UNIVERSIDADE DE SÃO PAULO

ESCOLA DE ENGENHARIA DE SÃO CARLOS

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

E DE COMPUTAÇÃO

Proposta de Sistema de Segurança Interativo Baseado em Conceitos de Internet of Things

Autor: Rodrigo Carpim de Almeida

Orientador: Prof. Dr. Evandro Luis Linhari Rodrigues

São Carlos

2016

Rodrigo Carpim de Almeida

Proposta de Sistema de Segurança Interativo Baseado em Conceitos de Internet of Things

Trabalho de Conclusão de Curso apresentado
à Escola de Engenharia de São Carlos, da
Universidade de São Paulo

Curso de Engenharia Elétrica

ORIENTADOR: Prof. Dr. Evandro Luis Linhari Rodrigues

São Carlos
2016

Almeida, Rodrigo Carpim
A447p Proposta de Sistema de Segurança Interativo Baseado em Conceitos de Internet of Things / Rodrigo Carpim Almeida; orientador Evandro Luis Linhari Rodrigues. São Carlos, 2016.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2016.

1. Segurança Residencial. 2. APP Inventor. 3. Arduino. 4. Internet of Things. 5. ESP8266. I. Título.

FOLHA DE APROVAÇÃO

Nome: Rodrigo Carpim de Almeida

Título: "Sistema de segurança interativo e a prova de quedas na rede elétrica"

Trabalho de Conclusão de Curso defendido e aprovado em 08 / 06 / 2016,

com NOTA 8,0 (oito, zero), **pela Comissão Julgadora:**

Prof. Associado Evandro Luís Linhari Rodrigues - (Orientador - SEL/EESC/USP)

Prof. Dr. Maximiliam Luppe - (SEL/EESC/USP)

Mestre André Luís Martins - (Doutorando - SEL/EESC/USP)

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Dr. José Carlos de Melo Vieira Júnior

Dedicatória

Dedico esse trabalho aos meus pais que sempre acreditaram no meu potencial e sempre me apoiaram em minhas decisões, mesmo, às vezes, não concordando com elas.

[Rodrigo Carpim de Almeida].

Agradecimentos

Agradeço ao meu orientador Prof. Dr. Evandro Luís Linhari Rodrigues por acreditar em minha ideia e me guiar durante o projeto.

Aos meus amigos de infância por nunca abandonar nossos laços.

Aos meus amigos de faculdade que me apoiaram e me ajudaram durante esses anos de graduação.

À república Paratudo que me permitiu vivenciar um outro lado da vida Universitária.

À Atlética CAASO por me dar a oportunidade de me desenvolver como pessoa me ensinando muitas coisas que não estão escritas em livros.

Ao meu amigo Igor Hueb.

À Marina dos Santos Crivillari por me apoiar e ser meu porto seguro durantes os momentos difíceis que passei. Esse projeto não seria possível sem você.

[Rodrigo Carpim de Almeida].

("When you want to succeed as bad as you wanna breath, then you will be successful")

[Eric Thomas]

("Pain is temporary. It may last for a minute, or an hour, or an day, or even a year. But eventually it will subside and something else will take it's place. If I quit however, it will last forever.")

[Eric Thomas]

("Life is a game of inches. The inches we need are everywhere around us. In every decision we make, every minute and every second.")

[Al Pacino]

("Nobody is gonna hit as hard as Life. It ain't about how hard you can hit it's about how hard you can get hit and keep moving forward")

[Rocky Balboa]

Resumo

No mundo em que vivemos acessibilidade e conectividade viraram requisitos mais do que essenciais no cenário tecnológico. Ter acesso a informação de modo fácil e conexão com diversas plataformas são pontos mandatórios em projeto de tecnologia que queira se sustentar no mercado. Nesse projeto é proposto a utilização de conceitos de *Home-Automation* e *Internet of Thing* que vão ao encontro com tendências citadas. O projeto consiste em um sistema de segurança residencial que permitirá: controle de entrada (com a utilização de cartão magnético), controle de periféricos (como sirenes, lâmpadas, etc), monitoramento da residência, alerta de invasão e interação desses atributos com SmartPhone. Para o controle de entrada foram utilizados componentes RFID e visor de LCD, controlados com Arduino UNO R3. Para realização do monitoramento foram utilizadas câmeras IP p2p (*peer-to-peer*) que possibilitam o controle de direção da mesma, filmagem e snapshots. Para o alarme de invasão foram utilizados sensores magnéticos acoplados a módulos Wi-fi ESP8266-01. Para o acionamento dos periféricos a placa Intel Galileo foi utilizada para o controle de relês. A alimentação dos periféricos (independente da rede elétrica) foi realizada através de um NoBreack e de uma bateria 12V. Para a utilização do projeto no SmartPhone foi desenvolvido um aplicativo através da plataforma APP Inventor do MIT. Para conexão do projeto o embarcado Intel Galileo gerou um WebService que possibilitou a conexão entre os módulos Wi-fi ESP8266, o SmartPhone e a câmera IP. O Arduino UNO, o visor de LCD e o componente RFID apresentaram funcionalidade compatível com a exposta pelos fabricantes. As câmeras IP apresentaram bom funcionamento, porém devido a sua configuração p2p sua acessibilidade é reduzida comparada aos modelos não p2p. Os sensores magnéticos não apresentaram nenhum tipo de problema, já os módulos ESP8266 mostraram alguns entraves, desde dificuldades físicas (como encaixe dos pinos e alimentação de 3,3V) até limitações de processamento e memória. O embarcado Intel Galileo teve funcionamento correto e não apresentou complicações após toda a implementação ser realizada. O aplicativo feito por meio do APP Inventor supriu as necessidades do projeto e a plataforma de desenvolvimento mostrou-se ser capaz de realizar outros tipos de aplicação, sendo uma ótima escolha para projetos de *Home-Automation*.

Palavras Chaves: Segurança residencial, *Internet of Things*, ESP8266, App Inventor, Arduino

Abstract

In our world, accessibility and connectivity have become the most essential requirements in the technological cenario. Having access to information and having an easy way to connect to multiple platforms are mandatory points in technology products that want to be sustainable in the market. In this project was proposed an use of concepts of Home-Automation and Internet of Thing following the cited tendencies. The project consists of a home security system that will: control of the access to the residence (with the use of magnetic card), control of gadgets (such as sirens, lights, etc.), monitoring the residence, intrusion alert and interaction of these attributes with SmartPhone. For the control of the access to the residence was used RFID components and LCD display controlled with Arduino UNO R3. To perform the monitoring was used IP cameras p2p (peer-to-peer) that enable the steering control of the camera, filming and snapshots. For the intrusion alarm, magnetic sensors were used coupled to ESP8266-01 Wi-fi modules. To active the gadgets the Intel Galileo was used to control relays. An NoBreack and a 12V battery were used as power sources of the gadgets. An app was made using the APP Iventor of the MIT to integrate all the parts of the Project on the SmartPhone. To make the connection the embarked Intel Galileo generated an WebService which enabled the connection between the Wi-fi ESP8266 modules, SmartPhone and IP camera. The Arduino UNO, the LCD display and the RFID componente have showed functionality compatible with exposed by the manufacturers. The IP cameras worked properly, but this cameras configurations are less accessible than the models not p2p. The magnetic sensors do not present any problem although the ESP8266 modules showed some obstacles, from physical obstacles (as plug pins and limited power source 3.3v) to processing power and memory limitations. Intel Galileo had correct operation and no complications since the implementation was carried out. The app made by the APP Inventor supplied the needs of the project and the development platform has proven to be able to perform other types of application, proving to be a great choice for Home-Automation projects.

Key words: home security, Internet of Things, ESP8266, App Inventor, Arduino.

Lista de Figuras

Lista de Tabelas

Siglas

ACPI	*Advanced Configuration and Power Interface*
App	Aplicativo
CI	Circuito Integrado
DHCP	*Dynamic Host Configuration Protocol* -
EEPROM	*Electrically-Erasable Programmable Read-Only Memory* -
GPIO	*General Purpose Input/Output*
HTML	*Hypertext Markup Language*
IDE	*Integrated Development Environment*
IoT	*Internet of Things*
ISA	*Industry Standard Architeture*
IP	*Internet Protocol*
I2C	*Inter-Integrated Circuit*
LCD	*Liquid Criytal Display*
LED	*Light Emitting Diode*
LUA	Tipo de codificação
MAC	*Media Access Control*
NFC	*Near Field Communication*
PCI	*Peripheral Component Interconnect*
PWM	*Pulse Width Modulation*
P2P	*Peer-to-Peer*
RAM	*Random Access Memory*
RF	Radio Frequência
RISC	*Reduced Instruction Set Computer* -
RFID	*Radio Frequency Identification*
RX	Recebimento

SD	*Secure Digital*
SOC	*System-on-a-Chip*
SPI	*Serial Peripheral Interface*
SRAM	*Static Random Access Memory*
UART	*Universal Asynchrounous Receiver/Transmiter -*
URL	*Uniform Resource Locator*
USART	*Universal Synchronous Asynchronous Receiver Transmitter -*
USB	*Universal Serial Bus*

Sumário

Capítulo 1

Introdução

Ao analisar a história da humanidade vemos que ela já passou por diversas eras de desenvolvimentos. Todas, sem dúvida, afetaram drasticamente o convívio entre as pessoas e mudaram totalmente os paradigmas da sociedade. Produção em larga escala com pouca mão de obra era impensável antes da revolução industrial. Comunicação quase instantânea com o outro lado do mundo era algo totalmente fora da realidade antes do *boom* da internet e a entrada da era digital.

Atualmente comenta-se muito sobre uma nova era de desenvolvimento, a pós-digital. Nessa nova era as pessoas estão tão acostumadas a lidar com novas tecnologias (plataformas *on-line*, integração entre aparelhos, comunicações a longa distância, interatividade, etc) que algo que não quebre paradigmas não terá nenhuma atenção das pessoas. Não é incomum ver vídeos de bebes e crianças pequenas tentando virar a página de uma revista como se estivessem mexendo em um *table* ou celular. As formas tradicionais de atuação já não têm mais o espaço de antigamente.

Um grande movimento que dá grande força a essa nova era é chamado de *Internet of Things (IoT)* [1]. A *Internet of Things* (internet das coisas em tradução livre) foca no conceito de integração entre os aparelhos eletrônicos ao nosso redor e o controle simplificado dos mesmos. Não é mais inconcebível um ar condicionado que ligue automaticamente toda vez que você chega em casa, ou então uma torradeira automatizada que prepare suas torradas com um toque no seu celular.

Este trabalho de conclusão de curso segue a linha do conceito *IoT* na área de segurança residencial. O projeto foca na utilização das novas tecnologias de comunicação local (utilizando o Wi-fi), acessibilidade e interatividade do sistema com SmartPhones, segurança do sistema e independência de outros sistemas não seguros como, por exemplo, a rede de ali-

mentação de energia de uma residência.

1.1 Motivação

Como citado anteriormente a era pós-digital põe em cheque os conceitos e padrões que existem hoje no mundo. Muitos serviços e atividades, anteriormente feitos por humanos, já vem sendo realizados por maquinas que agilizam e trazem mais confiança na questão qualitativa.

Os sistemas de segurança não escapam desse movimento. Hoje, a área de segurança residencial se concentra na prestação de serviço com utilização de muita mão de obra e pouco (ou nenhum) uso de tecnologia [2], porém com as novas inovações tecnológicas e com o grande aumento no uso de aplicativos e de Home-Automation é possível e extremamente viável o desenvolvimento de um sistema de monitoramento residencial de fácil utilização e integrado com o aparelho celular das pessoas.

1.2 Objetivos

Os objetivos desse projeto são:

1. Detecção de invasão residencial
2. Utilização de um aplicativo para *Android* que seja capaz de controlar periféricos
3. Ser possível de acompanhar o que acontece na residência, utilizando-se o Smartphone, de qualquer lugar onde se tenha acesso a internet.
4. Receber uma notificação no Smartphone quando algum dos sensores detectar abertura de alguma porta ou janela.
5. O sistema deve funcionar por um período mesmo que haja queda da alimentação da rede elétrica
6. Tornar o sistema eficiente nos pontos de vista energético e de transmissão de dados

1.3 Justificativas/Relevância

Este trabalho tem como função reforçar o movimento da *IoT* trazendo, com uma outra visão, um sistema de segurança residencial. O projeto tende a mostrar como atualmente é possível realizar inovações com componentes já disponíveis no mercado de forma simples e

eficiente.

1.4 Organização do Trabalho

Para facilitar, o projeto foi dividido em alguns blocos:

1. Acesso a residência.
2. Transmissão de Status dos Sensores.
3. Central de Operação.
4. Monitoramento.
5. Aplicativo.
6. Integração Completa.

Dentro dessa monografia primeiramente será apresentado um contexto teórico (**Embasamento Teórico**), seguido pelos aspectos técnicos de cada componente do projeto e pela construção de cada bloco (**Materiais e Métodos**) e finalizado com a comparação dos resultados e conclusão (**Resultados** e **Conclusão**).

Os capítulos **Materiais e Métodos** e **Resultados** seguem a mesma divisão de blocos. Além disso, todos os código utilizados no projeto se encontram nos **Apêndices**.

Capítulo 2

Embasamento Teórico

Nessa secção será apresentado um pouco do contexto teórico do projeto.

2.1 *Internet of Things*

O conceito de *IoT* surge, por meados de 2000, de uma fusão da comunicação utilizando RFID e a conexão de múltiplos módulos à internet através de um protocolo único e padronizado.

É interessante notar que por ser um conceito muito novo não há um definição fixa do que é *IoT*, mas independente da sua definição esse movimento trouxe uma visão completamente nova do futuro da eletrônica. Sua utilização já pode ser observada no cotidiano, camas inteligentes que regulam temperatura automaticamente, aquecedores ou ar condicionados que só ficam em funcionamento na presença de alguma pessoa são dois pequenos exemplo dentre muitos outros que mostram a tendência em se conectar aparelhos a embarcados e a internet.

Segundo o documento "*Towards a Definition of the Internet of Things (IoT)*" [1], publicado no site da IEEE, uma das possíveis descrições de *IoT* pode ser resumida pela seguinte frase: "*A network of items-each embedded with sensors-which are connected to the Internet*".

Essa descrição de embarcados conectados entre si e à internet se encaixa perfeitamente ao projeto. A utilização do módulo ESP8266 trás grande aliança com *IoT*, pela facilidade na utilização desse módulo para conectar outros componentes.

2.2 Sistemas Embarcados

Segundo o livro "*Embedded Systems Dictionary*" [5], um sistema embarcado (ou sistema embutido) é um sistema microprocessado no qual o computador é completamente encapsulado ou dedicado ao dispositivo ou sistema que ele controla.

Podemos colocar na lista de produtos que utilizam embarcados desde coisas complexas como um equipamento médico ou uma calculadora científica até coisas comuns ao cotidiano como um mp3 ou um rotiador. Praticamente tudo hoje em dia possui um pequeno computador embutido para a realização de tarefas pré determinadas, o que mostra a vasta gama de possíveis utilização de embarcados.

Assim como a outros desenvolvimentos tecnológicos a história dos sistemas embarcados tem início na área militar. O primeiro sistema embarcado de produção em massa foi o computador guia do míssil nuclear LGM-30 Míssil Minuteman, lançado em 1961, que constituiu o primeiro uso em grande volume de circuitos integrados. A utilização dessa tecnologia permitiu seu barateamento, reduzindo seu custo em mais de 300 vezes e garantindo o lugar dos sistemas embarcados no mundo.

2.3 *Home-Automation*

Segundo a Associação Espanhola de Domótica e Inmótica podemos definir *Home-Automation* como " a automatização e o controle aplicados à residência. Esta automatização e controle se realizam mediante o uso de equipamentos que dispõem de capacidade para se comunicar interativamente entre eles e com capacidade de seguir as instruções de um programa previamente estabelecido pelo usuário da residência e com possibilidades de alterações conforme seus interesses. Em consequência, a domótica permite maior qualidade de vida, reduz o trabalho doméstico, aumenta o bem-estar e a segurança, racionaliza o consumo de energia e, além disso, sua evolução permite oferecer continuamente novas aplicações."

Conceitualmente o termo domótica é utilizado como uma definição mais abrangente de *Home-Automation* por envolver por exemplo sistemas de sonorização. Nesse trabalho utilizaremos o termo *Home-Automation* para representar qualquer serviços proporcionados por sistemas tecnológicos integrados como o melhor meio de satisfazer as necessidades básicas de segurança, comunicação, gestão energética e conforto de uma habitação.

Ao unir *Home-Automation* com *IoT* e sistemas embarcados vemos que esses três conceitos

são complementares e facilmente presentes em projeto atuais. Apesar de ser algo novo (início por volta do ano 2000) se prevê que até o fim de 2020 70 % de todas as residências norte-americanas tenham alguma tipo de automação integrada.

2.4 Aplicativo

Para a construção do aplicativo e de suas funcionalidades, foi utilizada a plataforma de desenvolvimento *App inventor* criada no *MIT* [14].

A plataforma é bem simples de ser utilizada, sendo baseada em "programação por blocos"onde se adicionam blocos em dois ambientes diferentes (figura 2.1 e 2.2), um representando a interface que será apresentada no aplicativo e outra para a programação em si.

É interessante apontar que esse aplicativo só poderá ser utilizado em *Androids* e que o sistema de desenvolvimento do aplicativo permite teste em tempo real caso se tenha acesso a um SmartPhone, com sistema *Android*, conectado na mesma rede sem fio do computador onde se está desenvolvendo o aplicativo.

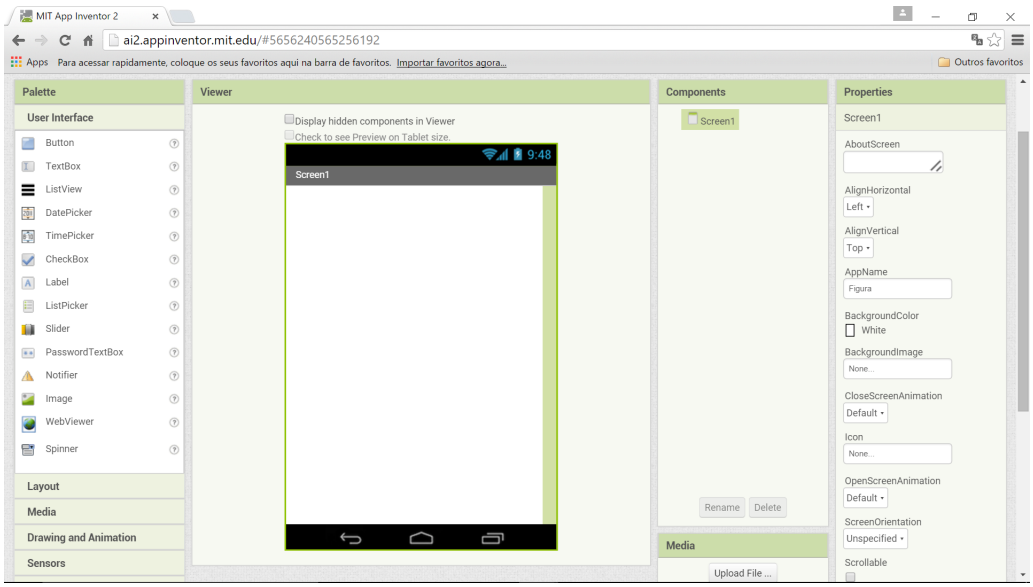


Figura 2.1: Página de Configuração da Interface do App

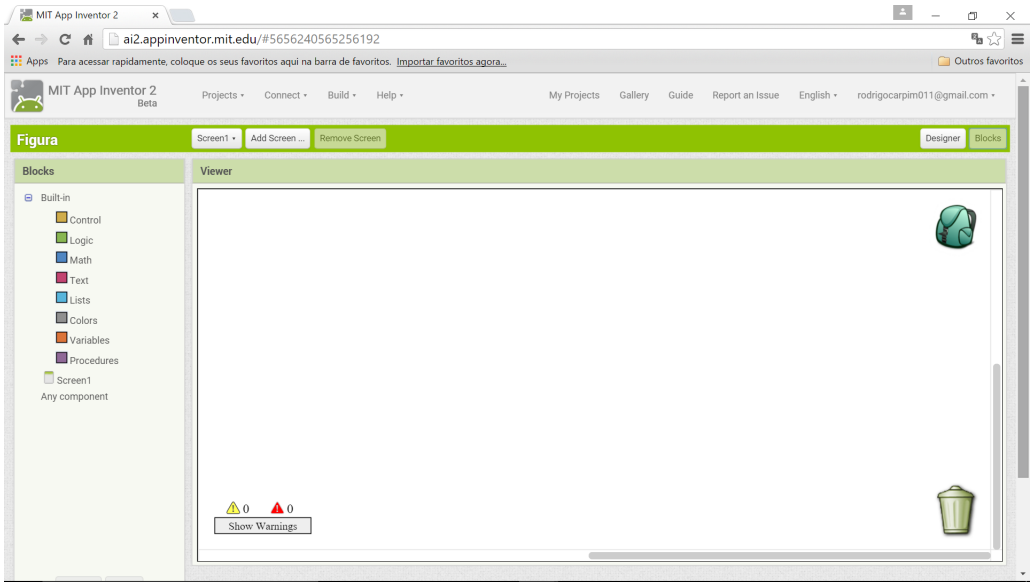


Figura 2.2: Página de Organização dos blocos para codificação do App

Pode-se acessar o site de desenvolvimento do aplicativo pelo endereço referenciado [14] na bibliografia.

Capítulo 3

Materiais e Métodos

Esse capítulo será divido em duas parte principais. Na primeira serão mostrados os componentes utilizados no projeto e suas especificações (**Materiais**) e na segunda será mostrado como cada um dos componentes foi interligado para se montar o sistema (**Métodos**).

3.1 Materiais

3.1.1 Módulo RFID e Display LCD

Para realizar o controle de acesso da residência foi utilizado um dispositivo RFID juntamente com um display LCD 16x2.

Como o próprio nome diz o sistema RFID (*Radio Frequency Identification*) se baseia na utilização de frequência de rádio para identificação das chamadas *tags*. A grande vantagem desse sistema está no fato das *tags* não precisarem ser componentes ativos, ou seja, apesar de não possuírem fonte de alimentação as *tags* ainda conseguem emitir um sinal resposta a partir do sinal do leitor de *tags*.

A figura abaixo mostra uma foto do dispositivo RFID utilizado no projeto:

Figura 3.1: Módulos RFID [8]

O componente leitor de *tags* possui 6 pinos:

Pino DAS - Comunicação

Pino SCK - Comunicação

Pino MOSI - Comunicação

Pino MISO - Comunicação

Pino NC - Não conectado (não é utilizado)

Pino GND - Terra

Pino RST - Reset

Pino 3,3 - Alimentação

O display LCD (*Liquid Criytal Display*) pode mostrar até duas linhas com 16 caracteres em cada uma de suas duas linhas (16x2). Os caracteres que podem aparecer na tela do display dependem dos *inputs* dados em seus pinos DB0-DB7. Por depender desses inputs o LCD não pode atuar sozinho e é dependente de outros componentes (EX.: um Arduino, uma RaspBarry, etc).

O display possui 16 pinos, como é mostrado na figura a seguir [20]:

VSS - Terra

VDD - Alimentação para controle lógico

V0 - Alimentação do display

RS - pino de *input* quando setado em nível alto; pino de instrução quando setado em nível baixo

RW - pino de leitura quando setado em nível alto; pino de escrita quando setado em nível baixo

E - *Enable*

DB0 - DB7 - Pinos de *input*, controle dos caracteres que serão escritos

A - Controle de Luminosidade (5v)

K - Controle de Luminosidade (0v)

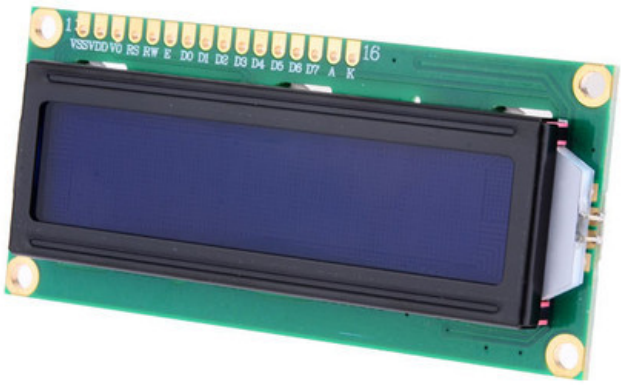


Figura 3.2: Display LCD [8]

3.1.2 Módulo Wi-fi ESP8266

O módulo Wi-fi ESP8266 é um componente que desde que entrou no mercado revolucionou a utilização de comunicação *wireless* para projetos de pequeno porte, barateando esse estilo de comunicação e incentivando fortemente a ideia da *IoT*. Esse componente possui mais de 12 modelos (figura 3.3) diferentes até o momento, nesse projeto o modelo utilizado é o ESP8266-01. Este modelo especificamente possui 8 pinos: Vcc, GND, RX, TX, *Chip*

Enable, *Reset*, GPIO-0 e GPIO-2. A figura 3.4 mostra como são distribuídos esses 8 pinos e a figura 3.5 mostra uma comparação do tamanho do componente com uma moeda.

Figura 3.3: Diferentes modelos da família ESP8266 disponíveis no mercado [7]

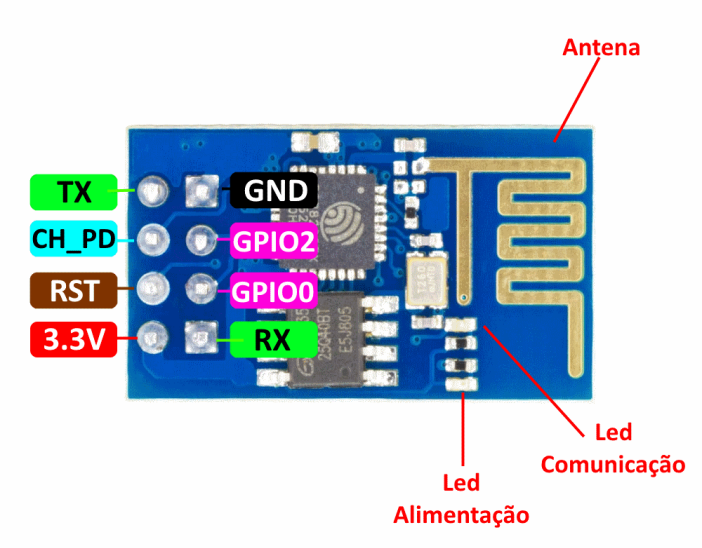


Figura 3.4: Modelo esquemático dos pinos do ESP8266-01 [8]

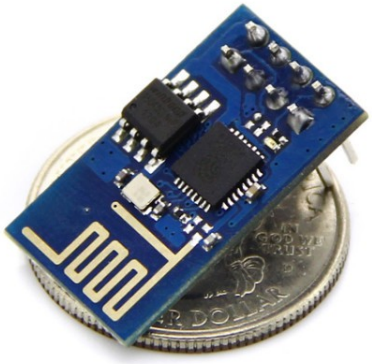


Figura 3.5: Comparação de tamanho do módulo ESP8266-01 com uma moeda [7]

Observando as figuras 3.4 e 3.5 fica claro que por esse módulo possuir tamanho reduzido pode ser utilizado em uma infinidade de projetos. Porém, também fica claro que a distribuição dos pinos do módulo não é nada amigável com o padrão de *protoboards*, por isso é necessária a utilização de adaptadores (como por exemplo *jumpers*) para a sua implementação o que pode causar pertubação na aquisição de sinal ou na transmissão de sinal serial pelo módulo.

Além disso, o módulo não opera com 5 Volts, mas sim com 3,3 Volts, o que o torna menos amigável a implementação. É importante colocar que apesar do módulo funcionar quando alimentado com 5 Volts sua temperatura dispara e muitas vezes o módulo acaba se auto-resetando como consequência.

Apesar de alguns pontos não favoráveis, o módulo possui uma capacidade incrível de processamento levando em consideração seu preço e tamanho. Algumas de suas características mais marcantes são [7]:

- Microcontrolador CPU que opera a 80MHz
- Arquitetura RISC de 32 bits
- 32Kbytes de memória RAM para instruções e 96Kbytes para dados.
- 64bytes de ROM para boot
- Memória Flash SPI Winbond W25Q40BVNIG

O *firmware* original do fabricante do módulo ESP8266-01 só funciona com protocolo simples, padrão AT [4], por comunicação serial, porém é possível trocar o *firmware*, o que

possibilita que o módulo funcione com outras linguagens.

Atualmente há duas mais utilizadas, uma baseada na codificação LUA [3] e uma na linguagem padrão de Arduino. Ambas conseguem operar bem no módulo e executar maiores funcionalidades, comparando-se com o *firmware* original. O *firmware* NodeMCU (baseado no LUA) apesar de apresentar boa quantidade de comando e de conseguir gravar uma rotina no próprio módulo, não possui um compilador padrão com interface simples e seus códigos tendem a ser relativamente maiores. Já o *firmware* IDE Arduino possui todas as vantagens do NodeMCU mais a facilidade de já utilizar o compilador padrão do Arduino o que acaba deixado-o mais popular e, com isso, novas versões de seu *firmware* são lançadas mais frequentemente.

Regulador

Apesar de operar com menos de 5 Volts o ESP8266-01 pode chegar a consumir muita corrente durante sua operação. Com isso, o ideal para o módulo é que ele seja alimentado por uma fonte independente que consiga fornecer no mínimo 300mA. A tabela 3.1 mostra os valores de corrente consumido com cada tipo de operação:

Para prevenir o projeto de problemas relacionados a corrente utilizou-se uma fonte ligada ao regulador de tensão Ywrobot (figura 3.6) para poder entregar até 700mA.

Figura 3.6: Regulador Ywrobot de 5 ou 3,3 Volts

Modo	Corrente Típica
Transmit 802.11b, CCK 1Mbps, POUT=+19.5dBm	215mA
Transmit 802.11b, CCK 11Mbps, POUT=+18.5dBm	197mA
Transmit 802.11g, OFDM 54Mbps, POUT =+16dBm	145mA
Transmit 802.11g, OFDM 54Mbps, POUT =+16dBm	145 mA
Transmit 802.11n, MCS7, POUT=+14dBm	135 mA
Receive 802.11b, packet length=1024 byte, -80dBm	60mA
Receive 802.11g, packet length=1024 byte, -70dBm	60mA
Receive 802.11n, packet length=1024 byte, -65dBm	62mA
Standby	0,9mA
Deep sleep	10uA
tower save mode DTIM 1	1,2mA
Power save mode DTIM 3	0,86mA
Power save mode DTIM 3 0.86 mA Total shutdown	0,5uA

Tabela 3.1: Tabela Consumo de Corrente ESP8266-01 [7]

Conversor USB-Serial

O único modo de se comunicar com o ESP8266-01, que veio direto da fábrica, é por meio de comunicação serial, inclusive para se fazer a troca do seu *firmware*. Pode-se realizar essa comunicação de várias maneiras diferentes, utilizando-se Arduinos de diferentes modelos, outros embarcados com RaspBerry ou, então, utilizando-se um adaptador USB-Serial.

No caso desse projeto, utilizou-se adaptador genérico que faz uso de um CI PL2303HX e que tem saída padrão de 5 Volts. É interessante notar que a maioria desses adaptadores USB-Serial possui *chips* que não trabalham com sistemas operacional Windows Vista, 8, 8.1, 10 ou Mac OS X. Para que esses conversores possam operar nesses sistemas, é necessário que sejam instalados *drivers* para reconhecimento desse adaptador.

Figura 3.7: Conversor USB-Serial

CI CD4050Be

Como visto anteriormente o módulo Wi-fi só pode receber 3,3 Volts em qualquer entrada. Como a maioria das saídas de componentes no geral operam com 5 Volts é preciso abaixar essa tensão para 3,3 Volts. O jeito mais simples de se realizar isso é utilizar um divisor de tensão com valores de resistências pré-determinados. Porém pode-se optar por conversores DC-DC que apresentam maiores praticidades, como tamanho reduzido e maior quantidade de I/O.

Como nesse projeto já é utilizada uma fonte de 3,3 Volts por conta da necessidade de corrente, a única precaução que precisou ser tomada foi diminuir a tensão de saída do pino TX do conversor USB-Serial de 5 Volts para 3,3 Volts para ser compatível com o máximo de

tensão de entrada do pino RX do módulo. A situação representou um ponto muito específico no projeto. Optou-se por utilizar o CI CD4050BE, um *buffer* não inversor, ligando o sinal de 5 Volts a sua entrada e limitando sua saída para 3,3 Volts. Apesar de não ser tão fiel em termos de qualidade o CI não apresentou nenhum problema.

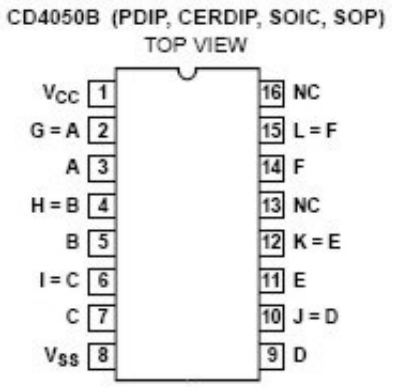


Figura 3.8: Pinagem do CD4050 [9]

3.1.3 Arduino Uno R3

Utilizou-se também o módulo Arduino Uno R3 e seu *Ethernet Shield* (figuras 3.9 e 3.10 respectivamente). O Arduino é um sistema *open source*, ou seja, toda sua estrutura (tanto de hardware quanto de software) é aberta para download, o que permite que terceiros façam bibliotecas ou lancem compiladores diferentes dos que a própria empresa desenvolve. Isso tem dois lados que são importantes pontuar. Essa ação de abertura aumenta o desenvolvimento dessa tecnologia e de sua popularidade em grande escala, o que traz enormes benefícios para a sociedade, porém por esse sistema ser completamente conhecido a sua falsificação é extremamente facilitada [10].

Figura 3.9: Arduino UNO R3

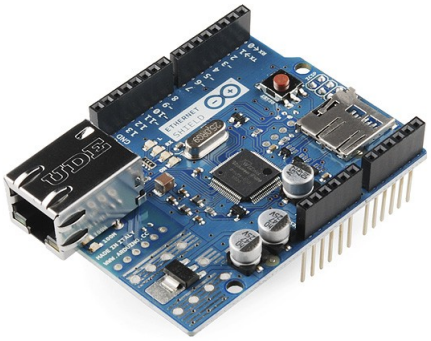


Figura 3.10: *Shield Ethernet* para Arduino

O Arduino Uno R3 é um dos mais simples módulos da sua família mas ainda assim possui características que permitem que o mesmo tenha grande utilidade em inúmeros projetos. Podemos citar como mais marcantes as seguintes características [11]:

- 14 saídas/entradas digitais

- 6 saídas/entradas analógicas

- Pinos com funcionalidades especiais como: PWM, Comunicação Serial e Interrupção interna

- Pinos de tensão contínua tanto de 5 Volts quanto de 3,3 Volts (apesar de poderem ofere-

cer no máximo 40mA)

- Microprocessador ATMEL ATMEGA328

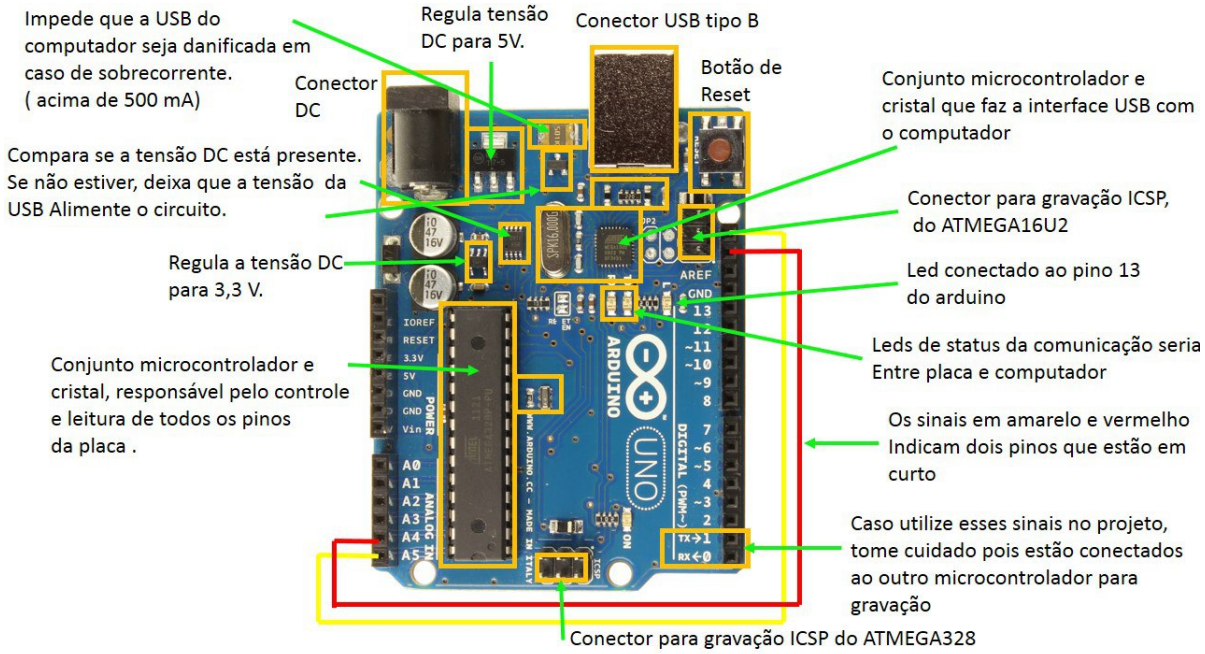


Figura 3.11: Resumo da Placa Arduino Uno R3 [11]

O componente principal da placa Arduino UNO é o microcontrolador ATMEL ATMEGA 328, um dispositivo de 8 bits da família AVR com arquitetura RISC avançada e com encapsulamento DIP28. Ele conta com 32 KB de Flash (mas 512 Bytes são utilizados para o /it bootloader), 2 KB de RAM e 1 KB de EEPROM. Pode operar a até 20 MHz, porém na placa Arduino UNO opera em 16 MHz, valor do cristal externo que está conectado aos pinos 9 e 10 do microcontrolador.

Esse microcontrolador pode operar com tensões bem baixas, de até 1,8 V, mas nessa tensão apenas opera até 4MHz. Possui dois modos de consumo super baixos, o *Power-down Mode* e o *Power-save Mode*, para que o sistema possa poupar energia em situações de espera. Possui, como periféricos, uma USART que funciona a até 250kbps, uma SPI, que vai a até 5MHz, e uma I2C que pode operar até 400kHz. Conta com um comparador analógico interno ao CI e diversos timers, além de 6 PWMs. A corrente máxima por pino é de 40mA, mas a soma da corrente de todo o CI não pode ultrapassar 200mA. Ele possui um oscilador interno de 32kHz que pode ser utilizado, por exemplo, em situações de baixo consumo [11].

Essas características somadas a fácil comunicação computador-placa (por meio de cabo USB-Serial), a sua independência de conexão com computador após ser programada e a fácil alimentação (funciona normalmente com qualquer alimentação contínua entre 7-12 Volts), mostram o quão útil pode ser essa placa.

Arduino Ethernet Shield Wiznet 5100

A comunicação da placa Arduino com a Internet é feita com o *Ethernet Shield* desenvolvido para trabalhar como uma extensão do Arduino Uno R3.

O *Shield* possui um microprocessador W5100 [12] que é o que permite fazer o acesso à internet. Como visto na figura 3.10 essa extensão deve se conectar ao roteador do local utilizando-se o cabo padrão RJ45. Além disso, é possível colocar um micro SD nesse *Shield* para se armazenar dados ou programas. É importante apontar que em alguns casos alguns pinos específicos não podem ser utilizados, pois estão reservados para fazer a comunicação entre a placa do Arduino e o *Ethernet Shield*.

3.1.4 Câmera IP

Nesse projeto foram utilizadas duas câmeras IP ,*wireless*, fabricadas na China, uma do modelo "Pan Tilt"(figura 3.12) P2P Network Camera e outra do modelo "TOP CAM". Ambas as câmeras já possuem aplicativo próprio para o celular que permite a visualização de suas imagens, controle de posição, controle de foco, fazer filmagens e tirar fotografias. Porém, somente a Camera "TOP CAM"possui acessibilidade a todas as configurações através do seu endereço de IP, dispensando o uso do software próprio da câmera, o que será muito útil durante a construção do aplicativo do projeto.

Figura 3.12: Câmera IP Pam Tilt

Tabela de Descarga a 25º - Corrente e Potência Constantes							
Tempo	A/W	5 min	15 min	30 min	1h	5h	20h
9,60v	A	21,6	10,5	6,9	3,6	1,05	0,31
9,60v	W	224	113	74,2	38,9	12	3,6
11,10v	A	16,0	7,80	6,00	3,05	0,93	0,29
11,10v	W	182	91	70,2	35,9	11,1	3,45

Tabela 3.2: Tabela de Descarga da Bateria 12V

Figura 3.13: Câmera IP Top Cam

3.1.5 Bateria e Carregador 12V

Para realizar a alimentação dos periféricos foi utilizado um carregador (figura 3.15) de 12V de saída (podemos considerar como uma fonte 12V) e uma bateria 12V (figura 3.14) para ser utilizada quando não houvesse alimentação da rede elétrica e, portanto, o carregador parasse de funcionar.

A tabela 3.14 mostra as especificações de carga da bateria:

Figura 3.14: Bateria 12V

Figura 3.15: Carregador 12V

3.1.6 NoBreak

Para garantir que o projeto funcionasse mesmo que não houvesse alimentação da rede elétrica foi utilizado um NoBreak 600VA save HOME, modelo "T1 - trivolte".

Podemos citar como principais características:

- Tecnologia Senoidal por aproximação [Nobreak interativo - NBR 15014];
- Microprocessador FLASH;
- Ampla faixa de tensão de entrada [80V - 142V] / [175V - 284V]
- 6 [seis] tomadas de saída;
- Battery save;
- DC-start: partida mesmo sem rede elétrica;
- Filtro de linha contra distúrbios na rede elétrica;
- Estabilizador com 4 [quatro] estágios de regulação;
- Proteção contra subtensão e sobretensão com retorno automático;
- Proteção contra sobrecarga e curto-circuito;

É importante notar que, apesar de ser uma fonte temporária de energia para o projeto, o NoBreak garante o funcionamento do sistema mesmo se a rede elétrica da residência for cortada.

3.1.7 Módulo Relé 5V

O módulo relé 5V foi utilizado para realizar o acionamento dos periféricos. O controle da Central de Operação dirá quando o relé deve abrir ou fechar a conexão, acionando assim os periféricos.

O módulo relé nada mais é do que dois relés fixados a uma placa, juntamente com diodos e resistores (para proteção). Esse componente está preparado para receber até 30 Volts DC ou 250 Volts AC e 10 A de corrente [22].

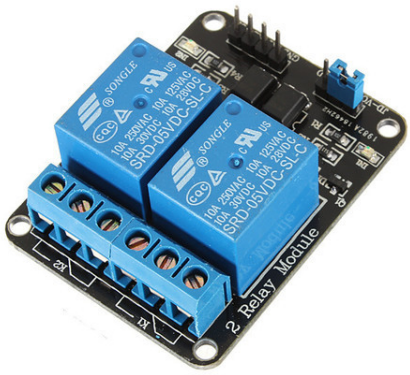


Figura 3.16: Módulo Rele 5 V [8]

3.2 Métodos

A figura 3.17 mostra a montagem final do sistema. Nas próximas secções serão apresentados os métodos utilizados em cada uma das parte do projeto.

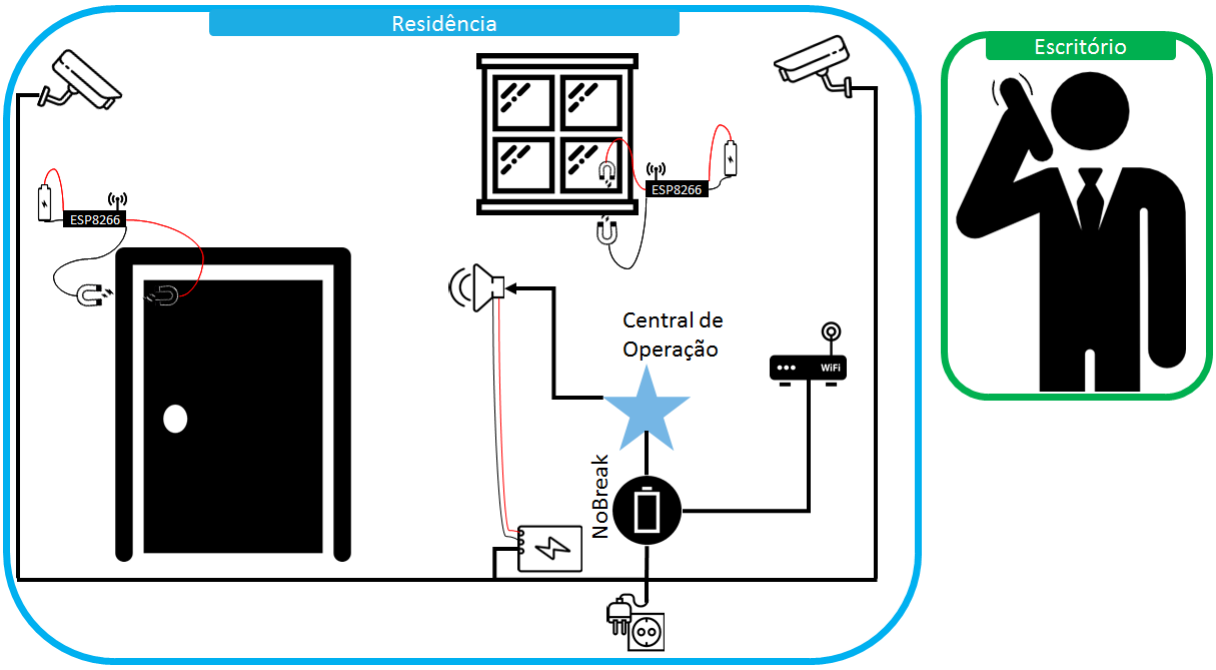


Figura 3.17: Esquemático do Projeto

3.2.1 Bloco 1 - Acesso

Para realizar o controle de acesso foram utilizados o módulo RFID, o display LCD e um Arduino UNO R3.

Antes de começar a criar o código na IDE é preciso baixar a biblioteca MFRC522, o que pode ser feito no link [23].

As imagens abaixo mostram a montagem do sistema e o código implementado no Arduino.

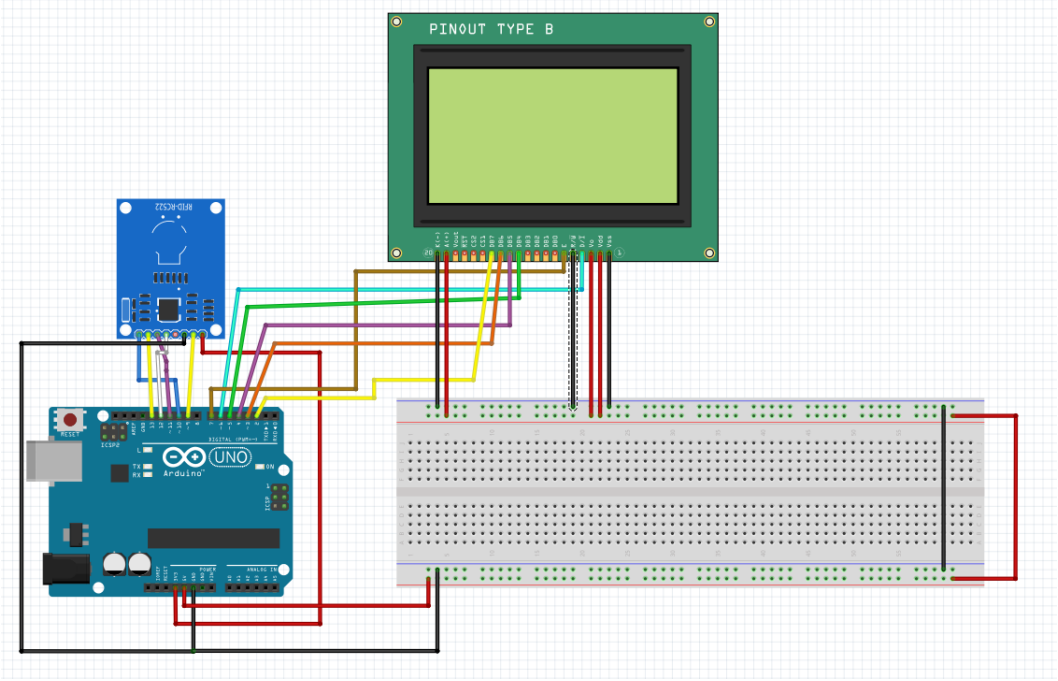


Figura 3.18: Montagem do sistema de acesso

Arquivo Editar Sketch Ferramentas Ajuda

Cart_o_RFID_V_2 §

```
//Programa : RFID - Controle de Acesso leitor RFID#include <SPI.h>
#include <MFRC522.h>
#include <LiquidCrystal.h>
#define SS_PIN 10
#define RST_PIN 9
MFRC522 mfrc522(SS_PIN, RST_PIN);
LiquidCrystal lcd(6, 7, 5, 4, 3, 2);
int acesso = 8;
int SensorMag = 1;
char st[20];

void setup()
{
  Serial.begin(9600);   // Inicia a serial
  SPI.begin();      // Inicia  SPI bus
  mfrc522.PCD_Init();   // Inicia MFRC522
  pinMode(acesso, OUTPUT);
  pinMode(acesso, INPUT);
  digitalWrite(acesso, HIGH);
  Serial.println("Aproxime o seu cartao do leitor...");
  Serial.println();
  //Define o número de colunas e linhas do LCD:
  lcd.begin(16, 2);
  mensageminicial();
}
```

```
void loop()
{
  // Look for new cards
  if ( ! mfrc522.PICC_IsNewCardPresent())
  {
    return;
  }
  // Select one of the cards
  if ( ! mfrc522.PICC_ReadCardSerial())
  {
    return;
  }
  //Mostra UID na serial
  Serial.print("UID da tag :");
  String conteudo= "";
  byte letra;
  for (byte i = 0; i < mfrc522.uid.size; i++)
  {
    Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
    Serial.print(mfrc522.uid.uidByte[i], HEX);
    conteudo.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
    conteudo.concat(String(mfrc522.uid.uidByte[i], HEX));
 }
 Serial.println();
 Serial.print("Mensagem : ");
 conteudo.toUpperCase();
```

```
if (conteudo.substring(1) == "A4 DB 6E B8") //UID 1 - Chaveiro
{
  Serial.println("Ola TCC !");
  Serial.println();
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Ola TCC !");
  lcd.setCursor(0,1);
  lcd.print("Acesso liberado!");
  delay(3000);
  digitalWrite(acesso, LOW);
  while {SensorMag = 1;}
  digitalWrite(acesso, HIGH);
  mensageminicial();
}

if (conteudo.substring(1) == "01 48 FB E5") //UID 2 - Cartao
  Serial.println("Desconhecido !!!");
  Serial.println();
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Desconhecido!");
  lcd.setCursor(0,1);
  lcd.print("Acesso Negado !");
  delay(3000);
  digitalWrite(acesso, HIGH);
  mensageminicial();
}
}

void mensageminicial()
{
  lcd.clear();
  lcd.print(" Aproxime o seu");
  lcd.setCursor(0,1);
  lcd.print("cartao do leitor");
}
```

Figura 3.19: Código Implementado para controle de accesso

Ressaltas-se que os números de identificação de cada *tag* (setas azuis) não são padrões e é preciso descobri-los. O retângulo azul garante que o número de identificação de todo cartão que não está cadastrado no código apareça no display LCD. Portanto, após realizar o *upload*do código no Arduino foi necessário ajusta-lo com o número dos *tag*.

A lógica de acesso se concentra nas setas amarelas e verdes e será discutida no **Bloco 7 - Integração Completa** .

3.2.2 Bloco 2 - Transmissão do Status dos Sensores

Nesse projeto foram testados dois métodos diferente para se transmitir o status da porta e janela. No primeiro método foi utilizado o modulo RF433MHz e no segundo foi utilizado o módulo Wi-fi ESP8266-01.

Ambos os métodos tiveram como input a leitura do sensor magnético e se comunicavam com a Central de Operação.

ESP8266-01

Nesse bloco é descrito todo o processo no qual foi utilizado o módulo Wi-fi ESP8266-01 antes da integração.

A primeira coisa em que se deve atentar nesse estágio do projeto é se o sistema operacional reconhece o adaptador USB-Serial. Caso o sistema operacional não reconheça é necessário fazer a instalação de extensões para que seja possível prosseguir. No seguinte endereço é possível encontrar diferentes extensões [15].

Instalação do *Firmware*

Feito isso, é possível passar para a análise do *firmware* do módulo. Como visto anteriormente é possível utilizar o módulo com 3 *firmware* diferentes e, por facilidade, foi escolhido utilizar o IDE Arduino. Para isso o *firmware* do componente não tem que ser atualizado ou mudado necessariamente, porém como não há somente um único distribuidor desse módulo, e para evitar futuros problemas, foi realizada a inserção do *firmware* que responde aos comandos padrões AT.

Foi utilizado o programa *esp8266-flasher* juntamente com o arquivo "ESP-8266-BIN0.92"para realizar a atualização de *firmware*. Tanto o programa de gravação quanto o arquivo .bin podem ser obtidos no endereço [16]. Para completar o uso desse programa só é necessário selecionar o arquivo bin descrito anteriormente, quando se clica no botão "Bin".

É importante apontar que para que o módulo entenda que seu *firmware* será modificado é necessário deixar o pino GPIO-0 aterrado. As figuras 3.20 e 3.21 mostram respectivamente o software "esp8266-flasher"e o esquema de ligação necessário para realizar a troca do *firmware*.

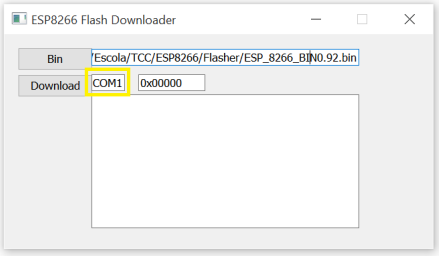


Figura 3.20: Programa esp8266 flasher

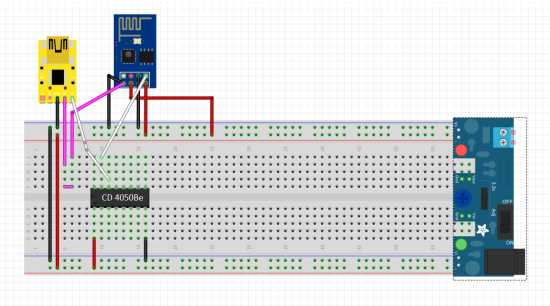


Figura 3.21: Esquema de ligação para gravação no ESP8266-01

A parte destacada na figura 3.20 evidencia que pode ser necessário trocar o inscrito "COM 1"para o corresponde a porta onde está inserido o adaptador USB-Serial. Para descobrir qual porta é, basta clicar no Botão Windows+X selecionar "Gerenciador de Dispositivos"e abrir a aba "Portas".

Instalação do IDE Arduino

Após certificar-se de que o módulo está com um *firmware* adequado deve-se instalar a versão 1.6.5 do compilador padrão do arduino, o que pode ser feito acessando o seguinte endereço [17]. Já há versões mais atualizadas do compilador do arduino, porém observou-se erro ao utilizar outra versão (mesmo que mais atualizada) sem ser a 1.6.5, uma vez que o compilador não conseguiu localizar a pasta de biblioteca.

Seguindo, foi feito o download da extensão IDE [18] . Depois seguiu-se os passos: Foi aberto o arquivo executável do compilador Arduino, pressionou-se então "control+,"e trocou-se o endereço destacado na figura 3.22 pelo da figura 3.23.

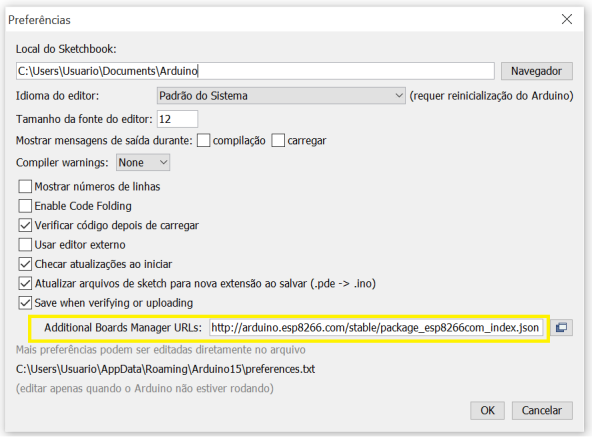


Figura 3.22: Janela "Preferências"do compilador arduino

- Enter `http://arduino.esp8266.com/stable/package_esp8266com_index.json` into *Additional Board*

Figura 3.23: Endereço para download dos arquivos IDE Arduino

O passo tomado consecutivamente foi entrar em "Ferremanetas -> Placas -> Gerenciador de Placas"achar a extensão esp8266 e instala-la. Após realizar esses passos será possível gravar código no módulo Wi-fi utilizando-se o compilador do Arduino.

Codificação do programa

Nessa secção será apresentado o código utilizado com o módulo ESP8266-01.

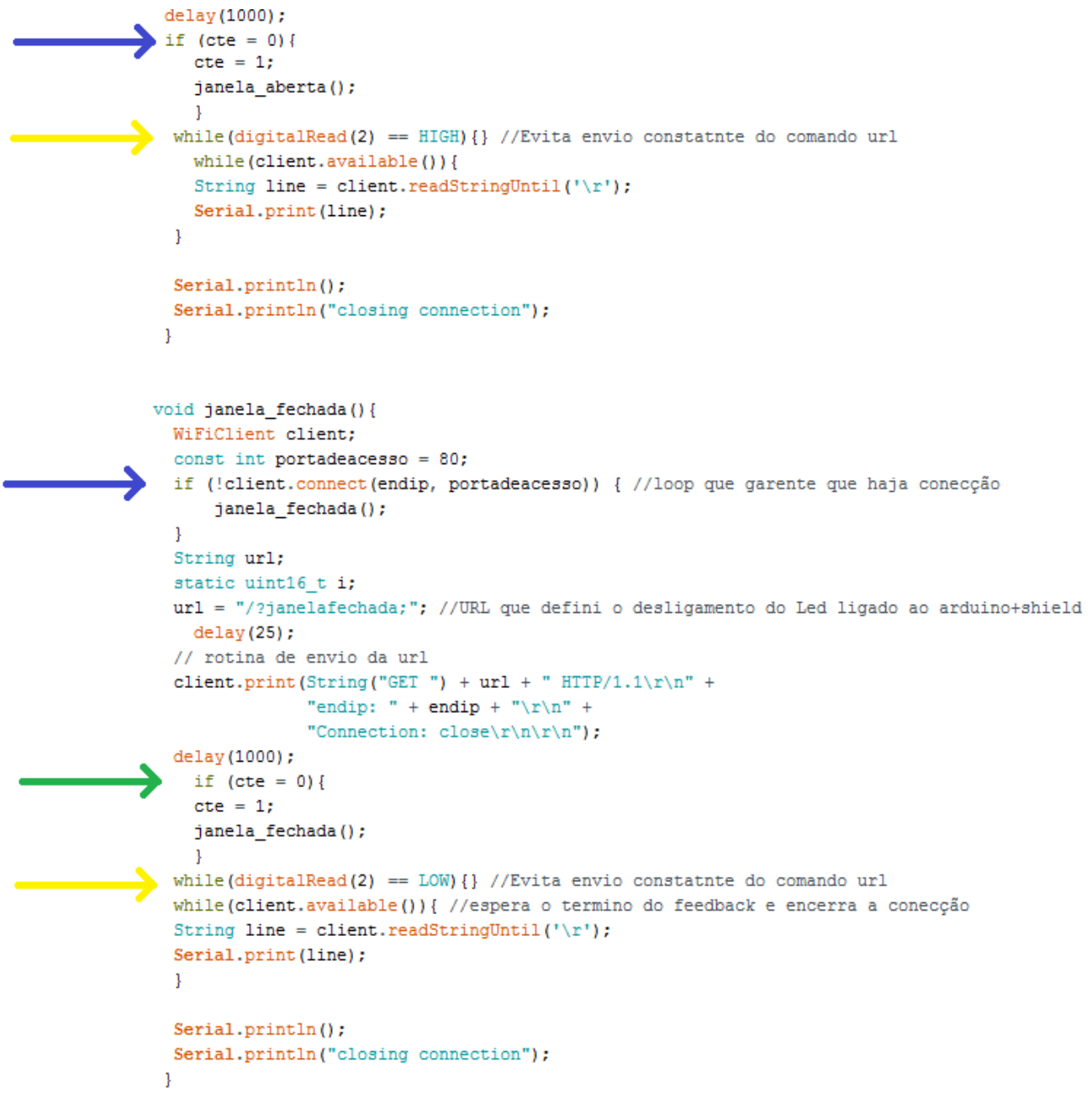
Foi implementado no módulo ESP8266 um código que interage com as instruções HTML processadas pela Central de Operação. A tarefa do módulo é verificar o valor do sensor magnético de porta (que está conectado ao GPIO-2 do módulo) e enviar um comando para o IP da Central de Operação para que a mesma realize o controle.

O código a seguir mostra como foi feita a conexão do módulo com a rede Wi-fi, assim como a verificação do estado do sensor magnético.

Arquivo Editar Sketch Ferramentas Ajuda

EnvioURL_V08_Janela

```
#include <ESP8266WiFi.h>
const char* nome     = "    "; //nome da rede wifi que se dejesa conectar
const char* senha = "          "; //senha dessa rede
const char* endip = "192.168.0.17"; //endereço de ip do arduino+shield
int cte = 0;


void setup() {
  pinMode(2, INPUT);
  Serial.begin(115200);
  delay(100);
  Serial.println();
  Serial.println();
  WiFi.begin(nome, senha); //conceta-se há rede wifi definida

  while (WiFi.status() != WL_CONNECTED) {//loop que garante que o módulo será conectado
    delay(500);
  }
}

void loop() {
  cte = 0;
  if(digitalRead(2) == LOW){janela_fechada();} //verifica se o sensor está aberto
  if(digitalRead(2) == HIGH){janela_aberta();} //verifica se o sensor está fechado

}


void janela_aberta(){
  WiFiClient client;
  const int portadeacesso = 80;
  if (!client.connect(endip, portadeacesso)) { //loop que garente que haja conecção
      janela_aberta();
  }
  String url;
  static uint16_t i;
  url = "/?janelaaberta;"; //URL que defini o ligamento do Led ligado ao arduino+shield
    delay(25);
  // rotina de envio da url
  client.print(String("GET ") + url + " HTTP/1.1\r\n" +
               "endip: " + endip + "\r\n" +
               "Connection: close\r\n\r\n");
```



```
delay(1000);
if (cte = 0){
   cte = 1;
   janela_aberta();
   }
 while(digitalRead(2) == HIGH){} //Evita envio constatnte do comando url
   while(client.available()){
   String line = client.readStringUntil('\r');
   Serial.print(line);
 }

 Serial.println();
 Serial.println("closing connection");
}


void janela_fechada(){
 WiFiClient client;
 const int portadeacesso = 80;
 if (!client.connect(endip, portadeacesso)) { //loop que garente que haja conecção
     janela_fechada();
 }
 String url;
 static uint16_t i;
 url = "/?janelafechada;"; //URL que defini o desligamento do Led ligado ao arduino+shield
   delay(25);
 // rotina de envio da url
 client.print(String("GET ") + url + " HTTP/1.1\r\n" +
              "endip: " + endip + "\r\n" +
              "Connection: close\r\n\r\n");
 delay(1000);
   if (cte = 0){
   cte = 1;
   janela_fechada();
   }
 while(digitalRead(2) == LOW){} //Evita envio constatnte do comando url
 while(client.available()){ //espera o termino do feedback e encerra a conecção
 String line = client.readStringUntil('\r');
 Serial.print(line);
 }

 Serial.println();
 Serial.println("closing connection");
}
```

Figura 3.24: Código Utilizado no módulo ESP que verifica o status da Janela

É interessante colocar que as linhas apontadas pelas setas verdes mostram "mini-loops" que garantem a conexão do módulo tanto no Wi-fi quanto no endereço de IP do Arduino. Essas precauções foram importantes no código pela dificuldade que o módulo tem em conectar

quando há algum tipo de interferência ou simplesmente por lentidão no sistema Wi-fi.

Já as setas amarelas mostram outro tipo de "mini-loop", o qual evita que o ESP8266-01 envie repetitivamente o comando URL correspondente à leitura do pino GPIO-2. Caso isso ocorresse o Arduino seria forçado a realizar rapidamente e repetitivamente um comando de controle específico o que se mostrou prejudicial a qualidade do comando, distorcendo a saída desejada e por muitas vezes não realizando o controle desejado.

Por sua vez as setas azuis mostram a lógica utilizada para que seja enviado duas vezes o comando URL cada vez que se detecta alteração no pino GPIO-0. Esse artifício foi utilizado para garantir que a Central de Operação recebesse a URL.

Vale ressaltar que a utilização dos "mini-loop"foi uma solução encontrada para a implementação, mas que apresentam um risco para o fluxo do programa, uma vez que o sistema não consegue sair dessa parte do código enquanto a ação não for realizada. Outras soluções, como por exemplo, a utilização de "watch dogs", flag ou uma determinada quantidade máxima de vezes que cada ação pudesse ser executada, cobriria esse problema.

3.2.3 Bloco 3 - Central de Operação

O bloco Central de Operação do projeto é responsável por criar o WebService que será utilizado na transmissão de informação entre os diversos componentes do projeto e também é responsável pelo acionamento dos periféricos.

Para o WebService foi utilizada codificação HTML padrão e comandos através de URLs. A lógica do sistema se baseia na leitura da URL enviada ao endereço de IP da Central, na decisão de *setagem* dos pinos em alto ou baixo e na decisão do que será escrito no WebService. Com essa lógica já é possível realizar todo o controle de acionamento dos periféricos.

Foram implementados dois modelos de Central de Operação diferentes, um utilizando o embarcado Intel Galileo e outro utilizando a placa Arduino UNO R3 juntamente com seu *Ethernet Shield*.

Arduino UNO + Shield Ethernet

Nesse segundo bloco será descrito todo o processo para o qual foi utilizado o Arduino e seu *Ethernet Shield*.

Codificação do programa

Primeiramente foi-se conectado o *Ethernet Shield* ao Arduino Uno R3 e ao roteador pelo cabo padrão RJ45. Conectando o Arduino ao computador pelo cabo Serial-USB o sistema já está pronto para receber o código.

Abaixo é mostrado o código implementado no Arduino:

Arquivo Editar Sketch Ferramentas Ajuda

Site_Luz_V05 §

```
//Programa do site que acende ou apaga a luz dependendo do sinal do ESP8266-01
#include <SPI.h>
#include <Ethernet.h>

byte mac[] = {0xde, 0xab, 0xbd, 0xaf, 0xfe, 0xed}; //physical mac address
byte ip[] = { 192,168,0,170 }; // IP fixo
EthernetServer server(80);
int panico = 8;

String readString;
void setup(){
pinMode(6, OUTPUT); //pino de indicaçãp da porta
pinMode(7, OUTPUT); //pino de indicação da janela
pinMode(8, OUTPUT); //pino de indicação do botao de panico
Ethernet.begin(mac, ip);
server.begin();
}

void loop(){
EthernetClient client = server.available();
if (client) {
while (client.connected()) {
if (client.available()) {
char c = client.read();
if (readString.length() < 100) {
readString += c;
}
 if(readString.indexOf("?portaaberta") >0)//Checa se o houve o comando URL de porta aberta
 {
 digitalWrite(6, HIGH); //Seta o pino 6 em alto
 }
 if(readString.indexOf("?portafechada") >0)//Checa se o houve o comando URL de porta fechada
 {
 digitalWrite(6, LOW); // Seta o pino 6 em baixo
 }
if(readString.indexOf("?janelaaberta") >0)//Checa se o houve o comando URL de janela aberta
{
  digitalWrite(7, HIGH); // Seta o pino 7 em alto
}
if(readString.indexOf("?janelafechada") >0)//Checa se o houve o comando URL de janela fechada
{
  digitalWrite(7, LOW); // Seta o pino 7 em baixo
}
if(readString.indexOf("?ledon") >0)//Checa se o houve o comando URL de led de teste
{
  digitalWrite(5, HIGH); // Seta o pino 5 em alto
}
if(readString.indexOf("?ledoff") >0)//Checa se o houve o comando URL de led de teste
{
  digitalWrite(5, LOW); // Seta o pino 5 em baixo
}
if(readString.indexOf("?panicoon") >0)//Checa se o houve o comando URL de led de teste
{
  digitalWrite(8, HIGH); // Seta o pino 8 em baixo
}
if(readString.indexOf("?panicooff") >0)//Checa se o houve o comando URL de led de teste
{
  digitalWrite(8, LOW); // Seta o pino 8 em baixo
}
```

```
if (c == '\n') {
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
client.println();
client.println("<HTML>");
client.println("<HEAD>");
client.println("<TITLE>Verificacao de portas aberta</TITLE>");
client.println("</HEAD>");
client.println("<BODY");
if(digitalRead(6) == LOW){ client.println("<H1>PortaOFF</H1>");}
if(digitalRead(6) == HIGH){  client.println("<H1>PortaON</H1>");}
if(digitalRead(7) == LOW){ client.println("<H1>JanelaOFF</H1>");}
if(digitalRead(7) == HIGH){  client.println("<H1>JanelaON</H1>");}
if(digitalRead(8) == HIGH){client.println("<H1>PANICO!!!!!</H1>");}
if(digitalRead(8) == LOW){client.println("<H1>NORMAL</H1>");}


client.println("<hr />");
client.println("<br />");

client.println("<a href=\"/?ledon\"\">LED ON</a>");
client.println("<a href=\"/?ledoff\"\">LED OFF</a><br />");

client.println("</BODY>");
client.println("</HTML>");

delay(10);
client.stop();

readString=""; //Limpa o vetor para o próximo comando

}
}
}
}
```

Figura 3.25: Código Utilizado no Arduino da Central de Controle

Os números de MAC e de IP (apontados respectivamente pela seta verde e azul) não são padrões e portanto devem ser obtidos.

O número de IP foi escolhido por decisão arbitrária para que houvesse um número fixo e o sistema DHCP não mudasse o endereço de IP.

Já o número MAC muitas vezes está escrito em uma etiqueta colada no próprio *Ethernet Shield*. No caso do projeto não havia etiqueta no *Ethernet Shield*, então para descobrir qual o número MAC foi preciso acessar o endereço de IP do roteador (um padrão muito comum é o 192.168.0.1) e verificar quais números de MAC estavam conectados à rede. O login e senha

do roteador estavam presentes em uma etiqueta colada na parte inferior do roteador.

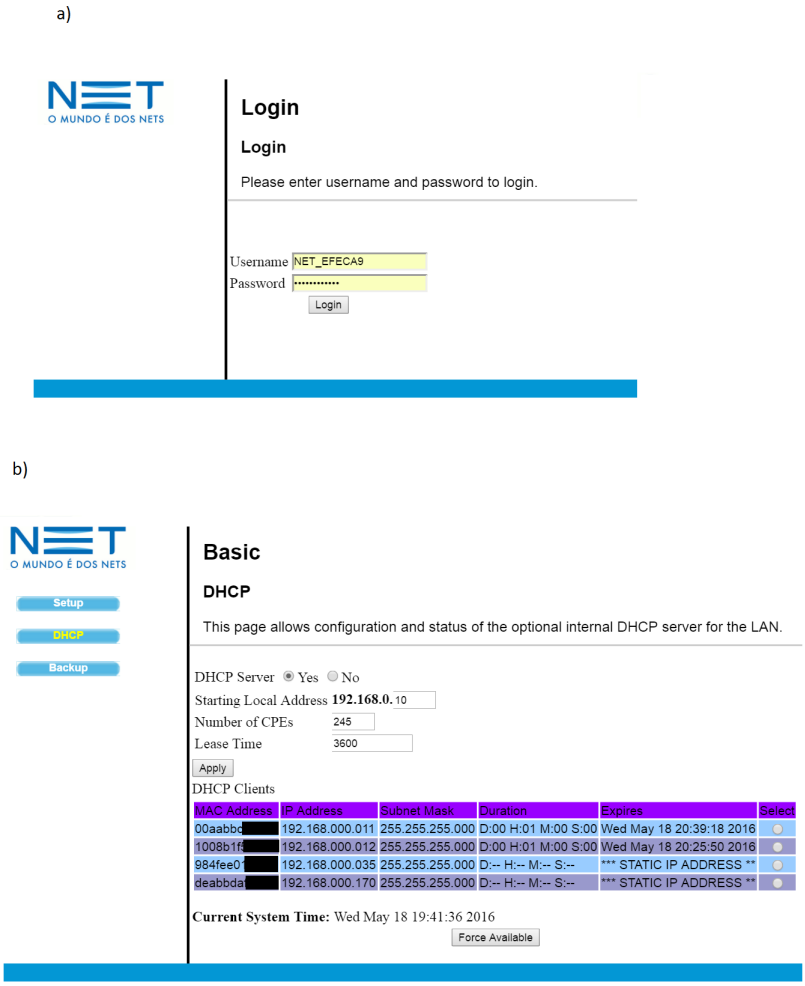


Figura 3.26: a) Página de Login do rotiador b) Página onde os números de MAC são exibidos

WebService

Para a criação do WebService ressaltamos duas partes do código.

A parte de dentro do retângulo verde é responsável pela leitura das URL enviadas ao WebService e pela decisão de setar o pino da placa da Central de Operação como alto ou baixo. A decisão de qual URL será enviada para o WebService é de responsabilidade dos módulos Wi-fi ESP8266.

Após esse primeiro bloco, temos a outra parte da codificação dentro do retângulo azul.

Nessa parte é verificado qual é o estado de cada pino e é tomada a decisão do que deve ser escrito no WebService. É importante ter precisão no que irá ser escrito no WebService pois essas palavras serão utilizadas pelo aplicativo para verificação do *status* da porta e da Janela. Aqui se nota que uma boa prática de implementação é a utilização de *debounces*, ou seja, realizar duas vezes a leitura do estado de cada pino separadas por um breve período, para evitar que ruídos e outros tipos de variações não intencionais acarretem na leitura de um valor errado. A imagem da figura 3.27 mostra o WebService em funcionamento.

Figura 3.27: a) WebService criado pelo Arduino UNO

3.2.4 Bloco 4 - Monitoramento

Para realizar o monitoramento da residência foram utilizadas duas câmeras IP.

Para acessar as configurações da câmera através do seu número de IP é necessário entrar com nome de usuário e senha. Como padrão de fabricação o nome do usuário é admin e não há senha. Porém para que o aplicativo possa acessar a câmera é necessário adicionar uma senha à câmera.

Além disso, dois passos precisam ser feitos para garantir a conexão correta ao aparelho: é necessário fixar o IP do aparelho para que o DHCP não o altere toda vez que a câmera foi inicializada e necessitas-se saber qual porta deve ser utilizada para o acesso a câmera IP.

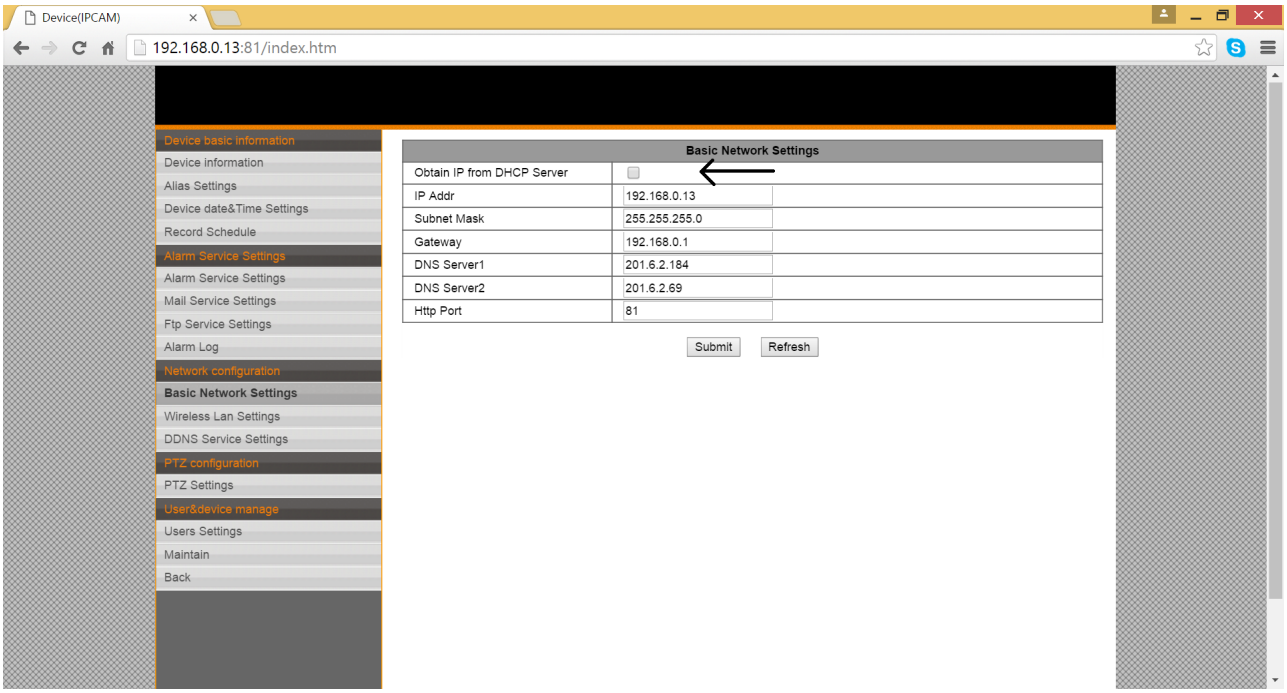
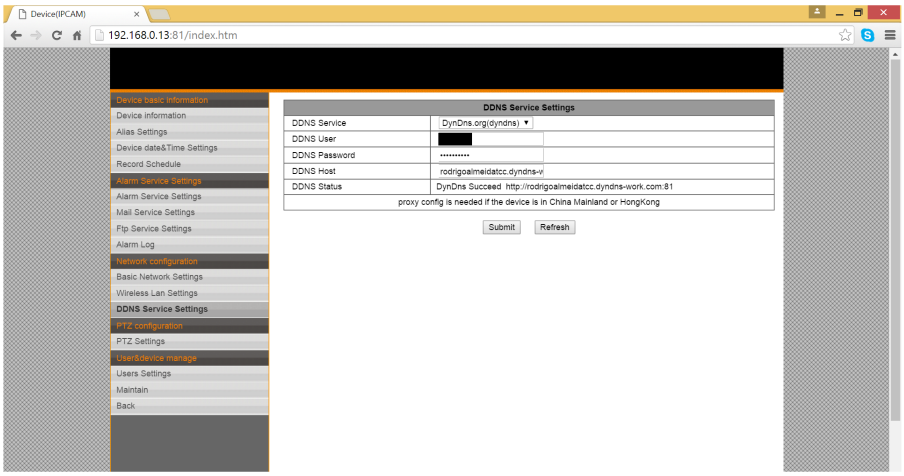


Figura 3.28: Fixando IP da câmera IP

Para poder acessar as imagens da câmera de outras redes que não a conectada ao aparelho é necessário criar-se um site que redirecione o acesso ao endereço de IP da câmera. Há vários serviços assim disponíveis gratuitamente. No caso do projeto foi utilizado o provedor DynDns que é um dos serviços aceitos pelo aparelho.

Após isso, deve-se fazer um PortFowarding no roteador para que o acesso externo ao endereço de IP da câmera seja bloqueado.

a)



b)

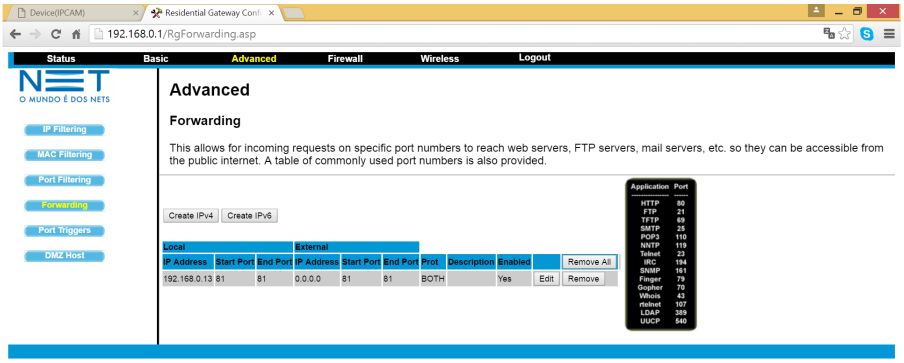


Figura 3.29: a) servidor DDNS b) Port Fowarding

3.2.5 Bloco 5 - Aplicativo

Nessa secção será mostrado o que foi desenvolvido utilizando-se a plataforma *on-line App Inventor*. Como apresentado anteriormente essa plataforma realiza o desenvolvimento do aplicativo em duas janelas diferentes.

Design do App

A primeira mostra como será a tela do aplicativo. Podemos ver que temos tantos componentes visíveis quanto componentes não visíveis (servem para alguma função interna).

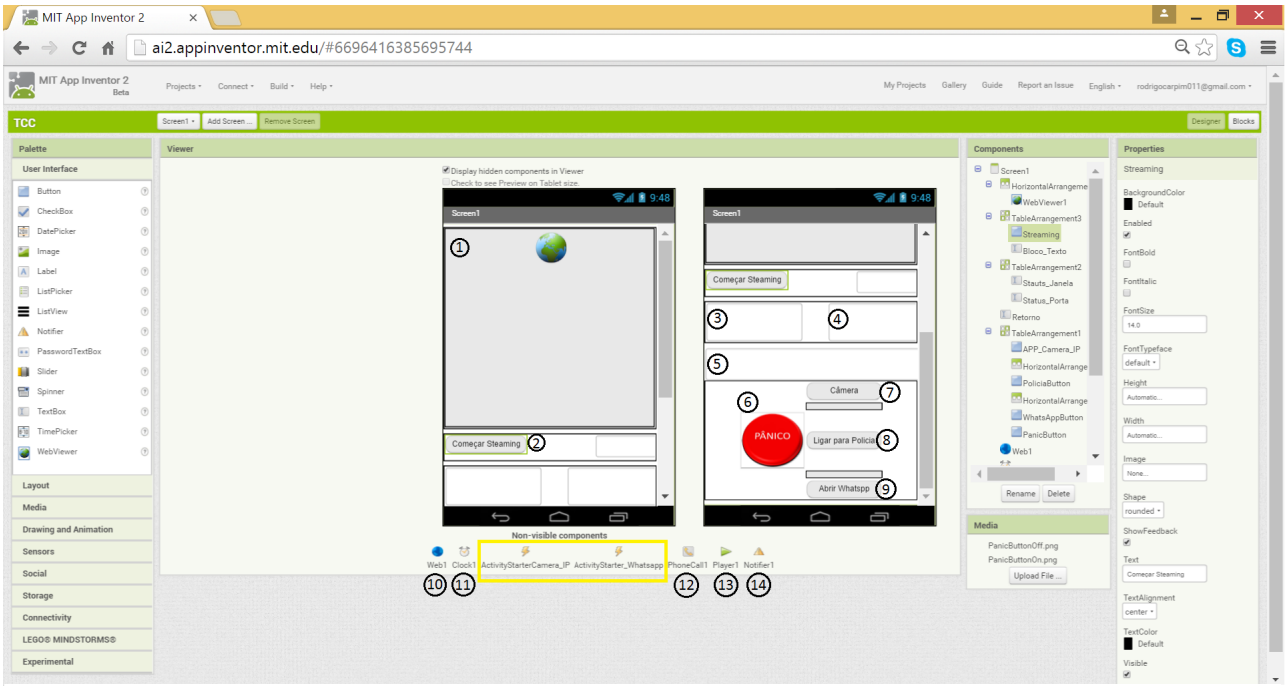


Figura 3.30: Tela de Design

Visíveis:

1. Web Viewer - Permite visualizar uma página da internet, no caso utilizado para se conectar a Câmera Ip TOP CAM.

2. Botão "Começar Streaming- Inicia a conexão com a câmera IP.

3. Caixa de Texto - Retorna o status da Janela

4. Caixa de Texto - Retorna o status da Porta

5. Caixa de Texto - Retorna o código fonte do WebService

6. Botão Pânico - Envia a URL "?panicoon"quando acionado e a URL "?panicooff"quando desativado, o que aciona ou desliga um periférico

7. Botão Câmera - abre o aplicativo da Câmera IP Pan Tilt

8. Botão Ligar para Policia - Liga para a policia

9. Botão Abrir o Whatsapp - Envia, no aplicativo Whatsapp, para um contato escolhido a mensagem: "Socorro! Estou sendo assaltado"

Não visíveis:

10. Web - Permite o recebimento de informações da Internet, tais como, código fonte, URL, etc.

11. Clock1 - Inicia um *timer* em loop de 1 segundo

12. PhoneCall - Possibilita a ligação para um número pré determinado
13. Player - Possibilita vibrar o celular.
14. Notifier - Possibilita a criação de *pop-ups* dentro do aplicativo

Activity Starter

O retângulo amarelo identifica os blocos não visíveis chamados Acivity Starter. É importante dar destaque para esses blocos, pois eles permitem o acesso a outros aplicativos instalados no celular e as configurações internas do mesmo. Isso permite interação entre diversos aplicativos e garante conectividade o que é algo fundamental hoje em dia .

Esse componente possui 8 inputs: Action, ActivityClass, ActivityPackage, DataType, DataUri, ExtraKey, ExtraValue, ResultName. Dependendo do papel que se deseja que o ActivityStarter faça é necessário ou não incluir os 8 inputs.

Abaixo segue o que cada componente Actvity Starter possui de input, assim como sua funcionalidade:

1. ActivityStarterCamera-IP - Permite abrir o aplicativo da Câmera IP Pan Tilt

a. Action: "android.intent.action.MAIN"

b. ActvityClass: "x1.Studio.Ali.GuideActivity"

c. ActivityPackage: "x.p2p.cam"

2. ActivityStarter-Whatsapp - Permite o envio de mensagem pré-determinada pelo Whatsapp

a. Action: "android.intent.action.SEND"

b. ActvityClass: "com.whatsapp.ContactPicker"

c. ActivityPackage: "com.whatsapp"

d. Data Type: "text/plan"

e.ExtraKey: "android.intent.extra.TEXT"

f.ExtraValue: "Socorro! Estou sendo assaltado"

Para abrir outros aplicativos ou configurações dentro do SmartPhone sempre será necessário pelo menos completar: **Action** (define a ação a ser realizada pelo aplicativo, tal como: inicializar, enviar, somente olhar, etc), **ActivityClass** (funcionalidade do aplicativo desejado que se pretende abrir, por exemplo "com.whatsapp.ContactPicker"abre direto a lista de contatos no Whatsapp) e **ActivityPackage** (nome do aplicativo que o sistema android lê).

O maior problema com esse componente é descobrir qual **ActivityClass** e **ActivityPackage** usar. Nesse projeto foi utilizado o aplicativo gratuito "System Info for Android"(figura 3.31) que permite abrir o código fonte de aplicativos instalados no SmartPhone.

A figura 3.32 mostra o código fonte de um aplicativo. Dentro do código fonte devemos procurar o bloco de código de "action"(seta azul) para poder encontrar o ActivityClass (seta verde). A seta amarela mostra onde podemos encontrar o ActivityPackage.

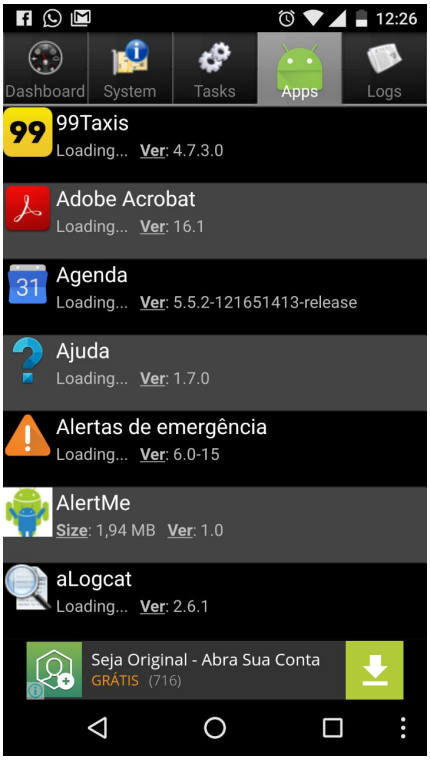


Figura 3.31: Aplicativo System Info for Android

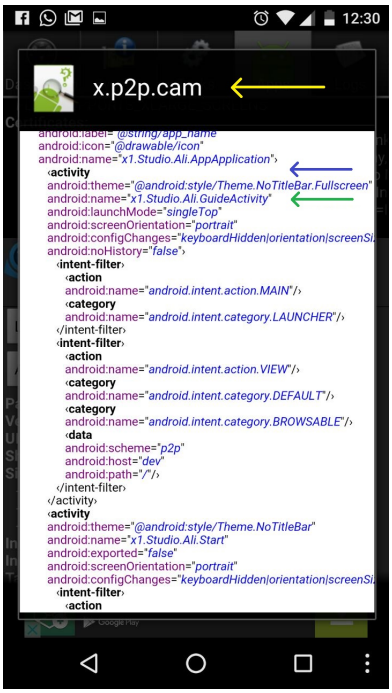


Figura 3.32: Código Fonte Aplicativo da Câmera Pan Tilt

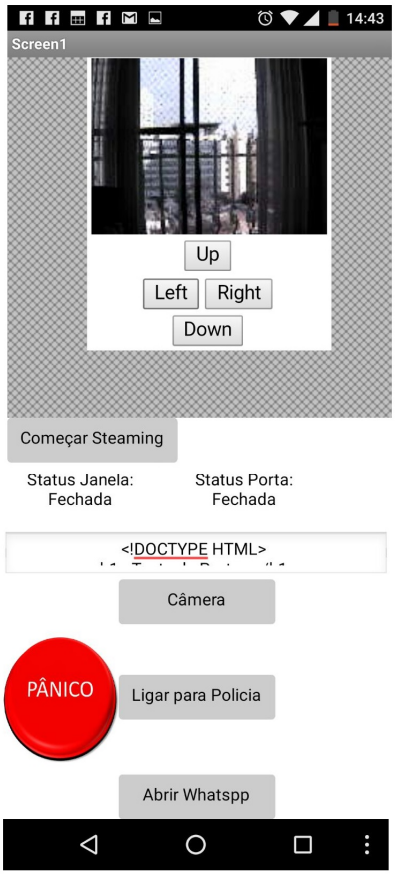


Figura 3.33: Tela do Aplicativo funiconando

A figura 3.33 mostra a tela do aplicativo funcionando. A letra a) da figura 3.34 mostra uma simulação do que acontece quando alguem abre a porta e a letra b) simula a resposta do aplicativo quando o botão "Pânico"é pressionado.

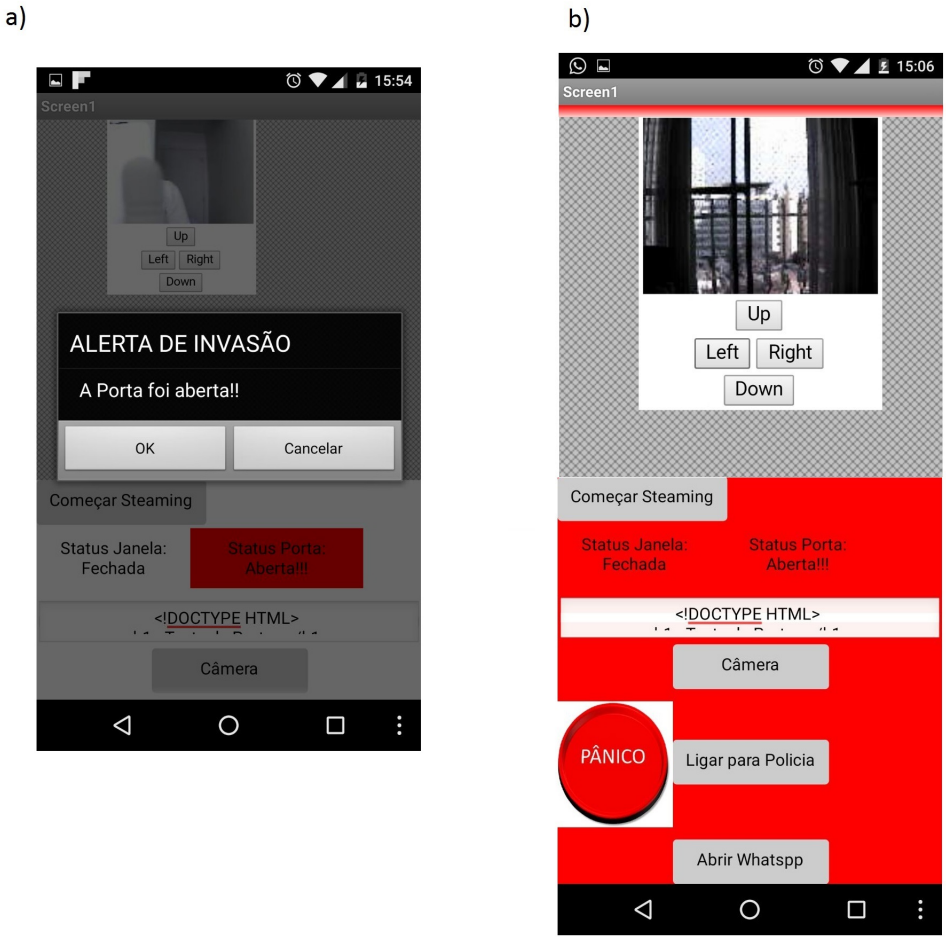


Figura 3.34: a) resposta do aplicativo quando abre-se a porta b) resposta do aplicativo quando se pressiona o botão Pânco

Blocks de codificação

A codificação dessa interface é feita na aba *blocks* da plataforma. Para melhor entendimento os blocos de codificação foram divididos em: **Variáveis globais**, **Leitura WebService**, **Notificação**, **Botões** e **Botão Pânico**.

Variáveis globais

A figura 3.35 mostra todas as variáveis utilizadas.

É importante notar que nessa plataforma o uso de variáveis como flags é comum, pois não há muito link entre os blocos, ou seja, não se consegue carregar um valor resultado do mesmo jeito que em um código C++ ou da IDE Arduino.

As variáveis indicadas pela seta amarela são utilizadas como flags, já a azul é utilizada para indicar o endereço em que pode ser encontrado o WebService e a seta em verde mostra a variável que receberá qual texto deve aparecer na notificação de invasão.

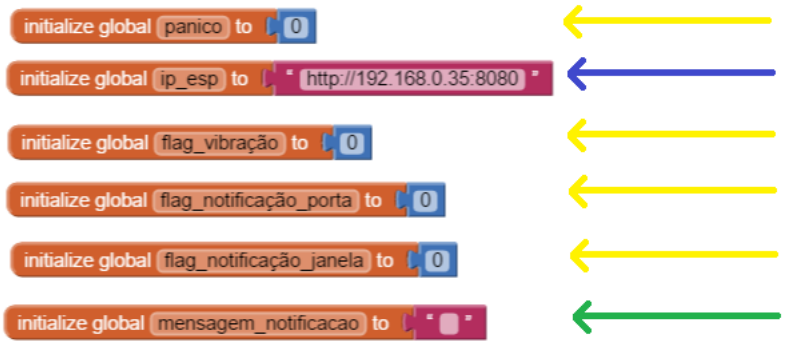


Figura 3.35: Variáveis Globais utilizadas na codificação do aplicativo

Leitura WebService

O bloco apresentado abaixo garante que o aplicativo leia o código fonte do WebService (criado pela Central de Operação) e verifique se existe alguma das seguintes palavras pré-definidas: JanelaOFF, PortaOFF, NORMAL, PANICO, JanelaON, PortaON.

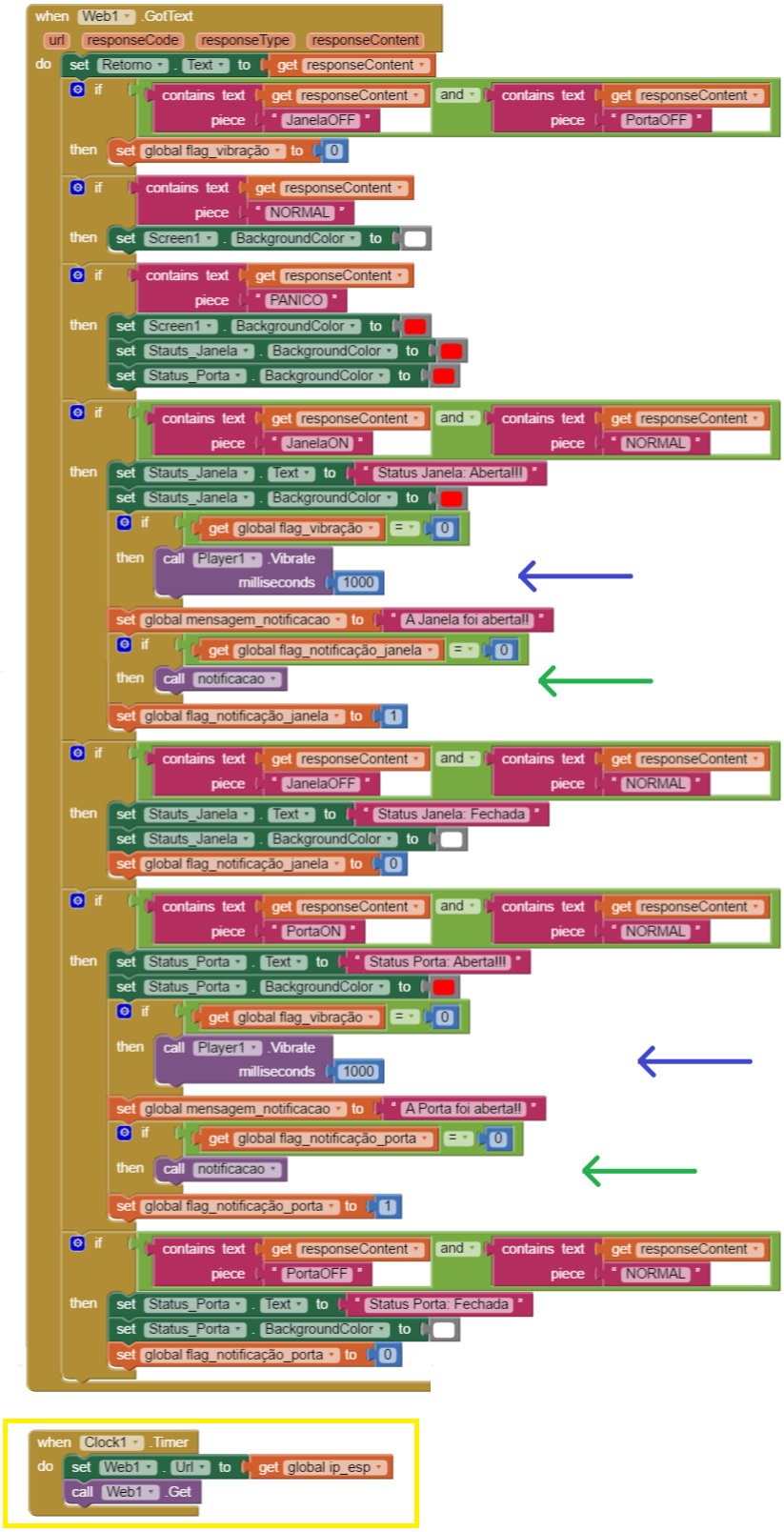


Figura 3.36: Bloco responsável pela Leitura do WebService

Toda a lógica do sistema depende dessa leitura, por isso é extremamente importante que as palavras pré-definidas batam com as programadas dentro da Central de Operação.

A seguinte lógica é usada:

- JanelaOFF: Indica que a janela está fechada
- PortaOFF: Indica que a porta está fechada
- NORMAL: Indica que o botão de pânico não foi acionado
- PANICO: Indica que o botão de pânico foi acionado
- JanelaON: Indica que a janela foi aberta
- PortaON: Indica que a porta foi aberta

É interessante notar que quando a porta ou janela é aberta o fundo da caixa de texto muda para vermelho, uma notificação é acionada (seta verde) e o celular vibra (seta azul).

Além disso, uma parte extremamente importante é o bloco destacado dentro do retângulo amarelo. Com esse bloco garantimos que o processo de leitura seja realizado a cada 1 segundo devido a componente Clock1 adicionada na tela de Desing.

Pop-up

O bloco abaixo cria um *pop-up* que avisa o que foi aberto, a janela ou a porta. É possível notar que após se responder ao *pop-up* (clicando no botão "OK- letra a) figura 3.34) o celular para de vibrar, uma vez que a variável global flag-vibração é setada para 1.

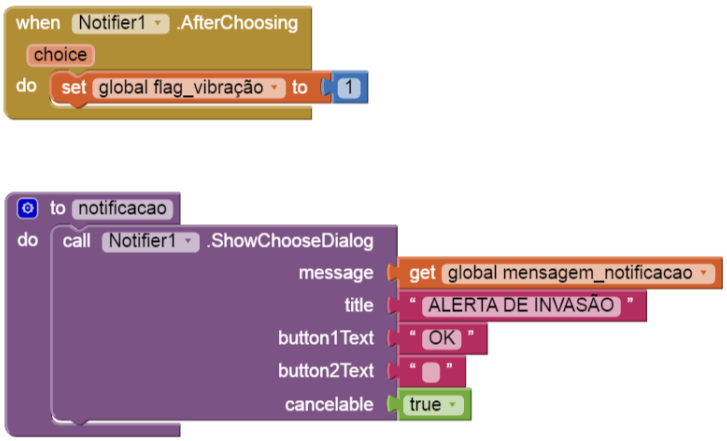


Figura 3.37: Bloco responsável pela Notificação

Botões

Para o botão "Ligar para Policia"basta colocar 190 no input na área de Desing.

Os botões "Câmera"e "Abrir Whatsapp"simplesmente iniciam as respectivas Activity Starter (explicadas anteriormente).

Nesse bloco a parte mais importante se trata do botão "Começar Streaming"que atrela ao WebViewer o endereço da Câmera IP TOP CAM. Além disso, a parte em negrito na URL mostra o usuário ("admin") e a senha ("tcc") necessários para acessar a câmera.

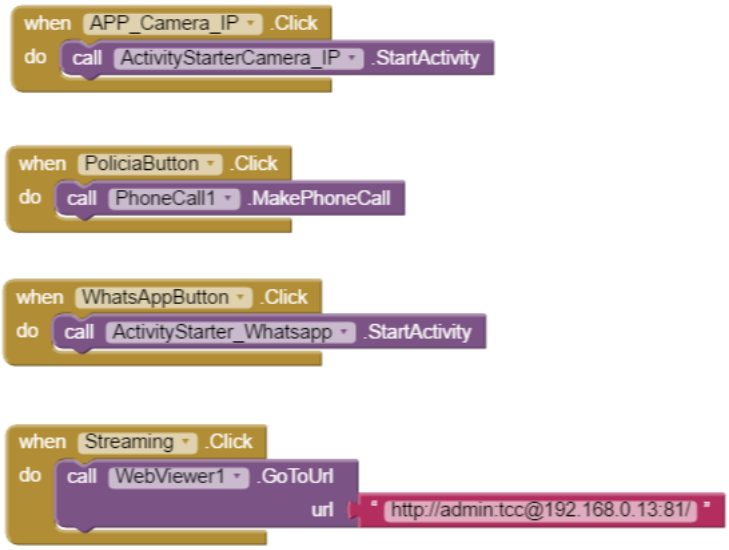


Figura 3.38: Bloco responsável pelos botões, sem ser o de Pânico

Botão Pânico

Primeiramente foram adicionadas duas imagens, uma com o botão de pânico sem estar pressionado e outra com ele estando pressionado. Isso foi necessário para se criar a ilusão de se estar apertando o botão Pânico.

O bloco abaixo garante a troca da imagem do botão Pânico toda vez que ele é apertado (função da variável "panico") e o envio das URLs ?panicoon e ?panicooff.

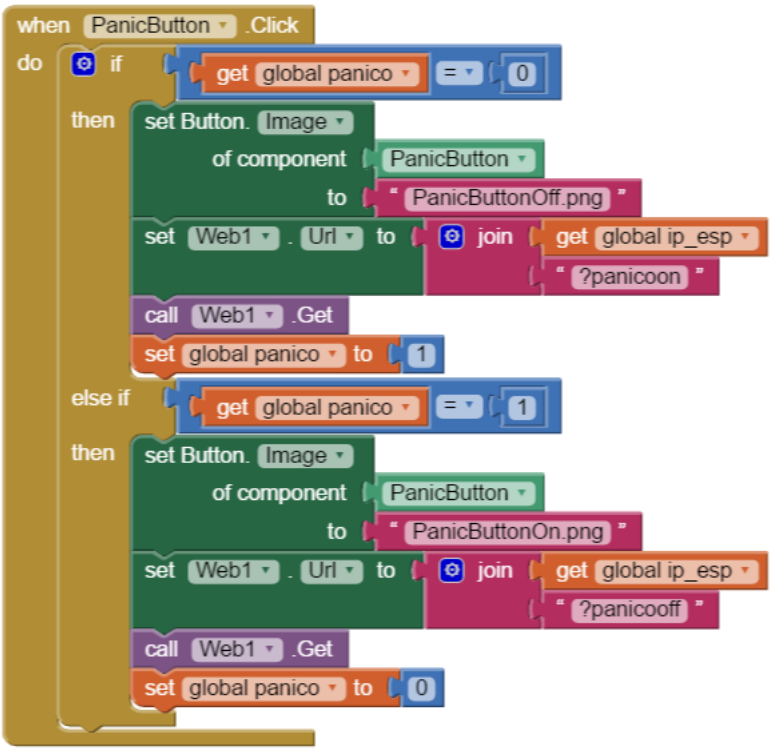


Figura 3.39: Bloco responsável pelo botão de Pânico

3.2.6 Bloco 6 - Integração completa

Nessa secção será mostrado como foram integrados todos os componentes utilizados no projeto.

A foto abaixo mostra todos os blocos conectados.

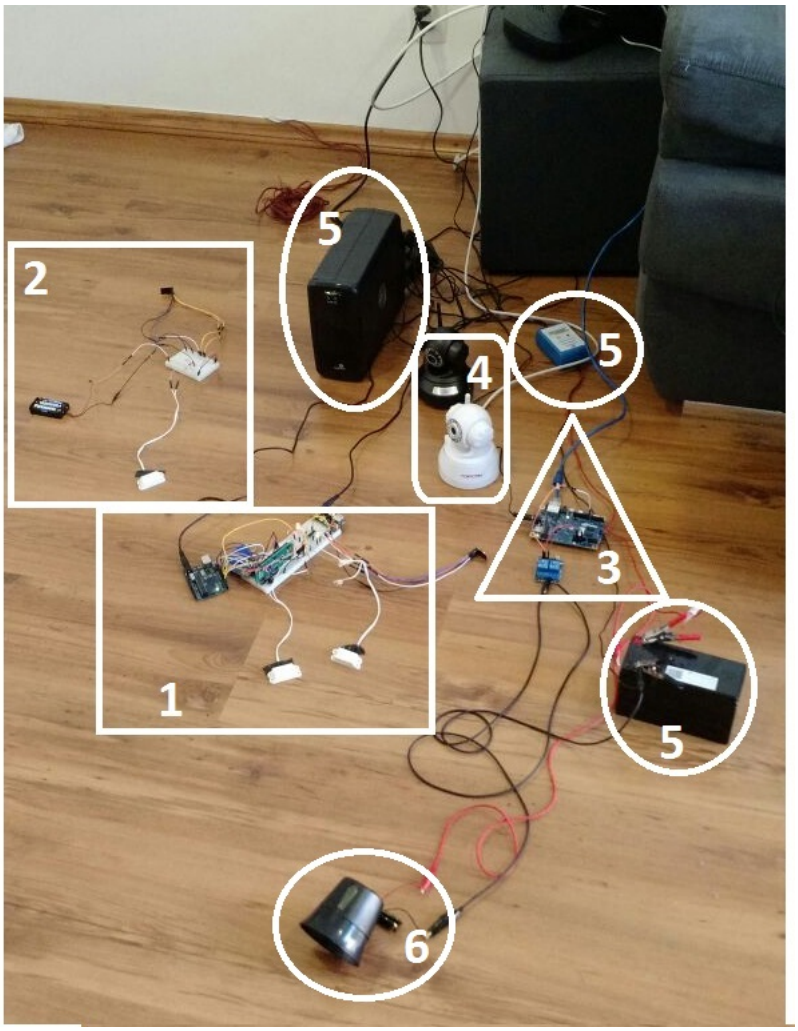


Figura 3.40: Sistema Integrado

1. Acesso: Arduino + Dois Sensores Magnético + ESP8266-01 + Fontes (fonte do arduino e a Ywrobot, ambas conectadas ao Nobreak)
2. Sensor da Janela: ESP8266-01 + Sensor Magnético + Pilhas
3. Central de Comando: Relé + Intel Galileo + Fonte do Galileo (ligada ao Nobreak)
4. Monitoramento: Câmeras IP + Fontes (Ambas ligadas ao Nobreak)
5. Independência da Rede Elétrica: Nobreak + Bateria 12V + Carregado 12V
6. Periférico: Sirene (ligada ao Relé e a Bateria 12V)

O esquema de ligações abaixo mostra no detalhe como ficou a integração dentro de cada Bloco:

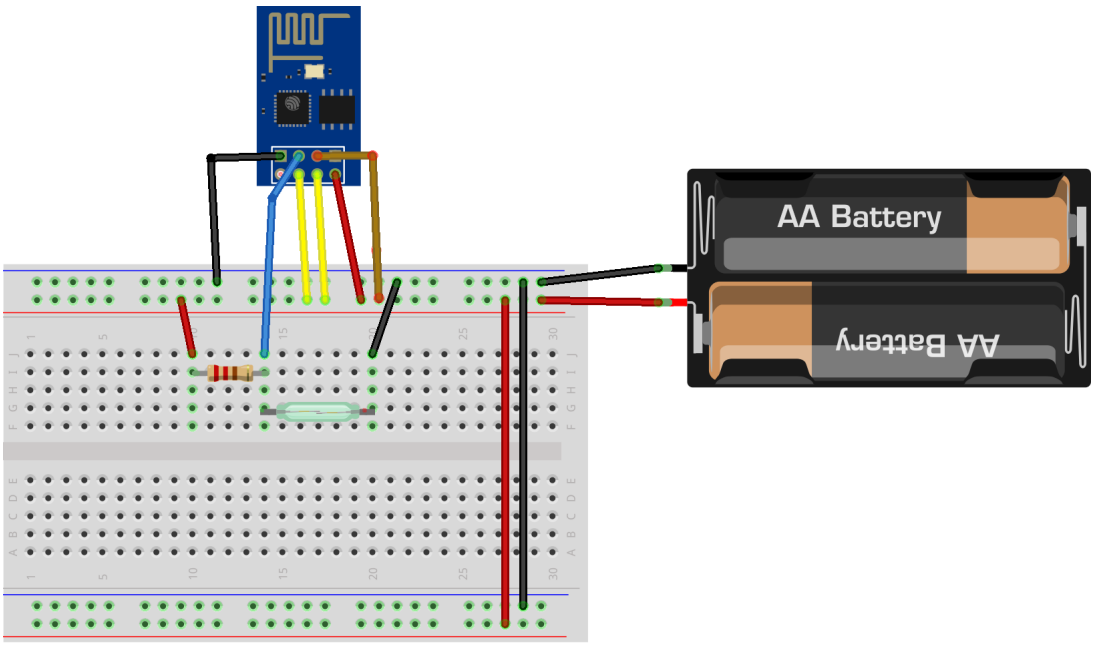


Figura 3.41: Montagem do Sistema de Transmissão de Staus da Janela

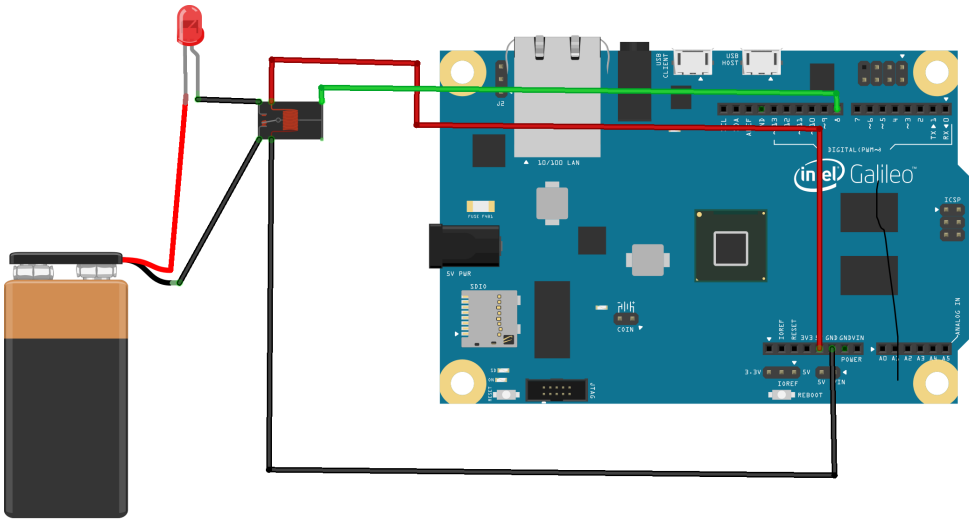


Figura 3.42: Monantagem da Central de Controle (periféricos representados pelo LED)

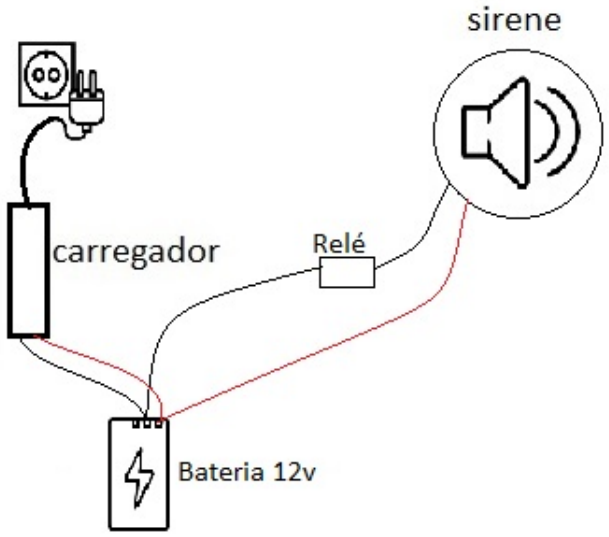


Figura 3.43: Imagem esquemática da conexão entre carregador 12V, bateria, relé e sirene

É importante notar na imagem 3.41 que o pino GPIO-0 dos dois módulos ESP8266-01 estão conectados na alimentação. Isso é necessário para que o componente entre em modo de operação utilizando o *firmware* instalado.

A figura 3.44 retoma o esquemático geral do projeto, seguido pelos fluxogramas de decisão de cada bloco:

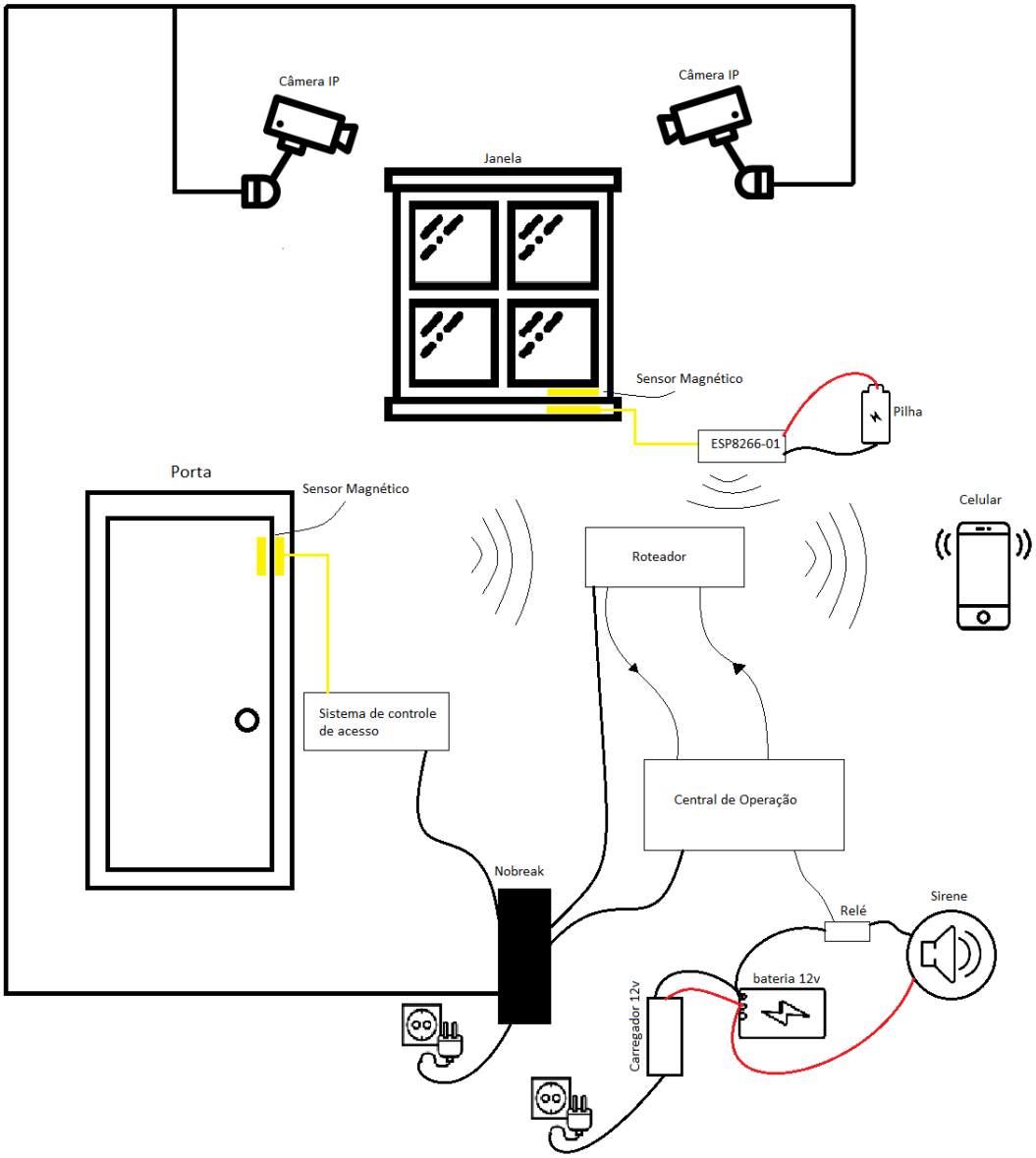


Figura 3.44: Imagem esquemática do projeto [26]

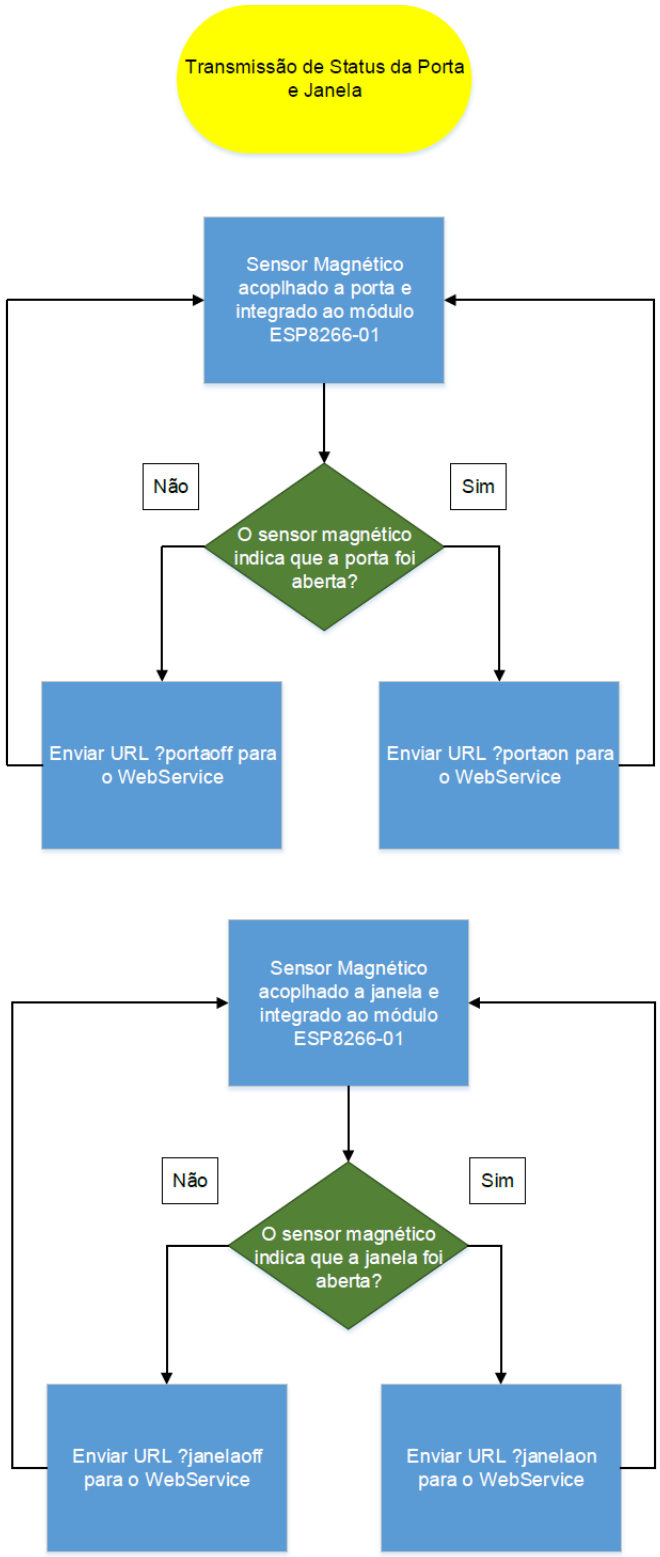


Figura 3.45: Fluxograma de transmissão de status

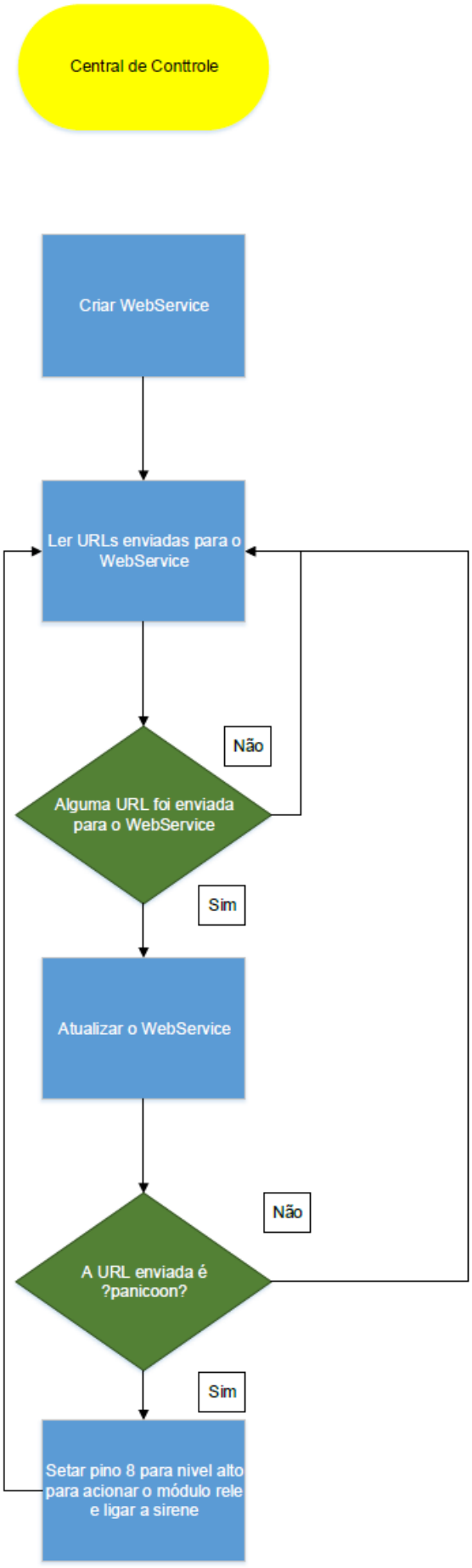


Figura 3.46: Fluxograma da Central de Controle

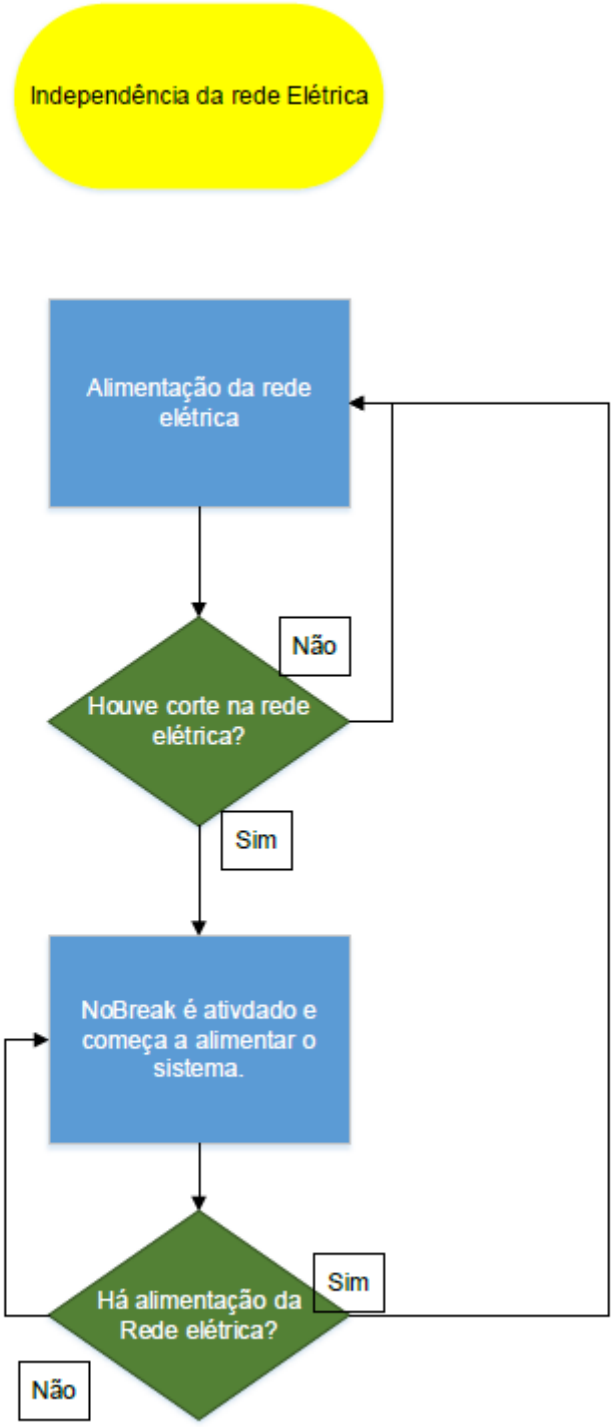


Figura 3.47: Fluxograma de utilização de fontes independentes da rede elétrica

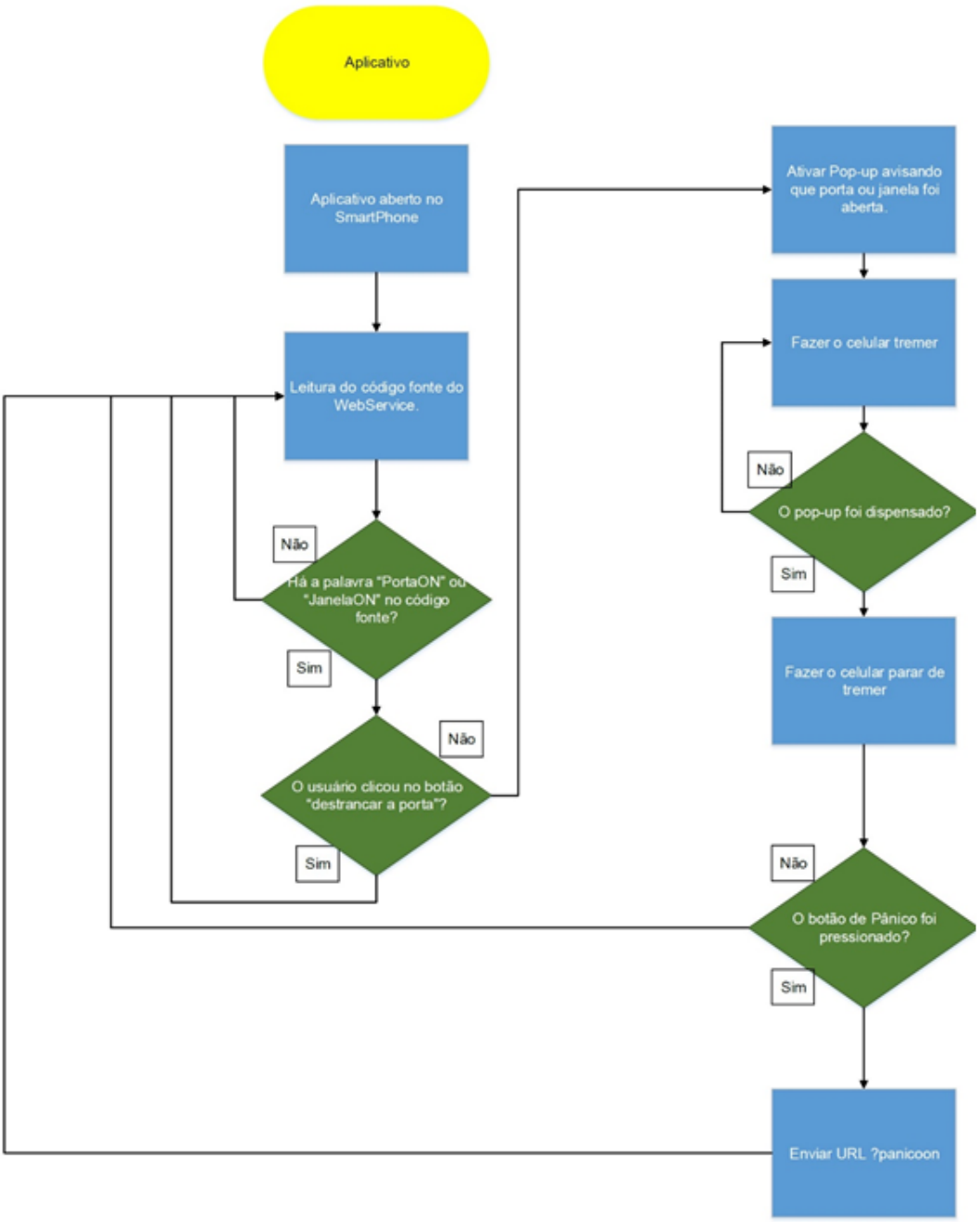


Figura 3.48: Fluxograma do aplicativo

Capítulo 4

Resultados

A seguir são apresentados os resultados do projeto. Esse capítulo também será divido em subsecções para facilitar a compreensão.

4.1 Bloco 1 - Acesso

O sistema implementado para controlar o acesso realizou sua função satisfatoriamente. Observou-se que por utilizar muitos *jumpers* para realizar a conexão entre protoboard, arduino, LCD e módulo RFID o sistema as vezes apresentou algumas pequenas falhas por mau contato.

As principais desvantagens se encontram em ter que ajustar o código no Arduino depois de descobrir qual o número de identificação das *tags*, e na utilização de um outro embarcado, além do módulo RFID, para fazer o controle e a identificação de acesso.

Outro ponto limitante é a necessidade de ter que usar algo físico (as *tags*) para se ter acesso a residência. É possível eliminar esse empecilho, implementando algo como por exemplo um sistema NFC (integrado com o SmartPhone) ou um botão no aplicativo.

4.2 Bloco 2 - Transmissão de Status

Para realizar a transmissão do status da porta e da janela foram testados dois modos diferentes: utilizando o RF433MHz e utilizando o módulo Wi-fi ESP8266-01.

A tabela a seguir mostra uma comparação dos componentes:

Componente:	Módulo Wi-fi ESP8266-01	RF433MHz
Raio de transmissão	até 350 m	até 200 m
Confiabilidade de transmissão	Alta. Não sofre muita interferência	Baixa. Qualquer obstáculo bloqueia a transmisão
Consumo de Corrente transmitindo	até 215mA	até 15mA
Consumo de Corrente sem transmitir	até 1,2mA	0 A
Taxa de Transferência	até 54Mbps	4Kbps
Necessidade de processamento externo	Não	Sim
Preço médio	15,00 reais	8,00 reais

Tabela 4.1: Tabela Comparação ESP8266-01 versus RF433MHz (Fontes: [25], [24], [27])

Pode-se observar que sem nenhuma dúvida o módulo Wi-fi supera o RF433MHz. Apesar de consumir mais corrente o módulo fica na frente em todos os outros quesitos. Maior distância máxima de transmissão com alta confiabilidade, maior taxa de transmissão e, o principal, independência de processamento externo. Esse último quesito é o que transforma o módulo ESP8266-01 em um grande componente eletrônico.

A necessidade da utilização de um processador externo (um arduino UNO por exemplo) infla o custo da transmissão utilizando o RF433MHz em pelo menos três vezes o preço do ESP8266.

Ressalta-se que apesar do módulo ESP8266-01 não vir pronto de fábrica para a utilização como um Arduino seu poder de processamento juntamente com sua simplicidade, tamanho e praticidade de conexão à rede Wi-fi o fazem uma parte central desse projeto e supre com facilidade todas as necessidades.

Essa família de módulos fortalece muito o movimento da *Internet of Things*. Ter a facilidade de conexão à internet em componentes tamanho reduzidos tornam a criatividade o limitante em projetos.

Para casos que sejam necessários outros requisitos, a família ESP8266 possui diversos outros módulos. Podemos citar alguns como:

- ESP8266-12E: Assim como o modelo 01, o ESP8266-12E possui uma antena interna,

mas como a vantagem de dispor de 11 pinos GPIO, o que abre possibilidades diferentes de uso para o módulo.

Figura 4.1: Módulo ESP8266-12E [8]

- Node MCU ESP8266-12E: O Módulo ESP8266 NodeMCU ESP-12E é uma placa de desenvolvimento completa, que além de possuir o componente ESP8266-12E conta com um conversor TTL-Serial e um regulador de tensão 3.3V. O modelo acaba com os problemas de prototipação da família ESP8266 tornando-o tão simples quanto o Arduino.

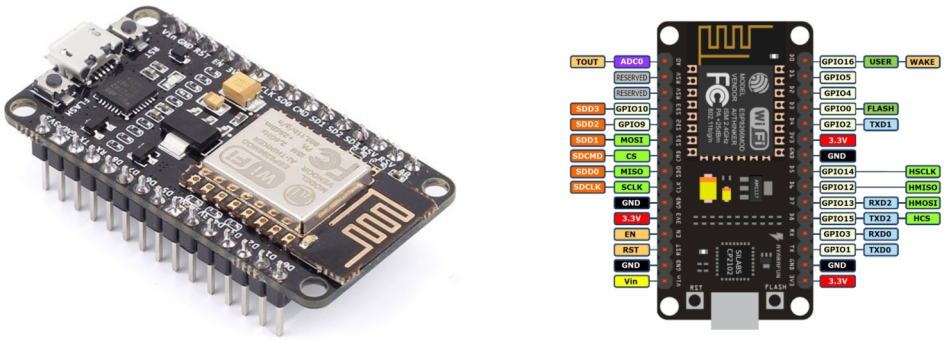


Figura 4.2: Módulo NodeMCU ESP8266-12E [8]

Em contrapartida aos seus atributos esses dois modelos acabam sendo mais caros do que o modelo básico 01 (30,00 reais e 60,00 reais comparados com 15,00 reais do modelo 01).

4.3 Bloco 3 - Central de operação

Para servir como Central de Operação foram testados dois modos diferentes: utilizando

um Arduino UNO mais um *Ethernet Shield* e utilizando o embarcado Intel Galileo.

Ambos os dois modos tiveram bons resultados, porém o Arduino mostrou-se incapaz de fornecer a corrente necessária para o acionamento correto do módulo rele. Abaixo iremos apontar as vantagens e desvantagens de cada modo.

Podemos citar como vantagens do Arduino UNO:

- Facilidade de Implementação: O aparelho vem direto da fábrica pronto para a utilização. A codificação é simples e há bastante suporte, tanto dos fabricantes quanto da comunidade utilizadora dos componentes.

- Preço: Por ser um sistema *Open Source* existem inúmeros fabricantes de placas tipo arduino o que torna seu preço bem acessível (por volta de 30,00 reais).

Podemos citar como desvantagens do Arduino UNO:

- Limitação de processamento: Por ser um dos mais simples da família arduino o embarcado não é capaz de rodar códigos muito grandes ou que exijam muita capacidade de processamento, tais como processamento de imagens/vídeo.

- Conectividade: O Arduino UNO em si não agrega muito nas tendências da *IoT*, é preciso a utilização de *Shields* ou outros módulos para que ela seja acessível *Wireless* o que aumenta o custo de sua utilização (o preço médio de um *Ethernet Shield* é por volta de 50,00 reais)

- Corrente: O Arduino UNO possui uma limitação na quantidade de corrente que ele consegue fornecer e receber. 40mA e 200mA são respectivamente a corrente máxima fornecida por um pino e a corrente total que pode ser oferecida pela placa. Esses fatores limitam a utilização de componentes agregados a placa e quando ultrapassados podem causar danos ao equipamento ou o desligamento do mesmo.

Podemos citar como vantagens da placa Intel Galileo:

- Utilização da IDE Arduino: A placa possibilita a utilização da interface simples do IDE para acesso e programação o que a torna muito mais amigável ao usuário.

- Linux embarcado: Além de poder servir como um Arduino, a Intel Galileo possibilita a utilização de sistemas Linux (32bits) o que a torna ideal para projetos que necessitem de processamento de algoritmos computacionais complexos.

- Corrente: Apesar de não poder receber muita corrente (25mA por entrada) a placa Galileo consegue oferecer até 800mA, quatro vezes mais do que o Arduino.

Podemos citar como desvantagens da placa Intel Galileo:

- Preço: Com um custo por volta de 500,00 reais (Intel Galileo Gen2) a placa se torna inviável para aplicações que não utilizarão todo o seu potencial.

- Necessidade de atualização da firmware: Apesar de possibilitar o uso da IDE para interface a Galileo precisa passar por um setup inicial.

Vemos que ambas as duas placas possuem características marcantes e que se há a necessidade de utilizar uma embarcado com maior poder de processamento, a Galileo supera muito o Arduino, compensando o seu preço mais elevado.

4.4 Bloco 4 - Monitoramento

Existem no mercado muitas camêras IP. Ambos os modelos tinham boa funcionalidade, porém a câmera TOP CAM não obteve regularidade no sinal e perdia o contato com a rede. Além disso, só foi possível acessar as configurações internas dessa câmera, pois a porta de acesso do modelo Pan Tilt não é informado no software do aparelho.

Esse modelo de câmera, por possuir preço relativamente acessível (encontra-se no mercado câmeras entre 130,00 a 250,00 reais) se mostrou excelente para monitoramento residencial e Home-Automation.

4.5 Bloco 5 - Aplicativo

O Aplicativo criado através da plataforma AppInventor mostrou grande velocidade de resposta e inúmeras possibilidades de uso.

Comparado com outras plataformas de desenvolvimento de aplicativos o AppInventor se demonstrou mais simples e superior a grande maioria.

Podemos citar como sua maior vantagem a utilização de blocos para a codificação do aplicativo e a possibilidade de teste do aplicativo durante o seu desenvolvimento o que permite grande flexibilidade durante a construção do mesmo. Sua maior desvantagem se encontra pela limitação de funcionalidades. Com um número limitado de blocos de codificação e sem

a possibilidade de criação de novos blocos algumas vezes para criar coisas simples como um delay é necessário há utilização de vários blocos.

Outro fator negativo da plataforma é a falta de conexão entre dois blocos. Isso força a utilização de variáveis globais como flags toda vez que se deseja carregar algum valor de um bloco para o outro. Fora isso a plataforma se mostrou muito útil na construção do projeto e uma grande aliada na temática de Home-Automation.

4.6 Bloco 6 - Integração completa

Durante a integração completa foram observados alguns problemas.

Devido à perda de conexão da câmera IP algumas vezes o aplicativo não pode encontrar o endereço da mesma e não exibiu nenhuma imagem (figura 4.3).

Figura 4.3: Falha na conexão com câmera IP

Inicialmente o ESP8266-01 apresentou algumas falhas no envio do status da porta e janela. Porém, ajustando-se sua codificação (seta verde figura 3.24) esse problema teve uma incidência muito pequena e o sistema conseguiu uma resposta adequada, perto de 0,5 segundo entre abertura da porta ou janela e o aviso no SmartPhone.

A conexão por DDNS também apresentou problema. Não foi possível identificar a causa do problema pois tanto a rede de internet quanto o site provedor do serviço de DDNS apresentaram instabilidade. Além disso, há a possibilidade do *firewall* do modem bloquear a conexão por motivos de segurança o que traz mais uma dificuldade para a identificação do problema raiz.

Fora esses pontos o sistema funcionou muito bem, com resposta adequada a invasão,

notificação chamativa (fazer o SmartPhone tremer até o *pop-up* no aplicativo ser dispensado), funcionalidade correta do acesso a residência e boa resposta ao acionamento de periféricos.

4.7 Resultados da Utilização do Nobreak, Bateria 12V e Pilhas

O aparelho de Nobreak obteve ótima atuação e, em testes realizados, possibilitou o funcionamento do sistema por até 35 minutos sem que houvesse alimentação da rede elétrica.

A bateria 12V também serviu perfeitamente para alimentar a sirene. A bateria continuou à funcionar após 90 minutos de uso contínuo, por isso, o Nobreak é o limitante energético nesse sistema.

Para a alimentação do módulo Wi-fi responsável pela transmissão de status da janela, utilizou-se uma montagem com pilhas Philips. Apesar de ser possível o acionamento do ESP8266 o mesmo funcionou somente 10 minutos até o esgotamento das pilhas.

Para o módulo responsável pela transmissão de status da porta foi-se decidido em utilizar uma fonte de 3,3 Volts ao invés de pilhas uma vez que pode se aproveitar a prototipação do sistema de acesso.

Além disso, não foi implementado um sistema de pilhas recarregáveis pois o preço de um carregador mais suas pilhas (por volta de 140,00 reais) não compensa sua eficiência energética, tendo em vista que o Nobreak (250,00 reais) teve um resultado satisfatório em relação ao tempo de funcionamento do sistema.

Capítulo 5

Conclusões

O projeto se apresentou como um grande desafio de finalização do curso de graduação, abordando desde conhecimentos ministrados durante o curso até assuntos atuais não contemplados na sala de aula.

Acima disso tudo, deve-se destacar que todos os conhecimentos novos trazidos com o projeto e principalmente a lógica e raciocínio utilizados em cada etapa de desenvolvimento, só foram possíveis por conta da base adquirida ao longo da graduação.

Esse trabalho possibilitou o contato com diversos conceitos atuais (*Home-Automation*, segurança eletrônica residencial, *Internet of Thing*, sistemas embarcados, aplicativos de Smart-Phone) e com uma comunidade ativa em todas as áreas abordadas facilitando o entendimento e o desenvolvimento de todas as partes do projeto.

Os resultados do projeto nos levam a concluir que:

- No controle de acesso à residência pode-se optar por utilizar modos com foco na facilidade de utilização (como o módulo RFID) ou com foco na utilização de meios digitais (como NFC ou uma função no aplicativo). Tendo em vista as linhas do projeto conclui-se que a melhor opção seria o uso de meios digitais que vão ao encontro com a ideia da *IoT*.

- Na transmissão de status conclui-se que o módulo ESP8266-01 foi excelente para o projeto e que o mesmo (ou outros da sua família) pode ser utilizados em diversos trabalhos de *Home-Automation*.

- Sobre a central de operação constatou-se que ao se fazer uma análise, a placa Intel Galileo não é a ideal para o projeto por ter um custo muito mais elevado do que o Arduino UNO (juntamente com seu *Ethernet Shielde*). Este último supriu as necessidades do projeto e por possuir fácil prototipação e interface vemos que os pontos de atenção (como o limite de corrente fornecida) podem ser facilmente contornados.

- As câmeras IP se mostraram grande aliadas na questão da segurança residencial. Com um preço acessível a câmera por si só já é um ótimo componente em qualquer residência. Na ótica do projeto as câmeras IP trazem o melhor custo/benefício (apesar dos problemas relatados em na secção 4.2.5) quando comparadas aos sistemas profissionais de filmagem e a câmera de preço e qualidade inferiores.

- Os sistemas de alimentação projetados para suprir as necessidades elétricas do sistema em caso de queda da rede elétrica se mostraram suficientemente eficientes. É importante notar aqui que o uso de Nobreak para realizar a independência da rede elétrica é a solução com o melhor custo benefício visto que consegue suportar a necessidade do projeto sem criar custos altos.

- O aplicativo desenvolvido, além de abrir portas para novos projetos, provou ser eficiente e simples de usar. A plataforma de desenvolvimento (o AppInventor) é um dos grandes centros desse trabalho e é uma recomendação à projetos de *Home-Automation*.

Por este trabalho estar organizado em forma de tutorial espera-se que ele possa auxiliar no desenvolvimento de soluções para segurança residencial que utilizem sistemas embarcados ou comunicação wireless ou sensores.

Por fim, conclui-se que o projeto contemplo de um modo geral o propósito do trabalho desde da integração dos sensores magnéticos com os módulos Wi-fi, passando pelo monitoramento da residência, pelo sistema de alerta de invasão, pelo controle de periférico até a criação do aplicativo para a integração de todos os atributos.

5.1 Trabalhos futuros

Como um dos objetivos do projeto, a análise para tornar o sistema eficiente nos pontos de vista energético e de transmissão de dados não foi realizada. Por isso, em trabalhos futuros esse ponto deve ser abordado.

Além disso, podemos citar alguns outros pontos que podem complementar o projeto:

- Estudo de outros tipos de embarcados para outras aplicações (como reconhecimento facial)

- Otimização dos componentes utilizados para maior velocidade de resposta.

- Maior integração entre câmeras IP e aplicativo.

- Utilização de outros periféricos para maior automatização residencial.

- Incorporação de módulo GSM para independência da rede Wi-fi

- Utilização dos modos low power dos componentes para economizar energia

- Criação de detecção e aviso do nível das baterias/pilhas

- Utilização de outros métodos de notificação para diminuir o tráfego de dados

- Utilização da central de comando de modo mais ativo (notificando o celular ao invés do celular buscar as informações no WebService)

Podemos finalizar constatando que o sistema poderia ser utilizado para a criação de um produto de segurança residencial, mas para isso seria necessário um estudo de fabricação em larga escala e utilização de chips e outros componentes com interface menos amigável para que o custo fosse barateado.

Referências Bibliográficas

[1] Towards a Definition of the Internet of Things (IoT) . http://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Revision1_27MAY15.pdf, Acesso em: 11 de Junho de 2016. Autor: Roberto Minerva, Abyi Biru, Domenico Rotondi.

[2] Autor: Paula, Marcela Elisa Jacob de. Estudo de Segurança Eletrônica Patrimonial. http://www.ppgee.ufmg.br/defesas/479M.PDF, Acesso em: 11 de Junho de 2016.

[3] About LUA. http://www.lua.org/about.html, Acesso em: 11 de Junho de 2016.

[4] Comandos AT. https://sites.google.com/site/toucatronic/zigbee/comandos-at, Acesso em: 11 de Junho de 2016.

[5] Autor: Jack G. Ganssle, Michael Barr. Embedded Systems Dictionary. , Edição: 2003.

[6] National Association of Home Builders. http://www.nahb.org/, Acesso em: 11 de Junho de 2016.

[7] Apresentando o modulo esp8266. http://www.embarcados.com.br/modulo-esp8266/, Acesso em: 29 de novembro de 2015.

[8] Site FlipeFlop, imagens. http://blog.filipeflop.com/, Acesso em: 20 de maio de 2016.

[9] Cd4050be datasheet (pdf) - texas instruments. http://pdf1.alldatasheet.com/datasheet-pdf/view/26878/TI/CD4050BE.html, Acesso em: 29 de novembro de 2015.

[10] How to spot a counterfeit Arduino. https://www.arduino.cc/en/Products/Counterfeit, Acesso em: 11 de Junho de 2016.

[11] Arduino uno. http://www.embarcados.com.br/arduino-uno/, Acesso em: 29 de novembro de 2015.

[12] W5100 Datasheet. https://www.sparkfun.com/datasheets/DevTools/Arduino/W5100_Datasheet_v1_1_6.pdf, Acesso em: 11 de Junho 2016.

[13] Intel galileo. https://www.embarcados.com.br/intel-galileo/, Acesso em: 15 de Abril de 2016.

[14] Mit app inventor. http://appinventor.mit.edu/explore/, Acesso em: 29 de novembro de 2015.

[15] Pl 2303 windows drive download. http://www.prolific.com.tw/US/ShowProduct.aspx?p_id=225&pcid=41, Acesso em: 29 de novembro de 2015.

[16] How to use esp8266 esp-01 as a sensor web client. https://importhack.wordpress.com/2014/11/22/how-to-use-ep8266-esp-01-as-a-sensor-web-client/#more-170, Acesso em: 29 de novembro de 2015.

[17] Previous ide releases. https://www.arduino.cc/en/Main/OldSoftwareReleases#previous, Acesso em: 29 de novembro de 2015.

[18] Esp8266 core for arduino. https://github.com/esp8266/Arduino, Acesso em: 29 de novembro de 2015.

[19] Downloads para placa intel galileo. https://software.intel.com/pt-br/iot/hardware/galileo/downloads/, Acesso em: 15 de Abril de 2016.

[20] Especificacoes LCD. http://img.filipeflop.com/files/download/Datasheet_Display_16x2.pdf, Acesso em: 18 de Maio de 2016.

[21] Sirene modulada para alarmes (ART525). http://www.newtoncbraga.com.br/index.php/artigos/54-dicas/3816-art525.pdf, Acesso em: 18 de Maio de 2016.

[22] Especificacoes tecnicas rele. http://img.filipeflop.com/files/download/Datasheet_Rele_5V.pdf, Acesso em: 20 de Maio de 2016.

[23] Biblioteca MFRC522. https://github.com/miguelbalboa/rfid, Acesso em: 20 de Maio de 2016.

[24] Vídeo testando distância de transmissão ESP8266-01 . https://www.youtube.com/watch?v=7BYdZ_24yg0, Acesso em: 20 de Maio de 2016.

[25] Especificações RF433MHz. http://www.filipeflop.com/pd-80dc1-modulo-rf-transmissor-receptor-433mhz-am.html, Acesso em: 20 de Maio de 2016.

[26] Site flaticon. http://www.flaticon.com/, Acesso em: 20 de Maio de 2016.

[27] Especificações transmissor RF433MHz. http://www.webtronico.com/documentos/transmissor_rf_link.pdf, Acesso em: 20 de Maio de 2016.

Apêndice A

Código utilizado no Arduino UNO para controle de acesso

```
//Programa : RFID - Controle de Acesso leitor RFID
# include <SPI.h>
# include <MFRC522.h>
# include <LiquidCrystal.h>
# define SS_ PIN 10
# define RST_ PIN 9
MFRC522 mfrc522(SS_ PIN, RST_ PIN);
LiquidCrystal lcd(6, 7, 5, 4, 3, 2);
int acesso = 8;
int SensorMag = 1;
char st[20];
void setup()
{
Serial.begin(9600); // Inicia a serial
SPI.begin(); // Inicia SPI bus
mfrc522.PCD_ Init(); // Inicia MFRC522
pinMode(acesso, OUTPUT);
pinMode(acesso, INPUT);
digitalWrite(acesso, HIGH);
Serial.println("Aproxime o seu cartao do leitor...");
```

```
Serial.println();
//Define o número de colunas e linhas do LCD:
lcd.begin(16, 2);
mensageminicial();
}
void loop()
{
// Look for new cards
if ( ! mfrc522.PICC_ IsNewCardPresent())
{
return;
}
// Select one of the cards
if ( ! mfrc522.PICC_ ReadCardSerial())
{
return;
}
//Mostra UID na serial
Serial.print("UID da tag :");
String conteudo = ;
byte letra;
for (byte i = 0; i < mfrc522.uid.size; i++)
{
Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? "0": );
Serial.print(mfrc522.uid.uidByte[i], HEX);
conteudo.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? "0": ));
conteudo.concat(String(mfrc522.uid.uidByte[i], HEX));
}
Serial.println();
Serial.print("Mensagem : ");
conteudo.toUpperCase();
if (conteudo.substring(1) == "A4 DB 6E B8") //UID 1 - Chaveiro
{
```

```
    Serial.println("Ola TCC !");
    Serial.println();
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Ola TCC !");
    lcd.setCursor(0, 1);
    lcd.print("Acesso liberado!");
    delay(3000);
    digitalWrite(acesso, LOW);
    while SensorMag = 1;
    digitalWrite(acesso, HIGH);
    mensageminicial();
    }
    if (conteudo.substring(1) == "01 48 FB E5") //UID 2 - Cartao { Serial.println("Desconhecido
!!!");
    Serial.println();
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Desconhecido!");
    lcd.setCursor(0, 1);
    lcd.print("Acesso Negado !");
    delay(3000);
    digitalWrite(acesso, HIGH);
    mensageminicial();
    }
    }
    void mensageminicial()
    {
    lcd.clear();
    lcd.print("Aproxime o seu");
    lcd.setCursor(0, 1);
    lcd.print("cartao do leitor");
    }
```

Apêndice B

Código para a transmissão de status da porta e janela

B.0.1 RF433MHz

```
//Programa : Recepção de porta aberta ou fechada avisando no email quando aberta
# include <VirtualWire.h>
# define time 1000 //Defini um delay defoud
int valor_ recebido_ RF;
char recebido_ RF_ char[4];
void setup()
{
Serial.begin(9600);
delay(time);
pinMode (5, OUPUT)
vw_ set_ rx_ pin(7); //Pino ligado ao pino DATA do receptor RF
vw_ setup(2000); //Velocidade de comunicacao (bits por
segundo)
vw_ rx_ start();//Inicia a recepcao
Serial.println("Tudo calmo por enquanto");
}
```

```
void loop()
{
uint8_ t buf[VW_ MAX_ MESSAGE_ LEN];
uint8_ t buflen = VW_ MAX_ MESSAGE_ LEN;
if (vw_ get_ message(buf, & buflen)==1)
{
int i;
for (i = 0; i < buflen; i++)
{
//Armazena os caracteres recebidos
recebido_ RF_ char[i] = char(buf[i]);
}
recebido_ RF_ char[buflen] = ”;
//Converte o valor recebido para integer
valor_ recebido_ RF = atoi(recebido_ RF_ char);
//Mostra no serial monitor o valor recebido
Serial.print("Recebido: ");
Serial.print(valor_ recebido_ RF);
Serial.print("Alguem abriu a porta");
envia();
//Altera o estado do led conforme o numero recebido
if (valor_ recebido_ RF == 1)
{ digitalWrite(5,HIGH);}
if (valor_ recebido_ RF == 0)
{ digitalWrite(5,LOW);

}
```

B.0.2 ESP8266-01

```
# include <ESP8266WiFi.h>
const char* nome = ; //nome da rede wifi que se dejesa conectar
const char* senha = ; //senha dessa rede
```

```
const char* endip = "192.168.0.17"; //endereço de ip do arduino+shield
int cte = 0;
void setup()
pinMode(2, INPUT);
Serial.begin(115200);
delay(100);
Serial.println();
Serial.println();
WiFi.begin(nome, senha); //conceta-se há rede wifi definida
while (WiFi.status() != WL_ CONNECTED) { //loop que garante que o módulo será conectado
delay(500);
}
void loop() {
cte = 0;
if (digitalRead(2) == LOW) {
janela_ fechada(); //verifica se o sensor está fechado
}
if (digitalRead(2) == HIGH) {
janela_ aberta(); //verifica se o sensor está aberto
}
}
void janela_ aberta() {
WiFiClient client;
const int portadeacesso = 80;
if (!client.connect(endip, portadeacesso)) { //loop que garente que haja conecção
janela_ aberta();
}
String url;
static uint16_ t i;
url = "/?janelaaberta;"; //URL que defini o ligamento do Led ligado ao arduino+shield
delay(25);
// rotina de envio da url
```

```
client.print(String("GET ") + url + "HTTP/1.1 r n"+
"endip: "+ endip + " r n"+
"Connection: close r n r n");
delay(1000);
if (cte = 0) {
cte = 1;
janela_ aberta();
}
while (digitalRead(2) == HIGH) { } //Evita envio constatnte do comando url
while (client.available()) {
String line = client.readStringUntil(’˚);
Serial.print(line);
}
Serial.println();
Serial.println("closing connection");
}
void janela_ fechada() {
WiFiClient client;
const int portadeacesso = 80;
if (!client.connect(endip, portadeacesso)) { //loop que garente que haja conecção
janela_ fechada();
}
String url;
static uint16_ t i;
url = "/?janelafechada;"; //URL que defini o desligamento do Led ligado ao arduino+shield
delay(25);
// rotina de envio da url
client.print(String("GET ") + url + "HTTP/1.1 r n"+
"endip: "+ endip + "° n"+
"Connection: close r n r n");
delay(1000);
if (cte = 0) {
cte = 1;
```

```
janelaz_ fechada();
}
while (digitalRead(2) == LOW) { } //Evita envio constatnte do comando url
while (client.available()) { //espera o termino do feedback e encerra a conecção
String line = client.readStringUntil(’ r’);
Serial.print(line);
}
Serial.println();
Serial.println("closing connection");
}
```

Apêndice C

Central de Controle

C.0.1 Código utilizado no Arduino UNO com seu *Ethernet Shield*

```
//Programa do site que acende ou apaga a luz dependendo do sinal do ESP8266-01
# include <SPI.h>
# include <Ethernet.h>
byte mac[] = 0xde, 0xab, 0xbd, 0xaf, 0xfe, 0xed; //Endereço do MAC
byte ip[] =  192,168,0,170 ; // IP fixo
EthernetServer server(80);
int panico = 8;
String readString;
void setup(){
pinMode(6, OUTPUT); //pino de indicaçãp da porta
pinMode(7, OUTPUT); //pino de indicação da janela
pinMode(8, OUTPUT); //pino de indicação do botao de panico
Ethernet.begin(mac, ip);
server.begin();
}
void loop(){
EthernetClient client = server.available();
if (client) {
while (client.connected()) {
if (client.available()) {
```

```
    char c = client.read();
    if (readString.length() < 100) {
    readString += c;
    }
    if(readString.indexOf("?portaaberta") >0)//Checa se o houve o comando URL de porta aberta
    {
    digitalWrite(6, HIGH); //Seta o pino 6 em alto
    }
    if(readString.indexOf("?portafechada") >0)//Checa se o houve o comando URL de porta fechada
    {
    digitalWrite(6, LOW); // Seta o pino 6 em baixo
    }
    if(readString.indexOf("?janelaaberta") >0)//Checa se o houve o comando URL de janela aberta
    {
    digitalWrite(7, HIGH); // Seta o pino 7 em alto
    }
    if(readString.indexOf("?janelafechada") >0)//Checa se o houve o comando URL de janela fechada
    {
    digitalWrite(7, LOW); // Seta o pino 7 em baixo
    }
    if(readString.indexOf("?ledon") >0)//Checa se o houve o comando URL de led de teste
    {
    digitalWrite(5, HIGH); // Seta o pino 5 em alto
    }
    if(readString.indexOf("?ledoff") >0)//Checa se o houve o comando URL de led de teste
    {
    digitalWrite(5, LOW); // Seta o pino 5 em baixo
    }
    if(readString.indexOf("?panicoon") >0)//Checa se o houve o comando URL de led de
```

```
teste
    {
    digitalWrite(8, HIGH); // Seta o pino 8 em baixo
    }
    if(readString.indexOf("?panicooff") >0)//Checa se o houve o comando URL de led de
teste
    {
    digitalWrite(8, LOW); // Seta o pino 8 em baixo
    }
    if (c == ' n') {
    client.println("HTTP/1.1 200 OK");
    client.println("Content-Type: text/html");
    client.println();
    client.println(«HTML>");
    client.println(«HEAD>");
    client.println(«TITLE>Verificacao de portas aberta</TITLE>");
    client.println(«/HEAD>");
    client.println(«BODY");
    if(digitalRead(6) == LOW){ client.println(«H1>PortaOFF</H1>");}
    if(digitalRead(6) == HIGH){ client.println(«H1>PortaON</H1>");}
    if(digitalRead(7) == LOW){ client.println(«H1>JanelaOFF</H1>");}
    if(digitalRead(7) == HIGH){ client.println(«H1>JanelaON</H1>");}
    if(digitalRead(8) == HIGH){ client.println(«H1>PANICO!!!!!</H1>");}
    if(digitalRead(8) == LOW){ client.println(«H1>NORMAL</H1>");}
    client.println(«hr />");
    client.println(«br />");
    client.println(«a href=  "/ ?ledon " »LED ON</a>");
    client.println(«a href=  "/?ledoff  " »LED OFF</a><br /> );
    client.println(«/BODY>");
    client.println(«/HTML>");
    delay(10);
    client.stop();
    readString=; //Limpa o vetor para o próximo comando
```

```
}
}
}
}
}
```

C.0.2 Código utilizado na Intel Galileo

```
// WebService do projeto
# include <SPI.h>
# include <Ethernet.h>
byte mac[] = {
0x98, 0x4F, 0xEE, 0x01, 0x13, 0x4B } ;
IPAddress ip(192,168,0,35);
EthernetServer server(8080);
String readString;
void setup() {
pinMode(6, OUTPUT); //pino de indicaçãp da porta
pinMode(7, OUTPUT); //pino de indicação da janela
pinMode(8, OUTPUT); //pino de indicação do botao de panic
digitalWrite(6, LOW); //
digitalWrite(7, LOW); //
digitalWrite(8, LOW); //
Serial.begin(9600);
// Inicializando a conexao com a internet
Ethernet.begin(mac, ip);
server.begin();
}
void loop() {
// Esperando novos clientes
EthernetClient client = server.available();
if (client) {
// especificamente no Intel Galileo o pedido de comando http termina com "boolean cur-
```

```
rentLineIsBlank = true;"
    boolean currentLineIsBlank = true;
    while (client.connected()) {
    if (client.available()) {
    char c = client.read();
    //Verificacao das URLs
    if (readString.length() < 100) {
    readString += c;
    }
    if(readString.indexOf("?portaaberta") >0)//Checa se o houve o comando URL de porta aberta
    {
    digitalWrite(6, HIGH); //Seta o pino 6 em alto
    }
    if(readString.indexOf("?portafechada") >0)//Checa se o houve o comando URL de porta fechada
    {
    digitalWrite(6, LOW); // Seta o pino 6 em baixo
    }
    if(readString.indexOf("?janelaaberta") >0)//Checa se o houve o comando URL de janela aberta
    {
    digitalWrite(7, HIGH); // Seta o pino 7 em alto
    }
    if(readString.indexOf("?janelafechada") >0)//Checa se o houve o comando URL de janela fechada
    {
    digitalWrite(7, LOW); // Seta o pino 7 em baixo
    }
    if(readString.indexOf("?ledon") >0)//Checa se o houve o comando URL de led de teste
    {
    digitalWrite(5, HIGH); // Seta o pino 5 em alto
    }
```

```
    if(readString.indexOf("?ledoff") >0)//Checa se o houve o comando URL de led de teste
    {
    digitalWrite(5, LOW); // Seta o pino 5 em baixo
    }
    if(readString.indexOf("?panicoon") >0)//Checa se o houve o comando URL de led de
teste
    {
    digitalWrite(8, HIGH); // Seta o pino 8 em baixo
    }
    if(readString.indexOf("?panicooff") >0)//Checa se o houve o comando URL de led de
teste
    {
    digitalWrite(8, LOW); // Seta o pino 8 em baixo
    }
    Serial.write(c);
    // Verificacao se o pedido http acabou
    if (c == ' n' &  & currentLineIsBlank) {
    //Criando o WebService
    client.println("HTTP/1.1 200 OK");
    client.println("Content-Type: text/html");
    client.println("Connection: close");
    client.println();
    client.println(«!DOCTYPE HTML>");
    client.println(«h1> Teste de Portas </h1>");
    client.println(«input type=button value=ABRIR_ PORTA
    onmousedown=location.href='/?portaaberta'>"); //Botão de teste para "abrir a porta"
    client.println(«br />");
    client.println(«input type=button value=FECHAR_ PORTA
    onmousedown=location.href='/?portafechada'>"); //Botão de teste para "fechar a porta"
    client.println(«br />");
    client.println(«html>");
    if(digitalRead(6) == LOW){
    client.println("PortaOFF");
```

```
client.println(«br />");
}
if(digitalRead(6) == HIGH){
client.println("PortaON");
client.println(«br />");
}
if(digitalRead(7) == LOW){
client.println("JanelaOFF");
client.println(«br />");
}
if(digitalRead(7) == HIGH){
client.println("JanelaON");
client.println(«br />");
}
if(digitalRead(8) == HIGH){
client.println("PANICO!!!!!");
client.println(«br />");
}
if(digitalRead(8) == LOW){
client.println("NORMAL");
client.println(«br />");
}
client.println(«/html>");
break;
}
if (c == ’ n’) {
// Terminando de escrever no WebSevice
currentLineIsBlank = true;
}
else if (c != ’ r’) {
currentLineIsBlank = false;
}
}
```

```
}
delay(1);
// fechando a comunicacao
client.stop();
Serial.println("client disonnected");
}
readString=; //Limpa o vetor para o próximo comando
}
```