CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA
CELSO SUCKOW DA FONSECA – CEFET/RJ

Projeto de Automação Residencial utilizando Comunicação entre Mobile Station(GSM), Arduino(GSM Shield), XBee e PC.

Manoel Felipe Costa Furtado

Prof. Orientador: Vinicius Coutinho de Oliveira

Rio de Janeiro
DEZEMBRO de 2015

CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA
CELSO SUCKOW DA FONSECA – CEFET/RJ

Projeto de Automação Residencial utilizando Comunicação entre Mobile Station(GSM), Arduino(GSM Shield), XBee e PC.

Manoel Felipe Costa Furtado

Projeto final apresentado em cumprimento às normas do Departamento de Educação Superior do CEFET/RJ, como parte dos requisitos para obtenção do título de Bacharel em Engenharia Elétrica

Prof. Orientador: Vinicius Coutinho de Oliveira

Rio de Janeiro
DEZEMBRO de 2015

M277 Furtado, Manoel Felipe Costa
Projeto de Automação Residencial utilizando Comunicação entre Mobile Station(GSM), Arduino(GSM Shield + XBee).
Furtado, Manoel Felipe Costa – 2015
x, 100f +anexos: il (algumas color), grafs, tabs.;enc

Projeto Final (Graduação) Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, 2015
Bibliografia:f. XXX-XXX (Ex.:130-131)

1.Engenharia Elétrica 2.Telecomunicações I.Título

CDD 658

DEDICATÓRIA

Dedico este Projeto Final aos meus parentes, que nos proporcionaram a oportunidade de concluir a graduação. A eles que sempre estão do nosso lado, nos apoiando e nos dando força para continuar e alcançar os objetivos que são o Vinicios Coutinho, Allan Alegretti, Sanitella Coppola Defelippe e a Vanessa Ayrão Franco.

AGRADECIMENTOS

Agradeço ao Prof. orientador Vinicius Coutinho de Oliveira pela orientação que me foi dada e pelo acompanhamento do projeto.

Agradeço a todas pessoas que nos ajudaram direta ou indiretamente na realização do trabalho, ao apoio de nossos amigos.

Agradeço a todos do corpo docente do CEFET pelos ensinamentos compartilhados durante o período da graduação.

RESUMO

O avanço de novas tecnologias de comunicação sem fio e na microeletrônica está facilitando a realização da automação residencial e comercial no dia a dia na sociedade globalizada em que vivemos, proporcionando até as classes menos favorecidas poder implementar esse serviço. Nesse trabalho, será apresentada uma aplicação do protótipo eletrônico de hardware livre chamado Arduino, que interligará o hardware proprietário assegurar a interoperabilidade de forma a garantir o funcionamento de um sistema de comunicação seguro (através do SMS), rápido (através da rede móvel 2G) e com gasto mínimo de energia (através do ZigBee). O Arduino proporciona inúmeras possibilidades para gerenciar, coletar informações interagindo com outras linguagens como Shell Script, MySQL, HTML, PHP, AJAX, JAVA.

Para a aplicação proposta, é empregado um aparelho celular (GSM) para enviar os comandos e receber as informações via SMS ao Arduino integrado com o PC. Para essa comunicação se estabelecer, é necessário o módulo GSM com um módulo Ethernet ou Wifi com o Arduino. Depois de estabelecida a referida comunicação, o módulo XBee, ligado no mesmo Arduino, pode se comunicar com outro XBee de modo a poder ser controlada por telefone móvel empregando o serviço de mensagem SMS fornecido pela tecnologia GSM. Essa comunicação entre XBee TX/RX é provida pelo Arduino, que recebe as instruções pelo SMS, acionando uma porta digital qualquer, utilizando a sua linguagem, no módulo XBee. Após o acionamento do módulo XBee receptor, modifica-se o estado lógico de uma das portas do XBee acionando uma carga - por exemplo, um relé. Um PC gerencia todos os eventos da comunicação e apresenta informações úteis.

ABSTRACT

The advance of wireless communication and microelectronics technology has growingly enabled the realization of residential and commercial automation in everyday life in the globalized society we live in, allowing even the lower-income classes to implement this service. In the present work, an application of electronic, free, non-proprietary hardware prototype called Arduino, which interconnects SIM900 and XBee proprietary hardware,is presented,. It assures interoperability in order to guarantee the operation of a secure communication system (via SMS), fast (via Mobile 2G network) and with minimum power consumption (through ZigBee). The Arduino provides many possibilities to manage and collect information, interacting with other languages such as Shell Script, MySQL, HTML, PHP, AJAX, JAVA and others.

For the proposed application, a GSM cell phone is utilized to send commands and receive information by SMS to the Arduino, which is integrated to the PC. To establish this communication, the GSM module with an Ethernet or Wifi module with Arduino is required. Once the communication is established , the XBee module, connected to Arduino, communicates with other XBee so that it can be controlled by mobile phone employing the SMS message service provided by GSM technology. This communication between XBee TX / RX is provided by Arduino, which receives the instructions via SMS, activating a digital gate circuit, using their language, at the XBee module. After the activation of the XBee reception module, the logic state of the gate changes and the XBee triggers a load – for instance, a relay. The PC manages all the communication events and displays useful information.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS E SIGLAS

2G:	Second Generation ;
3GPP:	Generation Partnership Project;
AC:	Alternating Current;
AES:	AbreviaturasAdvanced Encryption Standard;
AJAX:	Asynchronous Javascript and XML;
ANATEL:	Agência Nacional de Telecomunicações;
AODV:	Ad hoc On Demand Distance Vector;
API:	Application Programming Interface;
AREF:	Analogue REFerence;
AT:	Attention;
BER:	Bit Error Rate;
BI:	Beacon Interval;
BO:	macBeaconOrder;
BPSK:	Binary Phase Shift Keying;
CAP:	Contention Access Period;
CBR:	Constant Bit Rate;
CCA:	Clear Channel Assessment;
CFP:	Contention Free Period;
CLP	Controlador Lógico Programável
CRC:	Cyclic Redundancy Check;
CSMA-CA:	Carrier Sense Multiple Access with Collision Avoidance;
CSS:	Cascading Style Sheets;

DC: Direct Current;

DCS: Data-Centric Storage;

DSSS: Direct Sequence Spread Spectrum;

ED: Energy Detection;

EGSM: Extended GSM;

E/S Entrada e Saída

FCS: Frame Check Sequence;

FFD: Full-Function Device;

FH: Frequency Hopping;

FHSS: Frequency Hopping Spread Spectrum;

FL: Frame Length;

GCC: GNU Compiler Collection;

GNU: General Public License;

GPRS: General Packet Radio Services;

GPS: Global Positioning System;

GSM: Global System for Mobile Communications or Groupe Spécial Mobile;

GTS: Guaranteed Time Slot;

HMI: Human-Machine Interface;

HTML: HyperText Markup Language;

I²C: Inter-Integrated Circuit;

ICSP: In-circuit Serial Programming;

IDE Integrated Development Environment

IEEE: Institute of Electrical and Electronics Engineers;

IFS: Interframe Spacing;

IOREF: Input Output Reference;

IP: Internet Protocol;

ISM: Industrial, Scientific, and Medical;

ISO: International Organization for Standardization;

LAN: Local Area Network;

LED: Light Emitting Diode;

LGPL: Lesser General Public License;

LIFS: Long Interframe Spacing;

LLC: Logical Link Control;

LQI: Link Quality Indication;

LR-WPAN: Low-Rate Wireless Personal Area Network;

M2M: Machine-to-Machine;

MAC: Medium Access Control;

MCU: Microcontrolador;

MFR: MAC footer;

MHR: MAC header;

MIC: Message Integrity Code;

MISO: Master In Slave Out;

MOSI: Master Out Slave In;

MPDU: MAC Protocol Data Unit;

MPLS: Multi-Protocol Label Switching;

MSDU: MAC Service Data Unit;

MySQL: Michael Widenius's Structured Query Language;

NWK: Network (NWK) Layer;

O-QPSK: Offset Quadrature Phase-Shift Keying;

OFDM: Orthogonal Frequency Division Multiplexing;

OSI: Open Systems Interconnection;

PAN: Personal Area Network;

PC: Personal Computer;

PDU: Protocol Data Unit;

PHP: Hypertext Preprocessor;

PHR: PHY Header;

PHY: Physical Layer;

PLC: Programmable Logic Controllers;

POS: Personal Operating Space;

PPDU: PHY Protocol Data Unit;

PS: Preamble Sequence;

PSDU: PHY Service Data Unit;

PWM: Pulse-width Modulation;

QoS: Quality of service;

RF: Radio frequency;

RFD: Reduced-Function Device;

SAP: Service Access Point;

SCADA: Supervisory Control and Data Aquisition;

SCK: Serial Clock;

SCL: Serial Line;

SD: Superframe Duration;

SDA: Serial Data;

SFD: Start of Frame Delimiter;

SHR: Synchronization Header;

SIFS: Short Interframe Spacing;

SMS: Short Message Service;

SMT: surfaced mount technology;

SNR: Signal-to-noise ratio;

SO: macSuperframeOrder;

SPI: Serial Peripheral Interface;

SSCS: Service Specific Convergence Sublayer;

TCP/IP: Transmission Control Protocol / Internet Protocol;

TG4: Task Group or Teilifís na Gaeilge;

TTL: Time To Live;

TWI: Two-Wire Interface;

UART: Universal Asynchronous Receiver/Transmitter;

USB: Universal Serial Bus;

UTP: Unshielded Twisted Pair;

UWB: Ultra Wide Band;

Wi-Fi: Local Area Wireless Technology;

WLAN: Wireless Local Area Network;

WPAN: Wireless Personal Area Network;

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

A automação residencial está a cada dia facilitando a vida das pessoas. Com a chegada de novas tecnologias de comunicação sem fio, já é possível levar esta facilidade e conforto para a população de várias classes sociais, pois evita altos gastos para adequar a casa para implementar essa automação. Controlando o XBee pelo GSM-SMS, garantimos eficiência energética, segurança e facilidade. O projeto permite além da programação base usada que é a linguagem C/C++, outras como o Shell Script, MySQL, HTML, PHP, AJAX e comandos AT etc, possibilitando interação de outras tecnologias como o Bluetooth, Wi-Fi com diversos sensores e equipamentos. Essa interoperabilidade na rede é proporcionada pelo Arduino.

1.2 JUSTIFICATIVA

O uso do Arduino proporcionará a interoperabilidade mais fácil com outras tecnologias, viabilizando o objetivo do projeto. O Arduino é uma plataforma de prototipagem eletrônica de hardware livre, proporcionando a integração com diversas tecnologias. E não somente o hardware é livre mas a linguagem de programação também é livre, proporcionando uma interação com a comunidade que desenvolve e outras linguagens.

XBee é a marca da Digi International® Inc. (Digi®) para uma família de módulos de rádio, que utiliza o protocolo ZigBee 802.15.4. O padrão tem como finalidade a eficiência energética. Permite realizar a automação residencial ou comercial com custo energético menor, menos impacto na saúde, menor interferência na comunicação sem fio com outros dispositivos, mesmo com baixo throughput a velocidade de resposta na comunicação é mais rápida entre todos os outros protocolos usados no mercado.

Empregar um módulo GSM conectado ao Arduino garante segurança, cobertura, facilidade para realizar a comunicação pelo usuário. Utiliza a rede móvel oferecida pelas operadoras de telefonia celular, garante ao usuário realizar a comunicação em qualquer país que ofereça o serviço.

1.3 OBJETIVOS

Este projeto tem como objetivo estudar os protocolos de comunicações entre um aparelho celular (GSM) e o Arduino integrado com o PC por mensagem de texto SMS. Uma vez estabelecida essa comunicação, será utilizado o módulo XBee, ligado ao Arduino, para se comunicar com outro XBee de forma que possa ser controlado pelo aparelho celular usando a tecnologia GSM por SMS. O PC gerenciará todos os eventos da comunicação e apresentará informações úteis. Nessa gerência, haverá interação com outras linguagens como Shell Script, MySQL, HTML, PHP e AJAX.

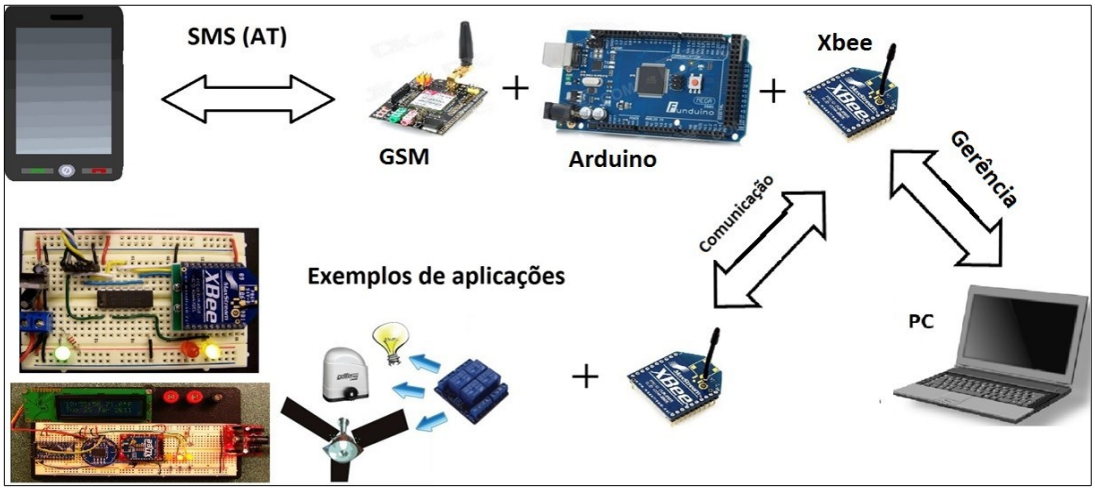


Figura 1.1: Arquitetura de rede proposta

1.4 METODOLOGIA USADA NO PROJETO

Empregou-se a seguinte metodologia para elaboração deste projeto.

A) Análise de Viabilidade do Projeto.

A viabilidade do projeto foi encontrada pesquisando sobre outros projetos de conclusão de curso, artigos científicos, conversas informais com colegas e faculdade, pesquisando na internet etc. Durante a pesquisa sobre a viabilidade, foi realizado um estudo se era possível realizar ele. Pesquisando os dispositivos a ser utilizado. Já era sabido que seria encontrado obstáculos na execução. Tanto que não foi concluído toda a proposta apresentada nos subtítulos 1.1, 1.2 e 1.3. Mas o desafio e motivação de tentar executar foi fundamental.

B) Especificação e compra de materiais para montagem do protótipo.

Será abordado no capítulo 3.

C) Montagem física do projeto. Diagrama das conexões das pinagens de todo o projeto.

D) Programação:

I) Programação da comunicação entre o Arduino e Módulo (Shield-GSM) acoplado no Arduino.

II) Programação da comunicação entre o Arduino e XBee acoplado no Arduino entre o XBee acoplado no Arduino e o outro XBee receptor.

III) Programação da comunicação entre o PC e o Arduino.

IV) A programação da comunicação das mensagens SMS através de comandos AT integrando todo o projeto.

V) Desenvolvimento de interface de usuário e o registro dos eventos através de banco de dados e html.

E) Testes, correções e ajustes do protótipo

1.5 ORGANIZAÇÃO DO TRABALHO

O presente trabalho se encontra organizado da seguinte forma. O capítulo 1 introduz o tema do projeto, sua motivação, os objetivos e sua organização. No capítulo 2 é feita uma abordagem teórica dos padrões IEEE 802.15.4 e Zig-Bee. O capítulo 3 apresenta a descrição e especificações técnicas dos materiais utilizados no projeto e suas caraterísticas. No capítulo 4 são apresentados a montagem física do protótipo e os desenhos do circuito. O capítulo 5 apresenta as bibliotecas para o Arduíno e alguns exemplos de sua utilização. No capítulo 6 é apresentada a programação utilizada no trabalho e testes, correções e ajustes do protótipo. No capítulo 7 é apresentada a visão do tema do trabalho do conceito da automação industrial e residencial.

2 IEEE 802.15.4 E ZIG-BEE

2.1 ORIGEM DO IEEE 802.15.4 E ZIGBEE

Ergen [1] sugere, que o surgimento da rede celular foi natural decorrente da rede de telefonia com fio, como alternativa, durante a segunda metade do século 20, devido à necessidade de dar mobilidade da comunicação existente, menores custos, aumento da cobertura e possibilidade de formar uma ligação independentemente da localização.

O grupo de trabalho IEEE criou um padrão a rede sem fio, chamado IEEE 802.11 para rede WLANs (Wireless Local Area Network). Esse padrão estava preocupado com algumas caraterísticas correspondentes à tecnologia Ethernet.

Os padrões IEEE 802.15 diferem do IEEE 802.11, pois se tratam de especificações a redes WPANs (Wireless Personal Area Network). Tem como objetivo ser de baixo custo, longo e curto alcance, baixo consumo, resposta muito rápida e tamanho reduzido do dispositivo.

Hoje o padrão IEEE 802.15 está dividido em 7 grupos [2]:

IEEE 802.15.1 - 2005	IEEE 802.15.5 - 2009
IEEE 802.15.2 - 2003	IEEE 802.15.6 - 2012
IEEE 802.15.3 - 2003	IEEE 802.15.7 -2011
IEEE 802.15.4 - 2011	

Tabela 2.1: IEEE 802.15 dividido em 7 grupos - [2]

A tecnologia Bluetooth foi revisada e publicada pelo IEEE 802.15.1, precursora do padrão IEEE 802.15. Embora não seja o padrão de base do presente trabalho, é um padrão próximo.

O padrão IEEE 802.15.4 atende o tema do trabalho. Em 2006 foi revisado e publicado o padrão IEEE 802.15.4 [3]. O grupo TG4 (Task Group 4) tem como objetivo investigar soluções para transmitir à baixa taxa de dados usando o mínimo de energia possível para garantir baixo consumo das baterias existentes e baixa complexidade. Algumas aplicações

potenciais são sensores, brinquedos interativos, crachás inteligentes, controles remotos e automação residencial.

O fato de o padrão IEEE 802.15 estar organizado em 7 grupos, mostra a diversividade das aplicações para suprir a necessidade da sociedade; Ergen [1] define três classes de WPANs:

1) IEEE 802.15.3 proporciona alta taxa de dados, é adequado para aplicações multimídias que requerem alto QoS.

2) IEEE 802.15.1 (Bluetooth): lida com uma variedade de tarefas como a conexão de telefone celular para PDA que atualmente foi substituído pelos smartphones, conexões com outros dispositivos como rádio automotivo (sistema de viva voz), consoles de videogame da nova geração, fones de ouvido sem fio, mouse, teclado, GPS, impressoras. A tecnologia Bluetooth tem uma singular característica que proporciona a mais alta velocidade em relação aos grupos IEEE 802.15, opera em curta distância; e tem um QoS equilibrado.

3) IEEE 802.15.4 (LR-WPAN - low-rate wireless personal area networks): é o foco deste trabalho. Se destina a servir aplicações industriais (usando sensores), residenciais e médicas, com baixo consumo de energia e baixo custo, garantindo a taxa de dados necessária e QoS. A baixa taxa de dados permite que o LR-WPAN consuma pouca energia.

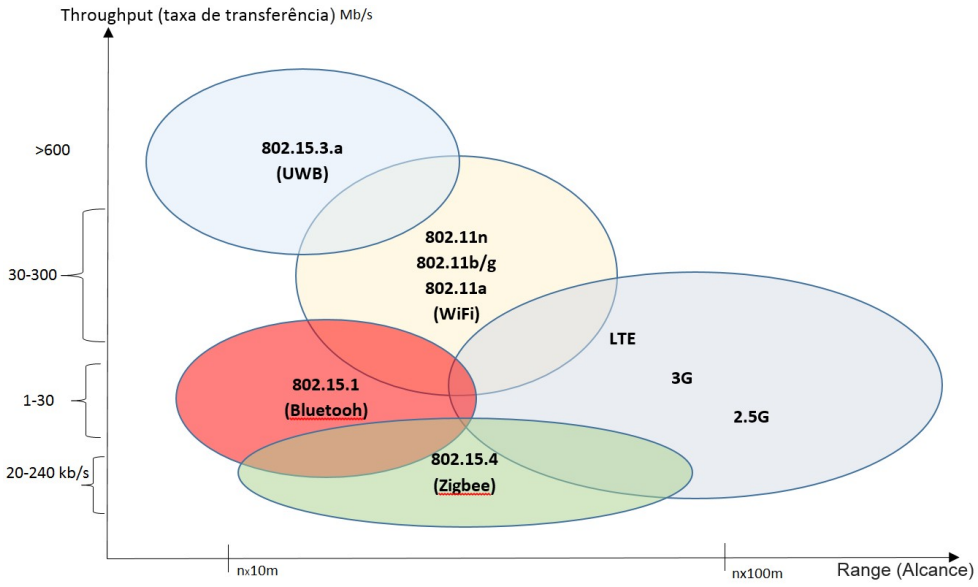


Figura 2.1: Comparação de taxa de transferência e cobertura. Adaptado [31] [10]

Características	802.11b (Wi-Fi)	802.15.4 (Bluetooh)	802.15.4 (ZigBee)
Autonomia da Bateria (Dias)	0,5-5 (Horas/Dias)	1-7 (Dias)	100-1000+ (Anos)
Complexidade	Muito Complexo	Complexo	Simples
Numero de Nós em uma rede	32	7	65000
Consumo na transmissão	30 mA TX, 0,2 µA Standby	300 mA TX, 20 mA Standby	45 mA (Classe 2) 150 mA (Classe1)
Frequência de Operação Básica	2,4 GHz	2,4 GHz	2,4 GHz
Taxa de Transferência (Kbps)	11000	1000	20-300
Rande (Metros)	100 m	10 m	70 m – 300 m
Latencia	~ 3 seg	~ 10 seg	~ 30 ms
Segurança	Authentication Service Set ID (SSID)	64 bits, 128 bits	128 bit AES e Appication Layer user defined
Atributos	Confiabilidade	Conveniência de custo	Velocidade e Flexibilidade
Aplicação	Acesso a Internet (Web, E-mail etc), Streaming (voz, video), Rede Local Pcs sem fio, etc	-Comunicação entre celulares e fones de ouvido sem fio ou sistemas viva voz para carros. -Comunicação sem fio entre PCs e dispositivos de E/S, como mouse, teclados e impressoras. -Substituição de dispositivos seriais tradicionais com fio em equipamentos de teste, receptores GPS, equipamentos médicos, leitores de código de barras.	-Automação de Controles Domésticos -Aplicações na Área de Saúde -Manutenção da Rede Elétrica -Aplicações industriais -Monitoramento e Controles de diversos sensores

Tabela 2.2: Comparação entre WiFi, Bluetooh e ZigBee [5]

A tecnologia Bluetooth, foi inventada pela fabricante Ericsson, desenvolvido por Jaap Haartsen e Sven Mattisson em 1994, e foi, ratificada pelo IEEE posteriormente através do padrão 802.15.1. Um processo similar aconteceu com a tecnologia ZigBee e o padrão correspondente, IEEE 802.15.4.

A tecnologia ZigBee tem como característica, baixa taxa de dados, baixo consumo de energia, baixo custo. Protocolo de rede sem fio voltado para aplicações de automação e controle remoto. O IEEE começou trabalhando em um padrão de baixa taxa de dados um pouco mais tarde, padronizando o IEEE 802.15.4. Como no Bluetooth, o surgimento da tecnologia veio antes. Em seguida, o grupo ZigBee Alliance e o IEEE decidiu unir uma força de trabalho e formou o nome ZigBee que é o nome comercial para essa tecnologia.

É esperado do ZigBee, fornecer conectividade de baixo custo e baixo consumo de energia para equipamentos onde a vida útil da bateria é extremamente importante, e que não necessite de altas taxas de dados, como aqueles utilizados pelo Bluetooth. O ZigBee pode ser implementada em redes de malha bem maiores que o Bluetooth.

As WPANs têm uma natureza pervasiva, possibilitando a integração de computadores a diversos objetos e equipamentos, e ao ambiente, de forma geral. Por utilizarem comunicação sem fio, sua integração com dispositivos móveis é também natural. A importância das WPANs vêm aumentando nos últimos anos. Elas têm sido usadas em diversos campos, como automação residencial, industrial e agrícola, embarcadas em veículos, para o monitoramento das condições de saúde à distância, entre outros. Algumas aplicações não eram possíveis antes do seu advento, outras se tornaram mais simples com seu uso.

2.2 DESCREVENDO O IEEE 802.15.4 E COMPARANDO O ZIGBEE

ZigBee Alliance visa proporcionar às camadas superiores da pilha de protocolos (a partir da camada de rede para aplicação) a interoperabilidade da rede de dados, serviços de segurança e uma gama de soluções de controle para redes sem fio residenciais e prediais. Esse controle pode ser chamado de automação residencial ou comercial, que é tema do trabalho. A interoperabilidade é a principal característica; ela garante que fabricantes diferentes possam se comunicar.

O IEEE 802.15.4 trabalha na especificação das camadas PHY e MAC, oferecendo blocos de construção para diferentes tipos de redes conhecidas como Star, Mesh e Cluster Tree. O roteamento da rede é projetado para garantir a conservação de energia e baixa latência pela garantia do tempo dos slots (time slots). Uma característica única da camada de rede no ZigBee é a redundância da comunicação eliminando ponto de falha único (Single Point of Failure) na rede Mesh. As principais características do PHY incluem energia e detecção de qualidade da ligação, avaliação de canal livre para melhorar a convivência com outras redes sem fio.

Algumas características do IEEE 802.15.4 TG4 [3]:

a) As taxas de dados de 250 kb/s, 40 kb/s e 20 kb/s.

b) Dois modos de endereçamento; 16-bit curto e de 64 bits IEEE.

c) Suporte para dispositivos com latência crítica, como joysticks.

d) Operação em topologia estrela, ponto a ponto e árvore (Star, Mesh e Cluster Tree).

e) Método de acesso CSMA-CA ou ALOHA.

f) Estabelecimento automático da rede pelo coordenador.

g) O protocolo usa o handshake para melhorar a confiabilidade na transferência.

h) O gerenciamento de energia para garantir baixo consumo de energia como a detecção de energia (ED - Energy detection (ED)) e o indicador que dá a qualidade do link (LQI(Link Quality Indication))

i) 16 canais na banda ISM de 2,4 GHz, 10 canais em 915MHz e 1 canal na banda de 868 MHz

j) Opcional. Alocação garantida do tempo de slots (GTSs (Guaranteed Time Slots))

Visto a diferença conceitual do IEEE 802.15 com IEEE 802.15.4/ZigBee.

Segue-se a estrutura resumida do padrão IEEE 802.15.4 [3], em 4 grupos.

2.2.1 COMPONENTES DA WPAN

Somente dois tipos de dispositivos podem participar de uma rede IEEE 802.15.4: dispositivos de – Função Completa (FFD – Full-Function) ou de Função Reduzida (RFD – Reduced-Function). O FFD é capaz de servir como um coordenador de pessoal (PAN – Personal Area Network) ou como um coordenador diferentemente do RFD, ou seja, o FFD pode assumir as funções do RFD. A rede deve incluir pelo menos um FFD, atuando com coordenador PAN. O RFD é destinado a aplicações que são extremamente simples, como acender e apagar uma lâmpada, um sensor infravermelho passivo, ou seja, ele não tem a necessidade de enviar grandes quantidades de dados e só se associa com um único FFD de

cada vez, logo o RFD pode ser implementado usando o mínimo de recursos e capacidade de memória. Nesse padrão basta ter dois ou mais dispositivos de comunicação, sendo 1 deles necessariamente do tipo FFD, na mesma camada física para formar a WPAN. [3] [4]

2.2.1.1 TOPOLOGIAS

Dependendo dos requisitos de uma aplicação, uma IEEE 802.15.4 LR-WPAN pode operar em três topologias: estrela, ponto a ponto e árvore, como mostrado na figura 2.2. Na topologia em estrela a comunicação é estabelecida entre os dispositivos e um único controlador central, chamado coordenador PAN.

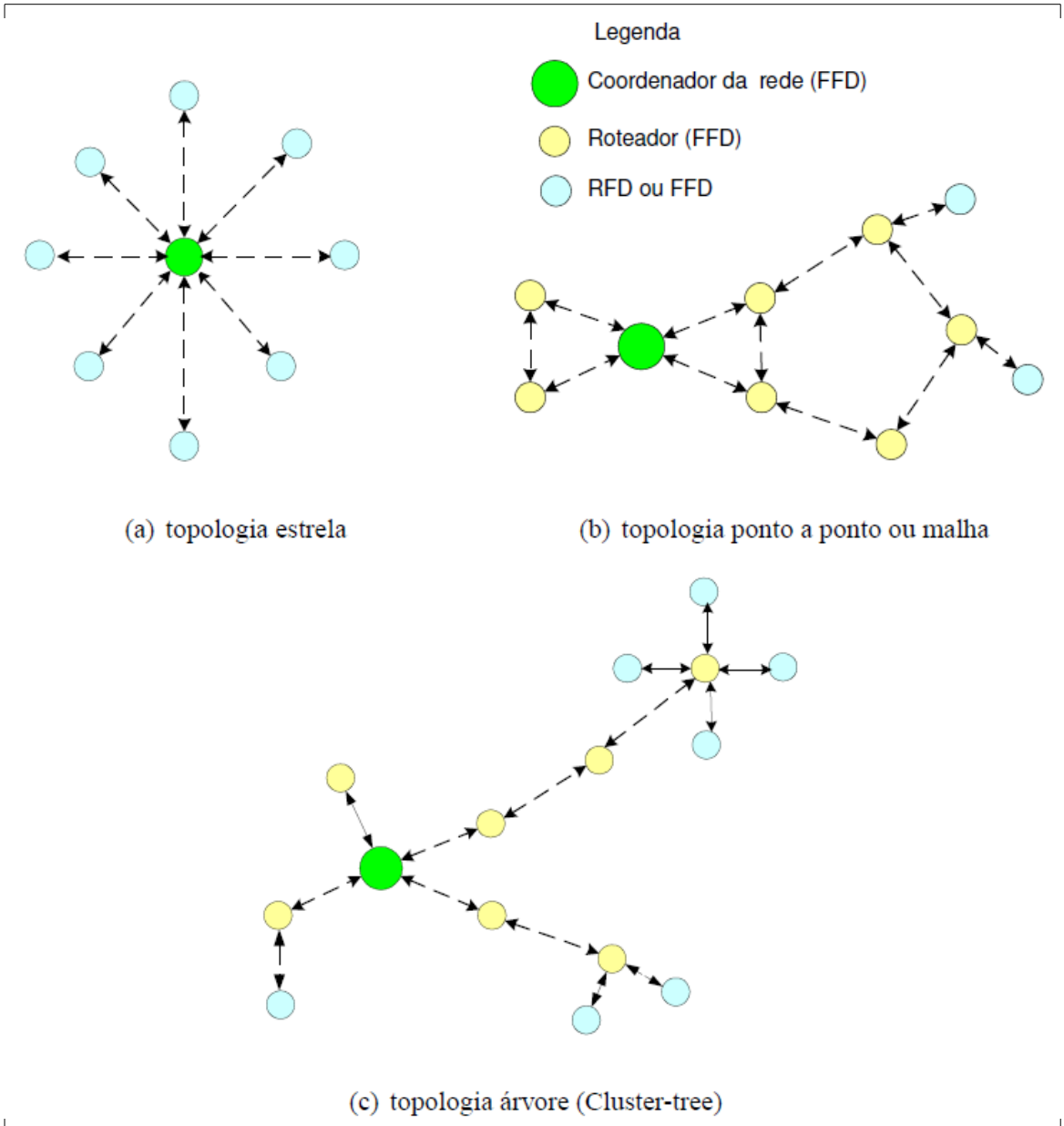


Figura 2.2: Topologias IEEE 802.15.4 LR-WPAN – [4]

2.2.1.1.1 TOPOLOGIA ESTRELA OU STAR TOPOLOGY

Na topologia estrela, a comunicação é estabelecida entre dispositivos e um único controlador central, chamado coordenador PAN. O coordenador PAN pode ser alimentado por energia contínua, enquanto os outros dispositivos normalmente seriam alimentados por bateria. As aplicações que se encaixam neste tipo de topologia, são: automação residencial, periféricos de computador pessoal, jogos e aplicações médicas. [4]

Após um FFD ser ativado pela primeira vez, ele pode estabelecer sua própria rede e tornar-se o Coordenador PAN. Cada rede inicializada escolhe um identificador PAN, que não é concorrentemente usado por alguma outra rede dentro da esfera de influência do rádio. Isto permite que cada rede estrela opere independentemente. Uma vez que é escolhido o identificador PAN, o coordenador permite que outros dispositivos se liguem à sua rede. Todos os dispositivos operando na rede, em qualquer topologia terão cada, um único endereço estendido de 64 bits. Este endereço poderá ser utilizado para comunicação direta dentro da PAN, ou pode ser trocado por um endereço curto alocado pelo coordenador PAN quando o dispositivo se associa. [4]

2.2.1.1.2 TOPOLOGIA PONTO A PONTO (PEER TO PEER) OU MESH (MALHA)

A topologia ponto a ponto (peer to peer) ou mesh (malha) também tem somente 1 coordenador PAN, contudo, difere da topologia em estrela pelo fato de que qualquer dispositivo FFD pode se comunicar com outro desde que ele esteja no seu raio de alcance de transmissão. Esta topologia permite a implementação de redes mais complexas, tais como formação em redes de malha ou em árvore (Cluster-tree).[4]

Aplicações como monitoramento e controle industrial, monitoramento na agricultura, e segurança se enquadram nesta topologia. Uma rede ponto a ponto pode também permitir múltiplos saltos para rotear mensagens de qualquer dispositivo para algum outro da rede. Tais funções são executadas pela camada de rede. Nessa topologia todos os dispositivos podem se comunicar entre si desde que estejam dentro do alcance. Essa topologia pode ser considerada uma rede ad hoc, com capacidade de se auto-organizar (self-organizing) e de se auto-estruturar (self-healing). Essa configuração permite também múltiplos caminhos ligando um dispositivo aos outros dispositivos da rede, de forma a permitir uma maior robustez. [5]

2.2.1.1.3 TOPOLOGIA EM ÁRVORE (CLUSTER-TREE)

A rede Cluster-tree é um caso especial de uma rede ponto a ponto, onde a maioria dos dispositivos são FFDs e um dispositivo RFD pode conectar-se no final de um ramo. Qualquer FFD pode agir como um coordenador e prover serviços de sincronização para outros dispositivos e coordenadores, porém somente um desses coordenadores será o coordenador PAN. [4]

O coordenador PAN fará o broadcast do beacon frame, anunciando a existência da rede. Os dispositivos recém-adicionados irão se estabelecer, podendo enviar quadros beacon buscando novos candidatos a se juntarem a rede. A principal vantagem dessa estrutura em árvore é aumentar a área de cobertura, ao custo de aumentar o atraso da mensagem.

Coordenador ZigBee	Roteador ZigBee	Dispositivo Final ZigBee	Função na Camada de Rede
X			Estabelecer uma nova rede ZigBee
X	X		Conceder endereço lógico de rede
X	X		Permitir que dispositivos entrem ou saiam da rede
X	X		Manter lista de vizinhos e rotas
X	X		Rotear pacotes da camada de rede
X	X	X	Transferir pacotes da camada de rede

Tabela 2.3: Tipos de dispositivos e suas Funções [6] [4]

Tipos de dispositvos	Funcionalidade disponíveis no protocolo	Fonte de alimentação típica	Configuração típica do receptor
FFD – Full Function Device	A maioria ou todas	Principal	Ligado quando em espera
RFD – Reduced Function Device	Limitada	Bateria	Desligado quando em espera

Tabela 2.4: Tipos de nós [7]

Na legenda da figura 2.2 a esfera azul corresponde ao nome de End Device, que pode ser RFD ou FFD. Como mostra a tabela 2.3 e 2.4 resume as características das funções dos tipos de nós.

2.2.2 LR-WPAN - DEVICE ARCHITECTURE

A arquitetura LR-WPAN (LR-WPAN (low-rate wireless personal area networks)) é definida em camadas baseadas no modelo OSI (Open Systems Interconnection). Cada camada é responsável por uma parte do padrão e oferece serviços para as camadas superiores. As interfaces entre as camadas servem para definir os enlaces lógicos que são descritos na norma IEEE 802.15.4. Um dispositivo LRWPAN compreende uma camada física (PHY) que contém os transceptores de radiofrequência com seu mecanismo de controle de baixo nível e uma subcamada MAC que provê acesso para a camada física. [4]

As camadas superiores mostradas nas figuras 2.3 e 2.4 consistem da camada de aplicação, do suporte à aplicação e de uma camada de rede que provê configuração da rede, manipulação e roteamento de mensagens definidas pelo grupo ZigBee Alliance. A camada de rede é responsável pela descoberta de rota e a entrega dos pacotes de dados. [4]

Em redes de sensores ad hoc, onde um grande número de nós é utilizado randomicamente, a descoberta de múltiplas rotas em uma topologia de malha é uma tarefa difícil. É igualmente desafiador manter e reparar rotas quando os nós são relocados ou desligam, por falta de bateria por exemplo. Inúmeros algoritmos de roteamento têm sido desenvolvidos para suportar redes ad hoc e especificamente neste trabalho foi utilizado o algoritmo AODV descrito mais adiante. [4]

A subcamada IEEE 802.2 LLC (Logical Link Control) pode acessar a subcamada MAC através da subcamada SSCS (Service Specific Convergence Sublayer). Ela provê acesso às funções primitivas definidas para a subcamada MAC que são essencialmente funções que são usadas para interagir com a subcamada MAC. Elas são usadas para executar funções tais como fazer uma requisição, receber uma notificação, e examinar ou modificar um atributo MAC. Dois atributos de particular importância são o macBeaconOrder (BO) e macSuperframeOrder (SO). Estes atributos definem o intervalo no qual são enviados quadros de sinalização (beacons) pelo coordenador e o comprimento do período ativo do superframe, respectivamente. [4]

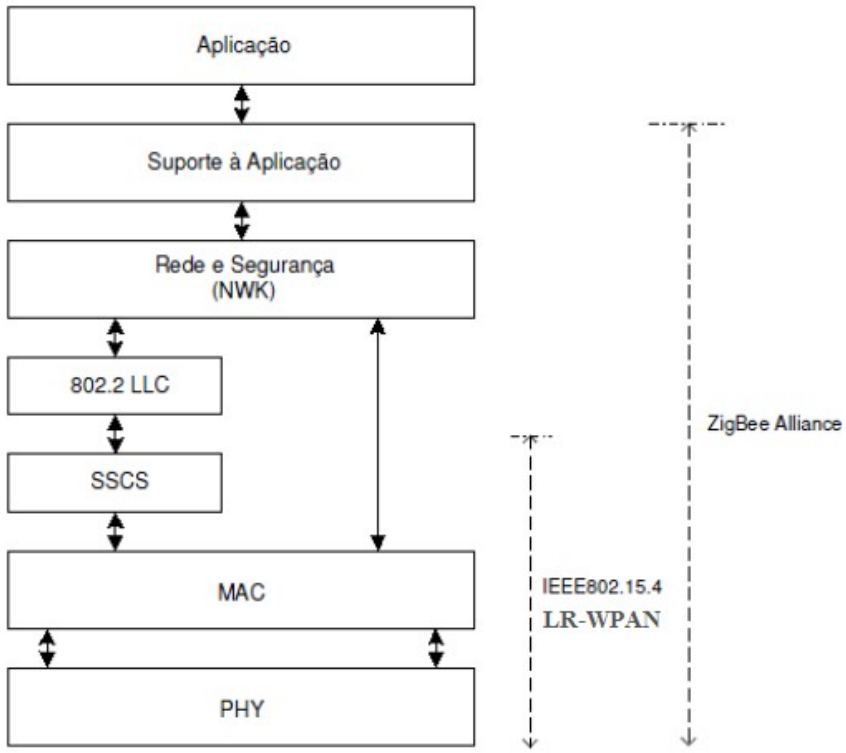


Figura 2.3: Camadas de Rede [4]

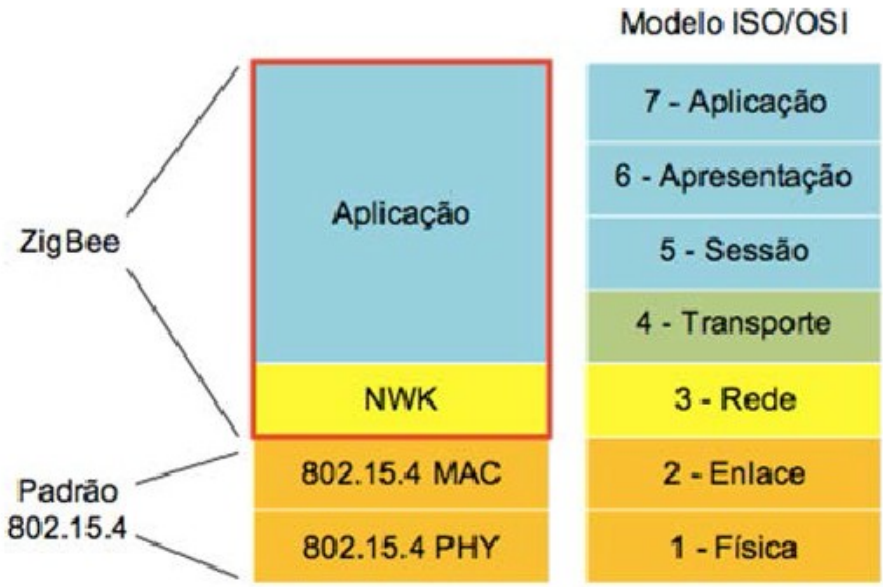


Figura 2.4: Comparação entre o padrão ZigBee e o modelo ISO/OSI. [5]

2.2.3 IEEE 802.15.4 PHY – PHYSICAL LAYER

A especificação da camada física descreve como os dispositivos IEEE 802.15.4 devem se comunicar através de um canal sem fio. Ela define as bandas ISM, que não requerem licenciamento, de 2.4 GHz e 868/915 MHz. A banda de frequência ISM 2.4 GHz é utilizada em todo o mundo, enquanto que as bandas ISM 868 MHz e ISM 915 MHz são utilizadas na Europa e América do Norte, respectivamente. [4] Como mostrado na tabela 2.5a.

PHY	Banda	Coverage	Taxa de bits	Numeros de Canais
2.4 GHz	ISM	Worldwide	250 kbps	16
868 MHz		Europe	20 kbps	1
915 MHz	ISM	Americas	40 kbps	10

Tabela 2.5a: As faixas de frequências disponíveis pelo IEEE 802.15.4 com taxa de bits apropriada por região e o número de canais. [8] [9]

Na figura 2.5 e na tabela 2.5a mostra 27 canais com três diferentes taxas de dados são alocadas pelo IEEE 802.15.4: 16 canais com uma taxa de dados de 250 kb/s em 2.4 GHz, 10 canais com uma taxa de dados de 40 kb/s na banda de 915 MHz e 1 canal com uma taxa de dados de 20 kb/s na banda de 868 MHz. Na figura 2.5b, a modulação BPSK (Binary Phase Shift Keying) é utilizada na banda de 868/915 MHz e a modulação O-QPSK (Offset Quadrature Phase-Shift Keying) na banda de 2.4 GHz. Ambas as modulações oferecem uma taxa de erro (BER) muito baixa com relação a um baixo nível de sinal ruído (SNR). Diferente do Bluetooth, o IEEE 802.15.4 não usa salto de frequências, mas é baseado em espalhamento de espectro de sequência direta (DSSS). Isto é muito útil em nossas medidas de interferência interna reportada no capítulo de simulações. [4]

PHY (MHz)	Banda de frequência (MHz)	Parâmetros de Espalhamento				
		Taxa de espalhamento (kchip/s)	Modulação	Taxa de bits (kb/s)	Taxa de símbolo	Símbolos
868/915	868-868,6	300	BPSK	20	20	Binário
	902-928	600	BPSK	40	40	Binário
868/915	868-868,6	400	ASK	250	12,5	20-bit PSSS
	902-928	1600	ASK	250	50	20-bit PSSS
868/915	868-868,6	400	O-QPSK	100	25	16 símbolos
	902-928	1000	O-QPSK	250	62,5	16 símbolos
2450	2.400-2.483,5	2000	O-QPSK	250	62,5	16 símbolos

Tabela 2.5b: As faixas de frequências disponíveis pelo IEEE 802.15.4 com suas modulações. [1] [7] [8] [9] [10]

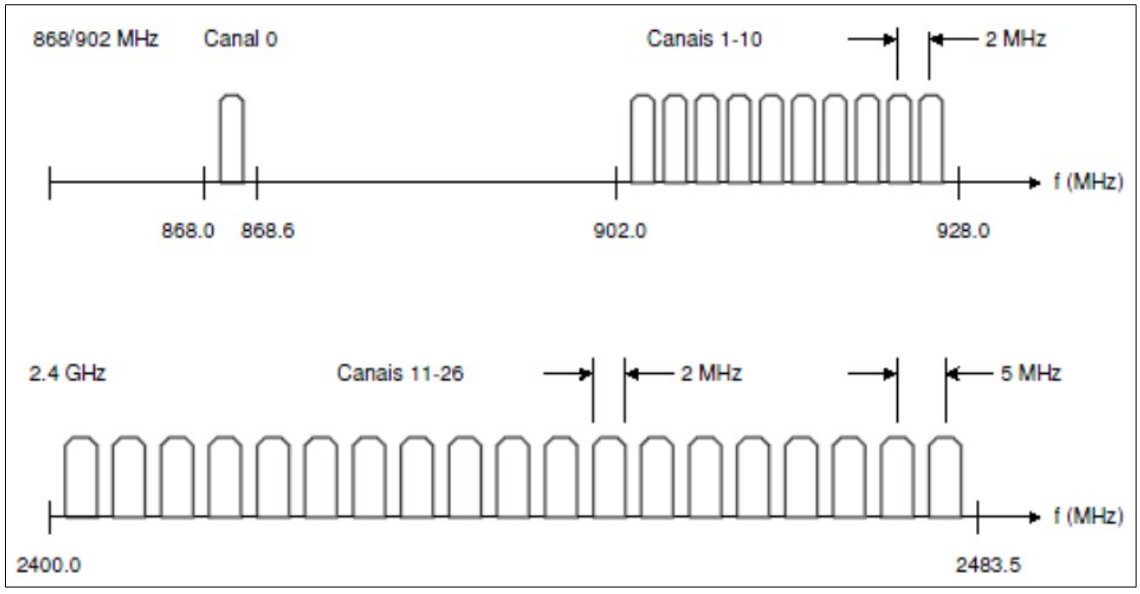


Figura 2.5: Canais por frequência de operação. [4]

Complementando no padrão IEEE Std 802.15.4™-2006 (Revision of IEEE Std 802.15.4-2003). Para estarem de acordo com a RESOLUÇÃO ANATEL N§ 506/2008 (Radiação Restrita) os equipamentos que utilizam Tecnologia de Espalhamento Espectral devem operar em 902-907,5 MHz, 915-928 MHz, 2400-2483,5MHz. [10]

A camada física é responsável pela ativação e desativação do transceptor, seleção do canal de frequência, transmissão e recepção de dados, detecção de energia no canal (ED),

indicação da qualidade do link (LQI) para os pacotes recebidos e CCA (clear channel assessment) para o protocolo CSMA-CA da camada MAC. A medida da detecção de energia do receptor (ED) é usada pela camada de rede como parte do algoritmo de seleção de canal. Corresponde a uma estimativa da potência do sinal recebido dentro da largura de banda do canal IEEE 802.15.4. [4]

A medida LQI (Link Quality Indication) é uma caracterização da intensidade e/ou qualidade do pacote recebido. A medida pode ser implementada usando a medida ED, uma estimativa da relação sinal/ruído ou uma combinação desses métodos. O LQI é reportado como um inteiro de 8 bits. Os valores máximo e mínimo LQI são associados com os valores de mais baixa e alta qualidade dos sinais IEEE 802.15.4 detectáveis pelo receptor, e os outros valores estariam uniformemente distribuídos entre esses dois limites. [4]

O CCA (Clear Channel Assessment) é executado de acordo com a configuração de um dos métodos descritos abaixo: [4]

- Energia acima do nível: CCA reportará o estado do meio como ocupado após detectar um nível de energia acima do nível ED;
- Detecta somente a portadora: CCA reportará o estado do meio como ocupado após a detecção do sinal da portadora. Este sinal pode estar acima ou abaixo do nível ED;
- Detecta portadora com energia acima do nível: CCA reportará o estado do meio como ocupado após a detecção da portadora com energia acima do nível ED.

A estrutura do pacote PPDU (PHY Protocol Data Unit), ilustrado na figura 2.9 consiste nos seguintes componentes básicos:

- SHR (Synchronization Header), que permite ao dispositivo receptor sincronizar com o feixe de bits, através de 4 Bytes correspondentes ao campo PS (Preamble Sequence) e um Byte no campo SFD (Start of Frame Delimiter);
- PHR (PHY Header), campo de 1 Byte (FL - Frame Length) que contém informação do comprimento em Bytes do quadro PSDU;
- PSDU (PHY Service Data Unit) que são os dados de comprimento variável vindos da camada MAC (MPDU).

O tamanho máximo do PSDU é fornecido pela constante aMaxPHYPacketSize, que é

igual a 127 Bytes.

No caso do quadro de reconhecimento os campos SHR e PHR são idênticos ao quadro de dados, porém o PSDU é composto somente de 5 Bytes vindos da camada MAC (vide figura 2.8). [4]

2.2.4 IEEE 802.15.4 MAC

A subcamada MAC trata todo acesso ao canal de rádio físico e é responsável pelas seguintes tarefas: geração e sincronização de beacons; suporte de associação e desassociação na rede PAN; suporte opcional à segurança do dispositivo; gerenciamento de acesso ao canal via CSMA-CA; manutenção dos tempos reservados (slots GTS e prover validação e reconhecimento de mensagem. Os beacons são pacotes de controle que delimitam quadros utilizados pelo coordenador para sincronizar com os demais dispositivos da rede. [4]

Uma rede PAN pode ser configurada com beacon habilitado ou desabilitado. No caso de uma rede com beacon desabilitado, os dispositivos podem comunicar-se em qualquer tempo após uma fase de associação. O acesso ao canal e a contenção são gerenciados usando o mecanismo CSMA-CA (Carrier Sense Multiple Access with Collision Avoidance). Cada vez que um dispositivo quer transmitir um quadro de dados ou comandos MAC, ele espera por um período randômico de tempo. Se após a espera o canal é encontrado livre, o dispositivo transmite seu dado. Se o canal está ocupado o dispositivo aguarda um outro período randômico antes de tentar acessar o canal novamente. Quadros de reconhecimento são enviados sem usar o mecanismo CSMA-CA. [4]

Em uma rede com beacon habilitado, o coordenador da rede PAN transmite um beacon periodicamente no qual os outros dispositivos o usam para sincronização e para determinação de quando estão liberados para transmissão e recepção de mensagens. A mensagem beacon é usada para definir uma estrutura chamada de superframe em que todos os nós na rede PAN a serem sincronizados. Esta estrutura é mostrada na figura 2.6. [4]

A norma IEEE802.15.4 permite o uso opcional de uma estrutura superquadro (SuperFrame). O formato de um superframe é definido pelo coordenador. O superframe é limitado pelos beacons e é enviado pelo coordenador. O quadro beacon é transmitido no primeiro intervalo de tempo (slot) de cada superframe. Se o coordenador não deseja usar a

estrutura de superframes, ele pode desligar as transmissões de beacons. [4]

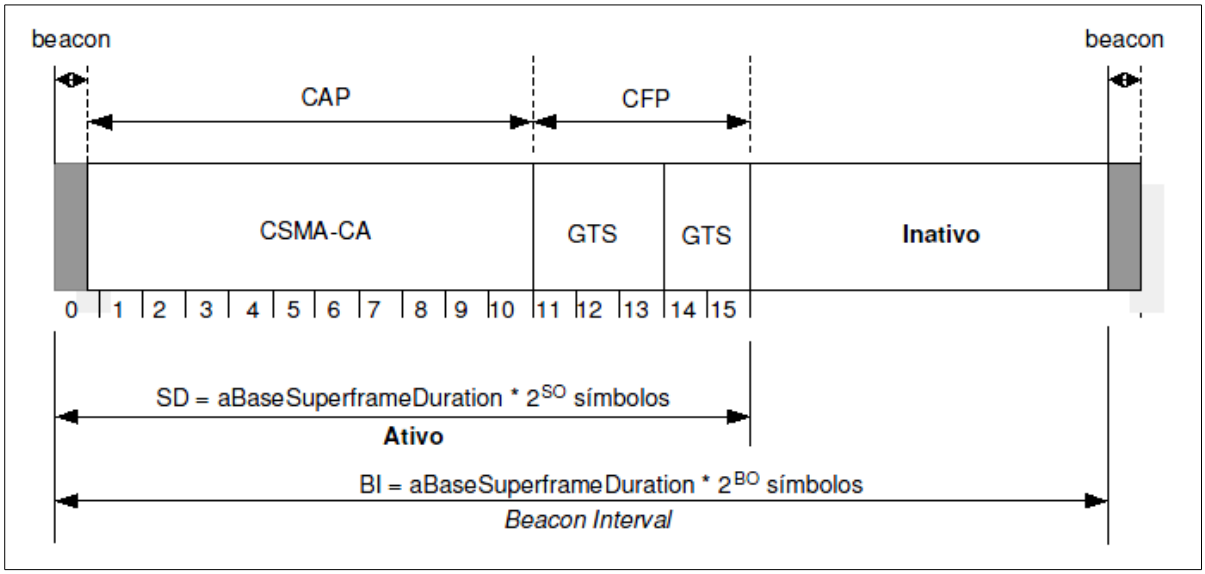


Figura 2.6: Quadro SuperFrame

Os beacons são usados para sincronizar os dispositivos associados, para identificar a PAN, e para descrever a estrutura dos superframes. Existe um período ativo durante o qual a comunicação se processa e um período inativo (opcional), durante o qual os dispositivos podem desligar seus transceptores para conservar energia. O período ativo é dividido em 16 períodos de tempo iguais (slots). Imediatamente seguindo o beacon vem o período de acesso de contenção (CAP). Durante este período os dispositivos devem se comunicar usando o mecanismo sloted CSMA-CA. Isto é similar ao CSMA-CA sem slots exceto pelo fato de que os períodos de backoff são alinhados com os limites dos slots. O CAP deve conter pelo menos oito períodos ativos mas pode chegar a até 16. Seguindo o CAP temos um opcional período livre de contenção (CFP), que pode ter até sete períodos ativos. Em um CFP o coordenador PAN reserva períodos (GTS) para algum dispositivo. Durante um GTS um dispositivo tem acesso exclusivo ao canal e não executa CSMA-CA. Durante um desses GTSs, um dispositivo pode transmitir dados ou receber dados de seu coordenador PAN, mas não ambos. Um GTS será reservado somente pelo coordenador da rede (PAN). O comprimento de um GTS deve ser um múltiplo inteiro de um período ativo (slot). Todos os GTSs devem ser contíguos no CFP e são localizados no final do período ativo do superframe. Um dispositivo pode desabilitar seu transceptor durante um GTS designado para outro dispositivo a fim de conservar energia. Para

cada GTS o coordenador armazenará no pacote beacon o intervalo de partida (slot), comprimento, direção e endereço do dispositivo associado. A direção do GTS é especificada como transmissão ou recepção. [4]

A estrutura do superframe é definida pelos valores dos atributos macBeaconOrder (BO) e macSuperframeOrder (SO). A Equação 1 mostra o intervalo de tempo no qual o coordenador transmitirá seus quadros beacon é definido como BI (Beacon Interval) e se relaciona com o parâmetro BO (macBeaconOrder) da seguinte maneira: para 0 <= BO <= 14

Onde aBaseSuperframeDuration = 960 simbolos.

$$BI = aBaseSuperframeDuration * 2^{BO} \, simbolos \quad (1)$$

Se BO=15 o superframe não existirá e o valor do macSuperframeOrder, SO, será ignorado. [4]

A unidade símbolo é uma unidade de tempo que depende do tipo de modulação utilizado em cada banda de frequência. Nas bandas de 902/868 MHz a modulação é BPSK e um símbolo corresponde a um 1 bit, enquanto na banda de 2.4 GHz com modulação O-QPSK um símbolo corresponde a 4 bits. [4]

O atributo MAC macSuperframeOrder (SO) define o comprimento da porção ativa do superframe. A Equação 2 o intervalo de tempo relativo à parte ativa do superframe SD (Superframe Duration) se relaciona com o parâmetro SO (macSuperframeOrder) da seguinte maneira: para 0 <= SO <= BO <= 14,

$$BD = aBaseSuperframeDuration * 2^{SO} \, simbolos \quad (2)$$

A porção ativa de cada superframe é dividida em 16 intervalos igualmente espaçados com duração de aBaseSlotDuration * 2^{SO} , onde aBaseSlotDuration = 60 simbolos. O superframe é composto de 3 partes: um beacon, um CAP (Contention Access Period) e um CFP (Contention Free Period). O beacon é transmitido sem o uso de CSMA, no início do slot 0, e o CAP começa imediatamente após o beacon. O CFP se presente, segue imediatamente após o CAP e se estende até o final da porção ativa do superframe. Todos os GTSs alocados estarão dentro do CFP. [4]

Durante SD, a parte ativa, os nós podem enviar seus quadros no começo de cada slot usando slotted CSMA/CA durante o período CAP. No caso do canal estar ocupado, o nó computa seu período de backoff baseado em um número de intervalos de tempo (slots). [4]

As redes PAN que quiserem usar uma estrutura de superframe configurarão o atributo macBeaconOrder (BO) com um valor entre 0 e 14 e macSuperframeOrder com um valor entre 0 e o valor de macBeaconOrder. Em caso contrário estes atributos seriam configurados com o valor de 15. Neste caso o coordenador não transmite beacons e todas as transmissões, com a exceção de quadros de reconhecimento e quadros de dados que imediatamente seguem o reconhecimento de um comando de pedido de dados usam o mecanismo CSMA-CA para acessar o canal. Além disso os GTSs não são permitidos. A rede LR-WPAN define quatro estruturas de quadros: quadro de beacon, quadro de dados, quadro de reconhecimento e quadro de comandos MAC. [4]

O formato geral do quadro de dados MAC (MPDU) é dado na figura 2.7 e consiste nos seguintes componentes básicos: [4]

- MHR (MAC header), que contém um campo de controle de 2 Bytes (FC – Frame Control), 1 Byte para número de sequência e de 4 a 20 Bytes para campo de endereçamento;
- MSDU (MAC Service Data Unit) são os dados de comprimento variável, que são provenientes da camada superior;
- MFR (MAC footer), é composto de 16 bits FCS (Frame Check Sequence).

Portanto o tamanho máximo do overhead na camada MAC (MHR + MFR) no quadro de dados é igual a 25 Bytes. [4]

A recepção com sucesso e a validação de um quadro de dados ou de comando MAC pode ser opcionalmente confirmado com um reconhecimento. Se o dispositivo fonte não recebe um reconhecimento após algum período de tempo, ele assume que a transmissão não teve sucesso e repete a transmissão do quadro.

O quadro de reconhecimento, mostrado na figura 2.8, consiste nos seguintes campos:

- MHR (MAC header), que contém um campo de controle de 2 Bytes (FC – Frame Control) e 1 Byte para número de sequência;
- MFR (MAC footer), é composto de 16 bits FCS (Frame Check Sequence).

Os campos MHR e MFR juntos formam o quadro de reconhecimento MAC (MPDU).

O período IFS (Interframe Spacing) define a quantidade de tempo que separa a transmissão de dois quadros consecutivos. De fato, a subcamada MAC necessita de uma quantidade finita de tempo para processar o dado recebido pela camada física. Se uma transmissão requer um reconhecimento, a separação entre o quadro de reconhecimento e a próxima transmissão será de, pelo menos, um período IFS. A duração de um período IFS é dependente do tamanho do quadro transmitido. Quadros (MPDUs) de até aMaxSIFSFrameSize = 18 Bytes de comprimento serão seguidos de um período SIFS de uma duração de no mínimo aMinSIFSPeriod = 12 símbolos. Quadros (MPDUs) com comprimentos maiores que aMaxSIFSFrameSize Bytes serão seguidos por um período LIFS de uma duração de no mínimo aMinLIFSPeriod = 40 símbolos. A figura 2.9 ilustra estes conceitos. O algoritmo CSMA-CA leva em conta esta requisição nas transmissões no CAP (Contention Access Period). [4]

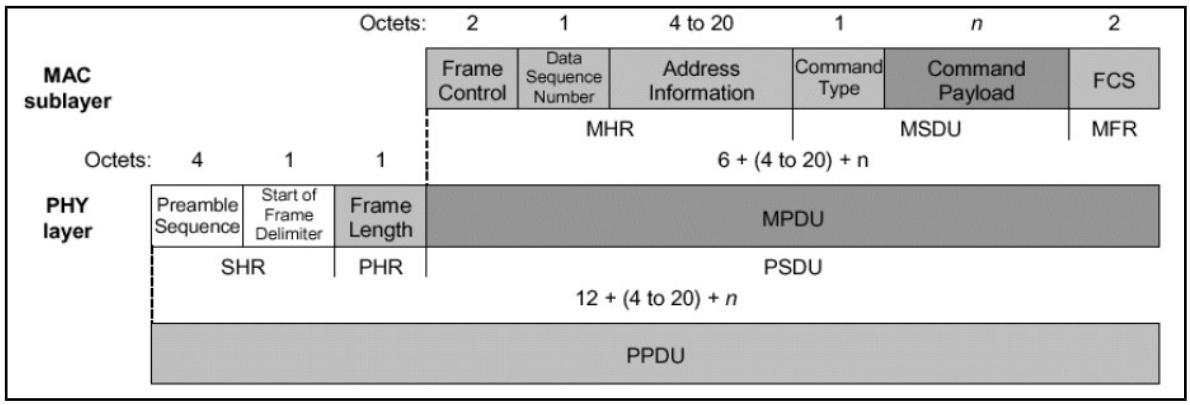


Figura 2.7: Formato do Pacote de Dados. [8] [3]

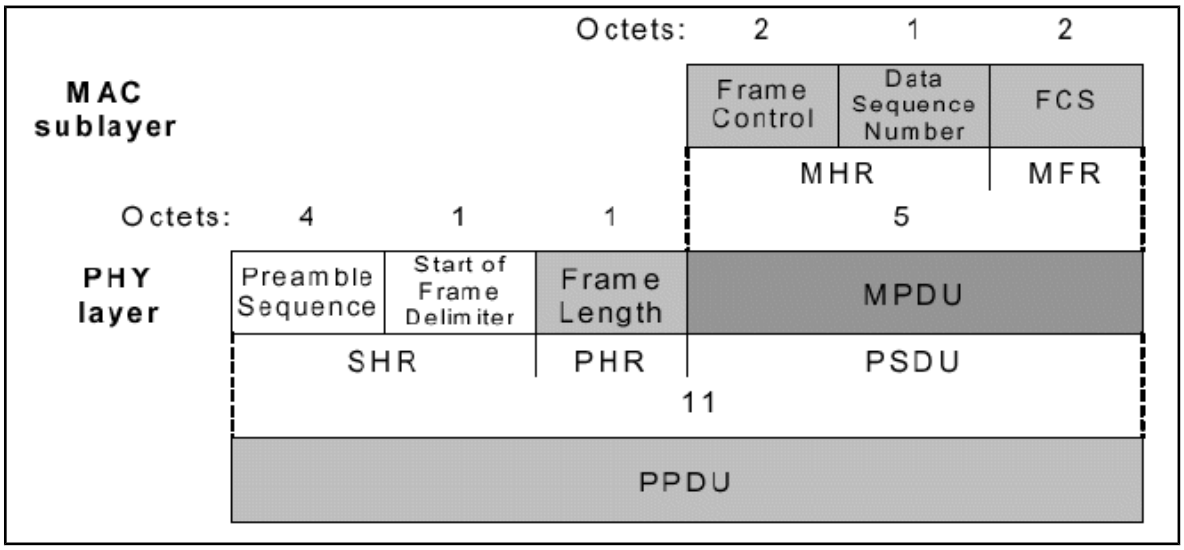


Figura 2.8: Formato do Pacote de Reconhecimento. [8] [3]

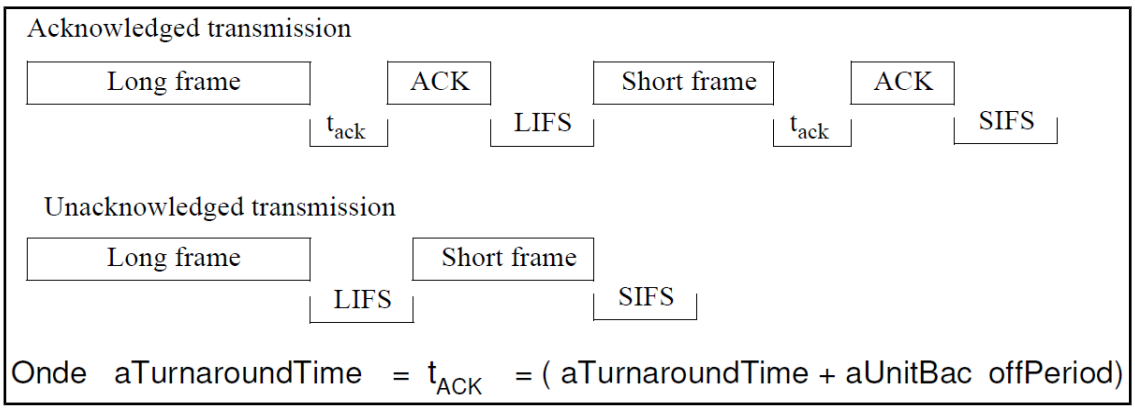


Figura 2.9: Intervalo entre quadros. [3]

2.2.5 DATA TRANSFER MODEL

Existem três tipos de transferência de dados. O primeiro é a transferência de dados do dispositivo para o coordenador. O segundo é a transferência de dados de um coordenador para um dispositivo. A terceira é a transferência de dados entre dois dispositivos ponto a ponto. Em uma topologia estrela somente duas dessas transferências são usadas, porque os dados somente podem ser trocados entre o coordenador e um dispositivo. Na topologia ponto a ponto os dados são trocados entre dispositivos na rede, e consequentemente todas as três

formas de transferência podem ser utilizadas. [4]

O mecanismo para cada tipo de transferência depende se a rede suporta a transmissão de beacons. Uma rede com beacon habilitado é usada para suportar dispositivos de baixa latência. Se a rede não necessita suportar tais dispositivos, ela pode não usar o beacon para transferências normais, contudo o beacon é ainda utilizado para associação de rede. O primeiro tipo de transferência de dados é um mecanismo que transfere dados de um dispositivo para um coordenador. Quando um dispositivo quer transferir dados para um coordenador em uma rede com beacon habilitado, ele primeiro aguarda o beacon da rede, e uma vez detectado, o dispositivo se sincroniza com a estrutura superframe. No ponto apropriado o dispositivo transmite seu quadro de dados, usando o CSMA-CA para o coordenador. O coordenador reconhece a recepção de dados correta transmitindo um quadro de reconhecimento opcional, conforme mostrado na figura 2.10(a). Quando o dispositivo quer transferir dados em uma rede sem habilitação de beacons, ele transmite seu quadro de dados usando CSMA-CA para o coordenador. O coordenador reconhece a recepção correta dos dados transmitindo um quadro de reconhecimento opcional, conforme mostrado na figura 2.10(b). [4]

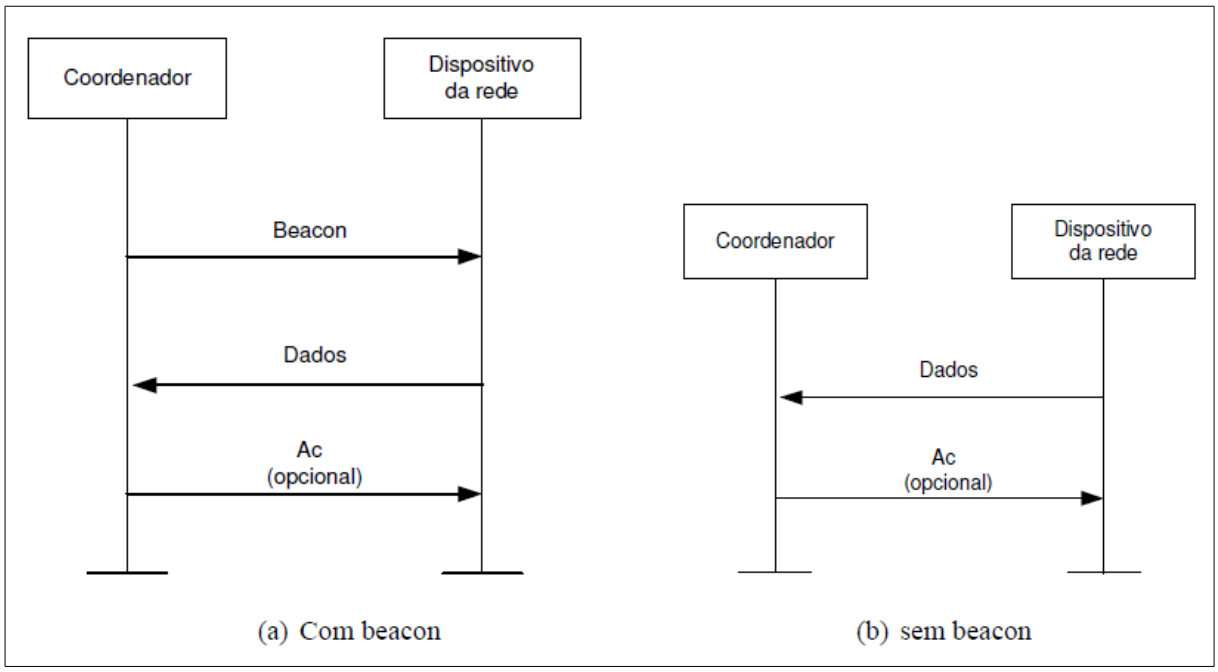


Figura 2.10: Transferência de Dados para o Coordenador

O segundo tipo de transferência de dados, conhecido como transmissão indireta é o mecanismo que transfere dados de um coordenador para um dispositivo. Quando o coordenador quer transferir dados para um dispositivo em uma rede com beacon habilitado, ele indica no beacon da rede que a mensagem de dados está pendente. O dispositivo periodicamente escuta os beacons da rede e se uma mensagem está pendente, transmite um comando MAC requisitando o dado, usando CSMA-CA. O coordenador reconhece a recepção correta da requisição de dados transmitindo um quadro de reconhecimento. O quadro de dados pendente é então enviado usando CSMA-CA. O dispositivo reconhece a recepção correta dos dados transmitindo um quadro de reconhecimento. Após receber o reconhecimento, a mensagem é removida da lista de mensagens pendentes no beacon. Esta sequência é mostrada na figura 2.11(a). [4]

Quando um coordenador quer transferir dados para um dispositivo em uma rede sem habilitação de beacons, ele armazena os dados do dispositivo apropriado e aguarda um contato e requisição dos dados. O dispositivo pode fazer contato transmitindo um comando MAC de requisição de dados, usando CSMA-CA para seu coordenador. O coordenador reconhece a recepção correta do pedido de dados, transmitindo um quadro de reconhecimento. Se os dados estão pendentes, o coordenador transmite o quadro de dados, usando CSMA-CA para o dispositivo. Se o dado não está pendente, o coordenador transmite um quadro de dados com comprimento zero para indicar que não havia dados pendentes. Sequência mostrada na figura 2.11(b). [4]

Na transferência de dados em uma rede PAN ponto a ponto, cada dispositivo pode se comunicar com qualquer outro dispositivo que esteja dentro do alcance de seu rádio de transmissão.

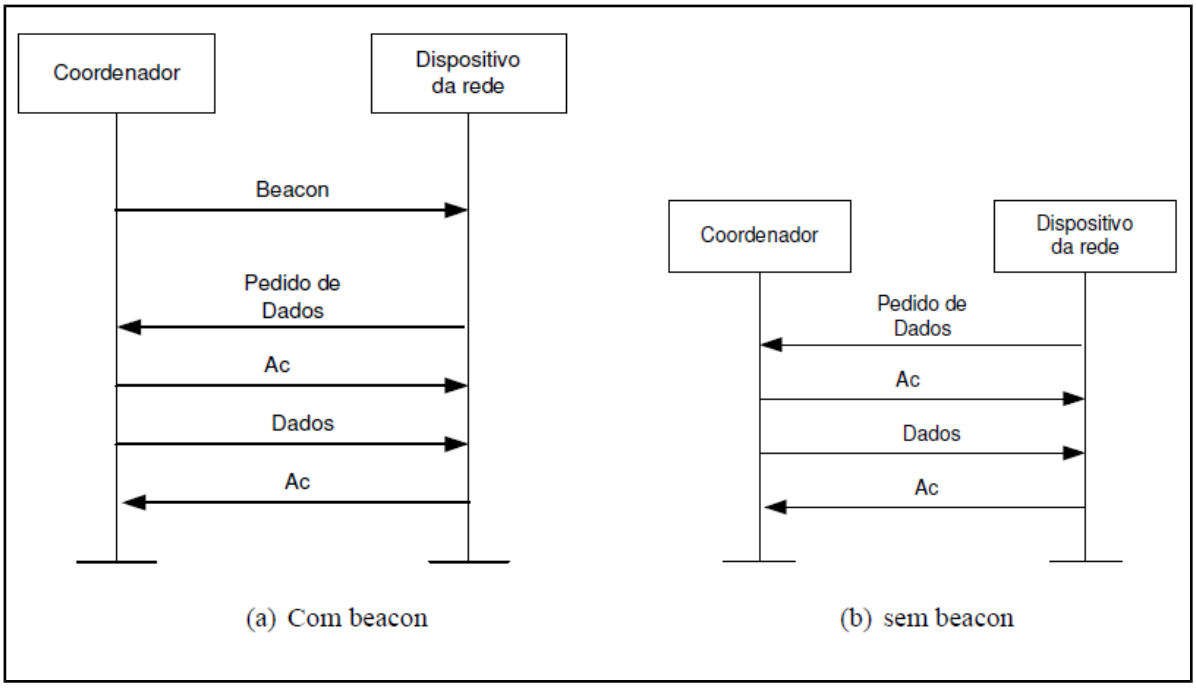


Figura 2.11: Transferência de Dados do Coordenador. [4]

2.2.6 ASSOCIATION AND DISASSOCIATION

A associação de um dispositivo parte após ele ter completado uma exploração ativa ou passiva do canal. A exploração passiva permite que um dispositivo localize algum coordenador transmitindo quadros beacons dentro de sua área de alcance (POS), enquanto que na exploração ativa o dispositivo transmite um comando de pedido de beacon. Os resultados da exploração são então usados para escolher uma PAN adequada caracterizada pelo seu canal físico (phyCurrentChannel), seu identificador (macPANId) e seu endereço curto (CoordShortAddress) ou longo (CoordExtendedAddress). [4]

Um dispositivo não associado iniciará seu procedimento enviando um comando de pedido de associação para o coordenador de uma rede PAN existente. Se o comando é recebido corretamente, o coordenador enviará um quadro de reconhecimento (acknowledgement). Este reconhecimento contudo não significa que o dispositivo tenha sido associado. O coordenador necessita de tempo para determinar se os recursos utilizados em uma PAN são suficientes para permitir mais um outro dispositivo como associado. Esta decisão é tomada dentro de um intervalo de tempo de aResponseWaitTime símbolos. Se

existirem recursos suficientes, o coordenador reservará um endereço curto para o dispositivo e gerará um comando de resposta de associação contendo o novo endereço e um estado indicando sucesso na associação. Se não existirem recursos suficientes, o coordenador gerará um comando de resposta de associação contendo um estado indicando falha. Esta resposta será enviada para o dispositivo usando transmissão indireta. [4]

No outro lado, o dispositivo, após obter o quadro de reconhecimento, espera pela resposta durante o tempo de aResponseWaitTime símbolos. Na recepção de um comando de resposta de associação, o dispositivo enviará um quadro de reconhecimento. Se a associação tiver sucesso, ele armazenará o endereço do coordenador. [4]

Quando o coordenador quer que um de seus dispositivos associados deixe a rede PAN, ele envia um comando de notificação de desassociação para o dispositivo, usando transmissão indireta. Após recepção do pacote, o dispositivo envia um quadro de reconhecimento. Mesmo que este reconhecimento não seja recebido, o coordenador considerará o dispositivo desassociado. [4]

Se um dispositivo associado quer deixar o PAN, ele envia um comando de notificação de desassociação para o coordenador. Após a recepção, o coordenador envia um quadro de reconhecimento. Mesmo se o reconhecimento não é recebido o dispositivo considerará desassociado. Ao desassociar-se, o dispositivo removerá todas as referências daquela rede PAN e o coordenador removerá todas as referências do dispositivo desassociado. [4]

2.2.7 SEGURANÇA NA CAMADA MAC USANDO NWK

A segurança na camada MAC como mostra a figura 2.12 protege os frames MAC transmitidos em um único salto na rede. Para saltos múltiplos, a segurança é feita nas camadas superiores: NWK e Aplicações. [5]

A segurança MAC utiliza o algoritmo AES (Advanced Encryption Standard) para criptografar e validar o dado que é enviado. A validação ou garantia de integridade do dado é feita por MIC (Message Integrity Code). [5]

Caso seja necessário utilizar segurança, um bit do cabeçalho MAC será setado. Com isso, é anexado ao frame o Cabeçalho Auxiliar de Segurança que determina o tipo de proteção utilizado (Security Control), o Contador de Frames (Frame Counter) que garante a sequência

e autenticação dos dados e guarda referência da chave (Key Identifier) de 128 bits a ser utilizada para determinado nó. [5]

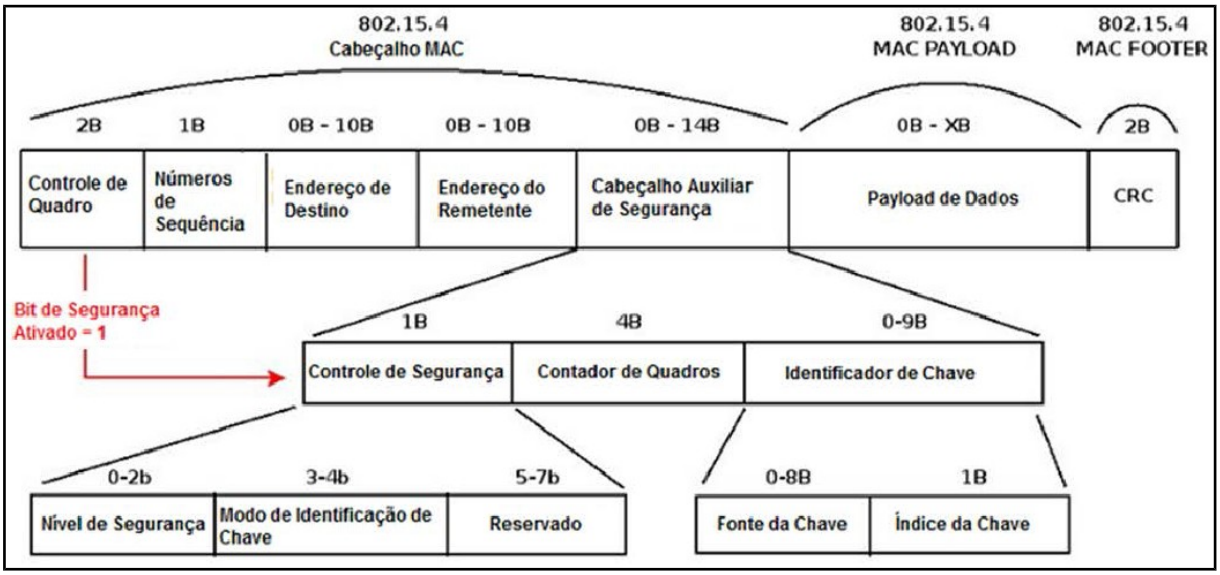


Figura 2.12: Segurança na camada MAC [4] [5]

Quando se é desejada segurança na camada MAC, ZigBee usa a segurança da camada MAC para assegurar (confiar) o comando MAC, beacons, e quadros de reconhecimento (acknowledgement frames). Para transmissão de mensagens através de um único nó o ZigBee utiliza quadros seguros de MAC, porém para vários nós o ZigBee se apóia em camadas superiores (como a camada NWK). A camada MAC utiliza AES (Advanced Encryption Standard) como seu principal algoritmo de criptografia e define uma variedade de suítes que utilizam o algoritmo AES. Essas suítes podem proteger a confidencialidade, integridade e autenticidade dos quadros MAC. A camada MAC faz a segurança, porém as camadas superiores são quem montam as chaves e define os níveis de segurança a serem usadas, controladas e processadas. Quando a camada MAC transmite (ou recebe) um quadro com segurança habilitada, ela procura pelo destino do quadro, recupera a chave associada com aquele destino, e então usa essa chave para o quadro de acordo com a suíte de segurança designada para a chave utilizada. Cada chave é associada com uma única suíte de segurança, e o cabeçalho do quadro MAC possui um bit que especifica se a segurança para o quadro está habilitada ou não. [11]

2.2.8 ZIGBEE ROUTING LAYER

O roteamento ZigBee executa dois roteamentos distintos, roteamento hierárquico, utilizado principalmente na topologia em árvore, e roteamento AODV (Ad hoc On Demand Distance Vector). AODV é um algorítmo reativo de roteamento, ou seja, um nó não tem que descobrir ou manter uma rota para outro nó, a não ser que haja necessidade de comunicação. Quando o nó fonte deseja transmitir dados para um nó de destino, dá-se o início da descoberta de rota. O nó fonte constrói um pacote de RREQ (Route Request) e o envia por difusão (broadcasting). Cada nó que o recebe e não conhece o destino, o reenvia apenas uma vez. Ao chegar no destino ou em algum nó que o conheça, é construído o pacote RREP (Route Reply) e enviado de volta pelo caminha reverso. Com a chegada deste pacote no nó fonte, fica então definido um caminho para a transmissão dos dados e o nó fonte pode iniciar a transmissão. [4]

Atribuição de Endereços na rede ZigBee. Como parte do processo de ingressar em uma rede, cada dispositivo recebe um endereço de rede lógico. Em redes ZigBee, os endereços de rede são atribuídos ou por um coordenador ou por um roteador, usando um algoritmo de árvore estruturada. [6]

No mais alto nível da estrutura (no dispositivo coordenador) da rede está definida uma entidade conhecida como "stack profile". O “stack profile” é um conjunto de parâmetros que incluem definições da profundidade máxima da rede, o número máximo de filhos roteadores em uma profundidade e o número máximo de filhos (dispositivos finais) que podem comunicar-se com um roteador individual. Estes parâmetros determinam a forma da árvore da rede. Por exemplo, a profundidade da rede determina o número máximo de saltos entre qualquer dispositivo e o coordenador de rede. [6]

Seguindo o raciocínio da referência bibliográfica [6] ele usou o exemplo abaixo para explicar as atribuições dos endereços.

Existe exatamente um coordenador ZigBee em cada rede. Este dispositivo agrega o maior número de funções, por exemplo, o coordenador é capaz de criar uma rede tornando-se a raiz da árvore dessa rede, sendo portanto, o único dispositivo capaz de comutar dados entre redes. [6]

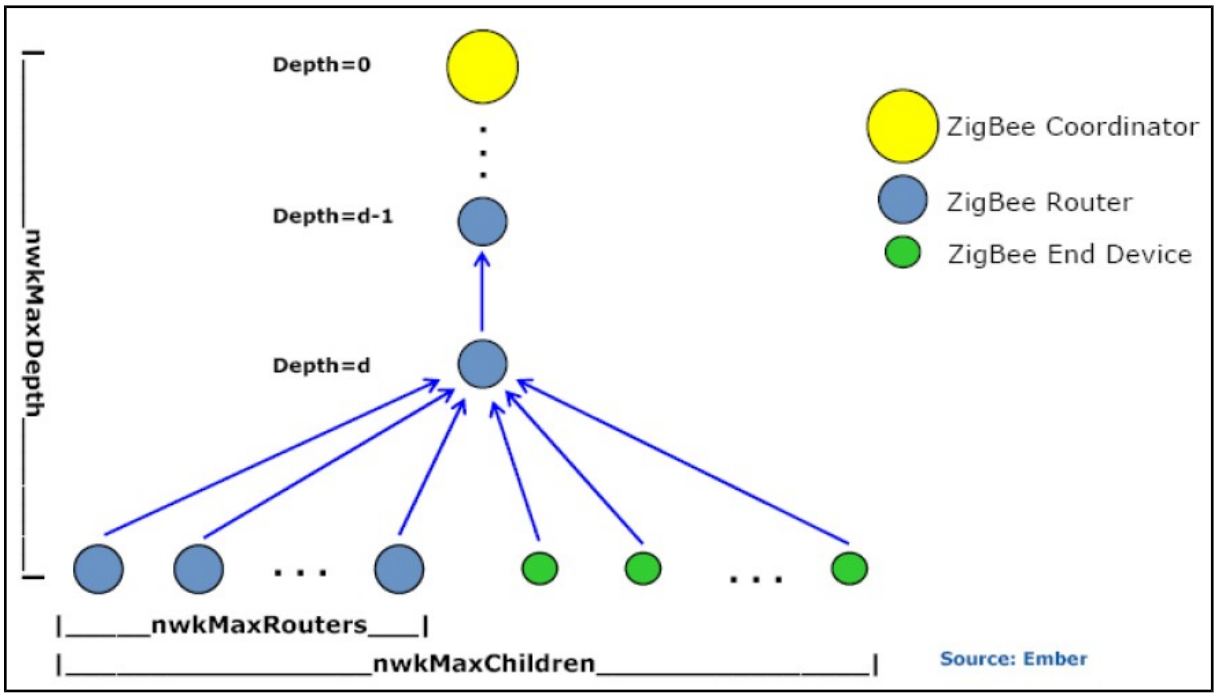


Figura 2.13: Árvore lógica de uma ZigBee. [6]

Além disso, o coordenador é capaz de armazenar informação sobre a rede como, por exemplo, um repositório de chaves seguras e determinar parâmetros como o número máximo de filhos (Cm) de um roteador ZigBee , o número máximo de roteadores filhos (Rm) de um nó pai, a profundidade da rede (Lm). [6]

A partir desses parâmetros, os dispositivos roteadores podem computar um parâmetro denominado Cskip, que é utilizado para computar o tamanho do "pool" de endereços de filhos. Este parâmetro é calculado pela seguinte fórmula apresentado na figura 2.14: [6]

$$Cskip(d) = \begin{cases} 1 + Cm \cdot (Lm - d - 1), & \text{se } Rm = 1 \quad \cdots\cdots\cdots\cdots (a) \\ \dfrac{1 + Cm - Rm - Cm \cdot Rm^{Lm-d-1}}{1 - Rm}, & \text{Caso contrário} \quad \cdots\cdots\cdots\cdots (b) \end{cases}$$

Figura 2.14: Fórmula para computar o parâmetro Cskip(d). [6]

Estes atributos da árvore lógica podem ser usados para computar o endereço lógico e determinar o lugar de um dispositivo na árvore. Isto simplifica a tarefa de roteamento na

árvore. Como opção, a especificação ZigBee também permite que a atribuição de endereços seja realizada pela aplicação. A atribuição de endereços é feita da seguinte forma: [6]

Na figura 2.15 mostra que se um nó pai (roteador ou coordenador) na profundidade d tem um endereço Aparente, então:

- ao n^{th} roteador filho é atribuido o endereço: $A_{parent} + (n-1) \times C_{skip}(d) + 1$
- *ao* n^{th} dispositivo final filho, é atribuido o endereço: $A_{parent} + Rm \times C_{skip}(d) + n$

Figura 2.15: Atribuição de Endereços. [6]

É importante notar que o roteamento ZigBee executa dois roteamentos distintos, roteamento hierárquico, caso o roteador não tenha "capacidade de roteamento" e roteamento sob demanda (semelhante ao AODV). Estes procedimentos intermediários entre outros, como processo de inundação serão abordados nas seções seguintes. Mais informações vide a referência bibliográfica [6]

Um dispositivo tem capacidade de tabela de rota se:

- É um coordenador ou roteador ZigBee;
- Possui uma tabela de rotas;
- Possui registro livre em sua tabela de rota ou um registro correspondente ao endereço destino na tabela de rotas;
- Um dispositivo tem capacidade de tabela de descoberta de rota se:
- Possui uma tabela de descoberta de rota
- Possui um registro livre em sua tabela de descoberta de rota.

Se um dispositivo tem ambos, capacidade de tabela de descoberta de rota e capacidade de tabela de rota então dizemos que ele possui Capacidade de Roteamento.

Este procedimento de descoberta de rota é baseado no roteamento AODV (Ad hoc On Demand Distance Vector) e é utilizado caso o roteador não esteja diretamente ligado ao dispositivo destino, mas possui “capacidade de roteamento”. Se o dispositivo não tem um registro na tabela de rotas para o destino, ele deve estabelecer um registro na tabela com o campo “status” igual a constante DISCOVERY_UNDERWAY. Depois ele enviará um frame com o comando de descoberta de rota em broadcast. Cada dispositivo que emite um frame de requisição de rota deve manter um contador usado para gerar identificadores destas requisições. [6]

3 DESCRIÇÃO E ESPECIFICAÇÕES TÉCNICAS DOS MATERIAIS UTILIZADOS

3.1 CUSTO DO PROJETO

Materiais	Quantidade	Valor	Modelo	Revendedora	Fabricante
Protoboard	x1	R$ 70,00	Protoboard 2420 Furos 4 Bornes MSB-400 ICEL	Ferramentaskennedy	ICEL Manaus
Arduino	x1	US$ 60.20	Arduino Mega 2560 mais acessórios para Funduino	DX – MVProduct	Fundino
Modulo GSM/GPRS para Arduino	x1	US$ 59.10	SIM900 (GSM/GPRS)	DX	ElecFreaks
Fonte para Protoboard	x1	R$ 35,00	Fonte de alimentação para Protoboard 5v / 3.3v:	MULTILOGICA-SHOP	
Xbee Shield Para Arduino	x1	R$ 44,90	Xbee Shield v.14 – XBE0003	TECHMOUNT	DFRobot
Módulos Xbee	x2	US$ 81.00	XB24-AUI-001 – Modulo Xbee de 1mw Série 1	Ebay – Duino Shop E Sparkfun	Digi E Maxstream

Tabela 3.1: Materiais usados no Projeto

Valor total aproximadamente em real: R$ 646,32

Link das fontes e valores sujeitos a variações. Cotação no valor de R$ 2,40 e o valor do IOF não foi levada em consideração.

3.2 ARDUINO

Arduino, palavra por vezes traduzida ao português como Arduíno, é uma plataforma de prototipagem eletrônica de hardware livre e de placa única, projetada com um microcontrolador Atmel AVR com suporte de entrada/saída embutido, uma linguagem de programação padrão, a qual tem origem em Wiring, que é essencialmente C/C++. O objetivo do projeto é criar ferramentas que são acessíveis, com baixo custo, flexíveis e fáceis de se usar por artistas e amadores, principalmente para aqueles que não teriam alcance aos

controladores mais sofisticados e de ferramentas mais complicadas. [12][13]

O surgimento do Arduino se dá no ano de 2005, na Itália, por um professor chamado Massimo Banzi, com intuito de ensinar eletrônica e programação de computadores para os alunos de design, para uso em projeto de arte, interativos e robótica. [12][13]

Foi pelo motivo de que ensinar programação para pessoas que não eram da área era muito difícil e além de ter placas caras baratas no mercado. Com isso o Massimo e David Cuartielles decidiram criar sua própria placa. Também foi preciso criar uma linguagem de programação. [12][13]

Ele faz parte do conceito de uma plataforma eletrônica "open-source". Isso significa que qualquer pessoa pode montar seu próprio Arduino e modificá-lo, tanto o software quanto o hardware. Existem inúmeras pessoas desenvolvendo diversas aplicações programas e placas que podem ser acopladas ao Arduino. [12][13]

O Arduino original é fabricado pela companhia italiana Smart Projects, porém a estadunidense SparkFun Electronics também possui algumas marcas comerciais sob a mesma licença. [12] [13]

Os projetos e esquemas de hardwares são distribuídos sob a licença Creative Commons Attribution Share-Alike 2.5, e estão disponíveis em sua página oficial. Arquivos de layout e produção para algumas versões também estão hospedadas. O código fonte para o IDE e a biblioteca de funções da placa são disponibilizadas sob a licença GPLv2 e hospedadas pelo projeto Google Code. [12] [13]

O Arduino é uma plaquinha a qual podemos programar através de computador para executar as mais variadas funções; podendo ser utilizado para leitura de sensores, controlar atuadores (motor, lâmpada, etc.), processando informações e com isso realizando o controle dos dispositivos. (Entretanto, é necessário utilizar computadores para programá-lo). [12] [13]

Além do próprio Arduino foram criados os Shields, por terceiros, que são placas complementares do Arduino que são especializadas para determinadas funções como por exemplo o Ethernet Shield que oferece a interface necessária para que o Arduino consiga comunicar em rede através da interface RJ45.

O Arduino foi concebido para ser de fácil entendimento, programação e aplicação; além de ser multiplataforma, podendo ser configurado em qual quer tipo de ambiente (Linux,

Windows e Mac OS).

O Arduino utiliza o microcontrolador Atmega. O microcontrolador (denominado MCU) é um computador em um chip, que contém toda a estrutura desde microprocessador, memória até os periféricos de entrada/saída. Pode ser embarcado no interior de outro dispositivo (que no caso é o Arduino), para que possa controlar suas funções e ações.

Pode ser usado para o desenvolvimento de objetos interativos independentes, ou ainda para ser conectado a um computador hospedeiro. Uma típica placa Arduíno é composta por um controlador, algumas linhas de E/S digital e analógica, além de uma interface serial ou USB, para interligar-se ao hospedeiro, que é usado para programá-la e interagir em tempo real. Ela em si não possui nenhum recurso de rede, porém é comum combinar um ou mais Arduinos deste modo, usando extensões apropriadas chamadas de shields. A interface do hospedeiro é simples, podendo ser escrita em várias linguagens. A mais popular é a Processing, mas outras que podem comunicar-se com a conexão serial são: Max/MSP, Pure Data, SuperCollider, ActionScript e Java. [12] [13]

A principal finalidade do Arduino num sistema é facilitar a prototipagem, implementação ou emulação do controle de sistemas interativos, em nível doméstico, comercial ou móvel, da mesma forma que o CLP controla sistemas de funcionamento industriais. Com ele é possível enviar ou receber informações de basicamente qualquer sistema eletrônico, como identificar a aproximação de uma pessoa e variar a intensidade da luz do ambiente conforme a sua chegada, ou abrir as janelas de um escritório de acordo com a intensidade da luz do sol e temperatura ambiente. Os campos de atuação para o controle de sistemas são imensos, podendo ter aplicações na área de impressão 3D, robótica, engenharia de transportes, engenharia agronômica e musical. [12] [13]

Em resumo, o Arduino é um conjunto (KIT) de desenvolvimento que pode com a posse de informações realizar os processos computacionais e com o resultado atuar no controle ou no acionamento de algum elemento eletroeletrônico conectado ao seu terminal de saída. Por ser baseado em microcontrolador ele é portanto programável, sendo possível realizar diversas aplicações diferentes, e podendo reutilizar o mesmo equipamento, através de uma nova programação. Sua programação e facilitada pela existência de diversas funções que controlam o dispositivo, com uma sintaxe similar a linguagens de programação comumente utilizadas (C e C++).

Começou a se utilizar essa tecnologia para automação residencial pois além de estabelecer uma eficiência desejável, e de certa forma bem barata, é de fácil implementação, podendo ser implementada pelo próprio dono da casa com um certo conhecimento técnico; diferente do antigo método de automação residencial onde era utilizado um CLP que era, de certa forma, uma implementação muito mais complicada e que exigia mais conhecimentos técnicos.

3.2.1 ESPECIFICAÇÕES DO HARDWARE ARDUINO MEGA 2560

O Arduino Mega 2560 é uma placa de microcontrolador baseada no ATmega2560. Suas principais características são 54 pinos de entradas/saídas digitais, das quais 15 podem ser usadas como saídas PWM, 16 entradas analógicas, 4 UARTs (portas seriais de hardware), um oscilador de cristal de 16MHz, uma conexão USB, uma entrada de alimentação, uma conexão ICSP e um botão de reset. O Mega 2560 é compatível com a maioria dos Shields desenhados para os Arduinos Uno, Duemilanove e para o Diecimila. Possui ainda o dobro de memória do antigo Arduino Uno. [13]

A tabela 3.2 resume as principais características do Arduino Mega 2560.

Microcontroller	ATmega2560
Voltagem de Operação	5 V
Voltagem de Entrada	7-12 V
Limite da Voltagem de Entrada	6-20 V
Pinos Digitais de E/S (I/O)	54 (of which 14 provide PWM output)
Pinos Analógicos de Entrada	16
Valor da Corrente DC E/S (I/O)	40 mA
Valor da Corrente por Pino DC E/S (I/O)	50 mA
Capacidade da Memória Flash (Flash Memory)	256 kB of which 8 kB used by bootloader
Capacidade da Memória SRAM	8 kB
Capacidade da Memória EEPROM	4 kB
Velocidade do Clock	16 MHz
Material	FR4

Tabela 3.2: Características do Hardware do Arduino Utilizado [13]

O método utilizado de comunicação serial é TTL (lógica transistor-transistor). A comunicação serial em um nível TTL permanecerá sempre entre os limites de 0 V e Vcc, que é muitas vezes 5 V ou 3,3 V. Um valor lógico alto ("1"); é representado por Vcc, enquanto um lógico baixo ("0") é 0 V. [13]

Mega 2560 R3 também adiciona pinos SDA (Serial Data) e SCL (Serial Clock) ao lado da AREF (Analogue REFerence). [13]

Há dois novos pinos colocados perto do pino de RESET. Um deles é o que permite que o IOREF, se adapte à voltagem fornecida a partir da placa. O outro está reservado para usos futuros. O Arduino Mega pode ser alimentado pela conexão USB ou com uma fonte de alimentação externa. A fonte de alimentação é selecionada automaticamente. Fonte externa de energia pode vir com um adaptador AC para DC ou bateria. O adaptador pode ser conectado, conectando um plug 2,1 milímetros de centro-positivo na entrada de alimentação da placa. [13]

A placa pode operar com fornecimento externo de 6 a 20 volts. Se for fornecido com menos de 7 V, no entanto, o pino de 5 V pode fornecer menos de cinco volts e a placa pode ser instável. Se usar mais do que 12 V, o regulador de voltagem pode superaquecer e danificar a placa. O intervalo recomendado é de 7 a 12 volts. [13]

Cada um dos 54 pinos digital de Arduino Mega 2560 pode ser utilizado como uma entrada ou uma saída, usando funções pinMode (), digitalWrite () e digitalRead (). Eles operam com 5 volts. Cada pino pode fornecer ou receber um máximo de 40 mA e tem um resistor pull-up interno (desconectado por padrão) de 20-50 kΩ. Além disso, alguns pinos têm funções especializadas. [13]

IOREF: Este pino na placa Arduino fornece a referência de tensão com que o microcontrolador opera. É necessário configurar a proteção corretamente para ler a tensão pino IOREF e selecionar a fonte de alimentação adequada ou habilitar tradutores de tensão nas saídas para o trabalho com a 5 V ou 3,3 V. 3V3: A alimentação de 3,3 volts gerada pelo regulador. A corrente máxima é de 50 mA. [13]

LED: Há um LED conectado ao pino digital 13. Quando o pino tem valor ALTO, o LED está ligado, quando o pino tem valor BAIXO, ele está desligado. [13]

TWI: 20 (SDA) e 21 (SCL). Comunicação TWI apoio usando a biblioteca Wire. O protocolo I2C é implementado no Arduino, embora receba outro nome: TWI (Two-Wire Interface), por questões de direitos autorais. [13]

I²C (Inter-Integrated Circuit) é um barramento serial multi-mestre desenvolvido pela Philips que é usado para conectar periféricos de baixa velocidade a uma placa-mãe, a um sistema embarcado ou a um telefone celular.

O Mega2560 tem 16 entradas analógicas, cada uma das quais com 10 bits de resolução (isto é, 1024 valores diferentes). Por padrão elas medem até 5 volts, embora seja possível mudar o limite superior de sua faixa usando o pino AREF e a função analogReference (). [13]

Serial: 0 (RX) e 1 (TX); Serial 1: 19 (RX) e 18 (TX); Serial 2: 17 (RX) e 16 (TX); Serial 3: 15 (RX) e 14 (TX). Utilizado para receber (RX) e transmitir dados seriais (TX) TTL. Pinos 0 e 1 são também conectados aos pinos correspondentes do chip serial ATmega16U2 USB-to-TTL. [13]

Interrupções Externo: 2 (interromper 0), 3 (interrupção 1), 18 (interromper 5), 19 (interromper 4), 20 (interromper 3), e 21 (interrupção 2). Estes pinos podem ser configurados para disparar uma interrupção para o valor baixo, a borda de subida ou queda, ou a mudança de valor. Para mais detalhes sobre interrupções e a função attachInterrupt() descrita na referência bibliográfica. [13]

PWM: 2-13 e 44-46 Fornece saída PWM de 8-bit com a função analogWrite().

SPI: 50(MISO), 51(MOSI), 52(SCK), 53(SS). Estes pinos dão suporte à comunicação SPI (Serial Peripheral Interface) utilizando a biblioteca SPI. Os pinos SPI são divididos no cabeçalho ICSP também, que é fisicamente compatível com o Uno, Duemilanove e Diecimila.

AREF. Tensão de referência para as entradas analógicas. Usado com analogReference (). [13]

Reset. Linha BAIXO para resetar o microcontrolador. Tipicamente usado para adicionar um botão de reset. [13]

USB Proteção contra sobrecorrente: O Arduino Mega2560 tem um POLYFUSE reajustável que protege as portas USB do seu computador de curto e sobrecorrente. Embora a maioria dos computadores forneça sua própria proteção interna, o fusível fornece uma camada

extra de proteção. Se houver mais de 500 mA aplicados à porta USB, o fusível rompe automaticamente a ligação até o curto ou a sobrecarga ser retirada. [13]

Divisão das caraterísticas no Hardware

Arduino Uno: O primeiro mais conhecido. Figura 3.1 mostra essa divisão básica.

Figura 3.1: Arduino Uno [13]

Arduino Mega 2560: O segundo mais conhecido. Com mais entradas. Figura 3.2 mostra essa divisão básica e a Figura 3.3 mostra a pinagem do Arduino Mega 2560.

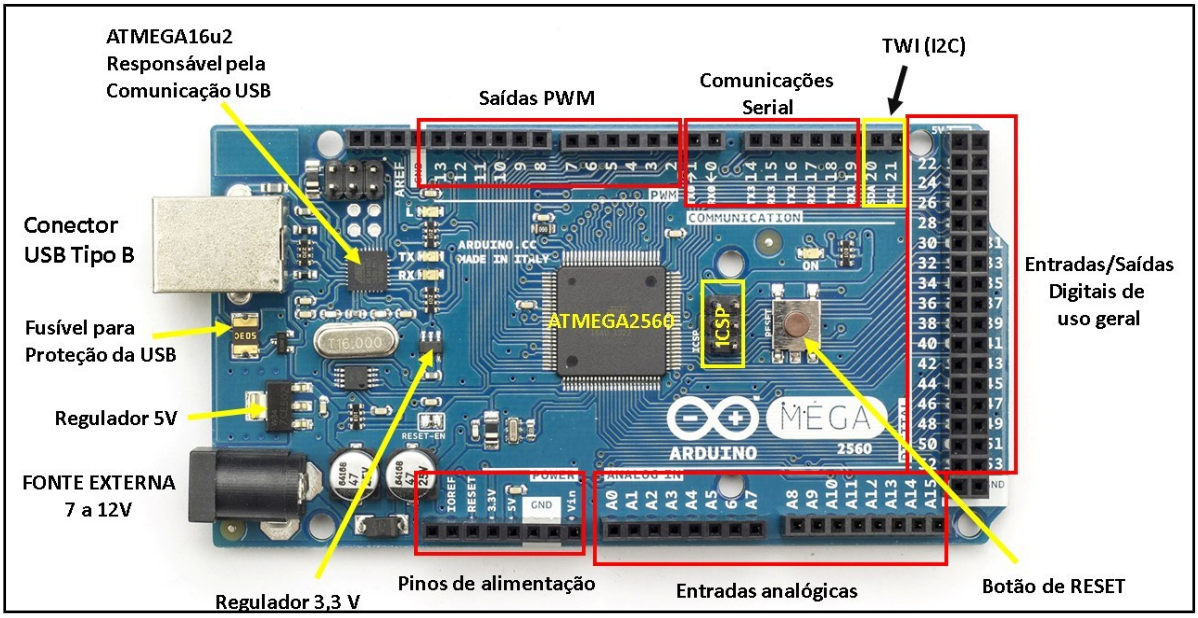


Figura 3.2: Arduino Mega 2560 [13]

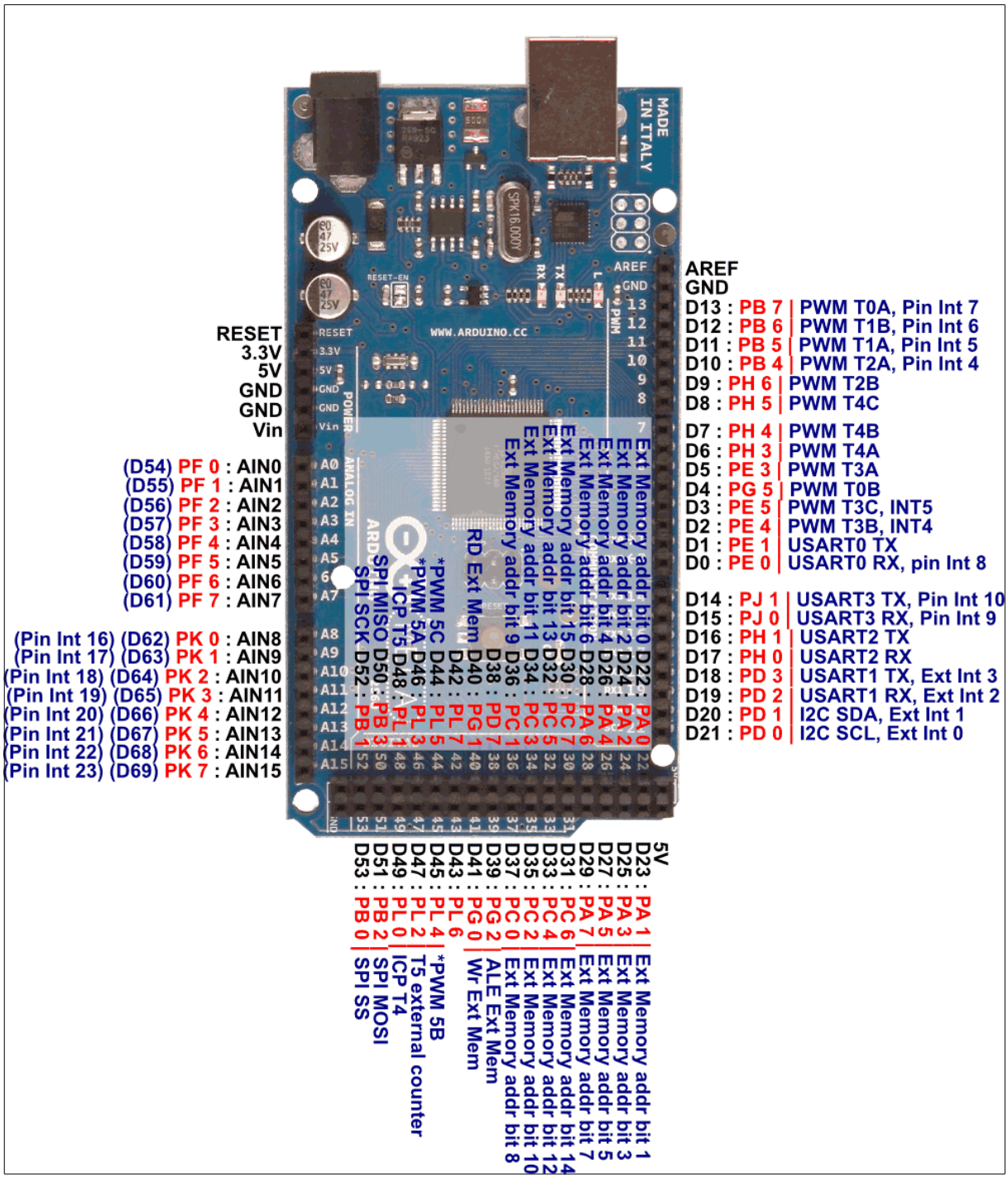


Figura 3.3: Pinagem do Arduino Mega 2560 [13]

Diagrama Lógico da Arquitetura do Hardware

A figura 3.4 resume as principais caraterísticas lógicas da sua arquitetura.

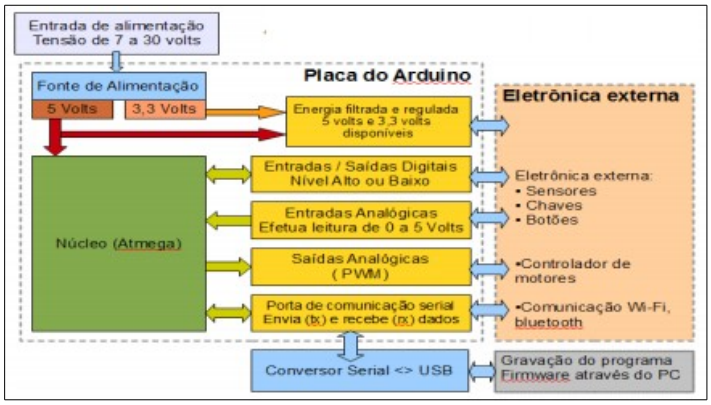


Figura 3.4: Diagrama Lógico da Arquitetura do Arduino

A tabela 3.3 mostra algumas vantagens do Arduino Mega em relação aos demais, por isso a escolha do mesmo para o projeto:

Nome	Arduino UNO R3	Sanguino	Arduino Mega
Microcontrolador	ATmega328	ATmega644p	ATmega2560
Velocidade	16 MHz	16 MHz	16 MHz
Memória Flash (Espaço para o código)	32 kB	64 kB	256 kB
Memória SRAM	2 kB	4 kB	8 kB
Memória EEPROM	1024 bytes	2048 bytes	4096 bytes
Quantidade de Pinos de Entrada e Saída E/S	14	32	70
Pinagens Digitais	14	24	54
Pinos PWM	6	6	14
Pinos Analógicos	6	8	16
Portas Seriais	1	2	4
I2C	SIM	SIM	SIM
SPI	SIM	SIM	SIM
JTAG	NÃO	SIM	NÃO
Voltagem de Operação	5 V	5 V	5 V
Voltagem de Entrada (Recomentada)	7-12 V	7-12 V	7-12 V
Voltagem de Entrada (Limites)	6-20 V	6-20 V	6-20 V
Corrente DC por pino de E/S	40 mA	40 mA	40 mA
Corrente DC por pino de 3.3 v	50 mA	50 mA	50 mA

Tabela 3.3: Comparação entre os Arduinos [13]

3.3 MÓDULO XBEE

Tecnologia baseada no padrão IEEE 802.15.4/ZigBee Alliance. Esse padrão foi desenvolvido para tornar mais simples a comunicação em redes sem fio. Devido à sua simplificação ele torna o seu custo de aquisição menor.

XBee é a marca da Digi International® Inc. (Digi®) International para uma família de módulos de rádio. Os primeiros rádios XBee foram introduzidos sob a marca MaxStream em 2005 e foram baseados no padrão 802.15.4/2003 projetado para ponto-a-ponto e comunicação estrela para taxas de transmissão “over-the-air” de 250 kb/s. Dois modelos foram introduzidos inicialmente, um custo menor de 1mW XBee e a de maior potência de 100 mW XBee-PRO. [15]

Os dispositivos operam numa faixa de frequência que não precisa de licença para o funcionamento; na faixa de 2,4 GHz em todos continentes, de 915 MHz na América no Norte e de 868 MHz na Europa, com taxas de transferência de dados de 250 kb/s, 40 kb/s e 20kb/s, respectivamente. [16]

O rádio XBee trabalha com 2 modos de operação de transmissão e recepção de dados. No primeiro modo, Transparent Operation ou AT, os dados são enviados e recebidos pela porta serial, utilizando uma interface simples e sendo mais fácil o desenvolvimento de aplicações, bastando a aplicação se conectar à porta serial do módulo e enviar os dados utilizando comandos AT.

O segundo modo, é o modo Application Programming Interface (API), baseado no envio e recepção de quadros de dados, especificando como comandos, respostas de comandos e mensagens sobre o estado de funcionamento do módulo são enviados e recebidos. Comandos AT também podem ser enviados e recebidos, permitindo a coexistência dos dois modos em rede.

O modelo do Módulo de XBee usado neste trabalho foi desenvolvido pela empresa MaxStream adquirida pela empresa Digi International® Inc. (Digi®). É possível encontrar o documento no site do fabricante Digi International® Inc. (Digi®). Também pode ser encontrado em outros sites que revendem esse módulo da Série 1 XB24-AWI-001, referencia bibliográfica. [16]

A figura 3.5 mostra a pinagem do XBee Modelo Série 2 que é quase igual da Série 1 usada no trabalho.

A tabela 3.5 especifica a função de cada porta.

A tabela 3.6 diz as características do módulo XBee comparando com XBee-PRO.

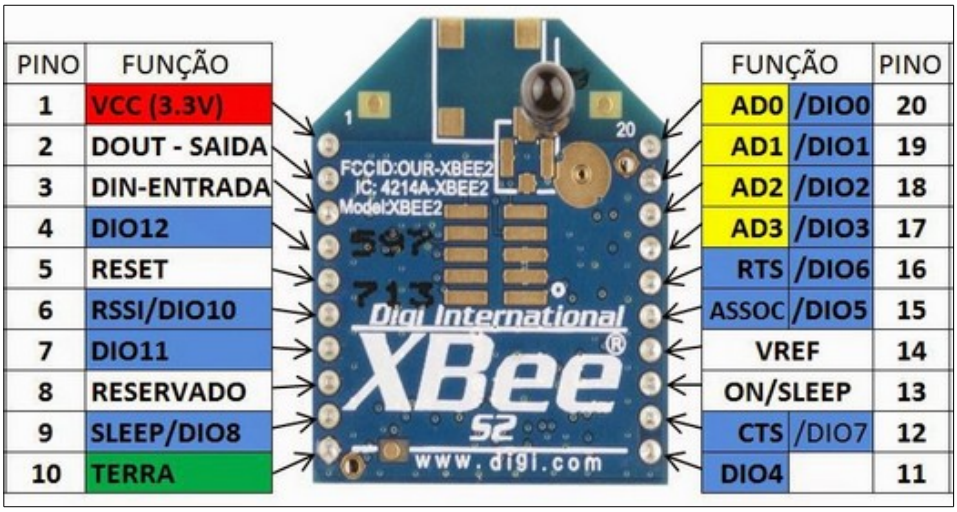


Figura 3.5: Pinagem do XBee Série 2

Pino	Nome	Direção	Descrição
1	VCC	-	Alimentação (3,3v)
2	DOUT	Saída	Saída de dados UART
3	DIN	Entrada	Entrada de dados UART
4	DO8	Saída	Saída Digital 8
5	RESET	Entrada	Reset do Módulo
6	PWM0/RSSI	Saída	Saída PWM0 ou indicador de força do sinal RSSI
7	PWM1	Saída	Saída PWM1
8	Reservado	-	Não Conectado
9	SLEEP/18	Entrada	Pino de controle do modo Sleep ou Entrada Digital 8
10	GND	-	Terra
11	AD4/DIO4	Ambos	Entrada analógica 4 ou E/S Digital 4
12	CTS/DIO	Ambos	Sinal de controle CTS ou E/S Digital 7
13	ON	Saída	Indicador de estado do módulo
14	VREF	Entrada	Voltagem de referência para as entradas A/D
15	AD5/DIO5	Ambos	Entrada analógica 5 ou E/S Digital 5
16	RTS/AD6/DIO6	Ambos	Sinal de controle RTS ou Entrada analógica 6 ou E/S Digital 6
17	AD3/DIO3	Ambos	Entrada analógica 3 ou E/S Digital 3
18	AD2/DIO2	Ambos	Entrada analógica 2 ou E/S Digital 2
19	AD1/DIO1	Ambos	Entrada analógica 1 ou E/S Digital 1
20	AD0/DIO0	Ambos	Entrada analógica 0 ou E/S Digital 0

Tabela 3.4: Descrição da Pinagem [16]

Plataforma	XBee	XBee-PRO
Desempenho		
Potência de Sáida	1mW (+0 dBm) North American & International	63 mW (+18 dBm) North American 10 mW (+10 dBm) International
Cobertura – Indoor/Urban	Até 100 ft (30 m)	Até 300 ft (90 m)
Cobertura – Outdoor/RF	Até 300 ft (90 m)	Até 1 mile (1.6 km) RF LOS
Sensibilidade do receptor	-92 dBm	-100 dBm (all variants)
Taxa de dados RF	250 Kbps	250 Kbps
Frequência de Operação	2.4 GHz	2.4 GHz
Taxa de dados na interface	Até 115.2 Kbps	Até 115.2 Kbps
Característica da Rede		
Método de Acesso – Spread spectrum type	DSSS (Direct Sequence Spread Spectrum)	
Topologias de Rede	Ponto ou Malha, Estrela, Arvore (Point-to-point, point-to-multipoint, & peer-to-peer)	
Tratamento de Erros	Tentativas e Confirmações (Retries & acknowledgements_)	
Opções de Filtragem	PAN ID, Channel, and 64-bit addresses	
Quantidades dos canais	16 Channels	12 Channels
Endereçamento	65,000 endereços disponível na rede por canal	
Potencia		
Tensão de Alimentação	2.8 - 3.4 VDC Recomendação: 3.0 – 3.4 VDC	2.8 - 3.4 VDC Recomendação: 3.0 - 3.4 VDC
Corrente de transmissão	45 mA (@ 3.3 V) boost mode ; 35 mA (@ 3.3 V) normal mode	215 mA (@ 3.3 V)
Corrente de Recepção	50 mA (@ 3.3 V)	55 mA (@ 3.3 V)
Corrente no modo Sleep	<10 µA até 25° C	<10 µA até 25° C
Informações Gerais		
Banda da frequência básica	2.4000 - 2.4835 GHz	
Certificações	FCC/IC – Europe/ETSI – Australia/Ctick – RoHS(Compliant)	
Opções de interface	3V CMOS UART	
Propriedades Físicas		
Tamanho	0.960 in x 1.087 in (2.438 cm x 2.761 cm)	0.960 in x 1.297 in (2.438 cm x 3.294 cm)
Peso	0.10 oz (3g)	
Opções das antenas	U.FL, Reverse Polarity SMA (RPSMA), Chip Antenna or Wired Whip Antenna	
Temperatura de operação	-40° C to 85° C (industrial)	

Tabela 3.5: Características Gerais do XBee e XBee-PRO [16]

A figura 3.6 mostra a diversidades de tipos de antenas e modelos existente no mercado.

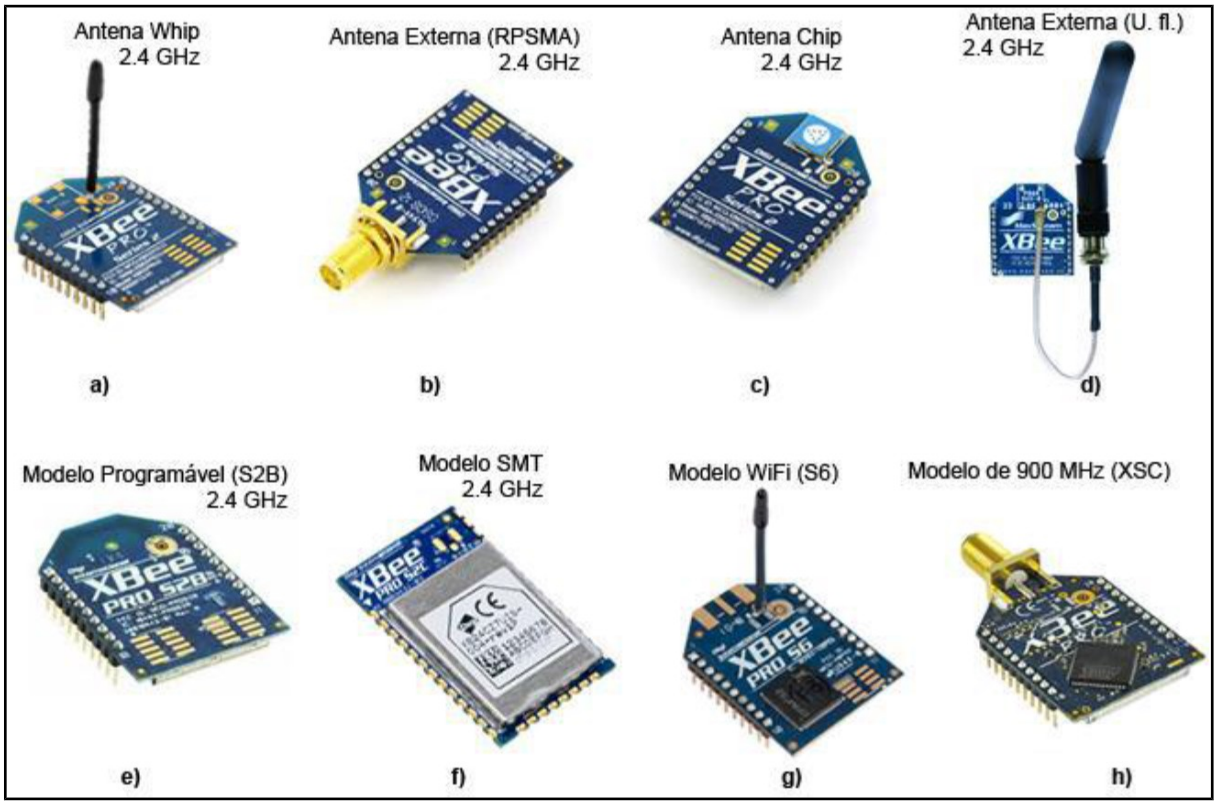


Figura 3.6: Tipos de Antenas – Referência ao trabalho [17]

A figura 3.7 mostra o produto usado no trabalho:

Figura 3.7: Módulo XBee e o Conector USB/Serial – Fotos tirado pelo Autor.

Já homologado na ANATEL. [28]

A tabela 3.6 mostra as características básica que o XBee deve obedecer.

Características técnicas básicas:					
Faixa de Frequências Tx (MHz)	Potência Máxima de Saída (W)	Designação de Emissões	Tecnologia	Tipo de Modulação	Taxa de Transmissão
2400,0 a 2483,5	0,00125	1M56X9D	Sequência Direta	O-QPSK	Até 230,4k bit/s

Tabela 3.6: Características Básicas do XBee S1 pela ANATEL. [28]

3.4 MÓDULO GSM

O SIM900 é um módulo GSM/GPRS quadband com pilha TCP/IP embutida e é do tipo board-to-board (fácil soldagem e integração, até mesmo para prototipagem) e pode ser usado em aplicações onde a transmissão se dê via tecnologia GSM/GPRS, seja voz, SMS, ou dados. Consome pouca energia, corroborando ao tema do trabalho. Seu pequeno tamanho (24x24 mm) o torna ideal para os mais exigentes requisitos das aplicações industriais, como M2M (Machine-to-Machine), telemetria ou qualquer outra forma de comunicação móvel.

O módulo SIM900 é fabricado pela SIMCOM (SHANGHAI SIMCOM LTD.) e distribuído no Brasil pela ME Componentes.

Principais Características

- Banda: Quad-Band GSM/GPRS 850/900/1800/1900 MHz
- Encapsulamento: SMT (surfaced mount technology).
- TCP/IP: Pilha TCP/IP já embutida.
- Antena: Externa.
- Aplicações: Rastreadores Veiculares, Rastreadores Pessoais, M2M, entre outras.

O SIM900 é um completo módulo GSM e por isso, requer um chip (cartão) GSM para que possa transmitir/receber voz, dados e SMS.

Já homologado na ANATEL. [21]

A tabela 3.7 mostra as características básica que o SIM900 deve obedecer.

Características técnicas básicas:			
Faixa de Frequências Tx (MHz)	Potência Máxima de Saída (W)	Designação de Emissões	Tecnologia
824,0 a 849,0	1,9054	200KG7W	GSM/GPRS/EDGE
898,5 a 901,0	1,8967	200KG7W	GSM/GPRS/EDGE
907,5 a 915,0	1,8967	200KG7W	GSM/GPRS/EDGE
1710,0 a 1785,0	0,8472	200KG7W	GSM/GPRS/EDGE
1895,0 a 1900,0	0,8433	200KG7W	GSM/GPRS/EDGE

Tabela 3.7: Características Básicas do SIM900 [21]

O Módulo SIM900 segue o padrão 3GPP TS 05.07. A tabela 3.8 mostra as características básica que o SIM900 deve obedecer.

Frequência	Típica Sensibilidade na Recepção	Máxima Sensibilidade na Recepção	Recepção	Transmissão
GSM850	-109 dBm	-107 dBm	869 ~ 894 MHz	824 ~ 849 MHz
EGSM900	-109 dBm	-107 dBm	925 ~ 960 MHz	880 ~ 915 MHz
DCS1800	-109 dBm	-107 dBm	1805 ~ 1880 MHz	1710 ~ 1785 MHz
PCS1900	-109 dBm	-107 dBm	1930 ~ 1990 MHz	1850 ~ 1910 MHz

Tabela 3.8: 3GPP TS 05.07 [19]

No trabalho foi utilizado o Shield produzido pela Elec Freaks, como mostra a figura 3.8. [20]

Figura 3.8: Shield do Módulo SIM900 para Arduino. [20]

A tabela 3.9 mostra as características Básicas do SIM900. [19]

Ficha Técnica	
GPRS multi-slot	Class 10 (Default) 8 (option)
GPRS mobile station	Class B
Controle	Via comandos AT (GSM 07.07, 07.05, mais os comandos AT da SIMCom)
Potência na transmissão	Class 4 (2W) at GSM 850 and EGSM 900. Class1 (1W) at DCS 1800 and PCS 1900
Nível	LVTTL
Tensão de Entrada	3.2V ~ 4.8V
Medidas	24mm x 24mm x 3mm
Pinagem	68 pinos
Economia da Potencia	Tipico consumo de potencia na modo sleep é 1mA (BS-PA-MFRMS=9)
SIM application toolkit	GSM 11.14 Release 99
Real Time Clock	Support RTC
Banda das Frequências	SIM900 Quad-band: GSM 850, EGSM 900, DCS 1800, PCS 1900. SIM 900 pode procurar as 4 bandas de frequências automaticamente. As bandas de frequência também pode ser definida pelo comando AT "AT+CBAND".
SIM interface	Support SIM card: 1.8v , 3V
Antena Externa	Antenna Pad
Temperatura de Operação	Normal Operation: -30^{0}C ~ +80^{0}C.
Data GPRS	GPRS data Downlink transfer: Max 85.6 kbps. GPRS data Uplink transfer: Max 42.8 kbps. Coding Scheme: CS-1, CS-2, CS-3 and CS-4
SMS	MT, MP, CB, Text and PDU Mode. SMS Storage: SIM card
Serial port	Full moden interface with status and control lines, unbalanced, asynchronous. 1200bps to 115200bps. Can be user for AT commands or data stream. Support RTS;CTS hardware handshale and software ON/OF flow control. Multiplex ability accordin to GSM 07.10 Mult

Tabela 3.9: Características Básicas do SIM900 [19]

A figura 3.9 mostra o diagrama lógico do hardware do Shield SIM900 Functional Diagram. E a figura 3.10 mostra a pinagem do SIM900.

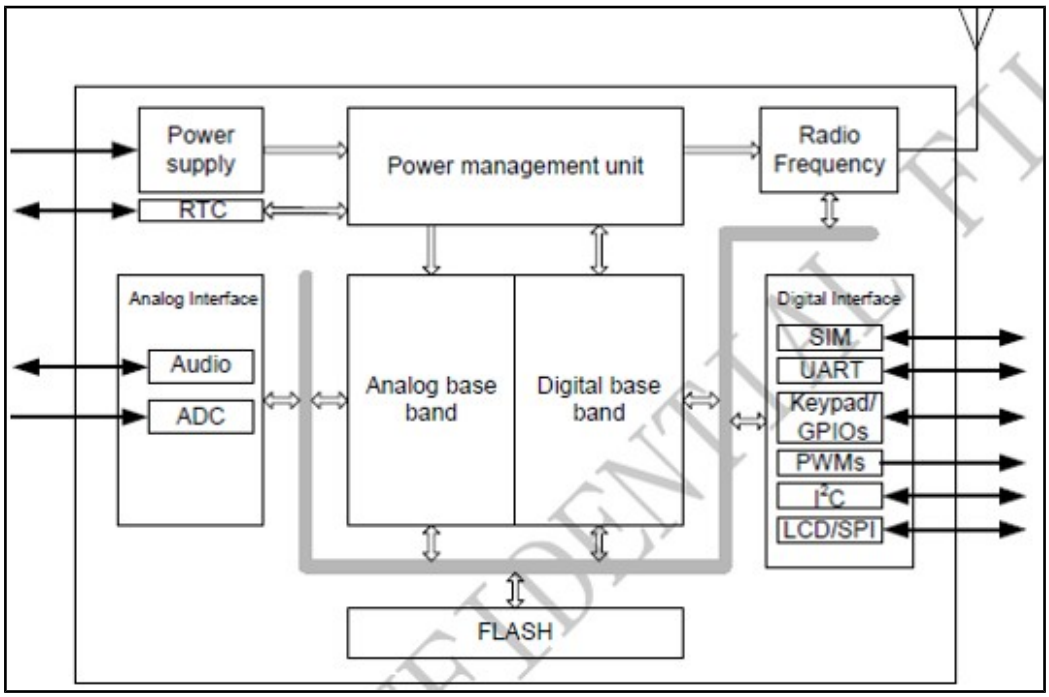


Figura 3.9: Diagrama lógico do Hardware do Shield SIM900 [19]

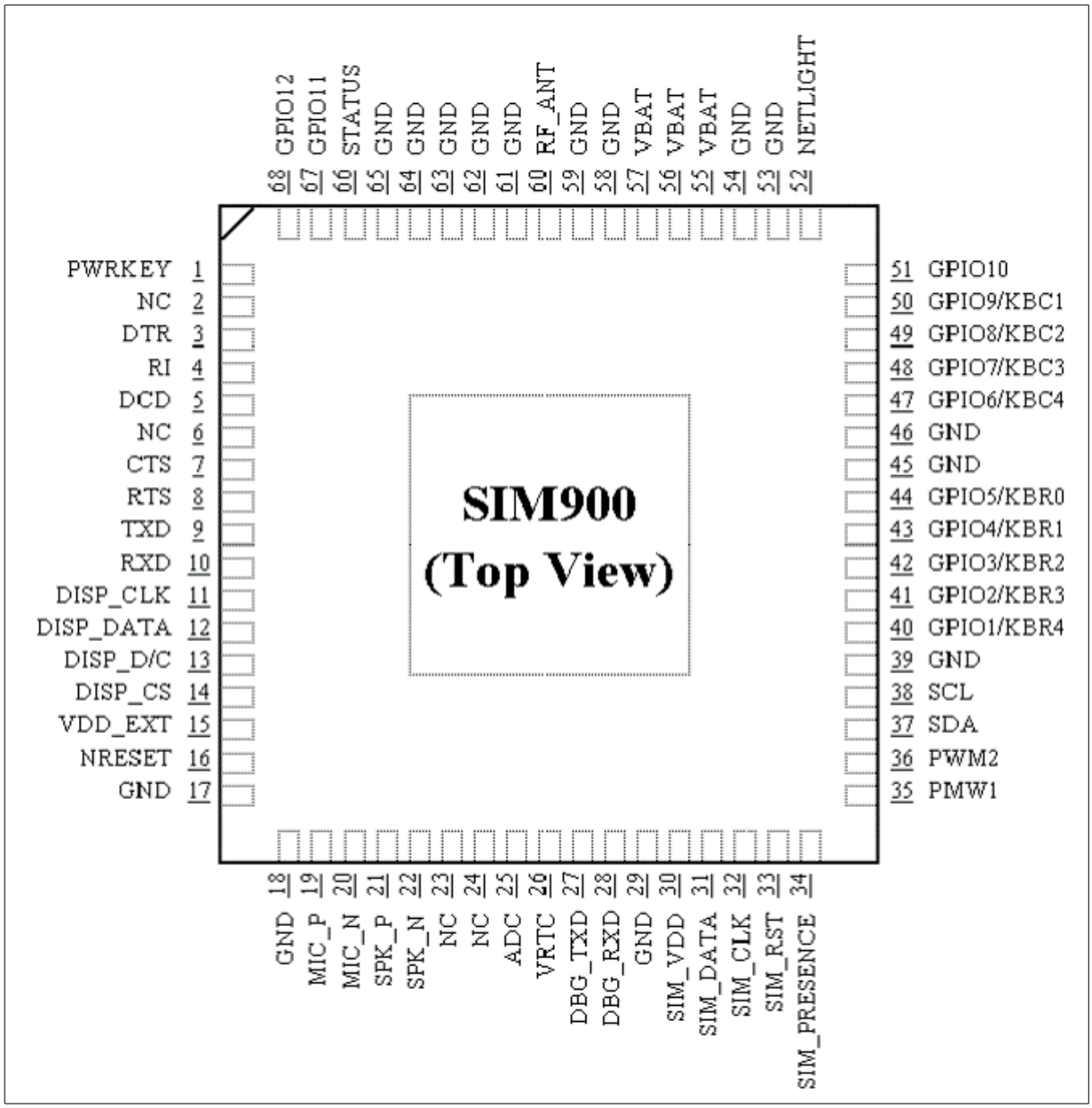


Figura 3.10: Pinagem do SIM900 [19]

A figura 3.11 mostra a pinagem do Shield SIM900, que estabelecerá a comunicação entre o SIM900 e o Arduino.

Na tabela 3.10 é descrito, somente a pinagem utilizada no trabalho, o restante se encontra no documento fornecido pelo fabricante do Shield ElecFreaks. [20]

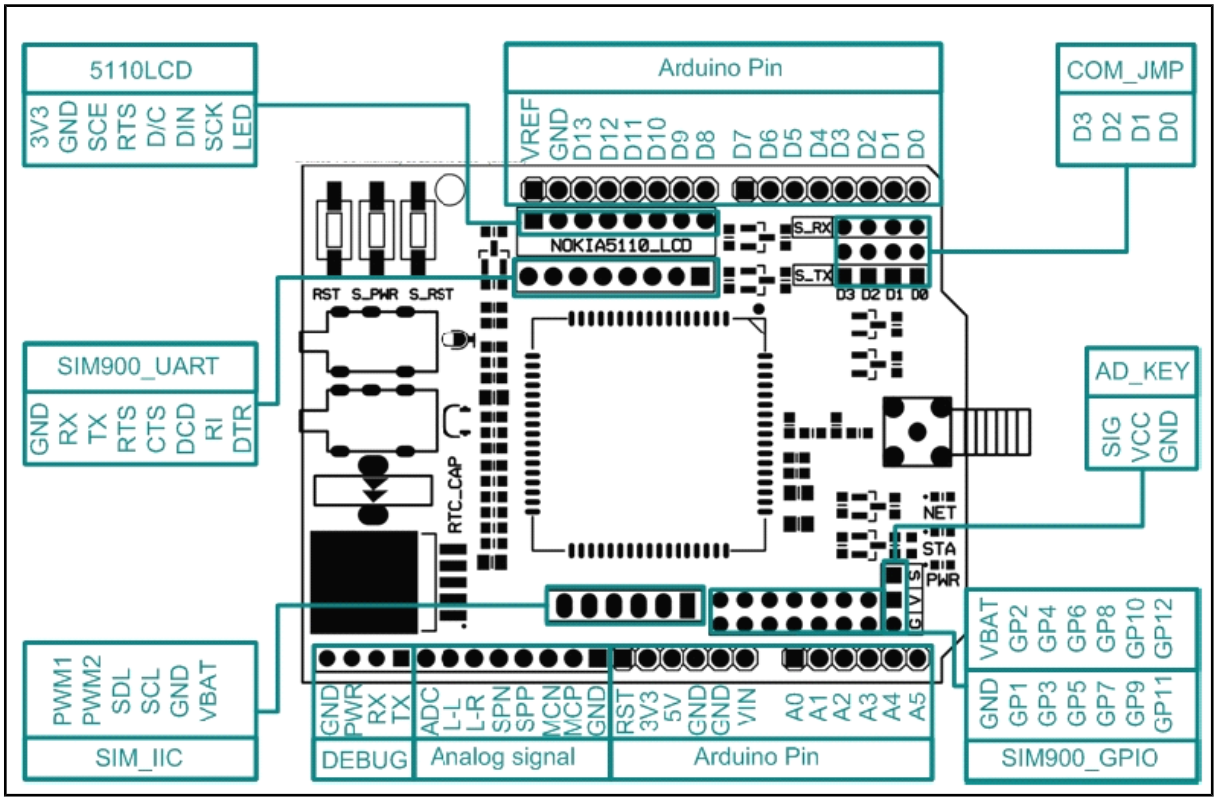


Figura 3.11: Pinagem do Shield SIM900 para o Arduino [20]

Tipo	Simbolo	Descrição
Pinagem entre Arduino e o SIM900	D0	Pino a comunicação, pode ser selecionado para RX ou TX
	D1	Pino a comunicação, pode ser selecionado para RX ou TX
	D2	Pino a comunicação, pode ser selecionado para RX ou TX
	D3	Pino a comunicação, pode ser selecionado para RX ou TX
	D4	Conecta o SIM900 UART Bus RI
	D5	Pino para controlar o Reset SIM900
	D6	Pino para controlar o Power on/off SIM900
DEBUG	GND	Pino para controlar o Power Ground
	PWR	Pino para controlar o Power Supply
	RX	SIM900 Debug Port Recevie. For debugging and uploading firmware
	TX	SIM900 Debug Port Transmit. For debugging and uploading firmware

Tabela 3.10: Descrição das Pinagens [20]

4 MONTAGEM FÍSICA DO PROTÓTIPO

O trabalho foi moldado pela montagem física e a programação para satisfazer a motivação descrita na introdução. Foram feitas adaptações para somente receber SMS transmitido por um dispositivo móvel (celular), que utiliza um SIM Card GSM fornecido por uma operadora de telefonia móvel do Estado do Rio de Janeiro. Esse SMS é recebido pelo Shield SIM900 conectado ao Arduino que processará a mensagem acionando o XBee, também conectado ao Arduino. O XBee acionará o relé, ligando ou desligando a carga (lâmpada).

Todos os diagramas de conexões do circuito encontram-se no apêndice I. Para desenhar, foi utilizado o software TinyCad [32].

O livro Robert Faludi [18], foi motivador e importante na montagem física do protótipo.

4.1 DIAGRAMA DE CONEXÃO ENTRE OS COMPONENTES DO PROJETO

Nesta seção será apresentado o desenho do circuito que vai receber o SMS, processar a mensagem pelo Arduino e acionar o XBee, que por sua vez acionará o relé ligado à lâmpada. Esses dois circuitos, que funcionam por encaixes pelos jumpers, serão chamados de Circuito RX-TX-RX Mega2560-SIM900-XBee e Circuito RX-TX-RX XBee-Relé-Lâmpada, respectivamente.

4.1.1 DESENHO DO CIRCUITO RX-TX-RX MEGA2560-SIM900-XBEE

Figura 4.1 mostra o diagrama do circuíto RX-TX-RX Mega2560-SIM900-XBee.

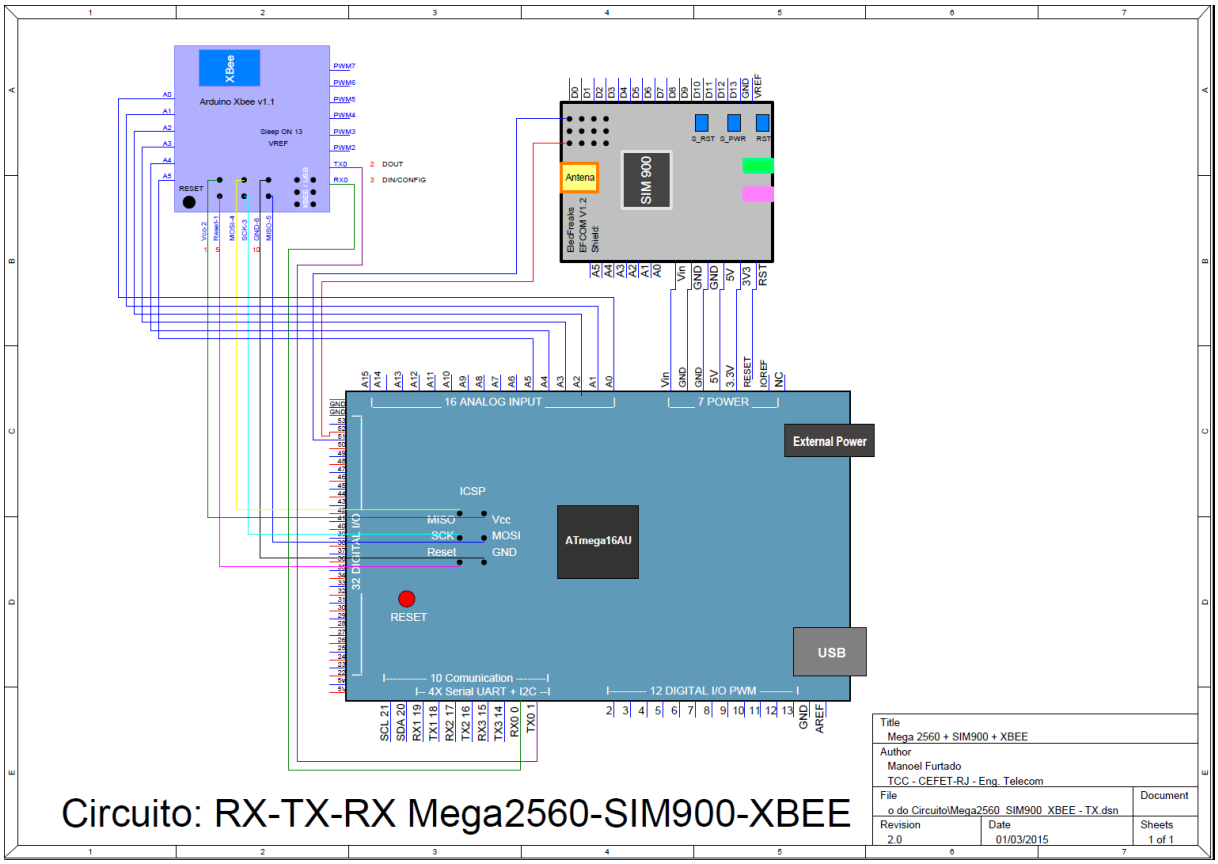


Figura 4.1: Desenho do Circuito de RX-TX-RX Mega2560-SIM900-XBee

As figuras 4.2.1 e 4.2.2 mostram a montagem do circuito de RX-TX-RX Mega2560-SIM900-XBee.

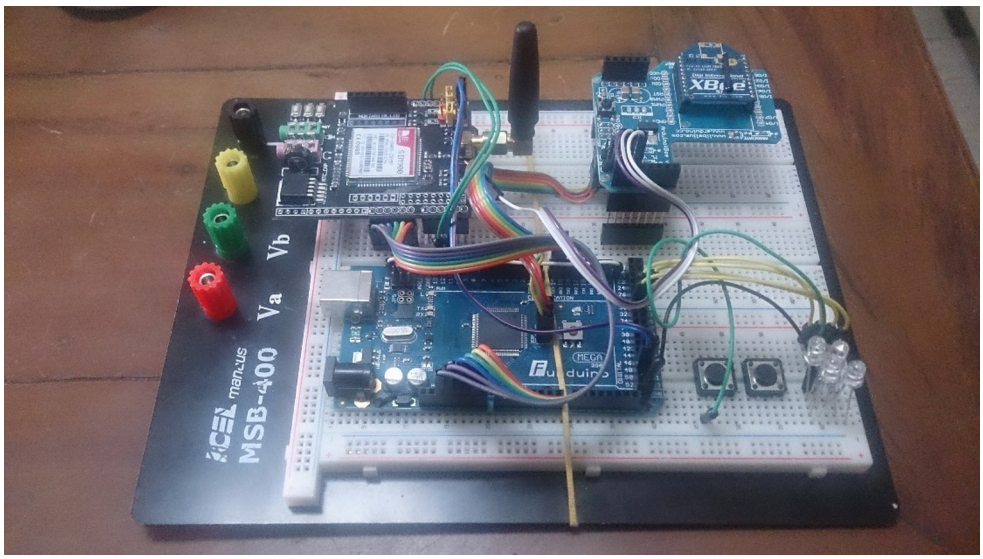


Figura 4.2.1: Foto do Circuito de RX-TX-RX Mega2560-SIM900-XBee

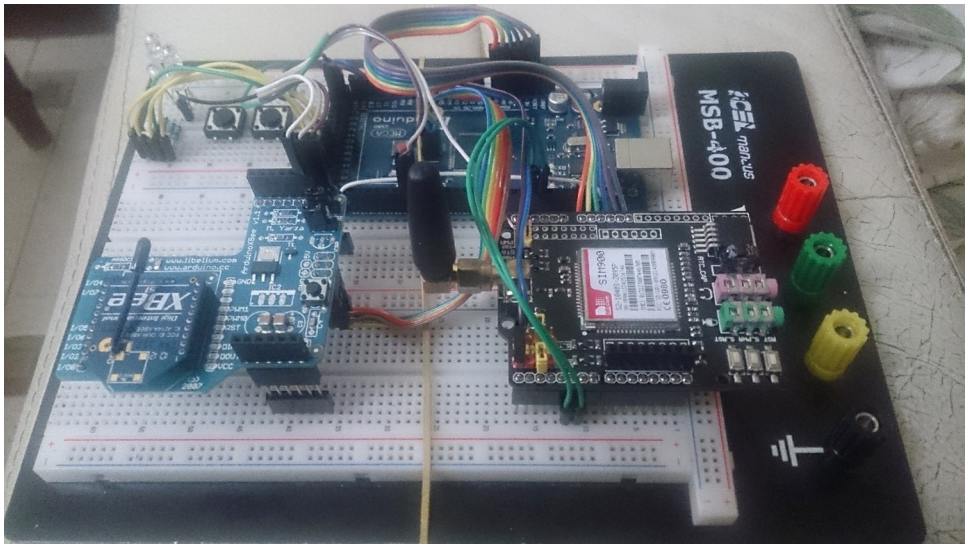


Figura 4.2.2: Foto do Circuito de RX-TX-RX Mega2560-SIM900-XBee

4.1.2 DESENHO DO CIRCUITO RX-TX-RX XBEE-RELÉ-LÂMPADA

Figura 4.3 mostra a montagem do circuito RX-TX-RX XBee-Relé-Lâmpada.

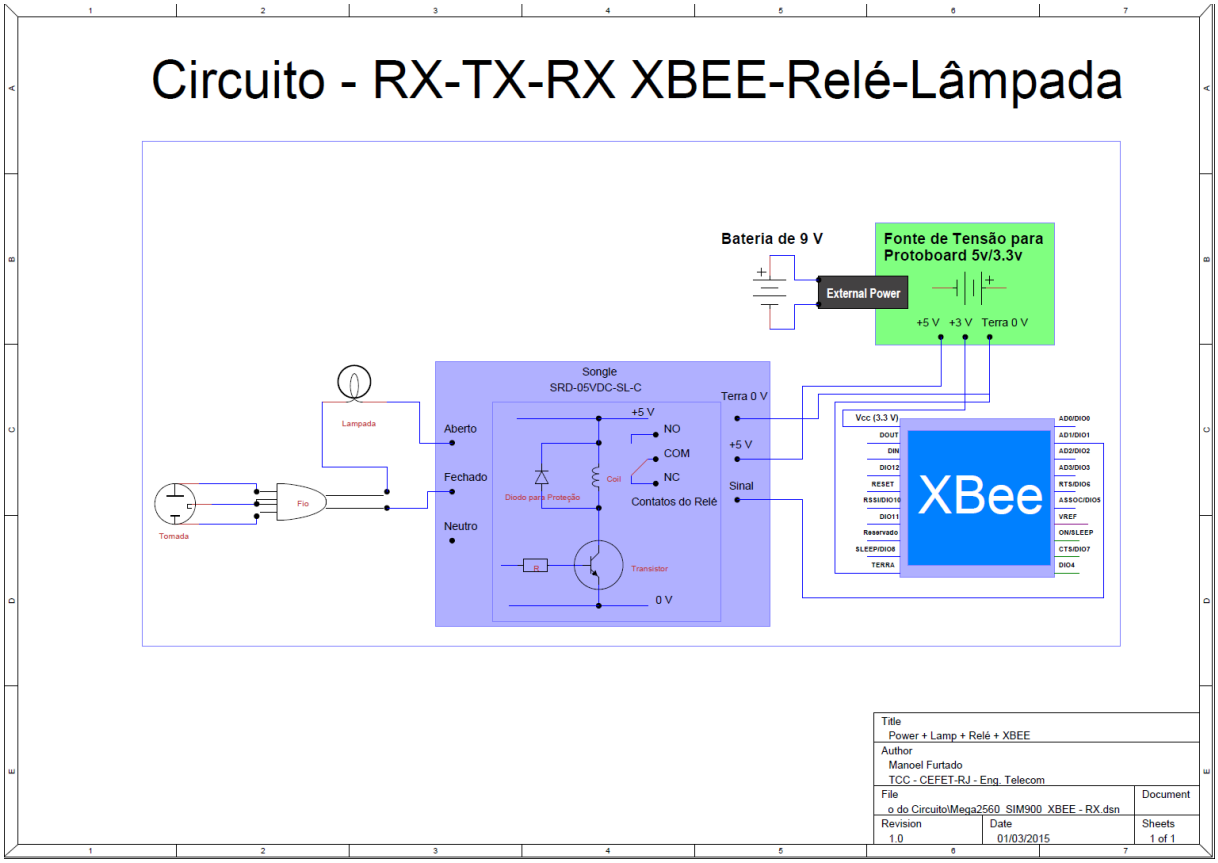


Figura 4.3: Desenho do Circuito de RX-TX-RX XBee-Relé-Lâmpada

As figuras 4.3.1, 4.3.2, 4.3.3, 4.3.4 mostram a montagem do circuito de RX-TX-RX XBee-Relé-Lâmpada.

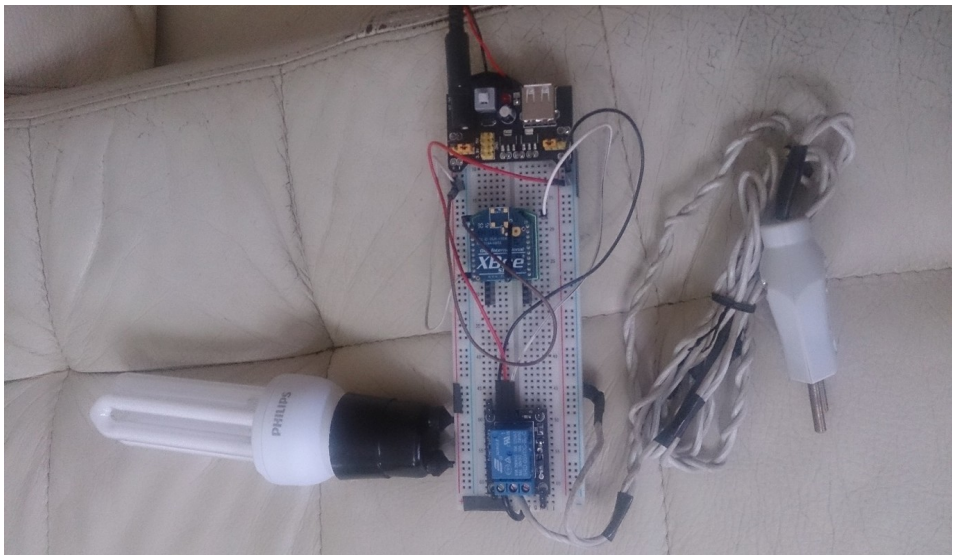


Figura 4.3.1: Circuito de RX-TX-RX XBee-Relé-Lâmpada

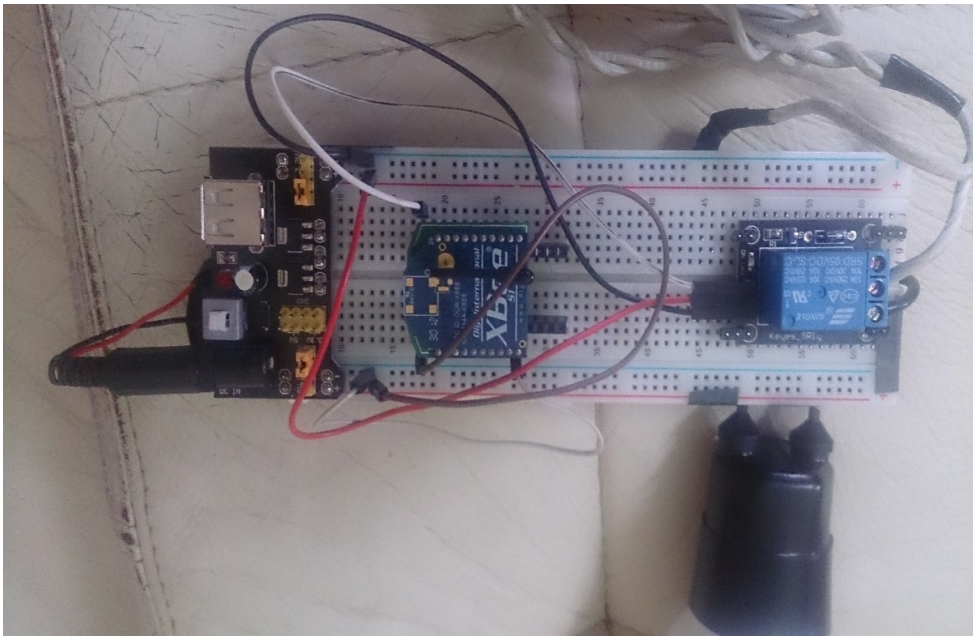


Figura 4.3.2: Circuito de RX-TX-RX XBee-Relé-Lâmpada

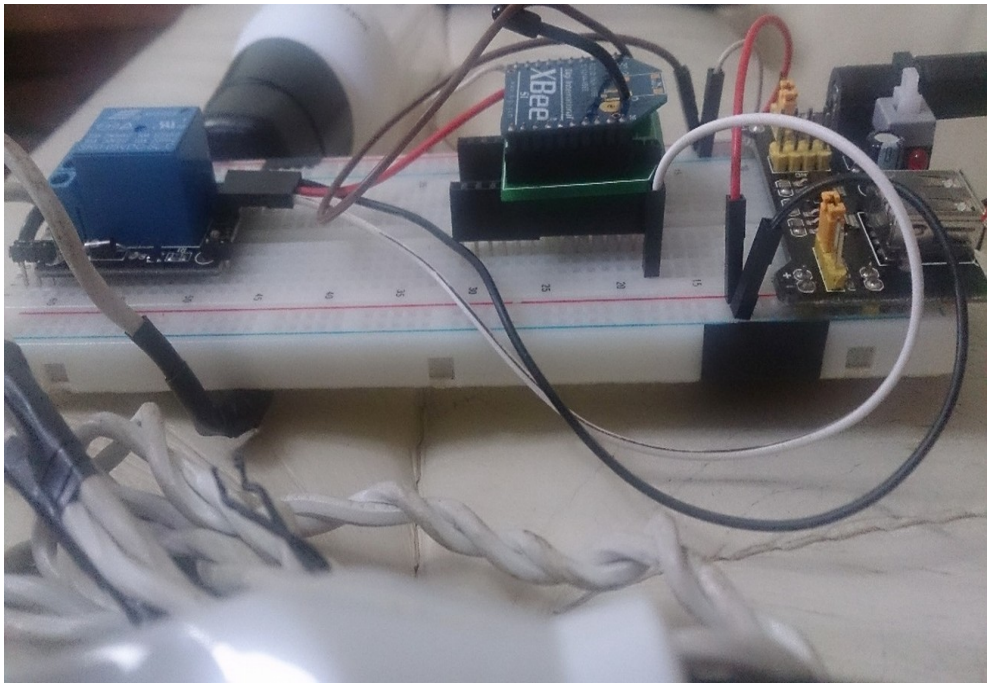


Figura 4.3.3: Circuito de RX-TX-RX XBee-Relé-Lâmpada

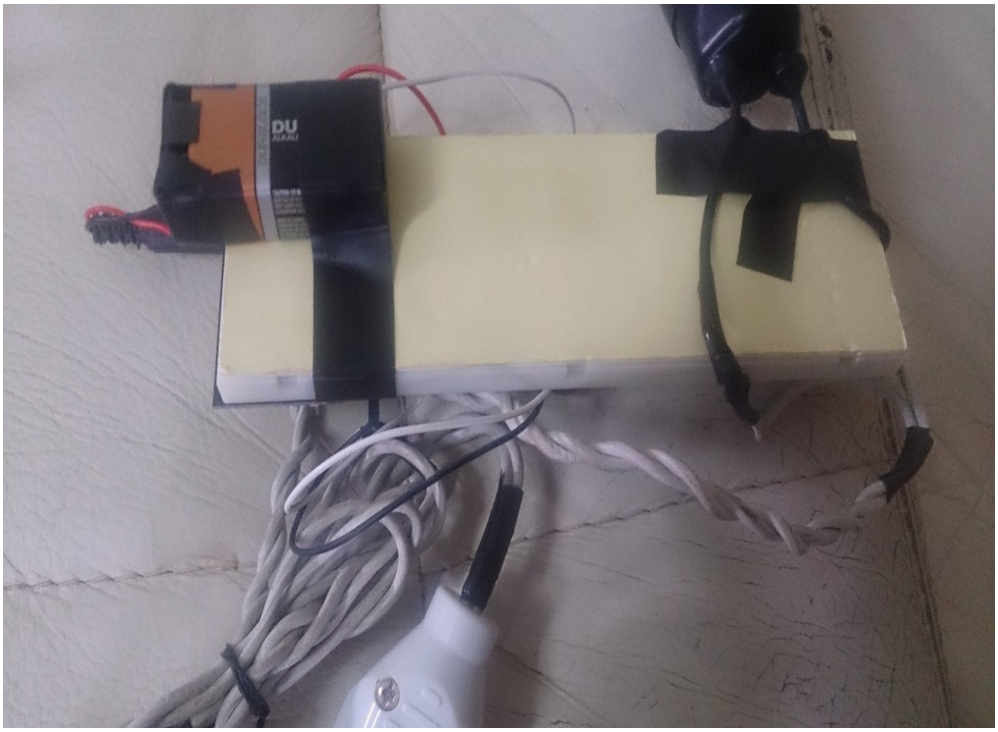


Figura 4.3.4: Circuito de RX-TX-RX XBee-Relé-Lâmpada

5 BIBLIOTECAS PARA ARDUINO

5.1 ARDUINO E SUAS BIBLIOTECAS

O hardware do Arduino com o seu software IDE proporciona o interfaceamento em dois níveis, hardware e software. Esse software pode ser estendo através das bibliotecas, assim como a maioria das plataformas de programação, fornece funcionalidades extras no sketches desenvolvidos. [34]

No nível de hardware permite conectar componentes através de módulos(shields) que contem algum hardware com alguma tecnologia como WiFi, Bluetooh, Ethernet, diversos tipos de sensores. [34]

No nível de software, as bibliotecas do Arduino permite que qualquer outra linguagem de programação com capacidade de se comunicar em série nativamente, para não precisar de intermediário, exemplos Python, Processing, C/C++, Linux TTY etc. [34]

Atualmente, da data de publicação desse trabalho, os plugins Cordova para Android são um ótimo exemplo da integração das bibliotecas. Cria toda uma ponte para proporcionar a interoperabilidade das tecnologias atuais, com os dispositivos móveis (smartphones). Referência dos plugins [35], a um esforço muito grande, mais de mil pessoas trabalham nesses plugins. Essa interoperabilidade é caracterizada desde a comunicação serial até esses aplicativos híbridos para os dispositivos móveis, através desses plugins.

PhoneGap é um framework de desenvolvimento de aplicativos móveis criado por Nitobi. Adobe Systems adquiriu Nitobi em 2011. Ele permite que os programadores de software para construir aplicações para dispositivos móveis usando JavaScript, HTML5, CSS3 e, em vez de depender de APIs específicas de plataforma como aqueles em iOS, Windows Phone ou Android. Ele permite que envolve acima de HTML, CSS, Javascript e código dependendo da plataforma do dispositivo. Os aplicativos resultantes são híbridas, porque toda a renderização de layout é feito através de vistas da web em vez interface do usuário(UI) pelo framework nativa da plataforma e porque eles não são apenas aplicações web, mas são embalados com base na Web como aplicativos para distribuição e ter acesso a APIs nativas do dispositivo. Adobe System e Nitobi doaram o código para Advanced Systems

Format(ASF) do PhoneGap se chamando então "Apache Callback" e posteriormente surge o Apache Cordova. Como o software open-source, Apache Cordova permite embalar(wrappers) sem o Adobe, como Intel XDK ou Appery.io. O Apache Cordova é uma comunidade ligada ao PhoneGap, que é a versão produzida pela Adobe em cima do Cordova. O Ionic open source SDK usa Cordova não PhoneGap para suas ferramentas principais. A Intel XDK também usa APIs Cordova. [36]

O software fornecido pelo Arduino para desenvolver a programação contém um editor de texto para escrever o código que utiliza um compilador GCC (que é baseado na arquitetura de C e C++) o qual contém basicamente uma interface gráfica construída em Java, uma área de programação e uma barra de ferramentas, cujas funções são enumeradas na figura 5.1. Ele é responsável por realizar o upload do programa no hardware do Arduino.

O ambiente de desenvolvimento integrado Arduino (IDE) é uma aplicação multiplataforma escrita em Java, derivada dos projetos Processing e Wiring. É incluso uma biblioteca de software chamado de "written" herdado do projeto original Wiring. Ele é projetado para apresentar a programação para artistas e outros recém-chegados que não estão familiarizados com desenvolvimento de software. Inclui um editor de código com recursos de realce de sintaxe, parênteses correspondentes e identificação automática, sendo capaz de compilar e carregar programas para a placa com um único clique. Um programa ou código escrito para Arduino é chamado de "Sketch".

Processing é uma linguagem de programação de código aberto e ambiente de desenvolvimento integrado (IDE), construído para as artes eletrônicas e comunidades de projetos visuais com o objetivo de ensinar noções básicas de programação de computador em um contexto visual e para servir como base para cadernos eletrônicos. O projeto foi iniciado em 2001 por Casey Reas e Ben Fry, ambos ex-membros do Grupo de Computação do MIT Media Lab. Um dos objetivos do Processing é atuar como uma ferramenta para não-programadores iniciados com a programação, através da satisfação imediata com um retorno visual.1 A linguagem tem por base as capacidades gráficas da linguagem de programação Java, simplificando características e criar alguns novos. [22] [23]

Wiring é uma plataforma de prototipagem eletrônica de hardware livre composta por uma linguagem de programação, um ambiente de desenvolvimento integrado (IDE) e um microcontrolador de placa única. O projeto foi iniciado em 2003 por Hernando Barragán

através do Interaction Design Institute Ivrea, Itália. Atualmente se desenvolve na Escola de Arquitetura e Design da Universidade de Los Andes, em Bogotá, Colômbia. Construído sobre o Processing, um projeto aberto de Casey Reas e Benjamin Fry, sua linguagem foi desenvolvida com a ajuda do Grupo de Computação e Estética da MIT Media Lab.1. [24] [25]

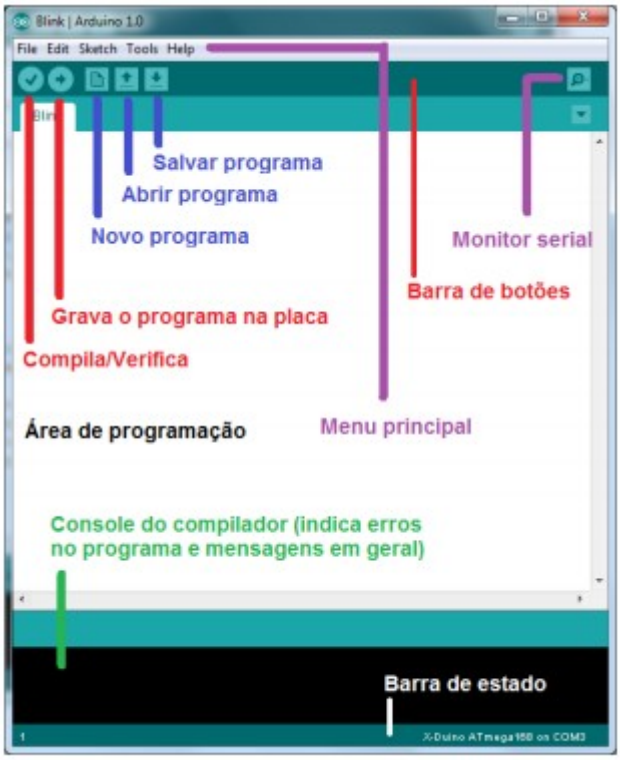


Figura 5.1: Detalhe das funções da janela principal do Software de compilação (versão 1.0 estável)

Como a base da programação do Arduino é o C e o C++, foi preservada sua sintaxe de declaração de variáveis, na utilização de operadores, na manipulação de vetores, na conservação de estruturas e é uma linguagem sensível a letras minúsculas e maiúsculas chamado de case-sensitive. Entretanto, existe uma mudança: em vez da função main ( ), o Arduino necessita de duas funções elementares: setup ( ) e loop ( ); sem a existência dessas o programa não funcionará (mesmo que não se escreva nada no interior delas, é necessário estarem presentes na sua programação).

Setup( )

A função setup é utilizada para iniciar variáveis, configurar o modo dos pinos e incluir bibliotecas. Esta função é executada uma única vez, assim que o Arduino é ligado ou quando o botão reset e pressionado. Sintaxe: void setup( ) {}

Loop ( )

A função loop é a que entra em looping (executa sempre o mesmo bloco de código), permitindo ao seu programa executar as operações que estão dentro desta função. Essa função deve sempre ser declarada após a função setup(). Sintaxe: void loop ( ) {}

Funções básicas de entrada e saída digital

Pinmode ( ) – Configura o pino definido para que se comporte como entrada ou saída. Deve-se informar o número do pino que deseja e em seguida a função que deseja que assuma; entrada (INPUT) ou saída (OUTPUT).

Sintaxe: pinMode (pino, modo);

digitalWrite ( ) – Escreve um valor 1 ou 0 no pino Digi International® Inc. (Digi®)tal. Se o pino for configurado como saída, sua voltagem será 5 V para nível alto e 0 V para nível baixo.

Sintaxe: digitalWrite(pino, valor);

digiRead( ) – Lê o valor de um pino digital especificado

Sintaxe: Int digitalRead(pino);

A figura 5.2 mostra o exemplo mais simples, que tem a função de acender e apagar um LED repetidamente no intervalo de 1 seg.

Para melhor compreensão, eis outro exemplo, desta vez utilizando um botão e 3 variáveis do tipo int:

```
/* Exemplo de função sobre de Entrada e Saída digital */

int ledPin = 13; // LED conectado ao pino digital 13

int bot = 7; // botão conectado ao pino digital 7

int val = 0; // variável para armazenar o valor lido
```

```
void setup()
{
        pinMode(ledPin, OUTPUT); // seta o pino digital 13 como uma saída
        pinMode(bot, INPUT); // seta o pino digital 7 como uma entrada
}
void loop()
{
        val = digitalRead(bot); // lê o pino de entrada
        digitalWrite(ledPin, val); // acende o LED de acordo com o botão 7
}
```

Esse programa (Sketch) transfere para o pino 13 o valor lido no pino 7, que foi definido como uma entrada. Como mencionado na seção 3.2.1, algumas portas digitais podem ser configuradas como entrada ou saída; quem define é a função pinMode( ), nesse exemplo.

Nesses dois exemplos simples não foi necessário declarar o uso de uma biblioteca específica, pois as funções utilizadas já estão na biblioteca padrão que o compilador utiliza. Todavia, no presente trabalho foi necessário empregar adicionalmente a biblioteca SoftwareSerial.h, a qual será apresentada ao longo desta seção.

O software (IDE) para programar a Sketch do Arduino pode ser expandido através da utilização de bibliotecas, assim como qualquer plataforma de programação. As bibliotecas oferecem funcionalidade extra para uso na Sketch, por exemplo, trabalhando com hardware ou manipulação de dados. Um número de bibliotecas vem instalados com a IDE, mas também é possível fazer o download ou criar o seu próprio. Para mais detalhes, consultar a referência. [13]

As tabelas 5.1a e 5.1b mostram a diversidade de bibliotecas para o Arduino.

Tipo de bibliotecas	Bibliotecas	Descrição
Standard Libraries	EEPROM	reading and writing to "permanent" storage
	Ethernet	for connecting to the internet using the Arduino Ethernet Shield
	Firmata	for communicating with applications on the computer using a standard serial protocol.
	GSM	for connecting to a GSM/GRPS network with the GSM shield.
	LiquidCrystal	for controlling liquid crystal displays (LCDs)
	SD	for reading and writing SD cards
	Servo	for controlling servo motors
	SPI	for communicating with devices using the Serial Peripheral Interface (SPI) Bus
	SoftwareSerial	for serial communication on any digital pins. Version 1.0 and later of Arduino incorporate Mikal Hart's NewSoftSeriallibrary as SoftwareSerial.
	Stepper	for controlling stepper motors
	TFT	for drawing text , images, and shapes on the Arduino TFT screen
	WiFi	for connecting to the internet using the Arduino WiFi shield
	Wire	Two Wire Interface (TWI/I2C) for sending and receiving data over a net of devices or sensors.
Due Only Libraries	Audio	Play audio files from a SD card.
	Scheduler	Manage multiple nonblocking tasks.
	USBHost	Communicate with USB peripherals like mice and keyboards.
Esplora Only Library	Esplora	this library enable you to easily access to various sensors and actuators mounted on the Esplora board.
Arduino Robot Library	Robot	this library enables easy access to the functions of the Arduino Robot
Arduino Yún Bridge Library	Bridge Library	Enables communication between the Linux processor and the Arduino on the Yún.
USB Libraries (Leonardo, Micro, Due, and Esplora)	Keyboard	Send keystrokes to an attached computer.
	Mouse	Control cursor movement on a connected computer.

Tabela 5.1a: Diversividades de Bibliotecas para o Arduino [13]

Tipo de bibliotecas	Bibliotecas	Descrição
Contributed Libraries	If you're using one of these libraries, you need to install it first. See these instructions for details on installation. There's also a tutorial on writing your own libraries.	
Communication (networking and protocols)	Messenger	for processing text based messages from the computer
	NewSoftSerial	an improved version of the SoftwareSerial library
	OneWire	control devices (from Dallas Semiconductor) that use the One Wire protocol.
	PS2Keyboard	read characters from a PS2 keyboard.
	Simple Message System	send messages between Arduino and the computer
	SSerial2Mobile	send text messages or emails using a cell phone (via AT commands over software serial)
	Webduino	extensible web server library (for use with the Arduino Ethernet Shield)
	X10	Sending X10 signals over AC power lines
	XBee	for communicating with XBees in API mode
	SerialControl	Remote control other Arduinos over a serial connection
Sensing	Capacitive Sensing	turn two or more pins into capacitive sensors
	Debounce	for reading noisy digital inputs (e.g. from buttons)
Displays and LEDs	GFX	base class with standard graphics routines (by Adafruit Industries)
	GLCD	graphics routines for LCD based on the KS0108 or equivalent chipset.
	Improved LCD library	fixes LCD initialization bugs in official Arduino LCD library
	LedControl	for controlling LED matrices or seven segment displays with a MAX7221 or MAX7219.
	LedControl	an alternative to the Matrix library for driving multiple LEDs with Maxim chips.
	LedDisplay	control of a HCMS 29xx scrolling LED display.
	Matrix	Basic LED Matrix display manipulation library
	PCD8544	for the LCD controller on Nokia 55100 like displays (by Adafruit Industries)
	Sprite	Basic image sprite manipulation library for use in animations with an LED matrix
	ST7735	for the LCD controller on a 1.8", 128x160 TFT screen (by Adafruit Industries)
Audio and Waveforms	FFT	frequency analysis of audio or other analog signals
	Tone	generate audio frequency square waves in the background on any microcontroller pin
Motors and PWM	TLC5940	16 channel 12 bit PWM controller.
Timing	DateTime	a library for keeping track of the current date and time in software.
	Metro	help you time actions at regular intervals
	MsTimer2	uses the timer 2 interrupt to trigger an action every N milliseconds.
Utilities	PString	a lightweight class for printing to buffers
	Streaming	a method to simplify print statements

Tabela 5.1b: Diversividades de Bibliotecas para o Arduino [13]

A biblioteca SoftwareSerial.h tem a função de estabelecer uma comunicação serial em quaisquer pinos digitais. Começando com o software IDE Arduino v1.0 (dezembro de 2011), NewSoftSerial.h [26] substituiu a biblioteca SoftwareSerial.h antiga biblioteca suportada oficialmente. Isto significa que se o projetista possui v1.0 ou posterior, não deve baixar esta biblioteca pois já está incluída nas novas versões com o nome SoftwareSerial.h (antiga NewSoftSerial) [27].

Na versão do software IDE para Arduino v1.63 o arquivo da biblioteca pode ser encontrado na pasta:

[…\Arduino\hardware\Arduino\avr\libraries\SoftwareSerial\SoftwareSerial.h]

Editando esse arquivo pode-se verificar todo o código. Nesse trabalho, foi necessário modificar o parâmetro [#define _SS_MAX_RX_BUFF 64] de 64 para 256 para solucionar problemas, relacionado a instabilidade do funcionamento do protótipo.

É possível encontrar diversos exemplos também como [SoftwareSerialExample.ino].

SoftwareSerialExample.ino: [13]

```
/* …
 modified 25 May 2012  by Tom Igoe  based on Mikal Hart's example
 This example code is in the public domain.
 */
#include <SoftwareSerial.h>
SoftwareSerial mySerial(10, 11); // RX, TX
void setup()
{
  // Open serial communications and wait for port to open:
  Serial.begin(57600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }
  Serial.println("Goodnight moon!");
  // set the data rate for the SoftwareSerial port
  mySerial.begin(4800);
  mySerial.println("Hello, world?");
}
void loop() // run over and over
{
  if (mySerial.available())
    Serial.write(mySerial.read());
  if (Serial.available())
    mySerial.write(Serial.read());
}
```

(Tom Igoe, 2012, [13])

Esse exemplo é importante para poder ajudar a estabelecer a comunicação entre o Arduino o Shield GSM e o XBee.

Esse mesmo exemplo está bem explícito na página oficial do Arduino. [13]

Mais informações, na página de referência [27].

5.2 BIBLIOTECA – XBEE-ARDUINO

5.2.1 COMANDOS AT

O Módulo XBee pode ser configurado por um programa chamado X-CTU, que foi desenvolvido pelo próprio fabricante do rádio XBee ou por um software como o HyperTerminal. Para configurar o módulo, é necessário inicialmente selecionar a porta serial, a taxa de transferência, controle de fluxo, bits de dados, paridade e bit de parada.

A seguir, será demonstrado um exemplo de como configurar de forma mais básica e segura para o Módulo XBee Série 1:

1. Instalar o Software X-CTU disponível no site do fabricante e atualizar o software pois ele atualizará o banco de dados dos Firmware disponível: [16]

Nome do software do usado no trabalho:

XCTU Next Gen Installer – Versão: 40003026_C

2. Conectar o módulo do X-Bee no hardware que estabelecerá uma conexão segura entre o módulo X-bee e o X-CTU da interface USB.

Figura 5.3: Módulo XBee 1m W S1 ZigBee Kit para Arduino

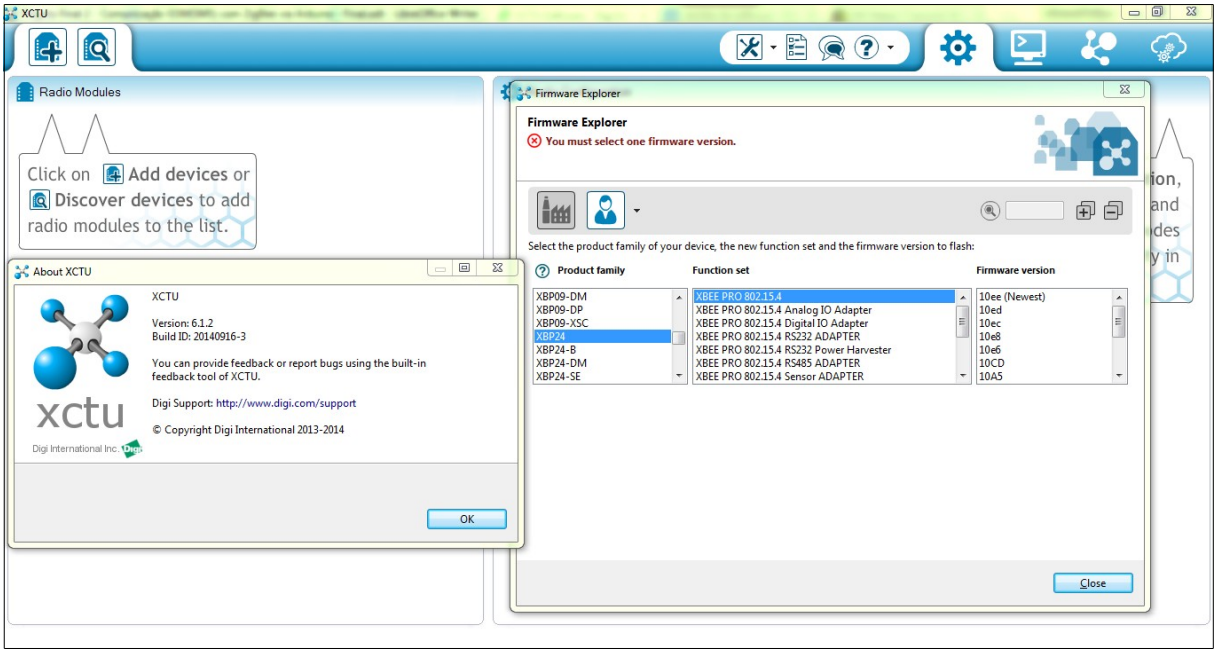


Figura 5.4: Atualizando o Firmware.

3. Escolher o produto correto e atualizar para a versão mais recente do Firmware apropriado para a aplicação, como mostra a Figura 5.4.

4. Configurar com os parâmetros corretos para a aplicação.

O modelo do Módulo de XBee usado no projeto foi desenvolvido pela MaxStream. A descrição dos parâmetros está disponível no datasheet do fabricante. [16]

5.2.2 COMANDO AT TRABALHANDO EM CONJUNTO COM ARDUINO.

Com a ajuda de um microcontrolador mais as bibliotecas do Arduino é possível associar à linguagem de programação do Arduino os comandos AT's disponíveis para o firmware do Módulo já configurado. Isto é mostrado em detalhes no Apêndice I. Já o modo em que API (Application Programming Interface) é empregada é mostrado como exemplo no Apêndice II.

No projeto foi utilizado um modo alternativo chamado de I/O Line Passing (Apêndice I), disponível somente nos módulos série 1, em razão de inúmeras dificuldades encontradas para fazer se comunicar no modo API. Uma vez que o modo API é o principal forma de se comunicar entre os módulos XBee, pois com ele é possível explorar todas as funcionalidades disponíveis, sugere-se como tema de futuros trabalhos o aperfeiçoamento desta parte do código.

5.3 BIBLIOTECA GSM PARA O ARDUINO

5.3.1 COMANDOS AT

Assim como no XBee, é possível conectar ao Arduino por meio de um software como o HyperTerminal e executar comandos. Como o SIM900 está soldado no shield para o Arduino, os comandos AT podem ser executados pelo software do Arduino. Após compilar o Sketch, é necessário apenas abrir o monitor serial, e os comandos já estão prontos para serem executados.

Depois que o shield GSM estiver com os LEDs estáveis, uma conexão segura (como a mostrada no Anexo III) é estabelecida com o software IDE do Arduino pelo serial monitor deste software ou pelo HyperTerminal, e é possível executar comandos ATs podendo enviar ou receber SMS, seguindo a documentação do SIM900. Esse exemplo pode ser encontrado no Anexo III no final do documento.

Foi necessário tomar algumas medidas de precaução para o bom funcionamento do protótipo do qual trata este projeto. A primeira foi garantir que a linha GSM estivesse cadastrada na operadora, com crédito para enviar SMS. A outra foi certificar que os LEDs estavam de acordo com a documentação do SIM900 e do fabricante do Shield. [19] [20]

5.3.2 COMANDOS AT TRABALHANDO EM CONJUNTO COM ARDUINO.

O ótimo exemplo pode ser encontrado em [29]. Se trata exatamente de um tutorial sobre o uso dos comandos AT com o Arduino. O **ANEXO A** desse documento mostra o código, que por ser grande e complexo foi aproveitado na íntegra, sem quaisquer modificações.

6 CÓDIGO DESENVOLVIDO PARA O TRABALHO

6.1 COMUNICAÇÃO ENTRE O SHIELD GSM E ARDUINO MEGA 2560

Depois de muito testes obteve-se uma versão relativamente simplificado do código.

O código foi compilado no Arduino 1.6 [13] transcrito no **APÊNDICE III**.

O objetivo dessa Sketch (código) é fazer o Arduino receber um SMS e acionar um LED ligando e desligando pelo texto chave do SMS. Com vistas a escrever um código o mais simples possível, foi adicionada uma criptografia muito básica.

A seguir são comentados os principais comandos do código.

Nas 12 primeiras linhas é mostrado como comentário (/* … */) no código de que forma é possível verificar o status da conexão com o SIM Card na rede 2G da operadora. Essa explicação é fornecida pelo fabricante do Efcom. [20]

Para esse código é necessário usar a biblioteca SoftwareSerial.h que está declarada na primeira linha útil da sketch [#include <SoftwareSerial.h>].

Declaram-se as portas digitais do Arduino 50 e 51 como TX e RX, respectivamente, [SoftwareSerial mySerial(50, 51)];

Declaram-se três variáveis do tipo String: linhaAtual, sms, smsAnterior com nenhuma informação; uma variável led do tipo int, com o valor 30 que representa a porta digital do Arduino; e outra, lendoSMS do tipo boolean. Mais detalhes sobre as diferenças de tipos de variáveis podem ser encontradas na referência bibliográfica [13].

… **APÊNDICE III**.

```
#include <SoftwareSerial.h>
   SoftwareSerial mySerial(50, 51);
   String  linhaAtual = "";
   String  sms = "";
   String  smsAnterior = "";
   boolean lendoSMS = false;
   int led = 30;

…
```

Foi definido baud rate como 9600; existem outros aceitáveis, mas, no caso, foi escolhido esse, pois se mostrou estável, [mySerial.begin(9600);]. Como explicado anteriormente, a variável led começa com o valor 30 que se refere ao pino 30 da porta digital do Arduino Mega 2560. Com a função pinMode(), declara-se a porta como de saída [pinMode(led, OUTPUT);]. Em seguida, é executado um comando AT que vai fazer o SIM900 ficar no modo de leitura de texto para facilitar a leitura do SMS [mySerial.println("AT+CMGF=1");]. Por último, executa-se [void mostraDadosSerial()], que explicado mais adiante.

… **APÊNDICE III**.

```
void setup()
{
   mySerial.begin(9600);
   Serial.begin(9600);
   pinMode(led, OUTPUT);
   Serial.println("Configurando SMS modo texto");
   mySerial.println("AT+CMGF=1");
   mostraDadosSerial();
}
...
```

A cada loop é executado o comando AT [mySerial.println("AT+CMGF=1");] para forçar o modo leitura e pede-se para mostrar os dados recebidos (no caso proveniente do SMS [mostraDadosSerial();]). Pede-se para esperar 1 segundo [delay(1000);].

Em seguida, define-se uma condição do tipo While [while (mySerial.available()>0)]. Essa condição vai verificar se a entrada no RX recebeu um valor maior que zero; no caso, quando chega um SMS com certeza é maior que zero. Para que isso funcione é chamada a rotina [void mostraDadosSerial()], que determina que quando a função definida pela biblioteca [mySerial.available()!=0] for diferente de 0, ou seja, trafega algum dado, se executa-se a função [Serial.write(mySerial.read());].

A função [Serial.write(mySerial.read());] executa o seguinte procedimento: se existe algum dado trafegando essa função vai ler de forma serial todos os dados com a função [mySerial.read()]; Logo, os dados estarão já lidos em [mySerial.read()]. Essa leitura será passada para a variável [char inChar] de forma dinâmica, ou seja, a cada loop dentro da condição While irá ler dado por dado (nesse caso irá ler os dados do SMS recebido). A cada Loop dentro do While será detectado se houve algum dado e será armanezado na variável local [char inChar]. Na ausência de dados o void loop() ficará em loop a cada 1 segundo.

Um passo importante e necessário é definir uma variável inChar do tipo Char porque a cada loop dentro da condição [while (mySerial.available()>0)],em cada leitura, [mySerial.read();], vai transformar o byte em caractere.

O trecho do código explicado no texto acima é assim:

```
… APÊNDICE III.
void loop()
{
   mySerial.println("AT+CMGR=1");
   mostraDadosSerial();
   delay(1000);
   while (mySerial.available()>0)
     {
         char inChar = mySerial.read();
         linhaAtual += inChar;
...                    ...
void mostraDadosSerial()
{
   while(mySerial.available()!=0)
   {
     Serial.write(mySerial.read());
   }
}
...
```

O próximo bloco do código vai da linha 43 à linha 83, e os comentários são o que seguem.

No momento em que a variável do tipo String linhaAtual começa a receber caractere por caractere a cada loop da condição do While através da variável inChar. Então esses caracteres é inserida numa outra variável sms do tipo String (linha 57), com isso a variável linhaAtual se torna uma variável como ponte entre o byte que a variável inChar, recebe e transforma em caractere, para a variável sms.

Quando houver quebra de linha para evitar problema do reconhecimento do caractere na variável sms do tipo String existe a condição do tipo IF, como mostrado no trecho do código abaixo.

```
… APÊNDICE III.
 if (inChar == '\n'){
              linhaAtual = "";}
…
```

Toda vez que existir um quebra de linha a variável linhaAtual irá ser limpa. Pois a cada byte lido pode haver uma quebra de linha, pois como explico anteriormente a função [mySerial.read();] está sendo escrita devido a função [Serial.write(mySerial.read());], isso é

importante para que a Sketch cumpra sua função, pois as variáveis linhaAtual e inChar vai servir para descriptografar a chave contida no texto do SMS, explicado a diante para interpretar a mensagem do SMS. Antes sobre essa questão da quebra de linha é profundado.

Em 9 de maio de 2011 foi publicado um código por Tom Igoe. Esse código tem o nome de Serial Event [14] e foi adaptado para utilização no trabalho. Disponível no **ANEXO B**.

A criptografia na Sketch se carateriza por dois caracteres especiais distintos definidos segundo o critério abaixo:

1) Os dois caracteres escolhidos não podem estar presentes no cabeçalho do SMS.

2) O carácter escolhido não pode estar presente no texto do SMS.

Esses caracteres determinam o início e o fim do texto.

Na Sketch do trabalho foi escolhido o caracter "@" que determinará o início da leitura do texto do SMS e o caracter "<" que determinará o final da leitura do texto do SMS.

A condição if, na linha 48, verificará se a função [string.endsWith] é verdadeira. Essa função verificará a string linhaAtual se termina com o caractere "@". Sendo verdadeira a variável lendoSMS do tipo bolean será verdadeira e fará que a variável sms seja limpa, segundo a condição [if (linhaAtual.endsWith("@"))]. Isso significa que a exemplo, uma mensagem de texto enviada por SMS “@Liga LED<”, acionará o preenchimento da variável SMS. Ou seja, todos os caracteres que a variável linhaAtual recebe antes do caractere “@” é ignorado.

```
… APÊNDICE III.
if (linhaAtual.endsWith("@"))
        {
        lendoSMS = true;
        sms = "";
        }
```

A condição if, na linha 55, verificará se a variável lendoSMS do tipo boolean é verdadeira, ou seja, é será incrementado o carácter na variável SMS do tipo string. Enquanto a variável lendoSMS for verdadeira e a condição [if (inChar != '<')] fará que o caractere armazenada na variável inChar, do tipo Char, coloque o valor do caractere na variável SMS que foi limpa como explicado no paragrafo acima. Dessa forma todo o texto depois do surgimento do caractere “@” e incluindo ele será armazenada na string SMS. Quando surgir o

caractere “<” entrará na condição contrária de [if (inChar != '<')], logo a estrutura do eles, linhas 59 á 79, entra em execução.

```
… APÊNDICE III.
if (lendoSMS)
        {
        if (inChar != '<')
          {
           sms += inChar;
          }
        else
...
```

Dentro do bloco, linhas 59 á 79, o estado variável lendoSMS muda para falsa [lendoSMS = false;] e escreve a variável sms na serial pela função [Serial.println(sms);], escreve só para monitorar pelo serial monitor figura 5.1.

Quando a string SMS receber todos os caracteres de forma atender uma das condições, linhas: 63, 68 e 73. A Sketch (código) terá realizado seu objetivo que será mudar o estado lógico da porta do Arduino, somente quando solicitado por SMS pelo usuário.

```
… APÊNDICE III.
else
          {
          lendoSMS = false;
          Serial.println(sms);
          if(sms=="@Liga LED" && smsAnterior!=sms)
           {
            Serial.println("Ligando LED");
            digitalWrite(led, HIGH);
           }
          if(sms=="@Desliga LED" && smsAnterior!=sms)
           {
            Serial.println("Desligando LED");
            digitalWrite(led, LOW);
           }
          if(sms == smsAnterior)
            {
             mySerial.println("AT+CMGD=1,4");
             sms="";
            }
          smsAnterior=sms;
          }
...
```

Condições:

a) [if(sms=="@Liga LED" && smsAnterior!=sms)],

b) [if(sms=="@Desliga LED" && smsAnterior!=sms)],

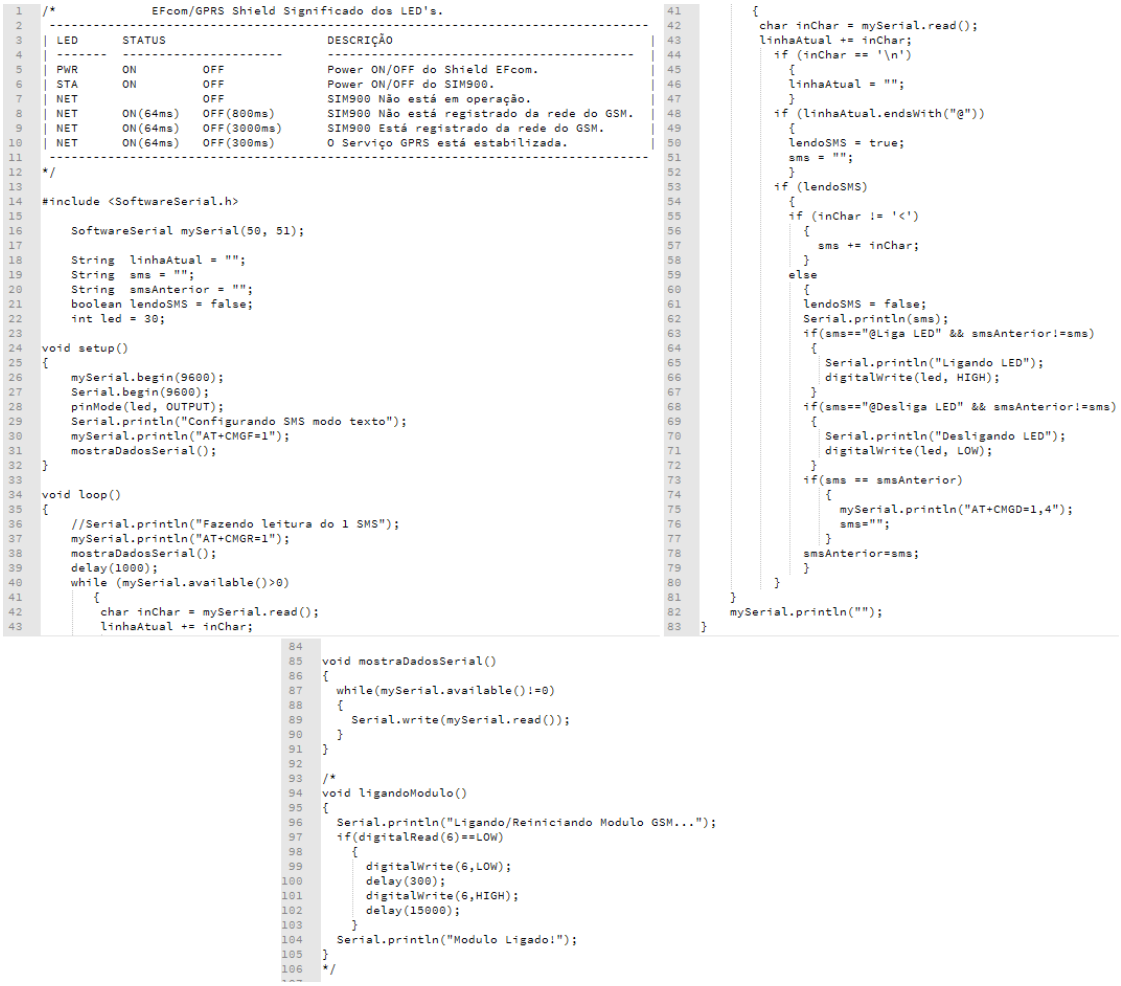
c) [if(sms == smsAnterior)]

Antes de explicar as condições, é fundamental para entendimento que a forma que a Sketch (código) foi desenvolvida, tem como objetivo de ser a forma mais simples para acionar com segurança. Então foi criado a variável smsAnterior do tipo string para evitar mensagem SMS repetida. O desenvolvimento do código para o trabalho foi para demonstrar o protótipo na defesa do trabalho, mostrado na sessão 4. Como não foi obtido sucesso com em operar o modo API com XBee dificultando completar objetivo do trabalho descrito em 1.3.

A ideia das 3 condições é acionar os LEDs tornando o estado lógico da porta digital 30 do Arduino Mega 2560 representado pela variável led. Para evitar acionamento desnecessário foi criada a variável smsAnterior, do tipo string, para fazer um AND lógico com a variável sms. Esse AND lógico é representado pelos caracteres &&. No final do else a variável smsAnterior recebe o texto da variável sms, para que no próximo loop o AND lógico funcione. Quando a variável sms é igual a variável smsAnterior executa-se o comando AT ["AT+CMGD=1,4"] pela função [mySerial.println("AT+CMGD=1,4");] e em seguida a variável sms é limpa.

Esse comando AT ["AT+CMGD=1,4"] deleta todos os SMSs da memória disponível para armazenar os SMS's no SIM900, tanto mensagens lidas como não lidas. Esse comando AT está disponível no documento SIM900 AT Commands Manual_V1.07 [19] na Página 99 em AT Commands According to GSM07.05.

Para facilitar a visualização do código do **APÊNDICE III**, as figuras 6.1 e 6.2 mostram o código completo numa única imagem.



```
/*            EFcom/GPRS Shield Significado dos LED's.
 ------------------------------------------------------------------------------------
| LED      STATUS                    DESCRIÇÃO                                       |
| -------  ----------------------    ----------------------------------------------  |
| PWR      ON          OFF           Power ON/OFF do Shield EFcom.                   |
| STA      ON          OFF           Power ON/OFF do SIM900.                         |
| NET                  OFF           SIM900 Não está em operação.                    |
| NET      ON(64ms)    OFF(800ms)    SIM900 Não está registrado da rede do GSM.      |
| NET      ON(64ms)    OFF(3000ms)   SIM900 Está registrado da rede do GSM.         |
| NET      ON(64ms)    OFF(300ms)    O Serviço GPRS está estabilizada.               |
 ------------------------------------------------------------------------------------
*/

#include <SoftwareSerial.h>

    SoftwareSerial mySerial(50, 51);

    String  linhaAtual = "";
    String  sms = "";
    String  smsAnterior = "";
    boolean lendoSMS = false;
    int led = 30;

void setup()
{
    mySerial.begin(9600);
    Serial.begin(9600);
    pinMode(led, OUTPUT);
    Serial.println("Configurando SMS modo texto");
    mySerial.println("AT+CMGF=1");
    mostraDadosSerial();
}

void loop()
{
    //Serial.println("Fazendo leitura do 1 SMS");
    mySerial.println("AT+CMGR=1");
    mostraDadosSerial();
    delay(1000);
    while (mySerial.available()>0)
       {
        char inChar = mySerial.read();
        linhaAtual += inChar;
       {
        char inChar = mySerial.read();
        linhaAtual += inChar;
          if (inChar == '\n')
            {
            linhaAtual = "";
            }
          if (linhaAtual.endsWith("@"))
            {
            lendoSMS = true;
            sms = "";
            }
          if (lendoSMS)
            {
            if (inChar != '<')
              {
                sms += inChar;
              }
            else
              {
              lendoSMS = false;
              Serial.println(sms);
              if(sms=="@Liga LED" && smsAnterior!=sms)
               {
                Serial.println("Ligando LED");
                digitalWrite(led, HIGH);
               }
              if(sms=="@Desliga LED" && smsAnterior!=sms)
               {
                Serial.println("Desligando LED");
                digitalWrite(led, LOW);
               }
              if(sms == smsAnterior)
                {
                  mySerial.println("AT+CMGD=1,4");
                  sms="";
                }
              smsAnterior=sms;
              }
            }
       }
    mySerial.println("");
}

void mostraDadosSerial()
{
  while(mySerial.available()!=0)
  {
    Serial.write(mySerial.read());
  }
}

/*
void ligandoModulo()
{
  Serial.println("Ligando/Reiniciando Modulo GSM...");
  if(digitalRead(6)==LOW)
    {
      digitalWrite(6,LOW);
      delay(300);
      digitalWrite(6,HIGH);
      delay(15000);
    }
  Serial.println("Modulo Ligado!");
}
*/
```

Figura 6.1: Código do Apêndice III

```
#include <SoftwareSerial.h>

    SoftwareSerial mySerial(50, 51);

    String  linhaAtual = "";
    String  sms = "";
    String  smsAnterior = "";
    boolean lendoSMS = false;
    int led = 30;

void setup()
{
    mySerial.begin(9600);
    Serial.begin(9600);
    pinMode(led, OUTPUT);
    Serial.println("Configurando SMS modo texto");
    mySerial.println("AT+CMGF=1");
    mostraDadosSerial();
}

void mostraDadosSerial()
{
  while (mySerial.available() != 0)
  {
    Serial.write(mySerial.read());
  }
}
```

```
void loop()
{
  mySerial.println("AT+CMGR=1");
  mostraDadosSerial();
  delay(1000);
  while (mySerial.available() > 0)
  {
    char inChar = mySerial.read();
    linhaAtual += inChar;
    if (inChar == '\n')
    {
      linhaAtual = "";
    }
    if (linhaAtual.endsWith("@"))
    {
      lendoSMS = true;
      sms = "";
    }
    if (lendoSMS)
    {
      if (inChar != '<')
      {
        sms += inChar;
      }
      else
      {
        lendoSMS = false;
        Serial.println(sms);
        if (sms == "@Liga LED" && smsAnterior != sms)
        {
          Serial.println("Ligando LED");
          digitalWrite(led, HIGH);
        }
        if (sms == "@Desliga LED" && smsAnterior != sms)
        {
          Serial.println("Desligando LED");
          digitalWrite(led, LOW);
        }
        if (sms == smsAnterior)
        {
          mySerial.println("AT+CMGD=1,4");
          sms = "";
        }
        smsAnterior = sms;
      }
    }
  }
  mySerial.println("");
}
```

Figura 6.2: Código do Apêndice III

6.2 COMUNICAÇÃO ENTRE O XBEE TX E O XBEE RX

Para a comunicação entre o XBee TX e o XBee RX, existe o software do fabricante, o XCTU [16]. Entretanto, optou-se por fazer configuração por um outro software terminal serial;

no caso, foi empregado o software Tera Term. [31]

No datasheet do fabricante, páginas 33, 34 e 40 até 89, encontram-se os comandos AT's padrão com suas respectivas descrições [16]. Neste datasheet, encontram-se também as especificações para a comunicação I/O line passing (páginas 23 e 24 dp datasheet)

Existem alternativas para configurar esse tipo de comunicação, que pode ser usado em conjunto com o modo API [33].

O módulo XBee Série 1 suporta um simples recurso chamado I/O line passing, que permite usar qualquer um dos 8 pinos (DIO 0 até 7) para detectar mudanças do estado lógico de Alta(~5 V) para Baixa (~3 V) do módulo XBee em um outro pino correspondente no módulo do XBee receptor. Basicamente, espelhar-se-á o estado lógico, como criação de um "fio virtual" livre de qualquer apoio de microcontrolador externo ou de análise de série.

O **APÊNDICE IV** mostra somente as alterações em relação à configuração padrão.

Já o **APÊNDICE V** são mostrados todos os parâmetros da versão do firmware 10EF, e já com as alterações efetuadas utilizando o software Tera Term [31] em relação à configuração padrão.

A tabela 6.1 mostra os principais parâmetros alterados do padrão, do firmware XB24-XBee 802.15.4-versão 10EF.

Parametros	Xbee Base	Xbee Remoto
CH Channel	C	C
ID PAN ID	FAB	FAB
DH Destination Address High	0	0
DL Destination Address Low	4321	8765
MY 16-bit Source Address	8765	4321
NI Note Identifier	Xbee Base	Xbee Remoto
Sleep Mode	No Sleep [0]	No Sleep [0]
BD Interface Data Rate	9600 [3]	9600 [3]
AP API Enable	API disable [0]	API disable [0]
D3 DIO3 Configuration	DI [3]	DO Low [4]
D2 DIO2 Configuration	DI [3]	DO Low [4]
D1 DIO1 Configuration	DI [3]	DO Low [4]
D0 DIO0 Configuration	DI [3]	DO Low [4]
IU I/O Output Enable	Disabled [0]	Disabled [0]
IC DIO Change Detect	F	F
IR Sample Rate * 1 ms	14	0
IA I/O Input Address	FFFFFFFFFFFFFFFF	8765
T0 D0 Output Timeout	FF	FF
T1 D1 Output Timeout	FF	FF
… T7 D7 Output Timeout	FF	FF

Tabela 6.1: Parâmetros alterados I/O Line Passing entre dois XBee(TX/RX)

Segue algumas descrições dos principais parâmetros modificados da configuração padrão, como mostrado na tabela 6.1. Por padrão existem mais de 60 parâmetros.

1) Parâmetro: Número do Canal [CH]

   Pode variar entre 0x0B – 0x1A. [16]

2) Parâmetro: Personal Area Network – [ID]

   Pode variar entre 0x0 – 0xFFF. [16]

3) Parâmetro: Destination Address High and Low Address [DH / DL]

   Pode variar entre 0x0 – 0xFFFFFFFF. [16]

Esses parâmetros têm a função de endereçar o endereço de destino. Os 32 bits dos 64 bits superiores para o DH e os 32 bits inferiores para o DL do endereço de destino de 64 bits. Define a DH registar a zero e DL menos de 0xFFFF para transmitir usando um endereço de 16 bits. 0x000000000000FFFF é o endereço de broadcast para o PAN. [16]

4) Parâmetro: Source Address [MY]

Pode variar entre 0x0 – 0xFFFF. [16]

Configura o endereço de origem de 16 bits. Definese = 0xFFFF para desativar a recepção de pacotes com endereços de 16 bits. Endereço de origem de 64 bits é o número de série e está sempre ativada. [16]

5) Parâmetro: Note Identifier [NI]

Pode variar entre 0 – 20 caracteres ASCII. [16]

Tem como função de identificar por String o módulo.

6) Parâmetro: Sleep Mode [SM]

Opções: No Sleep [0], Pin Hibernate [1], Pin Doze [2], Reserved [3], Cyclic Sleep Remote [4], Cyclic Sleep Remote w/ pin wakeup [5], *Sleep Coordinator-See descripton [6]. [16]

Pino Hibernação é reduz a energia economizando, Pino Doze fornece o wake up(despertador) mais rápido, Ciclo sono remoto com ou sem o pino wake up (despertador). Definição Coordenador sono é definido pelo parâmetro SM compatível com a versão 106 parâmetros SM 106. [16]

7) Parâmetro: Interface Data Rate [BD]

Opções: 1200 [0], 2400 [1], 4800 [2], 9600 [3], 19200 [4], 38400 [5], 57600 [6], 115200 [7]. [16]

Define a taxa de transmissão interface serial para a comunicação entre os modems seriais. [16]

8) Parâmetro: Modo API [AP]

I/O de dados é enviado para fora do UART usando um quadro(frame) de API. Todos os outros dados podem ser enviados e recebidos através da Operação transparente ou enquadramento(framing) API se o modo de API está habilitado (AP> 0). [16]

Operações API suportar dois RX (recepção) identificadores de quadros para dados de I/O (definir o endereço de 16 bits para 0xFFFE e o módulo fará endereçamento de 64 bits). [16]

9) Parâmetro: DIOX Configuration [DX]

Para configurar as opções para os pinos, tabela 6.2, do módulo XBee.

As opções: conversor analógico para digital, entrada e saída digital.

Funções dos pinos e seus números PIN associado e comandos. [16]

AD = conversor analógico-digital, DIO = Digital Input/Output

Funções dos pinos não aplicáveis na presente seção são indicadas dentro de parênteses.

Função do Pino (PIN)	Numero do Pino	Comando AT
AD0/DIO0	20	D0
AD1/DIO1	19	D1
AD2/DIO2	18	D2
AD3/DIO3/(COORD_SEL)	17	D3
AD4/DIO4	11	D4
AD5/DIO5/(ASSOCIATE)	15	D5
DIO6/(RTS)	16	D6
DIO7/(CTS)	12	D7
DI8/(DTR)/(Sleep_RQ)	9	D8

Tabela 6.2: Relação das funções com comando AT e seus respectivos pinis. [16]

A opção DIO detecta a mudança do estado lógico.

Quando “DIO Detectar Mudança” é habilitado (usando o comando IC), os pinos DIO 0-7 são monitorados. Quando a mudança é detectada em um pino de DIO, ocorrerá o seguinte:

1. Um pacote RF é enviado com os níveis de pinos atualizados DIO. Este pacote não conterá quaisquer amostras ADC. [16]

2. Todas as amostras em fila são transmitidas antes da mudança detectar. [16]

Quando a mudança é detectada não afetará o Pino (Pin Sleep wake-up). O pino D8 (DTR/Sleep_RQ/DI8) é a única linha que vai despertar um módulo de Pin sono. Se não todas as amostras são coletadas, o módulo ainda vai entrar no modo de sono após detectar uma mudança pacote é enviado. [16]

10) Parâmetro: I/O Output Enable [IU]

Habilita I/O dados recebidos para ser enviado UART. Os dados são enviados através de um quadro(frame) de API, independentemente do modo ATAP. [16]

11) Parâmetro: DIO Change Detection [IC]

Pode variar entre 0x0 – 0xFF. [16]

Configura o bit no campo do quadro(frame), com valores para monitorar a mudança. Cada bit permite o monitoramento de DIO0 até DIO7 para mudanças. Se detectado, os dados são transmitidos apenas com dados DIO. Quaisquer amostras em fila de espera para transmissão será enviado em primeiro lugar. [16]

12) Parâmetro: Sample Rate [IR]

Pode variar entre 0x0 – 0xFFFF. Este valor configurado será multiplicado por 1 ms. [16]

Definir taxa de amostragem. Quando define este parâmetro faz com que o modem aceite todas as amostras do DIO e ADC habilitado em um intervalo especificado.

13) Parâmetro: Input Address [IA]

Pode variar entre 0x0 – 0xFFFFFFFFFFFFFFFF. [16]

Configura os endereços de módulo para que saídas estão vinculadas. Definir todos os bytes para 0xFF não permitirá que qualquer I/O receba quadros(frames) da mudandas detectadas dos pinos de saida. Para configurar para receber todos, basta setar com 0xFFFF. [16]

14) Parâmetro: Output Timeout [TX]

Pode variar entre 0x0 – 0xFF. Este valor configurado será multiplicado por 100 ms.

Configura o valor de tempo de saída dos pinos no modo I/O Line Passing. Quando a saída apresenta um nível não-padrão é iniciado um temporizador. O cronômetro é reiniciado quando um pacote válido I/O é recebido. Pode ser configurado em todas os pinos DIO0 até DIO7, T0 até T7. [16]

6.3 DIFICULDADES ENCONTRADAS

As bibliotecas para Arduino que a comunidade desenvolve não funcionam 100% com os Shields desenvolvidos para o Arduino, então os fabricantes estão adaptando algumas bibliotecas fornecidas junto aos produtos. Logo, nem todas as funções nativas ou com outras bibliotecas funcionam juntas.

A exemplo, foi necessário mudar o valor de uma variável de 64 para 256 para não ter problema com o módulo do GSM.

Biblioteca: SoftwareSerial.h

#define _SS_MAX_RX_BUFF 64(256) // RX buffer size

Houve bastante dificuldade em se utilizar o modo API para comunicação do XBee, então a programação foi adaptada para o modo I/O Line Passing para a comunicação entre os módulos XBee. Atribui-se essa dificuldade ao fato de ter sido utilizado um modelo da Série 1. Com os modelos da Série 2, aparentemente com o firmware atualizado desenvolvido para o modo API, se espera mais facilidade em operar com estabilidade.

Um exemplo mal sucedido se encontra disponível no **APÊNDICE II**. A solução é apresentada no **APÊNDICE III e APÊNDICE IV**, tal solução se mostrou bem-sucedida durante o desenvolvimento do projeto e foi utilizada no trabalho. O código desenvolvido foi suficiente para o funcionamento do protótipo, mas limitou o alcance de alguns dos objetivos descritos na seção 1.3.

Como sugestão para trabalhos futuros, sugere-se a realização de estudos para dimensionar com ensaios os benefícios de se utilizar o protocolo ZigBee. No presente trabalho, o foco foi a realização de uma aplicação do protótipo na prática funcionando de forma estável com base neste padrão. De fato, pela revisão bibliográfica realizada, são inúmeros os trabalhos que demonstram o baixo consumo de energia proporcionado pelo padrão. Existem vários recursos disponíveis pelo protocolo ZigBee que influenciam bastante no consumo, além das diferenças entre cada módulo XBee comercializado pelo fabricante.

O maior desafio no projeto foi em relação a programação, em função da necessidade de se integrar todas as linguagens e hardware para fazer o sistema funcionar e, principalmente, para gerenciar os eventos durante a comunicação.

7 AUTOMAÇÃO INDUSTRIAL E RESIDENCIAL

Nos próximos parágrafos são transcritos alguns trechos de um artigo de autoria de Neves et. al [30] que traduz o pensamento do autor deste trabalho sobre a relevância do ramo de Automação.

A área de automação industrial está sendo repensada em função do grande desenvolvimento experimentado pelas técnicas digitais. No contexto industrial, há algumas décadas os problemas de automação são cada vez mais importantes. A sociedade depara-se com o avanço da tecnologia e com os seus desafios, que não são poucos. No entanto, observa-se que algumas perguntas precisam ser respondidas para melhor encaminhar esta importante área do conhecimento: como as instituições podem formar profissionais capazes de ter uma capacitação técnica suficiente para contornar suas próprias dificuldades? E garantir uma relação técnica com a sociedade sem assustá-la? O assunto é diversificado, pois abrange desde tópicos relativos à arquitetura de hardware e software, programação de controladores lógicos programáveis, controle de malhas contínuas até o gerenciamento estratégico de uma empresa, passando pela supervisão dos processos industriais e pela logística da produção. As técnicas desenvolvidas para o tratamento desses problemas atingiram hoje um relativo grau de sofisticação tecnológica e formal, exigindo pessoal técnico com formação específica para sua aplicação adequada. [30]

Os cursos de engenharia elétrica, engenharia mecânica, engenharia de produção, engenharia de controle e automação vêm se colocando na contingência de munir seus estudantes de ferramentas que os possibilitem, no menor tempo possível, se adequar ao quotidiano técnico de uma empresa e, pelo maior tempo possível, estarem preparados para se atualizar tecnicamente. Estes objetivos, em parte conflitantes, conduzem para a seguinte questão: qual o compromisso ideal entre profundidade e abrangência quando se leciona uma disciplina de automação industrial? De fato, as limitações de tempo num curso de engenharia obrigam que se opte ou por aprofundar certos tópicos da matéria, deixando o aluno sem visão de conjunto, ou por dar uma ideia geral do problema, deixando lacunas na formação do estudante que tornarão mais lento o acompanhamento dos avanços de seu campo de trabalho. E assim, a formação de engenheiros qualificados para o futuro é necessária equilibrando estas

decisões sobre profundidade e abrangência. Segundo Paulo Freire, a tecnologia de um país é sua educação de qualidade, e a tecnologia é a base de sustentação da economia e soberania de uma nação. [30]

Com base nisso, serão apresentadas dez situações como interessantes áreas nas quais a tecnologia possibilita novas perspectivas atuais e futuras para o desenvolvimento da automação. Em algumas delas são levantados problemas comuns que surgem da própria evolução tecnológica, como problemas sociais. A maioria, porém, discute os benefícios de algumas tecnologias consideradas relevantes para a automação. Na sequência, faz-se uma análise destes problemas.

Entre os muitos desafios da automação moderna, serão abordados neste trabalho os seguintes: no campo social, na formação técnica de profissionais e educação da sociedade quanto à evolução tecnológica proporcionada pela automação; em sistemas críticos, segurança e confiabilidade; na otimização de informações, no sentido de fornecer uma interface de software apropriada; na gerência de informações de tempo real; na área de reconhecimento de padrões; identificação de falhas em sistemas de automação; no que diz respeito à comunicação, implementação de comunicação segura entre dispositivos heterogêneos; em automação residencial, a utilização de um protótipo mínimo capaz de atender as necessidades dos mais variados tipos de residências; aplicações na área de medicina, instrumentos de precisão; e, por fim, impactos sociais e ambientais gerados pela automação. [30]

O trecho citado acima traduz o pensamento do autor, sobre a prática do trabalho aplicada na indústria, mas que podem ser implementados os mesmos conceitos para a automação residencial. O autor acredita que para criar um sistema de automação tanto industrial como residencial é muito mais produtivo discutir esses desafios, que têm influência no desenvolvimento de qualquer projeto, do que a parte técnica propriamente dita.. Para tanto, com base no artigo acima mencionado, cada situação é comentada, fazendo um paralelo entre o referido artigo e o pensamento do autor deste trabalho.

1) Formação Técnica de Profissionais e Educação da Sociedade quanto à Evolução Tecnológica Proporcionada pela Automação. [30]

Esse primeiro desafio traduz a dificuldade de no Brasil se conseguir projetar um sistema útil para sociedade brasileira, tanto pela dificuldade técnica, quanto pelo baixo desenvolvimento tecnológico e baixíssima educação da sociedade para usufruir dos sistemas.

2) Segurança e Confiabilidade em Sistemas Críticos. [30]

Esse desafio técnico influencia muito no desenvolvimento de qualquer projeto, pois aumentar a segurança inclui uma complexidade para garantir a confiabilidade no futuro sistema e principalmente nos cenários críticos que possa ocorrer que pode inviabilizar o projeto. A todo momento pode elevar o custo e forçar a revisão do planejamento.

3) Otimização de Informações, no Sentido de Fornecer uma Interface Homem-Máquina Apropriada. [30]

É um desafio que pode se tornar mais difícil devido à primeira situação; aliás, todos os desafios estão interligados e o mais importante é o primeiro desafio citado. O conhecimento técnico das linguagens HTML, PHP, CSS, JavaScript, AJAX, etc. ajudam a amenizar, melhorando a interface Homem-Máquina e favorecendo a interoperabilidade das interfaces.

4) Reconhecimento de Padrões. [30]

Esse desafio é o que garante a interoperabilidade das interfaces, a exemplo desse trabalho, que só foi viável pois foram relacionados os padrões para tal.

5) Identificação de Falhas em Sistemas de Automação. [30]

Faz parte do escopo de qualquer trabalho prever as falhas. Entretanto, sempre temos como desafio prever os possíveis falhas que desconhecemos, pois estas costumam aparecer durante a implementação do projeto e depois no funcionamento prático.

6) Comunicação Segura entre Dispositivos Heterogêneos. [30]

Esse desafio é o que melhor representa este trabalho, que se tornou viável graças às bibliotecas e ao hardware do Arduino, que abrem uma gama de possibilidades de desenvolvimento de inúmeros projetos e garante confidencialidade, autenticação, autorização e integridade entre Dispositivos Heterogêneos.

7) Sistemas de Automação Residencial. [30]

Sobre esse desafio será transcrito o trecho do artigo mencionado anteriormente.

Ao se falar em automação residencial, vários são os indicadores de muitas perspectivas de crescimento. O conceito de casa/escritório automatizado evoluiu bastante e hoje já não é mais visto com uma noção futurística e sim como uma opção de conforto, bem-estar e qualidade de vida ao permitir o controle da residência remotamente, economia de tempo e esforço com tarefas repetitivas e interação com usuários à distância. A própria evolução tecnológica contribuiu muito para que ideias que antes eram vistas como impraticáveis pudessem ser implementadas e hoje se vê muitos projetos ousados de automação residencial integrando vários dispositivos como computadores, celulares e a TV digital. Os avanços tecnológicos na área de microeletrônica baratearam os custos dos dispositivos, tornando fácil a disponibilidade de processadores, microcontroladores de tamanho reduzido, baixo consumo, bem como barramentos eficientes para melhoria de comunicação. A evolução da automação residencial passa atualmente por um período de migração de toda a tecnologia desenvolvida ao longo das décadas em automação predial e industrial, adequando-se à infraestrutura de uma residência. Dentre as funcionalidades a serem desenvolvidas e implementadas pelos sistemas de automação residencial podem-se destacar (ARAUJO E PEREIRA, 2003) o gerenciamento, monitoramento e otimização do consumo de energia; conforto térmico através do controle HVAC (Heating, Ventilation and Air Conditioning); controle de iluminação, controle de acesso, monitoramento e segurança física de dispositivos; dispositivos de manutenção inteligente, entre outros. [30]

8) Gerência de Informações de Tempo Real. [30]

É um desafio relacionado à otimização da informação e inclui, também, a linguagem para banco de dados (MySQL). Na opinião do autor é o melhor exemplo, onde a automação residencial e industrial se complementam.

O desenvolvimento da automação ligado à evolução de tecnologias tem, na comunicação industrial, importante ligação com o crescimento da tecnologia da informação, principalmente na relação entre os níveis gerencial e operacional de um ambiente de Automação Industrial. A TI (tecnologia da informação), que alterou bastante as hierarquias e estruturas no ambiente de escritório, passa agora a influenciar também o ambiente industrial, nos mais diversos níveis, desde indústrias de processos, prédios e sistemas logísticos,

entrando até no nível de automação residencial. A gerência de informações industriais surgiu na medida em que, com o amadurecimento das tecnologias dos equipamentos de campo inteligentes, uma grande quantidade de informações possibilitadas por eles pode agora ser disponibilizada para outras aplicações em outros níveis de tecnologias, sendo possível organizá-las, por exemplo, em formato conveniente para análise no nível de gerência de processos industriais. Os modernos sistemas de gerência de informações web permitem facilitar a análise e interpretação dos dados obtidos pela comunicação com outras camadas do sistema de automação industrial. Implementar tais ferramentas, em ambiente industrial, em tempo real, é um grande desafio para a automação. [30]

9) Aplicações na Área de Medicina. [30]

Nesse desafio o autor discorda, pois acredita que não só na área de medicina mas em todas outras também enfrentamos diversos obstáculos no Brasil, como nos meios de transporte urbano em geral, indústria de eletroeletrônica, eletromecânica, componentes robotizados, etc. O que falta, como dito no artigo, em todas as áreas é integração das universidades e as empresas para produção de tecnologia nacional, assim como com empresas multinacionais no sentido de produção de artigos e pesquisas, diminuindo os obstáculos entre os mundos acadêmico e industrial.

10) Impactos Sociais e Ambientes Gerados pela Automação. [30]

O desafio de mensurar os impactos da tecnologia na sociedade ainda é muito recente. Inúmeros trabalhos estão sendo desenvolvidos nesse sentido, pois desde a década de 80 do século XX é notória a mudança que a sociedade vem sofrendo como consequência do fenômeno da globalização.

8 CONCLUSÃO

O projeto foi bem-sucedido no sentido de se desenvolver um protótipo plenamente funcional, demonstrando sua aplicabilidade a uma situação prática. Foi possível apresentar todos os conceitos envolvidos na proposta do tema do trabalho através de uma versão simplificada do sistema, no qual uma carga – no caso, uma lâmpada – liga e desliga por comandos SMS, responsáveis por acionar o relé ao qual ela está conectada, a qualquer distância dentro da área de cobertura 2G da operadora.

A arquitetura de rede proposta foi plenamente implementada, à exceção da comunicação entre o Arduino e PC. Uma vez estabelecida essa comunicação, torna-se possível verificar os dados das variáveis e criar variáveis adicionais, como contadores, etc. Nesse sentido, é possível, sim, pensar em criar uma gerência para verificar ao menos os eventos. Esse tipo de análise de viabilidade foi pesquisada e estudada antes da realização do presente trabalho, e entende-se ser este um objeto de estudo interessante para trabalhos futuros. Para implementar essa gerência, o requisito é realizar a comunicação do XBee receptor para o XBee transmissor, ou seja, a comunicação deve ser bidirecional. No protótipo que apresentamos, a comunicação se dá de forma unidirecional, permitida somente no modo API.

Além disso, o conceito mais amplo de automação residencial também foi abordado de forma teórica, e a visão do autor sobre o tema foi mostrada.

Como extensão deste trabalho, sugere-se, futuramente, a realização de estudos para dimensionar com ensaios os benefícios de se utilizar o protocolo ZigBee. No presente trabalho, o foco foi a realização de uma aplicação prática e funcional com base neste padrão. De fato, pela revisão bibliográfica realizada, são inúmeros os trabalhos que demonstram o baixo consumo de energia proporcionado pelo padrão. Existem vários recursos disponíveis pelo protocolo ZigBee que influenciam bastante no consumo, além das diferenças entre cada módulo XBee comercializado pelo fabricante.

As possibilidades de implementação na prática são limitadas apenas pela imaginação do leitor. É uma realidade e existem diversos exemplos sendo publicados e comercializados.

9 REFERÊNCIA BIBLIOGRÁFICA

[1] ERGEN, S. C.; **ZigBee/IEEE 802.15.4 Summary.** Califórnia: Berkeley, 2004
Disponível em:
<http://www.prism.uvsq.fr/~mogue/Biblio/Sensor/AUTRES/zigbee_summary.pdf>
Acessado em 26 de junho 2015

[2] IEEE Computer Society; **IEEE 802.15 WPAN ™ Task Group 4 (TG4)**
Disponível em:
<http://ieee802.org/15/completed.html> <http://www.ieee802.org/15/about.html>
Acessado em 26 de junho 2015

[3] IEEE Computer Society; **Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)**. IEEE Std 802.15.4 2011(Revision of IEEE Std 802.15.4-2006)
Disponível em: <https://standards.ieee.org/findstds/standard/802.15.4-2011.html>
Acessado em 26 de junho 2015

[4] Torres dos Santos, Sergio; **Redes de Sensores Sem Fio em Monitoramento e Controle -** Junho/2007 - Dissertação apresentada à COPPE/UFRJ
Autor: Sergio Torres dos Santos; Orientadores:
Aloysio de Castro Pinto Pedroza e Luís Henrique Maciel Kosmalski Costa
Disponível em: <http://www.gta.ufrj.br/ftp/gta/TechReports/Sergio07/Sergio07.pdf>
Acessado em 14 de Agosto 2015

[5] Ferret dos Santos, Luiz; **A IEEE 802.15.4 COMO PLATAFORMA DE COMUNICAÇÃO DE DADOS** - Revista Ilha digital, ISSN 2177-2649, volume 4, páginas 97 – 105, 2013.
Everton Luiz Ferret dos Santos - Professor do Departamento Acadêmico de Eletrônica (DAELN), campus Florianópolis, IFSC <everton@ifsc.edu.br>.
Disponível em:
<http://ilhadigital.florianopolis.ifsc.edu.br/index.php/ilhadigital/article/download/55/50> Acessado em 14 de Agosto 2015

[6] Azevedo, Tiago; **Roteamento ZigBee**
CPE 825 - Roteamento em Redes de Computadores
Programa de Engenharia de Sistemas e Computação - UFRJ
Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia
Disponível em:
<http://www.gta.ufrj.br/ensino/CPE825/2006/resumos/TrabalhoZigbee.pdf>
Acessado em 14 de Agosto 2015

[7] Barbosa, G. V., Ferreira, J. L.; **Utilização do Protocolo ZigBee na Comunicação com PLC.**
Trabalho de Conclusão de Curso Engenharia Elétrica Telecomunicação e Automação PUC-MG Poços de Caldas 2009, Alunos: Guilherme Varela Barbosa; Jefferson Luiz Ferreira. Orientador: Ramiro Romankevicius Costa.
Disponível em:
<http://www.scribd.com/doc/222959967/102603796-Apresentacao Resumida-de Monografia-ZigBee> Acessado em 14 de Agosto 2015

[8] Ranjan, Ashish; **Wireless Communication using Zigbee** – Apresentado pelo: Ashish Ranjan – 200101244
Disponibilizado pelo Prof. Prabhat Ranjan's na web page para estudantes
Email: <prabhat_ranjan@daiict.ac.in> - <http://intranet.daiict.ac.in/~ranjan/>
Disponível em: <https://www.scribd.com/doc/205910435/Wireless-Communication-using-Zigbee-pdf> Acessado em 14 de Agosto 2015

[9] Lakhe, P. R.; **Wireless Sensor Network Using Zigbee** - VNCET-30 Mar'12. International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 National Conference on Emerging Trends in Engineering & Technology (VNCET-30 Mar'12)
Prof. Pravin R. Lakhe (Department Of Computer Engineering, Vidyavardhini's College Of Engineering and Technology, K. T. Marg, Vasai Road-401202
Email: pravinlakhe@rediffmail.com)
Disponível em: <http://www.ijera.com/special_issue/VNCET_Mar_2012/55.pdf>
Acessado em 14 de Agosto 2015

[10] ANATEL; **Republica o Regulamento sobre equipamentos de radiocomunicação de radiação restrita.** Resolução nº 506, de 1º de julho de 2008 - ANATEL
Disponível em: <http://legislacao.anatel.gov.br/resolucoes/2008/104-resolucao-506>
Acessado em 14 de Agosto 2015

[11] Frare, B. P.; Araki, F. N. T.; Xavier, M. F.; **Aplicação do ZigBee na Segurança.**
Bruno Pinaffi Frare, Flávio Nobuteru Tachikawa Araki, Marcelo Fantini Xavier
Orientador: Franscico Martins Portelinha - 15 de Junho de 2009
Universidade Federal de Itajubá - UNIFEI
Instituto de Engenharia de Sistemas e Tecnologias da Informação
Engenharia da Computação - Itajubá - MG
Disponível em:
<http://td-zigbee-project.googlecode.com/svn-history/r3/trunk/main.pdf>
Acessado em 14 de Agosto 2015

[12] Wikipédia; **Arduino**
Wikipedia, com acesso livre, enciclopédia livre de conteúdo de Internet, apoiada e organizada pela organização sem fins lucrativos Wikimedia Foundation.
Disponível em:
<http://en.wikipedia.org/wiki/Arduino> ; <http://pt.wikipedia.org/wiki/Arduino>
Acessado em 14 de Agosto 2015

[13] Arduino; **Web Site Oficial do Arduino**
Disponível em: <http://Arduino.cc/>; Acessado em 14 de Agosto 2015

[14] Igoe, Tom; **Serial Event –** Maio de 2011
Disponível em: <http://www.Arduino.cc/en/Tutorial/SerialEvent>
Acessado em 14 de Agosto 2015

[15] Wikipédia; **XBee**
Wikipedia, com acesso livre, enciclopédia livre de conteúdo de Internet, apoiada e organizada pela organização sem fins lucrativos Wikimedia Foundation.
Disponível em: <http://en.wikipedia.org/wiki/XBee> Acessado em 14 de Agosto 2015

[16] Digi International® Inc. (Digi®); **Web Site Oficial do XBee**
XBee® 802.15.4
Software: XCTU Next Gen Installer, Windows x32/x64 Versão: 40003026_C
Disponível em: <http://ftp1.digi.com/support/utilities/40003026_C.exe>
Acessado em 14 de Agosto 2015
Todos os documentos técnicos podem ser encontrados nos links abaixo.
Disponível em:
<http://www.digi.com/products/XBee-rf-solutions/modules/XBee-series1-module#resources> Acessado em 14 de Agosto 2015
<http://examples.digi.com/get-started/basic-XBee-802-15-4-chat/>
Acessado em 14 de Agosto 2015
<http://examples.digi.com/quick-reference/> Acessado em 14 de Agosto 2015

[17] Silva, Bruno S.P.; Soares, Sérgio A. F.; **Mini curso ao XBee**
Sérgio Aurélio Ferreira Soares e Bruno Sampaio Pinho da Silva
Graduandos em Engenharia de Computação em 15 e 16 de agosto de 2012
Universidade Federal do Vale do São Francisco – UNIVASF
Disponível em:
<https://www.scribd.com/doc/157329099/Minicurso-XBee-Ago-2012>
<http://www.brunopinho.com/novo/>
Acessado em 14 de Agosto 2015

[18] Faludi, Robert; **Wireless Sensor Networks**; ISBN: 978-0-596-80773-3 – December 2010: First Edition. Copyright © 2011 Robert Faludi. All rights reserved.
Printed in the United States of America. Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.
Disponível em:
<http://my.safaribooksonline.com>; <corporate@oreilly.com>;
<https://www.safaribooksonline.com/library/view/building-wireless-sensor/780596807757/> Acessado em 15 de Agosto 2015

[19] SIMCom Wireless Solutions; **Web Site Oficial do SIM 900**
SIMCom Wireless Solutions, a subsidiary of SIM Technology Group Ltd.
Documentos Disponível em:<http://old.simcomm2m.com/producten.aspx?id=1019>
Acessado em 15 de Agosto 2015

[20] EFCom GPRS/GSM Shield; **Web Site Oficial do EFCom GPRS/GSM Shield**
GPRS Shield - EFCom é um módulo sem fio ultracompacto e confiável para arduino.
<http://www.elecfreaks.com/wiki/index.php?title=EFCom_GPRS/GSM_Shield>
Acessado em 15 de Agosto 2015

[21] ANATEL; **Certificado de Homologação do SIM 900**;
Nº: 0685-14-5900 Emissão: 27/02/2014
Disponível em:
<http://sistemas.anatel.gov.br/sgch/HistoricoCertificado/Homologacao.asp?
NumRFGCT=54914&idtHistoricoCert=9647875>
Acessado em 15 de Agosto 2015

[22] Wikipédia; **Processing**
Wikipedia, com acesso livre, enciclopédia livre de conteúdo de Internet, apoiada e organizada pela organização sem fins lucrativos Wikimedia Foundation.
Disponível em: <https://en.wikipedia.org/wiki/Processing_(programming_language)>
Acessado em 15 de Agosto 2015

[23] Ras, Caseys; Fry, Benjamin; **Web Site Oficial sobre Processig –** 2001**.**
Tipo de programação: Programming Paradigm - Object-oriented programming (OOP)
Disponível em: <https://processing.org/> Acessado em 15 de Agosto 2015

[24] Wikipédia; **Wiring**
Wikipedia, com acesso livre, enciclopédia livre de conteúdo de Internet, apoiada e organizada pela organização sem fins lucrativos Wikimedia Foundation.
Disponível em: <https://en.wikipedia.org/wiki/Wiring_(development_platform)>
Acessado em 15 de Agosto 2015

[25] Barragán, Hernando; **Web Site Oficial sobre Wiring** – 2003
Wiring é um projeto aberto iniciado por Hernando Barragán.
Disponível em: <http://wiring.org.co/about.html> Acessado em 15 de Agosto 2015

[26] Hart, Mikal; Fried, Limor; **Biblioteca: NewSoftwareSerial.h –** 2011
Multi-instance software serial library Copyright (c) 2006 David A. Mellis.
Disponível em: <http://arduiniana.org/libraries/newsoftserial/> ;
<https://github.com/johnmccombs/arduino-libraries/blob/master/NewSoftSerial/NewSoftSerial.h>
Acessado em 15 de Agosto 2015

[27] Hart, Mikal; Fried, Limor; Stoffregen, Paul; Mace Garrett; Hagman Brett; **Biblioteca: SoftwareSerial.h** – 2015
Disponível em:
<https://www.arduino.cc/en/Main/Software> Acessado em 15 de Agosto 2015
Limor Fried
Disponível em: <http://ladyada.net/> Acessado em 15 de Agosto 2015
Mikal Hart
Disponível em: <http://www.arduiniana.org> Acessado em 15 de Agosto 2015
Paul Stoffregen
<http://www.pjrc.com/about/contact.html> Acessado em 15 de Agosto 2015
Garrett Mace <http://www.macetech.com/blog/> Acessado em 15 de Agosto 2015
Brett Hagman <http://www.roguerobotics.com/> Acessado em 15 de Agosto 2015

[28] ANATEL; **Certificado de Homologação do XBee**
Nº: 0991-10-1209 Emissão: 14/05/2010
Disponível em:
<http://sistemas.anatel.gov.br/sgch/HistoricoCertificado/Homologacao.asp?NumRFGCT=77410&idtHistoricoCert=9260398> Acessado em 15 de Agosto 2015

[29] Sigurðsson, G. H; **SMS Server – SIM900 e Arduino Mega 2560**
Novembro de 2013 - Autor: Guðjón Hólm Sigurðsson
Disponível em:
<https://www.youtube.com/watch?v=kIFa7IhpR8s> Acessado em 16 de Agosto 2015
<https://drive.google.com/file/d/0B9AZRKxOrL4aLUE4ZlRoM2NnTEE/edit>
Acessado em 16 de Agosto 2015

[30] Neves, Cleonor; Duarte, Leonardo; Viana, Nairon; Lucena, Jr; **OS DEZ MAIORES DESAFIOS DA AUTOMAÇÃO INDUSTRIAL AS PERSPECTIVAS PARA O FUTURO.** II Congresso de Pesquisa e Inovação da Rede Norte-Nordeste de Educação Tecnológica - João Pessoa – PB – II CONNEPI 2007
Autores: Cleonor NEVES; Leonardo DUARTE; Nairon VIANA;
Vicente Ferreira de LUCENA Jr.
Disponível em:
<http://www.redenet.edu.br/publicacoes/arquivos/20080109_085035_INDU-068.pdf>
Acessado em 16 de Agosto 2015

[31] Teranish, Robert T.; **Software Tera Term**
Autor Original: Robert T. Teranishi
Projeto: Tera Term Project
Open Source Development at OSDN
Disponível em: <http://ttssh2.sourceforge.jp/index.html.en>
Acessado em 18 de Agosto 2015

[32] Pyne, Matt; **Sofware Tinycad**
Autor Original: Matt Pyne
Disponível em: <http://sourceforge.net/projects/tinycad/>
Acessado em 18 de Agosto 2015

[33] Ahart Matt; **DXBee Tech Tip: Digital IO Line Passing with XBee ;**
Autor: Matt Ahart, Postado em 25 Agosto de 2015
Disponível em: <http://www.digi.com/blog/tag/line-passing/> ou
<http://www.digi.com/blog/videos/XBee-tech-tip-digital-io-line-passing-with-XBee/>
Acessado em 12 de Outubro 2015

[34] Arduino; **Web Site Oficial do Arduino - Variedades de bibliotecas**
Disponível em: <http://playground.arduino.cc/Main/Interfacing>;
Acessado em 15 de Outrubro 2015
Disponível em: <http://playground.arduino.cc/Main/ComponentLib>;
Acessado em 15 de Outrubro 2015
Disponível em: <https://www.arduino.cc/en/Tutorial/LibraryExamples>;
Acessado em 15 de Outrubro 2015
Disponível em: <http://playground.arduino.cc/Interfacing/Processing>;
Acessado em 15 de Outrubro 2015

[35] Comunidade com mais de mil autores; **PhoneGap e Cordova plugins list**
Contato autores: e-mail: hello@phonegap-plugins.com
Disponível em: <http://phonegap-plugins.com/> Acessado em 15 de Outrubro 2015
Disponível em: <http://phonegap-plugins.com/keywords/arduino>
Acessado em 15 de Outrubro 2015
Disponível em: <https://github.com/xseignard/cordovarduino>
Acessado em 15 de Outrubro 2015

[36] Wikipédia; **PhoneGap**

Wikipedia, com acesso livre, enciclopédia livre de conteúdo de Internet, apoiada e organizada pela organização sem fins lucrativos Wikimedia Foundation.

Disponível em: <https://en.wikipedia.org/wiki/PhoneGap>

Acessado em 15 de Outrubro 2015

Disponível em: <https://cordova.apache.org/docs/en/3.0.0/plugin_ref/plugman.html>

Acessado em 15 de Outrubro 2015

APÊNDICE I – ENVIANDO TEXTO DE UM XBEE PARA OUTRO XBEE

Programação da comunicação entre o Arduino e XBee acoplado no Arduino. E entre o XBee acoplado no Arduino e o outro XBee receptor conectado ao PC. Para isso é necessário configurar o XBee transmissor e o XBee receptor por comando AT no será mostrado na Parte A do Apêndice I e na Parte B será mostrado a programação que será copilado e enviado ao Arduino.

Apêndice I – Parte A – Enviando texto de um XBee para outro XBee

Configuração básica, via comando AT nos dois módulos XBee. Como é uma rede ponto a ponto não foi necessário definir coordenador e Dispositivo final (end-device).

XBee1:

+++	(Entra em modo de configuração)
ATIDFAB	(Personal Area Network ID: 0xFAB)
ATMY2	(16-bit Source Address: 0x2)
ATDL3	(Destination Address Low: 0x3)
ATBD3	(Define 9600 Baud)
ATWR	(Grava a config.)
ATCN	(Sai do Modo Configuração)

XBee2:

+++	(Entra em modo de configuração)
ATIDFAB	(Personal Area Network ID: 0xFAB)
ATMY2	(16-bit Source Address: 0x3)
ATDL3	(Destination Address Low: 0x2)
ATBD3	(Define 9600 Baud)
ATWR	(Grava a config.)
ATCN	(Sai do Modo Configuração)

Apêndice I – Parte B – Enviando texto de um XBee para outro XBee

Programação da comunicação entre o Arduino e XBee acoplado no Arduino.

```
/*****************************************************************
Pinagem no Hardware:
Encontrar o pino XBee's DOUT (TX) e conectar em um pino como a exemplo o 15 (Arduino Mega RX)
Encontrar o pino XBee's DIN (RX) e conectar em um pino como a exemplo o 14 (Arduino Mega RX)
*****************************************************************/
// Usando a biblioteca SoftwareSerial, disponível pelo software para comunicar com o XBee:

#include <SoftwareSerial.h>

// Definindo conexão entre RX do XBee e TX do Arduino
// e TX do XBee e RX do Arduino
SoftwareSerial XBee(15, 14); // RX, TX

// Definindo a variável X com valor 0
unsigned long x=0;
// Estabelecendo entrar de dados serial com 9600 Bauds.
// Que nesse exemplo de programa vai ser pré-definido na com a msg "Manoel Furtado"

void setup()
{
Serial.begin(9600);
}
// No XBee receptor mostrará a mensagem "Manoel Furtado" na saída serial, no caso monitor.
// E o valor da variável X incrementando +1 a cada loop com delay de 1000 ms.

void loop()
{
  Serial.print("Manoel Furtado");
  Serial.println(x++);
  delay(1000);
  //x=x+1;
}
```

APÊNDICE II – XBEE MODO API

Comunicação de um XBee para outro XBee via modo API

Esse exemplo não está correto então foi utilizado o modo I/O Line Passing no trabalho.

```
// Usando a biblioteca SoftwareSerial, disponível pelo software para comunicar com o
XBee:
#include <SoftwareSerial.h>
// Definindo conexão entre RX do XBee e TX do Arduino
// e TX do XBee e RX do Arduino
SoftwareSerial XBee(15, 14); // RX, TX
int led = 31;
void setup()
{
pinMode(led,OUTPUT);
Serial.begin(9600);
}
void loop()
{
 digitalWrite(led,HIGH);       // LED ON
                               // setRemoteState(0x5);
 delay(5000);                  // 5s
 digitalWrite(led,LOW);        // LED OFF
                               // setRemoteState(0x4);
 delay(5000);                  // 5s
}
void setRemoteState(char value)
{
Serial.write(0x7E);             // Start Byte
Serial.write((byte)0x0);        // Parte mais significativa (sempre zero)
Serial.write(0x08);             // Parte mais significativa
Serial.write(0x08);             // Ox17 para remoto comando AT
Serial.write(0x4D);             // Seta Frame ID para zero para não repetir.
```

```
// ID para recipient, ou usar OxFFFF para Broacast
 Serial.write((byte)0x0);
 Serial.write((byte)0x0);
Serial.write((byte)0x0);
Serial.write((byte)0x0);
Serial.write((byte)0x0);
Serial.write((byte)0x0);
Serial.write(0xFF);              // OxFF para Broadcast
Serial.write(0xFF);              // OxFF para Broadcast
                                 // 16 bit for recipient or OxFFFE if unknown
Serial.write(0xFF);
Serial.write(0xFE);
Serial.write(0x02);      //Ox02 para aplicar as mudanças imediatamente remotamente

 // Nome do comando em caracters ASCII
 Serial.write(0xFF);
 Serial.write(0xFE);
 Serial.write(0x02);
 Serial.write('D');
 Serial.write('3');

 // Dado do comando em bytes
 Serial.write(value);

 // Checksum para todo o comprimento dos bytes.
 long sum = 0x17 + 0xFF + 0xFF + 0xFE + 0x02 + 'D' + '3' + value;
 Serial.write(0xFF - (sum & 0xFF));   // Calculate the proper checksum
}
```

APÊNDICE III – ENVIANDO SMS AO ARDUINO

```
/*                          EFcom/GPRS Shield Significado dos LED's.
 ---------------------------------------------------------------------------------------------------
| LED    STATUS                        DESCRIÇÃO                                                    |
| -------  ------------------------------  ------------------------------------------------------|
| PWR    ON            OFF             Power ON/OFF do Shield EFcom                               |
| STA    ON            OFF             Power ON/OFF do SIM900                                     |
| NET                  OFF             SIM900 Não está em operação                                |
| NET    ON(64ms)      OFF(800ms)      SIM900 Não está registrado da rede do GSM                  |
| NET    ON(64ms)      OFF(3000ms)     SIM900 Está registrado da rede do GSM                      |
| NET    ON(64ms)      OFF(300ms)      O Serviço GPRS está estabilizada                           |
 ---------------------------------------------------------------------------------------------------
*/

#include <SoftwareSerial.h>

   SoftwareSerial mySerial(50, 51);

   String  linhaAtual = "";
   String  sms = "";
   String  smsAnterior = "";
   boolean lendoSMS = false;
   int led = 30;

void setup()
{
   mySerial.begin(9600);
   Serial.begin(9600);
   pinMode(led, OUTPUT);
   //ligandoModulo();
   Serial.println("Configurando SMS modo texto");
   mySerial.println("AT+CMGF=1");
   mostraDadosSerial();
}
```

```
void loop()
{
   //Serial.println("Fazendo leitura do 1 SMS");
   mySerial.println("AT+CMGR=1");
   mostraDadosSerial();
   delay(1000);
   while (mySerial.available()>0)
     {
      char inChar = mySerial.read();
      linhaAtual += inChar;
       if (inChar == '\n')
        {
        linhaAtual = "";
        }
       if (linhaAtual.endsWith("@"))
        {
        lendoSMS = true;
        sms = "";
        }
       if (lendoSMS)
        {
        if (inChar != '<')
          {
           sms += inChar;
          }
        else
          {
          lendoSMS = false;
          Serial.println(sms);
          if(sms=="@Liga LED" && smsAnterior!=sms)
           {
            Serial.println("Ligando LED");
            digitalWrite(led, HIGH);
           }

          if(sms=="@Desliga LED" && smsAnterior!=sms)
           {
            Serial.println("Desligando LED");
            digitalWrite(led, LOW);
           }
```

```
            if(sms == smsAnterior)
              {
               mySerial.println("AT+CMGD=1,4");
               sms="";
              }
            smsAnterior=sms;
            }
          }
    }
    mySerial.println("");
}

void mostraDadosSerial()
{
  while(mySerial.available()!=0)
  {
    Serial.write(mySerial.read());
  }
}

/*
void ligandoModulo()
{
  Serial.println("Ligando/Reiniciando Modulo GSM...");
  if(digitalRead(6)==LOW)
    {
     digitalWrite(6,LOW);
     delay(300);
     digitalWrite(6,HIGH);
     delay(15000);
    }
  Serial.println("Modulo Ligado!");
}
*/
```

APÊNDICE IV: I/O LINE PASSING ENTRE DOIS XBEE(TX/RX) – COMANDO AT

Configuração da comunicação entre o XBee TX e o XBee RX por I/O Line Passing por comandos AT.

XBee1: XBee Base

+++	(Entra em modo de configuração)
ATIDFAB	(Personal Area Network ID: 0xFAB)
ATMY8765	(16-bit Source Address: 0x8765)
ATDL4321	(Destination Address Low: 0x4321)
ATBD3	(Define 9600 Baud)
ATD33	(Configuração DI (digital Input) (3))
ATD23	(Configuração DI (digital Input) (3))
ATD13	(Configuração DI (digital Input) (3))
ATD03	(Configuração DI (digital Input) (3))
ATIC	(DIO Chance Detect: F)
ATNIXBee Base	(Node Identifier: XBee Base)
ATIU0	(I/O Output Disabled)
ATWR	(Grava a config.)
ATCN	(Sai do Modo Configuração)

XBee2: XBee Remoto

+++	(Entra em modo de configuração)
ATIDFAB	(Personal Area Network ID: 0xFAB)
ATMY4321	(16-bit Source Address: 0x4321)
ATDL8765	(Destination Address Low: 0x8765)
ATNIXBee Remoto	(Node Identifier: XBee Remoto)
ATD33	(Configuração DO (digital Output) LOW (4))
ATD23	(Configuração DO (digital Output) LOW (4))
ATD13	(Configuração DO (digital Output) LOW (4))
ATD03	(Configuração DO (digital Output) LOW (4))
ATIC	(DIO Chance Detect: F)
ATCE0	(CE Coordenador [1] End Divice [0])
ATIA8765	(Setting address to 0xFFFF will allow any received I/O packet to change outputs.)
ATWR	(Grava a config.)
ATCN	(Sai do Modo Configuração)

APÊNDICE V: I/O LINE PASSING ENTRE DOIS XBEE(TX/RX) – UTILIZANDO O SOFTWARE XCTU

Versão XML exportado do software XCTU [16]

XBee1: XBee - Base

<?xml version="1.0" encoding="UTF-8"?>

```
<data>
 <profile>
  <description_file>xb24_15_4_10ef.xml</description_file>
  <settings>
   <setting command="CH">C</setting>
   <setting command="ID">FAB</setting>
   <setting command="DH">0</setting>
   <setting command="DL">4321</setting>
   <setting command="MY">8765</setting>
   <setting command="MM">0</setting>
   <setting command="RR">0</setting>
   <setting command="RN">0</setting>
   <setting command="NT">19</setting>
   <setting command="NO">0</setting>
   <setting command="CE">0</setting>
   <setting command="SC">1FFE</setting>
   <setting command="SD">4</setting>
   <setting command="A1">0</setting>
   <setting command="A2">0</setting>
   <setting command="EE">0</setting>
   <setting command="KY"></setting>
   <setting command="NI">XBee BASE</setting>
   <setting command="PL">4</setting>
   <setting command="CA">2C</setting>
   <setting command="SM">0</setting>
   <setting command="ST">1388</setting>
   <setting command="SP">0</setting>
   <setting command="DP">3E8</setting>
   <setting command="SO">0</setting>
   <setting command="BD">3</setting>
   <setting command="NB">0</setting>
   <setting command="RO">3</setting>
   <setting command="AP">0</setting>
   <setting command="D8">0</setting>
   <setting command="D7">1</setting>
   <setting command="D6">0</setting>
```

```
      <setting command="D5">1</setting>
      <setting command="D4">0</setting>
      <setting command="D3">3</setting>
      <setting command="D2">3</setting>
      <setting command="D1">3</setting>
      <setting command="D0">3</setting>
      <setting command="PR">FF</setting>
      <setting command="IU">0</setting>
      <setting command="IT">1</setting>
      <setting command="IC">F</setting>
      <setting command="IR">14</setting>
      <setting command="P0">1</setting>
      <setting command="P1">0</setting>
      <setting command="PT">FF</setting>
      <setting command="RP">28</setting>
      <setting command="IA">FFFFFFFFFFFFFFFF</setting>
      <setting command="T0">FF</setting>
      <setting command="T1">FF</setting>
      <setting command="T2">FF</setting>
      <setting command="T3">FF</setting>
      <setting command="T4">FF</setting>
      <setting command="T5">FF</setting>
      <setting command="T6">FF</setting>
      <setting command="T7">FF</setting>
      <setting command="DD">10000</setting>
      <setting command="CT">64</setting>
      <setting command="GT">3E8</setting>
      <setting command="CC">2B</setting>
    </settings>
  </profile>
</data>
```

XBee2: XBee - Remoto

```
<?xml version="1.0" encoding="UTF-8"?>

<data>
 <profile>
  <description_file>xb24_15_4_10ef.xml</description_file>
  <settings>
   <setting command="CH">C</setting>
   <setting command="ID">FAB</setting>
   <setting command="DH">0</setting>
   <setting command="DL">8765</setting>
   <setting command="MY">4321</setting>
   <setting command="MM">0</setting>
   <setting command="RR">0</setting>
   <setting command="RN">0</setting>
   <setting command="NT">19</setting>
   <setting command="NO">0</setting>
   <setting command="CE">0</setting>
   <setting command="SC">1FFE</setting>
   <setting command="SD">4</setting>
   <setting command="A1">0</setting>
   <setting command="A2">0</setting>
   <setting command="EE">0</setting>
   <setting command="KY"></setting>
   <setting command="NI">XBee Remoto</setting>
   <setting command="PL">4</setting>
   <setting command="CA">2C</setting>
   <setting command="SM">0</setting>
   <setting command="ST">1388</setting>
   <setting command="SP">0</setting>
   <setting command="DP">3E8</setting>
   <setting command="SO">0</setting>
   <setting command="BD">3</setting>
   <setting command="NB">0</setting>
   <setting command="RO">3</setting>
   <setting command="AP">0</setting>
   <setting command="D8">0</setting>
   <setting command="D7">1</setting>
   <setting command="D6">0</setting>
   <setting command="D5">1</setting>
   <setting command="D4">0</setting>
   <setting command="D3">4</setting>
   <setting command="D2">4</setting>
   <setting command="D1">4</setting>
   <setting command="D0">4</setting>
```

```
      <setting command="PR">FF</setting>
      <setting command="IU">1</setting>
      <setting command="IT">1</setting>
      <setting command="IC">F</setting>
      <setting command="IR">0</setting>
      <setting command="P0">1</setting>
      <setting command="P1">0</setting>
      <setting command="PT">FF</setting>
      <setting command="RP">28</setting>
      <setting command="IA">8765</setting>
      <setting command="T0">FF</setting>
      <setting command="T1">FF</setting>
      <setting command="T2">FF</setting>
      <setting command="T3">FF</setting>
      <setting command="T4">FF</setting>
      <setting command="T5">FF</setting>
      <setting command="T6">FF</setting>
      <setting command="T7">FF</setting>
      <setting command="DD">10000</setting>
      <setting command="CT">64</setting>
      <setting command="GT">3E8</setting>
      <setting command="CC">2B</setting>
    </settings>
  </profile>
</data>
```

ANEXO A – GSM SERVER

```
//programmer gudjon@undri.com
//todo:sorry about the messy coding (not enough time to fix)

#include <SoftwareSerial.h>

//FOR MEGA decomment line below.  On the EFCom shield DIS-connect the jumpers
//and put jumperwires from
//SHIELD RX to MEGA 51 and from SHIELD TX to MEGA 50

SoftwareSerial mySerial(50, 51);

//FOR UNO decomment line below.  On the EFCom shield connect the jumers,
//TX to D2 and RX to D3.
//SoftwareSerial mySerial(2, 3);
/*                          EFcom/GPRS Shield Light meaning.
 ---------------------------------------------------------------------------------------------------
| LED                       STATUS          DESCRIPTION                                             |
| ------------------------- -----     ------------------------------------------------------------|
| PWR                       ON       Power on the EFcom                                           |
| PWR                       OFF      Power off the EFcom                                          |
| STA                       ON       Power on the SIM900 OFF Power Off the SIM900|
| NET                       OFF      SIM900 is not running                                        |
| NET 64ms ON/800ms         OFF      SIM900 not register the Network                              |
| NET 64ms ON/3000ms        OFF      SIM900 registered to the network                             |
| NET 64ms ON/300ms         OFF      GPRS communication is established                            |
 ---------------------------------------------------------------------------------------------------
*/
////////////////////////////////////////////////////////////////////////////////
//SELECT HERE WICH PINS YOU WANT TO CONTROL ON THE ARDUINO
                                                    (THEY must be in sequence)
#define OPIN_FIRST 30
#define OPIN_LAST  33
//when program starts turn all pins on or off.   allowed values are either HIGH or LOW
#define OPIN_STARTSTATE LOW
//The controlling phonenumber;
#define PHONENUMBER "1234567"
////////////////////////////////////////////////////////////////////////////////
```

```
String msg = String("");
int SmsContentFlag = 0;

#define GSM_POWER  6 /*EFCom power pin*/
#define GSM_RESET  5 /*EFCom reset pin**/

//  S T A R T    Borrowed from GSMSHIELD Library   S T A R T
// some constants for the IsRxFinished() method
// length for the internal communication buffer

#define COMM_BUF_LEN        300
#define RX_NOT_STARTED      0
#define RX_ALREADY_STARTED  1

// variables connected with communication buffer
   byte *p_comm_buf;                    // pointer to the communication buffer
   byte comm_buf_len;                   // num. of characters in the buffer
   byte rx_state;                       // internal state of rx state machine
   uint16_t start_reception_tmout;      // max tmout for starting reception
   uint16_t interchar_tmout;            // previous time in msec.
   unsigned long prev_time;             // previous time in msec.
   byte comm_buf[COMM_BUF_LEN+1];  // communication buffer +1 for 0x00
                                                          termination
enum rx_state_enum
{
 RX_NOT_FINISHED = 0,            // not finished yet
 RX_FINISHED,                    // finished, some character was received
 RX_FINISHED_STR_RECV,           // finished and expected string received
 RX_FINISHED_STR_NOT_RECV,  // finished, but expected string not received
 RX_TMOUT_ERR,                   // finished, no character received
                                 // initial communication tmout occurred
 RX_LAST_ITEM
};

void RxInit(uint16_t start_comm_tmout, uint16_t max_interchar_tmout)
{
 rx_state = RX_NOT_STARTED;
 start_reception_tmout = start_comm_tmout;
 interchar_tmout = max_interchar_tmout;
 prev_time = millis();
```

```
  comm_buf[0] = 0x00; // end of string
  p_comm_buf = &comm_buf[0];
  comm_buf_len = 0;
  mySerial.flush(); // erase rx circular buffer
}
byte IsRxFinished(void)
{
  byte num_of_bytes;
  byte ret_val = RX_NOT_FINISHED;  // default not finished
  if (rx_state == RX_NOT_STARTED) {
    // Reception is not started yet - check tmout
    if (!mySerial.available()) {
      // still no character received => check timeout
      if ((unsigned long)(millis() - prev_time) >= start_reception_tmout) {
        // timeout elapsed => GSM module didn't start with response
        // so communication is takes as finished
        comm_buf[comm_buf_len] = 0x00;
        ret_val = RX_TMOUT_ERR;
      }
    }
    else {
      // at least one character received => so init inter-character
      // counting process again and go to the next state
      prev_time = millis(); // init tmout for inter-character space
      rx_state = RX_ALREADY_STARTED;
    }
  }
  if (rx_state == RX_ALREADY_STARTED) {
    // Reception already started
    // check new received bytes
    // only in case we have place in the buffer
    num_of_bytes = mySerial.available();
    // if there are some received bytes postpone the timeout
    if (num_of_bytes) prev_time = millis();
    // read all received bytes
    while (num_of_bytes) {
      num_of_bytes--;
      if (comm_buf_len < COMM_BUF_LEN) {
        // we have still place in the GSM internal comm. buffer =>
        // move available bytes from circular buffer
```

```
      // to the rx buffer
      *p_comm_buf = mySerial.read();
      p_comm_buf++;
      comm_buf_len++;
      comm_buf[comm_buf_len] = 0x00;  // and finish currently received characters
                                      // so after each character we have
                                      // valid string finished by the 0x00
     }
     else {
      mySerial.read();
     }
    }
    if ((unsigned long)(millis() - prev_time) >= interchar_tmout) {
     comm_buf[comm_buf_len] = 0x00;  // for sure finish string again
                                     // but it is not necessary
     ret_val = RX_FINISHED;
    }
  }
  return (ret_val);
}
/*****************************************************
Method checks received bytes
compare_string - pointer to the string which should be find
return: 0 - string was NOT received
        1 - string was received
*****************************************************/
byte IsStringReceived(char const *compare_string)
{
  char *ch;
  byte ret_val = 0;
  if(comm_buf_len) {
#ifdef DEBUG_ON
        Serial.println("ATT: ");
        Serial.print(compare_string);
        Serial.print("RIC: ");
        Serial.println((char *)comm_buf);
#endif
    ch = strstr((char *)comm_buf, compare_string);
    if (ch != NULL) {
     ret_val = 1;
```

```
   }
else
{
 /*#ifdef DEBUG_PRINT
        DebugPrint("\r\nDEBUG: expected string was NOT received\r\n", 0);
 #endif
 */
}
 }
 return (ret_val);
}
byte WaitResp(uint16_t start_comm_tmout, uint16_t max_interchar_tmout)
{
 byte status;
 RxInit(start_comm_tmout, max_interchar_tmout);
 // wait until response is not finished
 do {
  status = IsRxFinished();
 } while (status == RX_NOT_FINISHED);
 return (status);
}
///////////////þarf eþtta kannski ekki sjá fyrir ofan
byte WaitResp(uint16_t start_comm_tmout, uint16_t max_interchar_tmout,
              char const *expected_resp_string)
{
 byte status;
 byte ret_val;
 RxInit(start_comm_tmout, max_interchar_tmout);
 // wait until response is not finished
 do {
  status = IsRxFinished();
 } while (status == RX_NOT_FINISHED);
 if (status == RX_FINISHED) {
  // something was received but what was received?
  // -----------------------------------------
  if(IsStringReceived(expected_resp_string)) {
   // expected string was received
   // ----------------------------
   ret_val = RX_FINISHED_STR_RECV;
  }
```

```
   else {
ret_val = RX_FINISHED_STR_NOT_RECV;
}
  }
  else {
   // nothing was received
   // --------------------
   ret_val = RX_TMOUT_ERR;
  }
  return (ret_val);
}
///////////////þarf eþtta kannski ekki endir
/************************************************************
Method sends AT command and waits for response
return:
          -1,   // no response received
           0,   // response_string is different from the response
           1,   // response_string was included in the response
************************************************************/
char SendATCmdWaitResp(char const *AT_cmd_string,
              uint16_t start_comm_tmout, uint16_t max_interchar_tmout,
              char const *response_string,
              byte no_of_attempts)
{
  byte status;
  char ret_val = -1;
  byte i;
  for (i = 0; i < no_of_attempts; i++) {
   // delay 500 msec. before sending next repeated AT command
   // so if we have no_of_attempts=1 tmout will not occurred
   if (i > 0) delay(500);
   mySerial.println(AT_cmd_string);
   status = WaitResp(start_comm_tmout, max_interchar_tmout);
   if (status == RX_FINISHED) {
    // something was received but what was received?
    // ---------------------------------------------
    if(IsStringReceived(response_string)) {
     ret_val = 1;
     break;  // response is OK => finish
    }
```

```
    else ret_val = 0;
   }
   else {
    // nothing was received
    // --------------------
    ret_val = -1;
   }
  }
  return (ret_val);
}//char SendATCmdWaitRes
/************************************************************
Method sends SMS
number_str:   pointer to the phone number string
message_str:  pointer to the SMS text string
return:
      ERROR ret. val:
      ------------------------------------------------------------
      -1 - comm. line to the GSM module is not free
      -2 - GSM module didn't answer in timeout
      -3 - GSM module has answered "ERROR" string
      OK ret val:
      ---------------------------
      0 - SMS was not sent
      1 - SMS was sent
an example of usage:
      GSM gsm;
      gsm.SendSMS("00XXXYYYYYYYYY", "SMS text");
************************************************************/
char SendSMS(char *number_str, char *message_str)
{
  char end[2];
  end[0]=0x1a;
  end[1]='\0';/* Ctrl+Z */

  mySerial.print("\r");
  delay(1000);                    //Wait for a second while the modem sends an "OK"
  mySerial.print("AT+CMGF=1\r");   //Because we want to send the SMS in text mode
  delay(1000);
```

```
 //mySerial.print("AT+CSCA=\"+919032055002\"\r");  //Setting for the SMS
                                                    Message center number,
 //delay(1000);                          //uncomment only if required and replace with
                                         //the message center number obtained from
                                         //your GSM service provider.
                                         //Note that when specifying a tring of characters
                                         // " is entered as \"
  mySerial.print("AT+CMGS=\"");mySerial.print(number_str);  mySerial.print("\"\r");
//Start accepting the text for the message
                                    //to be sent to the number specified.
                                    //Replace this number with the target mobile number.
 delay(1000);
 mySerial.print(message_str); mySerial.print("\r");   //The text for the message
 delay(1000);
 mySerial.print(end);


}
////          E N D   Borrowed from GSMSHIELD Library     E N D          //
////
//// S E T U P   S E T U P   S E T U P
//// S E T U P   S E T U P   S E T U P
//// S E T U P   S E T U P   S E T U P
void setup()
{
 mySerial.begin(9600);
 Serial.begin(9600);
 if (-1 == SendATCmdWaitResp("AT", 500, 100, "OK", 5))
 {
  digitalWrite(GSM_POWER, HIGH);delay(3000);
  digitalWrite(GSM_POWER, LOW);delay(2000);
 }
 for(int i=OPIN_FIRST;i<OPIN_LAST+1;i++)
 {
  pinMode(i, OUTPUT);
  digitalWrite(i,OPIN_STARTSTATE);
 }
 Serial.println( "Starting" );
mySerial.println( "AT+CMGF=1" );
delay(200);
}
```

```
boolean bFoundMSGandNumber=false;
void loop()
{
   char SerialInByte;
   if(mySerial.available())
   {
      SerialInByte = (unsigned char)mySerial.read();
     delay(5);
      if( SerialInByte == 13 ){
       ProcessGprsMsg();
      }
      if( SerialInByte == 10 ){
      }
      else {
        msg += String(SerialInByte);
      }
    }
}
void ProcessSms( String sms ){
 Serial.print("ProcessSms:");Serial.println(sms);
 String finna = String("xxxxx");
 for(int i=OPIN_FIRST;i<OPIN_LAST+1;i++)
 {  finna=String("Off");finna+=i;
    if( sms.indexOf(finna) >= 0 || sms.indexOf("Offall")  >= 0 ){digitalWrite(i, LOW);
Serial.println(finna);}
 }
   for(int i=OPIN_FIRST;i<OPIN_LAST+1;i++)
 {  finna=String("On");finna+=i;
    if( sms.indexOf(finna) >= 0 || sms.indexOf("Onall")  >= 0){digitalWrite(i, HIGH);
Serial.println(finna);}
 }
 if( sms.indexOf("List") >= 0 )
 {
   finna="stada";
   for(int i=OPIN_FIRST;i<OPIN_LAST+1;i++)
   {  finna+=" ";finna+=i;finna+="=";finna+=digitalRead(i);  }
   int iSize=finna.length()+1;
   char charBuf[iSize];
   finna.toCharArray(charBuf, iSize);
   SendSMS(PHONENUMBER, charBuf);
```

```
  }
 mySerial.println( "AT+CMGDA=\"DEL ALL\"" );
 delay(200);
}
// EN: Request Text Mode for SMS messaging
void GprsTextModeSMS(){
  mySerial.println( "AT+CMGF=1" );
}
void GprsReadSmsStore( String SmsStorePos ){
  mySerial.print( "AT+CMGR=" );
  mySerial.println( SmsStorePos );
}
// EN: Clear the GPRS shield message buffer
void ClearGprsMsg(){
  msg = "";
}
// EN: interpret the GPRS shield message and act appropiately
void ProcessGprsMsg() {
  Serial.print( "ProcessGprsMsg();");Serial.println(msg);
  if( msg.indexOf( "Call Ready" ) >= 0 ){
   Serial.println( "*** GPRS Shield registered on Mobile Network ***" );
     GprsTextModeSMS();
  }
  //todo: búa til define string úr PHONENUMBER
  char findMe[strlen(PHONENUMBER
+2];strcpy(findMe,PHONENUMBER);strcat(findMe,"\"");
  if( (msg.indexOf( "+CMT" ) >= 0) && (msg.indexOf( findMe )>0) ){
     Serial.println( "*** SMS Received ***" );
     bFoundMSGandNumber=true;
  }
  else
  {
    if(bFoundMSGandNumber==true)
    {
      ProcessSms( msg );
    }
     bFoundMSGandNumber=false;
  }
  ClearGprsMsg();
  SmsContentFlag = 0;
```

```
}
int getValidPin(char *pNum)
{
  int iNum=strlen(pNum);
  if (iNum<1 || iNum>2) return false;//pin must be from 0..99
  iNum = atoi (pNum);
  if ((iNum < OPIN_FIRST) || (iNum > OPIN_LAST)) return false;
 //num must be equal or between OPIN_FIRST and OPIN_FIRST
  return iNum;
}
```

ANEXO B – SERIAL EVENT

```
/*
  Serial Event example

 When new serial data arrives, this sketch adds it to a String.
 When a newline is received, the loop prints the string and
 clears it.

 A good test for this is to try it with a GPS receiver
 that sends out NMEA 0183 sentences.

 Created 9 May 2011 by Tom Igoe
 This example code is in the public domain.
 http://www.arduino.cc/en/Tutorial/SerialEvent
 */

String inputString = "";         // a string to hold incoming data
boolean stringComplete = false;  // whether the string is complete

void setup() {
  // initialize serial:
  Serial.begin(9600);
  // reserve 200 bytes for the inputString:
  inputString.reserve(200);
}

void loop() {
  // print the string when a newline arrives:
  if (stringComplete) {
    Serial.println(inputString);
    // clear the string:
    inputString = "";
    stringComplete = false;
  }
}
```

```
/*

 SerialEvent occurs whenever a new data comes in the
hardware serial RX.  This routine is run between each
time loop() runs, so using delay inside loop can delay
response.  Multiple bytes of data may be available.

*/

void serialEvent() {
  while (Serial.available()) {
    // get the new byte:
    char inChar = (char)Serial.read();
    // add it to the inputString:
    inputString += inChar;
    // if the incoming character is a newline, set a flag
    // so the main loop can do something about it:
    if (inChar == '\n') {
      stringComplete = true;
    }
  }
}
```