UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO
CURSO DE ENGENHARIA DE COMPUTAÇÃO

Modelo de estacionamento automatizado utilizando sensoriamento sem fio

Iuri Guerra de Freitas Pereira

Orientador: Prof. Dr. Luiz Eduardo Cunha Leite

Monografia apresentada à Banca Examinadora do Trabalho de Conclusão do Curso de Engenharia de Computação e Automação em cumprimento às exigências legais como requisito parcial à obtenção do título de Engenheiro de Computação e Automação.

Natal, RN, Dezembro de 2010

Modelo de estacionamento automatizado utilizando sensoriamento sem fio

Iuri Guerra de Freitas Pereira

Trabalho de Conclusão de Curso aprovado em 15 de julho de 2010 pela banca examinadora composta pelos seguintes membros:

Prof. Dr. Luiz Eduardo Cunha Leite (Orientador).....ECT/UFRN

Prof. Dr. Glaucio Bezerra Brandao.....DCA/UFRN

Prof. Me. Gustavo Bezerra Paz Leitao.....IFRN/SEDE

Aos meus pais, Ivanaldo e
Francinete e minha prima Karla
Karine Gurgel.

Agradecimentos

Agradeço primeiramente a Deus, que me deu a oportunidade de estar aqui hoje escrevendo este trabalho.

A minha família, que esteve comigo nos momentos mais difíceis e nunca deixaram de me dar apoio e carinho desde o início.

A minha irmã Isabelle que teve toda paciência para revisar esse texto e procurar por erros de português.

A todos os colegas de trabalho que conviveram comigo o dia-a-dia e que de uma forma ou de outra me ajudaram também para a realização deste trabalho.

Aos meus colegas de curso pela motivação e estiveram sempre me apoiando.

Aos meus amigos que estão comigo nos maus e bons momentos

Em especial a minha tia Fátima Gurgel que me deu um empurrãozinho final de caráter motivacional para que eu chegasse a este momento.

Resumo

O crescente uso de tecnologias sem fio, aliado à necessidade de se encontrar soluções informatizadas para problemas relacionados ao trânsito, são fatores que motivam o desenvolvimento deste trabalho. O presente trabalho tem como finalidade projetar um sistema que identifica e gerencia vagas de estacionamento através da comunicação entre sensores sem fio presentes em cada vaga.

Abstract

The growling demand fo wireless Technologies allied with the need to find computer solutions in order to solve problems related to transit are factors that motivates the development of this work. This paper aims to design a system that identifies and manages parking spaces through the communication between wireless sensors present in each parking space.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Códigos

Capítulo 1

Introdução

Nos dias de hoje vem se notando um aumento significativo da utilização de tecnologias sem fio na solução dos problemas cotidianos. Uma migração de tecnologias, que antes eram cabeadas, é vista atualmente sem as limitações físicas impostas pelos fios quando se diz respeito à montagem de redes de computadores, tanto corporativas quanto residenciais.

Tal notável crescimento se deve boa parte a maior mobilidade e a possibilidade de interconectar muitas pessoas sem os problemas que seriam inerentes à uma rede cabeada. Além disso, os custos e os prazos de implantação e de manutenção são reduzidos. O fato de a tecnologia wireless estar se transformando em uma tecnologia cada vez mais viável, pelo fato de ser de boa aceitação pelo mercado, faz com que ela se torne, a cada dia, mais eficiente e atenda cada vez mais um número maior de clientes, possibilitando assim uma evolução natural.

Essa evolução traz consigo várias possibilidades e ideias para o uso de redes sem fio, principalmente no que diz respeito a comunicação entre sensores ou dispositivos que requerem baixa potência. Futuramente o usuário dessa tecnologia poderá utilizá-la para interagir com o ambiente a sua volta e obter informações, por exemplo, do consumo de energia dos equipamentos instalados em sua residência ou empresa em tempo real e, assim, poder controlar o consumo em busca de soluções de economia.[Messias 2008] Outra possibilidade é a automatização de serviços do cotidiano como, por exemplo, um sistema que evite que o usuário fique muito tempo em um estacionamento à procura de uma vaga para estacionar, pois o sistema irá se comunicar com ele e informar que existe uma vaga a espera.

Essa última possibilidade é tema de estudo desta monografia. O crescente número de veículos nas cidades nos últimos anos tem provocado significativo aumento de problemas envolvendo o complexo viário urbano nas médias e grandes cidades. É corriqueiro em determinadas horas do dia várias vias de acesso a principais ruas e avenidas das principais

metrópoles do país e do mundo estarem engarrafadas e com trânsito lento, ocasionando perda de tempo e consequentemente prejuízos econômicos para a sociedade.

O uso de tecnologias avançadas na área de redes e automação podem ser cruciais para amenizar boa parte dos problemas envolvendo perda de tempo em ruas e estacionamentos aonde o grande número de carros vem trazendo problemas diariamente. Chegar a um estabelecimento e saber na hora da chegada aonde o seu veículo deve ser estacionado e ainda ter a opção visual de analisar aonde existe uma vaga para estacionar certamente é uma boa estratégia para diminuir o tempo gasto em um grande estacionamento.

1.1 Objetivos

1.1.1 Objetivo Geral

Este trabalho tem como objetivo implementar, utilizando tecnologias atuais, um projeto na área de redes de sensores sem fio e automação que ajude a diminuir o tempo que um usuário demora a encontrar uma vaga em um estacionamento de grande rotatividade.

Para tanto serão analisadas, em teoria e prática, tecnologias de transmissão de dados e rede de sensores sem fio pelo padrão IEEE 802.15.4 e o protocolo ZigBee. Uma abordagem com sensor de ultrassom ou sonar interligado com um microcontrolador para enviar os dados através de um módulo de transmissão/recepção sem fio.

1.1.2 Objetivos Específicos

- Pesquisa sobre o padrão IEEE 802.15.4 e o protocolo ZigBee para a criação de uma rede de sensores sem fio;
- Pesquisa sobre o microcontrolador PIC16F628A para atender as necessidades do projeto e interfaceamento com um sensor de ultrassom/sonar para captura de dados;
- Analisar a viabilidade do projeto, os prós e contras de acordo com os resultados obtidos no estudo de caso implementado;

1.2 Metodologia

Uma metodologia baseada em pesquisas bibliográficas: livros, revistas e artigos será utilizada. Levantamentos de vários conceitos serão realizados a respeito da abordagem do funcionamento das tecnologias envolvidas no projeto. Estes conceitos servirão de

base para a modelagem e implementação de um projeto de monitoramento de vagas de estacionamento através de sensores de ultrassom/sonar.

1.3 Estrutura do Trabalho

Este trabalho está dividido em cinco capítulos que são subdivididos em seções. O primeiro capítulo corresponde à introdução, mostrando o que foi escrito a respeito do tema sobre uma perspectiva geral, assinalando tanto a relevância como o interesse do trabalho e do que será tratado em outras seções.

O segundo capítulo traz uma revisão bibliográfica sobre os padrões IEEE 802.15.4 e ZigBee, seu funcionamento e importância e considerações de segurança.

No terceiro capítulo é descrito o funcionamento de um microcontrolador, mais especificamente o PIC16F628A, que foi utilizado na implementação e o funcionamento em conjunto com um sensor ultrassom/sonar para aquisição de dados.

O quinto capítulo agrega as tecnologias abordadas nos capítulos anteriores e descreve como foi realizado o projeto, expõe os resultados obtidos e dificuldades encontradas no decorrer do mesmo.

E por último, o capítulo das considerações finais mostrará como outras pessoas poderão contribuir com este trabalho de pesquisa e quais as contribuições são necessárias. Tentará abordar projetos possíveis e discutir melhorias no projeto implementado.

Capítulo 2

O padrão IEEE 802.15.4 e o protocolo ZigBee

2.1 O IEEE 802.15.4

O padrão IEEE 802.15.4, liberado em maio de 2003, é um padrão que define a camada de comunicação em dois níveis no modelo OSI. Especifica a camada física e a camada MAC (Media Access Control) para dispositivos que não precisem de alta taxa de transmissão de dados e que necessitem de baixa latência e baixo custo de energia, tais como nas LR-WPANs (Low Rate Wireless Personal Area Network). [Messias 2008]

Foi criada e é mantida pela IEEE (Institute of Electrical and Electronics Engineers). Instituição que tem como finalidade definir padrões para desenvolvimento tecnológico.

O 802.15.4 é, portanto, um protocolo de pacote de dados para rede sem fio. Utiliza como canal de acesso ao meio o protocolo CSMA-CA (Carrier Sense Multiple Access - Collision Avoidance), que é o mesmo utilizado pelo padrão 802.11 (Wi-Fi) e visa a detecção e abstenção de colisão de pacotes. Além do CSMA-CA, o 802.15.4 ainda possui time slotting opcional, reconhecimento de mensagem e uma estrutura sinalizadora, chamada *beacon*. A segurança é feita em multicamadas e será abordada mais adiante neste documento.[de Carvalho & Cunha 2010]

2.1.1 Entendendo o padrão 802.15.4

Este protocolo reside no segundo nível do modelo OSI. Esta camada é denominada de Data Link. Nela, os bits contendo as unidades de informação digital são manuseados e organizados de forma a se tornarem impulsos eletromagnéticos para a camada física logo abaixo. É um padrão que se mostra com bom desempenho quando se trata de relação sinal/ruído e interferências. As frequências definidas no padrão são espalhadas através de

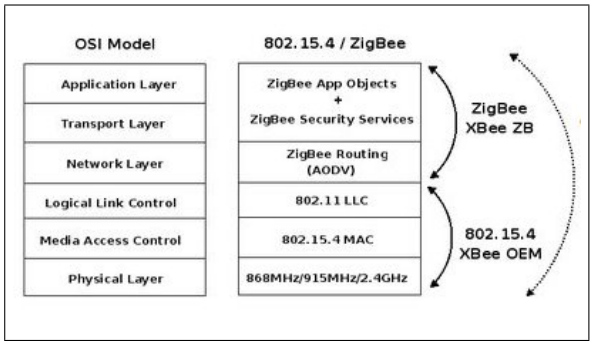

Figura 2.1: Localização do padrão no modelo OSI

27 canais diferentes divididas em três bandas principais: [Gascón 2009*a*]

- Europa - 868.0 à 868.6 MHz (1 canal)
- EEUU - 902.0 à 928.0 MHz (10 canais)
- Resto do mundo - 2.40 à 2.48 GHz (16 canais)

As taxas de dados são respectivamente:

- 868.0 à 868.6 MHz - 20/100/250 Kb/s
- 902.0 à 928.0 MHz - 40/250 Kb/s
- 2.40 à 2.48 GHz - 250 Kb/s

Por que o 802.15.4 é bom contra ruído?

O 802.15.4 utiliza DSSS (Direct Sequence Spread Spectrum) que é a sequência direta de espalhamento do espectro. É uma técnica de modulação do espectro de propagação. Utilizada extensamente em aplicações militares, fornece uma densidade espectral da potência muito baixa espalhando a potência do sinal sobre uma faixa de frequência muito larga. Consequentemente, este tipo de modulação requer uma largura de faixa muito grande para transmitir diversos Mbits/s.

O 802.15.4 utiliza a DSSS para modular a informação antes dela ser enviada para a camada física. Basicamente, cada bit de informação a ser transmitido é modulado em 4(quatro) diferentes canais. A característica dessa modulação traz como consequência menos interferência nas frequências de banda utilizadas e geram uma melhoria no SNR (Signal to Noise Ratio) no receptor devido ao fato de que é mais fácil detectar e decodificar a mensagem que esta sendo enviada pelo transmissor.

Existem diferentes modulações DSSS dependendo do tipo das limitações físicas do hardware do circuito e o número de símbolos que podem ser processados em um determinado tempo.

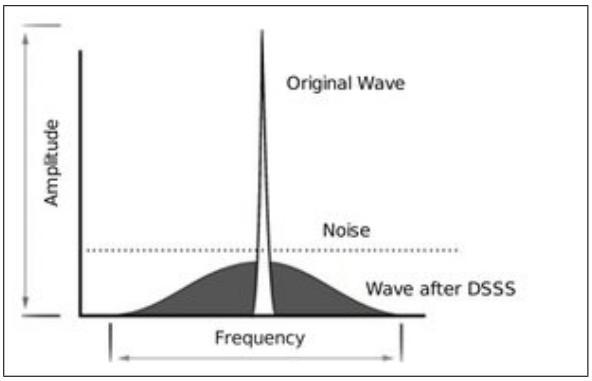

Figura 2.2: Exemplo de modulação de uma onda utilizando a técnica DSSS

Por que o 802.15.4 é bom contra interferências?

O padrão 802.15.4 utiliza duas técnicas para evitar que os pacotes comecem a ser transmitidos ao mesmo tempo. São elas o CSMA-CA e o GTS.

A mais comum é o CSMA-CA (Carrier Sense Multiple Access - Collision Avoidance). Nesse método, cada nó da rede escuta o meio para poder transmitir caso o meio esteja ocioso ou, nesse caso, em baixa energia. Se a energia encontrada é maior em um nível específico, o nó espera durante um tempo randômico e tenta novamente. Existe um parâmetro definido no padrão chamado de *macMinBE* que configura o expoente de *back-off* utilizado para calcular esse time-slot, que é o tempo de espera.

A segunda técnica, GTS (Guarantee Time Slot) utiliza um nó de rede centralizado denominado de coordenador PAN (Personal Area Network) que gera slot de tempo para cada nó para que assim eles saibam quando têm de transmitir. Existem 16 possíveis slots de tempo. Em um primeiro passo um nó deve enviar ao coordenador PAN uma mensagem de requisição GTS, como resposta o coordenador irá enviar uma mensagem de *beacon* contendo o slot alocado e o número de slots cadastrados.

Uma das funcionalidades implementadas no padrão 802.15.4 é o escaneamento de energia do canal (PLME-ED request). A ideia é permitir que se saiba o quanto de energia (atividade/ruído/interferências) existe em um ou mais canais em detrimento de utilizá-los. Desta forma pode-se economizar energia ao se escolher canais livres quando se configura a rede. Existem três diferentes escaneamentos de energia do meio possíveis:

- **Energia:** escaneia o canal e reporta a energia encontrada. Não interessa se é causada por uma tecnologia específica ou ruído. Apenas irá reportar se o espectro está sendo utilizado. Somente quando o valor recebido se encontrar abaixo de certo parâmetro de threshold, poderá ser realizada a transmissão.
- **Carrier Sense (CCA):** escaneia o meio e reporta se existem transmissões no padrão 802.15.4. Somente quando o canal estiver livre a transmissão será realizada.
- **CCA + Energia:** escaneia o meio e reporta se existem transmissões no padrão 802.15.4 acima do parâmetro de threshold especificado para então poder transmitir.

Protocolo de baixo consumo

O protocolo de baixo consumo tem por base o fato de o transceptor poder se encontrar em estado de hibernação (sleep) na maioria do tempo (99% em média) enquanto que as tarefas de recebimento e envio podem ser configuradas para tomar apenas uma pequena parte da energia do dispositivo. Essa porcentagem depende do tipo de modelo de comunicação utilizado. Se o modo *beacon* é utilizado (redes estrela ou PAN) o montante mínimo de tempo utilizado para transmitir/receber estes frames irá aumentar o tempo total com o qual o transceptor é utilizado.

Dispositivos da rede

O padrão 802.15.4 define dois tipos de dispositivos de rede. O primeira é denominado **Dispositivos de Função Completa (FFD)**. São dispositivos mais complexos e precisam de um hardware mais potente para a implantação da pilha de protocolos, consequentemente, consomem mais energia. Podem funcionar tanto como coordenador de uma rede quanto como um nó comum, seja ele roteador ou end device. São implementados em dispositivos com no mínimo 32KB de memória de programa e necessitam ter uma certa quantidade de memória RAM, para implementações de tabelas de rotas e configurações de parâmetros.

Por outro lado, existem os **Dispositivos de Função Reduzida (RFD)** que são dispositivos mais simples, onde sua pilha de protocolo pode ser implementada usando os mínimos recursos possíveis de hardware, como por exemplo, em microcontroladores de 8 bits com memória de programa próxima a 6KB, mas só podem se comunicar com dispositivos FFDs (Coordenador ou Roteador). Numa topologia de rede 802.15.4 eles assumem o papel de End Device (dispositivo final). Na prática podem ser: interruptores de iluminação, dimmers, controle de relês, sensores, entre outros.

Arquitetura de trasporte de dados

As unidades básicas de transporte de dados são denominadas frames, dos quais existem quatro tipos fundamentais: frames de dados (data frames), frames de reconhecimento, frames de *beacon* e frames de comando MAC. Estes tipos de frames proveem um meio termo racional entre simplicidade e robustez. Além do mais, uma estrutura denominada superframe, definida pelo coordenador, pode ser utilizada. Um superframe consiste de 16 slots de mesmo tamanho, que podem ser divididos futuramente em uma parte ativa e uma parte inativa onde o coordenador pode entrar em estado de economia de energia, sem necessidade de controlar sua rede.

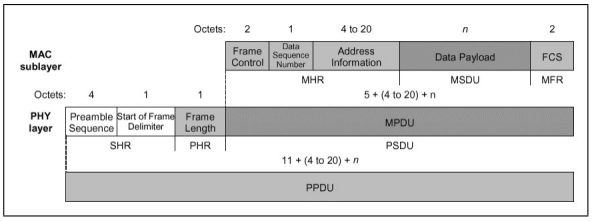

Figura 2.3: Estrutura do dataframe

A estrutura do frame de dados (data frame) pode ser vista na figura (2.3). Esta estrutura é uma das mais básicas e importantes do padrão 802.15.4. Tem capacidade de transferir até 104 bytes por pacote, o que já é o suficiente em se tratando de transferência de informações entre sensores. Para garantir a confiabilidade da entrega dos dados contém um campo com uma numeração sequencial dos dados e um campo de Frame Check Sequence (FCS).

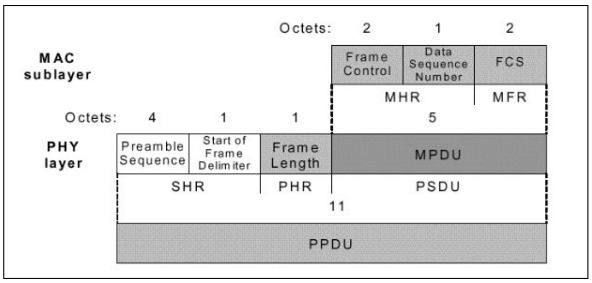

Figura 2.4: Estrutura do ACK Frame

No ACK Frame visto na figura 2.4, é enviado um retorno (ACK) eficaz do receptor para o emissor informando que o pacote foi recebido sem erros.

A figura (2.5) mostra o frame que representa um mecanismo para controle e configuração remota de nós da rede, permitindo que uma rede centralizada configure clientes

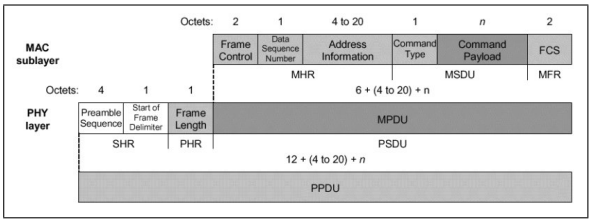

Figura 2.5: Estrutura do frame de comando MAC

individualmente sem importar o quanto a rede é extensa.

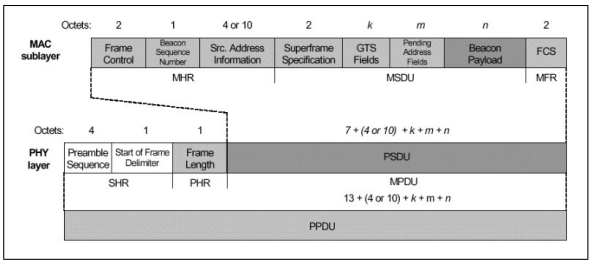

Figura 2.6: Estrutura do frame para o Beacon

Na figura (2.6) vemos uma estrutura opcional de sinalização. Os dispositivos podem *"acordar"* somente quando este sinal é transmitido, caso contrário, retornam a *"adormecer"*. É usado nas topologias de malha e estrela estendida para manter os nós sincronizados sem a necessidade deles consumirem energia por longos períodos de tempo.

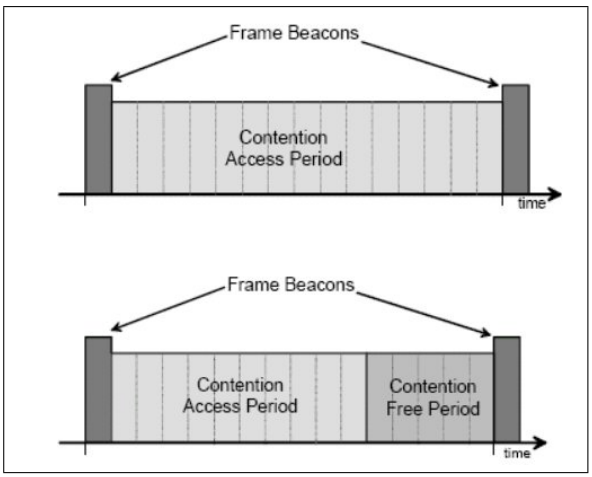

Figura 2.7: Estrutura do superframe

A estrutura de frame vista na figura 2.7 é uma estrutura opcional da camada MAC, ela é definida pelo nó coordenador, utiliza beacons para sinalização, slotting time e Guarantee

Time Slotting (GTS) para permitir que um nó continue utilizando o meio de acesso caso esteja mandando uma mensagem com alta prioridade.

A contenção de superframes ocorre entre seus limites, e é resolvida pelo CSMA-CA. Cada transmissão deve terminar antes da chegada do segundo *beacon*. Como mencionado anteriormente, aplicações com largura de banda bem definidas necessitam utilizar acima de sete domínios de um ou mais GTS (Guaranteed Time Slots) à direita no final do superframe. A primeira parte do superframe deve ser suficiente para fornecer serviço à estrutura de rede e seus dispositivos. Superframes são tipicamente utilizados no contexto de dispositivos de baixa latência, cujas associações devem ser mantidas mesmo em um período de tempo inativo ou de longa duração.

Transferência de dados para o coordenador requerem um sincronização de fase de *beacons*, se aplicável, seguida de uma transmissão CSMA-CA. O reconhecimento (acknowledgment) é opcional. Transferência de dados vindas do coordenador usualmente seguem as requisições do dispositivo: Se os *beacons* estão em uso, eles são utilizados para requisições de sinal; o coordenador reconhece a requisição e então envia os dados em pacotes que são reconhecidos pelo dispositivo. O mesmo é feito quando os superframes não estão em uso, somente neste caso não existem beacons para acompanhar as mensagens pendentes.

Redes ponto-a-ponto podem tanto utilizar o CSMA-CA sem slots ou mecanismos de sincronização; neste caso, a comunicação entre quaisquer dois dispositivos é possível, considerando que um dos dispositivos deve ser o coordenador da rede.

Em geral, todos os procedimentos implementados seguem uma arquitetura típica de classificação *requisição-confirmação/indicação-resposta*

2.2 O protocolo ZigBee

O ZigBee nada mais é do que um dos muitos protocolos que utilizam o padrão 802.15.4 em sua camada MAC. É com certeza o mais conhecido dos protocolos utilizados acima da camada física que utiliza o IEEE 802.15.4.[Gascón 2009*a*] A seguir buscaremos entender o funcionamento deste protocolo e suas implementações de segurança.

2.2.1 Entendendo o ZigBee

O ZigBee basicamente oferece quatro tipos diferentes de serviços:

- **Serviço de encriptação extra:** chaves de rede e aplicação implementam 128b estre de encriptação AES;

- **Associação e autenticação:** somente nós validados podem ingressar na rede;
- **Protocolo de roteamento:** *AODV*, um protocolo ad hoc reativo, tem sido implementado para realizar o roteamento de dados e processo de encaminhamento para qualquer nó na rede
- **Serviços de aplicações:** um termo abstrato denominado **"cluster"** é introduzido. Cada nó pertence a um cluster pré-definido e pode obter um pré-definido número de ações. Por exemplo: o *cluster do sistema de luz da casa* pode realizar duas ações; ligar a luz e desligar a luz.

O ZigBee é uma camada pensada para organizar a rede. A primeira coisa que um nó, seja ele roteador ou dispositivo final, que deseje ingressar na rede deve fazer é perguntar ao coordenador por um endereço de rede (**16 bits**), como parte do processo de associação. Toda a informação na rede é roteada utilizando esse endereço e não o endereço de 64 bits da camada MAC. Neste passo, processos de autenticação e encriptação são realizados.

Uma vez que um nó tenha ingressado na rede, ele pode enviar informações para os seus "irmãos"através dos roteadores, os quais estão sempre acordados à espera de pacotes. Quando o roteador recebe o pacote e o destino está no seu sinal de rádio, o roteador da uma primeira olhada e verifica se o destino final está acordado ou dormindo. Se estiver acordado, o roteador envia o pacote para o destino final, entretanto, em caso contrário, o roteador irá bufferizar o pacote até que o dispositivo final acorde e pergunte por um novo roteador.

2.2.2 Como trabalha uma rede ZigBee

No protocolo ZigBee existem três classes de dispositivos lógicos (Coordenador, Roteador e Dispositivo final) que definem a rede:

- **Coordenador:** Só pode ser implementado através de um dispositivo FFD. O coordenador é responsável pela inicialização, distribuição de endereços, manutenção da rede, reconhecimento de todos os nós, entre outras funções podendo servir como ponte entre várias outras redes que utilizem o mesmo padrão.
- **Roteador:** Só pode ser implementado através de um dispositivo FFD. Tem as características de um nó normal na rede, mas com poderes extras de também exercer a função de roteador intermediário entre nós, sem precisar do coordenador. Por intermédio de um roteador uma rede 802.15.4 pode ser expandida, e assim ter mais alcance. Na prática um roteador pode ser usado para amplificar o sinal da rede entre andares de um prédio.

- **Dispositivo Final (End Device):** É onde os atuadores ou sensores serão hospedados. Pode ser implementado através de um dos dispositivos FFD ou RFD. Assim ele é o nó que consome menos energia, pois na maioria das vezes ele fica em modo de hibernação (Sleep).

A redes podem ser construídas em topologias tanto ponto-a-ponto quanto estrela. Entretanto, toda rede necessita de ao menos um dispositivo FFD para funcionar como coordenador da rede. Essas redes são, portanto, formadas por grupos de dispositivos separados por uma determinada distância. Cada dispositivo tem um identificador único de 64 bits.[de Carvalho & Cunha 2010]

Redes ponto-a-ponto podem formar padrões arbitrários de conexões entre dispositivos, e suas extensões são limitadas apenas pela distância entre cada par de nós. Foram destinados para servir de base para redes ad hoc capazes de realizar autogestão e organização. Uma vez que o padrão não defina uma camada de rede, o roteamento não é diretamente suportado, mas uma camada adicional pode adicionar suporte para uma comunicação *multihop*. Algumas restrições topológicas podem ser observadas: o padrão menciona *cluster free* como uma estrutura que pode explorar o fato de que um dispositivo **RFD** pode apenas se associar com um dispositivo **FFD**, por vez para formar uma rede onde os dispositivos RFDs são exclusivamente as folhas de uma árvore na topologia e a maioria dos nós são textbfFFD; A estrutura também pode ser estendida como uma rede *mesh* genérica onde o nós são aglomerados em árvore da rede com um coordenador local para cada aglomeração, em adição ao coordenador global; um padrão mais estruturado seria a topologia em formato estrela, onde o coordenador da rede irá necessariamente ser o nó central. Tal rede pode ser originada quando um dispositivo **FFD** decide criar a sua própria Rede de Área Pessoal (**PAN - Personal Area Network**) e se declara coordenador, após escolher um único identificador **PAN**. Após isso, outros dispositivos podem se juntar a rede, os quais serão totalmente independentes de todas as outras redes estrela. Em consequência às restrições topológicas acima descritas, uma rede no padrão 802.15.4 aceita basicamente as seguintes topologias:

- **Malha ou Ponto-a-Ponto:** Na topologia em Malha a rede pode se ajustar automaticamente, tanto na sua inicialização como na entrada ou saídas de dispositivos na Rede. A Rede se organiza para otimizar o trafego de dados. Com vários caminhos possíveis para a comunicação entre os nós, este tipo de rede pode abranger em extensão, uma longa área geográfica, podendo ser implementada numa fábrica com vários galpões distantes; controle de irrigação ou mesmo num prédio com vários andares.

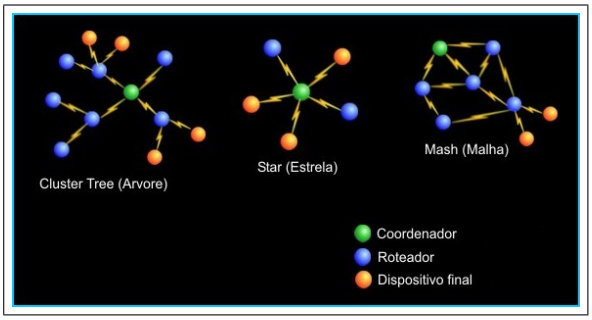

Figura 2.8: Topologias do protocolo ZigBee

- **Árvore (Cluster Tree):** Semelhante à topologia de Malha, uma rede em árvore, tem uma hierarquia muito maior e o coordenador assume o papel de nó mestre para a troca de informação entre os nós roteadores e finais.
- **Estrela:** É uma das topologias de rede mais simples de serem implantadas, é composta de um nó Coordenador, e quantos nós End Device forem precisos. Este tipo de rede deve ser instalada em locais com poucos obstáculos à transmissão e recepção dos sinais, como por exemplo, em uma sala sem muitas paredes ou locais abertos.

2.2.3 ZigBee vs ZigBee Pro

Em 2007, foi lançada uma nova versão do protocolo ZigBee denominada ZigBee Pro. A seguir serão estabelecidas as principais diferenças entre o protocolo ZigBee de 2006 e esse novo modelo.[Gascón 2009*c*]

- **Endereçamento Estocástico:** Na primeira implementação do ZigBee, o endereço era escolhido pelo coordenador de acordo com a posição do nó na árvore de rede. Agora o endereço de 16 bits é escolhido de forma randômica. Se os nós escolherem o mesmo endereço, então o endereço é resolvido através do padrão da camada MAC de 64 bits do IEEE 802.15.4;
- **Malha de gerenciamento de dados:** No ZigBee convencional, cada nó tinha que manter uma tabela de qualquer uma das rotas de e para o gateway para um dispositivo (se estivesse no caminho de rotas), agora os nós apenas salvam o caminho até chegarem no gateway (reduzindo o espaço de memória necessário). O gateway (um nó que suporta maiores recursos de memória RAM) guarda a rota (com todos os pulos) para algum nó. Quando o gateway tem que enviar um pacote para um nó específico, ele acrescenta a informação a respeito dos passos que tem que ser dados no mesmo pacote. Este procedimento é denominado /textbftextitmuitos para um

- **Fragmentação:** Pacotes de dados extensos podem ser facilmente fragmentados.
- **Escolha dinâmica do melhor canal:** Os nós irão mudar de canal se o canal em que estão possuir interferências ou ruído seguindo um parâmetro de threshold.
- **Conexões assimétricas:** Os links entre os nós nem sempre são simétricos e a qualidade da conexão é diferente de um nó A para um nó B do que de B para A, isto é devido a várias razões que incluem interferências entre canais e ruídos. Por esse motivo a versão PRO tenta levar isso em conta para construir as melhores rotas possíveis.
- **Segurança:** Na versão de 2006, a implementação do ZigBee utilizava de 128 bits acima do AES e uma chave de rede global para criar comunicações seguras. A nova versão tem um sistema mais complexo que permite que cada par de nós tenha sua própria chave para que assim a encriptação p2p possa ser realizada. Uma camada peer-to-peer de link de encriptação é adicionada.

2.2.4 802.15.4 vs ZigBee

Resumidamente, neste ponto do texto, podemos estabelecer algumas comparações entre o padrão IEEE 802.15.4 e o padrão de protocolo ZigBee:

- 802.15.4 foi idealizado para ser um protocolo que estabelece comunicações ponto-a-ponto com eficiência de energia.
- ZigBee define serviços extras (roteamento em topologia estrela, encriptação, serviços de aplicação) acima da camada 802.15.4.
- ZigBee cria redes semi-centralizadas aonde apenas o dispositivo final pode ficar em estado de hibernação (sleep).

2.3 Segurança no padrão IEEE 802.15.4 e protocolo ZigBee

Como visto anteriormente neste trabalho, a camada MAC do padrão IEEE 802.15.4 implementa alguns recursos que são utilizados pelo protocolo ZigBee nas camadas de rede e aplicação. Um desses recursos são os serviços de segurança.

IEEE 802.15.4 define o algoritmo de encriptação para ser usado quando os dados forem cifrados para serem transmitidos, no entanto, o padrão não especifica como as chaves têm de ser gerenciadas ou que tipos de políticas de autenticação têm de ser aplicadas. Es-

ses problemas são tratados nas camadas superiores que são gerenciadas por tecnologias tais como o ZigBee.[Gascón 2009*b*]

2.3.1 Um visão geral da segurança no IEEE 802.15.4

O algorítmo de encriptação utilizado é o **AES (Advanced Encryption Standard)** com uma chave de tamanho de 128 bits (16 bytes). É importante contar com um único tipo de método de encriptação devido ao fato de que a maioria dos transceptores 802.15.4/ZigBee têm um design específico de hardware para lidar com esse trabalho em nível eletrônico (dispositivos de baixo recursos embutidos).

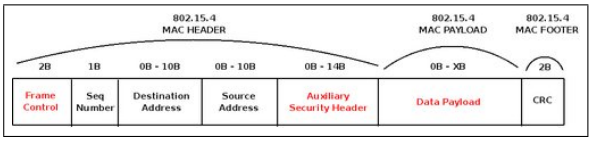

Figura 2.9: Quadro MAC do padrão IEEE 802.15.4

O algoritmo AES não é somente utilizado para encriptar a informação, mas também para validar os dados que estão sendo enviados. Este conceito é denominado de Integridade de Dados e é alcançado utilizando um Código de Integridade de Mensagem, (em inglês MIC) que é adicionado à mensagem. Este código garante integridade no cabeçalho MAC e carga de dados anexada. É criado encriptando partes do frame MAC do IEEE utilizando a chave da rede, assim se for recebida uma mensagem de um nó não confiável, poderá ser analisado que o MIC gerado para a mensagem enviada não corresponde àquele que seria gerado utilizando a mensagem com a chave secreta corrente, e então a mensagem é descartada. O MIC ou MAC (Message Authenticantion Code) pode ter tamanhos distintos: 32, 64, 128 bits, entretanto é sempre criado utilizando o algoritmo AES de 128 bits. Seu tamanho é apenas a quantidade de bits que são anexadas a cada frame. Quanto maior, mais seguro (apesar da menor carga de dados que a mensagem pode ter). A segurança de dados é realizada pela encriptação do campo de carga dados (payload) com a chave de 128 bits.

2.3.2 Insights de segurança do 802.15.4

Existem três campos no frame MAC do padrão IEEE 802.15.4 que são relacionados com a segurança:

- **Frame control:** localizado no cabeçalho MAC

- **Controle de segurança auxiliar:** localizado no cabeçalho MAC
- **Carga(payload) de Dados:** localizada no campo de carga do MAC

O frame de **Controle de Segurança Auxiliar** é habilitado somente se o subcampo de **Segurança Habilitada** do **Frame de controle** estiver ligado. Este cabeçalho especial tem 3 campos:

- **Controle de Segurança (1B):** especifica que tipo de proteção é utilizada.
- **Contador de Frame: (4B)** é um contador fornecido pela fonte do frame atual para proteger a mensagem contra repetição de proteção. Por esta razão cada mensagem tem um único ID de sequência representada por este campo.
- **Identificador de chave (0-9B):** especifica a informação necessária para saber que chave nós estamos usando com o nó que estamos nos comunicando.

O controle de segurança é o local onde a nossa política de segurança global é configurada. Utilizando os dois primeiros bits (campos de nível de segurança) escolhe-se o que será encriptado e quão longa a chave será:

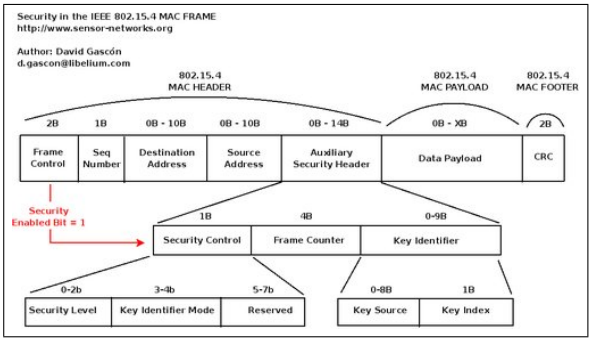

Figura 2.10: Segurança na camada MAC do padrão IEEE 802.15.4

Tabela 2.1: Tabela com as respectivas chaves para o tipo de encriptação de acordo com os bits do nível de segurança

0x00	Sem segurança		Dados não encriptados, autenticidade dos dados não validada
0x01	AES-CBC-MAC-32	MIC-32	Dados não encriptados, autenticidade dos dados validada
0x02	AES-CBC-MAC-64	MIC-64	Dados não encriptados, autenticidade dos dados validada
0x03	AES-CBC-MAC-128	MIC-128	Dados não encriptados, autenticidade dos dados validada
0x04	AES-CTR	ENC	Dados encriptados, autenticidade dos dados não validada
0x05	AES-CCM-32	AES-CCM-32	Dados encriptados, autenticidade dos dados validada
0x06	AES-CCM-64	AES-CCM-64	Dados encriptados, autenticidade dos dados validada
0x07	AES-CCM-128	AES-CCM-128	Dados encriptados, autenticidade dos dados validada

O valor **0x00** configura a não encriptação, então nem os dados são encriptados (sem confidencialidade de dados) nem a autenticidade é validada. De **0x01** à **0x03** os dados são autenticados utilizando a mensagem de autenticação de código (MAC) encriptada. O valor **0x04** encripta a carga de dados, certificando a confidencialidade de dados. Os valores de **0x05** à **0x07** certificam que os dados tenham confidencialidade e autenticidade. O subcampo do modo de identificação de chave configura o tipo (implícito ou explícito) que a chave deve ser utilizada pelo destinatário e o remetente. Possíveis valores são:

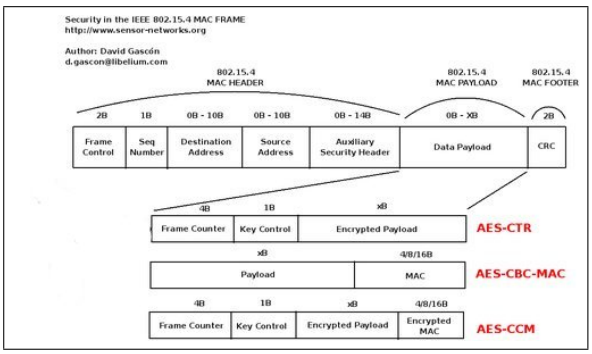

Figura 2.11: Segurança na camada MAC do padrão IEEE 802.15.4

- **0:** O ID da chave é implícita para o remetente e destinatário (não é especificada na mensagem).
- **1:** O ID da chave é determinada explicitamente pelo index de chave de 1 Byte vindo do campo identificador de chave e do **macDefaultKeySource**.
- **2:** O ID da chave é determinado explicitamente pelo index de chave de 1 Byte e os 4 Bytes da fonte de chave (Key Source).
- **2:** O ID da chave é determinado explicitamente pelo index de chave de 1 Byte e os 8 Bytes da fonte de chave (Key Source).

Como mencionado anteriormente, o campo **identificador de chave** é configurado quando o subcampo de modo de identificador de chave é diferente de zero. O subcampo fonte da chave (key source) especifica quem originou o grupo de chave. O index da chave (Subcampo de controle de chave) ajuda a identificar chaves diferentes das chaves de uma fonte de chave (key source) específica

O campo de carga de dados (payload data) pode ter três diferentes configurações dos campos de segurança previamente definidos:

- **AES-CTR:** Todos os dados são encriptados utilizando a chave definida de 128 bits e o algoritmo AES. O contador de frame configura uma única ID de mensagem, e o

contador de chave (key counter) no subcampo de controle de chave é utilizado pela camada de aplicação se o valor máximo do frame counter é atingido.

- **AES-CBC-MAC:** O **MAC** (Código de autenticidade de mensagem) é anexado ao final da carga de dados (data payload). Seu tamanho depende do nível de segurança especificado no campo de política de segurança (Security Policy). O MAC é criado encriptando informação do cabeçalho MAC do 802.15.4 e da carga de dados.
- **AES-CCM:** É a mistura dos métodos definidos anteriormente. Os subcampos correspondem com o modo **AES-CTR** mais o subcampo extra do **AES-CBC-MAC** encriptado.

2.3.3 A lista de controle de acesso

Cada transceptor 802.15.4 tem q gerenciar uma lista de controle para os seus *"irmãos confiáveis"* juntamente com a política de segurança. Por esta razão, cada nó tem que controlar sua própria **Lista de Controle de Acesso (ACL)** que guarda os seguintes campos:

- **Endereço (Address):** Endereço do nó que se deseja comunicar.
- **Suite de segurança:** A política de segurança que está sendo utilizada (AEC-CTR, AES-CCM-64, AES-CCM-128,,...).
- **Chave:** A chave de 128 bits utilizada no algoritmo AES.
- **Último vetor inicial(IV) e contador de repetição:** Ambos estão no mesmo campo. O último IV é utilizado pela fonte e o contador de repetição pelo destino como ID de mensagem em função de se evitar ataques repetidos.

Quando um nó quer enviar uma mensagem para um nó específico ou recebe um pacote, ele irá procurar na **ACL** para verificar se o nó é um **irmão confiável** ou não. Se for, o nó utilizará o dado contido na coluna específica para aplicar as medidas de segurança. Caso o nó não esteja na lista ou sua mensagem é rejeitada ou um processo de autenticação se dará início.

2.3.4 Segurança ZigBee

O ZigBee implementa duas camadas extras de segurança acima do padrão 802.15.4: as camadas de segurança de **rede** e **aplicação**. Todas as políticas de segurança confiam na encriptação do algoritmo AES de 128 bits, assim a arquitetura de hardware previamente implementada para o nível de link (camada MAC) é ainda válida. Existem três tipos de chave: master, link e de rede.

- **Master Keys:** São pré-instaladas em cada nó. Sua função é manter confidencial a troca de **Chaves de Link** entre dois nós no **Processo de Estabelecimento de Chave (SKKE)**.
- **Chaves de Link:** São únicas entre cada par de nós. Essas chaves são gerenciadas pelo nível de aplicação. São utilizadas para encriptar toda a informação entre cada dois dispositivos, por essa razão mais recursos de memória são necessários em cada dispositivo. Geralmente essa chave não costuma ser usada.
- **Chaves de Rede:** É uma chave única de 128 bits compartilhada ao longo dos dispositivos na rede. É gerado por um centro de confiança e re-gerada em diferentes intervalos. Casa nó precisa pegar sua chave de rede para ingressar em uma rede. Uma vez que o centro de confiança decida mudar a chave de rede, a nova chave é espalhada na rede utilizando a antiga chave de rede. Uma vez que essa nova chave é atualizada em um dispositivo, seu contador de frame é inicializado em zero. Este centro de confiança é normalmente o coordenador da rede, entretanto, pode ser que seja um dispositivo dedicado. Ele tem apenas que autenticar e validar cada dispositivo que tenta entrar na rede.

Cada par de dispositivos podem ter configurados tanto as chaves de rede quanto as de link. Nesse caso a chave de link é sempre utilizada (mais segurança, mais memória é necessária). Existem dois tipos de política de segurança que o centro de confiança pode seguir:

- **Modo Comercial:** O centro de confiança compartilha as chaves **master** e de **link** com qualquer dispositivo na rede. Este modo requer alto recurso de memória. Oferece um modelo completo e centralizado para controle de segurança de chave.
- **Modo Residencial:** O centro de confiança compartilha apenas a **chave de rede** (é o modo ideal quando dispositivos embarcados tem de lidar com esta tarefa devido aos baixo recursos que eles tem). Este é o modo normalmente escolhido para o modelo de redes de sensores sem fio.

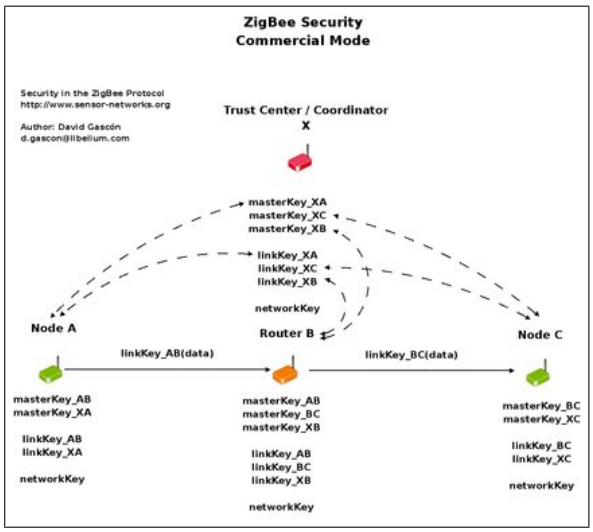

Figura 2.12: Política de segurança do modo comercial

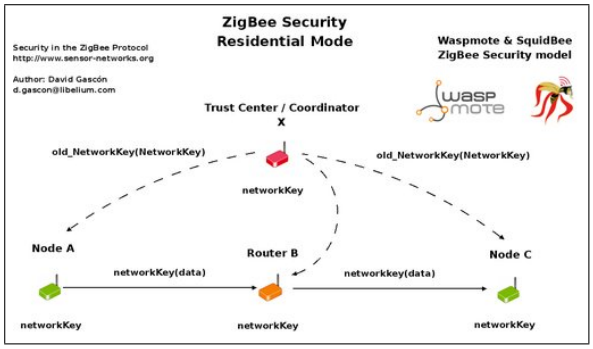

Figura 2.13: Política de segurança do modo residencial

Capítulo 3

O microcontrolador e o módulo sonar

3.1 O microcontrolador PIC16F628A

Pode-se dizer que um microcontrolador é um componente eletrônico dotado de uma inteligência programável utilizado no controle de processos lógicos [de Souza 2006].

O controle de processos é o controle de periféricos, tais como LEDs, botões e, no caso deste trabalho, sensores. São chamados de controles lógicos pois a operação do sistema baseia-se nas ações lógicas que demandam de execução de acordo com o estado das entradas e saídas.

O microcontrolador é programável, pois toda a lógica de operação é estruturada na forma de um programa e gravada dentro do componente. Após o programa ter sido gravado, toda vez que o microcontrolador for alimentado, o programa interno será executado.

3.1.1 O PIC16F628A

Para este projeto foi utilizado o PIC16F628A por ser um microcontrolador versátil, compacto e poderoso. Algumas características desse PIC são:

- Microcontrolador de 18 pinos, que facilita a montagem de hardwares experimentais;
- Possui até 16 portas configuráveis como entrada ou saída e dois osciladores internos (4 MHz e 37 kHz);
- 10 interrupções disponíveis (Timers, Externa, Mudança de Estado, EEPROM, USART, CCP e comparador);
- Memória de programação FLASH com 2.048 palavras, que permite a gravação do programa diversas vezes no mesmo chip, sem a necessidade de apaga-lo por meio de luz ultravioleta, como acontece nos microcontroladores de janela;
- Memória EEPROM (não-volátil) interna com 128 bytes;

- Recursos adicionais avançados: módulo CCP, Comparador interno e USART;
- Programação com 14 bits e 35 instruções

Nomenclaturas utilizadas

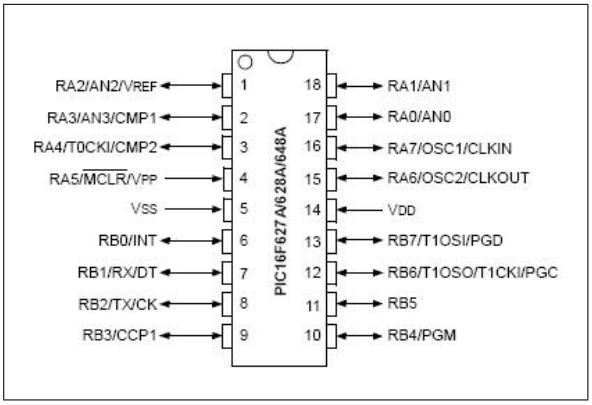

Figura 3.1: Pinagem do PIC16F628A

O PIC16F628A possui um total de 16 I/O separados em dois grupos cuja a denominação atende pelo nome de PORTA e PORTB.

O PORTA possui oito pinos que podem ser configurados como entrada ou saída e seus nomes são definidos como RA0, RA1, RA2, RA3, RA4, RA5, RA6 e RA7. Alguns pinos tem múltiplas funções e para utilizar uma determinada função é necessário abdicar de outra, por exemplo, para se utilizar o pino RA5, perdemos o MCLR externo.

O PORTB também possui oito pinos configuráveis como entrada ou saída indo de RB0 a RB7. O RB0 pode ser utilizado também para gerar interrupção externa.

Para que o microcontrolador possa funcionar, é necessário que o mesmo seja alimentado nos pinos 5 (Vss/GND) e 14 (Vdd/+5Vcc). A tensão nominal para alimentação é de 5Vcc, mas o alcance de variação desta tensão depende do modelo utilizado. No caso do PIC16F628A, ela vai de 2.0 a 5.5Vcc.

Um pino denominado MCLR (barrado) se refere ao Master Clear externo. Para que o PIC funcione, este pino tem que se encontrar em nível lógico baixo (GND). Caso contrário o programa será resetado.

Opções do PIC

Antes de gravar o código propriamente dito com a lógica desejada no PIC, é necessário verificar e configurar os bits que definem as opções de gravação do mesmo. Esses bits

podem ser configurados tanto em programas que fazem interface com o gravador de PIC utilizado quanto no próprio código, seja ele em *assembler* ou *C*.

Os parâmetros de configuração para o PIC16F628A são:

- **Tipo de oscilador:** Existem dois grupos de osciladores para uso com o PIC16F628A: internos e externos. O PIC em questão possui dois osciladores internos (37 kHz e 4 MHz) e capacidade de operação com vários osciladores externos. As opções disponíveis para configurar o oscilador desejado são:
  - **RC_CLKOUT:** Para oscilador externo tipo RC com o pino 15 operando como CLKOUT, ou seja, uma onda quadrada de um quarto da frequência.
  - **RC_IO:** Para oscilador externo tipo RC com o pino 15 operando como I/O (RA6).
  - **INTOSC_CLKOUT:** Para oscilador interno com o pino 15 operando CLKOUT, ou seja, onda quadrada de um quarto da frequência.
  - **INTOSC_I/O:** Para oscilador interno com o pino 15 operando como I/O (RA6).
  - **EC_IO:** Para clock externo (circuito auto oscilante) com o pino 15 operando como I/O (RA6).
  - **XT:** Para osciladores externos tipo cristal ou ressoadores.
  - **HS:** Para cristais ressoadores externos com frequências elevadas (acima de 4 MHz).
  - **LP:** Para cristais ou ressoadores externos com baixas frequências (abaixo de 200 KHz). Utilizado para minimizar consumo.
- **WATCHDOG TIMER:** O WDT também pode ser ativado ou não na hora da gravação, e esta configuração não poderá ser alterada posteriormente pelo programa. Esse parâmetro é um controle implementado para o caso do software gravado no PIC travar ou entrar em loop infinito por alguma razão. Após um certo intervalo de tempo se ele não for resetado pelo software e o timer dele chegar ao fim, o PIC será reinicializado.
- **POWER UP TIMER:** O *power up timer* interno pode ser habilitado ou não na hora da gravação. Esta operação faz com que o PIC só comece a operar cerca de 72 ms após o pino MCLR ser colocado em nível alto.
- **BROWN OUT DETECT:** Trata-se de um sistema de detecção automática de baixa tensão capaz de resetar o PIC. Isso significa que, se a tensão de alimentação for menor do que 4V (típico) por mais do que 100 μs, o sistema será reinicializado.
- **MASTER CLEAR ENABLE:** Esta é a opção que define o uso do pino 4, que

pode ser I/O ou *Master Clear* externo (MCLR). Ao habilitar esta opção, o pino 4 funciona como MCLR.

- **LOW VOLTAGE PROGRAM:** Um recurso relativamente novo para muitos modelos de PIC. Trata-se do sistema de programação do PIC em baixa tensão (5V). Normalmente essa programação por uma alta tensão (13 V) no pino MCLR. Acontece que hoje é possível criarmos sistemas onde um PIC possa gravar o programa de outro PIC, ou então efetuarmos um upgrade remoto. Quando habilitada esta opção, o pino 10 deixa de ser o RB4 e passa a operar como PGM.
- **DATA EE READ PROTECT:** Com esta opção ativada, não será possível ler a EEPROM interna antes do gravador de PIC. Durante o desenvolvimento, ou até em alguns tipos de projeto, essa leitura pode ser útil para encontrar erros e solucionar problemas.
- **CÓDIGO DE PROTEÇÃO:** É muito importante que esta opção esteja ativada quando se deseja gravar em série, pois isso impedirá que qualquer pessoa consiga ler o programa gravado dentro do PIC. É a única maneira de proteger seu sistema contra cópia indevida. Esse parâmetro impede que se leia a memória mas não impede que se grave outros programas por cima do anterior.

3.2 O módulo sonar

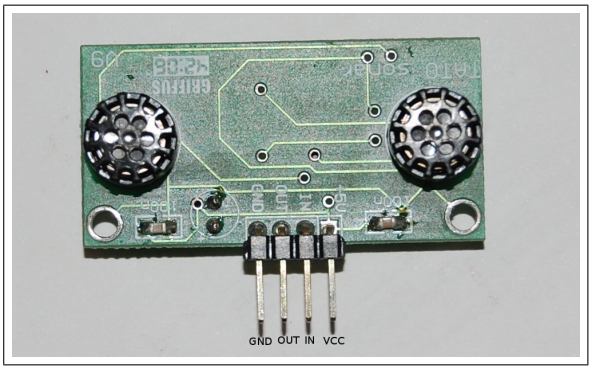

Figura 3.2: Módulo sonar da Tato Equipamentos Eletrônicos

A medição de distâncias é um problema que diversas áreas encontram, sejam elas industriais ou o segmento de consumo. Para resolver esse problema, a tecnologia de ultrassom é uma das tecnologias usadas pelas indústrias. Entretanto, um balanço entre custo e benefícios é importante. A medição de distâncias por ultrassom é normalmente utilizada quando uma medição sem contato é necessária.[Eletrônicos 2010]

3.2.1 O som

O som é uma vibração mecânica transmitida por um meio elástico. A faixa de frequências que o ser humano consegue ouvir varia aproximadamente de 20Hz a 20.000Hz. Esta faixa é por definição o espectro audível e varia de pessoa para pessoa e, geralmente, diminui com a idade. O ouvido é mais sensível a frequências em torno de 3.500Hz. Sons acima de 20.000Hz são conhecidos como ultrassom e sons abaixo de 20 Hz como infrassom.

3.2.2 Velocidade do som

A velocidade do som depende do meio que ele atravessa. Em geral, a velocidade do som é proporcional (a raiz quadrada da diferença) da "dureza"do meio e sua densidade. Esta é uma propriedade fundamental do meio. As propriedades físicas e a velocidade do som mudam de acordo com as condições do ambiente. A velocidade do som no ar depende da temperatura. No ar, a velocidade é de aproximadamente 345 m/s, na água de 1500 m/s e em uma barra de aço, de 5000 m/s.

Um uso comum do ultrasom é para medição de distâncias, isto também é chamado de sonar. Sonar trabalha de um modo similar ao radar. Um pulso ultrasônico é gerado em uma direção. Se existir um objeto no caminho deste pulso, o pulso é refletido de volta para o emissor como um eco, e é detectado. Medindo a diferença de tempo entre a emissão do pulso e a recepção do eco, é possível determinar a distância do objeto.

3.2.3 Reflexão do som

Para medir a distância que o som percorreu, ele precisa ser refletido de volta. Este som é uma onda transversal que bate em uma superfície plana. O som é então refletido, desde que as dimensões do objeto sejam grandes, comparado com o comprimento da onda.

Alguns parâmetros a serem considerados:

- **Superfície:** A superfície ideal do objeto é dura e lisa. Esta superfície reflete uma quantidade maior do sinal do que uma macia e porosa. Um eco fraco é resultado de um objeto pequeno ou macio. Isto reduz a distância de operação do sensor e reduz a sua precisão.
- **Distância:** Quanto menor for a distância do sensor ao objeto, mais forte será o eco. Deste modo, a medida que a distância aumenta, o objeto precisa ter melhores características de reflexão para retornar um eco suficiente.
- **Tamanho:** Um objeto grande tem mais superfície para refletir o sinal do que um menor. A área de superfície reconhecida como alvo é a área mais próxima ao sensor.

- **Ângulo:** A inclinação da superfície do objeto em relação ao sensor afeta o modo que o objeto reflete a onda. A porção perpendicular ao sensor retorna o eco. Se o objeto todo estiver a um ângulo grande, o sinal é então refletido para longe do sensor e o eco não é detectado.

3.2.4 Funcionamento básico do sensor

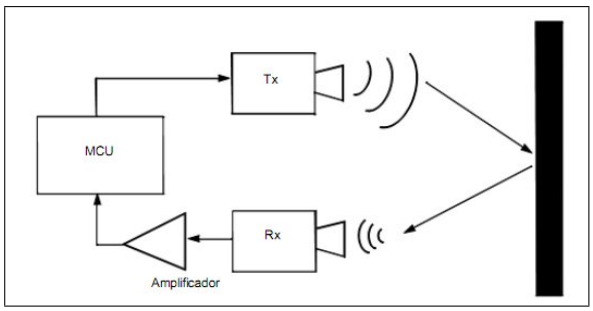

Figura 3.3: Esquema do funcionamento básico do sonar

O firmware do microcontrolador gera um trem de pulsos de 40 kHz. Depois que 10 pulsos são gerados, uma variável que mede o tempo é ativada. Esta variável guarda o tempo que o pulso leva até retornar e é usada para o cálculo da distância. Quando a onda é refletida pelo objeto, ela é capturada pelo receptor (Rx). Este sinal recebido é amplificado pelo amplificador, pois ela sofre uma atenuação no ar. Depois disto, o sinal retorna para o microcontrolador (MCU), onde é filtrado e usado para o cálculo da distância.

O sensor emite um tipo de sinal de saída chamado Eco, que é um pulso em nível alto correspondente ao tempo de ida e volta do som, ou seja, é necessário dividir este tempo por 2 para obter o cálculo da distância.

Capítulo 4

Implementação do projeto

Nos capítulos anteriores foram vistos alguns conceitos e especificações que abrangem a teoria das tecnologias que serão empregadas na implementação do projeto propriamente dito. Este capítulo irá descrever como o projeto foi desenvolvido, bem como discorrerá sobre como as tecnologias envolvidas e descritas anteriormente foram utilizadas em conjunto com o objetivo de chegar a um produto final, com resultados relevantes para os fins do presente trabalho.

A ideia por trás do projeto é fornecer um ambiente em que o usuário do estacionamento possa ter à sua disposição a opção de verificar visualmente se uma vaga está ou não livre através dos sinais luminosos (LEDs). Também é oferecida a possibilidade do usuário solicitar uma reserva de vaga. Após essa reserva o sistema procura uma vaga que esteja livre e a deixa reservada por um determinado intervalo de tempo à espera do usuário para que o mesmo possa estacionar o seu veículo. Se após esse intervalo de tempo nenhum veículo ocupar a vaga, o sistema volta ao seu funcionamento normal, liberando a vaga e esperando um veículo estacionar ou uma nova solicitação de reserva de vaga.

O projeto, portanto, consiste em elaborar um circuito transmissor/receptor que irá utilizar um módulo ZigBee acoplado ao circuito para transmissão e recepção de dados sem fio. Um sensor sonar que irá medir a distância e detectar se um obstáculo está ou não presente em uma determinada vaga e um microcontrolador que conterá a lógica responsável por processar os dados fornecidos pelo sonar e pelo ZigBee. Outro módulo ZigBee configurado como base ficará conectado a um computador e receberá os dados do módulo acoplado no circuito em que o sensor está presente. Na base receptora um software desenvolvido em linguagem Java TM irá gerenciar a vaga e estabelecer meio de contato entre os dois dispositivos.

A seguir, todo o processo que possibilita a transmissão, recepção e processamento dos dados serão detalhados passo a passo e individualmente. Ao final serão apresentados alguns resultados obtidos.

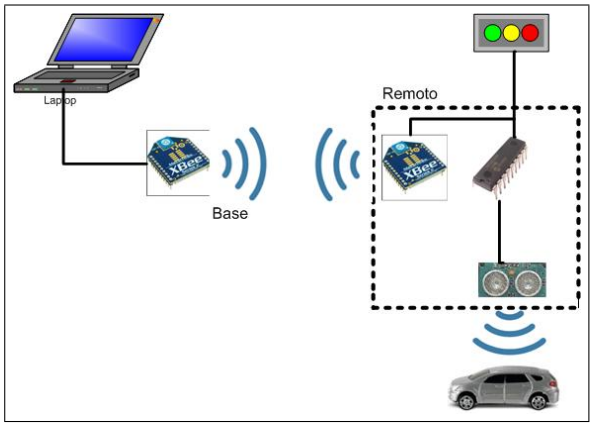

Figura 4.1: Esquema geral do modelo de estacionamento.

4.1 A lógica do PIC16F628A

O PIC16F628A será o meio de comunicação entre o sensor e o ZigBee como pôde ser visto no esquema geral do modelo na figura (4.1). Para realizar tal tarefa é necessário que o PIC contenha em sua programação uma lógica para tratar o sinal fornecido pela saída do sensor e, através desse processamento, fornecer como resposta o sinal para acender o LED respectivo. O PIC também receberá como entrada um sinal proveniente do módulo ZigBee no circuito, que por sua vez foi recebido do módulo base e corresponde a uma requisição de reserva de vaga.

Temos então a seguinte configuração para o PIC. Os pinos RA1, RA2 e RB0 são configurados como entradas. O pino RA1 é responsável por ativar o modo de reserva de vaga. Quando ele se encontrar em estado lógico alto, o LED amarelo, que indica reserva, ficará aceso e só apagara quando um obstáculo se encontrar no alcance definido como sendo de uma vaga ocupada ou quando a entrada RA1 receber nível lógico baixo.

O pino RA2 fica responsável por calibrar o sensor para uma distância específica. Quando um botão (push button) a ele relacionado é pressionado o tamanho do comprimento de onda do pulso do sensor naquele instante será armazenado em uma variável de referência e qualquer valor desse comprimento desse momento em diante será comparado com o valor de referência para decidir qual LED deve ser ativado.

O pino RB0 é conhecido como sendo o pino de interrupção externa do PIC. Quando ele é configurado como entrada e, mais especificamente para o projeto em questão, quando definido que a interrupção acontecerá toda vez que o nível baixo da entrada passar para o nível alto.

A interrupção pausa o código principal aonde quer que ele esteja sendo executado e

executa o código definido pelo desenvolvedor para tratar da interrupção. No caso deste projeto, a interrupção será responsável por medir o tamanho do comprimento de onda do pulso gerado pelo sensor.

Como visto anteriormente, o sensor retorna um pulso de onda quadrada com largura definida em milissegundos correspondente ao tempo que a onda leva para ir até um obstáculo e voltar. Esse tempo é o mesmo tempo em que a entrada RB0 ficará em estado lógico alto. Tendo em vista que a interrupção ocorreu quando a entrada RB0 estava em nível lógico baixo e passou para nível lógico alto e a interrupção agora está sendo executada, o trecho de código então contará quanto tempo a entrada RB0 ficou em estado alto e atribuirá ela a uma variável denominada tempo. Essa variável é, portanto, a medida do comprimento de onda naquele instante.

```
35 int16 ref = 0;
36 int16 tempo = 0;
37 int count;
38
39 #INT_EXT
40 void pulsin(){
41    disable_interrupts(GLOBAL);
42    disable_interrupts(INT_EXT);
43
44    count = 0;
45
46    while(B0){
47       count++;
48       delay_ms(1);
49    }
50    tempo = count;
51
52    enable_interrupts(GLOBAL | INT_EXT);
53 }
```

Figura 4.2: Trecho de código que trata a interrupção externa do PIC16F628A para medir o comprimento de onda do sinal gerado pelo sonar .

Os pinos 4, 5 e 14 são referentes ao MCLR, terra e Vcc respectivamente. Já os pinos 10, 11 e 12 correspondentes as entradas RB4, RB5 e RB6 são configurados como saída e vão direto para os respectivos LEDs que indicarão o estado da vaga monitorada. Uma dessas saídas é escolhida, no caso a saída correspondente ao LED vermelho que indica que a vaga está ocupada, para ser enviado ao ZigBee e de lá para o outro módulo ZigBee na base para que o software possa tratá-lo e gerenciar, na base, o seu estado. A lógica do PIC é, portanto, o que se pode ver no fluxograma na figura (4.3).

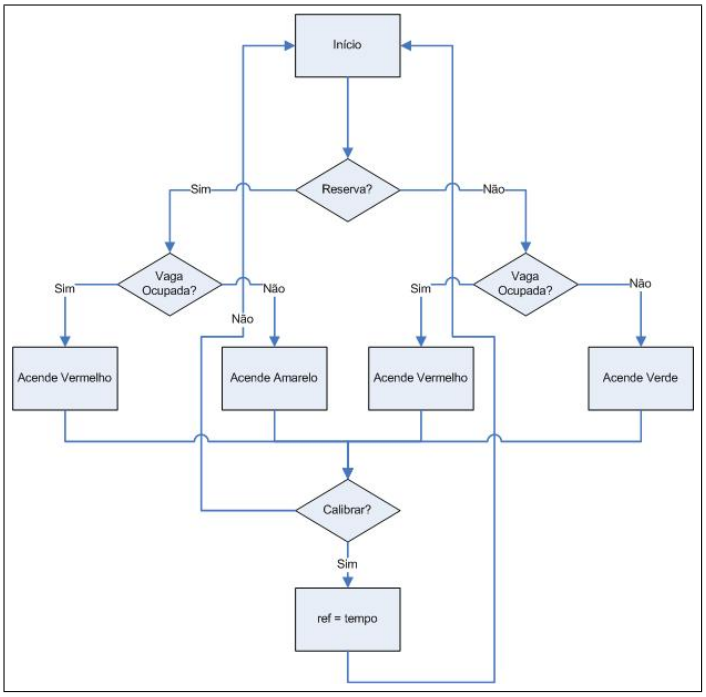

Figura 4.3: Fluxograma que explica a lógica por trás do código no PIC .

4.2 Os módulos XBee Pro Série 2

Foram utilizados no projeto dois módulos XBee Pro Série 2 da fabricante Digi International Inc. Algumas de suas principais características são:

- **Performance**
  - **Rendimento da Potência de saída:** 60 mW (18 dBm), 100 mW EIRP;
  - **Alcance em ambientes internos/zonas urbanas:** 100m;
  - **Alcance de RF em linha visível para ambientes externos:** 1,6Km;
  - **Sensibilidade do receptor:** -100 dBm (1% PER);
  - **Freqüência de operação:** ISM 2.4 GHz;
  - **Taxa de dados de RF:** 250.000 bps;
  - **Taxa de dados da Interface (Data Rate):** 115.200 bps;
- **Alimentação**
  - **Tensão de alimentação:** - 2.8 à 3.4v;
  - **Corrente de transmissão (típico):** - 215 mA @ 3.3 V;
  - **Corrente de Recepção (típico):** 55 mA @ 3.3 V;
  - **Corrente de Power-down Sleep:** $< 10\mu A$;

- **Propriedades físicas**
  - **Dimensões:** (2.438cm x 3.294cm);
  - **Peso:** 0.10 oz (3g);
  - **Temperatura de operação:** -40 to 85º C (industrial);
  - **Opções de antena:** Conector U.FL RF, Chip ou Chicote (whip);
- **Rede**
  - **Tipo de espalhamento espectral:** DSSS (Direct Sequence Spread Spectrum);
  - **Manipulação de erro:** Retransmite novamente (Retries) e reconhecimento (acknowledgements);
  - **Topologia de Rede:** Peer-to-peer(Par-a-par), ponto-a-ponto, ponto-a-multiponto e malha;
  - **Endereçamento:** 65.000 endereços de rede disponíveis para cada canal;
  - **Opções de filtros:** PAN ID, canais e endereços;
  - **Criptografia:** 128-bit AES;
  - **Número de canais selecionáveis via software:** 12 canais de seqüência direta;
- **Geral**
  - **Faixa de freqüência:** 2.4000 - 2.4835 GHz;

O módulo XBee possui 20 pinos. Para este projeto foram utilizados dois pinos de entrada/saída digital para envio/recebimento de dados proveniente da placa remota com o sensor e da base onde fica o computador gerenciador.

O bit de saída do PIC que diz se o LED vermelho está ou não aceso vai para a entrada digital no pino 17 do XBee remoto. Já o pino 18 está configurado como saída digital e transmite um bit proveniente do software de gerenciamento que vai estabelecer se uma vaga foi ou não foi reservada.

No projeto foi utilizada a topologia de rede mesh que vem por padrão configurada nos módulos XBee Pro Série 2.

4.2.1 Adaptador/Conversor USB

Para que os dados, após chegarem à base, possam ser processados por um software, é necessário que haja uma interface entre o módulo XBee receptor e qualquer interface de entrada/saída presente na maioria dos computadores no mercado.

No projeto foi utilizada uma placa adaptadora/conversora fabricada pela Rogercom. A placa tem interface USB e contém um driver que instala e configura uma porta serial

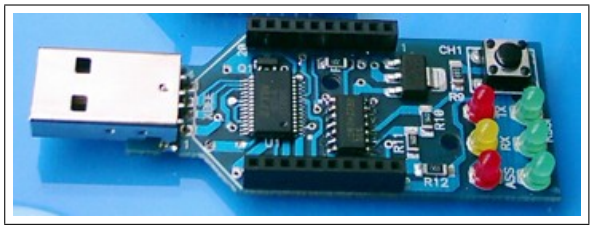

Figura 4.4: Adaptador/Conversor USB da Rogercom para módulos XBee Pro .

COMx para comunicação de dados. Dessa forma, os dados recebidos podem ser tratados como se tivessem sido recebidos por uma porta serial comum. Através dela pode-se também atualizar o firmware dos módulos e configurar seus parâmetros para a rede.

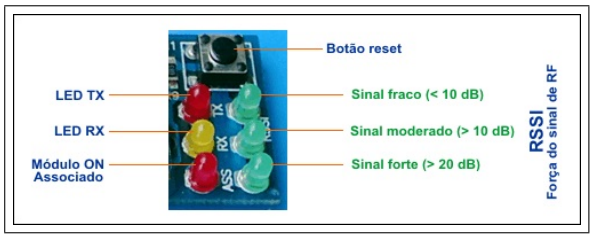

Figura 4.5: Detalhes dos indicadores da placa USB .

4.3 O Software de gerenciamento

Para fins de monitoramento de vagas de estacionamento e solicitação de reserva de vaga, foi desenvolvido um software em linguagem Java utilizando uma API open source denominada xbee-api.

Essa API fornece todas as funções necessárias para estabelecer comunicação serial entre o dispositivo que se encontra na base e o computador, bem como de receber e transmitir pacotes de dados entre os módulos presentes na rede.

O software estabelece uma conexão com uma porta COMx criada pelo driver da placa conversora/adaptadora e logo em seguida começa a ler o meio na espera de pacotes do módulo remoto. A medida que os pacotes vão chegando, o código trata os mesmos a fim de adquirir os dados relevantes referentes aos estado do bit de entrada. Uma vez que o bit de entrada, que vem transmitido pelo ZigBee remoto conectado ao PIC pela saída referente ao LED vermelho, encontra-se em nível alto, o programa sinaliza que a vaga em questão está ocupada, caso contrário, ela estará disponível.

Figura 4.6: Print Screen do software em funcionamento .

Um artifício extra foi permitir que o usuário solicite uma reserva de vaga. Nesse caso, o programa verifica se existe uma vaga livre, caso exista, ele pega a primeira vaga livre disponível e a configura como reservada. Ao mesmo tempo, envia para o módulo remoto correspondente àquela vaga um bit em nível lógico alto que será o bit de entrada na porta RA1 do PIC. Uma vez que esse bit está em alto no PIC, o mesmo fará a lógica já vista anteriormente com o objetivo de disponibilizar nos LEDs a configuração correta, ou seja, dispor o LED amarelo aceso.

No software, a classe que verifica esse processamento é a classe *CheckVaga.java*. Ela implementa uma Thread que fica rodando e verificando de acordo com o tempo de amostragem do módulo ZigBee os estados dos bits que estão sendo recebidos e enviados nos pacotes de dados. Essa Thread tem uma variável booleana que configura dois estados distintos. Em um denominado estado normal a comparação é feita levando em consideração que nenhuma vaga foi reservada. Já no outro estado, a vaga foi reservada e é verificado se em um intervalo de tempo algum veiculo estacionou no lugar previamente reservado, caso contrário, após esse intervalo de tempo ter se esgotado, o sistema volta ao estado normal e a vaga retorna ao estado livre. Caso algum veículo estacione no lugar que fora reservado previamente, o estado também retorna ao normal, uma vez que não há mais a reserva da vaga.

Um fluxograma que melhor organiza essa lógica do programa pode ser vista na figura (4.7)

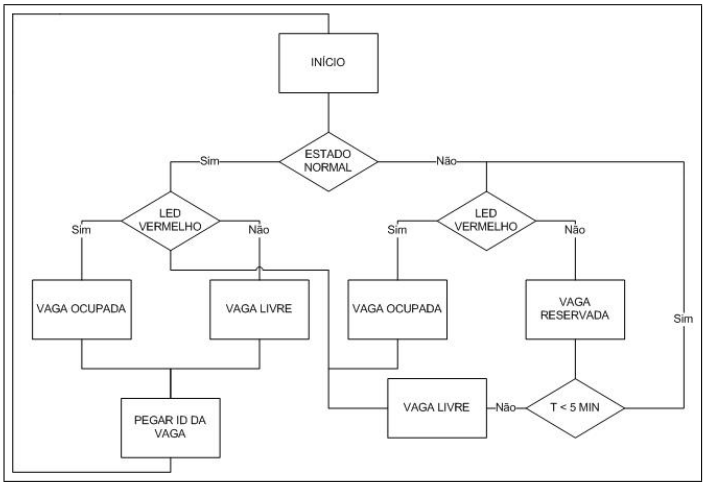

Figura 4.7: Fluxograma que explica a lógica do módulo base .

Capítulo 5

Conclusões

5.0.1 O sensor

O sensor utilizado obteve boas medidas e funcionamento adequado quando a distância do sensor ao obstáculo era pequena e o obstáculo era de superfície sólida e bem definida. Apesar da distância máxima, segundo a especificação fosse de 2,5 metros, na prática essa medida não chegou a ser alcançada com precisão.

O projeto, por sua vez, não se prende a um único tipo de sensor. O sonar utilizado por disponibilidade se mostrou capaz de atender aos requisitos do projeto e o mercado dispõe de sensores de ultrasom capazes de medir maiores distâncias e com poder de precisão melhorado.

5.0.2 O módulo XBee

O protocolo 802.15.4 da IEEE e os protocolos da camada superior que constituem o ZigBee se mostram extremamente robustos e eficientes na transmissão e recepção de pacotes de dados de maneira segura e consistente. É, sem dúvida, uma tecnologia que pelo fato de ser padronizada tende a ter um aumento na utilização em diversas áreas de aplicação.

5.0.3 O PIC16F628A

O PIC utilizado não é um PIC muito avançado. O projeto, portanto, não necessita de muitos recursos de programação de microcontroladores, embora essa programação seja necessária e sirva como interface entre o sensor e o transmissor. O processamento dentro do PIC foi realizado sem problemas e o funcionamento do circuito, apesar de alguns ruídos, foi contornado com o uso de capacitores.

5.0.4 XBee-api para Java

A API utilizada do lado da programação na base receptora foi realizada sem maiores problemas pela API xbee-api. Conexões, envio e recebimento de pacotes e comandos para os módulos foram feitas sem qualquer restrição. O único problema que poderia ser mencionado é o delay normal entre comandos que não necessariamente são frutos da API utilizada.

5.0.5 Considerações gerais

De modo geral o protótipo do produto que esboça um modelo de estacionamento automatizado se mostra viável. Através deste projeto, com a comunicação de sensor sem fio utilizando tecnologia ZigBee, um leque de possibilidades surge no quesito de adquirir informações importantes para fins específicos.

Uma das possibilidades consiste em trabalhar estatisticamente os dados pelos quais o software lida, na medida em que sabe quantos carros estão estacionados em um estabelecimento, quais dias da semana são mais movimentados, quais são menos e o lugar onde a maioria dos carros se concentra para estacionar. Em ambientes como um shopping center, tais informações são de grande relevância quanto a elaboração de campanhas de marketing entre lojas.

Além do mais, a comodidade e a economia de tempo ao utilizar o sistema e encontrar uma vaga a disposição mais rapidamente traz benefícios para a economia e para o bem estar do usuário.

Por fim, o investimento em tecnologias sem fio que busquem soluções para problemas corriqueiros do cotidiano podem ajudar a sociedade a viver com melhor qualidade de vida a medida que economiza tempo. Sem mencionar o fato de estar se adaptando a um meio cada vez mais inteligente em se tratando de tecnologias em desenvolvimento.

Referências Bibliográficas

de Carvalho, Leandro Ouriques M. & Pablo Salino Cunha (2010), '802.15.4 e zigbee', *Wireless Sensor Network, Research Group* .

de Souza, David Jośe (2006), *Desbravando o PIC*, Editora Érica.

Eletrônicos, Tato Equipamentos (2010), 'Medindo distâncias por ultrasom', *www.tato.ind.br* .

Gascón, David (2009*a*), '802.15.4 vs zigbee', *Wireless Sensor Network, Research Group* .

Gascón, David (2009*b*), 'Security in 802.15.4 and zigbee networks', *Wireless Sensor Network, Research Group* .

Gascón, David (2009*c*), 'Zigbee vs zigbee-pro', *Wireless Sensor Network, Research Group* .

Messias, Antônio Rogério (2008), 'Controle remoto e aquisição de dados via xbee/zigbee (ieee 802.15.4)', *www.rogercom.com* .

Apêndice A

Informações adicionais

A.1 Códigos fontes

Código Fonte A.1: Código fonte para o PIC16F6284A

```
/* ------ Includes referentes ao PIC ------ */
#include <16F628A.h>

#FUSES NOWDT                    //No Watch Dog Timer
#FUSES INTRC_IO                 //Clock interno de 4MHz
#FUSES PUT                      //Power Up Timer
#FUSES NOPROTECT                //Code not protected from reading

#FUSES BROWNOUT                 //Reset when brownout detected
#FUSES NOLVP                    //No low voltage prgming, B3(PIC16) or
    B5(PIC18) used for I/O
#FUSES NOCPD                    //No EE protection
#FUSES MCLR

#use delay(clock=4000000)

#use fast_io(a)
#use fast_io(b)

/* ------ Configuração das portas ------ */

#byte PORT_A = 0x05
#byte PORT_B = 0x06

#byte TRIS_A = 0x85
#byte TRIS_B = 0x86
```

```
#bit BUTTON = PORT_A.2     //BOTAO DE CALIBRACAO
#bit RESERVA = PORT_A.1 //ENTRADA VINDA DO ZIGBEE
#bit LED1 = PORT_B.1 //LED QUE VERIFICA SE O BOTAO FOI APERTADO
#bit B0 = PORT_B.0 //BIT DE INTERRUPCAO
#bit VERMELHO = PORT_B.4   //LED VERMELHO
#bit VERDE = PORT_B.5   //LED VERDE
#bit AMARELO = PORT_B.6 //LED AMARELO

int16 ref = 0;
int16 tempo = 0;
int count;

#INT_EXT
void pulsin(){
   disable_interrupts(GLOBAL);
   disable_interrupts(INT_EXT);

   count = 0;

   while(B0){
      count++;
      delay_ms(1);
   }
   tempo = count;

   enable_interrupts(GLOBAL | INT_EXT);
}


void main(){
   TRIS_A = 0b00000110; //
   TRIS_B = 0b00000001; //Habilitado para leitura

   VERDE = 0;
   VERMELHO = 0;
   AMARELO = 0;
   BUTTON = 0;
   RESERVA = 0;
   B0 = 0;
   LED1 = 0;

   ext_int_edge(L_TO_H);
   enable_interrupts(GLOBAL | int_ext);
```

```
70
    while(true){
        if(RESERVA){
            if(tempo < ref){
                VERMELHO = 1;
75              VERDE = 0;
                AMARELO = 0;
            }else{
                VERMELHO = 0;
                VERDE = 0;
80              AMARELO = 1;
            }
        }else{
            if(tempo < ref){
                VERMELHO = 1;
85              VERDE = 0;
                AMARELO = 0;
            }else{
                VERMELHO = 0;
                VERDE = 1;
90              AMARELO = 0;
            }
        }

        if(BUTTON){
95              ref = tempo;
                LED1 = 1;
            }else{
                LED1 = 0;
        }
100 }
}
```

Código Fonte A.2: Codigo fonte do sofware gerenciado. Lógica para gerenciar vagas

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
5 package classes;

import com.rapplogic.xbee.api.ApiId;
import com.rapplogic.xbee.api.RemoteAtRequest;
import com.rapplogic.xbee.api.RemoteAtResponse;
```

```
10 import com.rapplogic.xbee.api.XBee;
   import com.rapplogic.xbee.api.XBeeAddress64;
   import com.rapplogic.xbee.api.XBeeException;
   import com.rapplogic.xbee.api.XBeeResponse;
   import com.rapplogic.xbee.api.zigbee.ZNetRxIoSampleResponse;
15 import com.rapplogic.xbee.util.ByteUtils;
   import java.awt.Color;
   import java.util.logging.Level;
   import java.util.logging.Logger;
   import javax.swing.JLabel;
20 import javax.swing.JPanel;
   import javax.swing.JTextArea;

   /**
    *
25  * @author Iuri
    */
   public class CheckVaga extends Thread {


30     private static int MEIO_SEGUNDO = 500;
       private static int UM_MINUTO = 60000;
       private static int UM_SEGUNDO = 500;
       private static int CINCO_MINUTOS = 60000 * 5;

35     private boolean ESTADO_NORMAL;
       private boolean active;
       private JLabel status;
       private JLabel idvaga;
       private JPanel vaga;
40     private boolean verde;
       private XBee xb;
       private XBeeAddress64 end64;
       private StringBuilder textLog;
       private JTextArea log;
45     private XBeeResponse resposta;
       private long inicio, fim;

       public CheckVaga(JLabel status, JLabel idvaga, JPanel vaga, XBee xb
           , JTextArea log) {
           this.ESTADO_NORMAL = true;
50         this.active = true;
           this.status = status;
```

```
        this.idvaga = idvaga;
        this.vaga = vaga;
        this.verde = true;
        this.xb = xb;
        this.log = log;
    }

    public boolean isVerde() {
        return verde;
    }

    public void setVerde(boolean verde) {
        this.verde = verde;
    }

    public long getFim() {
        return fim;
    }

    public void setFim(long fim) {
        this.fim = fim;
    }

    public long getInicio() {
        return inicio;
    }

    public void setInicio(long inicio) {
        this.inicio = inicio;
    }

    public boolean isESTADO_NORMAL() {
        return ESTADO_NORMAL;
    }

    public void setESTADO_NORMAL(boolean ESTADO_NORMAL) {
        this.ESTADO_NORMAL = ESTADO_NORMAL;
    }

    public JTextArea getLog() {
        return log;
    }
```

```
95      public void setLog(JTextArea log) {
            this.log = log;
        }

        public boolean isActive() {
100         return active;
        }

        public void setActive(boolean active) {
            this.active = active;
105     }

        @Override
        public void run() {

110         while (active) {
                try {
                    if (ESTADO_NORMAL) {
                        resposta = xb.getResponse(15000);

115                     if (resposta.getApiId() == ApiId.
                            ZNET_IO_SAMPLE_RESPONSE) {
                            ZNetRxIoSampleResponse ioSample = (
                                ZNetRxIoSampleResponse) resposta;

                            if (ioSample.isD2On()) {
                                vaga.setBackground(Color.RED);
120                             status.setText("VAGA OCUPADA");
                                verde = false;
                            } else {
                                vaga.setBackground(Color.GREEN);
                                status.setText("VAGA LIVRE");
125                             verde = true;
                            }
                            //pegar o endereco de 16bits do sensor - saber
                                qual eh a vaga
                            end64 = ioSample.getRemoteAddress64();

130                     }
                        xb.sendAsynchronous(new RemoteAtRequest(end64, "NI"
                            ));
                        resposta = xb.getResponse(15000);
                        if (resposta.getApiId() == ApiId.REMOTE_AT_RESPONSE
```

```
                    ) {
                    RemoteAtResponse atResponse = (RemoteAtResponse
                        ) resposta;
                    if (atResponse.isOk()) {
                        idvaga.setText(ByteUtils.toString(
                            atResponse.getValue()));
                        System.out.println(ByteUtils.toBase16(
                            atResponse.getValue()));
                    }
                }
            } else {
                resposta = xb.getResponse(15000);

                if (resposta.getApiId() == ApiId.
                    ZNET_IO_SAMPLE_RESPONSE) {
                    ZNetRxIoSampleResponse ioSample = (
                        ZNetRxIoSampleResponse) resposta;

                    if (ioSample.isD2On()) {
                        vaga.setBackground(Color.RED);
                        status.setText("VAGA OCUPADA");

                        //enviar um bit low para o microcontrolador
                        xb.sendAsynchronous(new RemoteAtRequest(
                            end64, "D3", new int[]{4}));
                        resposta = xb.getResponse(15000);
                        if (resposta.getApiId() == ApiId.
                            REMOTE_AT_RESPONSE) {
                            RemoteAtResponse response = (
                                RemoteAtResponse) resposta;
                            if (response.isOk()) {
                                System.out.println("Bit enviado!");
                            } else {
                                System.out.println("Problema no
                                    envio do bit!");
                            }
                        }
                        ESTADO_NORMAL = true;
                    } else {
                        fim = System.currentTimeMillis();
                        if ((fim - inicio) <= UM_MINUTO) {
                            vaga.setBackground(Color.ORANGE);
                            status.setText("VAGA RESERVADA");
```

```
                                    //enviar um bit ativo para o
                                        microcontrolador
                                    xb.sendAsynchronous(new RemoteAtRequest
                                        (end64, "D3", new int[]{5}));
                                    resposta = xb.getResponse(15000);
                                    if (resposta.getApiId() == ApiId.
                                        REMOTE_AT_RESPONSE) {
                                        RemoteAtResponse response = (
                                            RemoteAtResponse) resposta;
                                        if (response.isOk()) {
                                            System.out.println("Bit enviado
                                                !");
                                        } else {
                                            System.out.println("Problema no
                                                envio do bit!");
                                        }
                                    }
                                } else {
                                    xb.sendAsynchronous(new RemoteAtRequest
                                        (end64, "D3", new int[]{4}));
                                    resposta = xb.getResponse(15000);
                                    if (resposta.getApiId() == ApiId.
                                        REMOTE_AT_RESPONSE) {
                                        RemoteAtResponse response = (
                                            RemoteAtResponse) resposta;
                                        if (response.isOk()) {
                                            System.out.println("Bit enviado
                                                !");
                                        } else {
                                            System.out.println("Problema no
                                                envio do bit!");
                                        }
                                    }
                                    vaga.setBackground(Color.GREEN);
                                    status.setText("VAGA LIVRE");
                                    ESTADO_NORMAL = true;
                                }
                            }
                        }
                    }

                } catch (XBeeException ex) {
```

```
                    System.out.println(ex.getCause());
200             }
                try {
                    sleep(100);
                } catch (InterruptedException ex) {
                    Logger.getLogger(CheckVaga.class.getName()).log(Level.
                        SEVERE, null, ex);
205             }
            }
        }
    }
```