Pontifícia Universidade Católica do Rio Grande do Sul
Faculdade de Engenharia – Faculdade de Informática
Curso de Engenharia de Computação

Braço Robótico Com Controle Remoto Bluetooth

Igor Bajerski

Vinicius Dal Bó Abella

Trabalho de conclusão do curso de

Engenharia de Computação

Orientador: Prof. Júlio César Marques de Lima

Porto Alegre, 29 de Junho de 2010

Igor Bajerski

Vinicius Dal Bó Abella

Braço Robótico com Controle Remoto Bluetooth

Trabalho de conclusão do curso de Engenharia de Computação, realizado como última atividade necessária para a obtenção do diploma de Engenheiro de Computação pela Pontifícia Universidade Católica do Rio Grande do Sul.

Aprovado em _____ de _____ de _____.

BANCA EXAMINADORA:

Prof. Anderson Royes Terroso

_____

Prof. Júlio César Marques de Lima

_____

Prof. Márcio Sarroglia Pinho

_____

AGRADECIMENTOS

Eu, Igor Bajerski gostaria de agradecer aos meus pais pelo incentivo que me deram em minha caminhada para me formar Engenheiro de Computação. Agradeço também a Priscila Alves pelo suporte emocional e compreensão. Também merecem o meu agradecimento os meus amigos que sempre estiveram ao meu lado me passando calma e bom humor.

Eu, Vinícius Dal Bó Abella, dedico esse trabalho aos meus pais pelo carinho e apoio durante a graduação, e a paciência durante a elaboração deste trabalho. Aos professores envolvidos, pela atenção e pelos conselhos imprescindíveis, em especial ao professor Júlio César Marques de Lima. Agradeço a Fillipe Welausen pelo incentivo na concepção do projeto piloto. Agradeço, também, a todos os colegas de graduação, amigos e parentes que apoiaram a idéia deste projeto, em especial a Tamirys Delazeri Sangali, pela compreensão e carinho.

RESUMO

Os alunos de Engenharia de Computação em busca de aprimorar os conhecimentos, pouco bordados, em robótica, propuseram se em realizar a construção total de um pequeno braço mecânico com uma interface de manipulação sem fio e inovadora. Tendo como desafio construir o braço mecânico com matérias de fácil acesso e apresentando um resultado final de total funcionalidade e intuitividade para o usuário. O pequeno braço mecânico foi construído utilizando-se servomotores empregados em aeromodelismo e automodelismo suportado por uma estrutura de alumínio. O objetivo da manipulação inovadora e sem fio foi atingido ao utilizar o controle remoto do console Nintendo Wii como dispositivo de manipulação remota do braço robótico. Ao final se obtém um pequeno braço robótico de manipulação simples, inovadora e intuitiva, construído com materiais facilmente encontrados no mercado. Braço mecânico totalmente funcional de acordo com as especificações descritas.

SUMÁRIO

1. **Introdução**

Países como o Brasil combinam duas realidades distintas, de um lado apresentando altos índices de desenvolvimento em determinadas regiões, compatíveis com as condições encontradas em países de primeiro mundo, em contraste com a dura realidade da miséria encontrada em países pobres e subdesenvolvidos. Este talvez seja o maior desafio dos governos que se sucedem, o de encontrar uma forma de equilibrar realidades tão distintas e promover a justiça social.

São exemplos clássicos de países que atingiram rápida e plenamente o desenvolvimento, aqueles que priorizaram a aplicação de recursos em Engenharia. É sabido que a educação é o meio pelo qual se atinge esta condição. Mas para isto é necessária uma educação de qualidade, principalmente a de cunho técnico-científico. Este tipo de educação requer a aplicação de grandes quantidades de recursos e acesso a bens e serviços de alta tecnologia.

Dentre as diversas áreas do conhecimento que abrangem a Engenharia, uma das mais promissoras e interessantes é a robótica, uma vez que ela se insere em diversos segmentos, como na cadeia produtiva, na área de pesquisa e desenvolvimento, de entretenimento, educação, segurança ou mesmo na de saúde. A importância em se dominar este conhecimento e a falta de recursos para acesso a dispositivos robóticos foram as razões pela qual ficou-se interessado pelo presente trabalho, ou seja, a construção de um braço robótico que apresentasse uma interface de controle flexível e moderna.

O curso de Engenharia de Computação tem como objetivo a formação de profissionais para atuar em processos de automação, integrando aspectos relacionados ao desenvolvimento e gerência de projetos de hardware e software. Tais aspectos envolvem sistemas embarcados, sistemas de tempo real, sistemas integrados de hardware e software, sistemas distribuídos, redes de computadores e sistemas de comunicação.

A necessidade da criação do curso de Engenharia de Computação pode ser justificada pelo crescente nível de automatização e informatização da sociedade moderna, que fez surgir já há alguns anos, nas mais conceituadas universidades do mundo, esta nova modalidade de atuação, denominada Engenharia de Computação. A necessidade de um contingente de

profissionais capazes de interagir dinamicamente com as demandas computacionais em mais baixo nível (hardware), até então normalmente realizadas por profissionais egressos dos cursos de Engenharia Elétrica, com sólidos conhecimentos das necessidades e tendências das demandas computacionais em mais alto nível (software), até então normalmente realizadas por profissionais egressos dos cursos de Ciência da Computação, deu espaço a criação e crescente procura do mercado, por profissionais desta nova área.

De um egresso de cursos de Engenharia, segundo a comissão de especialistas de Engenharia do MEC espera-se: "... uma sólida formação técnico-científica e profissional geral que o capacite a absorver e desenvolver novas tecnologias, estimulando a sua atuação crítica e criativa na identificação e resolução de problemas, considerando seus aspectos políticos, econômicos, sociais, ambientais e culturais, com visão ética e humanística, em atendimento às demandas da sociedade.".

Os autores deste trabalho buscaram nesse projeto, utilizando todo o conhecimento teórico adquirido durante o momento acadêmico no curso de Engenharia de Computação, aprimorar o conhecimento em algumas áreas que julgaram ser de interesse profissional. A presença de Engenheiros de Computação está cada vez mais presente no universo da automação, do qual faz parte a área da robótica. Procurando complementar os conhecimentos obtidos no curso, os alunos perceberam uma oportunidade interessante ao propor o desafio de construir um pequeno braço robótico, acrescentando ao mesmo, uma interface de comunicação flexível para seu controle, de baixo custo e de aplicação didática.

Um dos grandes assuntos abordados durante o curso de Engenharia de Computação é o estudo de redes de comunicação, e é sabido pelos alunos envolvidos nesse projeto, que esse conhecimento é imprescindível para o sucesso de um Engenheiro da Computação no mercado. Decidiu-se então também focar o estudo em na construção de uma interface de controle do braço, com alta conectividade e flexibilidade para se adaptar as necessidades do projeto, recursos estes que estão ausentes na maioria dos braços robóticos, e desta forma se apresentando como uma forma inovadora de controle.

1.1. **Justificativa**

Dado um cenário onde o conhecimento, cada vez mais, está fazendo a diferença e a preocupação dos alunos em buscar este conhecimento, que por algum motivo, não foi

abordado em quantidade e profundidade durante o curso, buscou-se complementar a formação acadêmica através da pesquisa de alguns assuntos considerados imprescindíveis, na visão e interesse dos autores deste trabalho, concluindo assim o curso com maior densidade de conhecimento e uma vivência prática de assuntos abrangentes como o da robótica.

Como todo o curso de graduação, uma grande carga de conhecimento teórico é aplicada sobre os alunos, porém em muitas disciplinas não existe uma vivência prática adequada, por quaisquer que sejam as razões disto, fazendo com que a experiência prática não seja treinada. Na visão dos alunos, a vivência prática e tão importante quanto o conhecimento teórico, e por esse motivo, buscou-se desenvolver na prática assuntos tratados, em sua maioria, de maneira teórica.

1.2. **Motivação**

Desenvolver e aguçar aptidões adquiridas durante o decorrer do curso de Engenharia de Computação, procurando estar completos, no escopo do conhecimento teórico e prático, para as necessidades do mercado, utilizando essa inteligência para nos tornarmos Engenheiros que saibam propor soluções viáveis, aplicáveis na prática e tornando-se pessoas de sucesso ao estarmos aptos a atuar em diversas áreas.

O instrumento para tal objetivo é o desenvolvimento de um pequeno braço robótico buscando desta forma exercitar os diversos conhecimentos e habilidades que este projeto requer. Braços robóticos são uma realidade na indústria e em aplicações de alta tecnologia como as da área aeroespacial, porém não são acessíveis na maior parte das escolas, entre outras razões, pelo seu elevado custo. Viu-se nesta combinação de fatores a oportunidade de desenvolver um projeto inovador, de baixo custo, para que outros estudantes e interessados na área, pudessem ter acesso a este dispositivo, nos mesmos moldes de cooperação que cada vez mais encontramos na internet. Entre as características inovadoras do projeto, buscou-se o de estabelecer a comunicação de dados com dispositivos Bluetooth e a utilização de acelerômetros para a construção da interface de controle.

2. Objetivos

São três os objetivos principais deste trabalho. O primeiro visa suprir as carências de mercado por manipuladores robóticos de baixo custo, que possam ser empregados em atividades de ensino ou entretenimento, contendo uma interface de controle totalmente flexível, empregando tecnologias atuais de conectividade e empregando dispositivos modernos para a construção da interface de controle, com a utilização de acelerômetros, tudo isto com foco no baixo custo e com a utilização de materiais e componentes de fácil acesso.

O segundo visa complementar as atividades acadêmicas de modo a obter uma formação mais completa de Engenharia de Computação, com especial interesse pela área de automação industrial, permitindo uma melhor fixação de conhecimentos, que julgados por nós, são imprescindíveis para uma boa colocação deste alunos no mercado de trabalho, bem como de constituir em um desafio na busca por uma aplicação prática interessante que se pudesse propor como trabalho de fim de curso.

O terceiro objetivo é o de inovar na escolha de dispositivos de controle do braço robótico, com a utilização de tecnologias modernas, como o uso de conectividade por Bluetooth e acelerômetros, construindo um pequeno braço robótico, com baixo investimento e com materiais de fácil acesso.

3. **Descrição do Projeto**

Este projeto consiste na construção de um pequeno braço robótico didático que possibilite a aplicação na prática de conceitos de automação com materiais facilmente acessíveis no mercado e que tenha um custo final baixo. A plataforma de controle do dispositivo será baseada em um micro controlador MSP 430F2619 acoplado a uma placa didática fornecida pela PUCRS. O KIT de Laboratório de Processadores I fornece todo o tipo de suporte de conexões de entradas e saídas. O KIT implementa o controle principal do braço robótico, o qual irá controlar todos os motores que movimentam as articulações do braço robótico.

Para a movimentação das juntas do braço robótico optou-se por utilizar servomotores de posição. Cada articulação do braço robótico é controlada por um ou mais servomotores, os quais permitem variações angulares entre 0 e 180 graus. O pequeno braço robótico possui 5 graus de liberdade. Este braço pode girar até 180º em torno de seu eixo e em sua extremidade possui uma garra do tipo pinça, a qual é utilizada para a manipulação de objetos. A pinça pode girar também, até 180 graus em torno de seu eixo, proporcionando maior flexibilidade na manipulação de objetos.

O movimento do braço robótico é feito com o uso de acelerômetros e botões encontrados nos controles do vídeo game Nintendo Wii, ou seja, o Wimote e o nunchuck, transmitidos através de tecnologia Bluetooth até a placa de controle. O objetivo foi a de buscar por dispositivos modernos e já difundidos no mercado que utilizem a tecnologia Bluetooth.

A estrutura do braço robótico foi feita em alumínio, a qual serve de suporte para os servomotores de cada articulação. A escolha por este material foi feita em função das dimensões e peso envolvidos. O pequeno braço utiliza servomotores de pequeno torque, o que limita a escolha por certos materiais. As dimensões do braço e o seu formato podem ser vistos na *figura 7*.

4. **Servomotores**

Servomotores são comumente utilizados para controlar movimentos angulares, tipicamente, entre 0º e 180º. Esses dispositivos são constituídos por um motor DC, um circuito eletrônico de controle, um potenciômetro, um conjunto de engrenagens e três condutores exteriores de ligação, como exemplificado na *figura 1*.

Figura 1: Exemplo de componentes de um servomotor

O motor do servomotor ao ser alimentado faz girar uma série de engrenagens redutoras (caixas de redução) que aumentam o torque do motor permitindo a movimentação de objetos acoplados ao eixo de forma muito precisa. Devido a combinação destas engrenagens redutoras obtém-se um aumento no torque do motor. Se o motor fosse ligado diretamente ao eixo de controle, para movimentar a mesma carga, seu tamanho físico deveria ser várias vezes maior do que os empregados nos servos.

4.1. **Principio de Funcionamento dos Servomotores**

O circuito eletrônico de controle e o potenciômetro formam um sistema interno de realimentação (*feedback*) para controle da posição do eixo do servo. Tipicamente, o eixo de um servomotor roda entre 0º e 180º e pode ser posicionado, entre estes dois valores angulares, pela aplicação de um sinal na entrada de controle. Aplicado e mantido o sinal, o servo manterá a posição angular do seu eixo. Se o sinal mudar, então a posição angular do eixo também irá mudar. Caso não seja aplicado sinal algum, só as forças de atrito mantêm o servo na sua posição angular.

O sistema interno de realimentação faz com que o servo gire em uma determinada posição em resposta a um determinado trem de impulsos. O potenciômetro de *feedback*, que está conectado mecanicamente ao eixo do servomotor, funciona como sensor (encoder absoluto) que indica a posição do eixo uma vez que a sua resistência varia em função do ângulo de rotação do motor.

O circuito eletrônico compara o valor da resistência do potenciômetro com os impulsos que recebe pela linha de controle, ativando o motor para corrigir qualquer diferença que exista entre ambos. Isto é, o potenciômetro permite ao circuito de controle verificar a todo o momento o ângulo de rotação do servomotor. Se o eixo estiver no ângulo correto, o motor não gira. Se o circuito verificar que o ângulo não é o correto, então o motor irá girar, no sentido adequado, até alcançar o ângulo correto. A comparação entre o valor do potenciômetro e a largura dos impulsos e as correções necessárias, são partes de um processo de controle conhecido por controle em malha fechada.

Os condutores exteriores de ligação consistem em dois sinais para a alimentação DC e outro para ligação do sinal de comando. A tensão de alimentação dos servomotores está normalmente compreendida entre 4.8 V e 6 V, sendo recomendável a tensão de 5 V. Quanto mais baixa é a tensão mais lenta é a resposta do servomotor e menor é o seu torque.

4.2. **Controle do Ângulo de Rotação dos Servomotores**

O ângulo de rotação do motor dos servomotores é determinado pela duração do impulso (nível lógico alto) que se aplica na entrada de comando. O servomotor funciona em PWM (*Pulse Width Modulation* ou modulação por largura de impulso), sistema que consiste em gerar um sinal quadrado em que se varia a duração do impulso, mantendo-se fixo o período deste sinal.

A largura mínima e máxima do impulso depende do tipo do servomotor. No entanto, e no caso geral, se o servomotor receber na sua entrada impulsos com a duração de 1.5ms, o seu eixo roda até ficar estável no centro do intervalo de rotação, a que corresponde o ângulo de 90º. Se receber impulsos com a duração de 1ms, roda, no sentido anti-horário, até atingir o limite do intervalo de rotação correspondente a 0º. Se receber impulsos com a duração de

2ms, roda, no sentido horário, até atingir o outro limite do intervalo de rotação correspondente a 180º ou um pouco mais.

Impulsos ente 1ms e 1.5ms farão com que o servomotor rode para posições intermédias entre 0º e 90º, enquanto impulsos entre 1.5ms e 2ms farão com que o servomotor rode para posições intermédias entre 90º e 180º, movimentos exemplificados na *figura 2*.

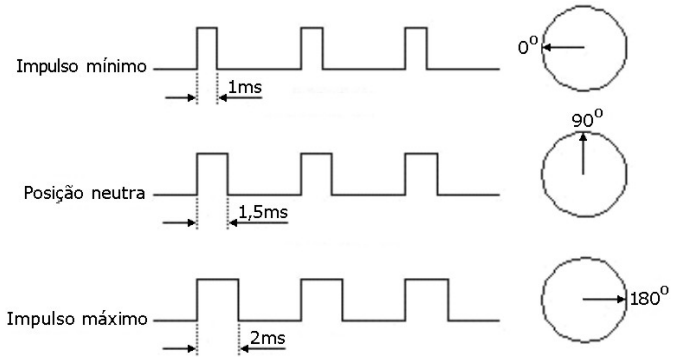

Figura 2 - Exemplo de pulsos de controle de um servomotor.

Os impulsos, para que o servomotor funcione corretamente, devem ser aplicados a cada 20 ms mas valores entre 10 ms e 30 ms também são aceitos, onde apenas o que se varia é o tempo em que o sinal se mantém em nível lógico alto (largura do impulso). A *figura 3* exemplifica essa característica.

Figura 3 - Exemplo de período de um pulso de controle de servomotor.

4.3. **Torque dos Servomotores**

O torque dos servomotores permite avaliar a força que os mesmos são capazes de exercer. Por exemplo, um servomotor que apresente um torque de 3 kg.cm a 4.8 V, significa que com uma polia, engrenagem ou braço de 1 cm de comprimento ligado ao eixo do servomotor, o mesmo está apto a levantar até 3 kg. Caso a polia, engrenagem ou braço

possua, por exemplo, 2,5 cm, o servo será capaz de levantar até 1,2 kg. Percebe-se desta forma que, para os tamanhos que possuem, os servos são extremamente potentes.

4.4. **Servomotores utilizados**

Foram utilizados três tipos diferentes de servomotores:

- Servomotor MG945
- Servomotor MG90S
- Servomotor SG90

4.5. **Disposição dos Servomotores**

A figura 4 apresenta o aspecto físico do braço robótico projetado, onde pode-se observar os 5 graus de liberdade que o braço apresenta. Na figura, os pontos marcados de 1 à 5 descrevem a posição de cada servomotor utilizado no projeto.

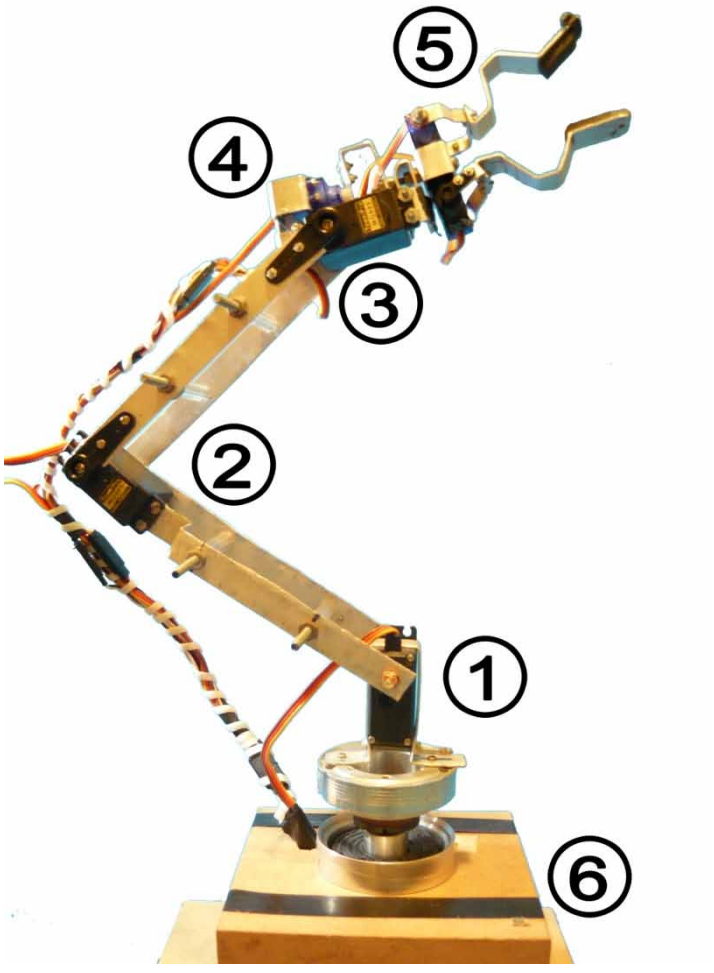

Figura 4 - Posição dos servomotores no braço.

A tabela 1 complementa a figura 4, descrevendo o nome de cada servomotor enumerado, em cada uma das juntas do braço.

Número	Nome
1	Servomotor 01
2	Servomotor 02
3	Servomotor 03
4	Servomotor 04
5	Servomotor 05
6	Servomotor 06

Tabela 1 - Servomotores do projeto.

O Servomotor 05 é composto na realidade por 2 servomotores, porém esses 2 servomotores são alimentados pelo mesmo sinal de controle.

4.6. **Servomotor MG945**

São os servomotores mais potentes utilizados nesse projeto. Na coloração preta, esses servomotores foram utilizados nas articulações que necessitam de maior torque. Os MG945 foram intitulados como os servomotores 01,02,03 e 06, por serem as articulações que estão mais próximas da base, ou seja, que necessitam exercer maior força dentre todos os outros servomotores.

Esse modelo de servomotor possui um ângulo de trabalho de 0° e 180°, e o tempo que o servomotor necessita para percorrer 60° é 0,23 segundo. Nota-se que esse modelo de servomotor não tem como prioridade a velocidade no movimento, e sim, o torque disponível. O torque esse é de 10kg.cm, de acordo com a especificação de tensão utilizada nesse projeto, a qual provê 5 volts na alimentação de todos os servomotores. O MG945 possui uma massa de 55g, o mais pesado servomotor utilizado no projeto, porém, como esse modelo foi utilizado em articulações próximas à base, ficou minimizado o efeito desta massa elevada.

Outra característica importante deste servomotor, que também foi decisiva para utilização do mesmo nesse projeto, está relacionada com o tipo de material empregado na construção das engrenagens da caixa de redução, de metal ao invés de plástico, o que possibilita uma maior vida útil do servomotor. Objetivando criar um protótipo que com a devida engenharia de produto possa ser comercializado, preocupou-se em projetar um dispositivo que possua uma vida útil que justifique o investimento realizado por quem adquirir o braço robótico.

4.7. **Servomotor MG90S**

O SG90R tem por característica principal ser um servomotor de baixa massa, velocidade alta e possuir engrenagens de ferro. Esse modelo foi empregado nos servomotores intitulados servomotores 05. Os servomotores 05 compõem o acionamento da garra mecânica do braço robótico.

O servomotor MG90R possui um ângulo de trabalho de 0° e 180°, e o tempo que o servomotor necessita para realizar este movimento é de 0,1 segundo. O torque desse modelo é de 2kg.cm, de acordo com a especificação de tensão desse projeto, o qual disponibiliza 5 volts na alimentação de todos os servomotores. Nota-se, que apesar da pouca massa, 13,4g, esse servomotor proporciona uma velocidade adequada com um torque apropriado para os objetos que esse projeto se propõe a manipular.

Devido à utilização desses servomotores na composição da garra mecânica, onde os mesmo podem ser tensionados por tempos longos, teve-se a preocupação de utilizar servomorores com engrenagens de redução fabricadas em metal, assim garantindo uma maior vida útil do dispositivo como um todo.

4.8. **Servomotor SG90**

Esse servomotor é o único no projeto que é composto por engrenagens de redução de nylon. O SG90 foi empregado no servomotor intitulado servomotor 4. O servomotor 4 tem a responsabilidade e realizar o giro da garra mecânica.

O servomotor SG90 possui um ângulo de trabalho de 0° e 180°, e o tempo que o servomotor necessita para realizar este movimento é de 0,1 segundo. O torque desse modelo é de 1,8kg.cm, de acordo com a especificação de tensão desse projeto, o qual fornece 5 volts na alimentação de todos os servomotores. A massa desse modelo é de 9g, o mais leve de todos os servomotores empregados nesse projeto. Esta característica foi explorada na criação do braço robótico, pois é importante reduzir a massa nas extremidades finais do braço robótico, de modo a garantir que os servomotores MG995 suportem as cargas exercidas pelos objetos a serem manipulados, além da massa do corpo do braço robótico.

5. O MSP430

O MSP430 é um micro controlador RISC simples de 16 bits fabricado pela Texas Instruments, com uma CPU compacta e econômica que opera na freqüência de 25MHz e contém 27 instruções e 16 registradores. Ele oferece vantagens que o fazem adequado para aplicações com baixo consumo de potência como aplicações wireless de RF.

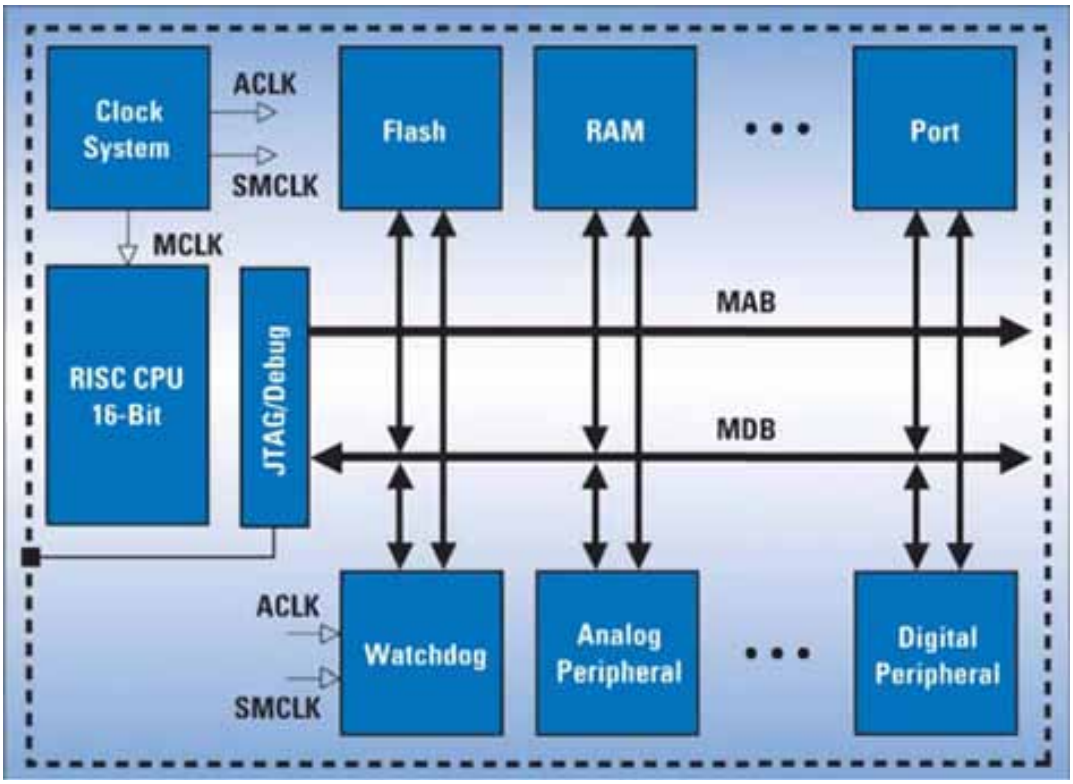

Figura 5: Arquitetura do MSP430.

São disponibilizados diversos periféricos como *timers*, USARTS, Conversores AD de 10, 12 e 16 bits, comparador analógico, amplificador operacional, conversores DA de 10 bits e/ou 12 bits, controlador de LCD, etc. A figura 6 exibe o micro controlador associado ao KIT de Processadores I.

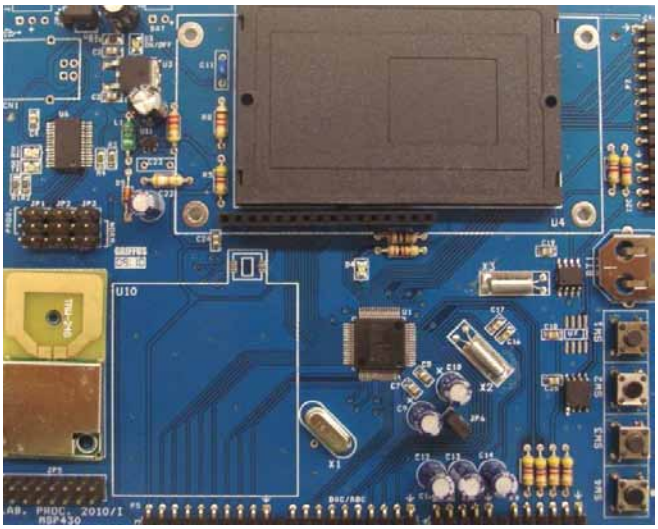

Figura 6: Placa de Laboratório de Processadores com MSP430 utilizada.

6. **Materiais Utilizados**

Além dos 7 servomotores mencionados anteriormente, também foram utilizados os seguintes materiais para a construção do braço robótico:

- Perfil de Liga de Alumínio
- Parafusos longos e curtos
- Porcas para os parafusos
- Arruelas
- Tubos de caneta esferográfica da marca BIC
- Extensores de fios de servomotores
- Cabeçote de Vídeo Cassete
- Caixas de MDF

Liga de Alumínio

Foi utilizada uma chapa de liga de alumínio, que consistem em alumínio e pequenas quantidades de cobre, manganês, silício, magnésio entre outros elementos. Essa chapa possuía 1 metro de comprimento, 2,8 centímetros de largura e uma profundidade de 1 milímetro. As características da liga metálica acrescida com a profundidade da chapa metálica foram decisivos para a escolha desse material para a utilização na construção da estrutura do braço robótico.

Devido às características de resistência mecânica da chapa metálica, a tarefa para construção do esqueleto do braço robótico ficou bastante facilitada. Foi necessário cortar a chapa metálica em alguns pontos. Para realizar estes cortes foi utilizada uma tesoura grande de metal. As dobras na chapa foram realizadas utilizando alicate e martelo.

Como a chapa de alumínio possuía 2,8 cm de largura, uma medida que não era apropriada para as dimensões projetadas do braço mecânico, decidiu-se cortar a chapa metálica pela metade. Com 1,8 cm de largura foi obtida a dimensão apropriada para a construção das principais unidades de sustentação do braço robótico.

7. **Dimensões do Projeto**

A figura 7 exibe as dimensões principais, da estrutura que interliga os servomotores 01, 02 e 03 do braço robótico. A figura 8 exibe as dimensões principais que envolvem os servomotores 04 e 05 (servomotores da pinça).

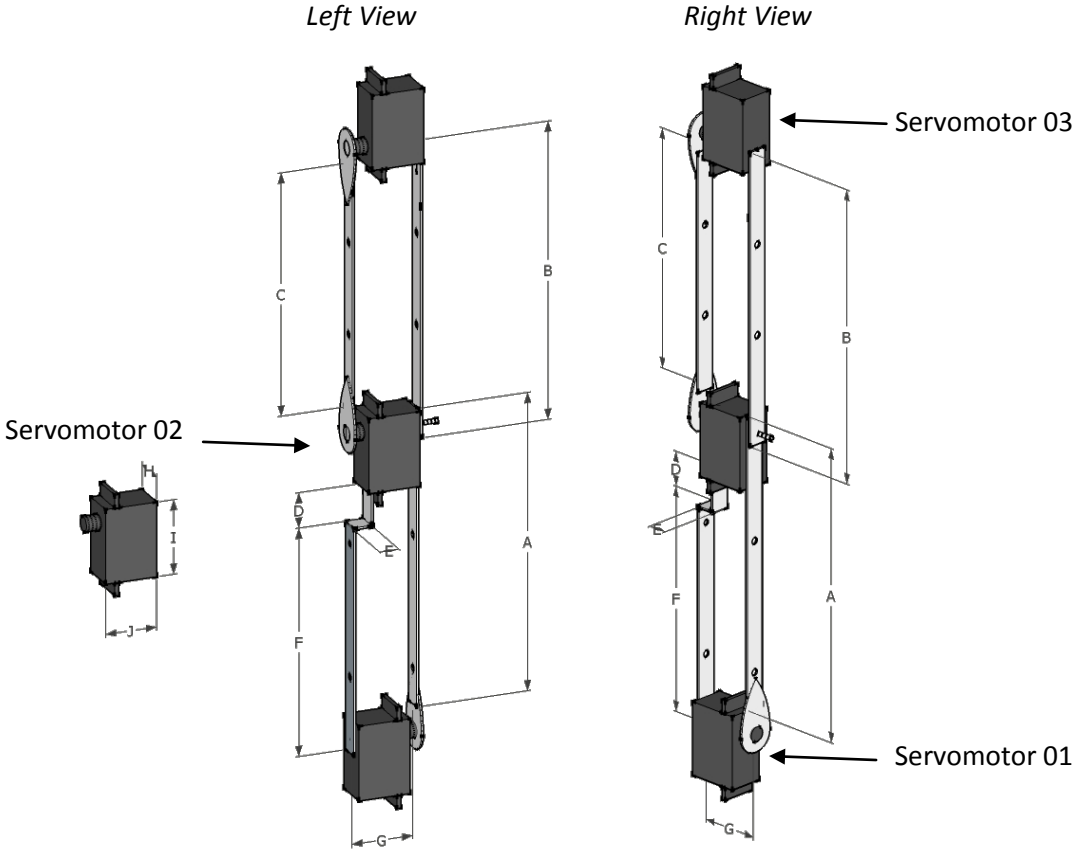

Figura 7: Dimensões que envolvem os servomotores 01, 02 e 03.

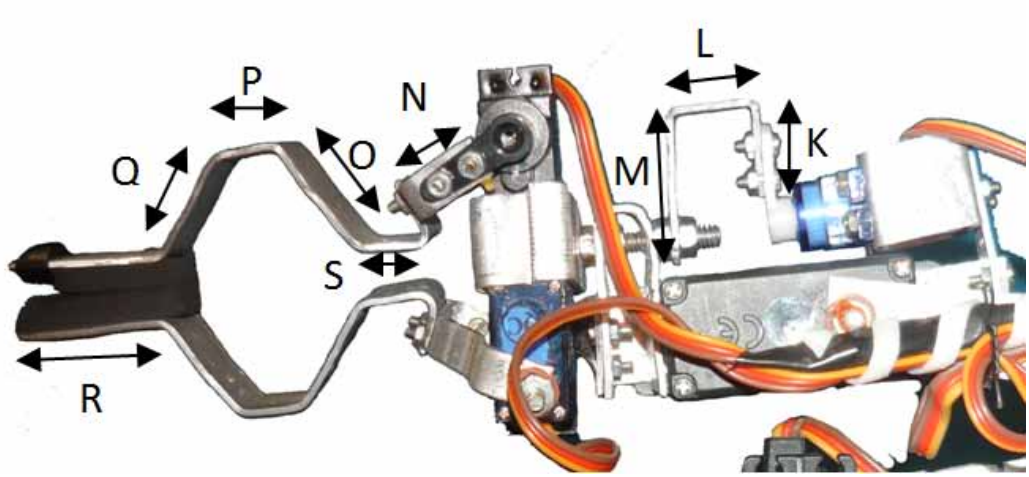

Figura 8: Dimensões que envolvem os servomotores 04 e 05.

Onde:

- A: 160mm
- B: 160mm
- C: 130,5mm
- D: 20mm
- E: 10,5mm
- F: 122,5mm
- G: 33mm
- H: 19mm
- I: 40mm
- J: 27,6mm
- K: 10,5mm
- L: 10,5mm
- M: 20,5mm
- N: 10,3mm
- O: 20,2mm
- P: 10mm
- Q: 20,2mm
- R: 20,5mm
- S: 10mm

8. **Requisitos**

Os itens abaixo são os requisitos necessários para o desenvolvimento do projeto. Definem o conjunto de diretrizes que regem a criação, desenvolvimento e estudo desse projeto.

- Construir do princípio um pequeno braço mecânico didático.
- O pequeno braço mecânico deve sustentar sua própria massa e ser capaz de manipular objetos de no máximo 50g.
- Os dados trafegados necessariamente devem percorrer um canal de comunicação sem fio.
- Identificar um dispositivo de manipulação inovador para a manipulação do braço robótico, e provar que o mesmo é eficaz nessa tarefa.
- Projeto de baixo custo.
- Utilizar tecnologias recentes e materiais de fácil acesso.

9. **Bluetooth**

Bluetooth é uma tecnologia de baixo consumo e baixo alcance, sem fios, originalmente desenvolvida para substituir cabos de modo a permitir a conexão de celulares, *headsets* e computadores. Desde então, tem evoluído para um padrão sem fio que conecta dispositivos eletrônicos para formar as chamadas *Personal Area Networks* (PANs), bem como redes *ad hoc*. Não somente os cabos são desnecessários para conectar os dispositivos, mas as conexões também são realizadas sem a necessidade de instalação de *softwares* de *drivers*. Com essa tecnologia, os dispositivos são capazes de descobrir qualquer outro dispositivo Bluetooth habilitado, determinar seus serviços e aplicações, bem como estabelecer conexões para troca de dados.

Bluetooth é uma solução sem fio para comunicar equipamentos entre si. Seus rádios estão divididos em três classes, conforme a *tabela 2*.

Classe	**Potência máxima permitida (mW/dBm)**	**Alcance (aproximadamente)**
Classe 1	100mW (20dBm)	Até 100 metros
Classe 2	2.5mW (4dBm)	Até 10 metros
Classe 3	1mW (0dBm)	~1 metro

Tabela 2: Características dos rádios Bluetooth.

Existe uma diversidade muito grande de produtos com diferentes tipos de interfaces de conexão, como cabos coaxiais, RS232, RS485, conexões paralela e serial de diversos tipos, etc. E com esses produtos vem uma variedade ainda maior de conectores, plugues e protocolos.

Esta diversidade dificulta a integração dos sistemas e para evitar combater esse problema, soluções como o Bluetooth vem surgindo, a qual não utiliza fios e evita a utilização de diversos protocolos para os mais variados equipamentos com as mais variadas funcionalidades que podem ocorrer.

A velocidade de transmissão de dados do Bluetooth é baixa, até a versão 1.2, a taxa pode alcançar, no máximo, 1 Mbps. Na versão 2.0, esse valor passou para até 3 Mbps. Essa é

uma preocupação constante do grupo de desenvolvimento do Bluetooth (SIG, *Special Interest Group*) e para a versão 3.0 já teremos uma taxa de transmissão de até 24Mbps.

O Bluetooth utiliza a freqüência de rádio aberta ISM (*Industrial*, *Scientifical*, *Medical*), que opera na freqüência de 2,45GHz e com isso é capaz de funcionar no mundo todo, com variações, dependendo do país, que vão de 2,4GHz a 2,5GHz.

Como a faixa ISM é aberta, isto é, pode ser utilizada por qualquer sistema de comunicação, é necessário garantir que o sinal do Bluetooth não sofra e nem gere interferências. O esquema de comunicação *FH-CDMA* (***F**requency **H**opping -**C**ode-**D**ivision **M**ultiple **A**ccess*), utilizado pelo Bluetooth, permite tal proteção, já que faz com que a frequência seja dividida em vários canais. O dispositivo que estabelece a conexão vai mudando de um canal para outro de maneira muito rápida. Esse esquema é chamado "salto de frequência" (*frequency hopping*). Isso faz com que a largura de banda da frequência seja muito pequena, diminuindo sensivelmente as chances de uma interferência. No Bluetooth, pode-se utilizar até 79 frequências (ou 23, dependendo do país) dentro da faixa ISM, cada uma espaçada da outra por 1 MHz.

Como um dispositivo se comunicando por Bluetooth pode tanto receber quanto transmitir dados (modo *full-duplex*), a transmissão é alternada entre slots para transmitir e slots para receber, um esquema denominado *FH/TDD* (***F**requency Hopping/**T**ime-**D**ivision **D**uplex*). Esses slots são canais divididos em períodos de 625 µs. Cada salto de frequência deve ser ocupado por um slot, logo, em 1 segundo, tem-se 1600 saltos.

No que se refere ao enlace, isto é, à ligação entre o emissor e receptor, o Bluetooth faz uso, basicamente, de dois padrões: *SCO* (***S**ynchronous**C**onnection-**O**riented*) e *ACL* (***A**synchronous **C**onnection-**L**ess*). O primeiro estabelece um link sincronizado entre o dispositivo master e o dispositivo escravo, onde é feito uma reserva de slots para cada um. Assim, o SCO acaba sendo utilizado principalmente em aplicações de envio contínuo de dados, como voz. Por funcionar dessa forma, o SCO não permite a retransmissão de pacotes de dados perdidos. Quando ocorre perda em uma transmissão de áudio, por exemplo, o dispositivo receptor acaba reproduzindo um som com ruído. A taxa de transmissão de dados no modo SCO é de 432 kbps, sendo de 64 kbps para voz.

O padrão ACL, por sua vez, estabelece um link entre um dispositivo mestre e os dispositivos escravos existentes em sua rede. Esse link é assíncrono, já que utiliza os slots

previamente livres. Ao contrário do SCO, o ACL permite o reenvio de pacotes de dados perdidos, garantindo a integridade das informações trocadas entre os dispositivos, assim, acaba sendo útil para aplicações que envolvam transferência de arquivos, por exemplo. A velocidade de transmissão de dados no modo ACL é de até 721 Kbps.

9.1. **Rede Bluetooth**

Quando dois ou mais dispositivos se comunicam através de uma conexão Bluetooth, eles formam uma rede denominada *piconet*. Nessa comunicação, o dispositivo que iniciou a comunicação assume o papel de mestre enquanto que os outros são escravos. É o mestre quem regula a transmissão de dados entre a rede e o sincronismo entre os dispositivos. Os escravos não podem mandar dados diretamente para outros escravos da rede, nesse caso, o mestre atua como um switch para a piconet e todo o tráfego deve passar pelo mestre e só depois ser direcionado para o destino. Qualquer dispositivo pode ser mestre ou escravo em uma rede *piconet* e eles podem trocar os papéis em qualquer ponto da conexão quando o escravo quiser tomar o papel de mestre. Pode haver no máximo 7 escravos ativos em uma rede *piconet* mas somente 1 será mestre.

Cada *piconet* pode suportar até 8 dispositivos (1 *master* e 7 *slaves*), no entanto, é possível fazer com esse número seja maior através da sobreposição de *piconets*. Em poucas palavras, isso significa fazer com que uma *piconet* se comunique com outra dentro de um limite de alcance, esquema esse denominado *scatternet*. Deve-se notar que um dispositivo slave pode fazer parte de mais de uma piconet ao mesmo tempo, no entanto, um *master* só pode ocupar essa posição em uma única *piconet*.

Cada dispositivo Bluetooth tem seu próprio *clock* e é identificado pelo seu endereço Bluetooth único. Escravos em uma *piconet* usam o endereço Bluetooth e o *clock* do mestre para determinar a sequência de *frequency hopping*. São adicionados offsets aos *clocks* nativos de cada um dos escravos para sincronizar com o *clock* do mestre durante a conexão. Além disso, o mestre também controla a transmissão de dados dos dispositivos. Os escravos somente podem transmitir quando estiver programado pelo mestre. O mestre decide como toda a banda disponível é distribuída para os escravos considerando a freqüência de comunicação com eles.

Um conjunto de duas ou mais *piconets* interconectadas formam *scatternets*. A *figura 9* mostra as *piconets* e *scatternets*. Uma unidade Bluetooth pode ser um escravo em duas ou

mais *piconets*, mas só pode ser mestre em uma única. Os dispositivos que participam de duas ou mais *piconets* podem atuar como *gateways*, repassando o tráfego de uma *piconet* para outra. Sabendo-se que unidades Bluetooth somente podem transmitir e receber dados em uma *piconet*, sua participação em muitas *piconets* é possível devido a multiplexação denominada de *Time Division Multiplex* (TDM). Isso significa que embora os dispositivos possam participar de várias *piconets*, eles só poderão estar ativos em uma única por vez. Os dispositivos participando de múltiplas *piconets* dividem seu tempo entre as *piconets*, gastando alguns *slots* de tempo em uma e outros *slots* em outra. As *piconets* podem ser identificadas pela identidade e o *clock* do mestre. Um dispositivo que deseje estar ativo em outra *piconet* deverá notificar o mestre da *piconet* atual que ele ficará inativo por um pré-determinado período de tempo. O dispositivo então vai ter de re-sincronizar seu *clock* (adicionando um *offset*) com o outro mestre. Quando o escravo se tornar inativo em uma *piconet*, as comunicações entre mestres e outros escravos ativos continuam normalmente. Por outro lado, quando um mestre se torna inativo na sua *piconet*, os escravos vão ter de esperar que o mestre volte a ficar ativo antes de poderem prosseguir com a comunicação.

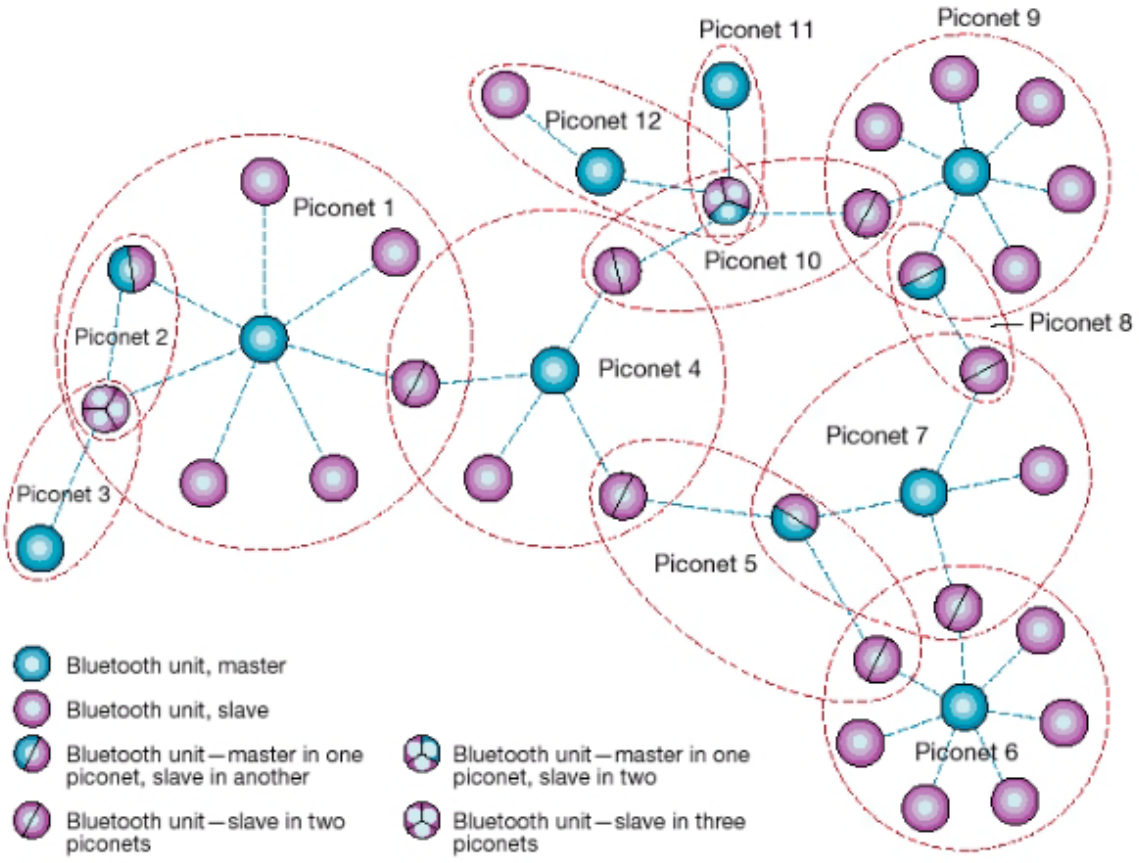

Figura 9: As *Piconets* e *Scatternets* do Bluetooth.

Para que cada dispositivo saiba quais outros fazem parte de sua *piconet*, é necessário fazer uso de um esquema de identificação. Para isso, um dispositivo que deseje estabelecer uma conexão em uma *piconet* já existente pode emitir um sinal denominado *Inquiry*. Os dispositivos que recebem o sinal respondem com um pacote *FHS* (***F****requency* ***H****opping* ***S****ynchronization*) informando a sua identificação e os dados de sincronismo da *piconet*. Com base nessas informações, o dispositivo pode então emitir um sinal chamado *Page* para estabelecer uma conexão com outro dispositivo.

Como o Bluetooth é uma tecnologia que também oferece como vantagem economia de energia, um terceiro sinal denominado *Scan* é utilizado para fazer com que os dispositivos que estiverem ociosos entrem em *stand-by*, isto é, operem em um modo de descanso, poupando eletricidade. Todavia, dispositivos neste estado são obrigados a "acordar" periodicamente para checar se há outros equipamentos tentando estabelecer conexão.

9.2. **Os Protocolos Bluetooth**

Os protocolos Bluetooth contém o padrão de procedimentos para conexões e troca de dados entre dispositivos Bluetooth. A *figura 10* mostra a pilha de protocolos do Bluetooth. O Rádio (*Radio*) é a interface entre o canal no ar e a Banda-base (*Baseband*). A camada de banda-base é responsável pela codificação e decodificação do canal. Ela digitaliza os sinais recebidos pelo rádio para passar para cima na pilha e formata o dado que recebe do *Link Controller* para transmissão no canal. O *Link Controller* é responsável por estabelecer e manter os links entre as unidades Bluetooth. O *Link Manager Protocol* (LMP) lida com o gerenciamento da *piconet* e a configuração do link. Ele também inclui procedimentos para melhorar a segurança do link, como encriptação e autenticação.

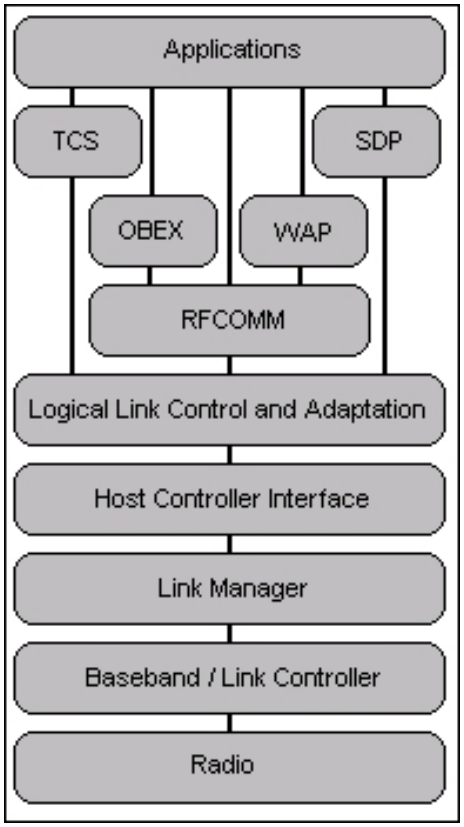

Figura 10: Pilha de protocolos Bluetooth.

A interface do *Host Controller* (HCI) define métodos uniformes para acessar e controlar as camadas inferiores da pilha de protocolos são elas a banda-base (*Baseband*) e o *link manager*. Diretamente acima disso, o *Logical Link Control and Adaption Protocol* (L2CAP) fornece serviços de dados orientados a conexão ou não para os outros níveis mais altos da camada de protocolos. Suas capacidades de multiplexação de protocolo permitem que diferentes protocolos e serviços usem um link de banda-base. Os protocolos restantes todos utilizam links L2CAP (e aqui são posicionados no topo daquele protocolo). O *Service Discovery Protocol* (SDP) define procedimentos para descobrir serviços de outros dispositivos, bem como determinar as características daqueles serviços. O protocolo RFCOMM define um protocolo de transporte para emulação de portas serial RS-232. A *Telephony Control Specification* (TCS) define a sinalização de controle de chamada para estabelecer conversa e chamadas de dados entre dispositivos Bluetooth, fornecendo eles com serviços telefônicos. O protocolo *Object Exchange* (OBEX) é uma especificação para a troca de objetos de dados através de links infravermelho (IR). Exemplos de uso do OBEX incluem a troca de cartões de negócios e sincronização de aplicações de calendário. A tecnologia Bluetooth utiliza a especificação do protocolo IrOBEX para permitir que as aplicações funcionem em ambos baixo-alcance RF e IR, permitindo que as aplicações escolham qual

usar. Da mesma forma, o protocolo *Wireless Application* (WAP) inclui requisitos de interoperabilidade para Bluetooth como um usuário WAP. Isso permite que um dispositivo use WAP através de links Bluetooth fornecendo serviços de valor agregado.

Existem três formas de implementar as pilhas de protocolo Bluetooth, como ilustrado na *figura 11*. Ela pode usar o padrão de arquitetura de dois processadores (figura 11a), arquitetura embarcada, ou de um único processador. No modelo de dois processadores, o host Bluetooth reside sobre o PC enquanto as camadas mais baixas do protocolo são encapsuladas em um módulo Bluetooth. Essa arquitetura é geralmente utilizada em módulos Bluetooth separados ou *dongles* USB para computadores pessoais e notebooks. A segunda aproximação (figura 11b) ainda usa dois processadores mas a maioria das camadas do protocolo estão no processador alvo. Essa arquitetura é geralmente utilizada em dispositivos com recursos limitados, tais como telefones celulares e *handhelds*. A terceira arquitetura (figura 11c) é de um único processador, usada em soluções com um único chip.

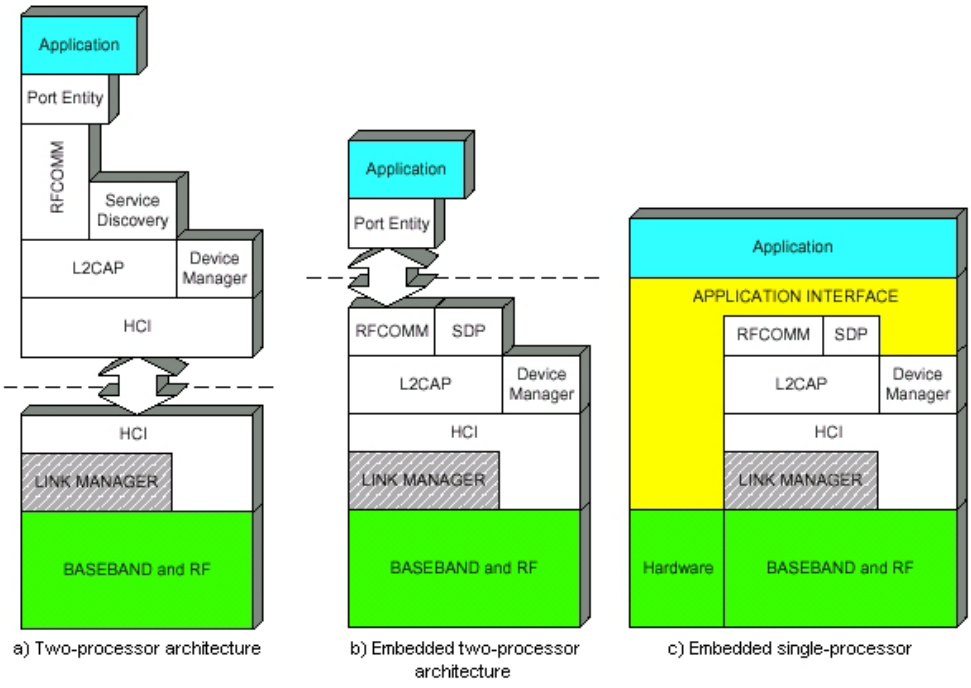

Figura 11: Particionamentos da pilha de protocolos Bluetooth.

9.3. **Perfis Bluetooth**

Os perfis Bluetooth contém regras de como as aplicações devem usar a pilha de protocolos Bluetooth. Essas regras descritas nos perfis garantem a interoperabilidade entre dispositivos independentemente de seus fabricantes. Todos os dispositivos que implementarem certas características o farão de acordo com o padrão especificado.

Perfis podem ser construídos em cima de outros perfis permitindo o reuso de características e funcionalidades descritas em outros perfis. Como pode ser visto na *figura 12*, todos os perfis são construídos em cima da *Generic Access Profile* (GAP).

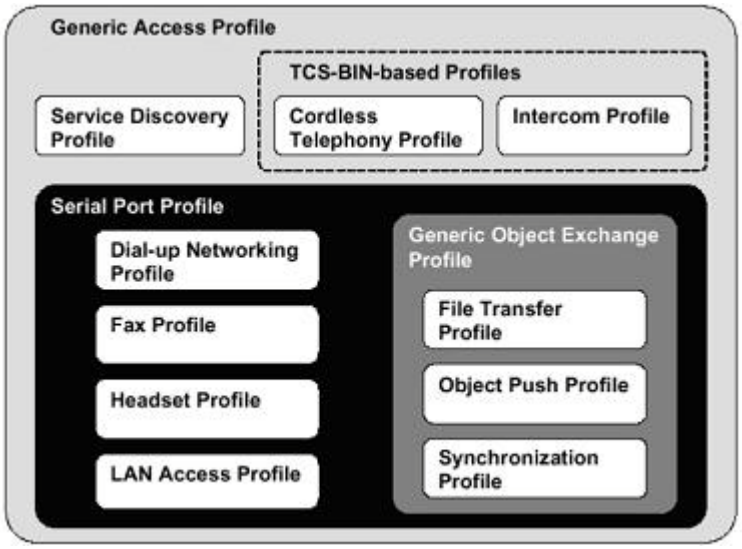

Figura 12: Generic Access Profile (GAP).

Os dispositivos não necessitam implementar todos os perfis. Um dispositivo somente precisa implementar os perfis necessários para suas aplicações. Uma excessão, entretanto, é a *Generic Access Profile*, a qual é requerida por todos os dispositivos. Esse perfil define o seguinte:

- Requisitos para características que devem ser implementadas em todos os dispositivos;

- Procedimentos genéricos para a descoberta de dispositivos Bluetooth;

- Facilidades no gerenciamento do link para conectar dispositivos Bluetooth;

- Procedimentos relacionados ao uso de diferentes níveis de segurança;

- Requisitos de formato comum para parâmetros de dispositivo acessíveis no nível de interface com o usuário.

Os procedimentos de descoberta e conexão garantem que todos os dispositivos Bluetooth irão reconhecer outros dispositivos Bluetooth e serem capazes de conectar-se entre si. Tendo as características necessárias para unidades Bluetooth vai permitir que

desenvolvedores façam suposições sobre outros dispositivos quando estiverem desenvolvendo suas aplicações. Por outro lado, terminologias a serem usadas no nível de interface com o usuário vão ajudar usuários a reconhecer as características e funções de diferentes dispositivos, plataformas e interfaces.

Quando a versão 1.1 da *Bluetooth Profile Specification* saiu, 13 perfis estavam descritos. Desde lá, muitos novos perfis foram criados, incluindo o *Human Interface Device Profile*, o *Personal Area Networking Profile* e o *Basic Printing Profile*. O *Human Interface Device Profile* define regras para dispositivos como teclados, *joysticks*, e outros controles utilizados para jogos e simuladores. O *Personal Area Networking Profile* descreve como dois ou mais dispositivos Bluetooth formam redes ad hoc e como podem usar pontos de acesso para acessar redes. A *Basic Printing Profile* define os requisitos e processos para suportar o modelo de uso da *Basic Printing*. Esse modelo de uso é definido para telefones celulares e PDAs para imprimir mensagens de texto, pequenas mensagens de email, e outros documentos formatados.

9.4. **Segurança**

O esquema de *frequency hopping* utilizado pela tecnologia Bluetooth já faz com que a escuta de links Bluetooth seja muito difícil. De fato, o exército dos EUA considera que um link para comunicação usando *frequency hopping* com 79 canais seja seguro. Ainda assim, o Bluetooth oferece encriptação e autenticação utilizando um algoritmo baseado no encriptador SAFER+ (*Secure And Faster Encryption Routine*). Esse algoritmo fera chaves criptografadas de 128 bits através de uma entrada de texto de 128 bits. Quando é inicializado um processo de segurança, uma chave de 128 bits gerada por um *Personal Identification Number* (PIN), o endereço do dispositivo Bluetooth requerente, e um número aleatório compartilhado entre o requerente e o verificador. O processo de autenticação verifica se os dois dispositivos estão usando a mesma chave de 128 bits para verificar que o mesmo número PIN foi introduzido nos dois dispositivos. Se o processo de autenticação tem sucesso, uma nova chave de 128 bits é gerada usando um novo número aleatório de cada unidade, os endereços Bluetooth das duas unidades e a chave atual de 128 bits. Essa chave é usada para produzir a string criptografada para criptografar e descriptografar os dados transmitidos.

O Bluetooth também apresenta três modos de segurança, que podem ser usados pelas aplicações dependendo dos seus requerimentos de segurança. O modo 1 não é seguro. Um dispositivo no modo 1 nunca inicia qualquer procedimento de segurança. Um dispositivo

operando no modo 2 aplica processos de segurança em um nível de serviço. Somente após um canal L2CAP ter sido estabelecido que algum procedimento de segurança será executado. Dependendo da aplicação, isso pode incluir autorização, autenticação e encriptação. Por outro lado, um dispositivo com modo de segurança 3 aplica procedimentos de segurança no nível de link. Então, quando um dispositivo falha nas medidas de segurança executadas por um dispositivo conectado, não é estabelecido o link entre os dois.

Em adição a essas medidas de segurança, os dispositivos podem também ficar “invisíveis” caso queiram ficar escondidos. Bluetooth permite que um dispositivo fique no modo de não poder ser descoberto.

10. **Hardware**

Para a interface entre o braço robótico e o controle remoto Bluetooth utilizamos a placa de Laboratório de Processadores com microcontrolador MSP430 e para a comunicação Bluetooth adquirimos um módulo Bluetooth da *SURE Electronics* (figura 13) ao preço de R$100,00.

10.1. **Módulo SURE**

O módulo utilizado é o *Bluetooth Serial Converter, UART interface* de 9600bps (GP-GC021).

- Compatível com a especificação Bluetooth 2.0+EDR;

- *Enhanced Data Rate* (EDR) com especificação V2.0.E.2 para ambos modos de modulação de 2Mbps e 3Mbps;

- Potência de saída tipo classe 2;

- Opera com velocidade máxima Bluetooth e suporte completo a *Piconet*;

- Suporta *Scatternet*;

- Opera na tensão de 3,3V;

- Interface UART;

- Suporta Flash externa de 8Mb;

- Suporta coexistência com o protocolo 802.11;

- Compatível com RoHS.

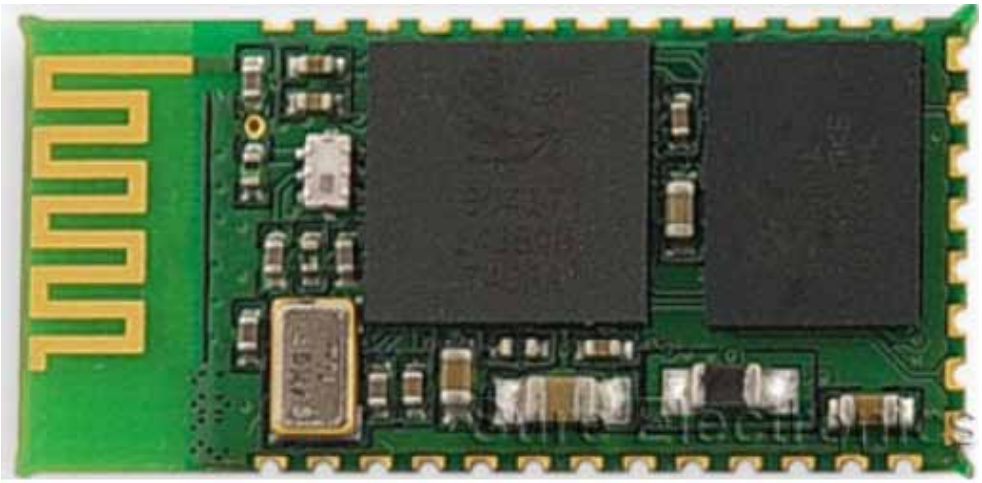

Figura 13: Foto do módulo Bluetooth Sure Electronics.

Esse é um módulo Bluetooth classe 2 que utiliza o *chipset* BlueCore4-AudioROM do fabricante *Cambridge Silicon Radio*.

Trata-se de um conversor que é usado para converter uma UART de 9600bps, um *start bit*, oito bits de dados, um *stop bit*, sem bit de paridade, para o protocolo UART Bluetooth. Ele não pode se comunicar com um dispositivo igual a ele, somente com um *dongle* Bluetooth para PC. Assim possibilitamos a comunicação entre a placa com MSP430 e um PC, que faz papel de *bridge* em nosso sistema. O controle Bluetooth conecta-se ao PC que faz a interface para o módulo Bluetooth da SURE.

As tabelas 3, 4 e 5 mostram todas as características do chip.

Especificações

Banda de freqüência de operação	2,4GHz-2,48GHz na banda não licenciada ISM
Especificação Bluetooth	V2.0+EDR
Classe de potência de saída	Classe 2
Tensão de operação	3,3V
Interface do Host	UART
Dimensões	26,9mm(C)x13mm(L)x2.2mm(A)

Tabela 3: Especificações do módulo SURE GP-GC021.

Características Elétricas

Estimativa	**Mínimo**	**Máximo**
Temperatura de armazenamento	-40°C	+150°C
Faixa de desempenho garantido RF	-40°C	+150°C
Tensão de alimentação	2,2V	4,2V

Tabela 4: Características elétricas do módulo SURE GP-GC021.

Consumo

Modo de operação	Tipo de conexão	Taxa da UART (kbps)	Média de consumo	Unidade
Scan de página	-	115,2	0,42	mA
ACL sem tráfego	Mestre	115,2	4,60	mA
ACL com transferência de arquivo	Mestre	115,2	10,3	mA
ACL 1,28s Sniff	Mestre	38,4	0,37	mA
ACL 1,28s Sniff	Escravo	38,4	0,42	mA
Conexão com Host em Standby	-	38,4	40	µA

Tabela 5: Consumo do módulo SURE GP-GC021.

Diagrama de blocos

A *figura 14* mostra o diagrama de blocos do chip da SURE.

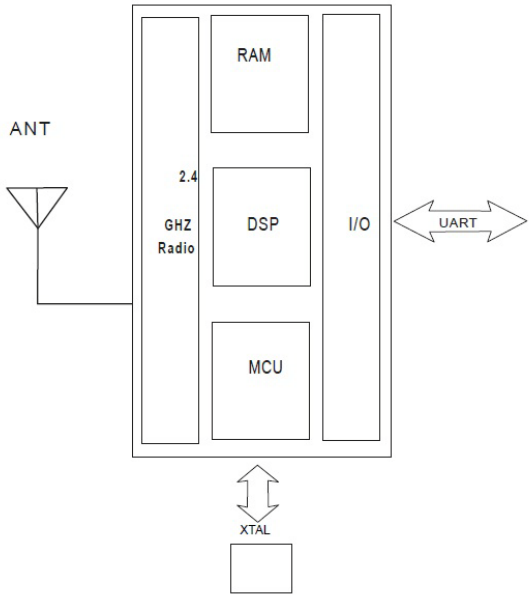

Figura 14: Diagrama de blocos do chip da Sure.

Dimensões

A *figura 15* mostra as dimensões do chip.

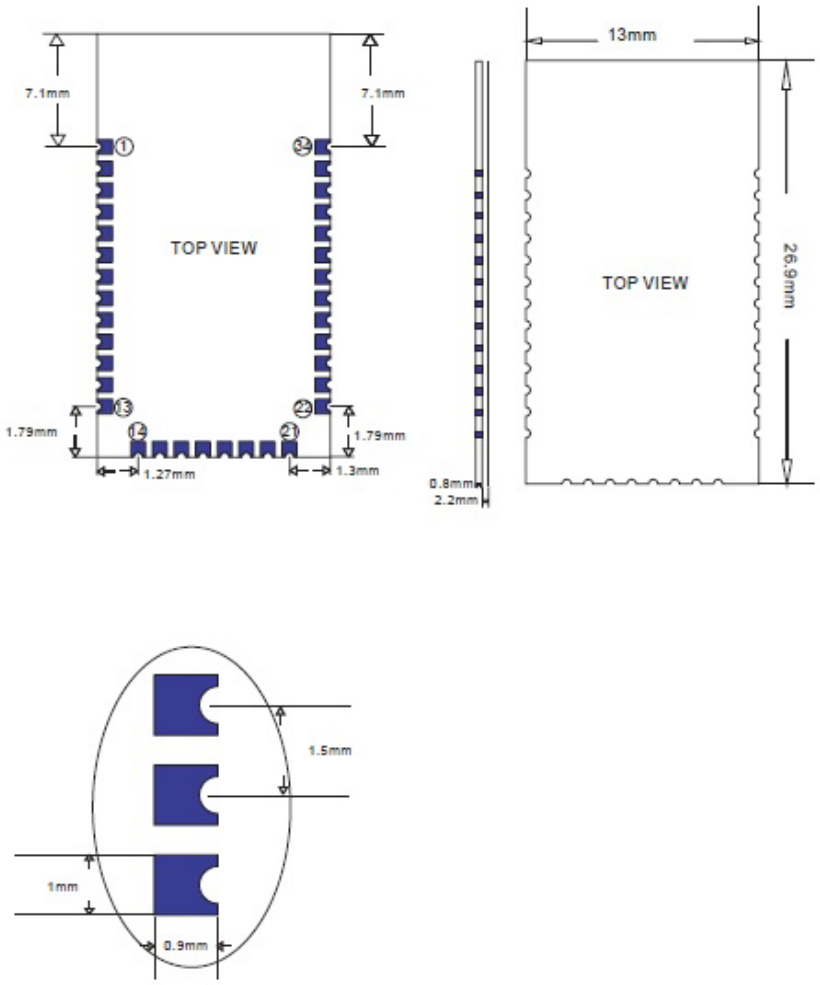

Figura 15: Dimensões do chip da Sure.

10.2. **Dispositivo de Manipulação**

Foi escolhido como dispositivo de manipulação do braço robótico o controle remoto do videogame Nintendo Wii, chamado *Wiimote*. Esse controle tem por característica ser complementado por um controle adicional, o *nunchuck*. A *figura 16* ilustra os controles remotos.

Figura 16: Wiimote a direita, com o nunchuck conectado.

10.3. **Por que o controle do Wii**

Tivemos a idéia de desenvolver o braço mecânico e depois pensamos que o mesmo poderia ser controlado via um controle sem fios, então decidimos usar o protocolo Bluetooth.

Para isso descobrimos que o controle do vídeo-game Wii da Nintendo é Bluetooth e possui muitas funcionalidades e serviria muito bem para controlar o braço robótico, dando várias possibilidades de movimentos para ele.

Através de pesquisa na internet descobrimos que o controle do Wii já havia passado por um processo de engenharia reversa e a maioria de suas funcionalidades já estavam disponíveis na internet em código C aberto. Então foi utilizada essa biblioteca no desenvolvimento do Braço Robótico com Controle Remoto.

11. **Wiiuse**

A biblioteca encontra-se disponível em www.wiiuse.net. Utilizamos a versão disponível para Windows, para isso bastou ter o *Microsoft Visual Studio 2008* conseguido por meio do convênio Microsoft-PUCRS. Lá criamos o nosso projeto incluindo a biblioteca wiiuse.h, e a dll wiiuse.dll.

Abaixo um sumário sobre a biblioteca:

Funções da API

Temos as seguintes funções disponíveis na API:

- wiiuse_init()
- wiiuse_cleanup()
- wiiuse_version()
- wiiuse_find()
- wiiuse_connect()
- wiiuse_disconnect()
- wiiuse_poll()
- wiiuse_poll()
- wiiuse_rumble()
- wiiuse_toggle_rumble()
- wiiuse_set_leds()
- wiiuse_motion_sensing()
- wiiuse_read_data()
- wiiuse_write_data()
- wiiuse_status()
- wiiuse_get_by_id()
- wiiuse_set_flags()
- wiiuse_set_smooth_alpha()
- wiiuse_set_bluetooth_stack()
- wiiuse_set_orient_threshold()
- wiiuse_set_accel_threshold()
- wiiuse_set_nunchuck_orient_threshold()
- wiiuse_set_nunchuck_accel_threshold()

- wiiuse_resync()
- wiiuse_set_timeout()
- wiiuse_set_ir()
- wiiuse_set_ir_vres()
- wiiuse_set_ir_position()
- wiiuse_set_ir_sensitivity()
- wiiuse_set_aspect_ratio()

11.1. Encontrando e se conectando a *wiimotes*

Podemos nos conectar a tantos *wiimotes* quantos o nosso sistema permitir. Primeiramente é necessário inicializar o número máximo de *wiimotes* que quisermos que o nosso programa use. Não há problema em inicializar menos controles do que o número definido. No nosso caso utilizaremos apenas um *wiimote*.

Inicializamos os controles através da função *wiiuse_init()*.

Através da seguinte linha inicializamos nosso único *wiimote*:

```
wimote** wiimotes = wiiuse_init(1);
```

Cada *wiimote* inicializado possui um identificador único associado ao controle . A função *wiiuse_init()* retorna um vetor de ponteiros para o objeto *wiimote* que foi inicializado mas não conectado.

Depois desta etapa é necessário encontrar alguns *wiimotes* que estão no modo de descoberta. A função *wiiuse_find()* é responsável por esta etapa.

```
int found = wiiuse_find(wiimotes, 1, 5);
```

Aqui escaneamos a porta COM em busca de um *wiimote* assim como inicializado acima durante um tempo máximo de 5 segundos. A informação de cada controle (no nosso caso apenas um) é armazenada no vetor de controles retornado pela função *wiiuse_init()*. A função irá retornar o número máximo de controles encontrado.

Se econtrarmos alguns controles vamos conectar a eles usando a função *wiiuse_connect()*. Para isso definimos o número máximo de controles no vetor, não quantos

foram encontrados. A função irá retornar o número total de *wiimotes* que conectaram com sucesso.

```
int connected = wiiuse_connect(wiimotes, 1);
 if (connected)
      printf("Connected to %i wiimotes (of %i found).\n", connected, found);
else {
      printf("Failed to connect to any wiimote.\n");
      return 0;
}
```

Depois disso, os controles estão conectados e podemos nos comunicar com eles.

Para desconectar algum controle chama-se a função *wiiuse_disconnect()* com o objeto particular do *wiimote* o qual se deseja desconectar:

```
void wiiuse_disconnect(struct wiimote_t* wm);
```

Ou então podemos chamar a função *wiiuse_shutdown()* para desconectar e limpar todas as conexões dos controles:

```
void wiiuse_cleanup(struct wiimote_t** wm, int wiimotes);
```

wiimotes é o tamanho do vetor *wm* que foi passado por *wiimote_init()*.

11.2. Auto-Detecção da Pilha Bluetooth do Windows

A função *wiiuse_find()* vai tentar detectar automaticamente a pilha Bluetooth que roda no sistema.

Se a biblioteca não conseguir encontra a pilha correta para usar, ou se já soubermos qual pilha utilizar, podemos manualmente configura-la antes de chamar a função *wiiuse_find()* com a seguinte função:

```
void wiiuse_set_bluetooth_stack(struct wiimote_t** wm, int wiimotes, enum
win_bt_stack_t type);
```

Aqui o parâmetro *wiimotes* é o tamanho do vetor que foi passado para a função *wiimote_init()*.

O parâmetro *type* pode ser qualquer uma das estruturas:

- WIIUSE_STACK_BLUESOLEIL

Esse parâmetro dirá a biblioteca para utilizar a pilha BlueSoleil.

- WIIUSE_STACK_MS

Esse parâmetro diz para a biblioteca utilizar outras pilhas.

11.3. O Sistema de Polling

A biblioteca *wiiuse* funciona como um sistema de **polling não bloqueante**. Isso significa que precisamos constantemente dizer ao *wiiuse* para procurar por eventos.

A função *wiiuse_poll()* vai retornar o número de *wiimotes* que tiveram um evento detectado. Se o número retornado for zero, não precisamos fazer nada, por outro lado deveremos percorrer cada *wiimote* e procurar qual evento foi detectado.

O código seguinte mostra o loop que utilizamos para detectar eventos nos controles:

```
while (1) {
    if (wiiuse_poll(wiimotes, 2)) {
        int i = 0;
        for (; i < 2; ++i) {
            switch (wiimotes[i]->event) {
            /* procura por eventos aqui */
            }
        }
    }
}
```

A variável *wiimote_t::event* é modificada pela função *wiiuse_poll()* para indicar se um evento aconteceu no *wiimote* que está no poll. A variável *event* é modificada para qualquer dos seguintes estados:

- WIIUSE_NONE

Nenhum evento aconteceu no *wiimote*.

- WIIUSE_EVENT

Um evento genérico aconteceu no *wiimote*.

- WIIUSE_STATUS

Um relatório de status foi obtido do *wiimote*.

- WIIUSE_DISCONNECT

O *wiimote* se desconectou.

- WIIUSE_READ_DATA

O dado que foi anteriormente requisitado foi retornado dos registradores ou da ROM do *wiimote*.

- WIIUSE_NUNCHUCK_INSERTED

O acessório do controle do *Wii* chamado de *Nunchuck* foi inserido.

Este é um caso especial de evento para WIIUSE_STATUS.

- WIIUSE_NUNCHUCK_REMOVED

Um *nunchuck* foi removido.

Este é um caso especial de evento para WIIUSE_STATUS.

- WIIUSE_CLASSIC_CTRL_INSERTED

Um controle clássico foi inserido.

Este é um caso especial de evento para WIIUSE_STATUS.

- WIIUSE_CLASSIC_CTRL_REMOVED

Um controle clássico foi removido.

Este é um caso especial de evento para WIIUSE_STATUS.

- WIIUSE_GUITAR_HERO_3_CTRL_INSERTED (não utilizado no projeto)

Um controle chamado *Guitar Hero 3* foi inserido.

Este é um caso especial de evento para WIIUSE_STATUS.

No Windows um sistema de *timeout* é usado quando acontece o *polling* nos *wiimotes*. Se acharmos que o *wiimote* está respondendo muito lentamente podemos tentar diminuir os valores de *timeout*, entretanto, diminuir muito esses valores pode causar outros problemas.

Os *timeouts* são dados em milisegundos.

Existem dois tipos de valores para *timeout*:

- O *timeout* normal. É usado para o *polling* normal.

- *Timeout* de expansão. É usado quando uma expansão é detectada até que realize o *handshake* com sucesso.

O *timeout* normal é usado sempre, exceto quando uma expansão é conectada pela primeira vez. Quando uma expansão é detectada, o *wiiuse* vai começar usando o *timeout* de expansão para aquele *wiimote* até que a expansão termine seu *handshake*.

Se acharmos que expansões não estão sendo detectadas propriamente podemos tentar aumentar o *timeout* de expansão. Isso vai causar a pausa do *wiiuse* por um tempo maior para esperar que o *handshake* termine antes de reverter para o valor normal de *timeout*.

A função é:

```
void wiiuse_set_timeout(struct wiimote_t** wm, int wiimotes, byte normal_timeout,
byte exp_timeout);
```

11.4. O Evento Generic

O evento *event* é modificado pelo *wiiuse* quando um evento genérico acontece em um *wiimote*.

Um evento é gerado quando acontece uma mudança significativa de estado.

Pode ser considerado como mudança significativa de estado o seguinte:

1. Botão pressionado;
2. Botão solto;
3. Movimento do joystick;
4. A orientação do dispositivo (quando a sensibilidade ao movimento está habilitada) se modificou significativamente;
5. A posição em que a câmera IR (Infravermelho) está apontando mudou.

11.5. Limiar de Orientação

O acelerômetro é muito sensível e produz muito ruído. Por causa disso, o ângulo está quase sempre se modificando em cada chamada a função *wiiuse_poll()*, significando que cada chamada vai resultar em aumento de um evento genérico se a sensibilidade ao movimento (*motion sensing*) está habilitada. Para evitar problemas, o *wiiuse* somente vai gerar um evento para a sensibilidade de movimento se acontecer uma mudança significativa na orientação do controle, ou se algum ângulo se modificar em um grau particular.

Por padrão, esse limiar é a metade de um grau (0,5 graus). Isso significa que se o ângulo mudar por menos do que isso um evento não vai ser gerado.

Podemos mudar o limiar de orientação chamando a seguinte função:

```
void wiiuse_set_orient_threshold(struct wiimote_t* wm, float threshold);
```

O parâmetro *threshold* (limiar) é a quantidade de graus que um ângulo (roll, pitch ou yaw) deve mudar para gerar um evento.

Deve-se notar que essa função leva somente uma estrutura de *wiimote* e com isso podemos dinamicamente fazer com que um *wiimote* seja mais sensível que outro. Um bom tempo para definir um diferente limiar se quisermos que seja aplicado a todos os controles seria após a chamada de *wiimote_init()*.

11.6. Limiar de Aceleração

A função

```
void wiiuse_set_orient_threshold(struct wiimote_t* wm, float threshold);
```

Funciona do mesmo jeito que a função *wiiuse_set_orient_threshold()* mas se aplica aos valores de aceleração ao invés dos valores de orientação.

11.7. O Evento de Status

O evento de status é modificado pelo *wiiuse* quando ocorre uma mudança de status em um *wiimote*.

Uma mudança de status acontece quando uma das seguintes condições são atingidas:

1. Uma extensão foi conectada na porta de extensão do *wiimote*.
2. Uma extensão foi desconectada da porta de extensão do *wiimote*.
3. A função *wiiuse_status()* foi chamada e o *wiimote respondeu.*

Informações importantes que podem ser obtidas através de um evento de status incluem:

- Se existe algum acessório conectado;

- Se o *speaker* está habilitado;

- Quais LED's estão ligados;

- Qual o nível restante de vida da bateria do controle (flutua entre 0 e 1).

11.8. Evento de Desconexão

Um evento de desconexão é modificado pela *wiiuse* quando um *wiimote* se desconectou.

Uma desconexão ocorre quando uma das seguintes condições são atingidas:

1. A conexão cai.
2. O botão *POWER* do controle é segurado por dois segundos.
3. A bateria termina e o controle se desliga.

11.9. Evento de Leitura de Dados

O evento de leitura de dados é definido pela *wiiuse* quando o *wiimote* retorna dados que foram requisitados previamente para serem lidos ou da memória ROM ou dos registradores.

Os dados podem ser lidos usando a seguinte função:

```
int wiiuse_read_data(struct wiimote_t* wm, byte* buffer, unsigned int offset, unsigned short len);
```

Onde *buffer* é um *buffer* alocado suficientemente grande para poder armazenar o dado a ser lido, *offset* é o endereço a ser lido do *wiimote* e *len* é o tamanho do bloco a ser lido.

Quando um evento de leitura é retornado podemos obter o dado da variável *wiimote_t::read_req*. Por exemplo, para ter certeza que o evento corresponde ao que solicitamos, devemos verificar se a variável *wiimote_t.read_req.addr* apresenta o mesmo valor de *offset* que passamos para a função *wiiuse_read_data()*. O dado retornado fica armazenado em *wiimote_t.read_req.buf*.

11.10. A Estrutura do Wiimote

A estrutura do *wiimote* é passada a cada chamada e tem todas as informações relacionadas ao estado corrente e configuração dos dispositivos. Essa estrutura é *read only* e deveria ser tratada como tal.

Somente membros chave são listados aqui, a estrutura completa está definida no cabeçalho *wiiuse.h*.

int unid

É o identificador único relacionado ao *wiimote* durante o estágio *wiiuse_init()*.

struct expansion_t exp

O dispositivo de expansão conectado no *wiimote*.

struct orient_t orient

A orientação do acelerômetro em cada eixo. Essa estrutura possui roll e pitch flutuando entre -180 e 180 graus.

O *yaw* flutua entre -26 e 26 graus e pode ser calculado somente se o IR estiver habilitado.

Se o IR estiver desabilitado, o *yaw* será sempre 0.

Essa estrutura também possui as variáveis *a_roll* e *a_pitch* que são o valor absoluto de roll e pitch. Esses valores representam o exato valor de roll e pitch que o *wiimote* reporta para um evento.

O acelerômetro produz muito ruído, por isso, para reduzir esse efeito o *wiiuse* implementa uma média de movimento exponencial para cada ângulo.
Podemos usar a função *wiiuse_set_smooth_alpha()* para mudar o valor alpha da equação.
Podemos também desabilitar essa característica desabilitando a flag correspondente com a função *wiiuse_set_flags()*.

struct gforce_t gforce

A força da gravidade em cada eixo reportada pelo acelerômetro.
O acelerômetro é sensível em mais ou menos 3 unidades de gravidade.
Essa estrutura tem um *x*, *y* e *z*.

Por exemplo, se *y* é 2,3 então existem 2,3 unidades de gravidade aplicados na direção positiva do eixo *y*.

struct ir_t ir

Essa estrutura tem toda a informação relacionada ao dispositivo de IR.

unsigned short btns

Os botões que acabaram de ser pressionados no evento atual.

unsigned short btns_held

Os botões que estão sendo pressionados.

unsigned short btns_released

Os botões que acabaram de ser liberados no evento atual.

11.11. A Estrutura de Expansão

A estrutura de expansão guarda o tipo de expansão que está conectada na porta de expansão e também todo o dado associado.

type

O parâmetro *type* guarda o tipo de expansão que está conectada, ele pode ser um dos seguintes:

- EXP_NONE
- EXP_NUNCHUCK
- EXP_CLASSIC
- EXP_GUITAR_HERO_3

Baseado em qual tipo é, podemos acessar o dado da extensão associada por um dos seguintes membros:

- struct nunchuck_t nunchuck
- struct classic_ctrl_t classic
- struct guitar_hero_3_t gh3

11.12. A Estrutura do Nunchuck

A estrutura do nunchuck é acessível através da estrutura do *wiimote* e tem toda a informação relacionada ao estado corrente e configuração dos dispositivos. Essa estrutura é *read only* e deve ser tratada como tal.

Somente membros chave são listados aqui, a estrutura completa está definida no cabeçalho *wiiuse.h*.

struct orient_t orient
A orientação do acelerômetro em cada eixo.

struct gforce_t gforce
A força da gravidade em cada eixo assim como reportada pelo acelerômetro.

struct joystick_t js
O joystick tem os membros *min*, *max* e *center*, cada um tem um byte para *x* e outro para *y* (unsigned char).

Os membros mais importantes são o *ang* e *mag*.

O *ang* é o ângulo no qual o joystick está sendo mantido.
Totalmente apontado para cima o ângulo é 0 graus, para a direita é 90 graus, para baixo é 180 graus e para a esquerda é de 270 graus.

mag é a magnitude em que o joystick está sendo mantido.
Quando estiver no centro o valor é 0 e para pontas distantes é 1. Então se a magnitude é de 0,5 então o joystick está na metade do caminho entre o meio e o centro.

byte btns
Os botões que acabaram de ser pressionados no evento atual.

byte btns_held
Os botões que estão sendo pressionados.
byte btns_released
Os botões que acabaram de ser liberados no evento atual.

11.13. A Estrutura do Controle Clássico

A estrutura do controle clássico é acessível através da estrutura do *wiimote* e tem toda a informação relacionada ao estado e configuração atual dos dispositivos.

Essa estrutura é *read only* e deve ser tratada como tal.

Somente membros chave são listados aqui, a estrutura completa está definida no cabeçalho *wiiuse.h*.

struct joystick_t ljs
Botão para a esquerda do joystick.
struct joystick_t rjs
Botão para a direita do joystick.
struct btns
Os botões que acabaram de ser pressionados no evento atual.
struct btns_held
Os botões que estão sendo pressionados.
struct btns_released
Os botões que acabaram de ser liberados no evento atual.
float whammy_bar
Essa variável é análoga a "botão".
Flutua entre 0 (não pressionado) e 1 (totalmente pressionado). Então 0,5 é pressionado até a metade.

11.14. Estado dos Botões

Para descobrir o estado dos botões (se pressionado ou não) fazemos uso de algumas macros.

Macros:
- IS_PRESSED()
Vai retornar 1 se o botão especificado está pressionado.

```
if (IS_PRESSED(dev, button)) {
        /* botão está pressionado */
} else {
        /* botão não está pressionado */
}
```

- IS_JUST_PRESSED()

Vai retornar 1 se o botão especificado acaba de ser pressionado no evento atual.

```
if (IS_JUST_PRESSED(dev, button)) {
        /* botão acaba de ser pressionado */
 } else {
        /* botão não acabou de ser pressionado */
}
```

- IS_RELEASED()

Vai retornar 1 se o botão especificado tiver acabado de ser liberado no evento atual.

```
if (IS_RELEASED(dev, button)) {
        /* botão acaba de ser liberado */
} else {
       /* botão não acaba de ser liberado */
}
```

- IS_HELD()

Vai retornar 1 se o botão especificado está sendo pressionado (isto é, foi pressionado em um momento anterior mas ainda não foi liberado).

```
if (IS_HELD(dev, button)) {
        /* botão está sendo segurado */
 } else {
       /* botão não está sendo segurado */
 }
```

Códigos dos botões do *wiimote*:

- WIIMOTE_BUTTON_ONE
- WIIMOTE_BUTTON_TWO
- WIIMOTE_BUTTON_B
- WIIMOTE_BUTTON_A
- WIIMOTE_BUTTON_MINUS
- WIIMOTE_BUTTON_HOME
- WIIMOTE_BUTTON_LEFT
- WIIMOTE_BUTTON_RIGHT
- WIIMOTE_BUTTON_DOWN
- WIIMOTE_BUTTON_UP
- WIIMOTE_BUTTON_PLUS

Códigos dos botões do nunchuck:

- NUNCHUK_BUTTON_C
- NUNCHUK_BUTTON_Z

Códigos do botões do controle clássico:

- CLASSIC_CTRL_BUTTON_UP
- CLASSIC_CTRL_BUTTON_LEFT

- CLASSIC_CTRL_BUTTON_DOWN
- CLASSIC_CTRL_BUTTON_RIGHT
- CLASSIC_CTRL_BUTTON_X
- CLASSIC_CTRL_BUTTON_A
- CLASSIC_CTRL_BUTTON_Y
- CLASSIC_CTRL_BUTTON_B
- CLASSIC_CTRL_BUTTON_PLUS
- CLASSIC_CTRL_BUTTON_HOME
- CLASSIC_CTRL_BUTTON_MINUS
- CLASSIC_CTRL_BUTTON_ZR
- CLASSIC_CTRL_BUTTON_ZL
- CLASSIC_CTRL_BUTTON_FULL_R *(R fully pressed)*
- CLASSIC_CTRL_BUTTON_FULL_L *(L fully pressed)*

11.15. Sensibilidade ao Movimento (*Motion Sensing*)

Para habilitar a sensibilidade ao movimento para um *wiimote* precisamos habilitá-lo chamando a função *wiiuse_motion_sensing()*.

Essa função pode também ser utilizada para desabilitar a sensibilidade ao movimento.

Sabendo que o acelerômetro produz muito ruído, *wiiuse* vai gerar um evento somente se algum ângulo mudar para um grau significativo.

A função é a seguinte:

```
void wiiuse_motion_sensing(struct wiimote_t* wm, int status);
```

Onde o status é 1 para habilitar o acelerômetro e 0 para desabilitar.

11.16. Verificando os Estados do *Wiimote*

Algumas macros são dadas para facilmente descobrir o estado de um *wiimote*.

Cada macro leva um parâmetro, um ponteiro para a estrutura do *wiimote*.

- WIIUSE_USING_ACC(wm)
  Retorna 1 se o acelerômetro do *wiimote* está em uso (sensibilidade ao movimento está ativada).
- WIIUSE_USING_EXP(wm)
  Retorna 1 se uma expansão está conectada ao *wiimote*.
- WIIUSE_USING_IR(wm)
  Retorna 1 se a câmera de IR do *wiimote* está em uso.

- WIIUSE_USING_SPEAKER(wm)
  Retorna 1 se o speaker do *wiimote* está ligado.
- WIIUSE_IS_LED_SET(wm, number)
  Retorna 1 se o led correspondente ao número *number* (varia entre 1 e 4) está ligado.

11.17. **Definindo as *Flags* do *Wiiuse***

Algumas *flags* podem ser definidas para modificar o comportamento da *wiiuse*.

Flags são definidas usando a função *wiiuse_set_flags()*, na seguinte forma:

```
int wiiuse_set_flags(struct wiimote_t* wm, int enable, int disable);
```

Os parâmetros *enable* e *disable* podem ser quaisquer dos seguintes:

- WIIUSE_SMOOTHING
  Essa *flag* é definida por padrão.

  Por causa do ruído gerado pelo acelerômetro, não se pode confiar nos valores de ângulo por ele retornados.
  Se essa flag é definida então os ângulos reportados pela *wiiuse* serão suavizados.

- WIIUSE_CONTINUOUS
  Por padrão *wiiuse* vai gerar somente um evento se o status do *wiimote* mudou (ex.: botão pressionado, movimento do joystick, etc).

  Habilitando essa *flag* fará com que a *wiiuse* constantemente gere eventos mesmo que o status do dispositivo não tenha sido alterado desde o último evento.

12. **Fluxo de dados**

Todos os dados de controle são enviados pelo *wiimote* através de Bluetooth. Como o *wiimote* envia informações de sensores que não são utilizados, o computador de mesa possui a tarefa de filtrar os dados enviados pelo *wiimote* que são realmente utilizados pelo braço robótico. O computador de mesa possui também a tarefa de enviar por Bluetooth esses dados ao KIT. O KIT recebe esses dados e os transforma em sinais para os servomotores. A *figura 17* exibe esse processo.

Figura 17: Processo do fluxo de dados.

13. **Lógica de Transferência de Dados**

Para a interpretação correta dos dados trafegados entre o computador de mesa e o KIT foi definida uma seqüência de estados. O primeiro estado tem por característica o KIT aguardar a solicitação de inicio de tráfego de dados por parte do computador de mesa, onde o KIT aguarda receber qualquer valor na sua interface serial para o início do processo. Logo em seguida o KIT responde ao computador de mesa a solicitação de início, escrevendo o valor 0xFF em sua interface serial. A *figura 18* ilustra essa etapa.

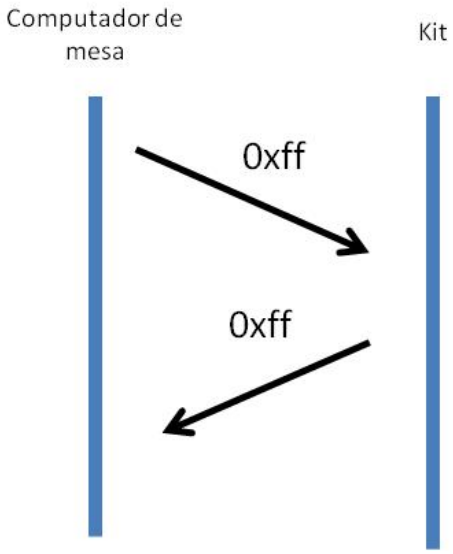

Figura 18: Início da transferência de dados.

Após a etapa de descoberta dos dispositivos inicia-se efetivamente a transmissão dos dados utilizados para a construção dos sinais que são enviados aos servomotores.

O primeiro valor enviado pelo computador de mesa é o referente ao servomotor 05, após o KIT receber o valor do servomotor 05 ele envia ao computador de mesa o valor 0x05. O computador de mesa então envia o valor referente ao servomotor 01, logo em seguida o KIT responde ao computador de mesa o valor 0x01. O próximo valor enviado pelo computador de mesa é o do servomotor 02, o KIT responde enviando o valor 0x02. O valor do servomotor 04 é enviado em seguida pelo computador de mesa, o KIT responde enviando o valor 0x04. O mesmo ocorre para o servomotor 06.

O computador de mesa envia o valor e o KIT responde enviando 0x06. O processo de transmissão dos valores dos servomotores está exemplificado na *figura 19*.

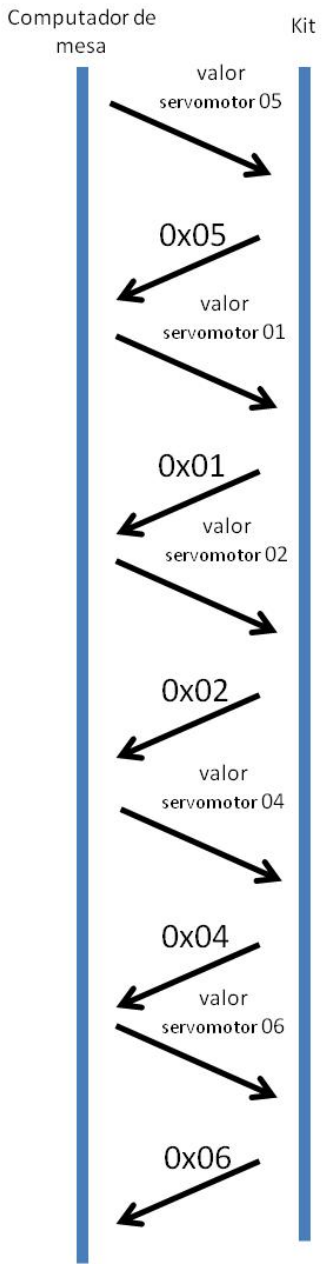

Figura 19: Processo de transferência de dados.

Após o KIT responder ao recebimento dos dados do servomotor 06 o processo retorna ao início, o envio dos dados do servomotor 05.

14. Manipulação do Braço Robótico

A tabela 6 descreve a função de cada sensor do *wiimote* nos movimentos do braço robótico. Para auxiliar, a *figura 20* informa, novamente, a posição de cada servomotor.

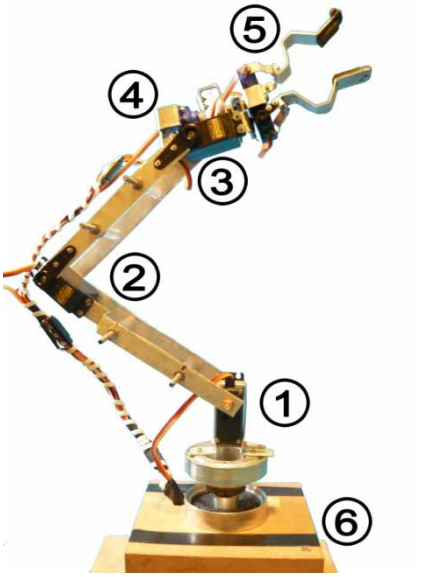

Figura 20: Posição dos servomotores no braço.

Servomotor	**Sensor do controle**
01	*Pitch* do wiimote
02	*Pitch* do nunchuck
03	Não utiliza dados do controle. O valor desse sinal é calculado automaticamente, com base nos servomotores 01 e 02, para se manter sempre paralelo a base do braço robótico.
04	*Left* do *wiimote* e *Right* do wiimote, dependendo do sentido que necessita se rotacionar o eixo do servomotor
05	Botão Z do nunchuck e C do nunchuck, dependendo do sentido que necessita se rotacionar o eixo do servomotor. Ao se pressionar o botão Q a garra mecânica se fecha gradativamente. Ao se pressionar o botão C a garra mecânica se abre gradativamente.
06	*Roll* do *wiimote* seguido do pressionamento do botão B do *wiimote*

Tabela 6: Relação entre os servomotores e sensores.

O braço mecânico ainda utiliza o sinal do botão A do *wiimote*, quando pressionado os dados de manipulação recebidos são ignorados. Esse recurso é utilizado quando se necessita

que o braço robótico mantenha a sua última posição. Os demais sinais enviados pelo *wiimote* não são utilizados nesse projeto. A *figura 21* ilustra os sensores do *wiimote*.

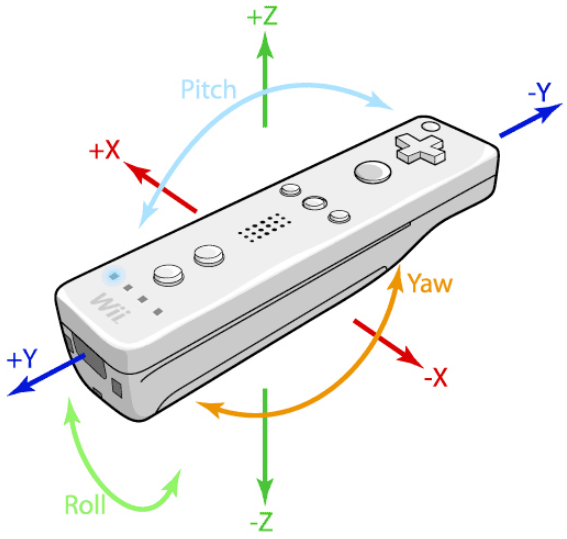

Figura 21: Funcionalidades do *wiimote*.

14.1. Suavização dos Dados Enviados pelo Controle Remoto

Os dados enviados pelo *wiimote*, aparentemente, não sofrem tratamento quanto a instabilidades de valores. Ou se sofrem tratamento, são enviados de forma instável tendo se em vista o objetivo de nosso projeto. O fato é que mesmo se tendo um momento de repouso aparente o *wiimote* não envia os mesmos valores, esses valores, mesmo com diferenças pequenas, precisam ser tratados para que o braço robótico não apresente vibração nos movimentos.

Para implementar esse tratamento do sinal foi utilizado um algoritmo de processamento digital de sinal. Foi aplicado um filtro passa baixa, onde buscou se atenuar as variações rápidas de valores enviados pelo *wiimote* e manter os valores que são considerados estáveis. Foi considerada a freqüência máxima de operação de 50Hz, a freqüência de corte de 10Hz com freqüência de *stop* de 15Hz, e uma atenuação de 60dB.

Com isso atingimos a suavidade desejada nos movimentos do braço robótico, além de proporcionar uma convergência satisfatória quando aplicado movimentos bruscos no controle remoto. Essa convergência satisfatória para movimentos bruscos é muito importante para a conservação da estrutura do braço robótico, uma vez que os movimentos, aparentemente, não exercem forças que possam gerar algum dano na estrutura do braço robótico. A *figura 22* tem

como objetivo exibir a diferença entre os dados enviados pelo *wiimote* e a saída do filtro DSP aplicado.

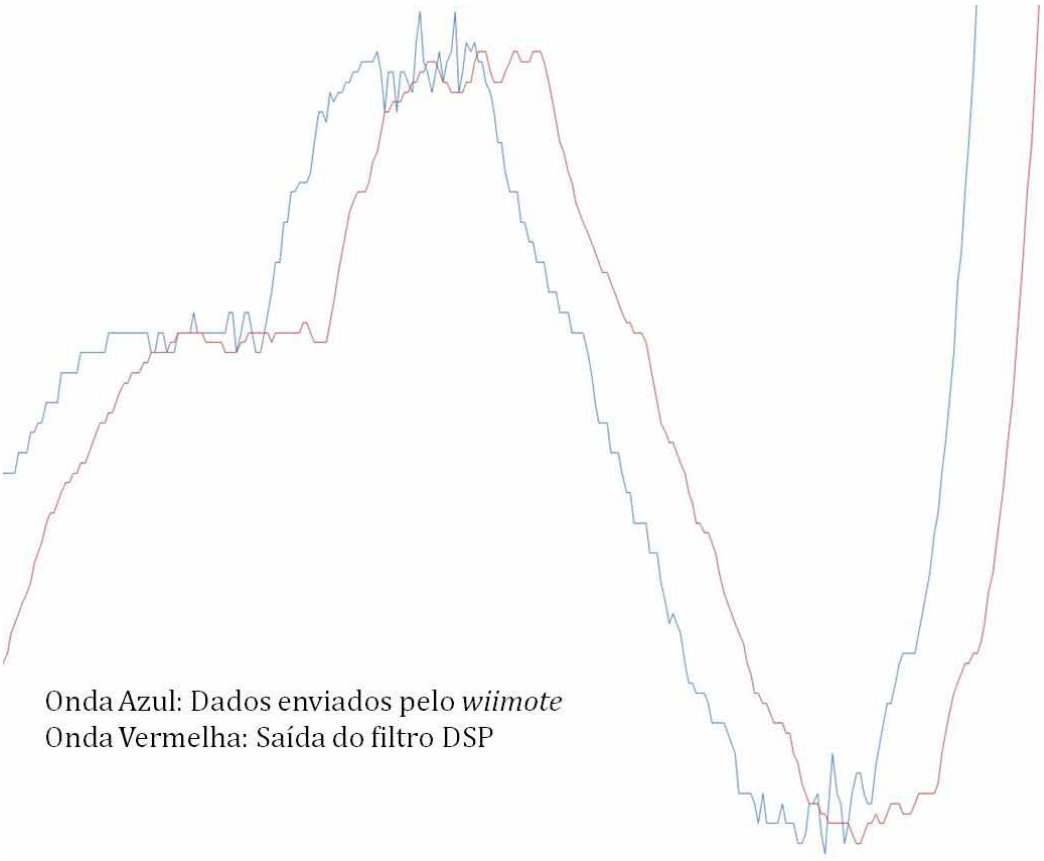

Figura 22: Desempenho do filtro DSP.

15. **Utilização Prática do Braço Robótico**

O braço pode ser utilizado em diversas aplicações, duas aplicações encontradas pelo grupo foi empilhar blocos de plástico e movimentar blocos em um jogo da velha "robótico".

16. **Problemas Encontrados**

Logo após adquirirmos o módulo da SURE encontramos alguns problemas por não saber utilizar o módulo e também por que faltava muita documentação. Então trocamos alguns emails com o pessoal da SURE e até alguns com a Texas Instruments e acabamos descobrindo que fazer o protocolo Bluetooth funcionar era muito mais simples do que o que estávamos tentando descobrir. Bastava sincronizar o módulo com o PC e o mesmo estaria disponível em uma porta COM pronto para ser tratado como uma simples porta serial.

Também tivemos problemas com a taxa da porta serial, que no caso do módulo é de 9600bps enquanto para o cabo é de 115200bps e não percebemos isso logo de início.

17. **Custo Aproximado do Projeto**

Foi possível a construção do pequeno braço robótico com servomotores que custam aproximadamente R$ 16,00 cada um. Foi utilizado 1 metro de um perfil de alumínio de uma polegada, para todo o projeto, perfil de alumínio esse custou aproximadamente R$ 20,00. Investiu-se aproximadamente R$ 10,00 entre parafusos e porcas.

O KIT de Laboratório de Processadores I custou R$ 100,00. O módulo Bluetooth SURE custou R$ 100,00. Custos extras contabilizaram R$ 45,00, entre extensores, caixas de MDF, etc.

Então se alcançou um custo aproximado de R$ 400,00.

18. **Conclusão**

Todos os objetivos propostos na especificação inicial do trabalho foram satisfeitos. Conseguiu-se construir um pequeno braço robótico, manipulável por um dispositivo Bluetooth, totalmente funcional. Dentro do tempo projetado foi possível a construção total do braço robótico didático utilizando materiais de fácil acesso, estudo dos dispositivos de mercado, utilização de tecnologias recentes, busca do conhecimento necessário e aplicação dos conhecimentos adquiridos durante o período acadêmico.

Com sucesso se identificou o controle remoto do videogame Nintendo Wii, *wiimote*, como um dispositivo apropriado para a manipulação do braço robótico. Esse projeto prova a capacidade do *wiimote* para não somente a manipulação de braços robóticos, mas também a possibilidade de empregá-lo em diversas situações. Um dispositivo de custo baixo, que utiliza tecnologias atuais e pode ser o substituto dos *joysticks* e botões, comumente utilizados em dispositivos de mercado, por exemplo, em processos de automação.

O *wiimote* permitiu a manipulação do braço robótico de forma inovadora, inteligente e intuitiva para o usuário. Com a utilização do *wiimote* também foi possível o estudo detalhado do protocolo de comunicação sem fio Bluetooth, e as oportunidades que esse protocolo pode fornecer para nosso projeto. Com sucesso conseguimos comunicar, sem restrições, o *wiimote* com o computador de mesa, contudo a comunicação entre o computador de mesa e o KIT, através de Bluetooth, não foi satisfatória devido a taxa de transmissão lenta do módulo Bluetooth SURE. O módulo Bluetooth SURE utilizado nesse trabalho suporta uma taxa máxima de transferência de 9600bps, uma taxa lenta se comparada com 115200bps, velocidade essa atingida utilizando se o cabo. A 9600bps o braço mecânico não exibe movimentos suaves, movimentos suaves são apresentados quando utilizada uma velocidade igual ou superior a 115200bps. O problema de baixa taxa de transferência do módulo Bluetooth poderia ser corrigido caso seja utilizado um módulo Bluetooth de taxa igual ou superior a 11520bps.

Atingiu se o objetivo, também, de estudar e aplicar na prática conhecimentos de robótica, conhecimento esse que não foi satisfatoriamente transmitido durante o curso de Engenharia de Computação. Com a conclusão com sucesso desse projeto os autores desse trabalho sentem se seguros em participar de estudos referentes a robótica.

19. **Referências Bibliográficas**

[FRA09] Francisco , António M. S. . "*Motores Eléctricos*". ETEP (Lidel), 2009.

[GRA03] Gratton, Dean A. "*Bluetooth Profiles: The Definitive Guide*". Upper Saddle River: Prentice Hall PTR, 2003.

[MIL01] Miller, Michael. "*Descobrindo Bluetooth*". Campus, 2001.

[MOR02] Morrow, Robert. "*Bluetooth operation and use*". McGraw-Hill, 2002.

[LAF08] Laforest, Michael. *"The Wiimote API"*. Lançado em 03/04/08. Disponível em http://www.wiiuse.net/