PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA – CCET
CURSO DE ENGENHARIA DE COMPUTAÇÃO

Alexander Pataki
Henrique Soares Hinke
José Eduardo da Silva Rodrigues
Matheus Augusto de Queiroz Sene

PROJETO INTEGRADO WIIROBOT

Curitiba - PR
2010

Alexander Pataki
Henrique Soares Hinke
José Eduardo da Silva Rodrigues
Matheus Augusto de Queiroz Sene

PROJETO INTEGRADO WIIROBOT

Documento apresentado ao curso Graduação em Engenharia de Computação da Pontifícia Universidade Católica do Paraná como requisito à avaliação da disciplina de Resolução dos Problemas de Engenharia e Física III, referente ao projeto integrado do 1º semestre.

Professores: Afonso Ferreira Miguel e Gil Marcos Jess.

Curitiba - PR
2010

DEDICATÓRIA

Dedicamos este trabalho primeiramente a Deus, pela saúde, fé e perseverança que têm nos dado. A nossos pais, pelo reconhecimento à nossa escolha do curso e futura profissão, e a quem honramos pelo esforço de nos mantermos vivos em um curso com tantas dificuldades como este. A todos os professores e professoras que muito nos ajudam para nossa futura formação, dos quais teremos boas lembranças.

AGRADECIMENTOS

Nossos sinceros agradecimentos aos professores e colaboradores que enriqueceram nosso projeto com críticas e sugestões. Em especial, os professores Afonso Ferreira Miguel e Gil Marcos Jess. Também gostaríamos de agradecer nossos colegas de turma Washington Luiz Peroni Balsevicius, Lucas Caldoncelli Rodrigues e ao senhor Abel Soares Hinke pela contribuição em equipamentos e idéias para melhoria de nosso projeto.

SUMÁRIO

1. INTRODUÇÃO.....06
2. OBJETIVOS.....06
3. DESCRIÇÃO DO PROJETO.....07
    - Controle Nunchuck.....07
    - Servos Motores.....08
    - Suporte Pan and Tilt.....10
    - Câmera.....12
    - Arduino.....13
        - Alimentação.....14
        - Memória.....15
        - Entrada e Saída.....15
        - Comunicação.....17
        - Programação.....17
        - Segurança.....17
        - Características Físicas.....18
    - Softwares.....18
        - Software Controle Nunchuck.....18
        - Software Filtro.....24
        - Software Arduino.....25
        - Software Captura Webcam.....29
4. ÉTICA.....34
5. CONCLUSÃO.....34
6. REFERÊNCIAS BIBLIOGRÁFICAS.....35
7. APÊNDICE.....35

1. INTRODUÇÃO

Para o nosso primeiro projeto na graduação de engenharia da computação decidimos realizar algo que estivesse dentro de nossos conhecimentos, que tivesse uma boa aceitação dos professores e que estivesse dentro dos pré-requisitos de elaboração do projeto.

O projeto em questão propõe o desenvolvimento de um robô para câmeras de segurança. O controle do robô é feito através do acelerômetro que o controle Nunchuck do Nintendo Wii possui. O suporte para a câmera e seus servos motores é denominado de pan and tilt e em nosso projeto foi feito utilizando acrílico. Os comandos são realizados através do microprocessador Arduino Duemilanove conectando-se ao computador através de porta USB embutida nele.

Outro aspecto relevante frente ao projeto é que atualmente existe um grande aumento de sistemas de segurança para diversas aplicações, os quais serão alvos de aplicações do projeto a ser desenvolvido.

2. OBJETIVOS

O principal objetivo deste projeto é o controle do movimento do suporte com a câmera de segurança através de um controle que possui um acelerômetro interno, permitindo o controle da câmera à distância e um maior aproveitamento do espaço de cobertura.

No projeto é utilizado o microprocessador Arduino Duemilanove, devido a algumas facilidades na programação e implementação do mesmo, evitar a inserção de componentes e tratamentos adicionais além do conhecimento prévio dos integrantes do grupo.

A interface homem máquina é feita pelo próprio controlador Arduino, que possui toda a lógica de controle. Ele possui um ambiente de desenvolvimento de suas funções que se comunica com o computador através de porta USB acoplada na placa.

Um dos principais desafios do projeto é com que ele seja adaptado facilmente nos mais variados sistemas de segurança presentes atualmente no mercado.

3. DESCRIÇÃO DO PROJETO

Em nossa descrição de projeto detalhamos tudo o que foi desenvolvido no mesmo. Desde uma pequena explicação sobre o equipamento até a forma que ele foi aplicado em nosso projeto.

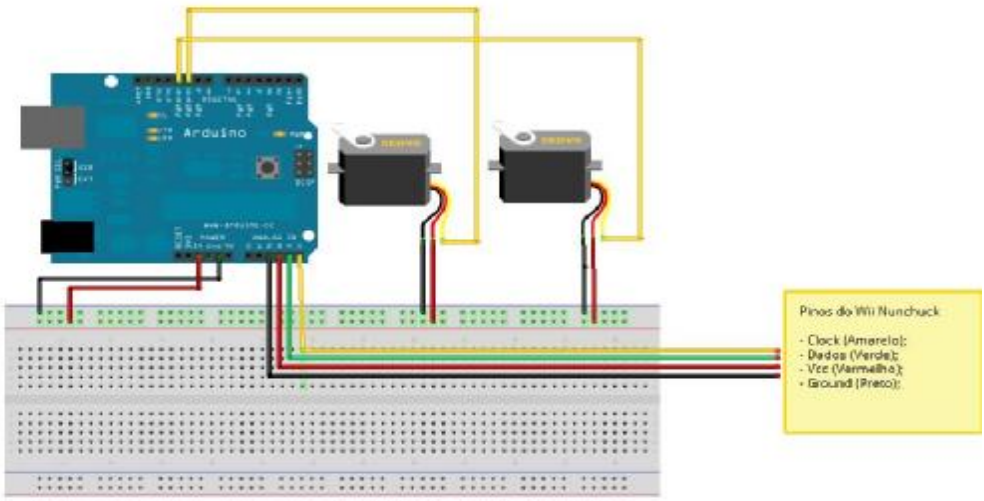

Diagrama Geral Wiirobot

3.1) CONTROLE NUNCHUCK

O controle Nunchuck faz parte do console Nintendo Wii, lançado oficialmente no final de 2006. O Nintendo Wii revolucionou o mercado de consoles de vídeo game, principalmente pela sua interface única com o usuário e a não utilização de cabos em seus controles.

Outra inovação está justamente no controle Nunchuck. Foi o primeiro controle desenvolvido com um acelerômetro interno capaz de detectar movimentos em três dimensões. É o controle com a maior precisão de controle de movimentos, por isso é usado em jogos onde a precisão é importante para o desenvolvimento.

Em nosso projeto o Nunchuck é parte essencial. Com ele utilizamos para o controle total dos movimentos do suporte e em seus botões existem funções especiais, tais como ativar o modo de gravação de vídeo da câmera ou de bater fotos e guardar em uma pasta no computador.

Figura 1 - Controle Nunchuck 1

Figura 2 - Console Nintendo Wii 1

3.2) SERVOS MOTORES

Servos motores são dispositivos de malha fechada, ou seja: Recebem um sinal de controle, verificam a posição atual, atuam no sistema indo para a posição desejada.

A principal diferença do servo motor em relação a outros tipos de motores existentes é o fato de ele se movimentar apenas com 180º, porém a precisão de posicionamento é muito maior em relação aos seus semelhantes.

O servo motor possui três componentes básicos de sua estrutura:

• **Sistema Atuador**: O sistema atuador é constituído por um motor elétrico, embora também existam servos motores de corrente alternada, a maioria utiliza motores de corrente continua, que é utilizado em nosso projeto Wiirobot. Também está presente um conjunto de engrenagens que formam uma caixa de redução com uma relação bastante longa, que ajuda a amplificar o torque. O tamanho, torque, velocidade do motor, precisão de posicionamento e material das engrenagens são os principais diferenciais do servo motor para os demais.

• **Sensor**: O sensor geralmente é um potenciômetro solidário ao eixo do servo. O valor de sua resistência elétrica indica a posição angular em que se encontra o eixo.

• **Circuito de Controle:** O circuito de controle é formado por componentes eletrônicos discretos ou circuitos integrados e geralmente é composto por um oscilador e um controlador PID (Controle Proporcional Integrativo e Derivativo) que recebe um sinal do sensor e o sinal do controle aciona o motor no sentido necessário para posicionar o eixo na posição desejada.

Em nosso projeto foram utilizados dois servos motores. A principal função deles é movimentar o suporte pan and tilt em 180º horizontalmente e verticalmente. Por existir diversos tipos de servo motor, escolhemos um que se adaptasse melhor a nossa estrutura física e que pudesse em todos os sentidos, tais como força, posição, ruído e tamanho. Para isso selecionamos dois servos motores com as seguintes características:

• As engrenagens são feitas em nylon com três pólos de ferrite;

• Top ball bearing;

• Tensão elétrica operacional: 4.8V ~ 6.0V

• Velocidade de operação: 0.10 sec / 60 degree

• Torque: 1.4kg/cm (19.6 oz / in)

• Dimensões: 22.8 x 11.8 x 20.6mm com peso de 9g

Figura 3 - Servo motor

Figura 4 - Servo motor pequeno

3.3) SUPORTE PAN AND TILT

O Suporte pan and tilt é um suporte conhecido e muito usado para filmagens em geral, principalmente quando se precisa filmar um mesmo local por um longo período de tempo.

O pan and tilt pode ser adaptado de diversas formas para melhor atender aquilo que se precisa. No caso do Wiirobot o suporte foi constituído de uma maneira para que os motores pudessem movimentar a câmera em 180º horizontalmente e verticalmente.

Utilizamos uma placa de acrílico para moldar a estrutura como melhor estudamos, de uma maneira que conseguíssemos tirar o máximo possível de aproveitamento dos motores, tivéssemos a maior área de visibilidade da câmera possível e utilizasse uma quantidade mínima de material para a fabricação.

O processo de criação e montagem da estrutura foi realizado nos laboratórios de maquetes da Pontifícia Universidade Católica do Paraná. O acrílico foi escolhido pelo baixo custo, moderada facilidade de modelagem e designer mais chamativo.

No Wiirobot ele foi desenvolvido em três etapas. A primeira foi o recorte e a montagem da estrutura base, que da a sustentação ao resto do suporte. Ela é composta de três pedaços de acrílicos que juntos formam uma base que é fixada no solo, bancada ou qualquer outro ambiente onde o projeto for implantado. A segunda etapa foi a criação das estruturas moveis, que compõe a parte que irá se movimentar na estrutura. Também é feita através de três bases

pequenas de acrílico que juntas formas um losango, feito para se movimentar e servir como suporte da câmera de segurança. A terceira parte é a montagem de toda a estrutura, que foi feita através de parafusos especiais para acrílico, que formaram nosso suporte pan and tilt. Abaixo estão a foto do protótipo montado no software de modelagem Rhino 3D e outros exemplos de suporte pan and tilt utilizados com freqüência nos mais diversos seguimentos.

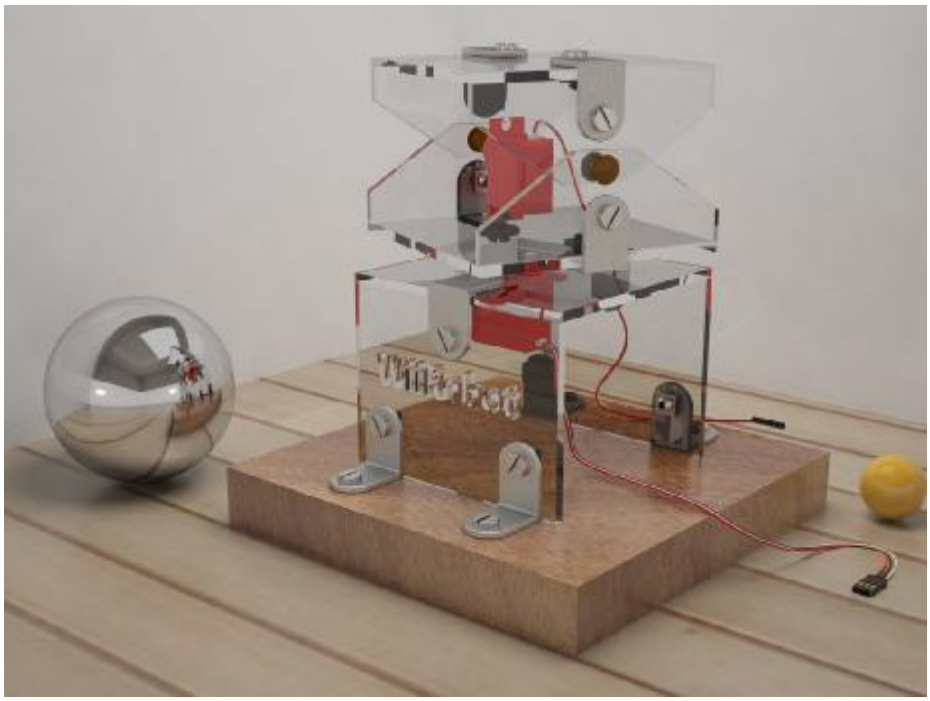

Figura 5 - Protótipo inicial Wiirobot 1

Figura 6 - Pan and Tilt 1

3.4) CÂMERA – WEBCAM

Uma webcam é uma câmera de vídeo de baixo custo que capta imagens e as transfere para um computador no qual esteja conectada.

As principais utilizações da webcam são para videoconferência, monitoramento de ambientes, produções de vídeos e imagens para exibição em outras aplicações. Atualmente existem webcams de baixa ou de alta resolução (acima de 2.0MP) e com ou sem microfones acoplados. Algumas dessas webcams possuem leds para iluminação em ambientes com precária luminosidade. A grande maioria das webcams é conectada ao computador através da porta USB, e a captura de imagens é realizada por um componente eletrônico denominado CCD.

No Wiirobot a webcam é utilizada para monitorar o ambiente escolhido pelo cliente. Ela poderá monitorar o ambiente completo ou apenas locais pré-determinados de acordo com os movimentos que serão feitos pelo controlador através do controle Nunchuck.

A Webcam que escolhemos para o projeto foi selecionada após alguns testes e pré-requisitos que colocamos como fundamentais para a escolha, são eles:

- Tamanho;
- Capacidade de filmagem em ambientes desfavoráveis;
- Preço;
- Tamanho das imagens geradas no computador;
- Qualidade de imagem e filme;
- Driver compatível a maior quantidade possível de sistemas operacionais;

- Facilidade de integração com software de captura desenvolvido pela equipe.

A webcam selecionada nos atende em todos esses pré-requisitos, o modelo dela é LGCam4 de 2.0MP com iluminação noturna.

Figura 7 - Webcam

3.5) ARDUINO

O Arduino é um computador físico baseado numa plataforma de hardware livre, projetada com um microcontrolador de placa única com suporte de entrada/saída embutido e uma linguagem de programação padrão, na qual tem origem em Wiring e é essencialmente em C/C++, tornando mais fácil a adaptação ao nosso projeto e ao entendimento dos integrantes da equipe.

Uma placa Arduino é composta por um controlador, algumas linhas de E/S digital e analógica, além de uma interface serial ou USB.

A interface do hospedeiro é bastante simples, podendo ser escrita em várias linguagens. A mais popular é denominada Processing, utilizada na grande maioria das aplicações desenvolvidas com o Arduino.

O projeto inicial do Arduino iniciou-se na cidade de Ivrea, na Itália em 2005, com o intuito de interagir em pequenos projetos e ter um orçamento de desenvolvimento menor do que outros sistemas semelhantes. Seu sucesso deu-se com uma menção honrosa na categoria de Comunidades Digitais em 2006, pela Prix Ars Eletrônica, além de uma marca considerável de 75mil placas vendidas até o meio do ano de 2009.

Para nosso projeto Wiirobot, selecionamos a placa Arduino Duemilanove, que de acordo com estudos realizado pelos integrantes do grupo, era a placa de melhor custo beneficio e melhor preparada para atender os requisitos do projeto.

Abaixo vão as principais características do microcontrolador utilizado no Wiirobot:

Microcontrolador	ATmega328
Tensão Operacional	5V
Tensão de alimentação (recomendada)	7-12V
Tensão de alimentação (limites)	6-20V
Pinos I/O digitais	14 (dos quais 6 podem ser saídas PWM)
Pinos de entrada analógica	6
Corrente contínua por pino I/O	40 mA
Corrente contínua para o pino 3.3V	50 mA
Memória flash	32 KB (2KB usados para o bootloader)
SRAM	2 KB
EEPROM	1 KB
Velocidade de clock	16 MHz

3.5.1) Alimentação

O Arduino Duemilanove pode ser alimentado pela conexão USB ou por qualquer fonte de alimentação externa. A fonte de alimentação é selecionada automaticamente.

A alimentação externa (SEM USB) pode ser tanto uma fonte, como uma bateria. A fonte pode ser conectada com um plug de 2,1mm (centro positivo) no conector de alimentação. Cabos vindos de uma bateria podem ser inseridos nos pinos GND (terra) e VIN (entrada de tensão) do conector de alimentação.

A placa pode operar com alimentação externa de 6 a 20 Volts. Entretanto, se a alimentação for inferior a 7V, o pino 5V poderá fornecer menos de 5V e a placa poderá ficar instável. Se a alimentação for superior a 12V o regulador de tensão pode superaquecer e avariar a placa, por isso a alimentação recomendada é de 7V a 12V.

Os pinos de alimentação presentes na placa, são:

- **VIN**: Entrada de alimentação para a placa Arduino quando uma fonte externa for utilizada.
- **5V**: A fonte de alimentação utilizada pelo microcontrolador e para outros componentes da placa. Pode ser proveniente do pino VIN através de um regulador On-board, ou ser fornecida pelo USB.
- **3V3**: Alimentação de 3,3V fornecida pelo chip FTDI. A corrente máxima é de 50mA.
- **GND**: É o pino terra.

3.5.2) Memória

O ATmega328 tem 32KB de memória flash para armazenar código, além de 2KB de SRAM e 1KB de EEPROM.

3.5.3) Entrada e Saída

Cada um dos 14 pinos digitais do Arduino Duemilanove podem ser usados como entrada e saída usando as funções **pinMode(), digitalWrite()** e **digitalRead()**. Eles operam com tensão de 5V. Cada pino pode fornecer ou receber um máximo de 40mA e tem um resistor pull-up interno de 20-50kOhms. Além disso, alguns pinos possuem funções especiais:

- **Serial 0(RX) e 1(TX):** Usados para receber(RX) e transmitir(TX) dados seriais TTL. Estes pinos são conectados aos pinos correspondentes do chip serial FTDI USB-TTL.

- **Interruptores externos 2 e 3:** Esses pinos podem ser configurados para disparar uma interrupção por um baixo valor, uma elevação, uma falha ou uma mudança de parâmetros.

- **PWM 3,5,6,9,10 e 11:** Fornecem uma saída analógica PWM de 8bit com a função **analogWrite()**.

- **SPI 10(SS), 11(MOSI), 12(MISO), 13(SCK):** Esses pinos suportam comunicação SPI, embora seja compatível com o hardware ela não esta disponível na linguagem do Arduino.

- **LED 13:** Há um led já montado e conectado ao pino digital 13. Quando o pino esta no valor HIGH, o led acende e quando esta na posição LOW, o led apaga.

O Duemilanove possui seis entradas analógicas e em cada uma delas existe uma resolução de 10bit. Por padrão, elas medem de 0V a 5V, embora essa característica possa ser mudada com códigos específicos na programação do hardware.

- **I²C 4(DAS) e 5(SCL):** Suportam comunicação I²C(TWI) usando a biblioteca Wire)

- **AREF:** Referência de tensão para as entradas analógicas. Usados com a função **analogReference()**.

- **Reset:** Esse botão tem a função de enviar um valor LOW para reinicializar o microcontrolador.

3.5.4) Comunicação

Com o Arduino Duemilanove a comunicação com um PC, com outro Arduino ou com outro modelo de microcontrolador é muito simples. O ATmega328 permite a comunicação serial no padrão UART TTL (5V), que está disponível nos pinos digitais 0(RX) e 1(TX). Um chip FTDI FT232RL presente na placa, encaminha esta comunicação serial através do USB e os drivers FTDI, que já estão no software Arduino, fornecem uma porta COM virtual para o software no computador. O software Arduino inclui um monitor serial que permite que dados simples de texto sejam enviados ao Arduino. Os LEDs RX e TX da placa piscam quando os dados estão sendo transferidos ao computador pelo chip FTDI através da conexão USB.

O ATmega328 também oferece suporte aos padrões de comunicação I²C(TWI) e SPI. O software do Arduino inclui uma biblioteca Wire para simplificar o uso de I²C.

3.5.5) Programação

O Arduino Duemilanove pode ser programado com o software Arduino, que acompanha a placa.

O ATmega328 no Arduino Duemilanove vem pré-gravado com um bootloader que permite enviar novos programas sem o uso de um programador de hardware externo. Ele se comunica utilizado o protocolo original STK500.

3.5.6) Segurança

O Arduino Duemilanove tem um polifusível resetável que protege a porta USB do seu computador contra curto-circuito e sobre corrente. Apesar da maioria dos computadores possuírem proteção interna própria, o fusível proporciona uma proteção extra. Se mais de 500mA foram aplicados na porta USB, o fusível irá automaticamente interromper a conexão até que o curto ou a sobrecarga seja removida.

3.5.7) Características Físicas

O comprimento e a largura máxima do Duemilanove são 2,7" (68,50 mm) e 2,1" (53,34 mm) respectivamente, com o conector USB e o jack de alimentação indo um pouco além destas dimensões

Figura 8 - Arduino Duemilanove 1

3.6) SOFTWARES

No projeto Wiirobot a parte de programação e integração hardware/software foi fundamental para o sucesso do que foi proposto inicialmente. Abaixo, estão os códigos desenvolvidos e aprimorados pela equipe e que foram utilizados para específicas funções dentro de cada estrutura.

3.6.1) Software Controle Nunchuck

O software do controle nunchuck foi desenvolvido juntamente com buscas feitas em sites especializados para conseguirmos adaptar o controle ao Arduino. Abaixo está o código utilizado,na íntegra.

```
#include <WProgram.h>


static uint8_t nunchuck_buf[6];   // array to store nunchuck data
// Uses port C (analog in) pins as power & ground for Nunchuck
static void nunchuck_setpowerpins()
{
#define pwrpin PORTC3
#define gndpin PORTC2
   DDRC |= _BV(pwrpin) | _BV(gndpin);
   PORTC &= ~ _BV(gndpin);
   PORTC |=  _BV(pwrpin);
   delay(100);  // wait for things to stabilize
}
// initialize the I2C system, join the I2C bus,
// and tell the nunchuck we're talking to it
static void nunchuck_init()
{
   Wire.begin();               // join i2c bus as master
   Wire.beginTransmission(0x52);// transmit to device 0x52
   Wire.send(0x40);// sends memory address
   Wire.send(0x00);// sends sent a zero.
   Wire.endTransmission();// stop transmitting
```

```
}
// Send a request for data to the nunchuck
// was "send_zero()"
static void nunchuck_send_request()
{
  Wire.beginTransmission(0x52);// transmit to device 0x52
  Wire.send(0x00);// sends one byte
  Wire.endTransmission();// stop transmitting
}
// Encode data to format that most wiimote drivers except
// only needed if you use one of the regular wiimote drivers
static char nunchuk_decode_byte (char x)
{
  x = (x ^ 0x17) + 0x17;
  return x;
}
// Receive data back from the nunchuck,
// returns 1 on successful read. returns 0 on failure
static int nunchuck_get_data()
{
  int cnt=0;
  Wire.requestFrom (0x52, 6);// request data from nunchuck
  while (Wire.available ()) {
    // receive byte as an integer
    nunchuck_buf[cnt] = nunchuk_decode_byte(Wire.receive());
```

```
      cnt++;
    }
    nunchuck_send_request();  // send request for next data payload
    // If we recieved the 6 bytes, then go print them
    if (cnt >= 5) {
      return 1;   // success
    }
    return 0; //failure
}

// Print the input data we have recieved
// accel data is 10 bits long
// so we read 8 bits, then we have to add
// on the last 2 bits.  That is why I
// multiply them by 2 * 2
static void nunchuck_print_data()
{
    static int i=0;
    int joy_x_axis = nunchuck_buf[0];
    int joy_y_axis = nunchuck_buf[1];
    int accel_x_axis = nunchuck_buf[2]; // * 2 * 2;
    int accel_y_axis = nunchuck_buf[3]; // * 2 * 2;
    int accel_z_axis = nunchuck_buf[4]; // * 2 * 2;

    int z_button = 0;
```

```
int c_button = 0;

// byte nunchuck_buf[5] contains bits for z and c buttons
// it also contains the least significant bits for the accelerometer data
// so we have to check each bit of byte outbuf[5]
if ((nunchuck_buf[5] >> 0) & 1)
   z_button = 1;
if ((nunchuck_buf[5] >> 1) & 1)
   c_button = 1;

if ((nunchuck_buf[5] >> 2) & 1)
   accel_x_axis += 2;
if ((nunchuck_buf[5] >> 3) & 1)
   accel_x_axis += 1;

if ((nunchuck_buf[5] >> 4) & 1)
   accel_y_axis += 2;
if ((nunchuck_buf[5] >> 5) & 1)
   accel_y_axis += 1;

if ((nunchuck_buf[5] >> 6) & 1)
   accel_z_axis += 2;
if ((nunchuck_buf[5] >> 7) & 1)
   accel_z_axis += 1;
```

```
    Serial.print(i,DEC);
    Serial.print("\t");
    Serial.print("joy:");
    Serial.print(joy_x_axis,DEC);
    Serial.print(",");
    Serial.print(joy_y_axis, DEC);
    Serial.print("  \t");
    Serial.print("acc:");
    Serial.print(accel_x_axis, DEC);
    Serial.print(",");
    Serial.print(accel_y_axis, DEC);
    Serial.print(",");
    Serial.print(accel_z_axis, DEC);
    Serial.print("\t");

    Serial.print("but:");
    Serial.print(z_button, DEC);
    Serial.print(",");
    Serial.print(c_button, DEC);

    Serial.print("\r\n");  // newline
    i++;
}
// returns zbutton state: 1=pressed, 0=notpressed
static int nunchuck_zbutton()
```

```
{
    return ((nunchuck_buf[5] >> 0) & 1) ? 0 : 1;  // voodoo
}
// returns zbutton state: 1=pressed, 0=notpressed
static int nunchuck_cbutton()
{
    return ((nunchuck_buf[5] >> 1) & 1) ? 0 : 1;  // voodoo
}
// returns value of x-axis joystick
static int nunchuck_joyx()
{
    return nunchuck_buf[0];
}
// returns value of y-axis joystick
static int nunchuck_joyy()
{
    return nunchuck_buf[1];
}
// returns value of x-axis accelerometer
static int nunchuck_accelx()
{
    return nunchuck_buf[2];   // FIXME: this leaves out 2-bits of the data
}
// returns value of y-axis accelerometer
static int nunchuck_accely()
```

```
{
    return nunchuck_buf[3];   // FIXME: this leaves out 2-bits of the data
}
// returns value of z-axis accelerometer
static int nunchuck_accelz()
{
    return nunchuck_buf[4];   // FIXME: this leaves out 2-bits of the data
```

3.6.2) Software Filtro

O software filtro foi aprimorado de uma classe encontrada na Internet. A principal utilidade é diminuir o ruído.

```
class Kalman
{
private:
  double q;
  double r;
  double x;
  double p;
  double k;
public:
  Kalman(double q, double r)
  {
    this->q = q;
    this->r = r;
    this->x = 0;
```

```
    this->p = 0;
    this->k = 0;
  }
  ~Kalman(void)
  {
  }
  void Update(double measurement)
  {
    this->p = this->p + this->q;
    this->k = this->p / (this->p + this->r);
    this->x = this->x + this->k * (measurement - this->x);
    this->p = (1 - this->k) * this->p;
  }
  double GetPrediction()
  {
    return this->x;
  }
};
```

3.6.3) Software Arduino

Software do Arduino que controla os servos motores e faz a integração com o controle nunchuck. Desenvolvido na plataforma do próprio Arduino na linguagem C++. No código existe uma especificidade que é a gravação dos movimentos feitos previamente. O usuário realiza uma seqüência de movimentos e os consegue gravar, segurando o botão X do controle. Após soltar o controle o Wiirobot fará os mesmos movimentos que foram gravados. Para cancelar, deve-se selecionar o mesmo botão X.

```
#include <Wire.h>
#include <Servo.h>
#include "nunchuck_funcs.h"
#include "kalman_filter.h"
#define TAM 10

int posicoesPan[TAM];
int contPan = 0;
int posicoesTilt[TAM];
int contTilt = 0;

int accx, accy, joyx, joyy, cbut, zbut;

Servo servoPan;
Servo servoTilt;

Kalman filterPan(0.0035, 32);
Kalman filterTilt(0.0035, 32);

void setup()
{
  Serial.begin(9600);

  nunchuck_setpowerpins();
  nunchuck_init();
```

```
  servoPan.attach(10);
  servoTilt.attach(11);

  for(int i = 0; i < TAM; i++)
  {
    posicoesPan[i] = 0;
    posicoesTilt[i] = 0;
  }
}

void loop()
{
  if(nunchuck_get_data()==1)
  {
    //nunchuck_print_data();
    accx = nunchuck_accelx();
    accy = nunchuck_accely();
    joyx = nunchuck_joyx();
    joyy = nunchuck_joyy();
    cbut = nunchuck_cbutton();
    zbut = nunchuck_zbutton();

    filterPan.Update((double)(accx));
    filterTilt.Update((double)(accy));
```

```
int angPan = map((int)(filterPan.GetPrediction()),70,182,0,180);
if(angPan<0){
 angPan = 0;
}
if(angPan>180){
 angPan = 180;
}
int angTilt = map((int)(filterTilt.GetPrediction()),70,182,0,180);
if(angTilt<0){
 angTilt = 0;
}
if(angTilt>180){
 angTilt = 180;
}
if(zbut == 1)
{
 if(cbut == 1)
 {
   if(contPan < TAM)
   {
      posicoesPan[contPan] = 180-angPan;
      contPan++;
      delay(250);
   }
```

```
    else
    {
      contPan = 0;
    }
    if(contTilt < TAM)
    {
      posicoesTilt[contTilt] = 180-angTilt;
      contTilt++;
      delay(250);
    }
    else
    {
      contTilt = 0;
    }
  }
  servoPan.write(180-angPan);
  servoTilt.write(180-angTilt);

  return;
}
if(zbut == 0)
{
  for(int i = 0; i < TAM; i++)
  {
    if(posicoesPan[i] != 0)
```

```
        {
          servoPan.write(posicoesPan[i]);
          delay(500);
        }
        if(posicoesTilt[i] != 0)
        {
          servoTilt.write(posicoesTilt[i]);
          delay(500);
        }
      }
    }
  }
}
```

3.6.4) Software Captura Webcam

```
using System.Windows.Forms;
using System.Drawing;
using AForge.Imaging.Filters;
using AForge.Imaging;
namespace CapturaWiiRobot
{
    public partial class Form1 : Form
    {
        Bitmap bmp1 = null;
```

```
Bitmap bmp2 = null;


// Abordagem 1
Difference differenceFilter = new Difference();
IFilter thresholdFilter = new Threshold(15);
IFilter erosionFilter = new Erosion();
IFilter extrachChannel = new ExtractChannel(RGB.R);
Merge mergeFilter = new Merge();
ReplaceChannel replaceChannel = null;
// Abordagem 2
Difference differenceFilter2 = new Difference();
IFilter thresholdFilter2 = new Threshold(15);
IFilter erosionFilter2 = new Erosion();
BlobCounter blobCounter = new BlobCounter();


public Form1()
{
   InitializeComponent();
}
public void LigarCamera()
{
   if(captureX.Devices.Count > 0)
      captureX.SelectDevice((Antheus.Capture.Device)captureX.Devices[0]);
```

```
    //captureX.ShowDeviceOptions();
}
public void DesligarCamera()
{
    timerCaptura.Stop();
    captureX.SelectDevice(null);
}
private void Form1_FormClosed(object sender, FormClosedEventArgs e)
{
    DesligarCamera();
}
private void timerCaptura_Tick(object sender, System.EventArgs e)
{
    Bitmap image = captureX.GrabBitmap24Bits();
    bmp1 = captureX.GrabBitmap8Bits();
    if (bmp1 != null)
    {
        System.Threading.Thread.Sleep(15);

        bmp2 = captureX.GrabBitmap8Bits();
        if (bmp2 != null)
        {
            // Abordagem 1
```

```
differenceFilter.OverlayImage = bmp1;
Bitmap tmp1 = differenceFilter.Apply(bmp2);
Bitmap tmp2 = thresholdFilter.Apply(tmp1);
Bitmap tmp3 = erosionFilter.Apply(tmp2);

Bitmap redChannel = extrachChannel.Apply(image);

mergeFilter.OverlayImage = tmp3;
Bitmap tmp4 = mergeFilter.Apply(redChannel);
replaceChannel = new ReplaceChannel(RGB.R, tmp3);
replaceChannel.ChannelImage = tmp4;
Bitmap tmp5 = replaceChannel.Apply(image);

pictureBoxImage.Image = tmp5;
// Abordagem 2
/*differenceFilter2.OverlayImage = bmp1;
Bitmap tmp1 = differenceFilter2.Apply(bmp2);
Bitmap tmp2 = thresholdFilter2.Apply(tmp1);
Bitmap tmp3 = erosionFilter2.Apply(tmp2);

blobCounter.ProcessImage(tmp3);
Rectangle[] rects = blobCounter.GetObjectsRectangles();
Graphics g = Graphics.FromImage(image);
```

```
            int count = 0;
            using (Pen pen = new Pen(Color.Red, 1))
            {
               foreach (Rectangle rc in rects)
               {
                  g.DrawRectangle(pen, rc);

                  if ((rc.Width > 15) && (rc.Height > 15))
                  {
                     g.FillRectangle(pen.Brush, new Rectangle(rc.X, rc.Y, 10, 15));
                     g.DrawString(count.ToString(), this.Font, Brushes.White, rc.Location);
                     g.DrawRectangle(pen, rc);
                     count++;
                  }
               }
            }
            g.Dispose();
            pictureBoxImage.Image = image;*/
         }
      }
   }
   private void pictureBoxImage_Paint(object sender, PaintEventArgs e)
   {
```

```
            Graphics g = e.Graphics;

            g.DrawLine(new Pen(Color.FromArgb(75, Color.Red), 2), new
Point(pictureBoxImage.Width / 2, 0), new Point(pictureBoxImage.Width / 2,
pictureBoxImage.Height));

            g.DrawLine(new Pen(Color.FromArgb(75, Color.Red), 2), new Point(0,
pictureBoxImage.Height / 2), new Point(pictureBoxImage.Width, pictureBoxImage.Height
/ 2));

        }

        private void Form1_Load(object sender, System.EventArgs e)

        {

            LigarCamera();

            timerCaptura.Start();

        }

        private void Form1_Resize(object sender, System.EventArgs e)

        {

            pictureBoxImage.Location = new Point(Width / 2 - pictureBoxImage.Width / 2,
Height / 2 - pictureBoxImage.Height / 2);

        }

    }

}
```

4. ÉTICA

O nosso projeto não visa afirmar em nenhum momento que o desenvolvimento partiu exclusivamente dos integrantes de nossa equipe. Nesse caso, alguns dos materiais foram retirados da Internet e publicações específicas do que iríamos utilizar em nosso projeto. A maioria dos integrantes ainda não possui sólidos conhecimentos em determinados assuntos que envolveram o

desenvolvimento, por isso a união e o esforço de todos foi fator determinante para o sucesso do Wiirobot.

5. CONCLUSÃO

O presente trabalho proporcionou uma compreensão ampliada do controle de movimento de uma estrutura mecânica, do uso de microcontroladores, tais como o Arduino, e a integração de hardware e software afim de realizar uma função.

No tangente ao trabalho como um todo fica claro a presença dos conhecimentos de forma integrada como os de fisica, no controle dos motores na estrutura pan and tilt, a disciplina de técnicas avançadas de programação e estrutura de dados, com o C++ e a disciplina de sistemas digitais, que nos ajudaram na composição da protoboard e na integração de todos os sistemas.

Como resultado, podemos afirmar que o projeto obteve sucesso ao integrar todos os conhecimentos obtidos e também na iniciação em desenvolvimento de projetos acadêmicos, desde o pré-projeto até o resultado final.

6. REFERENCIAS

Site acessado em 20/04/2010 – **http://www.arduino.cc/en**

Site acessado em 20/04/2010 – **http://interactive-matter.org/2009/12/filtering-sensor-data-with-a-kalman-filter/comment-page-1/**

Site acessado em 17/04/2010 -**http://pt.wikilingue.com/es/Nunchuk**

7. APÊNDICE

Segue anexo em formato digital.