Pontifícia Universidade Católica do Paraná

PROJETO INTEGRADO – SPY ROBOT

CURITIBA – PARANÁ

2011

André Colin

Valquíria Pires de Oliveira

SPY ROBOT

Projeto Integrado, apresentado as disciplinas de Fisica III, Sistemas Digitais, Resolução de Problemas de Engenharia e Circuitos Elétricos, do curso de Engenharia de Computação, 3º Período da Universidade Católica do Paraná.

CURITIBA – PARANÁ

2011

1.INTRODUÇÃO

O projeto SpyRobot surgiu da idéia de criar um robô espião controlado por um computador via rádio freqüência pois existem tarefas que apresentam riscos para serem executadas por pessoas, e antes de serem executadas, é necessaria uma avaliação do local para minimizar o risco de acidentes.

Neste projeto iremos demonstrar o uso de um robô controlado para reconhecimento de áreas de risco, que integra os conhecimentos em circuitos elétricos, física, sistemas digitais e programação.

2.OBJETIVOS

Este projeto tem como objetivo, disponibilizar uma ferramenta para o auxilio de governos ou empresas em tarefas de possivel risco, evitando qualquer perigo aos seus funcionarios. O SpyRobot entra como um mecânismo que avalia o perigo antes da interferência humana ou em algumas tarefas realiza todo o trabalho envolvido como na supervisão de equipamentos ou tubulações.

3.DESCRIÇÃO DO PROJETO

3.1.PROTÓTIPO

O SpyRobot é um robô que terá um completo sistema de comunicação por video sem fio. Sua propulsão é exercida por duas esteiras movidas a motores eletricos com redução por engrenagens.

O video que é enviado ao computador é capturado através de uma micro-camera presa na parte frontal do robô que movimenta-se horizontal e verticalmente. Também conta com um sistema de iluminação por LEDs brancos de alto brilho distribuidos por todo o robô, proporcionando seu uso na ausência de luz.

3.2.ESTEIRAS

As esteiras são feitas de segmentos retangulares de plástico que unidos envolvem nove rodas cada uma. Quatro dessas rodas possuem suspensão individual para melhor agilidade em terreno acidentado, outras duas rodas são para extensão da esteira e transmissão de tração. As últimas três pequenas em cima são apenas para sustentação e ajudar no alinhamento.

3.3.MOTORES

O robô possui dois motores que controlam as esteiras, um para cada esteira. Cada um dos motores possui um sistema de engrenagem para redução.

3.4.CÂMERA

A câmera possui Motores para movimentação de 270º horizontal e 90º vertical. Pode ser conectada a internet por wireless (tecnologia sem fio) a qualquer roteador vendido no mercado ou através de cabo RJ45 convencional.

Possui microfone e alto falante embutidos na câmera, permitindo assim formar um chat entre o administrador que controla a câmera e o ambiente que está sendo monitorados.

As imagens podem ser vistas ao vivo (tempo real) de qualquer lugar do mundo pela internet. Possui infravermelha - visão noturna (visão total na escuridão a distâncias superiores a 8 metros)

Possui sensor de movimento, com opção de envio automático de e-mail.

Possui modo panorâmico de vigilância com movimentação horizontal automática, os vídeos podem ser gravados remotamente em qualquer computador conectado a internet ou rede, além de poderem ser armazenados em servidor FTP.

Os movimentos da câmera podem ser controlados pela internet (navegador através do programa que acompanha a câmera).

3.5.ALIMENTAÇÃO

A alimentação do robô é gerada por duas baterias, uma de 6V e 1,3A que alimenta a placa controladora de radio e os motores.

A segunda bateria de 6V e 1,3A alimenta somente a câmera sem fio.

3.6.COMUNICAÇÃO

O robô é controlado pelo computador, que se comunica com o sistema RF através da porta serial, utilizando um arduino e transmite as informações para o receptor no carrinho.

4.CIRCUITOS

4.1.1. ARDUINO

Neste projeto usamos apenas a entrada e saída digital (IO) para o acionamento dos transistores que controlam o envio de dados para o robô através do RF.

5.SOFTWARE

O software foi implementado em c++ usando o Microsoft Visual 2010.

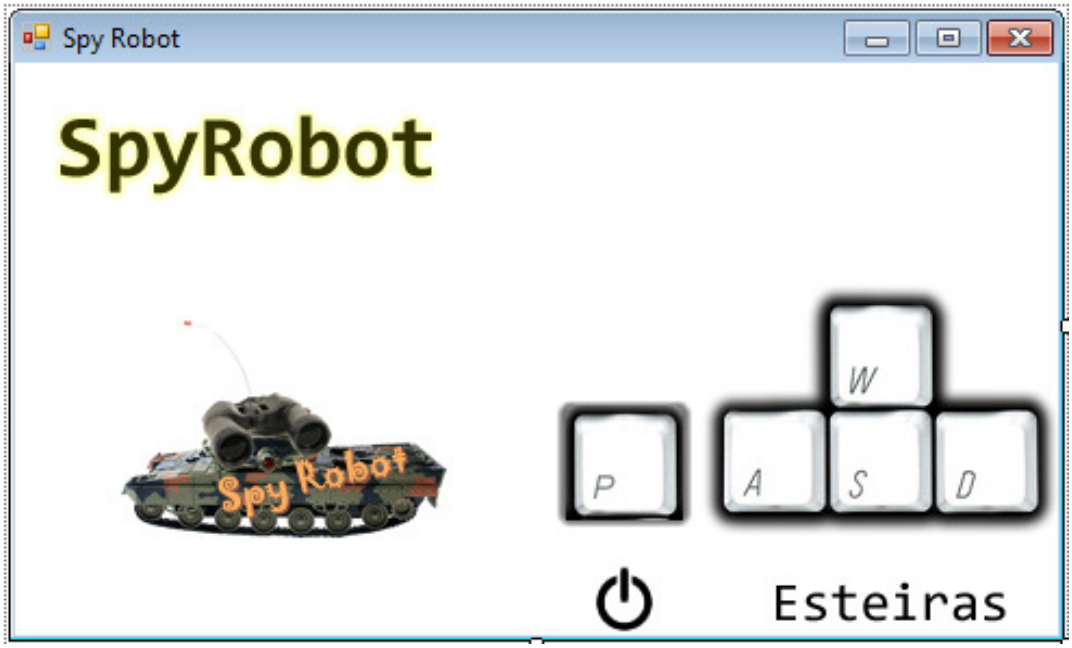

*imagem do programa rodando no Windows 7.

Código fonte:

Programa (keyboard.cpp):

```
// keyboard.cpp : main project file.

#include "stdafx.h"
#include "Form1.h"

using namespace keyboard;

[STAThreadAttribute]
int main(array<System::String ^> ^args)
{
	// Enabling Windows XP visual effects before any controls are created
	Application::EnableVisualStyles();
	Application::SetCompatibleTextRenderingDefault(false);

	// Create the main window and run it
	Application::Run(gcnew Form1());
	return 0;
}
```

Form1.h:

```
#pragma once
```

```
namespace keyboard {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Summary for Form1
	///
	/// WARNING: If you change the name of this class, you wipll need to
change the
	///          'Resource File Name' property for the managed resource
compiler tool
	///          associated with all .resx files this class depends on.
Otherwise,
	///          the designers will not be able to interact properly with
localized
	///          resources associated with this form.
	/// </summary>
	public ref class Form1 : public System::Windows::Forms::Form
	{
	public:

	//           property Keys KeyCode {
  //  Keys get ();
//}
	 int x;
		Form1(void)
		{
			InitializeComponent();

		serialPort1->Open();
			 x = 0;

		//
			//TODO: Add the constructor code here
			//
		}

	protected:
		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		~Form1()
		{

			serialPort1->Close();
			if (components)
			{
				delete components;

			}
		}
	private: System::IO::Ports::SerialPort^  serialPort1;

	protected:
	private: System::ComponentModel::IContainer^  components;
```

```
	private:
		/// <summary>
		/// Required designer variable.
		/// </summary>


#pragma region Windows Form Designer generated code
		/// <summary>
		/// Required method for Designer support - do not modify
		/// the contents of this method with the code editor.
		/// </summary>
		void InitializeComponent(void)
		{
			this->components = (gcnew
System::ComponentModel::Container());
			System::ComponentModel::ComponentResourceManager^  resources
= (gcnew System::ComponentModel::ComponentResourceManager(Form1::typeid));
			this->serialPort1 = (gcnew
System::IO::Ports::SerialPort(this->components));
			this->SuspendLayout();
			//
			// serialPort1
			//
			this->serialPort1->PortName = L"COM7";
			//
			// Form1
			//
			this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
			this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
			this->BackColor = System::Drawing::SystemColors::Window;
			this->BackgroundImage =
(cli::safe_cast<System::Drawing::Image^  >(resources-
>GetObject(L"$this.BackgroundImage")));
			this->BackgroundImageLayout =
System::Windows::Forms::ImageLayout::Center;
			this->ClientSize = System::Drawing::Size(481, 264);
			this->Cursor = System::Windows::Forms::Cursors::Default;
			this->FormBorderStyle =
System::Windows::Forms::FormBorderStyle::FixedSingle;
			this->Name = L"Form1";
			this->Text = L"Spy Robot";
			this->Load += gcnew System::EventHandler(this,
&Form1::Form1_Load);
			this->KeyDown += gcnew
System::Windows::Forms::KeyEventHandler(this, &Form1::Form1_KeyDown);
			this->KeyPress += gcnew
System::Windows::Forms::KeyPressEventHandler(this, &Form1::Form1_KeyPress);
			this->KeyUp += gcnew
System::Windows::Forms::KeyEventHandler(this, &Form1::Form1_KeyUp);
			this->ResumeLayout(false);

		}
#pragma endregion
	private: System::Void Form1_KeyDown(System::Object^  sender,
System::Windows::Forms::KeyEventArgs^  e) {


				if ( e->KeyCode == Keys::W){

					serialPort1->Write("1");
```

```
                        }
                        if ( e->KeyCode == Keys::A){
                                serialPort1->Write("2");

                        }
                        if ( e->KeyCode == Keys::S){
                                serialPort1->Write("8");

                        }
                        if ( e->KeyCode == Keys::D){
                                serialPort1->Write("4");

                        }
                        if ( e->KeyCode == Keys::P){
                                serialPort1->Write("5");

                        }




                    }
       private: System::Void Form1_KeyPress(System::Object^  sender,
System::Windows::Forms::KeyPressEventArgs^  e) {
                    }
       private: System::Void Form1_KeyUp(System::Object^  sender,
System::Windows::Forms::KeyEventArgs^  e) {

                         if ( e->KeyCode == Keys::W){
                                serialPort1->Write("0");

                         }
                            if ( e->KeyCode == Keys::A){
                                serialPort1->Write("0");

                         }
                                 if ( e->KeyCode == Keys::S){
                                serialPort1->Write("0");
```

```
                        }
                                if ( e->KeyCode == Keys::D){

                        serialPort1->Write("0");


                        }

                                if ( e->KeyCode == Keys::P){

                        serialPort1->Write("0");


                        }


                    }
      private: System::Void Form1_Load(System::Object^  sender,
System::EventArgs^  e) {
                    }
private: System::Void pictureBox1_Click(System::Object^  sender,
System::EventArgs^  e) {
             }
};
}
```

Stdafx.h:

```
// stdafx.h : include file for standard system include files,
// or project specific include files that are used frequently, but
// are changed infrequently
#pragma once

// TODO: reference additional headers your program requires here
```

Stdafx.cpp

```
// stdafx.cpp : source file that includes just the standard includes
// keyboard.pch will be the pre-compiled header
// stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"
```

Assemblyinfo.cpp:

```
#include "stdafx.h"

using namespace System;
using namespace System::Reflection;
using namespace System::Runtime::CompilerServices;
using namespace System::Runtime::InteropServices;
```

```
using namespace System::Security::Permissions;

//
// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
//
[assembly:AssemblyTitleAttribute("keyboard")];
[assembly:AssemblyDescriptionAttribute("")];
[assembly:AssemblyConfigurationAttribute("")];
[assembly:AssemblyCompanyAttribute("Microsoft")];
[assembly:AssemblyProductAttribute("keyboard")];
[assembly:AssemblyCopyrightAttribute("Copyright (c) Microsoft 2008")];
[assembly:AssemblyTrademarkAttribute("")];
[assembly:AssemblyCultureAttribute("")];

//
// Version information for an assembly consists of the following four values:
//
//      Major Version
//      Minor Version
//      Build Number
//      Revision
//
// You can specify all the value or you can default the Revision and Build
Numbers
// by using the '*' as shown below:

[assembly:AssemblyVersionAttribute("1.0.*")];

[assembly:ComVisible(false)];

[assembly:CLSCompliantAttribute(true)];

[assembly:SecurityPermission(SecurityAction::RequestMinimum, UnmanagedCode =
true)];
```

Código Fonte “Arduino”

```
int io8 = 8, io9 = 9, io10 = 10, io11 = 11, io12 = 12;
char incomingByte = -1;

void allLow (){
    digitalWrite(io8,LOW);
    digitalWrite(io9,LOW);
    digitalWrite(io10,LOW);
    digitalWrite(io11,LOW);
    digitalWrite(io12,LOW);}

void setup(){

    Serial.begin(9600);

    pinMode(io8,OUTPUT);
    pinMode(io9,OUTPUT);
    pinMode(io10,OUTPUT);
```

```
    pinMode(io11,OUTPUT);
    pinMode(io12,OUTPUT);
    allLow();}


void loop(){
// allLow();
  if(Serial.available()){incomingByte = Serial.read();}

  switch (incomingByte){
          case 48:
          allLow();
          break;

         case 49:
         digitalWrite(io11,HIGH);
         digitalWrite(io10,HIGH);

         break;

         case 50:
         digitalWrite(io12,HIGH);

         break;

         case 56:
         digitalWrite(io12,HIGH);
         digitalWrite(io8,HIGH);

         break;

         case 52:
         digitalWrite(io8,HIGH);

         break;

         case 53:
         digitalWrite(io9,HIGH);
         break;
      }


incomingByte = -1;
}
```

6.CONCLUSÃO

Com o desenvolvimento do projeto teve-se muita dificuldade, pois todo o material principal para o desenvolvimento foi comprado diretamente da China, e devido ao tempo não foi possível trabalhar o tempo necessário, devido ao prazo estabelecido.

Mas mesmo tendo esses imprevistos, o projeto foi entregue dentro do prazo previsto.

6. SITE E VÍDEOS

Site do Projeto:

http://www.spyrobot.hd1.com.br

Videos:

http://www.youtube.com/watch?v=l7NZIzqJWFI

7.REFERÊNCIAS

Sites:

http://www.arduino.cc

Professores Colaboradores:

- Gil Marcos Jess
- Afonso Ferreira Miguel
- Ivan Chueiri