CENTRO UNIVERSITÁRIO DE BRASÍLIA -UniCEUB

CURSO DE ENGENHARIA DE COMPUTAÇÃO

GABRIEL SANTOS DE FREITAS

DISPOSITIVO ELETRÔNICO COM SENSOR DE DISTÂNCIA PARA AUXÍLIO NA APRESENTAÇÃO DE DATASHOW DE DOCUMENTOS EM AMBIENTE VIRTUAL

Orientadora: MSc. Maria Marony Sousa Farias Nascimento

Brasília

Junho, 2011

GABRIEL SANTOS DE FREITAS

DISPOSITIVO ELETRÔNICO COM SENSOR DE DISTÂNCIA PARA AUXÍLIO NA APRESENTAÇÃO DE DATASHOW DE DOCUMENTOS EM AMBIENTE VIRTUAL

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.
Orientadora: MSc. Maria Marony Sousa Farias Nascimento.

Brasília

Junho , 2011

GABRIEL SANTOS DE FREITAS

DISPOSITIVO ELETRÔNICO COM SENSOR DE DISTÂNCIA PARA AUXÍLIO NA APRESENTAÇÃO DE DATASHOW DE DOCUMENTOS EM AMBIENTE VIRTUAL

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.
Orientadora: MSc. Maria Marony Sousa Farias Nascimento.

Este Trabalho foi julgado adequado para a obtenção do Título de Engenheiro de Computação, e aprovado em sua forma final pela Faculdade de Tecnologia e Ciências Sociais Aplicadas - FATECS.

_____

Prof. Abiezer Amarilia Fernandez
Coordenador do Curso

Banca Examinadora:

_____

Prof. Maria Marony, Mestre em Engenharia Elétrica
Orientadora

_____

Prof. Luis Cláudio Lopes de Araújo, Mestre em Matemática Pura
Instituição

_____

Prof. Thiago de Miranda Leão Toríbio, Mestre em Física Teórica
Instituição

AGRADECIMENTOS

A Deus, pela Sua eterna e constante presença em nossas vidas, nos protegendo e abençoando.

A minha família, meu pai Octávio, minha mãe Mariângela, minha irmã Carolina, meus familiares e minha namorada Nathália, por todo apoio, carinho, cuidado, paciência, compreensão e atenção que eu poderia precisar e desejar.

Aos meus amigos, que sempre estiveram comigo desde o ensino fundamental e médio, pela amizade, companheirismo e atenção.

Aos meus novos amigos de faculdade que entraram junto comigo, em especial ao Alcides Rafael, Bruno Passos, Guilherme Silva, Gustavo Suzukawa, Jean Matheus, Rafael Alfarone e Samyr Alves e às minhas amigas Camilla Cristine, Vanessa Miranda. Também aos meus amigos de faculdade Leonardo Lima, Vinicius Tonhá, Thiago Alencar, Marco Aurélio, Reinaldo Oliveira por toda ajuda e amizade nesses cinco anos de faculdade. Ao aluno José Carlos que sempre apoiou e ajudou no laboratório.

Ao meu amigo Guilherme da Costa Silva pela criação de alguns desenhos nas ferramentas VISIO e AutoCAD.

À minha orientadora, professora Maria Marony Sousa Farias, pela orientação, atenção e conhecimentos indispensáveis na realização e conclusão deste trabalho.

Aos meus professores pela orientação, ensinamentos e ajuda e principalmente pelo conhecimento indispensável para a minha formação. Agradecer também aos professores Luis Cláudio, Thiago e Javier pelas orientações na banca e no projeto final.

“A mente que se abre a uma nova idéia jamais voltará ao seu tamanho original.”

Albert Einstein

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Neste projeto, é apresentada uma proposta de criação de um dispositivo eletrônico para auxiliar a exposição/apresentação de *slides* por meio do projetor multimídia. Este dispositivo tem o objetivo de dinamizar a exibição evitando que o apresentador interrompa a explicação para mudar o *slide*. Para isso, o protótipo construído, utiliza o movimento da mão do apresentador na frente do sensor para avançar ou voltar os *slides*. Este protótipo conta com um sensor infravermelho de distância, o *software* gratuito *Gobetwino* e o microcontrolador *Arduino* com *ATmega328p*. Para seu funcionamento, o sensor de distância utiliza um feixe de infravermelho para medir a distância. O *software* gratuito *Gobetwino* é responsável pela comunicação entre *Arduino* e o computador. Por fim, o *slide* apresentado no computador é avançado ou recuado, dependendo da direção que a mão segue em frente ao sensor.

Palavras-Chave: Sensor infravermelho de distância, apresentação de *slides*.

ABSTRACT

In this Project is proposing the creation of an electronic device that is capable of assisting people in a datashow presentation. This device has the objective to make the presentation easier to explain because the presenter won´t need to stop the presentation to change the slides. For this, the created prototype, use the movement of the presenter´s hand in front of a infrared sensor to forward or rewind the current slide. This prototype is composed by a infrared sensor of distance, a software Gobetwino and a microcontroller Arduino with ATmega328p for its perfect operation. The infrared sensor uses a infrared laser to measure the distance. The Gobetwino is responsible for the communication between Arduino and the computer. At the end, the slide shown in computer will be forward ou rewind depending of the direction that the presenter´s hand do in front of the sensor.

CAPÍTULO 1 – INTRODUÇÃO

1.1 – Motivação e Posicionamento

A realização deste projeto possui duas motivações básicas: a primeira se deve à observação de professores durante uma apresentação de *slides* na aula, na qual, esses fizeram movimentos na tela de projeção, simulando um avançar de *slides*. Essa observação foi motivadora pelo fato de não ser possível realizar essa função, de passar *slides*, sem um controle remoto ou um dispositivo que captasse o movimento da mão. A segunda motivação se deve ao fascínio do autor por novas tecnologias, nas quais o usuário não tem em posse algum dispositivo físico para executar uma tarefa. (Ex. Projeto Natal da *Microsoft Kinect*). Esta forma de apresentar *slides* elimina a necessidade de o apresentador ter que a todo momento ir até o computador, que é o repositório da apresentação, para avançar ou voltar o *slide* o que facilita e dinamiza a apresentação em um projeto multimídia (*Datashow)*.

Com o intuito de amenizar este problema de deslocamento até o computador para avançar ou retroceder *slides*, este projeto propõe a construção de um dispositivo eletrônico que faça a leitura do movimento da mão ou de um objeto qualquer em frente ao sensor e informe ao computador se o mesmo deve avançar ou retroceder *slides*. Neste dispositivo, a leitura do movimento é feita por um sensor infravermelho de distância responsável por lançar um feixe infravermelho. Quando esse feixe é interrompido pela mão do apresentador, ele se reflete no sensor, que mede as várias distâncias do movimento e o programa compilado no *Arduino com ATmega328p* interpreta a leitura do sensor e envia ao computador, por meio do *software* gratuito *Gobetwino*, o comando necessário para avançar/retroceder *slides*.

Este projeto se restringe a mostrar essa solução em forma de protótipo, pois o custo do sensor e do microcontrolador ultrapassa os valores de um controle-remoto que faz a função de avançar ou voltar *slides*.

1.2 – Visão Geral do Projeto

O projeto simula um dispositivo que auxilia na apresentação de *slides* para exibição em *datashow*. Na Figura 1.1 é apresentado o dispositivo eletrônico do projeto e o seu posicionamento durante a apresentação.

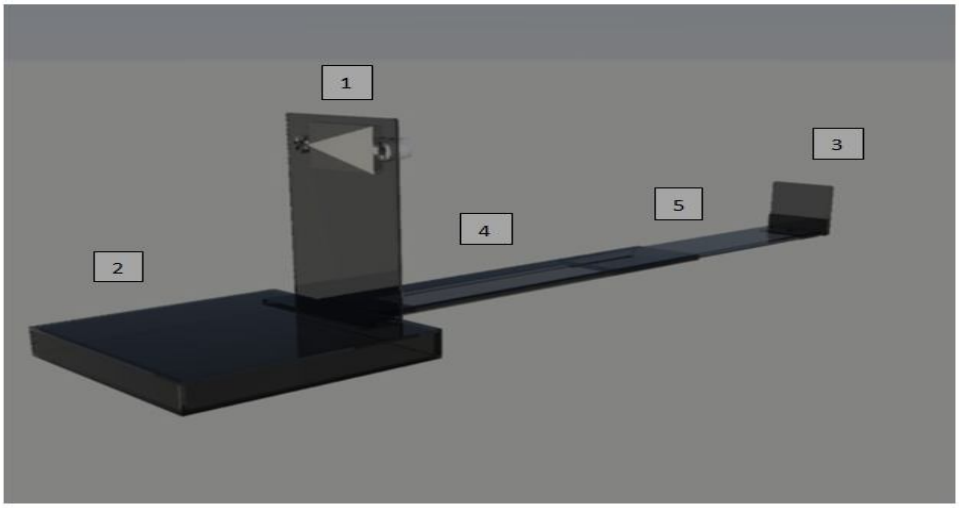

Figura 1.1 – Esboço do suporte que serve para alocar os componentes e conectá-los. (Fonte: *AutoCad*, 2011)

1- A parte número um (1), é destinada ao suporte (5 x 9cm) do sensor infravermelho de distância SHARP GP2D120.

2- A parte número dois (2), é destinada ao suporte (8 x 11cm) do microcontrolador *Arduino ATmega328p Duemilanove*, na qual este está conectado ao sensor de distância e ao computador.

3- A parte número três (3), é destinada ao suporte da barreira de segurança para a calibração do sistema. Esse suporte é composto por duas placas fixas (2 x 2cm e 2 x 5cm), separadas por 2 mm, permitindo alocar uma folha de papel,.

4- A parte número quatro (4), é destinada ao corredor (15 x 3cm) que liga o suporte do microcontrolador e do sensor à barreira de segurança.

5- A parte número cinco (5), é destinada a um corredor secundário (13 x 2cm) fixado diretamente ao suporte da barreira de segurança que permite alongar o corredor.

1.3 – Objetivos do Tabalho

O objetivo geral deste trabalho é apresentar um protótipo de um dispositivo eletrônico, utilizando um sensor infravermelho de distância, que é responsável por transmitir a informação de avançar/retroceder *slides* ao computador. Este dispositivo utiliza um movimento da mão em frente ao sensor para que este consiga captar se o movimento se aproxima ou se afasta dele. Este procedimento provê as informações necessárias para avançar ou voltar o *slide*. Para que este dispositivo consiga realizar a leitura do movimento e transmita a informação correta para o computador, é necessário que algumas tarefas sejam executadas:

- desenvolver um código, em uma linguagem própria do microcontrolador *Arduino com ATmega328p*, linguagem *Wiring*, para que este receba as informações passadas pelo sensor de distância e realize um tratamento dessas informações e passe para o computador o comando correto;

- implementar no *Arduino* um sensor infravermelho de distância, ou seja, conectar o sensor nas portas analógicas do microcontrolador a fim de prover energia e comunicação entre ambos, para a captura do movimento da mão pelo sensor;

- calibrar a distância limite de leitura do sensor infravermelho para que apenas os movimentos capturados, entre o sensor e a barreira de segurança, sejam tratados e utilizados no código compilado *Arduino*;

- configurar no software *Gobetwino* a apresentação que será exibida para que esta receba as informações passadas pelo microcontrolador ao *Gobetwino*;

- executar o software *Gobetwino* logo após o *update* do código no microcontrolador e pressionar o botão de *reset* presente no microcontrolador, para que o *Gobetwino* faça a leitura da mesma porta serial utilizada pelo *Arduino*, para que haja conexão do microcontrolador com o computador e a passagem de comandos.

1.4 – Estrutura da Monografia

Além deste capítulo composto de introdução e objetivos, esta monografia está estruturada em mais quatro capítulos:

Capítulo 2 – Apresentação do Problema – Neste capítulo é apresentado o contexto do problema, como ele é tratado atualmente, as soluções existentes, e como a proposta aqui apresentada pretende solucionar o problema.

Capítulo 3 – Referencial Teórico e Tecnológico – Neste capítulo é apresentado o referencial teórico e tecnológico que compõe a teoria que embasa o projeto. Inicialmente, são descritos os sensores de modo geral. Em seguida, apresenta-se uma visão geral sobre o microcontrolador *Arduino com ATmega328p* e sobre o *software* gratuito *Gobetwino*.

Capítulo 4 – Modelo Proposto – Este capítulo trata do desenvolvimento e a visão do projeto, bem como especifica as questões de *hardware e software* em uma explicação detalhada bem como a forma de funcionamento.

Capítulo 5 – Análise do Modelo Proposto – Este capítulo trata especificamente da aplicação, da descrição e dos resultados do protótipo do projeto.

Capítulo 6 – Conclusão – Este capítulo trata especificamente do final do projeto com suas conclusões e apresentação de propostas para projetos futuros.

CAPÍTULO 2 – CENÁRIO ATUAL

2.1 – Contexto Geral do Problema

Constantemente, empresas, faculdades e escolas utilizam apresentações em *slides* para transmitir as informações desejadas para seus ouvintes. Com a constante necessidade de técnicas e tecnologias para dinamizar e sofisticar as apresentações de *slides,* visando um aumento na interação com o público alvo, surgem inúmeras tecnologias.

Nos ambientes corporativos, as empresas utilizam apresentações em *slides* para vender seus produtos e idéias. Como os executivos precisam convencer seus clientes a comprarem seus produtos, as apresentações em *slides* de forma dinâmica provêem bons resultados, tendo em vista que mantém os clientes atentos. Já no ambiente acadêmico, as apresentações em *slides* permitem que o professor utilize informações sucintas para explicar aos alunos a matéria ministrada.

Um dos problemas verificados no processo de apresentação de *slides* é que em muitos casos, o computador que é utilizado para executar os *slides* não está próximo do local onde a projeção dos mesmos se encontra. Outro problema é o fato de o apresentador ter de interromper a apresentação a todo instante que for mudar de *slides*. Este processo de mudança de *slide* prejudica muito o andamento da apresentação, ocasionando quebra do raciocínio dos ouvintes e pouco dinamismo da apresentação.

Buscando minimizar esse tipo de problema, este projeto apresenta um protótipo que tem como base um microcontrolador *Arduino com ATmega328p,* acoplado a ele um sensor infravermelho de distância, e a conexão microcontrolador e computador é via *USB*. O protótipo construído deve ficar próximo ao apresentador, que para acioná-lo deve passar a mão no espaço delimitado pelo protótipo na direção correta para avançar ou voltar *slides,* que o sistema criado se encarrega de mudar o *slide*.

2.2 – Tecnologias Existentes

Os chamados passadores de *slides*, ou *sliders*, são aparelhos utilizados para passar os *slides* sem a necessidade de o apresentador ir ao computador e apertar um botão do teclado para que a apresentação mude de *slide*. Hoje no mercado, existem produtos prontos que fazem a função de passadores de *slides*. Este projeto busca utilizar um sensor de distância infravermelho no protótipo para realizar essa função. Para exemplificar produtos existentes, são mencionados dois produtos de diferentes fabricantes que atendem ao problema apresentado.

A Logitech é uma empresa que foi fundada em 1981 em Apples, Suiça. Ela é uma empresa focada no desenvolvimento de periféricos. Um dos produtos da *Logitech* é o *Logitech Professional Presenter R800,* figura 2.1. Este produto tem a função de um *slider* e é do tipo *plug and play* e sem fio, na qual um dispositivo *USB* é acoplado a uma das entradas *USB do* computador para fazer a conexão do controle remoto, que fica com o apresentador. Por meio desse controle é possível avançar ou voltar os *slides*.

Figura 2.1 – Aparelho *Logitech Professional Presenter R800* (Fonte: <http://www.logitech.com/en-us/mice-pointers/presentation-remote/devices/5873>)

A empresa *DekCell* é uma empresa que trabalha com componentes eletrônicos, dentre eles, computadores, *laptops*, câmeras e dispositivos de computadores. Um dos produtos que satisfaz o problema apresentado neste projeto é o *RF Wireless Laser Pointer Presentation Device*, figura 2.2. Esse dispositivo trabalha com a conexão sem fio, *plug-and-play*, na qual um dispositivo *USB* é acoplado a uma das entradas *USB* para fazer a conexão do controle remoto, que fica com o apresentador, com o computador.

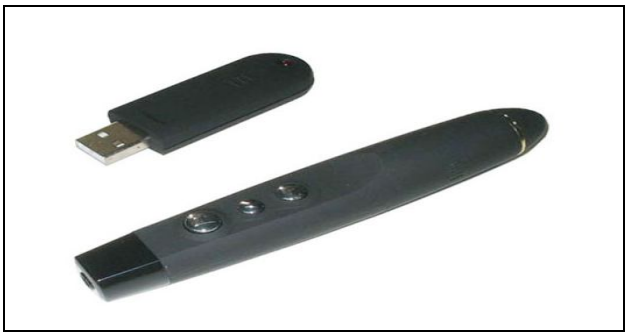

Figura 2.2 – Aparelho *RF Wireless Laser Pointer Presentation Device* (Fonte: <http://www.dekcell.com/product-image.php?pid=3518&img=cpa-1384-black.jpg>)

Apesar de no mercado já existirem produtos que possuem a função de um *slider*, este projeto tem por objetivo criar um protótipo que serve para o mesmo propósito. Um outro aspecto negativo é com relação ao preço, já que para a construção deste protótipo são necessárias peças caras e o custo de produção deste projeto é bem mais alto do que dos outros dispositivos. Outra questão é com relação à estrutura de conexão do dispositivo, isto que esse protótipo é composto de vários componentes que são conectados de forma cabeada, além da plataforma que é criada para apoiar o dispositivo.

No entanto, este projeto permite que o protótipo também seja utilizado por pessoas que possuam alguma deficiência física, que impossibilite o manuseio desses aparelhos existentes no mercado. Por se tratar de um sensor infravermelho de distância acoplado ao microcontrolador, bastaria que um objeto, de preferência com um alto índice de reflexão e de baixo índice de refração, se colocasse em frente ao sensor e fizesse um movimento, pré-determinado, para que o dispositivo entendesse se ele deve avançar ou voltar *slides*.

CAPÍTULO 3 – REFERENCIAL TEÓRICO E TECNOLÓGICO

Para que o protótipo proposto pelo projeto alcance o objetivo de um dispositivo eletrônico que avance ou volte *slides* pelo movimento da mão, alguns métodos, *softwares* e ferramentas são necessários. Para a composição do dispositivo, é necessário um sensor de distância, especificamente um sensor infravermelho de distância, uma linguagem de programação para microcontrolador *Wiring*, um microcontrolador, conexão *USB* e o *software* gratuito *Gobetwino*. A razão pela qual esses dispositivos foram escolhidos está diretamente relacionada ao custo e à conexão entre todos os dispositivos necessários para perfeito funcionamento do projeto.

3.1 – Sensor

Sensor é um termo utilizado para designar dispositivos sensíveis a alguma forma de energia do ambiente que pode ser luminosa, térmica, cinética, relacionando informações sobre grandeza que precisa ser medida, como temperatura, pressão, velocidade, corrente, aceleração, posição, etc. (THOMAZINI).

O sinal de saída de um sensor deve ser manipulado antes de sua leitura no sistema de controle. Isto geralmente é realizado com um circuito de interface para produção de um sinal que possa ser lido pelo controlador (THOMAZINI).

3.1.1 – Sensores Ópticos

Sensores ópticos são componentes eletrônicos de sinalização e comando que executam detecção de qualquer material sem que haja contato mecânico entre eles. O princípio de funcionamento do sensor óptico baseia-se na existência de um emissor e de um receptor. A luz gerada pelo emissor deve atingir o receptor com a intensidade suficiente para fazer com que o sensor comute sua saída (THOMAZINI).

O sinal de luz gerado pelo emissor do senso óptico é modulado numa frequência, ou seja, o emissor gera um sinal com um certo número de lampejos por segundo. O receptor do sinal do sensor é acoplado a um filtro que somente considera sinais com a mesma frequência do emissor, essa característica é empregada no sensor óptico para minimizar os efeitos de possíveis interferências causadas por outras fontes luminosas que não o emissor (THOMAZINI).

3.1.2 – Sensor Óptico por Retrorreflexão

Nesse tipo de sensor, o emissor e o receptor estão montados no mesmo corpo. Um feixe de luz é estabelecido entre o emissor e o receptor por intermédio do refletor. O sensor é ativado quando o objeto interrompe o feixe de luz. O objeto detectado pode deixar passar uma baixa intensidade luminosa desde que o limiar de detecção seja atingido. Ele também pode refletir a luz de maneira direta ou difusa, desde que não seja detectada pelo receptor do sensor com intensidade suficiente para acioná-lo. Por esta razão, objetos muito transparentes, claros ou brilhantes podem eventualmente não ser detectados por esse tipo de sensor. Caso ocorra uma falha no emissor, o sensor talvez faça uma interpretação de que o objeto está presente (THOMAZINI). A figura 3.1 mostra um exemplo do funcionamento de um sensor óptico por retrorreflexão, na qual é o mesmo princípio utilizada neste projeto.

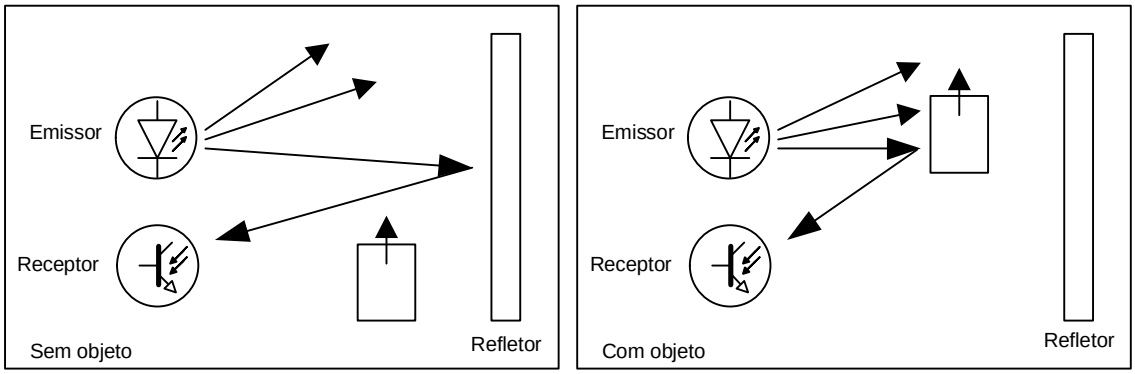

Figura 3.1 – Figura sobre Sensor Óptico por Retrorreflexão (Fonte: Adaptação THOMAZINI, Visio 2011)

3.1.3 – Sensor Infravermelho de Distância SHARP GP2D120

O sensor *SHARP GP2D120* é o sensor escolhido para compor o dispositivo eletrônico mencionado nesta monografia. O *GP2D120* mede a distância lançando um feixe de luz infravermelha em um alvo e captando a reflexão com um fotodiodo infravermelho, presente no receptor, que converte a luz que entra em uma variação de tensão. O detector então reporta a posição do ponto para o processador na qual determina a distância ou a altura. O tipo de detector utilizado no projeto é o *Position Sensitive Detector (PSD)* (FRAUENFELDER).

A figura 3.2 mostra o método de triangulação que ocorre quando o emissor do sensor de distância envia um feixe de luz, esse feixe é refletido e é recebido pelo receptor.

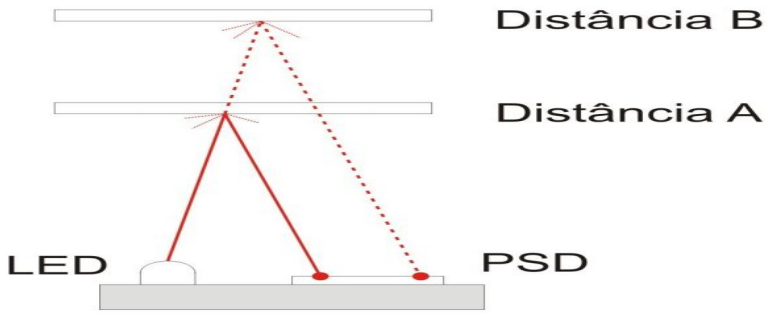

Figura 3.2 - Sensor de Distância com feixe refletido (Fonte: http://www.maxwellbohr.com.br/downloads/robotica/mec1000_kdr5000/tutorial_eletronica_-_aplicacoes_e_funcionamento_de_sensores.pdf)

PSD é um tipo especial de fotodetector analógico. Os sensores de triangulação tipo PSD usam correntes elétricas como informações de saída em cada terminal. Na figura 3.3 mostra o PSD composto pelos terminais *X1* e *X2*, por onde saem as correntes e por uma área ativa *(Active Detector Area*) que receberá o posição do ponto luminoso e passará para um processador. A corrente total de cada terminal é proporcional a posição do ponto no detector. Se o ponto estiver no meio do detector o valor da corrente nos terminais serão iguais. Se o ponto sair do centro, as saídas dos terminais também mudarão e a posição do ponto poderá ser calculada por essa mudança. O valor transmitido ao microcontrolador é fornecido em volts *(V)* pelo sensor, na qual, o microcontrolador utiliza um conversor *AD* para converter o sinal analógico em digital (SENSORS TRIANGULATION /ANDERSON).

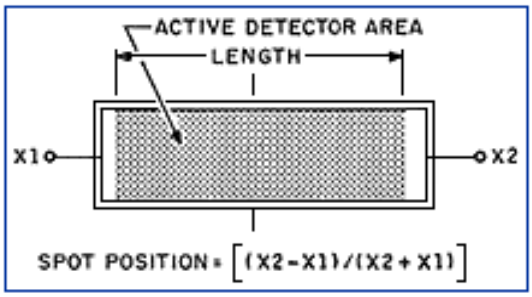

Figura 3.3 – Área ativa (Active Detector Area) que receberá o posição do ponto luminoso e passará para um processador.

(Fonte: http://archives.sensorsmag.com/articles/0598/tri0598/)

Para este projeto, o valor da distância não é considerado no cálculo projetado para o microcontrolador. A razão disso é pelo fato de existir uma relação entre a distância e a tensão. Como a tensão varia de acordo com a distância, é possível utilizar o valor da tensão para saber se objeto está se aproximando do sensor ou se afastando. A figura 3.4 é composta por duas imagens do sensor. Na primeira se encontram o emissor e receptor e na segunda a parte de trás do sensor.

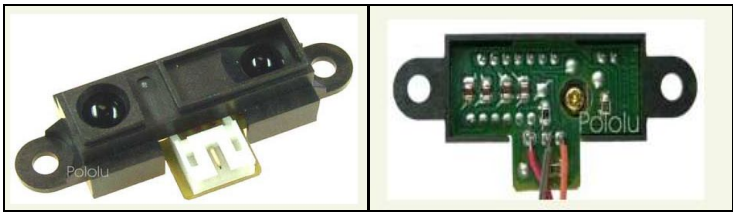

Figura 3.4 – Sensor *SHARP GP2D120* (Fonte: http://multilogica-shop.com/sensor-de-dist%C3%A2ncia-sharp-gp2d120xj00f-4-30cm)

Na figura 3.5, encontra-se o diagrama de blocos do sensor *SHARP GP2D120*. Este diagrama mostra de forma generalizada os componentes internos do sensor. Na figura 3.6 se encontram as dimensões físicas do sensor.

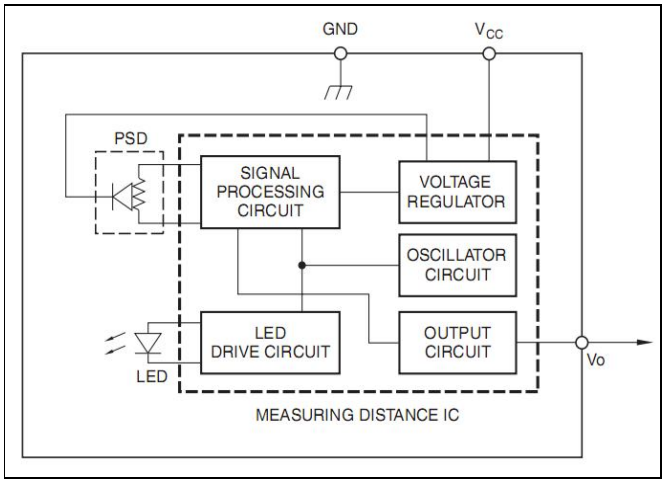

Figura 3.5 – Diagrama de blocos do sensor. (Fonte: DATASHEET)

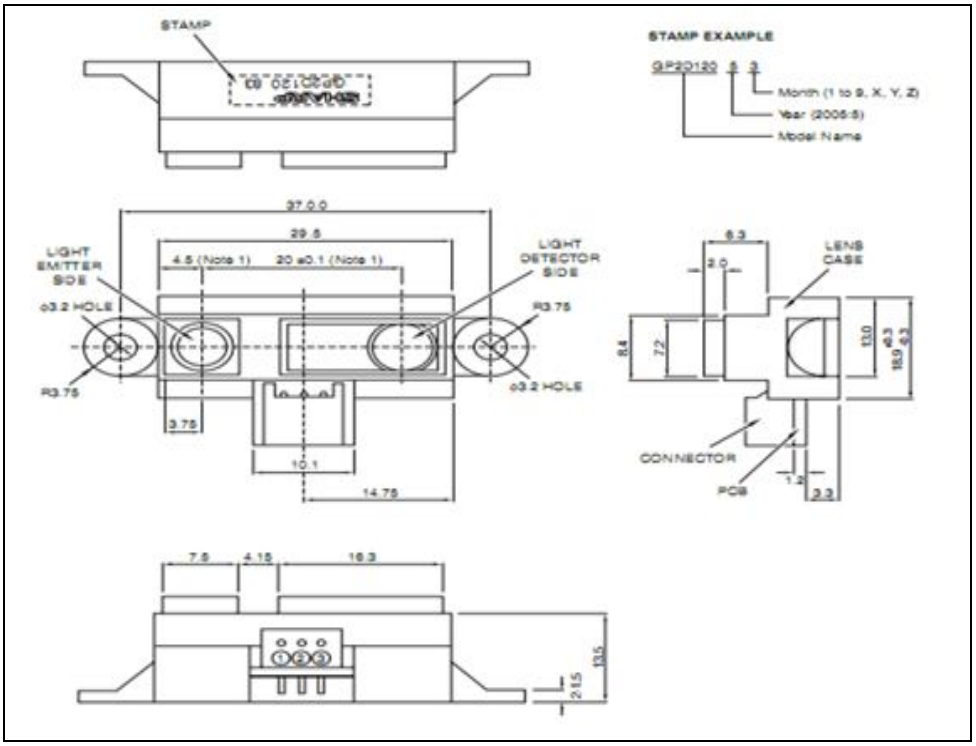

Figura 3.6 – Dimensões do sensor. (Fonte: DATASHEET).

Na figura 3.7, encontra-se a configurações de pinos do GP2D120. O pino1 é o pino responsável pela saída analógica do sinal processado, ou seja, após a captura do sinal, é função do pino1 transmitir a informação capturada. O pino2 é o pino que exerce a função de terra. O pino3 é o pino responsável pelo abastecimento de energia do sensor. (DATASHEET)

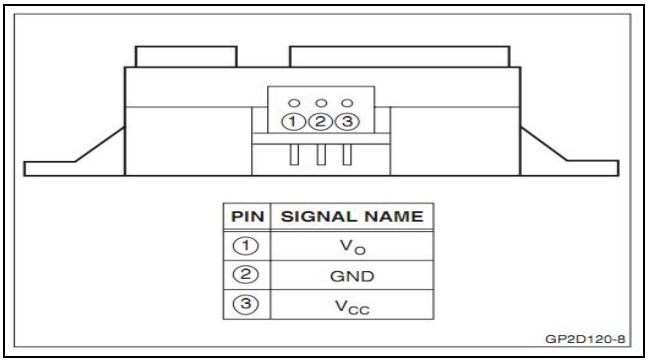

Figura 3.7 – Pinos de conexão do sensor. (Fonte: DATASHEET)

O sensor *SHARP GP2D120* opera, geralmente, a uma tensão de *VCC*=5 volts, podendo esta, sofrer variações e a uma tensão de saída que varia de (-0.3 volts até 0,3 + *VCC*). A tabela 3.1 possui as especificações eletrônicas desse sensor de forma mais detalhada, mostrando as classificações absolutas máximas e tensão operacional necessária para o funcionamento do sensor.

Tabela 3.1 – Especificações eletrônicas do sensor (Fonte: DATASHEET).

ELECTRICAL SPECIFICATIONS			
Absolute Maximum Ratings			
Ta = 25°C, V = 5 VDC			
PARAMETER	**SYMBOL**	**RATING**	**UNIT**
Tensão de Alimentação	VCC	-0.3 to +7	V
Tensão de Saída	VO	-0.3 to (VCC +0.3)	V
Temperatura Operacional	Topr	-10 to +60	°C
Temperatura Armazenada	Tstg	-40 to +70	°C
Tensão de Alimentação Operacional	VCC	4.5 to 5.5	V

O sensor *SHARP GP2D120* trabalha com uma faixa de distância que se estende de 4 a 30 cm, na qual, quando seu pico de tensão de saída se encontra em distâncias menores ele decresce, em uma curva, a medida em o objeto refletido se afasta do sensor. Na figura 3.8, encontra-se um gráfico que mostra a relação entre a tensão de saída que é fornecida e a distância. Neste gráfico também existe um comparativo quanto ao tipo de objeto que foi refletido pelo feixe infravermelho, se é um objeto cinza ou branco. Neste gráfico fica evidente o aumento de tensão de saída nas primeiras distâncias, ou seja, quanto mais próximo do sensor, maior é a sua tensão de saída (DATASHEET).

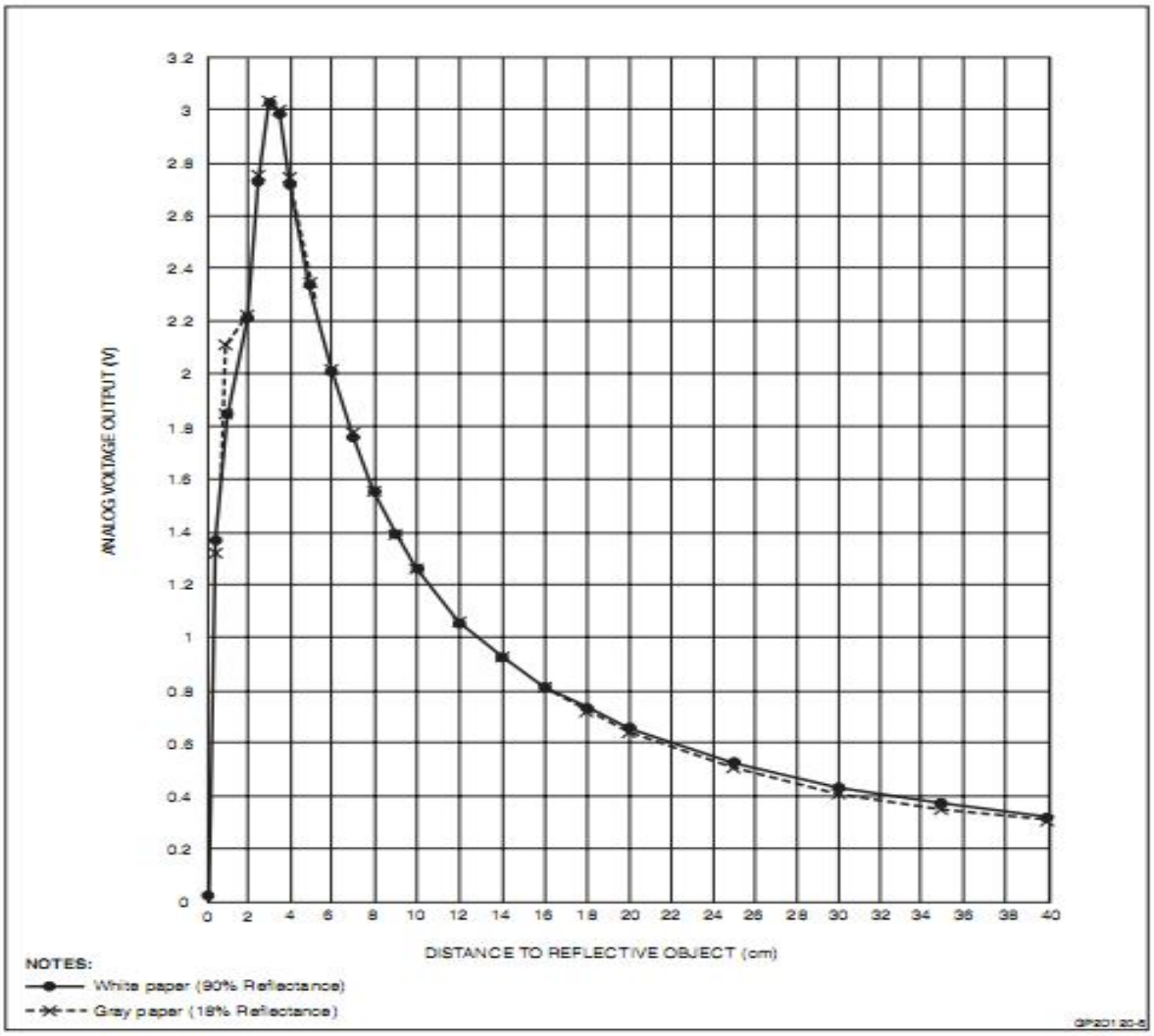

Figura 3.8 – Relação da tensão de saída pela distância. (Fonte: DATASHEET).

Também é possível verificar, na figura 3.8, que o comportamento da curva equivalente a relação direta da tensão de saída e a distância possui um formato exponencial. Esse comportamento dificulta a forma de calcular uma distância exata, entretanto a utilização de uma aproximação linear resolveria essa questão.

3.2 – Microcontroladores

A função do microcontrolador, neste projeto, consiste em receber as informações lidas pelo sensor infravermelho de distância, tratar essas informações com o código programado e transmitir o comando correto ao computador.

Um microcontrolador, tipicamente, já possui um aglomerado de dispositivos eletrônicos em um único *chip*. Esses dispositivos são um microprocessador, que consiste na Unidade Central de Processamento, a memória *ROM* (*Read Only Memory*) e a memória *RAM* (*Randon Access Memory)* (NICOLOSI, 2004).

3.2.1 – Microcontrolador ATmega328P

O *ATmega328P* é um microcontrolador de 8 bits de baixa potência que é desenvolvido pela empresa ATMEL. Este microcontrolador de arquitetura RISC trabalha a uma freqüência de 16MHz. Este microcontrolador possui as memórias Flash, EEPROM e RAM. O *ATmega328P* possui 32K Bytes de memória Flash, 1k Byte de memória EEPROM e 2k bytes de RAM. Ele também é composto por catorze portas digitais e seis portas analógicas (ATMEL, 2011).

3.2.2 – Placa Arduino Duemilanove

Uma das razões pela qual se escolheu a placa *Arduino Duemilanove* é devido ao seu tamanho de aproximadamente 37,1 cm^2 (7 x 5,3 cm), e a outra razão é o fato de existir uma conexão serial que se conecta diretamente a um computador via USB.

A placa *Arduino Duemilanove 328P* possui entrada e saída (*E/S*) integrante de uma plataforma de código aberto. Esta placa possui um *ATmega328P* como microcontrolador. A plataforma de programação da placa chamada *Arduino Alfa*, é uma *IDE* para o desenvolvimento de códigos em uma interface amigável para programadores. Para que ocorra a comunicação com o computador, a placa oferece uma porta *USB*, na qual, é utilizada tanto para provimento de energia quanto para transmissão de informação. Esse processo ocorre de forma paralela (ATMEL).

A placa Arduino, figura 3.9, possuí alguns componentes que são utilizados para funcionamento do projeto. Um dos componentes é o *LED* acoplado junto a pino digital 13, um botão de *reset*, e os pinos analógicos *(ANALOG IN)* e os de energia e aterramento (*POWER)*.

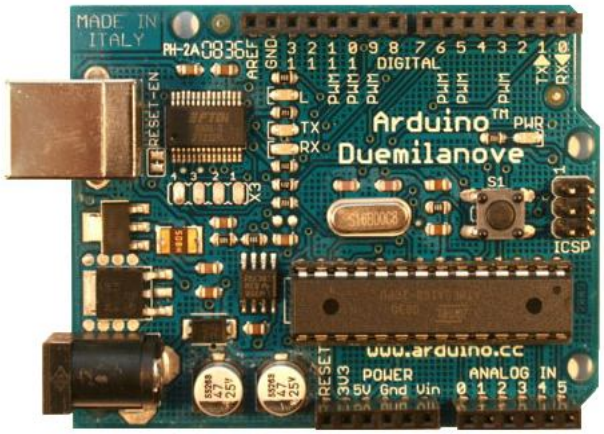

Figura 3.9 – Placa *Arduino Duemilanove com ATmega328P*. (Fonte: http://arduino.cc/en/Main/ArduinoBoardDuemilanove)

3.3 – Conexão USB

A *USB* (*Universal Serial Bus*) tem a particular função de permitir a conexão de muitos periféricos simultaneamente (pode-se conectar até 127 dispositivos em um barramento *USB*) ao barramento e este, por uma única tomada, se conecta à placa-mãe (MONTEIRO).

No caso do projeto proposto, para que a apresentação, que está iniciada no computador, sofra alteração de avançar ou voltar *slides* a partir das informações tratadas no microcontrolador, é necessário que haja conexão do microcontrolador com o computador, que além de prover energia também seja utilizada para a transmissão de informação. Para esta conexão, é utilizada a conexão *USB*.

Uma das razões pela qual foi escolhida a conexão *USB* é por conta de sua simplicidade na configuração e manuseio. Por se tratar de uma conexão confiável devido aos protocolos próprios, possuir compatibilidade com grande parte dos sistemas operacionais, possuir baixo custo. É importante observar que os computadores, atualmente, possuem várias portas *USB*. Outro fator que evidencia o porquê de se utilizar essa conexão é a questão da placa *Arduino Duemilanove* só possuir uma entrada e saída de dados via *USB*. Tendo em vista estas informações, fica evidente que a conexão *USB* é a mais indicada para o projeto. (AXELSON, 2009)

3.4 – Software Gobetwino

Cada componente do projeto possui sua característica própria e sua funcionalidade. O software *Gobetwino* é gratuito e é utilizado para agregar ao *Arduino com ATmega328p*, funções que o microcontrolador não consegue fazer por conta própria. O *Gobetwino* pode ser encontrado no site <http://www.mikmo.dk/gobetwino.html>. O *software* em questão funciona como uma espécie de *proxy* “genérico” (GOBETWINO).

A definição de *proxy* em ambiente de rede é um aplicativo que é configurado para agir em benefício da rede atribuída. Quando um aplicativo em execução em um *host* interno emite

uma solicitação de dados para fora da rede, o servidor *proxy* intercepta a solicitação, converte e passa-a para a rede-alvo (SCRIMGER).

A razão pela qual o *Gobetwino* é considerado um *proxy* "genérico" é pelo fato de ele ser um aplicativo que intercepta as informações passadas pelo *Arduino*, interpreta essas informações e envia os comandos configurados para o computador.

O *Gobetwino* intercepta as informações através da escuta da porta serial de modo que os comandos configurados no *Arduino* são interpretados e passados ao computador. O *Gobetwino* faz a função de um *driver* entre o *Arduino* e o computador (GOBETWINO).

A seguir, algumas funções que o *Gobetwino* é capaz de executar:

- iniciar um programa no computador;
- iniciar um programa e esperar até que ele se encerre;
- enviar um dado a qualquer programa do *Windows*, como se fosse um teclado;
- enviar um *e-mail;*
- realizar um *download* de um arquivo da internet;
- realizar um *ping* de um *host* ou um endereço *IP* (GOBETWINO).

O *Gobewino* necessita de certos requisitos recomendados pelo manual do usuário. Esse *software* foi desenvolvido para trabalhar, unicamente, com plataforma *Windows* e requer o *Microsoft .net 2.0* instalado. Esse software foi criado e testado no sistema operacional da *Microsoft Windows XP ServicePack 3* e ainda não foi testado nos sistemas operacionais *Windows Vista* e *Windows 7*. A interface de comunicação que o *Gobetwino* utiliza com o usuário é uma interface gráfica.

Na aba *Setting* é possível configurar a porta serial e configurar conta de email. Na aba *Commands* é possível configurar, dentre os comandos prontos, aquele comando que melhor satisfaz as necessidades do projeto. Na tela inicial do *Gobetwino*, é possível verificar o quê, exatamente, o código programado no *Arduino* está executando, juntamente com os comandos programados no microcontrolador.

A figura 3.10 mostra a tela inicial do *Gobetwino*, na qual a *string* de comando *TESTE* é ativada para que se abra o documento do *Power Point* configurado neste comando. A figura

3.11 é composta de duas imagens que são abas adjacentes a aba *Setting*. Essas abas são *Mail* e *Serial Port*.

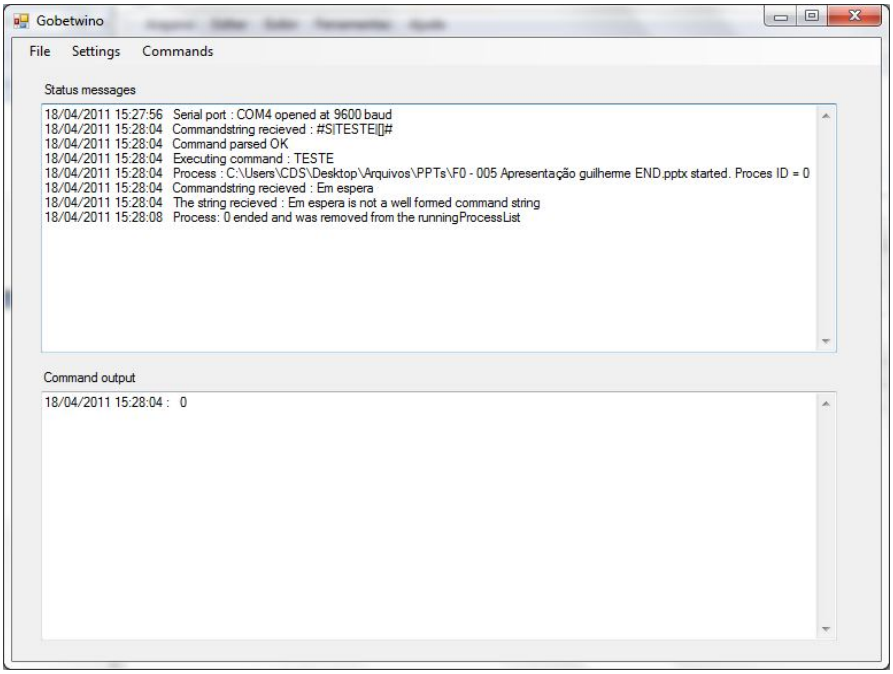

Figura 3.10 – Tela inicial do Software *Gobetwino*. (Fonte: Autor).

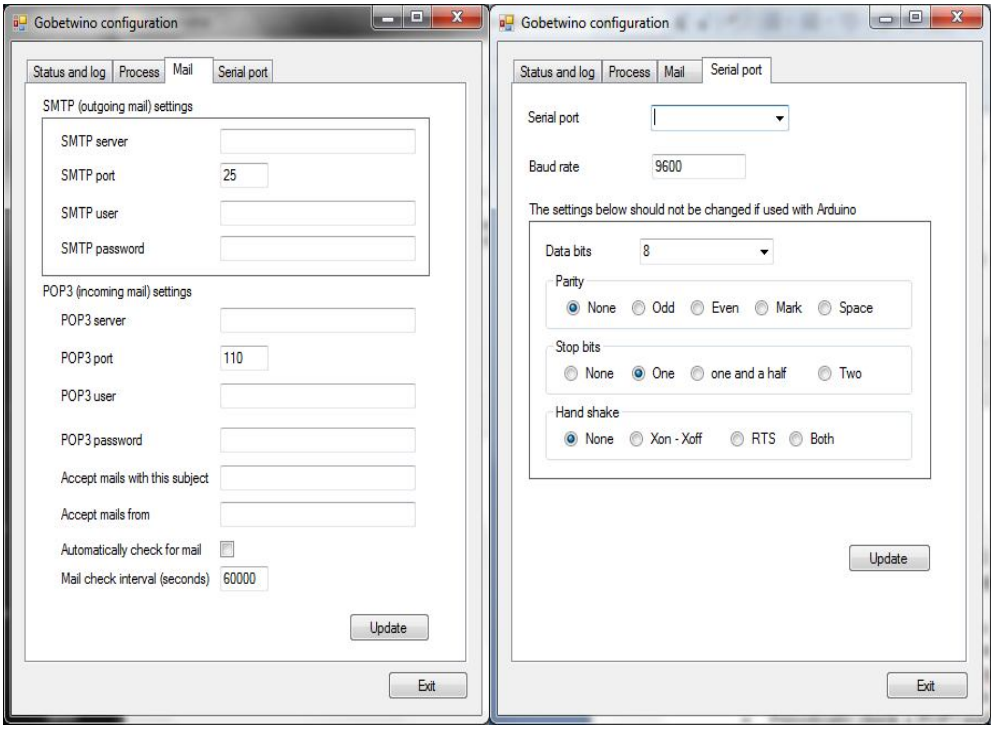

Figura 3.11 – Abas adjacentes à aba *Setting* da tela inicial do Software *Gobetwino, Mail* e *Serial Port*. (Fonte: Autor).

A figura 3.12 mostra a terceira aba da tela inicial do *Gobetwino* que é referente à aba *Commands*. Nesta terceira aba ocorre a configuração de vários comandos padrão que possuem diversas funções. O comando *SPRID*, em específico, é um exemplo de comando configurado com o nome *ABREPPT*.

O *SPRID* é um tipo de comando que inicia um programa no computador, na qual, o *Gobetwino* utiliza um identificador para esse programa. Esse identificador pode ser colocado no programa do *Arduino* para que este envie atalhos de teclas, por exemplo, *Page up, Page down e F8*. O comando *SPRID* possibilita iniciar qualquer arquivo executável, como por exemplo, *(.exe), (.bat) ou (.cmd)*. Este comando também permite que se execute um arquivo de extensão (*.txt)*, por exemplo, e com isso abre um editor de texto como o *Notepad*. (GOBETWINO).

A sintaxe que serve para iniciar o programa no *Arduino* que é fornecida pelo manual do usuário do *Gobetwino*, esta disposta conforme apresentado a seguir.

Serial.println("#S|NAME|[]#");

No caso deste projeto, o programa que é executado é a própria apresentação, em *Power Point*, que recebe os comandos para avançar ou voltar *slides*.

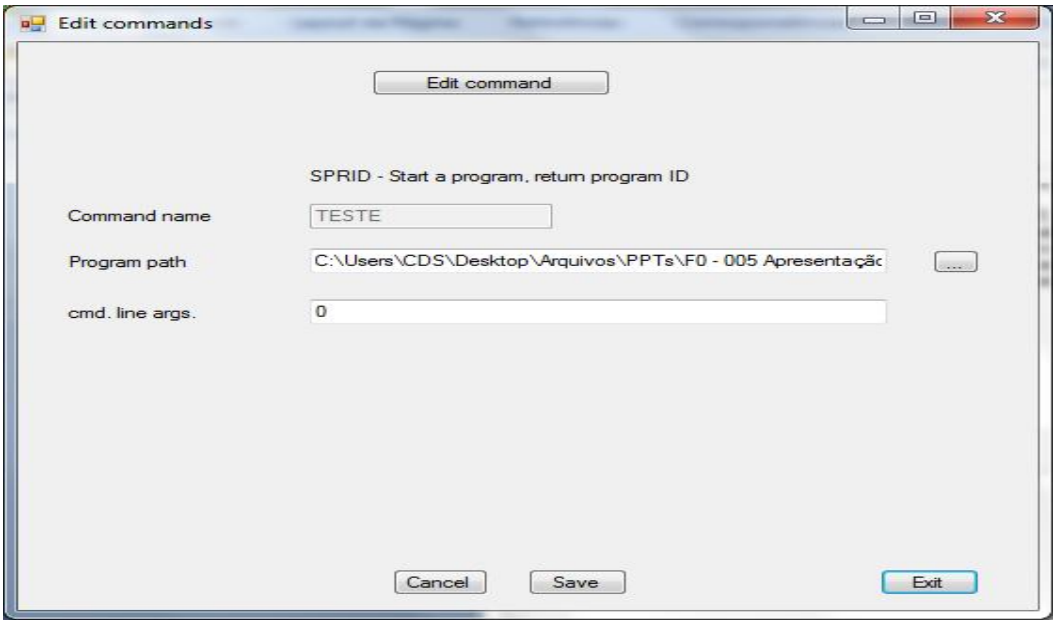

Figura 3.12 – Aba *Commands* da tela inicial do *software Gobetwino*. (Fonte: Autor).

Esse comando de avançar ou retroceder *slides* necessita de um comando chamado *SENDK*. O *SENDK* é um dos comandos especiais do *Gobetwino* que não possuem parâmetros internos. Este comando tem a capacidade de simular o envio de uma tecla de um teclado de computador ao programa iniciado pelo comando *SPRID*. Com isso o programa iniciado irá reagir a essa tecla pressionada alterando seu estado atual. O *SENDK* é considerado um comando com um privilégio alto, pois consegue, literalmente, controlar a maioria dos programas do *Windows* (GOBETWINO).

A sintaxe que serve para enviar uma chave ou comando de um teclado, por exemplo, é fornecida pelo manual do usuário do *Gobetwino*, na qual o *PID* é o identificador que foi colocado no programa executado pelo *SPRD*, que permite que o *Arduino* envie a informação e o *keystrokes to send* é a *string* com informação da suposta tecla pressionada. A tabela 3.2 mostra as chaves e seus respectivos argumentos necessários para a construção desta sintaxe.

Serial.println("#S|SENDK|[PID&keystrokes to send]#")

Tabela 3.2 – Tabela contendo algumas *keystrokes* juntamente com sua sintaxe. (Fonte: GOBETWINO).

Key	Argument	Key	Argument
BACKSPACE	{BACKSPACE}, {BS}, or {BKSP}	TAB	{TAB}
BREAK	{BREAK}	UP ARROW	{UP}
CAPS LOCK	{CAPSLOCK}	F1	{F1}
DEL or DELETE	{DELETE} or {DEL}	F2	{F2}
DOWN ARROW	{DOWN}	F3	{F3}
END	{END}	F4	{F4}
ENTER	{ENTER} or ~	F5	{F5}
ESC	{ESC}	F6	{F6}
HELP	{HELP}	F7	{F7}
HOME	{HOME}	F8	{F8}
INS or INSERT	{INSERT} or {INS}	F9	{F9}
LEFT ARROW	{LEFT}	F10	{F10}
NUM LOCK	{NUMLOCK}	F11	{F11}
PAGE DOWN	{PGDN}	F12	{F12}
PAGE UP	{PGUP}	F13	{F13}

Na tabela 3.3, se encontram algumas teclas consideradas especiais, pois suas combinações com as teclas acima permitem diferentes comandos.

Tabela 3.3 - Tabela contendo algumas *keystrokes* especiais juntamente com sua sintaxe. (Fonte: GOBETWINO)

Key	Argument
SHIFT	+
CTRL	^
ALT	%

3.5 – Linguagem *Wiring*

Wiring é uma plataforma *open source* de protótipo eletrônicos composto por um ambiente de programação *(IDE),* uma placa prototipo eletrônica e a documentação criada por designer. Muitos estudantes, pesquisadores e desenvolvedores utilizam a linguagem *Wiring* para prototipagem e trabalhos profissionais (WIRING).

A linguagem com as estrutura utilizadas no código criada para o microcontrolador é a linguagem *Wiring* e sua estrutura pode ser encontrada em < http://wiring.org.co/reference/ >.

CAPÍTULO 4 – MODELO PROPOSTO

4.1 – Apresentação do Modelo Proposto

A proposta deste projeto é criar um dispositivo que avance ou retroceda *slides* com um movimento da mão em uma determinada direção. Neste dispositivo são necessários componentes de *hardware e software* que atuando de forma conjunta permitem que o dispositivo funcione corretamente. Para que esta solução seja concluída, alguns passos são necessários.

A figura 4.1 mostra o processo de funcionamento do dispositivo que se inicia com o sistema criado a espera de um objeto que interrompa o raio de atuação do sensor. A partir do momento em que o objeto, ou no caso a mão do usuário, entra neste raio, um valor é obtido pelo sensor e armazenado, conforme o programa. Na medida em que o objeto avança em uma direção, novos valores são computados. Quando o código assimila e interpreta todos os valores, é possível compreender o comportamento do objeto, e com isso o microcontrolador aciona um comando, que faz parte do código criado, e o software *Gobetwino* se encarrega de transmitir a informação ao computador.

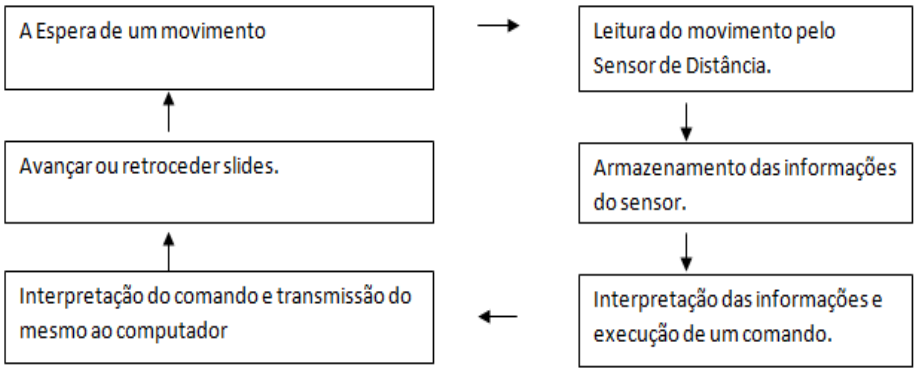

Figura 4.1 – Processos necessários para funcionamento do sistema. (Fonte: Autor)

A parte de conexão entre microcontrolador e computador é feita por *USB*, e por meio dessa conexão é que o *software Gobetwino* recebe as informações necessárias para enviar um comando ao computador para este avançar ou retroceder *slides*. A figura 4.2 mostra o processo que se estende desde o sensor até a mundança de *slides*.

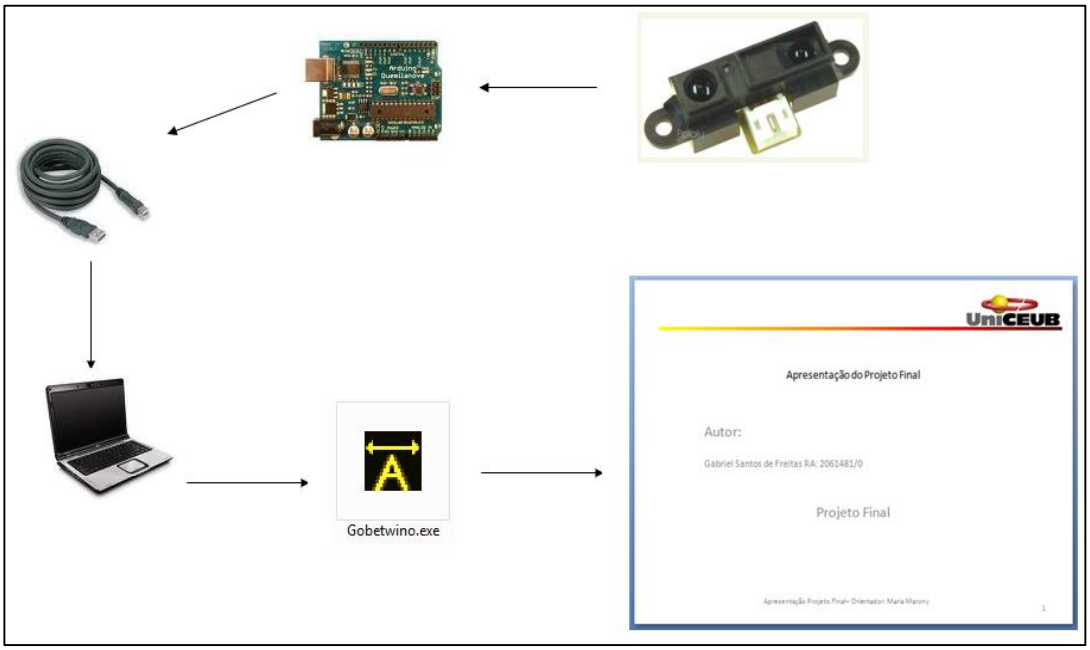

Figura 4.2 – Ilustração do diagrama esquemático do projeto físico. (Fonte: Autor)

4.2 – Diagrama Esquemático do Protótipo do Projeto

No decorrer do processo de montagem, programação e teste, é necessário traçar e construir um sistema físico que funcione como suporte para a integração de todos os componentes do projeto.

O projeto é composto, fisicamente, por um microcontrolador e um sensor infravermelho de distância. Esse sensor, quando em funcionamento, obtém valores diferentes mesmo que não haja a interrupção de um objeto. Quando um objeto interrompe o raio de alcance do sensor e permanece parado naquela distância, os valores obtidos pelo sensor sofrem alterações pequenas entre si, tanto para valores maiores quanto menores. A figura 4.3, composta de duas figuras que representam, respectivamente, valores calculados pelo sensor sem objeto e com objeto. Estes valores são exibidos com a utilização do monitor *serial* da

IDE, *Arduino Alpha*, utilizada para a programação do microcontrolador. Com esta figura é possível verificar que mesmo com o objeto parado em uma posição, os valores oscilam de forma não padronizada.

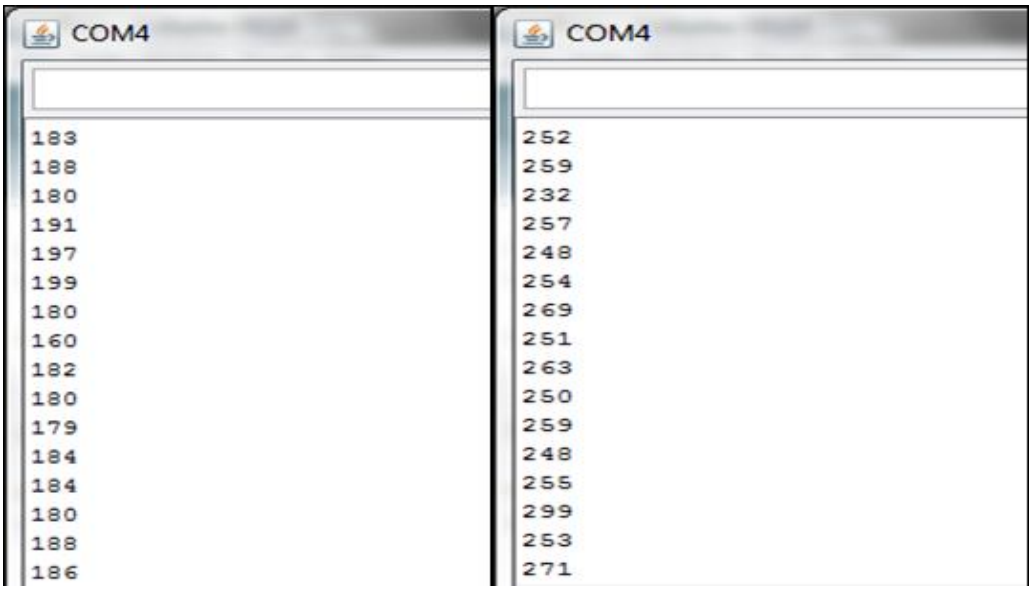

Figura 4.3 – Comparação dos valores sem o objeto e com o objeto em uma posição fixa, respectivamente. (Fonte: Autor)

A idéia proposta pelo autor para a programação do microcontrolador é a de utilizar a variação dos valores obtidos pelo sensor em diferentes distâncias e analisar o comportamento delas em pequenas amostras. Em outras palavras, quando um objeto interrompe o raio de atuação do sensor, novos valores são encontrados e ficam variando entre si em pequenos valores.

A partir do momento em que se move o objeto para uma distância maior ou menor, os valores, nesta nova distância, se alteram consideravelmente em relação a valores obtidos na distância inicial, mas mesmo assim continuam alterando entre si em pequenos valores. A figura 4.4, é composta por duas imagens que mostram, respectivamente, um objeto mais distante do sensor e a outra mais próxima do sensor.

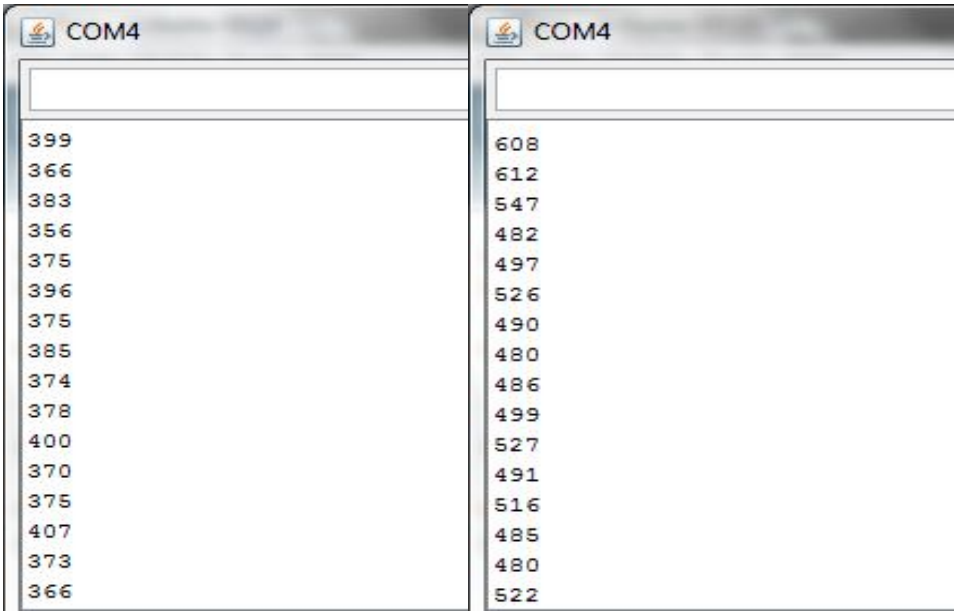

Figura 4.4 – Comparando os valores de um objeto a uma distância inicial e uma final respectivamente. (Fonte: Autor)

Os valores obtidos em diferentes distâncias permitem analisar se o objeto se aproxima ou se afasta do sensor. Baseado nisso é desenvolvido um código para o microcontrolador. No entanto, as pequenas variações nos valores, que ocorrem quando se tem ou não um objeto disposto no raio de alcance do sensor, é assimilado pelo código e pela lógica. Esses novos valores são tratados como se o objeto estivesse se afastado ou aproximado do sensor.

Essas variações que ocorrem implicam ao código criado que está ocorrendo a movimentação do objeto. Após a análise de uma amostra de valores, o código envia ao computador o comando necessário e o *Gobetwino* realiza o processo de enviar um comando para alterar os *slides*.

Para evitar esse tipo de problema, é proposto alocar uma barreira de segurança a uma distância fixa do sensor. A idéia de colocar esta barreira é para evitar que as alterações nos valores forcem o código criado a avançar ou retroceder *slides* sem que ninguém realize nenhum movimento e, principalmente, para traçar um ambiente de ação delimitado pelo sensor e a barreira. Em outras palavras, criar um limite de atuação para que apenas os valores encontrados entre o sensor e a barreira sejam de fato utilizados para avançar ou voltar *slides*. Para isso é realizado um processo de calibração do sensor para que a maior distância encontrada em uma amostragem seja o limite de atuação. Após o processo de calibração, não

há necessidade de se manter a barreira de segurança, pois o valor máximo calculado é armazenado em uma variável, fazendo com que qualquer valor que esteja a uma distância maior que a da barreira seja desconsiderado.

4.3 – *Hardwares e Softwares* do modelo proposto

Para a consolidação do projeto é utilizado o *software* gratuito *Gobetwino* e o código desenvolvido, pelo autor, para ser utilizado no microcontrolador.

4.3.1 – Configuração do Gobetwino

O processo de funcionamento se inicia com a configuração e conexão entre o software *Gobetwino* e o microcontrolador. Após essa configuração é possível utilizar o protótipo para avançar ou retroceder os slides.

Nesta etapa, abre-se o aplicativo para criar e configurar o nome do comando e o documento que deve ser apresentado. Para isso é necessário acessar a aba *Setting* e em seguida a aba *Serial port*. A figura 4.5 mostra a tela inicial e a tela *Setting* do *Gobetwino.*

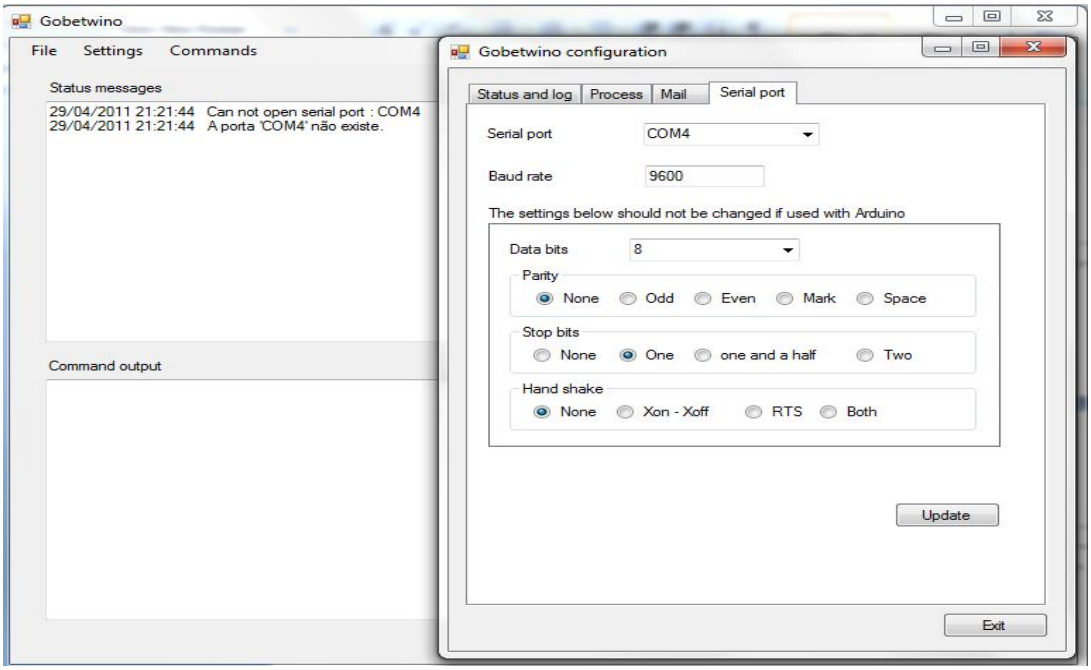

Figura 4.5 – Tela inicial do *Gobetwino* e a aba *Serial port* da aba principal *Settings*. (Fonte: Autor)

Na aba *Serial port* é configurado a porta *serial* e *Bound rate* que possui a configuração padrão 9600. A porta serial configurada é a *COM5*, e essa porta *serial* varia de acordo com o microcontrolador. Essa é a porta *serial* padrão que a *Arduino* utiliza para realizar conexão com o computador. A razão pela qual a porta *COM5* é utilizada, ao invés de outra porta *serial*, para configurar o *Gobetwino* é o fato de que o *Gobetwino* funciona como um intermediador entre o microcontrolador e o computador para realizar certas funções, ou seja, para que esse *software* consiga transmitir a informação desejada, é necessário que ele fique escutando a porta serial *COM5*, que é a porta de saída do microcontrolador, a espera de um comando transmitido pelo microcontrolador que faça com que ele mande um comando para o computador. Esse comando é passado direto para um aplicativo que é configurado nele, como por exemplo, uma apresentação em *Power Point*, um documento do *Word*, um bloco de notas, dentre outros (GOBETWINO).

Um ponto importante a ser tratado sobre a conexão é a forma como o *Gobewino* entende o comando passado pelo *Arduino.* Dentro do programa compilado e gravado no microcontrolador é necessário colocar um comando que faz parte da linguagem *Wiring* que é o *Serial.println().* A partir dessa linha de código e o que está configurado dentro dos parênteses, que o *Gobetwino* interpreta e processa a informação e a manda para o computador (GOBETWINO).

Após a configuração da porta *serial*, é necessário criar um comando e associar o mesmo a um aplicativo. A criação de comando e configuração é feita na aba *Commands,* figura 4.6, que se encontra da janela principal do *Gobetwino.*

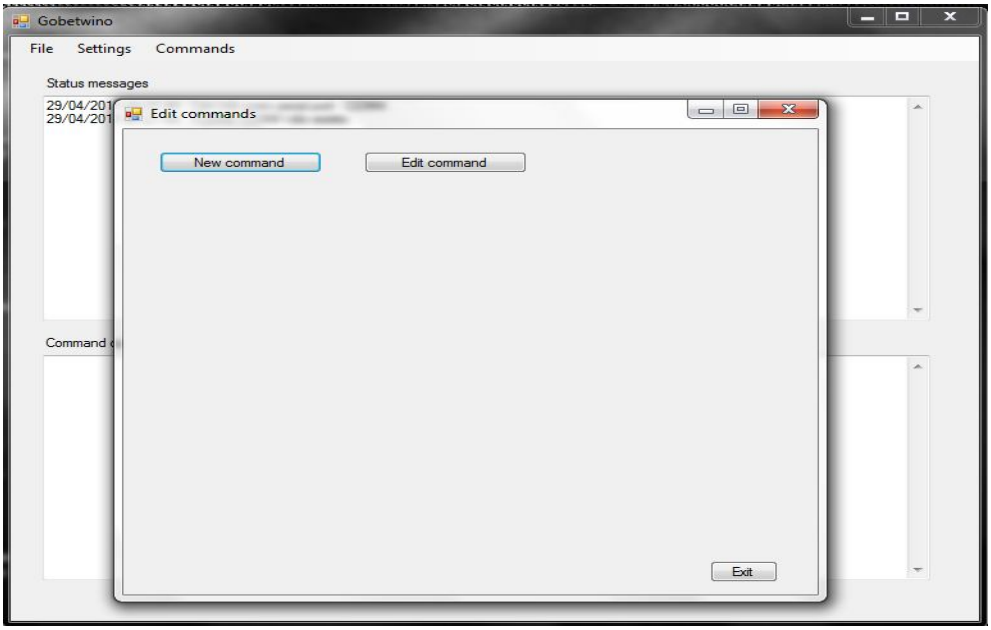

Figura 4.6 – Aba *Commands*, para a criação e configuração de um comando. (Fonte: Autor).

Na aba *New command* é que se escolhe um dos comandos que são padrões que vieram pré-programados. Estes comandos são:

- *SPRID* – Inicia um programa do *Windows*;

- *DLFIL* – Realiza *download* de um programa;

- *SPWEX* – Inicia um programa e esperar pelo término do mesmo;

- *RFLIN* – Lê a linha de um arquivo;

- *LGFIL* – Realiza *log* de arquivos;

- *CPFIL* – Copia um arquivo;

- *SMAIL* – Envia um *e-mail*;

- *PING* – Realiza um comando de *ping*. (GOBETWINO)

Para o protótipo deste projeto, é necessário apenas o comando *SPRID*. Este comando tem a função de iniciar um programa e permanecer com ele aberto a espera de comandos para alterações no mesmo. Por se tratar de um protótipo de um *slider*, ou seja, de um dispositivo para avançar ou retroceder *slides*, basta enviar à apresentação os atalhos padrões utilizados nos teclados convencionais para alterar os *slides*. Geralemente, algumas teclas, de um teclado comum de computador, quando pressionadas, como o *Page Down* e *Page Up*, permitem que um programa, em *Power Point*, por exemplo, avance e retroceda *slides*.

A figura 4.7 mostra o comando *SPRID* sendo configurado no *Gobetwino*, juntamente com um arquivo do *Power Point*. No parâmetro *Program path*, é necessário colocar o caminho do arquivo que se deseja executar. No paramento *Cmd line args* coloca-se qualquer argumento que se deseje passar ao programa iniciado (GOBETWINO).

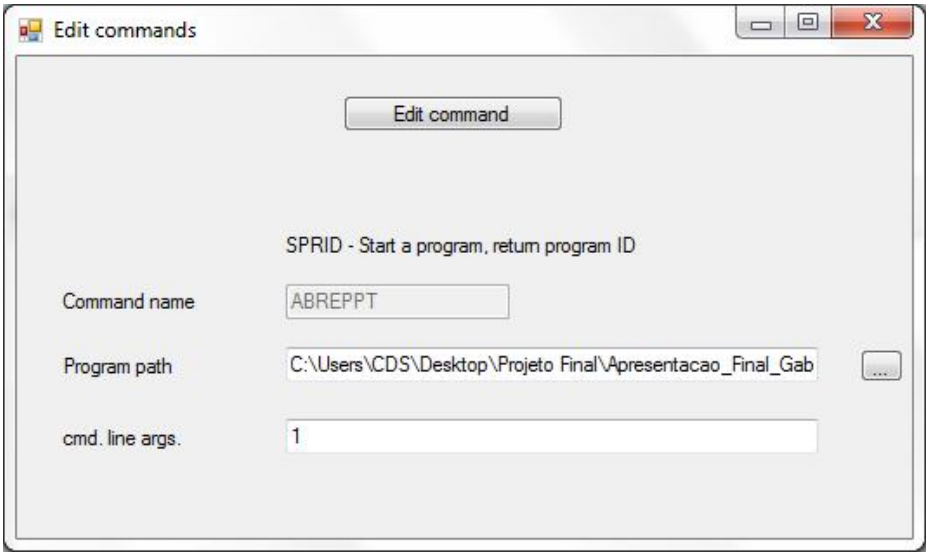

Figura 4.7 – Aba *Commands* configurada para abrir um *Power Point*. (Fonte: Autor)

Para que o programa configurado no *Gobetwino* seja alterado tendo como base as informações passadas pelo microcontrolador, basta colocar o comando *Serial.println()* no código do microcontrolador.

A linha do código utilizado no microcontrolador para este projeto para iniciar a apresentação é:

Serial.println("#S|ABREPPT|[]#");

Para que sejam enviados comandos para o programa aberto é necessário colocar, dentro desses parênteses, um comando chamado *SENDK*, anteriormente mencionado.

Abaixo, está a linha do código utilizado no microcontrolador para enviar o comando de retroceder slide, que no caso foi a tecla *Page up*:

Serial.println("#S|SENDK|[0&{PGUP}]#");

4.3.2 – Configuração Gobetwino com o Arduino

Como já foi mencionado anteriormente, tanto o microcontrolador *Arduino com ATmega328p* quanto o *Gobetwino* utilizam a porta serial *COM5*, ou então outra porta desde que sejam as mesmas, para entrada e saída de informação. No entanto, não é possível ter dois dispositivos utilizando a mesma porta serial no mesmo instante, tendo em vista que o *Arduino* não realiza o *upload* de um programa se outro dispositivo já estiver utilizando porta *serial* configurada. Para que isto seja possível, é necessário realizar um procedimento de *reset* no microcontrolador.

Para o processo de utilização simultânea na porta *serial* é necessário, primeiramente, realizar o *upload* do código, tendo em vista que nenhum dispositivo esteja usando a porta serial no mesmo instante do *upload*, a ser executado no Arduino, no *Arduino Alpha* a *IDE* de programação. A figura 4.8 mostra a *IDE Arduino Alpha* com um trecho do código.

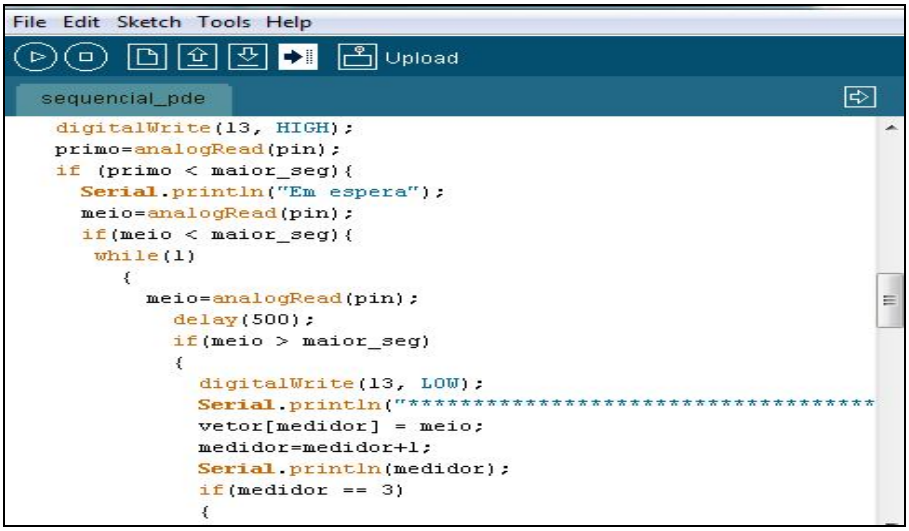

Figura 4.8 – Ilustração do *IDE*, com um trecho do código a ser compilado no microcontrolador. (Fonte: Autor).

Após a realização do *upload* do programa, é necessário iniciar o aplicativo *Gobetwino.exe*. Em seguida, é necessário pressionar o botão de *reset* que existe no microcontrolador, figura 3.9.

O botão *reset* tem a função de recompor o microcontrolador, na qual é executado novamente, o processo de compilação e a inicialização do programa. Quando ocorre o *upload* do programa no mesmo, a porta serial *COM5* fica alocada ao microcontrolador servindo de conexão entre o *Arduino* e o computador. Após o *upload* do programa, é necessário executar o *gobetwino.exe* para que o *Gobetwino* passe a escutar a porta *serial.*

Quando botão *reset* do microcontrolador é pressionado, ocorre uma nova execução do programa compilado. No entanto, o *software Gobetwino* está, constantemente, conectado a porta *serial* após a sua execução. No momento em que o botão de *reset* é pressionado, o código compilado e executado, anteriormente, no *Arduino*, passa a ser lido e interpretado pelo *Gobetwino*.

Na figura 4.9, encontra-se um diagrama esquemático que menciona o processo por *upload* comum, e com o botão *reset*.

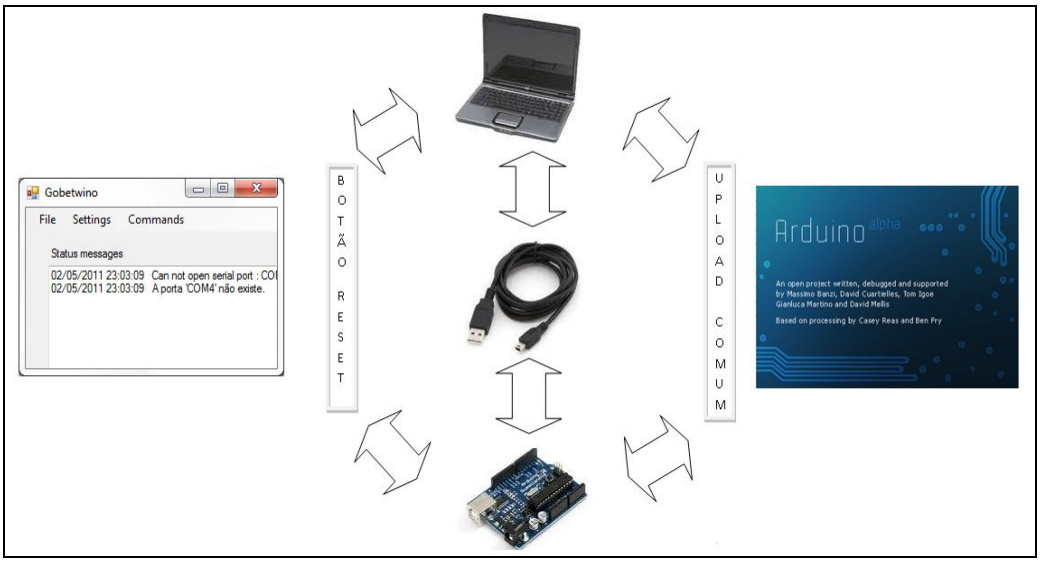

Figura 4.9 – Ilustração de um diagrama esquemático que mostra a comunicação com o botão de *reset* e o *upload* comum. (Fonte: Autor).

4.3.3 – Ligação do sensor ao microcontrolador

Em nível de *hardware*, para que seja possível tabalhar o sensor infravermelho de distância juntamente com o microcontrolador, é necessário fornecer ao sensor um canal de energia, terra e comunicação.

O sensor *SHARP GP2D120* possui três pinos de conexão. O pino 1 responsável pelo canal de comunicação entre o sensor e o microncontrolador. O pino 2 é responsável por Terra. O pino 3 é responsável pelo fornecimento de energia, figura 3.7. (DATASHEET)

Por se tratar de um sensor óptico, as comunicações entre o sensor e microcontrolador são em pinos de entrada analógico do microncontrolador, as seção *ANALOG IN*. A relação de fornecimento de energia e terra serão trabalhos na seção *POWER*.

De acordo com o *DATASHEET* do sensor, o sensor *SHARP GP2D120* trabalha em um faixa de tensão operacional de 4.5 a 5.5 *volts*. Na seção *POWER* existe um pino que fornece a tensão de 5 volts (5V) e dois pinos de terra (Gnd).

4.3.4 – Calibração do Sensor em Programação.

Na programação do microcontrolador, ocorreu um processo de calibração do sensor de distância tendo em vista a barreira de segurança. Como mencionado, anteriormente, essa barreira de segurança tem o objetivo de evitar que os *slides* sofram alteração sem a interferência de ninguém.

Na figura 4.10, encontra-se o código criado em linguagem *Wiring* utilizado para pegar o maior valor dentre uma amostra de quinhentos (500) registros. Esse valor de quinhentos foi escolhido para que tivesse uma amostragem completa. Com a diminuição desses valores alguns dados esperados foram desconsiderados, tornando assim o valor de quinhentos (500) o mais adequado. O maior valor reflete diretamente na menor distância encontrada para a barreira de segurança.

Figura 4.10 – Ilustração da *IDE* da sobre o trecho do código referente a calibração do sensor de distancia me relação a barreira de segurança. (Fonte: Autor).

Após obter o maior valor que fica armazenado na variável *maior*, ocorre também o cálculo de um valor de segurança que equivale ao maior valor mais dez por cento (10%) do mesmo. O valor de segurança é armazenado na variável *maior_seg* e este é o valor que de fato é utilizado para limite de atuação, mencionado anteriormente. Este valor de segurança serve para evitar que o sensor de distância encontre um valor que seja maior do que o maior valor encontrado na amostra de quinhentos (500).

4.4 – Execução do Processo

Para que se possa executar e utilizar o protótipo do projeto, é necessário preparar o sistema. O primeiro passo é configurar o *Gobetwino*, na seqüência, realizar o *upload* do programa no microcontrolador e pressionar o botão *reset* para que o *Gobetwino* assuma controle da porta *serial* e receba as informações do microcontrolador. Com o sistema já preparado, já é possível utilizar o dispositivo criado para avançar ou retroceder *slides*.

4.4.1 – Funcionamento do Sensor junto ao LED do microcontrolador

Para que o sensor faça a leitura correta dos valores, é necessária, no momento em que se executa o movimento, que se observe um *LED* acoplado a placa *Arduino* junto ao pino 13 da sessão digital. Esse *LED* é responsável por avisar ao apresentador o momento correto para que ele possa mover a mão.

Quando o objeto ou a mão do apresentador interrompe o campo de ação do sensor, a leitura da tensão, naquela primeira posição, é convertida e armazenada em um vetor. Após a armazenagem desse valor, o *LED*, que anteriormente estava aceso, se apaga por um curto período para que o objeto ou o apresentador realize um movimento. Em seguida, ao tempo delimitado, o *LED* torna a acender indicando que o processo de leitura já se encerrou, permitindo ao usuário retirar a mão ou objeto do campo de ação do sensor. Para a apresentação do protótipo para a banca, é utilizado um *LED* verde apenas para representar o *LED* da placa de forma evidente. A figura 3.9 mostra o microcontrolador com o *LED* junto a porta digital 13.

4.4.2 – Funcionamento do sensor junto ao sistema

Primeiramente, o sensor de distância fica enviando, constantemente, um feixe infravermelho. A partir do momento em que o feixe encontra a barreira de segurança, o feixe é refletido no sensor, fornecendo assim um valor para o microcontrolador.

Esse valor corresponde à tensão de saída do sensor de distância. Quando o feixe infravermelho é refletido no receptor do sensor, é gerada uma diferença de potencial (V) que é fornecido ao microcontrolador. No entanto, esse valor recebido é convertido, digitalmente, pelo microcontrolador. Na figura 4.11 mostra o processo de quando feixe emitido é refletido na barreira de segurança.

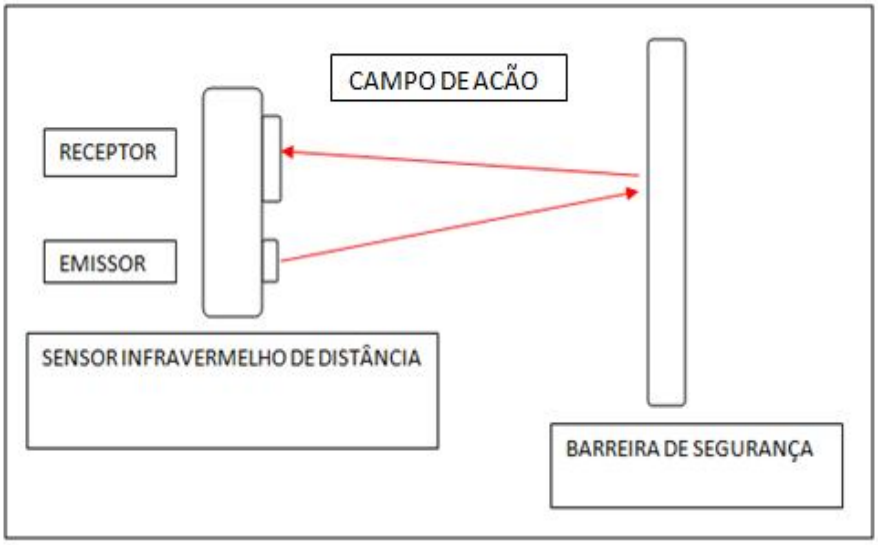

Figura 4.11 – Ilustração de um feixe infravermelho saindo do sensor de distância e sendo refletido na barreira de segurança. (Fonte: Autor).

O cálculo da tensão é realizado no sensor de distância e passado para o microcontrolador, que através do código compilado nele, armazena esses valores em um vetor numérico. Antes de o valor ser armazenado no vetor, ocorre uma validação para saber se a tensão obtida pelo sensor é correspondente a um objeto que interrompeu a distância entre o sensor e a barreira de segurança ou se é a tensão obtida pela reflexão do infravermelho na barreira de segurança. Caso a tensão seja de um objeto diferente da barreira de segurança, o valor digital é armazenado no vetor. Na figura 4.12 é ilustrada um trecho do código que exemplifica a validação da tensão e o processo de armazenamento do valor no vetor.

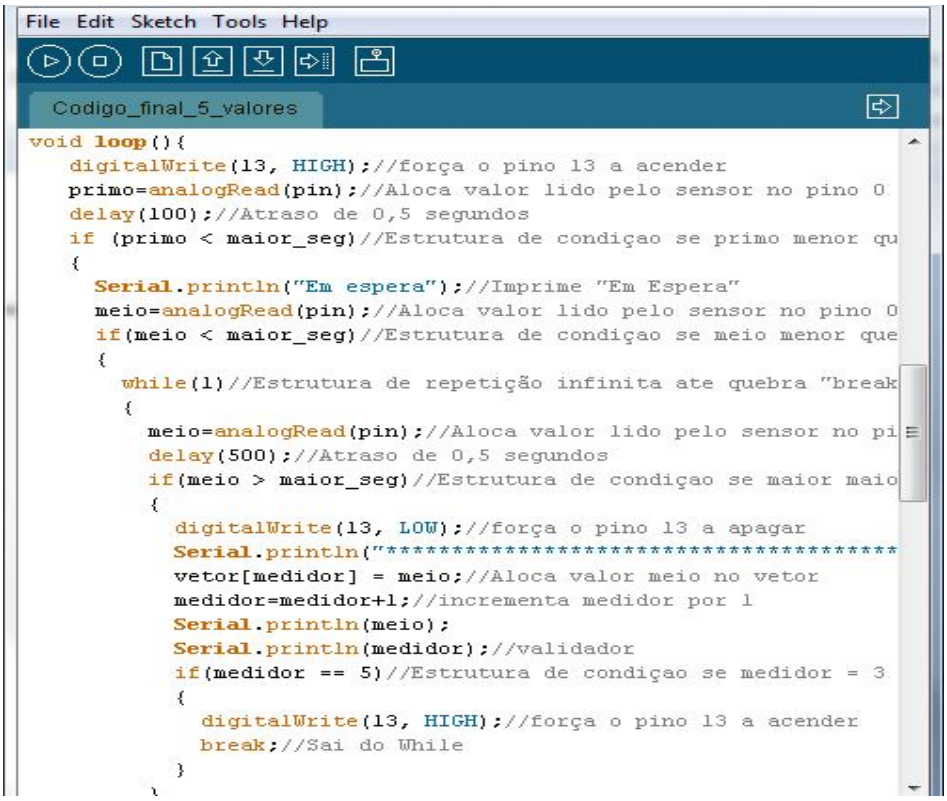

Figura 4.12 – Ilustração sobre o processo de armazenagem dos valores no vetor. (Fonte: Autor).

Primeiramente é calculado o valor da variável *primo*. Esta variável tem a função de armazenar o primeiro valor calculado pelo sensor que equivale a um valor qualquer baseado na reflexão do feixe infravermelho na barreira de segurança. Necessariamente, o primeiro valor calculado será menor que o valor armazenado em *maior_seg*. Isto se deve ao fato de que após a calibração do sensor, a variável *maior_seg* recebe o maior valor dentre os 500 valores obtidos além da adição ao seu valor original de dez por cento (10%) do mesmo.

As condições, *if(primo < maior_seg)* e *if(meio < maior_seg)*, servem unicamente, para garantir que o sistema está funcionando. Ou seja, garantem que os valores calculados pelo sensor são, necessariamente, menores que a distância de segurança, garantindo que nenhum objeto se interpôs no campo entre o sensor e a barreira de segurança.

A partir do momento que se passa pelas duas condições, a estrutura de repetição *while(1)* assegura que o sistema só passará para a próxima etapa se algum objeto interromper o campo de ação. Enquanto nenhum objeto for identificado, os valores digitais das tensões são armazenados, temporariamente, na variável '*meio*' que em seguida é validado pela condição *if(meio > maior_seg)*.

Caso a valor armazenado em *meio* seja maior que a tensão armazenada em *maior_seg*, significa que algum objeto se interpôs no campo de ação fazendo com que a condição mostrada acima, passe a ser verdadeira e que o valor registrado na variável *meio* seja alocado no primeiro espaço do vetor, sobrescrevendo qualquer valor, anteriormente armazenado.

Quando um objeto interrompe o campo de ação em um ponto qualquer, este primeiro ponto é o marco de início do processo de avançar ou voltar *slides*. Se o primeiro ponto for mais próximo do sensor, a tendência do movimento é o sentido da barreira de segurança, caso contrário, a tendência do movimento é na direção do sensor. Estando o objeto no meio existem as duas possibilidades de movimento podendo avançar ou voltar *slides*. Na figura 4.13, é mostrado o objeto em três posições diferentes na qual o sensor, em cada uma das etapas, está à esquerda e a barreira de segurança à direita.

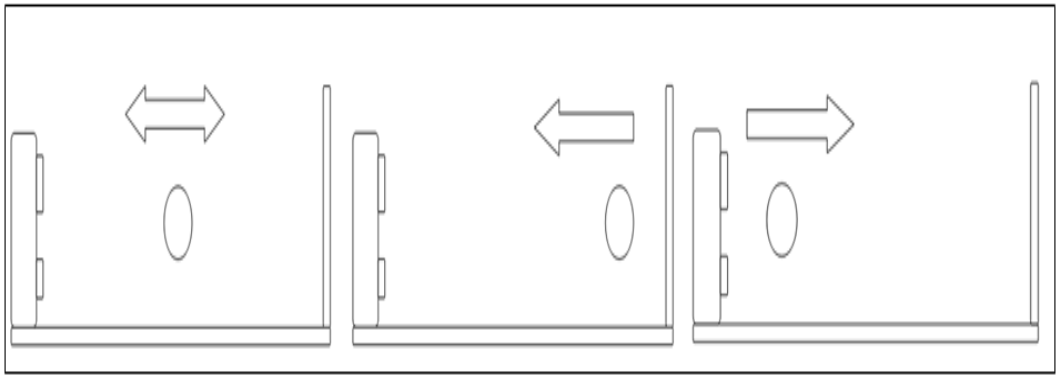

Figura 4.13 – Ilustração sobre o posicionamento de um objeto em três posições diferentes e a tendência do movimento. (Fonte: Autor).

Após o primeiro valor ter sido armazenado no vetor, o objeto se locomove em uma das direções dependendo do ponto inicial. À medida que o objeto se locomove, novos valores de tensão são registrados pelo sensor de distância e passados, digitalmente, para o microcontrolador. O código compilado nele, como explicado anteriormente, realiza testes e armazena estes novos valores no vetor.

Para esse projeto e código, é adotada a opção de armazenar apenas cinco medidas digitais de tensão no vetor. E é a partir da análise dessas cinco tensões que se decide qual é de fato a tendência do movimento e o quê deve ser enviado ao microcontrolador para que ocorra o processo avançar ou retroceder *slides.*

4.4.3 – Analise dos Valores do Vetor

Após o objeto ter se locomovido pelo campo de ação, que é o espaço entre o sensor e a barreira de segurança, e os valores das tensões de cinco distâncias diferentes terem sido gravadas no vetor, inicia-se o processo de análise da tendência para saber se deve avançar ou voltar *slides.*

A seguir, é apresentada figura 4.14, na qual, se ilustra um trecho do código compilado no microcontrolador, que é responsável por analisar os valores armazenados no vetor para saber se o movimento feito pelo objeto implica em avançar ou voltar *slides*.

```
medidor=0;//Seta 0 a variavel medidot
ct=0;//Seta 0 a variavel ct
for(ct=1;ct < 6 ; ct++)//Estrutura de Repetiçao for enquanto ct
{
  if(vetor[medidor] >= vetor[ct])//Estrutura de condiçao se val
  {
    flag_ava= flag_ava + 1;//incrementa ct_2 por 1
  }
  else //senao
  {
    flag_rec=flag_rec+1;//incrementa ct_2 por 1
  }
  if(vetor[ct] >= vetor[ct_2])//Estrutura de condiçao se valor
  {
    flag_ava= flag_ava + 1;//incrementa ct_2 por 1
  }
  else//senao
  {
    flag_rec=flag_rec+1;//incrementa ct_2 por 1
  }
  ct_2=ct_2+1;//incrementa ct_2 por 1
}
```

Figura 4.14 – Ilustração da *IDE* com o código responsável por analisar os valores armazenados para tendência do movimento. (Fonte: Autor).

Para verificar se de fato o movimento do objeto implica em avançar ou voltar *slides*, a lógica do código programado trabalha com *flags*. Essas *flags* têm o objetivo de armazenar um valor unitário a partir da comparação dos valores armazenados no vetor.

Primeiramente, são utilizadas duas variáveis com a função de *flags*. A primeira é a variável *flag_ava*, que tem a função de armazenar os valores unitários indicando que a tendência do movimento começa com o marco inicial mais próximo do sensor e que vai se afastando em direção à barreira de segurança. A outra variável é a *flag_rec* que tem a função de armazenar os valores unitários indicando que a tendência do movimento começa com o marco inicial mais próximo da barreira de segurança e que vai se afastando em direção ao sensor. Estas variáveis servem para validar uma condição presente no código que tem a função de enviar ao microcontrolador o comando correto para avançar ou voltar *slides*. É por meio do acréscimo desses valores unitários que a comparação entre as *flags* permite ao código enviar o comando certo de avançar ou retroceder *slides*.

O código citado anteriormente funciona a partir das comparações dos valores do vetor entre si. Primeiramente, são utilizadas três variáveis, *medidor, ct e ct2* com a função de contadores. Esses contadores permitem que os valores armazenados neles sejam utilizados para extrair os valores armazenados no vetor, como demonstrado na figura 4.14

Após os contadores assumirem seus valores iniciais, o código entra em uma estrutura de repetição *for(ct = 0; ct < 6; ct++)* que serve para analisar várias vezes os valores armazenados no vetor.

A primeira análise consiste em comparar o valor armazenado no primeiro campo do vetor com o valor do segundo campo. Se o valor do primeiro campo do vetor for maior que o valor do segundo campo, visto no trecho *if(vetor[medidor] >= vetor[ct])*, é um indício de que o objeto começou o movimento mais próximo do sensor e foi se afastando. Depois de validado isso, a variável *flag_ava* recebe um valor unitário evidenciando que tendência do movimento é para avançar *slides*. Caso essa condição não seja confirmada, a *flag_rec* recebe esse valor unitário e o processo de análise da outra condição é confirmado.

A segunda condição, *if(vetor[ct] >= vetor[ct2])*, tem o papel de verificar se o primeiro campo do vetor maior que o terceiro. Tendo essa condição confirmada, a variável *flag_ava* é acrescida de mais um valor unitário confirmando nesta primeira validação que o objeto começou o movimento mais próximo do sensor de distância e foi se afastando em direção a barreira de segurança. Caso a medida armazenada no terceiro seja menor que a do primeiro, a condição é invalidada fazendo com que a variável *flag_rec* receba um valor unitário.

Para evitar que código, após essa primeira análise, emitisse ao microcontrolador um sinal errado são feitas também novas validações com os mesmos valores. No entanto, ao final da passagem pela estrutura de repetição *for* ocorre uma mudança nos valores das variáveis, fazendo com que na nova passagem pela estrutura de repetição, ocorra a permutação dos valores do vetor e novas análises sejam feitas.

4.4.4 – Envio do comando ao Computador

Após os processos de armazenagem de tensões calculadas pelo sensor de distância no vetor e o processo de análise e validação desses valores a fim de concretizar a tendência do movimento inicia-se a parte final do processo. Nesta parte o microcontrolador utiliza os valores armazenados nas *flags* para concluir qual foi o movimento do objeto do campo de ação entre o sensor de distância e a barreira de segurança.

O comando, *Serial.println("#S|SENDK|[PID&keystrokes to send]#")*, serve para que o microcontrolador envie um sinal para computador de modo que o *software Gobetwino* intercepte esse comando, interprete-o e execute o comando necessário. A figura 4.15 mostra um trecho do código que utiliza apenas uma estrutura de condição para identificar qual comando deverá ser enviado para que apresentação de avance ou retroceda *slides*.

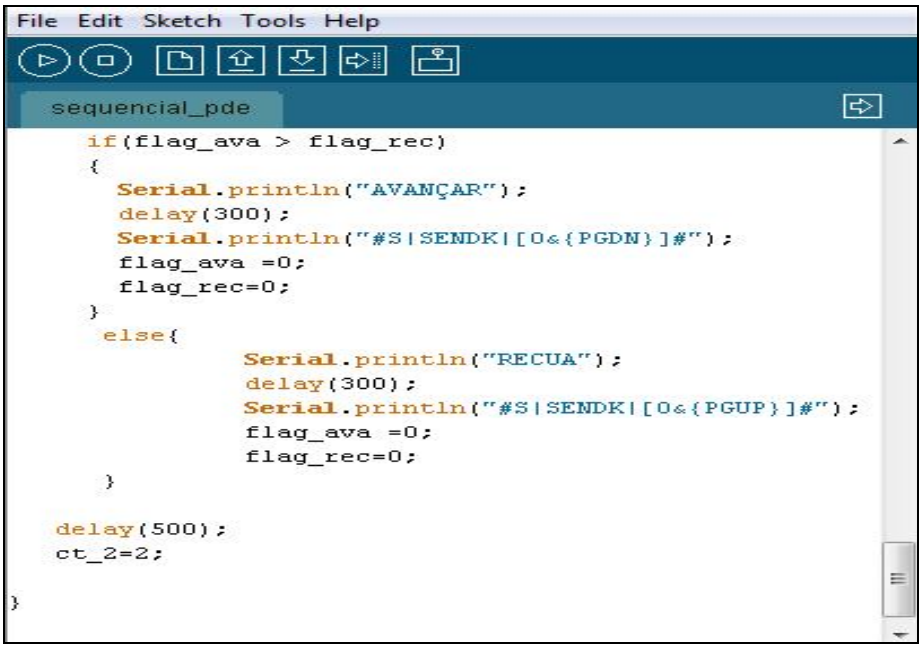

Figura 4.15 – Ilustração da *IDE* sobre o trecho do código na qual o comando é enviado ao computador para avançar ou retroceder *slides*. (Fonte: Autor)

No código citado anteriormente, primeiramente é feita a avaliação das duas *flags*, *flag_ava* e *flag_rec*. Como na etapa anterior, é utilizado o acréscimo de um valor unitário em cada verificação, à medida que essa avaliação prossegue, as *flags* vão aumentando de valor ou estagnam. Esta avaliação permite confirmar a tendência do movimento.

Na condição *if(flag_ava > flag_rec)* é averiguado se a análise da tendência do movimento do objeto é voltada para a aproximação do sensor de distância ou voltada para se afastar do sensor. Caso a condição seja verdadeira, o comando, *Serial.println(“#S|SENDK|[0&{PGDN}]#”)*, deve ser enviado do microcontrolador para o computador para que o *software Gobewino* intercepte e interprete este comando, primeiramente, e envie para o computador um outro comando, como o *Page up*. Este novo comando tem a mesma função da tecla *Page up* do teclado convencional. Caso a condição não seja verdadeira, o microcontrolador deve enviar ao computador o comando *Serial.println(“#S|SENDK|[0&{PGUP}]#”)* que contem a função de *Page down*, que por sua vez possui a mesma função da tecla *Page down* do teclado convencional.

As teclas *Page up* e *Page down* quando pressionados por um usuário durante uma apresentação em *Datashow* permitem a mudança do *slide* atual. Quando a tecla *Page down* é pressionada no teclado, o *slide* subseqüente ao atual passa a ser o atual, ou seja, ocorreu o avanço de *slides*. Caso a tecla *Page up* tenha sido pressionada, o *slide* anterior ao atual passa a ser o *slide* atual, ou seja, voltou o *slide*.

Após este três processos, o código se mantém em um *loop* infinito retornando a primeira fase à espera de um novo movimento de um objeto. Esse processo de *loop* se deve à estrutura de repetição *void loop()*.

CAPÍTULO 5 – APLICAÇÃO DO MODELO PROPOSTO

5.1 – Aplicação do Protótipo Proposto

A aplicação do protótipo é voltada para, basicamente, dois ambientes. Estes ambientes são tanto os ambientes acadêmicos quanto ambientes empresariais. No entanto, é mostrado, por meio dos resultados e custos, que este protótipo se adequa melhor em um ambiente acadêmico. O projeto é considerado válido se ao final de todo o processo, considerando desde a calibração até o comando de avançar ou voltar *slides*, se o dispositivo criado realizar de fato essa tarefa.

5.2 – Descrição da Aplicação do Protótipo

Inicialmente, um objeto, ou a própria mão do apresentador, interrompe o campo de ação do sensor e a partir de um movimento e certa direção, é dado inicio ao processo de calibração, análise de valores obtidos pelo sensor e a definição do comando correto para avançar ou voltar *slides*.

Após a definição do comando de avançar ou voltar *slides*, o microcontrolador envia um código ao computador, que ao ser interceptado pelo *software Gobetwino* que fica escutando a porta *serial*, fica responsável por transformar este comando em um atalho de uma tecla de teclado.

5.3 – Resultados do Projeto

5.3.1 – Resultados Esperados

Nos quesitos de *software e hardware*, é esperado que ao final do movimento, de fato, ocorra uma mudança nos *slides*, tanto para avançar quanto para retroceder. O processo inicial de o sistema ficar em aguardo, a espera de um movimento sem que haja qualquer alteração nos slides da apresentação, é necessário e esperado. Assim como é esperado que o dispositivo funcione no formato *plug-and-play*.

Também é esperado que o movimento do objeto ou da mão do apresentador não seja sempre no mesmo lugar, permitindo um uso mais dinâmico do dispositivo.

Especificamente, em nível de *software*, é esperado que o *software Gobetwino* consiga sempre assimilar os códigos passados pelo microcontrolador, e transmita o comando certo para o computador. É esperado também que a apresentação, após ter sido iniciada pelo *Gobetwino*, possa sofrer alterações pelo autor.

Durante o processo de análise de proposta de projeto, foi cogitada a criação de um *driver* para realizar função de interpretar os códigos do microcontrolador e passar para o computador o comando correto para mudar de *slides*. Durante o estudo de métodos para criação de um *driver*, é encontrado o *software Gobetwino* que realiza, de certa forma, a função do *driver* necessário. Ou seja, é esperado que o *software Gobetwino* exerça a função de um *driver* para intermediar a conversa entre o microcontrolador e o computador.

Quanto ao *hardware*, é esperado que o sensor infravermelho de distância capture as várias distâncias exatas do objeto em relação ao sensor e repassasse as informações para o microcontrolador. É esperado que o microcontrolador, por meio do código compilado nele, consiga assimilar os valores recebidos do sensor, analise-os e os transmitam para o computador.

5.3.2 – Resultados Obtidos

O protótipo do projeto foi capaz de avançar ou voltar *slides* por meio do movimento da mão ou do objeto em frente ao sensor de distância. Os dados obtidos pelo sensor infravermelho de distância *SHARP GP2D120* e passados para o microcontrolador *ATmega328P* para análise e validação do mesmo foi satisfatório. No entanto, ocorreu a necessidade de se mudar o sensor de distância do *SHARP GP2Y0A02YK* para *SHARP GP2D120* após uma série de testes com o código final. O *software Gobetwino* exerceu sua função como esperado no código final, no entanto, em alguns testes sua leitura e transmissão não foram satisfatórias.

Apesar de os sensores infravermelho de distância *SHARP* terem resolvido a questão da leitura da distância, os valores das distâncias não foram, de fato, utilizadas. Como esperado, o valor a ser trabalhado no código era para ser uma distância, no entanto, durante testes foi possível verificar que o valor de saída do sensor era o valor de sua tensão digitalizada. Como em diferentes distâncias, as tensões variavam, bastava observar o comportamento delas em diferentes distâncias para projetar o código completo do microcontrolador. Com isso, não houve a necessidade de utilizar a distância exata do objeto como pretendido e sim a tensão de saída obtida pelo sensor.

Ainda em relação ao *hardware,* o sensor *SHARP GP2Y0A02YK* foi o primeiro sensor utilizado. Este sensor possuía uma faixa de atuação variando de 20 cm a 150 cm. Para o projeto proposto, é necessário se trabalhar com no máximo 20 cm. Por esta razão, em alguns testes, os valores das tensões que deveriam variar em diferentes distâncias, não estavam variando, pois os movimentos estavam sendo trabalhados a menos de 20 cm. No entanto, ao se afastar o suficiente do sensor para o range programado para ele, ocorriam as falhas na leitura. Para sanar estes problemas iniciais, foi proposto mudar o sensor do *SHARP GP2Y0A02YK para SHARP GP2D120*. Este segundo sensor, que trabalha no intervalo de 4 cm a 30 cm, serviu melhor as necessidades do projeto, realizando medidas mais exatas, que eram necessárias para perfeito funcionamento do projeto.

O *software Gobetwino* é um *software* gratuito. No entanto, ele não é considerado um *software* livre por não possuir seu código aberto. Tendo esse agravante de não poder trabalhar no do seu código fonte, o *Gobetwino* possui apenas as funções prontas criadas pelo desenvolvedor desse *software*. Nos testes realizados, ocorreram alguns erros de leitura, por parte do *software*, estes erros ocorreram devido aos comandos enviados do microcontrolador ao computador e interceptados pelo *Gobetwino*.

O *Gobetwino* possui uma interface gráfica que mostra a execução do código pelo microcontrolador, na qual ele fica a espera de comandos conhecidos para executá-los. Apesar de em alguns casos, ele ler o comando certo e não executá-lo, foi possível verificar que a falta de um *delay* maior no código do microcontrolador entre o comando recebido e a necessidade de enviá-lo ao computador, compreendia o erro. Bastou colocar um *delay* entre algumas linhas do código para que ocorresse uma melhora na leitura do *software*.

Existe um ponto que não foi satisfeito com relação à edição da apresentação. Durante o processo de testes, era esperado que quando a apresentação é iniciada, é possível realizar algumas alterações, por exemplo, é possível colocar os *slides* em tela cheia para a apresentação, o que a principio é satisfatório. No entanto, após iniciar o processo de avançar ou voltar *slides*, não possível realizar alterações na apresentação, pois a relação de controle do *software Gobetwino* sobre esta, é perdido por limitações do *software*. Para que seja possível trabalhar com o dispositivo novamente, é necessário realizar as alterações desejadas na apresentação já aberta, salvar essa apresentação, fechá-la e pressionar novamente o botão físico de *reset* no microcontrolador, para que o código, anteriormente compilado, abra uma nova apresentação já alterada e o usuário possa trabalhar novamente.

Originalmente, no código do microcontrolador, foram analizadas três distâncias para o cálculo da tendência do movimento. Após uma serie de testes e análise dos valores obtidos pelo sensor, foi constatado que devido a ruídos, alguns valores de tensão, durante o movimento, oscilavam causando um desacordo com a teoria de que à medida em que a distância aumenta a tensão diminui. Esta oscilação foi armazenada e utilizada para cálculo da tendência do movimento e por conta dessa oscilação a tendência foi calculada de forma errada impedindo que o *slide* mudasse de forma correta. Com o aumento de cinco distâncias para cálculo da tendência, as oscilações são comparadas com mais valores aumentado a precisão e o acerto da tendência de movimento.

5.3.3 – Comparação entre Resultados Esperados e Obtidos

Tanto em *software* quanto *hardware*, os resultados foram satisfatórios, pois o protótipo, criado com o objetivo de avançar ou retroceder *slides*, atendeu às expectativas propostas. Apesar da necessidade de trocar os sensores infravermelho de distância, e de não precisar criar um *driver*, já que foi possível encontrar um *software* que, mesmo com as limitações, possuía as funções necessárias apesar das limitações.

No sensor, os resultados esperados e os obtidos, apesar de não ter sido utilizada a distância exata entre o objeto e o sensor e sim o valor digitalizado da tensão de saída do sensor ao encontrar um objeto, foram os mesmos, pois no código criado no microcontrolador, apenas a variação dos valores em diferentes distâncias permitiam que fossem feitas análises de tendência do movimento, possibilitando o cálculo da direção do movimento e, consequentemente, o comando correto para avançar ou retroceder *slides*.

No *software* de comunicação entre o microcontrolador e o computador, a escolha do uso do *Gobewino*, ao invés da criação do *driver*, permitiu que o resultado esperado e o obtido fossem os mesmos. No entanto, não era esperado, mas foi necessário abrir o aplicativo *Gobewino.exe* e pressionar o botão de *reset* no microcontrolador para que houvesse a conexão do *Gobetwino* com o microcontrolador, além de fechar a apresentação e pressionar novamente o botão de *reset* caso alguma alteração física fosse feita na apresentação. Mesmo com as limitações e de ações não esperadas, por parte do *Gobetwino*, os resultados foram satisfatórios.

5.4 – Custos do Projeto

Apesar de possuir características de um projeto acadêmico, a busca para diminuir o custo do projeto é algo necessário em todos os ambientes. No entanto, devido dispositivos que queimaram e falta de planejamento, não foi possível construir um protótipo que fosse mais barato do que aqueles existentes no mercado, considerando a produção de apenas um dispositivo. Os custos do projeto foram basicamente de *hardware* e conectores, tendo em vista que o *software* utilizado, o *Gobetwino*, é o um *software* gratuito.

O orçamento e custo do microcontrolador, o *Arduino ATmega328P Duemilanove*, foi o mais alto do projeto. Para esse projeto foi comprado o *Kit Arduino Duemilanove – iniciantes*, figura 5.1, que custa em torno de R$ 218,00. Apenas o *Arduino Duemilanove com ATmega328* custa R$ 108,00. A aquisição de outro *Arduino Duemilanove* foi necessária, pois o *Arduino* que veio no *kit* queimou durante alguns testes.

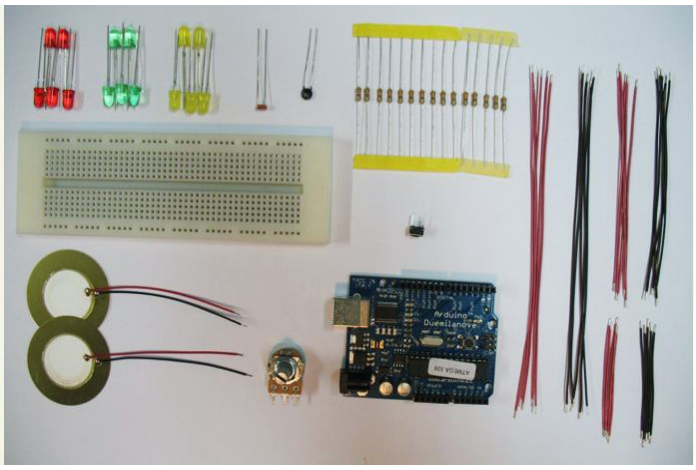

Figura 5.1 – Ilustração do kit iniciante do *Arduino Duemilanove*. (Fonte: http://www.multilogica-shop.com/Kit-Arduino-iniciantes).

O custo para os sensores infravermelhos de distância *SHARP GP2Y0A02YK* foi de R$ 83,00 e para o sensor *SHARP GP2D120* de R$ 82,00. Na figura 5.2 estão os sensores respectivamente.

Figura 5.2 – Figura ilustrativa sobre o sobre os sensores infravermelhos de distancia. Primeiramente o sensor *SHARP GP2Y0A02YK* e segundo *SHARP GP2D120.* (Fonte: www.multilogica-shop.com/catalogo/sensores/sensores-de-dist%C3%A2ncia)

Para a conexão entre o microcontrolador e o sensor infravermelho de distância, é necessário um conector *JST* de três pinos, figura 5.3, que custa em média R$ 6,00

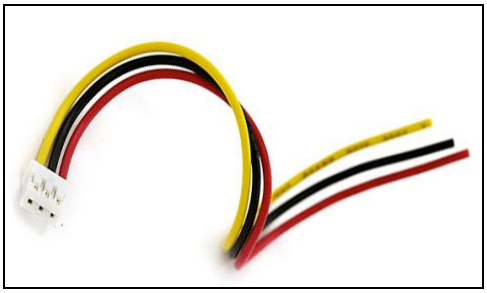

Figura 5.3 – Figura ilustrativa sobre o sobre o cabo *JST* de três pinos para o sensor. (Fonte: http://www.multilogica-shop.com/cabo-para-sensor-de-infravermelho)

No quadro 5.1, é comparado o preço do projeto juntamente com um produto já existente no mercado.

Quadro 5.1 – Quadro referente ao custo do projeto juntamente com a comparação entre o projeto e um produto do mercado. (Fonte: Autor).

	Descrição do Item	**Valor Unitário (R$)**	**Quantidade**
PROJETO DESENVOLVIDO	Microcontrolador Arduino com ATmega328p	108,00	1
	Sensor Infravermelho de Distância SHARP GP2D120	82,00	1
	Sensor Infravermelho de Distância SHARP GP2Y0A02YK	83,00	1
	Gobetwino	0,00	1
	Kit Arduino	218,00	1
	Conectores JST de três pinos	6,00	5
	TOTAL DO PROJETO	**521,00**	
PRODUTO DO MERCADO	*Logitech Professional Presenter R800*	288,00	1
	TOTAL DO PROJETO	**288,00**	

CAPÍTULO 6 – CONCLUSÃO

6.1 – Conclusões Acerca do Projeto

A experiência e oportunidade de desenvolver esse projeto foram de extrema importância, pois permitiu ao autor agregar conhecimentos em relação a certas práticas da engenharia. Essas práticas envolvem todo o processo de construção do projeto como o processo de planejamento do tema, a pesquisa dos materiais necessários tanto eletrônicos quanto teóricos e a necessidade de buscar saídas e opções para cada dificuldade. Para desenvolver todo o processo também foram necessárias varias horas de programação e testes, além do longo processo de escrita da monografia.

A idéia inicial do projeto foi criar um dispositivo eletrônico que auxiliasse na apresentação de *datashow* alterando os slides a partir do movimento da mão em uma determinada direção. Foi proposto o protótipo de um dispositivo para realização da função de passar *slides* baseado em um microcontrolador e um sensor infravermelho de distância, sem a necessidade de o apresentador pressionar qualquer tecla em um controle remoto ou de ter que acessar o computador, repositório da apresentação, para mudar os *slides*.

Basicamente, o dispositivo consiste na captação do movimento da mão ou de um objeto em frente a um sensor. A partir desse movimento, um código criado no microcontrolador interpreta os valores obtidos pelo sensor, analisa a tendência do movimento e repassa para o computador o comando para avançar ou retroceder *slides*.

Durante o projeto ocorreram alguns problemas com relação a leitura do sensor, e de conexão entre o microcontrolador e o computador para passar o comando correto de avançar ou retroceder *slides*. Após compilar o código no microcontrolador e utilizar o primeiro sensor *SHARP*, mencionado em capítulos anteriores, foram encontrados erros de leitura e inconsistência, por se tratar de um sensor que trabalha com um campo de ação de 20 a 150 cm e o trabalho necessitar de apenas 3 a 20 cm. Para solucionar esse problema, foi proposto alterar o sensor *SHARP* por um sensor da mesma marca mas com o campo de ação reduzido, trabalhando com intervalo de 4 a 30 cm. Em nível de conexão entre o microcontrolador e o

computador foi cogitada, durante o planejamento e aprovação da proposta de projeto, a criação de um *driver* que faria a função de interpretar os valores fornecidos pelo microcontrolador e passá-los para o computador a fim de avançar ou voltar *slides*. Durante algumas pesquisas foi encontrado o *software Gobetwino*, um software gratuito e pronto que possui funções que se assemelham a um *driver* facilitando a comunicação do microcontrolador com computador.

Os pontos que não foram satisfatórios são os relativos à questão do *Gobewino* necessitar que se execute o seu aplicativo e pressione o botão de *reset* no microcontrolador para que ele passe a funcionar em conjunto com o microcontrolador. Outro fato é que qualquer alteração necessária na apresentação em slides deve ser feita, salva e fechada. Após isso, é necessário pressionar o botão de *reset* no microcontrolador, novamente, para que, novamente, o *Gobetwino* abra a apresentação em *slides*. Esse processo é necessário pois o *Gobetwino* perde referência com a apresentação me slides caso alguma alteração ocorra durante a apresentação.

Foi possível concluir nesse projeto que em relação ao custo, este projeto não é favorável à relação custo/benefício, pois o valor total do projeto foi superior ao um produto considerado caro dentre os existentes no mercado. No entanto, o alto custo se deve ao fato de não se tratar de uma produção em massa.

Ao final do projeto é possível verificar que, de fato, o objetivo traçado e planejado foi cumprido e o dispositivo funcionou de forma satisfatória.

6.2 – Sugestões de Futuros Projetos

Como evidenciado no dia a dia, e nesse projeto, a necessidade de dispositivos que auxiliem na apresentação de slides buscando dinamismo e agilidade é muito alta. Em relação a esse projeto e utilizando o mesmo conceito, é possível aprimorar as questões de conexão entre o microcontrolador e computador, que é realizada via USB, e também a substituição do *software Gobetwino* por um *driver* ou por um *software* que pode ser desenvolvido utilizando as bibliotecas de conexão do *Arduino* com outras plataformas de desenvolvimento e programação que ficam alocadas no próprio computador, eliminando assim a necessidade do *Gobetwino*.

Como esse projeto busca melhorar o dinamismo nas apresentações, a substituição do cabo USB por uma conexão sem fio, permitiria ao apresentador alocar o suporte do sistema em qualquer lugar que lhe fosse conveniente, sem a necessidade de se preocupar com o tamanho do cabo.

Com relação à substituição do *Gobetwino*, seria referente a questão dos dispositivos *plug-and-play*. Esse protótipo não funciona dessa maneira, tendo a necessidade de pressionar o botão *reset* do microcontrolador para que haja conexão do deste com o computador por intermédio do *Gobetwino.*

REFERÊNCIAS

ANDERSON, Henrik – Position Sensitive Detectors - Device Technology and Applications in Spectroscopy Disponível em: <miun.diva-portal.org/smash/get/diva2:1939/FULLTEXT01> Acesso em: 24 jun.2011

ATMEL. ATmega328P Preliminary Summary. San Jose: 2010. 1,2. Disponível em: http://www.atmel.com/dyn/resources/prod_documents/doc8161.pdf. Acesso em: 17 abr. 2011.

AXELSON, Jan. USB Complete: The Developer‘s Guide. Madison: Lakeview Research, 2009. 506 p.

DATASHEET, GP2D120, Distance Measuring Sensor, Sharp Microelectronics of the Americas. Disponível em: <http://www.sharpsma.com/optoelectronics/sensors/distance-measuring-sensors/GP2D120>. Acesso em: 3 jun. 2011.

DEKCEL, RF Wireless Laser Pointer Presentation Device with Page Up/Down Function (Black) – 2011. Disponível em: < http://www.dekcell.com/product-cpa_1384_black-3518.html >. Acesso em: 3 jun. 2011.

FRAUENFELDER, Mark – Make: Technology on Your Time. Volume 5. O'Reilly Media, Inc., 2006 - 224 páginas

GOBETWINO, *Gobetwino – 2011*. Disponível em: http://www.mikmo.dk/gobetwino.html. Acesso em: 18 abr. 2011

LOGICTECH, Logitech Professional Presenter R800 – 2011. Disponível em: <http://www.logitech.com/en-us/mice-pointers/presentation-remote/devices/5873>. Acesso em: 3 jun. 2011

MONTEIRO, Mário A. Introdução à Organização de computadores. Rio de Janeiro: Editora LTC, 2002. 4 ª edição. 215 p. (qnt de paginas)

NICOLOSI, Denys. Microcontrolador 8051 Detalhado. 4ª edição. São Paulo: Editora Érica, 2004. 227 p.

REAS, Casey; FRY, Ben. Getting Started with Processing. 1ª edição. Sebastopol: O'Reilly Media, 2010. 208 p.

RESET GOBETWINO, Using Gobetwino to Control Windows through Arduino – Electronics – 2010 – Disponível em:

<https://sites.google.com/a/divinechildhighschool.org/electronics/Home/Arduino-Lessons/using-gobetwino-to-control-windows-through-arduino>. Acesso em: 3 jun. 2011>.

SCRIMGER, Rob; LASALLE, Paul; PARIHAR, Mridula. TCP/IP - A BIBLIA. Editora Gulf Professional Publishing, 269 p.(qnt de paginas)

SENSOR, Relatório sobre a calibração do sensor de distância GP2D02 da SHARP. Disponível em:

<http://wiki.dcc.ufba.br/pub/Mecateam/DownloadsMecateam/relatriodog2p02.pdf>. Acesso em: 3 jun.11

SENSORS TRIANGULATION – Triangulation Sensors – 2011. Disponível em: http://archives.sensorsmag.com/articles/0598/tri0598/. Acesso em: 24 jun. 2011.

THOMAZINI, Daniel; ALBUQUERQUE, Pedro Urbano Braga. Sensores Industriais, Fundamentos e Aplicações. São Paulo: Editora Érica, 2005. 1 ª edição. 220 p.

WIRING, Wiring. Disponível em: < http://wiring.org.co/reference/ >. Acesso em: 23 jun. 2011.

APÊNDICE A – Código armazenado no microcontrolador

```
int pin = 0; // pin de saida do sensor
int maior=0; // inteiro para calibragem
int primeiro=0; //inteiro para calibragem
int ct=0; //inteiro para calibragem
int vetor[10]; // vetor que armazena os informaçoes do sensor a fim de traçar a direcao da mao
int medidor=0; // caminha pelo vetor
int medidor_2=0;// armazena medidor para uso no for
int meio=0; //intermediar os ifs
int primo=0; //primeiro valor
int maior_seg=0;//maior + maior * 0,1
int flag_ava =0;//flag para saber se ao final avanca slides
int flag_rec=0;//flag para saber se ao final recua slides
int ct_2=2;//contador para valores do vetor


void setup(){
  Serial.begin(9600);//Inicia Serial
  delay(1000);//Atraso de 1 segundo
  pinMode(13, OUTPUT);//Seta pino treze para receber comando de "write"
  delay(500);//Atraso de 0,5 segundos
  //While responsavel por guardar a maior distancia a fim de calibrar o sensor(inicio)
  while(ct!=500){ //Estrutura de repetiçao enquanto diferente de 500
    primeiro = analogRead(pin);//Aloca valor lido pelo sensor no pino 0 em primeiro
```

```
    if (primeiro > maior)//Estrutura de condiçao se primeiro maior que maior
    {
      maior=primeiro;//Aloca valor de primeiro em maior
    }
    ct=ct+1;//Ct é incrementado por 1 a cada passada pelo while
  }
  maior_seg=(maior *  1.1); //Aloca em maior_seg o valor de maior * 1,1
  //While responsavel por guardar a maior distancia a fim de calibrar o sensor(fim)
  delay(1000);//Atraso de 1 segundo
  digitalWrite(13, HIGH); //força o pino 13 a acender
  delay(1000);//Atraso de 1 segundo
  digitalWrite(13, LOW);  //força o pino 13 a apagar
  delay(3000);//Atraso de 3 segundos
  Serial.println("#S|ABREPPT|[]#");//Abre PPT
}

void loop(){
   digitalWrite(13, HIGH);//força o pino 13 a acender
   primo=analogRead(pin);//Aloca valor lido pelo sensor no pino 0 em primo
   delay(100);//Atraso de 0,5 segundos
   if (primo < maior_seg)//Estrutura de condiçao se primo menor que maior_seg
   {
     Serial.println("Em espera");//Imprime "Em Espera"
     meio=analogRead(pin);//Aloca valor lido pelo sensor no pino 0 em meio
     if(meio < maior_seg)//Estrutura de condiçao se meio menor que maior_seg
     {
```

```
    while(1)//Estrutura de repetição infinita ate quebra "break"
    {
      meio=analogRead(pin);//Aloca valor lido pelo sensor no pino 0 em meio
      delay(500);//Atraso de 0,5 segundos
      if(meio > maior_seg)//Estrutura de condiçao se maior maior que maior_seg
      {
        digitalWrite(13, LOW);//força o pino 13 a apagar
        Serial.println("****************************************");//validador
        vetor[medidor] = meio;//Aloca valor meio no vetor
        medidor=medidor+1;//incrementa medidor por 1
        Serial.println(meio);
        Serial.println(medidor);//validador
        if(medidor == 5)//Estrutura de condiçao se medidor = 3
        {
          digitalWrite(13, HIGH);//força o pino 13 a acender
          break;//Sai do While
        }
      }
    }
  }
}
Serial.println("saiu"); //validador
delay(500);//Atraso de 0,5 segundos
Serial.println("valores do vetor"); //validador
medidor=0;//Seta 0 a variavel medidot
ct=0;//Seta 0 a variavel ct
```

```
  for(ct=1;ct < 6 ; ct++)//Estrutura de Repetiçao for enquanto ct menor que 4
  {
    if(vetor[medidor] >= vetor[ct])//Estrutura de condiçao se valor de vetor em um espaço de memoria é  maior do que em outro
    {
      flag_ava= flag_ava + 1;//incrementa ct_2 por 1
    }
    else //senao
    {
      flag_rec=flag_rec+1;//incrementa ct_2 por 1
    }
    if(vetor[ct] >= vetor[ct_2])//Estrutura de condiçao se valor de vetor em um espaço de memoria é  maior do que em outro
    {
      flag_ava= flag_ava + 1;//incrementa ct_2 por 1
    }
    else//senao
    {
      flag_rec=flag_rec+1;//incrementa ct_2 por 1
    }
    ct_2=ct_2+1;//incrementa ct_2 por 1
  }
  delay(100);//Atraso de 0,1 segundos
  if(flag_ava > flag_rec)//Estrutura de condiçao se flag_ava maior que flag_rec
  {
     Serial.println("AVANÇAR");//Imprime "avanca"
```

```
      Serial.println(flag_ava);
      delay(300);//Atraso de 0,3 segundos
      Serial.println("#S|SENDK|[0&{PGDN}]#");//comando para avancar slides
      flag_ava =0;//Seta 0 a variavel flag_ava
      flag_rec=0;//Seta 0 a variavel flag_rec
   }
   else{//senao
            Serial.println("RECUA");//Imprime "RECUA"
            Serial.println(flag_rec);
            delay(100);//Atraso de 0,3 segundos
            Serial.println("#S|SENDK|[0&{PGUP}]#");//comando para voltar slides
            flag_ava =0;//Seta 0 a variavel flag_ava
            flag_rec=0;//Seta 0 a variavel flag_rec
   }
   delay(500);//Atraso de 0,5 segundos
   ct_2=2;//Seta 2 a variavel ct_2

}
```

APÊNDICE B – Código armazenado no microcontrolador para a banca

```
int pin = 0; // pin de saída do sensor
int maior=0; // inteiro para calibragem
int primeiro=0; //inteiro para calibragem
int ct=0; //inteiro para calibragem
int vetor[10]; // vetor que armazena as informações do sensor a fim de traçar a direção da mão
int medidor=0; // caminha pelo vetor
int medidor_2=0;// armazena medidor para uso no for
int meio=0; //intermediar os ifs
int primo=0; //primeiro valor
int maior_seg=0;//maior + maior * 0,1
int flag_ava =0;//flag para saber se ao final avança slides
int flag_rec=0;//flag para saber se ao final recua slides
int ct_2=2;//contador para valores do vetor


void setup(){
  Serial.begin(9600);//Inicia Serial
  delay(1000);//Atraso de 1 segundo
  pinMode(13, OUTPUT);//Seta pino treze para receber comando de "write"
  delay(500);//Atraso de 0,5 segundos
  //While responsável por guardar a maior distancia a fim de calibrar o sensor(inicio)
  while(ct!=500){ //Estrutura de repetição enquanto diferente de 500
```

```
    primeiro = analogRead(pin);//Aloca valor lido pelo sensor no pino 0 em primeiro
    if (primeiro > maior)//Estrutura de condição se primeiro maior que maior
    {
      maior=primeiro;//Aloca valor de primeiro em maior
    }
    ct=ct+1;//Ct é incrementado por 1 a cada passada pelo while
  }
  maior_seg=(maior *  1.1); //Aloca em maior_seg o valor de maior * 1,1
  //While responsável por guardar a maior distancia a fim de calibrar o sensor(fim)
  delay(1000);//Atraso de 1 segundo
  digitalWrite(13, HIGH); //força o pino 13 a acender
  delay(1000);//Atraso de 1 segundo
  digitalWrite(13, LOW);  //força o pino 13 a apagar
  delay(1000);//Atraso de 3 segundos
  Serial.println("#S|ABREPPT|[]#");//Abre PPT
}

void loop(){
   digitalWrite(13, HIGH);//força o pino 13 a acender
   primo=analogRead(pin);//Aloca valor lido pelo sensor no pino 0 em primo
   delay(100);//Atraso de 0,5 segundos
   if (primo < maior_seg)//Estrutura de condição se primo menor que maior_seg
   {
     Serial.println("Em espera");//Imprime "Em Espera"
     meio=analogRead(pin);//Aloca valor lido pelo sensor no pino 0 em meio
     if(meio < maior_seg)//Estrutura de condição se meio menor que maior_seg
```

```
  {
    while(1)//Estrutura de repetição infinita ate quebra "break"
    {
      meio=analogRead(pin);//Aloca valor lido pelo sensor no pino 0 em meio
      delay(200);//Atraso de 0,5 segundos
      if(meio > maior_seg)//Estrutura de condição se maior maior que maior_seg
      {
        digitalWrite(13, LOW);//força o pino 13 a apagar
        //Serial.println("****************************************");//validador
        vetor[medidor] = meio;//Aloca valor meio no vetor
        medidor=medidor+1;//incrementa medidor por 1
        //Serial.println(meio);
        //Serial.println(medidor);//validador
        if(medidor == 5)//Estrutura de condição se medidor = 3
        {
          digitalWrite(13, HIGH);//força o pino 13 a acender
          break;//Sai do While
        }
      }
    }
  }
}
//Serial.println("saiu"); //validador
delay(300);//Atraso de 0,5 segundos
//Serial.println("valores do vetor"); //validador
medidor=0;//Seta 0 a variável medidor
```

```
ct=0;//Seta 0 a variável ct
for(ct=1;ct < 6 ; ct++)//Estrutura de Repetição for enquanto ct menor que 4
{
 if(vetor[medidor] >= vetor[ct])//Estrutura de condição se valor de vetor em um espaço de memória é  maior do que em outro
 {
  flag_ava= flag_ava + 1;//incrementa flag_ava por 1
 }
 else //senão
 {
  flag_rec=flag_rec+1;//incrementa flag_rec por 1
 }
 if(vetor[ct] >= vetor[ct_2])//Estrutura de condição se valor de vetor em um espaço de memória é  maior do que em outro
 {
  flag_ava= flag_ava + 1;//incrementa flag_ava por 1
 }
 else//senão
 {
  flag_rec=flag_rec+1;//incrementa flag_rec por 1
 }
 ct_2=ct_2+1;//incrementa ct_2 por 1
}
delay(100);//Atraso de 0,1 segundos
if(flag_ava > flag_rec)//Estrutura de condição se flag_ava maior que flag_rec
{
```

```
    //Serial.println("AVANÇAR");//Imprime "avança"
    //Serial.println(flag_ava);
    delay(100);//Atraso de 0,3 segundos
    Serial.println("#S|SENDK|[0&{PGDN}]#");//comando para avançar slides
    flag_ava =0;//Seta 0 a variável flag_ava
    flag_rec=0;//Seta 0 a variável flag_rec
  }
  else{//senao
        //Serial.println("RECUA");//Imprime "RECUA"
        //Serial.println(flag_rec);
        delay(100);//Atraso de 0,3 segundos
        Serial.println("#S|SENDK|[0&{PGUP}]#");//comando para voltar slides
        flag_ava =0;//Seta 0 a variável flag_ava
        flag_rec=0;//Seta 0 a variável flag_rec
  }
  delay(500);//Atraso de 0,5 segundos
  ct_2=2;//Seta 2 a variável ct_2

}
```