CONTROLANDO REMOTAMENTE UM ROBÔ USANDO JOYSTICK

Andrés Vidal Berriel[1], Augusto Zanella Bardini[1], Guilherme Fontoura[1], Silvia de Castro Bertagnolli[2], Patrícia Nogueira Hübler[2]

[1]INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO SUL – CÂMPUS CANOAS
Rua Dra. Maria Zélia Carneiro de Figueiredo, 870
CEP 94412-240 – Canoas – RS

[2] INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO SUL – CÂMPUS CANOAS
Rua Dra. Maria Zélia Carneiro de Figueiredo, 870
CEP 94412-240 – Canoas – RS

Resumo: Ao realizar uma análise da literatura é possível encontrar trabalhos que apresentam a robótica sendo usada, principalmente, para o ensino de física e matemática. Alguns trabalhos também demonstram como a robótica pode ser utilizada como uma ferramenta para o ensino de programação. Assim, o objetivo deste trabalho, integrado a um projeto de pesquisa que investiga como a robótica pode qualificar o ensino de programação, é através do uso do kit de robótica da Xbot (robô Curumin) propor desafios de programação com as linguagens de programação C/C++. No caso deste artigo, resolveu-se descrever de forma detalhada os desafios encontrados na integração de um controle *joystick* às funcionalidades do robô Curumim. Para tanto, foram organizados alguns passos e foram realizados registros dos problemas que podem ser encontrados neste processo de integração mostrando as soluções encontradas para cada situação.

Palavras Chaves: Robótica para ensino de programação, Programação C/C++, Kit Robótico Curumin.

Abstract: Once performed an analysis of the literature, it's possible to find work which show robotics being mainly used for physics and mathematics' studies. Some work also show how robotics can be used as a tool for teaching programing. So, this work's objective , integrated to a research project which investigates how robotics can qualify teaching of programming, is, through the use of Xbot's robotics kit (Curumim robot), to propose programing challenges with the programming languages C / C + +. In this article's case, it was decided to describe detailedly the challenges found in the integration of a joystick to control the Curumim robot features. Then, we organized a few steps and records were taken of the problems which may be encountered in this integration process, showing the solutions for each situation.

Keywords: Robotics for teaching of programming, C/C++ programming, Robotics Kit Curumin.

1 INTRODUÇÃO

Este trabalho está vinculado a um projeto de pesquisa que investiga como a robótica pode qualificar o ensino de programação, denominado “A robótica como ferramenta para qualificar o ensino no IFRS - Câmpus Canoas”.

O foco do subprojeto, a partir do qual este artigo foi elaborado, compreende propor desafios de programação com as linguagens de programação C/C++ utilizando-se do kit de robótica da Xbot (robô Curumin).

Ao realizar uma análise do robô Curumin percebeu-se que havia, basicamente, duas maneiras de interagir com ele [XBOTa, 2012]:

(i) programação em blocos - através de uma interface gráfica com o usuário é possível arrastar blocos de comandos, informando a sequencia de ações esperadas, e conectar o robô de modo que ele execute os passos definidos;

(ii) programação com a linguagem de programação C++ - usando o IDE (*Integrated software Development*) Visual C++ é possível criar programas utilizando classes pré-definidas de modo a movimentar e controlar o robô.

A partir desta investigação percebeu-se que a interação com o robô poderia ser melhorada, e assim, este foi o desafio imposto pela equipe envolvida com o projeto “utilizar controles externos para movimentar o robô”.

Assim, escolheu-se o *joystick* como periférico a ser integrado e como modelo funcional. Espera-se com este periférico estabelecer uma padronização no modo de integrar o *joystick* ao Curumim e na forma de desenvolver aplicações com este tipo de periférico.

Cabe destacar que, todos os alunos que fazem parte do projeto cursam o 2º e o 3º ano do curso técnico de informática integrado ao ensino médio. Já os bolsistas que fazem parte da equipe que propôs o presente trabalho são alunos do 2º ano que estão estudando a programação estuturada com a linguagem de programação C. Eles ainda não estudaram os conceitos de

orientação a objetos, logo alguns dos código para a programação dos robôs são mais dificeis de desenvolver.

O artigo prossegue apresentando na seção 2 algumas observações sobre o *kit* de robótica utilizado, a seção 3 aborda o desafio e o modelo funcional e a aplicação avaliativa. Já a seção 4 descreve a integração com o *joystick* e as biliotecas utilizadas. Na seção 5 são abordados alguns dos problemas encontrados. Finalmente, a seção 6 apresenta algumas conclusões obtidas com os experimentos já realizados, bem como as perspectivas futuras.

2 O KIT DE ROBÓTICA

Este artigo utiliza o robô Curumim, da Xbot, utilizado, principalmente, para fins de aprendizagem em escolas ne nível fundamental e médio.

Ele disponibiliza o software Curumin que viabiliza a programação em blocos. A programação em blocos é realizada em um editor que permite ao usuário arrastar blocos de comandos para construir, de forma rápida, os programas que irão controlar o robô. Estes blocos delimitam o início e o fim do programa, bem como as estruruturas de controle de seleção e de repetição que farão o robô seguir as instruções definidas. Além disso, este software ainda possibilita que o robô possa andar reto, rotacionar, realizar moviumentos em curva e ativar os motores de forma individual [XBOTa, 2012].

Outra possibilidade de uso do robô é através do Visual C++ que possibilita a programação integral da arquitetura de hardware do robô com a linguagem de programação C/C++. Através da criação de um projeto e da inclusão de bilbiotecas de funções específicas do Curumin é possível realizar os mesmos movimentos que a programação em blocos possibilita, além do uso de sensores e do reconhecimento de imagens [XBOTa, 2012].

O robô Curumin (Figura 1) possui três rodas omnidirecionais, cinco sensores infravermelhos de proximidade sendo quatro nas diagonais e um na frente. Além de uma câmera *onboard* que é acompanhada de uma biblioteca de processamento de imagens (OpenCV) a qual permite, entre as suas opções de processamento de imagens, a leitura de QR Code. Ele é acompanhado por um rádio base (*trascreiver*) com interface USB (*Universal Serial Bus*) [USB, 2012] que opera na faixa de 2,4 GHz, necessário para a conexão do programa com o robôCurumim e dois *packs* de baterias com um carregador duplo [XBOTb, 2012].

Figura1 – Kit de Robótica Curumin

3 O DESAFIO

Basicamente, o foco do robô Curumim é a programação utilizando-se raciocínio lógico, com uso de câmera e sensores de proximidade, de forma a garantir certo grau de autonomia. A manipulação do robô é extremamente simples, pois basta utilizar comandos de entrada e saída de dados e estruturas condicionais (como if/else ou switch/case) ou estruturas de repetição (como for, while ou do/while) na linguagem de programação C/C++ .

A parte mais complexa da programação seria o uso de conceitos de orientação a objetos, visto que a equipe envolvida com o projeto está no 2º ano do Ensino Médio e ainda está estudando programação estruturada.

Assim, a equipe envolvida pensou em criar diferentes tipos de interfaces entre o robô e o usuário: dispositivos/controles externos e interfaces gráficas intuitivas.

O primeiro desafio imposto à equipe foi enviar comandos ao robô Curumin através de dispositivos que não o teclado (entrada padrão). O dispositivo selecionado foi o *joystick*, que é um dispositivo especial de interface física com o usuário, que em conjunto com o robô Curumim permite interagir de forma mais dinâmica e realista, um modelo de *joystick* pode ser visualizado na Figura 2.

Figura2 – Exemplo de um *Joystick* e Botões de Movimentação (imagem superior) e Mapeamento dos Botões do *Joystick* com o emulador Xpadder (imagem inferior)

Entende-se como “modelo funcional” o dispositivo em cujas opções e capacidades de entrada se baseiam as atividades nas quais o programa resultante de pesquisa seria aplicado. Assim, colocou-se como “opção de entrada” cada botão do *joystick* em específico, e como “capacidade de entrada” as delimitações que o controle terá em relação às suas opções de entrada, tais como localização física, comportamento (analógico ou não) e o número de opções disponíveis.

Desse modo, estabelece-se um padrão sobre o qual as atividades devem ser desenvolvidas para otimizar o desempenho do periférico em cada aplicação e garantir uma

melhor interação entre o usuário e o robô Curumim. Espera-se da sistematização decorrente da utilização do modelo funcional uma melhoria na praticidade de criação de novas aplicações para a programação do controle remoto, sem causar diferenças significativas na lógica dos códigos implementados nem no jeito de integração de dispositivos

Foi selecionado, como especificação do modelo funcional, em um primeiro momento, um *joystick* com fio. Logo, esse modelo foi modificado e migrou-se para um *joystick* sem fio, modelo DUAL SHOCK da marca Multilaser.

O primeiro desafio foi fazer o robô funcionar corretamente, seguindo as interações impostas pelo usuário através do controle remoto. A próxima seção descreve como foi o processo de integração do *joystick* e quais os recursos computacionais usados para viabilizar essa integração.

4 INTEGRAÇÃO DO JOYSTICK

A primeira etapa da integração com o *joystick* foi a elaboração de um programa, na linguagem C/C++, que atendesse aos comandos abaixo descritos, mas cuja origem fosse o teclado:

(i) andar para frente – que informava ao robô que ele deveria andar 20 centímetros em linha reta para frente;

(ii) andar para trás – que informava ao robô que ele deveria andar 20 centímetros em linha reta para trás;

(iii) rotacionar andar à direita – que informava ao robô que ele deveria andar aproximadamente 90 graus em linha reta à direita;

(iv) rotacionar andar à esquerda – que informava ao robô que ele deveria andar aproximadamente 90 graus em linha reta à esquerda;

(v) andar rápido, ou modo turbo, que informava ao robê que ele deveria usar a velocidade 3 percorrendo 30 centímetros para frente;

(vi) rotacionar 180 graus– que informava ao robô que ele deveria andar aproximadamente 180 graus;

(vii) corrigir à direita – que informava ao robô que ele deveria andar aproximadamente 45 graus à direita;

(viii) corrigir à esquerda – que informava ao robô que ele deveria andar aproximadamente 45 graus à esquerda;

A estrutura lógica do programa encontra-se esquematiza na Figura 3.
Uma vez concluída a versão básica do programa partiu-se para a integração do mesmo com o *Joystick*. No início, pensou-se que seria uma tarefa trivial, mas ao se realizar consultas na literatura e na Internet ela tornou-se mais complexa, devido, principalmente, à falta de documentação disponível sobre as possibilidade de integração de um *joystick* a um programa escrito na linguagem de programação C/C++.
Mesmo com a falta de documentação foram identificadas algumas alternativas para fazer a integração Curumim-*joystick*. Dentre as quais se pode incluir software de terceiros, bibliotecas especiais voltadas ao desenvolvimento de jogos e bibliotecas pertencentes a SDKs (*Software Development Kit) kits* de desenvolvimento de software C/C++.

```
do{

    comando = tolower(_getch());
    switch (comando){
        case 'w': //FRONTAL
            curumim->move(curumim->getTarget(), curumim->getSource(), 100, 0, 200);
            curumim->waitStopped(curumim->getTarget(), curumim->getSource());
            printf("%c ",536);
            break;
        case 's': //TRASEIRO
            curumim->move(curumim->getTarget(), curumim->getSource(), -100, 0, 200);
            curumim->waitStopped(curumim->getTarget(), curumim->getSource());
            printf("%c ",537);
            break;
        case 'a': //ESQUERDA
            curumim->rotate(curumim->getTarget(), curumim->getSource(), 103, 100);
            curumim->waitStopped(curumim->getTarget(), curumim->getSource());
            printf("%c ",539);
            break;
        case 'd': //DIREITA
            curumim->rotate(curumim->getTarget(), curumim->getSource(), -103, 100);
            curumim->waitStopped(curumim->getTarget(), curumim->getSource());
            printf("%c ",538);
            break;
        case '0': //TURBO
            if (cTurbo<3){
                curumim->move(curumim->getTarget(), curumim->getSource(), 600, 0, 300);
                curumim->waitStopped(curumim->getTarget(), curumim->getSource());
                printf("TURBO ");
                cTurbo++;
            }else printf("NOTURBO ");
            break;
        case '1': //180
            curumim->rotate(curumim->getTarget(), curumim->getSource(), 136, 100);
            curumim->waitStopped(curumim->getTarget(), curumim->getSource());
            printf("%c%c ",539,538);
        default:
            if(comando=='4'||comando=='6'||comando=='q'){ //DIAGONAL ESQUERDA
                curumim->rotate(curumim->getTarget(), curumim->getSource(), 51, 100);
                curumim->waitStopped(curumim->getTarget(), curumim->getSource());
                printf("DE ");
            }else if(comando=='5'||comando=='7'||comando=='e'){ //DIAGONAL DIREITA
                curumim->rotate(curumim->getTarget(), curumim->getSource(), -51, 100);
                curumim->waitStopped(curumim->getTarget(), curumim->getSource());
                printf("DD ");
            }
    }
}while(comando!=27);//ENQUANTO NÃO TECLA ESC
```

Figura 3 - Demonstração do código-base da aplicação do controle remoto do robô Curumim com o teclado como entrada de dados

4.1 Software de terceiros

Como primeira alternativa para a integração, foram identificados aplicativos que conseguem interceptar mensagens enviadas pelo *joystick* ao computador. Desse modo, eles conseguem converter estas mensagens em mensagens de teclado definidas pelo usuário, isto é, criam uma correspondência entre os dados dos botões do *joystick* e os dados das teclas do teclado, como, por exemplo, os dados ilustrados pela Figura 4.

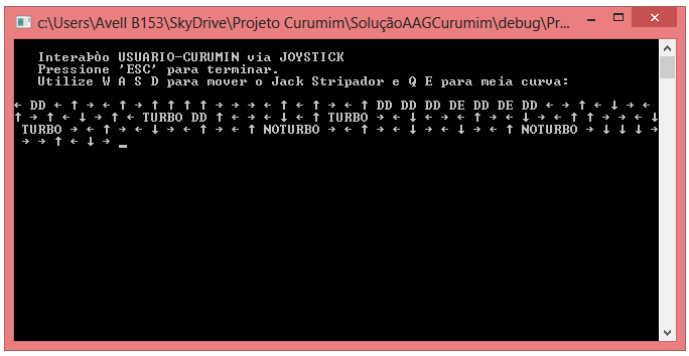

Figura 4 – Correspondência entre os dados dos botões do joystick e os códigos do teclado

Pode-se afirmar que estes programas emulam os códigos enviados pelas teclas do teclado no *joystick*. Assim, é possível fazer a correspondência entre o programa e o *joystick* de forma rápida e prática, uma vez que o código-base não precisa ser editado sempre.

Dentre esses "emuladores" o que mais se destaca é o Xpadder [Xpadder, 2013], um software do tipo proprietário que tem como objetivo principal a utilização de *joystick* em jogos que não possuem uma configuração de entrada nativa para estes dispositivos. Considerando o objetivo do programa Xpadder, este é adequado às nossas necessidades,uma vez que, mesmo

não se tratando de um jogo de computador, o robô não possui configuração de entrada nativa para dispositivos alheios ao teclado, fazendo com que uma interface deste tipo seja realmente útil. Esta alternativa foi a mais prática, porém menos eficiente, uma vez que as mensagens do *joystick* passarão por duas interfaces de tradução (Xpadder e Visual C++ 2005), ao invés de uma só, até chegar ao robô.

4.2 Bibliotecas de jogos

Como segunda alternativa, foram selecionadas diversas bibliotecas voltadas ao desenvolvimento de jogos. Como os *joysticks* foram os dispositivos pensados para a utilização em jogos, é imprescindível que esse tipo de biblioteca possua o recurso adequado para utilizá-los. Desse modo, pensou-se na possibilidade de fazer a integração com este tipo de recurso.

Dentre as bibliotecas analisadas, a que mais se destacou foi a Allegro [Allegro, 2013]. Porém a sua versão para o ambiente Visual C++ 2005, único software compatível com o *kit* do robô Curumim, era pouco intuitiva e não possuía a documentação apropriada para a sua utilização com *joysticks*.

Esta alternativa resultou mais eficiente que a anterior, software "emuladores", se considerado o código final produzido. Porém, ela é menos eficiente se observado o aspecto da carga de bibliotecas, pois para usá-la é necessário carregar uma biblioteca inteira composta por muitas funcionalidades, que não são pertinentes para a integração com *joysticks.* Isso ocorre, porque para usar o *joystick* apenas alguns comandos seriam utilizados não sendo necessária inclusão de uma biblioteca de funções muito grande.

4.3 Bibliotecas de SDKs

A terceira alternativa identificada foi a de utilizar bibliotecas anexadas a *kits* de desenvolvimento de software, mais especificamente as funcionalidades do chamado DirectInput, incluído no DirectX SDK [DirectInput, 2013].

Este SDK encontra-se integrado ao Windows SDK, o qual fornece comandos úteis para a gestão e utilização de *joysticks* em programas desenvolvidos por diversas linguagens de programação, dentro as quais C/C++ está incluída.

Esta alternativa demonstrou ser a mais eficiente, desde que o *kit* do robô Curumin seja utilizado no sistema operacional Windows, da Microsoft [Joysticks, 2013]. Uma vez que a migração do programa para outros sistemas operacionais fica impossibilitada através do uso deste recurso.

Como o nosso objeto de pesquisa é o robô Curumim, e ele tem condições restritas e limitadas de uso ao sistema operacional Windows acredita-se que esta opção é a mais viável. Destaca-se ainda que, o *kit* do robô Curumin é fortemente limitado a condições específicas desse sistema operacional, pois o desenvolvimento dos programas é todo via Visual C++ 2005 o qual deve ser, preferencialmente, instalado no Windows XP. Além disso, toda questão relacionada com migração para outros sistemas operacionais seria irrelevante, pois outros sistemas operacionais não seriam compatíveis com o hardware que é utilizado por este projeto.

5 PROBLEMAS ENCONTRADOS

Durante o processo de integração *joystick*-programa-Curumim foram encontrados diversos problemas e impedimentos que levaram a equipe a optar por um ou outro recurso para alcançar o objetivo proposto.

A maioria destes problemas se deu por motivos de incompatibilidade e ausência de documentação sobre como resolver os problemas. A arquitetura de hardware e os software por ela exigidos também foram definitivos para descartar certos métodos e/ou soluções.

O primeiro problema encontrado foi referente ao uso de software de terceiros. Ao testar o Xpadder no sistema operacional dos computadores dedicados à pesquisa, Windows 7, o mesmo funcionava perfeitamente. Contudo, o programa não conseguia interceptar e/ou converter as mensagens do *joystick* para o programa-base do robô, uma vez que não era compatível com o console de depuração, único modo no qual o robô mencionado consegue receber os comandos do programa, do Visual C++ 2005. Posteriormente, percebeu-se que este problema ocorria devido à incompatibilidade entre a versão de Xpadder utilizada (2012.05.01) e do compilador no Windows 7.

Outro ponto que merece destaque é que o ambiente de programação Visual C++ 2005 executa no Windows 7 somente no modo de compatibilidade. Assim, foi testada a funcionalidade sob estas mesmas circunstâncias no sistema operacional da Microsoft, o Windows 8, o qual funcionou perfeitamente sem exigir modo de compatibilidade de nenhum dos dois software utilizados. Assim, adotou-se como solução o uso do sistema operacional Windows 8 para as demais etapas do projeto de pesquisa.

No que diz respeito à biblioteca Allegro, a mesma possuía documentação praticamente nula para a versão selecionada e às funcionalidades que se desejava aplicar. Concluiu-se que era praticamente impossível a sua utilização uma vez que aconteciam erros múltiplos de vinculação e compilação. Assim, o uso dessa biblioteca foi descartado para aplicação no projeto.

Considerando o uso do DirectInput do Windows SDK, esta alternativa seria a mais adequada e a mais eficaz se existisse melhor documentação sobre as funcionalidades necessárias para a utilização dos *joysticks*. A documentação encontrada é pobre e pouco intuitiva, fator que, somado à falta de tempo para a sua aplicabilidade na pesquisa e à complexidade de implementação dos recursos do SDK, resultou suficiente para descartar esta opção momentaneamente. O maior problema com o uso do SDK foi a obtenção do retorno das funções, pois os valores de retorno que constavam na documentação oficial do SDK não coincidiam com os valores de retorno reais das funções, deixando a tarefa ainda mais complexa, e prejudicando o entendimento e a aplicabilidade da biblioteca.

Ao concluir esta seção, convém destacar que a solução que mais se adequou ao uso do *joystick*-programa-Curumim compreendeu o uso do "emulador" Xpadder aliado ao sistema operacional Windows 8, conforme ilustra a Figura 5.

Figura 5. Esquematização da trajetória das mensagens e comandos entre o *joystick* e o robô Curumim.

6 CONCLUSÃO

Pode-se afirmar que o desafio foi concluído com sucesso: o uso do sistema operacional Windows 8, aliado com o ambiente de programação Visual C++ 2005 e o uso de software de terceiros, Xpadder, para fazer a integração com o *joystick* possibilitou realizar os mapeamentos do teclado esperados e qualificar o processo de interação do usuário com o robô.
No decorrer do texto, apontou-se que essa solução seria menos eficiente, uma vez que os comandos passam por duas interfaces diferentes em vez de por uma só (Figura 5). Porém, esta solução demandou menos código e menos tempo de programação e integração.

Das aplicações relativas à avaliação e teste do programa desenvolvido, as mesmas podem ser pensadas de modo a haver competição entre dois ou mais robôs. Dessa forma, pode ser testada a interação do robô com o usuário, e vice-versa, de modo que a responsabilidade de acertar cada movimento seja mútua, e não só do Curumim. Pode-se, então, testar a interação nas suas duas dimensões, e otimizar o programa para facilitar e simplificar a interação do usuário.

Mesmo sendo priorizadas as atividades de competição que, ao exigir mais precisão do robô, ajudam a polir o código de forma mais eficaz, espera-se que o controle remoto com *joystick* seja aplicável em mais modalidades. Estas modalidades poderiam, com os ajustes e equipamentos necessários, chegar a integrar a câmera do curumim com o controle desenvolvido. Outra possibilidade de utilização está em limitar algumas funcionalidades do programa às quais os participantes das atividades poderiam ter acesso. Assim, possibilita-se também a existência de juízes moderados (que poderiam interferir no jogo dos participantes através de dispositivos que possuam todas as funcionalidades, como um teclado), garantindo uma redução das probabilidades de jogadas não permitidas ou erros inadvertidos por parte dos participantes.

O próximo desafio do projeto compreende em aprofundar a tarefa proposta a níveis mais complexos, onde o robô Curumim receberia os parâmetros de funcionamento, que, normalmente, são pré-definidos (velocidade, rotação e distância de locomoção) sejam modificáveis pelo usuário em tempo real através de interfaces gráficas intuitivas. No entanto, devido à falta de conhecimentos teórico-práticos da equipe com relação à orientação a objetos este desafio será realizado posteriormente, pois será necessário estudar o modelo orientado a objetos com maior profundidade.

AGRADECIMENTOS

A equipe do projeto gostaria de agradecer ao CNPq por financiar as bolsas PIBIC-EM/CNPq dos alunos Andrés Vidal Berriel, Augusto Zanella Bardini e Guilherme Fontoura envolvidos com o projeto, à PETROBRAS e ao BNDES pela doação dos *kits* robóticos e ao IFRS por fornecer apoio financeiro à aquisição de alguns materiais de consumo e permanentes necessários à execução do projeto.

REFERÊNCIAS BIBLIOGRÁFICAS

"Allegro: A game programming library". 2013. Disponível em: <alleg.sourceforge.net/>. Acesso em: 17 ago. 2013.

"DirectInput". Disponível em: <msdn.microsoft.com/en-us/library/windows/desktop/ee416842(v=vs.85).aspx>. Acesso em: 17 ago. 2013.

Joysticks (Windows) (2013). Disponível em: <msdn.microsoft.com/en-us/library/windows/desktop/dd757116(v=vs.85).aspx/css>. Acesso em: 17 ago. 2013.

USB Joystick With C++ And Direct Input (2012). Disponível em: <stackoverflow.com/questions/9742611/usb-joystick-with-c-and-directinput>. Acesso em: 17 ago. 2013.

XBOTa. (2012) "Curumim – Apostila de software V1.1". Disponível em: www.xbot.com.br/externo/cd_curumim/Apostila_Curumim_Software_v1.1.pdf. Acesso em abril de 2013.

XBOTb (2012) "Curumin - Apostila de Hardware V2.0 Hardware". Disponível em: www.xbot.com.br/externo/cd_curumim/Apostila_Curumim_Hardware_v2.0.pdf. Acesso em: abril de 2013.

XPadder (2013) Xpadder. Disponível em: http://www.xpadder.com. Acesso em: agosto de 2013.