AMBIENTE DE SIMULAÇÃO EM PYTHON BASEADO NO ROBODECK E MODELAGEM COM ARQUITETURA PLANEJADOR-REATOR

José Ailton Batista da Silva[1], Artur Ferreira Moreira[1], Andressa da Silva Fernandes[1], Alan Vinicius de Araújo Batista[1], Pedro Henrique Almeida Miranda[2]

[1] INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO CEARÁ CAMPUS CEDRO
Alameda José Quintino, s/n - Prado
63400-000 – Cedro – CE

[2] UNIVERSIDADE FEDERAL DO CEARÁ
Rua Cel. Estanislau Frota, s/n, Centro
62010-560 – Sobral – CE

Resumo Com a utilização da robótica cada vez mais presente nas indústrias cria-se a necessidade por profissionais capacitados na área. Uma categoria de robôs que vem ganhado bastante espaço no mercado são os robôs móveis, por isso em centros de pesquisas universitárias é comum ter-se plataformas robóticas, como por exemplo, o Robodeck. Este trabalho apresenta o desenvolvimento de um ambiente de simulação baseado nesse robô, utilizando a linguagem Python, visando facilitar o estudo de arquiteturas e comportamentos robóticos.

Palavras Chaves: Pygame, Python, Robodeck, Robótica.

Abstract: With the use of robotics frequently present in the industries, there's the need for trained professionals in the area. One category of robots has gained a lot of space on the market are mobile robots, so in University Research Centers is common exist robotics platforms, like the platform Robodeck. This paper present the development of an simulation environment based on this robot, coded in Python aiming to make it easy the study of architectures and robotic behaviors.

Keywords: Pygame, Python, Robodeck, Robotics.

1 INTRODUÇÃO

O uso da robótica está cada vez mais presente em nosso cotidiano. Hoje é comum ver robôs industriais que soldam, pitam e movimentam grandes objetos. Robôs que desativam minas e bombas, que trabalham submersos no mar sob grandes pressões e em alto grau de periculosidade [Secchi, 2008].

Dentre as variedades de arquitetura robótica existente no mercado e nas industrias, uma vem recebendo bastante destaque no cenário atual que são os robôs móveis. A definição de robô móvel é um agente artificial que atua de forma racional, com capacidade de movimentação no ambiente que fora projetado. Esse tipo de robô pode extrair informações sobre o ambiente o qual está imerso e usar seu conhecimento sobre um determinado domínio realizando alguma tarefa especifica [Siegwart, Nourbakhsh e Scaramuzza, 2011].

Segundo Kerschbaumer[2014] o processo de construção de dispositivos robóticos físicos necessita de tempo e recursos que nem sempre estão disponíveis as universidades ou aos alunos. Em geral quando tal aplicação é construída normalmente resulta em experimentos limitados e com poucos sensores.

Visando acabar com essa dificuldade foi que o Instituto Federal de Educação, Ciência e Tecnologia do Ceará campus Cedro adquiriu uma plataforma móvel de robótica universal chamado RoboDeck. Essa plataforma móvel contém diversos sensores e pode ser programado usando várias linguagens de programação, criando então os mais diversos comportamentos para o robô e auxiliando nas aulas de robótica do curso de Mecatrônica Industrial.

Apesar dessa plataforma auxiliar nas aulas de robótica móvel, um outro problema comum é que em geral a quantidade de alunos em sala de aula é grande para usar somente uma plataforma e não existe um ambiente de teste para prever o comportamento do robô, além é claro, de se tratar de uma plataforma cara sendo necessário tomar bastante cuidado contra colisões ou outro fator que venha a danificá-lo.

O objetivo desse trabalho é então apresentar um ambiente de simulação baseado na geometria do RoboDeck. Esse ambiente foi desenvolvido utilizando a linguagem Python através da *framework* de criação de jogos PyGame [PyGame, 2017]. A linguagem Python foi escolhida porque já vem embutida nos sistemas operacionais Linux facilitando a migração do código gerado para a plataforma robótica.

Nos próximos tópicos serão então apresentados alguns detalhes do Robodeck e desenvolvimento do ambiente de simulação. Para a validação desse ambiente virtual será utilizado uma das arquiteturas estudadas na disciplina de robótica móvel. A arquitetura escolhida foi o *Planejador-Reator* por ter características híbridas e utilizar somente dois módulos.

2 A PLATAFORMA ROBÓTICA ROBODECK

O RoboDeck foi desenvolvido pela empresa C. Associados Desenvolvimento Tecnológico Ltda, sendo comercializado pela Xbot [Xbot, 2014]. É um robô móvel que foi elaborado no intuito de promover o desenvolvimento educacional nas

diversas áreas base da Robótica, como por exemplo: eletrônica, mecânica, sistemas embarcados, sensoriamento, programação, entre outros [Robodeck, 2011].

Por possuir diversas características e recursos disponíveis essa plataforma é bastante customizável, atuando como ponto de partida para vários tipos de projeto de pesquisa. Dois módulos são responsáveis pelo controle do Robodeck. Um módulo com *ARM9* responsável pelas funções básicas de movimentação e aquisição de dados pelos sensores. O outro é um microcontrolador *Jennic* que controla as interfaces de redes de comunicação, esses módulos se comunicam por uma UART [Robodeck, 2011]. No próprio veículo também existe uma placa-mãe *NanoITX* com o sistema operacional *Debian Squeeze*, pela qual existe um módulo de alta performance que pode ser adicionados algoritmos de controle do robô [Pissardini, 2014].

Entre os sensores disponíveis nessa plataforma móvel podemos destacar: câmera, bússola, acelerômetro, GPS, sensor infravermelho, ultrassônico e temperatura [Xbot, 2010]. Na **Erro! Fonte de referência não encontrada.** está ilustrado a localização de cada um desses sensores, que estão fixados na carcaça do robô. Observa-se que há quatro sensores ultra-som, um em cada lado do carro. Esses sensores são capazes de detectar objetos de 3 centímetros até 6 metros de distância. Outro sensor importante é o sensor bússola capaz de sentir a rotação do robô variando de 0º a 359,9º do norte magnético, podendo então saber a pose do robô [Netto, 2012] [Robodeck, 2011].

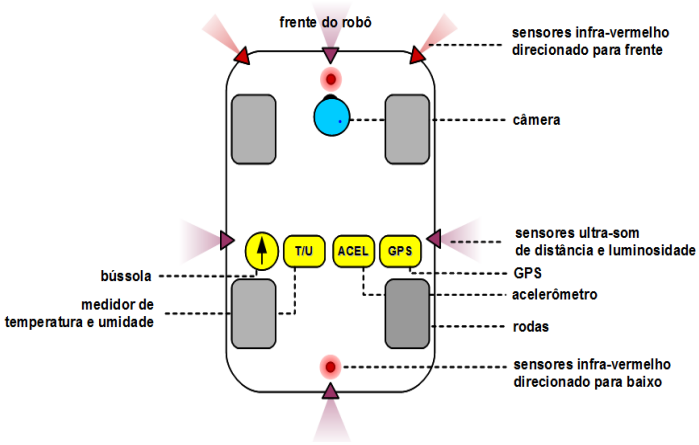


Figura 1 - Sistema de sensoriamento [Robodeck, 2011].

Além da programação local (através da placa-mãe extra) disponível nessa plataforma, o usuário pode usar comunicação sem fio como Wi-Fi, Bluetooth ou ZigBee. Com o uso da comunicação ZigBee, por exemplo, podemos comandar e receber informações a até 1000 metros de distância [Xbot, 2010].

Utilizando a comunicação sem fio o controle do robô é feito através de um protocolo nativo dessa plataforma com comandos de baixo e alto nível [Netto, 2012]. Também é possível utilizar uma SDK desenvolvida em C++/CLI e existe uma versão para Java ME bem documentada. Assim pode-se criar aplicações usando sua linguagem desejada como VisualBasic, C#, C++, Iron Python, ou Java. Essa plataforma é de código aberto dando total liberdade para o desenvolvimento de aplicações [MUNOZ, 2011] [Xbot, 2010].

Com relação a sua mecânica, o robô foi projetado para ser omnidirecional, onde o controle de direção é baseado na geometria *Ackermann*. Suas quatro rodas são independentes entre si e controladas através de servo-motores de direção, um para cada roda. As rodas dianteiras são responsáveis pela tração e também estão acopladas a *encoders* capazes de medir a velocidade e aceleração do robô. Para melhor controle em condições de velocidade longitudinal o *firmware* do robô se baseia em um modelo matemático não linear com dois graus de liberdade que são o deslocamento lateral e ângulo de guinada [Xbot, 2010].

Na figura 2 está em destaque a parte lateral e superior do Robodeck. Observa-se que a parte superior é possível adicionar outros dispositivos tais como garras, braços, sensores, e outros. Isso porque essa plataforma móvel é modular [MUNOZ, 2011] e contém alguns conectores extras para acionar esses módulos, além é claro das entradas USB disponível na placa-mãe e estendida através de *hubs*.

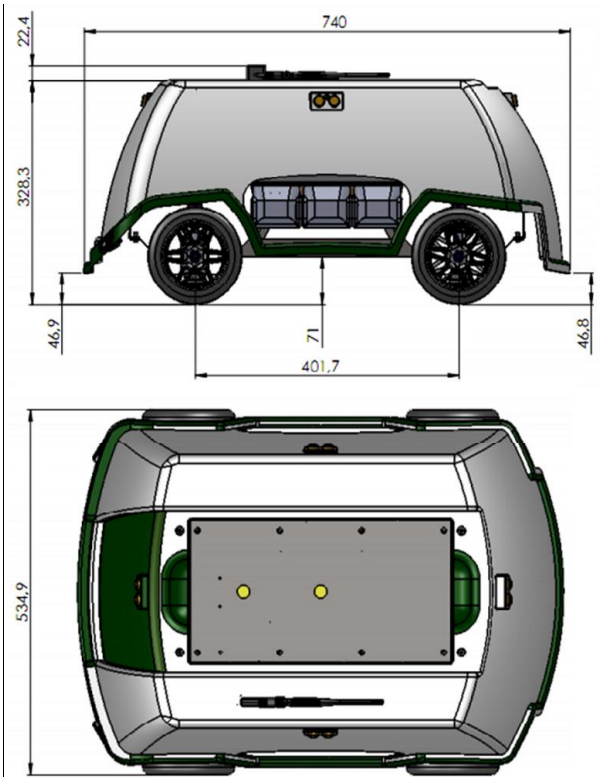


Figura 2 - Visão frontal e superior [Robodeck, 2011].

3 MATERIAIS E MÉTODOS

Nessa seção serão descritas as características e funcionamento desse ambiente de simulação, bem como todo o processo de implementação utilizando a linguagem *Python* e a biblioteca *Pygame*. O Python é uma linguagem de programação de propósito geral, tem suporte a programação estruturada e também orientada a objetos, capaz de gerar aplicações altamente interativas e com pouca digitação de código, além de ter um grande conjunto de bibliotecas [Lutz, 2007] [McGugan, 2007] [De Souza, 2014]. O Pygame foi criado utilizando as funções da biblioteca SDL (*Simple Direct Media Layer*) o qual simplifica tarefas de aplicações multimídia e principalmente desenvolvimento de jogos. Além da funcionalidades da SDL, foram acrescentados ao Pygame funções como detecção de colisão, *sprites*, *render groups* e outros [Marques, 2011] [De Souza, 2014]. No entanto o Pygame não é um modulo padrão do Python sendo necessário o seu *download* e instalação, e, assim como o Python está disponível gratuitamente [Sweigart, 2012].

O simulador foi desenvolvido para ser um módulo do Python e utiliza os paradigmas de orientação a objeto. Todo o módulo está disponível em uma única pasta chamada *robodeck_pkg*, o

qual o aluno deve copiar para sua aplicação. A figura 3 representa um diagrama de classes básico desse ambiente, nele estão representados duas classes principais: *Robodeck* e *CollisionRobotException*.

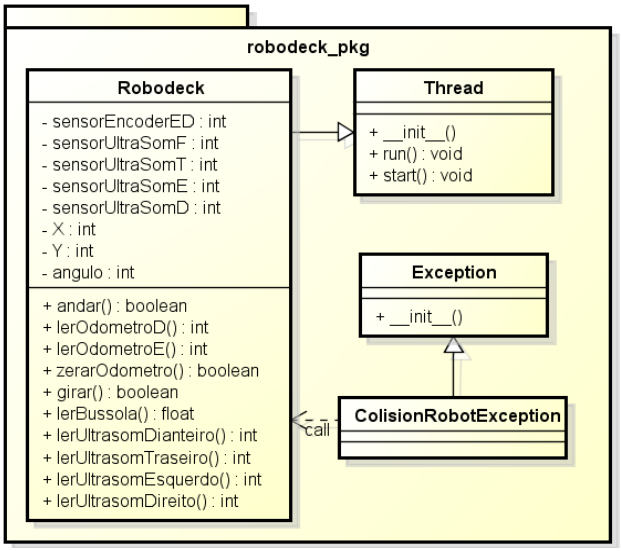


Figura 3 - Diagrama de classes do simulador.

A classe *Robodeck* é uma especialização da classe de pseudo multiprocessamento disponível na linguagem usada, logo toda a simulação usando *Pygame* ocorre em paralelo com as instruções do usuário, podendo inclusive fazer testes em tempo real usando o terminal do sistema. Essa classe é uma interface com o desenho que representa um robô real na janela de animação em duas dimensões.

Os únicos dispositivos implementados nesse simulador foram os *encoders*, os quatro sensores ultrassônicos, o sensor bussola e as quatros rodas. Para ter acesso a esses dispositivos usam-se os métodos do objeto, por exemplo, a instrução *andar()* faz com que todas as rodas se alinhem e gire para a frente, já a instrução *lerUltrasomDianteiro()* retorna a distância de um objeto a frente detectada pelo sensor.

Na janela de animação do simulador, representada na figura 4, é possível o usuário adicionar obstáculos no ambiente o qual é detectável pelo sensores ultrassônicos e caso o robô colida com alguma, a exceção *CollisionRobotException* é chamada parando a execução da simulação. Nessa janela há uma rosa dos ventos que indica a direção do norte, uma imagem de um alvo e o robodeck. As linhas que se concentram no centro do robô indicam onde os sensores ultrassônicos detectaram obstáculos, é possível modificar as posições desses objetos e também ocultar essas linhas através do arquivo de inicialização disponível na pasta do módulo. O cenário de obstáculos criado pelo usuário pode ser salvo e recuperado usando teclas de atalho, bem como carregar cenários pré-definidos.

É importante salientar que o simulador não faz qualquer conexão com o robô físico real, sendo usado somente para testar comportamentos e implementações antes de utilizar o robô físico propriamente dito. Também não há nenhuma classe disponível para isso sendo necessário o aluno criar sua própria implementação da classe Robodeck que conecte a plataforma real de acordo com o diagrama de classes já discutido.

4 RESULTADOS E DISCUSSÃO

Nessa seção será descrito a validação do simulador desenvolvido, para isso foi implementado uma arquitetura robótica chamada *Planejador-Reator* utilizando todos os recursos disponíveis no ambiente simulado. Além disso foi adicionado ao comportamento dessa implementação a capacidade de mapeamento do ambiente através dos sensores ultrassônicos, mas não o foi utilizado na manipulação do robô, servindo somente para demonstrar o potencial desse ambiente de simulação.

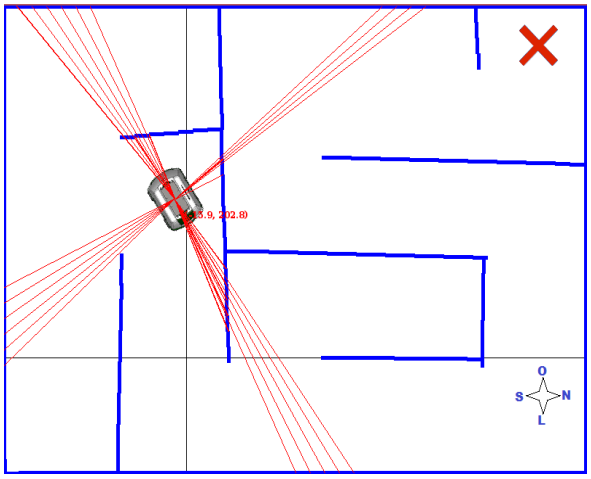


Figura 4 - Janela de animação do simulador.

A arquitetura *Planejador-Reator* possui duas partes principais que trabalham de forma paralela: um reator que controla o robô de forma reativa adaptável, e um planejador responsável pela deliberação modificando alguns comportamentos do reator [Siegwart, Nourbakhsh e Scaramuzza, 2011] [Junior, 2006]. Essa arquitetura é então dita como híbrida por ter elementos de planejamento e ao mesmo tempo reativo, conforme mostrado na figura 5.

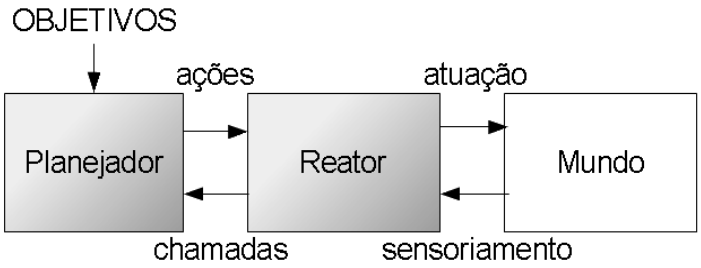


Figura 5 - Arquitetura do Planejador-Reator.

Utilizando os conceitos apresentados sobre essa arquitetura foi criado uma classe chamada *Reator* que contém um objeto do tipo *Robodeck*, ou seja, o *Reator* importa as classes disponível no módulo como ilustrado na figura 6. Também foi criado a classe *Planejador* o qual contém um objeto *Reator* e também ficou responsável por gerar um mapa do ambiente.

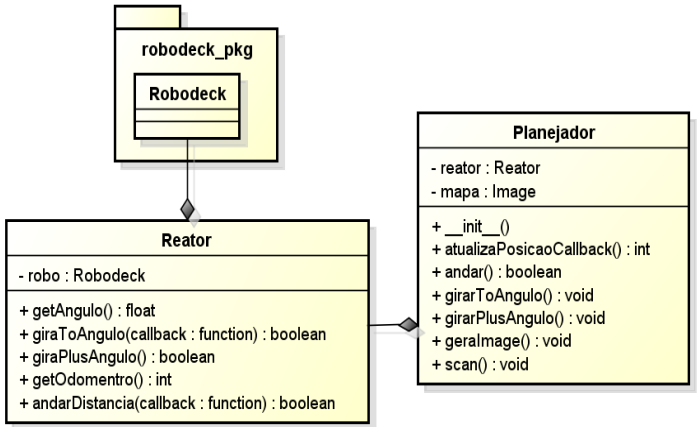
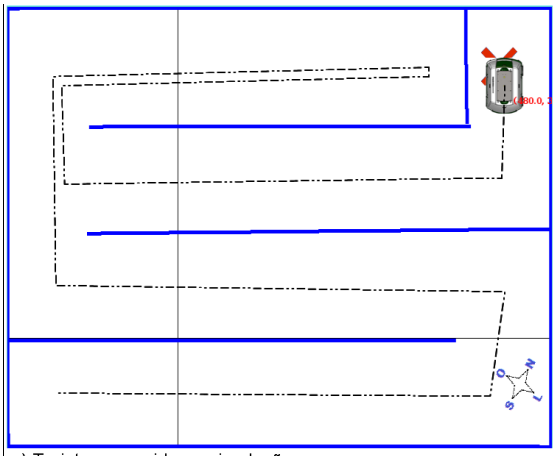


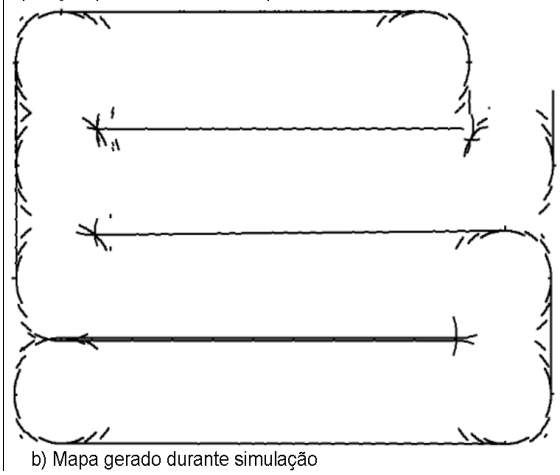
Figura 6 - Usando o módulo *robodeck_pkg* para implementar o *Planejador-Reator*.

Foi configurado no simulador, para esse exemplo em especifico, que o norte apontasse 60 graus com relação ao eixo X do sistema de coordenadas, o robô seria iniciado na posição -180 em X e -80 em Y e teria que percorrer um caminho com obstáculos em forma de labirinto até chegar a posição de coordenadas X=470 e Y=370, sendo esse seu objetivo.

O principal comportamento implementado no planejador foi a capacidade de guardar posições, se existir mais de uma entrada em seu percurso atual, podendo voltar caso chegar numa situação em que não há mais como seguir. A figura 7 (a) está apresentado um dos labirintos criado para o teste do robô, o qual foi adicionado o trajeto percorrido pelo robô para melhor visualização sendo que este não é gerado pelo simulador. Na figura 7 (b) representa o mapa (imagem) gerado durante a interação do robô com os obstáculos.



a) Trajeto percorrido na simulação



b) Mapa gerado durante simulação

Figura 7 - Mapeamento do ambiente percorrido pelo robô.

5 CONCLUSÕES

Este artigo apresentou um ambiente de simulação desenvolvido em Python utilizando a biblioteca Pygame para uso em aplicações tendo como inspiração a plataforma de robótica móvel Robodeck. Além disso foi implementada a arquitetura *Planejador-Reator* para percorrer labirintos e gerar um mapa do ambiente o qual está interagindo.

O ambiente desenvolvido tornou a produção de algoritmos relacionado ao comportamento do Rododeck mais ágil e interativo devido a capacidade do programador poder ver como seu código esta sendo tratado pelo robô, ganhado tempo e melhorando o trabalho em equipe.

Apesar do código gerado para o simulador não ser totalmente compatível para a plataforma (caso a linguagem python não seja usada), ainda assim é bastante aconselhado o seu uso devido a fácil migração de código.

Entre trabalhos futuros, prevê-se implementar os sensores restantes do Robodeck e também criar um ambiente tridimensional utilizando bibliotecas de física com modelagem de gravidade, colisões e inércia afim de aumentar seu potencial de simulação.

REFERÊNCIAS BIBLIOGRÁFICAS

JUNIOR, Valdir Grassi. *Arquitetura Híbrida para Robôs Móveis Baseada em Funções de Navegação com Interação Humana*. Tese de Doutorado. Universidade de São Paulo, 2006.

KERSHBAUNER, Ricardo; LIMA, Carlos Raimundo E. O Simulador de Robôs V-REP no Ensino das Técnicas de Controle de Robôs Autônomos. *COBENGE*, 2014.

KV, Noufal Ibrahim. Creating physics aware games using PyGame and PyODE. *Python Papers*, v. 5, n. 3, 2010.

LUTZ, Mark; ASCHER, David. *Aprendendo Python*. 2 ed. Porto Alegre, Bookman, 2007.

MARQUES, Diego Lopes et al. Atraindo Alunos do Ensino Médio para a Computação: Uma Experiência Prática de Introdução a Programação utilizando Jogos e Python, 2011.

MCGUGAN, Will. *Beginning Game Development with Python and Pygame*. Will McGugan, 2007.

MUNOZ, Mauro Enrique de Souza. Projeto do software do RobotDeck versao 0.2, 2011.

NETTO, Antonio Valerio; MIRANDA, Felipe Antunes; PINTO, Miguel Angelo Gaspar. Driver Control for Mobile Robot and application Control Trajectory with Computer Vision, 2012.

PISSARDINI, Rodrigo de Sousa. *Veículos autônomos de transporte terrestre: proposta de arquitetura de tomada de decisão para navegação autônoma*. Tese de Doutorado. Universidade de São Paulo, 2014.

PYGAME Community. 2016. Disponível em: <http://pygame.org>. Acesso em: 14 set. 2017.

ROBODECK. Apostila Mecatrônica versão 1.0., 2011.

SECCHI, Humberto Alejandro. Uma Introdução aos Robôs Móveis, 2008.

DE SOUZA, Marco Antonio Furlan; DENIS, Everson; FERNANDES, João Carlos Lopes. Projeto de um Console de Jogos Multiplataforma com Raspberry Pi. *Anais do Computer on the Beach*, p. 85-94, 2014.

SIEGWART, Roland; NOURBAKHSH, Illah Reza; SCARAMUZZA, Davide. *Introduction to autonomous mobile robots*. MIT press, 2011.

SWEIGART, Albert. *Making Games with Python & Pygame*. CreateSpace, 2012.

XBOT (Brasil). Robodeck versão 1.0 Manual do Usuário. 2010.

XBOT (Brasil). *XBot - Aprimoramento o ensino com tecnologia*}. 2014. Disponível em: <http://www.xbot.com.br/>. Acesso em: 14 set. 2017.