Uma Abordagem Integrada de Hardware e Software para o Aprendizado de Subsistemas de Entrada/Saída em Projetos de Sistemas Computacionais

Guilherme Álvaro Esmeraldo
LEDS/IFCE
Instituto Federal do Ceará
Crato, Brazil
guilhermealvaro@ifce.edu.br

Edson Barbosa Lisboa
LEA/IFS
Instituto Federal de Sergipe
Aracaju, Brazil
edson.lisboa@academico.ifs.edu.br

Luiz Fernando Morato
LEA/IFS
Instituto Federal de Sergipe
Aracaju, Brazil
luiz.morato0259@academico.ifs.edu.br

Cícero Samuel Mendes
LEDS/IFCE
Instituto Federal do Ceará
Crato, Brazil
mr.samuelmendes@gmail.com

Camila Valdez Ribeiro
LEA/IFS
Instituto Federal de Sergipe
Aracaju, Brazil
camila.ribeiro00@academico.ifs.edu.br

Milena Santos do Nascimento
LEA/IFS
Instituto Federal de Sergipe
Aracaju, Brazil
milena.nascimento068@academico.ifs.edu.br

Lucas Fontes Cartaxo
LEDS/IFCE
Instituto Federal do Ceará
Crato, Brazil
lfonteesc@gmail.com

Pedro Silva dos Santos
LEA/IFS
Instituto Federal de Sergipe
Aracaju, Brazil
pedro.santos061@academico.ifs.edu.br

***Resumo*—O estudo dos conceitos relacionados à interação do sistema com o ambiente externo é um dos pontos mais importantes no aprendizado em projetos de sistemas computacionais modernos, tais como os da Internet das Coisas, Robótica e Automação. Esses conceitos são tão amplos e complexos, pois envolvem aspectos como interfaces, canais e protocolos de comunicação, bem como dispositivos de entrada/saída, que são tratados de forma superficial em aulas predominantemente teóricas e, na literatura, não possui o suporte adequado dos simuladores computacionais, os quais têm sido amplamente utilizados para apoio à aplicação prática dos conceitos teóricos. Este artigo apresenta um novo recurso do Subsistema de Entrada/Saída do simulador computacional CompSim, que conta com uma abordagem integrada de hardware e software, para suportar os processos de aprendizagem prática no projeto da interação entre o sistema computacional e seus respectivos periféricos. O recurso proposto inclui uma interface de entrada/saída padronizada que automatiza a conexão de novos periféricos, sendo eles desenvolvidos em software ou com uso de componentes eletrônicos físicos, ao sistema computacional simulado. A abordagem proposta tem sido empregada em diferentes cursos e os resultados mostram que, ao permitir simulação e construções práticas de periféricos, aumenta-se o potencial do processo de ensino-aprendizagem em subsistemas de Entrada/Saída.**

Palavras-chave—projeto de sistemas computacionais, subsistemas de entrada/saída, simulador computacional, apoio ao aprendizado

I. INTRODUÇÃO

Em cursos técnicos e superiores nas áreas de computação e engenharia eletrônica, há disciplinas que tratam de aspectos de projetos de sistemas computacionais, tal como a de Arquitetura e Organização de Computadores (AOC)[1][2]. Nessas disciplinas, os estudantes adquirem conhecimentos e habilidades para projetar novos sistemas computacionais, apontar e diagnosticar problemas relacionados ao desempenho de sistemas completos ou subsistemas, bem como determinar soluções para otimizar seu uso e desempenho. Geralmente, os conteúdos trabalhados nas disciplinas são extensos e com um alto grau de detalhes, pois abordam os aspectos estruturais e funcionais dos componentes do computador, a comunicação deles entre si, e até mesmo a sua programação em linguagens de alto e de baixo nível.

Dentre esses conteúdos, a interação do sistema computacional com o ambiente externo é um dos mais importantes. Os sistemas computacionais modernos, como são os casos da Internet das Coisas (IoT), Computação de Alto Desempenho, Automação e Robótica, considerando as características e funcionalidades das atividades para os quais foram projetados, necessitam realizar interações com o ambiente e/ou com outros sistemas computacionais. Essa interação envolve diversos elementos como interfaces, barramentos, modelos e protocolos de comunicação, módulos de entrada/saída e os próprios periféricos. O aprendizado desses conceitos, em específico, é uma atividade considerada complexa e não trivial, devido ao emprego de métodos de ensino que limitam-se à realização de aulas predominantemente teóricas e/ou aos altos níveis de abstração das ferramentas de software utilizadas para apoiar as aulas práticas, não possibilitando a exploração dos detalhes técnicos que os subsistemas de entrada/saída reais apresentam [3]. É importante ressaltar que existe a necessidade de mais estudos para a educação nessa área [7] e que é um campo importante não só para o projeto de sistemas computacionais, mas também de sistemas operacionais, linguagens de programação e interação homem-computador.

Com o uso de ferramentas de software, surge um outro problema, que é a ausência de exposição dos estudantes ao hardware físico, onde o contato com os componentes eletrônicos físicos incentiva o aprendizado de todas as fases de

desenvolvimento de um sistema computacional e/ou periférico [10]. Desta forma, um cenário ideal de aprendizado em entrada/saída envolve o uso de laboratórios de hardware especializados, com suporte de técnicos de laboratório experientes e a disponibilidade dos materiais a serem utilizados nas aulas práticas. No entanto, é fato que tais laboratórios não são comumente encontrados na grande maioria das instituições de ensino.

Na literatura, tem-se optado pela utilização de simuladores para apoio ao aprendizado dos diferentes aspectos de projetos de sistemas computacionais [4]. Os simuladores são ferramentas que buscam representar cenários reais no projeto de sistemas computacionais e têm como principais benefícios a abstração dos diferentes recursos do computador, não necessitarem de laboratórios de hardware e técnicos especializados, bem como permitem simplificar a configuração de diferentes cenários de projetos e feedback rápido nas simulações [14]. Alguns trabalhos, como os apresentados em [5][7][13][15], realizaram comparações entre diferentes simuladores, buscando estabelecer critérios para adoção de algum deles para apoio ao aprendizado nos conceitos de projetos de sistemas computacionais. Entre os critérios de comparação, destacam-se: granularidade (nível de detalhes, opções de configuração e variabilidade de componentes de hardware); desempenho e precisão dos resultados da simulação; suporte à simulação de aplicações de usuário; e interfaces gráficas interativas.

Entre os simuladores computacionais, podem ser encontrados aqueles que buscam abstrair os detalhes dos componentes de hardware visando aumentar o desempenho das simulações, os que abordam desde componentes específicos de hardware, que buscam fidelizar as características dos componentes do computador para tratar de conceitos específicos ou mais avançados com maiores detalhes [6] – como são os casos dos simuladores de processadores e de memórias cache [18] –, até os de sistemas completos [5], que trazem uma visão macro das funcionalidades e de comunicação entre componentes.

No tocante aos Subsistemas de Entrada/Saída, o tema ou não é tratado adequadamente pelos simuladores, por conta das abstrações adotadas que fazem com que os conceitos fundamentais sejam tratados vagamente, ou não é abordado [20]. Assim, outros trabalhos incluem simuladores que atuam ao nível de hardware, como os descritos em linguagem de descrição de hardware (HDL) [17], que visam abordar diferentes níveis de abstração no projeto de um sistema computacional, e os que utilizam componentes físicos de hardware [10][11][12][14], os quais fazem com que os estudantes sejam motivados a produzirem sistemas reais que podem fazer uso de periféricos para interação com o ambiente. No entanto, alguns estudos, como os apontados nos trabalhos em [16] e [17], demonstram que a utilização de hardware físico, em práticas laboratoriais, pode apresentar desvantagens, tais como: alta demanda de tempo para configuração e montagem dos experimentos, considerando o curto período de tempo das aulas práticas em laboratório; dependendo do cenário de projeto, pode-se necessitar de maiores habilidades técnicas em eletrônica; com o tempo, há o desgaste de seus componentes eletrônicos; dependência de vários componentes eletrônicos complementares que podem elevar rapidamente o custo; e, estudantes que não possuem o nível adequado de conhecimentos em eletrônica podem projetar circuitos sem os devidos cuidados técnicos e, desta forma, podem danificá-los e/ou a seus módulos complementares.

Este artigo apresenta um novo recurso do subsistema de entrada/saída do simulador CompSim [14], o qual vem com a proposta integrada de hardware e software de apoiar o aprendizado e exploração, de forma prática, da interação entre o sistema computacional e seus respectivos periféricos. Esse recurso consiste de uma interface de entrada/saída padronizada que permite conectar, de forma simplificada e automatizada, o sistema computacional simulado a diferentes tipos de periféricos, com objetivo de apoiar a realização de experimentos práticos que envolvam desde o projeto de periféricos virtuais, desenvolvidos apenas em software, e físicos, desenvolvidos em hardware e software. Com isso, espera-se atender aos requisitos dos processos de ensino-aprendizagem em subsistemas de entrada/saída, em diferentes cursos, disciplinas e níveis de formação.

II. Fundamentação Teórica

Dispositivos de entrada/saída são componentes fundamentais em sistemas computacionais, pois possibilitam a interface para a interação entre o sistema e o ambiente exterior. Tais interações podem se configurar entre homem-máquina (IHM), máquina-máquina (IMM) e de comunicação [8]. Portanto, há diferentes dispositivos de entrada/saída, que apresentam grandes variações com relação às suas funcionalidades, tecnologias empregadas na sua constituição, taxas de transmissão de dados, além de natureza, volume e formato dos dados na comunicação. Essa heterogeneidade inviabiliza que, em sistemas computacionais, o processador possa se comunicar e tratar diretamente as particularidades de cada dispositivo periférico. Assim, o projeto de Subsistemas de Entrada/Saída é uma solução integrada de hardware e software, que deve ser adotada para cada dispositivo periférico específico, para que este se conecte e se comunique com o restante do sistema.

De uma forma geral, um Subsistema de Entrada/Saída é subdivido em Módulo de Entrada/Saída, também conhecido como Controlador, e o próprio periférico, também conhecido como dispositivo externo. A Fig. 1 ilustra esses dois elementos e como estão relacionados.

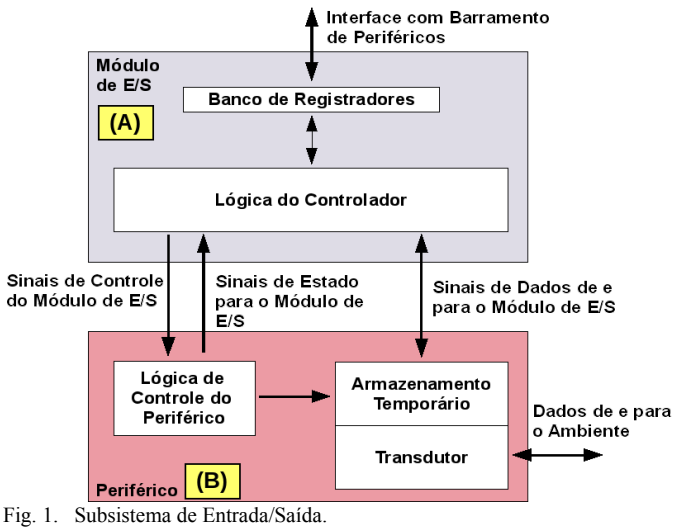

Fig. 1. Subsistema de Entrada/Saída.

O primeiro, o Módulo de Entrada/Saída, que pode ser visto na Fig. 1-A, é um componente de hardware que faz interface com o barramento de conexão com periféricos, também chamado de barramento de periféricos, e possui: 1) Banco de Registradores, utilizados para armazenar dados, informações de controle, comandos e estados; e 2) Lógica do Controlador das funções do periférico, responsável por traduzir as requisições do processador em comandos para controle do periférico e converter as respostas do periférico em formatos de dados que podem ser compreendidos pelo processador.

Já o periférico, ilustrado na Fig. 1-B, contém: 1) Lógica de Controle do Periférico, que se encarrega de interpretar os comandos enviados pelo Módulo de Entrada/Saída e faz com que a função relativa ao comando seja executada; 2) Transdutor, fundamental para realizar as devidas conversões entre sinais de diferentes naturezas (e.g. conversão entre sinal elétrico proveniente do sistema computacional em uma forma de energia compatível com o ambiente externo, tais como ondas magnéticas ou sonoras); e 3) Armazenamento temporário, que está associado ao Transdutor para que os dados possam ser transferidos entre o periférico e o Módulo de Entrada/Saída.

O projeto dos Subsistemas de Entrada/Saída pode variar de acordo a natureza, funcionalidades, tecnologias empregadas e complexidade do periférico. Desta forma, percebe-se que o estudo, compreensão e desenvolvimento de periféricos não são atividades imediatas. Assim, abordagens alternativas têm surgido como mecanismos para otimizar tanto os processos de ensino-aprendizagem quanto para aumentar o desempenho no desenvolvimento de software e hardware, em projetos de dispositivos de entrada/saída, tal como será apresentado a seguir.

III. Materiais e Métodos

O CompSim é desenvolvido na linguagem Python versão 3.x, com *framework* gráfico Tkinter, os quais permitem a distribuição do simulador em diferentes plataformas operacionais, tais como MS Windows e GNU/Linux, e em arquiteturas de 32 e 64-bits[1]. Com esse suporte, é possível aumentar a abrangência de sua adoção/uso em diferentes instituições e cursos, bem como por docentes e estudantes.

Em versões anteriores, o CompSim contava com periféricos integrados ao sistema computacional simulado, Entre os periféricos disponíveis, haviam aqueles do tipo virtual, implementados em software e que emulam o comportamento de periféricos reais (e.g. Vídeo e Teclado), e físico, que inclui uma interface em software para suportar a comunicação do sistema computacional simulado com uma *board* da plataforma aberta de prototipação Arduino UNO [9]).

Mesmo com o suporte dos periféricos virtuais e físico, observou-se a necessidade de ampliar o espectro de possibilidades de estudos em Subsistemas de Entrada/Saída, dadas as diferentes naturezas, comportamentos, tecnologias empregadas e modelos de comunicação dos periféricos. Nesse sentido, a partir de um levantamento teórico aprofundado sobre o tema de Entrada/Saída em projetos de sistemas computacionais, realizou-se estudos de técnicas de programação (Programação Orientada a Objetos, Padrões de Projeto, Componentes de Software e Arquiteturas de Software), em que concebeu-se o modelo de interface, aqui apresentado, para permitir a conexão simplificada e automatizada de diferentes periféricos, sejam eles virtuais ou físicos.

IV. Simulador CompSim

CompSim é um simulador computacional, que segue a abordagem de projetos baseados em plataforma [19], no qual há uma plataforma de hardware simulável customizável, conhecida por "Mandacaru", que inclui os principais componentes do computador. São eles: 1) CPU: um processador de 16-bits, chamado de "Cariri", que é baseado em acumulador, com arquitetura mista (RISC e CISC), composta por 16 instruções para realização de operações de transferência de dados, lógicas e aritméticas, desvio de fluxo de programa e de entrada/saída com periféricos. Possui ainda os seguintes submódulos: banco de registradores, contador de instrução, unidade de controle, unidade lógica e aritmética e subsistema de tratamento de interrupções de software e hardware; 2) Memória cache: é utilizada para otimizar o desempenho dos programas (aproveitando as características de localidade espacial e temporal) e suporta diferentes tipos de configuração, como de técnicas de mapeamento e políticas de substituição, entre outras; 3) Memória RAM: é utilizada para armazenamento de dados, instruções e pilha dos programas que serão executados no simulador; 4) Barramentos: o simulador inclui dois barramentos, sendo um de sistema, que permite a comunicação entre o processador e as memórias Cache e RAM, e um de periféricos, que permite que o processador se comunique com o Subsistema de Entrada/Saída; e 5) Subsistema de Entrada/Saída: inclui uma interface padronizada com módulos de entrada/saída, que possibilita a comunicação do processador com diferentes tipos de periféricos.

O simulador também conta com uma interface gráfica para dar suporte à configuração dos componentes virtuais da plataforma Mandacaru, ajustar parâmetros de simulação e suportar a codificação de aplicações em baixo nível (Assembly). A Fig. 2 mostra a interface gráfica do CompSim, na qual pode-se visualizar os seguintes componentes gráficos: A) Editor de código: inclui recursos para simplificar a codificação de uma aplicação, como número de linhas, teclas de atalho para recursos de edição (*copy*, *cut*, *paste*, *do/undo*, *got to line*, etc.), um assistente de codificação (permite auxiliar a construção de instruções de código com a sintaxe correta), destaque de palavras-chave (*syntax highlight*), entre outros. Este componente está integrado a um montador (*Assembler*) que realiza análises léxica, sintática e semântica no código-fonte da aplicação, tradução deste para *bytecodes*, carregamento dos *bytecodes* na memória RAM, além de gerar um relatório do programa, que inclui a tabela de símbolos, endereços de memória dos segmentos e *bytecodes* gerados; B) Processador: durante uma simulação, exibe os registradores do processador Cariri e respectivos valores assumidos. Os registradores de endereçamento possuem cores diferenciadas, onde as respectivas cores são utilizadas para indicar as diferentes posições referenciadas na memória RAM; C) Memória cache: exibe as linhas da cache e respectivos valores, destaca também as linhas e as palavras endereçadas pelo processador durante uma simulação; D) Memória RAM: exibe os conteúdos (instruções e dados) de todos os endereços do componente virtual memória RAM (os componentes gráficos Memória RAM e processador estão integrados, de forma que, durante uma simulação, as posições de memória são destacadas

1 http://compsim.crato.ifce.edu.br/

de acordo com as respectivas cores dos registradores de endereçamento); e E) Componentes de controle de configuração e execução de simulação: inclui controles que permitem configurar o tempo total de simulação e a frequência de relógio de sistema (*clock*), iniciar, executar (em modos normal, passo a passo ou rápido), parar e reiniciar uma simulação.

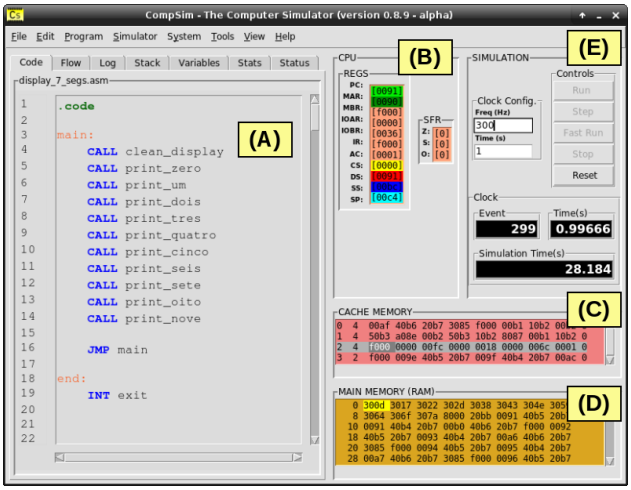

Fig. 2. Interface Gráfica do Simulador CompSim.

O CompSim também inclui outros componentes de visualização, que são: 1) *Logs*: apresenta, durante uma simulação, os eventos e notificações gerados por todos os componentes da plataforma; 2) *Stack*: permite acompanhar os estados da pilha do programa (dados adicionados e endereçamento do apontador de topo da pilha), durante uma simulação; 3) *Variables*: permite visualizar os valores assumidos pelas variáveis e estruturas de dados do programa, durante uma simulação; e 4) *Stats*: após o encerramento de uma simulação, apresenta gráficos estatísticos que sumarizam os eventos gerados pela CPU, Memórias RAM e Cache, bem como barramentos (e.g. são apresentados os tipos e quantidades de instruções executadas pela CPU, contabilizações de blocos lidos e escritos na memória RAM, totais de *hits* e *misses* de cache, operações de entrada/saída realizadas no barramento de periféricos, entre outros).

A. *Suporte de Operações de Entrada/Saída*

O processador Cariri, presente na plataforma virtual de hardware do CompSim, possui, em seu conjunto de instruções da arquitetura, a instrução "INT", que é utilizada, dentre outras opções, para realização de operações de entrada/saída. Para tanto, a instrução necessita, como parâmetros, dos códigos de operação 20 e 21 para leitura e escrita de dados a partir de e para um determinado periférico, respectivamente.

As operações de entrada/saída devem atender às seguintes condições: 1) Em operações de leitura/escrita, os 8 bits mais significativos do registrador acumulador (AC) serão utilizados para endereçamento de periféricos. Com isso, podem ser endereçados até 256 portas diferentes (8 bits $\rightarrow 2^8 = 256$ portas); 2) Em operações de escrita, os 8 bits menos significativos do registrador acumulador serão utilizados para guardar o byte que será escrito em algum periférico; e 3) Em operações de entrada, após a leitura de dados de periférico, o dado lido estará disponível no registrador acumulador, nos 8 bits menos significativos, caso seja um byte, ou, nos 16 bits do acumulador, caso seja um número inteiro. Desta forma, nas operações de entrada/saída, o registrador acumulador do processador Cariri passa a incluir os campos "AC High" (utilizado para endereçamento de periférico) e "AC Low" (utilizado para transferência de dados), como podem ser vistos na Fig. 3.

Fig. 3. Campos do registrador acumulador em operações de Entrada/Saída.

A Fig. 4 apresenta um exemplo de escrita de um caractere "A" em dispositivo de saída de vídeo.

```
1   .code
2      LDA a
3      ADD video
4      INT output
5
6      INT halt
7
8   .data
9      a:      DB 'A'
10     video:  DD 0x200
11     output: DD 21
12     halt:   DD 25
13
14  .stack 10
```
Fig. 4. Exemplo de operação de saída.

No exemplo da Fig. 4: nas linhas 1, 8 e 10, estão contidos os delimitadores de segmentos de código, dados e pilha do programa, respectivamente; no segmento de dados, na linha 9, está definido o caractere que será enviado ao periférico, na linha 10 define-se a porta de endereçamento do periférico de vídeo, na linha 11 está definido o código que será utilizado pela instrução "INT" para realizar uma operação de saída; no segmento de código, na linha 2, a instrução "LDA" carrega o caractere "A" no registrador acumulador (neste caso, será carregado o respectivo valor inteiro ASCII do caractere em "AC Low"), na linha 3 acrescenta-se ao acumulador a porta de endereçamento do periférico de video (o número da porta será acrescentado ao registrador acumulador no campo "AC High") e, na linha 4, será realizada a operação de saída, através da instrução "INT" com o parâmetro 21, contido na variável "output".

B. *Subsistema de Entrada/Saída*

O Subsistema de Entrada/Saída, presente na plataforma de hardware virtual do CompSim, permite que novos periféricos sejam conectados automaticamente ao barramento de periféricos. Para tanto, o projeto de cada dispositivo de entrada/saída deve incluir dois arquivos: 1) o primeiro (arquivo com extensão ".csd") contém uma especificação de sua interface, a qual inclui as seguintes definições: número(s) da(s) porta(s) de entrada/saída; número da interrupção (IRQ), caso o periférico suporte este tipo de comunicação; dados para instanciar o componente de software (nome do pacote e da classe de software); e uma curta descrição textual do respectivo periférico; 2) o segundo arquivo inclui o programa que será utilizado para implementar o comportamento do periférico (o código-fonte do programa, com extensão ".py", deve ser

descrito na linguagem Python v3.x, para manter compatibilidade com o CompSim).

Desta forma, ao se criar uma plataforma computacional no simulador CompSim, seu Subsistema de Entrada/Saída buscará pelos arquivos de descrição de interface de componente (arquivo ".csd") e, de forma automatizada, instanciará os respectivos componentes de software (arquivo ".py"), que serão conectados ao barramento virtual de periféricos (esses processos são ilustrados na Fig. 5-A e 5-B, respectivamente). Com isso, os novos periféricos já estarão prontos para interação com os demais componentes de hardware do sistema computacional simulado.

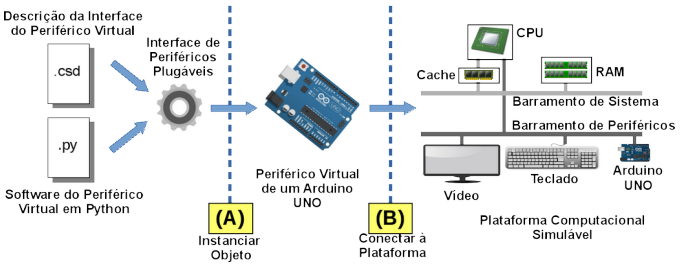

Fig. 5. Criação de um periférico plugável e conexão à plataforma de hardware virtual do CompSim.

C. *Assistente de Criação de Interface de Periféricos*

O CompSim inclui um assistente para dar suporte à criação de novos periféricos, chamado de "Device Interface Creator" (ou simplesmente "DICreator") (ilustrado na Fig. 6). Basicamente, ele consiste de um formulário, que deve ser preenchido, pelo projetista, com os dados da interface do novo periférico e de criação do respectivo componente de software. Com essas informações, o assistente gera automaticamente os respectivos arquivos ".csd" e ".py", ou, em outras palavras, cria o respectivo Módulo de Entrada/Saída do novo dispositivo de entrada/saída. Deste ponto em diante, para concluir a criação, o projetista deve apenas focar no desenvolvimento do comportamento funcional do periférico, ao complementar sua codificação no arquivo-fonte ".py".

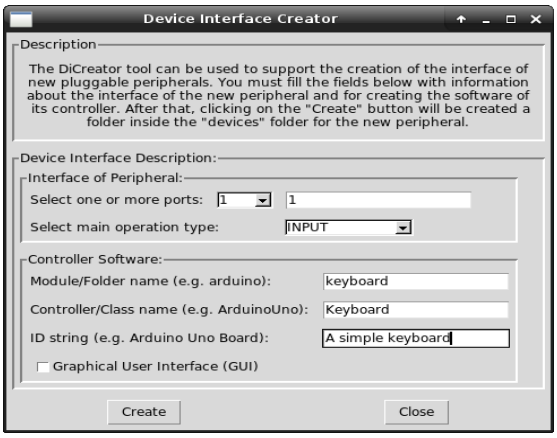

Fig. 6. DICreator: assistente para criação de interface de periféricos.

O cenário apresentado na Fig. 6 pode consistir no projeto de um novo periférico que simula um teclado ("keyboard"). Esse periférico poderá ser utilizado para que o projetista interaja com a aplicação do sistema computacional, durante uma simulação, através de entrada de dados (por exemplo, strings digitadas no próprio teclado do computador onde está sendo executado o simulador). Na descrição da interface, observa-se que o periférico será endereçado na porta 1 e que não suportará comunicação por interrupções (provavelmente, o objetivo seja utilizar a técnica de comunicação *polling*). Já do ponto de vista do projeto do software, a classe que implementará o controlador do novo periférico se chamará "SimpleKeyboard", ela estará contida no módulo "keyboard" e não terá interface gráfica (provavelmente, o projetista deseja utilizar um terminal de comandos, por exemplo, para digitação das strings).

A Fig. 7 apresenta o arquivo ".csd" gerado com o assistente de criação de interfaces de periféricos a partir do cenário apresentado na Fig. 6. No arquivo, pode-se ver que: o periférico será endereçado na porta 1 (linha 2); ele não suportará comunicação por interrupções (linha 3); os arquivos gerados estarão localizados no diretório "keyboard" (linha 4); o código-fonte para criação do periférico estará no módulo de software chamado de "keyboard" (linha 5); a classe que instanciará o objeto de software do periférico se chamará "SimpleKeyboard" (linha 6); e, por último, a linha 7 inclui uma string com a descrição do novo periférico.

```
1  [Device]
2  ports = 1
3  irq =
4  folder = keyboard
5  module = keyboard
6  controller = SimpleKeyboard
7  identification = A simple keyboard
```

Fig. 7. Arquivo "keyboard.csd" gerado com o assistente DICreator.

Na Fig. 8, é possível ver o arquivo ".py" gerado pelo assistente de criação de interface de periféricos, a partir do cenário apresentado na Fig. 6.

```
1  from device import INPUT_DEVICE
2  from queue import Queue
3
4  class SimpleKeyboard(INPUT_DEVICE):
5    def __init__(self, master=None):
6      self.gui_log = Queue()
7      print("Device SimpleKeyboard created!")
8
9    def read(self, data):
10     self.gui_log.put("INPUT DEVICE: has read something")
11     data = int(format(data, "016b")[8:16], 2)
12
13 if __name__ == "__main__":
14   c = SimpleKeyboard()
```

Fig. 8. Arquivo "keyboard.py" gerado com o assistente DICreator.

No arquivo, pode-se ver que: nas linhas 1 e 2, serão importadas, respectivamente, a classe de software "INPUT_DEVICE", que consiste da classe base para periféricos que realizam entrada de dados, e a fila de comunicação ("Queue"), utilizada pelo periférico, durante uma simulação, para informar eventos e/ou notificações gerados por ele, que serão exibidos na interface gráfica do simulador CompSim (mais especificamente, no componente gráfico *Logs*); na linha 4, a definição da classe "SimpleKeyboard", que implementará o novo periférico (observar que ela herda os atributos e métodos da classe "INPUT_DEVICE"); nas linhas 5 a 7, está o construtor da classe, onde, na linha 6, será criada a fila de comunicação; nas linhas 9 a 11, é definido o método "read", que implementará o comportamento de leitura (entrada)

de dados do periférico (observar na linha 10 que, sempre que o periférico realizar uma operação de entrada, será adicionada uma notificação na fila de comunicação, e, na linha 11, a variável "data", que tem uso opcional, pode ser utilizada como uma entrada de controle do periférico); e, por fim, nas linhas 13 e 14, pode-se instanciar o software do periférico para testes durante sua implementação (este recurso permite isolar o desenvolvimento do periférico, durante esta fase, não necessitando do simulador CompSim).

D. Dispositivos de Entrada/Saída Plugáveis

Atualmente, o CompSim conta oficialmente com duas categorias de dispositivos de entrada/saída: 1) virtuais, que são aqueles desenvolvidos apenas em software e emulam o comportamento de periféricos reais, e 2) físicos, que incluem uma interface de software, para integração com CompSim, e componentes físicos de hardware, utilizados para confeccionar um periférico real.

Entre os dispositivos virtuais de entrada/saída oficiais, há Teclado, Vídeo e Arduino UNO [9], como podem ser vistos nas Figs. 9-A, 9-B e 9-C, respectivamente.

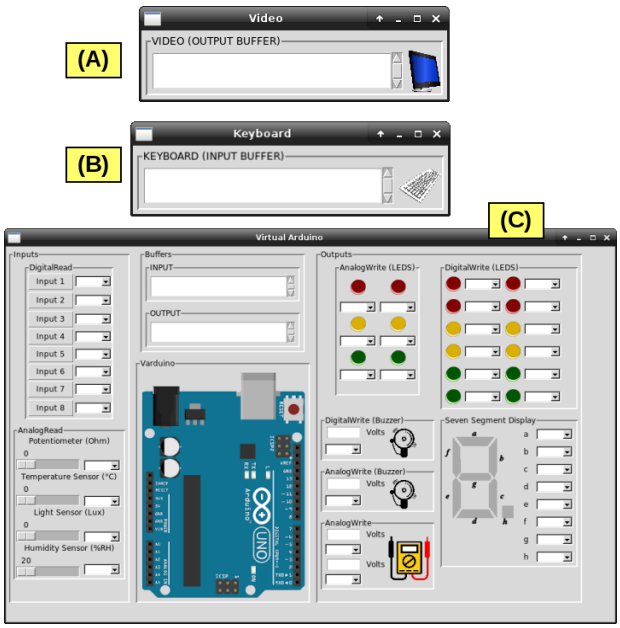

Fig. 9. Periféricos Virtuais do CompSim.

O dispositivo Vídeo é utilizado para impressão de strings (saída de dados) pela aplicação, durante uma simulação. Já o dispositivo Teclado é utilizado para entrada de dados, sendo que ele possui duas versões, em que diferem pelo mecanismo de comunicação com o processador, sendo um por polling e o outro por interrupções. Nesses componentes, a interface gráfica inclui campos de texto para exibir os estados dos *buffers* de *input* ou *output*, dados lidos ou escritos durante as operações de comunicação.

O Arduino UNO virtual, também chamado de "VArduino", apresentado na Fig. 9-C, é composto por um Arduino UNO virtual e por diferentes componentes gráficos que emulam dispositivos eletrônicos reais. A interface gráfica está dividida em três macroblocos de componentes gráficos, onde: do lado esquerdo, estão os componentes de entrada de dados, que emulam 8 botões (*push buttons*), 1 potenciômetro e 3 sensores analógicos (temperatura, luminosidade e umidade); ao centro, estão contidos os componentes gráficos que apresentam os *buffers* de entrada (*input*) e saída (*output*), usados na comunicação entre o simulador CompSim e o VArduino, e, logo abaixo, há um diagrama do Arduino UNO, utilizado para consulta de portas e de numeração de pinos de comunicação; por fim, à direita, estão os componentes de saída de dados, que emulam 18 LEDs e 2 *Buzzers* (campainhas), os quais podem ser utilizados em operações de escrita digital/analógica, um *Display* de 7 segmentos e dois dispositivos genéricos de escrita analógica, que podem simular, por exemplo, o acionamento de *coolers*/exaustores de computadores, acionar e controlar o movimento de motores, gerar sinais modulados para controle de um LED infravermelho (e.g. para uso como controle remoto), gerar efeitos sonoros, entre outros.

Entre os dispositivos físicos de entrada/saída oficiais, há um *Buzzer*, Arduino UNO e Arduino MEGA [9], como podem ser vistos nas Figs. 10-A, 10-B e 10-C, respectivamente.

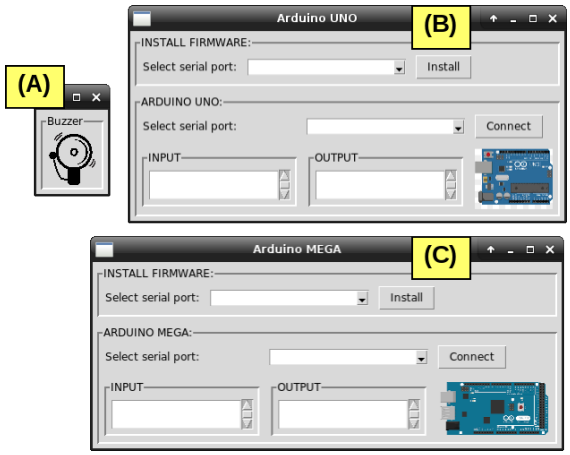

Fig. 10. Periféricos físicos do CompSim.

Os dispositivos físicos de entrada/saída, como citado anteriormente, são divididos em dois submódulos, sendo que: o primeiro implementa, em software, o Módulo de Entrada/Saída e permite integração do periférico ao barramento de periféricos da plataforma virtual; e o segundo submódulo é o periférico propriamente dito.

No caso do *Buzzer*, o periférico físico consiste do alto-falante do computador que executa o simulador. Nos demais, os periféricos consistem das respectivas *boards* físicas das plataformas abertas de prototipação Arduino UNO e Arduino MEGA. A interface gráfica desses dois últimos inclui mecanismos nos quais: é possível selecionar a porta serial de comunicação (interface USB) entre o computador e a *board* (e.g. "COM4", no sistema operacional MS Windows, e "/dev/ttyUSB0", no GNU/Linux); instalar o *firmware* e realizar a conexão lógica da *board*, via protocolo serial, para que seja possível a integração entre o simulador CompSim e a *board*; e visualizar os estados dos *buffers* de entrada/saída, durante uma simulação (os periféricos Arduino UNO e MEGA podem operar tanto para entrada quanto para saída de dados digitais e analógicas).

A Tabela I ilustra quais recursos estão disponíveis no dispositivo de entrada/saída Arduino UNO, com respectivos modos de operação. Na tabela, as portas B e D, que possuem interface com os pinos de uma *board* Arduino UNO, rotulados de 8 a 13 e de 2 a 7, respectivamente, podem ser utilizadas para operações de entrada ou saída digitais (e.g. acender LEDS ou leitura de estados de *push buttons*). A porta C, com pinos rotulados de A0 a A5, pode ser utilizada para leitura de dados analógicos (e.g. leituras de dados aferidos por sensores). Já as portas PWM, com pinos 3, 5, 6, 9, 10 e 11, podem ser utilizadas para saída analógica (e.g. acionar motores), através da técnica de *Pulse Width Modulation*.

TABELA I. SERVIÇOS DO DISPOSITIVO DE ENTRADA/SAÍDA ARDUINO UNO.

Recurso	Modo de Operação
Porta B (pinos 8 a 13)	Entrada/Saída Digital
Porta D (pinos 2 a 7)	Entrada/Saída Digital
Porta C (pinos A0 a A5)	Entrada Analógica
Porta PWM (pino 3)	Saída Analógica
Porta PWM (pino 5)	Saída Analógica
Porta PWM (pino 6)	Saída Analógica
Porta PWM (pino 9)	Saída Analógica
Porta PWM (pino 10)	Saída Analógica
Porta PWM (pino 11)	Saída Analógica

Para que estas operações estejam disponíveis, é necessário que na definição do arquivo ".csd", referente à interface do dispositivo de entrada/saída Arduino UNO, cada recurso esteja endereçado em uma porta diferente. A Fig. 11 mostra o arquivo de descrição da interface do dispositivo de entrada/saída Arduino UNO, no qual pode-se observar, na linha 2, o uso dos números de portas de endereçamento de 2 a 10.

```
1  [Device]
2  ports = [2,3,4,5,6,7,8,9,10]
3  irq =
4  folder = arduino_uno
5  module = arduino_uno
6  controller = ArduinoUNO
7  identification = device Arduino UNO
```

Fig. 11. Arquivo "arduino_uno.csd" com descrição da interface do dispositivo Arduino UNO.

Com essa definição de interface, a implementação do comportamento do dispositivo de entrada/saída Arduino UNO (arquivo ".py") considerará que, dependendo da porta endereçada e do tipo de operação (entrada ou saída), será acionado um determinado recurso da *board* física.

Da mesma forma, a Tabela II e a Fig. 12 ilustram os recursos, e respectivos modos de operação, do dispositivo de entrada/saída Arduino MEGA, bem como seu arquivo de definição de interface ".csd".

TABELA II. SERVIÇOS DO DISPOSITIVO DE ENTRADA/SAÍDA ARDUINO MEGA.

Recurso	Modo de Operação
Porta A (pinos 22 a 29)	Entrada/Saída Digital
Porta B (pinos 50 a 53,10 a 13)	Entrada/Saída Digital
Porta C (pinos 30-37)	Entrada/Saída Digital
Porta D (pino 38)	Entrada/Saída Digital
Porta G (pinos 39 a 41)	Entrada/Saída Digital
Porta L (pinos 42 a 49)	Entrada/Saída Digital
Porta F (pinos A0 a A7)	Entrada Analógica
Porta K (pinos A8 a A15)	Entrada Analógica
Porta PWM (pino 2)	Saída Analógica
Porta PWM (pino 3)	Saída Analógica
Porta PWM (pino 4)	Saída Analógica
Porta PWM (pino 5)	Saída Analógica
Porta PWM (pino 6)	Saída Analógica
Porta PWM (pino 7)	Saída Analógica
Porta PWM (pino 8)	Saída Analógica
Porta PWM (pino 9)	Saída Analógica
Porta PWM (pino 10)	Saída Analógica
Porta PWM (pino 11)	Saída Analógica
Porta PWM (pino 12)	Saída Analógica
Porta PWM (pino 13)	Saída Analógica
Porta PWM (pino 44)	Saída Analógica
Porta PWM (pino 45)	Saída Analógica
Porta PWM (pino 46)	Saída Analógica

```
1  [Device]
2  ports = [2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,
   22,23,24]
3  irq =
4  folder = arduino_mega
5  module = arduino_mega
6  controller = ArduinoMEGA
7  identification = device Arduino MEGA
```

Fig. 12. Arquivo "arduino_mega.csd" com descrição da interface do dispositivo Arduino MEGA.

V. DISCUSSÃO DOS RESULTADOS

O Simulador CompSim vem sendo utilizado por estudantes de diferentes turmas de um curso de Técnico em Eletrônica e de um Bacharelado em Sistemas de Informação. Com o Subsistema de Entrada/Saída do CompSim, os estudantes podem lidar de maneira prática com os conceitos de entrada/saída na criação de diferentes tipos de periféricos virtuais (software) e físicos (software e hardware), bem como com aspectos de sua programação, através da criação de aplicações computacionais interativas e de *device drivers*.

Com os dispositivos físicos de entrada/saída, é possível criar cenários de projetos de sistemas computacionais completos, em que se pode realizar a simulação da plataforma computacional virtual (CPU, memórias RAM e Cache, e barramentos) interagindo com hardware real, com o apoio dos

Arduinos UNO e MEGA. Ressalta-se que, a opção de criação de dois dispositivos de entrada/saída derivados da especificação aberta do Arduino, deve-se ao fato de que o Arduino MEGA, por possuir maiores recursos de hardware, tais como maior capacidade de armazenamento das memórias *Flash*, RAM e EEPROM, bem como maior número de pinos de GPIO (*General Purpose Input/Output*), é mais indicado para projetos de periféricos com maiores requisitos de espaço de programação e armazenamento, bem como de conectividade.

Desta maneira, os estudantes puderam criar, de forma simplificada, diferentes periféricos físicos, para interação do usuário ou do meio externo com os programas em execução no simulador, tais como: saída digital e analógica em LEDS, uso de displays de 7 segmentos, entrada de dados digitais pelo uso de teclados numéricos e chaves tácteis, e entrada analógica pelo uso de potenciômetros e sensores de luz (LDR), temperatura (LM35) e umidade do solo (higrômetro, que é composto por uma sonda e um módulo com comparador LM393). Além destes, com suporte de um FPGA (*Field Programmable Gate Array*) [17], foi possível criar periféricos digitais, tais como contadores digitais e registradores dos tipos *buffer*, de deslocamento e de carregamento paralelo de dados, os quais foram conectados a um Arduino (UNO ou MEGA) para interação com sistema computacional virtual simulável.

Já os dispositivos virtuais de entrada/saída são particularmente interessantes quando não há disponibilidade de hardware físico para as práticas laboratoriais, pois, mesmo com essa carência, pode-se criar cenários que se assemelham aos reais. Além disso, ao abstrair o uso de dispositivos eletrônicos, pode-se eliminar assim os custos na aquisição e manutenção dos componentes eletrônicos, bem como reduzir os ônus no aprendizado dos conceitos básicos de eletrônica e no manuseio desses dispositivos. É nesse contexto que surgiu a proposta do VArduino, um dispositivo virtual de entrada/saída que possui comportamento semelhante a uma *board* física do Arduino UNO. Além de emular as operações do Arduino UNO (escritas e leituras digitais e analógicas), ele inclui uma gama de componentes eletrônicos virtuais, os quais podem ser facilmente conectados ao Arduino UNO virtual com apoio de controles gráficos do tipo *combobox*. Desta forma, é possível produzir rapidamente experimentos que fazem uso de LEDS, *push buttons*, potenciômetros, sensores, *Display* de 7-segmentos e *Buzzers*.

Cabe ressaltar que os programas criados em Assembly e que serão executados na plataforma de hardware virtual são completamente portáveis entre o uso dos dispositivos virtual VArduino e físico do Arduino UNO. Isso significa que pode-se, inclusive, estabelecer um fluxo de projeto de subsistema de entrada/saída que, por exemplo, na fase de projeto da interface de interação, no primeiro instante, pode-se fazer uso do dispositivo virtual VArduino para validar os aspectos funcionais e de comunicação de entrada/saída, para, posteriormente, substituí-lo pelo dispositivo físico Arduino UNO e respectivos componentes eletrônicos reais, permitindo a validação dos requisitos do circuito eletrônico em si, tais como custo, consumo de energia e de espaço físico, bem como desempenho.

Apesar de todo o potencial no aprendizado e projeto de novos dispositivos de entrada/saída, ainda há restrições de desempenho na interação entre a plataforma de hardware virtual com os periféricos físicos. Um dos exemplos identificados, pelos estudantes, consiste no projeto de um sistema computacional que faz uso do sensor ultrassônico HC-SR04, como periférico. Este sensor é composto de dois componentes, um emissor e um receptor de ultrassom, em que o primeiro é responsável por enviar pulsos de ultrassom, os quais, ao colidirem com algum obstáculo, são refletidos e capturados pelo receptor. Neste estudo de caso, em que se utilizou uma plataforma de hardware virtual, com dispositivo de entrada/saída Arduino UNO fazendo interface com o sensor HC-SR04, verificou-se que o desempenho da simulação não permitiu que o sistema computacional simulado conseguisse realizar a captura dos pulsos, em tempo, devido à alta frequência de operação do sensor (em torno de 40 kHz).

VI. Conclusões

Este artigo apresentou uma proposta de uso do simulador CompSim para suporte ao aprendizado prático de conceitos de Subsistemas de Entrada/Saída em projetos de sistemas computacionais. O simulador proposto inclui uma plataforma virtual de hardware que contém um Subsistema de Entrada/Saída, o qual permite conectar, de forma automatizada, a ela, novos dispositivos de entrada/saída, sendo eles virtuais ou físicos. Os dispositivos virtuais são aqueles que emulam, em software, componentes de hardware reais. Já os dispositivos físicos incluem uma interface de software, para conexão com a plataforma de hardware virtual do simulador, e o periférico físico (composto por componentes eletrônicos reais).

Durante as práticas laboratoriais, os estudantes são estimulados a aplicar os conceitos aprendidos nas aulas teóricas na criação de diferentes dispositivos de entrada/saída, utilizando diferentes tecnologias, tais como a linguagem de programação Python e componentes eletrônicos (e.g. *boards* do Arduino UNO, resistores, LEDS, *push buttons*, *displays* e sensores). A abordagem proposta tem sido empregada em diferentes cursos e os resultados mostram que, ao permitir simulação e construções práticas de periféricos, aumenta-se o potencial do processo de ensino-aprendizagem em subsistemas de Entrada/Saída.

Como trabalhos futuros, pretende-se: realizar estudos com o objetivo de aumentar o desempenho de simulação da plataforma de hardware virtual, de forma que ela possa suportar uma gama maior de componentes eletrônicos, tal como o sensor de ultrassom HC-SR04; a criação de novos dispositivos de E/S, incluindo periféricos digitais com suporte de FPGA, e estender as funcionalidades dos existentes, tal como adicionar suporte aos protocolos I^2C, CAN, SPI e Serial, bem como conversores ADC/DAC ao periférico Arduino UNO; adicionalmente, elaborar cenários de projetos de sistemas computacionais, tornando o aprendizado dos conceitos de Subsistemas de Entrada/Saída mais diversificado, abrangente, motivante e dinâmico.

Referências

[1] ACM. Association for Computing Machinery and IEEE Computer Society, “Curriculum Guidelines for Undergraduate Degree Programs in Computer Science,” 2013.

[2] A. F. Zorzo, D. Nunes, E. Matos, I. Steinmacher, J. Leite, R. M. Araujo, R. Correia and S. Martins, “Referenciais de formação para os cursos de graduação em computação,” in Sociedade Brasileira de Computação (SBC), 2017.

[3] E. Larraza-Mendiluze and N. Garay-Vitoria, “Approaches and tools used to teach the computer input/output subsystem: A survey,” in IEEE Transactions on Education, v. 58, n. 1, pp. 1-6, 2014.

[4] S. R. Fernandes and I. S. Silva, “Relato de Experiência Interdisciplinar Usando MIPS,” in International Journal of Computer Architecture Education (IJCAE), V. 6, n. 1. pp. 52-61, 2017.

[5] P. H. Penna and H. C. Freitas, “Análise e Avaliação de Simuladores de Sistemas Completos para o Ensino de Arquitetura de Computadores”, in International Journal of Computer Architecture Education, V. 2, N.1. pp. 13-16, 2013.

[6] L. Duenha and R. Azevedo, “Utilização dos Simuladores do MPSoCBench para o Ensino e Aprendizagem de Arquitetura de Computadores,” in International Journal of Computer Architecture Education (IJCAE), V. 5, n. 1. pp 26-31, 2016.

[7] G. A. R. M. Esmeraldo, C. S. R. Mendes, L. F. Cartaxo and E. B. Lisboa, “Apoio ao Aprendizado em Arquitetura e Organização de Computadores: Um Estudo Comparativo entre Simuladores Computacionais,” in Revista Tecnologias na Educação, v. 31, pp. 1-17, 2019.

[8] W. Stallings, “Computer Organization and Architecture. 10th Edition,” Pearson, 2017.

[9] A. Kurniawan, "Introduction to Arduino Boards and Development," in Arduino Programming with. NET and Sketch, Apress, pp. 1-19, 2017.

[10] M. Black, “Export to arduino: a tool to teach processor design on real hardware,” in Journal of Computing Sciences in Colleges, 31(6), pp.21-26, 2016.

[11] A. M. A. Neto, J. A. dos S. Borges and G. P. Silva, “Extensão do Simulador SimuS com uso do Protocolo Firmata,” in XVIII Workshop de Iniciação Científica do XVII Simpósio em Sistemas Computacionais de Alto Desempenho (WIC-WSCAD), pp.123-128, 2017.

[12] R. A. Hexsel and R. Carmos, “cMIPS – uma Ferramenta Pedagógica para o Estudo de Arquitetura,” in International Journal of Computer Architecture Education, v.2, n.1, pp. 29 -32, 2013.

[13] B. Nikolic, Z. Radivojevic; J. Djordjevic and V. A. Milutinovic, “Survey and Evaluation of Simulators Suitable for Teaching Courses in Computer Architecture and Organization,” in IEEE Transactions on Education, Vol. 52, No. 4, 2009.

[14] G. A. Esmeraldo and E. B. Lisboa, “Uma Ferramenta para Exploração do Ensino de Organização e Arquitetura de Computadores”, in International Journal of Computer Architecture Education, v.6, n.1, pp. 68 -75, 2017.

[15] P. F. Gonçalves and J. Duraes, “An Arduino Simulator for Practical Embedded Programming Teaching,” in International Symposium on Ambient Intelligence and Embedded Systems, 2019.

[16] T. Katayama, T. Nishida, Y. Kita, H. Yamaba, K. Aburada and N. Okazaki, “Implementation of Arduino Simulator ADVIS Visualizing the Value of Voltage on the Circuit,” in Journal of Robotics, Networking and Artificial Life, 5(4), pp. 249-252, 2019.

[17] M. Awedh and A. Mueen, "Teaching Computer Organization Using Field Programmable Gate Array: An Incremental Approach," in Asian Journal Of Advanced Basic Sciences, vol.4, n.1, pp. 5-11, 2016.

[18] M. A. S. Xavier, J. C. Rodrigues and O. A. L. Júnior, “Simuladores de Memória Cache, um Estudo Comparativo Direcionado ao Ensino,” in Workshop sobre Educação em Arquitetura de Computadores (WEAC 2011), pp. 7-12, 2011.

[19] K. Keutzer, A. R. Newton, J. M. Rabaey and A. Sangiovanni-Vincentelli, "System-level design: orthogonalization of concerns and platform-based design," in IEEE transactions on computer-aided design of integrated circuits and systems, 19(12), pp. 1523-1543, 2000.

[20] G.. Esmeraldo, E. Barbosa, L. F. Cartaxo, C. S. R. Mendes, “Aprendizado Prático de Subsistemas de Entrada/Saída em Projetos de Sistemas Computacionais com Suporte do Simulador CompSim,” in Comunicações em Informática, v. 4, pp. 15-19, 2020.