

APRENDIZAGEM ATIVA PARA SISTEMAS EMBARCADOS: UMA ABORDAGEM COM SIMULINK/STATEFLOW

Pedro Ivo de Oliveira Tironi – pedroivoexatas@gmail.com
Gustavo Lobato Campos – gustavo.lobato@ifmg.edu.br
Patrick Santos de Oliveira – patrick.oliveira@ifmg.edu.br
Instituto Federal de Minas Gerais, IFMG – Campus Formiga
Rua Padre Alberico, 440 – Bairro São Luiz
35570-010 – Formiga – Minas Gerais, MG

Resumo: Este artigo tem como objetivo apresentar um tutorial dos passos bases de desenvolvimento de sistemas embarcados, por meio da plataforma Arduino em conjunto ao software Simulink. Destaca-se ainda o intuito de contribuir para formação de futuros engenheiros, por meio de uma alternativa à usualmente empregada no ensino da área de microcontroladores. Este tema é baseado nas dificuldades encontradas pelos alunos na elaboração de um raciocínio lógico estruturado para a solução de problemas computacionais, na compreensão de conceitos e na complexidade/sintaxe dos algoritmos, que são frequentemente obstáculos para o aprendizado. Problemas que podem ser suprimidos pelo uso de alternativas a programação convencional (bottom-up) por lógicas visuais (blocos). Estas representam, por meio de estados e transições, toda a complexa cadeia de funções e estruturas contidas em um algoritmo. Alicerçado a este tópico, o texto apresenta ao leitor os conceitos chave para desenvolvimento de sistemas embarcados com duas ferramentas da MathWorks®: Simulink e Stateflow. Através de uma aprendizagem ativa é exposto um exemplo real, de forma que a partir deste o leitor se torne hábil a desenvolver diversos sistemas embarcados conforme seu interesse. A aplicação do Arduino a todo conceito envolvido conduz o leitor a um aprendizado facilitado por ser uma ferramenta de fácil uso.

Palavras-chave: Sistemas Embarcados. Simulink. Stateflow. Tutorial. Aprendizagem Ativa.

1 INTRODUÇÃO

Estima-se que dentre todos os novos processadores fabricados atualmente (BARR, 2020), apenas 2% compõem partes de novos computadores, os outros 98% tornaram-se sistemas embarcados (SEs). Fato dado em virtude que todo dispositivo eletrônico moderno, desde brinquedos, semáforos, a até controladores de pouso em espaçonaves são desenvolvidos a partir de um SE (BARR, 2020). Estes processadores são embarcados em sistemas para que gerenciem fábricas, fluxo de pessoas, de informações e produtos, sendo que cada sistema opera em uma tarefa em específico (TIRONI *et al*, 2018).

Basicamente, um SE é um sistema microcontrolado que interage com dispositivos periféricos de entrada ou saída (HEATH, 2003). No contexto dos SEs existem plataformas microcontroladas como o Arduino, que se destaca por ter código aberto e ferramentas que facilitam seu uso. Suas placas são capazes de “ler” entradas por meio de sensores (KRISHNA, 2020) e através de um processamento interno transformar a informação de entrada em uma saída, por meio de atuadores (ARDUINO, 2020).

Logo, percebe-se cada vez mais a importância do tema na formação de engenheiros associados principalmente às áreas de eletrônica, elétrica, automação e controle, assim como dos cientistas e engenheiros da computação, sendo fundamental o domínio de ferramentas que

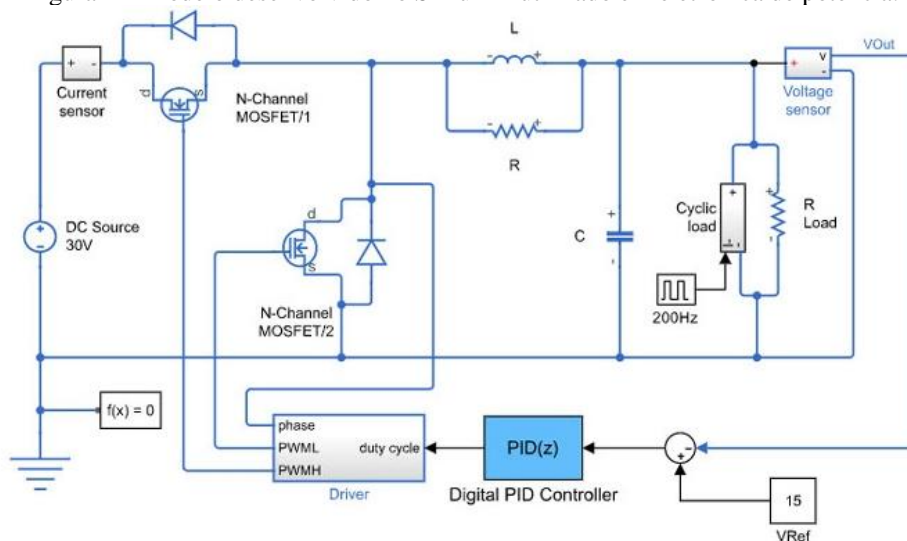
possibilitem o desenvolvimento de SEs. Em contrapartida, estudos (MOREIRA *et al*, 2018), (SALES *et al*, 2014), (CAMPOS, 2010), (GOMES, 2010) e (FERREIRA *et al*, 2010) demonstram a dificuldade de alunos realizarem a interpretação e abstração de soluções e aplicá-las com o uso de linguagens de programação, aumentando-se cada vez mais a evasão escolar, além de um baixo desempenho geral. Conforme Moreira (2018), dentre as principais dificuldades para a compreensão da programação como um todo 42,72% dos estudantes afirmaram possuir maior dificuldade no desenvolvimento da lógica de programação, 34,54% entendimento da sintaxe e somente 15,45% dificuldade na interpretação dos problemas.

Neste ponto de vista, os autores deste trabalho apresentam uma alternativa ao desenvolvimento convencional de algoritmos de programação que são base na elaboração de SEs. Alternativa descrita aqui em forma de tutorial a fim de proporcionar ao leitor conhecimentos chaves que possibilitem a programação em Arduino por meio de diagramas de blocos nas plataformas Simulink/Stateflow da MathWorks®. Observa-se também a aplicação dos conceitos de Aprendizagem Ativa, onde motiva-se o aluno a ser protagonista de seu aprendizado. Portanto, pretende-se contornar os principais obstáculos presentes na dificuldade de programação com uma alternativa por meio da programação em blocos e estados, a fim de possibilitar ao leitor uma nova ferramenta em substituição as tradicionalmente empregadas.

2 SIMULINK\STATEFLOW

O Simulink é um produto complementar ao *software* MATLAB, no qual, apresenta uma interface gráfica, interativa e intuitiva, em um ambiente de modelagem computacional (SIMULINK..., 2019). Proporciona ao usuário a elaboração rápida de protótipos virtuais (que podem ser embarcados nas mais diversas plataformas) para explorar conceitos de *design* em qualquer nível de abstração e detalhamento (SIMULINK..., 2019). Seu ambiente de trabalho oferece uma interface gráfica do usuário para construir modelos através de diagramas de blocos, conforme apresenta a Figura 1 onde é desenvolvido um modelo de eletrônica de potência.

Figura 1 – Modelo desenvolvido no Simulink utilizado em eletrônica de potência.



Fonte: (SIMULINK, 2020).

Dentro do sistema inclui uma biblioteca de blocos predefinidos para construir modelos gráficos por comandos arrastar e soltar com o *mouse*. Conforme apresentado em (SIMULINK,

2019) os alunos aprendem eficientemente com *feedback* frequente do *software*, a natureza interativa do Simulink incentiva os usuários a testarem as mais diversas soluções.

Além destas funcionalidades, existe o ambiente de trabalho Stateflow que é uma *toolbox* que pode ser adicionada ao Simulink para proporcionar ao desenvolvedor alternativas ao desenvolvimento de sistemas discretos (WHAT..., 2020). Sua programação é basicamente construída por estados e transições, fato que facilita ao desenvolvedor, pois não é necessário decorar sintaxes de programação.

3 TUTORIAL DE DESENVOLVIMENTO DE SISTEMA EMBARCADO

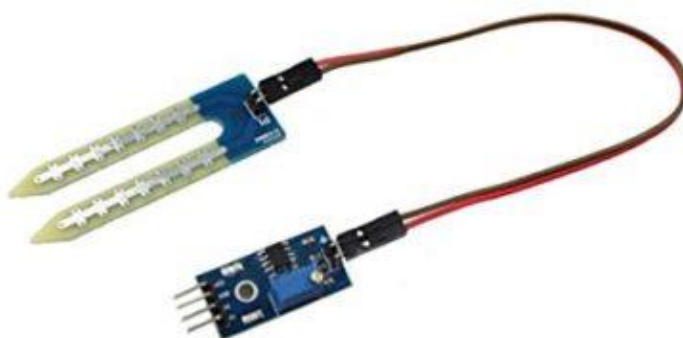
3.1 Descrição do problema

Nesta seção será apresentado um tutorial do desenvolvimento de um sistema embarcado em uma aplicação específica, para tanto, será dada uma introdução ao problema considerado.

Dados mostram que a produção das lavouras irrigadas é consideravelmente mais significativa quando comparadas com lavouras sem irrigação adequada (BARBOSA, 2013). Com o intuito de melhorar a produtividade de uma lavoura será projetado um sistema de irrigação automático, onde por meio de um sensor será captada a umidade do solo e o sistema microcontrolado irá identificar a necessidade de irrigação (SENSOR, 2020).

Para realizar a captação dos dados referentes a umidade do solo devem ser utilizados sensores de umidade (*moisture*), conforme exemplifica a Figura 2. Este é composto por duas sondas responsáveis por medir a quantidade de volume de água no solo, e criam uma corrente elétrica que permite mensurar um parâmetro de tensão de forma analógica. De modo que, quanto menor a tensão, menor será a quantidade de água no solo (BARBOSA, 2013).

Figura 2 – Sensor de umidade de solo.



Fonte: (SOIL, 2020).

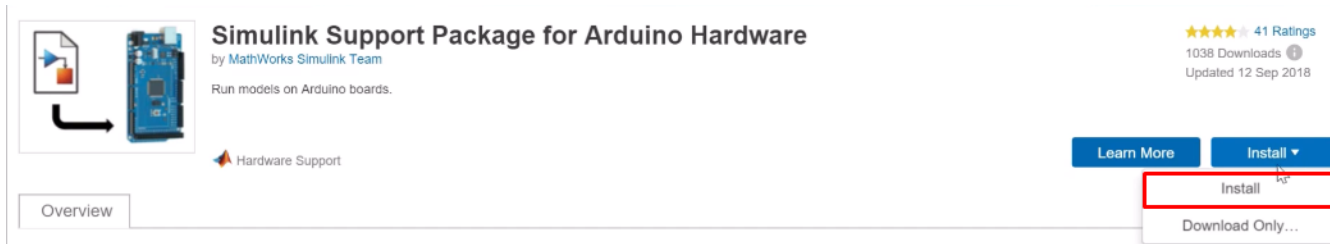
No exemplo apresentado, sem perda de generalidade, será empregado um potenciômetro para representar este sensor. É válido ressaltar que tanto o sensor quanto o potenciômetro, é aplicada uma tensão de 0 a 5V, que será conectada a entrada analógica do Arduino, convertido internamente para valores de 0 a 1023 (2^{10}), por meio de seu conversor A/D.

A saída deste sistema será responsável por acionar um relé que fará o controle da parte de potência. Em escala de prototipagem será acionado um sinal luminoso (LED) a fim de representar os diferentes estados de acionamento.

3.2 Complementos do MATLAB

No MATLAB primeiramente é necessário instalar os complementos a serem utilizados na programação. Na tela inicial na aba *Add-ons* selecione o menu *Get Hardware Support Packages*, logo após escolha o complemento *Simulink Support Package for Arduino Hardware*, conforme apresentado na Figura 3, e realize a instalação no computador.

Figura 3 – Adicionando complemento de integração entre Simulink e Arduino.

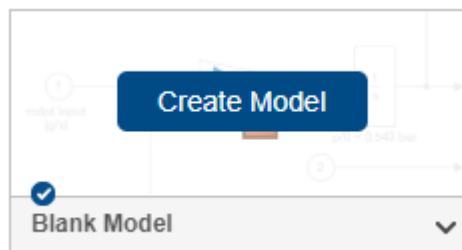


Fonte: Próprio do Autor.

3.3 Criando design virtual

Na área de trabalho inicial do MATLAB deve-se selecionar o ícone Simulink inserido na aba *Home*. O software irá inicializar uma nova interface de trabalho (*Simulink Start Page*), dentro desta será possível criar um modelo em branco conforme apresenta a Figura 4.

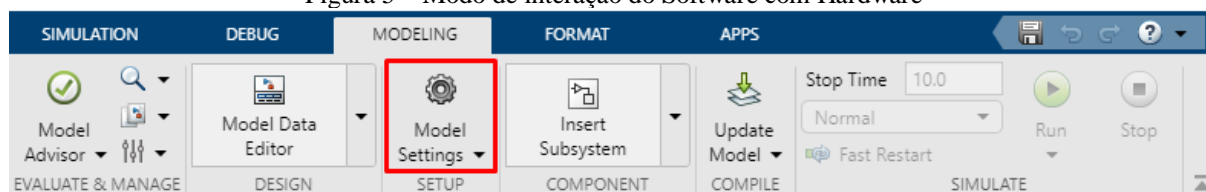
Figura 4 – Criando um modelo Simulink em branco.



Fonte: Próprio do Autor.

Uma nova tela será aberta, sendo este o ambiente de trabalho. Antes de editar o documento é necessário configurar o modelo para a placa de Arduino a ser utilizada. Isto é feito ao alternar entre as abas de configurações até *Modeling* e posteriormente em *Setup* selecione o campo com figura de engrenagens nomeado *Model Settings*, a Figura 5, demonstra o local exato para configuração.

Figura 5 – Modo de interação do Software com Hardware

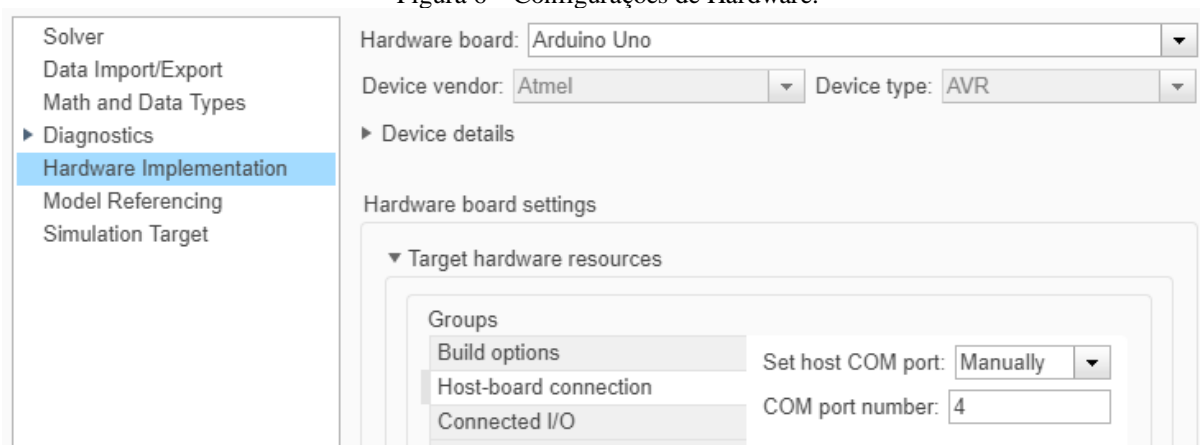


Fonte: Próprio do Autor.

Aberto o menu de configuração selecione *Hardware Implementation* e configure os campos com o seu microcontrolador. A Figura 6, apresenta um caso particular aonde foi

conectado um Arduino Uno e foi configurado manualmente a sua porta de comunicação (COM4).

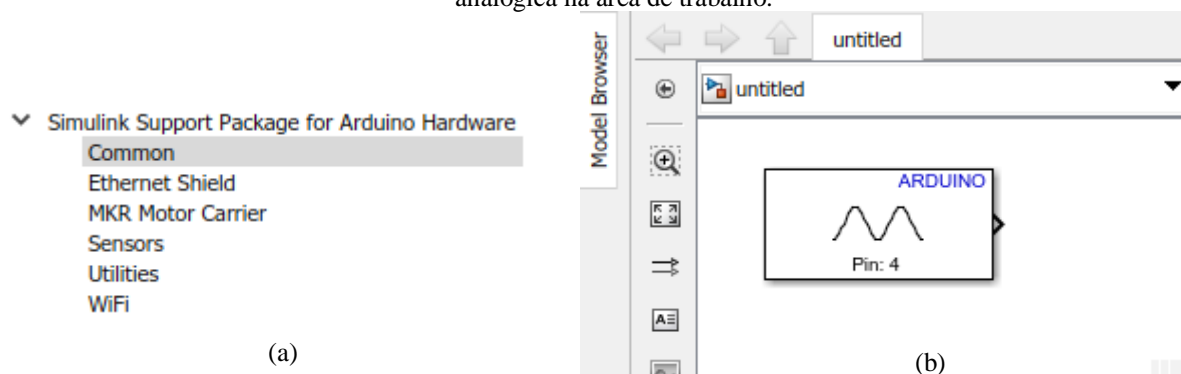
Figura 6 – Configurações de Hardware.



Fonte: Próprio do Autor.

Com todas as configurações feitas é possível iniciar a modelagem do SE em questão. Primeiramente, deve-se inserir um bloco para receber dados do potenciômetro. Para isto, na aba *Simulation* clique em *Library Browser* e escolha o menu que contém os blocos de suporte para Arduino, conforme apresenta a Figura 7 - a. Identifique o bloco *Analog Input*, através do mouse é possível com um clique sobre o mesmo arrastá-lo para a área de trabalho, no local desejado solte o bloco, a Figura 7 - b representa o resultado deste procedimento.

Figura 7 - (a) Opções de blocos de desenvolvimento do pacote Arduino e (b) Inserido o bloco de leitura analógica na área de trabalho.

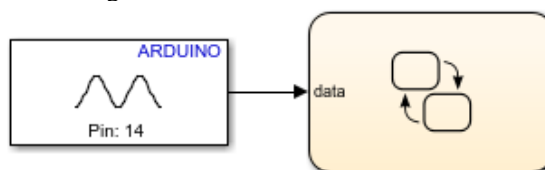


Fonte: Próprio do Autor.

Com um clique duplo sobre o bloco é permitido editar suas configurações para adequá-las ao problema. Para isto no campo *Pin number*, insira a porta 0 que representa a porta A0 do Arduino Uno. Para mais detalhes sobre as portas, o *software* disponibiliza um link dentro do próprio bloco com explicações (em inglês) de pinos dos mais diversos modelos de Arduino.

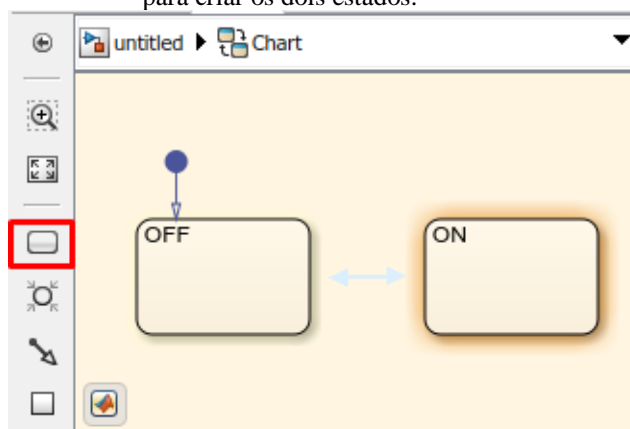
Para controle deste sistema é necessário adicionar o bloco *chart* que está presente na *Library Browser* inserido no menu Stateflow. Adicionado os dois blocos na área de trabalho é possível com o comando arrastar e soltar conectá-los, conforme demonstra a Figura 8.

Figura 8 – Conexão entre dois blocos.



Fonte: Próprio do Autor.

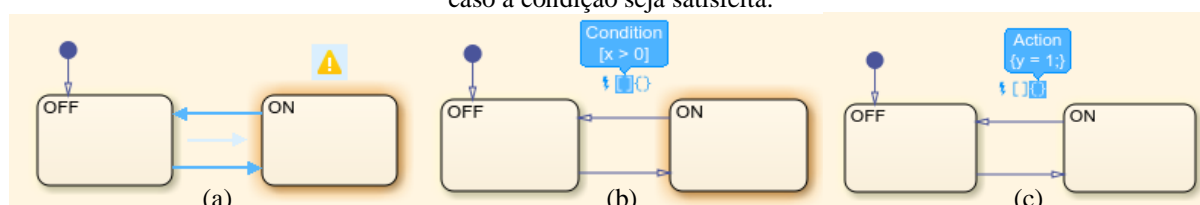
Com um clique duplo sobre o bloco *chart* entra-se no ambiente de trabalho do Stateflow. Neste ambiente cria-se dois estados, que são *OFF* e *ON*, este procedimento é feito ao arrastar o bloco com nome *State* para a área de trabalho e com um clique duplo é possível renomeá-lo-á, conforme apresenta a Figura 9.

 Figura 9 – Desenvolvido dois estados para o sistema, é destacado em vermelho o bloco *State* utilizado para criar os dois estados.


Fonte: Próprio do Autor

Após criado os estados é necessário inserir as *transitions* para o sistema transitar entre os *states*, isto é feito conforme apresenta a Figura 10 – a. Para adicionar uma condição na transição o texto deve ser redigido entre colchetes, conforme apresenta a Figura 10 – b, o significado disto é que para o sistema mover de *OFF* para *ON* ele deve satisfazer aquela condição. Outra configuração existente conforme apresenta a Figura 10 – c é a ação, ou seja, caso a transição aconteça irá ocorrer uma determinada ação que estiver entre chaves.

Figura 10 – (a) transições entre ações do sistema, (b) inserindo condição para transições e (c) inserindo uma ação caso a condição seja satisfeita.

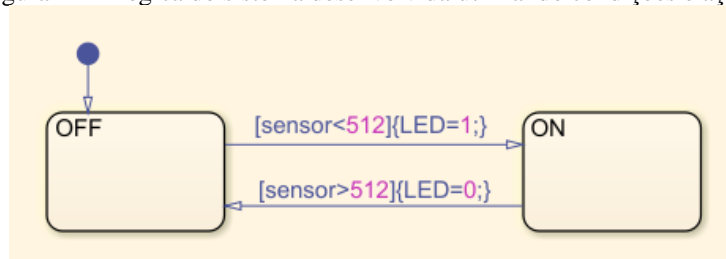


Fonte: Próprio do Autor.

No problema exemplificado, que trata da modelagem de um sistema de irrigação, o sistema irá ter seu primeiro estado em *OFF* (seta com círculo apontando para o estado *OFF*). E caso a quantidade do volume de água no solo apresente uma resposta menor do que 512 o sistema irá fazer a ação de ligar o sistema de irrigação (representado pelo estado do LED igual a 1). Caso

o sistema esteja ligado (estado *ON*) e apresente um nível satisfatório de umidade do solo (maior que 512) então o sistema irá realizar uma transição para estado *OFF* que tem como ação desligar a irrigação (estado do LED igual a 0). Todos estes passos são compilados na Figura 11.

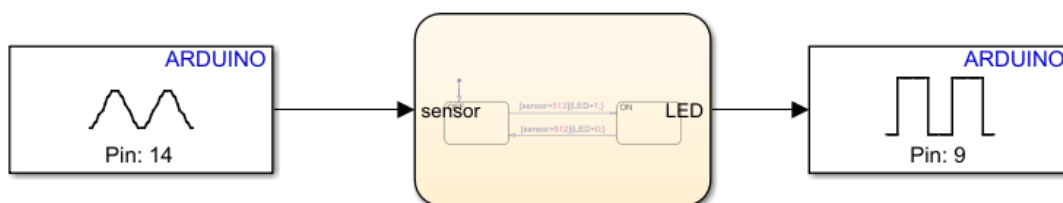
Figura 11 – Lógica do sistema desenvolvida utilizando condições e ações.



Fonte: Próprio do Autor.

Configurada toda a lógica de controle do sistema, o usuário retorna ao ambiente de trabalho do Simulink. Neste ambiente, é necessário conectar uma saída ao bloco *chart* do Stateflow, para isto na *Library Browser* dentro dos componentes seleciona-se o bloco *Digital Output*, e configura-se *Pin: 9*. O resultado deste procedimento é exposto na Figura 12.

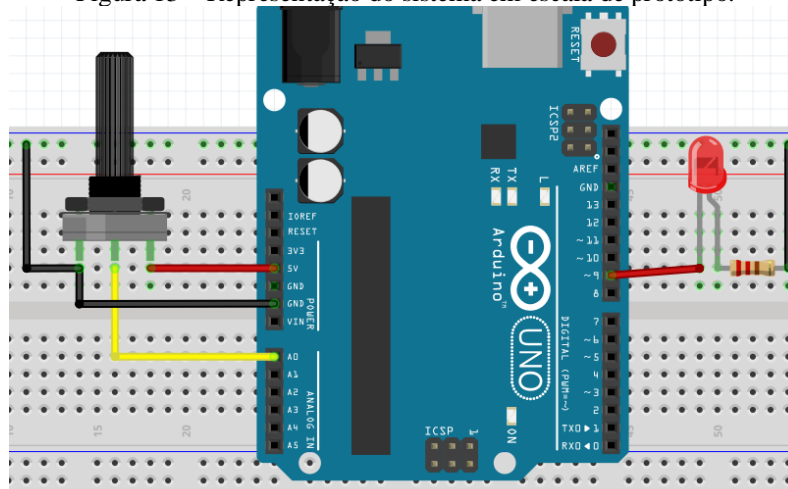
Figura 12 – Conectada a saída ao sistema de controle.



Fonte: Próprio do autor.

Portanto, o sistema está apto para ser embarcado no microcontrolador. Para isto, é necessário salvar o modelo e pressionar o botão *Run*. Mas antes disto, deve-se elaborar o circuito eletrônico que irá representar o sistema real, conforme apresentado na Figura 13 desenvolvido no *software* Fritzing (gratuito).

Figura 13 – Representação do sistema em escala de protótipo.

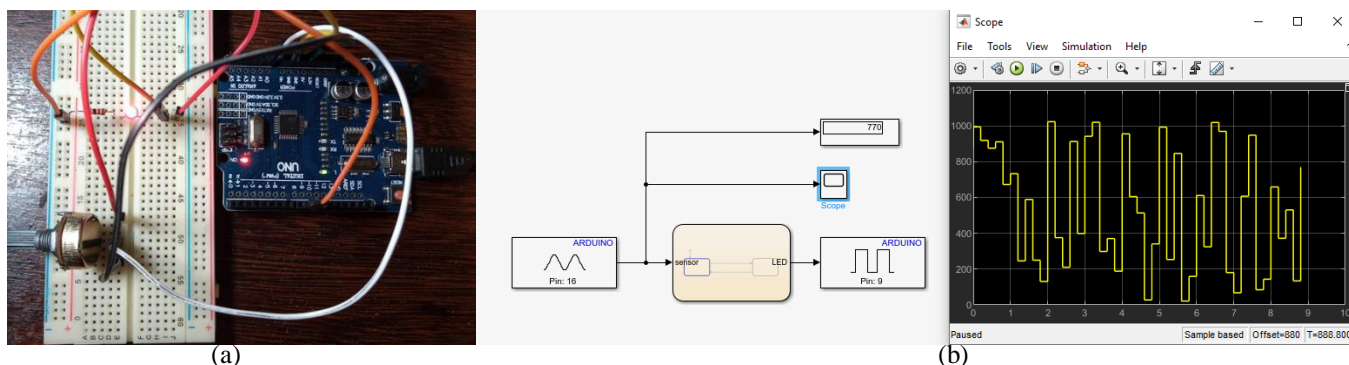


Fonte: Próprio do Autor.

3.4 Resultados e validação

Ao simular o sistema pode-se verificar seu funcionamento tanto através de ferramentas de visualização inseridas no *hardware*, apresentado na Figura 14 – a, quanto em *software* como demonstra a Figura 14 - b.

Figura 14 – (a) Sistema em funcionamento na *protoboard* e (b) *Software* em funcionamento interagindo o protótipo, varia-se o fator quantidade de água no solo através do potenciômetro e visualiza-se através do *Scope*.



Fonte: Próprio do Autor.

O sistema responde à medida que o usuário gira o potenciômetro, fato que pode ser visualizado pela ferramenta *Scope* do Simulink. E o estado que o LED se encontra é possível ser visualizado no bloco *chart*, apresentando em bordas azul o estado atual.

4 CONSIDERAÇÕES FINAIS

O presente trabalho teve como objetivo trazer uma alternativa ao desenvolvimento de algoritmos de programação aliado ao uso em sistemas embarcados, no qual apresenta-se a ferramenta Simulink/Stateflow da MathWorks. O conteúdo foi desenvolvido de forma que o leitor pudesse realizar o mesmo exemplo sem dificuldades, a fim de motivar a Aprendizagem Ativa. É válido ressaltar a relevância deste trabalho, pois apresenta de forma sucinta e interativa diversos tópicos que não são encontrados na literatura nacional.

O exemplo desenvolvido ao longo do texto apresenta uma aplicação real e é um dos mais diversos modelos de sistemas embarcados existentes. Teve como objetivo servir como base para apresentação de conceitos chave a fim de familiarizar o leitor com o assunto, no qual após a leitura do mesmo, torna-se apto a desenvolver sistemas mais complexos.

Por fim, o trabalho se torna uma alternativa a professores e alunos que desejam iniciar no mundo da programação, visto que contorna os principais desafios encontradas por estudantes que são: a dificuldade na lógica de programação e na sintaxe, pois todo o procedimento foi realizado utilizando blocos e ligações entre os mesmos.

Agradecimentos

A todos os integrantes do Grupo de Pesquisa CNPq, GSE (Grupo de Soluções em Engenharia) e também ao Polo de Inovação EMBRAPPII - IFMG, pela licença cedida do *software* MATLAB Simulink com a toolbox do Stateflow.

REFERÊNCIAS

ARDUINO Introduction: what is Arduino?. **Arduino**. Disponível em: <<https://www.arduino.cc/en/guide/introduction>>. Acesso em: 13 jul. 2020.

BARBOSA, J. W. **Sistema de irrigação automatizado utilizando plataforma Arduino**. 2013. 57 f. Trabalho de Conclusão de Curso (Graduação). Instituto Municipal de Ensino Superior de Assis. 2013.

BARR, M. Embedded Systems Glossary. **Barr Group**. Disponível em: <<https://barrgroup.com/embedded-systems/glossary>>. Acesso em: 14 jul. 2020.

CAMPOS, R. **Metodologia ERM2C**: para melhoria do processo de ensino-aprendizagem de lógica de programação. In: Workshop de Ensino em Informática, 2010. Belo Horizonte, MG. 2010.

GOMES, A. J. **Dificuldades de aprendizagem de programação de computadores**: contributos para a sua compreensão e resolução. 2010. 492 f. Tese (Doutorado em Engenharia Informática). Faculdade de Ciências e Tecnologia, Universidade de Coimbra. 2010.

HEATH, S. **Embedded systems design**. EDN series for design engineers: An embedded system is a microprocessor-based system that is built to control a function or a range of functions. 2013. Newnes. ISBN 978-0-7506-5546-0.

KRISHNA, R. Embedded Systems Tutorial: History, Types, Advantages, Examples. **Guru99**. Disponível em: <<https://www.guru99.com/embedded-systems-tutorial.html>>. Acesso em: 13 jul. 2020.

MOREIRA, G. L. *et al.* **Desafios na aprendizagem de programação introdutória em cursos de TI da UFERSA, campus Pau dos Ferros**: um estudo exploratório. In: Encontro de Computação do Oeste Potiguar (ECOP). Pau dos Ferros, RN. 2018. ISSN 2526-7574.

SALES, A. *et al.* **Dificuldades para o ingresso e permanência na ciência e engenharia da computação**: um olhar feminino. 18º REDOR, UFRPE. Recife, PE. 2014.

SENSOR de Umidade do Solo Higrômetro. **FILIFE FLOP**. 2020. Disponível em: <<https://www.filife flop.com/produto/sensor-de-umidade-do-solo-higrometro/>>. Acesso em: 13 jul. 2020.

SIMULINK. **MathWorks**. Disponível em: <<https://www.mathworks.com/products/simulink.html>>. Acesso em: 13 jul. 2020.

SIMULINK tutorial. **IEEE**: Electrical Safety Workshop. Disponível em: <<https://ewh.ieee.org/r1/ct/sps/PDF/MATLAB/chapter8.pdf>>. Acesso em: 13 jul. 2020.

SOIL Moisture Sensor Market. **Market Research**. 2020. Disponível em: <<https://marketresearch.biz/report/soil-moisture-sensor-market/>>. Acesso em: 14 jul. 2020.



TIRONI, P. *et al.* **Rede CAN Automotiva:** Perspectivas gerais e vulnerabilidades. In: Conferência de Estudos em Engenharia Elétrica (CEEL). Uberlândia, MG. 2018. ISSN 2178-8308. doi: 10.14295/2596-2221.xviceel.2018.166.

WHAT Is Stateflow? - Stateflow Overview. **MATLAB**. 2013. Disponível em: <<https://www.youtube.com/watch?v=hr4UJS0O52UI>>. Acesso em: 13 jul. 2020.

ACTIVE LEARNING FOR EMBEDDED SYSTEMS: A SIMULINK/STATEFLOW APPROACH

Abstract: *This article aims to present a tutorial on the basic steps of developing embedded systems, through the Arduino platform together with the Simulink software. Also noteworthy is the intention to contribute to the training of future engineers, through an alternative to the one usually used in the teaching of the microcontroller area. This theme is based on the difficulties encountered by students in the preparation of a structured logical reasoning for the solution of computational problems, in the understanding of concepts and in the complexity / syntax of the algorithms, which are often easy for learning. Problems that can be suppressed by using alternatives to conventional programming (bottom-up) by visual logic (blocks). These represent, through states and transitions, a whole complex chain of functions and structures contained in an algorithm. Based on this topic, the text presents the reader with the key concepts for the development of embedded systems with two MathWorks® tools: Simulink and Stateflow. Through active learning, a real example is exposed, so that from this tournament the reader is able to develop various embedded systems according to his interest. The application of Arduino to any theoretical concept leads the reader to a learning process facilitated by being an easy to use tool.*

Keywords: *Embedded Systems. Simulink. Stateflow. Tutorial. Active Learning.*