OKIoT: Simulador de Dispositivos de Internet das Coisas integrado com o Protocolo MQTT

Fabio Hirotsugu Hayashi[1], **Victor Takashi Hayashi**[2], **Felipe Valencia de Almeida**[2], **Reginaldo Arakaki**[2]

[1]Laboratórios Didáticos Secos
Universidade Federal do ABC (UFABC) – São Paulo – SP – Brasil

[2]Departamento de Engenharia de Computação e Sistemas Digitais
Escola Politécnica da Universidade de São Paulo (USP) – São Paulo – SP – Brasil

fabio.hayashi@ufabc.edu.br

{victor.hayashi,felipe.valencia.almeida,reg}@usp.br

Resumo. *O Arduino é uma plataforma de código aberto utilizada no ensino de programação para a robótica educacional e o pensamento computacional. Para garantir menor custo e acessibilidade, existem diversas ferramentas de simulação que permitem o desenvolvimento de soluções independentes para Arduino. Contudo, soluções conectadas são necessárias para o advento de verticais previstas no Plano Nacional de Internet das Coisas como cidades inteligentes e Indústria 4.0. Neste artigo, a ferramenta de código aberto OKIoT é apresentada para fomentar o aprendizado de conceitos de computação distribuída necessários para a construção de soluções conectadas.*

1. Cenário de Uso

O Arduino é uma plataforma de código aberto que conta com ampla comunidade de colaboração. Na literatura, é possível encontrar iniciativas e propostas do uso do Arduino como uma forma de potencializar o interesse de alunos em aulas de geometria [Sales et al. 2017], em robótica educacional [Silva et al. 2018], além de sua integração com o *Scratch* para explorar o pensamento computacional para o ensino fundamental [Pereira and Franco 2018] e médio [Silva et al. 2019].

Ferramentas de simulação para Arduino possuem um menor custo associado que alternativas de laboratório residencial ou remoto, e podem aumentar a acessibilidade por parte de seus usuários, pois a maioria das ferramentas podem ser utilizadas a partir de um computador com Internet, com baixo custo computacional. Há diversas alternativas proprietárias como o *Autodesk Tinkercad*[1], e soluções de código aberto como o *Wokwi*[2].

O uso de Internet das Coisas (IoT) para a construção de cidades inteligentes pode tornar possível o monitoramento em tempo real, diagnóstico e levantamento de pontos de otimização de processos e redução de custos em ações relacionadas à iluminação pública, Indústria 4.0 e sustentabilidade, considerando o Plano Nacional de IoT[3]. Contudo, a capacitação profissional com tecnologias IoT ainda é um ponto em aberto no

[1]https://www.tinkercad.com/things/dEOWWJatCtv-arduino-simulator
[2]https://wokwi.com/arduino/new?template=arduino-uno
[3]https://www.gov.br/mcti/pt-br/acompanhe-o-mcti/transformacaodigital/internet-das-coisas

DOI: 10.5753/wcbie.2021.218486

Brasil. Como capacitar profissionais que atuarão na construção das cidades inteligentes usando tecnologias de IoT? Esta educação pode começar em que nível?

Neste artigo, é apresentada a ferramenta de código aberto OKIoT com interface de usuário e dispositivo virtual baseado em HTML e *javascript*. Considerando a Taxonomia de Bloom revisada [Krathwohl 2002], o objetivo de aprendizagem considerado é o seguinte: ”usando o dispositivo IoT e interface de usuário da ferramenta OKIoT, o aluno deve aplicar o protocolo MQTT para a comunicação entre dois dispositivos virtuais”. O público-alvo considerado é o superior em cursos de extensão de IoT, ou disciplinas de Laboratório de Engenharia de Software ou Introdução à Engenharia em que o *Project Based Learning* (PBL) [Solomon 2003] é implementado. Em especial, a simplicidade e amplo escopo de integração do protocolo MQTT pode facilitar a integração dos sistemas distribuídos idealizados nos projetos dos alunos.

2. Desenvolvimento

O simulador foi desenvolvido à semelhança de um protótipo em placa real de desenvolvimento ESP8266 apresentado na Figura 1. O código funcional do exemplo da biblioteca MQTT para ESP8266 PubSubClient[4] também foi utilizado para que o comportamento da saída serial e os códigos específicos para comunicação MQTT apresentados na ferramenta fossem os mesmos que o código real. O código foi compilado e carregado por meio da ferramenta de desenvolvimento Arduino IDE, conforme ilustrado na Figura 1. O simulador foi desenvolvido em linguagem *javascript* e HTML para as interfaces de usuário e simulador de dispositivo, sendo fundamentado pelo protocolo de padrão aberto MQTT.

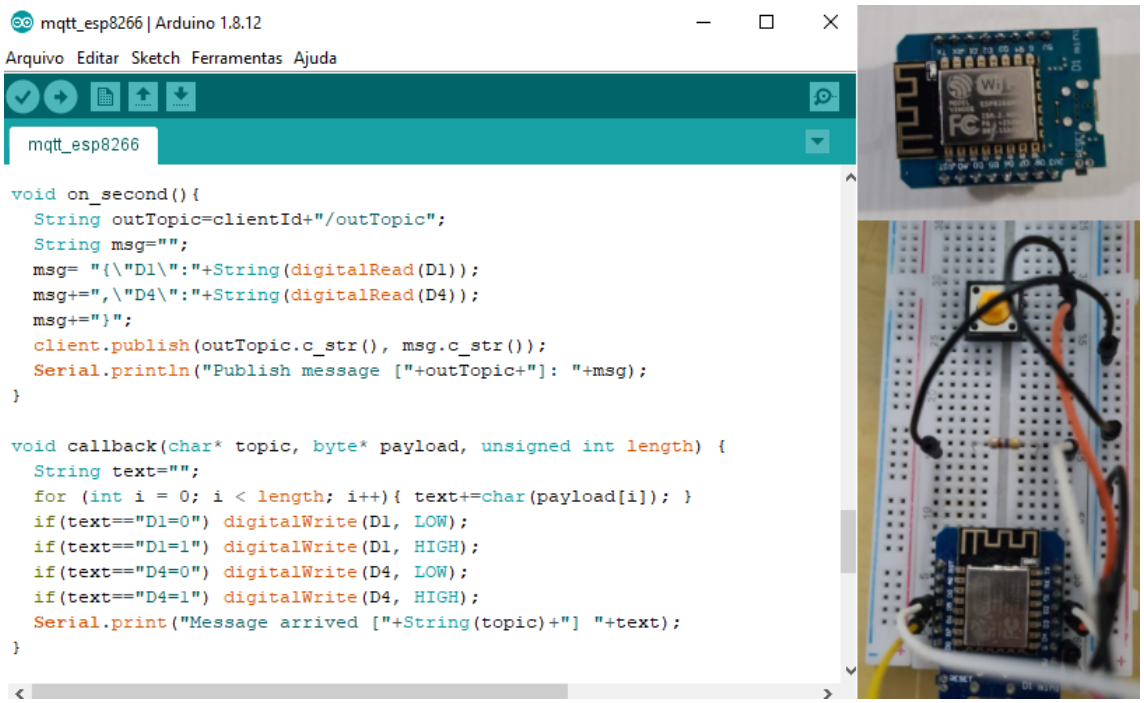

Figura 1. Arduino IDE (esquerda) e protótipo real usado como modelo para o desenvolvimento do simulador (direita).

[4] https://github.com/knolleary/pubsubclient

O MQTT (*Message Queuing Telemetry Transport*) é um protocolo de troca de mensagens leve, com ampla possibilidade de aplicação em projetos IoT devido ao seu baixo custo computacional. O protocolo baseia-se em um paradigma denominado *publish-subscriber*. Nele, determinado dispositivo conectado à rede é um cliente, e este pode ser do tipo *publisher*, ou seja, um publicador de conteúdo (mensagens) ou então do tipo *subscriber*, ou seja, um assinante de conteúdo. O MQTT também se utiliza do conceito de tópicos, onde cada tópico é um canal de comunicação separado.

Para facilitar o entendimento, é possível fazer uma analogia entre o MQTT e uma situação simples do dia a dia. Tomando por exemplo o YouTube, os usuários podem interagir ativamente com a plataforma por meio de uma conta Google. É possível fazer um paralelo entre este usuário e o cliente MQTT, pois o usuário pode ter um canal na plataforma, onde ele posta seus vídeos, se tornando um *publisher* do seu canal. Ao mesmo tempo outros usuários podem optar por se inscrever no canal, se tornando *subscribers* deste. Assim, o canal do Youtube é uma analogia interessante para o paradigma utilizado no protocolo MQTT.

Para que esta comunicação funcione corretamente é necessário existir um intermediário entre o *publisher* e o *subscriber* sendo este o MQTT *broker*. O *broker* é um servidor responsável por rotear as mensagens entre os diversos clientes da rede por meio da informação correspondente ao tópico. Existem diversas opções de *brokers* disponíveis para uso atualmente. Alguns são de uso público, como por exemplo o disponível pela HiveMQ[5]. Outros são gratuitos e possuem código aberto, podendo ser instalados em máquinas próprias ou na nuvem como por exemplo o Mosquitto [Light 2017]. Para o desenvolvimento do protótipo, um *broker* público[6] foi utilizado.

Uma validação inicial ocorreu por meio de apresentação da ferramenta construída aos docentes de um curso de extensão de IoT da Escola Politécnica da Universidade de São Paulo (USP), que confirmaram a utilização da ferramenta no próximo oferecimento do curso no segundo semestre de 2021. Cabe o destaque ao uso da ferramenta em atividade de absorção neste curso, como uma forma de complementar os questionários semanais e preparar os alunos para o projeto prático também previsto no curso de extensão. Os projetos dos alunos podem ter temas relacionados à aplicação de IoT, como a gestão energética por meio de medidores de energia conectados, manutenções preditivas em indústrias, e monitoramento dos processos de reciclagem em cidades, por exemplo.

3. Apresentação do Software

A ferramenta é composta por dois arquivos, sendo eles o *esp8266.html* e o *mqtt_esp8266client.html*. O primeiro deles é a interface gráfica da ferramenta, enquanto o outro atua como interface de configuração do cliente MQTT. A seguir é apresentada uma visão geral de cada interface. A interface gráfica, apresentada na Figura 2, é composta por três recursos visuais, descritos a seguir.

1. Diagrama Esquemático: recurso visual que apresenta a montagem simulada. Tanto o *led built in* do ESP quanto o *led* do *protoboard* são funcionais, e podem estar acesos ou apagados. Outros sensores e atuadores como o *display* e relé presentes de forma ilustrativa podem ser integrados em versões futuras;

[5]http://www.hivemq.com/demos/websocket-client/
[6]broker.mqtt-dashboard.com

2. Monitor Serial: recurso visual que atua como *log* da comunicação via MQTT. Nele é possível identificar as novas mensagens recebidas em conjunto com o tópico correspondente a cada mensagem;
3. Arduino IDE: recurso visual para o usuário identificar qual código estaria sendo executado no dispositivo real ESP. O usuário deverá ser capaz de entender este código e relacionar os conceitos do MQTT e da programação do ESP relacionados ao código de maneira prática, detalhando o funcionamento das funções *on_second* que é executada a cada segundo, e o *callback* que é executada quando o simulador recebe um evento a partir de um tópico MQTT no qual está subscrito.

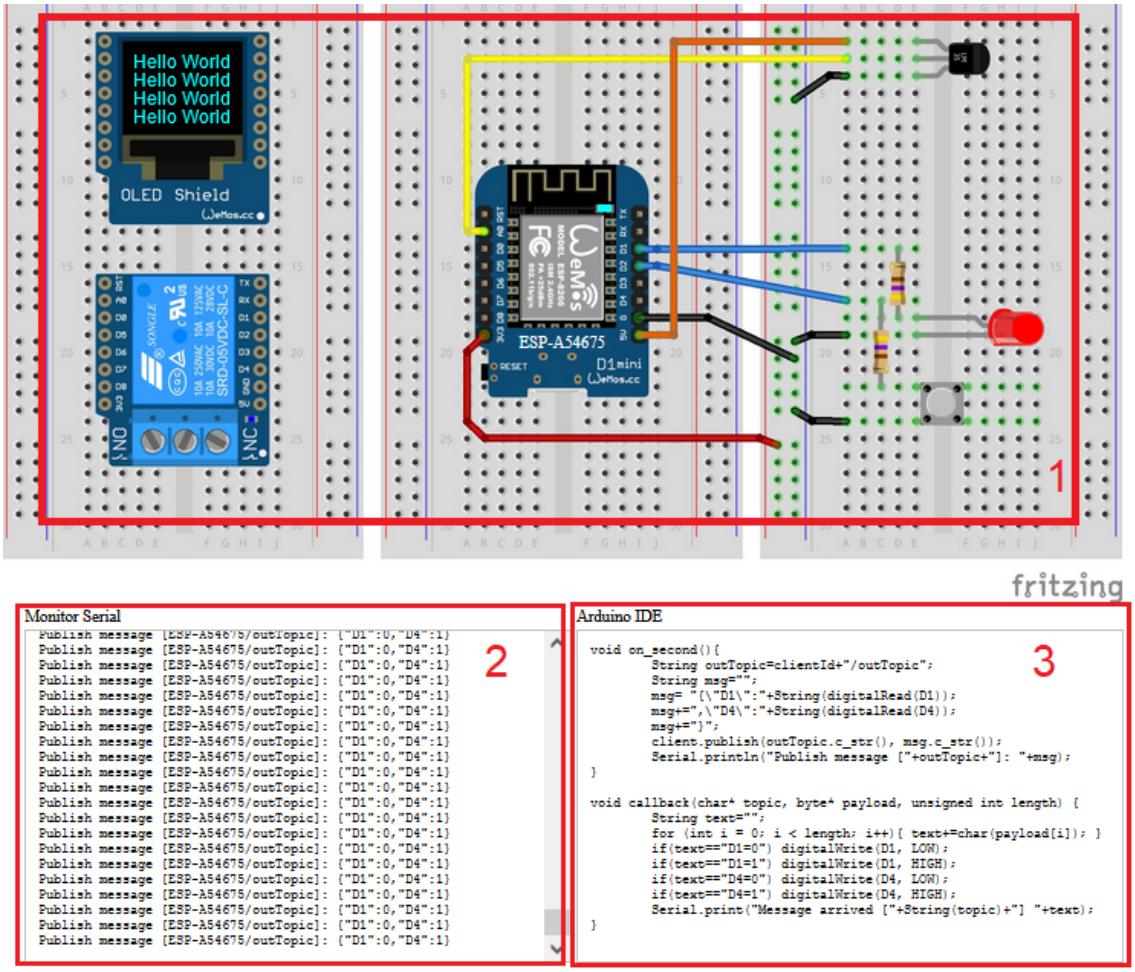

Figura 2. Interface gráfica do simulador de dispositivo ESP8266.

A interface de configuração do cliente MQTT, apresentada na Figura 3, é composta por duas sub-interfaces descritas a seguir.

1. Sub-interface de configuração do cliente: Nesta sub-interface o usuário configura o cliente MQTT;
2. Sub-interface de envio das mensagens: Permite ao usuário alterar o conteúdo de dois pinos de entrada/saída virtuais por meio de botões.

Uma visão alto nível da arquitetura MQTT da ferramenta é ilustrada pela Figura 4. O tópico *inTopic*, que contém as informações de estado de ambos os leds da interface gráfica é assinado por ela e publicado pela interface de configuração do cliente MQTT.

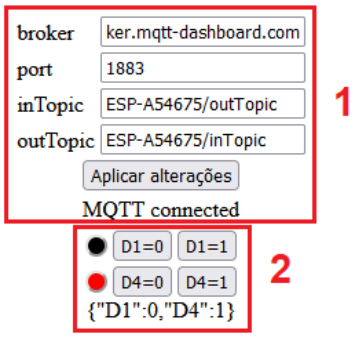

Figura 3. Interface de configuração do cliente MQTT.

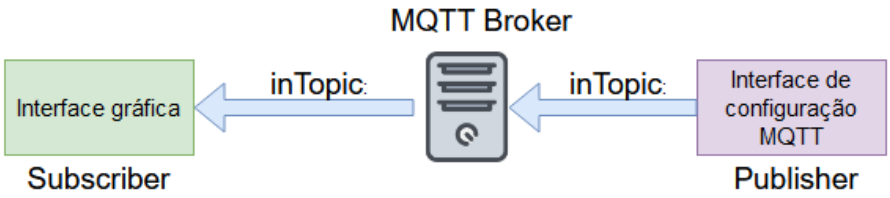

Figura 4. Arquitetura MQTT da ferramenta OKIoT.

Para utilizar a ferramenta, o usuário pode obter o projeto disponível no GitHub em um empacotamento com extensão ZIP. Em seguida os arquivos deverão ser descompactados, gerando uma pasta contendo com arquivos HTML descritos anteriormente. Ambos os arquivos deverão ser abertos lado a lado, facilitando a rápida visualização na interação entre eles. O primeiro passo consiste em configurar o cliente MQTT.

A Figura 5 apresenta a configuração do cliente MQTT na ferramenta. Um identificador do ESP é gerado aleatoriamente na interface gráfica, sendo necessário inserilo na interface de configuração do cliente MQTT, alterando os nomes dos tópicos. As informações correspondentes ao *broker* e a porta não precisam ser alteradas, exceto em uma situação onde o usuário deseja utilizar-se de um outro *broker* que não seja público.

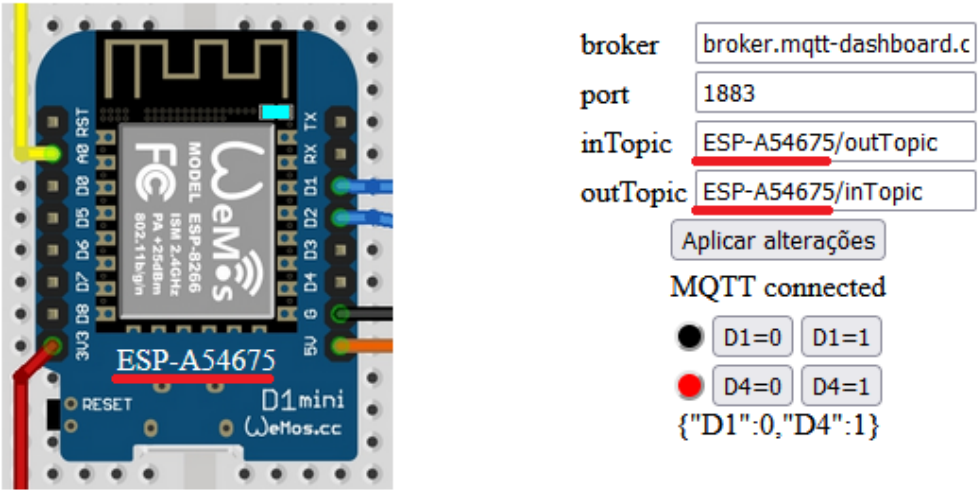

Figura 5. Configuração do cliente MQTT na ferramenta OKIoT.

Após a configuração do cliente o usuário pode alterar os valores de ambos os *leds* do diagrama esquemático clicando nos botões correspondentes com os valores desejados. O pino D1 controla o *led* do *protoboard*, que é ativo alto, já o pino D4 controla o *led built in*, que é ativo baixo. A Figura 6 apresenta os resultados dos casos de teste. Também é possível acompanhar os valores de ambos os pinos por meio do Monitor Serial.

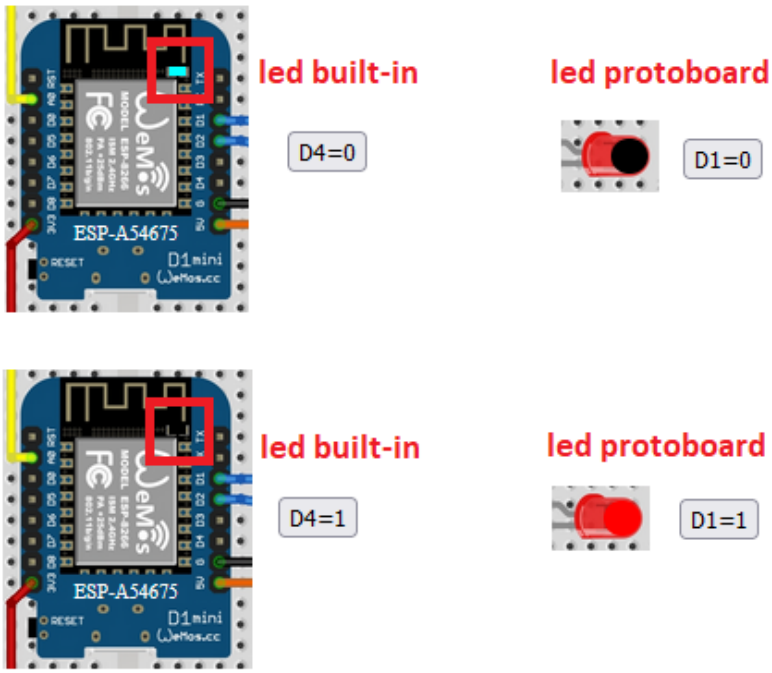

Figura 6. Casos de Teste do simulador OKIoT.

Além do uso das duas interfaces de maneira local, existe a possibilidade de utilização de dois sistemas diferentes, como por exemplo um computador e um celular. Neste caso a interface de configuração do cliente MQTT poderia ser controlada pelo celular por exemplo, agregando conceitos de arquiteturas distribuídas e computação pervasiva nas atividades de absorção que utilizem a ferramenta.

A ferramenta disponível sob a licença *GNU General Public License v3.0* pode ser obtida em: **`https://github.com/vthayashi/okiot-simulator`**

Um vídeo com a demonstração do simulador está disponível em: **`https://youtu.be/AlA83Y4Hzfo`**. Recomenda-se o uso do navegador Google Chrome para sua utilização.

4. Considerações finais

Considerando o melhor conhecimento dos autores, a ferramenta OKIoT é uma das pioneiras na simulação de dispositivos IoT com código aberto, considerando a integração com aplicações externas por meio do protocolo MQTT. A ferramenta cumpriu seu objetivo de suportar atividades que permitem um aprendizado prático sobre o protocolo MQTT, um exemplo de protocolo de comunicação para sistemas de computação distribuídos.

A ferramenta será utilizada no próximo oferecimento do curso de extensão de IoT na Escola Politécnica da Universidade de São Paulo (USP) no segundo semestre de 2021,

e está disponível como um Recurso Educacional Aberto (REA) para qualquer um utilizar ou expandir a ferramenta. O uso como um recurso adicional em disciplinas de Laboratório de Engenharia de Software ou Introdução à Engenharia em que o *Project Based Learning* (PBL) é implementado está sendo avaliado no momento de escrita deste artigo.

Uma oportunidade interessante é a integração da ferramenta com um Ambiente Virtual de Aprendizagem (AVA). Foram realizados testes de integração com os ambientes Moodle e Tidia-AE, e foram atestadas limitações nas duas paltaformas. O moodle permite somente código HTML estático (sem *javascript*), enquanto o Tidia-AE permite a execução do *javascript*, mas não permite a realização da conexão MQTT na porta padrão 1883 utilizada pelo *broker* público. Uma alternativa é a execução em um ambiente externo ao AVA com integração por meio de interfaces *Learning Tools Interoperability* (LTI) [Severance et al. 2010].

Outra possível expansão é incluir os procedimentos de ligações em *protoboard* no simulador, considerando possíveis erros dos alunos e como estes podem ser diagnosticados de forma didática. Além disso, a adição de mais dispositivos virtuais (servomotor, sensor ultrassônico, sensor de luminosidade, sensor de temperatura e umidade) com integração de mecanismos de interação virtual (aquecedor ou resfriador, alvo, umidificador) pode complementar a ferramenta com novas atividades para casos específicos.

Referências

Krathwohl, D. R. (2002). A revision of bloom's taxonomy: An overview. *Theory into practice*, 41(4):212–218.

Light, R. A. (2017). Mosquitto: server and client implementation of the mqtt protocol. *Journal of Open Source Software*, 2(13):265.

Pereira, A. C. and Franco, M. E. (2018). Desenvolvendo o pensamento computacional no ensino fundamental com arduino e scratch. In *Anais do V Encontro Nacional de Computação dos Institutos Federais*. SBC.

Sales, S. B., Silva, R. B., Sobreira, E. S. R., and do Nascimento, M. D. R. (2017). Utilizando scratch e arduino como recursos para o ensino da matemática. In *Anais do XXIII Workshop de Informática na Escola*, pages 538–547. SBC.

Severance, C., Hanss, T., and Hardin, J. (2010). Ims learning tools interoperability: Enabling a mash-up approach to teaching and learning tools. *Technology, Instruction, Cognition and Learning*, 7(3-4):245–262.

Silva, A., de Melo, R. F., de Sousa, R. P., and Nascimento, K. (2019). Estimulando o pensamento computacional em alunos do ensino médio com o uso do scratch for arduino. In *Anais do XXV Workshop de Informática na Escola*, pages 783–791. SBC.

Silva, J., Cristiano, F., Martins, D., and da Silva, W. (2018). Storytelling e robótica educacional: a construção de carros robôs com arduino e materiais recicláveis. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 29, page 1806.

Solomon, G. (2003). Project-based learning: A primer. *Technology and learning-dayton-*, 23(6):20–20.