SoftwareX 20 (2022) 101218

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

TinkercadNetConnector: Connecting emulated IoT devices to the outside world

Joshua Ellul [a,*], Carl James Debono [b]

[a] Department of Computer Science, University of Malta, Malta
[b] Department of Communications and Computer Engineering, University of Malta, Malta

ARTICLE INFO

Article history:
Received 24 January 2022
Received in revised form 29 August 2022
Accepted 26 September 2022

Keywords:
Arduino
Emulator
Test-beds

ABSTRACT

Tinkercad.com, amongst other things, provides an easy to use environment to emulate Arduino Uno devices along with connected virtual hardware components and is controlled by Arduino sketch code. Whilst the platform is a useful tool for learning, since it does not provide a means of communicating with the outside world, as soon as a project requires any form of communication with Internet enabled services, one has to resort to physical hardware (or other means of emulating/simulating Arduinos). In this paper, we introduce TinkercadNetConnector, which allows for Tinkercad emulated Arduino Uno devices to communicate with the outside world.

Code metadata

Current code version	1.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-22-00029
Code Ocean compute capsule	N/A
Legal Code License	GNU General Public License v3.0
Code versioning system used	git
Software code languages, tools, and services used	JavaScript
Compilation requirements, operating environments & dependencies	Chrome Extension
If available Link to developer documentation/manual	https://github.com/joshuaellul/tinkercad-net-connector/blob/main/README.md
Support email for questions	joshua.ellul@um.edu.mt

1. Motivation and significance

Tinkercad.com is a useful learning tool [1] for those that want to (amongst other things) learn how to design and code Arduino Uno based devices. It is also a suitable tool for early stage Arduino Uno firmware system development and testing. However, as soon as any interaction is required with any Internet service outside the emulated environment, one would need to resort to using physical hardware,[1] desktop based software,[2] or less intuitive user interfaces.[3] While such tools are useful, it would be ideal to allow for emulated devices to communicate with other external devices and web-based services using an online and easy to use platform. Given that Tinkercad.com provides an easy to use web interface [2–4] if support were to be added to allow for communication with external web services and devices, then this would meet the requirements laid out above that would support a platform that is easy to use, web based and allows for external connectivity without requiring physical hardware.

In this paper, we present TinkercadNetConnector, a Google Chrome Extension that: (i) turns Tinkercad emulated Arduino serial output into HTTP GET requests (and in turn responds back with the associated HTTP GET response as serial input); and (ii) polls a server for incoming messages that are turned into Arduino serial input.

The extension provides the following benefits for experimentation that makes use of Internet of Things (IoT) devices, it enables to: (i) develop and test IoT device code that communicates with external web services (and devices); and (ii) scale up a network of internet connected emulated IoT devices without requiring physical IoT devices which can both minimise cost and time

* Corresponding author.
E-mail address: joshua.ellul@um.edu.mt (Joshua Ellul).

1 https://xevro.be/
2 https://www.simulide.com https://www.labcenter.com/whyvsm/
3 https://lcgamboa.github.io/

https://doi.org/10.1016/j.softx.2022.101218

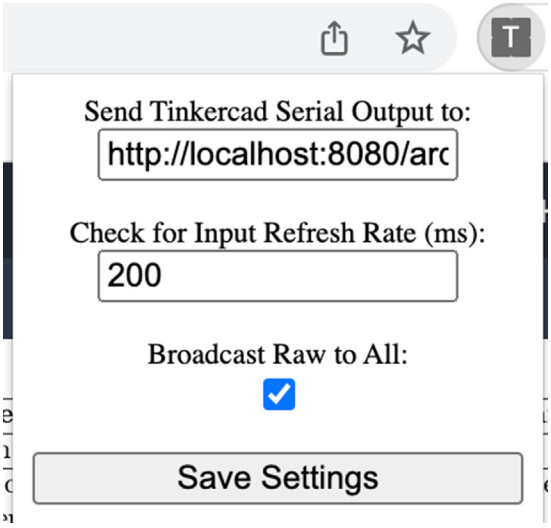

Fig. 1. Extension configuration pop window.

(to set up devices, and wait for firmware to be uploaded). Tinkercad already supports emulation of multiple Arduinos, however does not support their interaction with external web services.

To make use of the extension, users are required to: (i) install the extension in Chrome; (ii) configure the 'base url' where to send the output to, and how often to check for incoming messages. The minimal configuration required is depicted in Fig. 1.

2. Software description

TinkercadNetConnector facilitates integrating emulated Arduino Uno devices with: (i) other Tinkercad emulated devices in other tabs; (ii) external web services; (iii) physical Arduinos; or (iv) any other hardware or software required. Fig. 2 depicts a high level system architecture overview of TinkercadNetConnector. The extension (depicted in orange) is what enables for Tinkercad emulated devices to communicate with the external world — for which user-programmed or configured routing software (depicted in blue) can handle how and where to redirect messages to between the Tinkercad emulated devices and other system components. The routing software could be coded in any language/framework required (subject to being able to handle HTTP requests) — which could also include network/event-based configuration software (e.g. Node-RED[4]). We will now follow by providing an overview of TinkercadNetConnector's software architecture, and then describe the emulated device-to-routing software protocol.

2.1. Software architecture

An overview of the software architecture is depicted in Fig. 3. The extension is defined in the `manifest.json` file which configures:

- `popup.html` as a popup window depicted in Fig. 1 where users can configure the router url, refresh rate and the data format mode for incoming messages (discussed later). Client-side JavaScript for the pop-up window is stored in `popup.js`.
- `content.js` is the extension component that both listens to and makes changes to the Tinkercad web page.
- `background.js` which handles plugin tasks that need to execute in the background (independent of the content page).

The communication channels that exist between the components follow:

- **base-url**: The base url and refresh rate are stored in the extension's storage area which the background task then uses to: (i) communicate with the configured routing software; and (ii) set timers to poll the routing software for any incoming messages destined for the emulated Arduino devices.
- **process-input**: When the background component polls the external router for incoming messages and messages are returned, those messages are sent to the content component, which thereafter handles redirecting the message as serial input to the destined emulated device.
- **send-output**: When the content component parses changes in Tinkercad and recognises new messages output on the emulated Arduino's serial port, the message is sent to the background component that handles sending the message to the routing software that can then decide what to do with it — potentially sending it to another Tinkercad emulated device in another tab, a web service, a physical device or anywhere else.

2.1.1. Emulated device-to-routing software protocol

Since the intention of this extension is to facilitate external communication for any IoT (Arduino) software system and should not impose protocol requirements on the IoT system being developed (by the user of the extension), the extension's protocol to redirect messages from emulated devices to external routing software (and vice versa) was designed to be as light-weight as possible. Two direction-specific data redirection features are implemented:

Transmitting data from an emulated Arduino: Data sent by an emulated Arduino to its serial port is displayed in Tinkercad's *Serial Monitor*. The extension listens for any changes to the text in the *Serial Monitor* and forwards any detected data to the routing software (that is configured in the popup window). Data is forwarded to the base url configured with the following query string parameters:

- `msg`: The value for this parameter will be set to "output".
- `out`: The value for this parameter will be the data being redirected (that was sent from the emulated Arduino device to the serial port).
- `device`: The value for this parameter will be the Chrome's Tab ID — a unique identifier associated with each Chrome tab. The Tab ID can thereafter be used by the routing software to relay other messages intended for the specific emulated device.

For example, for routing software listening on the same machine on port 8080, and Arduino output "testing", when Tinkercad.com is running in Tab ID 203, the extension would initiate a GET request to:
http://127.0.0.1:8080/arduinoserver?msg=output&out=testing&device=203

The response data received back from the routing software will be redirected to the emulated Arduino by modifying the DOM to include the response data in the *Serial Monitor*'s input text field, and then automating the pressing of the send button by invoking its `click()` JavaScript function.

Sending data to an emulated Arduino: Since Chrome extensions do not support incoming connections, the extension was

[4] https://nodered.org/

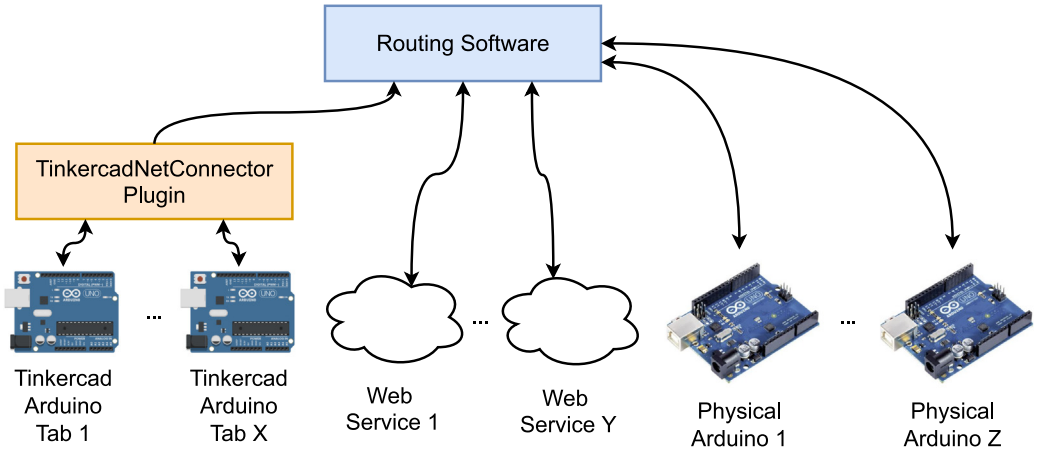

Fig. 2. TinkercadNetConnector System Overview.

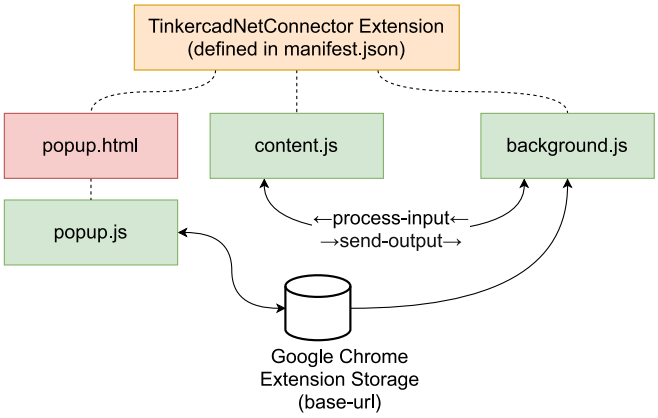

Fig. 3. TinkercadNetConnector Software Architecture Overview.

designed to poll the routing software for any data that is meant to be forwarded to an emulated Arduino — the extension polls the routing software periodically as dictated by the configured refresh rate. A GET request to the base url will be made with the following query string: "?msg=allinputs". The extension supports two different modes for handling incoming data: (i) when the extension's 'Broadcast Raw to All' checkbox is ticked, all data returned to the 'allinputs' HTTP GET request is sent to all Tinkercad tabs as serial input; and (ii) when the same checkbox is not ticked, the reply expected back from the routing software is a JSON object containing an array of input values targeted for specific emulated Arduinos (identified by the Chrome Tab ID within which they are running). This mode allows for specific data to be sent to specific Tinkercad tabs. An example response is provided in Listing 1.

```
{
  "inputs": [
    {"device": 203, "value": 777},
    {"device": 421, "value": 123},
  ]
}
```

Listing 1: An example response JSON object expected back from the routing software.

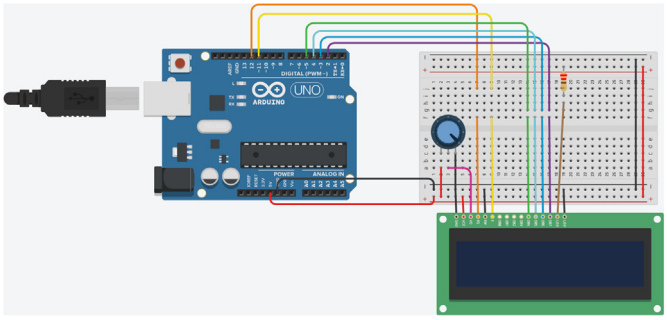

Fig. 4. Arduino Board for the Random Number Display example.

3. Illustrative examples

We now provide the following two illustrative examples of TinkercadNetConnector in use: (i) an Arduino that displays a random number generated from a NodeJS server; (ii) an Arduino that beeps based upon the number of 'bitcoin' tweets in the last 5 s that is retrieved via a Node-RED[5] server. All examples are available in the GitHub repository, including Arduino code, Arduino board design and server code.

3.1. Display random number from a NodeJS server

An Arduino board was set up with a connected LCD screen as depicted in Fig. 4. Salient aspects of the Arudino sketch code is provided in Listing 2.

```
void setup() {
  //...
  Serial.println("Register with router");
}

void loop() {
  if (Serial.available() > 0)
        lcd.println(receiveData());
}
```

Listing 2: Sketch Arduino code for Random Number Display from NodeJS server.

[5] https://nodered.org/

On `setup()` the extension outputs a message to the serial port that is directed to the NodeJS server so that the server is notified of the Arduino and its device ID (or rather the Chrome Tab ID which the server will use later on to send messages to). Thereafter, any received data will be displayed on the connected LCD screen.

The NodeJS server code is provided in Listing 3. The extension in this particular example was set up to redirect messages to the "/arduinoserver" endpoint, but this can be changed to whatever a user requires. The server handles the two HTTP GET requests that the extension sends, which follow:

- "output": The NodeJS server keeps track of Arduinos by first requiring them to send a message to the server (the message sent in the Arduino `setup` function).
- "allinputs": When the extension polls for any input to be forwarded to Arduinos, the NodeJS server generates a random number for each registered Arduino and sends them back to the extension for forwarding.

`receiveData()` listens for any incoming data and buffers characters received, until a '+' character is received at which point `receiveData()` will return the buffered characters received. This protocol could have been designed differently and is not imposed by the extension — in the next use-case a different protocol will be used.

```
var devices = new Array();
app.get("/arduinoserver", (req, res) => {
    if (req.query.msg === "output") {
        devices.push(parseInt(req.query.
            device));
    } else if (req.query.msg === "
        allinputs") {
        var inputsArray = [];
        devices.forEach(element => {
            inputsArray.push({
              "device": element,
              "value": getRandomNr().
                  toString() + "+"
            });
        });
        res.send(JSON.stringify({
            "inputs": inputsArray
        }));
    }
});
```

Listing 3: NodeJS server code for providing Random Numbers to the emulated Arduino.

3.2. Retrieve tweet count from node-RED and beep

An emulated Arduino board was set up with 3 connecting piezo buzzers as depicted in Fig. 5. The different buzzers will beep based upon the number of 'bitcoin' tweets since the last time they were checked — the number of tweets are being checked every 5 s. If 10 tweets have since been tweeted then 1 buzzer will beep; if 20 then 2 buzzers; while if 30 tweets occur then 3 buzzers will beep.

The Arduino continuously checks for incoming messages from the Node-RED server that have the following message structure:

`<STX><TWEET COUNT><ETX>`

When a message of this format is received, the Arduino will process it and beep the buzzers according to the aforementioned logic. The Arduino sketch code is provided in Listing 4.

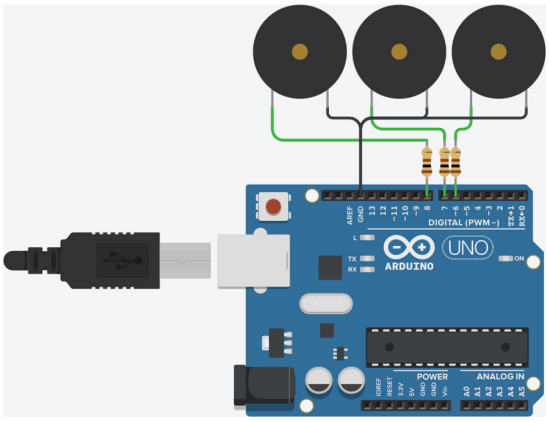

Fig. 5. Arduino Board with 3 connecting piezo buzzers.

```
long prevTweets = 0;

void setup() {
    Serial.println("Register with node
        red");
}

void loop() {
    long tweets = getTweets();
    long diffTweets = tweets - prevTweets
        ;
    prevTweets = tweets;
    if (diffTweets >= 10) beep(6);
    if (diffTweets >= 20) beep(7);
    if (diffReading >= 30) beep(8);
}
```

Listing 4: Sketch Tweet Beep Arduino code.

The Node-RED server flow is depicted in Fig. 6. Incoming HTTP requests will either be 'output' or 'allinputs' messages as per the extension implementation. If the request is an incoming 'output' message (the top part of the flow) then Node-RED saves the emulated device's ID. When the extension polls for incoming data via the 'allinputs' message (the bottom part of the flow), the Node-RED flow will get the number of tweets associated with 'bitcoin' and build a response including the number of tweets and saved Device ID, and thereafter send the response back to the extension (for forwarding to the emulated device).

4. Other examples in the repository

To further demonstrate the use of the plug-in, two other use-cases (available in the github repository) have been implemented to demonstrate:

1. A more compute intensive example which compresses data on the Arduino and transmits it to the server, whilst it decompresses data received back from the server. The technique implemented is a recently proposed light-weight compression technique [5].
2. An example to demonstrate and test how many different Tinkercad tabs can be run at the same time. A heartbeat example was implemented in which a server sends a ping to all connected Tinkercad devices, for which each device blinks an LED upon receiving the ping.

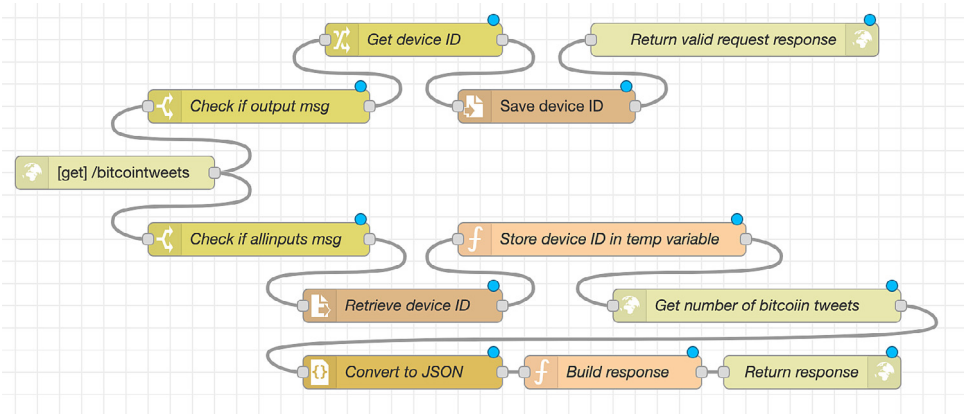

Fig. 6. Node-RED flow handling (i) saving of Arduino Device ID, and (ii) returning the total number of bitcoin tweets.

Given that the devices are emulated in Chrome one would expect that the number of simultaneous tabs that could be run will be low. Whilst this is the case, larger networks could be tested for use-cases where a single master Arduino node can be used to coordinate other Arduino nodes emulated in the same Tinkercad tab. In future we will investigate whether the extension can be extended to directly communicate to all different Arduino nodes emulated in the same tab. Given that each tab executes only a single JavaScript thread, this would help reduce overheads induced through the requirement of many chrome tabs.

Tests were run on a 3.1 GHz Dual-Core Intel Core i5 MacBook Pro with 8 GB memory to see whether any visual slow down could be seen. First OBS screen recording software was started to record the screen and the server side code available in the github repository was started. Thereafter, 15 new Tinkercad web page instance were loaded and each simulation started. Different chrome instances means that a separate JavaScript thread would be running for each. The screen recording[6] demonstrates that the first 5–6 instances run without any signs of speed degradation — thereafter when loading new chrome instances one can notice the simulations slowing down (and later catching up). Once an instance is loaded and the different instances have settled the different heart beats seem to be in sync until the 7th instance, and thereafter the instances can be seen to fall out of sync and back in sync at times. This is what one would expect with different JavaScript engines running in different Chrome instances.

5. Impact

The Internet of Things (IoT) is changing and has the potential to change not only our daily lives, but also to enhance scientific research in various domains including environmental [6], space [7], healthcare [8], marine [9], chemical [10] sciences amongst others. Since experiments can often involve setting up different IoT devices, configurations and sensors, initial system design and testing can involve an extensive amount of hardware configuration and set up times. Especially when labs are used to build, develop and test different IoT systems. More so, software developers involved in programming the required software need to often be limited to those that have access to physical hardware setups.

Various simulation [11,12] and emulation [13,14] tools have been proposed in aim of enabling for ease of development, modelling of large networks and other aspects. Yet various tools available either require access to physical hardware,[7] have non-intuitive user interfaces,[8] or require users to make use of desktop based software.[9] Whilst such solutions may be adequate for many situations, it would be ideal to support web-based Arduino development — for example in cases where individuals may be limited to Chromebook-like laptops (potentially in areas that have a large digital divide and such laptops are cheaper or handed out [15]). Whilst, Tinkercad.com provides a solution to this, problems are posed as soon as emulated Arduinos require interaction with the outside world beyond the emulated environment.

TinkercadNetConnector can, by enabling Tinkercad.com to interact with outside the emulated environment, support future research that makes use of IoT devices through facilitating ease of IoT Arduino firmware development and testing for applications that require to communicate with external services. Furthermore, such a solution can enable for individuals having access to only Chromebook-like laptops to learn and develop Arduino solutions that are not limited to the emulated environment (which could further support research and innovation through enabling more versed IoT device software developers). Furthermore, for such systems, hardware set up and configuration time associated with typical IoT system development can be removed, facilitating faster Arduino firmware development.

6. Limitations and future directions

The number of devices that can be emulated at the same time (on the same machine) is dependent upon: (i) a specific machine's processing and memory resources; and (ii) a chrome tab resource requirements and overheads incurred by running a Tinkercad emulated device. To further support emulating more devices, in future work we plan to extend the extension to easily support

6 Available at: https://www.um.edu.mt/l/bbXnd

7 https://xevro.be/

8 https://lcgamboa.github.io/

9 https://www.simulide.com, https://www.labcenter.com/whyvsm/

communication across different machines using the extension. Whilst this is already possible using routing software, we plan to provide an interface that would allow for different instances of the extension on different machines to route messages directly without having to provide separate routing software.

Furthermore, currently users are required to set up the different tabs and start them accordingly. For experiments or test-beds, whilst setting up the emulated devices is less of a pain than on actual hardware, in future work we aim to allow for users to configure a test-bed configurable environment that will automatically set-up and start the different emulated devices automatically.

Tinkercad currently supports Arduino Uno and Micro:bit boards — of which a serial input/output monitor is only supported for Arduino Unos. Therefore, the extension currently only supports emulated Arduino Uno devices. In future work we plan to support more IoT platforms by modularising the code to better support extensibility for different web-based platforms.

7. Conclusions

In this paper, TinkercadNetConnector, a Chrome extension to support communication outside of the emulated environment was presented. We presented two use-cases that demonstrate how it can be used to communicate with an external device or service through a routing software component that can take any form including intuitive web-based flow applications like Node-RED.

Data availability

No data was used for the research described in the article.

Acknowledgements

The work was partially funded by the European Regional Development Fund - INTERREG V-A Italia-Malta project NATIFLife C1-1.1-90.

References

[1] Mohapatra BN, Mohapatra RK, Joshi J, Zagade S. Easy performance based learning of arduino and sensors through tinkercad. Int J Open Inf Technol 2020;8(10):73–6.
[2] Abburi R, Praveena M, Priyakanth R. Tinkercad-a web based application for virtual labs to help learners think, create and make. J Eng Educ Transf 2021;34(SP ICTIEE):535–41.
[3] Eryilmaz S, Deniz G. Effect of tinkercad on students' computational thinking skills and perceptions: A case of ankara province.. Turkish Online J Educ Technol-TOJET 2021;20(1):25–38.
[4] Mohapatra BN, Mohapatra RK, Jagdhane V, Ajay CA, Sherkar SS, Phadtare VS. Smart performance of virtual simulation experiments through arduino tinkercad circuits. Perspect Commun Embedded-Syst Signal-Process-PiCES 2020;4(7):157–60.
[5] Ramanathan A. Unishox: A hybrid encoder for short unicode strings. J Open Source Softw 2022;7(69):3919.
[6] Hart JK, Martinez K. Toward an environmental internet of things. Earth Space Sci 2015;2(5):194–200. http://dx.doi.org/10.1002/2014EA000044.
[7] Dabbakuti JK, Ch B. Ionospheric monitoring system based on the internet of things with ThingSpeak. Astrophys Space Sci 2019;364(8):1–7.
[8] Chakraborty C, Banerjee A, Kolekar MH, Garg L, Chakraborty B. Internet of things for healthcare technologies. Springer; 2021.
[9] Al-Zaidi R, Woods J, Al-Khalidi M, Ali Alheeti KM, McDonald-Maier K. Next generation marine data networks in an IoT environment. In: 2017 Second international conference on fog and mobile edge computing. FMEC, 2017, p. 50–5. http://dx.doi.org/10.1109/FMEC.2017.7946407.
[10] Wen F, He T, Liu H, Chen H-Y, Zhang T, Lee C. Advances in chemical sensing technology for enabling the next-generation self-sustainable integrated wearable system in the IoT era. Nano Energy 2020;78:105155. http://dx.doi.org/10.1016/j.nanoen.2020.105155.
[11] D'Angelo G, Ferretti S, Ghini V. Simulation of the internet of things. In: 2016 International conference on high performance computing simulation. HPCS, 2016, p. 1–8. http://dx.doi.org/10.1109/HPCSim.2016.7568309.
[12] Chen M, Miao Y, Humar I. OPNET IoT simulation. Springer Nature; 2019.
[13] Kuwabara Y, Yokotani T, Mukai H. Hardware emulation of IoT devices and verification of application behavior. In: 2017 23rd Asia-Pacific conference on communications. APCC, 2017, p. 1–6. http://dx.doi.org/10.23919/APCC.2017.8304040.
[14] Ly-Trong N, Dang-Le-Bao C, Le-Trung Q. Towards a large-scale IoT emulation testbed based on container technology. In: 2018 IEEE Seventh international conference on communications and electronics. ICCE, 2018, p. 63–8. http://dx.doi.org/10.1109/CCE.2018.8465578.
[15] Lee J. Bridging the digital divide through the use of chromebooks in ethiopia. Library Hi Tech News 2020.