Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2023.042793

ARTICLE

Approach to Simplify the Development of IoT Systems that Interconnect Embedded Devices Using a Single Program

Enol Matilla Blanco[1], **Jordán Pascual Espada**[1] **and Rubén Gonzalez Crespo**[2,*]

[1]Department of Computer Science, University of Oviedo, Oviedo, 33007, Spain

[2]Department of Computer Science and Technology, Universidad Internacional de La Rioja, Logroño, 26006, Spain

*Corresponding Author: Rubén Gonzalez Crespo. Email: rubenagc@gmail.com

Received: 12 June 2023 Accepted: 19 September 2023 Published: 29 November 2023

ABSTRACT

Many Internet of Things (IoT) systems are based on the intercommunication among different devices and centralized systems. Nowadays, there are several commercial and research platforms available to simplify the creation of such IoT systems. However, developing these systems can often be a tedious task. To address this challenge, a proposed solution involves the implementation of a unified program or script that encompasses the entire system, including IoT devices functionality. This approach is based on an abstraction, integrating the control of the devices in a single program through a programmable object. Subsequently, the proposal processes the unified script to generate the centralized system code and a controller for each device. By adopting this approach, developers will be able to create IoT systems with significantly reduced implementation costs, surpassing current platforms by more than 10%. The results demonstrate that the single program approach can significantly accelerate the development of IoT systems relying on device communication.

KEYWORDS

IoT; interconnected devices; IoT platform; programing devices; devices coordination and communication

1 Introduction

The term "Internet of Things" refers to systems composed of multiple interconnected physical devices that collaborate to achieve a common objective [1]. In recent years, the utilization of these systems has experienced significant growth due to their widespread adoption across various sectors [2,3], like smart homes [4], healthcare [5,6], engineering, monitoring and automation [7,8].

Programmable microcontrollers play a crucial role in IoT systems, owing to their versatility and wide ranging applications. These compact computers efficiently manage electronic devices, including sensors and actuators. Microcontrollers find utility in diverse systems that require data acquisition from the physical world and the execution of actions through physical actuators. In the context of IoT, inter device communication assumes paramount importance. This involves establishing message definitions for device to device communication, selecting suitable communication protocols, and determining communication flows and interaction patterns.

Developing systems that involve interconnecting various smart devices can be a challenging task for developers, as it requires a solid understanding of smart devices, programming, and communication protocols. In most cases, when developers undertake the implementation of an entire system, it entails creating a centralized system along with programs to run on individual devices. This often results in a substantial amount of code being written [9].

Systems that facilitate device interconnection involve the exchange of data between devices, enabling them to trigger actions based on received data [10]. For instance, consider a scenario where multiple devices equipped with temperature sensors interact with a device controlling an air conditioner. When the temperature sensors detect a high average temperature, they send a command to the air conditioning device, instructing it to turn on. Although these systems usually do not require complex data analysis, they do involve data retrieval, analysis, and command execution. The mere definition of interfaces, communication protocols, and device coordination typically necessitates a significant amount of programming effort. This codebase may further expand if additional factors like authentication [11], security [12], updates [13], and device management [14] need to be considered.

To simplify and streamline the development process of such systems, there are numerous commercial and research platforms [15,16] as well as middleware solutions available [17]. These tools aim to mitigate complexity and address common requirements encountered during the creation of interconnected systems.

There are several robust commercial platforms, such as ThingSpeak [18], Amazon Web Services (AWS) IoT Core [19], Azure IoT Hub [20], Arduino Cloud [21] each other, that significantly reduce the implementation and management costs of IoT systems. These platforms offer a range of services including communication services, data storage from sensors, and data analysis capabilities. To minimize the implementation costs, many of these platforms provide mechanisms for defining business logic or rules to manage microcontrollers. These logic/rules often involve coordination among multiple devices, making them highly suitable for coordinating mechanisms. Some platforms even offer agile rule creation systems, including visual interfaces [22,23]. However, they typically require specific software implementations on the devices to connect to the service running the rules. The motivation of this research is to further evolve the paradigms used in previous platforms, aiming to streamline the creation of IoT applications.

The objective of this research is to simplify the development process of IoT systems that require communication among multiple devices. To achieve this goal, we propose implementing a unified program or script that encompasses the entire system, including IoT devices, using an abstraction approach that leverages programming objects. Subsequently, this proposal involves processing the unified script to generate the controller for each device.

The main contribution of this research is a novel platform that allows the development of the entire system in a single program which included the code of the devices and the server. By adopting this approach, developers can create IoT systems with a significantly reduced codebase while facilitating seamless intercommunication between devices. Using that proposal, the development of IoT systems will be achieved with a much lower time cost and without requiring knowledge of intercommunication and coordination between devices.

The paper is structured as follows. Section 2 shows the related work. Section 3 describes the proposal. Section 4 details a use case of the proposal. Section 5 presents the evaluation and discussion of results. Finally, Section 6 contains the conclusions and lines of future work.

2 Background

Small microcontrollers serve as the fundamental building blocks for numerous IoT systems. Despite being less powerful compared to other computers or devices, their cost effectiveness, compact size, and efficient power consumption make them a popular choice for a wide range of IoT projects [24]. Depending on the IoT project the requirements could be significantly different, even outside the functional scope, including aspects such as energy efficient [25] security [26].

There are some commercial platforms widely used in the industry to create IoT systems. The most relevant platforms are analyzed below, emphasizing the mechanisms they offer to specify the system logic and abstract communications and coordination between devices (Table 1).

Table 1: Analyzed platforms

	Abstraction in communication and coordination	Mechanisms to specify the logic of the system in a simple way
Amazon AWS	–	X
Azure IoT Central	–	X
ThingSpeak	–	X
Arduino Cloud	X*	X
Google IoT Cloud	–	X

Microcontrollers play a pivotal role in smart and connected objects, which are physical devices equipped with sensors to gather data, actuators to perform actions, and algorithms to make informed decisions [16,27]. In IoT systems, these smart objects establish connections with internet platforms to exchange data [28]. The software or firmware executed by these smart objects constitutes the foundation of their functionality.

ThingSpeak, one of the pioneering IoT platforms, offers a comprehensive system for defining login rules based on triggers. It enables the definition of business logic through trigger based forms, allowing developers to establish condition-reaction relationships. The platform provides software templates to simplify the development of controllers that are executed on IoT devices. While the platform abstracts communication features within the editor for business logic rules, it falls short in providing extensive configuration options for communication protocols. Despite this limitation, the efficient solution has gained widespread adoption and has been successfully utilized in numerous projects and solutions [29,30]. Incorporating ThingSpeak into IoT systems could prove to be an effective approach for reducing code complexity.

Google Cloud IoT provides an extensive solution that includes a wide array of tools aimed at tackling common challenges in the realm of IoT, such as deployment, monitoring, algorithm implementation, data analysis, and predictive capabilities. This platform has served as a fundamental framework for numerous research systems [31]. Among the specific tools offered within Google Cloud IoT, options like Dataflows and Cloud Functions hold significant promise in reducing code complexity for IoT applications. In the architecture of this platform, devices transmit data to either Cloud IoT Core or Cloud Pub/Sub. Developers are responsible for implementing the software on these devices, which then send data to the Google platform using various communication standards, primarily Message Queuing Telemetry Transport (MQTT) but with support for other protocols as well. Additionally, device functionality should define the commands it can interpret. The utilization of Google Cloud IoT can be immensely valuable in streamlining IoT systems that necessitate smooth communication between devices. Through the adoption of the platform's tools, developers can effectively diminish code complexity and improve the overall efficiency of their IoT solutions.

Amazon AWS IoT is a comprehensive solution that provides a wide range of tools to address common IoT challenges. With features such as the lightweight Free Real Time Operating System (FreeRTOS) operating system and IoT ExpressLink, it offers ways to simplify software implementation. The Rules engine allows developers to define the software logic for communication between devices. By leveraging these tools, AWS IoT enables developers to reduce complexity and enhance efficiency in their IoT systems [31].

Azure IoT is a highly popular platform for managing IoT systems [32]. Azure IoT offers seamless integration with Azure Event Grid and serverless compute, along with Azure IoT Edge support for hybrid IoT applications development. Within the platform, developers have the flexibility to configure rules and actions, allowing certain communication and coordination tasks to be handled by the platform. The rules can be easily defined using a user friendly form. Similar to other platforms, connecting devices to Azure IoT Hub necessitates the creation of specific software controllers for each device, which are then smoothly integrated into the platform.

Numerous research projects have resulted in platforms specifically designed to minimize the code complexity involved in implementing IoT systems. One such platform is Tiny Link, which offers users a simplified method to specify platform functionalities without the need to delve into the intricacies of underlying hardware components [33]. This platform adopts a low code approach for implementing the controller program on devices, resulting in a significant reduction in code complexity [34]. The main advantage of this platform is its ability to simplify the end to end decoding process for exchanging messages between devices, which proves especially beneficial for heterogeneous devices using diverse message specifications. It is essential to mention that the IoT market offers an extensive array of over 100 different platforms tailored to IoT devices [35], but the low code approach specifically focuses on simplifying the coding aspect of IoT systems and doesn't cover other features.

In the pursuit of simplifying code implementation, most platforms operate in a similar manner. Initially, developers create a specific controller for devices, enabling them to connect to the platform. This controller program defines the messages transmitted by the device and the commands/actions it can comprehend. Depending on the platform and device type, certain solutions may offer support for implementing the driver program, such as through templates or libraries, which alleviates developer effort. Libraries offer a convenient level of abstraction for communication protocols, but it is important to note that numerous platforms also provide the flexibility for developers to select the desired protocol and communication features according to their specific requirements.

The next step involves configuring coordination among devices. This typically entails sending data to the platform, processing it in varying degrees of complexity, and executing actions, often involving the devices themselves. Indeed, all IoT systems can be classified into two distinct components: (1) the device controller software, responsible for handling the information transmission and accepting commands, and (2) the system's rules or logic that governs the communication and coordination among devices. These two components play pivotal roles in ensuring the smooth functioning of IoT systems. This two component structure imposes additional work on developers, necessitating modifications in two separate areas that are often programmed differently and must synchronize with each other.

The objective of our proposal is to simplify the complexity by enabling the definition of the entire IoT system in a single program. By eliminating the need for code that connects devices to the platform and manages data transmission and reception, we can greatly reduce the number of lines of code and the effort required to define business logic rules. This streamlined approach simplifies the system implementation process and enhances overall efficiency.

3 Proposed Platform

3.1 Architecture

The proposal entails an IoT platform that simplifies the process of connecting microcontrollers for users. Users can effortlessly download a firmware that requires no modification. Upon downloading and executing the connection firmware on their microcontrollers, users gain the ability to program the entire system that interconnects their devices using a web based Integrated Development Environment (IDE). The web IDE is built on a programming language that closely resembles the native programming language of the microcontrollers.

For this prototype, we have designed a version of the framework that is compatible with Raspberry Pi and Arduino WeMos D1 Wi-Fi Espressif modules 266 (ESP266) microcontrollers. However, it is important to note that this is just one use case, as the platform can be extended to interact with various other types of microcontrollers. The platform should provide additional firmware options to enable users to connect different types of devices. Once the generic firmware is uploaded to the microcontrollers, it establishes a connection that links all the inputs and outputs of the devices with the IoT platform. This connection enables the execution of business logic on the platform's server or cloud (Fig. 1).

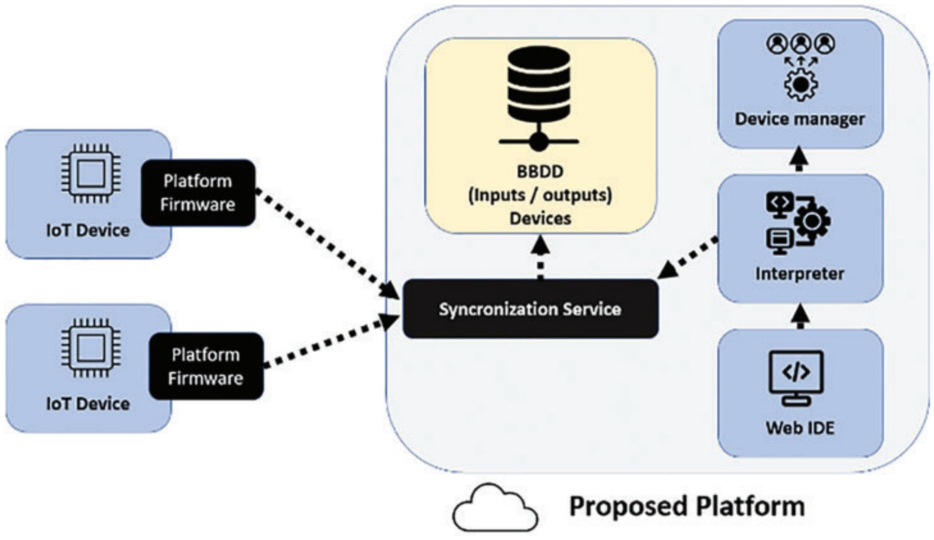
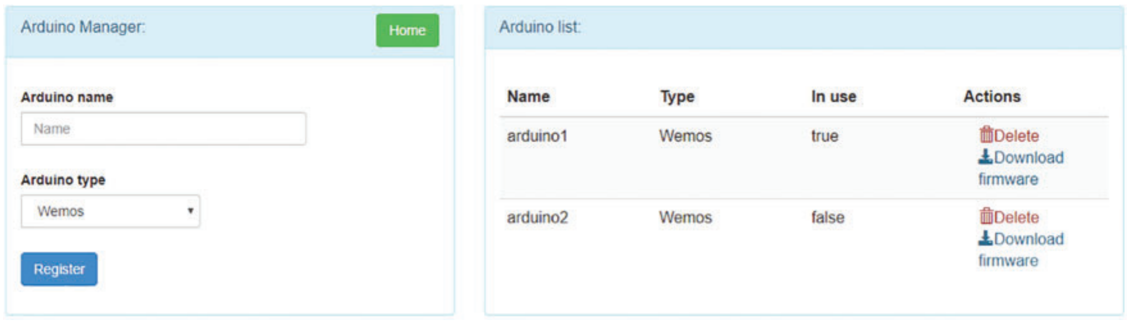


Figure 1: Solution conceptual scheme

The proposed platform architecture consists of two key components:

a) The Server: This component serves as the business logic hub of the system. It is a web application accessible to users through a login. Users can manage their microcontrollers within the platform (Fig. 2). Upon registering a new microcontroller, the platform automatically generates a unique ID for the device and requests the user to input internet connection parameters, such as Wi-Fi Service Set IDentifier (SSID) and password. Upon registration, the platform generates a connection firmware specific to the microcontroller. This connection firmware includes the device's ID and the internet connection parameters.



Name	Type	In use	Actions
arduino1	Wemos	true	Delete Download firmware
arduino2	Wemos	false	Delete Download firmware

Figure 2: Device manager

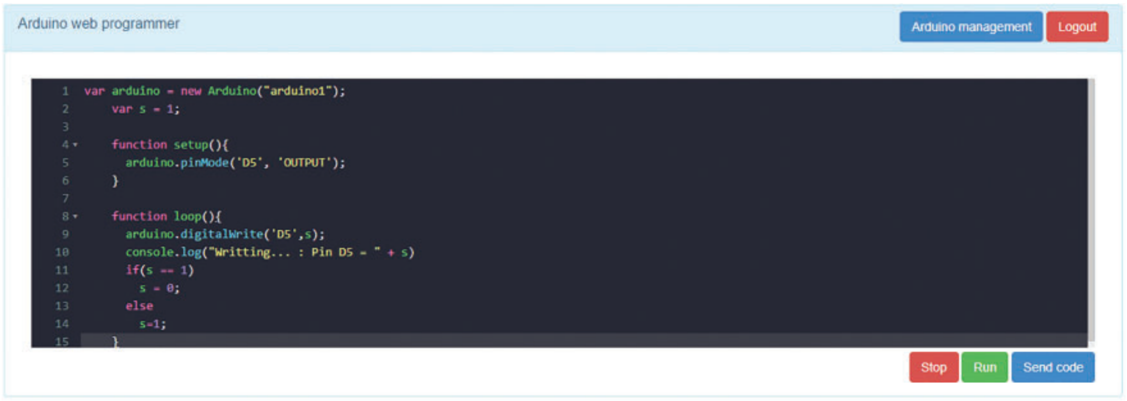
b) Microcontroller Connection Firmware: This component's primary responsibility is to establish the connection between the microcontrollers and the platform. The connection firmware, generated by the platform during registration, facilitates seamless communication between the microcontroller and the server. It incorporates essential information, including the unique ID of the device and the internet connection parameters provided by the user.

These two components work in tandem to facilitate the management and connectivity of microcontrollers within the proposed platform.

Once the microcontroller has the connection firmware running, the user can access the associated IDE "code editor" (Fig. 3). This code editor allows the user to implement the business logic of the microcontroller using a real programming language. The platform supports five functions that are specific to the microcontroller:

- function setup(): This function is executed only once at the start of the program.
- function loop(): This function is executed repeatedly every second after the setup() function finishes.
- pinMode(mode): This function is used to set the mode of a microcontroller pin as either INPUT or OUTPUT. It requires two parameters: the mode and the pin number to be modified.
- digitalRead(pin): This function returns the value of the specified microcontroller pin. Function receives as a parameter the number of the pin.
- digitalWrite(pin, value): This function changes the state of the specified microcontroller pin to the desired value. Function receives as a parameters the number of the pin and the new value for the pin.

When the user launches the program, it is executed on the platform. Prior to program execution, the platform retrieves the values of the digital and analog inputs from the microcontroller. Based on this data, the platform executes the program. Throughout the program execution, the digital and analog outputs of the microcontroller may undergo changes. The platform promptly sends the updated output values back to the microcontroller in response.



```
var arduino = new Arduino("arduino1");
    var s = 1;

    function setup(){
      arduino.pinMode('D5', 'OUTPUT');
    }

    function loop(){
      arduino.digitalWrite('D5',s);
      console.log("Writting... : Pin D5 = " + s)
      if(s == 1)
        s = 0;
      else
        s=1;
    }
```

Figure 3: Code editor

3.2 *The Device's Firmware*

The firmware empowers the microcontroller to operate as a web client, consistently sending the data from its analog and digital inputs (sensors) to the platform. These input values are generated by the sensors linked to the microcontroller. In return, the firmware receives updated values for all the microcontroller outputs (actuators). Changes in the output values can subsequently modify the behavior of actuators connected to the microcontroller, such as turning on a light (Fig. 4).

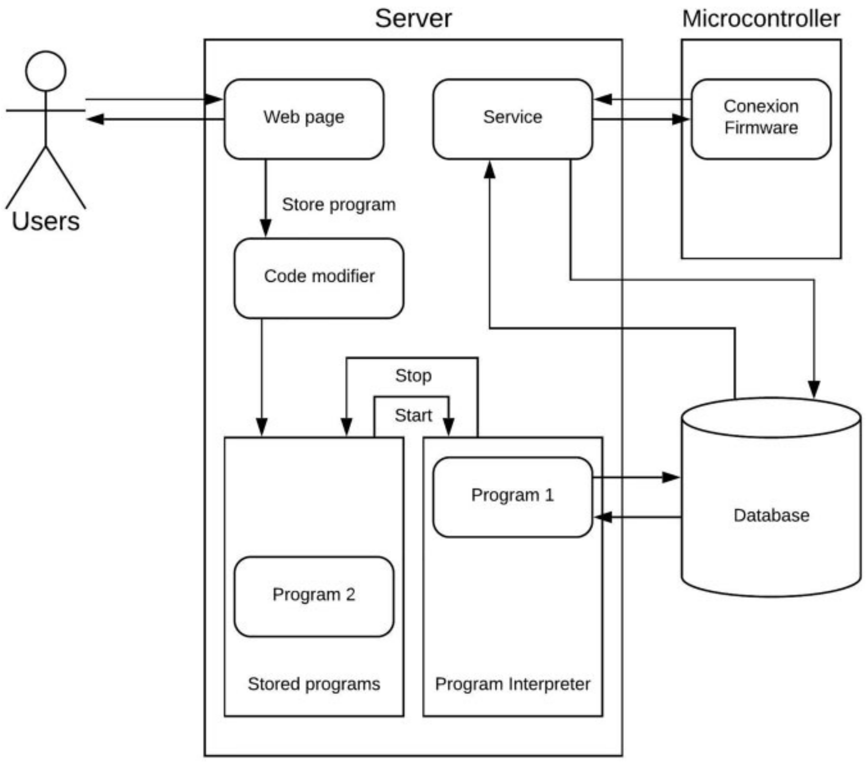


Figure 4: Basic interaction scheme

3.3 Code Redefinition

The Program Interpreter is responsible for executing the user created program within the web IDE. Before execution, the code undergoes a modification process by the Code Modifier. This process involves adding an internal library declaration, variables to identify the program owner, two customizable empty functions (setup and loop) that the user can utilize, and a try catch block to ensure the server continues running even in the event of errors in the user program. Additionally, the code modifier includes a call to the setup function, and a setInterval function for the loop function to execute every second with an initial delay of two seconds to ensure the setup is properly applied. In Fig. 4, you can see the modified program from Fig. 3 after passing through the Code Modifier.

Upon initialization of the "Device" class, an internal timer is activated to facilitate the retrieval and transmission of data between the database (BBDD) and the object. This mechanism enables efficient subsequent method calls on the "Device" object, allowing data retrieval directly from the object itself and avoiding delays associated with accessing the database.

3.4 Device Connection

The firmware, downloaded through the Device Manager, contains essential information such as the Wi-Fi network data, device ID, password (in the form of a generated hash), and the IP address of the platform server. Upon powering on, the microcontroller attempts to connect to the network specified in the firmware and initiates the program loop.

Within the program loop, the microcontroller retrieves data from all its pins, sends it to the server via an Hypertext Transfer Protocol (HTTP) request, and receives in response the pin mode for each pin and the data for the output pins. The microcontroller then updates the pin modes and writes the received values to the corresponding output pins. To accommodate more advanced sensors and actuators, like the digital temperature and humidity sensor (DHT11) or servomotors, the solution can seamlessly incorporate third party libraries. These libraries abstract the control of these devices, making it easier to interact with and utilize them in the IoT system.

Currently, the loop executes once per second, but this time interval can be adjusted within the platform, provided that the device supports sending data at that frequency.

4 Use Case

To illustrate the application of the proposed platform, let us consider a scenario where a company aims to automate the lights in a room using three interconnected devices (Fig. 5).

- Device 1: This device controls the room lights and is connected to a relay, which is operated through the digital pin D2.
- Device 2: This device incorporates a light sensor, which is an analog sensor connected to the analog pin A1. It measures the ambient light level in the room.
- Device 3: Similar to Device 2, this device is configured as a sensor and is also equipped with a light sensor. It is strategically placed in an area with minimal lighting within the room.

Devices used in this evaluation were: 2 Arduino Wemos D1 (Wi-Fi ESP-12F ESP266 802.11 b, g y n) Based on microcontroller ESP-8266EX 32-bit includes 11 Digital I/O pins and 1 pin for analog Input pin. 1 Arduino Wemos Mini (Wi-Fi ESP-12F ESP266 V.3 802.11 b, g y n). Based on microcontroller ESP-8266EX, includes 11 Digital I/O pins and 1 pin for analog Input pin. Centralized platform was running in a Windows 2016 Server Intel® Core™ i7 12700F with 32 GB or RAM and PCIe solid-state drive (SSD) disk.

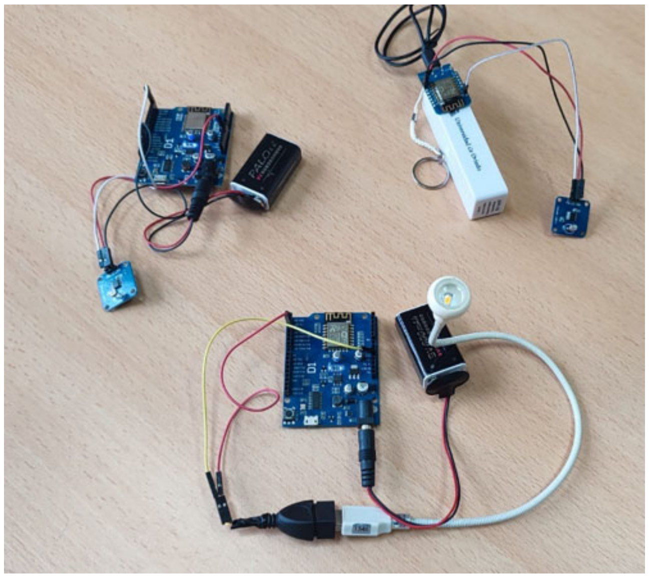


Figure 5: Three devices interconnected using the proposal. Device 1 Wemos D1 Wi-Fi ESP8266 with a light control system. Device 2 WeMos D1 Wi-Fi ESP8266 connected to an analog light sensor. Device 3 WeMos Mini D1 Wi-Fi ESP8266 connected to an light analog light sensor

By utilizing the platform, these three devices can seamlessly communicate and collaborate to automate the lighting process in the room. The system intelligently determines when to turn the lights on or off by analyzing the sensed light levels in the environment. This ensures that the lights are appropriately adjusted to provide optimal illumination based on the room's conditions.

The IoT system continuously and actively monitors the light levels at two different locations within the room in realtime. Should any of the light sensors detect a value below 600, the system promptly activates the lights. On the other hand, if the light level is above the specified threshold, the system will deactivate the lights.

In a traditional IoT implementation of this system, individual programs would need to be created for each device. Specifying the actions supported by each device. For instance, Device 1 would have actions like "turn on light" and "turn off light," while Devices 2 and 3 would have the action of "get light level". Numerous IoT platforms provide code templates to minimize the code developers need to write for each device. These three devices would be connected to a centralized system like ThingSpeak, ThingWorx, Azure IoT Hub, or Amazon AWS. Depending on the selected platform, developers might need to create a program or set up rules and notifications within the centralized system.

In our proposed approach, the firmware is installed on the devices. To generate this firmware, developers need to provide the network details on the website but should not modify the firmware program itself. Once this is done, the three devices can be connected to the platform. Developers can create a new script that incorporates the three devices using the Device class. In the setup function, the digital pins are configured as outputs, and the analog pins, which function as inputs, do not require any configuration. The main logic of the system is implemented in the loop function, which executes every 500 milliseconds. This function retrieves the values from the two analog sensors and, if either value is below 600, it triggers the lights to turn on. This logic serves as a basic example, but the loop function can include any JavaScript algorithm to define more complex conditions or analyses. Additionally, the system can incorporate third-party libraries using Node.js npm packages, enabling advanced features such as data analysis and artificial intelligence.

```
let device1 = new Device("d1");
let device2 = new Device("d2");
let device3 = new Device("d3");

let d1_pinLight = 'D2';
let d2_pinSensor = 'A1';
let d3_pinSensor = 'A1';

setup = () => {
  console.log('Setup');
  device1.pinMode(d1_pinLight, OUTPUT);
  // device2 A1 analog read not need configuration
  // device3 A1 analog read not need configuration
}

loop = () => {
  let valued2 =
    device2.analogRead(d2_pinSensor);
  let valued3 =
    device3.analogRead(d3_pinSensor);

  if(valued2 <= 600 || valued3 <= 600){
    device1.digitalWrite(d1_pinLight, HIGH);
  } else {
      device1.digitalWrite(d1_pinLight, LOW);
  }
}
```

The code based approach for configuring the system's functionality offers several advantages, particularly in reducing the time required to create the system. By leveraging programming structures, configurations and business logic can be simplified, especially in systems involving repetitive actions. For instance, consider the following system that effectively coordinates 16 devices with less than 50 lines of code. This program retrieves data from 8 devices equipped with analog light sensors and analyzes the number of sensors detecting low light levels. Using this count, the program activates a corresponding number of lights to compensate for the low light situation. By employing functional programming in JavaScript, this approach efficiently manages sensor structures and measurements, leading to a significant reduction in the required code compared to alternative implementation approaches.

```
let devicesSensors = []
let devicesSensorsNames =
    ["s1","s2","s3","s4","s5","s6","s7","s8"]
let devicesLights = []
```

(Continued)

(continued)

```
let devicesLightsNames =
   [“u1”,“u2”,“u3”,“u4”,“u5”,“u6”,“u7”,“u8”]

let pinLight = ‘D2’; // for all devices
let pinSensor = ‘A1’; // for all devices

setup = () => {
  console.log(‘Setup’);
  devicesSensorsNames.forEach (name, i => {
       let device = new Device(name)
       devicesSensors.push(device)
  })
       devicesLightsNames.forEach (name, i => {
           let device = new Device(name)
           device.pinMode(d1_pinLight, OUTPUT);
           devicesLights.push(device)
  })
}

loop = () => {
  let valuesSensors = []
  devicesSensors.forEach(device => {
       let value = device.analogRead(pinLight);
       valuesSensors.push(value);
  })

  let sensorsLows = valuesSensors.reduce(
  (accumulator, currentValue) => {
       if (currentValue < 600) {
        return accumulator + 1;
       } else {
        return accumulator;
       }
  }, 0);

  devicesLights.forEach(device, i => {
       if (i <= sensorsLows)
        device.digitalWrite(d1_pinLight, HIGH)
       else {
         device.digitalWrite(d1_pinLight, LOW);
       }
  })

}
```

5 Evaluation

The objective of this proposal is to streamline the process of creating IoT systems that involve interconnecting multiple devices for coordination and data collection, enabling them to perform actions based on conditions or analysis.

Currently, there are various IoT platforms available that facilitate the creation of such systems. These platforms eliminate the need for developers to individually implement the software for each device and the centralized system that coordinates them. Some popular examples include ThingSpeak, Azure IoT Central, and ArduinoCloud.

To evaluate the feasibility of implementing a centralized IoT system for enabling communication between two devices, we will consider the cost aspect. The system comprises Device 1 (Arduino Wemos) equipped with a light sensor and Device 2 (Arduino Wemos) equipped with a light switch. The centralized system will receive data from Device 1 and, based on that information, send an on or off command to Device 2. Although a simple use case has been designed for the evaluation, it is worth noting that the proposed solution can be extended to accommodate more complex coordination systems that may arise in the future.

To assess the process of defining this IoT system, we developed the functionality using different IoT platforms and closely monitored several parameters throughout the system creation process. The systems were created by a user who had prior knowledge of the IoT platforms, ensuring that no mistakes were made during the system creation. The following parameters were analyzed:

- (L) Lines of code added or modified in the code.
- (C) Characters modified in the code, excluding copy/paste actions. The parameter captures the extent of modifications made during the coding process.
- (N) Special configuration actions, which encompass additional tasks beyond the standard platform usage, such as installing software or IDEs for devices.
- (I) Interaction with the web interface, including clicks and keystrokes during program coding. The parameter excludes the registration process in any of the platforms. The Mousetron tool was used to measure this parameter.
- (M) Distance covered by the mouse cursor during the entire process. This parameter was measured using the Mousetron tool.
- (S) Number of different screens involved in configuring the system, such as managing devices, channels, and defining rules. Each unique screen represents a distinct step in the configuration process.
- (R) Number of different programs or websites used for creating the system. For example, using a website and the Arduino IDE. Some environments, like Arduino IoT Cloud, enable performing all actions directly from the website.

By analyzing these parameters, we can assess the efficiency and ease of use of different IoT platforms for creating the desired system.

The evaluated alternatives for creating the IoT system were as follows:

- Proposal: The developer registers the two devices and downloads the firmware, providing the necessary Wi-Fi credentials (SSID and password). The Arduino IDE is used to execute the firmware on both devices. Following this, the developer creates a program (code) to retrieve data from one device and define the coordination process.
- Azure IoT Central: Two Arduino firmwares are implemented based on the Azure/azure-sdk-c-arduino template. The developer needs to modify the firmware code and configure essential

parameters such as WiFi credentials, Azure host, and device information. Telemetry is used for sending data, and commands are used to execute actions. The Arduino IDE is used to execute the firmware on both devices. To implement synchronization, a rule is created in the Azure IoT portal, which offers a web interface without requiring code.

- ThingSpeak: Two Arduino firmwares are created based on the ThingSpeak Arduino template. The template includes communication and configuration parameters. Device 1 sends data to a ThingSpeak channel, and Device 2 requests updates from a ThingSpeak TalkBack, allowing remote command execution. The Arduino IDE is used to execute the firmware on both devices. Two conditions are added using ThingSpeak Reacts, a web interface that does not require coding.
- Arduino IoT Cloud: The developer registers the 2 devices in the Arduino IoT Cloud. To facilitate device communication, a shared cloud variable is created. Within the web IDE, the developer implements the 2 firmwares (code) that utilize the shared variable to coordinate their functionality. This platform enables coordination using code, similar to the proposal, without the need for an external IDE to deploy the firmware. However, the "Arduino Create Agent" software needs to be installed to access the devices from the website.

During the evaluation of these alternatives, multiple factors were taken into account, encompassing code length, configuration process, web interface interactions, mouse movement, and the number of screens utilized in the system creation process (Table 2).

Table 2: Parameters analyzed in all alternatives

	L	C	N	I	M	S	R
Proposal	8	234	1	307	8	4	2
Azure IoT central	33	977	1	1106	60	6	2
ThingSpeak	32	691	1	1269	25	5	2
Arduino cloud	**7**	**138**	1	390	22	5	2
No platform	229	4381	2	4868	57	2	1

When examining the code related parameters (L, C), Arduino Cloud emerged as the most efficient option, particularly in terms of character count (Fig. 6). This reduction is achieved by utilizing shorter code sentences that exclude device declarations and references. Nonetheless, Arduino Cloud demands the creation of two separate programs, although they are relatively short. Moreover, the additional step of creating a shared variable for program synchronization adds to the effort required for implementation. It is important to note that this synchronization is accomplished through web interfaces rather than code, resulting in the second group of parameters (N, I, M, S, R) being notably less favorable compared to the proposal.

When considering parameters that measure complexity outside of the code, we observed that the proposal stands out as the least demanding option. The reason for this notable reduction in interactions, such as keystrokes (I), mouse movement (M), and configuration screens (S), is mainly attributed to the emphasis on code based actions rather than relying on specific web interfaces (Fig. 7). This code centric approach streamlines the process and minimizes the need for extensive efforts when configuring the system. The proposal excels in minimizing the number of screens, mouse movements, and keystrokes needed for data entry. It outperforms the second ranked option, "Arduino Cloud," across all these indicators, and it is notably superior to Azure and ThingSpeak in this regard.

In order to establish a comprehensive comparison, we assigned weights to each analyzed parameter ($L = 10$, $C = 1$, $N = 90$, $I = 2$, $M = 7$, $S = 40$, $R = 100$) based on their significance in the IoT system creation process. These weights were assigned to ensure that the parameters that have a greater impact on the overall complexity of the system were appropriately considered in the evaluation.

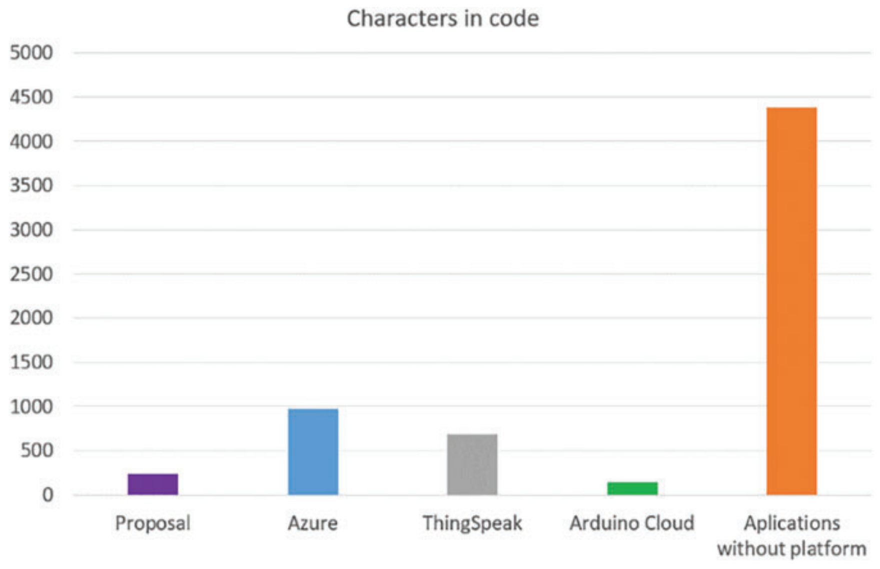


Figure 6: Comparison on characters in code. Vertical axis shows the number of characters

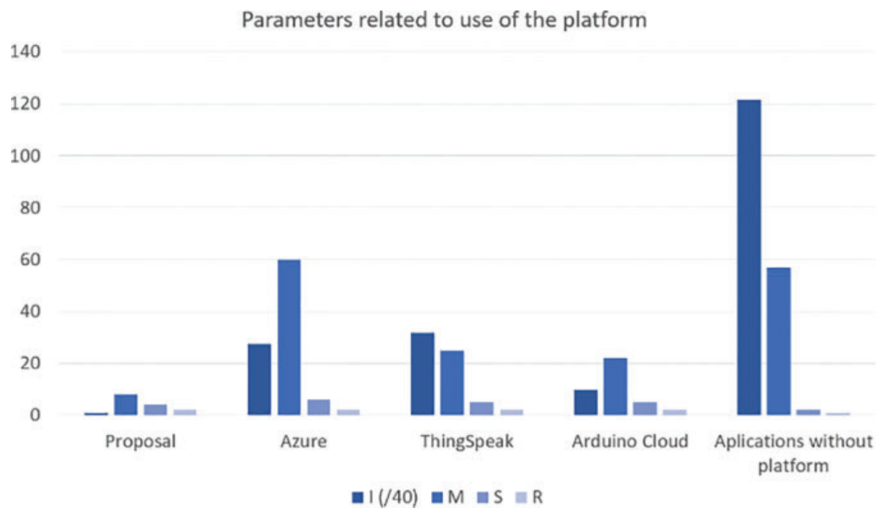


Figure 7: Comparison on parameters related to the use of web interfaces of the platforms and IDEs. Vertical axis shows the number of parameters (I, M, S, R)

In overall terms, the proposal exhibits a complexity index of 1434, which is 12.13% lower than the best alternative, Arduino Cloud (1632), and 65.49% lower than the third best alternative, ThingSpeak. It is worth noting that ThingSpeak and Azure have similar complexity indexes (Fig. 8). These findings suggest that the proposal offers a more streamlined and efficient approach, compared to the other evaluated alternatives. These results are explained, since the proposal is the one that defines the greatest abstraction in the communication and coordination between devices.

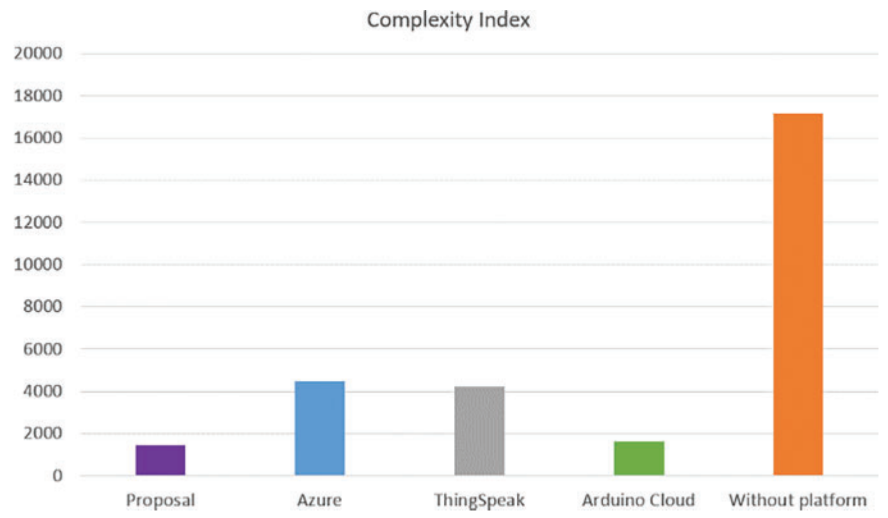


Figure 8: Global complexity. Vertical axis shows the total number of the index of completeness

6 Conclusions

This paper introduces a platform designed to simplify the creation of IoT systems that involve interconnecting smart objects. The proposal is based on the abstraction on devices communication and interconnection, and allowing the implementation of the entire IOT system in a single program.

While there are already several alternatives available to simplify the creation of IoT platforms for object interconnection, the primary objective of this research is to further enhance the ease of system creation. The hypothesis revolves around consolidating the entire system development into a single program, integrating both device logic and the centralized system. This approach sets itself apart from many existing platforms that often merge code editors with multiple web interfaces for rule definition, lacking a unified program for the entire IoT system. Based on the conducted evaluation, the proposal enables the swift creation of systems that effectively coordinate connected devices. It achieved the lowest overall complexity index among the 1434 alternatives, surpassing the second ranked "Arduino Cloud" by a margin of 12.13% and outperforming platforms like ThingSpeak and Azure IoT by almost 70%. These results indicate that the proposal holds potential as an agile solution for developing IoT systems of this nature.

The proposal presents a new development approach that can be very useful for the implementation of IoT systems. It allows to significantly reduce development costs compared to current alternatives. Moreover, the proposed approach enables developers with limited technical expertise to effectively communicate and interconnect IoT devices. This makes it particularly valuable for IoT systems that involve the coordination of multiple devices, each with distinct functionalities.

The proposal is efficient for the creation of IoT systems that do not have low data traffic or very low response times as essential requirements. It is based on a generic communication and coordination system, which may not be as efficient as a system implemented by an expert developer in communication, data compression, or high performance real-time systems.

We have identified several lines of future work within this proposal. Firstly, it facilitates the creation of dashboards and monitoring or remote control systems within the platform. Secondly, it includes protocol selections and coordination configurations based on artificial intelligence. These dynamic configurations can adapt during the system's lifecycle to improve efficiency and response times.

Acknowledgement: The authors are thankful to the School of Computer Science of the University of Oviedo for providing materials and resources for developing this research work.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: The authors confirm contribution to the paper as follows: design of the main idea of the research: Jordán Pascual Espada, Ruben Gonzalez Crespo; study of related work and definition of objectives: Jordán Pascual Espada, Ruben Gonzalez Crespo; solution design: Jordán Pascual Espada, Enol Matilla Blanco; solution implementation: Enol Matilla Blanco; analysis and interpretation of results: Jordán Pascual Espada, Enol Matilla Blanco. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Source code of the proposal solution can be downloaded at the following url https://lc.cx/mTUtLC, password: iot.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] J. Gubbi, R. Buyya, S. Marusic and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013. https://doi.org/10.1016/j.future.2013.01.010

[2] D. Gregorczyk, T. Bußhaus and S. Fischer, "A proof of concept for medical device integration using Web services," in *Int. Multi-Conf. on Systems, Signals & Devices*, Chemnitz, Germany, pp. 1–6, 2012. https://doi.org/10.1109/SSD.2012.6198124

[3] M. Yuriyama and T. Kushida, "Sensor-cloud infrastructure-physical sensor management with virtualized sensors on cloud computing," in *13th Int. Conf. on Network-Based Information Systems*, Takayama, Gifu, Japan, pp. 1–8, 2010. https://doi.org/10.1109/NBiS.2010.32

[4] S. H. Gampa, P. Yellamma, V. Ganta, C. Siram, A. R. S. Kamal *et al.*, "A review on smart home automation system using IoT with cloud computing," in *4th Int. Conf. on Electronics and Sustainable Communication Systems (ICESC)*, Coimbatore, Tamil Nadu, India, pp. 361–368, 2023. https://doi.org/10.1109/ICESC57686.2023.10193584

[5] A. Lakhan, M. A. Mohammed, J. Nedoma, R. Martinek, P. Tiwari *et al.*, "DRLBTS: Deep reinforcement learning-aware blockchain-based healthcare system," *Scientific Reports*, vol. 13, no. 1, pp. 4124, 2023. https://doi.org/10.1038/s41598-023-29170-2

[6] A. Lakhan, M. A. Mohammed, M. Elhoseny, M. D. Alshehri and K. H. Abdulkareem, "Blockchain multi-objective optimization approach-enabled secure and cost-efficient scheduling for the Internet of Medical Things (IoMT) in fog-cloud system," *Soft Computing*, vol. 26, no. 13, pp. 6429–6442, 2022. https://doi.org/10.1007/s00500-022-07167-9

[7] A. Hachani and N. Ajailia, "Towards new generation of civil engineers in the IoT era: PBL as a tool for integrating IoT in civil engineering curricula," in *IEEE Global Engineering Education Conf. (EDUCON)*, Kuwait, pp. 1–5, 2023. https://doi.org/10.1109/EDUCON54358.2023.10125234

[8] V. V. Reddy S, B. Jaison, A. Balaji, D. Indumathy, S. Vanaja *et al.,* "Agri-IoT: A farm monitoring and automation system using Internet of Things," in *Second Int. Conf. on Electronics and Renewable Systems (ICEARS)*, Tamil Nadu, India, pp. 639–642, 2023. https://doi.org/10.1109/ICEARS56392.2023.10085235

[9] S. G. Pantelimon, T. Rogojanu, A. Braileanu, V. D. Stanciu and C. Dobre, "Towards a seamless integration of IoT devices with IoT platforms using a low-code approach," in *IEEE 5th World Forum on Internet of Things (WF-IoT)*, Limerik, Ireland, pp. 566–571, 2019. https://doi.org/10.1109/WF-IoT.2019.8767313

[10] R. B. Gallinas, M. M. Sánchez, M. E. Beato and A. F. García, "An event mesh for event driven IoT applications," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 7, no. 6, pp. 54, 2022. https://doi.org/10.9781/ijimai.2022.09.003

[11] B. Kim, S. Yoon and Y. Kang, "PUF-based IoT device authentication scheme on IoT open platform," in *Int. Conf. on Information and Communication Technology Convergence (ICTC)*, Jeju Island, Korea (South), pp. 1873–1875, 2021. https://doi.org/10.1109/ICTC52510.2021.9620848

[12] J. Jung, J. Cho and B. Lee, "A secure platform for IoT devices based on ARM platform security architecture," in *14th Int. Conf. on Ubiquitous Information Management and Communication (IMCOM)*, Taichung, Taiwan, pp. 1–4, 2020. https://doi.org/10.1109/IMCOM48794.2020.9001713

[13] Y. Chai, Q. Wang and L. Yao, "Improvement and implementation of IoT device's program upgrade system on cloud platform," in *IEEE 5th Int. Conf. on Electronic Information and Communication Technology (ICEICT)*, Hefei, China, pp. 81–85, 2022. https://doi.org/10.1109/ICEICT55736.2022.9909393

[14] R. D. Murthy and M. Liu, "An elastic IoT device management platform," in *IEEE Global Conf. on Artificial Intelligence and Internet of Things (GCAIoT)*, Alamein New City, Egypt, pp. 1–6, 2020. https://doi.org/10.1109/GCAIoT51063.2020.9345907

[15] T. Rogojanu, M. Ghita, V. Stanciu, R. Ciobanu, R. Marin *et al.,* "NETIoT: A versatile IoT platform integrating sensors and applications," in *Global Internet of Things Summit (GIoTS)*, Bilbao, Spain, pp. 1–6, 2018. https://doi.org/10.1109/GIOTS.2018.8534552

[16] J. Gil, J. M. Torres and R. González-Crespo, "The application of artificial intelligence in project management research: A review," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 6, no. 6, pp. 54–66, 2021. https://doi.org/10.9781/ijimai.2020.12.003

[17] M. N. Fakhri, K. N. Bin Ramli, M. F. L. Abdullah, K. A. Omar, A. A. Qasim *et al.,* "Modeling middleware platform for supporting active communication between IoT devices," in *Int. Conf. on Information Science and Communication Technology (ICISCT)*, Karachi, Pakistan, pp. 1–7, 2019. https://doi.org/10.1109/CISCT.2019.8777420

[18] S. Pasha, "Thingspeak based sensing and monitoring system for IoT with matlab analysis," *International Journal of New Technology and Research*, vol. 2, no. 6, 2016.

[19] O. Jukić, I. Heđi and E. Ciriković, "IoT Cloud-based services in network management solutions," in *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)*, Opatija, Croatia, pp. 419–424, 2020. https://doi.org/10.23919/MIPRO48935.2020.9245117

[20] S. Forsström and U. Jennehag, "A performance and cost evaluation of combining OPC-UA and microsoft azure IoT Hub into an Industrial Internet-of-Things system," in *Global Internet of Things Summit (GIoTS)*, Geneva, Switzerland, pp. 1–6, 2017. https://doi.org/10.1109/GIOTS.2017.8016265

[21] C. N. Oton and M. T. Iqbal, "Low-cost open source IoT-based SCADA system for a BTS site using ESP32 and arduino IoT cloud," in *IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conf. (UEMCON)*, New York, USA, pp. 681–685, 2021. https://doi.org/10.1109/UEMCON53757.2021.9666691

[22] Z. Chaczko and R. Braun, "Learning data engineering: Creating IoT apps using the node-RED and the RPI technologies," in *16th Int. Conf. on Information Technology Based Higher Education and Training (ITHET)*, Ohrid, Macedonia, pp. 1–8, 2017. https://doi.org/10.1109/ITHET.2017.8067827

[23] M. Lekić and G. Gardašević, "IoT sensor integration to node-RED platform," in *17th Int. Symp. INFOTEH-JAHORINA (INFOTEH)*, East Sarajevo, Bosnia and Herzegovina, pp. 1–5, 2018. https://doi.org/10.1109/INFOTEH.2018.8345544

[24] P. Roy, J. Saha, N. Dutta and S. Chandra, "Microcontroller based automated room light and fan controller," in *Emerging Trends in Electronic Devices and Computational Techniques (EDCT)*, Kolkata, India, pp. 1–4, 2018. https://doi.org/10.1109/EDCT.2018.8405090

[25] M. A. Mohammed, A. Lakhan, K. H. Abdulkareem, D. A. Zebari, J. Nedoma *et al.,* "Energy-efficient distributed federated learning offloading and scheduling healthcare system in blockchain based networks," *Internet of Things*, vol. 22, pp. 100815, 2023. https://doi.org/10.1016/j.iot.2023.100815

[26] A. Lakhan, M. A. Mohammed, J. Nedoma, R. Martinek, P. Tiwari *et al.,* "Blockchain-enabled cybersecurity efficient IIOHT cyber-physical system for medical applications," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 5, pp. 2466–2479, 2023. https://doi.org/10.1109/TNSE.2022.3213651

[27] T. S. López, D. C. Ranasinghe, B. Patkai and D. McFarlane, "Taxonomy, technology and applications of smart objects," *Information Systems Frontiers*, vol. 13, no. 2, pp. 281–300, 2011. https://doi.org/10.1007/s10796-009-9218-4

[28] G. Kortuem, F. Kawsar, V. Sundramoorthy and D. Fitton, "Smart objects as building blocks for the Internet of Things," *IEEE Internet Computing*, vol. 14, no. 1, pp. 44–51, 2010. https://doi.org/10.1109/MIC.2009.143

[29] K. A. Pranoto, Y. R. Ramadhan, W. Caesarendra, A. Glowacz, S. K. Dash *et al.,* "Comparison analysis of data sending performance using the cayenne and ThingSpeak IoT platform," in *Int. Conf. on Informatics, Multimedia, Cyber and Information System (ICIMCIS)*, Jakarta, Indonesia, pp. 337–342, 2022. https://doi.org/10.1109/ICIMCIS56303.2022.10017648

[30] M. Irfan, H. Jawad, B. B. Felix, S. F. Abbasi, A. Nawaz *et al.,* "Non-wearable IoT-based smart ambient behavior observation system," *IEEE Sensors Journal*, vol. 21, no. 18, pp. 20857–20869, 2021. https://doi.org/10.1109/JSEN.2021.3097392

[31] A. S. Muhammed and D. Ucuz, "Comparison of the IoT platform vendors, microsoft azure, amazon web services, and google cloud, from users' perspectives," in *2020 8th Int. Symp. on Digital Forensics and Security (ISDFS)*, Beirut, Lebanon, pp. 1–4, 2020. https://doi.org/10.1109/ISDFS49300.2020.9116254

[32] R. Stackowiak, "Azure IoT solutions overview," in *Azure Internet of Things Revealed: Architecture and Fundamentals*. Berkeley, CA: Apress, pp. 29–54, 2019. https://doi.org/10.1007/978-1-4842-5470-7_2

[33] G. Guan, W. Dong, Y. Gao and J. Bu, "Towards rapid and cost-effective prototyping of IoT platforms," in *IEEE 24th Int. Conf. on Network Protocols (ICNP)*, Singapore, pp. 1–5, 2016. https://doi.org/10.1109/ICNP.2016.7785320

[34] S. G. Pantelimon, T. Rogojanu, A. Braileanu, V. D. Stanciu and C. Dobre, "Towards a seamless integration of IoT devices with IoT platforms using a low-code approach," in *IEEE 5th World Forum on Internet of Things (WF-IoT)*, Limerick, Ireland, pp. 566–571, 2019. https://doi.org/10.1109/WF-IoT.2019.8767313

[35] S. Forsström, U. Jennehag, P. Österberg, V. Kardeby and J. Lindqvist, "Surveying and identifying the communication platforms of the Internet of Things," in *Global Internet of Things Summit (GIoTS)*, Bilbao, Spain, pp. 1–6, 2018. https://doi.org/10.1109/GIOTS.2018.8534556