

We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

5,800

Open access books available

142,000

International authors and editors

180M

Downloads

Our authors are among the

154

Countries delivered to

TOP 1%

most cited scientists

12.2%

Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



Virtual Internet of Things Laboratory Using Node-RED

*Kalapatapu Venkata Sai Krishna Raja Shiva Kumar,
Attaphongse Taparugssanagorn and
Arunya Prasantha Senadeera Senadiri Dumunnage*

Abstract

Due to the pandemic or any other tough situations, attending the student laboratory session physically is difficult or even impossible. To overcome this problem, a virtual laboratory can be introduced. In many universities, virtual teaching methodology is not implemented widely. While there are several existing visual programming languages developed for various applications, an open-source visual programming language named Node-RED, which is particularly used for Internet of Things (IoT)-based applications, is taken into consideration in the book chapter. A complete outlook of Node-RED, which can be applied for an IoT-based virtual laboratory, is described. Its simplicity providing many built-in entities makes it possible to evolve innovative platforms with less coding complexity. The configuration, the flow development, the requirements, and the usage of the Node-RED are explained with respect to handling all the various types of errors. A few experimental cloud-based services are implemented on an IoT platform using serial port and message queuing telemetry transport (MQTT) broker service as well as providing live camera capturing feature with artificial-intelligence-based object detection. The results show that real-time IoT sensor data can be efficiently measured and visualized in dashboard and live object detection can be done with good accuracy.

Keywords: node-RED, visual programming language, open-source, Arduino MQTT, amazon cloud

1. Introduction

In this modern era, there has been a lot of development and upgradation in the field of Internet of things (IoT). The development has been actively growing in various sectors, for instance, industries, healthcare, education, agriculture, just to name a few. Consequently, it creates more employment opportunities. Thus, IoT should come into intensive usages with very careful awareness. The development of IoT programs can be done in various ways into the society. Although there have been a plenty of IoT development and applications, only very few academic institutions throughout the world are fully IoT equipped in their laboratories.

Since 2019, the COVID-19 pandemic has resulted in shutdown of educational institutions across the world. Globally, over 1 billion children are out of the classroom. As a result, education has changed dramatically with the distinctive rise of e-learning, whereby teaching is undertaken remotely and on digital platforms. While lecture sessions have been quite well organized, the laboratory sessions might be still challenging and dependent on the types of laboratories. To overcome this problem for an IoT design class, a virtual laboratory can be implemented. With a virtual laboratory, a laboratory session can be launched with flexibility of time. Self-learning process can be achieved by becoming more self-determining. It also supports collaborative group works and helps us develop critical thinking skills while doing individual assignments. The most important advantage is that it can be accessed from any location through Internet. Thus, any materials from the Internet can also be used. Nevertheless, there are some challenges in the virtual laboratory, which includes setting up the virtual laboratory environment, lack of social interaction between students and instructors, accessibility to various laboratory required technologies, software requirement and software inconsistency, student's issues while teaching simultaneously, etc. A visual programming language (VPL) is any programming language that lets users create programs by manipulating program elements graphically rather than by specifying them textually. A VPL allows programming with visual expressions, spatial arrangements of text and graphic symbols used either as elements of syntax or secondary notation. Many VPLs known as dataflow programming are based on the idea of boxes and arrows, where boxes or screen objects are treated as entities connected by arrows, lines, or arcs representing relations. Different prototyping boards have its own programming language, such as C, Python, Java, and similar. To develop an IoT system, at least one of these programming languages is required. However, it is fortunate that during these days, several VPLs have been developed to help us start programming without knowing the programming language. These IoT visual programming tools have a user-friendly approach of programming. It has a graphical user interface (GUI), where the user can just drag and drop moving code blocks and execute a simple piece of logic.

Node-RED VPL is an open-source software, which is a flow-based development tool originally developed by International Business Machines (IBM) for wiring hardware devices together with application programming interface (API) and few online services as part of the IoT. Node-RED can be used flexibly under the Apache2 license. Some developed their own services based on Node-RED, while others changed to their own user interface (UI) and deployed it as built-in. It can be established as a platform where we can publish our own developed node so that anyone can use it. The open-source software is an alternative to interact with the class. Students can practice, study, and understand with basic engineering skills or even develop an IoT-based system with artificial intelligence (AI) capabilities, e.g., prediction, regression, clustering, and classification. It can be installed locally on a personal computer (PC) or a laptop. With its simplicity for learning and using as well as several built-in entities, it can be used in evolving innovative platforms providing ability to assign code to all interfaces with less coding complexity. In addition, it supports several IoT prototyping board such as Arduino, Raspberry, and Android as well as cloud-based platforms.

The objectives of the book chapter are as follows.

- To describe the environment and features of Node-RED software with its abilities of numerous functions.

- To deliver a complete outlook on each function of Node-RED that can be applied for an IoT-based virtual laboratory.
- To talk over the suitability and rightness of machine learning and deep learning centered explanations in numerous practical fields.
- To arrange for a broad opinion on machine learning algorithms, which can be put to build-up the abilities of a data-driven approach.
- To highlight and summarize the possible study for smart systems including cloud technology.
- To bring out the virtual awareness/importance of IoT including laboratory sessions.

The rest of the book chapter is organized as follows. Section 2 presents the past related existing works on VPL and NodeRED. In Section 3, the step-by-step methodology of the work is described in detail, which includes getting started, configuration, utilization, and machine learning (ML) package for Node-RED. The experimental examples are provided along with the corresponding results in Section 4. Section 5 discusses about the results. Lastly, the conclusion is drawn in Section 6.

2. Related work

IoT has been playing an important role throughout the globe. It is a combination of both software and hardware tools. With the support of VPL, IoT can be applied into our day-to-day essential things. According to the survey by Ray in 2017, there were 13 VPLs for the usage and for the upgradation of IoT [1]. The VPL is classified into two types, namely open-source and proprietary-source with the four following main features, i.e., programming atmosphere, license, project source, and platform support [1]. Students can use open-source VPL such as Node-RED to learn and implement an IoT system or device with basic engineering skills and low-code programming. It helped many programmers to develop new software [2, 3]. Node-RED was initiated and was developed by IBM for connecting hardware devices with web-based editors [2, 3]. It was then applied further in the field of IoT. Rajalakshmi and Shahnasser came with a problem while making a cloud-system for IoT devices [4]. This is difficult to update the firmware by reinstalling the devices. To solve this issue, Node-RED was used without reinstalling the device and changing the programming code with quick setup [4]. In 2018, a model based on LoPy, which is a MicroPython triple-network development platform doubling up as a long-range (LoRa) Nano gateway, was connected to a system that uses Node-RED for interfacing with a local actuator and the external data with protocols [5, 6]. Their IoT development could be done in fog/clouds. The introduction of the IoT in education allows Internet-based communications to occur between things, sensors, and actuators. This has improved educational institutions [7–14]. In 2015, Giang et al. implemented a distributed Node-RED (D-NR) framework for building various types of IoT applications, which can work efficiently with simple designing process and less time [15]. Abdel-Basset et al. could make an efficient framework by new ideas with lower cost and greater security [16].

The advantages in maximizing IoT became more for the institutions, i.e., bringing out an affluent knowledge, better quality of working efficiency, and gaining real-time experiences [17]. In addition, they could keep track record of all the university resources with secured data accessibility. Using an open-source platform such as Node-RED, it helps for the creation of new ideas, i.e., linking up with many other specialized courses, for instance, information and communication technologies (ICTs), embedded system design, humanities, agricultural, just to name a few, yielding a good IoT program of study for students to learn and make research [16]. Another model was proposed by Marquez et al. as an IoT educational platform for virtual academic communities [17]. In 2020, Torres et al. used VPL with Node-RED to improve their IoT system by reducing the time taken for the development, i.e., reducing the number of failed attempts while deploying an IoT system [18]. Home automation consisting of water heater, cooling systems, electrical outlets was raised all over the IoT using cloud-platform where Node-RED is used to make fast setup for remote monitoring and control of data with a mutual communication [19]. In 2021, David et al. worked on indoor crop agriculture by monitoring parameters such as humidity, temperature, and light intensity with the help of IBM-Bluemix, which is the IBM open cloud platform providing mobile and web developers access to IBM software for integration security transaction and other key functions [20]. The data was transformed into Node-RED platform through a mobile application for tracking purpose of the farmers. Thuluva et al. made a solution for interoperability problem for IoT semantic web technologies in industrial field using semantic Node-RED models with feasibility and scalability approach [21]. A rapid and low-cost IoT prototype was developed by Ferencz and Domokos within less time on Node-RED using the combination of various cycle power plant dataset [22]. A low-power wide area network (LPWAN) technology called LoRa with its medium access control (MAC) layer protocol called LoRaWAN was deployed by Fox et al. in 2019, and these end devices interact with a gateway using Node-RED connection to an IBM-IoT platform [23]. Node-RED was applied and analyzed by Olsson and Eric in terms of modeling and security with misuse of API and providing security guarantees [24]. Clerissi et al. made a model for testing and developing IoT platforms using Node-RED with the functional behavior and the static view of the system. The class diagrams were used by testing, defining, and generating in java script using a Mocha test framework [25]. Proper regulations and rules for all the Node-RED developers were given by Clerissi et al. [26]. They are about the comprehensibility issues, which can be used to increase efficiency by reducing errors and time to complete tasks.

VPL is any programming language that lets users create program by manipulating program elements graphically rather than by specifying them textually [27, 28]. A VPL allows programming with visual expressions, spatial arrangements of text and graphic symbols used either as elements of syntax or secondary notation. For instance, many VPLs known as dataflow or diagrammatic programming are based on the idea of “boxes and arrows,” where boxes or other screen objects are treated as entities, connected by arrows, lines, or arcs representing relations [29–31]. There are several different VPLs, which can be divided into different fields of applications, such as education (23 languages), multimedia (26 languages), video games (18 languages), systems (34 languages), automation (4 languages), data warehousing (8 languages), legacy (5 languages), and miscellaneous (10 languages) [29–31]. The difference between a regular programming and visual programming, in terms of type, nature, flexibility, speed, efficiency, interface, learning complication, space usage, and example, is shown in **Table 1**.

No	Context	Regular programming	Visual programming
1	Type	Use only text	Use only graphics
2	Nature	Not user-friendly	User-friendly
3	Customizable	Very high	Moderate
4	Flexibility	High	Low
5	Speed	Very high	Low
6	Efficiency	Very high	Moderate
7	Interface	Not good	Great
8	Learning Complication	Take time	Easy to learn
9	Space Usage	Less	High
10	Example	Python, Java, etc.	Drakon, Helix, etc.

Table 1.
Difference between regular programming and visual programming.

Node-RED is preinstalled in many devices, such as Raspberry Pi, Intel, Fujitsu, just to name a few. Moreover, there are several different cloud services including Cisco, Nokia, IBM, Hitachi, etc., using Node-RED as a VPL and a dataflow programming.

3. Methodology

This section describes the complete scenario of methodology process. As can be seen in **Figure 1**, the first stage is installation, which involves the three following types, i.e., local machine, Raspberry Pi, and cloud services. After the correct installation, the Node-RED is configured involving all the basic nodes required to make a workflow with deployment activities. Next, the utilization is the main stage in which the development of the flows, handling errors of all the core nodes are explained. Accordingly, when the utilization process is completed, any type of experiments can be implemented with the expected and accurate outcomes.

3.1 Getting started

Getting started by installing Node-RED can be done on various criteria, which are explained as follows.

1. Running on a local machine for Windows 7 and above

Windows 7 and above versions have the capability of installing Node.js, Node-RED, and node package manager (npm) using command prompt (cmd) or Powershell. The procedure is described as follows.

- Node.js (version 14.x long term support (LTS) or above) is installed.
- All the local administrators are given their rights.

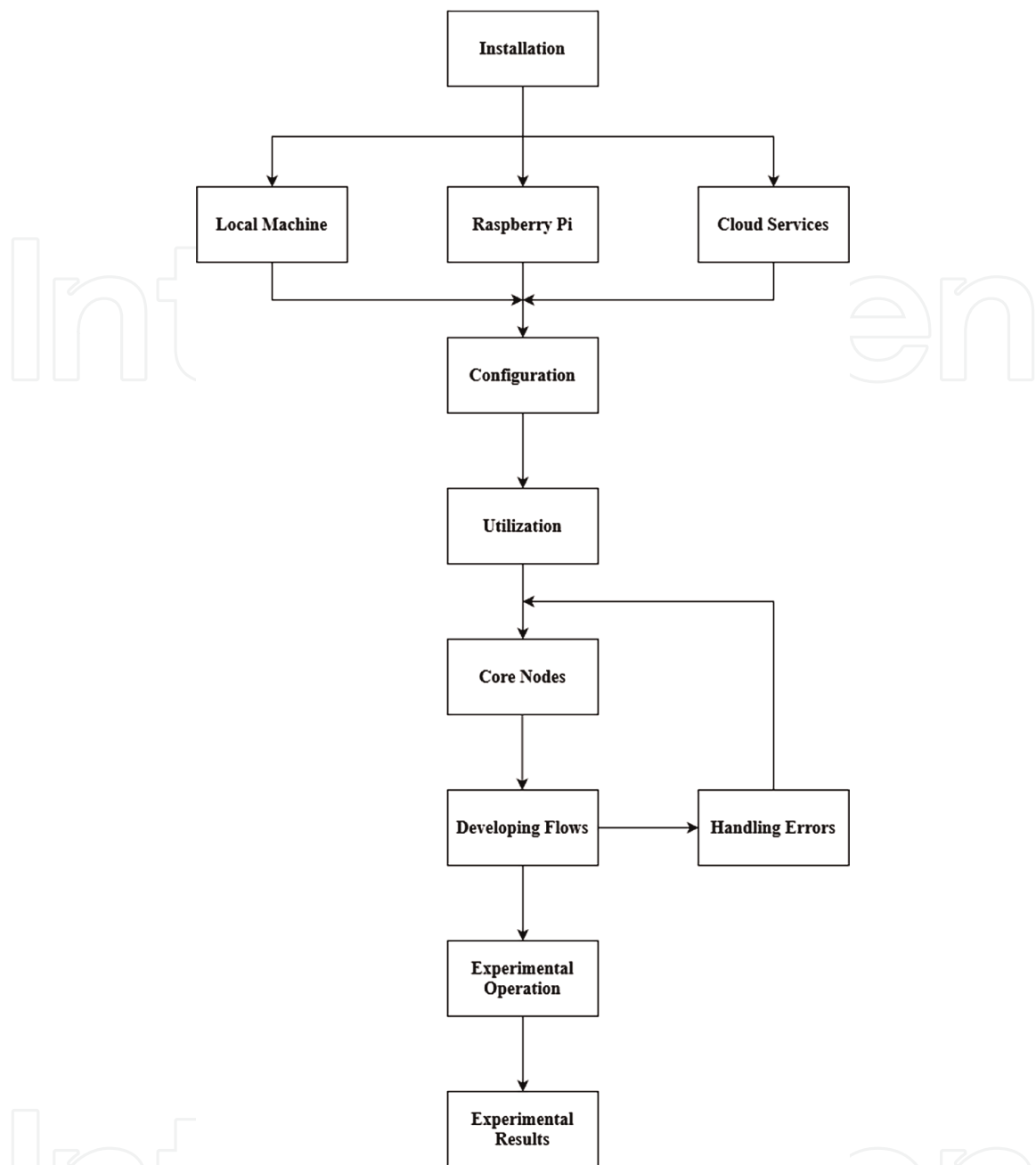


Figure 1.
Block diagram representing methodology workflow.

- Cmd or Powershell is opened and the following code is run to check if it is installed correctly or not installed.
- “node –version; npm –version” (Powershell),
- “node –version && npm –version” (cmd)
- Node-RED is installed by the command “npm install -g –unsafe-perm node-red.”
- As soon as installed, the cmd and type “node-red” is opened. As an output, a terminal log of Node-RED is displayed on the screen.

2. Running on a Raspberry Pi or any other IoT module

Buster is the supported version for Node-RED at present for all the Raspberry Pi Operating Systems (OS).

- The subsequent command is used to install node.js, npm and node-red on any Raspberry Pi.

```
"bash < (curl -sL https://raw.githubusercontent.com/node-red/linuxinstallers/master/deb/update-nodejs-and-nodered) "
```

(This command is used on any Debian-based OS like Ubuntu, DietPI.)

```
"bash < (curl -sL https://raw.githubusercontent.com/node-red/linuxinstallers/master/rpm/update-nodejs-and-nodered) "
```

(This command is used on any Red-Hat packet management (RPM) based OS like Red Hat, Fedora, CentOS, Oracle Linux.)

3. Running on different cloud services (IBM cloud, Amazon web services, Microsoft Azure)

Node-RED is used on different cloud-based services with many features depending on the user or the client requirements.

- For International Business Marketing (IBM) cloud: It is highly virtualized with high power, storage, networking, security, data management, analytics, developer tools, IoT, and integration and migration of virtual servers.
- For Microsoft Azure: It is also having the same abilities like IBM cloud with some extra added features such as replacing as an supplement for many other on-premise virtual servers with best recovery support.
- For Amazon web services (AWS): It has various new features when compared with many other cloud-services for developing new innovative smart devices, where it includes all the required API with less cost for the third-party usage.

3.2 Configuration

This section describes about various types of configurations that are needed for the deployment of each different application with default parameters in the default file directory.

- Normal application: Entire structure is loaded by default settings file, which is a built-in source.
- Embedded application: It is passed into property called RED.init(), where embedded.
- Run-time configuration: It defines the time-value of each node during the deployment.

- Logging configuration: Only console logging is supported in Node-RED.
- Node configuration: It is in the hierarchical format used for the application deployment.
- External module configuration: It defines about the run-time handling external npm modules and decides whether the editor allows new node modules to be installed such as the function node to have their own dynamic configured dependencies.
- Editor configuration: It is a set of files with different coding styles with various text editor plugins.

1. adminAuth: It permits security for user in the editor and admin API.

2. paletteCategories: It describes the sequence of types in the palette. By default, the pattern is subflow, common, function, network, sequence, parser, storage.

The run-time configuration is explained as follows.

- Flow file: It is used to store the flows.
- Userdir: It is used to store user data, credentials, and library data.
- Nodesdir: It is used to search additional installed nodes.
- Uihost: It is an interface to listen all connection on IPv4.
- Uiport: It is a port to serve ui editor.
- Httpadminroot: It is the root url, which contains both API and editor UI.
- httpAdminAuth: It allows HTTP validation on the editor UI.
- httpAdminMiddleware: It is an array of all functions, which is added to all admin routes.
- httpNodeRoot: It is the node root for all the urls that run HTTP at all endpoints.
- httpNodeAuth: It enables HTTP Basic Authentication
- httpRoot: It enables the root url to run on both admin and node endpoints through overriding httpAdminRoot and httpNodeRoot value.
- https: It permits https.
- httpStaticAuth: It supports basic confirmation and validation of HTTP with the static content.

- **httpNodeCors:** It is source distribution for the nodes, which are responsible for HTTP endpoints with cross-origin authentication.
- **httpNodeMiddleware:** It permits custom processing. For example, validation is required for the node.

Various stages of logging configuration used in Node-RED are described as follows.

- **Fatal:** Errors that make application unworkable are tracked.
- **Error:** Tracked errors for requests and fatal errors.
- **Warn:** Record of the problems about non-fatal and fatal errors.
- **Info:** Tracked information of application, warnings, error, and fatal errors.
- **Debugging (Debug):** Tracked information, which is more verbose than information, warnings, error, and fatal errors.
- **Tracing (Trace):** Tracks about all complete logging, debugging, info, warnings, error, and fatal errors.

The node configuration is explained as follows.

- **Function Node:** It is for gathering bits and pieces to attach into universal functions.
- **functionExternalModules:** It allows adding additional modules that are available to the function.
- **Debug Node:** Any message directed to the debug sidebar tab with maximum size and characters.
- **MQTT Nodes:** If the link is misplaced, how much time to pause in milliseconds before trying to connect.
- **Serial Nodes:** How much time to pause in milliseconds before making an effort to revive into serial port
- **socketReconnectTime:** How much time to pause in milliseconds before trying to connect again.
- **socketTimeout:** How much time to pause in milliseconds before scheduling out any port.

3.3 Utilization

3.3.1 Core nodes

The important nodes used for the basic functioning of the Node-RED are called “core nodes.” The main six types of core nodes are explained as follows.

1. Inject: It starts any flow manually by clicking the inject button. It can be at required intervals or any time within the editor.
2. Debug: It is used to show messages in the Debug sidebar in the editor, and the control on the node can be used to permit or restrict its outcome.
3. Function: It permits JavaScript code to run beside the messages delivered from it.
4. Change: It is used to transform message properties and fix context properties without changing a Function node with multiple operations such as set, change, move, delete.
5. Switch: It permits messages to be routed into different divisions of a flow by means of calculating set of instructions compared with each message with rules values, sequence, expression, otherwise property.
6. Template: It is used to produce text by message properties to fill the template.

3.3.2 Developing flows

It is the section in which all the necessary placement and arrangement of the nodes are done for the successful execution on the Node-RED.

1. Flow structure: It helps us organize flows, approaches for splitting into smaller, reusable components, and how to modify them to make use in different platforms.
2. Message design: It helps how to design messages to create nodes and flows, which can work together with any number of nodes and are easier to maintain.
3. Documenting flows: It helps about making or providing documentation on what tools and techniques Node-RED provides.

3.3.3 Handling errors

It is described as tracing out the bug or an error, which helps to reduce the developing time and correct the errors easily. The different types of errors on Node-RED are explained as follows.

1. Logging error: It displays the error with the date and time of the error and the node, which is noted as an error.
2. Catchable error: It will not be logged, but it informs about run-time error. Then, the Catch node will be used to produce a flow which can handle it.
3. Sub-flow error: It will not be logged but it informs about run-time error. Then, the Catch node will be used to produce a flow, which can handle it.

- 4. Uncatchable errors: If the error is written in the log, then a message is seen in the Debug sidebar and log outcome. But creation of a flow is not possible to handle it.
- 5. Uncaught errors: It causes the Node-RED run-time to shut down and cannot be controlled in the flow as they are produced by bugs in nodes.
- 6. Status changing errors: It is used to control modifications in node position by including the position property which provides the data about the position with the node that caused the incident.

3.4 Machine learning (ML) package for node-RED

This node-red-contrib-machine learning module for Node-RED contains a set of nodes offering machine learning functionalities. Such nodes have a python core that takes advantage of common ML libraries such as SciKit-Learn and Tenserflow. Classification and outlier detection can be performed using this package.

3.4.1 Usage

These flows create a dataset, train a model, and then evaluate it. Models, after training, can be used in real scenarios to make predictions.

Flows and test datasets are available in the “test” folder. We need to make sure that the paths specified inside nodes’ configurations are correct before trying to execute the program. “node-red” can be run from the folder “.node-red/node-modules/node-red-contrib-machine-learning” and the paths will be automatically correct. The flow shown in **Figure 2** loads a csv file, shuffles it, and creates a training and a test partition.

The flow shown in **Figure 3** loads a training partition and trains a “decision tree classifier” and then saves the model locally.

The flow shown in **Figure 4** loads a test partition and evaluates a previously trained model.

Figure 5 shows the flow of how to use a trained model during deployment. Data is received via mqtt, predictions are made and then sent back.



Figure 2.
Flow for loading a csv file, shuffling it, and creating a training and a test partition.

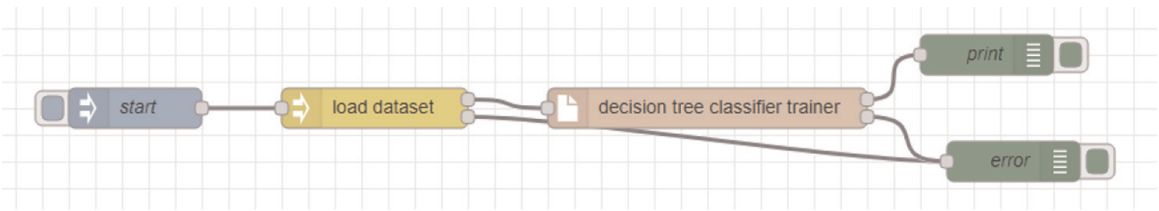


Figure 3.
Flow for loading a training partition and training a “decision tree classifier”.

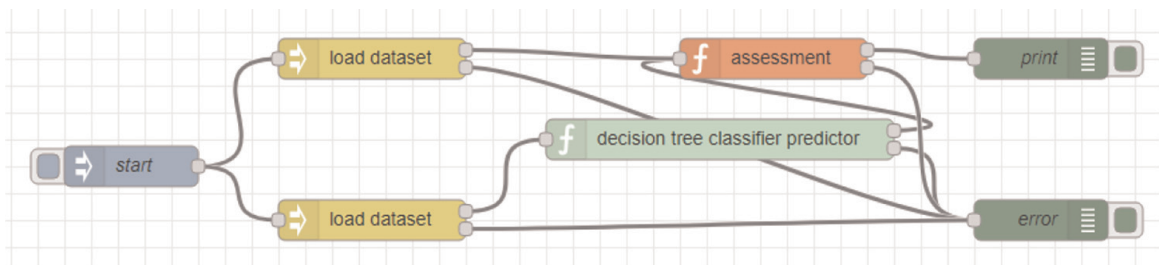


Figure 4.
Flow for loading a test partition and evaluating a trained model.

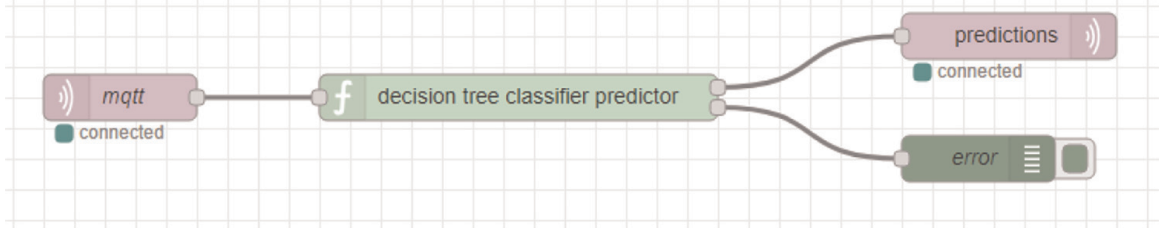


Figure 5.
Flow for showing how to use a trained model during deployment.

4. Experimentation

In this section, various types of experiments related to IoT are provided. ESP32, which is a microcontroller chip manufactured by Espressif Systems, is applied. It consists of a low-cost and a low-power chip with features such as Wi-Fi (IEEE802. 11 b/g/n), Bluetooth, and built-in antenna. The distributed hash table (DHT11) is a basic, ultralow-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air and spits out a digital signal (using an 8-bit microcontroller unit (MCU)) on the data pin. An Arduino integrated development environment (IDE) is used to write and upload programs to the Arduino compatible boards supporting the languages C and C++. Message queuing telemetry transport (MQTT), which is a lightweight messaging protocol, is used on such a small microcontroller that allows messaging between device to cloud and cloud to device with supporting several IoT devices. By using MQTT commands can be sent to control outputs, data can be read and published from sensors. Therefore, communications between multiple devices can be established. We can send a command with a client to control outputs, or we can read data from a sensor and publish it to a client.

4.1 Experimental examples

Example 4.1.1 DHT11 Sensor Data on Node-RED using a Serial Port (ESP32).
The procedure of this task is described as follows.

1. The ESP32 is connected to the DHT11 sensor as shown in **Figure 6** (GND TO GND, 3.3 V TO VCC, GPIO4 TO DATA PIN).
2. If the connections are correct, then a light present on the DHT11 sensor is turned on.

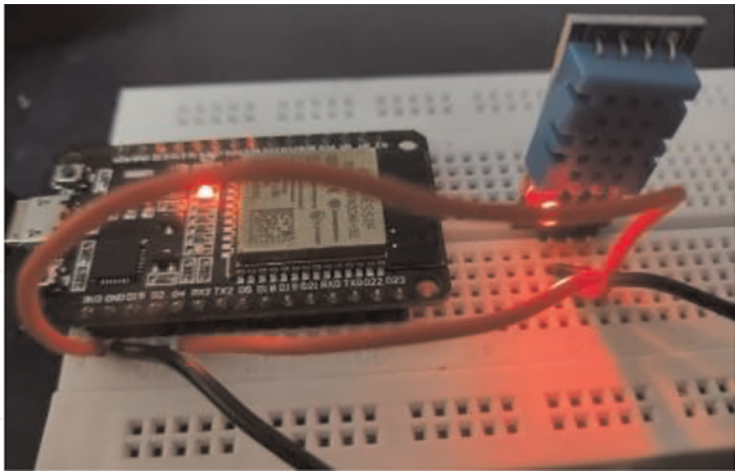


Figure 6.
ESP32 connection with DHT11 temperature sensor (serial port).

3. The DHT11 sensor is programmed into the Arduino integrated development environment (IDE) and uploaded into the ESP32 to show the corresponding temperature and humidity.
4. If the required packages are absent, then the Arduino IDE shows an error while uploading the code.
5. After the code is successfully uploaded, we can click on the serial monitor to see the corresponding temperature and humidity readings of the DHT11 sensor.
6. The Node-RED software is installed into the system. Then, the local hosting address of the Node-RED is open. This can be run on any browsers.
7. The required nodes, i.e., dashboard and serial port, are installed.
8. A flow is created as shown in **Figure 7**.

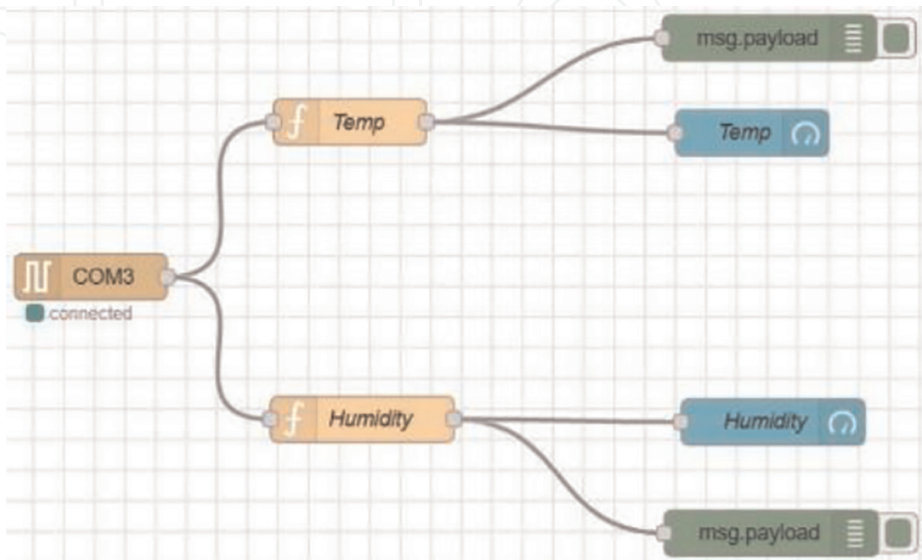


Figure 7.
Node-RED serial port connection for temperature and humidity.

9. A serial is inserted into the port node, one function node for temperature and another function node for humidity, which display the DHT11 sensor data.
10. To get the representation, two-gauge nodes for temperature and humidity are inserted and connected to their respective function nodes.
11. Finally, the overall workflow is deployed.

Example 4.1.2 MQTT Broker Service (ESP32) on Node-RED with DHT11 Sensor Data. The procedure of this task is described as follows.

1. The ESP32 is connected to the DHT11 sensor as shown in **Figure 8** (GND TO GND, 3.3 V TO VCC, GPIO4 TO DATA PIN).
2. If the connections are correct, then a light present on the DHT11 sensor will be turned on.
3. The DHT11 sensor is programmed into the Arduino (IDE) and uploaded into the ESP32 to show the temperature and humidity.
4. If the required packages are absent, then the Arduino IDE shows an error while uploading the code.
5. After the code is successfully uploaded, click on the serial monitor to see the temperature and humidity readings of the DHT11 sensor.
6. The Node-RED software is installed into the system. Then, the local hosting address of the Node-RED is open. This can be run on any browsers.
7. The required nodes, i.e., dashboard, MQTT-in, and MQTT-out nodes, are installed.
8. A flow is created as shown in **Figure 9**.

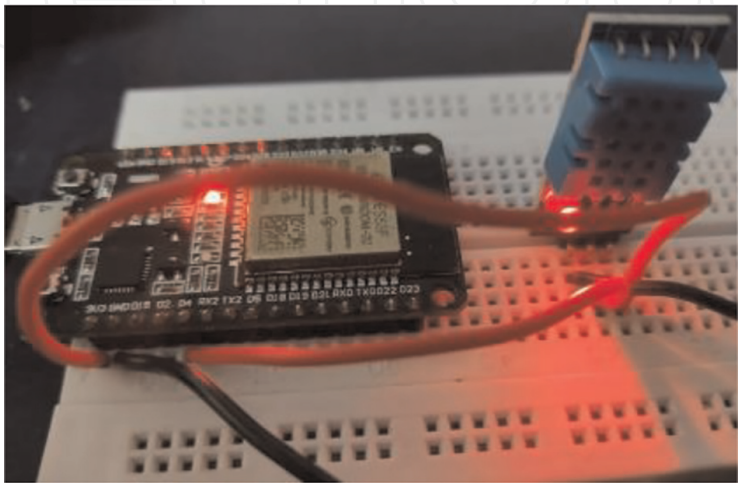


Figure 8.
ESP₃₂ and DHT₁₁ temperature sensor (MQTT broker service).

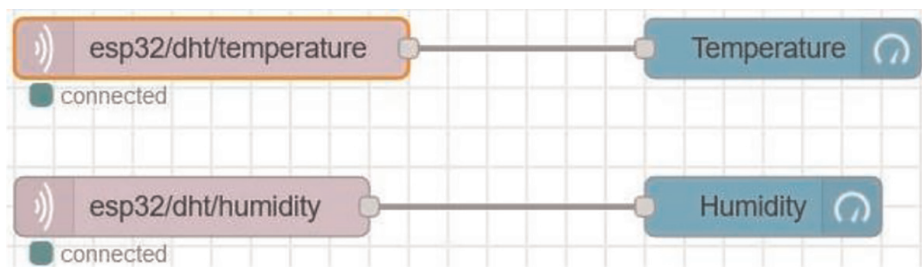


Figure 9.
MQTT broker service for temperature and humidity with node-RED.

- Two MQTT-in nodes, one for temperature and another for humidity, are inserted.
- This extracts the sensor data from the ESP32 through the MQTT broker service and gets into the Node-RED platform with the help of the local-host ip address.
- To display this data, the one gauge is connected to the temperature node and the other is connected to the humidity node.
- Finally, the overall workflow is deployed.

Example 4.1.3 Live Camera Capture and Object Detection using Machine Learning (ML).

In this example, a user interface (UI) has a major role for making an interface between the user and the Node-RED dashboard by the means of a system camera and a UI-table. **Figure 10** shows the procedure of this task, which is described as follows.

- The Node-RED software is installed into the system. The local hosting address of the Node-RED is opened. This can be run on any browser.
- The required nodes, i.e., random, dashboard, UI-table, UI-webcam, tfjs-coco-ssd, tf-model, tfjs-node (tf = tensorflow), are installed.

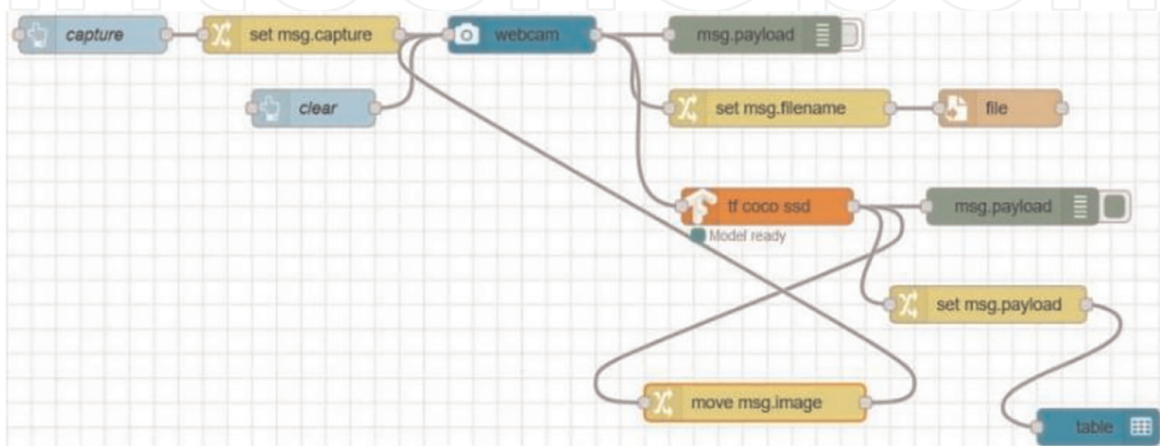


Figure 10.
Node-RED flow for live camera capture and object detection using machine learning (ML).

3. We connect the nodes by inserting button node = 2 (capture and clear), change node = 4 (set msg.capture, set msg.filename, set msg.image, set msg.payload), UI-webcam node = 1, UI-table node = 1, tfjs-coco-ssd = 1, debug node = 2 (msg.payload), file node = 1.
4. We check whether the camera of the PC/laptop is working or not.
5. After deploying successfully, we go to the hosting ip address followed by /ui, which opens another new webpage showing the output of the live web camera.

The object detection flow recognizes objects in an image and annotates objects with bounding boxes. An image can be loaded from a built-in camera, the file system, or by injecting the default image. We need to make sure that we have the node-red-contrib-browser-utils package installed for all these input nodes to work. This flow uses three of the custom nodes mentioned above (tf-function, tf-model, and post-object-detection). The loaded image is passed into the preprocessing node as msg.payload. The msg object is a JavaScript object that is used to carry messages between nodes. By convention, it has a payload property containing the output of the previous node. The preprocessing function node is an example of tf-function that directly calls the tf.node.decodeImage method with the predefined tf variable. The node produces a Tensor4D image representation as the payload and then passes it to the COCO SSD lite node, which is an instance of the tf-model custom node. This loads the COCO-SSD lite model.json from an external URL and runs inference on the model.

The result of the model goes through the postprocess node that returns an object array containing bbox, className, and score properties. The objects node combines an additional property, complete that is set to true, to the msg with the image object. Then, the bounding-box node draws bounding boxes on the input image and displays it in the browser.

4.2 Experimental results

Example 4.2.1 DHT11 Sensor Data on Node-RED using a Serial Port (ESP32).

Once after the successful deployment, we go to the hosting ip address followed by:1880/ui, which opens another new webpage representing the gauge readings of the temperature and the humidity of the DHT11 sensor. The temperature and humidity readings are directly sent from the serial port to the Node-RED directly. **Figure 11**

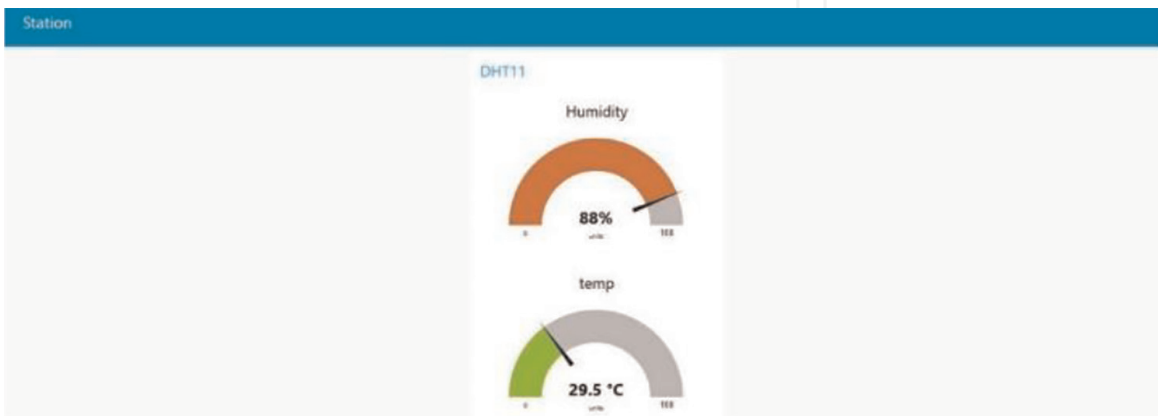


Figure 11.
Dashboard of temperature and humidity from DHT11 sensor (serial port).

shows an example of the dashboard showing temperature and humidity from the DHT11 sensor (serial port).

Example 4.2.2 MQTT Broker Service (ESP32) on Node-RED with DHT11 Sensor Data.

Once after the successful deployment, go to the hosting ip address followed by:1880/ui, which opens another new webpage representing the gauge readings of the temperature and the humidity of the DHT11 sensor. Temperature and humidity readings are sent from the sensor to MQTT broker (as a cloud), then finally sent to the Node-RED as a third-party service. **Figure 12** shows an example of the dashboard showing temperature and humidity from the DHT11 sensor (MQTT broker service).

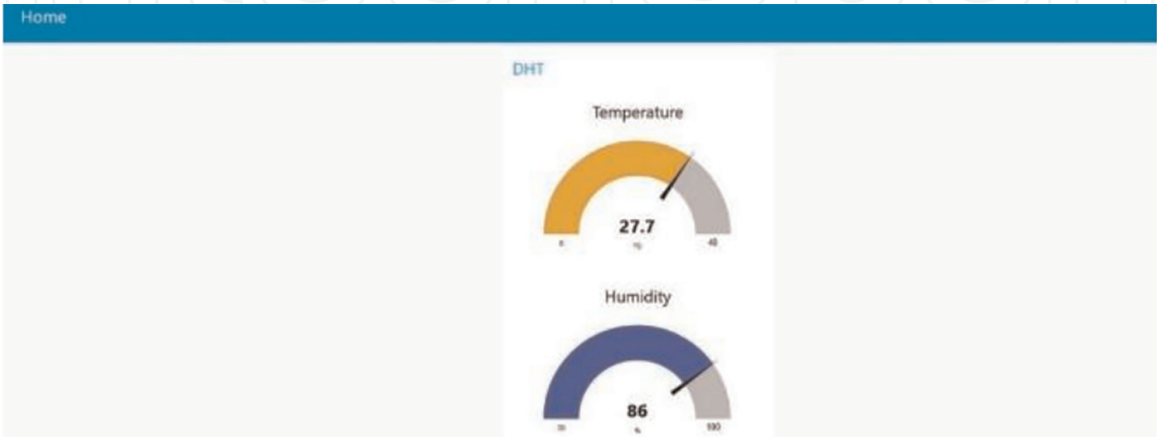


Figure 12.
Dashboard of temperature and humidity from DHT11 sensor (MQTT broker service).

Example 4.2.3 Live Camera Capture and Object Detection.

When the deployment is successful, go to the hosting ip address followed by:1880/ui, which opens another new webpage asking for the permission to allow camera

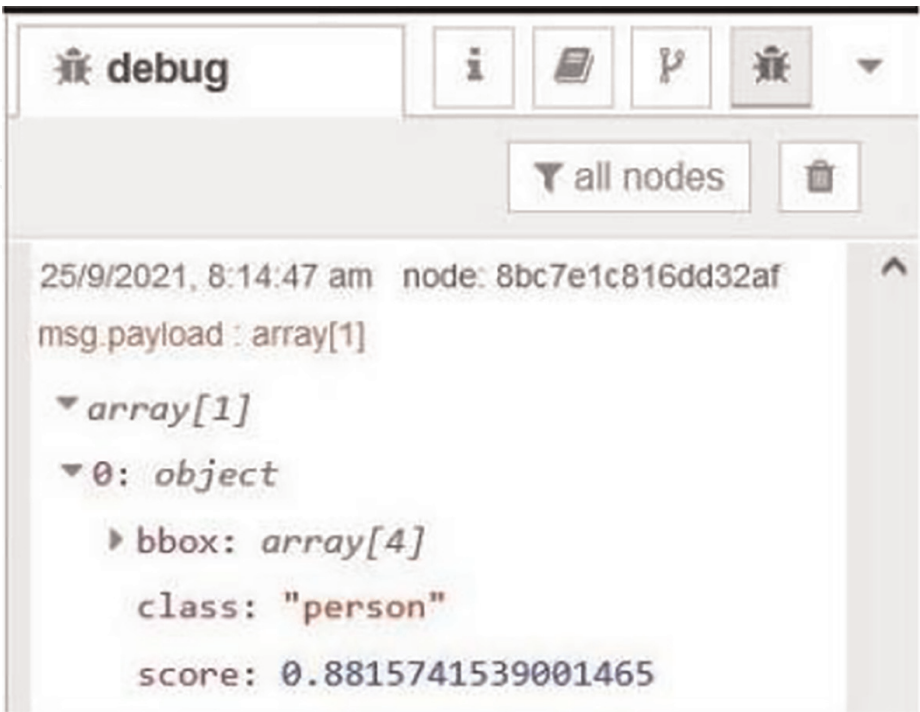


Figure 13.
Debug message showing single object class and accuracy score.

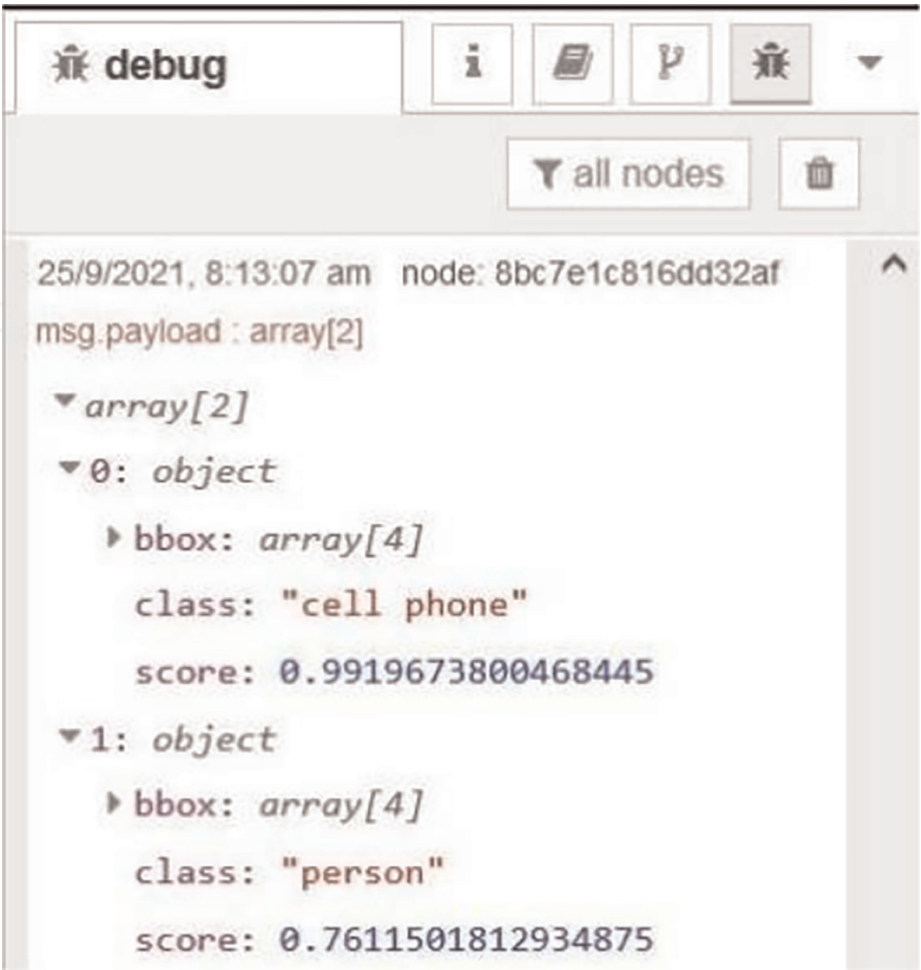


Figure 14.
Debug message showing dual object class and accuracy score.

(Press allow). Then, we can see the live camera working and press the capture button to capture. It detects the object with the help of TensorFlow analysis node as earlier describe and the output can be seen on the debug panel with the accuracy score of its correctness. The examples of the single object and the dual object detection are shown in **Figures 13** and **14**, respectively.

4.3 Implementing as a mobile application

From the given examples, they can also be made visible on a mobile phone for both iOS and Android. It is done through remote access just by installing the application named “RemoteRED” from the mobile app store. The steps are explained as follows.

1. Install the Remote-RED node in the Node-Red.
2. Open the Remote-RED settings and configure it to get the QR code.
3. Open the Remote-RED application from the mobile phone and scan the QR code.
4. Wait for a minute as it is asynchronous to get updated on the mobile phone.

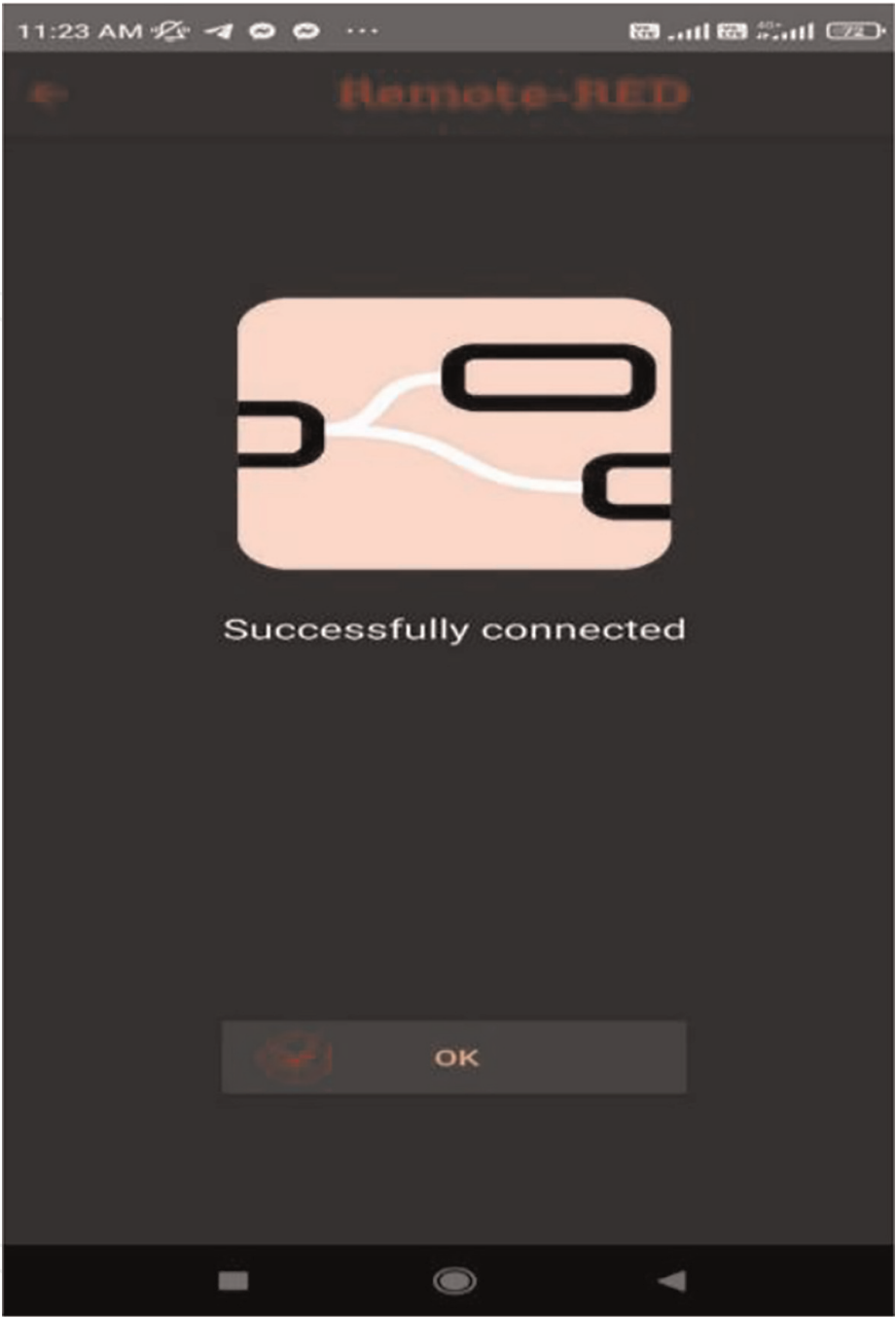


Figure 15.
Connecting to remote-RED application (iOS/android).

- 5. Now we can see the same output of the Node-RED on the mobile phone Remote-RED application as shown in **Figure 15**.
- 6. This process is the same and it works for any Node-RED workflow.

5. Discussion

The results show that simulated systems developed using the Node-RED can be deployed in the real world without changing the system parameters. Moreover, the

simulated system can be connected with hardware platforms easily. VPL-based hands-on tools are effectively smoothing the beginner's learning curve on IoT. The stranded network protocols and IoT protocols are easy to understand using hands-on experience and deploy in the Node-RED environment. Node-RED system models are easy to debug compared with the traditional programming debug methods. The platform works well in both Internet and Intranet mode as all the components are virtually presented in Node-RED. After all, the laboratory instructors can effectively convey the basics of network and IoT concepts to the students through a VPL language including the Node-RED.

6. Conclusion

The book chapter described a complete outlook of Node-RED that can be applied for an IoT based virtual laboratory. The configuration, the flow development, the requirements, and the usage of the Node-RED were explained with respect to handling all the various types of errors. We modeled, implemented, and tested the IoT virtual laboratory using Node-RED. The implemented virtual laboratory system is currently serving for flexible postgraduate programs and broadcasting completely online. Students obtain great encouragement and motivation toward virtual IoT hands-on practices as they can manage their own and convenient time and a location. The virtual laboratory concept utilizes the available hardware (Wi-Fi IEEE802. 11 b/g/n routers, classical Bluetooth hardware, and system-on-chip (SoC) like MCU) at the student location. The virtual IoT laboratory concept was proved to help students to learn faster than a classical theory class or a video-recorded lesson. With the simplicity of Node-RED and its built-in entities a few examples, which can be used for an IoT-based virtual laboratory, were done to evolve innovative platforms with less coding complexity. It provides flexibility such that the remote laboratory can run on several operating systems or on a mobile application. The proposed solution is platform-independent, and therefore, it can be implemented on low-cost hardware for smaller systems, and the clients can run on mobile devices. The first and second examples are typical scenarios in IoT on Node-RED platform including Arduino and third-party cloud service, whereas the third example is live camera capture with object detection capability. Finally, it was also shown about how Node-RED can be used as a mobile application remotely. After the completion of the book chapter, the readers are supposed to be able to develop IoT systems using VPL-Node-RED, integrate IoT with Node-RED, to design various workflows for IoT on Node-RED, develop real-time IoT applications, and apply security features for IoT and Node-RED while using cloud-based services. Hence, IoT with Node-RED has the capability to change the entire education system, which makes better learning with good interaction and flexibility.

The variety of IoT devices in the future is expected to dramatically grow. Our Node-RED IoT platform was designed to be extendible. To model these future scenarios, we expect to support new IoT device types by using the generic framework for creating new devices. An example of such devices could be wearables that generate movement data. This type of data can be emitted at high volumes but small size. Another device type, which is expected, is an IoT device that is configurable at runtime. This would give us the ability to change the behavior of the device according to an operation plan. An example of this type of device would be a smart home thermostat. Furthermore, we also want to improve the intelligence of our smart testing framework providing machine learning libraries that can continuously learn

from past data to improve prediction accuracy. Finally, empirical research can be done collecting a large amount of data in order to know the impact of virtual IoT laboratories on the students with their involvement as it helps the instructors to improve their laboratory sessions further.

Conflict of interest


The authors declare no conflict of interest.

Author details

Kalapatapu Venkata Sai Krishna Raja Shiva Kumar, Attaphongse Taparugssanagorn* and Arunya Prasantha Senadeera Senadiri Dumunnage
School of Engineering and Technology, Asian Institute of Technology, Phatum Thani, Thailand

*Address all correspondence to: attaphongset@ait.asia

IntechOpen

© 2022 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/3.0>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. 

References

- [1] Ray PP. A survey on visual programming languages in internet of things. *Scientific Programming*. 2017; **2017**:1-6. DOI: 10.1155/2017/1231430
- [2] Chaczko Z, Braun R. Learning data engineering: Creating iot apps using the node-red and the rpi technologies. In: *Proceedings of the 16th International Conference on Information Technology Based Higher Education and Training (ITHET'17)*; 10–12 July 2017. Ohrid, Macedonia: IEEE; 2017
- [3] Node_RED Node-Red Contribution [Internet]. 2017. Available from: <https://flows.nodered.org/node/node-red-contrib-pdf> [Accessed: 2022-02-28]
- [4] Rajalakshmi A, Shahnasser H. Internet of things using node-red and alexa. In: *Proceedings of the 17th International Symposium on Communications and Information Technologies (ISCIT'17)*; 25–27 September 2017. QLD, Australia: IEEE; 2017
- [5] Debauche O, Mahmoudi S, Mahmoudi SA. Internet of things: Learning and practices. Application to smart city. In: *Proceedings of the 4th International Conference on Cloud Computing Technologies and Applications (Cloudtech'18)*; 26–28 November 2018. Brussels, Belgium: IEEE; 2018
- [6] Alejandro A. Pycom Launches World's First Triple Network Low Power WAN, WIFI and Bluetooth Dev Module on Kickstarter [Internet]. 2016. Available from: <https://pycom.io/world-s-first-triple-network-dev-module/> [Accessed: 2022-02-28]
- [7] Giang NK, Blackstock M, Lea R, Leung VC. Developing iot applications in the fog: A distributed dataflow approach. In: *Proceedings of the 5th International Conference on the Internet of Things (IOT)*; 26–28 October 2015. Seoul, Korea (South): IEEE; 2015
- [8] Abdel-Basset M, Manogaran G, Mohamed M, Rushdy E. Internet of things in smart education environment: Supportive framework in the decision-making process. *Concurrency and Computation: Practice and Experience*. 2018;**31**(10). DOI: 10.1002/cpe.4515
- [9] Veeramanickam M, Mohanapriya M. IOT enabled Futurus smart campus with effective E-learning: i-campus. *GSTF Journal of Engineering Technology (JET)*. 2016;**3**(4):81
- [10] Elyamany HF, AlKhairi AH. IoT-academia architecture: A profound approach. In: *Proceedings of the IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*; 1–3 June 2015. Takamatsu, Japan: IEEE/ACIS; 2015
- [11] Marquez J, Villanueva J, Solarte Z, Garcia A. IoT in education: Integration of objects with virtual academic communities. In: *Proceedings of the World Conference on New Advances in Information Systems and Technologies*; 02 March 2016. Recife, Brazil: Springer; 2016. pp. 201-212
- [12] Chin J, Callaghan V. Educational living labs: A novel internet-of-things based approach to teaching and research. In: *Proceedings of the 9th International Conference on Intelligent Environments (IE)*; 16–17 July 2013. Athens, Greece: IEEE; 2013
- [13] Wang Y. English interactive teaching model which based upon internet of things. In: *Proceedings of the*

International Conference on Computer Application and System Modeling (ICCASM); 22–24 October 2010. Taiyuan, China: IEEE; 2010

Conference on Green Computing and Internet of Things (ICGCIOT'18); 16–18 August 2018. Bangalore, India: IEEE; 2018. pp. 386-390

[14] Bagheri M, Movahed SH. The effect of the internet of things (IoT) on education business model. In: Proceedings of the 12th International Conference on Signal-Image Technology and Internet-Based Systems (SITIS); 28 November-1 December 2016. Naples, Italy: IEEE; 2016

[20] David V, Ragu H, Duraiswamy RK, Sasikumar P. Iot based automated indoor agriculture system using node-red and ibm bluemix. In: Proceedings of the 6th International Conference on Inventive Computation Technologies (ICICT'21); 20–22 January 2021. Coimbatore, India: IEEE; 2021

[15] Aldowah H, Ul Rehman S, Ghazal S, Naufal UI. Internet of things in higher education: A study on future learning. Journal of Physics: Conference Series. 2017;892:012017. DOI: 10.1088/1742-6596/892/1/012017

[21] Thuluva AS, Anicic D, Rudolph S, Adikari M. Semantic node-red for rapid development of interoperable industrial iot applications. Semantic Web. 2020;11(6):949-975. DOI: 10.3233/sw-200405

[16] Akiyama K, Ishihara M, Ohe N, Inoue M. An education curriculum of iot prototype construction system. In: Proceedings of the 6th Global Conference on Consumer Electronics (GCCE'17); 24–27 October 2017. Nagoya, Japan: IEEE; 2017

[22] Ferencz K, Domokos J. Rapid prototyping of iot applications for the industry. In: Proceedings of the IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR'20); 21–23 May 2020. Cluj-Napoca, Romania: IEEE; 2020

[17] Marquez J, Villanueva J, Solarte Z, Garcia A. Iot in education: Integration of objects with virtual academic communities. New Advances in Information Systems and Technologies. 2016;444:201-212. DOI: 10.1007/978-3-319-31232-319

[23] Fox J, Donnellan A, Doumen L. The deployment of an iot network infrastructure, as a localised regional service. In: Proceedings of the 5th IEEE World Forum on Internet of Things (WF-IoT'19) 15–18 April 2019. Limerick, Ireland: IEEE; 2019. pp. 319-324

[18] Torres D, Dias JP, Restivo A, Ferreira HS. Real-time feedback in node-red for iot development: An empirical study. In: Proceedings of the 24th International Symposium on Distributed Simulation and Real Time Applications (DS-RT'20); 14–16 September 2020. Prague, Czech Republic: IEEE/ACM; 2020. pp. 1-8

[24] Mohammad MA, Musard B, Daniel H, Lars EO, Andrei S. Securing node-RED applications. In: Protocols, Strands, and Logic. Sequence Number 1: Springer International Publishing; 2021. DOI: 10.1007/978-3-030-91631-2_1

[19] Kodali RK, Anjum A. Iot based home automation using node-red. In: Proceedings of the 2nd International

[25] Clerissi D, Leotta M, Reggio G, Ricca F. Towards an approach for developing and testing node-red iot systems. In: Proceedings of the 1st ACM Sigsoft International Workshop on Ensemble-Based Software Engineering;

November 2018. New York, NY, USA:
ACM; 2018. pp. 1-8

[26] Clerissi D, Leotta M, Ricca F. A set of empirically validated development guidelines for improving node-red flows comprehension. In: Proceedings of the 15th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE'20). SciTePress; 25-26 April 2022. Online Streaming Conference: 2020. pp. 108-119

[27] Jost B, Ketterl M, Budde R, Leimbach T. Graphical programming environments for educational robots: Open roberta - yet another one? In: Proceedings of the IEEE International Symposium on Multimedia; 10-12 Dec. 2014. Taichung, Taiwan: IEEE; 2014. pp. 381-386

[28] Kuhail MA, Farooq S, Hammad R, Bahja M. Characterizing visual programming approaches for end-user developers: A systematic review. IEEE Access. 2021;9:14181-14202. DOI: 10.1109/ACCESS.2021.3051043

[29] Repenning A. Moving beyond syntax: Lessons from 20 years of blocks programing in agentsheets. Journal of Visual Languages and Sentient Systems. 2017;3:68-91. DOI: 10.18293/vlss2017-010

[30] Johnston WM, Hanna JRP, Millar RJ. Advances in dataflow programming languages. ACM Computing Surveys. 2004;36(1):1-34. DOI: 10.1145/1013208.1013209

[31] Rekers J, Schürr A. Defining and parsing visual languages with layered graph grammars. Journal of Visual Languages and Computing. 1997;8(1): 27-55. DOI: 10.1006/jvlc.1996.0027