Chapter

Teaching IIoT through Hands-on Activities

Gustavo Sanchez and Devika Kataria

Abstract

This chapter describes a hands-on educational approach to teach Industrial Internet of Things (IIoT), including activities like problem analysis, programming, testing and debugging. Students are given autonomy to propose and evaluate different solutions, using adequate tools and following best practices. In parallel, key competencies like team management, project planning, costing and time scheduling, are imbibed in students to prepare them to become deployable automation engineers. To illustrate the proposed approach, we elaborate on the experience gained from teaching an elective course to undergraduate engineering students, in terms of learning outcomes, methodology, assessment and feedback. This course was centered on the Node Red platform (based on Node.js), using hardware devices like Arduino Uno, Nano and Raspberry Pi. Sensors commonly used and protocols like Modbus RTU/TCP, OPC UA, MQTT are discussed in the framework of common industrial applications.

Keywords: IIoT, experiential learning, Node Red, communication protocols, development boards

1. Introduction

The skill gap for careers in a changing industrial sector has been identified by numerous authors [1], which has prompted educators to quickly adapt their courses, in order to prepare future engineers to excel in this new environment.

Typically, the following basic skills are in general required for engineers to succeed:

- ability to design, operate and troubleshoot processes and equipment, following best recommended practices, to maximize efficiency and productivity.
- teamwork, discipline and time management.

More formally, a document defining the skills and competencies needed in the automation field was proposed by The Automation Federation and International Society of Automation (ISA) [2]. It is made up of following tiers: personal effectiveness, academic, workplace, industry-wide technical, automation technical, occupation-specific knowledge, occupation-specific technical, occupation-specific requirements, and management (see **Figure 1**).

In this model, it is possible to observe that competencies related to Communication, Integration, Software and Cybersecurity are placed in tier 5.

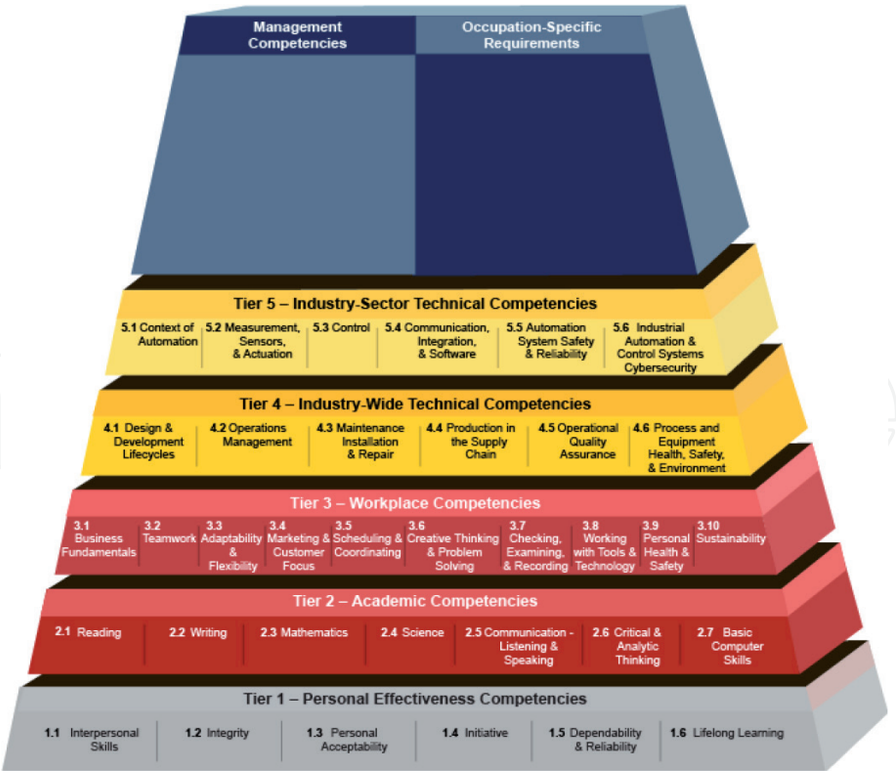

Figure 1.
Automation competency model [2].

Therefore, this is the natural place for the training program that will be described in next sections.

The Internet of Things (IoT) can be defined as a global dynamic network where physical and virtual objects interact to enable a set of services. In this context, the Industrial Internet of Things (IIoT) is the extension of this network to industrial sectors like logistics, transportation, manufacturing, utilities, oil and gas, etc. This extension enables to gather real-time data, necessary to make better decisions across all business functions: procurement, production, shipping, maintenance, etc.

To prepare for this chapter, several reports of teaching experiences related to IIoT have been consulted. In [1], the author describes his personal experience, working with educators and practitioners. It is stated that the path toward creating the Industry 5.0 workforce should begin in elementary school, and a specific curriculum is proposed for each level.

In [3] an on-line learning infrastructure is proposed, that allows to engage in a range of programming of real-world sensing applications, using a board based on the Arduino microcontroller, with several onboard I/O devices, including a slider, a pushbutton switch, a bank of six LEDs, and analog inputs for additional sensors. In [4] a syllabus is proposed, which offers guidelines for the quality assurance and safeguarding of IoT solutions, suitable for advanced studies at postgraduate level.

This chapter describes a hands-on educational approach to teach IIoT. In Section 2, we discuss common educational challenges in this domain and how to overcome them. In Section 3, we elaborate on the experience gained from teaching an elective course to undergraduate engineering students, in terms of learning outcomes, methodology, assessment and feedback. Wherever possible, we provide the link to possible solutions of proposed problems, that we have developed in order to make it available for interested readers to test and adapt them for their own projects. Finally, we conclude this chapter.

2. Educational challenges

To the best of our knowledge, there is no official document specifically describing desired competencies in the field of IIoT. Therefore, we have extracted from [2], the following main technical IIoT-related desired competencies:

- Design, document, install, and support the integration of automation systems with other systems, including Enterprise Resource Planning (ERP) and Manufacturing Operations Management (MOM)
- Design and operate databases for automation systems. Perform data historian duties: curation, archiving, retrieval
- Determine and implement the appropriate tools and methods for cybersecurity

The required technical knowledge includes:

- Network configuration, diagnostics and management
- Industrial digital field protocols (including but not limited to): AS-I, Ethernet/IP, DeviceNet, Foundation fieldbus, HART, INTERBUS, Modbus, PROFIBUS
- Industrial communication protocols (including but not limited to) XML, JSON, ASN.1, BACnet, ControlNet, Ethernet-TCP/IP, LonWorks, OPC UA, PROFINET
- Data contextualization (online/offline), modeling (UML, Entity Relation), storage and retrieval

Therefore, it is possible to observe that there is a broad range of topics to be addressed, which is the first pedagogical challenge that instructors will encounter when trying to design an IIoT course. Here, we propose to select only a basic sub-set of skills and content, which is equivalent to focus on the expected quality and depth of learning, rather than on the number of tools or protocols included in the syllabus (see **Table 1**, Section 3).

Unit	Topic	Content
1	IIoT Fundamentals 3 weeks	Industrial communication: principles, protocols and technologies. IIoT definition, architectures and use cases. Convergence of IT and OT. Design methodology.
2	Interfacing sensors and actuators 3 weeks	Proximity sensors, temperature sensors, vibration sensor, color sensors. Controlling DC/AC motors.
3	Programming with Node Red 3 weeks	Injecting nodes, debugging, managing palettes, designing dashboard.
4	Cloud services 3 weeks	Basic concepts. Device management. Applications: predictive maintenance, quality monitoring, personalized dashboards.

Table 1.
Course units: Theoretical content.

The second challenge is the complexity of real world IIoT applications, which may impede their study, keeping in mind time and resource constraints [5]. Here, we propose to break down problems into simpler sub-problems, which can be solved within the allocated time, using available tools.

For example, the typical integration problem of a given control system to a remote dashboard can be divided into the following 5 sub-problems:

1. Design and implementation of a local dashboard, considering only devices able to communicate through Modbus RTU
2. Assuming the previous system is working, integration of devices able to communicate through Modbus TCP/IP
3. Integration of devices able to communicate through OPC UA
4. Setting up communication to remote broker through MQTT protocol
5. Implementation of more complex applications like computer vision, anomaly detection, etc.

These problems will be further explained in next section, in the framework of a case study centered in our experience teaching an elective course to undergraduate engineering students.

3. Case study

The course Industrial Internet of Things (IIoT) aims at creating the fundamentals skills required to design, implement, and maintain industrial IoT systems. It is taught as elective course to undergraduate engineering students in their prefinal year. A previous exposure to embedded system programming, instrumentation and control systems is recommended. On successful completion of this course students are able to:

- Explain the key components that make up an Industrial IoT system.
- Discuss protocols and standards employed at each layer of the IIoT stack.
- Design, deploy and test a basic Industrial IoT system, including data analysis functionalities.
- Apply best practices to meet desired requirements for IIoT applications.
- Analyze the environmental effects and incorporate robustness in design of IIoT system.
- Choose technology for constrained nodes and network while maintaining real time data collection.
- Explain the importance of cybersecurity for IIoT networks.

The course delivery is planned in online mode and three sessions per week are conducted for 18 weeks. These sessions include concept discussions, hands-on

activities, projects, and assessments. The course description document containing the syllabus (see **Table 1**), learning outcomes, assessment rubric and references for learning materials is shared with students at the beginning of course. All the software is open source and sessions to install Node-Red, VNC viewer, Raspbian Busters operating systems, etc. are held at the beginning of the course. It is recommended for students to have a Desktop/Laptop able to run Windows 10.

3.1 Learning and assessment activities

The hands-on, problem-based learning or experiential learning approach means students are given a set of problems, and while trying to solve them they learn theoretical concepts. **Figure 2** summarizes the concept map for the learnings in this course, showing the topics discussed and demonstrated during hands-on sessions.

Next, we describe the set of problems that were proposed to students. Note that, as discussed in previous section, they correspond to the breaking down of a more complex control system integration problem.

3.1.1 Problem 1. Design and implementation of a local dashboard, considering only devices able to communicate through Modbus RTU

Consider the input/output variables shown in **Table 2**. We assume they correspond to a set of sensors and actuators connected to a device, e.g., PLC, Raspberry Pi, Arduino board, etc., able to act as a Modbus RTU slave, at address 01. It is required to design and implement a dashboard to supervise and control this process, which will also run at edge level, in a second device able to run Node Red [6], e.g., Desktop PC, Laptop or Raspberry Pi. This second device will act as Modbus RTU master.

A low-cost solution for this problem is to set up an Arduino/Genuino Uno as MODBUS slave, which is a microcontroller board based on the ATmega328P microprocessor. It has 14 digital input/output pins, of which 6 can be used as PWM outputs, 6 analog inputs, and runs with a 16 MHz quartz crystal (see **Figure 3**). Note that, in this problem, no real sensors/actuators will be connected to this board, because we are only interested in testing communication features. This means that the board will always be sending "dummy" data to the master. This also means that students do not need to have any sensor or actuator at home during on-line classes, to work on this problem.

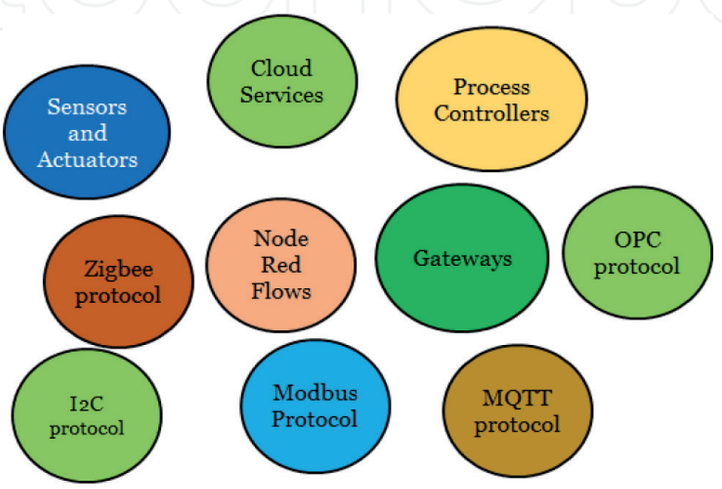

Figure 2.
Concept map for the domain knowledge imparted in IIoT course.

Data address	Type	Internal tag	ISA S5.1Tag	Range
0	Analog input register – read only	AI0	TC01.PV	0–100°C
1	Analog input register – read only	AI1	FC02.PV	0–150 lt/min
2	Analog input register – read only	AI2	PC03.PV	0–200 psi
3	Analog input register – read only	AI3	SC04.PV	0–1000 RPM
4	Analog input register – read only	AI4	VC05.PV	0–10 mm/s
5	Analog output register – read/write	AW0	TC01.SP	0–100°C
6	Analog output register – read/write	AW1	FC02.SP	0–150 lt/min
7	Analog output register – read/write	AW2	PC03.SP	0–200 psi
8	Analog output register – read/write	AW3	SC04.SP	0–1000 RPM
9	Analog output register – read/write	AW4	VC05.SP	0–10 mm/s
10	Discrete input coil – read only	DI0	YC06.PV	0/1
11	Discrete input coil – read only	DI1	YC07.PV	0/1
12	Discrete input coil – read only	DI2	YC08.PV	0/1
13	Discrete input coil – read only	DI3	YC09.PV	0/1
14	Discrete input coil – read only	DI4	YC10.PV	0/1
15	Discrete output coil – read/write	DW0	YC06.SP	0/1
16	Discrete output coil – read/write	DW1	YC07.SP	0/1
17	Discrete output coil – read/write	DW2	YC08.SP	0/1
18	Discrete output coil – read/write	DW3	YC09.SP	0/1
19	Discrete output coil – read/write	DW4	YC10.SP	0/1

Table 2.
Input/output variables for Problem 1.

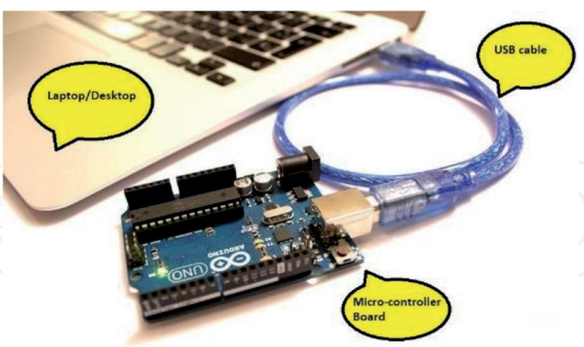

Figure 3.
Example of required set-up for Problem 1.

We propose to use the library SimpleModbusSlave [7] which allows the Arduino board to communicate using Modbus RTU protocol. Note however that it does not fully comply with Modbus specifications, as only functions 3, 6 and 16 are implemented. Similarly, the check for inter character time-out and frame time-out are combined by checking a maximum time allowable when reading from the message stream. This library implements an unsigned int. return value on a call to modbus_update(), which is the total error count since the slave started. Once this function is called, the input/output register defined during setup with function

modbus_configure() will be updated. Note that we have also successfully tested this library with Arduino Nano.

In addition to communication setup, it is possible to add other features in this Arduino program, to make it dynamic when visualizing the dashboard, as follows:

- For variable TC01.PV it is required to program a counter that increments from 0 to 100 and then is reset to 0.
- Between variables VC05.PV and VC05.SP it is required to implement a first order transfer function, to simulate a real process.
- Status of discrete output variable YC06.SP must be updated to YC06. PV and Arduino built-in LED.

An example of code complying with these specifications is available in this link: https://bit.ly/3eqHmxB. It is possible to test this code, previously to developing user dashboard, with QModbus, which implements a master application through a graphical user interface, allowing communication with slaves over serial line interface [8]. Students are able to analyze Modbus frames, from master and slave.

The previous explanation corresponds to the edge layer. Now considering the gateway layer, Node Red is able to run in different devices. We have used a laptop for convenience. The following palettes must be installed:

- node-red-contrib-modbus, version 5.13.3
- node-red-dashboard, version 2.28.1

An example of Node-Red code is available in this link https://bit.ly/2RnA9q0, as shown in **Figure 4**. The corresponding dashboard is shown in **Figure 5**.

3.1.2 Problem 2. Integration of devices able to communicate through Modbus TCP/IP

For the same process described in problem 1, include the input/output variables shown in **Table 3**, connected through Modbus TCP/IP at address 02.

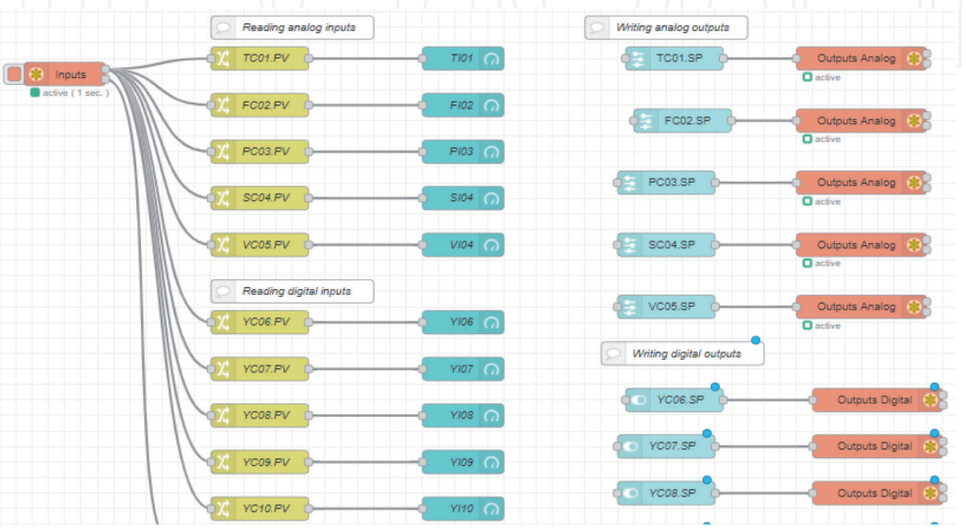

Figure 4.
Example of Node-Red code for Problem 1.

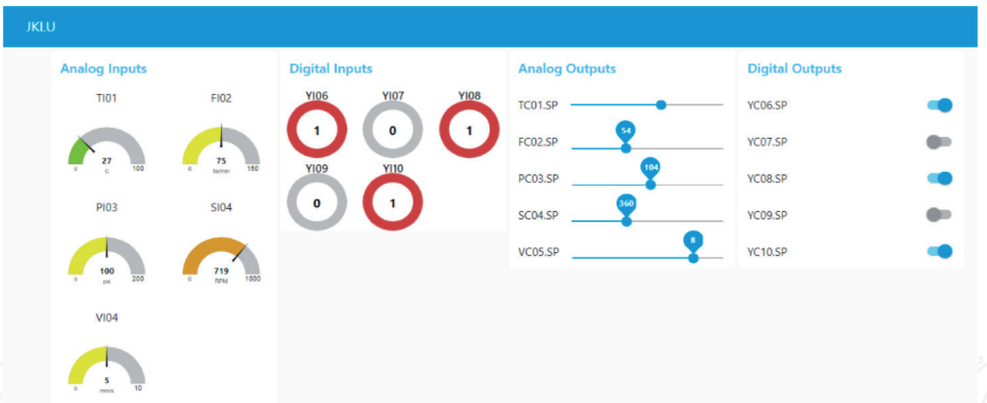

Figure 5.
Example of Node-Red dashboard for Problem 1.

Data address	Type	ISA S5.1Tag	Range
0	Analog input register – read only	TC11.PV	0–100°C
1	Analog output register – read/write	FC12.SP	0–150 lt/min
2	Discrete input coil – read only	YC13.PV	0/1
3	Discrete output coil – read/write	YC14.SP	0/1

Table 3.
Input/output variables for Problem 2.

An example of Node-Red code for Problem 2 is available in this link https://bit.ly/3tpnv7x. It is possible to test this code with ModbusSlave, which enables simulation of slave devices [9]. The limitation is that this software runs only in Windows operating system. The dashboard integrating measurements from both sources Modbus RTU and TCP is shown in **Figure 6**.

3.1.3 Problem 3. Integration of devices able to communicate through OPC UA

Design a dashboard to display the OPC UA tags shown in following **Table 4**, which will be randomly generated by Integration Objects' Server Simulator, which is a free to use tool [10].

An example of Node-Red code to solve this problem is available in this link https://bit.ly/3er1QqZ. The following palette must be previously installed: node-red-contrib-opcua. The dashboard displaying required OPC UA tags is shown in **Figure 7**.

3.1.4 Problem 4. Setting up communication to remote broker through MQTT protocol

It is required to design and implement a remote dashboard, which will run in Cloud, using Message Queueing Telemetry Transport (MQTT) protocol.

It is recommended to have at least some hardware setup (sensors, microcontrollers, power supply, etc.) available with the instructor. In case students do not have any hardware at home, they write the code and send it to the instructor for testing purpose. A schematic diagram showing the architecture for interfacing sensors and uploading the data to Cloud is shown in **Figure 8**. The data from the analog pin is sent to serial port of Arduino. The data from the controller board serial port is sent to the Internet gateway. Node-Red flow is run on the gateway and enable the data to be sent to Cloud.

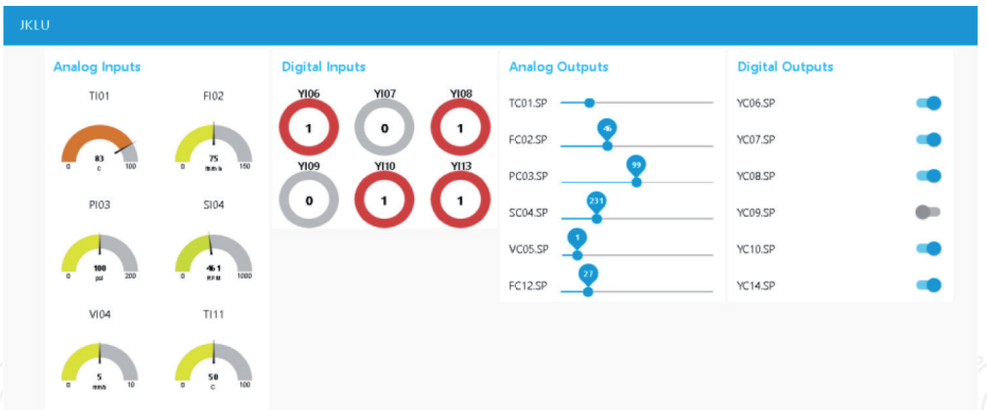

Figure 6.
Example of Node-Red dashboard for Problem 2.

Tag	Address	Type
Tag11	ns = 2; s = Tag11	Int16
Tag12	ns = 2; s = Tag12	Int32
Tag13	ns = 2; s = Tag13	Int64
Tag14	ns = 2; s = Tag14	UInt16
Tag15	ns = 2; s = Tag15	UInt32
Tag16	ns = 2; s = Tag16	UInt64
Tag17	ns = 2; s = Tag17	Double
Tag18	ns = 2; s = Tag18	String
Tag19	ns = 2; s = Tag19	Byte
Tag20	ns = 2; s = Tag20	Boolean

Table 4.
OPC UA tags for Problem 3.

A snapshot of flow where a SW-420 vibration sensor has been interfaced to an Arduino Nano, which sends the values to dashboard and display them in form of chart, is shown in **Figure 9**.

The sensors may be interfaced to the microcontrollers using various protocols. An example of this is the Zigbee protocol where sensors communicate with an end point device, which in turn sends the sensor data through routers to the Zigbee coordinator. The advantage of this type of connection is that multiple sensors can be connected to endpoint devices, and many such endpoint devices may be connected in star topologies to the controller through routers.

The Zigbee protocol is known to be secure and low power consuming as the endpoints which are inactive may sleep for the inactive duration. A typical application could be connecting crop monitoring sensors to end points and sending the data from endpoints to coordinator, which in turn may send the data to an MQTT server using the node-red-contrib-zigbee pallete (see **Figure 10**).

Various other wireless protocols like the Sigfox and LoRa WAN may be used for interfacing the sensors to the master coordinator/controller. The advantage of some of these emerging protocols are low power consumption and high data rates [11, 12]. The IoT Gateway is used to convert the data format received from any of these protocols to internet protocols like the HTTP, MQTT, XMPP or any other light weight protocol. The Gateway also implements security for the Endpoint and Coordinator devices and may do some edge computing or data analytics before sending the data to the Cloud storage.

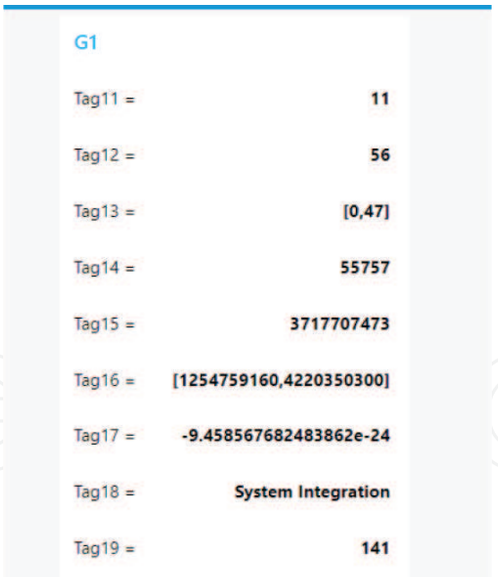

Figure 7.
Dashboard displaying required OPC UA tags.

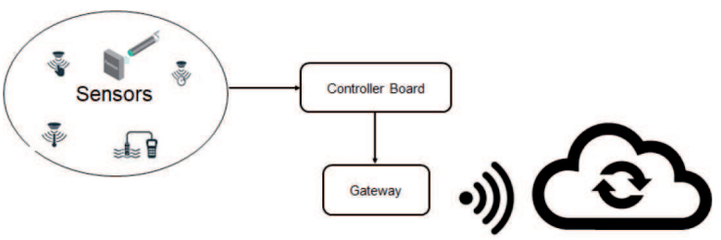

Figure 8.
Architecture for interfacing and uploading data to Cloud.

3.1.5 Problem 5. Implementation of more complex applications like computer vision, anomaly detection, etc

Students were able to interface the Raspberry Pi camera to upload images to Cloud, trying to optimize bandwidth usage. MQTT protocol and associated libraries for image transmission using Python programming are used in some of these projects for uploading data to Cloud. The Node-Red palletes required for implementing the flows were identified and installed.

Machine Learning services available on Cloud like IBM Watson were used by students, where algorithms for image recognition and classification, text recognition and other resources of AI/ML deployed. Knowledge of Raspberry Pi boards and Python programming as well as running Node-Red using Docker container was introduced. Node-Red flow was used to upload camera images and to classify objects using pretrained models from Tensorflow.js (Common Objects in Context dataset), available from pallete node-red-contribtensorflow [13, 14]. A simple flow for interfacing the camera and sending images to Cloud is shown in **Figure 11**. The flow consists of an inject node followed by execute node which contains the command to run the python program and a message payload node to debug the messages.

3.2 Project

Once students have completed all the previous hands-on activities, they were requested to work on a project, so they can apply the methods they have learned. First, they must submit a project charter, describing the project goals,

DOI: http://dx.doi.org/10.5772/intechopen.100217

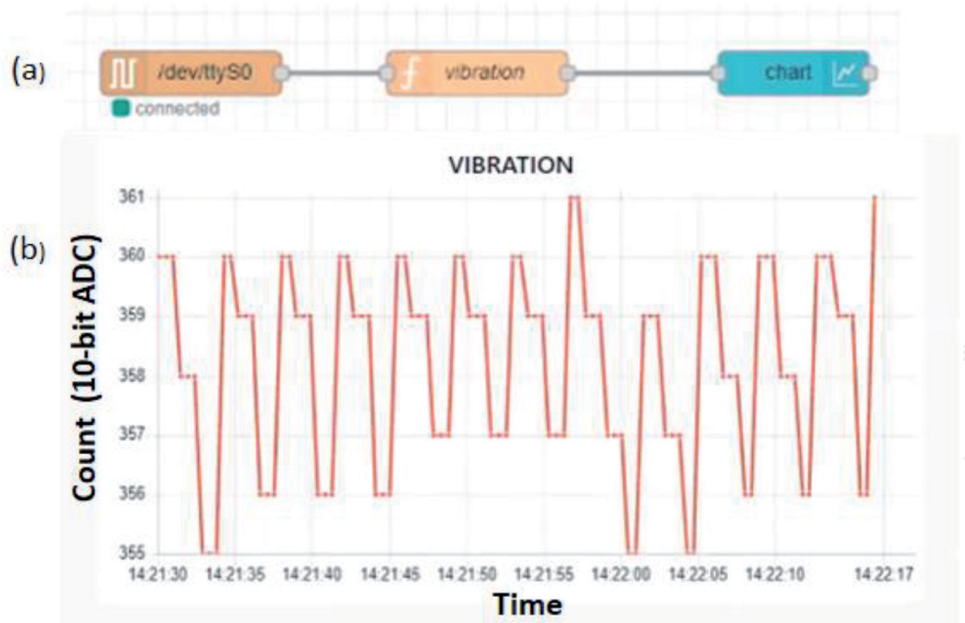

Figure 9.
Displaying SW-420 vibration sensor data in form of chart.

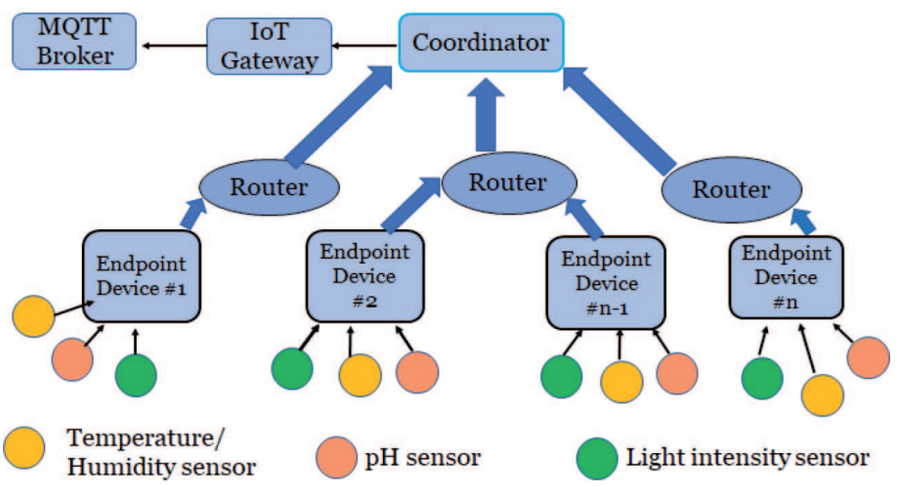

Figure 10.
Example of Zigbee architecture for agriculture application.

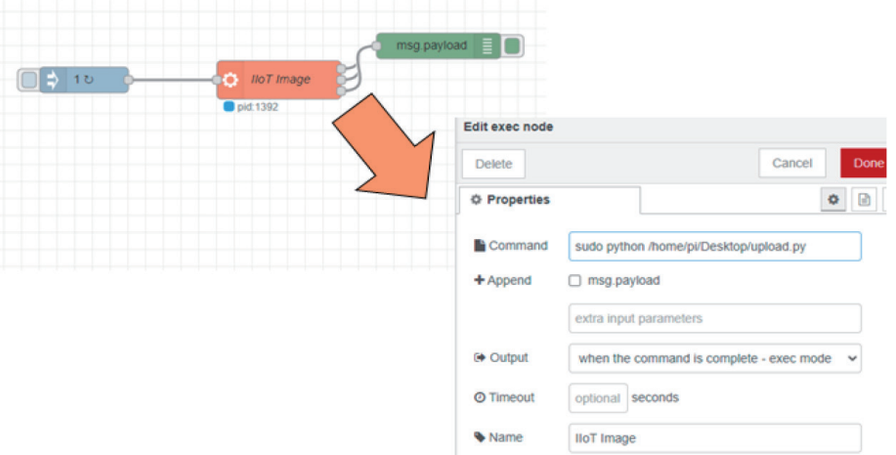

Figure 11.
Simple flow for interfacing the camera and sending images to Cloud.

responsibilities of team members, resources/bill of materials, references to literature and timing charts. Students should be made conscious of the professional ethics while working on this project. We consider the following facts as academic dishonesty offenses:

- Cheating: using unauthorized information. Receiving or giving unauthorized assistance.
- Fabrication: invention or falsification of any information.
- Plagiarism: deliberately representing the ideas, results, reports, drawings, notes, computational code, or any other product prepared by another person as one's own.

Some of the project ideas identified by the students for this course are:

a. monitoring safety in personnel in industry by monitoring camera images for helmet usage,

b. facemask detection for crowd at public places using live video transmission.

c. home automation using Zigbee and MQTT protocol and Node-Red flow(s),

d. language translation: speech to text and vice versa for real time audio signal,

e. surveillance using infrared camera and live video transmission to remote control station.

3.3 Assessment

Quizzes are recommended to be conducted periodically for assessing the learning outcomes. The aim of these evaluations should be to determine the understanding of concepts for implementation. As a sample a quiz may comprise following questions (with marks break) as follows:

Q1. Write Node-Red and Arduino code to solve the following problem:

a. Communication between Arduino and Node-Red can be implemented using any protocol (4 marks)

b. Two values A in [0, 100] and B in [0, 100] will be generated through Node-Red dashboard and written to Arduino board (3 marks)

c. Average M of these A and B will be calculated by Arduino board (3 marks)

d. Average M needs to be displayed back in Node-Red dashboard (3 marks)

e. If M is greater than 80 during 5 seconds, an alarm H will be displayed in the Node Red dashboard and Arduino built-in LED, until a RESET button (also in dashboard) is pressed (3 marks)

f. All values A, B, M and H will be published to Mosquitto MQTT broker, to following topics: IIoTQuiz1/Name/A, IIoTQuiz1/Name/B, IIoTQuiz1/Name/M, IIoTQuiz1/Name/H, where Name is student's name. (4 marks)

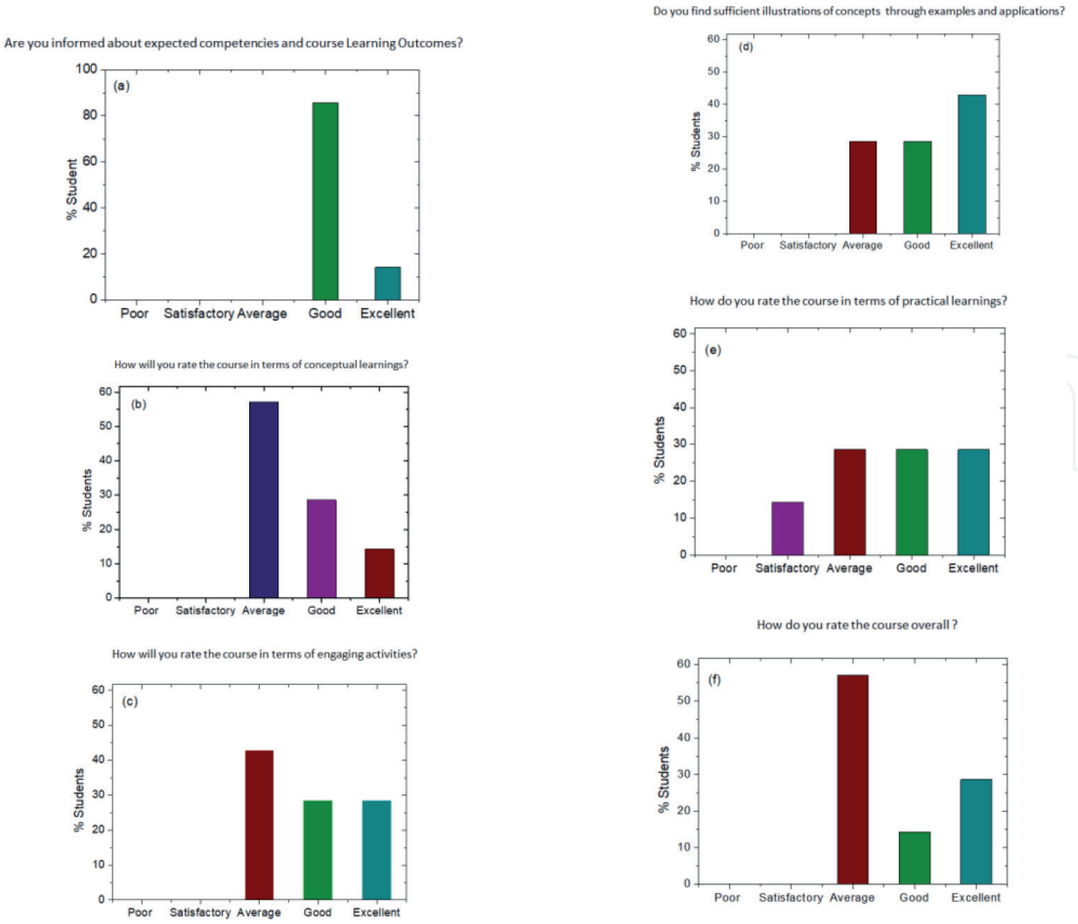

Figure 12.
(a–f) Feedback questionnaire and responses by students for the first time launch of the course.

3.3 Learners feedback

Feedback is recommended fortnightly to understand the learning process. A sample questionnaire and the responses are shown in **Figure 12** here for the course conducted by the authors for the first time for undergraduate students. The bar charts are self-explanatory and the feedback will be considered for further improvements.

4. Conclusion

In this chapter, a hands-on educational approach to teach Industrial Internet of Things (IIoT) was proposed. Because the set of required skills is large, we propose to focus on a basic sub-set of skills and content, trying to achieve the best possible quality and depth of learning. To overcome the complexity of real world IIoT projects, we propose to identify simpler sub-problems, which can be solved within the allocated time, using available tools. To illustrate our approach, specific examples, in terms of learning outcomes, methodology, assessment and feedback were presented. Wherever possible, link to solutions was provided for interested readers to test and adapt them for their own projects. The feedback received from students and their final performance is encouraging, as they seem to appreciate the proposed approach. We believe the same can be extended to teach similar courses like Digital Computer Networks, SCADA systems, Programmable Logic Controllers, etc. Currently, authors are planning to scale up this course, as an international MOOC, to reach a broader audience.

Acknowledgements

We are grateful to the Atal Incubation Center at JK Lakshmipat University for funding the projects for this course. We also acknowledge the support and motivation from the management in helping us to launch this course for students pursuing Bachelor in Technology degree in Electrical and Electronics Engineering.

Author details

Gustavo Sanchez and Devika Kataria*
Department of Electrical and Electronics Engineering, Institute of Engineering and Technology, JK Lakshmipat University, Jaipur, India

*Address all correspondence to: devikakataria@jklu.edu.in

References

[1] M. D. Kirchner. Teaching the Industrial Internet of Things. Preparing Students and Learners for Industry 4.0. September 2017. Available from: https://labmidwest.com/wp-content/uploads/2017/09/Teaching-IIoT-Preparing-Students-and-Learners-for-Industry-4.0-2.pdf [Accessed: 04 April 2021]

[2] The Automation Federation. Automation Competency Model. Available from: https://www.careeronestop.org/competencymodel/competency-models/automation.aspx [Accessed: 06 April 2021]

[3] G. Kortuem, A. K. Bandara, N. Smith, M. Richards, and M. Petre. Educating the internet-of-things generation. Computer 2013;46(2):53-61. DOI:10.1109/MC.2012.390

[4] A. Boukhris, et al. Quality engineering for the internet of things. Foundation level syllabus. Software Quality and Advanced Training Working Group (ASQF); 2018.

[5] F. Salewski and R. Schmidt. Teaching industrial automation: An approach for a practical lab course. In: Proceedings of the WESE'15: Workshop on Embedded and Cyber-Physical Systems Education, Amsterdam, Netherlands, October 2015. pp. 1-7. DOI:10.1145/2832920.2832921

[6] https://nodered.org/

[7] https://github.com/jecrespo/simple-modbus/blob/master/Modbus%20RTU%20libraries%20for%20Arduino/SimpleModbusSlaveV10/SimpleModbusSlave.h

[8] http://qmodbus.sourceforge.net/

[9] https://github.com/ClassicDIY/ModbusTool

[10] https://integrationobjects.com/sioth-opc/sioth-opc-unified-architecture/opc-ua-server-simulator/

[11] https://lora-alliance.org/wp-content/uploads/2020/11/what-is-lorawan.pdf

[12] https://www.sigfox.com/en/sigfox-story

[13] https://cocodataset.org/#home

[14] https://developer.ibm.com/technologies/artificial-intelligence/tutorials/building-a-machine-learning-node-for-node-red-using-tensorflowjs/