Bonsai in the Fog: an Active Learning Lab with Fog Computing

Antonio Brogi, Stefano Forti, Ahmad Ibrahim, Luca Rinaldi
Department of Computer Science
University of Pisa, Italy
name.surname@di.unipi.it

***Abstract*—As for every new technological trend, education of new scientists and engineers will be instrumental in shaping the implementation of the Fog architecture, including software development.**

In this paper, we describe a 2-hour hands-on lab activity that we designed and run during an M.Sc. course at the Department of Computer Science of the University of Pisa, Italy. The proposed activity is designed to practically (and incrementally) show different deployment models for IoT applications – viz., IoT+Edge, IoT+Cloud and Fog – over a meaningful yet simple application example. In addition to featuring a quick learning curve and limited cost, the proposed activity shows some possible extensions to design other lab sessions.

***Index Terms*—Fog computing, education, hands-on lab, active learning, micro:bit, software development**

I. Introduction

Recent research efforts are investigating how to better exploit capabilities along the continuum from the edge of the Internet to Cloud data centres, to support new IoT applications and their needs. Computational nodes closer to the edge will act both as *filters* – reducing the amount of data sent to the Cloud – and as *processing capabilities* – producing analytics – closer to where data is being sensed or used.

Current deployment models [1] for IoT applications consist either of

(1) *IoT+Edge* deployments (i.e., Edge computing [2], [3]) where data is processed locally at the edge of the Internet to determine reactions to sensed events (Figure 1(a)), or of
(2) *IoT+Cloud* deployments [4] where Things send data to Cloud data centres for further processing/analytics purposes, awaiting for a response, and only minor computation happens *in situ* (Figure 1(b)).

Despite supporting low-latency responses to sensed events, IoT+Edge deployments do not permit to easily share collected data across different systems, nor to perform real-time analytics on substantial amounts of data, due to the limited available resources at the edge (e.g., constrained or battery powered devices). Conversely, IoT+Cloud requires a stable Internet connection, and induces higher latencies and costs to transfer data to the Cloud. Also, IoT+Cloud deployments may cause performance degradation due to network congestion at the edge and core of the Internet.

Overall, both the depicted models are insufficient to support the IoT momentum, either overloading edge capabilities or requiring stable Internet connectivity, without supporting the IoT applications demand for low latencies and prompt decision-making [5].

In this context, *Fog computing* [6] aims at better supporting time-sensitive and bandwidth hungry IoT applications by selectively pushing computation closer to where data is produced and exploiting a geographically distributed multitude of heterogeneous devices (e.g., personal devices, gateways, micro-data centres, embedded servers) spanning the continuum from the Cloud to the IoT (Figure 1(c)).

The characteristics of Fog computing include [7]–[11]:

- *contextual location awareness* and *low latency*, enabling processing to happen as close as possible to the source of data,
- *bandwidth savings*, avoiding time-consuming data transfer to the Cloud when unnecessary,
- support for *mobility*, both in terms of IoT devices (e.g., smart-phones, vehicles) and associated Fog nodes,
- *geographical distribution*, to perform decentralised decision making exploiting widely distributed nodes,
- *heterogeneity* and *interoperability* of different IoT, Fog and Cloud nodes and communication technologies (e.g., wireless, serial, radio).

Introducing novel technological trends – such as Fog computing – in Computer Science (CS) and Engineering curricula is always a challenging task to accomplish. The OpenFog Consortium (OFC) [12] is calling for the design of curricula and courses including Fog computing in higher education programs, with a particular focus on software development in Fog scenarios [1].

For the first time this year, we have introduced Fog computing to M.Sc. students in CS attending our *Advanced Software Engineering* course[1]. The course includes hands-on lab sessions to promote students' positive learning in a supportive teaching environment, employing constructivist teaching methods [13] and implementing active learning [14].

In this spirit, we designed a two-hours practical session on Fog computing which had to:

[1]The *Advanced Software Engineering* course is a 9 ECTS course offered to M.Sc. students at the Department of Computer Science of the University of Pisa. This year, we overall devoted four hours – a two hours lecture and a two hours practical lab – to give a general introduction to Fog computing.

978-1-5386-5896-3/18/$31.00 ©2018 IEEE

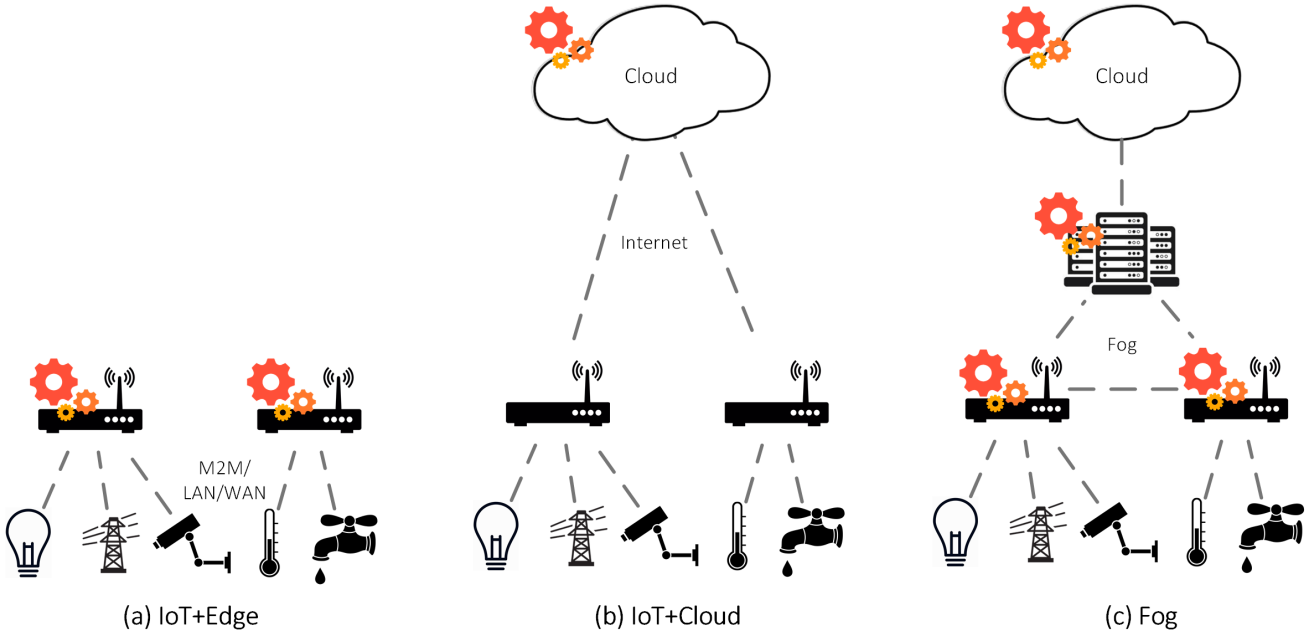

Fig. 1. Alternative deployment models of IoT applications.

O$_1$. constitute a first *hands-on programming* lab on Fog computing, practically showing the difference between IoT+Edge, IoT+Cloud and Fog deployment models, by exploiting *active learning* methodologies,

O$_2$. have a *quick learning curve*, only requiring students familiarity with high-level programming languages[2] (so that the activity could fit in a two-hours session),

O$_3$. have *limited cost* (i.e., hundreds of euro[3]) with respect to enterprise solutions (i.e., thousands of euro), possibly being *cross-platform* with respect to different (students' laptops) operating systems.

In this paper, we describe the active learning activity that we designed and proposed to our students. As we will show, the activity is specially designed to fulfil the set objectives (viz., **O**$_1$, **O**$_2$ and **O**$_3$), and it can constitute the basis for building (individual or group) projects and/or pieces of assessment.

The rest of this paper is organised as follows. After giving a bird's-eye view of the practical session and its overall organisation (Section II), we describe how to set up the learning environment for the activity (Section III). We then describe the whole session in detail (Section IV), and review some related work (Section V). We conclude by assessing how the active learning activity achieves the set objectives and by mentioning some possible extensions to obtain other practical sessions (Section VI).

[2]For instance, students attending our course hold a B.Sc. in Computer Science, or an equivalent academic degree.

[3]We use the spelling *euro* for both singular and plural as per the official EU recommendations [15].

II. Lab Overview

As mentioned in Section I, our first goal was to design a highly practical hands-on session to help students in getting in touch with Fog computing (**O**$_1$). Hence, to experientially involve students in the activity, we, first of all, identified a (toy) motivating scenario, which led the design of the active learning session:

> *Consider a bonsai greenhouse company planning to adopt a software solution to monitor and visualise soil moisture of their cultivation.*

In addition to being quite a lifelike example, precision and smart agriculture is one of the vertical markets to which Fog computing is expected to be fruitfully applicable [16].

Still, in line with **O**$_1$, we wanted our students to play with different deployment models of IoT applications, i.e. IoT+Edge, IoT+Cloud and Fog. Therefore, we structured the activity along the following three incremental phases (each covering a different deployment model):

(*i*) setting up an *IoT testbed* and coding of a simple *IoT+Edge* application to monitor soil moisture of each bonsai in the greenhouse,

(*ii*) coding of a gateway module running on general purpose laptops and streaming collected data to the Cloud for visualisation purposes, as in *IoT+Cloud* scenarios, and

(*iii*) extending the gateway module so that some computation was performed at the laptop level, enabling lower latencies and bandwidth savings as in *Fog* scenarios.

Each of the three parts of the learning activity was triggered by few introductory slides and concluded by a checkpoint. After completion of each phase, class discussion was opened before introducing the next phase, which was achieved as

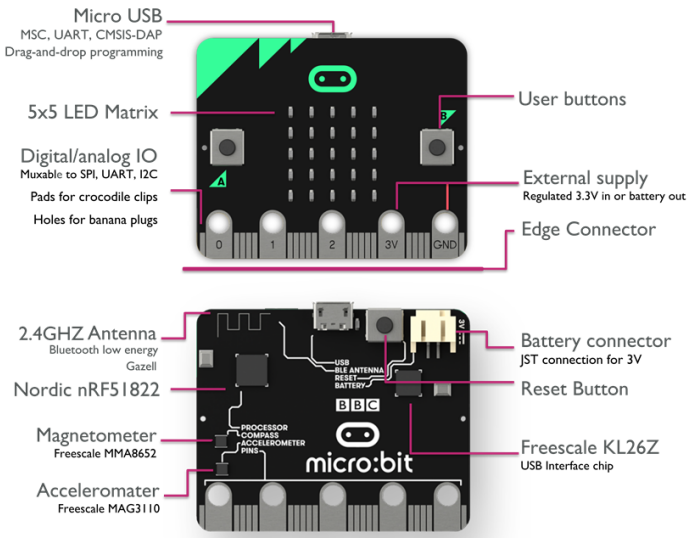

Fig. 2. Scheme of a micro:bit [18].

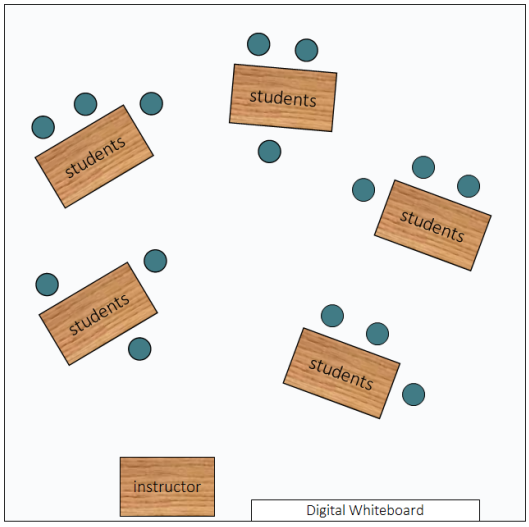

Fig. 3. Classroom setup.

an extension of the previous one. At the end of the session a summary of the overall activity was given. Doing so, we managed to guarantee that all students could successfully complete the lab session, whilst verifying the achievement of the learning objectives declared in $\mathbf{O}_1$.

To set up the experimental IoT testbed capable of monitoring moisture of the plants growing in the greenhouse, we employed micro:bits [17] (Figure 2), which are embedded systems designed in the UK for teaching CS. Each micro:bit features an ARM Cortex-M0 processor, an accelerometer, a magnetometer and a light sensor, Bluetooth, radio and USB connectivity, a display consisting of 25 LEDs, two programmable buttons, and it can be powered by either USB or via an external battery pack. Furthermore, micro:bits can input and output electrical signals through the 23-pin edge connector from/to external devices, such as thermometers or moisture detectors.

Programming of micro:bits can be done within an online editor[4], exploiting either a visual programming language or JavaScript (which was our choice). Therefore, by using micro:bits, we did not require to our students any previous technical experience in IoT programming or Fog computing. Familiarity with any high-level programming language (preferably JavaScript) was sufficient to actively participate in the session, in accordance with $\mathbf{O}_2$.

Finally, the low cost of micro:bits[5] and the possibility of exploiting a largely supported language such as JavaScript to program them as well as the entire greenhouse application made us achieve our last goal $\mathbf{O}_3$.

III. Setup

A. Classroom

The laboratory setup for the proposed practical session can follow a *Bring Your Own Design* (BYOB) approach. We highly recommend organising the classroom in such a way to facilitate active learning and teamwork among students. Internet connectivity is required to access the online editor and documentation.

For instance, we exploited a laboratory at our Department and we organised the class into five groups of three students, having each group working around the same table on a single laptop (Figure 3). Our lab is also equipped with a digital whiteboard, however, a projector is more than fine to complete the proposed activity. The activity is naturally suitable for any type of active or collaborative learning environment, e.g. SCALE-UP, TEAL or ACL classrooms [19].

B. Materials

In addition to a (BYOB) laptop for each group of students, each table must be equipped with:

- a micro:bit with its USB cable and two crocodile clips,
- two plastic cups and two paper clips, and
- dry soil (or a thirsty bonsai) and some water.

C. Software

The proposed learning activity is totally cross-platform and the code runs on Windows, MacOS and Linux. The only requirements that students' laptops must fulfil are having a Web browser and (the latest version of) NodeJS properly installed.

IV. Running the Lab

In this section, we describe in detail how to run the three phases of the proposed active learning activity, as sketched in Figure 4. For the sake of readability, Figure 4 shows – for each scenario – only two among the students' laptops and micro:bits used to program and test the greenhouse application. All code presented hereinafter is also publicly available and documented on GitHub[6].

[4] *Microsoft MakeCode*. Available at: https://makecode.microbit.org.

[5] In Italy, the current cost of a micro:bit is around €20.

[6] *BonsaiFog*. Available at: https://github.com/di-unipi-socc/bonsaifog.

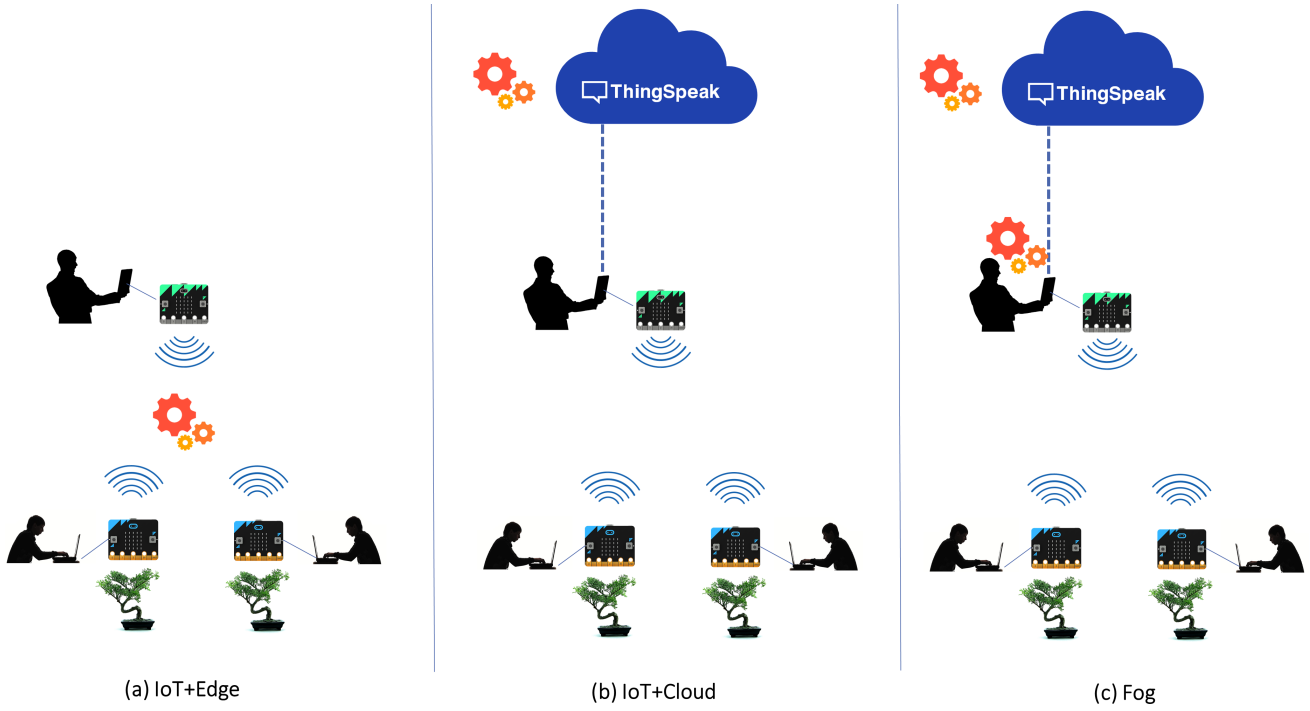

Fig. 4. The three scenarios of the learning activity.

A. IoT+Edge Scenario

The preliminary step for each table consists of building the IoT testbed to be used during the lab session and programming a first IoT+Edge application for moisture monitoring, as sketched in Figure 4(a). For this first part, we took inspiration from the official micro:bit tutorials [20]. At the end of this phase, the class will have coded a simple IoT+Edge application where the students' micro:bits act as IoT moisture sensors which transmit data to an Edge device acting as a sink (the instructor's micro:bit).

The instructor starts by asking students to connect one side of each crocodile clip to a paper clip, to be inserted in the soil (either of a bonsai or in a plastic cup), as shown in Figure 5. One of the crocodiles is then connected to pin `P0` of the micro:bit, whilst the other is connected to the `3V` pin. The described circuit can be naturally used to measure the electrical conductivity of the soil which gets higher the more the soil is watered (actually thanks to salts and nutrients that dilute in the water).

Listing 1 shows the JavaScript code to be burnt into the micro:bit of each bonsai. The instructor gives this first code snippet to students, but for the *if* of lines 7–9, asking them to complete the code so that the value of `reading` is shown when pressing the `A` button of their micro:bits. Students can easily achieve the goal of coding by themselves the *if* of lines 7–9, by autonomously browsing micro:bit online documentation and tutorials [21].

To stimulate active learning the instructor can also ask students to try reducing energy consumption by lowering the brightness of the micro:bit's LED screen and by probing soil moisture only every second, avoiding the continuous power waste due to having the crocodile clip connected to the `3V` pin[7].

Fig. 5. A bonsai connected to a micro:bit.

Finally, the instructor shows to the students how to include in the micro:bit firmware the serial writing (of tuples $\langle$`moisture`, `reading`$\rangle$) and radio transmission of the value of `reading` (normalised to the range $[0, 255]$). Solution code to

[7]In our class, students who finished before the others were also asked to extend the code so to display particular symbols on the micro:bit LEDs when the sensed moisture was too low (≤ 250) or too high (≤ 950).

```
let reading = 0

basic.forever(() => {
    reading = pins.analogReadPin(AnalogPin.P0)
    led.plotBarGraph(reading, 1023)

    if (input.buttonIsPressed(Button.A)) {
        basic.showNumber(reading)
    }
})
```

Listing 1. micro:bit firmware.

```
radio.setTransmitSerialNumber(true)
radio.setGroup(4)
led.setBrightness(64)
let reading = 0

basic.forever(() => {
    pins.analogWritePin(AnalogPin.P1, 1023)
    reading = pins.analogReadPin(AnalogPin.P0)
    radio.sendNumber(reading / 4);
    pins.analogWritePin(AnalogPin.P1, 0)
    led.plotBarGraph(reading, 1023)

    if (input.buttonIsPressed(Button.A)) {
        basic.showNumber(reading)
    }

    serial.writeValue("moisture", reading)
    basic.pause(1000)
})
```

Listing 2. Serial and radio micro:bit firmware.

these tasks is provided in Listing 2 and it requires connecting a crocodile clip to the `P0` and the other to the `P1` pin.

The instructor carries the same activity in parallel to students, performing checkpoints to discuss (all possible) solution code(s). Furthermore, an additional micro:bit at the instructor table runs a radio dashboard script like the one proposed in the official micro:bit documentation[8], suitably extended with serial writing of all received values (i.e., `serial.writeValue("moisture", packet.receivedNumber * 4)`). The dashboard associates each of the instructor's micro:bit LEDs to one of the students micro:bits, with LED brightness indicating moisture of the soil cup at each students' table. As a result, every time a group of students completes this part of the activity, one of the dashboard micro:bit LED turns on so that the instructor exactly knows when the IoT testbed is ready and that the class can continue.

B. IoT+Cloud Scenario

In this second phase, students have to include exploitation of a Cloud service – ThingSpeak [22] – within the experimental testbed, as per Figure 4(b). After completing this phase, all students will have coded an IoT+Cloud application where sensed moisture values are directly streamed to the Cloud by a trivial JavaScript gateway module, running on their laptops.

[8] *Radio Dashboard*. https://makecode.microbit.org/examples/radio-dashboard

The instructor starts by asking students to create an account on ThingSpeak, a Cloud-based platform featuring MATLAB analytics and data visualisation for IoT data. After registering to ThingSpeak, students can create a new channel and take note of the related *Write API key*.

The instructor provides the students with the primer code of Listing 3 to be completed with the *Write API key*, the micro:bit serial port – which can be retrieved on any platform by executing the commands in Listing 4 – and the code to push data coming from the micro:bit to ThingSpeak (by using the provided `pushData()` function). Figure 6 shows the result of running the completed code with NodeJS on laptops connected to a micro:bit running the firmware of Listing 2. Moisture data streamed by the micro:bit is visualised in the ThingSpeak dashboard. Groups can play with the application, trying to water the soil at their table and see changes in the plot made by ThingSpeak.

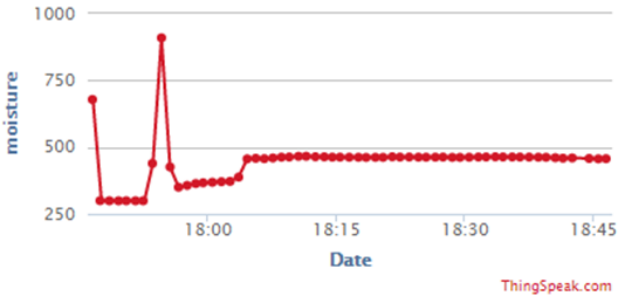

Fig. 6. ThingSpeak data visualisation.

Using the same gateway code to stream all sensed data from the dashboard micro:bit to ThingSpeak, the instructor can point out an interesting phenomenon. Since the free version of ThingSpeak only allows users to write to a given channel every 15 seconds, not all the sent readings will be plotted. Such observation can be used to introduce the need for data to get filtered and processed at an intermediate Fog layer.

C. Fog Scenario

The natural candidate to be extended with Fog functionalities is the JavaScript gateway module running within laptops. Figure 4(c) sketches the last phase of this learning activity, during which some Fog intelligence is added to the gateway code to aggregate and filter collected data before visualisation. Obviously, even if students can code and test extensions to the gateway module on their own laptops, the instructor's laptop will be eventually the Fog node of our scenario, aggregating and filtering data coming from all the micro:bits in the room.

To start this last part of the lab, the instructor asks students to extend the gateway code with a function `fogFun()` – input to `startFog(f)` – to send to the Cloud the average of the received readings, computed every 15 seconds. To simplify the task, one can give the students some hints about modifying the code executed on data reception so to (1) accumulate readings in a new `sum` variable, and (2) count the number `n` of readings

```
var SerialPort = require('serialport');
var request = require('request');

// TO DO: add Write Key
const CLOUD_URL = 'https://api.thingspeak.com/update?api_key=';
//TO DO: add micro:bit serial port
const MICROBIT_PORT = '';
const UPDATE_TIME = 1; // seconds
const MOISTURE = 'field1';

// Connect to the micro:bit
var port = new SerialPort(MICROBIT_PORT, {
  baudRate: 115200,
  parser: SerialPort.parsers.readline('\n')
}, function (err) {
  if (err)
    return console.log('Error: ', err.message);
  console.log('connected');
});

// Receive new data from micro:bit
port.on('data', function(data){
  var data = data.split(':');
  if (data.length == 2){
    var name = data[0];
    var value = parseInt(data[1]);
    console.log(name + ': ' + value);
    // TODO: push data to ThingSpeak as "MOISTURE=value"
  }
});

// LIBRARY CODE - do not change code below
function startFog(f){
  setTimeout(() => {f(); startFog(f)}, UPDATE_TIME * 1000);
}

function pushData (params, cb=()=>{}) {
  var url = `${CLOUD_URL}&${params}`;
  request.get(url, (error, response, body) => {
    if (error) return cb(error);
    console.log(`request on "${url}" status ${response.statusCode} body ${body}`);
    cb (null, body);
  });
}
```

Listing 3. Gateway code primer.

```
//WINDOWS:
List ports with
 ".\node_modules\.bin\serialport-list.cmd"
Example port on Windows 'COM4'
//LINUX/MAC:
List ports with
 "node_modules/.bin/serialport-list"
Example port on Linux/Mac '/dev/ttyACM0'
```

Listing 4. Commands to list serial ports.

received in the last 15 seconds. Listing 5 lists the changes to be made to (complete the version of) Listing 3.

Subsequently, the instructor can discuss with the class how to further improve the proposed solution, so to include other Fog techniques in the gateway code. For instance, during our practical session, the students observed that the amount of data

```
// Receive new data from micro:bit
var sum = 0
var n = 0

port.on('data', function(data){
  var data = data.split(':');
  if (data.length == 2){
    var name = data[0];
    var value = parseInt(data[1]);
    console.log(`${name}: ${value}`);
    sum += value;
    n++;
  }
});

startFog(fogFun);

function fogFun() {
  pushData(`${MOISTURE}=${sum/n}`);
  sum = 0;
  n = 0;
}
```

Listing 5. Changes to the gateway code.

transmitted to the Cloud can be further reduced by sending an update only when the difference between the current and the previous average is considered significant, i.e., greater than a set threshold.

Furthermore, they pointed out that – even when no significant change in the moisture is registered – there is a need to keep the ThingSpeak plot up-to-date. Hence, they decided to send an update whenever no data was streamed to the Cloud for a long time period, e.g. an hour.

As shown by the wheels symbol in Figure 4(c), in this last example scenario, the application intelligence spreads the available continuum of devices that we set up from the IoT to the Cloud, as in Fog scenarios. Indeed, the instructor's laptop acts as a Fog node that filters out unnecessary data, either because it will not be processed at the Cloud level (i.e., due to ThingSpeak constraints), or being not meaningful to final users (e.g., when moisture has not changed much in the last time period).

V. Related Work

Most of the research work done in creating course CS and Engineering curricula focuses on IoT only, IoT+Edge and IoT+Cloud scenarios.

The research work related to IoT-only [23]–[29] focused on the use of single-board computers to teach computing skills to high school or freshmen students. [23] discusses the experience of using Arduino to teach basic principles of programming and computer hardware. Target participants viewed computers as a black box with little understanding of the inner workings. Students in the course were required to design projects that can access and actuate different sensors (e.g., lighting an LED bulb, user identification using fingerprint sensor). Pre- and post-surveys on student learning revealed positive feedback in learning programming skills. [25] discusses the experience of creating a semester course on

IoT for freshman students. Course participants were subjected to hands-on session on using Arduino to access different sensors and actuators. To overcome the fear of mistake, students were encouraged to make extensive use of online simulators and learn from experience. Using the skills learnt from the course, students were able to program a navigation robot and temperature monitoring and cooling project.

Similarly, [26] and [27] discuss the result of teaching activities involving Arduino, for instance, controlling a LED light using Arduino and marking the attendance using RFID sensors. [29] discusses a hands-on approach and project-based learning to teach an IoT-based course, using Arduino and Raspberry Pi. The projects scenarios discussed include smart home (exploiting temperature sensors), smart classroom (exploiting of NFC tag) and smart library (which makes use of noise sensing sensors) use cases. [28] describes a course on Artificial Intelligence (AI) using the LEGO Mindstorms toolkit. Course participants were asked to implement core concepts of AI (e.g., reactive and deliberative agents, rule-based systems, graph search algorithms, and planning methods) by exploiting such toolkit. The course feedback revealed that the hands-on activity helped students in better learning the proposed AI concepts. [24] discusses the experience of teaching basic concepts of IoT devices to students using just a simulator. With the help of Android phone simulator, students participated in exercises to access and manipulate different sensors (e.g., vibration, accelerometer and GPS sensors).

On the other hand, some research work [30], [31] have focused on IoT+Edge scenarios in teaching. [30] describes an IoT+Edge use case scenario using Raspberry Pi as Edge device. The main goal was to explore single-board computers like Raspberry Pi for client-server communication with the help of different wireless communication technologies (i.e., Wi-Fi, ZigBee). Raspberry Pi acted as an Edge device by processing different file sharing requests coming from different users. The example demonstrated the use of Raspberry Pi for introducing the concept of Edge computing. Similarly, [31] describes the use of Raspberry Pis as an edge device in a smart campus scenario. Such edge devices perform different computation such as aggregation of local data and user queries executing on-device.

Other researchers [32], [33] focused on developing course curriculum to teaching IoT+Cloud scenarios. [32] discusses the experience of teaching a course on modelling and simulation of IoT+Cloud scenarios. Students were taught to write programs that can access different devices such as infra-red (IR), humidity and temperature sensors via a Raspberry Pi. As part of the course project, students wrote an IoT application to address irrigation problems in a smart farming scenario. [33] described the experience on a course on IoT called *My Digital Life* for The Open University. The focus of the learning activity was not only to let students understand basic IoT programming concepts but also to let them understand the impact of the increase in the IoT generated data. The students were asked to develop applications for various domains (i.e., energy, transport, health, business and daily life) that were based on collecting data from various IoT sensors and storing them on a back-end cloud servers. This activity triggered collaborative learning among students and resulted in some interesting projects from different domains.

To the best of our knowledge, most of the previous research was directed towards teaching IoT-only, IoT+Edge and IoT+Cloud scenarios and none focused on developing hands-on lab activities, practically illustrating the difference between Fog computing and other existing deployment models for IoT applications.

VI. Concluding Remarks

In this paper, we have described a highly practical active learning lab which can be useful for introducing Fog computing in CS and Engineering curricula. We run the proposed lab within our *Advanced Software Engineering* course, having all students able to fully complete and enjoy it.

The proposed activity also meets the objectives we set for it at design time. As per $\mathbf{O}_1$, it helps to practically understand the need for Fog computing, whilst showing some peculiar properties of Fog computing, against other existing deployment models for IoT applications (viz., IoT+Edge and IoT+Cloud). It highlights contextual location awareness of the Fog node that aggregates data coming from all the micro:bits that are transmitting moisture measurements within the radio range of the dashboard micro:bit (i.e., around 70 metres [21]). It clearly shows the efficacy of Fog computing to achieve bandwidth savings and to locally process data, whilst supporting mobility of IoT devices (plants (cups) in the greenhouse can be moved exploiting battery-powered micro:bits). It simply demonstrates the possibility of integrating heterogeneous hardware (e.g., embedded systems, laptops), software (i.e., firmware, different OSs) and communication technologies (i.e., radio, serial, Wi-fi Internet access) in Fog scenarios.

Secondly, the activity meets $\mathbf{O}_2$, thanks to the possibility of programming both the micro:bits and the gateway module with a high-level language (JavaScript). A lab session to host this activity can be completed within a 2-hour class. Students with some background in programming can successfully complete the activity with the help of the instructor and by reading the available online documentation. Thirdly, the activity can be organised with a limited cost for buying the micro:bits and all necessary materials, that is $\mathbf{O}_3$ is met.

Naturally, being a didactic example, the activity shows some limitations and room for further improvements. Indeed, some important characteristics of the Fog paradigm, such as security issues, mobility of Fog nodes and availability of differently capable devices (e.g., smartphones, routers, micro data centres) in the Fog layer were considered orthogonal to the first design of the lab session. In fact, it would be interesting to investigate the possibility of further modifying the testbed architecture from the IoT+Edge to the Fog scenario. Additionally, despite a dashboard micro:bit can potentially handle 25 micro:bits measuring moisture (i.e., around $75-100$ students in a single lab session, considering groups of $3-4$ people), we left testing scalability of our approach to future editions of the course.

Also, quantitatively measuring actual learning outcomes is left for future work.

Before concluding, we wish to highlight that the practical lab session we have described can constitute the basis for other active or project-based learning (or assessment) activities. Possible extensions include (just to mention some):

- extending the IoT testbed and application by exploiting other sensors and actuators (e.g., the light and temperature sensors embedded in the micro:bits) as well as different communication technologies (e.g., micro:bit's Bluetooth), and by experimenting with different form of synchronisation among the devices,
- (re-)structuring and extending the available codebase so to build a microservice-based architecture [34], including at least a database component, an intermediate Fog dashboard and some more advanced processing happening both in the Fog and in the Cloud layer,
- exploiting a larger variety of hardware (e.g., Raspberry Pi, Arduino) and programming languages (e.g., Python, Java, C) so to better show the Fog potentials in integrating heterogeneous systems, and
- collecting experimental data about the experiment so to actually quantify the benefits and overhead of adopting Fog computing with respect to IoT+Edge and IoT+Cloud deployment models.

ACKNOWLEDGEMENTS

We wish to warmly thank the students that enthusiastically took part in the lab sessions of the *Advanced Software Engineering* course. Thanks are also due to the *Pisa CoderDojo* club (https://pisa.coderdojo.it) that kindly lent us their micro:bits to successfully carry out the described lab session on Fog computing.

REFERENCES

[1] OpenFog, "OpenFog Reference Architecture," 2016. [Online]. Available: https://www.openfogconsortium.org/ra

[2] M. Satyanarayanan, "The Emergence of Edge Computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[3] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile Edge Computing-A key technology towards 5G," *ETSI White Paper*, vol. 11, no. 11, pp. 1–16, 2015.

[4] H. L. Truong and S. Dustdar, "Principles for Engineering IoT Cloud Systems," *IEEE Cloud Computing*, vol. 2, no. 2, pp. 68–76, Mar 2015.

[5] A. Brogi, S. Forti, and A. Ibrahim, "How to best deploy your Fog applications, probably," in *Proceedings of 1st IEEE International Conference on Fog and Edge Computing (ICFEC), Madrid*, O. Rana, R. Buyya, and A. Anjum, Eds., May 2017, pp. 105–114.

[6] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog Computing: A Platform for Internet of Things and Analytics," in *Big Data and Internet of Things: A Roadmap for Smart Environments*, 2014, pp. 169–186.

[7] A. Brogi and S. Forti, "QoS-aware Deployment of IoT Applications Through the Fog," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1185–1192, Oct. 2017.

[8] M. Iorga *et al.*, "The NIST Definition of Fog Computing (draft SP 800-191)," Aug. 2017. [Online]. Available: https://csrc.nist.gov/csrc/media/publications/sp/800-191/draft/documents/sp800-191-draft.pdf

[9] Y. Ai, M. Peng, and K. Zhang, "Edge Computing Technologies for Internet of Things: A Primer," *Digital Communications and Networks*, 2017.

[10] A. V. Dastjerdi and R. Buyya, "Fog Computing: Helping the Internet of Things Realize its Potential," *Computer*, vol. 49, no. 8, pp. 112–116, 2016.

[11] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on Fog Computing: Architecture, Key Technologies, Applications and Open Issues," *Journal of Network and Computer Applications*, 2017.

[12] "OpenFog Consortium," http://www.openfogconsortium.org/.

[13] M. Ben-Ari, "Constructivism in Computer Science Education," vol. 30, no. 1, pp. 257–261, 1998.

[14] O. Hazzan, T. Lapidot, and N. Ragonis, *Guide to Teaching Computer Science - An Activity-Based Approach*. Springer, 2015.

[15] "Spelling of the words euro and cent in official community languages as used in community legislative acts – European Commission Directive," Jan. 2009. [Online]. Available: http://ec.europa.eu/economy_finance/publications/pages/publication6336_en.pdf

[16] C. C. Byers, "Architectural Imperatives for Fog Computing: Use Cases, Requirements, and Architectural Techniques for Fog-Enabled IoT Networks," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 14–20, 2017.

[17] G. Halfacree, "Meet the BBC micro: bit," *The Official BBC micro:bit® User Guide*, pp. 1–15.

[18] "Micro:bit Developer Community – Hardware Description." [Online]. Available: http://tech.microbit.org/hardware/

[19] E. L. Park and B. K. Choi, "Transformation of Classroom Spaces: Traditional versus Active Learning Classroom in Colleges," *Higher Education*, vol. 68, no. 5, pp. 749–771, 2014.

[20] "Micro:bit Docs – Soil Moisture." [Online]. Available: https://makecode.microbit.org/projects/soil-moisture

[21] "Micro:bit Docs." [Online]. Available: https://makecode.microbit.org/reference/

[22] "Thingspeak." [Online]. Available: https://thingspeak.com

[23] R. Shultz, D. E. Ueda, J. S. Ward, and A. K. Fontecchio, "A Hands-On, Arduino-Based Approach to Develop Student Engineering Skills and Introduce Cybersecurity Concepts to K-12 Students," in *2015 ASEE Annual Conference & Exposition*, no. 10.18260/p.23395. Seattle, Washington: ASEE Conferences, June 2015.

[24] S. Abraham, "Using Internet of Things (IoT) as a Platform to Enhance Interest in Electrical and Computer Engineering," in *2016 ASEE Annual Conference & Exposition*, no. 10.18260/p.27149. New Orleans, Louisiana: ASEE Conferences, June 2016, https://peer.asee.org/27149.

[25] N. Wu and K. Zeng, "Embedded System Based First-Year Engineering Course with Aid of Online Simulation and Social Media," in *8th Annual First Year Engineering Experience (FYEE) Conference*, Columbus, August 2016.

[26] Y. Jang, J. Kim, and W. Lee, "Development and Application of Internet of Things Educational Tool based on Peer to Peer network," *Peer-to-Peer Networking and Applications*, Oct 2017. [Online]. Available: https://doi.org/10.1007/s12083-017-0608-y

[27] N. Yaácob, A. Yusof, A. Azhar, N. Naim, N. Nur, and A. Mahmon, "Rfid lab management system using Arduino microcontroller approach associate with webpage," *Journal of Scientific Research and Development*, vol. 3, no. 2, pp. 92–97, 2016.

[28] M. P. Cuéllar and M. Pegalajar, "Design and Implementation of Intelligent Systems with LEGO Mindstorms for Undergraduate Computer Engineers," *Computer Applications in Engineering Education*, vol. 22, no. 1, pp. 153–166, 2014.

[29] Z. Bogdanovic, K. Simic, M. Milutinovic, B. Radenkovic, and M. Despotovic-Zrakic, "A Platform for Learning Internet of Things." *International Association for Development of the Information Society*, 2014.

[30] C. W. Zhao, J. Jegatheesan, and S. C. Loon, "Exploring IOT Application Using Raspberry Pi," *International Journal of Computer Networks and Applications*, vol. 2, no. 1, pp. 27–34, 2015.

[31] K. Hentschel, D. Jacob, J. Singer, and M. Chalmers, "Supersensors: Raspberry Pi Devices for Smart Campus Infrastructure," in *Future Internet of Things and Cloud (FiCloud), 2016 IEEE 4th International Conference on*. IEEE, 2016, pp. 58–62.

[32] S. S. Patil, S. Katwe, U. Mudengudi, R. B. Shettar, and P. Kumar, "Open Ended Approach to Empirical Learning of IOT with Raspberry Pi in Modeling and Simulation lab," in *IEEE 8th Int. Conf. on T4E*. IEEE, 2016, pp. 258–259.

[33] G. Kortuem, A. K. Bandara, N. Smith, M. Richards, and M. Petre, "Educating the Internet-of-Things generation," *Computer*, vol. 46, no. 2, pp. 53–61, 2013.

[34] S. Newman, *Building Microservices: Designing Fine-Grained Systems*. " O'Reilly Media, Inc.", 2015.