Creative Internet of Things (IoT) for Undergraduates

Suk Jin Lee
TSYS School of Computer Science
Columbus State University
Columbus, GA
lee_suk@columbusstate.edu

Andrew Jung
Department of Computing Sciences
University of Hartford
West Hartford, CT
jung@hartford.edu

Mira Yun
Department of Computing Science & Networking
Wentworth Institute of Technology
Boston, MA
yunm@wit.edu

Abstract— **This article presents results of our pilot study for a new Internet of Things (IoT) course for CS undergraduates. Most of the current IoT courses proposed focused on designing and developing new classroom activities and projects. However, we strongly believe that open-ended student project can enhance students' creative thinking. After providing fundamental theories and related hands-on activities, we allowed students to choose their own final project topic. Since our course covers hardware platforms, programming, networking protocols, and data storage of IoT systems and applications in the classroom, students can be ready for exploring new real-world problems at the end. By choosing their own project topic, students have a chance to apply their classroom leanings to solve new problems creatively and learn new skills by themselves. Course evaluation based on student feedback shows high level of enthusiasm and engagement for the course especially for hands-on activities and final open-ended project.**

Keywords—Internet of Things for undergraduate, Hands-on activities, Open-ended project, Creative Learning

I. INTRODUCTION

Internet of Things (IoT) is the fast growing network paradigm that interconnects physical computing devices and everyday objects, and collects and shares information over the Internet. It has wide range of application such as smart home, wearables, smart city, and digital health care, and all applications have been developed to improve our everyday quality of life. According to this rapid growth of IoT applications and services, the demand for experienced professionals in the area has been increased. Universities and institutes all over the world have developed various courses and programs to offer IoT concepts and practices [1].

A current educational paradigm in STEM (Science, Technology, Engineering, and Mathematics) education is "Lean-by-Doing" [2][3], where students learn through hands-on activities within a practical environment. Our classroom experience clearly showed that hands-on activities in a collaborative team environment is very important to prepare our computer science students for careers in the future. In addition, it is important that computer science students understand how their programming code works with hardware [4]. Since IoT is rapidly growing as the next generation technology and integrates hardware and software, computer programming, wireless and mobile networking, cloud computing, and big data, IoT is a good but not easy topic to deliver in the classroom that gives students a motivating environment to learn the technology by doing hands-on activities in a team oriented environment.

Most of the current IoT courses proposed focused on designing classroom activities and projects by using off-the-shelf products such as Raspberry Pi (RPi) and Arduino. However, we strongly believe that open-ended student project can enhance students' creative thinking. The course we designed and presented in this paper gives students an opportunity to learn necessary skillset to develop the product through the creative learning process that includes projects, peers, passion, and play [5].

We designed the IoT course such that students can learn and practice their skills to enhance technical competency to keep up with trends in the growing technology-driven world. The course is 3000 level course that is for junior and/or senior students who are majoring computer science. The course contents introduce the fundamentals of IoT, its related concepts, such as open IoT architecture, fog computing and cloud computing. This course also provides platforms and solutions supporting development and deployment of IoT application with commercially available off-the-shelf products. It also covers how to measure a large volume of data from sensors, how to communicate end devices with gateway, and how to apply the data generated from IoT end devices to backend services. Based on lecture, we also designed 10 hands-on labs that can practice what they learn immediately. The course also gives students a project that can implement and develop their own application in the given platform.

Course evaluation based on student feedback shows high level of enthusiasm and engagement for the course especially for the labs and a project. The rest of the paper organized as follows: in Section II, we present the motivation and background of IoT education. In Section III we describe the details about course contents and findings. Finally, Section IV concludes our work and outlines ideas for future development.

II. MOTIVATION AND BACKGROUND

We designed the course contents for the students who are majoring in Computer Science. The IoT course requires students to have a number of pre-requisite courses, including programming, computer architecture, and computer networks. There have been many efforts to implement IoT concepts and skills into the classroom, and we found several researches that implement IoT courses in their curriculum [6][7][8][9][10].

Jing He, et al [6], developed courses for undergraduate students. Their courses are based on modules that can be integrated into various courses in STEM area. They implement eleven modules that are IoT based frameworks to embedded system. They used RPi for their experiments, and gotten good feedback from students. They surveyed with Linkert-scale questions. Their results showed that using RPi helped students to prepare for complex topics that were discussed later in the course (6.17 out of 7); using RPi made them easy to design, write and test software (5.67 out of 7); RPi prepared students for connecting sensors to the tool (6.0 out of 7.0); and also responded that C/C++ language is more appropriate than python in embedded system course (5.5 out of 7.0).

Koji Akiyama, et al [7], proposed and implemented IoT prototype system for students who are not in STEM area. They mentioned that their system lead students to create ideas for IoT system. They used Arduino and RPi for hardware; LED, a buzzer, and motor as actuator; ZigBee as sensor network; windowPC or RPi as a gateway. Using their system, students follow their stepwise framework to create IoT system without serious programming involvement. They applied their system to humanity courses. They mentioned that students were able to create their ideas as an IoT system design in that course.

Stan Kurkovsky, et al [8], presented the IoT-centric course for Computer Science curriculum using RPi, Arduino, and sensors. This course was designed for students who have programming background with data structures. They divided the IoT project into three incremental phases for students. They mentioned that RPi is a good device that allows students to experience the four layers of IoT functionalities. They got very positive feedback from students using RPi to study IoT.

Simon, et al [9], designed and proposed the IoT course for graduate level students. In addition to general classroom activities such as lecture and test, they gave students programming projects for building user interfaces, industrial design projects for large-scale project with several companies around Silicon Valley in California, and survey paper for having the research opportunity. They mentioned that students were interested in doing the programming and industrial projects, and some students' survey papers were published.

Most IoT courses proposed used RPi and Arduino for hardware to get information from sensors, and analyzed it not only in the classroom activities but also for student projects. Our course also used RPi and Arduino as a hardware platform, and gave students a chance to experience it. However, we give students an independent open-ended project with given sensors and hardware platforms that enhances students' creative thinking, not the ordinary project that based on lecture only. The course we designed gives students an opportunity to learn necessary skillset to develop the product through the creative learning process [5].

III. IoT FOR UNDERGRADUATES

The IoT needs several technologies, such as hardware, networking and communication, computer programming, and cloud computing, to accomplish the quality of services. Hence, it is an important topic to bring into the classroom environment for students, not only to learn the IoT foundation but also to experiment and practice techniques they had learned previously. Our IoT course has the following four modules:

1. *Hardware Platforms*: there are several hardware platforms available nowadays. We identified the following features as requirements to select the hardware platforms for applied IoT: electronic interface, programmability, wired or wireless Internet stack, low cost and commercially available, and future proof [11]. The most popular prototyping platforms are Arduino and RPi. Arduino is getting popular among educators because their software libraries make allow students at all levels to easily write code and help them expand challenging projects without knowledge of electronics. We made use of Arduino to implement applications that require multiple sensors and/or actuators. The RPi is one of low-cost single-board computers. It has distinctive mark of four USB ports, an Ethernet port, an HDMI port, and GPIO ports. The RPi is a useful platform if Ethernet connection is required to support real-time applications. One of the roles of RPi in the course is a network gateway for end devices to access Internet. All those hardware are off-the-shelf low cost products. In addition to the basic hardware platforms, the students used many different types of sensors for their specific applications.

2. *Programming*: the Arduino integrated development environment (IDE) is a cross-platform application and supports the languages C and C++. This environment allows students to write and upload programs to the Arduino board through Microsoft Windows OS. The SoftwareSerial library allows serial communication on other digital pins of the Arduino so that application developers can access ESP8266 Wi-Fi module on top of breadboard, by wiring different types of sensors. The Python and/or C languages are available for programming on Raspbian, which is a debian-based computer operating system for RPi. For the application development, the course used Node.js to support web standards and protocols, because Node.js is an open-source, cross-platform JavaScript run-time environment and processed by all browsers asynchronously, which allows us to execute JavaScript code outside of a browser.

3. *Networking Protocols* (Wireless Technology): the hypertext transfer protocol (HTTP) is one of the most widely used application protocols for communicating over the Internet. We establish communication between RPi and the server by using HTTP. Meanwhile, message queue telemetry transport (MQTT) is an extremely lightweight messaging protocol based on the publish/subscribe model over TCP. We also set up Mosquitto broker/client, which is an open source message broker to implement the MQTT protocol on Raspbian OS. For wireless technology, we used Wi-Fi add-on module with Arduino and set up an RPi as an access point in a standalone network.

4. *Data Storage* (Cloud computing): the data collected from end devices are forwarded to backend cloud storage. For our laboratory activities, we used ThingSpeak, which is an open-source IoT application and API to store and retrieve data from things using the HTTP protocol over the Internet.

TABLE I. COURSE SCHEDULE

Week	Day	Topic	Assignment
1	Day 1	Lecture: Class Administration Lab: Overview on IoT, intro to Raspberry Pi	Quiz Hands-on activity
	Day 2	Lecture: OPEN IoT Architecture Lab: Programming on Raspbian	Quiz Hands-on activity
	Day 3	Lecture: Device-Cloud Collaboration Framework Lab: HTTP Server and Client	Quiz Hands-on activity
	Day 4	Lecture:Fog Computing Lab: Control GPIO with web server	Quiz Hands-on activity
	Day 5	Lecture: Programming Frameworks for IoT Lab: Message Queuing Telemetry Transport	Quiz Hands-on activity
2	Day 6	**Test 1** Lab: Wi-Fi Hotspot with Raspberry Pi	Quiz Hands-on activity
	Day 7	Lecture: Stream Processing in IoT Lab: Programming on Arduino	Quiz Hands-on activity
	Day 8	Lecture: Security in IoT Lab: DHT11 sensor	Quiz Hands-on activity
	Day 9	Lecture: Obfuscation and Diversification Lab: Raspberry Pi security	Quiz Hands-on activity
	Day 10	Lecture: Applied IoT Lab: ESP8266 with Arduino Uno	Quiz Hands-on activity
3	Day 11	Lecture: Internet of Vehicles and Applications Lab: Temperature Data Upload on Cloud	Quiz Hands-on activity
	Day 12	**Test 2** Final Project	Final Project
	Day 13	Final Project	
	Day 14	Final Project	
	Day 15	Final Project	

A. Course Schedule and Contents

The course is a 3-credit course for computer science majoring junior/senior students. We designed the student's learning experience with hands-on activities (labs) and a project that extends their learning to develop their own applications. Thus, we give a short lecture relatively meaning that 30 minutes lecture, and 2 hours labs meaning hands-on activity. We designed this course for regular semester course originally, but it had been adapted to winter session that is 3 weeks program. Each week had 5 days classes, each class had two and half hours length. Thus, the class was the same time amount as the regular semester class. Table I presents details of the course topics for each day.

The course contents of the first week introduce IoT architecture and frameworks with initial RPi settings. Day 1 lecture introduces overview of the course and IoT. For the lab, students practice how to set up RPi and execute remote desktop access from Windows OS to RPi. Because Raspbian is an RPi Linux OS, students are able to practice many basic Linux commands. Day 2 lecture covers open standards for IoT Services toward interoperability. For the lab, students use C and/or Python programming languages and start to practice how to use and control the hardware, such as blinking LEDs by wiring to the breadboard. Day 3 lecture introduces a big picture of collaboration framework between end devices and cloud computing. In order to accommodate the collaborative framework, students learn how to communicate with each entity and transmit the data over the Internet. The course module introduces the TCP/IP network protocols and a dynamic language, JavaScript, as a bridge to extend an application layer protocol HTTP. The students conduct practical hands-on activities; setting up HTTP server/client, installing Node.js on Raspbian and/or Windows OS and sending HTTP requests to their HTTP server using methods such as GET and POST [12]. Day 4 lecture covers principles, architectures, and applications of fog computing [13]. Students are aware of a distributed computing paradigm, extending the cloud services to the edge of the network. For the lab, students practically control RPi GPIOs via web interface using RPi Node.js server. Day 5 lecture introduces a software architectural style, Representational State Transfer (REST), and message passing protocols to communicate between heterogeneous devices, such as MQTT and Constrained Application Protocol (CoAP). For the lab, students install Mosquitto broker/client on Raspbian OS and/or Windows OS, publishing a sample message to a topic and subscribing to the topic with multiple clients.

Fig. 1. Wi-Fi Hotspot with Raspberry.

The second week covers wireless and security components with RPi hotspot and Arduino sensors. Day 6 in Lab teaches how the RPi can be used as a wireless access point, running a standalone network. We configure the DHCP server and the access point host in RPi, running Wi-Fi hotspot, as shown in Fig. 1. The Ethernet port is connected to Internet through the local network.

```
#include <SoftwareSerial.h>

#define RX 10
#define TX 11

String AP = "RPiNetwork";          // CHANGE ME
String PASS = "CPSC1234567890";    // CHANGE ME

String API = "WKUNV7AMNYROMXB6";   // CHANGE ME
String HOST = "api.thingspeak.com";
String PORT = "80";
String field = "field1";

int countTrueCommand;
int countTimeCommand;
boolean found = false;
int valSensor = 1;

SoftwareSerial esp8266(RX,TX);
```

Fig. 2. Use of SoftwareSerial library

Day 7 lecture goes over the data processing paradigms and the characteristics of stream data. For the lab, students start to write and upload programs to Arduino board. This activity continues to the next day using a physical sensor, e.g. temperature and humidity sensor, so that the students are familiar with Arduino IDE. Day 8 lecture introduces the security issues of IoT and continues to the next day with general approaches for securing IoT. In Day 9 Lab, students practice how to make a secure connection to the server in Raspbian. Day 10 lecture goes over key components of IoT architecture including sensors and actuators, gateway and backend services, and requirements of gateway hardware/ software. Day 10 Lab teaches students how to use ESP8266 Wi-Fi module as an add-on and write code in the Arduino to talk to it. After testing ESP8266 Wi-Fi module directly, we teach students how to set up the virtual serial for the Arduino module, include the library, and define the variables, as shown in Fig 2.

```
void setup() {
   Serial.begin(9600);
   esp8266.begin(115200);
   sendCommand("AT",5,"OK");
   sendCommand("AT+CWMODE=1",5,"OK");
   sendCommand("AT+CWJAP=\""+ AP +"\",\""+ PASS
               +"\"",20,"OK");
}
```

Fig. 3. `Setup` code for Arduino Uno

We used SoftwareSerial library to connect ESP8266 Wi-Fi module through pins 10 and 11. From here, we will talk to Arduino through "ESP8266" and talk to the IDE through "Serial". We set the serial connection to a 9600 baud rate and ESP8266 connection to a 115200 baud rate, as shown in Fig. 3. The `setup` code establishes the connection between Wi-Fi module and Wi-Fi hotspot implemented in Day 6 Lab.

After successfully initializing the setup stage, as shown in Fig. 4, the `Loop` code executes all the instructions repeatedly unless the power supply of Arduino stopped working. For a simple operation test, we generated a random number and uploaded the value to one of cloud service providers. The students can replace this value with other measurement, e.g. temperature, humanity, etc., for their own projects later.

```
void loop() {
 valSensor = getSensorData();
 String getData = "GET /update?api_key="+ API +"&"+ field
+"="+String(valSensor);

 sendCommand("AT+CIPMUX=1",5,"OK");  // CIPMUX=0: single
connection, 1: multiple connection
 sendCommand("AT+CIPSTART=0,\"TCP\",\""+  HOST  +"\","+
PORT,15,"OK");  // CIPSTART = 0, id of connection
 sendCommand("AT+CIPSEND=0,"
+String(getData.length()+4),4,">");        // ID no. of
transmit connection

esp8266.println(getData);delay(1500);countTrueCommand++;
 sendCommand("AT+CIPCLOSE=0",5,"OK");
}
int getSensorData(){
  return random(1000); // Replace with real data
}
```

Fig. 4. `Loop` code for Arduino Uno

```
void  sendCommand(String  command,  int  maxTime,  char
readReplay[]) {
  Serial.print(countTrueCommand);
  Serial.print(". at command => ");
  Serial.print(command);
  Serial.print(" ");
  while(countTimeCommand < (maxTime*1))
  {
    esp8266.println(command);//at+cipsend
    if(esp8266.find(readReplay))//ok
    {
      found = true;
      break;
    }
    countTimeCommand++;
  }
  if(found == true)
  {
    Serial.println("OYI");
    countTrueCommand++;
    countTimeCommand = 0;
  }
  if(found == false)
  {
    Serial.println("Fail");
    countTrueCommand = 0;
    countTimeCommand = 0;
  }
  found = false;
 }
```

Fig. 5. `sendCommand` function code for Arduino Uno

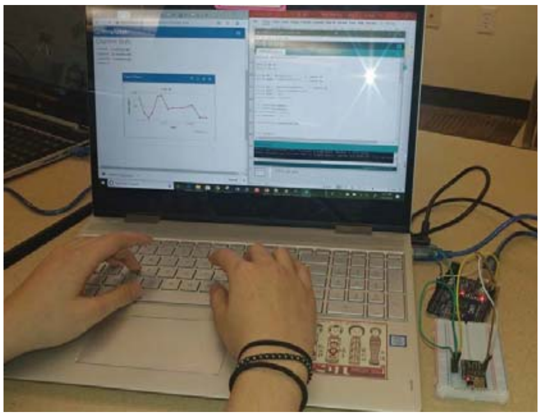

Fig. 6. A student works on uploading a random number to a cloud service provider, Thingspeak, using Wi-Fi add-on module with Arduino.

The `sendCommand` function codes create a package to exchange messages between ESP8266 Wi-Fi module and RPi Wi-Fi hotspot. It has three arguments; 1) string variable `command`; 2) integer variable `maxTime`; and 3) character variable `readReplay`. The string variable sends the operational command to ESP8266 Wi-Fi module. This command repeats `maxTime` times. If the expected response arrived within `maxTime` times, the operational command is successful; otherwise, the function alerts the user to recognize the failure of sending a message.

Students are also requested to create an endpoint they can send some data to for the test. For the simplicity, we employ Thingspeak, which is an open-source IoT application and API to store and retrieve data from things using the HTTP protocol over the Internet [14]. Fig. 6 shows that a student uploads a random number to a cloud service provider, Thingspeak, using Wi-Fi add-on module with Arduino through Wi-Fi hotspot. We use Wi-Fi hotspot implemented in Day 6 Lab, where Wi-Fi hotspot allocate a local network IP address for Wi-Fi add-on module.

```
#include <DHT.h>
#define DHTPIN 7    // where the dht11 is connected
DHT dht(DHTPIN, DHT11);
...
void setup() {
   Serial.begin(9600);
   esp8266.begin(115200);
   dht.begin();
   ...
   }
void loop() {
   // valSensor = getSensorData();
   float c = dht.readTemperature(); // celsius
   float f = c*9/5 + 32; // fahrenheit
   String getData = "GET /update?api_key="+ API +"&"+
field +"="+String(f);
...
}
/*
  int getSensorData(){
  return random(1000); // Replace with real data
}*/
```

Fig. 7. Include DHT library in the code to measure temperature data.

The last week of this course focuses on the design and development of IoT applications. Day 11 lecture shows that the dawn of a new era of IoT will drive the evolution of conventional vehicular ad-hoc networks (VANETs) into the Internet of Vehicles (IoV). For the lab, students use ThingSpeak as a cloud service provider and temperature-humidity sensor (DHT11) as an end device to measure temperature data. Students are instructed to include DHT library for humidity sensor to define the variable in the code, and define setup and loop functions to get temperature data, depicted in Fig. 7. Students upload temperature data to a cloud service provider, Thingspeak, using Wi-Fi add-on module with Arduino through RPi Wi-Fi hotspot. A portable battery powers the end devices including Arduino, Wi-Fi add-on module and temperature-humidity sensor (DHT11), as shown in Fig. 8, which makes the hands-on activity one of real-world applications with real-time data.

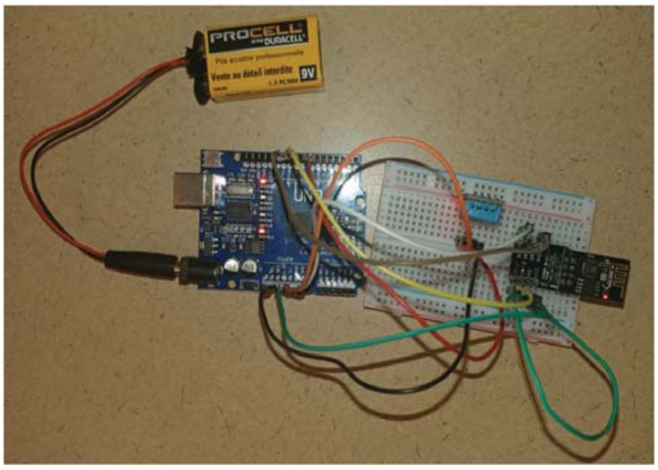

Fig. 8. A portable battery powers the end devices including Arduino, Wi-Fi add-on module and temperature-humidity sensor (DHT11).

TABLE II. SUMMARY OF STUDENTS' WORKS

Number	Project Topics	Sensors Used
1	Fire Alarm	Flame sensor, buzzer, LED light, ESP8266
2	Outside Light Triggered on By Sensitive Sounds	Big sound sensor, ESP8266
3	Obstacle Alarm	Avoidance sensor, buzzer, ESP8266
4	Laser Focused: A Tap Module and Laser Equipped Cloud Deterrent Device	Tap sensor, laser emitter, ESP8266
5	Motion Detection System using Tilt Sensor	Tilt Sensor, LED, buzzer, ESP8266
6	Shock Detection	Shock sensor, buzzer sensor, LED, ESP8266
7	TEMP TO LIGHT COLOR!	RGB light, LCD LED module, DHT 11 sensor, ESP8266

B. *Open-ended Project*

In the last week, students are allowed to choose their own final project topic in the given platforms. The course implicates the final projects with many different types of sensors, exemplified by temperature-humidity sensor, buzzer, tilt sensor, flame sensor, tap sensor, laser emitter, avoidance sensor, sound sensor, LCD LED module, etc., as shown in Table II.

We found that the skillsets used in the student projects were tightly related to the lessons learned throughout the semester. We also recognize that each team choose a unique project topic to create their own IoT system such as Fire Alarm, Obstacle Alarm, Laser Focused, Software-Based Emergency Button, etc. *Students designed and built their own specific projects based on their creative ideas.* For example, one of student projects uses flame sensor and buzzer, which was never covered in the regular lecture hours. If the fire activity is detected, the system activates a buzzer and the notification message is sent to a cloud service provider. Another project uses avoidance sensor and buzzer. If an obstacle is approached, the buzzer immediately starts buzzing until the obstacle is out of range. The system also sends the signals “0” or “1” to a cloud service provider. Table II summarizes students’ work produced throughout the course.

IV. Conclusion and Future Work

In this paper, we present a newly designed IoT course that is for computer science junior and senior students. We designed our course contents with four different modules: hardware platforms, programming, networking protocols and data storage. The design of this course may motivate students to enhance their creative thinking through the process of their project adapting the central concepts of IoT.

Based on our experiences, students actively participate in the labs. Our hands-on lab activities serve as a motivating environment to engage classroom learning. When students complete their hands-on activities, they were willing to help out other students in the classroom. Students remarked that they wanted to complete their lab activities successfully regardless of their grades, which was unusual, based on our experiences in other courses.

We designed this course so that students choose any topic related to IoT for their final project, meaning that the final project is an open-ended project. After selecting a topic, the students start to work on their topic. We recognized that open-ended projects give students a chance to think creatively and motivates them to learn new skills by themselves. For example, throughout the project, students studied by themselves third party libraries that are not discussed in the classroom to communicate between hardware platform and sensors such as Serial Peripheral Interface (SPI) Library [15], Wire Library [16] and Adafruit_SSD1306 Library [17]. Throughout learning new skills by themselves, they were developing and creating new IoT systems. We believe that creative thinking is an important part of higher education because we want to guide students to build strong problem solving skills.

This is our pilot study for the newly proposed IoT course. We plan to extend this course into the regular semester and explore the student self-efficacy through self-reporting survey and interviews including students’ program analysis. We also plan to implement the designed course at more institutions as a part of their curriculum.

References

[1] K. K. Rout, S. Mishra and A. Routray, "Development of an Internet of Things (IoT) Based Introductory Laboratory for Undergraduate Engineering Students," *2017 International Conference on Information Technology (ICIT)*, Bhubaneswar, 2017, pp. 113-118.

[2] M. M. Kombardi, “Authentic learning for the 21st century: An overview,” Educause learning initiative, Report No. 1, pp. 1-12, 2007.

[3] G. Frache, H. E. Nistazakis, and G. S. Tombras, “Reengineering engineering education: Developing an constructively aligned learning-by-doing pedagogical model for 21st century education,” IEEE Global Engineering Education Conference, pp. 1119-1124, 2017.

[4] S. Ristov, N. Ackovska, V. Kirandziska, and M. Gusev, “ Is the computer science curriculum ready to teach students towards hadwarizing?,” IEEE Global Engineering Education Conference, pp. 397-402, 2016.

[5] M. Resnick, “Give P’s a chance: Projects, Peers, Passion, Play,” Constructionism and Creativity Conference, pp. 13-20, 2014.

[6] J. He, D. Lo, Y. Xie, and J. Lartigue, “Integrating Internet of Things (IoT) into STEM undergraduate education: Case study of a modern technology infused courseware for embeded system course,” IEEE Frontiers in Education Conference, pp. 1-9, 2016.

[7] K, Akiyama, M. Ishihara, N. Ohe, and M. Inoue, “An education curriculum of IoT prototype construction system,” IEEE 6th Global Conference on Consumer Electronics, pp. 1-5, 2017.

[8] S. Kurkovsky, and C. Williams, “Raspberry Pi as a platform for the Internet of Thinigs projects: Experiences and Lessons,” ACM Conferences on Innovation and Technology in Computer Science Education, pp. 64-69, 2017.

[9] G. Simon, and M. Koo, “An integrated curriculum for Internet of Things: Experience and Evaluation,” IEEE Frontiers in Education Conference, pp. 1-4, 2015.

[10] H. Maenpaa, S. Carjonen, A. Hellas, S. Tarkoma, and T. Mannisto, “Assessing IoT projects in university education – A framework for problem-based learning,” IEEE/ACM 39th International Conference on Software Engineering: Softwre Engineering Education and Training Track, pp. 37-46, 2017.

[11] S.J. Johnston, M. Apetroaie-Cristea, M. Scott, S.J. Cox, Internet of Things: Principles and Paradigms, Ch. 15 - Applied Internet of Things, pp. 277-298, 2016, Morgan Kaufmann.

[12] Paul Golding, "HTTP, WAP, AJAX, P2P and IM Protocols," in Next Generation Wireless Applications: Creating Mobile Applications in a Web 2.0 and Mobile 2.0 World , , Wiley, 2008

[13] H. Hong, "From Cloud Computing to Fog Computing: Unleash the Power of Edge and End Devices," *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, Hong Kong, 2017, pp. 331-334.

[14] https://en.wikipedia.org/wiki/ThingSpeak

[15] https://www.arduino.cc/en/Reference/SPI

[16] https://www.arduino.cc/en/reference/wire

[17] https://github.com/adafruit/Adafruit_SSD1306