2023 Annual Conference & Exposition
Baltimore Convention Center, MD | June 25 - 28, 2023
The Harbor of Engineering Education for 130 Years
ASEE
Paper ID #37489

Introducing Internet-of-Things (IoT) – A Remote Approach

Dr. Samia Tasnim, The University of Toledo

Samia Tasnim, PhD, is an Assistant Professor in the department of Electrical Engineering and Computer Science at the University of Toledo. She received her M.S. and Ph.D. degrees, along with the prestigious dissertation year fellowship award, in Computer Science from Florida International University. Her research interests include Internet of Things (IoT), mobile computing, security, and data mining. Application domains include smart cities, intelligent transportation networks, healthcare and environment monitoring.

Teaching Internet-of-Things (IoT) – A Remote Approach

Samia Tasnim
Department of Electrical Engineering and Computer Science
The University of Toledo
Toledo, OH, USA
Samia.Tasnim@utoledo.edu

Abstract

There has been rapid growth in internet-of-things (IoT) over the last few years. According to grand view research, the IoT market value will reach $933.62 billion by 2025. Moreover, the number of connected devices will become 1 trillion by 2025, per HP's report. To prepare the students to be well aware of these novel technologies, we need to update our curriculum and course design. In this paper, I present some laboratories (labs) that the students conducted as a part of a course project in the ubiquitous computing class. This course is an elective for undergraduate Computer and Information Sciences or Information technology students. The students who take this course are either juniors or seniors. Covid-19 has taught us how remote teaching is useful to ensure proper education during the time of the pandemic. This project aims to design different lab modules that the students can conduct without purchasing hardware. I designed this course at the time of covid pandemic to ensure student learning and success in an economical way. I devised multiple assignments that helped the students to conduct hands-on lab using a remote simulation tool (e.g., Tinkercad). The end-of-the-semester positive student reviews support the success of the course design.

1 Introduction

Internet of things (IoT) is a system that connects people and devices (e.g., sensors, smartphones, wearables) anytime and anywhere. The fast-growing IoT system has influenced and materialized many industries, such as smart cities, smart grid, smart healthcare, and so on. Kevin Ashton first coined the term Internet of Things as "refers to scenarios where network connectivity and computing capability extends to objects, sensors and everyday items not normally considered computers, allowing these devices to generate, exchange and consume data with minimal human intervention" [1].

According to grand view research, the IoT market value will reach $933.62 billion by 2025. Moreover, the number of connected devices will become 1 trillion by 2025, per HP's report. To prepare the students to be well aware of these novel technologies, we need to update our curriculum and course design for courses such as: Ubiquitous Computing and Mobile Computing.

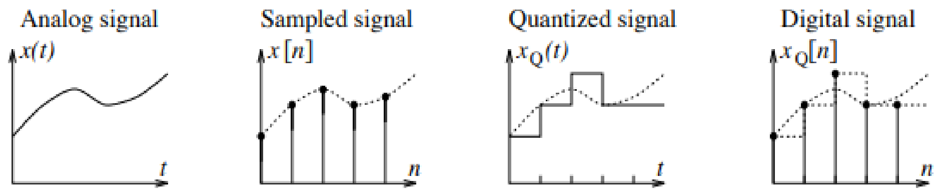


Figure 1: Analog, sampled, quantized and digital Signals [2].

In this paper, I present some laboratories (labs) that the students conducted in the ubiquitous computing course.

Large universities such as: Stanford University, University of Urbana Champaign, University of California Irvine, University of California Sandiego, and Florida International University, among others have recently introduced new courses or even a degree programs on Internet of Things. In every university or college, offering a new program or course might not be feasible due to resource constraint (e.g., limited faculty to cover the courses). But, like our approach, an existing CSE or IT or CIS course curriculum can be modified to include the emerging IoT concepts. As a results, the students will be well aware of these novel technologies and become a deserving candidate in the job market or for graduate studies.

One of the objectives of this course is to give students the ability to identify the sensors and other devices needed for different ubiquitous IoT solutions. In addition, they get to know about the analog and digital system in detail. The students learns the basic elements of IoT, and Iot programming with arduino.

The paper is organized as follows. I provide the background information in Section 2. Next, in Section 3 I discuss in detail two laboratories. In section 4, semester-end student comments are presented. Finally, Section 5 concludes the paper.

2 Background

2.1 Analog and Digital Data

In real-world, there are myriad examples of signals. Such as: the variation in air pressure when we speak, the daily highs and lows in temperature, and the periodic electrical signals generated by the heart [2]. In the past, we had analog signals only. But, now because of the advent of communication and wireless technology digital signals are more prominent. In physical world we see analog signals but the IoT devices understand in terms of digital. Thus, analog to digital conversion is a core educational component.

Some important concepts related to the analog and digital conversions are described as follow.

- Sampling – converts the continuous signal into a series of discrete analog signals at periodic intervals.

- Quantization - each discrete analog is converted into one of a finite number of (previously defined) discrete amplitude levels
- Encoding – discrete amplitude levels are converted into digital code

2.2 IoT Sensors

Sensing is core component of the IoT. There are various applications of IoT such as: environment monitoring [3, 4], agriculture, healthcare, air quality monitoring, traffic surveillance [5], smart home, etc. Different human, environment or objects are sensed for the better realization of the experimental environment and to make intelligent decisions.

In environmental sensing, different sensors are deployed to sense various environmental properties (e.g., humidity, temperature, and ozone) [6, 7, 8, 9]. The sensors are mounted on top of different vehicles (e.g., bicycle, bus, private car, and tram) or carried by a human being, which change their positions very frequently and unpredictably. Moreover, sensors have been used for IoT-based air quality monitoring applications [10, 11, 12].

A sensor is a device that receives a stimulus and responds with an electrical signal (voltage, current) [13]. Some examples of stimulus include: motion, velocity, pressure, temperature, sound, among others. To better apprehend the existing Iot devices and design future smart applications that deals with different sensors, proper understanding of sensors is necessary. This course aims to introduce the students into IoT sensors. Arduino is the most popular IoT device development platform and will be used in the laboratories. More details is presented in the next section.

3 Lab Instructions

In this section, I present the detailed description of two labs that I designed for the ubiquitous computing course. This course (3 credits) does not have additional lab hours, so the students conduct the labs during the regular class time of one hour fifteen minutes. The students are provided additional days outside the lab hours to write the report and submit.

3.1 Analog to Digital Lab

Tinkercad Circuits is a browser-based program. It allows one to design and simulate circuits at free of cost that in the same way it would have behaved if implemented using hardware (e.g., breadboard, wire, sensors, etc.).

3.1.1 Procedure Details

A student should log in to tinkercad [14] using personal gmail account. Then click the “New” button, and from the options select circuit. Clicking on the top left text allows one to rename the project. It is advisable to give a proper name that insinuates the purpose of the project. Students are recommended to append their name at the end of the project name as well.

Step 1: Circuit Building

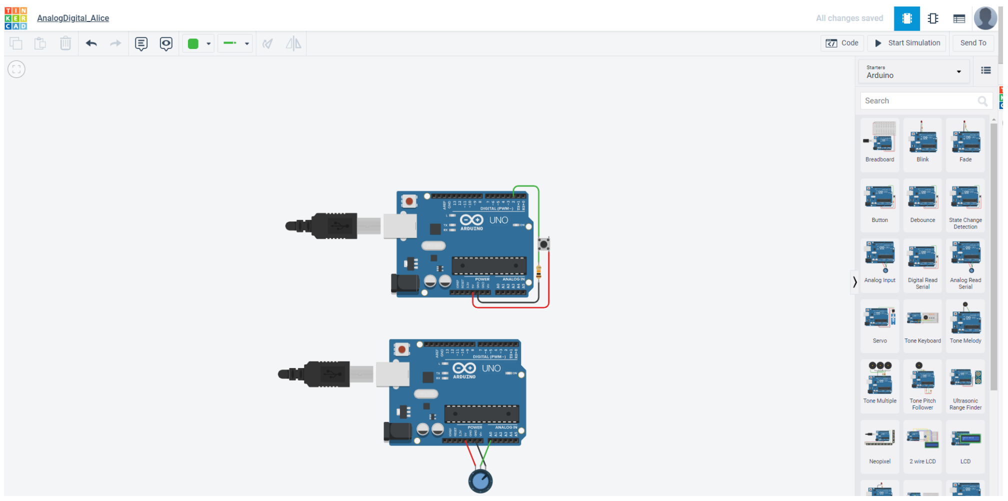


Figure 2: Circuit View of the Analog to Digital Lab.

A student can choose from different circuit components that are located at the top right corner of the browser. First one needs to click on the components drop down menu to select the desired component, and then drag it to the middle of the project workspace. For this lab, we will be using two arduino circuits named as “Digital Signal Read” and “Analog Signal Read” in the components list. After incorporating the above mentioned components, the circuit will look similar to Fig. 2. The tinkercad website allows one to zoom in or out the workspace area by simply using the mouse roller ball. Zoom out is very helpful when the circuit contains multiple components.

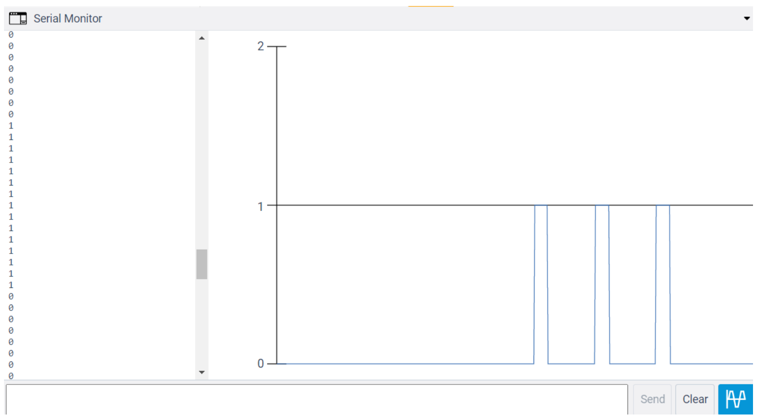
Step 2: Printing to the Serial Monitor With Blocks

(i) We will utilize the code blocks editor for listening to an Arduino input pin. Next, the analog value or digital state will be print out in the $SerialMonitor$ window. To open the code panel, a student needs to click the “Code” button.

(ii) Next, s/he is asked to click on the $SerialMonitor$ which is located at the bottom of the code panel.

(iii) To run the Arduino code, s/he should click “Start Simulation”, and observe the numbers in the Serial Monitor during the interaction with the potentiometer. As the potentiometer input value changes by moving the pointer on the dial, the serial output value will change accordingly. Since the circuit includes two independent Arduinos, students can click back and forth between the two Arduinos while the simulation is running in Tinkercad Circuits. However, the output in the serial monitor will reflect whichever Arduino was selected at the beginning of the simulation start. For example, while interacting with the potentiometer, only the analog circuit (lower one in Fig. 2) will display output in the serial monitor. In contrast, if the student clicks on the upper arduino before clicking the $StartSimulation$ button, s/he will notice the serial monitor starts displaying 0 (the default digital output when push button is not pressed) on the serial monitor. As soon as s/he presses the push button, the serial monitor will print 1 and then go back to the default 0 state.

Graph Output: Tinkercad allows to visualize the circuit output data in graph format. Though,



```
0
0
0
0
0
0
0
0
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
0
0
0
0
0
0
0
0
```

Figure 3: Serial Monitor and Graph Output of digital data read circuit.

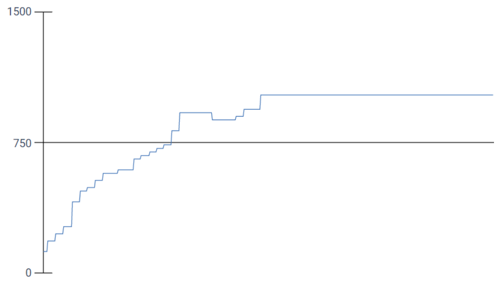


Figure 4: Graph Output of Analog data read circuit.

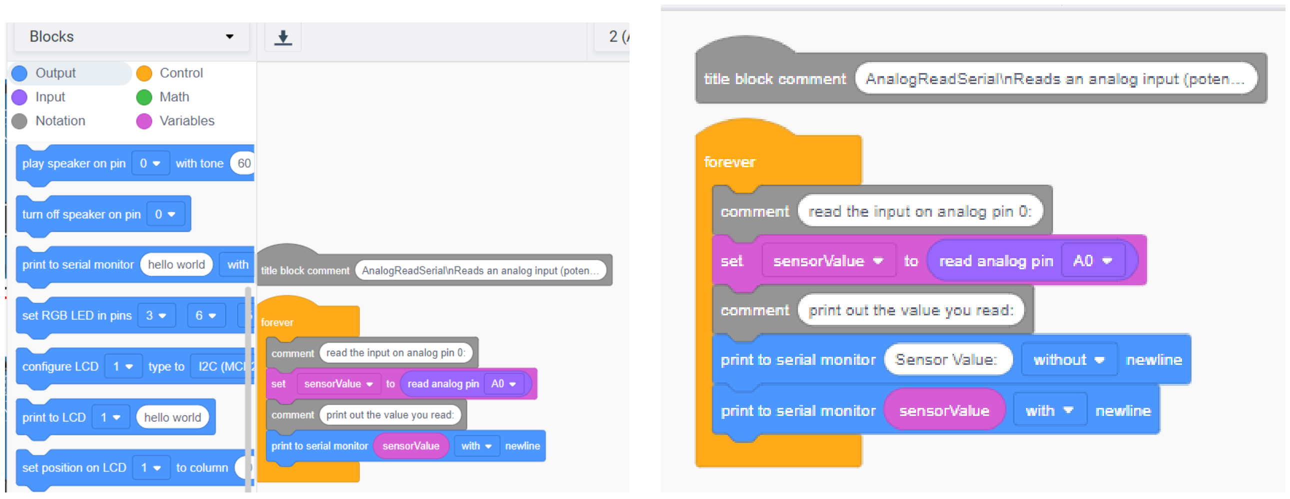
for printing value in the graph, the serial monitor print should not contain any characters, only numbers are allowed. This is a limitation of the graph showing option.

Fig. 3 displays the simulation output of the “digital signal read” (Top Arduino of Fig. 2). The output displayed 1 when the push button was pressed. The graph reflects the serial monitor values. On the other hand, Fig. 4 displays the graph output reflecting difference potentiometer values. The output value range is [0,1023]. Definitely, the output graph will vary based on the potentiometer interaction during the running the simulation. We will explain the reason behind this data range in the next section.

(iv) In Fig. 5 (a), we can see different options available under the “Blocks” section. Navigate to the Output code category, then drag out a “print to serial monitor” block and place it just before the serial block that is already in the program. A student can change the default text to label the Serial data, such as “Sensor Value: ”, and from the dropdown menu either choose to print with or without a new line. Please note, in case of Fig. 4, the default block code has been used, where a number is printed on the serial monitor. In contrast, after the code block configuration as shown on 5 (b), the serial monitor output looks similar to Fig. 6. A student can stack similar serial blocks in any future project for printing useful relevant feedback messages, as well as append relevant comment blocks.

Step 3: Arduino Code

In Fig. 5, only the “Blocks” drop down menu is selected under the “Code” editor. In this step, the


Blocks
Output
Control
Input
Math
Notation
Variables
play speaker on pin 0 with tone 60
turn off speaker on pin 0
print to serial monitor hello world with
set RGB LED in pins 3 6
configure LCD 1 type to I2C (MC
print to LCD 1 hello world
set position on LCD 1 to column
title block comment AnalogReadSerial\nReads an analog input (poten...
forever
comment read the input on analog pin 0:
set sensorValue to read analog pin A0
comment print out the value you read:
print to serial monitor Sensor Value: without newline
print to serial monitor sensorValue with newline

(a) (b)

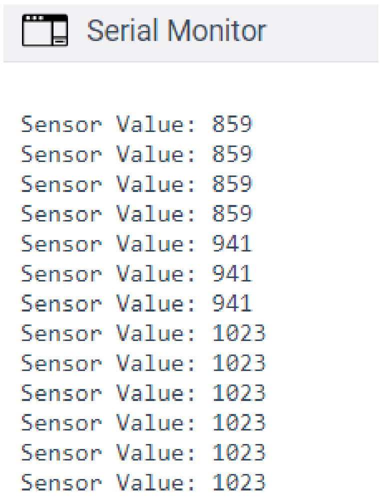
Figure 5: Using Code Block for Output Configuration (a) Default scenario, and (b) Reconfigure by prepending "Sensor Value: ".

student will explore the "Blocks + Text" drop down item. After s/he clicks on the "Blocks + Text" option, the arduino code will be made visible beside the code block, as shown in Fig. 7. Please note, the Arduino code is auto generated while the block items were added in the "Blocks" view. The first function in the code section, named $setup()$ as shown on Fig. 7 and Fig. 8, a variable "INPUT" is created to hold the state of the input. To enable the message sending feature, an Arduino requires opening a new communication channel. That is being done in the Serial.begin() in line 22 and line 17 of Fig. 7 and Fig. 8, respectively. The argument of this function defines the communication speed. In this case, the transmission speed is 9600 baud (1 baud = 1 bit per second).

Inside the loop() method, corresponding input data is read and assigned to the declared variables. In Fig 7, the input is read from the analog pin A_0 and assigned to the variable $sensorValue$. In contrast, in Fig. 8, the input is read from the digital pin 2, and then assigned to $buttonState$. Inside the arduino, the analog pins are names A_0, A_1, ..., A_5. Tinkercad allows "Code" only option as well where none of the "Blocks" are used. However, only "Code" option is out of the scope of this course.

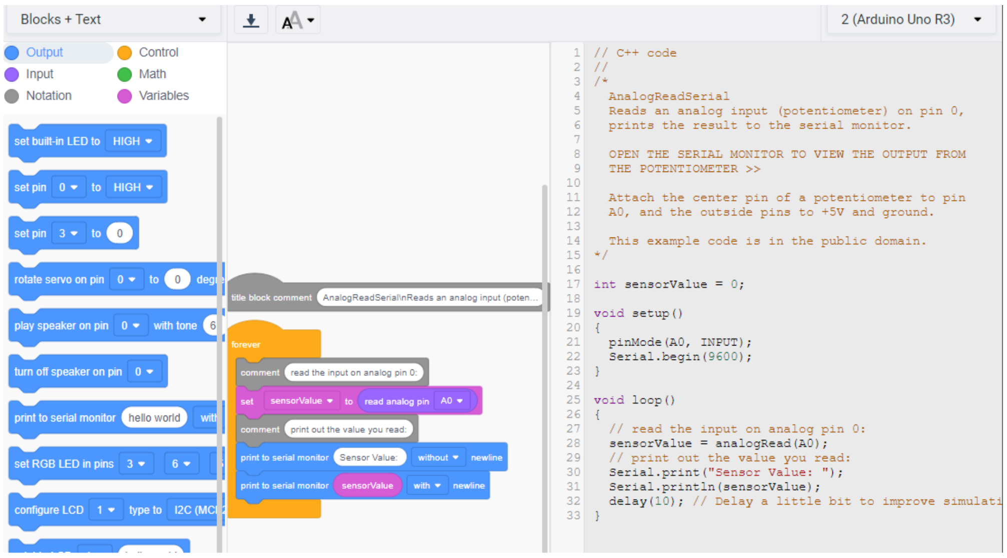
Step 4: Code Debugger

Like state-of-the-art integrated development environments (IDEs), Tinkercad circuits provides the programmers an option to debug their code. It allows to step through the code and peek into variables while the simulation is going on. To enable the debugger, the student needs to select either Block+Text mode or Text only option from the dropdown list. The, s/he needs to click on a line number for adding a break point (e.g., line 28 in Fig. 9). Then, while the simulation is running, the debugger will stop at that particular line each time through the loop. S/he has to hover over the variables to peek the values during the pause time.



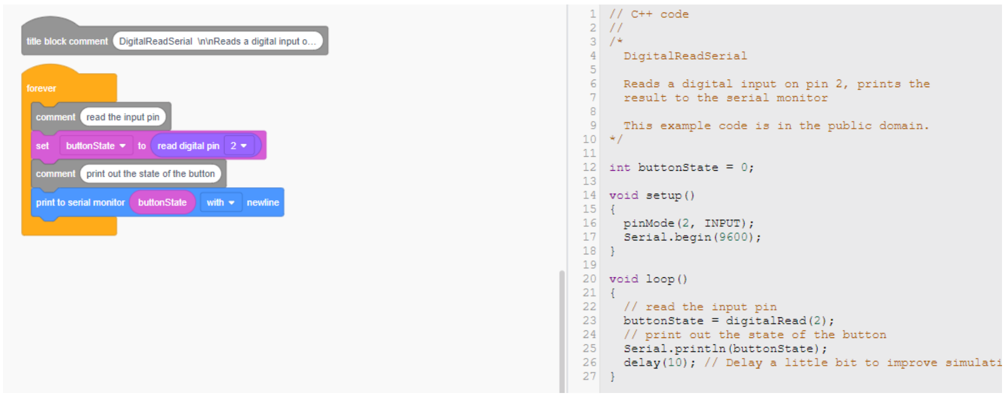
```
Sensor Value: 859
Sensor Value: 859
Sensor Value: 859
Sensor Value: 859
Sensor Value: 941
Sensor Value: 941
Sensor Value: 941
Sensor Value: 1023
Sensor Value: 1023
Sensor Value: 1023
Sensor Value: 1023
Sensor Value: 1023
Sensor Value: 1023
```

Figure 6: Serial Monitor output of Analog data read circuit post code block reconfiguration.


Blocks + Text
2 (Arduino Uno R3)
Output
Control
Input
Math
Notation
Variables
set built-in LED to HIGH
set pin 0 to HIGH
set pin 3 to 0
rotate servo on pin 0 to 0 degr
play speaker on pin 0 with tone 6
turn off speaker on pin 0
print to serial monitor hello world with
set RGB LED in pins 3 6
configure LCD 1 type to I2C (MCI
title block comment AnalogReadSerial\nReads an analog input (poten...
forever
comment read the input on analog pin 0:
set sensorValue to read analog pin A0
comment print out the value you read:
print to serial monitor Sensor Value: without newline
print to serial monitor sensorValue with newline

```
// C++ code
//
/*
  AnalogReadSerial
  Reads an analog input (potentiometer) on pin 0,
  prints the result to the serial monitor.

  OPEN THE SERIAL MONITOR TO VIEW THE OUTPUT FROM
  THE POTENTIOMETER >>

  Attach the center pin of a potentiometer to pin
  A0, and the outside pins to +5V and ground.

  This example code is in the public domain.
*/

int sensorValue = 0;

void setup()
{
  pinMode(A0, INPUT);
  Serial.begin(9600);
}

void loop()
{
  // read the input on analog pin 0:
  sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.print("Sensor Value: ");
  Serial.println(sensorValue);
  delay(10); // Delay a little bit to improve simulati
}
```

Figure 7: Block and Text View of Analog Circuit.


title block comment DigitalReadSerial \n\nReads a digital input o...
forever
comment read the input pin
set buttonState to read digital pin 2
comment print out the state of the button
print to serial monitor buttonState with newline

```
// C++ code
//
/*
  DigitalReadSerial

  Reads a digital input on pin 2, prints the
  result to the serial monitor

  This example code is in the public domain.
*/

int buttonState = 0;

void setup()
{
  pinMode(2, INPUT);
  Serial.begin(9600);
}

void loop()
{
  // read the input pin
  buttonState = digitalRead(2);
  // print out the state of the button
  Serial.println(buttonState);
  delay(10); // Delay a little bit to improve simulati
}
```

Figure 8: Block and Text View of Digital Read Circuit.

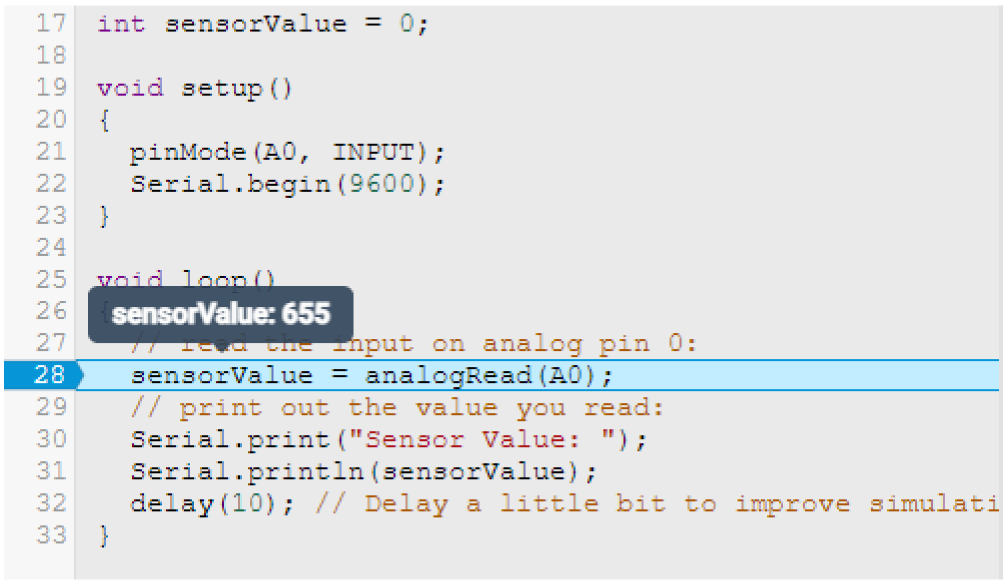

```
int sensorValue = 0;

void setup()
{
  pinMode(A0, INPUT);
  Serial.begin(9600);
}

void loop()
{
  // read the input on analog pin 0:
  sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.print("Sensor Value: ");
  Serial.println(sensorValue);
  delay(10); // Delay a little bit to improve simulati
}
```

Figure 9: Debugging the SensorValue variable.

3.1.2 Deliverable

After completing the lab following the above-mentioned instructions, a student needs to submit a document (word/pdf) using designated learning management system (LMS), such as: blackboard, canvas, etc, that will include the following items:

(a) A short description (4/5 sentences) on what is an Arduino (use internet search).

(b) Screenshot of the block and code section.

(c) 3 screenshots of serial monitor values based on the variation of potentiometer and push-button.

A Student can include his/her designed circuit link. To do so, s/he needs to click the button $SendTo$ (located beside start simulation), after that click $Invitepeople$ and finally click the $copylink$ option on the tinkercad.

3.2 Temperature Sensor Lab

In this lab assignment, the students will get familiarized with a temperature sensor, which has myriad applications in real-world starting from healthcare or body temperature measurement [15, 16] to agriculture [17, 18] and environment monitoring [3, 8]. A student needs to login to tinkercad and create a new circuit similar to the instructions mentioned at the beginning of section 3.1.1.

3.2.1 Procedure Details

The step by step instructions for completing the temperature sensor lab is mentioned as follows.

Step 1: LED Circuit Building

This lab will incorporate more components than the previously discussed analog to digital lab. To better arrange multiple components and to ensure that they are connected properly, a student will use a breadboard in this lab.

(i) First one needs to click on the components drop down menu to select the desired component, and then drag it to the middle of the project workspace. For this lab, we will be fetching an Arduino Uno and breadboard (under Arduino components, first item named as breadboard) from the components panel to the workplace. Here, the 5 volt and ground pins on the Arduino are connected to the power (+) and ground (-) rails on the breadboard by wires. As a general practice, black wire is used for connecting to ground, and red colored wire for connecting to the 5V. Tinkercad allows one to change the default wire colors by making a color selection from the color dropdown menu. Alternatively, after selecting the wire one can type a number, then the corresponding color from the dropdown list will be direcly reflected on the selected wire.

(ii) The student needs to drag three LEDs and place them on the breadboard in row D, spaced 2 breadboard sockets apart. The placement row can vary, the current placement is advised for proper spacing. Generally, the LED id is assigned as 1,2, etc. It is advised to rename 1 as LED_1. Similar to wires, the student can change the LED color using the inspector that pops up while clicking on the LED.

(iii) Next, s/he should use a resistor to connect each LED's cathode (left leg) to the ground rail (black) of the breadboard. In Tinkercad Circutis, a student can change a resistor's value by highlighting it and using the dropdown menu in the inspector. The default resistance value is $1k\Omega$, for this experiment the student should update the resistance value to 220Ω.

(iv) Finally, the student needs to connect the three LED anodes (right, longer legs) to digital pins 4, 3, and 2, respectively on the Arduino. The LED anode (+) is the terminal that current flows into. On the other hand, the cathode (-) is the source terminal of current and connected to the ground rail of breadboard.

Step 2: Adding Temperature Sensor

(i) The student needs to drag the temperature sensor from the component lists and place on the row D (same row as the LEDs) of the breadboard. The search option facilitates finding a desired item from the component list in a time-efficient manner.

(ii) The temperature sensor (TMP36) has three legs, hovering on each leg of the sensor, s/he will understand the functionality of that particular leg. The left leg is connected to power, middle one to Voltage output, and right leg is connected to ground. Thus, a red wire is used to connect the left leg to the 5V rail on the breadboard, and black wire to connect the right leg of the sensor with the ground (-) rail. The middle leg is connected to the A_0 analog pin of the Arduino. Similar to the Analog read circuit of Fig. 2, the changing voltage is input from the temperature sensor to the Arduino.

(iii) When a student clicks on the start simulation, a panel appears on top of the temperature sensor, that allows to change the temperature value (similar to a regulator). In this case, the temperature range is [-40^{0}C, 125^{0}C] . A temperature sensor creates a changing voltage signal

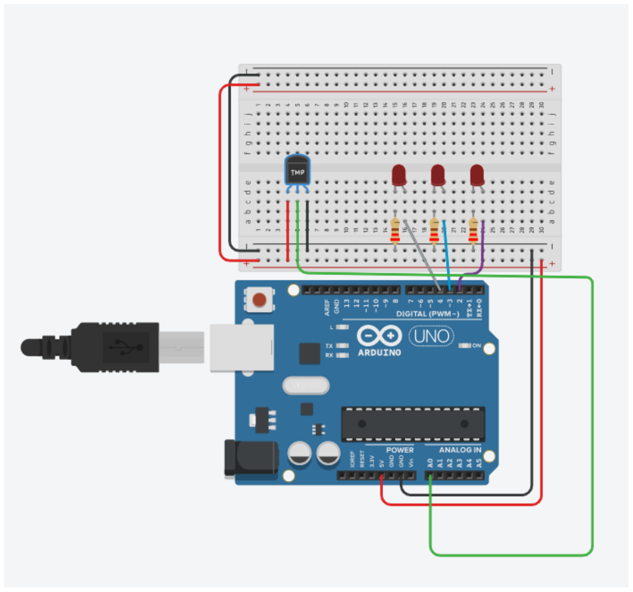


Figure 10: Circuit View of the Temperature Sensor Lab.

depending on the temperature it senses. This temperature sensor model (TMP36) is a good option because of it's capability of generating output voltage directly proportional to the sensed temperature in degree Celsius.

After incorporating the above mentioned components, the circuit will look similar to Fig. 10. In addition to the circuit view, tinkercad allows one to observe the schematic diagram. Fig. 11 shows the schematic view of the circuit developed in the temperature sensor lab.

Arduino has a built-in Analog- to-Digital Converter (ADC). The five analog pins A0 to A5 can interpret voltages between[0V, 5V], and translate that voltage to a value between [0, 1023].

Step 3: Block Code

(i) We will utilize the code blocks editor for listening to an Arduino input pin. Next, the analog value (temperature sensor read data in terms of celcius and Farenheit) will be print out in the Serial Monitor window. Also, the LEDs will lit up based on the temperarute sensor value. To open the code panel, a student needs to click the "Code" button.

(ii) Click the "Code" button to open the code editor. Then the student needs to use the block option from the dropdown menu. This project will utilize more complex blocks with logic and math functions in comparison to the Arduino to Digital lab. To enhance the readability of the program, a student is encouraged to include the comment option available under the "Notation" blocks.

(iii) Next, different useful variables are created by clicking on the "Variables" menu. The student needs to create three variables for this lab, named baselineTemp, celsius and fahrenheit. Please note, the naming is upto the programmer. A student needs to use the set variable block whenever a value needs to be assigned to a user declared variable. First, s/he needs to assign the value 20 to baseline Temp. The celsius variable is used for storing the temperature sensor data in Celsius unit.

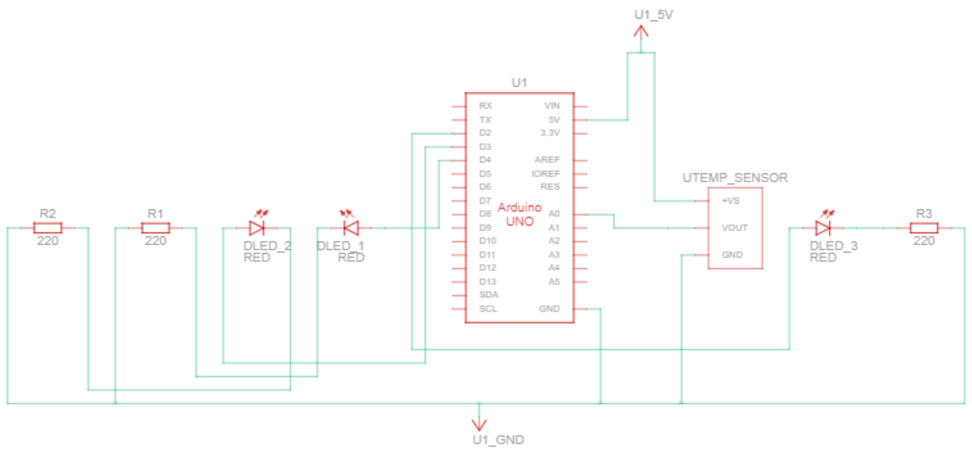


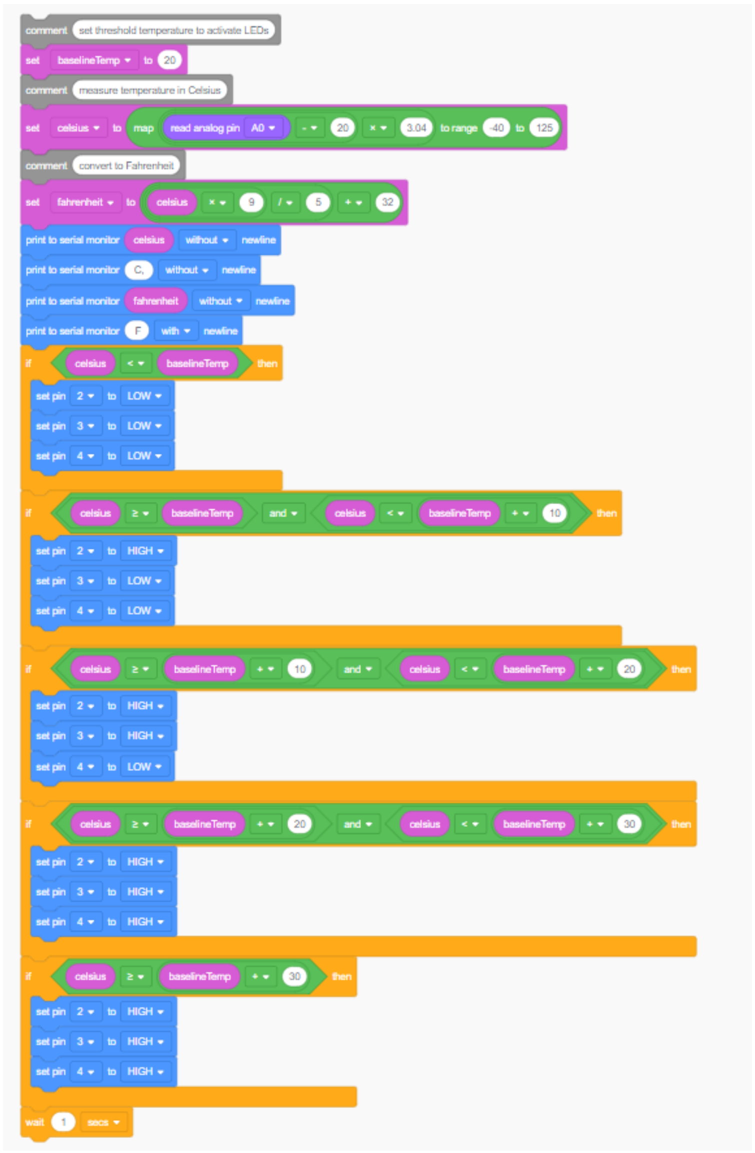
Figure 11: Schematic View of the Temperature Sensor Lab.

The temperature range is [-40, 125]. From Math category, a student needs to drag out a ”map” block, and nest two arithmetic blocks (“1 + 1”) within its first field. Then, to match with the temperature sensor’s input range, s/he needs to adjust the range from -40 to 125. The student needs to drag drag out “read analog pin A0” block from the input category, and place it into the first arithmetic field inside the “map” block. After that, s/he needs to adjust the arithmetic blocks to “(read analog pin A0 - 20) x 3.04”.

(iv) The default unit of temperature sensor is Celsius. As we know, in the USA Farenheit is used for temperature measurement, we will incorporate a coversion from celsius to farenheit and print values in both units on the serial monitor. For that, the student needs to use set block and some arithmetic blocks to read “set fahrenheit to (celsius x 9)/5 + 32”.

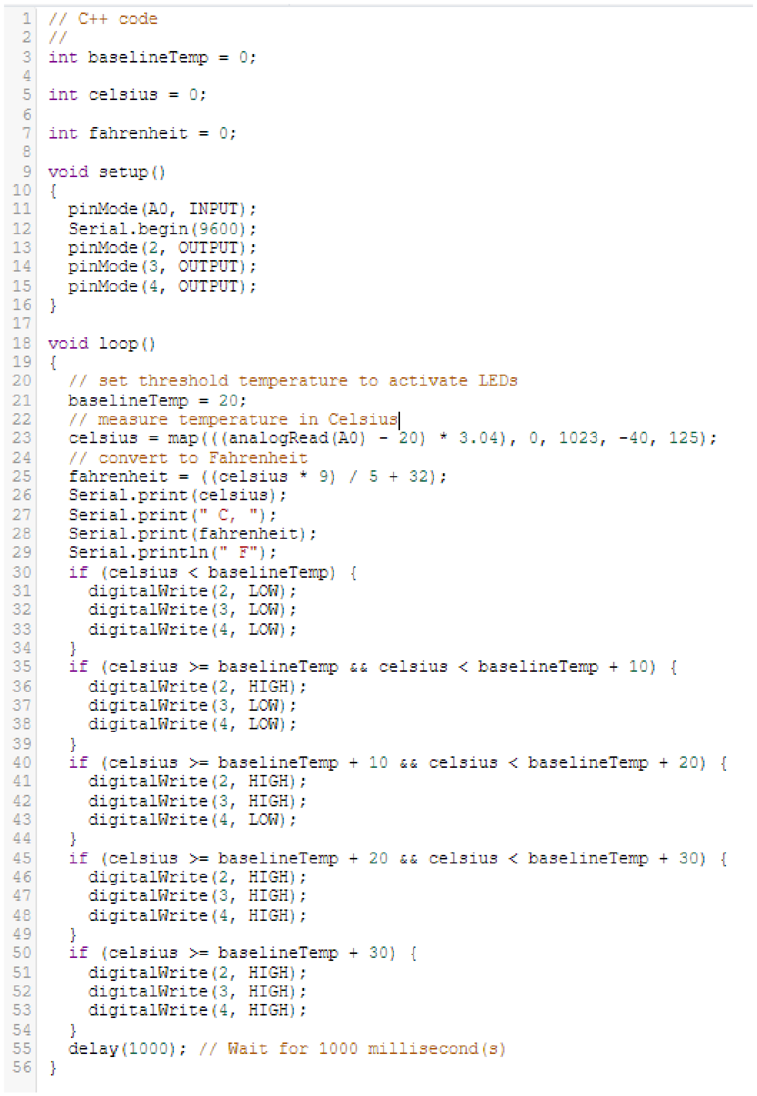
(v) Tinkercad allows to implement if else conditions inside the block editor. For this lab, we will categorize the temperature values (in Celsius) into five ranges and based on that decide which LED to turn on. For implementing this, a student needs to first click the Control category and drag out an if then block, then navigate to Math and drag a comparator block onto the if block. From the Variables category, s/he will drag the “celsius” and the “baselineTemp” variablea into the comparator block, adjusting the dropdown so it reads ”if celsius ¡ baselineTemp then”. The three LEDs are connected to three digital pins (D2, D3 and D4 in Fig 11). Thus, the student has to add three digital output blocks inside the if statement to set pins 2, 3, and 4 LOW for this temperature value condition.

(vi) One can select a block to copy and paste in tinkercad. The student can use this feature to save sometime. S/he can copy the if else blocks of previous step and paste 4 time inside the code blocks. As shown in Fig. 12, When the temperature is greater than or equal to the baselineTemp and less than baselineTemp+10, the condition is set to light up only pin 2’s LED. When the temperature is between baselineTemp+10 and baselineTemp+20, then will light up two LEDs. On the otherhand, when the sensed temperate is greater than or equal to baselineTemp+20, then all three LEDs will turn on. That is being implemented by making the digital output of pin 2, 3 and 4 changing to state “HIGH”.



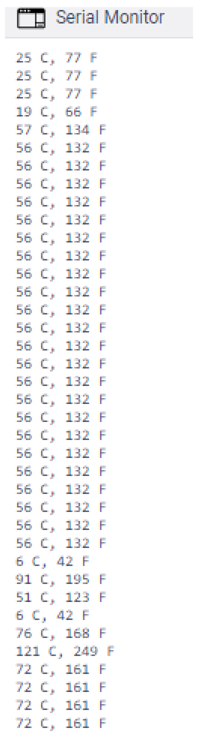
```
comment set threshold temperature to activate LEDs
set baselineTemp to 20
comment measure temperature in Celsius
set celsius to map read analog pin A0 - 20 × 3.04 to range -40 to 125
comment convert to Fahrenheit
set fahrenheit to celsius × 9 / 5 + 32
print to serial monitor celsius without newline
print to serial monitor C, without newline
print to serial monitor fahrenheit without newline
print to serial monitor F with newline
if celsius < baselineTemp then
    set pin 2 to LOW
    set pin 3 to LOW
    set pin 4 to LOW
if celsius ≥ baselineTemp and celsius < baselineTemp + 10 then
    set pin 2 to HIGH
    set pin 3 to LOW
    set pin 4 to LOW
if celsius ≥ baselineTemp + 10 and celsius < baselineTemp + 20 then
    set pin 2 to HIGH
    set pin 3 to HIGH
    set pin 4 to LOW
if celsius ≥ baselineTemp + 20 and celsius < baselineTemp + 30 then
    set pin 2 to HIGH
    set pin 3 to HIGH
    set pin 4 to HIGH
if celsius ≥ baselineTemp + 30 then
    set pin 2 to HIGH
    set pin 3 to HIGH
    set pin 4 to HIGH
wait 1 secs
```

Figure 12: Block view of the Temperature Sensor Lab.



```
// C++ code
//
int baselineTemp = 0;

int celsius = 0;

int fahrenheit = 0;

void setup()
{
  pinMode(A0, INPUT);
  Serial.begin(9600);
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
}

void loop()
{
  // set threshold temperature to activate LEDs
  baselineTemp = 20;
  // measure temperature in Celsius
  celsius = map(((analogRead(A0) - 20) * 3.04), 0, 1023, -40, 125);
  // convert to Fahrenheit
  fahrenheit = ((celsius * 9) / 5 + 32);
  Serial.print(celsius);
  Serial.print(" C, ");
  Serial.print(fahrenheit);
  Serial.println(" F");
  if (celsius < baselineTemp) {
    digitalWrite(2, LOW);
    digitalWrite(3, LOW);
    digitalWrite(4, LOW);
  }
  if (celsius >= baselineTemp && celsius < baselineTemp + 10) {
    digitalWrite(2, HIGH);
    digitalWrite(3, LOW);
    digitalWrite(4, LOW);
  }
  if (celsius >= baselineTemp + 10 && celsius < baselineTemp + 20) {
    digitalWrite(2, HIGH);
    digitalWrite(3, HIGH);
    digitalWrite(4, LOW);
  }
  if (celsius >= baselineTemp + 20 && celsius < baselineTemp + 30) {
    digitalWrite(2, HIGH);
    digitalWrite(3, HIGH);
    digitalWrite(4, HIGH);
  }
  if (celsius >= baselineTemp + 30) {
    digitalWrite(2, HIGH);
    digitalWrite(3, HIGH);
    digitalWrite(4, HIGH);
  }
  delay(1000); // Wait for 1000 millisecond(s)
}
```

Figure 13: Codesnippet of the Temperature Sensor Lab.



```
25 C, 77 F
25 C, 77 F
25 C, 77 F
19 C, 66 F
57 C, 134 F
56 C, 132 F
56 C, 132 F
56 C, 132 F
56 C, 132 F
56 C, 132 F
56 C, 132 F
56 C, 132 F
56 C, 132 F
56 C, 132 F
56 C, 132 F
56 C, 132 F
56 C, 132 F
56 C, 132 F
56 C, 132 F
56 C, 132 F
56 C, 132 F
56 C, 132 F
56 C, 132 F
56 C, 132 F
56 C, 132 F
56 C, 132 F
56 C, 132 F
56 C, 132 F
6 C, 42 F
91 C, 195 F
51 C, 123 F
6 C, 42 F
76 C, 168 F
121 C, 249 F
72 C, 161 F
72 C, 161 F
72 C, 161 F
72 C, 161 F
```

Figure 14: Serial monitor output of temperature sensor data.

Step 4: Arduino Code

In Fig. 12, only the “Blocks” drop down menu is selected under the “Code” editor. In this step, the student will explore the “Blocks + Text” drop down item. After s/he clicks on the “Blocks + Text” option, the arduino code will be made visible beside the code block, as shown in Fig. 13.

(i) At the beginning of the code, specifically in line 3, 4, and line 7, three variables are declared and initialized with the value 0.

(ii) The first function in the code section, named $setup()$ as shown on line 11 of code snippet, a variable “INPUT” is created to hold the state of the input. To enable the message sending feature, an Arduino requires opening a new communication channel. That is being done in the Serial.begin() in line 12. Input and output pins are configured using the pinMode() function. Arduino Pin A0 is configured as an input for listening to the electrical state of the temperature sensor. On the otherhand, in line 13-15, Pins 2, 3, and 4 are configured as outputs to control the LEDs.

(iii) Different conditions for turning on LEDs and temperature conversion from celsius to farenheit are defined inside the loop() funcion. On line 21, the baselineTemp variable is assigned with 20, which is the threshold value for the first LED to turn on. Based on the program requirements, these values can be changed accordingly. The temperature unit conversion function, Eq. 1, is implemented on line 25. Then the temperature values stored in the variables celsius and farenheit are printed on the serial monitor by appending “C” and “F” respectively.

(iv) Five if conditions categorize the celsius value and depending on the true condition, the digital pins 2,3, and 4 state is assigned. For example, when the celsius value is less than 20^0C, the digital output value for all three pins is LOW. If celsius value is in the range of greater than 20^0C, but less than 30^0C, then pin 2 becomes HIGH causing the connected LED to turn on in line 36. In line 40, the condition (temperature greater than equal to 30^0C and less than 40^0C) for turning on two LEDS connected to pin 2 and 3 is implemented.

$$\frac{C}{5} = \frac{(F-32)}{9} \tag{1}$$

3.2.2 Deliverable

After completing the lab following the above-mentioned instructions, a student needs to submit a document (word/pdf) using designated learning management system (LMS), such as: blackboard, canvas, etc, that will include the following items:

(a) Describe the functionality of a breadboard. Summarize your experience.

(b) Screenshot of the block and code section.

(c) 3 screenshot of output.

A Student can include his/her designed circuit link. To do so, s/he needs to click the button $SendTo$ (located beside start simulation), after that click $Invitepeople$ and finally click the $copylink$ option on the tinkercad.

4 Student Comments

At the end of the semester, a student survey was conducted to get their feedback regarding this Ubiquitous Computing course. In response to the question ”What did you like the best about this course?”, one student answered “I liked the TinkerCAD assignments that demonstrated the functionality of circuits. I think they gave me a good appreciation for some of the fundamentals of embedded systems and IoT.” Some other student responses to this question are as follows. “developing real world work”,“The professor really helped us in anyway she could.”

Next, the students were asked “Did you understand what was expected of you in this course?”. In response to this question, 50% replied Extremely well, 25% Well and 25% reasonably well. As a reply of the question “Were you adequately prepared in the prerequisite course to take this course ?”, 50% response was Extremely Well, 25% replied Well and 25% students responded poorly.

5 Conclusions

In this paper, I presented some laboratories that the students conducted using a remote simulation tool: Tinkercad. The pedagogical approach of remote teaching was briefly described and its effectiveness was assessed through a survey. The end-of-the-semester positive student reviews support the success of the course design. Even though, students could experiment numerous interesting IoT applications in Tinkercad utilizing varied sensors , there are some advanced sensors (e.g., particle sensor) that are still unavailable in tinkercad. The basic understanding of

IoT and sensors that were taught in this course well-prepares students to take advanced courses (e.g., Senior Design) in their future studies. For example, in senior design course when they get some funding for hardware purchase, they can enhance the designs implemented in this course and implement for a senior design.

References

[1] K. Rose, S. Eldridge, and L. Chapin, "The internet of things: An overview. internet society,(october), 53," 2015.

[2] A. Ambardar *et al.*, *Analog and digital signal processing*. PWS BOSTON, MA, 1995.

[3] S. Tasnim, N. Pissinou, and S. Iyengar, "A novel cleaning approach of environmental sensing data streams," in *2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pp. 632–633, IEEE, 2017.

[4] Y. Tang, S. Tasnim, N. Pissinou, S. Iyengar, and A. Shahid, "Reputation-aware data fusion and malicious participant detection in mobile crowdsensing," in *2018 IEEE International Conference on Big Data (Big Data)*, pp. 4820–4828, IEEE, 2018.

[5] S. Tasnim, J. Caldas, N. Pissinou, S. Iyengar, and Z. Ding, "Semantic-aware clustering-based approach of trajectory data stream mining," in *2018 International Conference on Computing, Networking and Communications (ICNC)*, pp. 88–92, IEEE, 2018.

[6] O. S. D. H. J. J. Li, B. Faltings and J. Beutel, "Sensing the air we breathe." http://www.opensense.ethz.ch/trac/, May 2012.

[7] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[8] M. A. Alswailim, H. S. Hassanein, and M. Zulkernine, "A reputation system to evaluate participants for participatory sensing," in *2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 2016.

[9] P. Kulkarni and P. Kute, "Internet of things based system for remote monitoring of weather parameters and applications," *Int. J. Adv. Electron. Comput. Sci*, vol. 3, no. 2, pp. 68–73, 2016.

[10] S. A. A. Elmustafa and E. Y. Mujtaba, "Internet of things in smart environment: Concept, applications, challenges, and future directions," *World Scientific News*, vol. 134, no. 1, pp. 1–51, 2019.

[11] GSMA, "Air quality monitoring using iot and big data." https://www.gsma.com/iot/wp-content/uploads/2018/02/iot_clean_air_02_18.pdf, Accessed January 2023.

[12] S. Tasnim, A. Ferguson, B. Gordon, C. Gordon, K. Ahmed, and I. Mkpong-Ruffin, "A smart environment monitoring application for mobile internet of things," in *Proceedings of the*

27th International Conference on Systems Engineering, ICSEng 2020, pp. 223–233, Springer, 2021.

[13] T. Inaoka, H. Shintaku, T. Nakagawa, S. Kawano, H. Ogita, T. Sakamoto, S. Hamanishi, H. Wada, and J. Ito, "Piezoelectric materials mimic the function of the cochlear sensory epithelium," *Proceedings of the National Academy of Sciences*, vol. 108, no. 45, pp. 18390–18395, 2011.

[14] Autodesk, "Tinkercad." https://www.tinkercad.com/dashboard, Accessed January 2023.

[15] A. H. Kioumars and L. Tang, "Wireless network for health monitoring: heart rate and temperature sensor," in *2011 Fifth International Conference on Sensing Technology*, pp. 362–369, IEEE, 2011.

[16] H. Mansor, M. H. A. Shukor, S. S. Meskam, N. Q. A. M. Rusli, and N. S. Zamery, "Body temperature measurement for remote health monitoring system," in *2013 IEEE International conference on smart instrumentation, measurement and applications (ICSIMA)*, pp. 1–5, IEEE, 2013.

[17] Z. Dong, F. Li, B. Beheshti, A. Mickelson, M. Panero, and N. Anid, "Autonomous real-time water quality sensing as an alternative to conventional monitoring to improve the detection of food, energy, and water indicators," *Journal of Environmental Studies and Sciences*, vol. 6, pp. 200–207, 2016.

[18] L. García, L. Parra, J. M. Jimenez, J. Lloret, and P. Lorenz, "Iot-based smart irrigation systems: An overview on the recent trends on sensors and iot systems for irrigation in precision agriculture," *Sensors*, vol. 20, no. 4, p. 1042, 2020.