Student name: Hoang Long Tran

Student ID: s223128143

SIT225: Data Capture Technologies

Activity 3.1: Arduino IoT Cloud and Dashboard with Arduino Nano 33 IoT devices

Arduino Cloud is your next exciting journey to build, control and monitor your connected projects. You can connect anything to Arduino Cloud including a wide range of compatible Arduino boards such as Arduino Nano 33 IoT or a third-party device that speaks Python. Arduino Cloud is an all-in-one IoT solution that empowers makers to create from anywhere, control their devices with stunning dashboards.

In this activity, you will connect your Arduino board to the cloud as a device, register a thing (in terms of a cloud variable) with your device, create a dashboard with a graphical widget which show value that is sent from the sketch running in your Arduino board.

Hardware Required

Arduino Nano 33 IoT Board

Wi-Fi hotspot (preferably your smartphone hotspot)

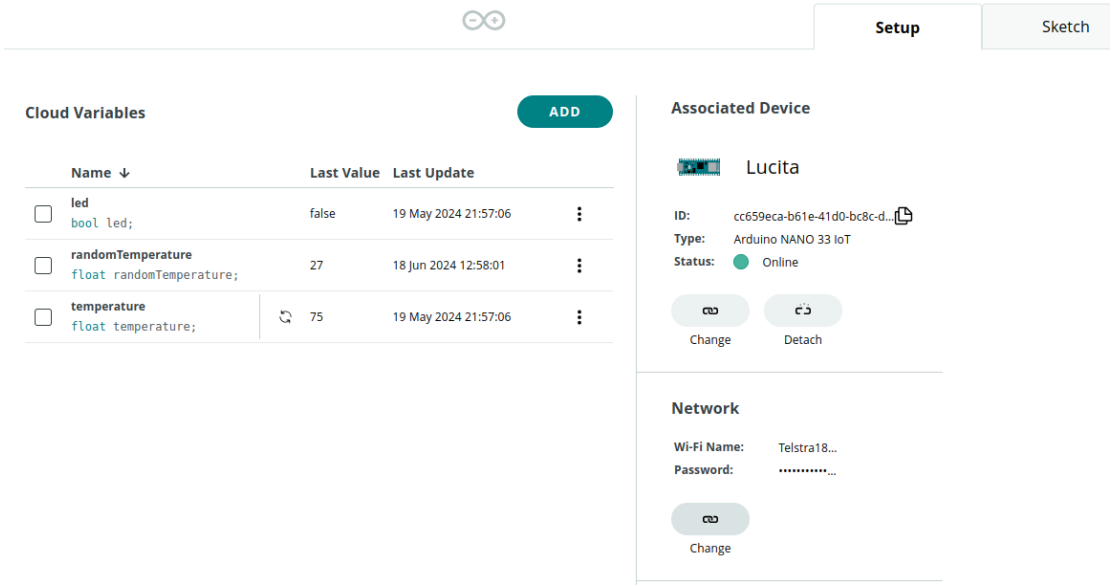
USB cable

Software Required

Arduino programming environment (Arduino IDE)
Arduino IoT Cloud (https://app.arduino.cc)

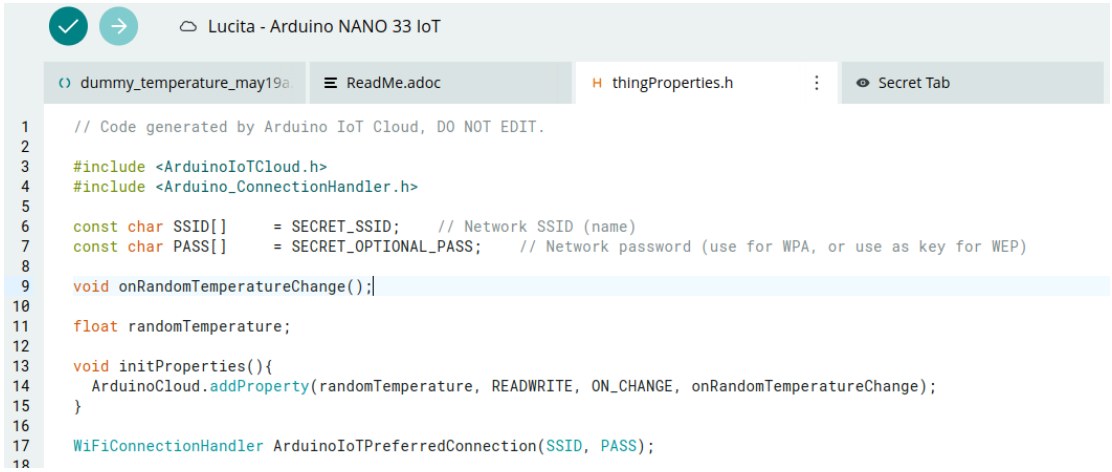
Steps

Step	Action
1	**Account creation:** Create an account in Arduino IoT Cloud (https://app.arduino.cc). Once done, login to your account.

	You can follow Arduino tutorial (https://support.arduino.cc/hc/en-us/articles/360016495559-Add-and-connect-a-device-to-Arduino-Cloud) for detail. This tutorial is referred to in the steps below.
2	**Add the device**: Follow step 1 in the tutorial to add your Arduino Nano 33 IoT device.
3	**Create and configure a Thing**: Follow step 2 in the tutorial to create a Thing (sensor) and attach it to the device created in step 2 above. Note that a Thing is created as a cloud variable of specific data types. In your sketch, there should exist the same variable name and type.  Consider only a single cloud variable ‘randomTemperature’ for this activity and ignore the rest (led or temperature). To keep things simple, the “Sketch” tab in the above screenshot prepares
4	**Configure your Wi-Fi**: In the above screenshot, the right column shows Associated Device and Network sections. You can see “Telstra” network is shown at the time this document was prepared. This should be your smartphone hotspot which is the most convenient considering the mobility – you can carry your laptop and projects without changing in the sketch.
5	**Sketch tab**: There is a Sketch tab at the top right corner of the screenshot above which gives you code ready to deploy in your Arduino board.

There are 3 files in the project. The first tab shows the sketch (.ino) is generally the project name. Two other sketches are header files (.h) - thingProperties.h and arduino_secrets.h.

The thingProperties.h header looks like below.



It shows several items including -

- Your wifi variables (such as ssid and password which you define in arduino_secrets.h header).
- The cloud variable exactly with the same name and type you have created while creating your Thing in step 3 above.
- An initialisation function called *void initProperties() {...}* which registers the Thing variable to the cloud.
- A template function *ArduinoIoTPreferredConnection(SSID, PASS)* which works behind the scenes to connect to the Arduino Cloud using the Wi-Fi you have configured.

The arduino_secrets.h header file contins Wi-Fi ssid and password. The Secret Tab shows fields where you can input Wi-Fi information. If you want to deploy using Cloud IDE shown above (called OTA upload), it needs to upgrade your

account. Instead, you can copy the sketch to your Arduino IDE installed on your computer.

sketch_iot_cloud.ino | arduino_secrets.h | thingProperties.h

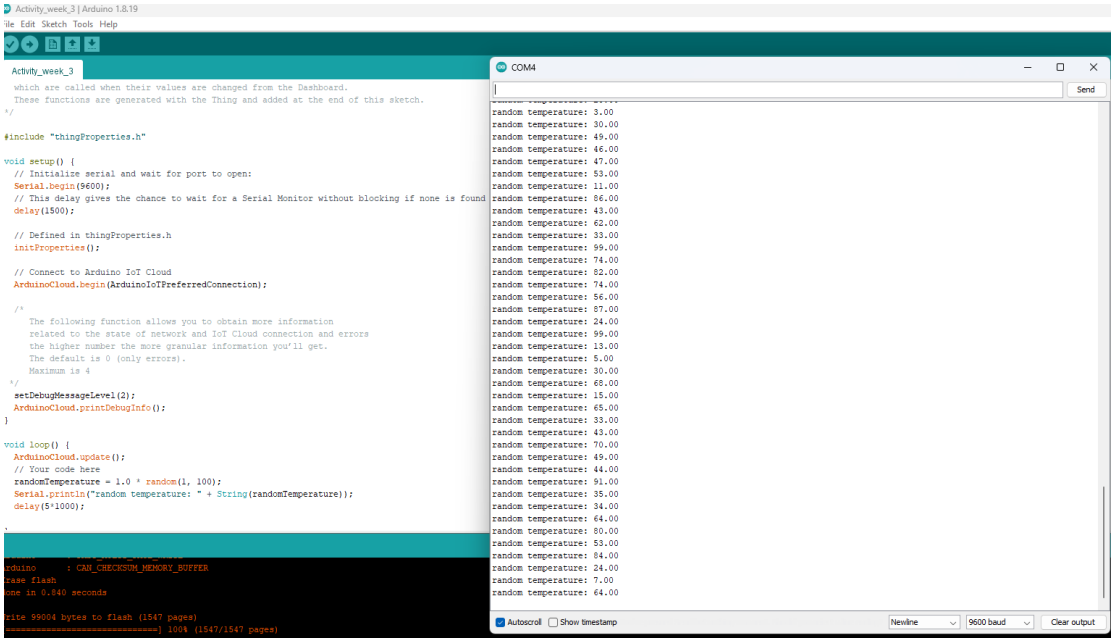
```
1  #define SECRET_SSID "[redacted]";
2  #define SECRET_OPTIONAL_PASS "[redacted]"
```

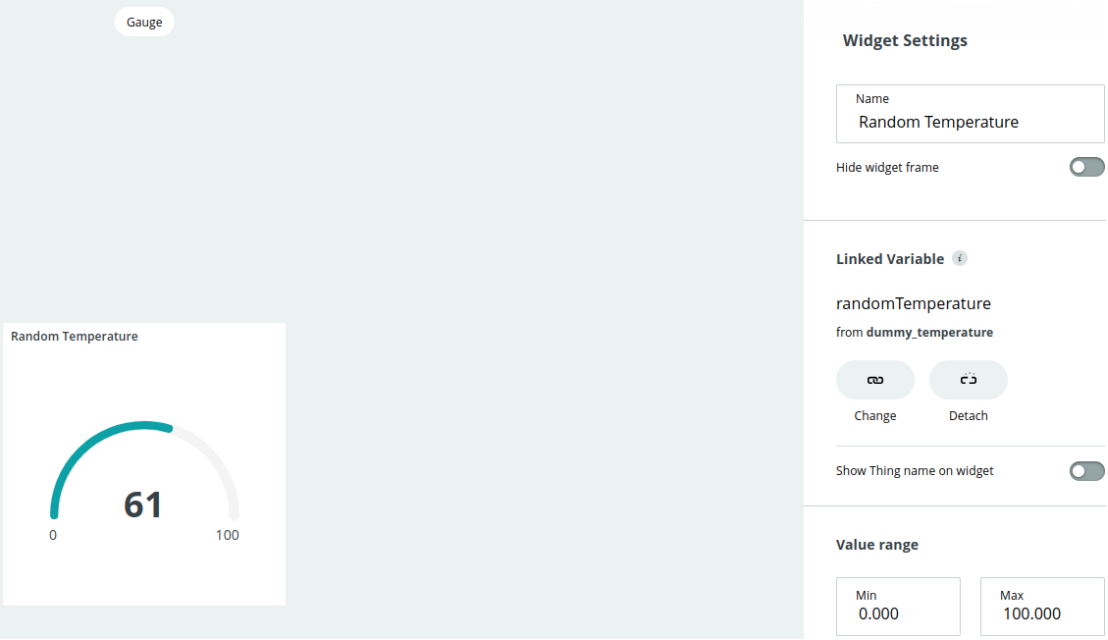
You can prepare the sketch and 2 header files as mentioned. Alternatively, you can download the code from here (https://github.com/deakin-deep-dreamer/sit225/tree/main/week_3).

The sketch in the GitHub link above shows the sketch below (*sketch_iot_cloud.ino*).

sketch_iot_cloud.ino | arduino_secrets.h | thingProperties.h

```
1  #include "thingProperties.h"
2
3  void setup() {
4    // Initialize serial and wait for port to open:
5    Serial.begin(9600);
6    // This delay gives the chance to wait for a Serial Monitor without
7    delay(1500);
8
9    // Defined in thingProperties.h
10   initProperties();
11
12   // Connect to Arduino IoT Cloud
13   ArduinoCloud.begin(ArduinoIoTPreferredConnection);
14
15   /*
16       The following function allows you to obtain more information
17       related to the state of network and IoT Cloud connection and erro
18       the higher number the more granular information you'll get.
19       The default is 0 (only errors).
20       Maximum is 4
21   */
22   setDebugMessageLevel(2);
23   ArduinoCloud.printDebugInfo();
24 }
25
26 void loop() {
27   ArduinoCloud.update();
28   // Your code here
29   randomTemperature = 1.0 * random(1, 100);
30   Serial.println("random temperature: " + String(randomTemperature));
31   delay(5*1000);
32 }
33
34 /*
35   Since Temperature is READ_WRITE variable, onRandomTemperatureChange(
36   executed every time a new value is received from IoT Cloud.
37 */
38 void onRandomTemperatureChange()  {
39   // Add your code here to act upon Temperature change
40   Serial.println("--onRandomTemperatureChange");
41 }
```

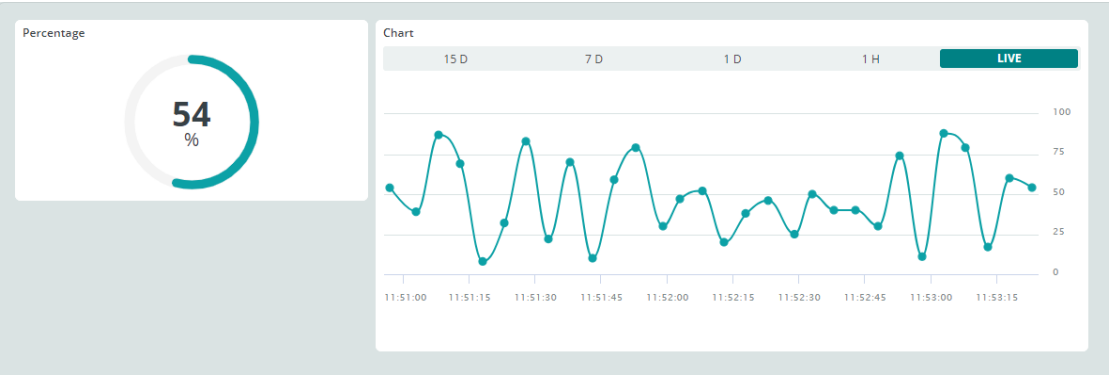
	The loop function generates a random value between 1 and 100 and assigns to *randomTemperature* variable which is a cloud variable and write in serial port and waits for 5 seconds.
6	**Deploy code to board**: You can upload code to Arduino board from Arduino IDE. You can observe the random temperature readings in the serial monitor. Question: Screenshot the output of the serial monitor with random temperature values along with the initial Wi-Fi connection codes. Comment on the output lines. Answer:  The output shows that the sketch is successfully generating and outputting random temperature values at 5 seconds intervals.
7	**Create a dashboard**: A dashboard in Arduino Cloud can be created to visualse the sensor readings sent by the Things connected to your Arduino Nano 33 IoT board. A list of Dashboard widgets is described in this tutorial (https://docs.arduino.cc/arduino-cloud/cloud-interface/dashboard-widgets). You can create a new dashboard from the Dashboards left menu items (https://app.arduino.cc/dashboards) where there are other menu items such as Devices and Things you have seen earlier. Creating a dashboard is simply choosing a dashboard widget, such as a Guage and link it to the Thing cloud variable you have created.



Once you click the Done button, you should look the Gauge widget is updating based on the value shown in the Arduino IDE serial monitor.

Question: Screenshot the output of the serial monitor with random temperature values and the dashboard output so the Gauge value can be found in the serial monitor output.

Answer:



8

Question: If you recall, there is a function *initProperties()* in *thingProperties.h* file where there is a single line -

```
13  void initProperties(){
14    ArduinoCloud.addProperty(randomTemperature, READWRITE, ON_CHANGE, onRandomTemperatureChange);
15  }
```

	A function *onRandomTemperatureChange()* exists (in sketch_iot_cloud.ino file) to respond to ON_CHANGE event. Can you explore what is the use of this function and when the ON_CHANGE event will trigger? Answer: The `initProperties` syncs with Arduino Cloud, where `randomTemperature` in our case is the variable will be synced, `READWRITE` is just permissions, `ON_CHANGE` that is called by the `onRandomTemperatureChange` function whenever value of `randomTemperature` changes. The application of this is to update cloud dashboards.

Activity 3.2: Arduino IoT Cloud with custom Python devices

You can connect anything to Arduino Cloud including a wide range of compatible Arduino boards such as Arduino Nano 33 IoT or a third-party device that **speaks Python**.

In this activity, you will connect a custom Python board to Arduino IoT Cloud and synchronise to another cloud variable, the one you created earlier, *randomTemperature*. This will enable you to receive data from Arduino board in your Python script.

Hardware Required

Arduino Nano 33 IoT Board

Wi-Fi hotspot (preferably your smartphone hotspot)

USB cable

Software Required

Arduino programming environment (Arduino IDE)
Arduino IoT Cloud (https://app.arduino.cc)
Python 3

Steps

Step	Action
1	**Thing & Device Configuration**: Following the tutorial (https://docs.arduino.cc/arduino-cloud/guides/python), a. Create a new Thing, by clicking on the "Create Thing" button. b. Click on the "Select Device" in the "Associated Devices" section of your Thing. c. Click on "Set Up New Device" and select the bottom category ("Manual Device"). Click continue in the next window and choose a name for your device. d. Finally, you will see a new Device ID and a Secret Key generate. You can download them as a PDF. Make sure to save it as you cannot access your Secret Key again.
2	**Create Variables**: Follow the same tutorial mentioned above and do the following steps:

	a. While in Thing configuration, click on "Add Variable" which will open a new window. b. Name your variable ***temperature*** and select it to be of a float type. c. Just below the variable name, there is a link '*Sync with other Things*'. Click the link and it will show a list of all the variables you have created so far in your Arduino Cloud account in all the devices. d. Select '***randomTemperature***' variable from Arduino board and click the button '**Synchronise Variables**'. e. At this point the ***randomTemperature*** data sent from your Arduino Nano board will arrive in Arduino Cloud, then it will be written to Python device variable ***temperature***. This value assignment triggers on_write event in the listening client and corresponding callback function is called.
3	**Create Python script**: Now you need to create a Python script to register to Arduino Cloud, register for a cloud variable called temperature and write a callback function for on_write event handling. You can write the code as below or download the code from here (https://github.com/deakin-deep-dreamer/sit225/blob/main/week_3/arduino_variable_sync.py).

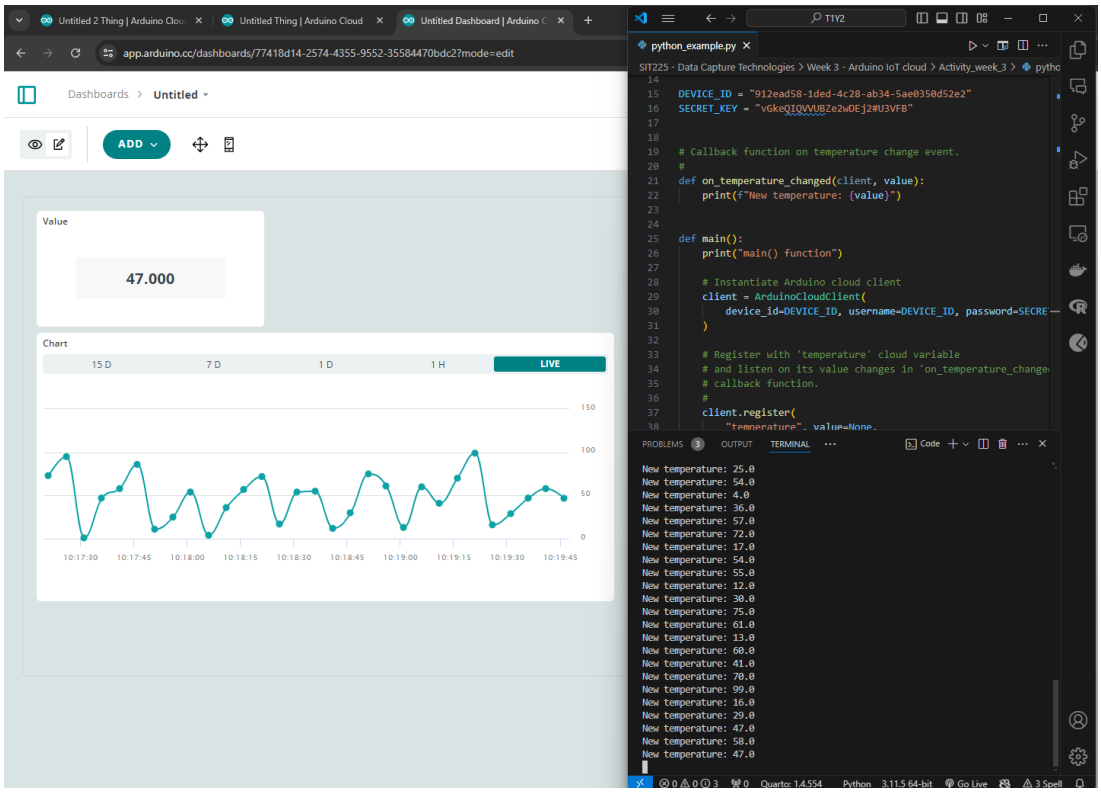
```python
"""
    Requirement: arduino_iot_cloud
    Install: pip install arduino-iot-cloud

    @Ahsan Habib
    School of IT, Deakin University, Australia.
"""

import sys
import traceback
import random
from arduino_iot_cloud import ArduinoCloudClient
import asyncio

DEVICE_ID = "bc5c0fe9-e6ef-4eb0-90de-05032ffd9a83"
SECRET_KEY = "3oJYfrkmSNjM4YwKGJgVObbBn"


# Callback function on temperature change event.
#
def on_temperature_changed(client, value):
    print(f"New temperature: {value}")


def main():
    print("main() function")

    # Instantiate Arduino cloud client
    client = ArduinoCloudClient(
        device_id=DEVICE_ID, username=DEVICE_ID, password=SECRET_KEY
    )

    # Register with 'temperature' cloud variable
    # and listen on its value changes in 'on_temperature_changed'
    # callback function.
    #
    client.register(
        "temperature", value=None,
        on_write=on_temperature_changed)

    # start cloud client
    client.start()


if __name__ == "__main__":
    try:
        main()  # main function which runs in an internal infinite loop
    except:
        exc_type, exc_value, exc_traceback = sys.exc_info()
        traceback.print_tb(exc_type, file=print)
```

You should replace DEVICE_ID and SECRET_KEY as you were given in step 1-d above.

Question: Study the code and describe in your word how the statements match to the purpose mentioned in this step-3 above?

Answer: I will explain from the main function. In this function, we instantiate the Arduino cloud client, and register it with the `temperature` cloud variable, with the changes based on the `on_temperature_changed` function where it prints the value it gets from the Arduino.

4	**Run Python script**:

	Run the python script from command line below - $ python arduino_variable_sync.py At this point, the command line output should show connecting to Arduino cloud and print new temperature values periodically. Question: Screenshot the Arduino Cloud Dashboard gauge showing the Arduino side randomTemperature value and screenshot the Python command-line output showing similar values. Note that there might be some lag due to network connectivity. Explain your answer accordingly. Answer: 
5	Question: Research how you can download data from Arduino IoT Cloud, say the *randomTemperature* variable if you continue the setup running for 10 minutes or so? Answer: Just hit the download button on dashboard and the Arduino service will send you an email.
6	Question: Discuss how you can save the data while you are receiving in Python script in a file? You can discuss in a group and come up with a solution.

Hint: An algorithm of appending data can be as below. Write Python code for the algorithm. You can put the code in the call-back function *on_temperature_changed()*.

Algorithm: Append timestamp and data value to a file:

a. Open a file in append mode
b. Create a CSV string <timestamp>, <value> <NEWLINE>
c. Call write function and pass CSV string, otherwise, call write_line function and pass the CSV string removing the ending <NEWLINE> (otherwise, a blank new line will be written to file).
d. Call flash to push data to be written to file immediately.
e. Close the file.

You can modify the algorithm to make it efficient, such as instead of opening/closing the file every time, move them to the beginning and end of the execution and keeping the CSV string formation and writing and flashing to file inside the callback function.

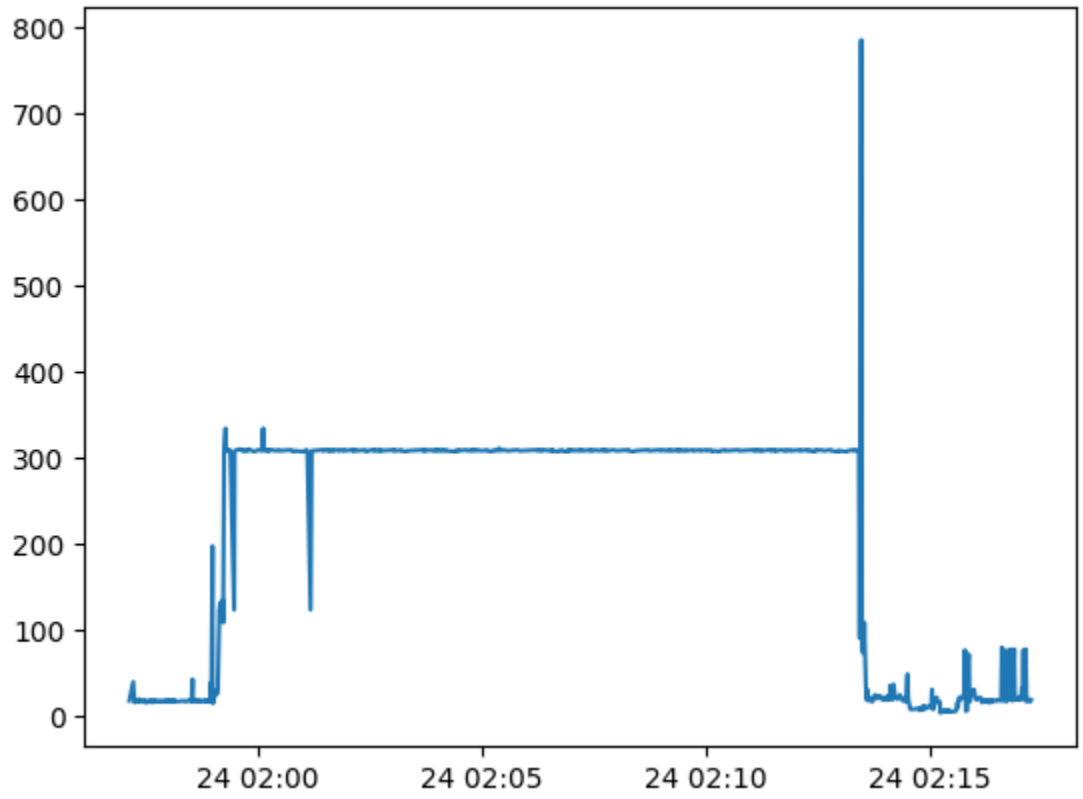
Answer:

```
# Direct path to save the csv
filename =
os.path.join(r'C:\Users\tomde\OneDrive\Documents\Deakin\Deakin-Data-
Science\T1Y2\SIT225 - Data Capture Technologies\Week 3 - Arduino IoT
cloud\Activity_week_3', 'random_temp.csv')

# Function to get the current time
def timestamp():
    return datetime.now().strftime('%Y%m%d%H%M%S')

# Callback function on temperature change event.
def on_temperature_changed(client, value):
    # String to save data
    formatted_data = f"{timestamp()}, {value}"

    # Add to csv file
    try:
        with open(filename, 'a') as file:
            file.write(formatted_data + '\n')
            file.flush()
        print(f"{formatted_data}") # print to terminal
    except Exception as e:
        print(f"Error in writing to csv: {e}")
```

	First, I created a direct path to save the csv. Then, I created a function to save the exact time that python receives the data. In the callback function, I have modified it to write the new data and flush it. I also printed the string that has been written in the csv file in the terminal.

Weekly activity

Q2:



This graph depicts the HC-SR04 Ultrasonic sensor sensing the distance of the object it was detecting. I put the sensor facing where I sit, so when I sit down on the chair, the distance is closer and when I don't the distance is further away. From the graph, in the initial period, I was sitting on my chair, so the distance measured is less than 100 cm, but when I went up and go elsewhere, the distance measured is constant around 300 cm. When I came back to sit on my chair to work, we could see that the distance measured went back down to around 30 40 cm.

Q3:

https://www.youtube.com/watch?v=98gky3LnSPs

Q4:

https://github.com/tomadonna1/SIT225_2024T2/tree/main/Pass%20Task%20Working%20with%20sensors