Student name: Yasmin Pokia

Student ID: s222245206

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)

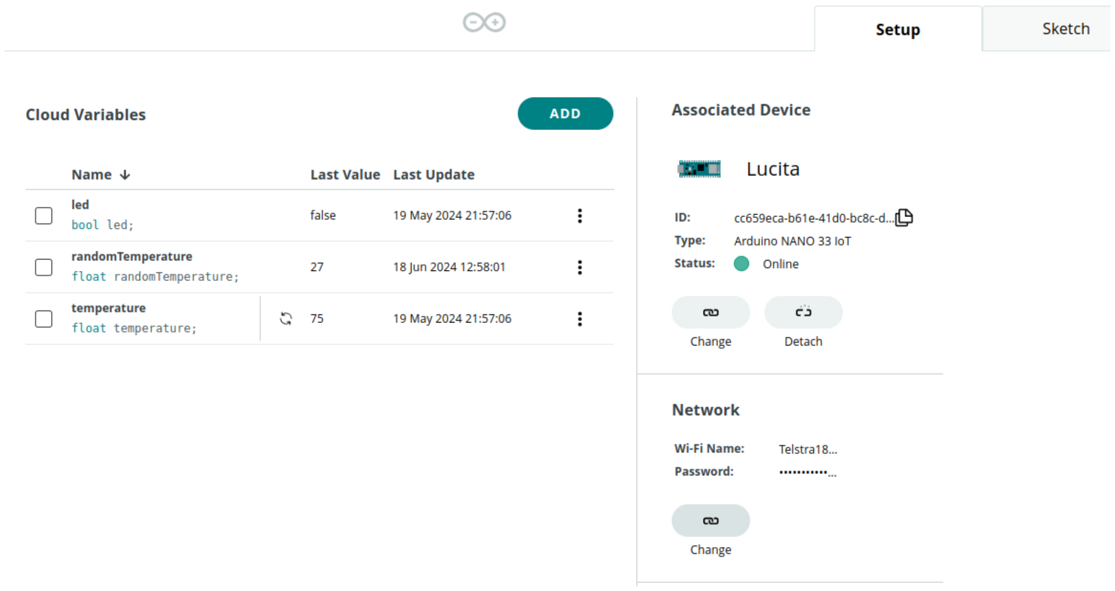
USB cable

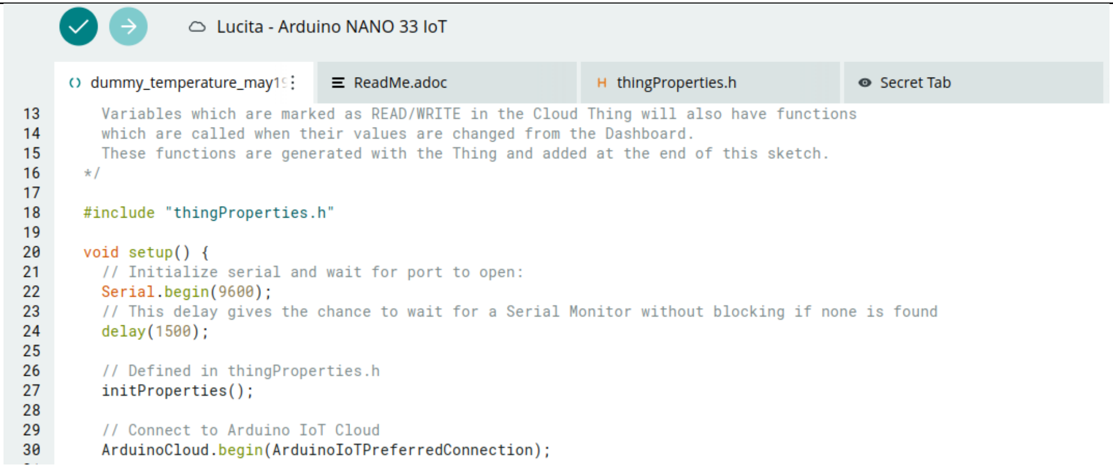
Software Required

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.

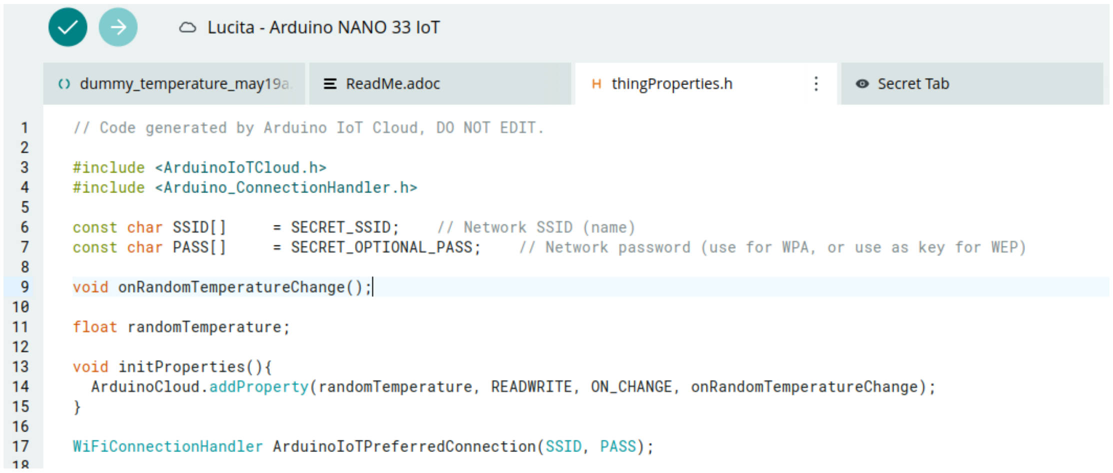
	You can follow Arduino tutorial (https://support.arduino.cc/hc/en-us/articles/360016495559-Add-and-connect-a-device-to-Arduino-Cloud) for detail. This tutorial is referred to in the steps below.
2	**Add the device**: Follow step 1 in the tutorial to add your Arduino Nano 33 IoT device.
3	**Create and 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.  Setup · Sketch Cloud Variables · ADD Name · Last Value · Last Update led `bool led;` · false · 19 May 2024 21:57:06 randomTemperature `float randomTemperature;` · 27 · 18 Jun 2024 12:58:01 temperature `float temperature;` · 75 · 19 May 2024 21:57:06 Associated Device Lucita ID: cc659eca-b61e-41d0-bc8c-d... Type: Arduino NANO 33 IoT Status: Online Change · Detach Network Wi-Fi Name: Telstra18... Password: .... Change 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.



```
  Variables which are marked as READ/WRITE in the Cloud Thing will also have functions
  which are called when their values are changed from the Dashboard.
  These functions are generated with the Thing and added at the end of this sketch.
*/

#include "thingProperties.h"

void setup() {
  // Initialize serial and wait for port to open:
  Serial.begin(9600);
  // This delay gives the chance to wait for a Serial Monitor without blocking if none is found
  delay(1500);

  // Defined in thingProperties.h
  initProperties();

  // Connect to Arduino IoT Cloud
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);
```

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.

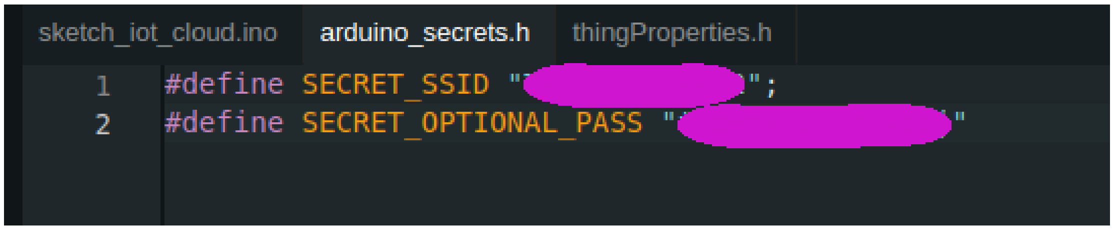


```
// Code generated by Arduino IoT Cloud, DO NOT EDIT.

#include <ArduinoIoTCloud.h>
#include <Arduino_ConnectionHandler.h>

const char SSID[]     = SECRET_SSID;    // Network SSID (name)
const char PASS[]     = SECRET_OPTIONAL_PASS;    // Network password (use for WPA, or use as key for WEP)

void onRandomTemperatureChange();

float randomTemperature;

void initProperties(){
  ArduinoCloud.addProperty(randomTemperature, READWRITE, ON_CHANGE, onRandomTemperatureChange);
}

WiFiConnectionHandler ArduinoIoTPreferredConnection(SSID, PASS);
```

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.

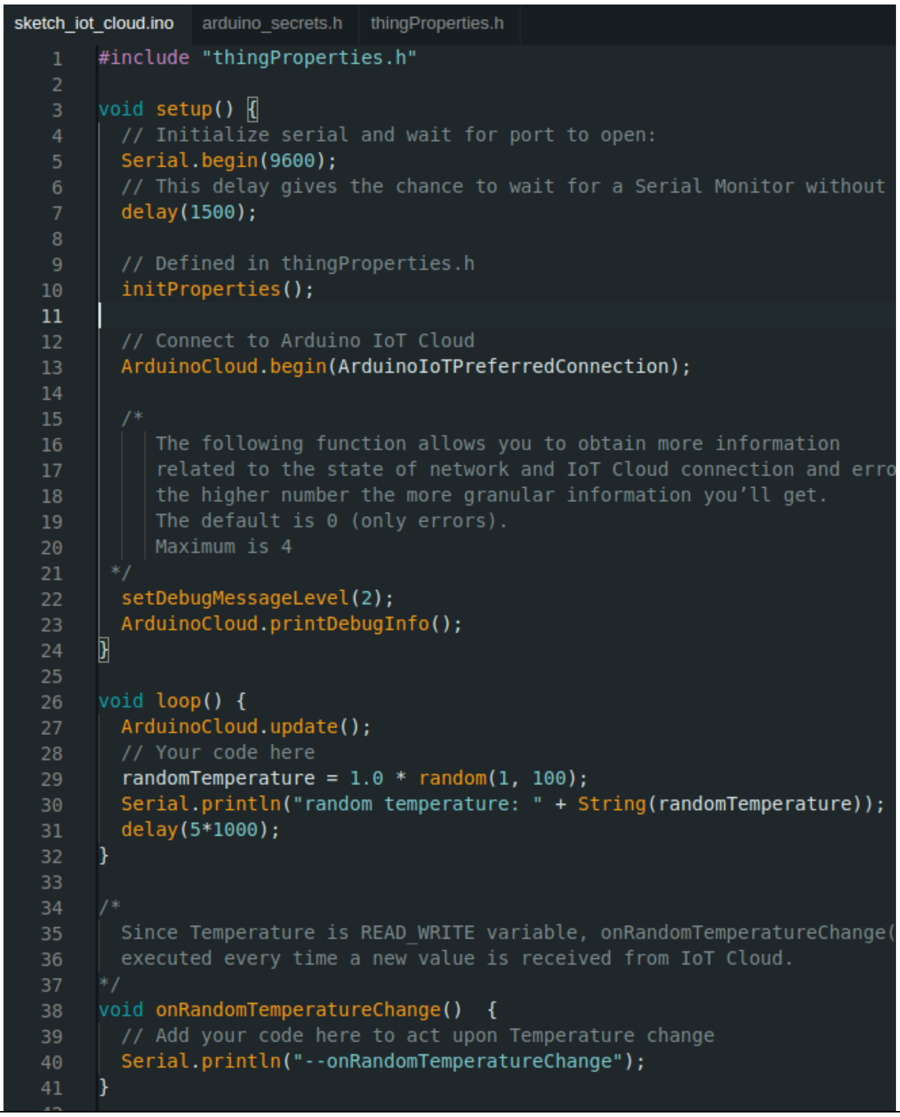
The arduino_secrets.h header file contins Wi-Fi ssid and password. The Secret Tab shows fields where you can input Wi-Fi information. If you want to deploy using Cloud IDE shown above (called OTA upload), it needs to upgrade your account. Instead, you can copy the sketch to your Arduino IDE installed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



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

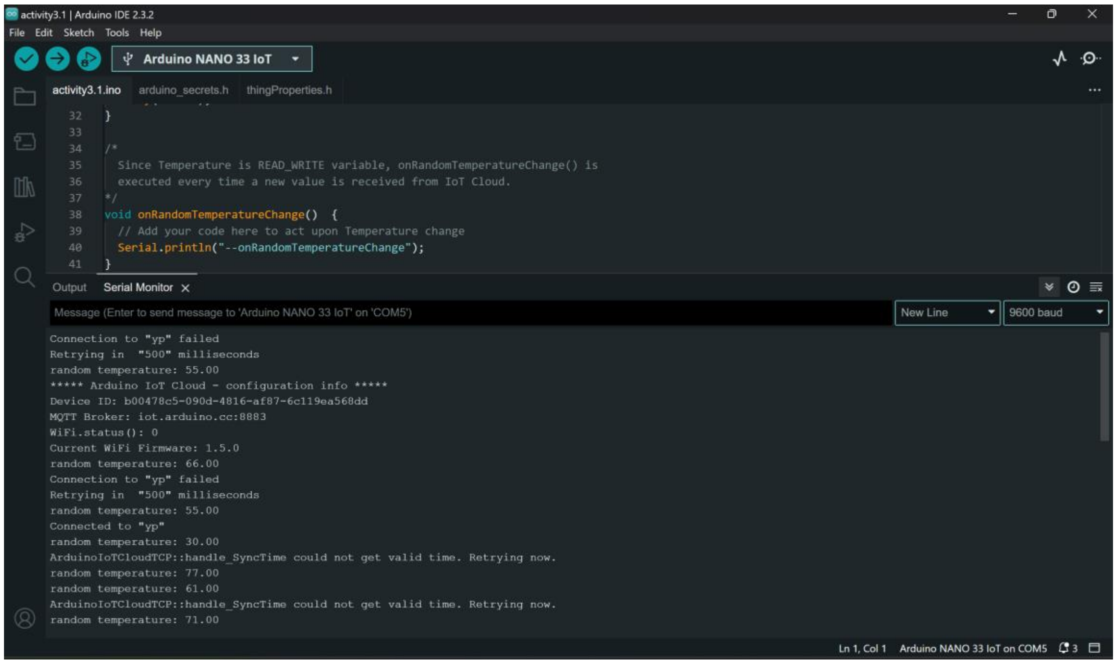
You can prepare the sketch and 2 header files as mentioned. Alternatively, you can download the code from here (https://github.com/deakin-deep-dreamer/sit225/tree/main/week_3).

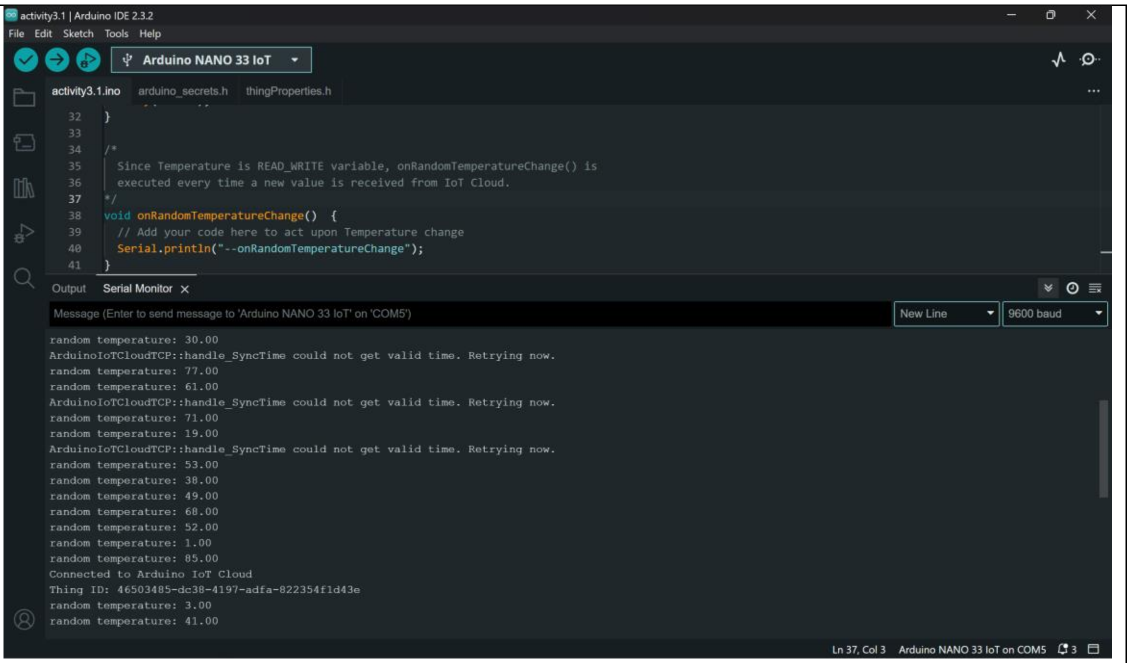
The sketch in the GitHub link above shows the sketch below (*sketch_iot_cloud.ino*).



```
#include "thingProperties.h"

void setup() {
  // Initialize serial and wait for port to open:
  Serial.begin(9600);
  // This delay gives the chance to wait for a Serial Monitor without
  delay(1500);

  // Defined in thingProperties.h
  initProperties();

  // Connect to Arduino IoT Cloud
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);

  /*
     The following function allows you to obtain more information
     related to the state of network and IoT Cloud connection and erro
     the higher number the more granular information you'll get.
     The default is 0 (only errors).
     Maximum is 4
  */
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();
}

void loop() {
  ArduinoCloud.update();
  // Your code here
  randomTemperature = 1.0 * random(1, 100);
  Serial.println("random temperature: " + String(randomTemperature));
  delay(5*1000);
}

/*
  Since Temperature is READ_WRITE variable, onRandomTemperatureChange(
  executed every time a new value is received from IoT Cloud.
*/
void onRandomTemperatureChange()  {
  // Add your code here to act upon Temperature change
  Serial.println("--onRandomTemperatureChange");
}
```

	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:
	
	`}`
	
	`/*`
	`  Since Temperature is READ_WRITE variable, onRandomTemperatureChange() is`
	`  executed every time a new value is received from IoT Cloud.`
	`*/`
	`void onRandomTemperatureChange()  {`
	`  // Add your code here to act upon Temperature change`
	`  Serial.println("--onRandomTemperatureChange");`
	`}`
	
	Message (Enter to send message to 'Arduino NANO 33 IoT' on 'COM5')
	New Line
	9600 baud
	
	`Connection to "yp" failed`
	`Retrying in  "500" milliseconds`
	`random temperature: 55.00`
	`***** Arduino IoT Cloud - configuration info *****`
	`Device ID: b00478c5-090d-4816-af87-6c119ea568dd`
	`MQTT Broker: iot.arduino.cc:8883`
	`WiFi.status(): 0`
	`Current WiFi Firmware: 1.5.0`
	`random temperature: 66.00`
	`Connection to "yp" failed`
	`Retrying in  "500" milliseconds`
	`random temperature: 55.00`
	`Connected to "yp"`
	`random temperature: 30.00`
	`ArduinoIoTCloudTCP::handle_SyncTime could not get valid time. Retrying now.`
	`random temperature: 77.00`
	`random temperature: 61.00`
	`ArduinoIoTCloudTCP::handle_SyncTime could not get valid time. Retrying now.`
	`random temperature: 71.00`
	
	Ln 1, Col 1 Arduino NANO 33 IoT on COM5



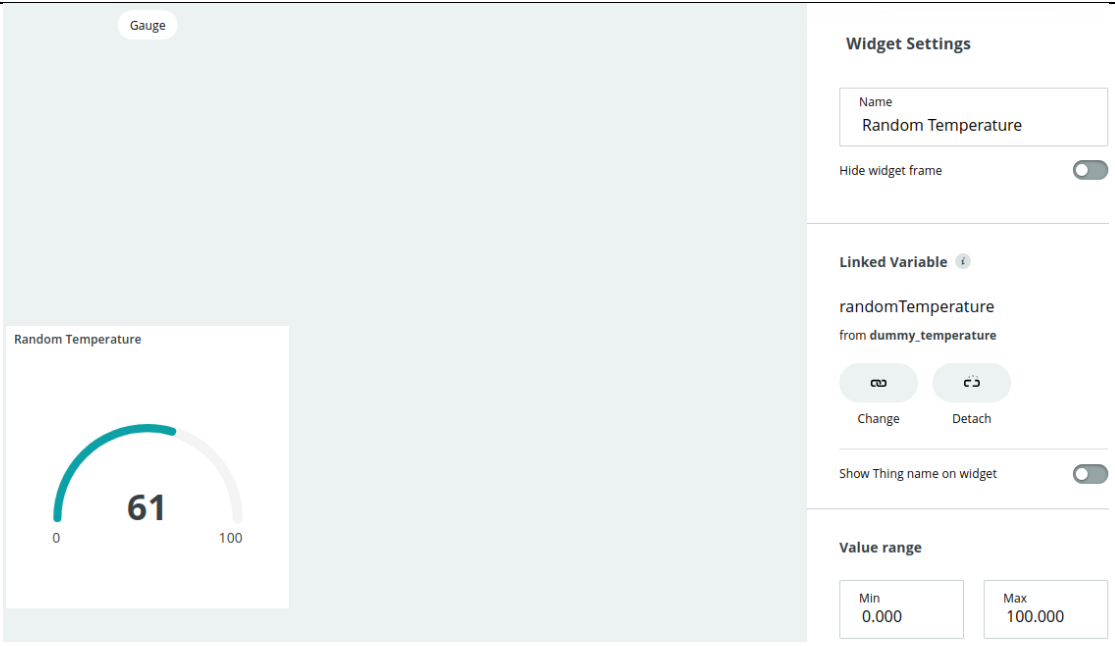
```
}

/*
  Since Temperature is READ_WRITE variable, onRandomTemperatureChange() is
  executed every time a new value is received from IoT Cloud.
*/
void onRandomTemperatureChange()  {
  // Add your code here to act upon Temperature change
  Serial.println("--onRandomTemperatureChange");
}
```

Message (Enter to send message to 'Arduino NANO 33 IoT' on 'COM5')

```
random temperature: 30.00
ArduinoIoTCloudTCP::handle_SyncTime could not get valid time. Retrying now.
random temperature: 77.00
random temperature: 61.00
ArduinoIoTCloudTCP::handle_SyncTime could not get valid time. Retrying now.
random temperature: 71.00
random temperature: 19.00
ArduinoIoTCloudTCP::handle_SyncTime could not get valid time. Retrying now.
random temperature: 53.00
random temperature: 38.00
random temperature: 49.00
random temperature: 68.00
random temperature: 52.00
random temperature: 1.00
random temperature: 85.00
Connected to Arduino IoT Cloud
Thing ID: 46503485-dc38-4197-adfa-822354f1d43e
random temperature: 3.00
random temperature: 41.00
```

ArduinoIoTCloudTCP::handle_SyncTime could not get valid time. Retrying now.
This line indicates that the Arduino IoT Cloud is attempting to synchronize the time but is facing issues. This may be due to network delays or connection problems.
random temperature: 19.00
The Arduino board is generating a random temperature reading. This value is printed to the Serial Monitor.
Connected to Arduino IoT Cloud
This line confirms that the Arduino board has successfully connected to the Arduino IoT Cloud.
Thing ID: 46503485-dc38-4197-adfa-822354f1d43e
This line provides the unique identifier for the "thing" (device) in the Arduino IoT Cloud. This ID is used to manage and interact with the device from the cloud.

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.



Name: Random Temperature

Hide widget frame

Linked Variable

randomTemperature
from **dummy_temperature**

Show Thing name on widget

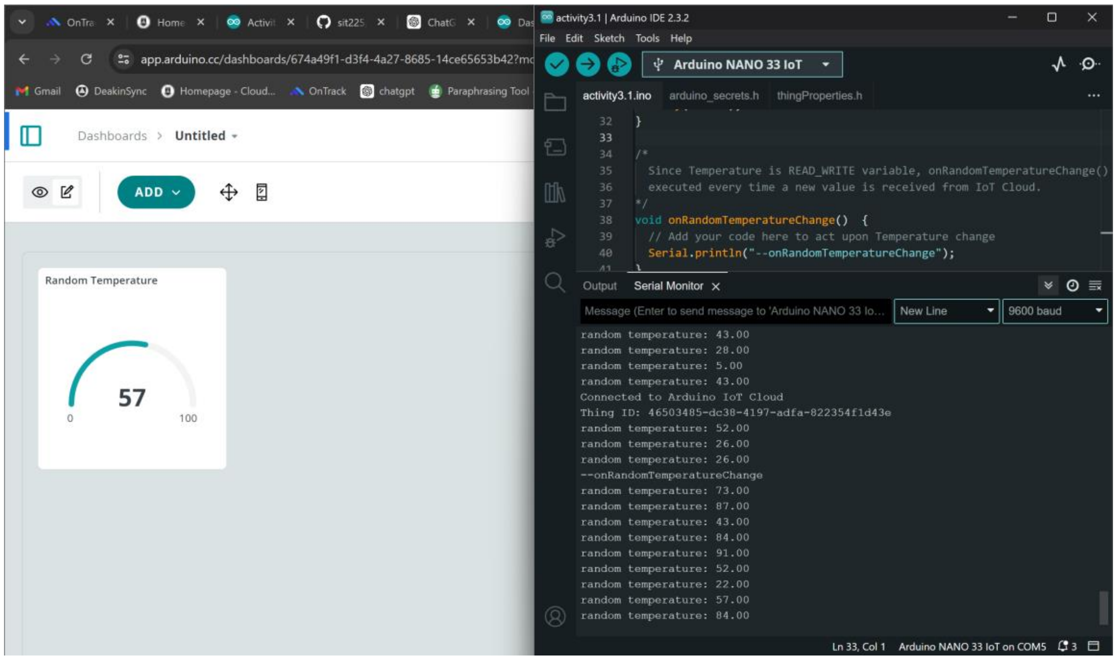
Value range

Min: 0.000
Max: 100.000

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.

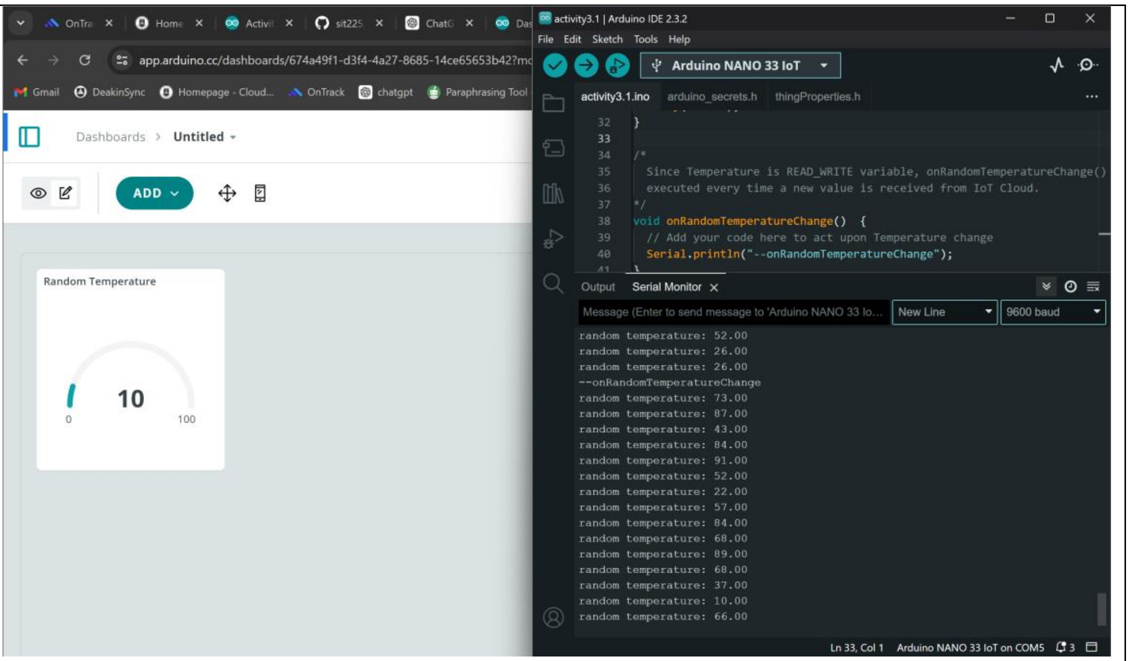
Question: Screenshot the output of the serial monitor with random temperature values and the dashboard output so the Gauge value can be found in the serial monitor output.

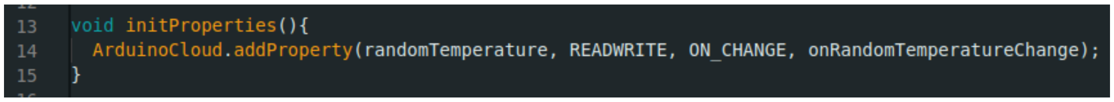
Answer:



```
}

/*
  Since Temperature is READ_WRITE variable, onRandomTemperatureChange()
  executed every time a new value is received from IoT Cloud.
*/
void onRandomTemperatureChange()  {
  // Add your code here to act upon Temperature change
  Serial.println("--onRandomTemperatureChange");
```

```
random temperature: 43.00
random temperature: 28.00
random temperature: 5.00
random temperature: 43.00
Connected to Arduino IoT Cloud
Thing ID: 46503485-dc38-4197-adfa-822354f1d43e
random temperature: 52.00
random temperature: 26.00
random temperature: 26.00
--onRandomTemperatureChange
random temperature: 73.00
random temperature: 87.00
random temperature: 43.00
random temperature: 84.00
random temperature: 91.00
random temperature: 52.00
random temperature: 22.00
random temperature: 57.00
random temperature: 84.00
```

	
	`/*`
	`  Since Temperature is READ_WRITE variable, onRandomTemperatureChange()`
	`  executed every time a new value is received from IoT Cloud.`
	`*/`
	`void onRandomTemperatureChange()  {`
	`  // Add your code here to act upon Temperature change`
	`  Serial.println("--onRandomTemperatureChange");`
	`random temperature: 52.00`
	`random temperature: 26.00`
	`random temperature: 26.00`
	`--onRandomTemperatureChange`
	`random temperature: 73.00`
	`random temperature: 87.00`
	`random temperature: 43.00`
	`random temperature: 84.00`
	`random temperature: 91.00`
	`random temperature: 52.00`
	`random temperature: 22.00`
	`random temperature: 57.00`
	`random temperature: 84.00`
	`random temperature: 68.00`
	`random temperature: 89.00`
	`random temperature: 68.00`
	`random temperature: 37.00`
	`random temperature: 10.00`
	`random temperature: 66.00`
	As shown above with 2 examples of screenshots.

8	Question: If you recall, there is a function *initProperties()* in *thingProperties.h* file where there is a single line -
	
	`void initProperties(){`
	`  ArduinoCloud.addProperty(randomTemperature, READWRITE, ON_CHANGE, onRandomTemperatureChange);`
	`}`
	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 use of the function ‘onRandomTemperatureChange()’, is to respond to changes in the ‘randomTemperature’ property. When the ‘randomTemperature’ value changes, the Arduino IoT Cloud triggers the ‘ON_CHANGE’ event, and this function is called. The ‘ON_CHANGE’ event will trigger and call the ‘onRandomTemperatureChange()’ function when the ‘randomTemperature’ property is updated from the Arduino IoT Cloud dashboard. The ‘ON_CHANGE’ event will also trigger if the ‘randomTemperature’ value is updated locally in the Arduino sketch and the change is synchronized with the cloud.

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

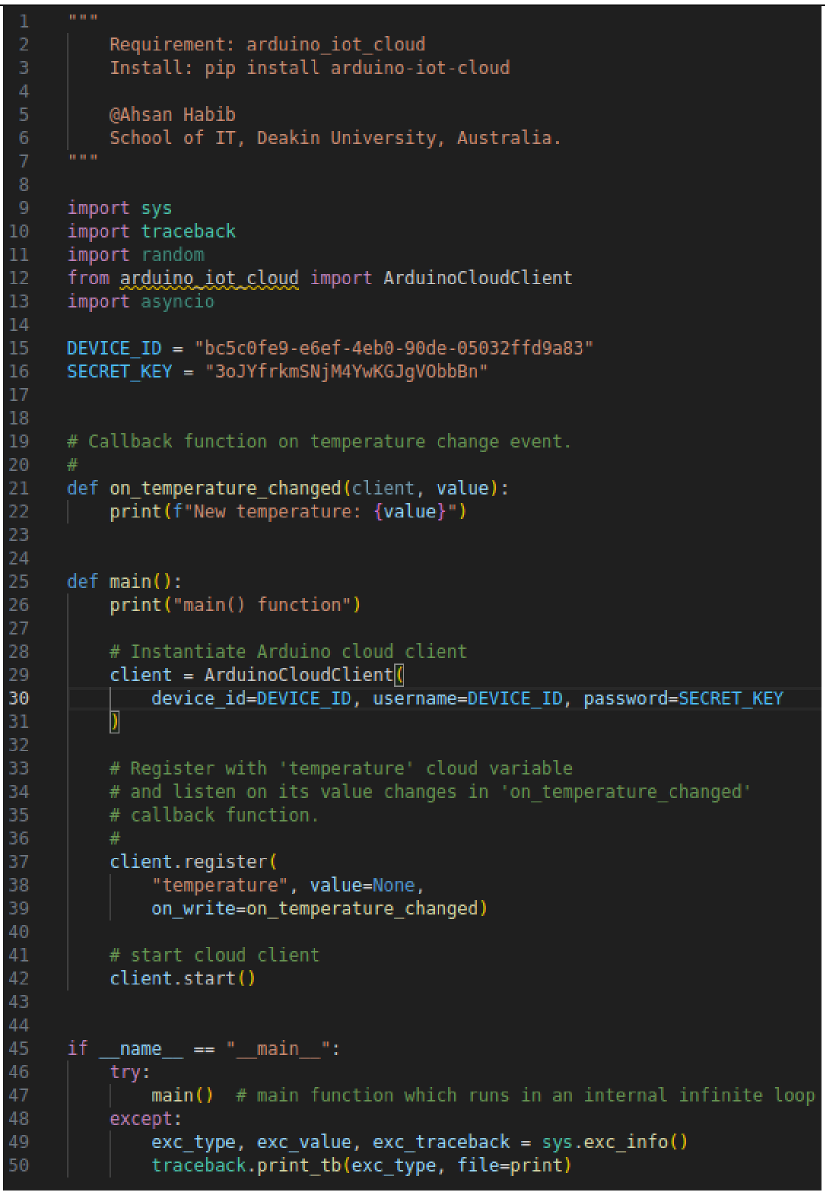
Software Required

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. Device ID: a1143159-86be-43cf-b604-d397ead8b800 Secret key: Msg?JldOtN#2VUoWj4RJsdefM
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)
```

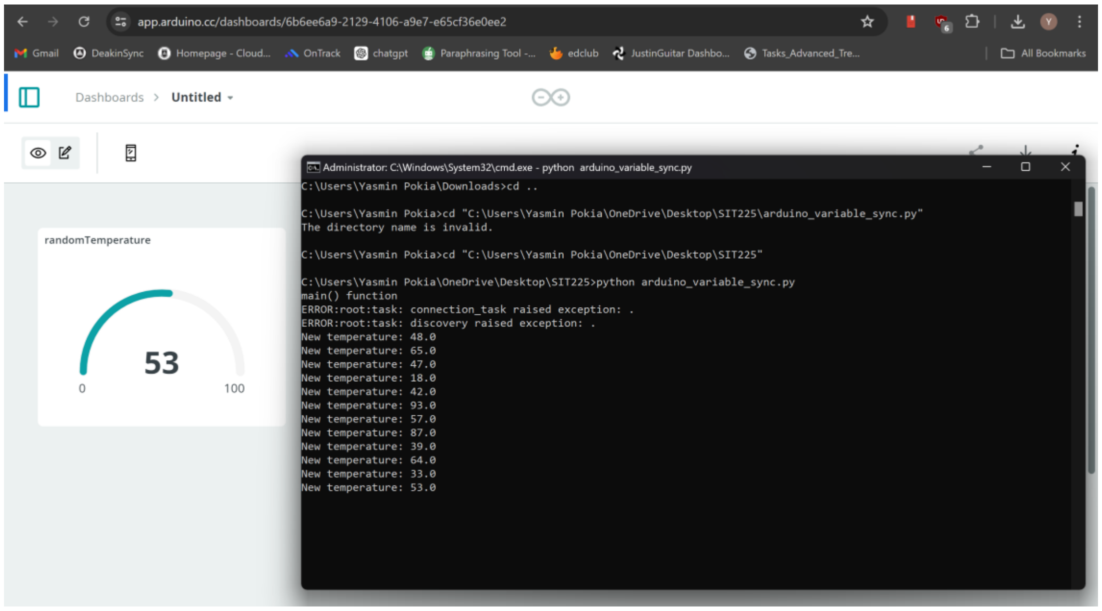
You should replace DEVICE_ID and SECRET_KEY as you were given in step 1-d above.

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

Answer: The Python script that is provided is intended to establish a connection with the Arduino IoT Cloud, register a cloud variable named temperature, and utilise a callback function to manage modifications to this value. Step 3 is to handle the on_write event and register for a cloud variable. Here is an explanation of how each section of the code serves this purpose:

	Callback Function: To manage variations in the temperature variable, the on_temperature_changed function is defined. It satisfies the need to manage changes to the cloud variable by printing the updated temperature value to the console each time the on_write event is triggered. In the main function: Instantiate Client: To connect to the Arduino IoT Cloud, the ArduinoCloudClient is instantiated using the DEVICE_ID and SECRET_KEY supplied. Register Variable: The temperature cloud variable is registered via the client.register method. Every time the temperature value changes (an on_write event), the callback method designated to be called is on_temperature_changed. Start Client: The cloud client is started via the client.start method, which also allows temperature variable monitoring and calls the callback code when it varies.
4	**Run Python script**: Run the python script from command line below - `$ python arduino_variable_sync.py` At this point, the commxand 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:


app.arduino.cc/dashboards/6b6ee6a9-2129-4106-a9e7-e65cf36e0ee2
Dashboards > Untitled
randomTemperature
53
0
100
Administrator: C:\Windows\System32\cmd.exe - python arduino_variable_sync.py

```
C:\Users\Yasmin Pokia\Downloads>cd ..

C:\Users\Yasmin Pokia>cd "C:\Users\Yasmin Pokia\OneDrive\Desktop\SIT225\arduino_variable_sync.py"
The directory name is invalid.

C:\Users\Yasmin Pokia>cd "C:\Users\Yasmin Pokia\OneDrive\Desktop\SIT225"

C:\Users\Yasmin Pokia\OneDrive\Desktop\SIT225>python arduino_variable_sync.py
main() function
ERROR:root:task: connection_task raised exception: .
ERROR:root:task: discovery raised exception: .
New temperature: 48.0
New temperature: 65.0
New temperature: 47.0
New temperature: 18.0
New temperature: 42.0
New temperature: 93.0
New temperature: 57.0
New temperature: 87.0
New temperature: 39.0
New temperature: 64.0
New temperature: 33.0
New temperature: 53.0
```

The gauge on the Arduino Cloud Dashboard shows the same values to the ones printed by the Python script. The Python script's command-line output shows that it is successfully connecting to the Arduino IoT Cloud and receiving new temperature values. Each "New temperature: X.X" line corresponds to a temperature reading received from the cloud, triggered by the 'on_write' event in the Python script. As shown, the gauge displays 53, and the Python script has shown a new temperature 53.00. The above screenshot shows the synchronized values from the Arduino board, Arduino Cloud, and the Python script, demonstrating the successful data flow and handling of the 'on_write' event.

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: You can use the Arduino IoT Cloud API to download data from the Arduino IoT Cloud, including the randomTemperature variable. You can programmatically access the properties and historical data of your Thing with this API. Here are the steps;

1. API Key and Thing ID:
   - Have your API key, Thing ID, and Variable ID from the Arduino IoT Cloud.
2. Headers:
   - Include the API key in the headers for authorization.
3. Time Calculation:
   - Calculate the start and end times for the last 10 minutes.
4. API Request:

	- Construct the URL to request historical data for the specified time range. - Make a GET request to the Arduino IoT Cloud API to fetch the data. 5. Save Data: - If the request is successful, save the fetched data to a JSON file for further analysis.
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: You can change the algorithm to open the file once at the beginning and close it at the end, appending data inside the callback function, in order to efficiently save the data received in the Python script to a file. This method is more effective because it reduces the overhead associated with often opening and closing the file. Here is the modified Algorithm: 1. Open a file in append mode at the start of the program. 2. In the callback function: a. Create a CSV string with <timestamp>, <value> <NEWLINE>. b. Write the CSV string to the file. c. Call flush to push the data to the file immediately. 3. Close the file at the end of the program.