Air quality sensor

Setting the scene

In this project, you are going to build a device to measure air quality. You will use the MKR 1000 board, MKR connector carrier and Grove Air Quality sensor. The connector carrier can be used to connect any of the Grove sensors.

Success criteria

- Understand what a **forever** loop does
- Understand what the air quality sensor measures
- Understand that if code is only to be executed if a condition is true, then an ‘if then else’ statement can be used
- Be able to report readings from a sensor to the Arduino IoT cloud

Instructions

The first step is to assemble your device.

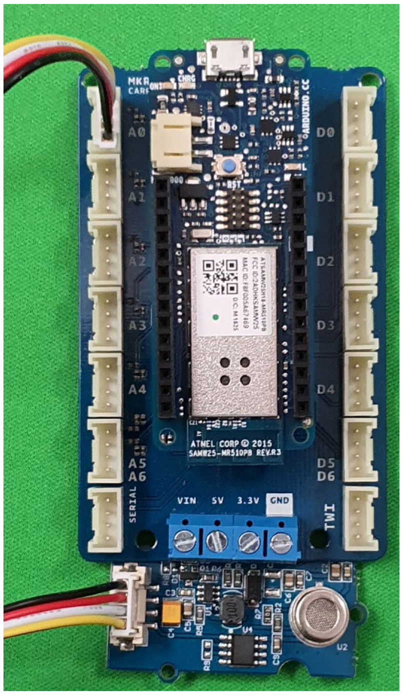
Plug the MKR1000 into the MKR Connector Carrier. Be careful to ensure that you line up the pins correctly and don’t cause any of the pins to bend. You need to ensure that all the pins on the MKR 1000 tie up with the equivalent labels on the connector carrier.

You should then plug in the Grove Air Quality sensor into port A0 on the connector carrier.

The Grove Air Quality sensor is an example of a shield. A shield is a modular circuit board which you can add to an Arduino to give it additional functionality.

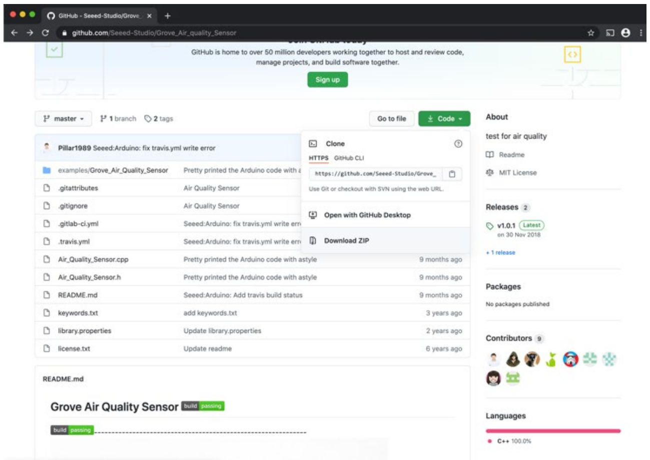
Pro-tip

The plug will only connect to the connector carrier one way around. If it doesn’t fit, don’t force it. You may have it the wrong way around.



Your device is now assembled and ready to be programmed.

Initially you need to install the relevant libraries. A library is a set of pre-written routines which a program can use. The Air Quality libraries can be downloaded from: https://github.com/Seeed-Studio/Grove_Air_quality_Sensor.



GitHub is home to over 50 million developers working together to host and review code, manage projects, and build software together.

```
https://github.com/Seeed-Studio/Grove_
```

Use Git or checkout with SVN using the web URL.

test for air quality

Pillar1989 Seeed:Arduino: fix travis.yml write error

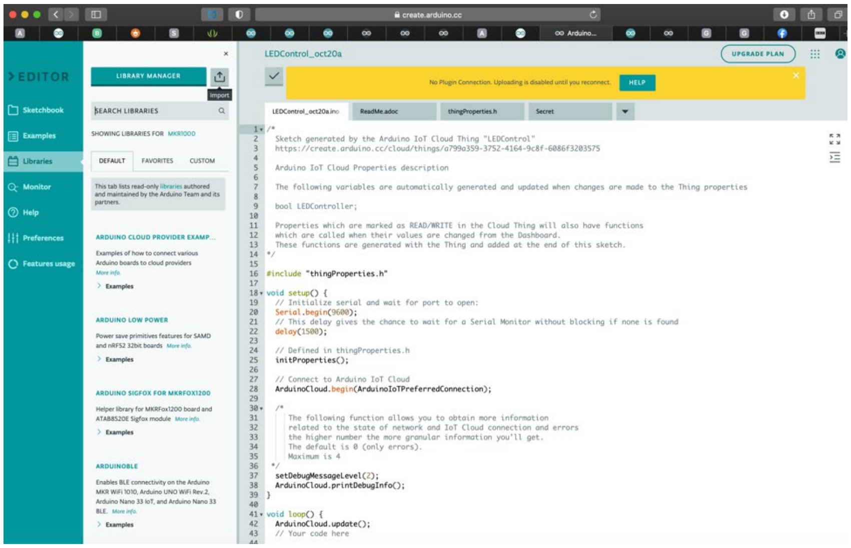
examples/Grove_Air_Quality_Sensor	Pretty printed the Arduino code with a	
.gitattributes	Air Quality Sensor	
.gitignore	Air Quality Sensor	
.gitlab-ci.yml	Seeed:Arduino: fix travis.yml write err	
.travis.yml	Seeed:Arduino: fix travis.yml write err	
Air_Quality_Sensor.cpp	Pretty printed the Arduino code with astyle	9 months ago
Air_Quality_Sensor.h	Pretty printed the Arduino code with astyle	9 months ago
README.md	Seeed:Arduino: Add travis build status	9 months ago
keywords.txt	add keywords.txt	3 years ago
library.properties	Update library.properties	2 years ago
license.txt	Update readme	6 years ago

README.md

Grove Air Quality Sensor

You should the click on ‘Code’ and download the zip file.

You now need to install the library into you IDE. Sign into the Arduino Cloud and go to the text editor. You should then click on ‘Libraries’.



This tab lists read-only libraries authored and maintained by the Arduino Team and its partners.

ARDUINO CLOUD PROVIDER EXAMP...

Examples of how to connect various Arduino boards to cloud providers More info.

Examples

ARDUINO LOW POWER

Power save primitives features for SAMD and nRF52 32bit boards More info.

Examples

ARDUINO SIGFOX FOR MKRFOX1200

Helper library for MKRFox1200 board and ATAB8520E Sigfox module More info.

Examples

ARDUINOBLE

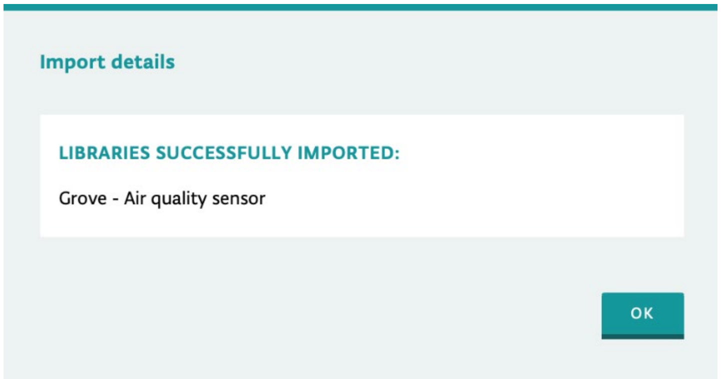
Enables BLE connectivity on the Arduino MKR WiFi 1010, Arduino UNO WiFi Rev.2, Arduino Nano 33 IoT, and Arduino Nano 33 BLE. More info.

Examples

No Plugin Connection. Uploading is disabled until you reconnect.

```
/*
  Sketch generated by the Arduino IoT Cloud Thing "LEDControl"
  https://create.arduino.cc/cloud/things/a799a359-3752-4164-9c8f-6086f3203575

  Arduino IoT Cloud Properties description

  The following variables are automatically generated and updated when changes are made to the Thing properties

  bool LEDController;

  Properties 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);

  /*
     The following function allows you to obtain more information
     related to the state of network and IoT Cloud connection and errors
     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
```

Once in the Libraries menu, you will see an option to import a library. Click on this button and then select your recently downloaded zip file.



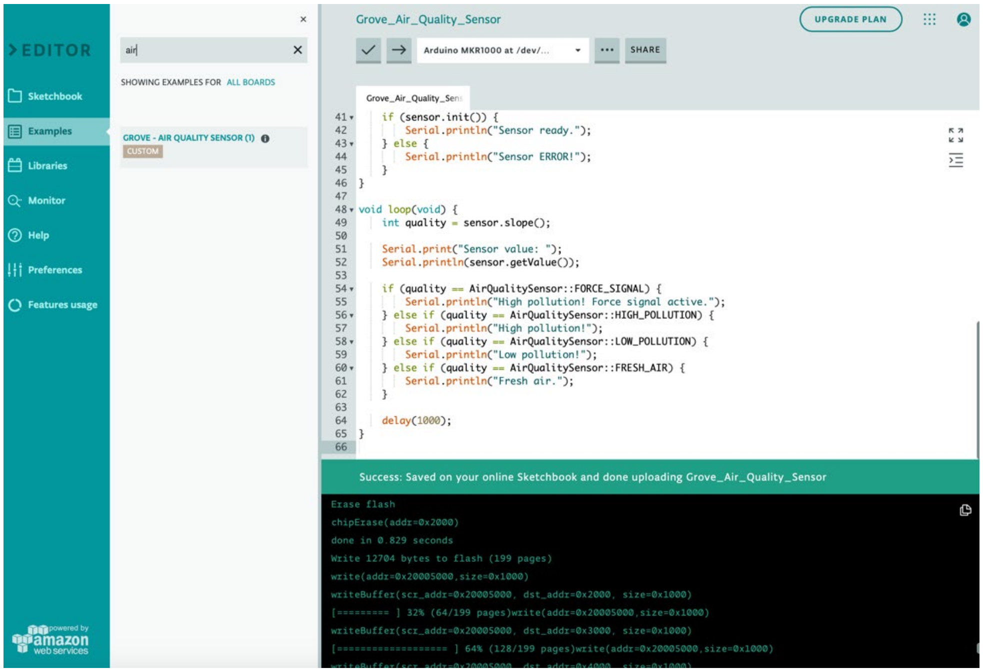
Import details

LIBRARIES SUCCESSFULLY IMPORTED:

Grove - Air quality sensor

You will then receive a confirmation message to inform you whether the library has been successfully imported.

You are now going to test your device using the example code that comes with the library.



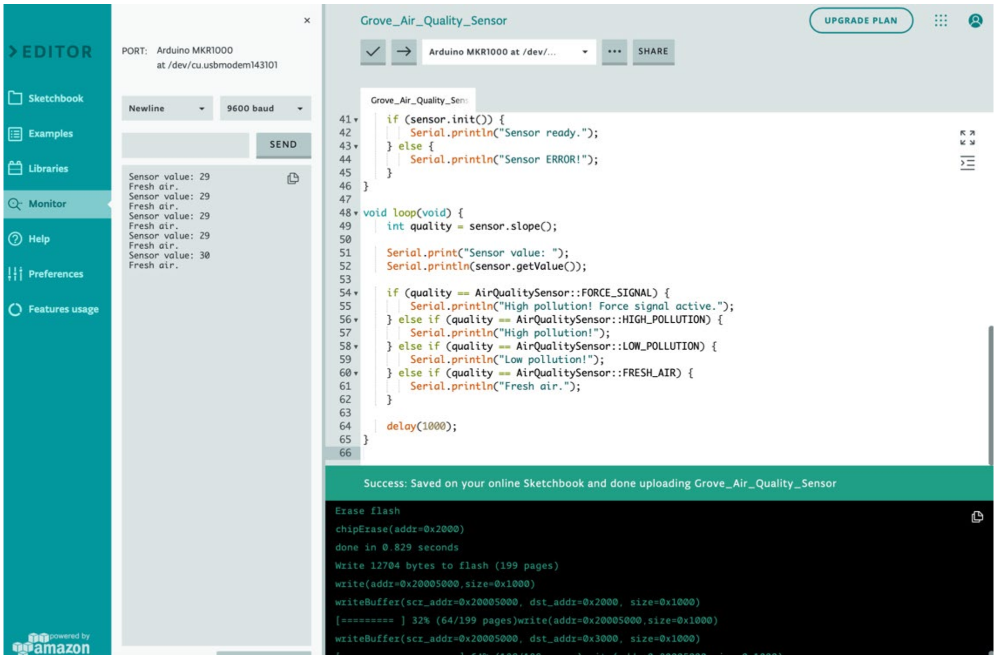
```
    if (sensor.init()) {
        Serial.println("Sensor ready.");
    } else {
        Serial.println("Sensor ERROR!");
    }
}

void loop(void) {
    int quality = sensor.slope();

    Serial.print("Sensor value: ");
    Serial.println(sensor.getValue());

    if (quality == AirQualitySensor::FORCE_SIGNAL) {
        Serial.println("High pollution! Force signal active.");
    } else if (quality == AirQualitySensor::HIGH_POLLUTION) {
        Serial.println("High pollution!");
    } else if (quality == AirQualitySensor::LOW_POLLUTION) {
        Serial.println("Low pollution!");
    } else if (quality == AirQualitySensor::FRESH_AIR) {
        Serial.println("Fresh air.");
    }

    delay(1000);
}
```

Success: Saved on your online Sketchbook and done uploading Grove_Air_Quality_Sensor

```
Erase flash
chipErase(addr=0x2000)
done in 0.829 seconds
Write 12704 bytes to flash (199 pages)
write(addr=0x20005000,size=0x1000)
writeBuffer(scr_addr=0x20005000, dst_addr=0x2000, size=0x1000)
[=========  ] 32% (64/199 pages)write(addr=0x20005000,size=0x1000)
writeBuffer(scr_addr=0x20005000, dst_addr=0x3000, size=0x1000)
[===================  ] 64% (128/199 pages)write(addr=0x20005000,size=0x1000)
```

Click on examples and search for air. You will then see a sample file for the Air Quality Sensor. Click on this file and then upload it to your device.

Once you have successfully uploaded it to your device, you should then select monitor from the left-hand menu. This will show you the readings that your device is currently taking. Please note that there may be a delay between uploading the sketch and the readings appearing.



PORT: Arduino MKR1000 at /dev/cu.usbmodem143101

Newline

9600 baud

```
Sensor value: 29
Fresh air.
Sensor value: 29
Fresh air.
Sensor value: 29
Fresh air.
Sensor value: 29
Fresh air.
Sensor value: 30
Fresh air.
```

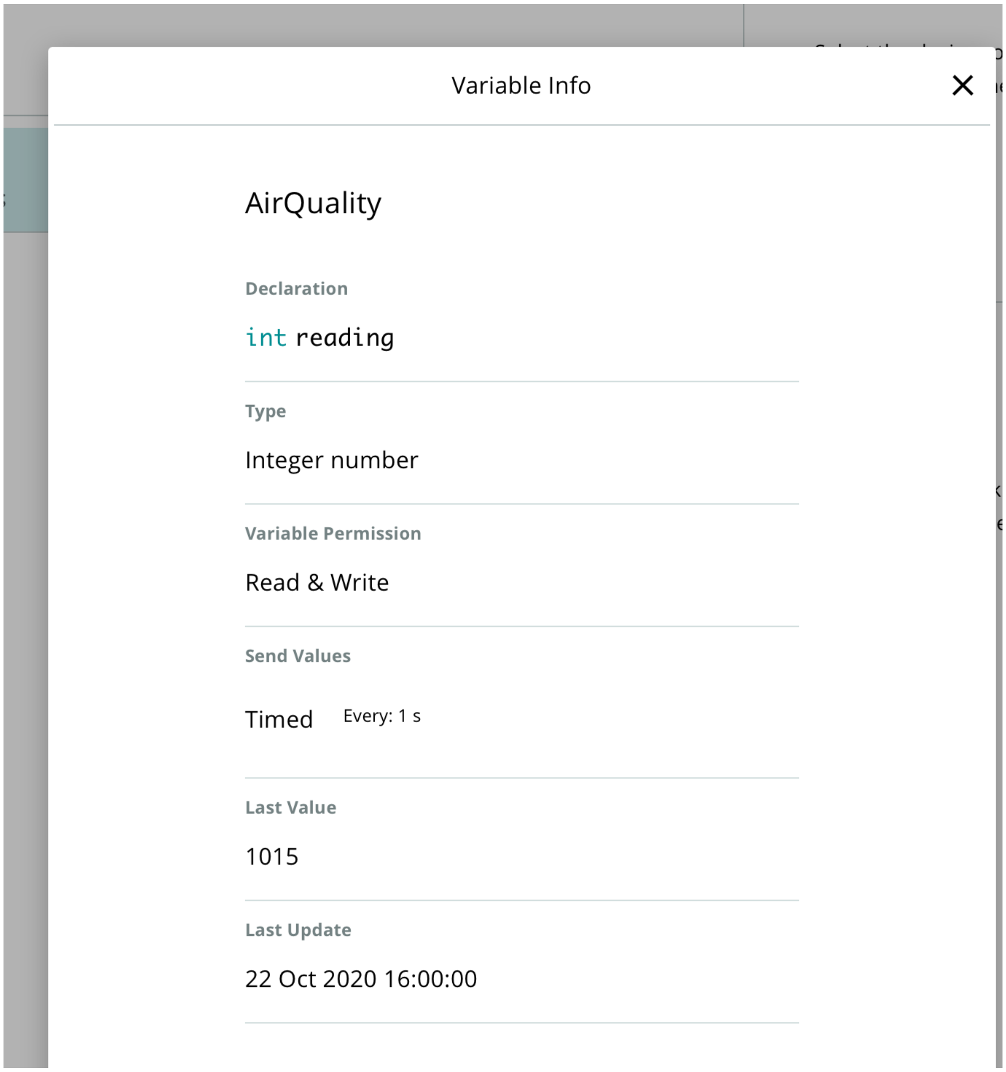
```
    if (sensor.init()) {
        Serial.println("Sensor ready.");
    } else {
        Serial.println("Sensor ERROR!");
    }
}

void loop(void) {
    int quality = sensor.slope();

    Serial.print("Sensor value: ");
    Serial.println(sensor.getValue());

    if (quality == AirQualitySensor::FORCE_SIGNAL) {
        Serial.println("High pollution! Force signal active.");
    } else if (quality == AirQualitySensor::HIGH_POLLUTION) {
        Serial.println("High pollution!");
    } else if (quality == AirQualitySensor::LOW_POLLUTION) {
        Serial.println("Low pollution!");
    } else if (quality == AirQualitySensor::FRESH_AIR) {
        Serial.println("Fresh air.");
    }

    delay(1000);
}
```

Success: Saved on your online Sketchbook and done uploading Grove_Air_Quality_Sensor

```
Erase flash
chipErase(addr=0x2000)
done in 0.829 seconds
Write 12704 bytes to flash (199 pages)
write(addr=0x20005000,size=0x1000)
writeBuffer(scr_addr=0x20005000, dst_addr=0x2000, size=0x1000)
[=========  ] 32% (64/199 pages)write(addr=0x20005000,size=0x1000)
writeBuffer(scr_addr=0x20005000, dst_addr=0x3000, size=0x1000)
```

At this stage you now know whether or not your device is taking readings. We now want to send these readings to the Arduino Cloud so that we can see a log of readings from throughout the day.

You should now return to the Arduino IoT Cloud and create a new thing. In this example it has been named AirQualityMonitor. Add the variable AirQuality using the properties detailed below and then click on 'Select device' and choose the Arduino board which you will use to run the device.



Variable Info

AirQuality

Declaration

`int reading`

Type

Integer number

Variable Permission

Read & Write

Send Values

Timed Every: 1 s

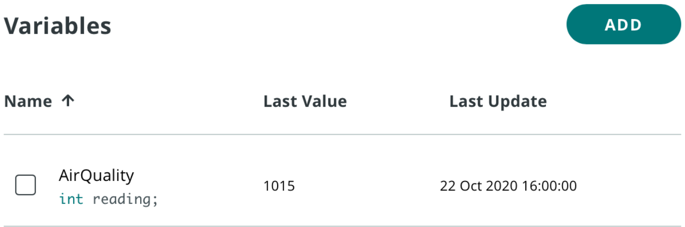
Last Value

1015

Last Update

22 Oct 2020 16:00:00

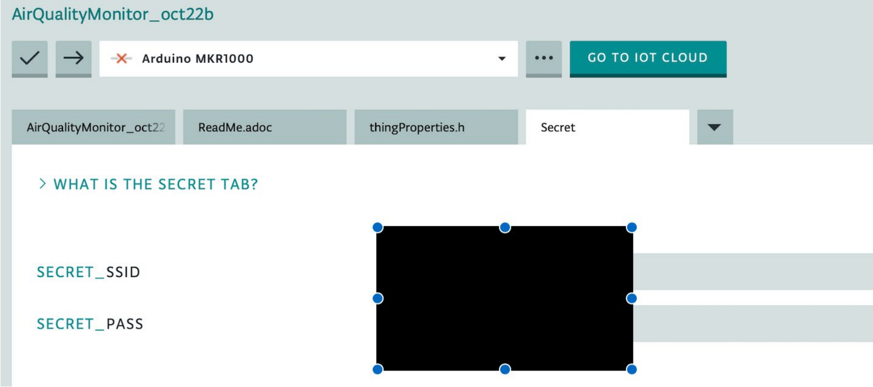
In the example above the variable is named reading. The sensor will return a whole number, so it is important that the data type is set as int. Click on ‘Save’.



Variables

	Name ↑	Last Value	Last Update
□	AirQuality `int reading;`	1015	22 Oct 2020 16:00:00

You should now select sketch and full editor. This will take you to the basic code which has been automatically generated. You should now click on the ‘Secret’ tab and enter the SSID and Password for your network.

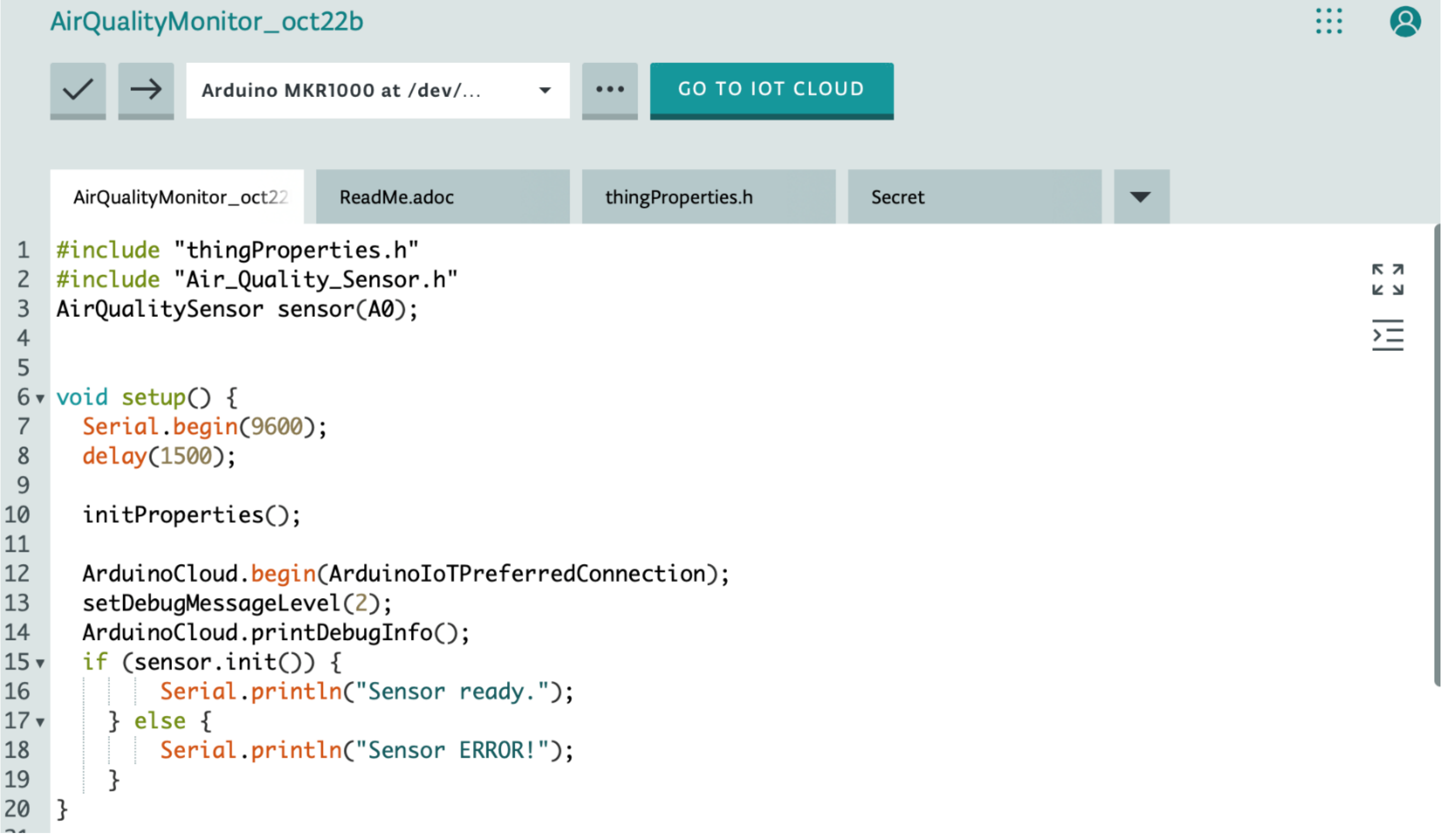


> WHAT IS THE SECRET TAB?

SECRET_SSID [redacted]

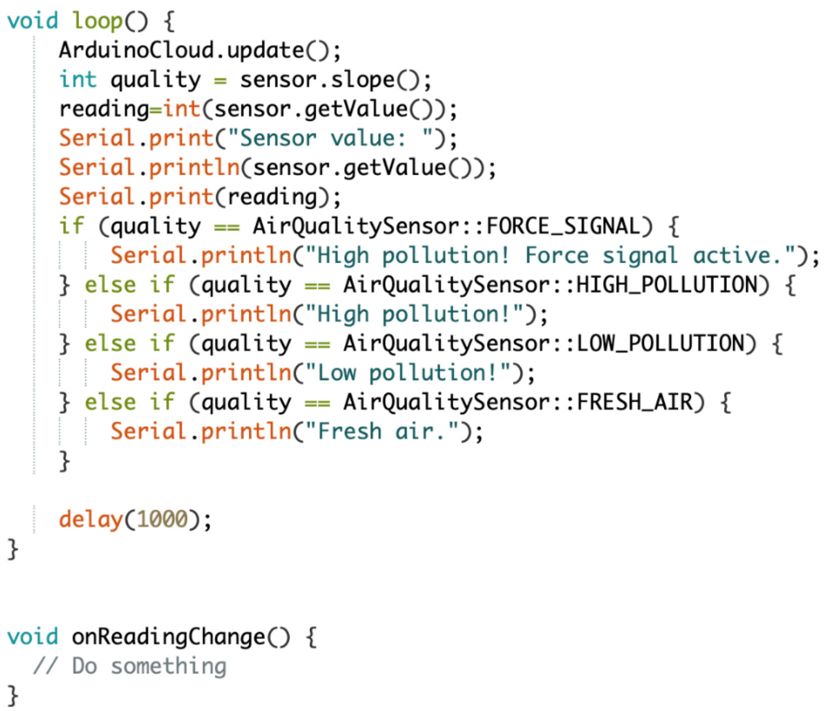
SECRET_PASS

You now need to write the sensor value back to the variable ‘Reading’ so that it can be sent back to the IoT cloud.



```
1  #include "thingProperties.h"
2  #include "Air_Quality_Sensor.h"
3  AirQualitySensor sensor(A0);
4
5
6  void setup() {
7    Serial.begin(9600);
8    delay(1500);
9
10   initProperties();
11
12   ArduinoCloud.begin(ArduinoIoTPreferredConnection);
13   setDebugMessageLevel(2);
14   ArduinoCloud.printDebugInfo();
15   if (sensor.init()) {
16         Serial.println("Sensor ready.");
17     } else {
18         Serial.println("Sensor ERROR!");
19     }
20 }
```

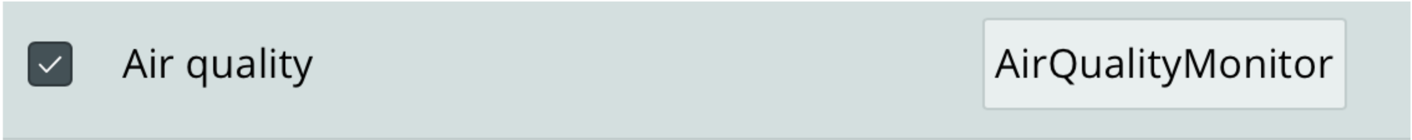
First, you should import the air quality sensor library using the code on line 2. You also need to say which port the sensor is plugged into which is defined on line 3. Inside the set-up loop, you should add the various lines relating to the sensor. These has been taken from the example file.

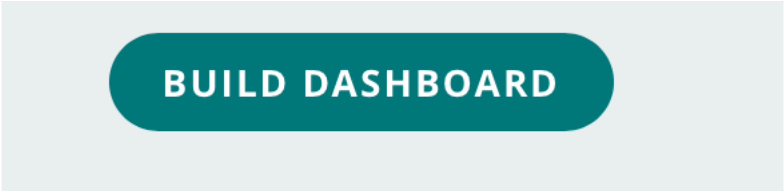


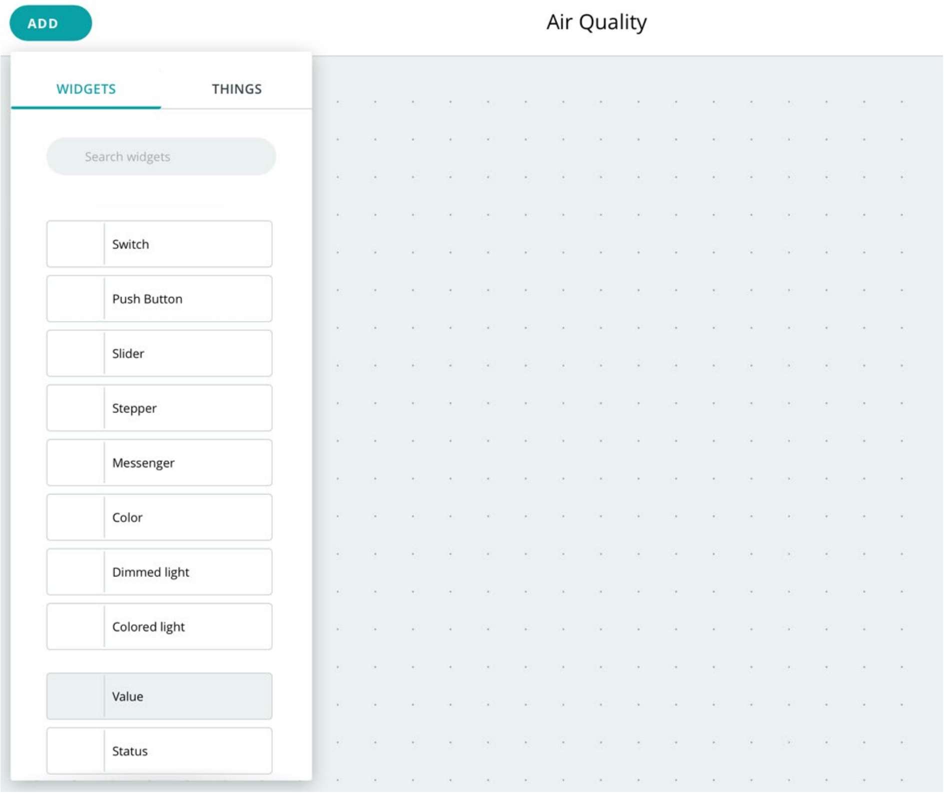
```
void loop() {
    ArduinoCloud.update();
    int quality = sensor.slope();
    reading=int(sensor.getValue());
    Serial.print("Sensor value: ");
    Serial.println(sensor.getValue());
    Serial.print(reading);
    if (quality == AirQualitySensor::FORCE_SIGNAL) {
        Serial.println("High pollution! Force signal active.");
    } else if (quality == AirQualitySensor::HIGH_POLLUTION) {
        Serial.println("High pollution!");
    } else if (quality == AirQualitySensor::LOW_POLLUTION) {
        Serial.println("Low pollution!");
    } else if (quality == AirQualitySensor::FRESH_AIR) {
        Serial.println("Fresh air.");
    }

    delay(1000);
}


void onReadingChange() {
  // Do something
}
```

It is important to state that the reading is an integer in order to ensure that the data is successfully transferred to the cloud. The if statement has been added from the example file, but this information isn't added to the cloud. You can now upload your program to your device and check in the monitor tab that values are recorded successfully. The Serial.print lines are the commands which are outputting to the monitor.

It is now time to create your IoT dashboard. Click on Goto IoT cloud and then dashboards. Select your 'Think' and then click on 'Build dashboard'.

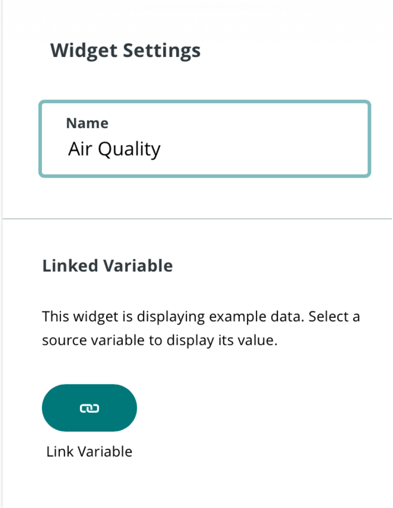






Click on ‘Add’ and choose the value widget.

Give your widget a name and then select ‘Link variable’.

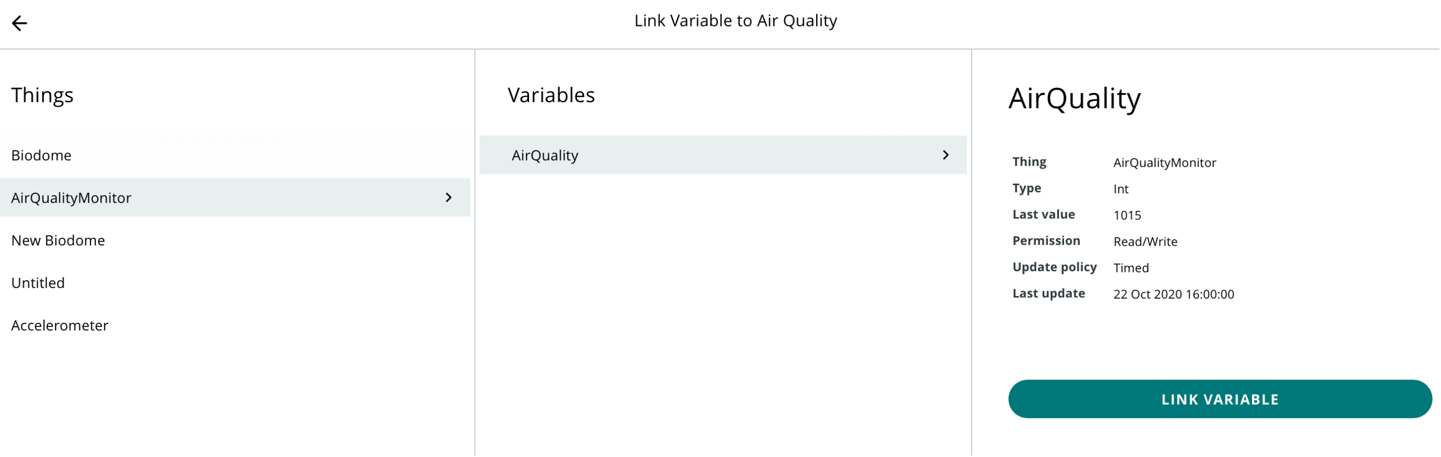

Widget Settings

Name: Air Quality

Linked Variable

This widget is displaying example data. Select a source variable to display its value.

You should then choose the appropriate variable and click on ‘Link variable’.



Link Variable to Air Quality

Things

- Biodome
- AirQualityMonitor
- New Biodome
- Untitled
- Accelerometer

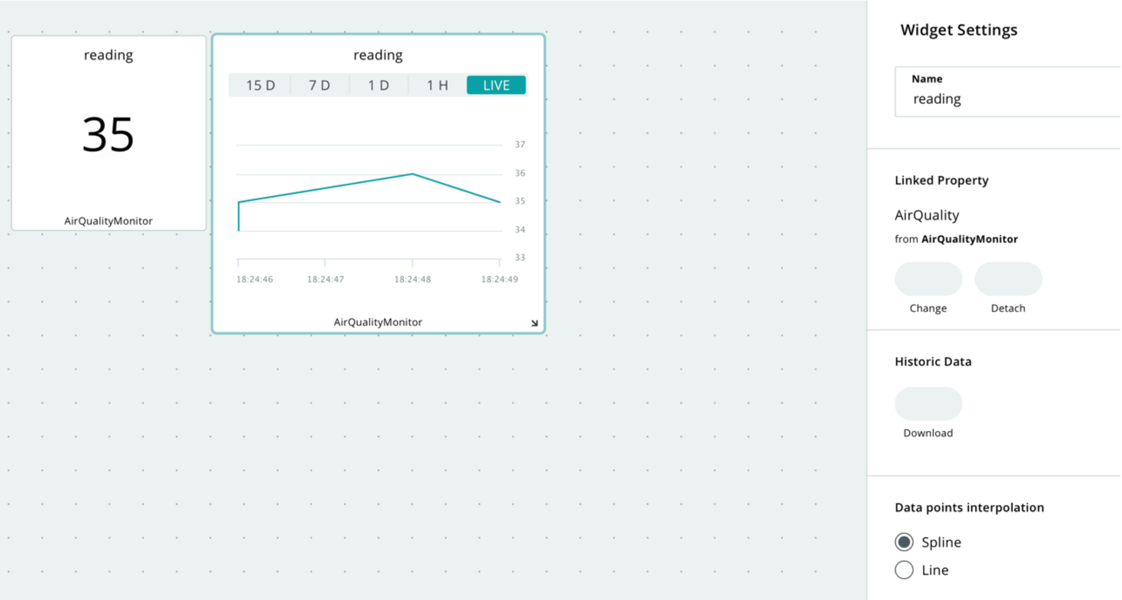
Variables

- AirQuality

AirQuality

Thing	AirQualityMonitor
Type	Int
Last value	1015
Permission	Read/Write
Update policy	Timed
Last update	22 Oct 2020 16:00:00

Repeat the same process for a chart widget. You will then see your data being displayed live.



Widget Settings

Name: reading

Linked Property

AirQuality

from **AirQualityMonitor**

Historic Data

Data points interpolation

● Spline

○ Line

Once you have a set of data ready for analysis, you could download all recorded data as a csv file by selecting download. You can then either analyse it using a spreadsheet or a coded application.

Testing

When you look at the dashboard you should be able to see live readings being taken. Try blowing into the sensor to see the impact that it has on the readings. All of the readings will be recorded onto a csv file. Open the file by clicking on the button under historic data. If your device has been built and programmed correctly, you should see all of the readings that it has taken.

Stretch tasks

Try to create a program which analyses the data that has been recorded in the CSV file. You may also wish to create a holder for the device and mount it around school to see how air quality changes during the day.

You could create a waterproof case to hold your device so that you can monitor the air quality around your environment at different times of the day. Please note that the device will need to be within wifi range. You could do this using an old plastic margarine tub. You will need to cut out a hole roughly the size of the sensor so that readings can be taken. Mount the holder to that the hole where the sensor is mounted points downwards so that rain will not get in. The device will also require a power source. You can use the battery connected provided with the kit. Connect the red wire to 5V and the black wire to GND.

A CSV file is a type of text file which separates the different values using a comma. To download the readings into a CSV file click on ‘Historic data’ and then ‘Download’ from the IoT dashboard screen.