# Air quality sensor

# Setting the scene

In this tutorial you are going to build a device to measure air quality. You will use the Arduino, breadboard and MQ-135 air quality sensor.

# 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 Arduino into the breadboard. Be careful to ensure that you line up the pins correctly and don’t cause any of the pins to bend.

Then, plug in the air quality sensor into the breadboard and connect port A0 on the Arduino to port A0 on the sensor. VCC should go to 5V and ground should go to ground..

[CH9\_IMG2.jpeg]

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 MQ135 library is part of the Arduino suite and can be installed from the ‘Libraries’ (in the Arduino Cloud) or ‘Manage Libraries’ (in the Arduino Software) menu.

[ CH9\_IMG3]

Once in the libraries menu you will see an option to **install**.

[NOTE: I’ve not included the burn in/calibration here – that should probably only be done once and takes 24 hours so may be best if the teacher organises that as per the book?]

Paste in this code:  
#include "MQ135.h" // MQ-135 library

const int ANALOGPIN = 0; // Pin to read the MQ-135

const float RZERO = 67.24; // Define the calibration setting

MQ135 gasSensor = MQ135(ANALOGPIN, RZERO); // Initialise the MQ-135

void setup() {

Serial.begin(9600); // Sets the serial port to 9600

}

void loop() {

float ppm = gasSensor.getPPM(); // Read the parts per million

Serial.println(ppm); // Output the parts per million

delay(1000); // Wait 1s

}

Run the code. At this stage you now know whether or not your device is taking readings. Now we 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. Don’t forget to add your WiFi details.

[L3\_IMG1]

Once you have created your thing, you should create a new variable. In the example above, the variable has been named reading. The sensor will return a float number, so it is important that the data type is set as float. It is only going to be read, so click **Read** Only. Click on ‘Add Variable.

[L3\_IMG2]

You should now select ‘Sketch’. This will take you to the basic code which has been automatically generated. You now need to write the sensor value back to the variable ‘reading’ so that it can be sent back to the IoT cloud.

[L3\_IMG3]

First, you should import the air quality sensor library using the code on line 1. You also need to say which port the sensor is plugged into which is defined on line 3. We define the RZERO value (which your teacher should give you) on line 4 and then create the gasSensor object (line 5). This is all very similar to the test code, with the additional thingProperties include.

[L3\_IMG4]

Inside the setup loop, we’ve turned on the in-built LED (lines 8-9) and turn it off and on again (lines 22 and 24-26) to indicate the Arduino is in the setup loop.

The important bit we have done here is simply to set reading to 1 so we can check it is coming through. This is on line 23.

This completes the setup loop. You now need to add the main loop which handles the readings.

[L3\_IMG5]

We have again flashed the built-in LED quickly to show the Arduino is in the main loop – then we read the PPM from the sensor and copy the value into the variable **reading**. You can now upload your program to your device and check in the setup tab that values are read successfully.

It is now time to create your IoT dashboard. Click on ‘Dashboards’ and then +DASHBOARD

[L3\_IMG6]

Create a new dashboard, give it a meaningful name and then add a value widget.

Give your widget a name and select **Link Variable**.

[L3\_IMG7]

Follow through the steps to link your variable.

[L3\_IMG8]

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

[L3\_IMG9] [L3\_IMG10]

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.

So far we have only read the data from the sensor – now we will attempt to interpret it. In **setup** add a new variable, **quality** of type String.

[L3\_IMG11]

In the sketch, add three new constant values to the top of the code (before the setup):  
[L3\_IMG12]

In the main loop we will now build a number of tests comparing the ppm against the constant values we created earlier, setting the quality String to an appropriate message each time. Each of these tests must be **Boolean** – that is must give us a yes/no (true/false) answer.

[L3\_IMG13]

You can then add another value to your dashboard and link it to the quality variable.

[L3\_IMG14]

# 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.