# Gas Leak Architecture Diagram

Microsoft Azure (IoT Cloud)

 

Gas Leak Sensor

Gas Leak Dashboard

The gas leak sensor on detecting a gas leak will send device-to-cloud messages to the IoT hub. The gas leak dashboard device continuously polls the IoT Hub, it reads the device-to-cloud messages and displays the messages to the user on the browser.

# Gas Leak Sensor

This is a Java console app that simulates a device that sends device-to-cloud messages to an IoT Hub. When the app is run it generates sample telemetry data to send to IoT Hub and waits for an acknowledgement before sending the next message:

Random rand = **new** Random();

**for**(**int** i = 1; i < 1001; i++ ) {

**int** currentGasLeak = rand.nextInt(1000-1) + i;

TelemetryDataPoint telemetryDataPoint =

**new** TelemetryDataPoint();

telemetryDataPoint.gasLeak = currentGasLeak;

String msgStr = telemetryDataPoint.serialize();

Message msg = **new** Message(msgStr);

System.***out***.println("Sending: " + msgStr);

Object lockobj = **new** Object();

EventCallback callback = **new** EventCallback();

*client*.sendEventAsync(msg, callback, lockobj);

**synchronized** (lockobj) {

lockobj.wait();

}

Thread.*sleep*(1000);

}

} **catch** (InterruptedException e) {

System.***out***.println("Finished.");

}

To run the application at command prompt in the gasleak-device folder, run the following command to begin sending telemetry data to IoT hub:

mvn exec:java -Dexec.mainClass="com.mycompany.app.simulated\_device.App"

# Gas Leak Dashboard

The dashboard reads device-to-cloud messages from IoT Hub and saves the messages to a mysql data and then displays the messages to the user in a browser window. To fetch new messages from IoT Hub, the user clicks refresh button:

