ARCES :: project by Edoardo Carrà date: 2/7/2019

Air Quality Sensor

Grove Air Quality Sensor V1.3

Introduction

ARCES IoT network is composed by several sensors in Brazil and in Bologna, which allows to monitoring some environment properties like temperature, pressure and humidity. The network is implemented using the MQTT protocol where the broker, in addition of guarantee the correct communication between heterogeneous sensor systems, share the data through the SEPA platform.
*The aim is making the correct configuration of the air pollution sensor :"**Grove Air Quality Sensor V1.3**" , inside the ARCES IoT network, using LinkIt 7697 like a MQTT client which will publish the values of the sensor to the broker server.*
This project enters in the contemporary IoT expansion challenge of interconnecting several sensor systems accessible by different people in all the world.

Here are shown two possible solutions for the correct configuration of a generic WiFi node inside the ARCES IoT network: one can be called the direct solution, the sensor communicate directly with the broker through WiFi connection, the second, instead, is made by the using of a bridge broker, so the sensor send the measures to a broker server in his local network which makes a bridge to the real MQTT broker.
To allow the communication with the broker it's necessary using a WiFi node, which publishes the sensor measures like a MQTT client. In this project it's used the board LinkIt 7697 which supports analogical input and WiFi connection, also it can be programmed with Arduino IDE

Grove Air Quality Sensor V1.3

The sensor used in this project is the "Grove Air Quality Sensor V1.3", which is designed for indoor air quality testing. It allows to make a qualitative acquisition of several gasses in the air, like: carbon monoxide, alcohol, acetone, thinner, formaldehyde. The sensor works at 5V and 3.3V and it's compatible with Arduino, Raspberry and other Grove systems. It needs 48 hours to warm-up and it can measuring gas in a range from 10 to1000 ppm.

sensor datasheet :
https://www.robotstore.it/rsdocs/documents/Sensore_MP503_datasheet.pdf

The output is analogical but it can be used the sensor library which makes the following AD conversion:

- Value : 3 FRESH AIR;
- Value : 2 LOW POLLUTION;
- Value : 1 HIGH POLLUTION;
- Value : 0 HIGH POLLUTION, FORCE SIGNAL.

So it's possible reading the sensor in two different ways: with the AD conversion of the library or directly from the analogical input.

When the library converts, it uses this scale:

- *Value 0 :* the currentVoltage - lastVoltage > 400 or currentVoltage > 700;
- *Value 1* : (the curentVoltage -lastVoltage > 400 and currentVoltage < 700) or currentVoltage - voltageAverage* >150;
- *Value 2* : (the curentVoltage -lastVoltage > 200 and currentVoltage < 700) or currentVoltage - voltageAverage* >50;
- *Value 3* : if the other condition aren't respected.

* The voltageAverage is the average of the all voltages measures by the sensor.

LinkIt 7697

This kind of board is powerful as it can run more complicated and store larger program, IoT application friendly and cost effective. It allows WiFi connections, it's compatible with the "Grove Air Quality Sensor V1.3" and it can be programmed by Arduino IDE.

For more information and features about the board: https://www.seeedstudio.com/LinkIt-7697-p-2818.html

Procedure for the direct implementation

Follow the procedure for the "direct" implementation:

1. Configure the node using Arduino IDE:
   - Install Arduino(https://www.arduino.cc/en/Main/Software) and then download the sensor library from Git Hub: https://github.com/Seeed-Studio/Grove_Air_quality_Sensor, you can add the library on Arduino IDE clicking on: *Sketch → Include Library → Add .ZIP library…*
   - Add the LinkIt 7697 board support package to Arduino IDE following this guide: https://docs.labs.mediatek.com/resource/linkit7697-arduino/en/environment-setup/setup-arduino-ide ;
   - Connect the sensor to the board following this table:

Black	GND
Red	5V or 3.3 V
White	Pin 15
Yellow	Pin 14 (or A0)

   - Make sure your configuration is ready by testing a sample sketch inside sensor library.

2. Configure the LinkIt 7697 like a MQTT client:
   - Install the MQTT client library here: https://github.com/knolleary/pubsubclient/releases/tag/v2.7 and add it to Arduino IDE like at the step 1;
   - It's necessary a WiFi connection which will allow the sensor node to reach the broker server.

```
/*********************************************************
*                                                        *
*        ARCES::GROVE AIR QUALITY SENSOR V1.3           *
* - Output Range:                                        *
*                 - 3 : Fresh Air;                       *
*                 - 2 : Low Pollution;                   *
*                 - 1 : High Pollution;                  *
*                 - 0 : High Pollution, Force Signal.    *
*                                                        *
*                                                        *
*                                  Developed by          *
*                                  Carrà Edoardo          *
*                                                        *
*********************************************************/

#include <PubSubClient.h> // import mqtt library
#include "Air_Quality_Sensor.h" // import air quality sensor grove v1 3
#include <LWiFi.h> // import wifi library

int server_port = ***; // port of mqtt broker
IPAddress server(***,***,***,***);  //ip of mqtt broker
//char server[] = "***";  //if you want to use dns instead ip

char ssid[] = "***";      //your network SSID
char pass[] = "***”;  //your network password

int status = WL_IDLE_STATUS;

WiFiClient clientWiFi;
PubSubClient client(clientWiFi);


AirQualitySensor sensor(A0); //change here the analogic pin of the air pollution sensor

//Reading message from the broker
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i=0;i<length;i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();
}

void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("AirQualitySensor")) {
      Serial.println("connected");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}

void setup() {
  Serial.begin(57600);
  Serial.println("Waiting sensor to init...");
  delay(2000);

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

  // attempt to connect to Wifi network:
  while (status != WL_CONNECTED) {
      Serial.print("Attempting to connect to SSID: ");
      Serial.println(ssid);
      // Connect to WPA/WPA2 network. Change this line if using open or WEP network:
      status = WiFi.begin(ssid, pass);
  }
  Serial.println("Connected to wifi");

  client.setServer(server, server_port);
  client.setCallback(callback);
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  else{
    //publishing the air quality index
    //sensor.slope() returns the index of air pollution ( a number from 3,fresh air, to 0, danger air pollution)
    String valore = String(sensor.slope());
    client.publish("AQI", valore.c_str()); //Air_quality_index(topic), value

    //publishing the voltage value of the sensor
    //sensor.getValue() returns directly the output of the sensor, which is analogical (0-1023)
    float x = sensor.getValue()*100; // formatting the analogical value in percentage value
    float aqs = x/1023;
    String val = String(aqs);
    client.publish("AQS",val.c_str()); //Air_quality_sensor(topic), value

    delay(60000);
  }
  client.loop();
}
```

As it’s possible to see, the values are sent in two different topics, the first is the air quality index (AQI), which is the library AD conversion, and the second is the air quality sensor (AQS), which is the analogical sensor output. It’s also possible sending the measures in one topic using JSON format, but it’s necessary guarantee the correct communication with your data format and the SEPA system.

Bridge implementation

This particular implementation is very useful when your sensor cannot access to main network for any reasons (too far from the access point, particular configuration of the network etc…). So this configuration consists in using a computer, which will be connected to main network and at the same time to the sensor with another local network(using the Hotspot of the computer or a router). On the computer will run a MQTT server which will be the local broker server of the sensor, but local broker server will act like a bridge to the real broker server of the main network. So the sensor will publish to the local broker server its measures through the local network, then the local broker server will publish to the real broker server through the main network.

It’s also possible creating more “local broker server” if it’s necessary.

This was the theoretical line. In this case there is air pollution sensor which it’s too far from the router and so it can’t communicate with the broker server of the ARCES IoT network. It’s possible using a computer, with Internet access, linked through local network to the sensor.

The procedure to create a broker bridge server inside your computer is as followed:

1. Install Mosquitto on your computer, the server will start automatically on 1883 port;
2. Create the bridge inside Mosquitto configurations:
   - you need to create a file inside /etc/mosquitto/conf.d/mosquitto.conf and write inside:

```
# bridge
connection bridge
address real_server_broker_ip_or_domain:broker_server_port
# if you want to publish everything
topic # out
```

3. Restart mosquitto service using:

```
sudo systemctl restart mosquitto
```

4. Follow the procedure for the “*direct implementation*” and insert within the code the ip and the port of your local broker server.

To verify the correct bridge implementation it’s possible to listen from your computer the publishing of your sensor, open another shell:

```
mosquitto_sub --host *** --port *** --verbose --topic '***/#’
```

It’s also possible establishing secure MQTT connections adding too:

```
bridge_cafile  /etc/mosquitto/certs/***
bridge_certfile /etc/mosquitto/certs/***
bridge_keyfile /etc/mosquitto/certs/***
```

All the certificates must be in the following directory */etc/mosquitto/certs/* .