Universidade de Coimbra
Faculdade de Ciências e Tecnologias da Universidade de Coimbra
Mestrado em Segurança Informática

Trabalho 4 – SmartBuilding Controller

Introdução:

Este trabalho foi realizado âmbito da cadeira de Sistemas de Comunicação Móvel com o intuito de construir um sistema de comunicação entre vários clientes MQTT. Este sistema pretende simular um *SmartBuilding Controller*.

Todo este trabalho foi realizado utilizando arduinos esp8266.

Modo de funcionamento do sistema:

O sistema é constituído por vários sensores (arduinos) que avaliam a temperatura, CO2, luz e atividade humana numa determinada sala e diversas aplicações (desenvolvidas em Python) de modo a gerir os ar-condicionados e luzes de uma sala. De modo a tornar isto possível, teremos 5 clientes MQTT (ou mais, dependendo do número de arduinos que analisam pacotes na rede):

- Arduino: é responsável por avaliar o estado da temperatura, CO2, luz e presença humana na sala em que está inserido. Sempre que houver alterações na temperatura e/ou nos níveis de CO2, esta informação irá ser reportada diretamente ao gestor de ar-condicionados e ao gestor central do sistema. Caso o nível de luz seja alterado, o gestor central do sistema também será notificado dessa mesma modificação. Se houver alteração de presença humana, irá informar em primeiro lugar o gestor central que, por sua vez, fica encarregue de avisar o gestor de ar-condicionados. Assim que se liga envia uma mensagem para o gestor central a informar que está em funcionamento. De 2.5 em 2.5 segundos verifica se existe algum novo update a efetuar ao software (recebido do gestor central).
- Gestor Central do Sistema: é o cérebro de todo o sistema. Neste podem ser executadas diversas ações como: update de software do arduino, mudar os limiares de temperatura, CO2 e luz referentes a cada sala, visualizar informação de cada sala (temperatura, CO2, luz, se o ar-condicionado e/ou as luzes estão ligadas, versão em que se encontra o software do sensor, se há ou não presença humana) e, por último, solicitar ao gestor das luzes que ligue ou desligue as luzes numa determinada sala. Esta aplicação além de também definir quais os limiares *default* a utilizar (no início da sua execução), irá receber também um aviso caso a luz de uma certa sala esteja abaixo do limiar estipulado de luz.
- Gestor de Ar-Condicionados: tem como função gerir os ar-condicionados de todas as salas. Sendo uma aplicação "inteligente", recebe a informação diretamente dos sensores e age consoante o resultado. Deste modo, assim que um arduino se conectar, este gestor não só é avisado do sucedido pelo gestor central como também recebe deste os parâmetros *default* para esse arduino. Perante qualquer atualização da temperatura e/ou CO2 (vinda dos sensores), este também é devidamente informado do ocorrido. Além disso, para saber se deve comparar as temperaturas recebidas com os limiares de presença ou ausência humana, é informado através do gestor central caso haja uma alteração. Se ocorrer alterações nos limiares de um certo arduino (efetuadas pelo gestor central), este é notificado e atualiza os mesmos.
- Gestor de Luzes: apenas recebe a informação vinda do gestor central para ligar ou desligar as luzes de uma sala específica.

João Carlos Borges Silva nº2019216753
Inês Martins Marçal nº2019215917

- Debug: sempre que uma mensagem for enviada para um dos tópicos definidos no sistema, o debug irá mostrar essa mensagem juntamente com o tópico onde foi efetuada a sua publicação.

Arquitetura e esquema de comunicação:

Para haver comunicação entre as diferentes aplicações e sensores foram definidos vários tópicos, os quais terão a função de transmitir toda a informação entre clientes MQTT:

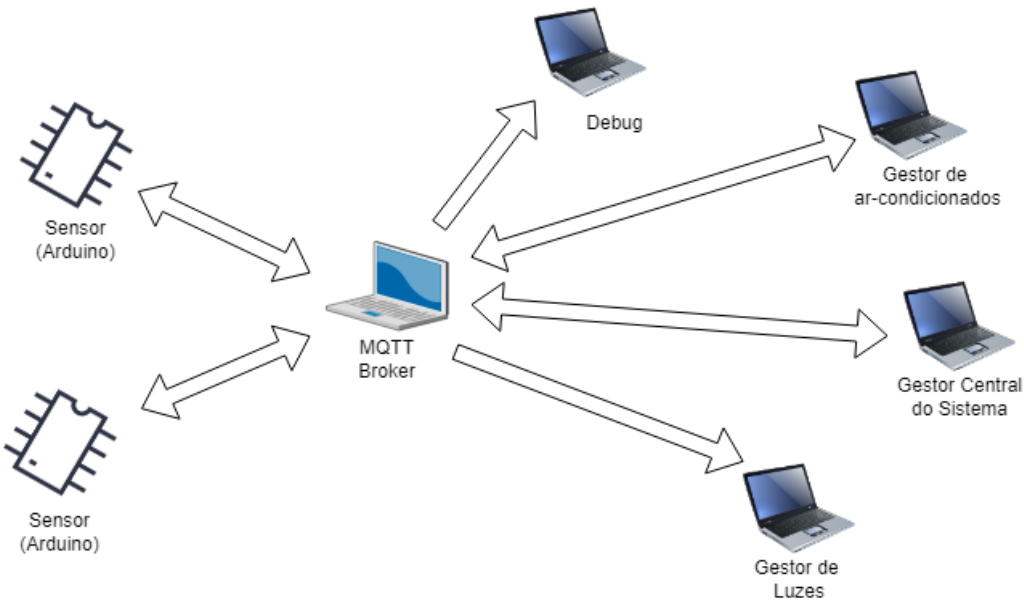
- Arduino:
  - Está subscrito:
    - **mqtt/update/ + [nome da sala]:** receber update vindo do gestor central. Formato da mensagem: "[versão nova]", exemplo: "1.01".
  - Irá publicar em:
    - **mqtt/light/from_sensor/ + [nome da sala]:** enviar atualização de luz para o gestor central. Formato da mensagem: "light:[valor da luz]", exemplo: "light:65".
    - **mqtt/avac/from_sensor/ + [nome da sala]:** enviar atualização de temperatura para o gestor de ar-condicionados. Formato da mensagem: "temp/co2:[valor da temperatura]", exemplos: "temp:34", "co2:56".
    - **mqtt/human_presence/from_sensor/ + [nome da sala]:** enviar atualização de presença humana para o gestor central. Formato da mensagem: "human:H/S", H significa que há presença humana, S significa que não há presença humana.
    - **mqtt/connect/from_sensor/ + [nome da sala]:** enviar uma mensagem apenas a informar que se ligou. Formato da mensagem: "ola".
- Gestor Central do Sistema:
  - Está subscrito:
    - **mqtt/connect/from_sensor/#:** receber mensagem de um arduino novo que se conecte. Formato da mensagem: "ola".
    - **mqtt/light/from_sensor/#:** receber atualização de luz vindo do arduino. Formato da mensagem: "light:[valor da luz]", exemplo: "light:65".
    - **mqtt/avac/from_sensor/#:** receber atualização de temperatura vindo do arduino. Além do gestor de ar-condicionados, este gestor está subscrito a este tópico para atualizar a sua própria informação acerca da condição de uma certa sala. Formato da mensagem: "temp/co2:[valor da temperatura]", exemplos: "temp:34", "co2:56".
    - **mqtt/human_presence/from_sensor/#:** receber atualização de presença humana vindo do arduino. Formato da mensagem: "human:H/S"
    - **mqtt/avac/condition/#:** receber quaisquer alterações vindas do gestor de ar-condicionados, nomeadamente se o ar-condicionado de uma sala está a aquecer, arrefecer ou ventilar. Formato da mensagem: "[0/1];[0/1];[0/1]", exemplo: "1;0;1", o primeiro valor serve para indicar se está ou não a aquecer, o valor seguinte serve para indicar se está ou não a arrefecer e o último para indicar se está ou não a ventilar, 0 - desligado, 1 - ligado.
    - **mqtt/will:** receber a mensagem *will* caso o gestor de ar-condicionados e/ou gestor de luzes se desconectem do *broker* MQTT. Formato da mensagem: "Avac leaving…" ou "Lights leaving…".

- Irá publicar em:
  - **mqtt/avac/info/ + [nome da sala]:** enviar parâmetros *default* do ar-condicionado de uma sala no qual o arduino tenha acabado de se conectar para o gestor de ar-condicionados. Formato da mensagem: "[temp. min com pres. humana];[temp. max com pres. humana];[temp. min sem pres. humana];[temp. max sem pres. humana];[valor CO2 seguro(mínimo)];[max de CO2];[0/1 - aquecer];[0/1 - arrefecer];[0/1 - ventilar];[1 - há presença humana no ínicio]", 0 - desligado, 1 - ligado, exemplo "10;20;5;25;10;80;70;0,0,0,1". NOTA: no início, como o ar-condicionado está desligado, os valores referentes ao seu estado irão estar sempre a 0.
  - **mqtt/update/ + [nome da sala]:** enviar update para um arduino. Formato da mensagem: "[versão nova]", exemplo: "1.01".
  - **mqtt/avac/update/ + [nome da sala]:** enviar qualquer alteração que seja feita aos limiares de temperatura ou CO2 de uma certa sala para o gestor de ar-condicionados. Também pode servir para informar este mesmo gestor se houve alterações ou não na presença humana. Formato da mensagem: "[número do valor a alterar (1-6 ou 10)];[valor]", exemplo: "3:45". Números dos valores: 1 - temp. min com pres. humana, 2 - temp. max com pres. humana, 3 - temp. min sem pres. humana, 4 - temp. max sem pres. humana, 5 - valor CO2 seguro(mínimo), 6 - max de CO2, 10 - presença humana (neste caso o valor será 0 ou 1).
  - **mqtt/light/update/ + [nome da sala]:** informar o gestor de luzes se deve ligar ou desligar as luzes de uma sala. Formato: "on/off", on para ligar, off para desligar.
- <u>Gestor de Ar-Condicionados</u>:
  - Está subscrito:
    - **mqtt/avac/from_sensor/#:** receber atualização de temperatura vindo do arduino. Formato da mensagem: "temp/co2:[valor da temperatura]", exemplos: "temp:34", "co2:56".
    - **mqtt/avac/info/#:** receber os parâmetros *default* do ar-condicionado de uma sala a que o arduino se tenha acabado de conectar, provenientes, por sua vez, do gestor central. Formato da mensagem: "[temp. min com pres. humana];[temp. max com pres. humana];[temp. min sem pres. humana];[temp. max sem pres. humana];[valor CO2 seguro(mínimo)];[max de CO2];[0/1 - aquecer];[0/1 - arrefecer];[0/1 - ventilar];[1 - há presença humana no ínicio]", 0 - desligado, 1 - ligado, exemplo "10;20;5;25;10;80;70;0,0,0,1".
      NOTA: no início, como o ar-condicionado está desligado, os valores referentes ao seu estado irão estar sempre a 0.
    - **mqtt/avac/update/#:** receber qualquer alteração que seja feita aos limiares de temperatura ou CO2 de uma certa sala, sendo esta indicada pelo gestor central. Também pode servir para informar este mesmo gestor se houve alterações ou não na presença humana. Formato da mensagem: "[número do valor a alterar (1-6 ou 10)];[valor]", exemplo: "3:45". Números dos valores: 1 - temp. min com pres. humana, 2 - temp. max com pres. humana, 3 - temp. min sem pres.

humana, 4 - temp. max sem pres. humana, 5 - valor CO2 seguro(mínimo), 6 - max de CO2, 10 - presença humana (neste caso o valor será 0 ou 1).

- Irá publicar em:
  - **mqtt/avac/condition/ + [nome da sala]:** enviar quaisquer alterações para o gestor central, nomeadamente se o ar-condicionado de uma sala está a aquecer, arrefecer ou ventilar. Formato da mensagem: "[0/1];[0/1];[0/1]", exemplo: "1;0;1", o primeiro valor serve para indicar se está ou não a aquecer, o valor seguinte serve para indicar se está ou não a arrefecer e o último para indicar se está ou não a ventilar, 0 - desligado, 1 - ligado.
  - **mqtt/will:** enviar a mensagem *will* caso o gestor de ar-condicionados se desconecte do *broker* MQTT. Formato da mensagem: "Avac leaving…".
- Gestor de Luzes:
  - Está subscrito:
    - **mqtt/light/update/#:** informar o gestor de luzes se deve ligar ou desligar as luzes de uma sala. Formato: "on/off", on para ligar, off para desligar.
  - Irá publicar em:
    - **mqtt/will:** enviar a mensagem *will* caso o gestor de luzes se desconecte do *broker* MQTT. Formato da mensagem: "Lights leaving…".
- Debug:
  - Está subscrito:
    - **mqtt/#:** para visualizar todas as mensagens que passam por todos tópicos utilizados pelo sistema.

É de notar que é utilizada a *wildcard* # em vários tópicos, para que seja possível abranger todos os arduinos contidos nas diferentes salas, excepto no caso do tópico "mqtt/#", onde o objetivo passa por abranger todos os tópicos que tenham como início "mqtt/", ou seja, todos os tópicos usados pelos sistema.



Detalhes de Implementação:

Arduino: São inicializados no início do programa variáveis com tópicos a subscrever ou para onde mais tarde se irá enviar informação, sendo, além disso, também criado um cliente MQTT. Dentro da função setup() é iniciada uma ligação *WiFi*, o cliente MQTT subscreve ao tópico para mais tarde receber updates do gestor central e envia uma mensagem para um o tópico "mqtt/connect/from_sensor/ + [nome da sala]" para informar que está em funcionamento (mais detalhes sobre os tópicos no capítulo anterior).

Dentro da função *loop()* encontra-se um *if_else*. Caso o utilizador introduza alguma coisa no Serial Monitor (ou seja, vai atualizar algum valor de temperatura, CO2, luz ou presença humana) entra-se dentro do primeiro *if*, conecta-se ao *broker* MQTT e envia-se a informação introduzida para os respectivos tópicos. Por outro lado, caso não se tenha introduzido nada apenas se verifica se há algum update publicado no tópico subscrito. No final da função *loop()*, o cliente MQTT desconecta-se do *broker* e efetua um *sleep* de 2.5 segundos, de modo a poupar no consumo de energia.

Foi criada também uma função chamada *MQTT_connect()* que, por sua vez, permitirá iniciar uma ligação ao *broker* MQTT, sendo chamada em todos os pontos em que os arduinos necessitam de comunicar com o *broker*, seja para ler o tópico anteriormente referenciado ou para publicar informação.

(Todas as aplicações detalhadas daqui em diante foram criadas usando código Python)

Gestor Central do Sistema: A aplicação começa por pedir como *input* os parâmetros *default* que cada sala terá ao conectar-se e de seguida é inicializada uma *thread* que terá como propósito suportar o cliente MQTT.

No processo principal irá ser efetuado a introdução de inputs, ou seja, comandos para realizar as diversas ações anteriormente referidas. Procede-se à introdução destes mesmos *inputs* dentro de um ciclo *while(True)* com um *time.sleep(1),* de modo a não causar uma espera ativa. Os comandos que podem vir a ser introduzidos encontram-se listados em seguida:

- **"condition [nome da sala]":** para visualizar o estado de uma sala específica, seja temperatura; nível de CO2; nível de luz; se há presença humana; visualizar se o ar-condicionado está a aquecer, arrefecer, ventilar ou desligado; se as luzes estão ou não ligadas; e ver a versão do software.
- **"config [nome da sala] [1-7] [valor]":** configurar os limiares de uma sala específica. Os números de 1 a 7 identificam, respetivamente, limiares de [temp. min com pres. humana], [temp. max com pres. humana], [temp. min sem pres. humana], [temp. max sem pres. humana], [valor CO2 seguro(mínimo)], [max de CO2] ou [de luz].
- **"update [nome da sala]":** atualizar o software de uma sala específica
- **"help":** para visualizar todos os comandos

Todos os comandos executados encontram-se interligados com uma *thread* que, por sua vez, suporta o cliente MQTT. Dependendo do comando introduzido, a mesma é responsável por publicar essa informação nos tópicos definidos ou efetuar apenas o seu *print* no terminal. É ainda efetuado o handle do *Ctrl+C* que, quando executado, irá encerrar não só o programa como também a ligação do cliente MQTT ao *broker*.

Na thread que suporta o cliente MQTT são subscritos os tópicos anteriormente referidos e sempre que recebida uma nova informação esta é tratada conforme o que foi descrito na arquitetura. Além destes tópicos, existem também aqueles que serão utilizados para enviar informação (ver capítulo da Arquitetura). Há métodos definidos nesta thread que estão ligados ao processo principal, sendo utilizados para publicar algo nos tópicos definidos ou efetuar o *print* da informação requisitada no terminal. Toda a informação acerca dos sensores (arduinos) é guardada durante a execução do programa num dicionário.

Gestor de Ar-condicionados: tal como no programa anterior temos tópicos que são subscritos para receber informação perante alguma alteração na temperatura, CO2, indicador de presença humana, limiares de um sensor ou quando um arduino novo se conecta e cria consequentemente uma entrada no dicionário para os limiares do mesmo. Procede-se ainda à definição de alguns tópicos, onde, posteriormente, irá ser efetuada a publicação de dados como a alteração no estado do ar-condicionado de uma sala ou quando este gestor encerra o seu cliente MQTT (sendo definida uma *will* de forma saber que o mesmo encerrou).

Como já referido, este gestor apresenta um dicionário com entrada para todos os sensores ligados, onde os valores dos limiares, funcionamento do ar-condicionado e indicação de presença humana são guardados. Isto permite, assim, evitar casos como os seguintes: caso um ar-condicionado estiver a aquecer uma sala e ao mesmo tempo ventilar por causa dos níveis de CO2, só se desliga o mesmo apenas quando a temperatura e o CO2 tiverem ambos com valores admissíveis e não só quando um deles apresentar valores seguros; ou, se um ar-condicionado já estiver a arrefecer e ainda receber uma temperatura acima do limiar superior, não indicar que voltou a ligar o arrefecimento. É de notar que caso haja alterações nos ar-condicionados esta informação é apresentada no terminal.

Gestor de Luzes: nesta aplicação é apenas criado um cliente MQTT e o mesmo será subscrito apenas a um tópico, onde irá receber ordens para apagar ou acender luzes de uma sala específica. Quando pressionado o *Ctrl+C,* o cliente é desconectado do *broker* e o programa é encerrado. À semelhança do gestor de ar-condicionados é ainda definida uma *will* com vista a informar que esta aplicação encerrou.

Debug: nesta aplicação é apenas criado um cliente MQTT e o mesmo será subscrito unicamente a um tópico. Através deste, irá ser possível visualizar todas as mensagens que passam pelos tópicos relativos ao sistema, ou seja, os que comecem por “mqtt/”. Quando pressionado o *Ctrl+C,* o cliente é desconectado do *broker* e o programa é encerrado.

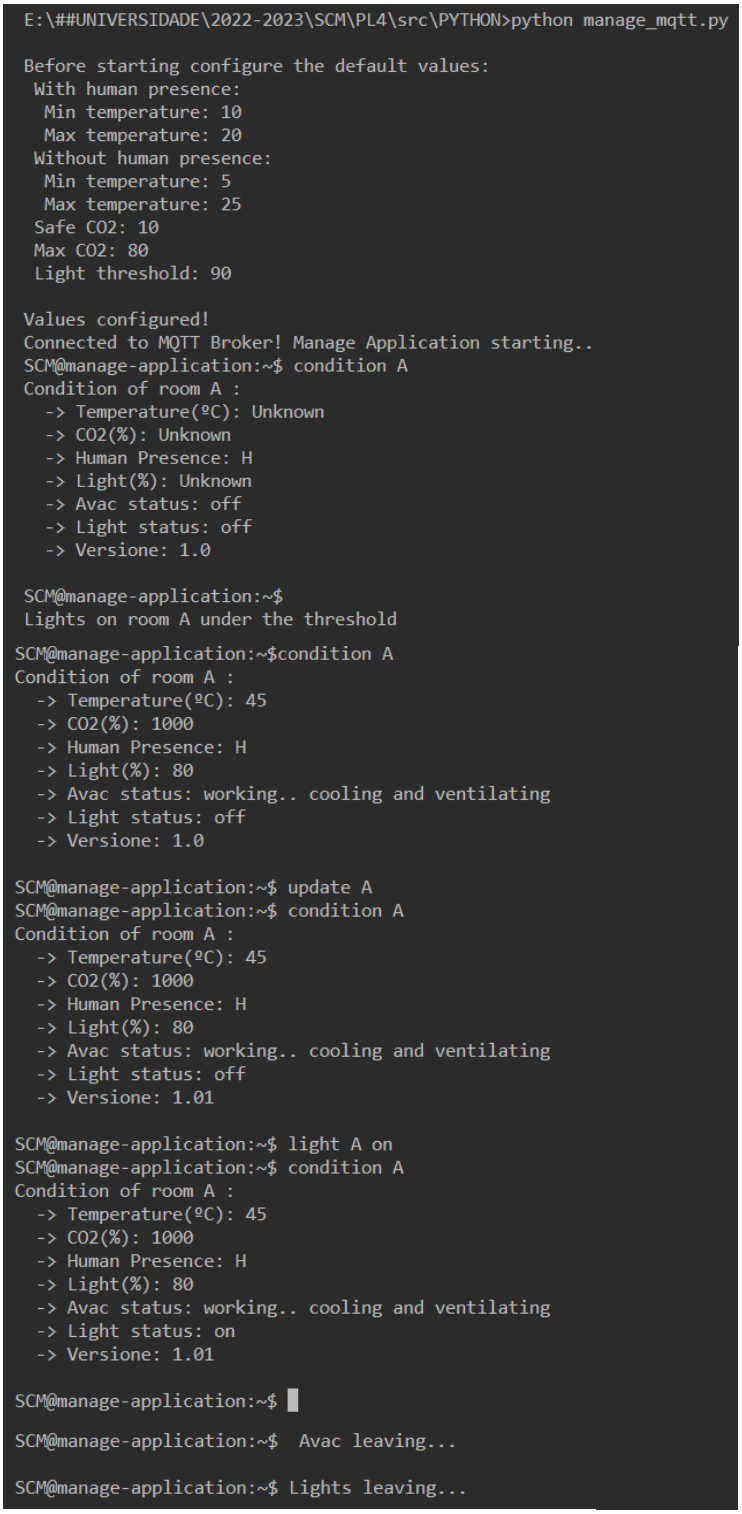
Bibliotecas e código utilizado:

- Arduino: Arduino, ESP8266WiFI, Adafruit_MQTT, Adafruit_MQTT_Client.
- Python: signal, time, sys, random, threading, paho.mqtt

Problema encontrado:

Infelizmente, devido à biblioteca usada nos sensores (arduinos) não conter um método de retenção nas mensagens publicadas, não é possível existir um cenário em que os arduinos se liguem primeiro ao *broker* e só posteriormente ao gestor central.

Algumas imagens de como funcionam os diferentes clientes MQTT:



Gestor Central do Sistema

João Carlos Borges Silva nº2019216753
Inês Martins Marçal nº2019215917

```
E:\##UNIVERSIDADE\2022-2023\SCM\PL4\src\PYTHON>python avac_mqtt.py
connect_mqtt
Values for room A received.
Avac booting on room A
Avac cooling on room A
Avac ventilating on room A
Avac not heating/cooling on room A
Avac not ventilating on room A
Avac shutingdown on room A
```

Gestor de Ar-Condicionados

```
E:\##UNIVERSIDADE\2022-2023\SCM\PL4\src\PYTHON>python light_mqtt.py
connect_mqtt
Lights of room A turned on
```

Gestor de Luzes

```
E:\##UNIVERSIDADE\2022-2023\SCM\PL4\src\PYTHON>python debug.py
connect_mqtt
Topic: mqtt/connect/from_sensor/A
Content: ola

Topic: mqtt/avac/info/A
Content: 10;20;5;25;10;80;0;0;0;1

Topic: mqtt/avac/from_sensor/A
Content: temp;45

Topic: mqtt/avac/condition/A
Content: 1;0;0

Topic: mqtt/avac/from_sensor/A
Content: co2;1000

Topic: mqtt/avac/condition/A
Content: 1;0;1

Topic: mqtt/light/from_sensor/A
Content: 80

Topic: mqtt/update/A
Content: 1.01

Topic: mqtt/light/update/A
Content: on
Topic: mqtt/will
Content:  Avac leaving...

Topic: mqtt/will
Content:  Lights leaving...
```

Debug

João Carlos Borges Silva nº2019216753
Inês Martins Marçal nº2019215917

```
............
WiFi connected
Connecting to MQTT... Connection failed
Retrying MQTT connection in 5 seconds...
MQTT Connected!
Connecting to MQTT... MQTT Connected!
Connecting to MQTT... MQTT Connected!
temp: 45
Connecting to MQTT... MQTT Connected!
Sending temperature...
Connecting to MQTT... MQTT Connected!
co2: 1000
Connecting to MQTT... MQTT Connected!
Sending co2...
Connecting to MQTT... MQTT Connected!
light: 80
Connecting to MQTT... MQTT Connected!
Sending light...
Connecting to MQTT... MQTT Connected!
Connecting to MQTT... Connection failed
Retrying MQTT connection in 5 seconds...
MQTT Connected!
Connecting to MQTT... MQTT Connected!
Connecting to MQTT... MQTT Connected!
1.01
Connecting to MQTT... MQTT Connected!
Connecting to MQTT... MQTT Connected!
Connecting to MQTT... MQTT Connected!
Connecting to MQTT... MQTT Connected!
Connecting to MQTT... MQTT Connected!
Connecting to MQTT... Connection failed
Retrying MQTT connection in 5 seconds...
MQTT Connected!
temp: 19
Connecting to MQTT... MQTT Connected!
Sending temperature...
Connecting to MQTT... MQTT Connected!
co2: 56
```

Arduino