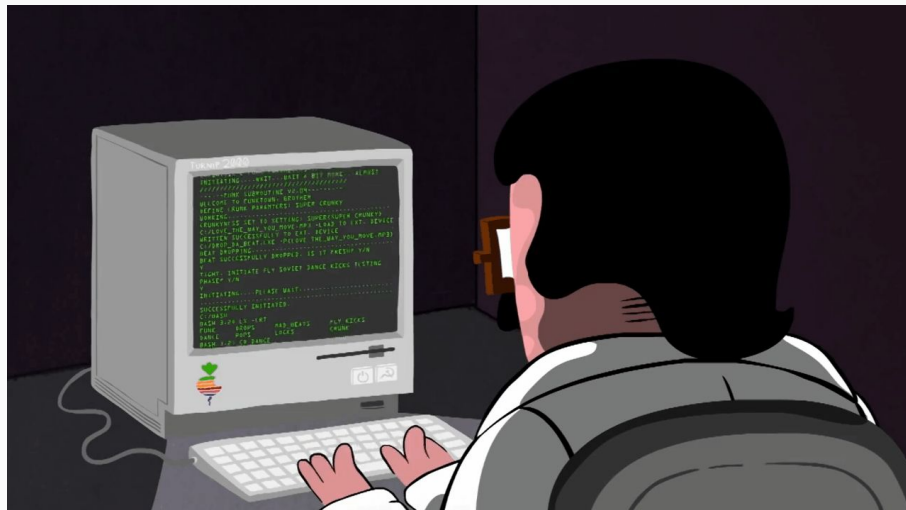


Professor: Luiz Felipe Oliveira

Publicando na Nuvem
com ESP32 e MQTT

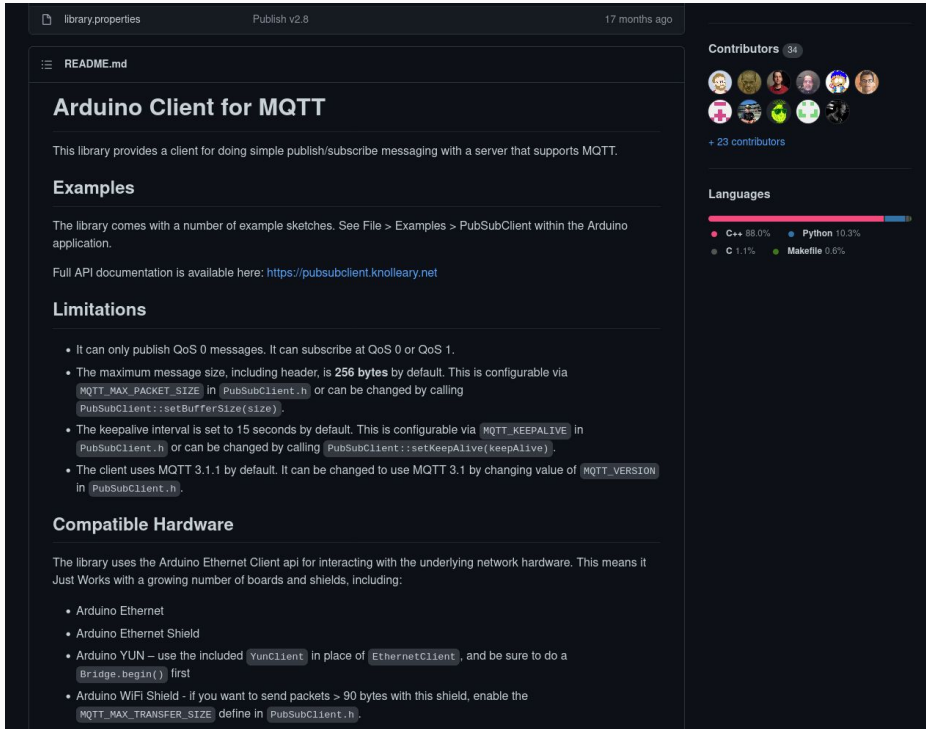


Parte 0: Explicando o MQTT

Parte 1: Publicação dos Dados

Parte 2: Visualização dos Dados

<https://github.com/knolleary/pubsubclient>



library.properties Publish v2.8 17 months ago

README.md

Arduino Client for MQTT

This library provides a client for doing simple publish/subscribe messaging with a server that supports MQTT.

Examples

The library comes with a number of example sketches. See File > Examples > PubSubClient within the Arduino application.

Full API documentation is available here: <https://pubsubclient.knolleary.net>

Limitations

- It can only publish QoS 0 messages. It can subscribe at QoS 0 or QoS 1.
- The maximum message size, including header, is **256 bytes** by default. This is configurable via `MQTT_MAX_PACKET_SIZE` in `PubSubClient.h` or can be changed by calling `PubSubClient::setBufferSize(size)`.
- The keepalive interval is set to 15 seconds by default. This is configurable via `MQTT_KEEPALIVE` in `PubSubClient.h` or can be changed by calling `PubSubClient::setKeepAlive(keepAlive)`.
- The client uses MQTT 3.1.1 by default. It can be changed to use MQTT 3.1 by changing value of `MQTT_VERSION` in `PubSubClient.h`.

Compatible Hardware

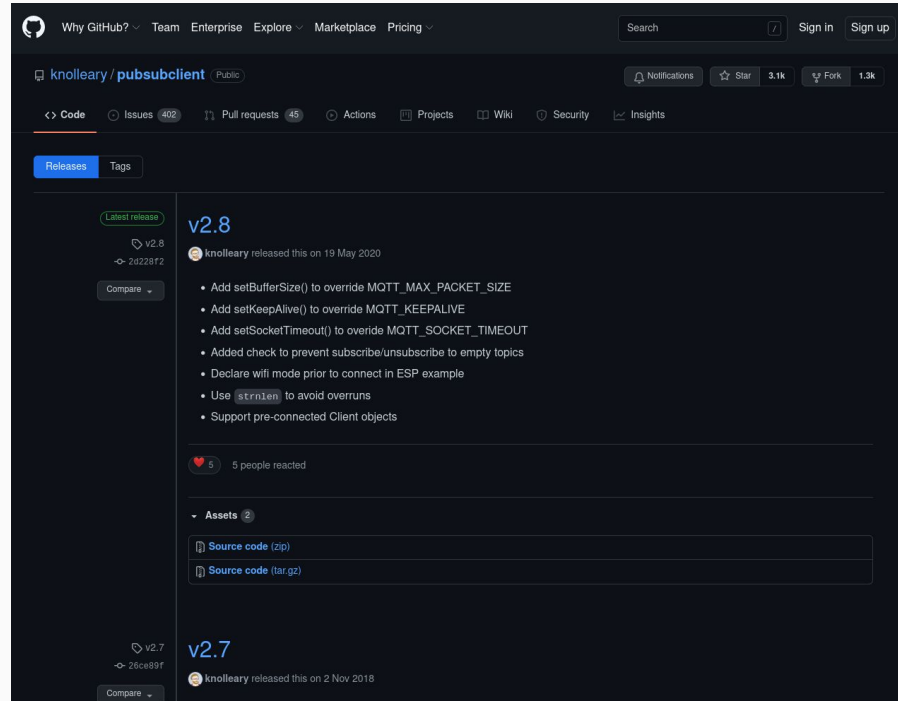
The library uses the Arduino Ethernet Client api for interacting with the underlying network hardware. This means it Just Works with a growing number of boards and shields, including:

- Arduino Ethernet
- Arduino Ethernet Shield
- Arduino YUN – use the included `YunClient` in place of `EthernetClient`, and be sure to do a `Bridge.begin()` first
- Arduino WiFi Shield - if you want to send packets > 90 bytes with this shield, enable the `MQTT_MAX_TRANSFER_SIZE` define in `PubSubClient.h`.

Contributors 34

Languages

- C++ 88.0%
- Python 10.3%
- C 1.1%
- Makefile 0.6%



Why GitHub? Team Enterprise Explore Marketplace Pricing

Search [7] Sign In Sign up

knolleary/pubsubclient Public

Notifications Star 3.1k Fork 1.3k

<> Code Issues 402 Pull requests 45 Actions Projects Wiki Security Insights

Releases Tags

Latest release

v2.8

knolleary released this on 19 May 2020

- Add `setBufferSize()` to override `MQTT_MAX_PACKET_SIZE`
- Add `setKeepAlive()` to override `MQTT_KEEPALIVE`
- Add `setSocketTimeout()` to override `MQTT_SOCKET_TIMEOUT`
- Added check to prevent subscribe/unsubscribe to empty topics
- Declare wifi mode prior to connect in ESP example
- Use `strlen` to avoid overruns
- Support pre-connected Client objects

5 5 people reacted

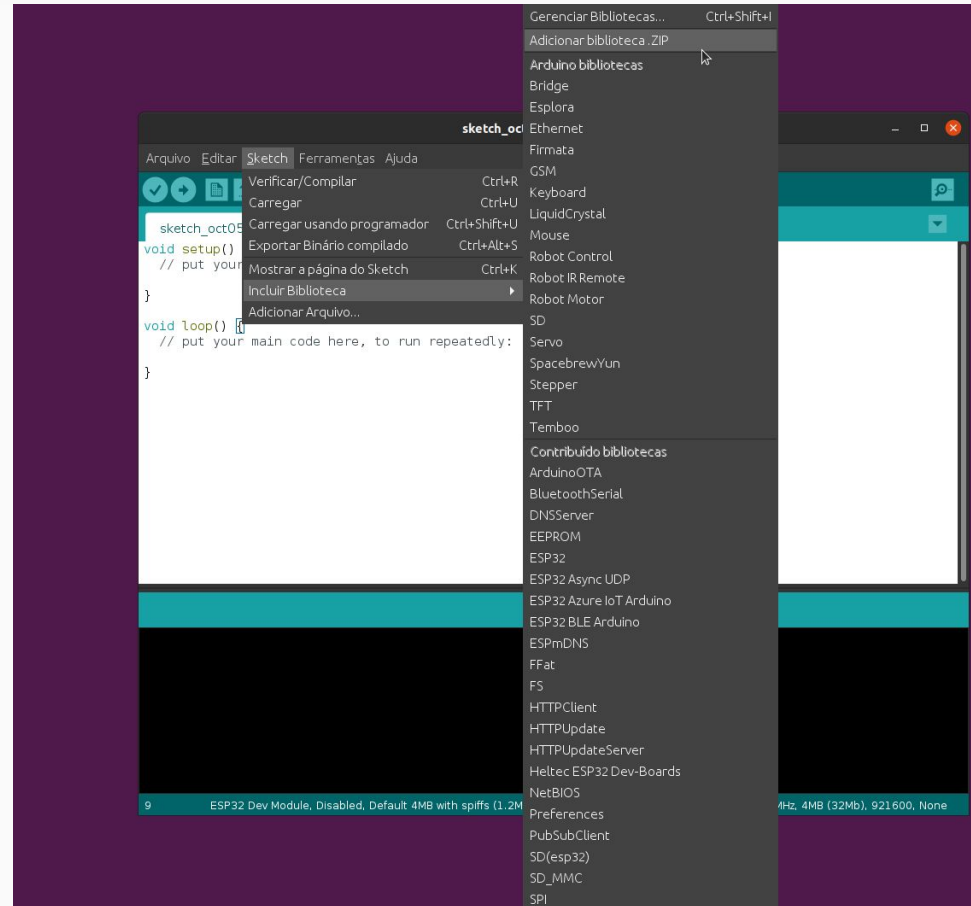
Assets 2

- Source code (zip)
- Source code (tar.gz)

v2.7

knolleary released this on 2 Nov 2018

<https://github.com/knolleary/pubsubclient>



Modo Station + Cloud + App

CLOUD

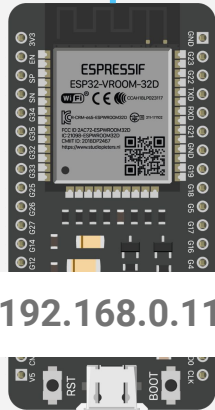
WIFI MODE
STATION

CLOUD



ARDUINO
IDE

millis();



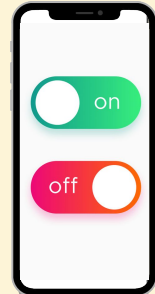
DISPOSITIVO

MQTT Client



MQTT Dash

Control the Things via MQTT protocol



CLIENTE



```
#include <WiFi.h>
#include <PubSubClient.h>
#include <ESPmDNS.h>

//Net Setup
#define NET_SSID "LabIoT"
#define NET_PASSWORD "XXXXXXXX"

//MQTT Setup
#define MQTT_ID "luiz-esp32-0800313131"
#define MQTT_BROKER "broker.hivemq.com"
#define MQTT_PORT 1883
#define MQTT_MILLIS_TOPIC "ifrj_cnit_luizfelipe_millis"
WiFiClient espClient; //Cliente de rede
PubSubClient MQTT(espClient); //Cliente MQTT

char millis_str[10] = "";
```

```
void setupWifi () {
  //Configura a conexão à rede sem fio
  if (WiFi.status () == WL_CONNECTED)
    return;
  Serial.println ();
  Serial.print ("Connecting to ");
  Serial.println (NET_SSID);

  WiFi.begin (NET_SSID, NET_PASSWORD );

  while (WiFi.status () != WL_CONNECTED) {
    delay (500);
    Serial.print (".");
  }

  Serial.println ("");
  Serial.println ("WiFi connected");
  Serial.println ("IP address: ");
  Serial.println (WiFi.localIP ());
}
```

```
void setupMQTT () {  
    MQTT.setServer (MQTT_BROKER, MQTT_PORT ); //informa qual broker e porta deve ser  
conectado  
  
    while (!MQTT.connected())  
    {  
        Serial.print ("*Tentando se conectar ao Broker MQTT: " );  
        Serial.println (MQTT_BROKER);  
        if (MQTT.connect (MQTT_ID))  
        {  
            Serial.println ("Conectado com sucesso ao broker MQTT!" );  
        }  
        else  
        {  
            Serial.println ("Falha ao reconectar no broker.");  
            Serial.println ("Havera nova tentativa de conexao em 2s" );  
            delay (2000);  
        }  
    }  
}
```



```
void setup(void) {  
  //Configura o baudrate da comunicação serial  
  Serial.begin(115200);  
  
  setupWifi();  
  
  setupMQTT();  
  
}
```

```
void loop(void) {  
  
    sprintf(millis_str, "%d", millis());  
  
    MQTT.publish(MQTT_MILLIS_TOPIC, millis_str);  
  
    setupWifi();  
    setupMQTT();  
    delay(2000);  
}
```

```
int sprintf ( char * str, const char * format, values );
```

corresponding argument:

specifier	Output	Example
d or i	Signed decimal integer	392
u	Unsigned decimal integer	7235
o	Unsigned octal	610
x	Unsigned hexadecimal integer	7fa
X	Unsigned hexadecimal integer (uppercase)	7FA
f	Decimal floating point, lowercase	392.65
F	Decimal floating point, uppercase	392.65
e	Scientific notation (mantissa/exponent), lowercase	3.9265e+2

Composes a string with the same text that would be printed if *format* was used on [printf](#), but instead of being printed, the content is stored as a *C string* in the buffer pointed by *str*.

corresponding argument:

specifier	Output	Example
d or i	Signed decimal integer	392
u	Unsigned decimal integer	7235
o	Unsigned octal	610
x	Unsigned hexadecimal integer	7fa
X	Unsigned hexadecimal integer (uppercase)	7FA
f	Decimal floating point, lowercase	392.65
F	Decimal floating point, uppercase	392.65
e	Scientific notation (mantissa/exponent), lowercase	3.9265e+2
E	Scientific notation (mantissa/exponent), uppercase	3.9265E+2
g	Use the shortest representation: %e or %f	392.65
G	Use the shortest representation: %E or %F	392.65
a	Hexadecimal floating point, lowercase	-0xc.90fep-2
A	Hexadecimal floating point, uppercase	-0XC.90FEP-2
c	Character	a
s	String of characters	sample
p	Pointer address	b8000000
n	Nothing printed. The corresponding argument must be a pointer to a signed int. The number of characters written so far is stored in the pointed location.	
%	A % followed by another % character will write a single % to the stream.	%

