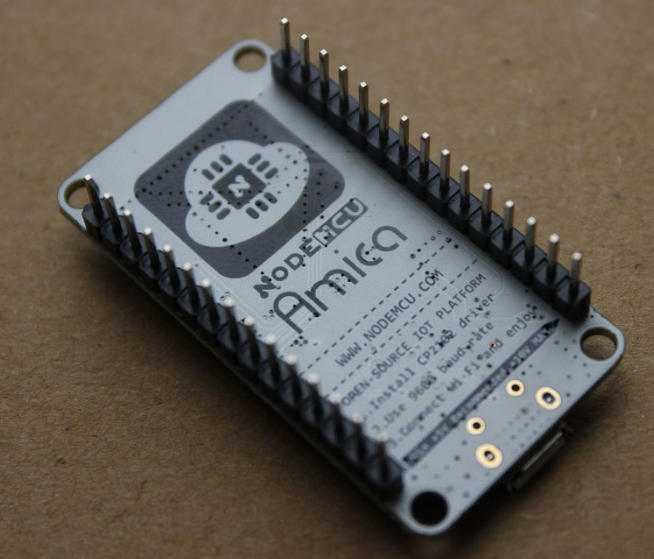
Using NodeMCU (ESP 8266) with **Microsoft Azure**

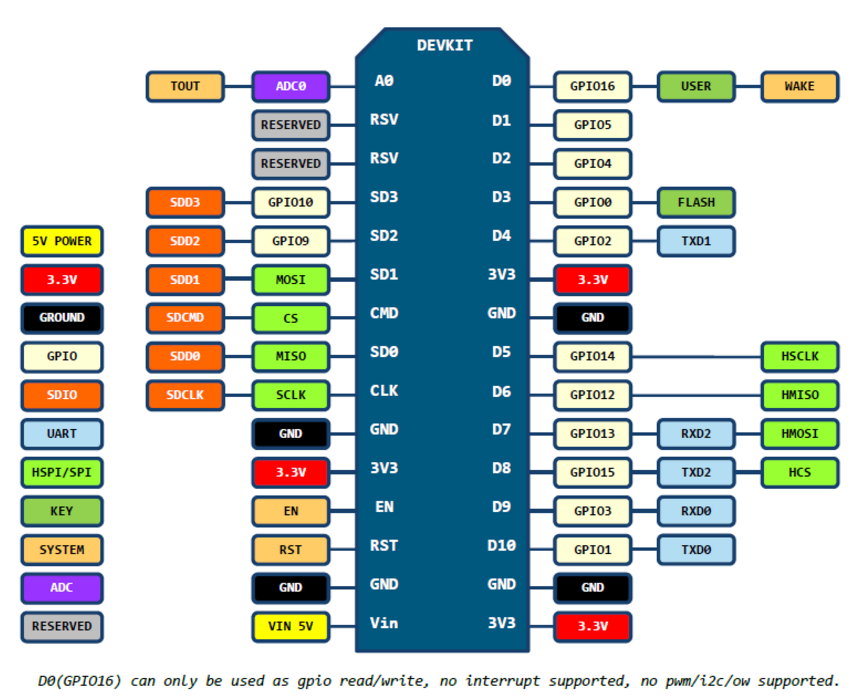
* ESP 8266 Hardware
* Configure Arduino IDE for ESP 8266
* ESP 8266 Hello World
* Programing ESP 8266 in Visual Studio
* MQTT Protocol with ESP 8266
* MQTT in C#
* Install ESP 8266 MQTT Lib
* D2C & C2D using ESP 8266

HARDWARE

* The ESP8266 is a low-cost Wi-Fi chip with full TCP/IP stack and MCU (Micro Controller Unit) capability
* **NodeMCU** V2 LUA based ESP8266-12E Development Kit

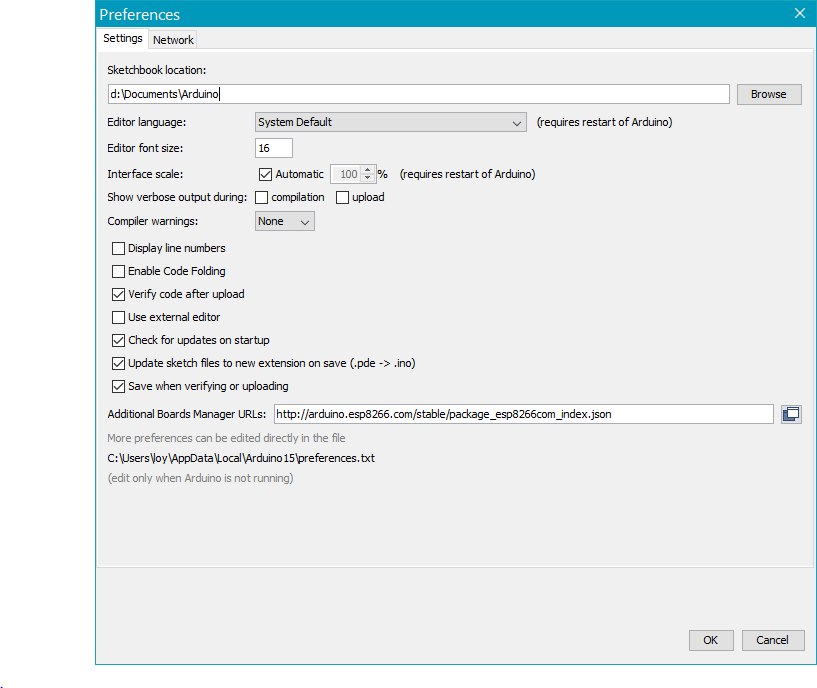


NodeMCU V2 pin definition

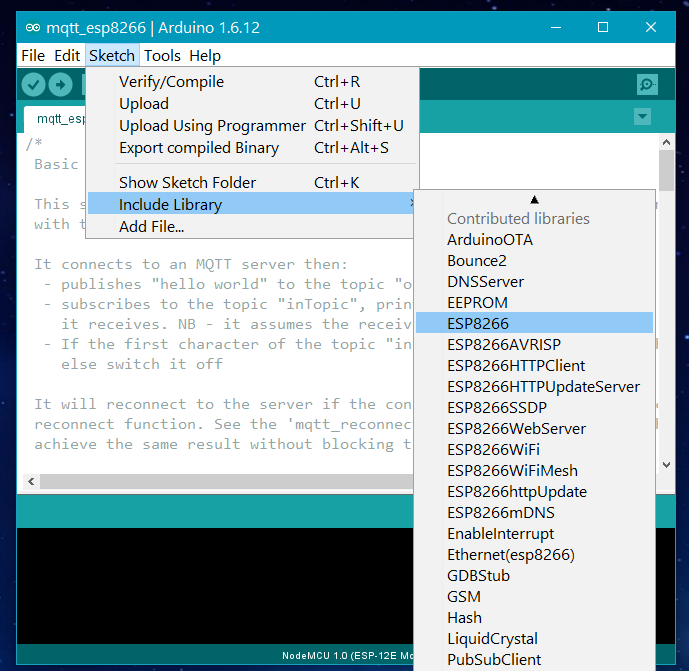


Configure Arduino IDE

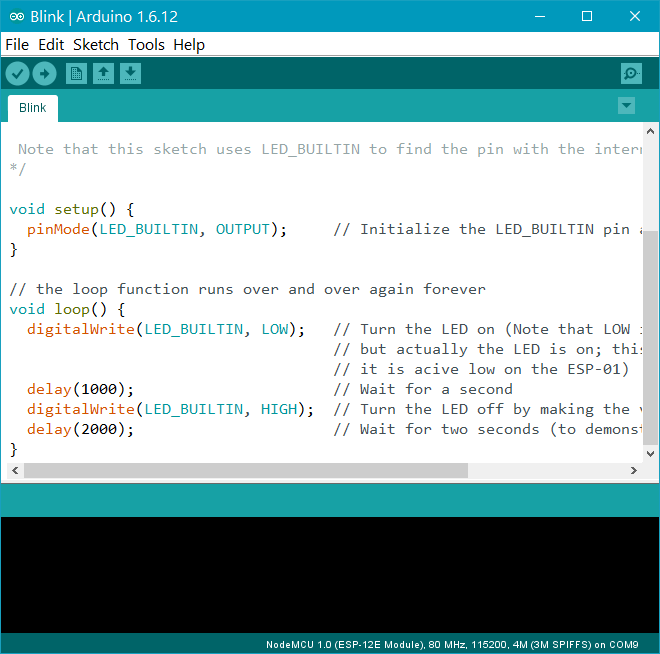
http://arduino.esp8266.com/stable/package\_esp8266com\_index.json



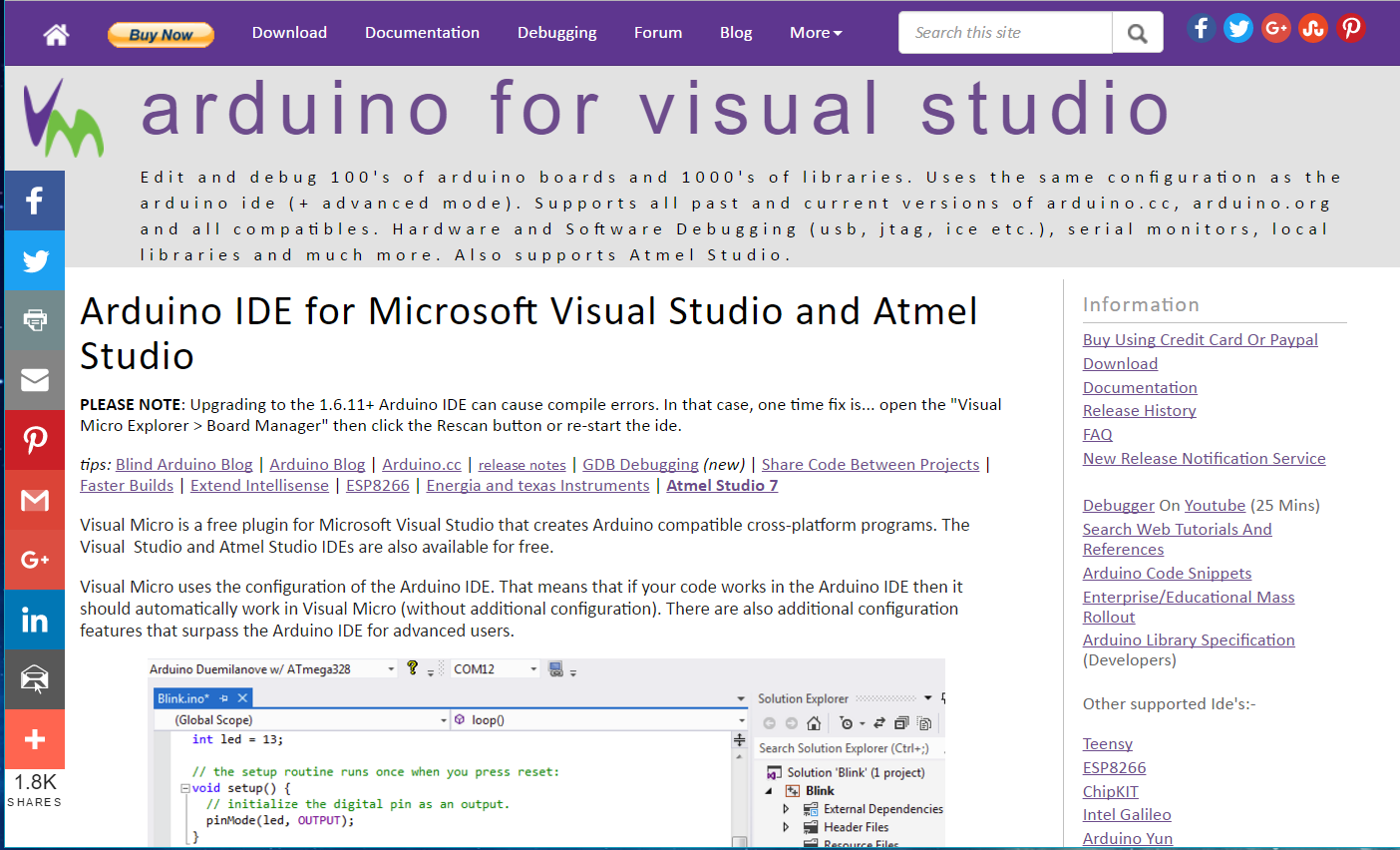
INCLUDE LIBRARY



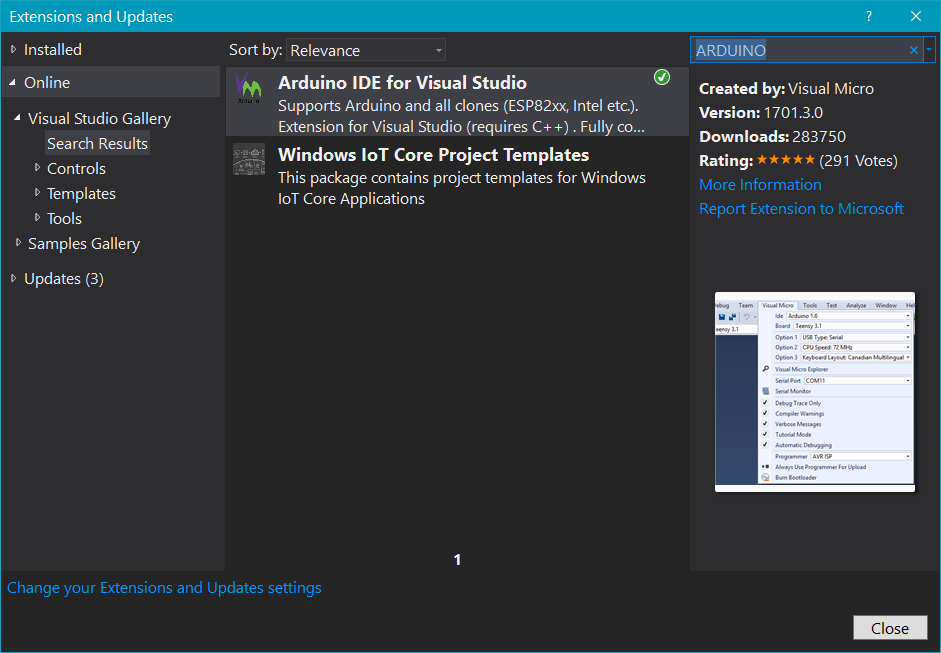
Hello World



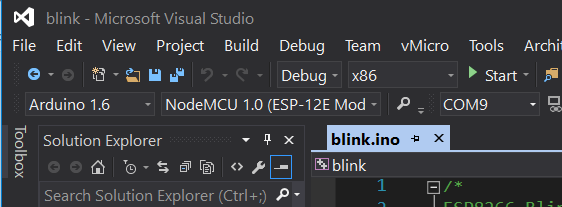
Programing in Visual Studio



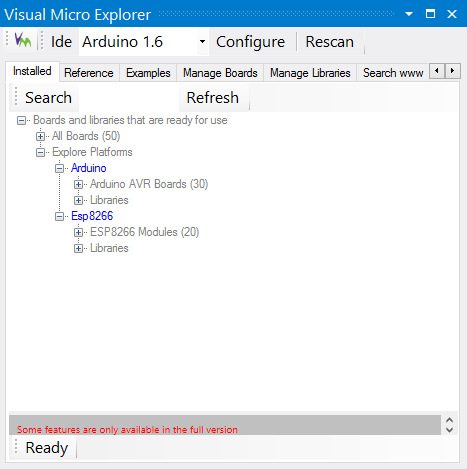
Extensions and Updates



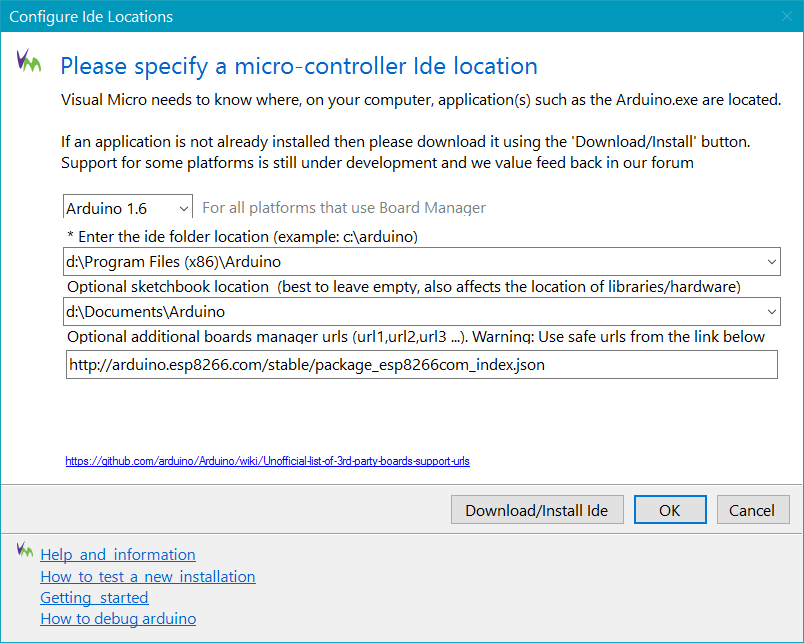
Open Visual Micro Explorer



Visual Micro Explorer Configure



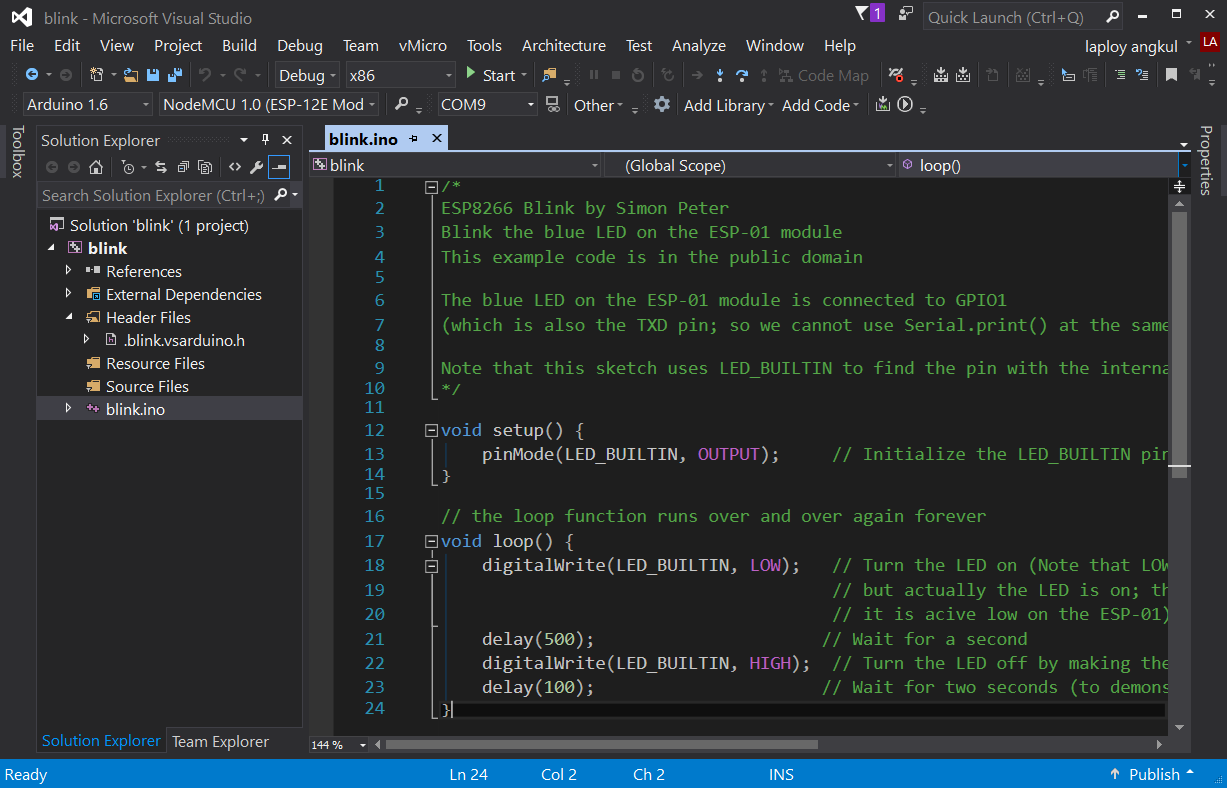
Configure Ide Locations



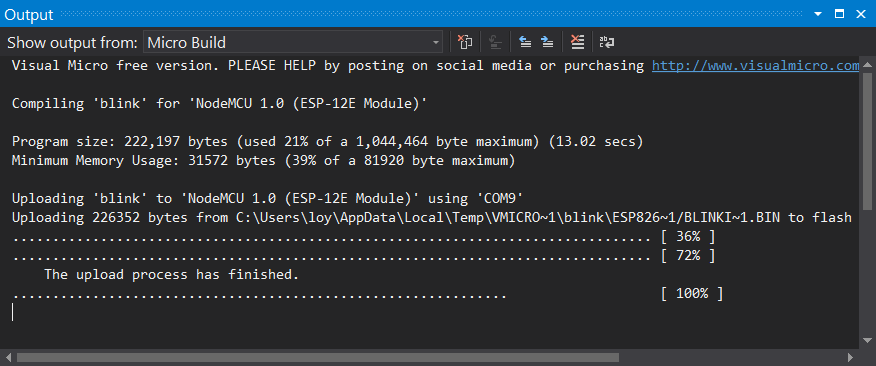
ESP 8266 PROGRAMMING IN VISUAL STUDIO

* Blinky
* Hello World!
* Read temperature censor
* Wi-Fi connection test
* MQTT D2C test
* MQTT C2D test
* MQTT D2C/C2D

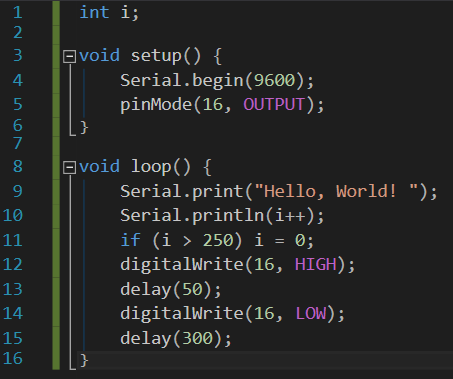
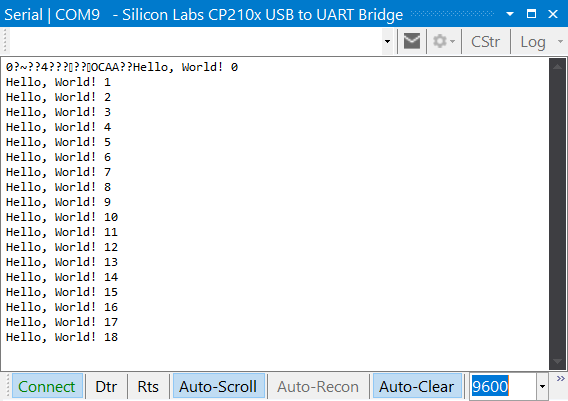
Blinky



COMPILE AND UPOLAD

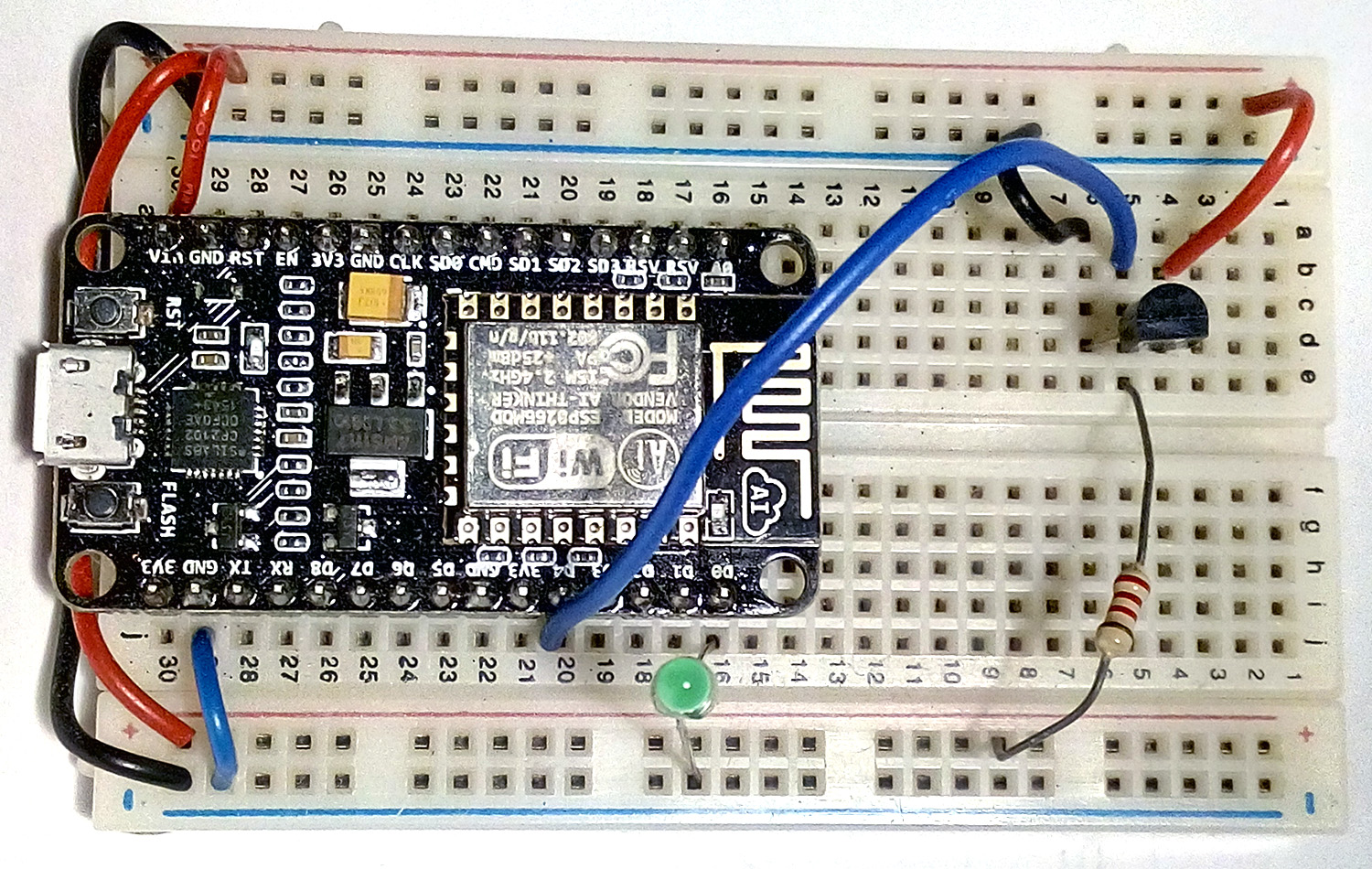


Hello World!



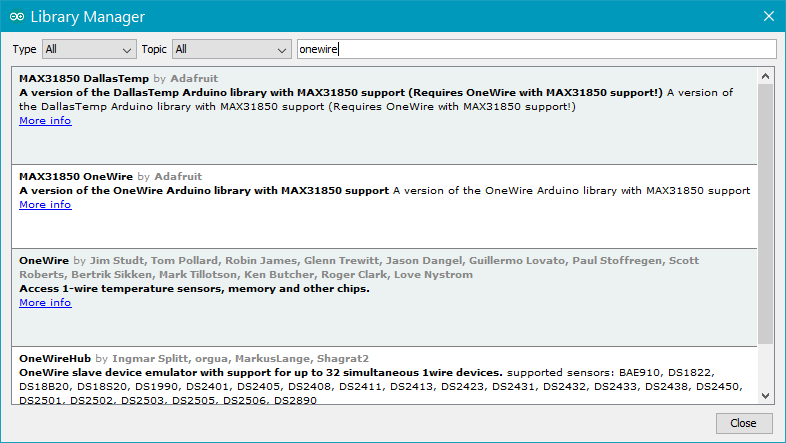
TEMPERATURE READING

Hardware setup



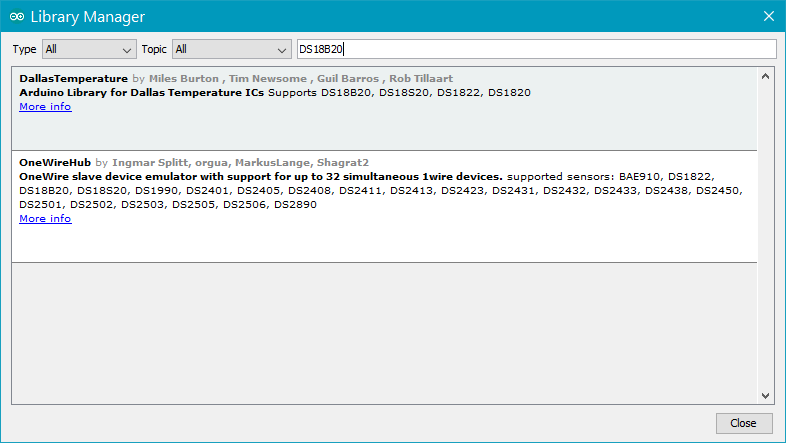
TEMPERATURE READING

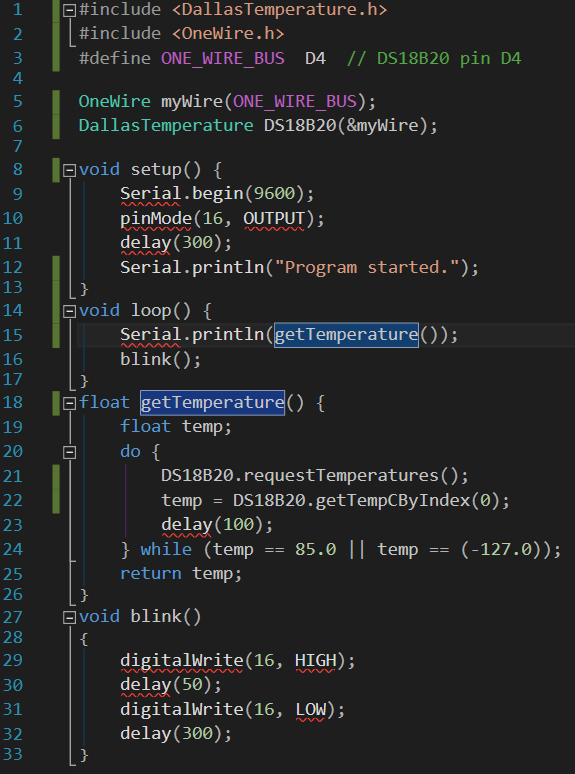
Add OneWire Lib



TEMPERATURE READING

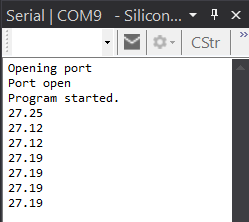
Add DallasTemperature Lib

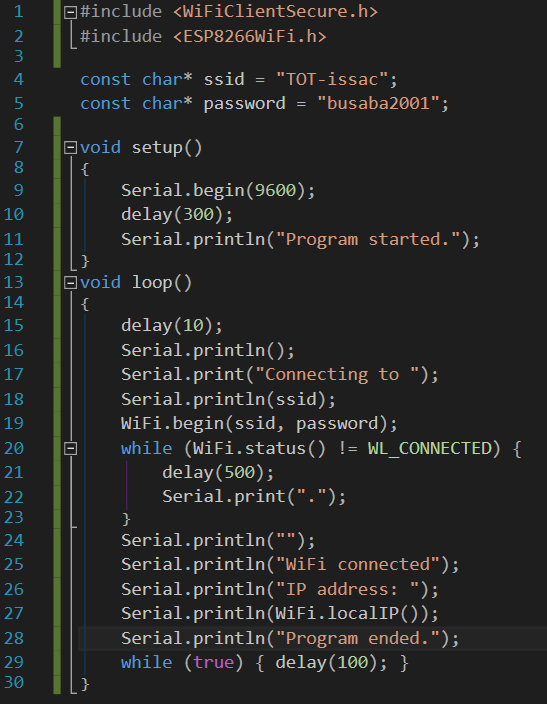


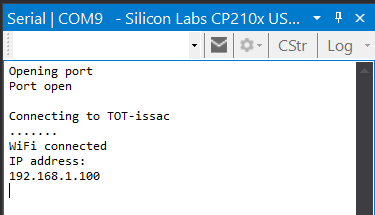
TEMPERATURE

READING

Source code



Wi-Fi Test



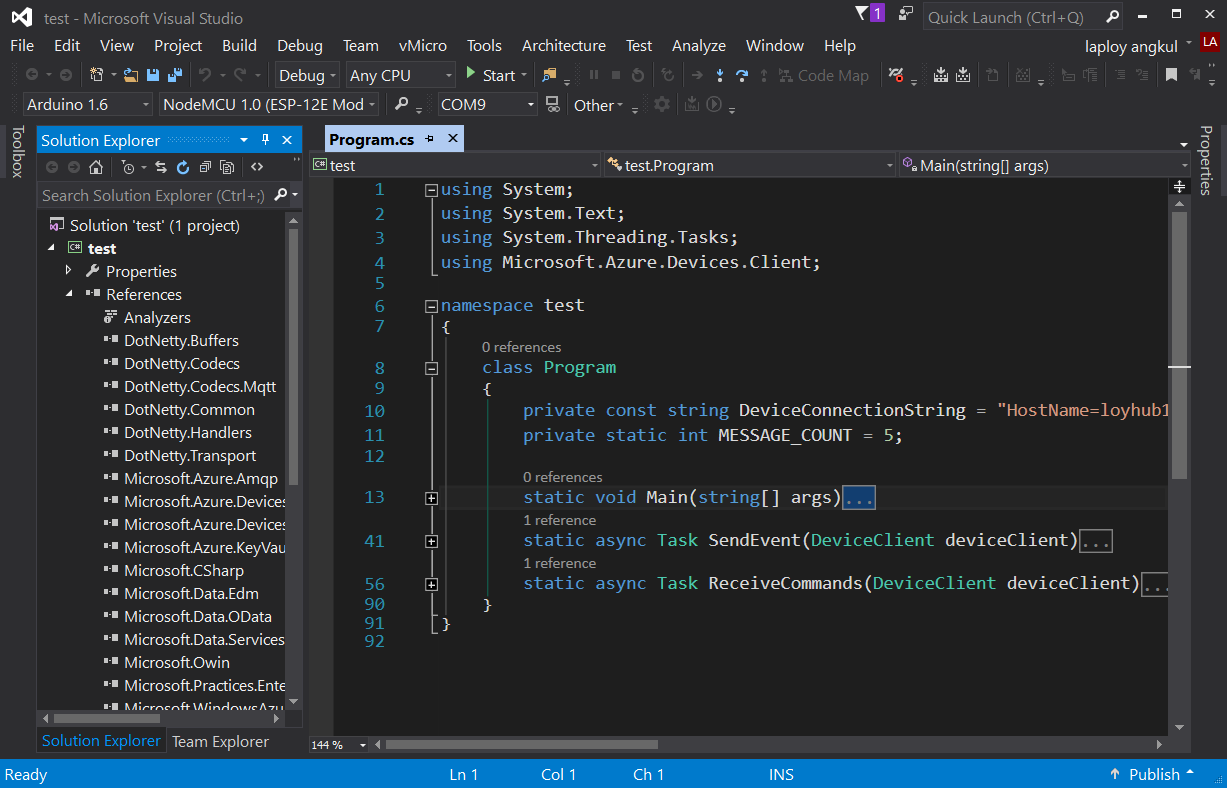
MQTT PROTOCOL

Message Queue Telemetry Transport



* Much more simple and focused than those of AMQP
* Provides publish-and-subscribe messaging (no queues)
* Specifically designed for resource-constrained devices
* Low bandwidth, high latency networks such as dial up lines and satellite links
* Used effectively in embedded systems.

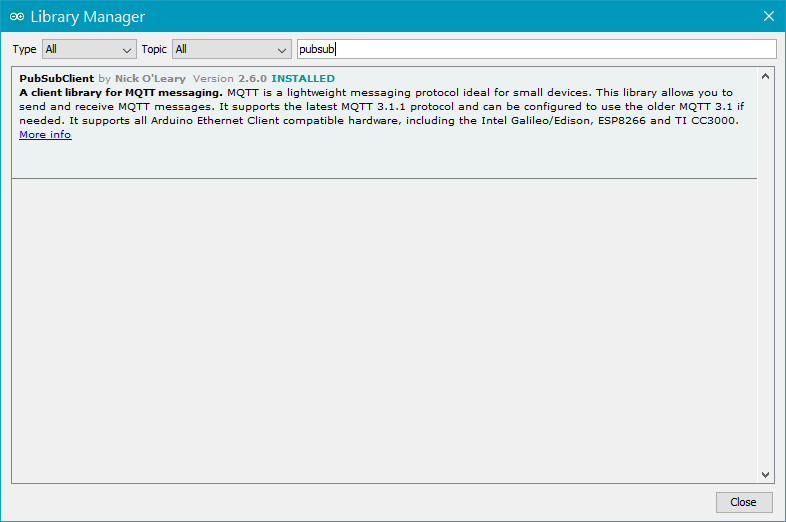
MQTT in C#



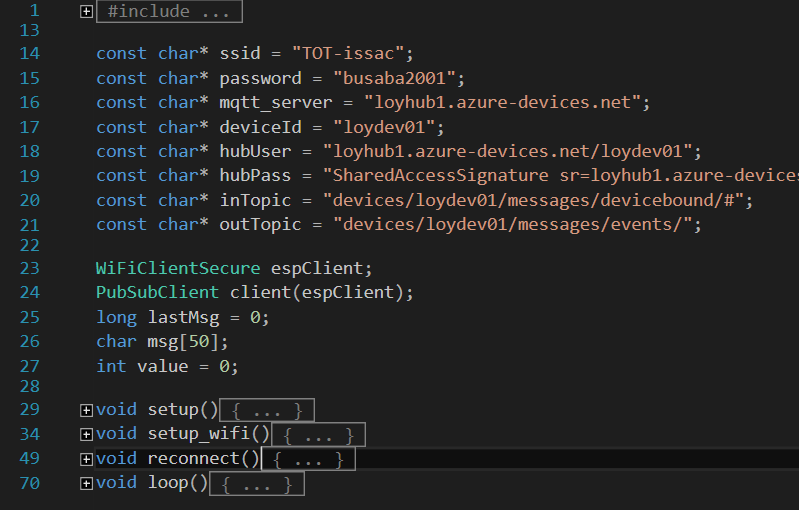
Install ESP 8266 MQTT Lib

PubSubClient by Nick O'Leary Version 2.6.0

A client library for MQTT messaging.



DEVICE TO CLOUD



CONSTANCE

const char\* ssid = "xxxxxxx";

const char\* password = "xxxxxx";

const char\* mqtt\_server = "loyhub1.azure-devices.net";

const char\* deviceId = "loydev01";

const char\* hubUser = "loyhub1.azure-devices.net/loydev01";

const char\* hubPass = "SharedAccessSignature sr=loyhub1.azure-devices.net%2Fdevices%2Floydev01&sig=pZCYksE6NDkmftmOnJ0PeFqj1Wc9IS4%2FW2OnTvdkbno%3D&se=1515634239";

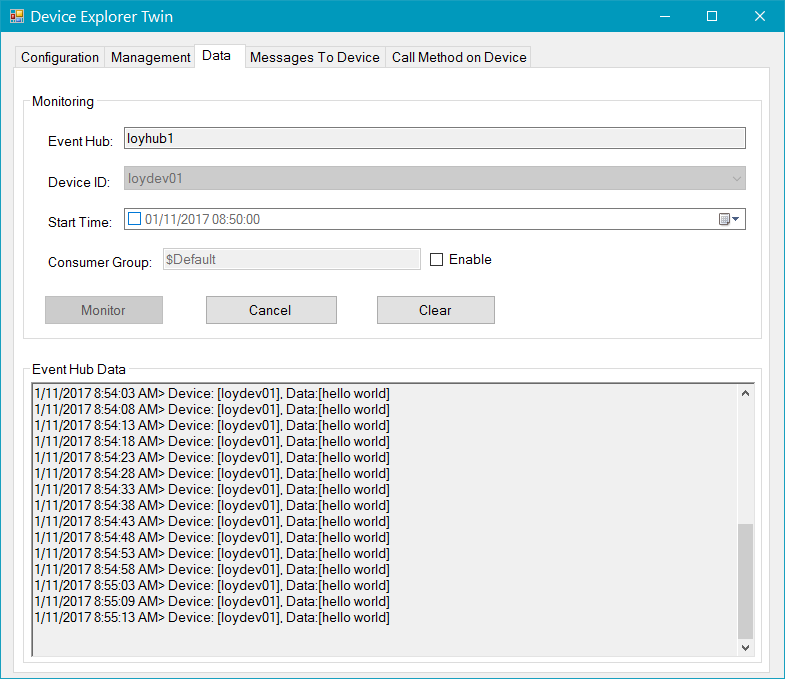
const char\* inTopic = "devices/loydev01/messages/devicebound/#";

const char\* outTopic = "devices/loydev01/messages/events/";

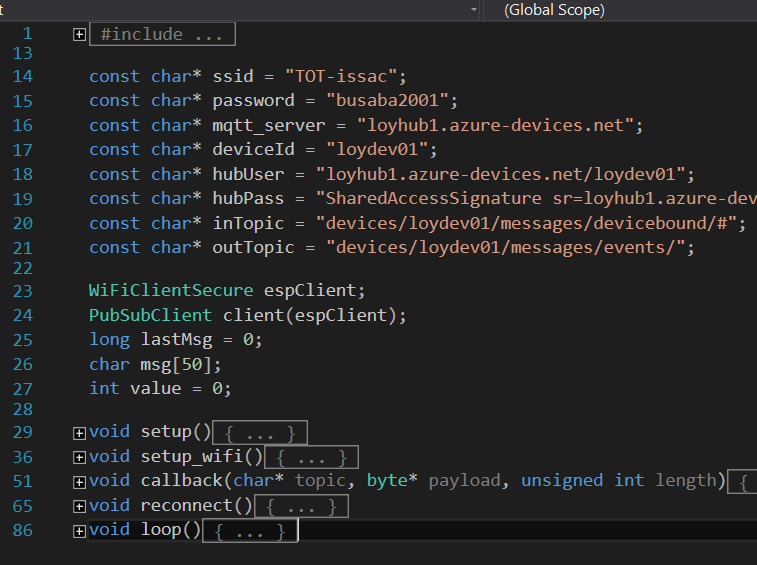
GET HUB PASS

1. Go to Device Explorer
2. Click Management Tab
3. Click device
4. Click SAS Token
5. Set the number of day = 365
6. Click Generate
7. Select part "SharedAccessSignature sr=loyhub1.azure-devices.net%2Fdevices%2Floydev01&sig=pZCYksE6NDkmftmOnJ0PeFqj1Wc9IS4%2FW2OnTvdkbno%3D&se=1515634239";

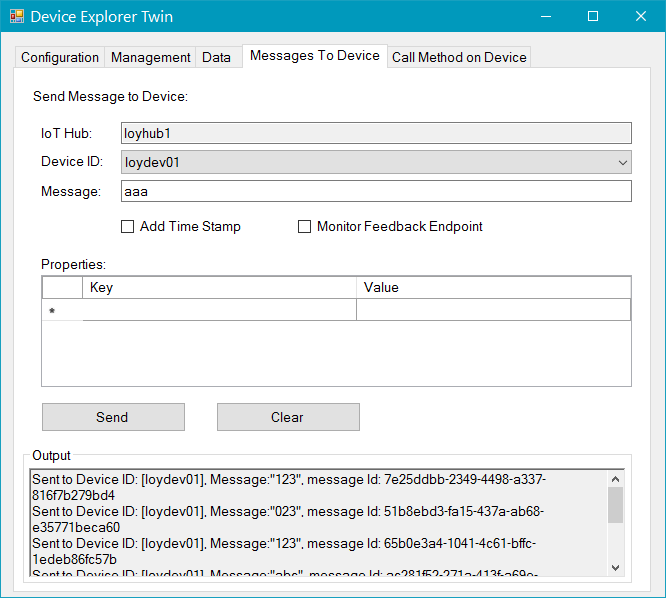
DEVICE EXPLOROR SHOW D2C MESSAGE



CLOUD TO DEVICE



DEVICE EXPLOROR SEND C2D MESSAGE



MQTT D2C /C2D

