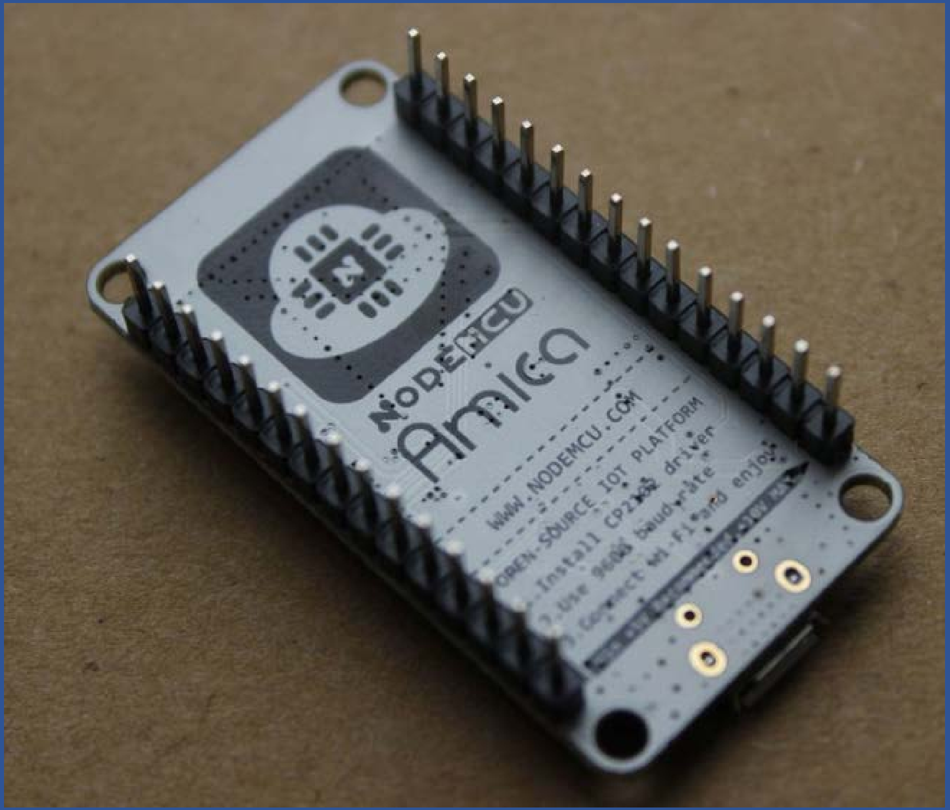
Using NodeMCU (ESP 8266) with Microsoft Azure

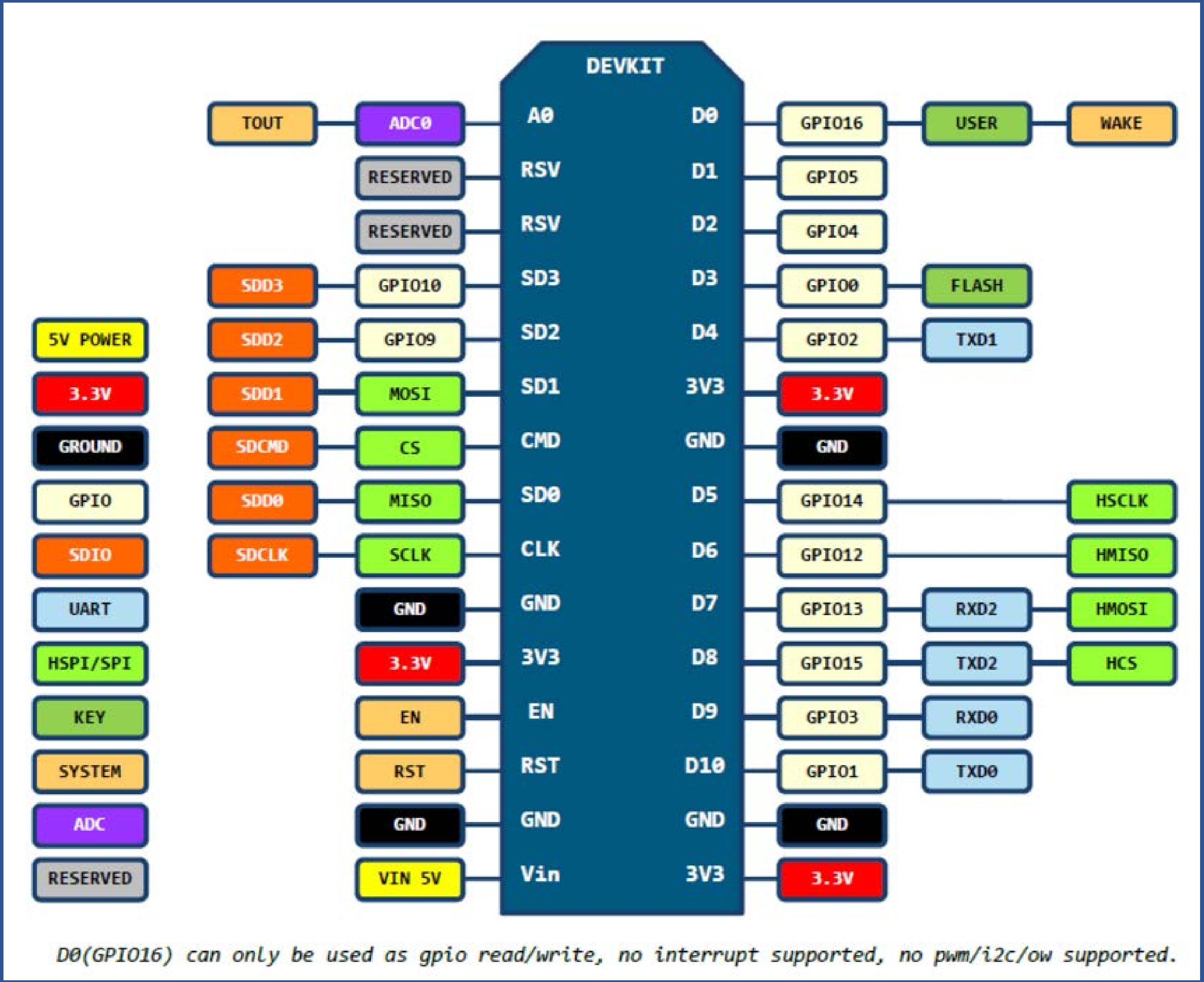
- ESP 8266 Hardware
- Configure Arduino IDE for ESP 8266
- ESP 8266 Hello World
- Programing ESP 8266 in Visual Studio
- MQTT Protocol with ESP 8266
- MQTT in C#
- Install ESP 8266 MQTT Lib
- D2C & C2D using ESP 8266

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

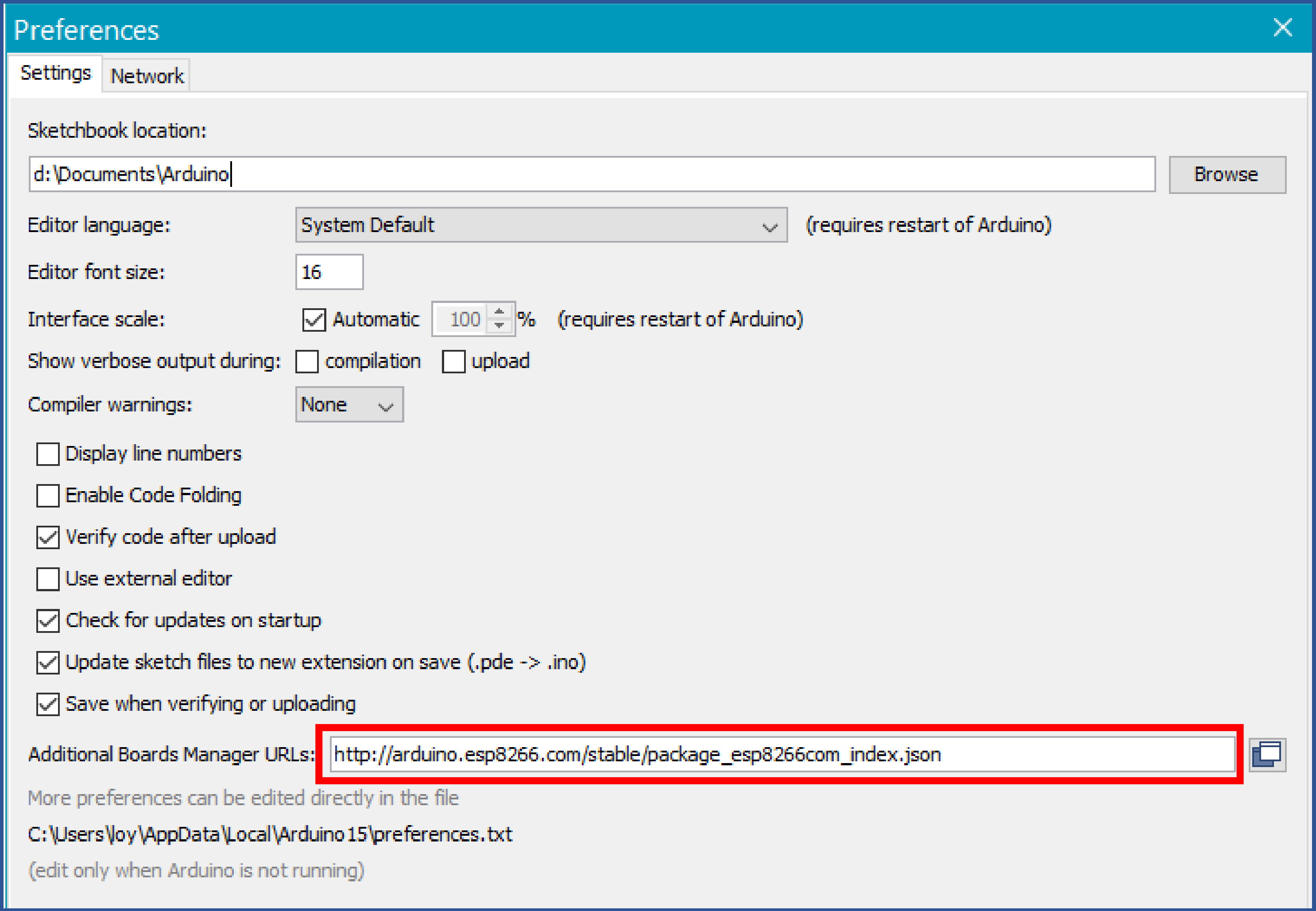


NodeMCU V2 pin definition



Configure Arduino IDE

http://arduino.esp8266.com/stable/package_esp8266com_index.json



Preferences

Sketchbook location: d:\Documents\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 16

Interface scale: [✓ Automatic] 100 % (requires restart of Arduino)

Show verbose output during: ☐ compilation ☐ upload

Compiler warnings: None

☐ Display line numbers

☐ Enable Code Folding

☑ Verify code after upload

☐ Use external editor

☑ Check for updates on startup

☑ Update sketch files to new extension on save (.pde -> .ino)

☑ Save when verifying or uploading

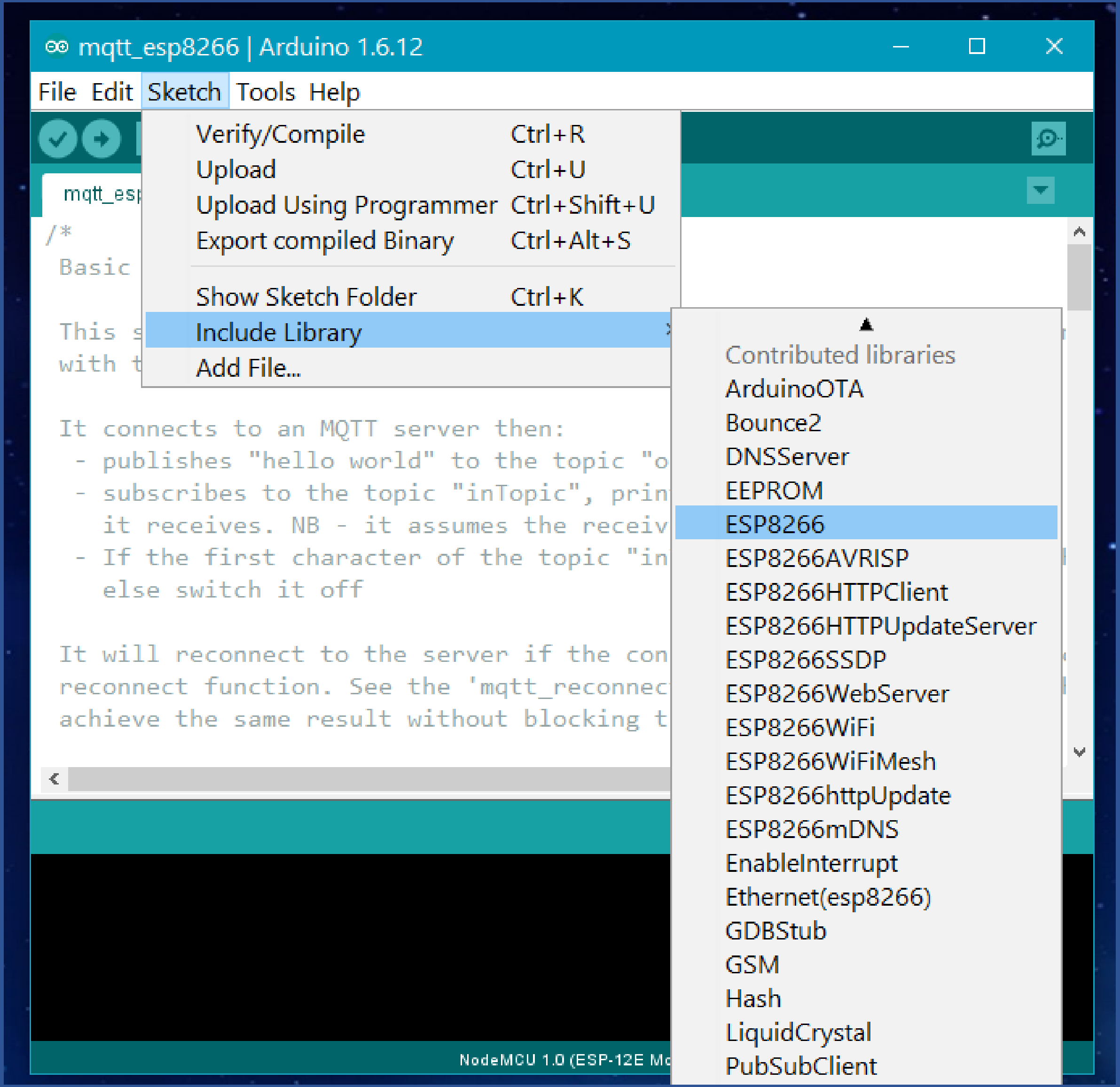
Additional Boards Manager URLs: http://arduino.esp8266.com/stable/package_esp8266com_index.json

More preferences can be edited directly in the file

C:\Users\loy\AppData\Local\Arduino15\preferences.txt

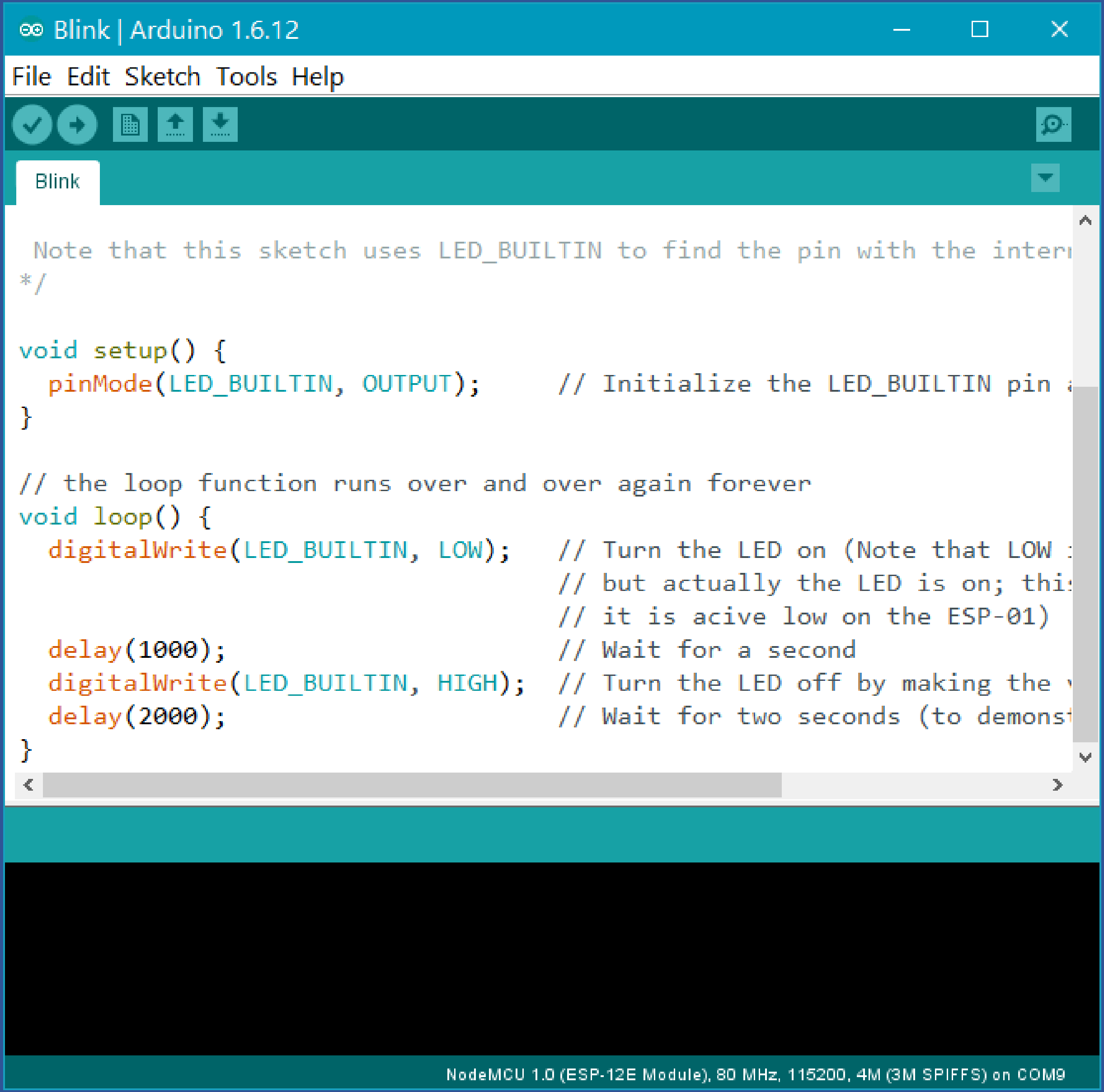
(edit only when Arduino is not running)

INCLUDE LIBRARY



```
/*
 Basic

 This s
 with t

 It connects to an MQTT server then:
  - publishes "hello world" to the topic "o
  - subscribes to the topic "inTopic", prin
    it receives. NB - it assumes the receiv
  - If the first character of the topic "in
    else switch it off

 It will reconnect to the server if the con
 reconnect function. See the 'mqtt_reconnec
 achieve the same result without blocking t
```

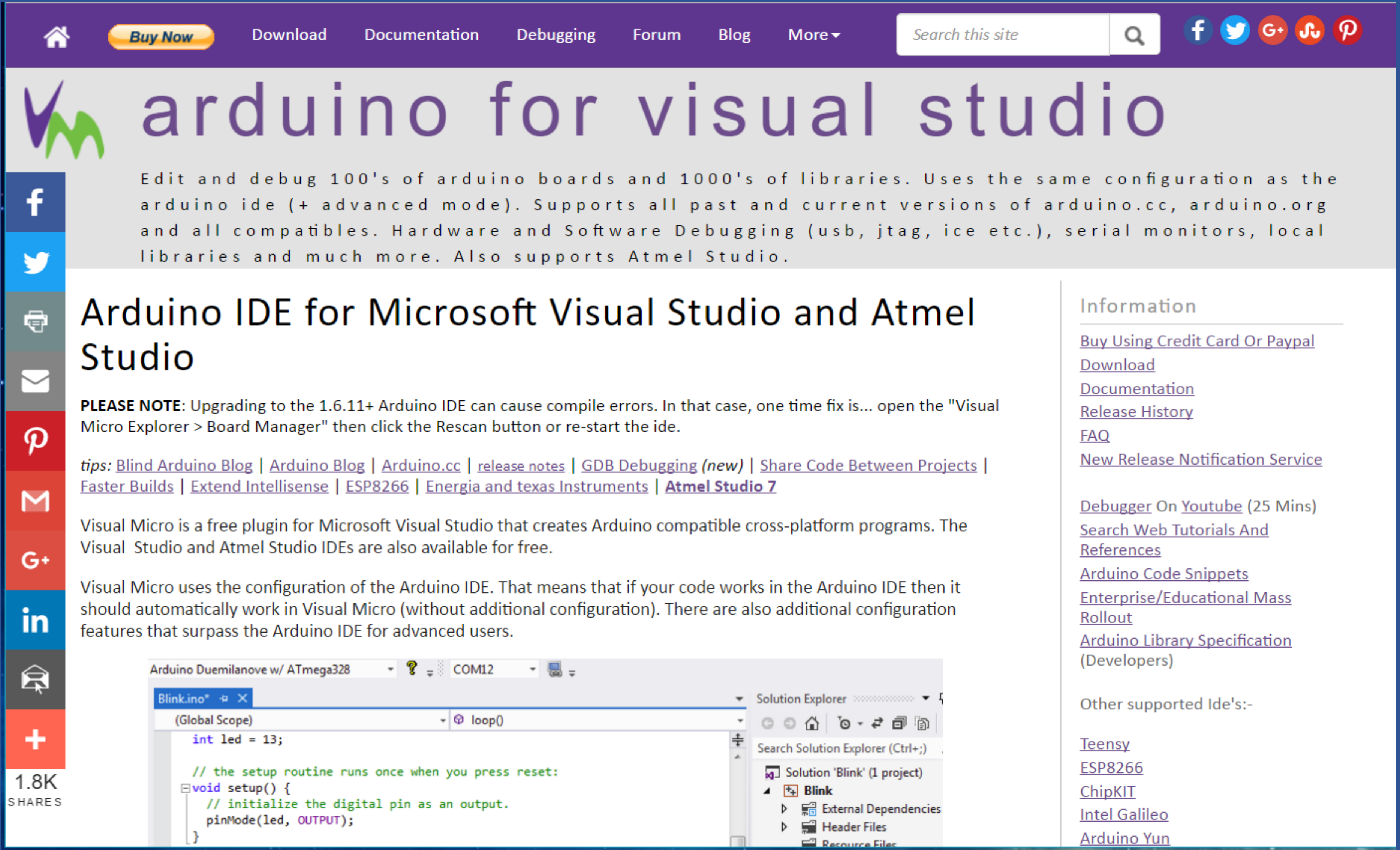
Hello World



```
 Note that this sketch uses LED_BUILTIN to find the pin with the intern
*/

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);     // Initialize the LED_BUILTIN pin a
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, LOW);   // Turn the LED on (Note that LOW i
                                    // but actually the LED is on; this
                                    // it is acive low on the ESP-01)
  delay(1000);                      // Wait for a second
  digitalWrite(LED_BUILTIN, HIGH);  // Turn the LED off by making the v
  delay(2000);                      // Wait for two seconds (to demonst
}
```

NodeMCU 1.0 (ESP-12E Module), 80 MHz, 115200, 4M (3M SPIFFS) on COM9

Programing in Visual Studio



arduino for visual studio

Edit and debug 100's of arduino boards and 1000's of libraries. Uses the same configuration as the arduino ide (+ advanced mode). Supports all past and current versions of arduino.cc, arduino.org and all compatibles. Hardware and Software Debugging (usb, jtag, ice etc.), serial monitors, local libraries and much more. Also supports Atmel Studio.

Arduino IDE for Microsoft Visual Studio and Atmel Studio

PLEASE NOTE: Upgrading to the 1.6.11+ Arduino IDE can cause compile errors. In that case, one time fix is... open the "Visual Micro Explorer > Board Manager" then click the Rescan button or re-start the ide.

tips: Blind Arduino Blog | Arduino Blog | Arduino.cc | release notes | GDB Debugging *(new)* | Share Code Between Projects | Faster Builds | Extend Intellisense | ESP8266 | Energia and texas Instruments | **Atmel Studio 7**

Visual Micro is a free plugin for Microsoft Visual Studio that creates Arduino compatible cross-platform programs. The Visual Studio and Atmel Studio IDEs are also available for free.

Visual Micro uses the configuration of the Arduino IDE. That means that if your code works in the Arduino IDE then it should automatically work in Visual Micro (without additional configuration). There are also additional configuration features that surpass the Arduino IDE for advanced users.

```
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}
```

Information

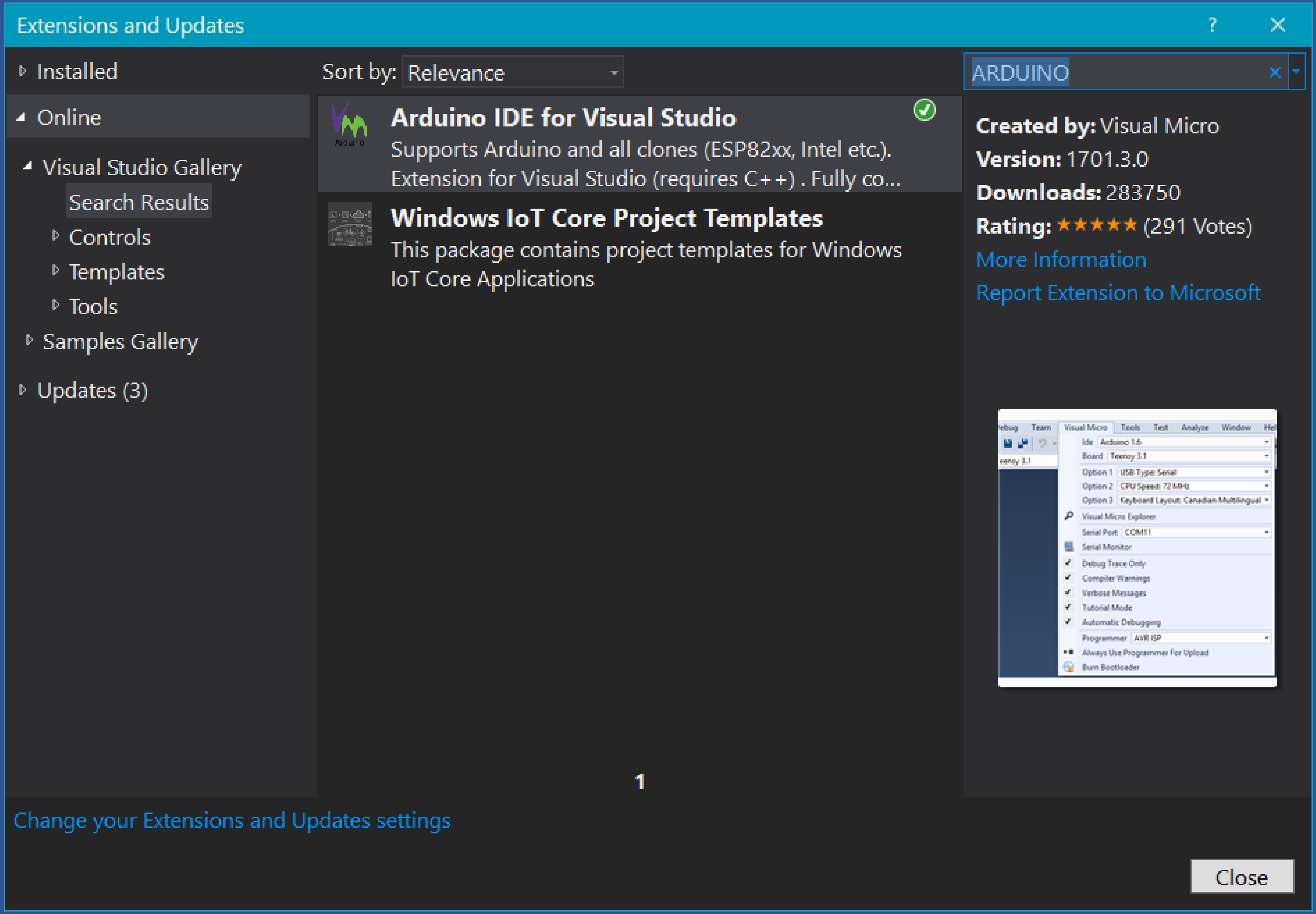
Buy Using Credit Card Or Paypal
Download
Documentation
Release History
FAQ
New Release Notification Service

Debugger On Youtube (25 Mins)
Search Web Tutorials And References
Arduino Code Snippets
Enterprise/Educational Mass Rollout
Arduino Library Specification (Developers)

Other supported Ide's:-

Teensy
ESP8266
ChipKIT
Intel Galileo
Arduino Yun

Extensions and Updates



Extensions and Updates

Installed
Online
Visual Studio Gallery
Search Results
Controls
Templates
Tools
Samples Gallery
Updates (3)

Sort by: Relevance

Arduino IDE for Visual Studio
Supports Arduino and all clones (ESP82xx, Intel etc.). Extension for Visual Studio (requires C++) . Fully co...

Windows IoT Core Project Templates
This package contains project templates for Windows IoT Core Applications

1

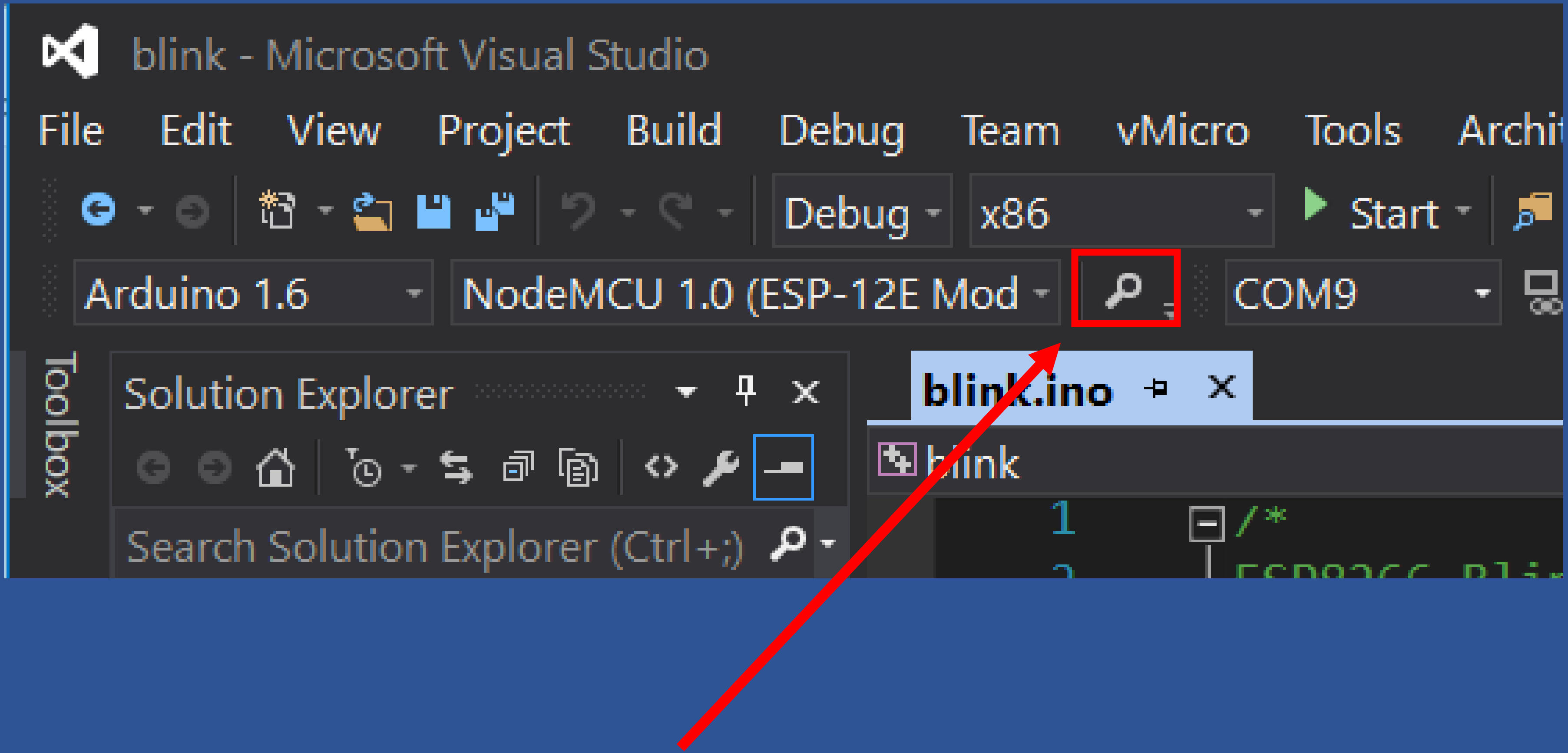
ARDUINO

Created by: Visual Micro
Version: 1701.3.0
Downloads: 283750
Rating: ★★★★★ (291 Votes)
More Information
Report Extension to Microsoft

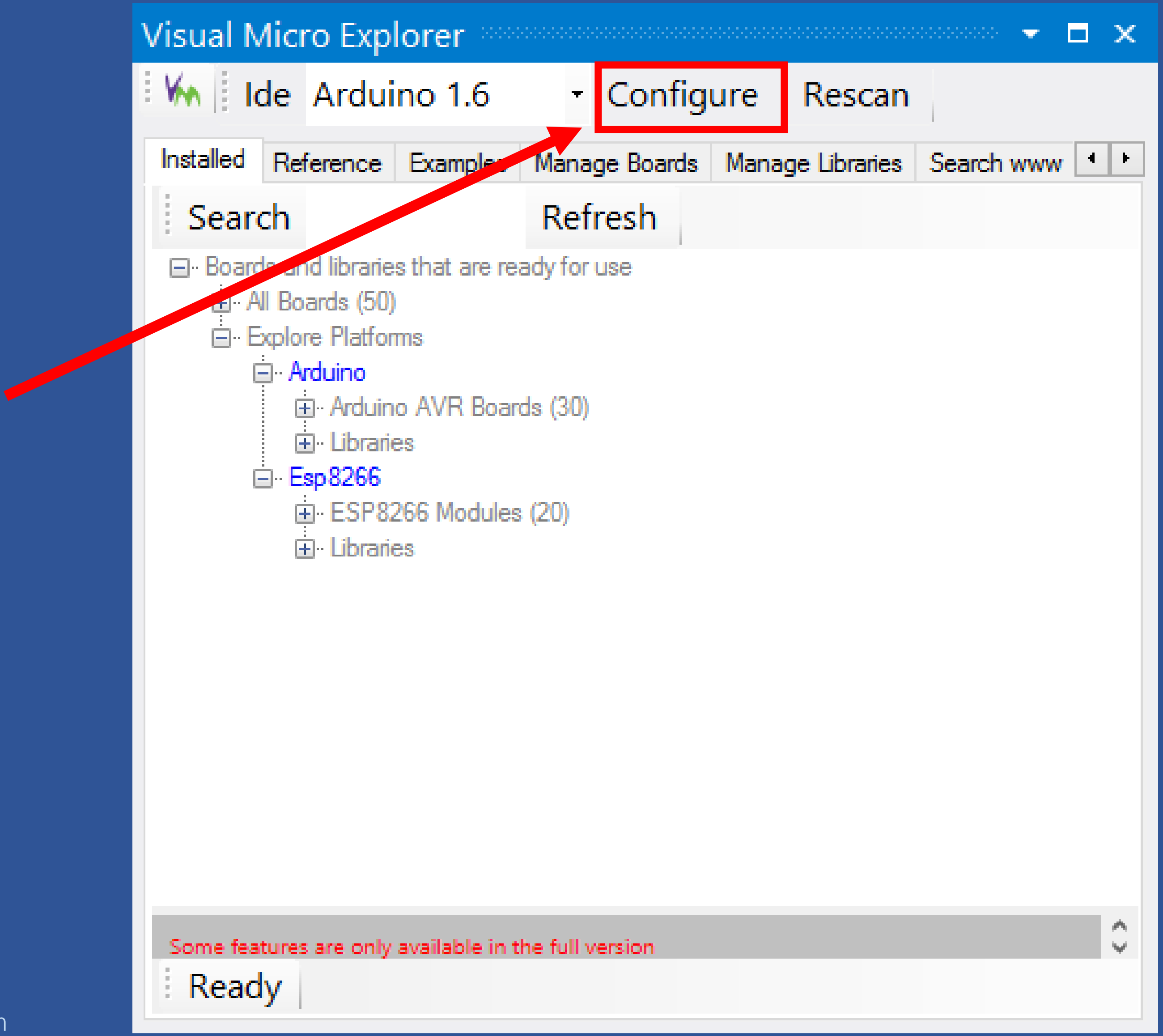
Change your Extensions and Updates settings

Close

Open Visual Micro Explorer



Visual Micro Explorer Configure

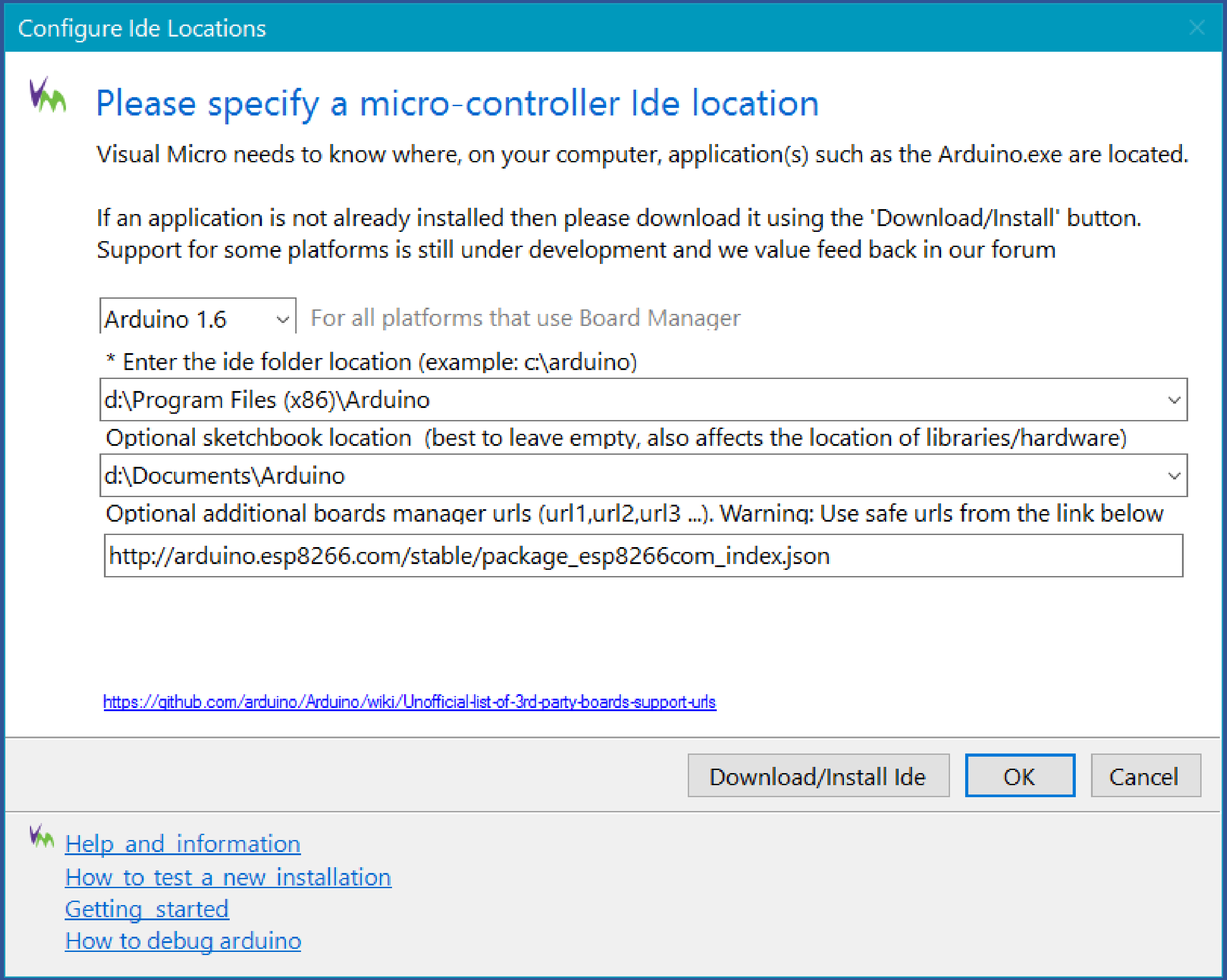


Boards and libraries that are ready for use
- All Boards (50)
- Explore Platforms
  - Arduino
    - Arduino AVR Boards (30)
    - Libraries
  - Esp8266
    - ESP8266 Modules (20)
    - Libraries

Some features are only available in the full version

Ready

Configure Ide Locations



Configure Ide Locations

Please specify a micro-controller Ide location

Visual Micro needs to know where, on your computer, application(s) such as the Arduino.exe are located.

If an application is not already installed then please download it using the 'Download/Install' button.
Support for some platforms is still under development and we value feed back in our forum

Arduino 1.6 For all platforms that use Board Manager

* Enter the ide folder location (example: c:\arduino)

d:\Program Files (x86)\Arduino

Optional sketchbook location (best to leave empty, also affects the location of libraries/hardware)

d:\Documents\Arduino

Optional additional boards manager urls (url1,url2,url3 ...). Warning: Use safe urls from the link below

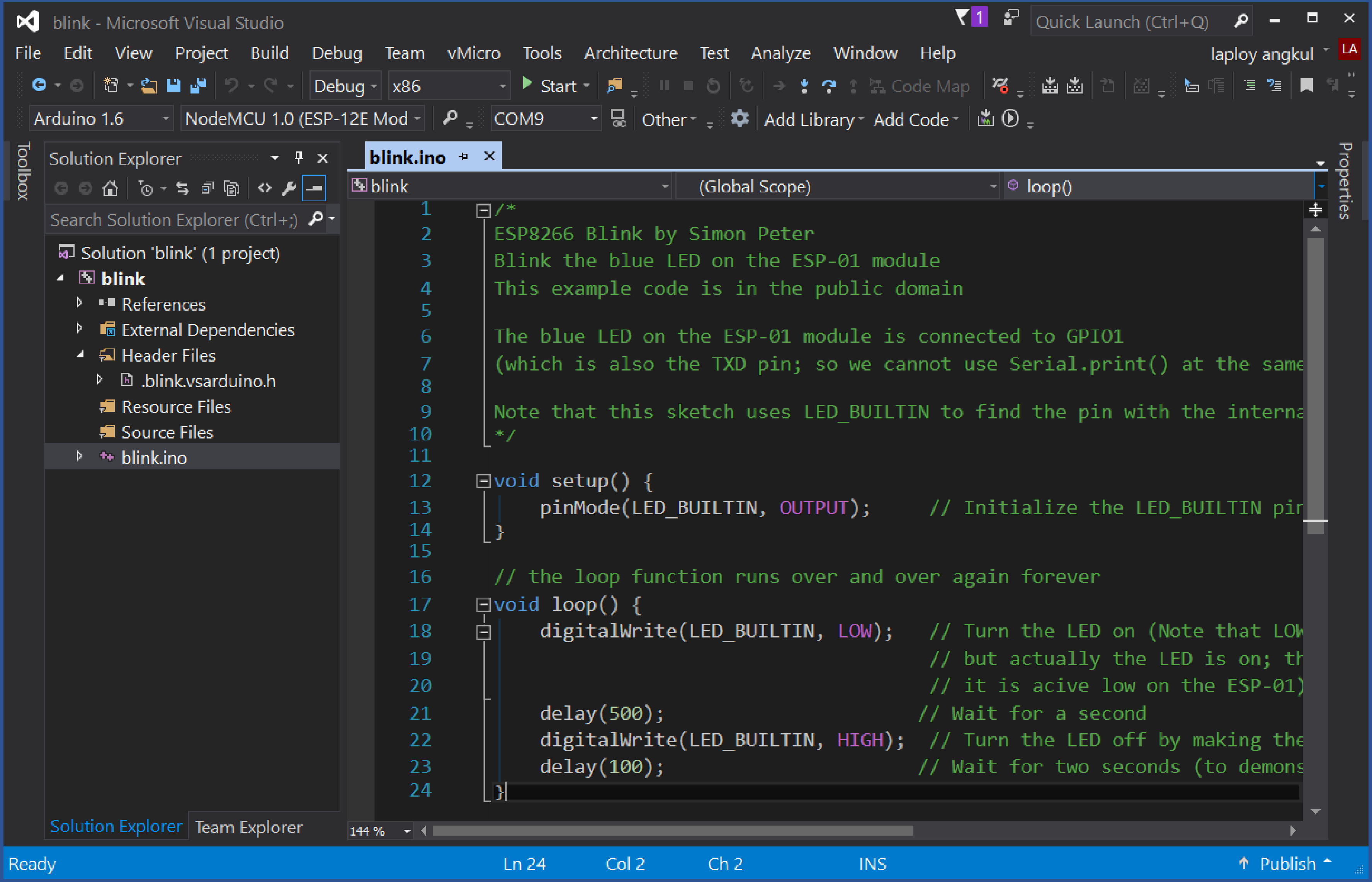
http://arduino.esp8266.com/stable/package_esp8266com_index.json

https://github.com/arduino/Arduino/wiki/Unofficial-list-of-3rd-party-boards-support-urls

ESP 8266 PROGRAMMING IN VISUAL STUDIO

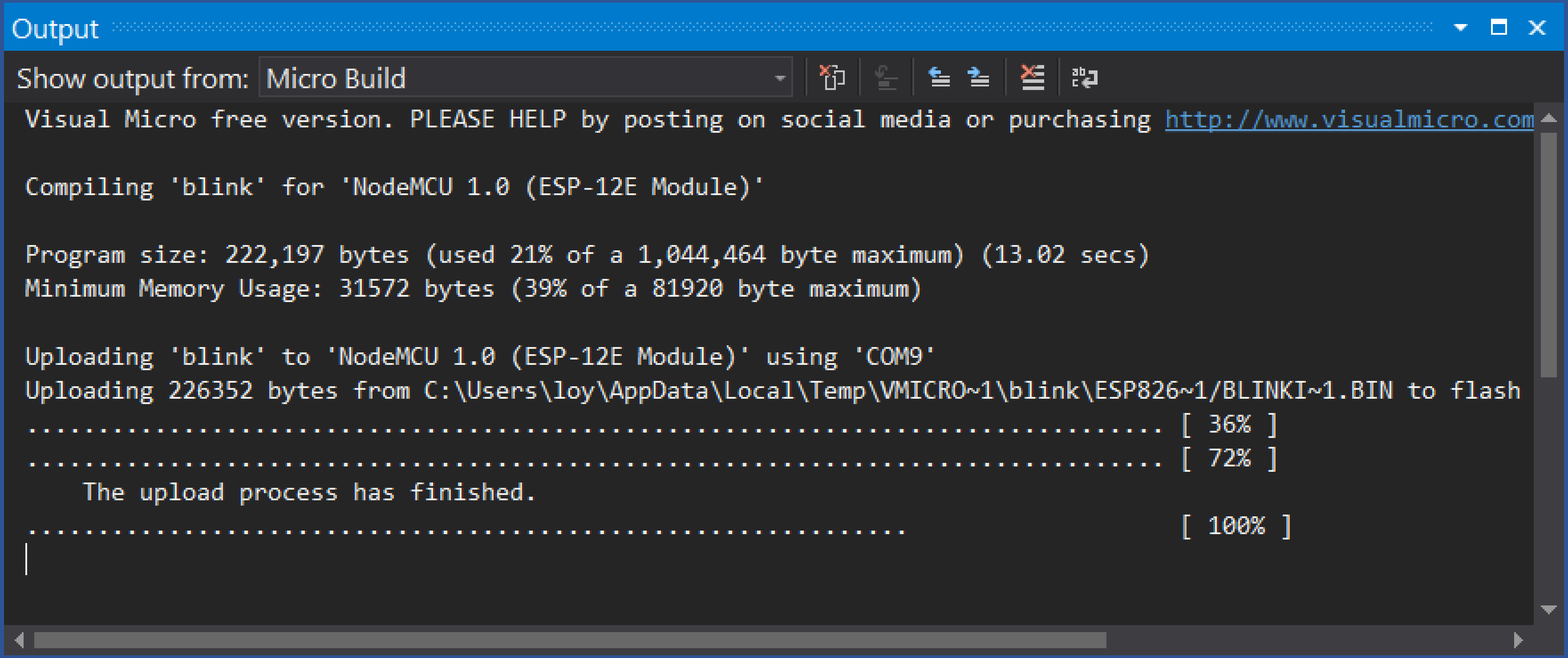
- Blinky
- Hello World!
- Read temperature censor
- Wi-Fi connection test
- MQTT D2C test
- MQTT C2D test
- MQTT D2C/C2D

Blinky


blink - Microsoft Visual Studio
File Edit View Project Build Debug Team vMicro Tools Architecture Test Analyze Window Help
Debug x86 Start
Arduino 1.6 NodeMCU 1.0 (ESP-12E Mod COM9 Other Add Library Add Code
Solution Explorer
Solution 'blink' (1 project)
blink
References
External Dependencies
Header Files
.blink.vsarduino.h
Resource Files
Source Files
blink.ino
blink.ino
(Global Scope)
loop()
Ready Ln 24 Col 2 Ch 2 INS Publish

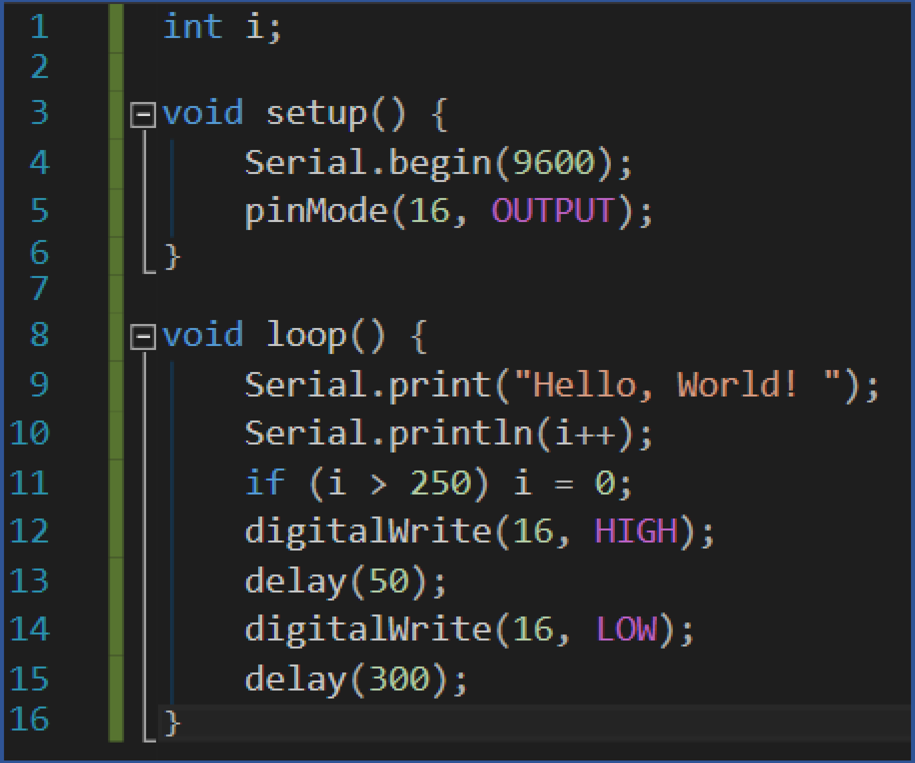
```
/*
ESP8266 Blink by Simon Peter
Blink the blue LED on the ESP-01 module
This example code is in the public domain

The blue LED on the ESP-01 module is connected to GPIO1
(which is also the TXD pin; so we cannot use Serial.print() at the same

Note that this sketch uses LED_BUILTIN to find the pin with the interna
*/

void setup() {
    pinMode(LED_BUILTIN, OUTPUT);     // Initialize the LED_BUILTIN pir
}

// the loop function runs over and over again forever
void loop() {
    digitalWrite(LED_BUILTIN, LOW);   // Turn the LED on (Note that LOW
                                      // but actually the LED is on; th
                                      // it is acive low on the ESP-01)
    delay(500);                      // Wait for a second
    digitalWrite(LED_BUILTIN, HIGH);  // Turn the LED off by making the
    delay(100);                      // Wait for two seconds (to demons
}
```

COMPILE AND UPOLAD

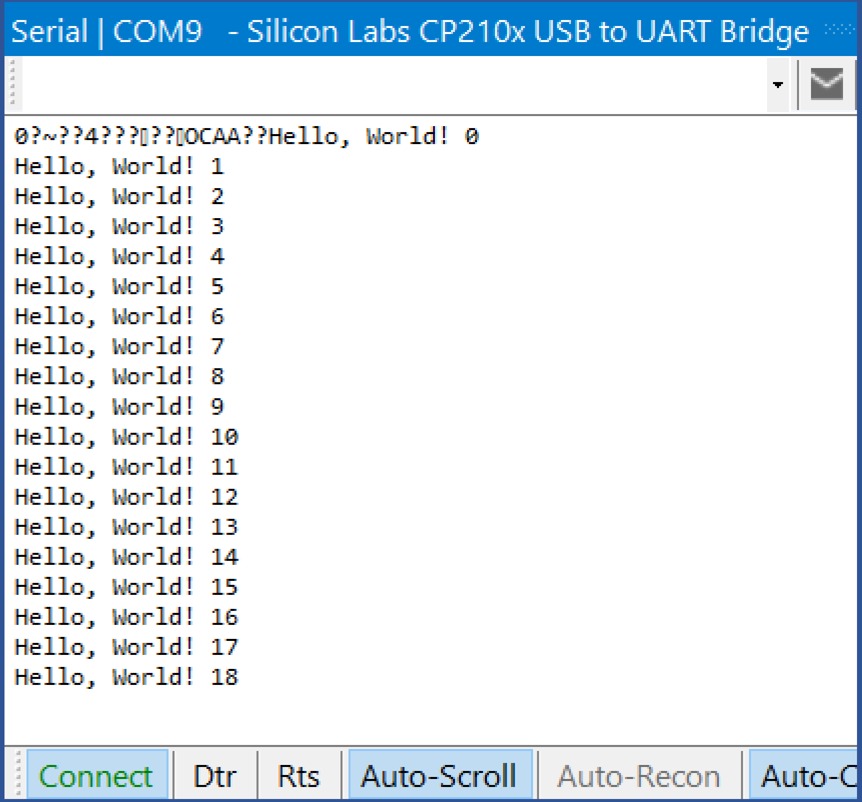


```
Visual Micro free version. PLEASE HELP by posting on social media or purchasing http://www.visualmicro.com

Compiling 'blink' for 'NodeMCU 1.0 (ESP-12E Module)'

Program size: 222,197 bytes (used 21% of a 1,044,464 byte maximum) (13.02 secs)
Minimum Memory Usage: 31572 bytes (39% of a 81920 byte maximum)

Uploading 'blink' to 'NodeMCU 1.0 (ESP-12E Module)' using 'COM9'
Uploading 226352 bytes from C:\Users\loy\AppData\Local\Temp\VMICRO~1\blink\ESP826~1/BLINKI~1.BIN to flash
................................................................................ [ 36% ]
................................................................................ [ 72% ]
    The upload process has finished.
..............................................................                   [ 100% ]
```

Hello World!



```
int i;

void setup() {
    Serial.begin(9600);
    pinMode(16, OUTPUT);
}

void loop() {
    Serial.print("Hello, World! ");
    Serial.println(i++);
    if (i > 250) i = 0;
    digitalWrite(16, HIGH);
    delay(50);
    digitalWrite(16, LOW);
    delay(300);
}
```

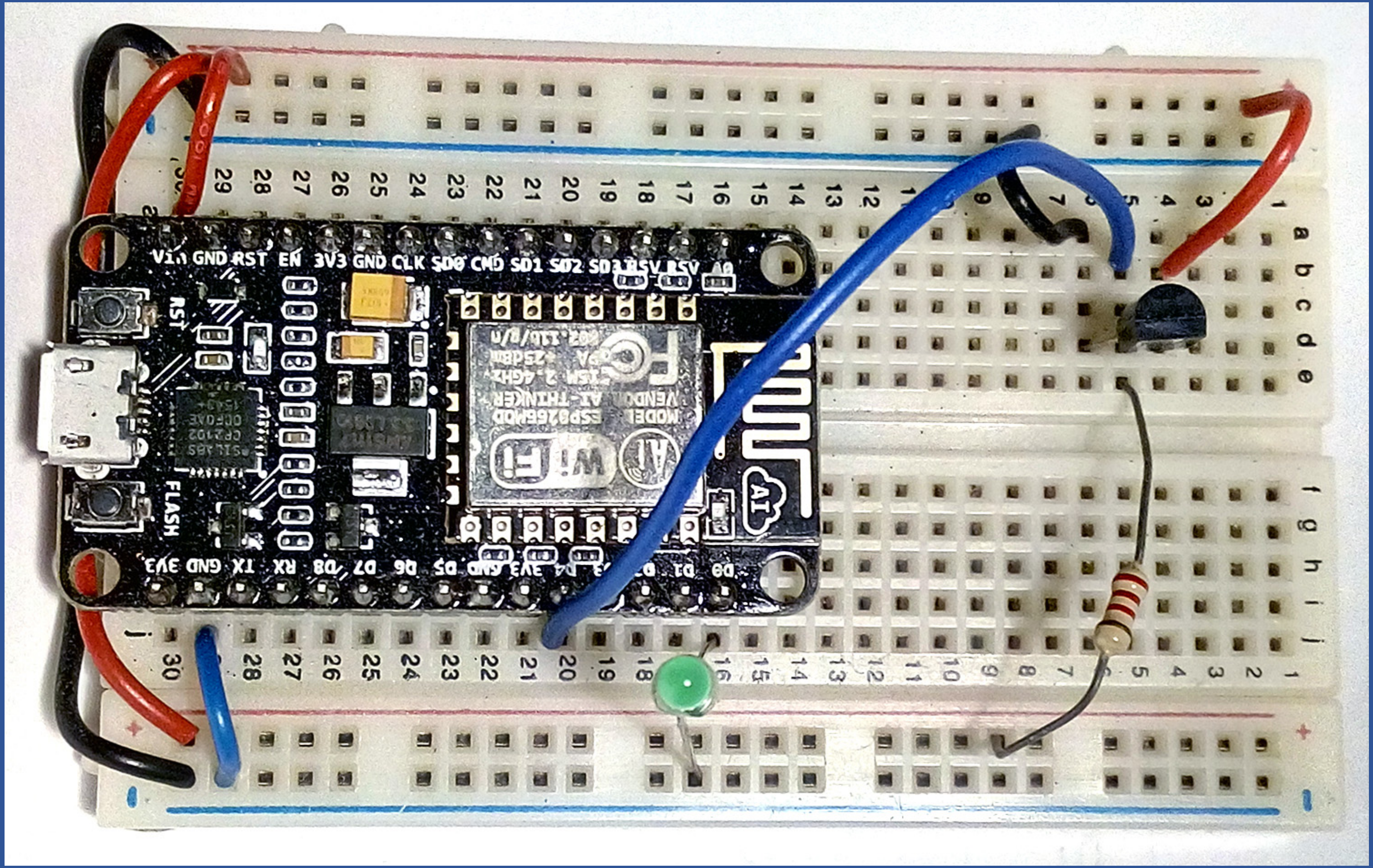


Serial | COM9 - Silicon Labs CP210x USB to UART Bridge

```
0?~??4???▯??▯OCAA??Hello, World! 0
Hello, World! 1
Hello, World! 2
Hello, World! 3
Hello, World! 4
Hello, World! 5
Hello, World! 6
Hello, World! 7
Hello, World! 8
Hello, World! 9
Hello, World! 10
Hello, World! 11
Hello, World! 12
Hello, World! 13
Hello, World! 14
Hello, World! 15
Hello, World! 16
Hello, World! 17
Hello, World! 18
```

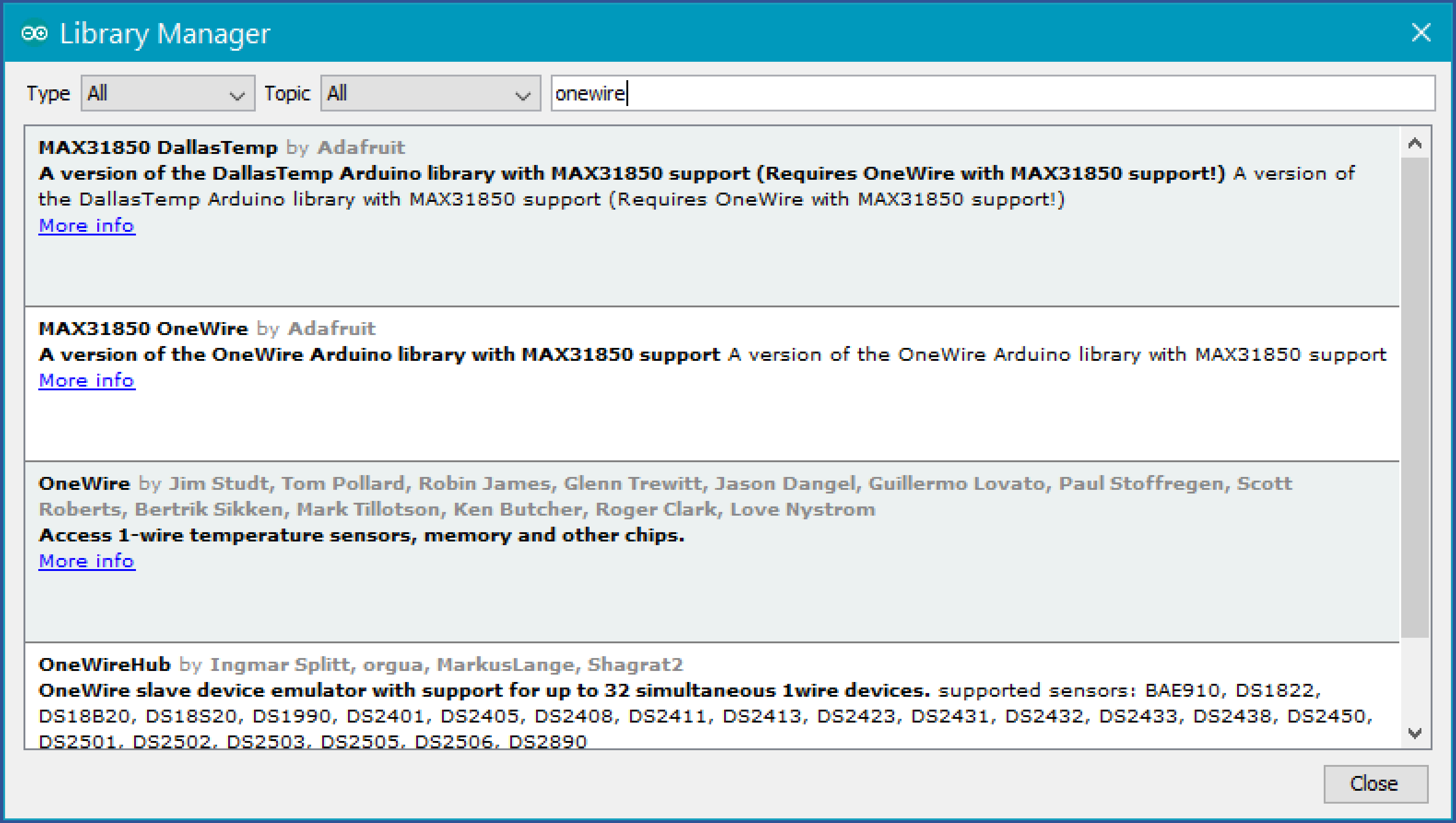
TEMPERATURE READING

Hardware setup



TEMPERATURE READING

Add OneWire Lib



onewire

MAX31850 DallasTemp by Adafruit
A version of the DallasTemp Arduino library with MAX31850 support (Requires OneWire with MAX31850 support!) A version of the DallasTemp Arduino library with MAX31850 support (Requires OneWire with MAX31850 support!)
More info

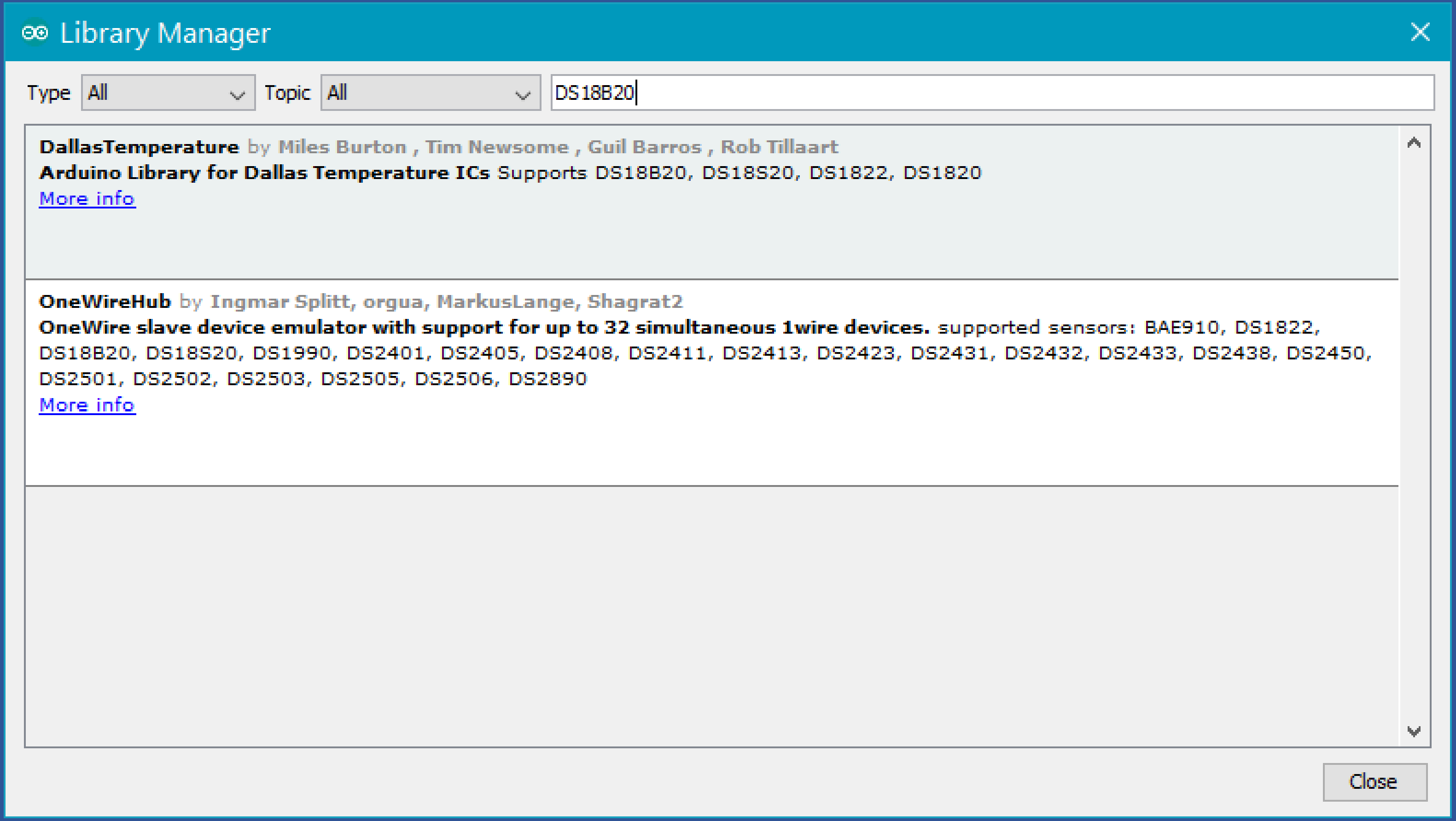
MAX31850 OneWire by Adafruit
A version of the OneWire Arduino library with MAX31850 support A version of the OneWire Arduino library with MAX31850 support
More info

OneWire by Jim Studt, Tom Pollard, Robin James, Glenn Trewitt, Jason Dangel, Guillermo Lovato, Paul Stoffregen, Scott Roberts, Bertrik Sikken, Mark Tillotson, Ken Butcher, Roger Clark, Love Nystrom
Access 1-wire temperature sensors, memory and other chips.
More info

OneWireHub by Ingmar Splitt, orgua, MarkusLange, Shagrat2
OneWire slave device emulator with support for up to 32 simultaneous 1wire devices. supported sensors: BAE910, DS1822, DS18B20, DS18S20, DS1990, DS2401, DS2405, DS2408, DS2411, DS2413, DS2423, DS2431, DS2432, DS2433, DS2438, DS2450, DS2501, DS2502, DS2503, DS2505, DS2506, DS2890

TEMPERATURE READING

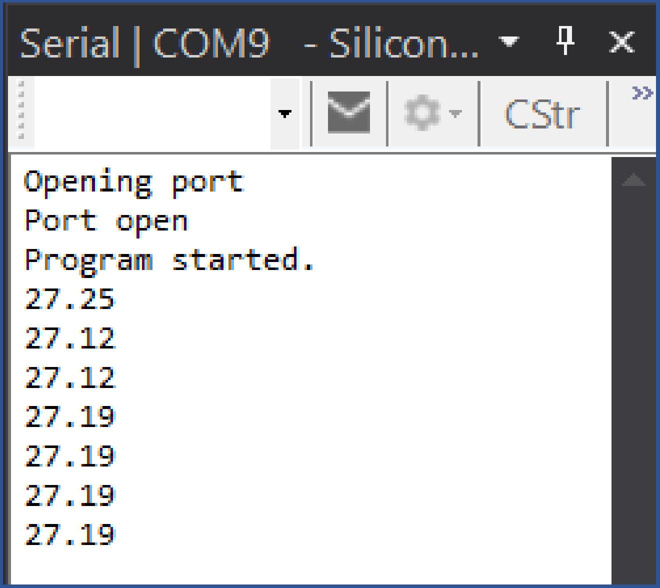
Add DallasTemperature Lib



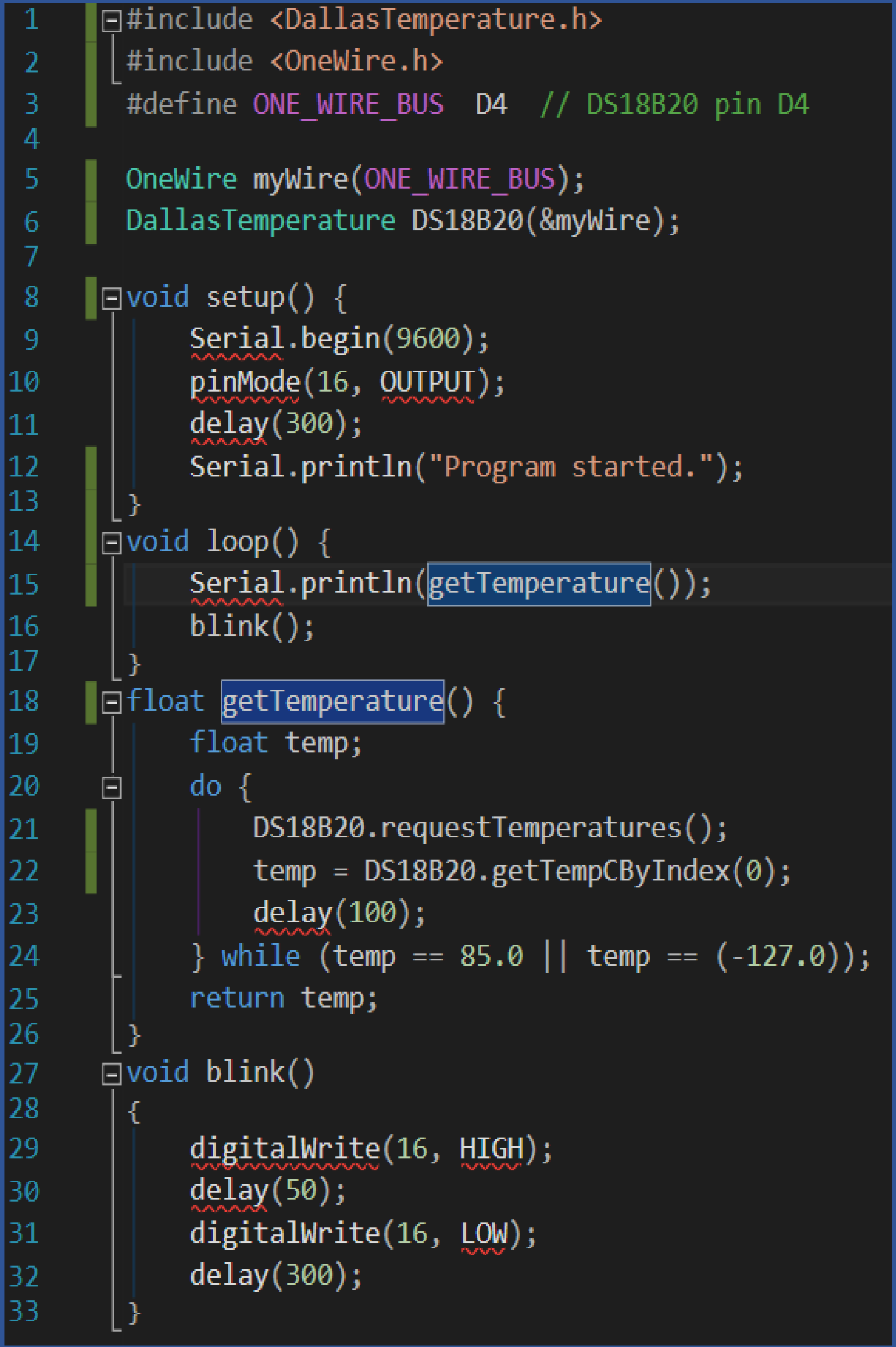
DallasTemperature by Miles Burton , Tim Newsome , Guil Barros , Rob Tillaart
Arduino Library for Dallas Temperature ICs Supports DS18B20, DS18S20, DS1822, DS1820
More info

OneWireHub by Ingmar Splitt, orgua, MarkusLange, Shagrat2
OneWire slave device emulator with support for up to 32 simultaneous 1wire devices. supported sensors: BAE910, DS1822, DS18B20, DS18S20, DS1990, DS2401, DS2405, DS2408, DS2411, DS2413, DS2423, DS2431, DS2432, DS2433, DS2438, DS2450, DS2501, DS2502, DS2503, DS2505, DS2506, DS2890
More info

TEMPERATURE READING Source code

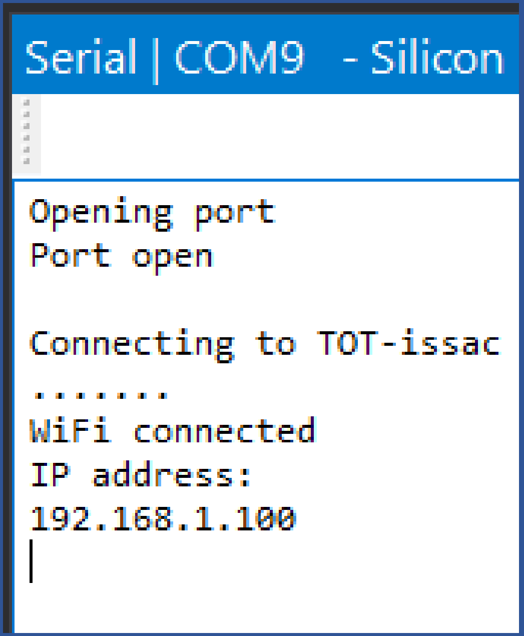


```
Opening port
Port open
Program started.
27.25
27.12
27.12
27.19
27.19
27.19
27.19
```



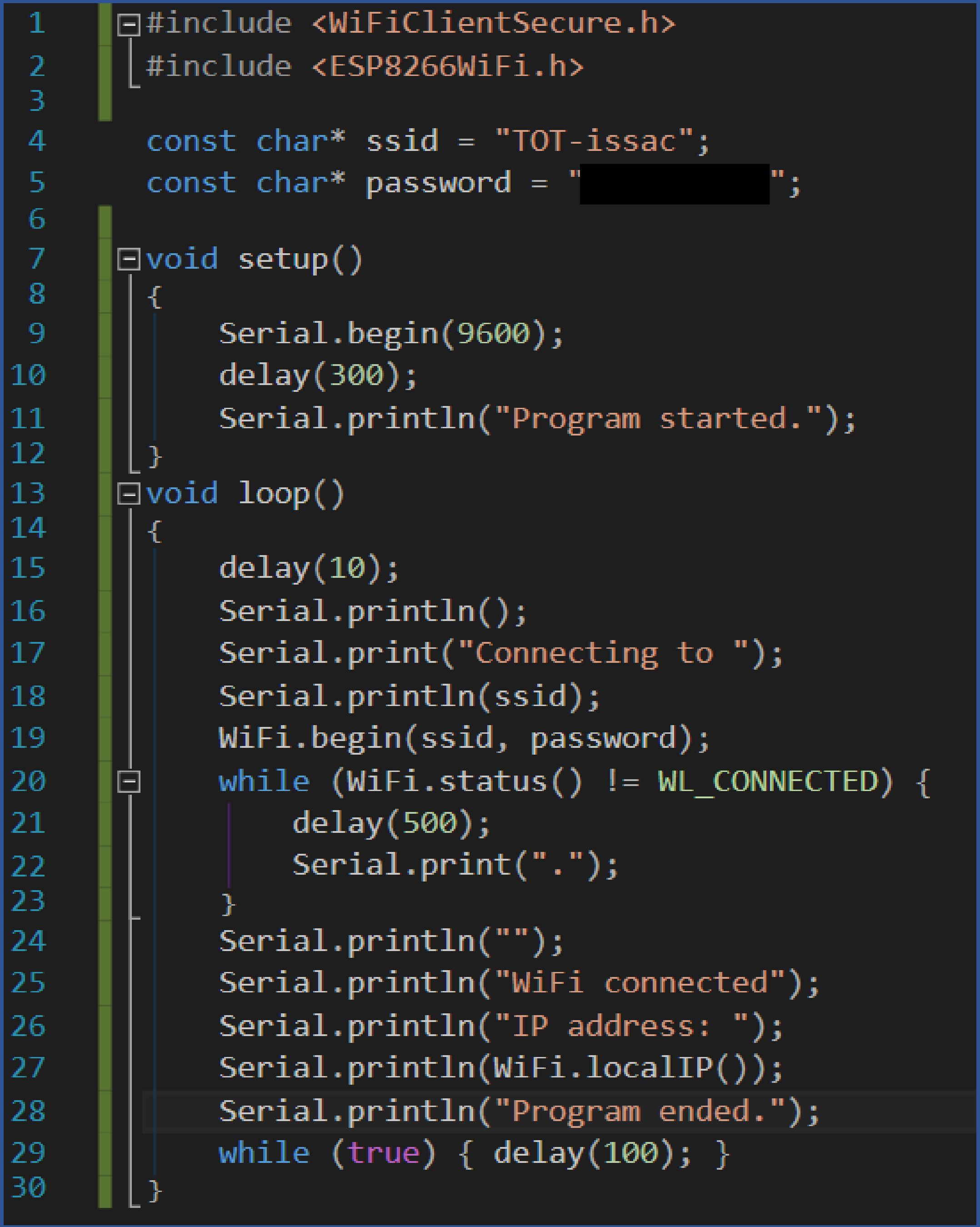
```
#include <DallasTemperature.h>
#include <OneWire.h>
#define ONE_WIRE_BUS  D4  // DS18B20 pin D4

OneWire myWire(ONE_WIRE_BUS);
DallasTemperature DS18B20(&myWire);

void setup() {
    Serial.begin(9600);
    pinMode(16, OUTPUT);
    delay(300);
    Serial.println("Program started.");
}
void loop() {
    Serial.println(getTemperature());
    blink();
}
float getTemperature() {
    float temp;
    do {
        DS18B20.requestTemperatures();
        temp = DS18B20.getTempCByIndex(0);
        delay(100);
    } while (temp == 85.0 || temp == (-127.0));
    return temp;
}
void blink()
{
    digitalWrite(16, HIGH);
    delay(50);
    digitalWrite(16, LOW);
    delay(300);
}
```

Wi-Fi Test



Serial | COM9 - Silicon

```
Opening port
Port open

Connecting to TOT-issac
........
WiFi connected
IP address:
192.168.1.100
```



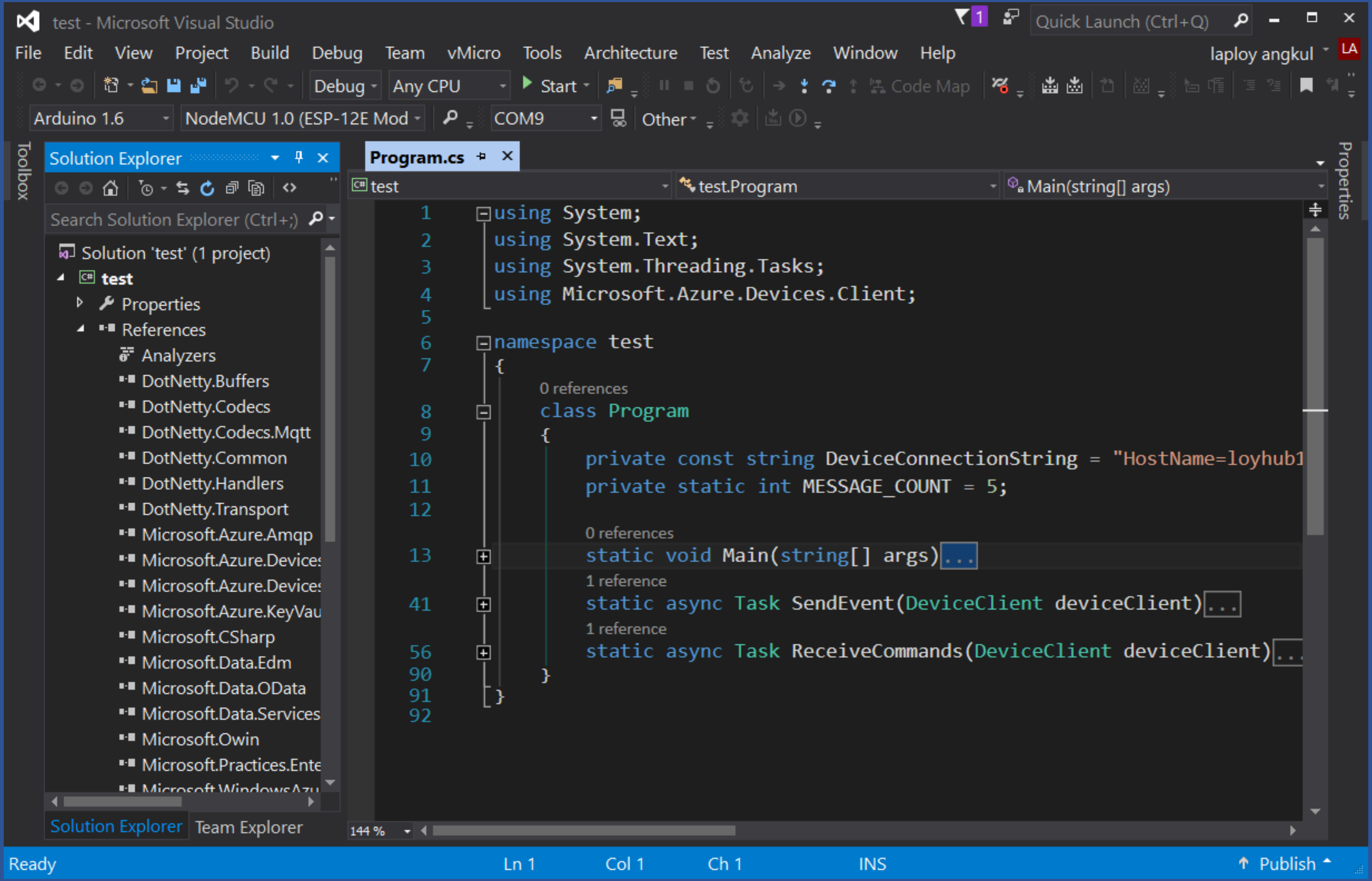
```
#include <WiFiClientSecure.h>
#include <ESP8266WiFi.h>

const char* ssid = "TOT-issac";
const char* password = "[redacted]";

void setup()
{
    Serial.begin(9600);
    delay(300);
    Serial.println("Program started.");
}
void loop()
{
    delay(10);
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
    Serial.println("Program ended.");
    while (true) { delay(100); }
}
```

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#

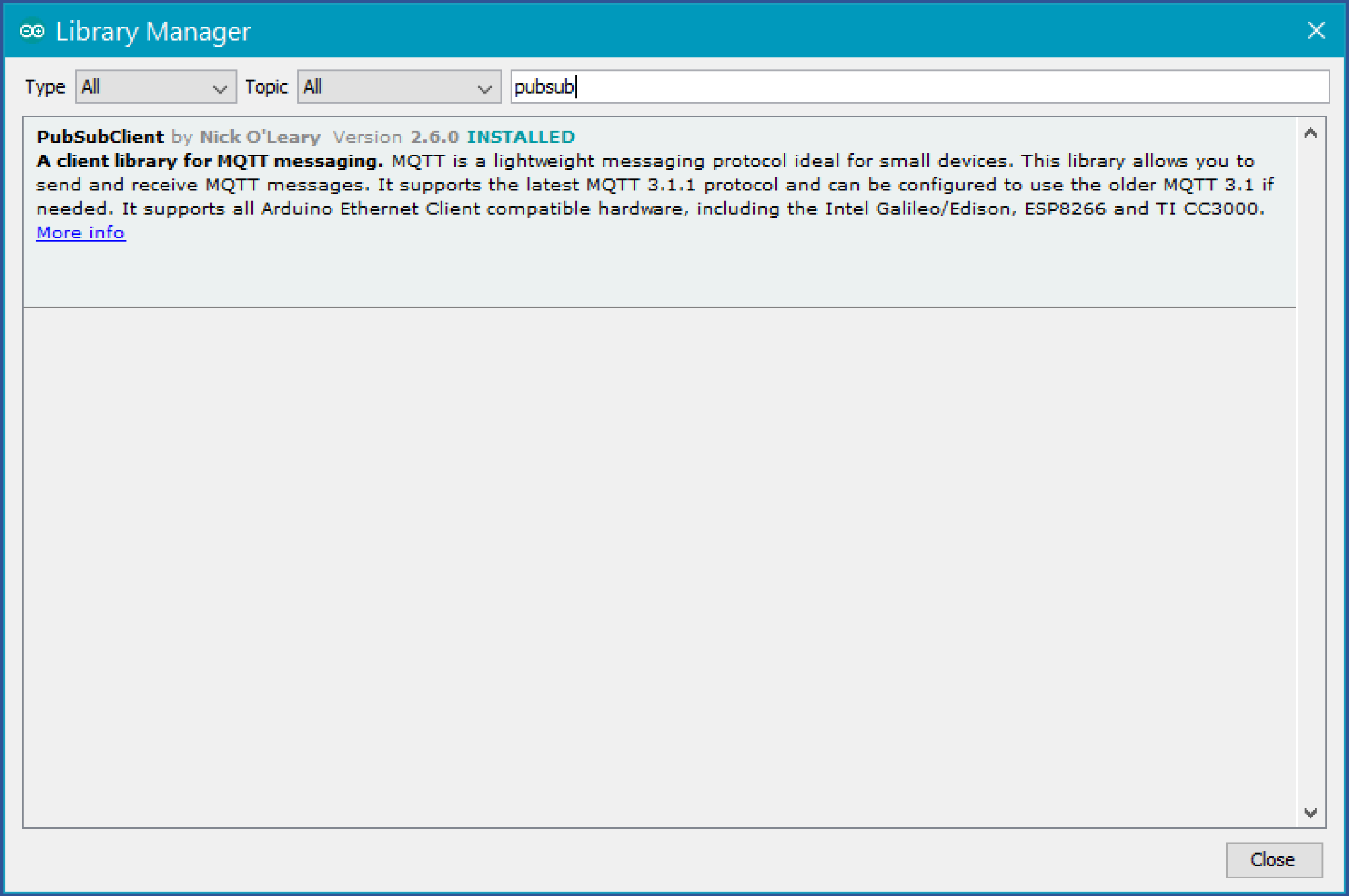


```
using System;
using System.Text;
using System.Threading.Tasks;
using Microsoft.Azure.Devices.Client;

namespace test
{
    0 references
    class Program
    {
        private const string DeviceConnectionString = "HostName=loyhub1
        private static int MESSAGE_COUNT = 5;

        0 references
        static void Main(string[] args)...
        1 reference
        static async Task SendEvent(DeviceClient deviceClient)...
        1 reference
        static async Task ReceiveCommands(DeviceClient deviceClient)...
    }
}
```

Install ESP 8266 MQTT Lib

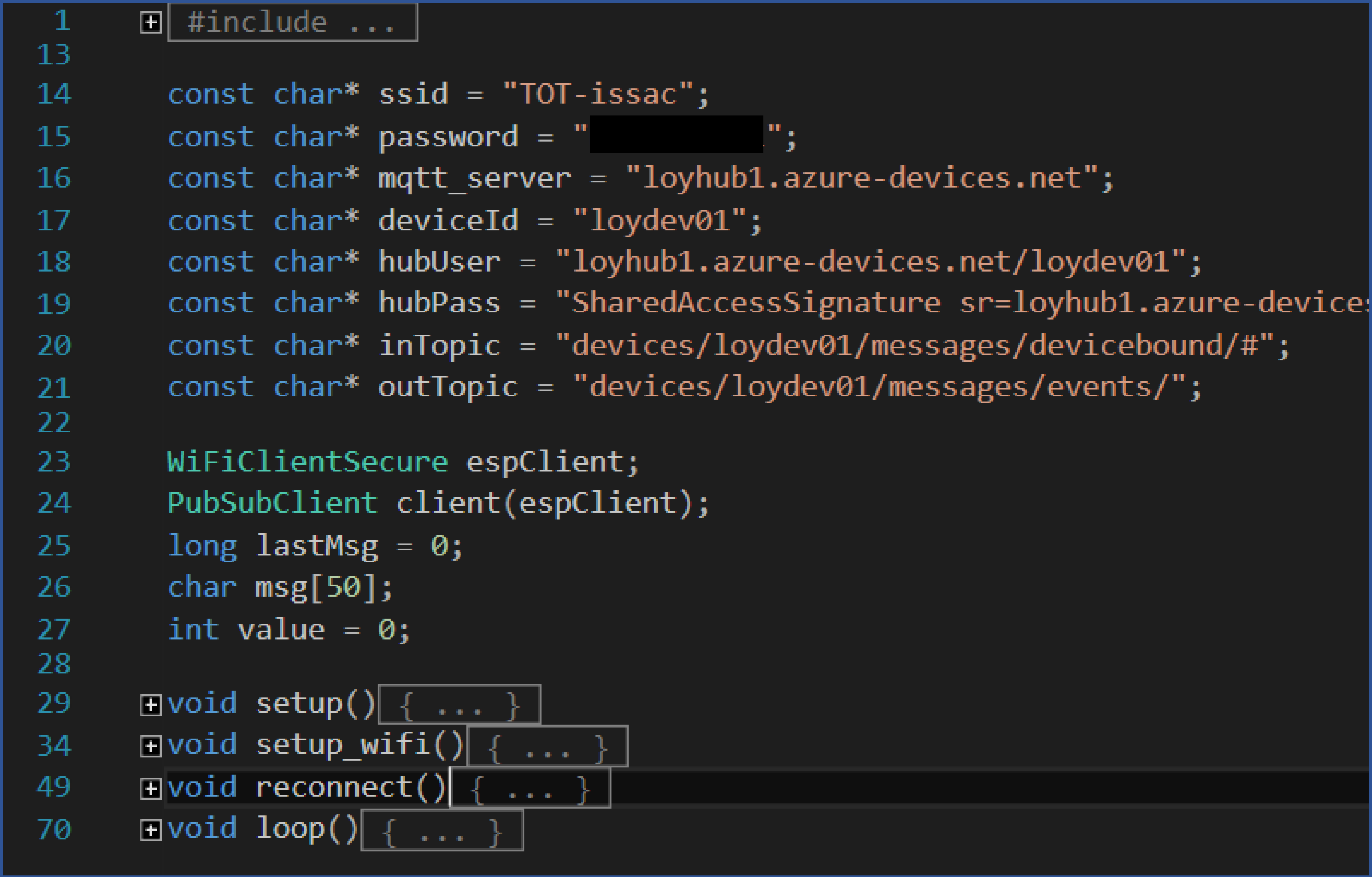
PubSubClient by Nick O'Leary Version 2.6.0

A client library for MQTT messaging.



PubSubClient by Nick O'Leary Version 2.6.0 INSTALLED
A client library for MQTT messaging. MQTT is a lightweight messaging protocol ideal for small devices. This library allows you to send and receive MQTT messages. It supports the latest MQTT 3.1.1 protocol and can be configured to use the older MQTT 3.1 if needed. It supports all Arduino Ethernet Client compatible hardware, including the Intel Galileo/Edison, ESP8266 and TI CC3000.
More info

DEVICE TO CLOUD



```
#include ...

const char* ssid = "TOT-issac";
const char* password = "[redacted]";
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-device
const char* inTopic = "devices/loydev01/messages/devicebound/#";
const char* outTopic = "devices/loydev01/messages/events/";

WiFiClientSecure espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int value = 0;

void setup() { ... }
void setup_wifi() { ... }
void reconnect() { ... }
void loop() { ... }
```

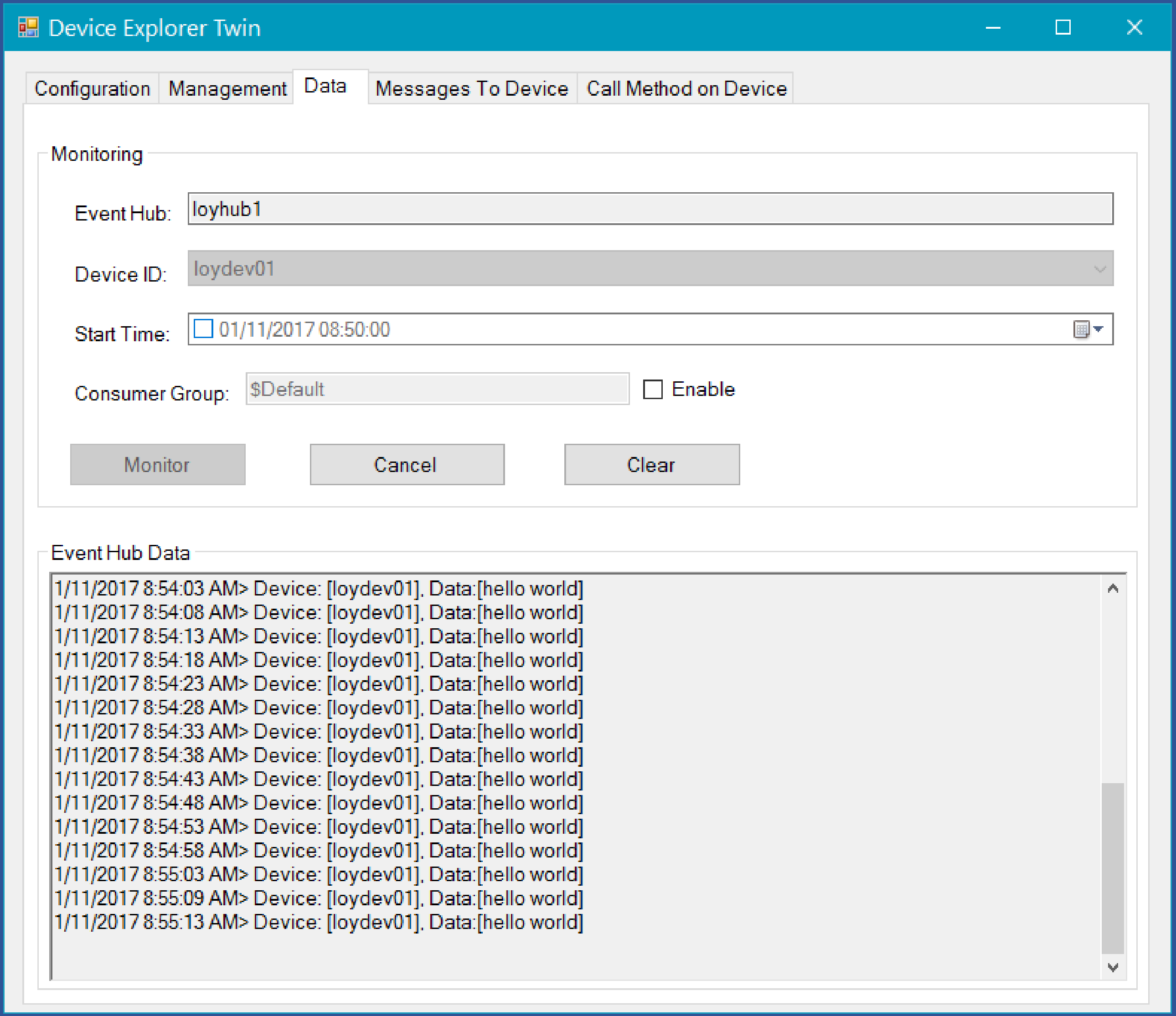
CONSTANCE

```
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%2
FW2OnTvdkbno%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=pZCYksE6NDkmftmOnJ0PeFqj1Wc9lS4%2FW2OnTvdkbno%3D&se=1515634239";

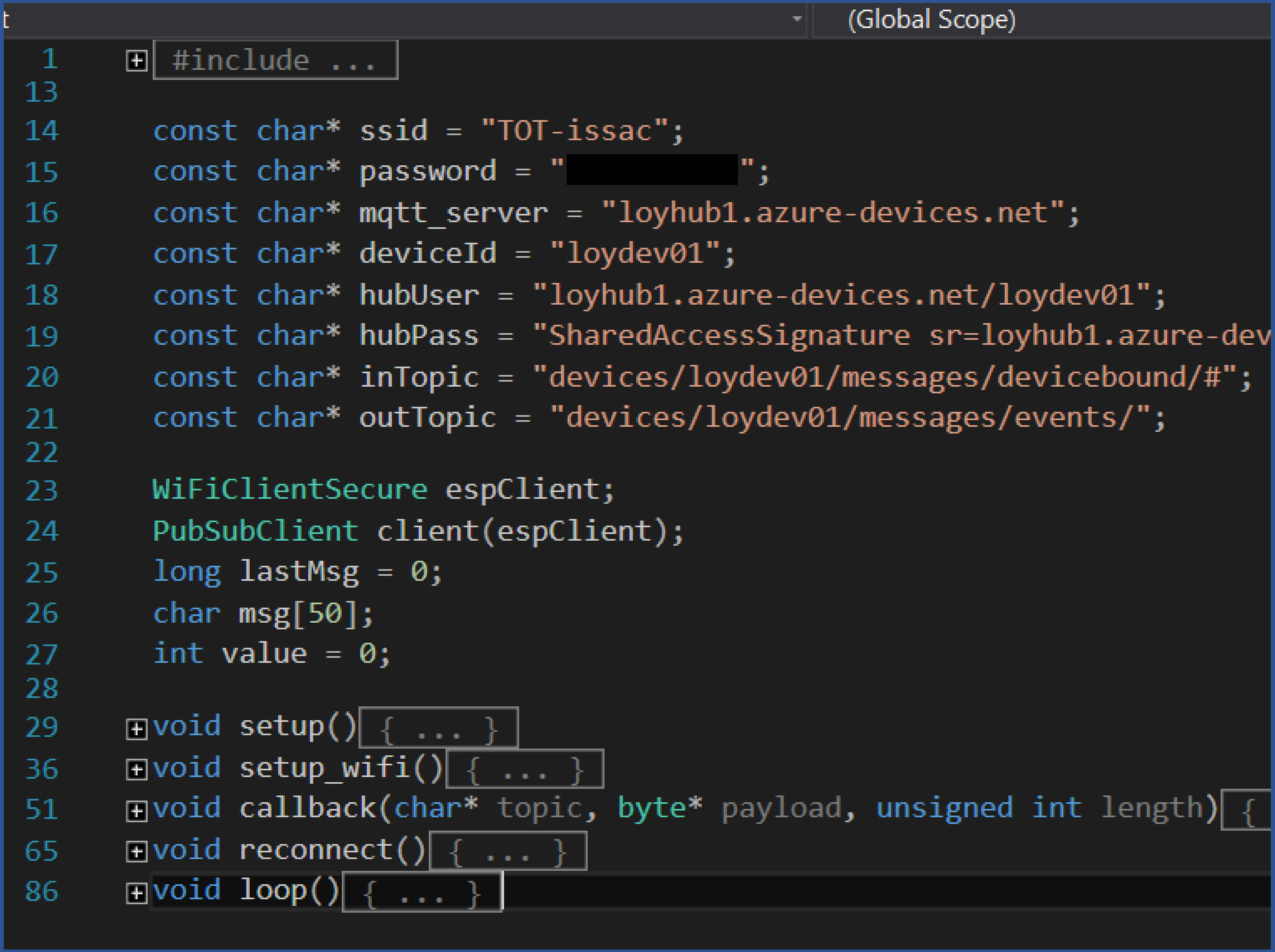
DEVICE EXPLOROR SHOW D2C MESSAGE



Event Hub: loyhub1
Device ID: loydev01
Start Time: 01/11/2017 08:50:00
Consumer Group: $Default ☐ Enable

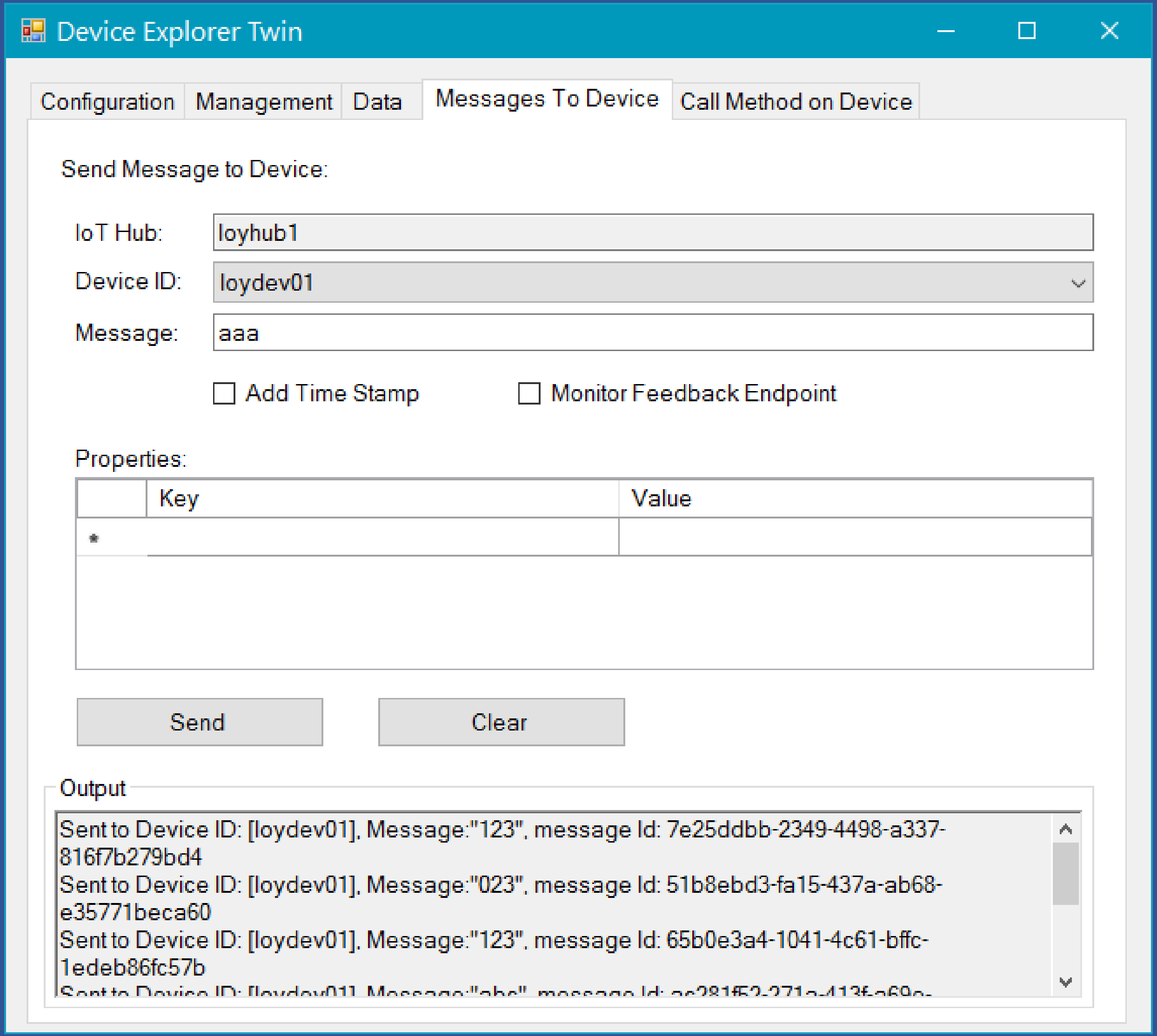
```
1/11/2017 8:54:03 AM> Device: [loydev01], Data:[hello world]
1/11/2017 8:54:08 AM> Device: [loydev01], Data:[hello world]
1/11/2017 8:54:13 AM> Device: [loydev01], Data:[hello world]
1/11/2017 8:54:18 AM> Device: [loydev01], Data:[hello world]
1/11/2017 8:54:23 AM> Device: [loydev01], Data:[hello world]
1/11/2017 8:54:28 AM> Device: [loydev01], Data:[hello world]
1/11/2017 8:54:33 AM> Device: [loydev01], Data:[hello world]
1/11/2017 8:54:38 AM> Device: [loydev01], Data:[hello world]
1/11/2017 8:54:43 AM> Device: [loydev01], Data:[hello world]
1/11/2017 8:54:48 AM> Device: [loydev01], Data:[hello world]
1/11/2017 8:54:53 AM> Device: [loydev01], Data:[hello world]
1/11/2017 8:54:58 AM> Device: [loydev01], Data:[hello world]
1/11/2017 8:55:03 AM> Device: [loydev01], Data:[hello world]
1/11/2017 8:55:09 AM> Device: [loydev01], Data:[hello world]
1/11/2017 8:55:13 AM> Device: [loydev01], Data:[hello world]
```

CLOUD TO DEVICE



```
#include ...

const char* ssid = "TOT-issac";
const char* password = "[redacted]";
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-dev
const char* inTopic = "devices/loydev01/messages/devicebound/#";
const char* outTopic = "devices/loydev01/messages/events/";

WiFiClientSecure espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int value = 0;

void setup() { ... }
void setup_wifi() { ... }
void callback(char* topic, byte* payload, unsigned int length) {
void reconnect() { ... }
void loop() { ... }
```

DEVICE EXPLOROR SEND C2D MESSAGE



IoT Hub: loyhub1

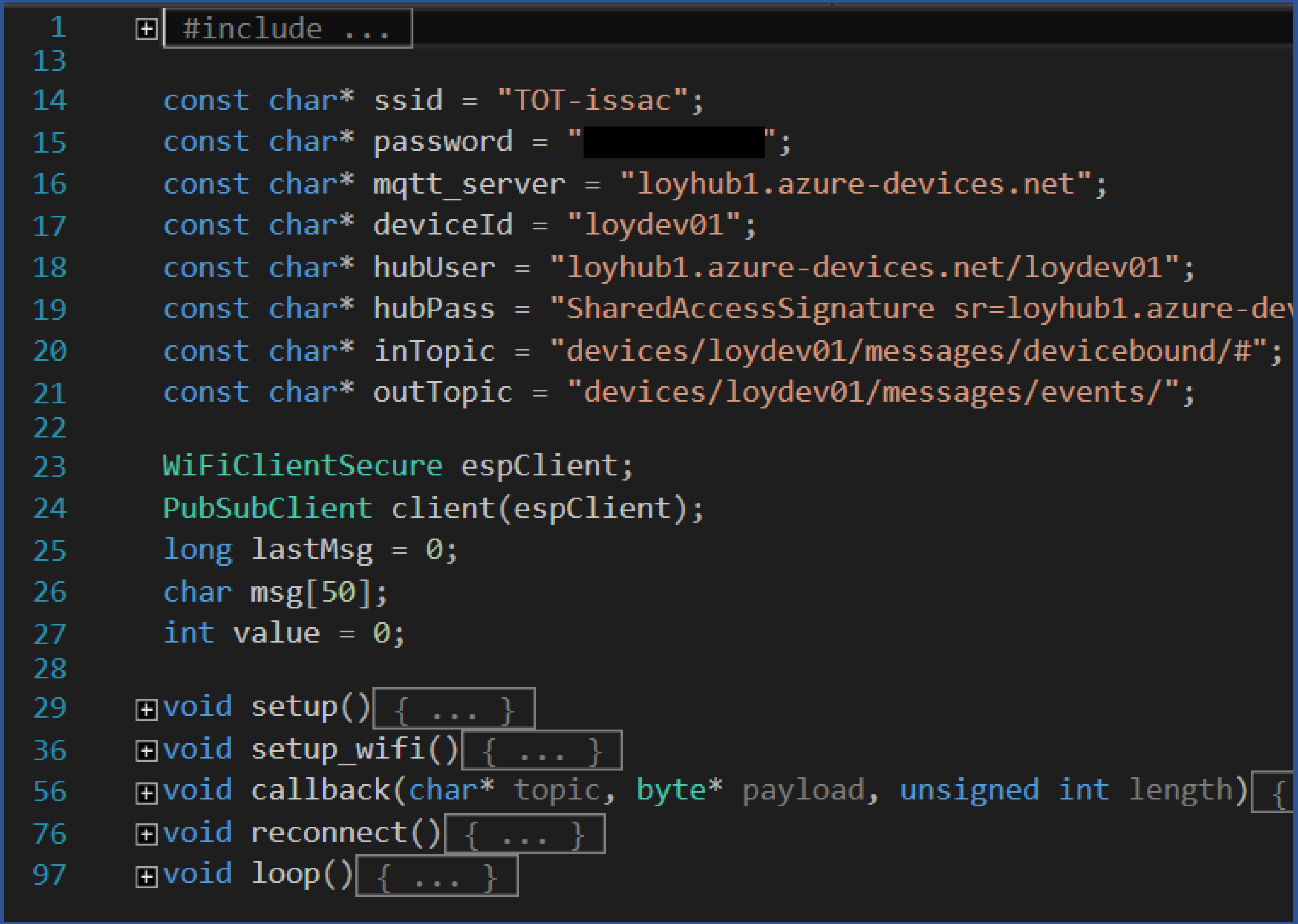
Device ID: loydev01

Message: aaa

	Key	Value
*		

```
Sent to Device ID: [loydev01], Message:"123", message Id: 7e25ddbb-2349-4498-a337-
816f7b279bd4
Sent to Device ID: [loydev01], Message:"023", message Id: 51b8ebd3-fa15-437a-ab68-
e35771beca60
Sent to Device ID: [loydev01], Message:"123", message Id: 65b0e3a4-1041-4c61-bffc-
1edeb86fc57b
```

MQTT D2C /C2D



```
#include ...

const char* ssid = "TOT-issac";
const char* password = "[redacted]";
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-dev
const char* inTopic = "devices/loydev01/messages/devicebound/#";
const char* outTopic = "devices/loydev01/messages/events/";

WiFiClientSecure espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int value = 0;

void setup() { ... }
void setup_wifi() { ... }
void callback(char* topic, byte* payload, unsigned int length) {
void reconnect() { ... }
void loop() { ... }
```