

### บทที่ 3 Node32s โค้ดโปรแกรมด้วย Platformio

เมื่อก่อนจะโค้ดโปรแกรม Node32s ต้องเขียนโค้ด esp-idf บน Ubuntu Linux ซึ่งรันบน VirtualBox หรือ VMware บนระบบปฏิบัติการ Windows แต่ตอนนี้ผู้เขียนขอแนะนำเครื่องมือเขียนโค้ดที่เป็นที่นิยมนั่นคือ Atom Platformio จุดเด่นคือ สามารถเขียนได้ทั้งแบบ esp-idf และแบบ Arduino IDE ใช้งานง่าย มีตัวช่วยในการเขียนโค้ดมากมาย หากคุณสามารถลองโค้ดโปรแกรมด้วย Platformio แล้วคุณ一定会

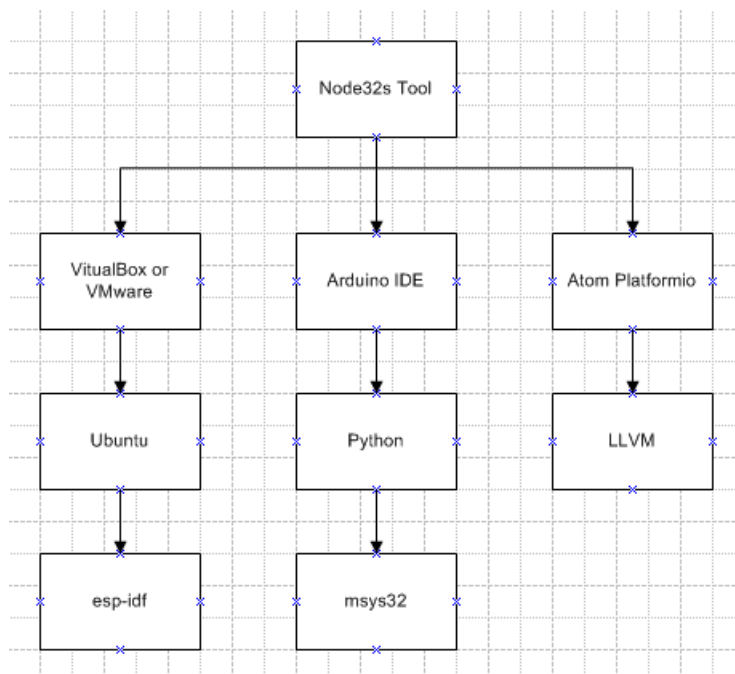
#### 3.1 วิธีโค้ดโปรแกรมลง Node32s

การเขียนโค้ดโปรแกรมลง Node32s เท่าที่ผู้เขียนรู้จักในตอนนี้มี 3 วิธีคือ

3.1.1 เขียนโค้ดแบบ C (esp-idf) บน Ubuntu ซึ่งรันอยู่บน VirtualBox หรือ VMware และบน Windows

3.1.2 เขียนโค้ดแบบ wiring บน Arduino IDE (รายละเอียดเพิ่มเติมโปรดอ่านบทที่ 2)

3.1.3 เขียนโค้ดแบบ C (esp-idf) หรือ wiring (Arduino IDE) บน Atom Platformio



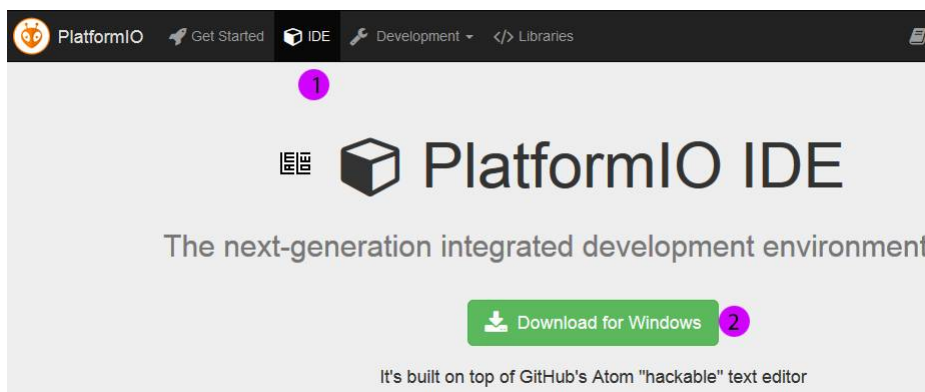
<<ch03-01.tif>> รูปที่ 3-1 โค้ดโปรแกรมลง Node32s

## 3.2 วิธีติดตั้ง Platformio

Platformio เป็นแพลตฟอร์มเสริมการทำงานของ Editor ที่โด่งดังนั่นคือ Atom ทำให้สามารถเขียนโค้ดโปรแกรมลงบอร์ดทดลองได้หลากหลาย (Embedded Board) หนึ่งในนั้นก็คือบอร์ด Node32s ซึ่งเป็นบอร์ดหลักที่ใช้ในหนังสือเล่มนี้เอง

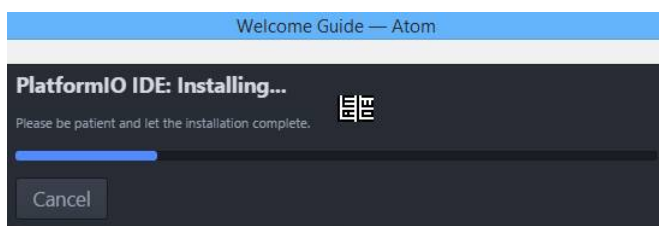
3.2.1 ให้คุณเปิดเบราว์เซอร์ไปที่ <http://platformio.org> คลิก IDE

3.2.2 คลิก Download for Windows เพื่อดาวน์โหลดไฟล์ platformio-atom-windows.exe



<<ch03-02.tif>> รูปที่ 3-2 คลิก IDE > Download for Windows

3.2.3 ดับเบิลคลิกไฟล์ platformio-atom-windows.exe ที่คุณเพิ่งดาวน์โหลดมา รอจนกระทั่งได้นหน้าจอ PlatformIO IDE: Installing... แล้วรอจน Atom เรียกแพ็คเกจ Platformio มาติดตั้งให้โดยอัตโนมัติ ขั้นตอนนี้ใช้เวลาพอสมควร



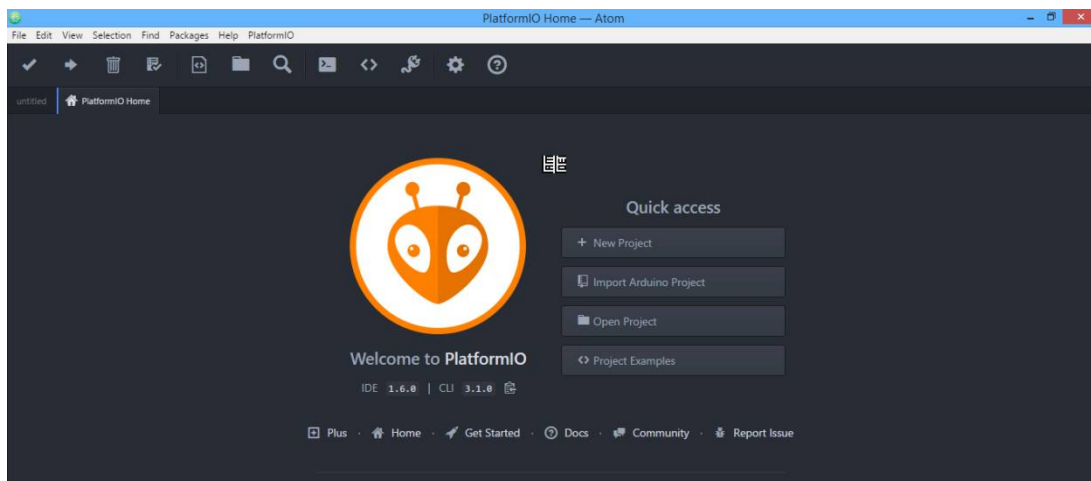
<<ch03-03.tif>> รูปที่ 3-3 ดับเบิลคลิกไฟล์ แล้วรอจนโปรแกรมติดตั้งเสร็จ

3.2.4 เมื่อโปรแกรมติดตั้งเสร็จแล้ว จะขึ้นหน้าจอ Atom PlatformIO IDE has been successfully installed! คลิก Reload Now เพื่อเริ่มค่าใหม่ ดังรูป 3-4



<<ch03-04.tif>> รูปที่ 3-4 คลิก Reload Now

3.2.5 หน้าจอ PlatformIO Home – Atom เห็นข้อความ Welcome to PlatformIO แสดงว่าคุณได้ติดตั้งโปรแกรมเสร็จเรียบร้อยแล้ว



<<ch03-05.tif>> รูปที่ 3-5 หน้าจอ PlatformIO Home – Atom

มีโฟลเดอร์ที่น่าสนใจ ดังนี้คือ

C:\Users\admin\AppData\Local\atom – สำหรับเก็บโปรแกรมหลัก

C:\Users\admin\.atom – สำหรับเก็บโปรแกรมเสริม

### 3.3 วิธีติดตั้ง LLVM

Platformio ใช้ LLVM เป็น Clang เพื่อทำให้การคอมไพล์ได้รวดเร็วมากขึ้น

3.3.1 ให้คุณเปิดบราวเซอร์ไปที่ <http://llvm.org> คลิก LLVM 3.9.0

### 3.3.2 จากนั้นคลิก Clang for Windows (64-bit) เพื่อดาวน์โหลดไฟล์ LLVM-3.9.0-win64.exe



## The LLVM Compiler Infrastructure

**Site Map:**  
Overview  
Features  
Documentation  
Command Guide  
FAQ  
Publications  
LLVM Projects  
Open Projects  
LLVM Users  
Bug Database  
LLVM Logo  
Blog  
Meetings  
LLVM Foundation

**Download!**  
Download now:  
**LLVM 3.9.0**  
All Releases  
APT Packages  
Win Installer  
View the open-source license

### LLVM Overview

The LLVM Project is a collection of modular and reusable compiler and toolchain technologies. Despite its name, LLVM has little to do with traditional virtual machines, though it does provide helpful libraries that can be [used to build them](#). The name "LLVM" itself is not an acronym; it is the full name of the project.

LLVM began as a [research project](#) at the [University of Illinois](#), with the goal of providing a modern, SSA-based compilation strategy capable of supporting both static and dynamic compilation of arbitrary programming languages. Since then, LLVM has grown to be an umbrella project consisting of a number of subprojects, many of which are being used in production by a wide variety of [commercial and open source](#) projects as well as being widely used in [academic research](#). Code in the LLVM project is licensed under the ["UIUC" BSD-Style license](#).

The primary sub-projects of LLVM are:

1. The **LLVM Core** libraries provide a modern source- and target-independent [optimizer](#), along with [code generation support](#) for many popular CPUs (as well as some less common ones). These libraries are built around a [well specified](#) code representation known as the LLVM intermediate representation ("LLVM IR"). The LLVM Core libraries are [well documented](#), and it is particularly easy to invent your own language (or port an existing compiler) to use [LLVM as an optimizer and code generator](#).
2. **Clang** is an "LLVM native" C/C++/Objective-C compiler, which aims to deliver amazingly fast compiles (e.g. about [3x faster than GCC](#) when compiling Objective-C code in a debug configuration), extremely useful [error and warning messages](#) and to provide a platform for building great source level tools. The [Clang Static Analyzer](#) is a tool that automatically finds

### LLVM Download Page

**SVN Access**

If you'd like access to the "latest and greatest" in LLVM development, please see the instructions for accessing the [LLVM SVN Repository](#). The major changes and improvements that SVN contains relative to the previous release are listed in the [Release Notes](#) for the next release.

**Download LLVM 3.9.0**

**Sources:**

- [LLVM source code \(.zip\)](#)
- [Clang source code \(.zip\)](#)
- [compiler-rt source code \(.zip\)](#)
- [libc++ source code \(.zip\)](#)
- [libc++abi source code \(.zip\)](#)
- [libunwind source code \(.zip\)](#)
- [LLD Source code \(.zip\)](#)
- [LLDB Source code \(.zip\)](#)
- [OpenMP Source code \(.zip\)](#)
- [Polly Source code \(.zip\)](#)
- [Clang tools-extra \(.zip\)](#)
- [LLVM Test Suite \(.zip\)](#)

**Documentation:**

- [LLVM \(release notes\)](#)
- [Clang \(release notes\)](#)
- [clang-tools-extra \(release notes\)](#)
- [LLD \(release notes\)](#)
- [libc++](#)
- [LLVM Docsytem \(.tar.xz\)](#)
- [Clang Docsytem \(.tar.xz\)](#)
- [clang-tools-extra Docsytem \(.tar.xz\)](#)

**Pre-Built Binaries:**

- [Clang for Mac OS X \(.zip\)](#)
- [Clang for FreeBSD10 AMD64 \(.zip\)](#)
- [Clang for FreeBSD10 i386 \(.zip\)](#)
- [Clang for Arch64 Linux \(.zip\)](#)
- [Clang for armv7a Linux \(.zip\)](#)
- [Clang for Fedora23 i686 Linux \(.zip\)](#)
- [Clang for Fedora23 x86\\_64 Linux \(.zip\)](#)
- [Clang for OpenSUSE 13.2 i386 Linux \(.zip\)](#)
- [Clang for OpenSUSE 13.2 x86\\_64 Linux \(.zip\)](#)
- [Clang for x86\\_64 Ubuntu 14.04 \(.zip\)](#)
- [Clang for x86\\_64 Ubuntu 16.04 \(.zip\)](#)
- [Clang for x86\\_64 Debian 8 \(.zip\)](#)
- [Clang for MIPS \(.zip\)](#)
- [Clang for MIPSEL \(.zip\)](#)
- [Clang for Windows \(32-bit\) \(.zip\)](#)
- [Clang for Windows \(64-bit\) \(.zip\)](#)

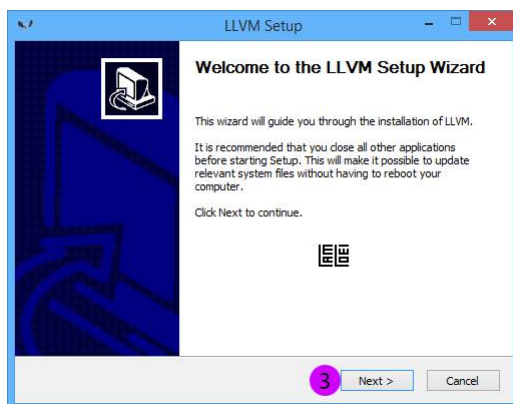
OpenMP run-time included in packages targeting Windows, x86\_64 Linux, and x86\_64 FreeBSD.

<<ch03-06.tif>> รูปที่ 3-6 คลิก LLVM 3.9.0

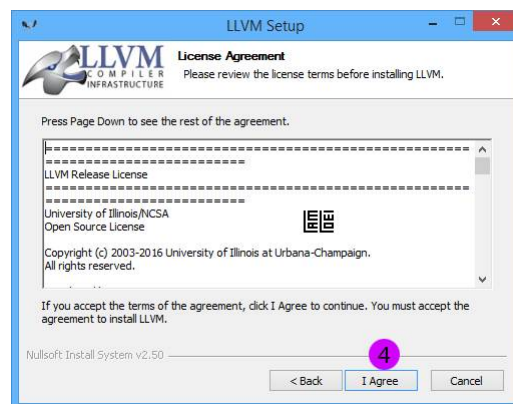
<<ch03-07.tif>> รูปที่ 3-7 คลิก Clang for Windows (64-bit)

3.3.3 ดับเบิ้ลคลิกไฟล์ LLVM-3.9.0-win64.exe ที่คุณเพิ่งดาวน์โหลดมา จะได้หน้าจอ LLVM Setup ข้อความต้อนรับสู่การติดตั้ง ให้คลิก Next >

3.3.4 หน้าจอ LLVM Setup ข้อกำหนดและเงื่อนไขการใช้งาน LLVM เมื่อคุณอ่านเข้าใจแล้ว คลิก I Agree



<<ch03-08.tif>> รูปที่ 3-8 คลิก Next >



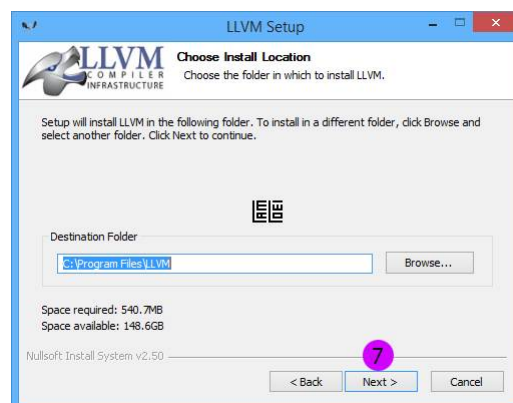
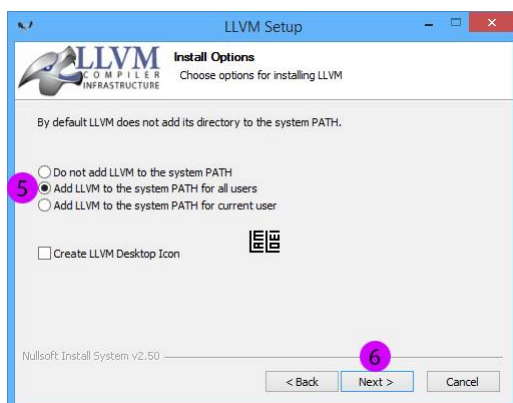
<<ch03-09.tif>> รูปที่ 3-9 คลิก I Agree

3.3.5 หน้าจอ LLVM Setup ตัวเลือกการติดตั้งให้คลิกวงกลมหน้า Add LLVM to the system PATH for all users เพื่อเพิ่มพาธของ LLVM ให้ทุกยูสเซอร์

- Do not add LLVM to the system PATH – ไม่ต้องเพิ่มพาธของ LLVM
- Add LLVM to the system PATH for all users – เพิ่มพาธให้ทุกยูสเซอร์
- Add LLVM to the system PATH for current user – เพิ่มพาธให้เฉพาะยูสเซอร์นี้
- Create LLVM Desktop Icon – สร้างไอคอน LLVM บนเดสก์ท็อป

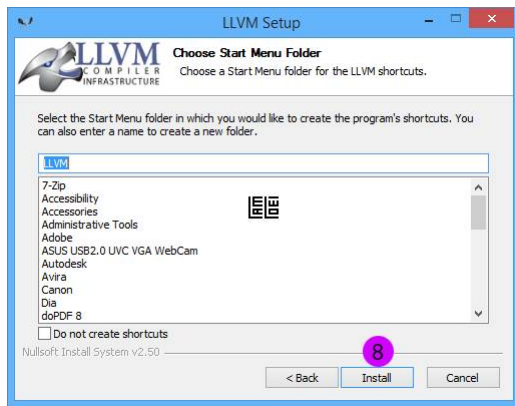
3.3.6 แล้วคลิก Next >

3.3.7 หน้าจอ LLVM Setup เลือกโฟลเดอร์ที่ต้องการเก็บ LLVM (หากต้องการเปลี่ยนคลิก Browse...) แล้วคลิก Next >

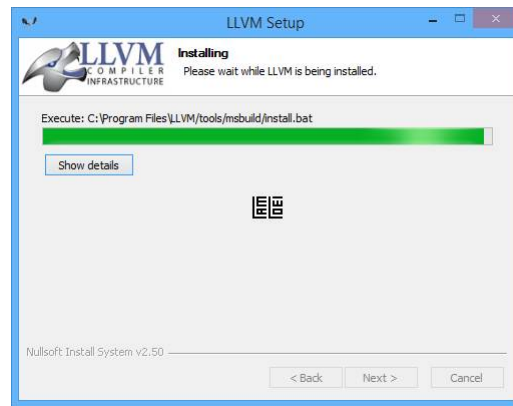


<<ch03-10.tif>> รูปที่ 3-10 คลิก Add LLVM > Next > <<ch03-11.tif>> รูปที่ 3-11 คลิก Next >

3.3.8 หน้าจอ LLVM Setup เพื่อเลือกสร้างเมนู LLVM ในเมนูสตาร์ท ในที่นี้เลือกเป็น LLVM แล้วคลิก Install ดังรูป 3-12 แล้วรอจนโปรแกรมติดตั้งเสร็จ ดังรูป 3-13



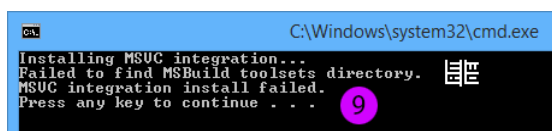
<<ch03-12.tif>> รูปที่ 3-12 คลิก Install



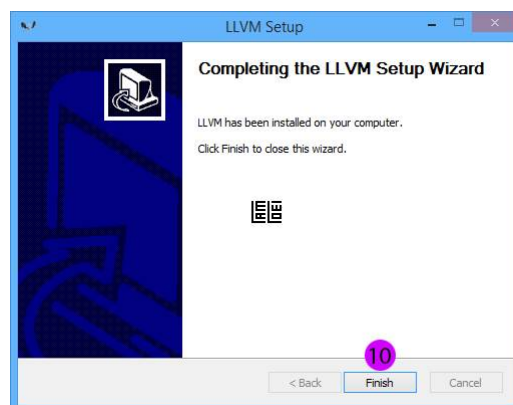
<<ch03-13.tif>> รูปที่ 3-13 รอสักครู่

3.3.9 หน้าจอ cmd ให้กดคีย์ Enter (หรือกดคีย์ใดคีย์หนึ่งบนแป้นพิมพ์ก็ได้)

3.3.10 หน้าจอ LLVM Setup ติดตั้งเสร็จเรียบร้อยแล้ว ให้คุณ Finish



<<ch03-14.tif>> รูปที่ 3-14 กดคีย์ Enter



<<ch03-15.tif>> รูปที่ 3-15 คลิก Finish

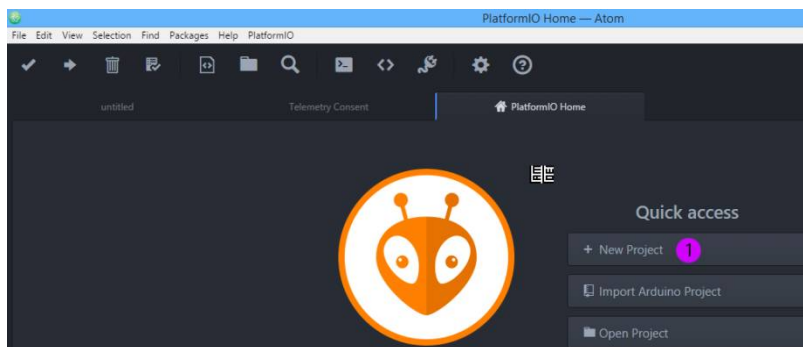
ข้อแนะนำ เมื่อติดตั้งทั้ง 2 โปรแกรมอันได้แก่ Atom PlatformIO, LLVM และตั้งค่าต่างๆ เสร็จเรียบร้อยแล้ว คุณควรจะ Restart คอมพิวเตอร์ เพื่อให้คอมพิวเตอร์เริ่มค่าใหม่ได้อย่างถูกต้อง

### 3.4 โปรแกรมแรก printf Hello world

มาเริ่มต้นด้วยโปรแกรมน่ายๆ กันก่อน โค้ดโปรแกรมสั้นๆ นี้มีเพียง 6 บรรทัด เพื่อแสดงข้อความที่อยู่ในคำสั่ง printf ข้อความในที่นี้คือ พิมพ์ Hello world ออกทางหน้าจอ Serial Monitor

#### 3.4.1 ที่หน้าจอ PlatformIO Home ให้คุณคลิก New Project เพื่อสร้างโครงการใหม่

(**โน้ต:** คุณสามารถคลิกเมนู Platformio > Home Screen เพื่อมาหน้าจอ PlatformIO Home)

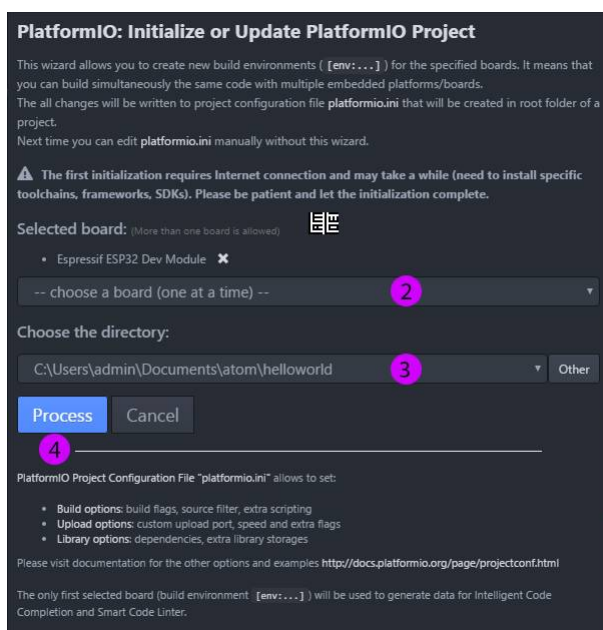


<<ch03-16.tif>> รูปที่ 3-16 คลิก New Project

#### 3.4.2 ในช่อง Select board: เลือกบอร์ดเป็น Espressif ESP32 Dev Module

#### 3.4.3 ในช่อง Choose the directory: เลือกโฟลเดอร์เป็น C:\Users\admin\Documents\atom\helloworld

#### 3.4.4 เมื่อคุณเลือกบอร์ด และโฟลเดอร์ที่เก็บโค้ด ที่ต้องการเรียบร้อยแล้ว ให้คลิก Process

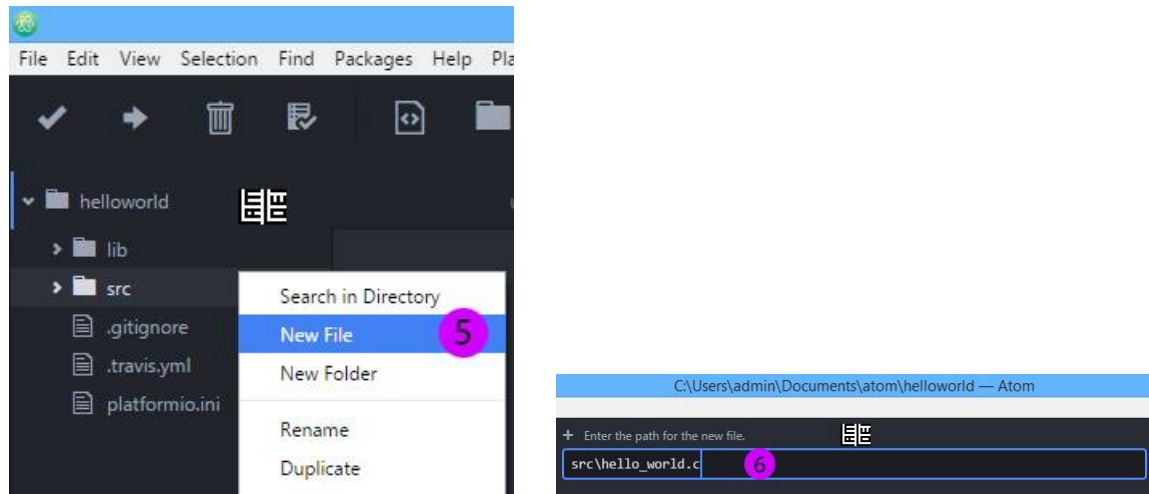


<<ch03-17.tif>> รูปที่ 3-17 คลิก Process



3.4.5 จะได้โฟลเดอร์ helloworld ขึ้นทางด้านซ้ายมือ จากนั้นคลิกขวาที่ src แล้วคลิกซ้ายที่ New File

3.4.6 จะมีช่อง Enter the path for the new file. ให้พิมพ์ชื่อไฟล์ src\hello\_world.c แล้วกดคีย์ Enter



<<ch03-18.tif>> รูปที่ 3-18 คลิกขวา src > คลิกซ้าย New File

<<ch03-19.tif>> รูปที่ 3-19 พิมพ์ชื่อไฟล์ แล้วกดคีย์ Enter

3.4.7 จะได้หน้าจอ hello\_world.c ให้พิมพ์โค้ดข้างล่างนี้ลงไป

```
#include <stdio.h>
```

```
void app_main()
```

```
{
```

```
    printf("Hello world\n");
```

```
}
```

3.4.8 เมื่อคุณพิมพ์เสร็จเรียบร้อยแล้ว ให้คลิกปุ่ม Serial Monitor

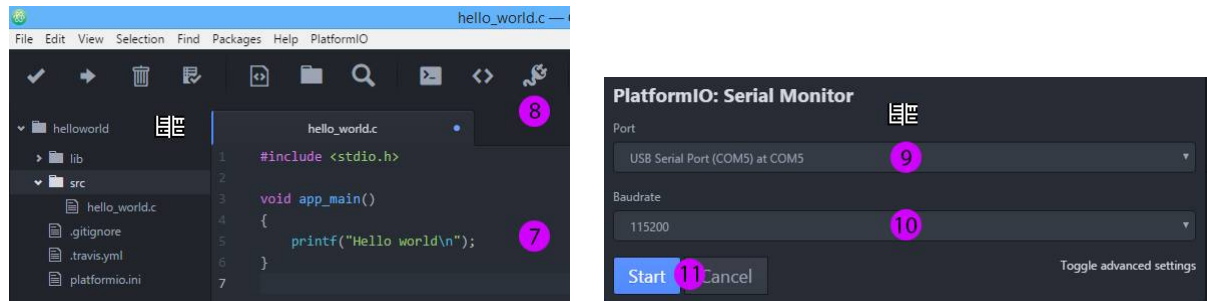
3.4.9 ในช่อง Port ให้เลือกพอร์ตที่บอร์ดของคุณเชื่อมต่อกับคอมพิวเตอร์ ในที่นี้คือ พอร์ต COM5

3.4.10 ในช่อง Baudrate ให้เลือกที่ 115200



### 3.4.11 คลิกปุ่ม Start

(**โน้ต:** โปรแกรม Platformio จะเลือกพอร์ตกับ baudrate ให้คุณแล้วโดยอัตโนมัติ)

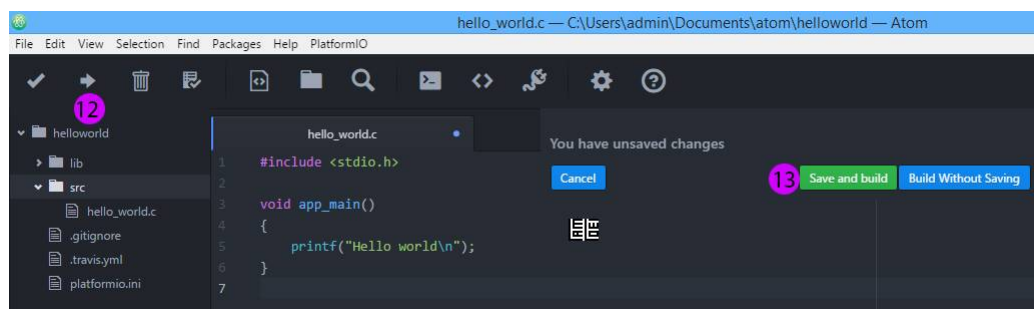


<<ch03-20.tif>> รูปที่ 3-20 พิมพ์โค้ด

<<ch03-21.tif>> รูปที่ 3-21 เลือกพอร์ตและ baudrate

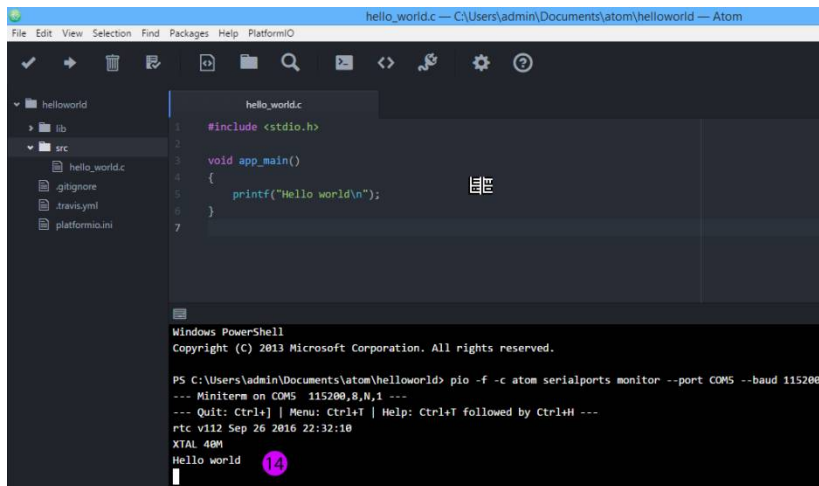
### 3.4.12 คลิกปุ่ม Upload

### 3.4.13 คลิกปุ่ม Save and build เพื่อบันทึกโค้ด และอัปโหลดโค้ดลงบอร์ด Node32s



<<ch03-22.tif>> รูปที่ 3-22 คลิก Upload > Save and build

3.4.14 กรุณารอจนกว่าโปรแกรมอัปโหลดเสร็จเรียบร้อยแล้ว โปรแกรมจะแสดงข้อความที่อยู่ในคำสั่ง printf ออกมาทางหน้าจอ Serial Monitor ในที่นี้แสดงข้อความ Hello world ออกทางหน้าจอ Serial Monitor ดังรูป 3-23

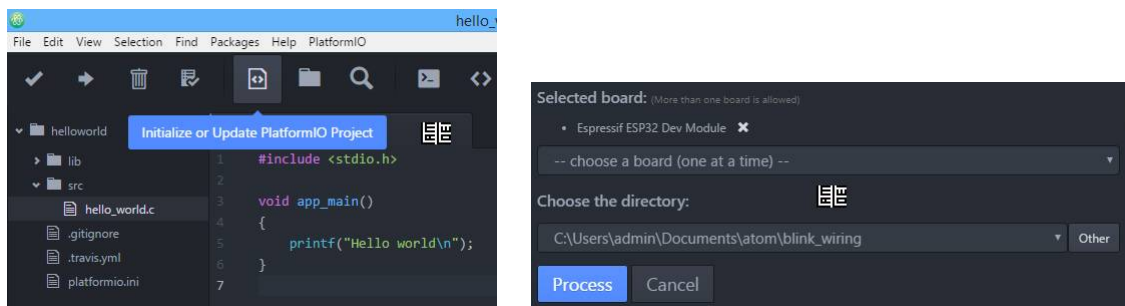


<<ch03-23.tif>> รูปที่ 3-23 แสดงข้อความ Hello world ออกทางหน้าจอ

### 3.5 ไฟกระพริบแบบ wiring

โค้ดโปรแกรมเขียนเหมือน Arduino IDE ทุกประการ แต่ต้องเพิ่ม #include "Arduino.h" ในบรรทัดแรก

3.5.1 คลิกปุ่ม Initialize or Update PlatformIO Project เพื่อสร้างโครงการใหม่ จากนั้นเลือกบอร์ดและโฟลเดอร์ที่ต้องการ



<<ch03-24.tif>> รูปที่ 3-24 สร้าง project ใหม่ <<ch03-25.tif>> รูปที่ 3-25 เลือกบอร์ดและโฟลเดอร์

3.5.2 สร้างไฟล์ใหม่ชื่อ blink\_wiring.cpp แล้วพิมพ์โค้ดข้างล่างนี้ (นามสกุลของไฟล์ต้องเป็น .cpp นะครับ) แล้วอัปโหลดเข้า Node32s จะเห็นว่าไฟบนบอร์ดจะกระพริบ จากนั้นลองเปลี่ยนตัวเลข delay เพื่อดูความเปลี่ยนแปลง

```
#include "Arduino.h"
```

```
#define LED_BUILTIN 2
```

```
void setup() {
```

```
    pinMode(LED_BUILTIN, OUTPUT);
```

```
}
```

```
void loop() {
```

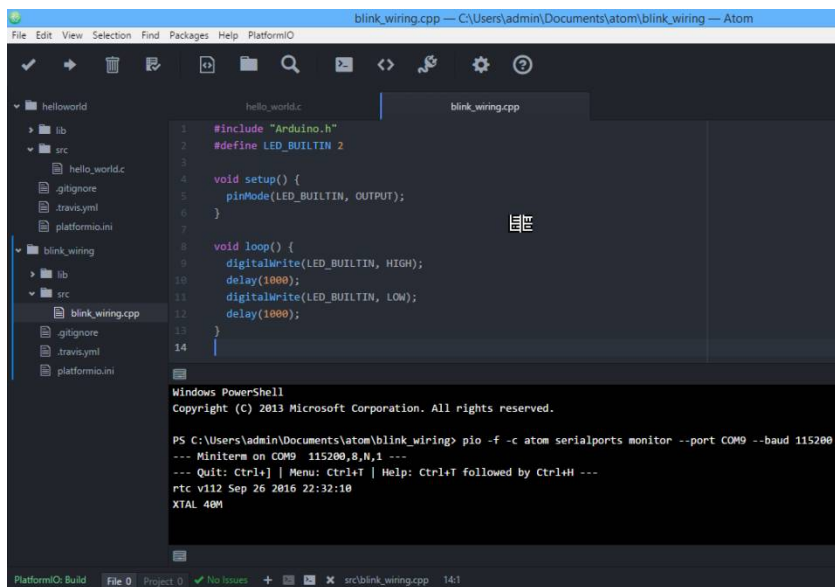
```
    digitalWrite(LED_BUILTIN, HIGH);
```

```
    delay(1000);
```

```
    digitalWrite(LED_BUILTIN, LOW);
```

```
    delay(1000);
```

```
}
```

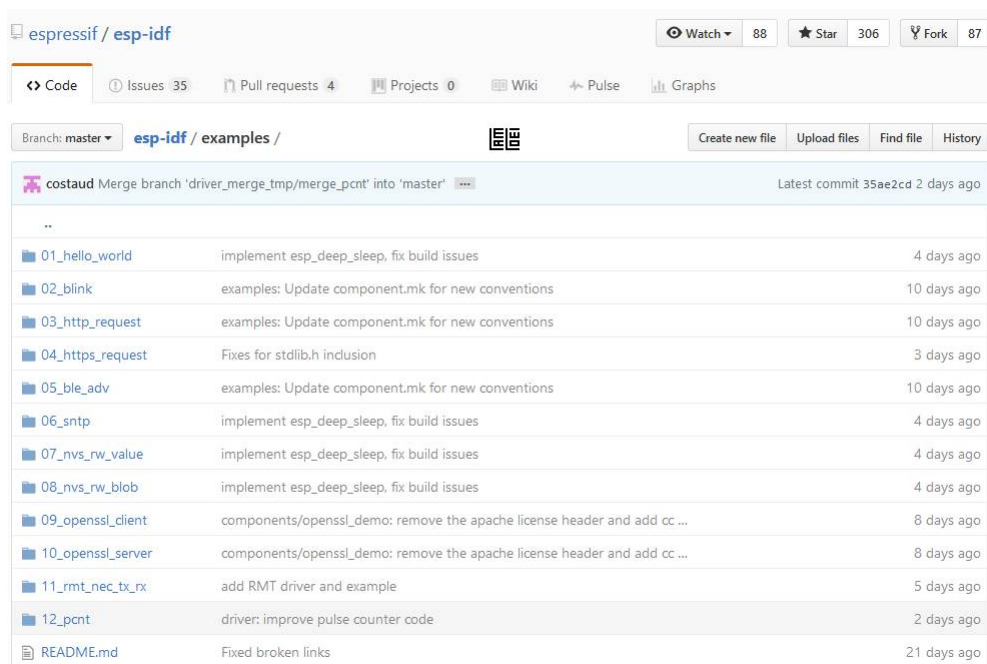


<<ch03-26.tif>> รูปที่ 3-26 โค้ดโปรแกรมลง Node32s

### 3.6 ไฟล์ตัวอย่าง esp-idf

คุณสามารถเปิดไฟล์ตัวอย่างขึ้นเพื่อมาศึกษาได้นะครับ แต่โค้ดตัวอย่างนี้ยังเป็น beta อยู่ อาจต้องมีการปรับแก้โค้ดบางส่วน เพื่อให้โค้ดโปรแกรมสามารถทำงานได้ ในหัวข้อนี้จะนำไฟล์ตัวอย่าง hello world มาโค้ดโปรแกรมลง Node32s

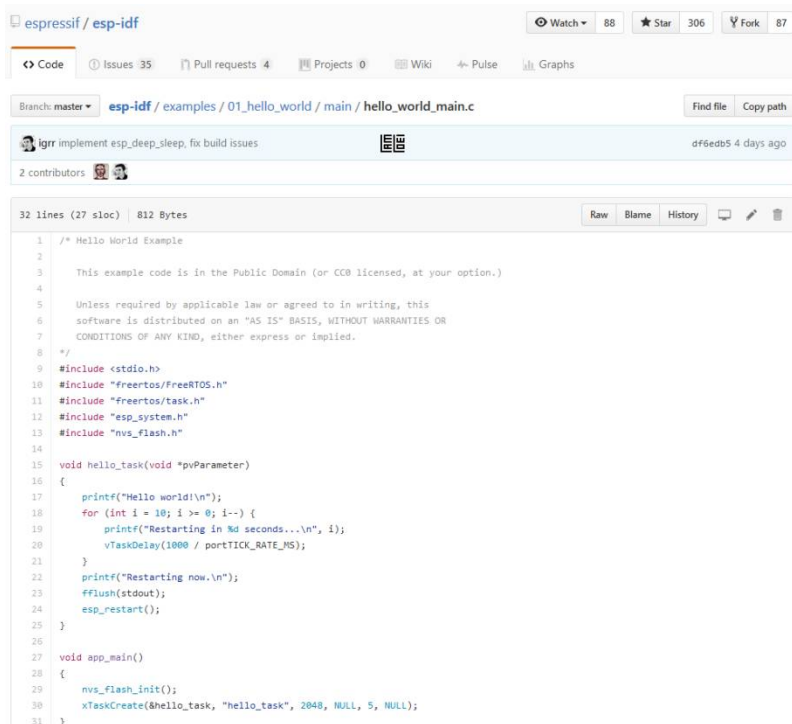
3.6.1 เปิดบราวเซอร์ไปที่ <https://github.com/espressif/esp-idf/tree/master/examples> คุณจะเห็นตัวอย่างโค้ด esp-idf ในที่นี้มี 12 ตัวอย่าง แต่เพื่อไม่ให้หนังสือเล่มนี้มีขนาดหนามาก ผู้เขียนจะโค้ดโปรแกรม 01\_hello\_world เพียงโปรแกรมเดียวเท่านั้น



espressif / esp-idf		Watch 88	Star 306	Fork 87
Code Issues 35 Pull requests 4 Projects 0 Wiki Pulse Graphs				
Branch: master esp-idf / examples / Create new file Upload files Find file History				
costaud Merge branch 'driver_merge_tmp/merge_pcmt' into 'master' Latest commit 35ae2cd 2 days ago				
01_hello_world	implement esp_deep_sleep, fix build issues	4 days ago		
02_blink	examples: Update component.mk for new conventions	10 days ago		
03_http_request	examples: Update component.mk for new conventions	10 days ago		
04_https_request	Fixes for stdlib.h inclusion	3 days ago		
05_ble_adv	examples: Update component.mk for new conventions	10 days ago		
06_snmp	implement esp_deep_sleep, fix build issues	4 days ago		
07_nvs_rw_value	implement esp_deep_sleep, fix build issues	4 days ago		
08_nvs_rw_blob	implement esp_deep_sleep, fix build issues	4 days ago		
09_openssl_client	components/openssl_demo: remove the apache license header and add cc ...	8 days ago		
10_openssl_server	components/openssl_demo: remove the apache license header and add cc ...	8 days ago		
11_rmt_nec_tx_rx	add RMT driver and example	5 days ago		
12_pcmt	driver: improve pulse counter code	2 days ago		
README.md	Fixed broken links	21 days ago		

<<ch03-27.tif>> รูปที่ 3-27 ไฟล์ตัวอย่างทั้งหมด

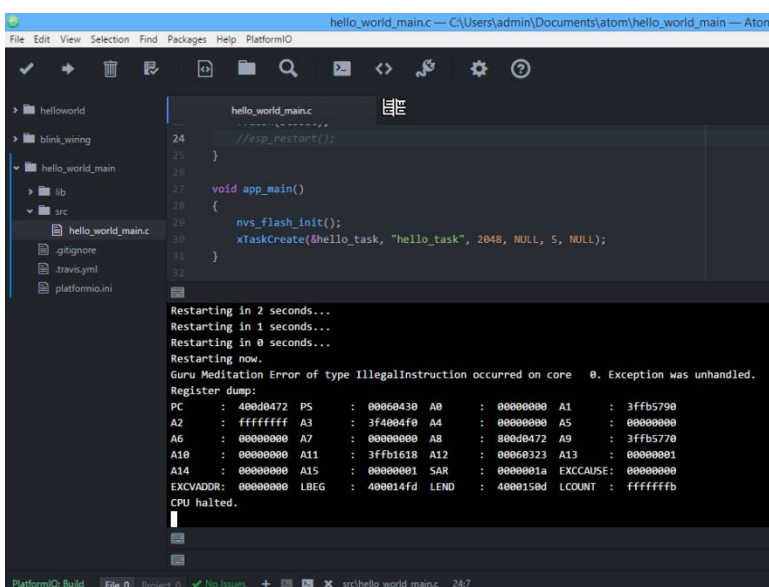
3.6.2 คลิก 01\_hello\_world แล้วคลิก main แล้วคลิก hello\_world\_main.c คุณจะเห็นโค้ดโปรแกรม ให้คลิกปุ่ม Raw เพื่อแสดงโค้ด จากนั้นให้คัดลอกโค้ดทั้งหมด



```
1  /* Hello World Example
2
3  This example code is in the Public Domain (or CC0 licensed, at your option.)
4
5  Unless required by applicable law or agreed to in writing, this
6  software is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
7  CONDITIONS OF ANY KIND, either express or implied.
8  */
9  #include <stdio.h>
10 #include "freertos/FreeRTOS.h"
11 #include "freertos/task.h"
12 #include "esp_system.h"
13 #include "nvs_flash.h"
14
15 void hello_task(void *pvParameter)
16 {
17     printf("Hello world!\n");
18     for (int i = 10; i >= 0; i--) {
19         printf("Restarting in %d seconds...\n", i);
20         vTaskDelay(1000 / portTICK_RATE_MS);
21     }
22     printf("Restarting now.\n");
23     fflush(stdout);
24     esp_restart();
25 }
26
27 void app_main()
28 {
29     nvs_flash_init();
30     xTaskCreate(&hello_task, "hello_task", 2048, NULL, 5, NULL);
31 }
```

<<ch03-28.tif>> รูปที่ 3-28 คัดลอกไฟล์ตัวอย่าง hello\_world\_main.c

3.6.3 ให้สร้างโปรเจกต์ใหม่ สร้างไฟล์ชื่อ hello\_world\_main.c แล้ววางโค้ดที่คุณคัดลอกไว้ลงในไฟล์นี้ เนื่องจากยังเป็นโค้ดตัวอย่างทดลอง ให้คุณคอมเมนต์ // บรรทัดที่ 24 เป็น //esp\_restart(); ดังรูป 3-29 จากนั้นอัปโหลดลง Node32s จะเห็นมีข้อความ Restarting in 9 – 0 second... แล้ว cpu ก็หยุดทำงาน



```
24 //esp_restart();
25 }
26
27 void app_main()
28 {
29     nvs_flash_init();
30     xTaskCreate(&hello_task, "hello_task", 2048, NULL, 5, NULL);
31 }
32
```

Restarting in 2 seconds...  
Restarting in 1 seconds...  
Restarting in 0 seconds...  
Restarting now.  
Guru Meditation Error of type IllegalInstruction occurred on core 0. Exception was unhandled.  
Register dump:  
PC : 400d0472 PS : 00060430 AB : 00000000 A1 : 3ffb5790  
A2 : ffffffff A3 : 3f4004f0 A4 : 00000000 A5 : 00000000  
A6 : 00000000 A7 : 00000000 A8 : 800d0472 A9 : 3ffb5770  
A10 : 00000000 A11 : 3ffb1618 A12 : 00000323 A13 : 00000001  
A14 : 00000000 A15 : 00000001 SAR : 0000001a EXCCAUSE : 00000000  
EXCVADDR : 00000000 LBEG : 400014fd LEND : 4000150d LCOUNT : ffffffff  
CPU halted.

<<ch03-29.tif>> รูปที่ 3-29 ผลลัพธ์ที่ได้จากการโค้ดโปรแกรม hello\_world\_main.c

สุดท้ายของบท คุณจะใช้ Arduino IDE เขียนโค้ดก็ได้ แต่ถ้าจะใช้คุณสมบัติพิเศษของ Node32s เช่น Bluetooth ในขณะที่ผู้เขียนเขียนหนังสือเล่มนี้อยู่ ต้องเขียนโค้ดแบบ esp-idf เท่านั้นถึงจะใช้งานได้ ฉะนั้นลองศึกษาโค้ดโปรแกรม esp-idf จากตัวอย่างอื่นๆ ทั้ง 12 ตัวอย่าง ตัวอย่างเหล่านี้ยังเป็น beta อยู่ นั่นก็หมายความว่า ต้องมีการปรับแก้โค้ดบางส่วน เช่น ต้องใส่ // คอมเมนต์ลงในโค้ดส่วนที่ทำให้เกิด error ออกไป เพื่อให้โค้ดที่เหลือสามารถทำงานได้นั่นเอง

== END ==