CavyIoT-DevBoard

All about CavyIoT-DevBoard

Click here to download pdf

CavyIoT.nodemcu-V0.03.bin firmware converts your hardware (NodeMCU, ESP8266 and WeMos D1 R1) to CavyIoT-DevBoard

Index

Introduction

Firmware is released as free open source, for the developers by Cavy Agrotronics. This firmware is useful to engineering students, teachers, hobbyist, enthusiast, programmers, for availing CavyIOT services. This firmware is free for all who want to add the power of IoT in their product.

It is language independent, means no matters what programming language the user is using in his project. User can interface **CavyIoT Development Board** with simple **Send-Response protocol.**

Flashing firmware converts IC ESP8266 into CavyIoT Development Board (CavyIoT-DevBoard). Developer of any community can develop and deploy a powerful IoT system/application to the end user. Simple interfacing API architecture makes easier for developers in development process.

Features of our platform and development board-

Hardware-

- Platform Independant
- Simple API
- Settings for data usage
- Local data backup

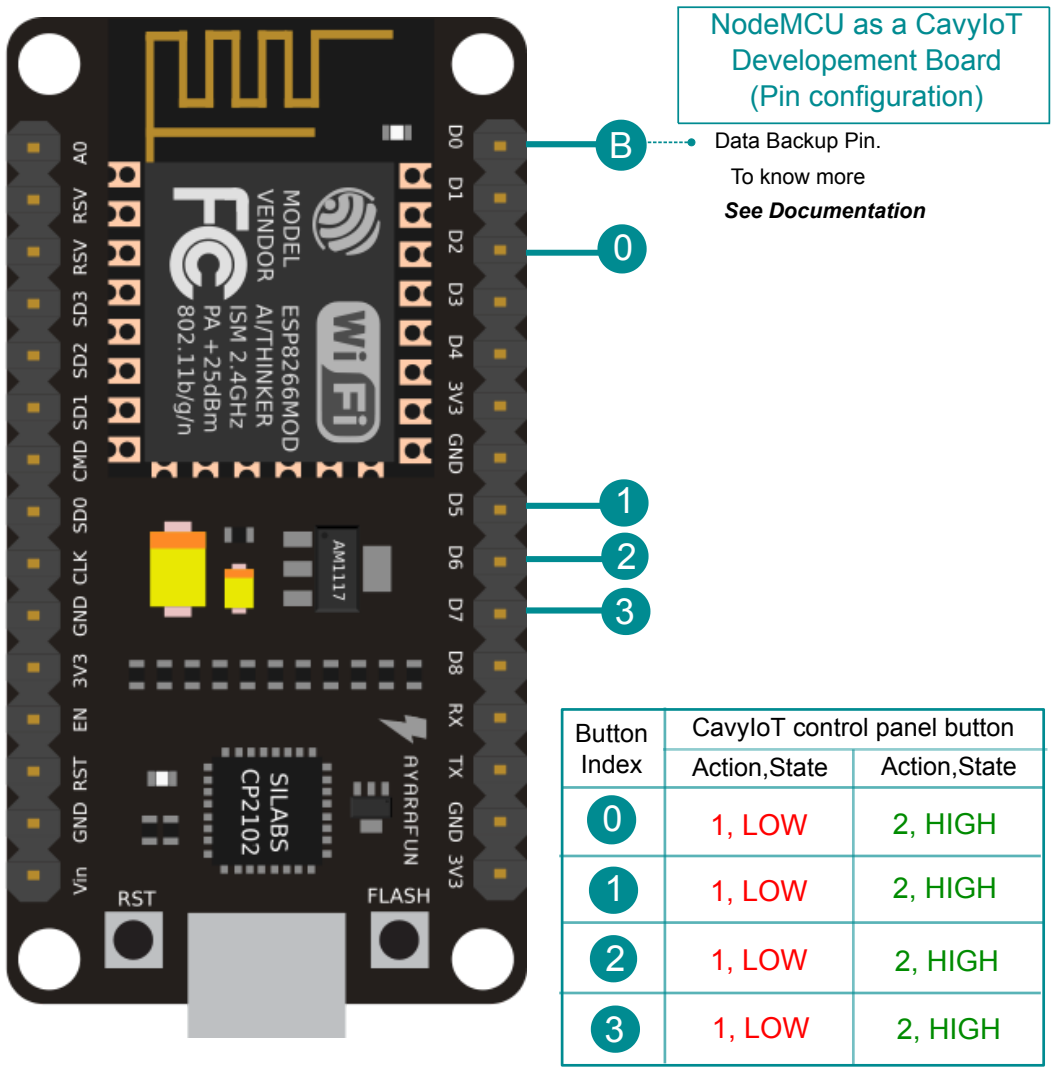
Software-

The CavyIoT service provides the subscriber access to CavyIoT-DevBoard all over the internet anytime/anywhere via Control Panel Control Panel is standard GUI software accessible over internet using standard internet browser.

Features of Control Panel-

- Data visualization
- Report generation
- Full control over system
- Bi-directional & Synchronized with DevBoard
- Buttons to set the output of board’s output pins
- Automation
- Log file of all operations and device status
- Triggers
- Online tools for data analysis
- Automatic and manual mode changing for operation

- Live charts and Gauges
- Minimum Data consumption and maximum utility



Button Index	CavyIoT control panel button	
	Action,State	Action,State
0	1, LOW	2, HIGH
1	1, LOW	2, HIGH
2	1, LOW	2, HIGH
3	1, LOW	2, HIGH

Pin mapping of NodeMCU after flashing firmware.

CavyIoT-DevBoard's baud rate is 9600 and is fixed for serial communication (via rx , tx pins). CavyIoT-DevBoard recognize a line of Instruction when is send within enclosed chevrons followed by a carriage return. Device sends request to the CavyIoT server via internet, in defined interval to get the status of Control Panel, for an example if the user clicks the button on control panel, the device will receive control signal issued in the next ping (request). The pinging keeps the device in synchronization with the Control Panel. Default ping interval is 5 sec, setting ping interval to higher values slows the response time of device and vice versa . And sensor data sent to CavyIoT-DevBoard is get displayed on Control Panel for monitoring and processing.

The **Output Pins** of CavyIoT-DevBoard shown above are responsive to the corresponding Buttons on the Control Panel. The frequency of sending sensor data to server and the frequency of receiving control signal from Control Panel of device, depends upon ping interval and it is configurable. User can set **Ping Interval** via programming device or from the Control Panel.

***Local Data Backup-**

In Dev-Board there is inbuilt provision for keeping records of the all operations and state of device. It keeps the record of records all data, operations and status of the device. As you know, the output pins of DevBoard are responsive to the corresponding Buttons on the Control Panel. If the button is clicked on Control Panel causes the state of output pin of device get changed or if user switches operation mode ie from **AUTO** to **MANUAL** causes a change in status of device. This change in operation mode or the change in state of output pin generates an event to write record to the end of log file in DevBoards memory. Even if any such event does not occurs the device records the state of device in predefined interval. *Local data backup is the most important as it is useful for data analysis, and for tracing the track of operation done. Recording interval is configurable and can be set from Control Panel too!*

How To download data (CSV file) from device-

Keep the data-backup pin **B** (shown in above image) LOW and restart the device.This will create a Wi-Fi hotspot to serve a page to download data log file. The default Wi-Fi ssid is **CavyIoT** and password **admin@123**. Browse to url ***http://100.100.100.100*** where the page with link for download will be available by local Wi-Fi web server of device. User can set **Hotspot parameters (SSID, password)** for his privilege, with the help of hardware API.

Interfacing

(CavyIoT-DevBoard is abbreviated as **device** for our convenience)

The basic interface is to send device a line of Instruction within enclosed chevrons- <command>

```
<ShowDemo(WiFiSSID, WiFiPassword, CavyIoT-username,CavyIoT-password, Device-name)>
```

then wait for the proper response message with open chevrons (>) followed by ok! or error!(**>ok!** or **>Error!**) . This signals device has completed the parsing and executed the command. At times, CavyIoT-DevBoard may not respond immediately. This happens when device is busy doing something else mostly happened with

```
<ShowDemo()>
```

The **response messages** are strings terminated by a return. These messages are "pushed" from device to the user in response to a query or instruction to let the user what happened.

The primary way to talk to device is performed by sending it a command string of characters (Instruction within enclosed chevrons **< >**), followed by a carriage return. Device will then process the string and then reply back with a response message, also terminated by a return, to tell you how it went.

Writing an Interface for CavyIoT-DevBoard

Streaming Protocol: Simple Send-Response [Recommended]

The send-response streaming protocol is the most fool-proof and simplest method to stream a command to device. The host PC/ MCU interface simply sends a line of command to CavyIoT-DevBoard and waits for an **Ok!** or **Error!** Response message before sending the next line of command.

So, no matter if device needs to wait for room in the look-ahead planner buffer to finish parsing and executing the last line of command or if the the host computer is busy doing something, this guarantees both to the host PC /MCU and device, the programmed command has been sent and received properly.

(Note : All commands for CavyIoT-DevBoard are not **Case Sensitive!**)

Commands-

1. **DefineButtonLables**
   - **Syntax-**

     ```
     <DefineButtonLables(Label (0), Action1, Action2, Label (1), Action1, Action2, Label (2),
     Action1, Action2, Label (3), Action1, Action2)>
     ```

   - **Description-**

     Sets the Label for Button(index) and Labels for the its actions.

     Button when is in Action1 OUTPUT Pin(index) is **LOW**
     Button when is in Action2 OUTPUT Pin(index) is **HIGH**

   - **Default-**

Button(n)	Label	Action1	Action2
0	LED1	**ON**	**OFF**
1	LED2	**ON**	**OFF**
2	LED3	**ON**	**OFF**
3	LED4	**ON**	**OFF**

   - **Response Message-**

     onSuccess- <Buttons labelled >Ok!
     onError- <>Error!

2. **StartDevice**
   - **Syntax-**

     ```
     <StartDevice( WiFiSSID, WiFiPassword, CavyIoT-username, password, device)>
     ```

   - **Description-**

     Connects to the Wi-Fi and starts pinging with regular interval.

- **Default-**

  no defaults

- **Response Message-**

  onSuccess- <Device Started ! Ready to send data >Ok!
  onError- <>Error!

- **Ping Response Message-**

  onSuccess- <JSON String of device status >Ok!
  onError- <>Error

3. **UpdateSensorData**
   - **Syntax-**

     ```
     <UpdateSensorData(Sensor1, S1_Value, S1_Unit, Sensor2, S2_Value, S2_Unit, Sensor3,
     S3_Value, S3_Unit, Sensor4, S4_Value, S4_Unit)>
     ```

   - **Description-**

     Sends data to the server.

   - **Default-**

     no defaults

   - **Response Message-**

     onSuccess- <JSON String of device status >Ok!
     onError- <>Error

   - **Example-**

     ```
     <UpdateSensorData( Temperature, 35.12, C, humidity, 19.00, Rh, Speed, 16.00, km/hr,
     pressure, 4.12, Psi)>
     ```

   - **Response Message-**

     onSuccess- < {"Device":"Demo","Mode":"MANUAL","Buttons":["OFF","ON","OFF","OFF"]} >Ok!
     onError- <>Error

4. **SetPingInterval**
   - **Syntax-**

     ```
     <SetPingInterval(seconds)>
     ```

   - **Description-**

     Sets the *Ping Interval.*

   - **Default-**

     5

   - **Response Message-**

     onSuccess- < Ping Interval is set >Ok!
     onError- <>Error!

   - **Example-**

     ```
     <SetPingInterval(10)>
     ```

   - **Response Message-**

     onSuccess- < Ping Interval is set >Ok!
     onError- <>Error!

5. **SetRecordingInterval**
   - **Syntax-**

     ```
     <SetRecordingInterval(minutes)>
     ```

   - **Description-**

     Sets the *Recording Interval.*

   - **Default-**

     5

   - **Response Message-**

     onSuccess- < Recording Interval is set >Ok!
     onError- <>Error!

   - **Example-**

     ```
     <SetRecordingInterval(5)>
     ```

   - **Response Message-**

     onSuccess- < Recording Interval is set >Ok!
     onError- <>Error!

6. **SetBackupHotspot**
   - **Syntax-**

     ```
     <SetBackupHotspot(Hotspot_name, Hotspot_password)>
     ```

   - **Description-**

     Sets the *Wi-Fi Hotspot credentials* for data backup

   - **Default-**

     Hotspot_name- **CavyIoT**
     Hotspot_password- **admin@123**

   - **Response Message-**

     onSuccess- < Hotspot For BACKUP is defined >Ok!
     onError- <>Error!

   - **Example-**

     ```
     <SetBackupHotspot(BlueSky, jki125nmio)>
     ```

   - **Response Message-**

     onSuccess- < Hotspot For BACKUP is defined >Ok!
     onError- <>Error!

7. **ShowDemo**
   - **Syntax-**

     ```
     <ShowDemo(WiFiSSID, WiFiPassword, CavyIoT-username, password, device)>
     ```

   - **Description-**

     Connects to the Wi-Fi and starts pinging. And returns Device Status after each ping.

   - **Default-**

     no defaults

- **Response Message-**

  onSuccess- < A JSON string of device stautus >Ok!
  onError- <>Error

- **Example-**

```
<ShowDemo(BlueSky, jki125nmio, robert555, oltrtrtv45pA2@, Demo)>
```

- **Response Message-**

  onSuccess- < {"Device":"Demo","Mode":"MANUAL","Buttons":["OFF","ON","OFF","OFF"]} >Ok!
  onError- <>Error

Guide to flash CavyIoT firmware to ESP8266,NodeMCU or Wemos D1 R1 using Arduino IDE.

Step by step guide on how to flash firmware over the internet using arduino IDE

Install CavyIoTdevelopmentBoard.h as ZIP file in Arduino IDE

Using Example sketches in CavyIoTdevelopmentBoard.h you can flash firmware to **ESP8266**,**NodeMCU** or **Wemos D1 R1**. and library helps in communication between your hardware, CavyIoT Cloud and IoT Dashboard (Control Panel). ***CavyIoTdevelopmentBoard.h*** library is available as a downloadable ZIP. Starting with Arduino IDE version 1.0.5, you can install 3rd party libraries in the IDE

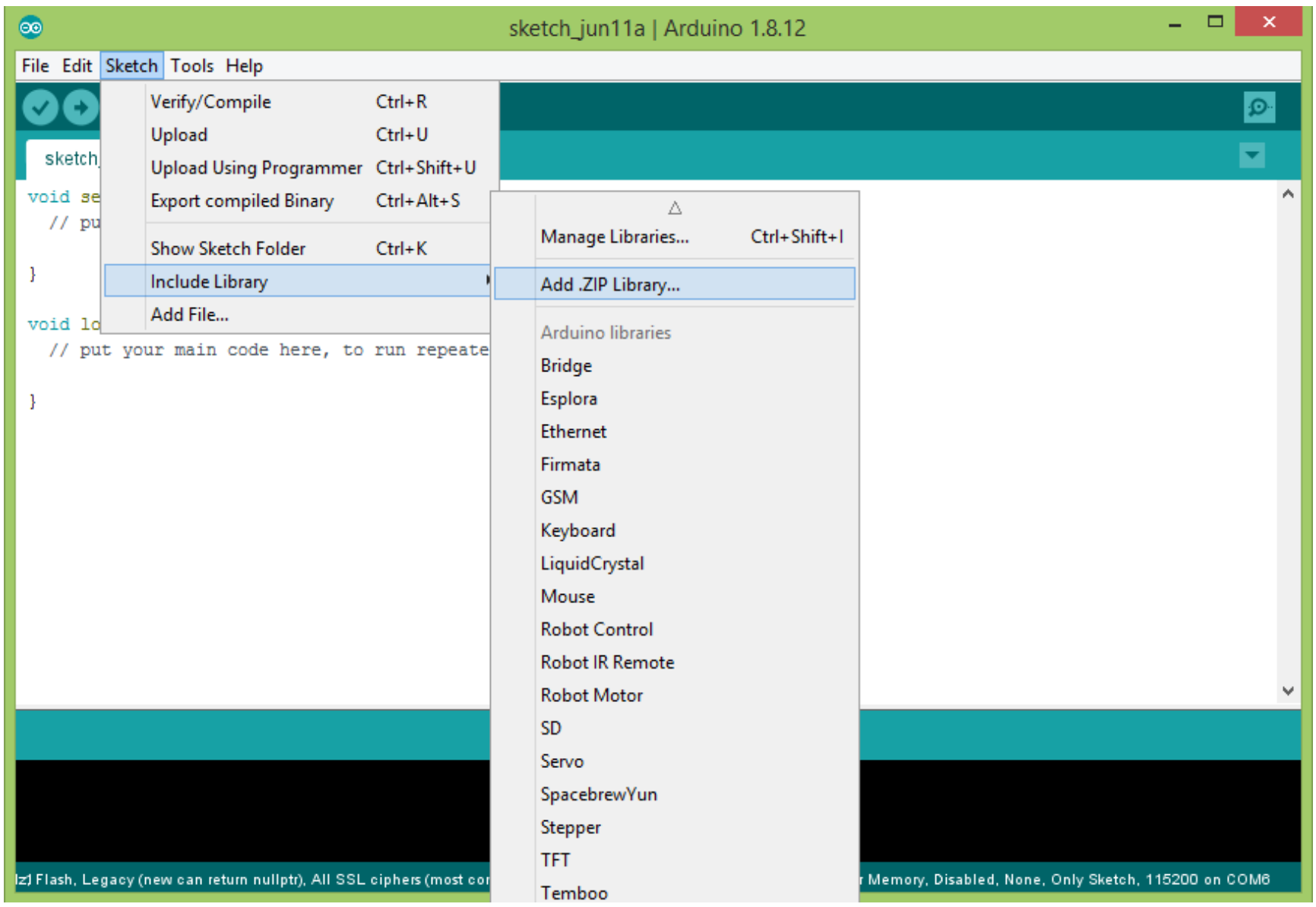
CavyIoTdevelopmentBoard.h Library by clicking the button:

Download CavyIoT Library

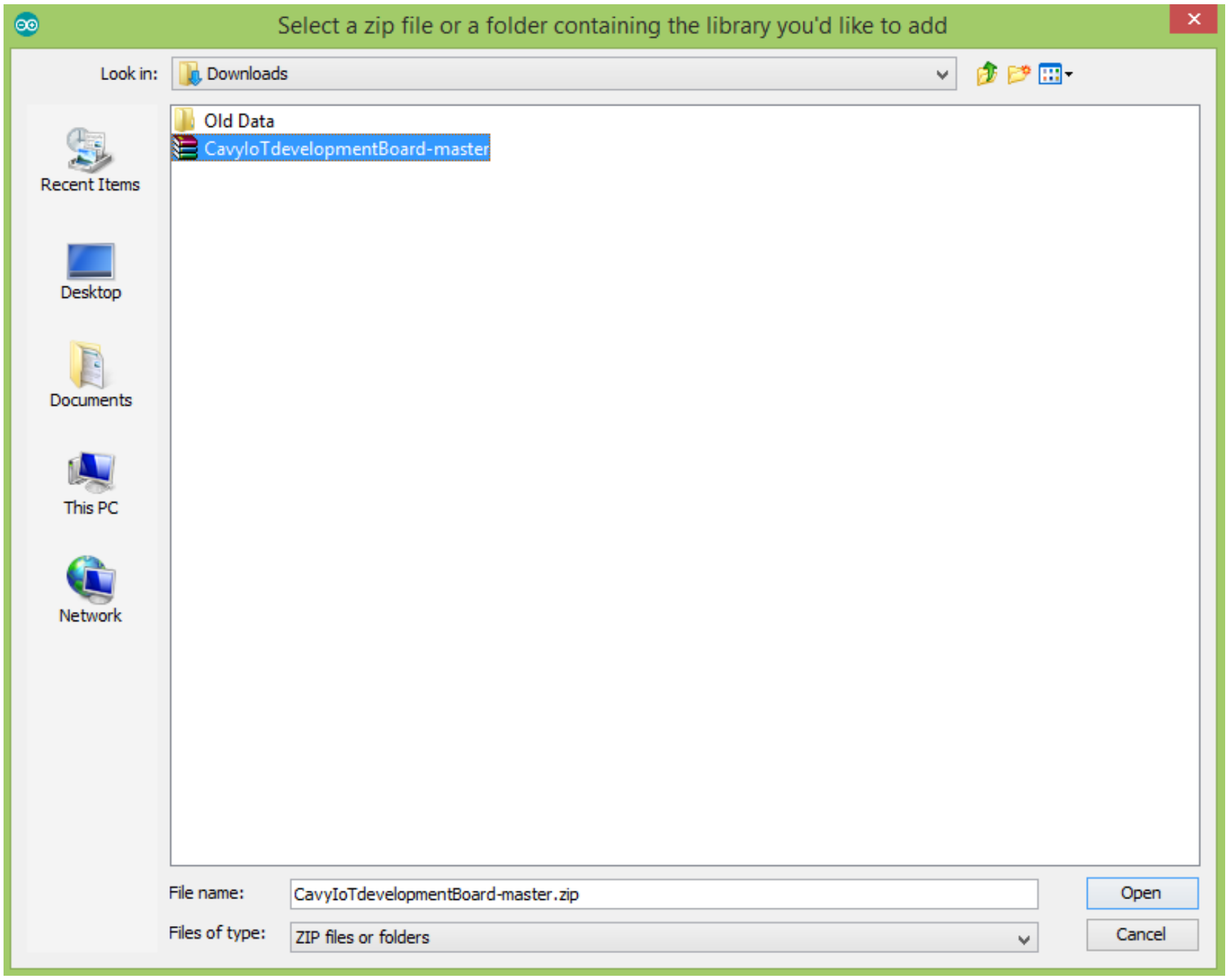
or Download it from GitHub GitHub

Do not unzip the downloaded library, leave as it is.

1. In the Arduino IDE, navigate to Sketch > Include Library > Add .ZIP Library. At the top of the drop down list, select the option to "Add .ZIP Library".



then select downloaded ZIP file and click 'Open'



2. Return to the Sketch > Include Library menu. menu. You should now see the library at the bottom of the drop-down menu. It is ready to be used in your sketch. The zip file will have been expanded in the libraries folder in your Arduino sketches directory.

The Library will be available to use in sketches, but with older IDE versions examples for the library will not be exposed in the File > Examples until after the IDE has restarted.

Troubleshooting

1. If you have troubles installing CavyIoT library, check Arduino guide for additional information.
2. Try deleting and re-installing CavyIoT Library - it helps in many cases
3. Make sure you don't have any duplicates of CavyIoTdevelopmentBoard.h in library folder.

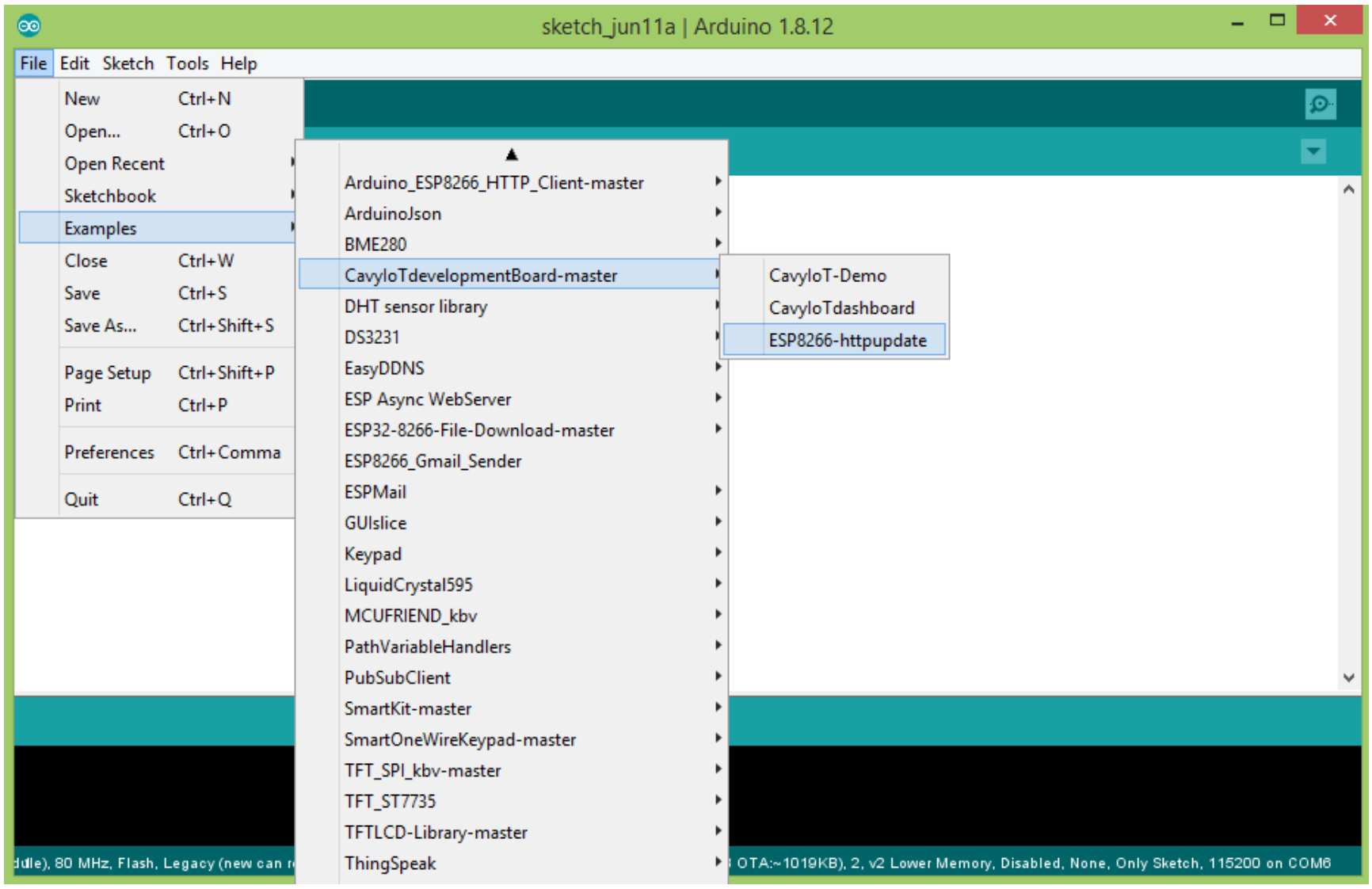
CavyIoT firmware converts your hardware (NodeMCU, ESP8266 and WeMos D1 R1) to CavyIoT-DevBoard

Flashing firmware in NodeMCU

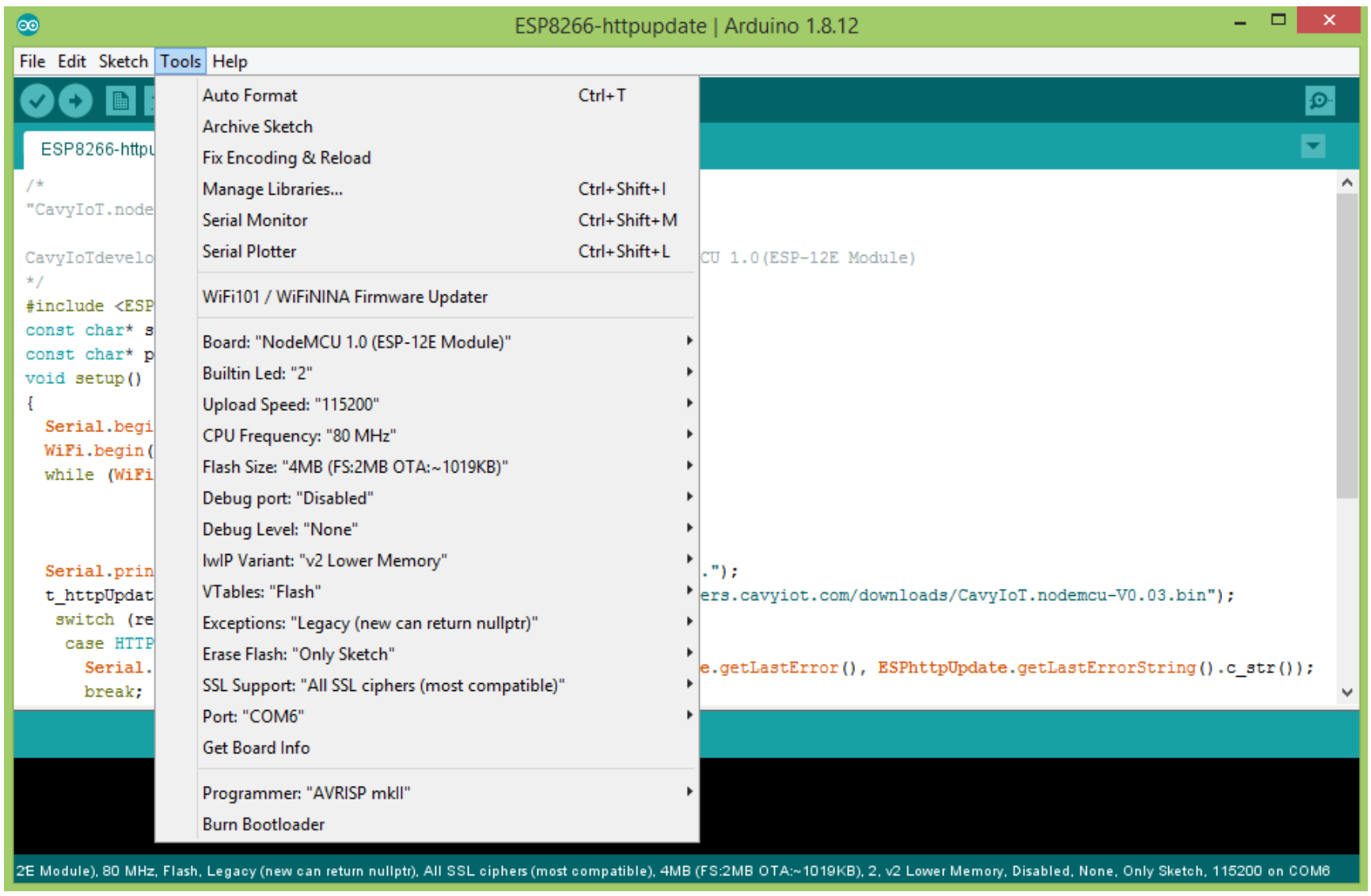
1. **If ESP8266 board in installed in your Arduino IDE Skip this step.**
   The ESP8266 community created an add-on for the Arduino IDE that allows you to program the ESP8266 using the Arduino IDE and its programming language.

   Start Arduino and open Preferences window.
   Enter ***https://arduino.esp8266.com/stable/package_esp8266com_index.json*** into Additional Board Manager URLs field. You can add multiple URLs, separating them with commas.
   Open Boards Manager from **Tools > Board menu** and find esp8266 platform. Select the version from a drop-down box. Click **install** button. Don’t forget to select your **NodeMCU 1.0 (ESP-12E Module)** board from **Tools > Board** menu after installation.
2. **Upload ESP8266-httpupdate sketch**

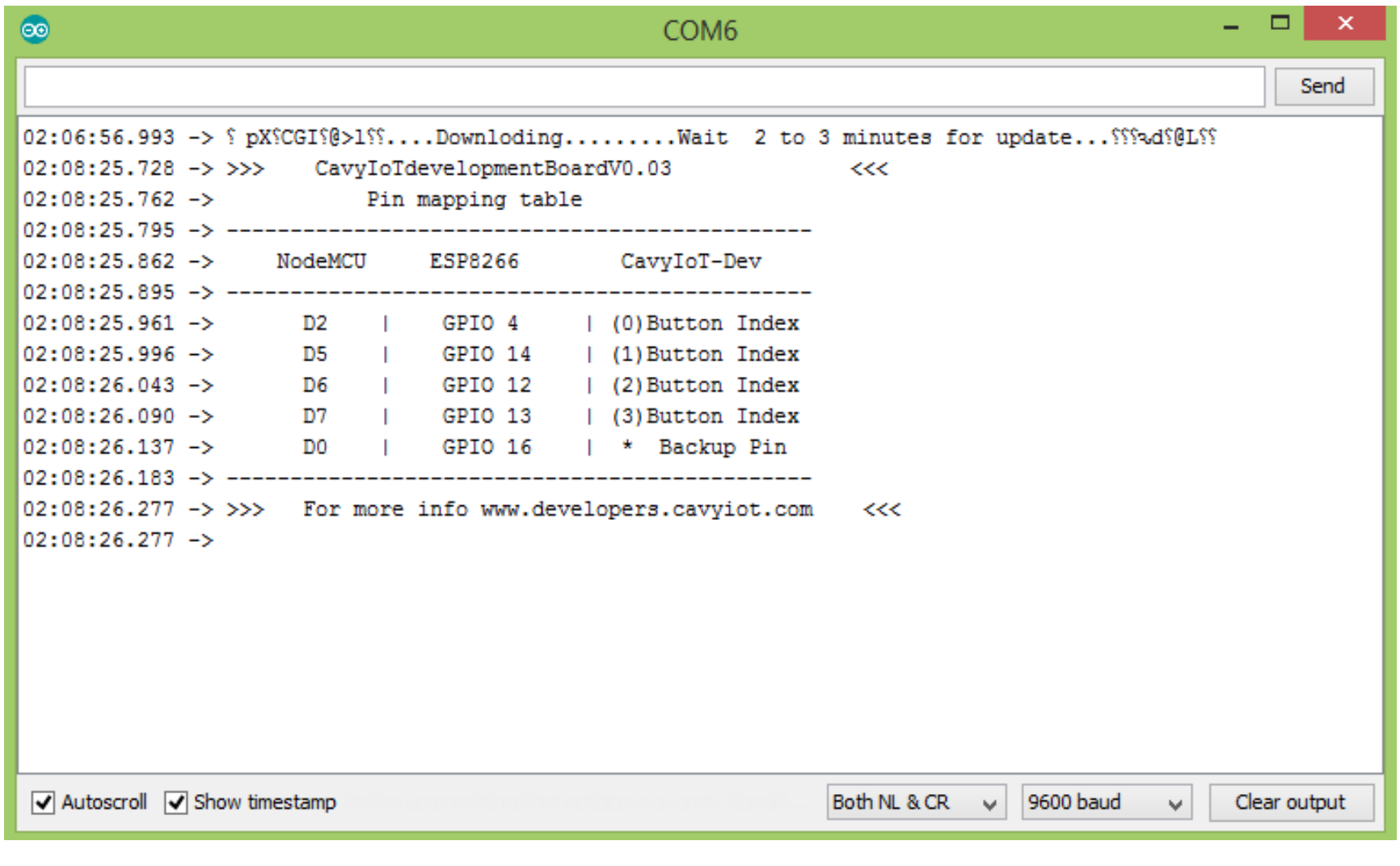
   In the Arduino IDE, navigate to **File > Examples > CavyIoTdevelopmentBoard-master > ESP8266-httpupdate** and open this example sketch.



Then select board and COM port properly. (NodeMCU is selected in this example)



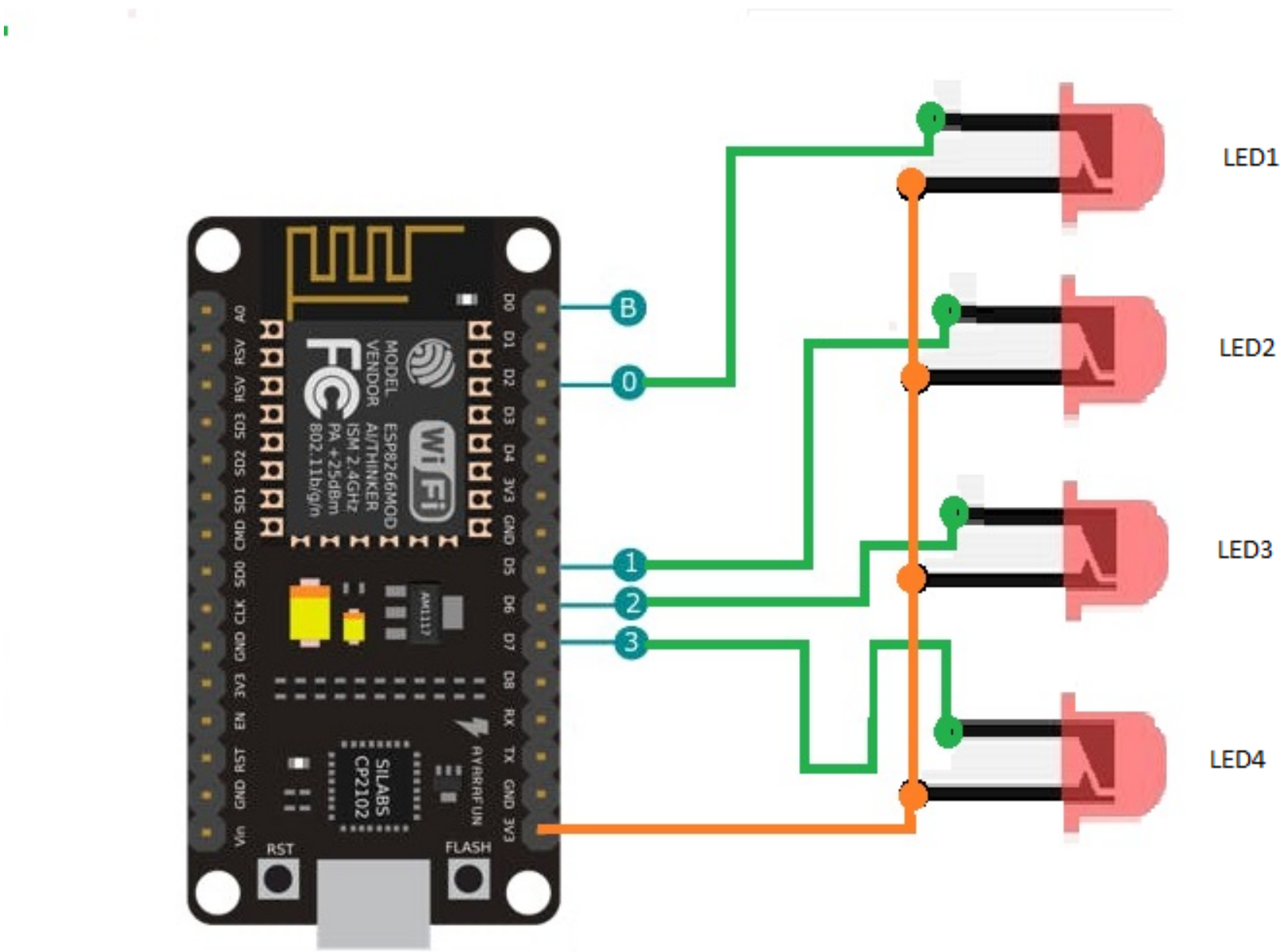
3. Before uploading **ESP8266-httpupdate.ino** sketch you need to replace Wi-Fi SSID and Password with your own. After Done Uploading open serial monitor **(Baudrate:9600,Both NL & CR)** and **Reset** the NodeMCU and wait for 2 minutes to complete flashing. You will see output on serial monitor as shown below:



Congratulations! Your CavyIoT-DevBoard is ready to use.

Testing Demo

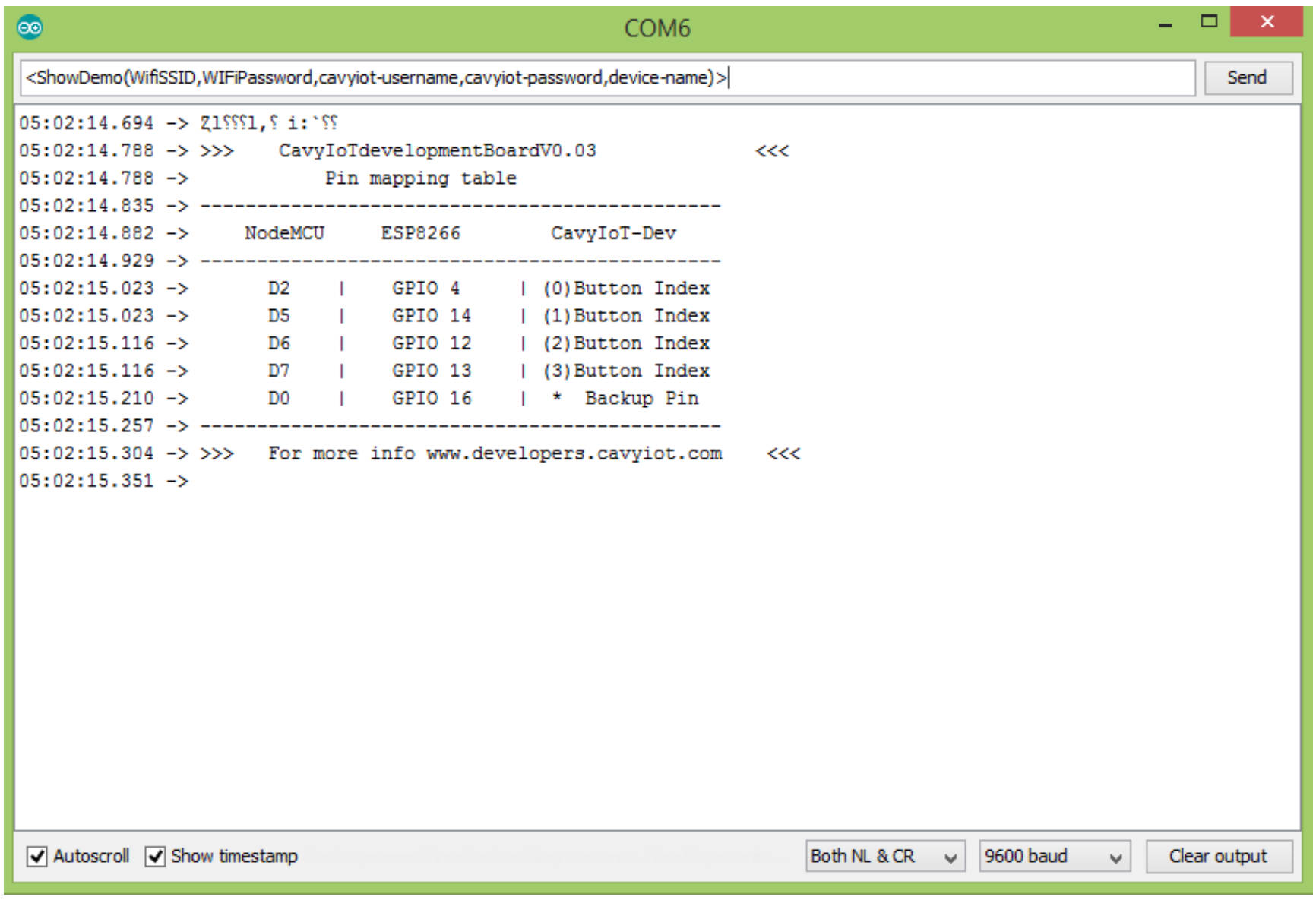
1. To test Dev-Board and to see how it works with remote control panel, connect four LED to NodeMCU(CavyIoT-DevBoard) as below.



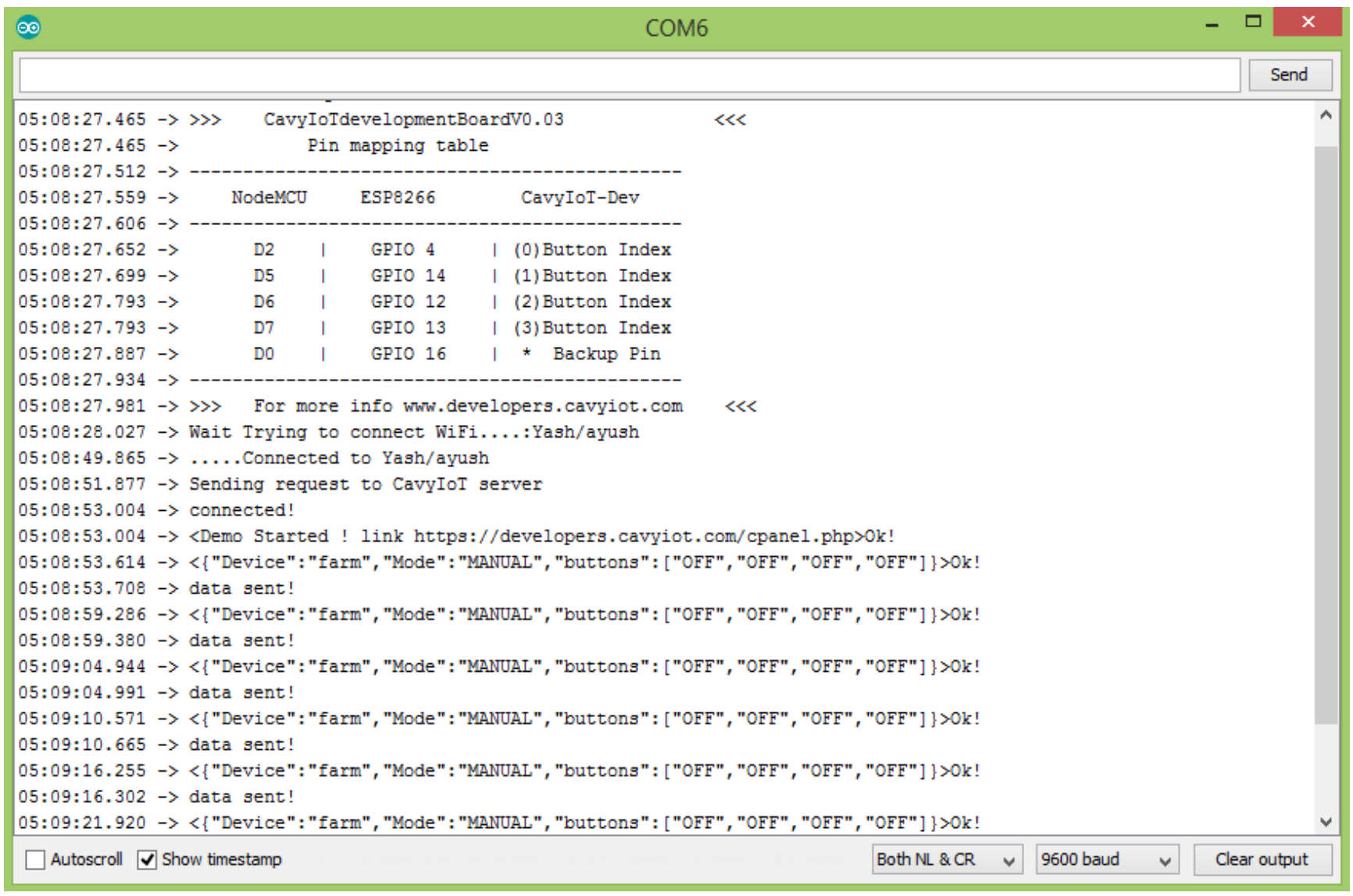
2. Open serial monitor and try typing the following commands, into the top area of the Serial Monitor that is level with the 'Send' button.

```
<ShowDemo(WiFiSSID, WiFiPassword, CavyIoT-username,CavyIoT-password, Device-name)>
```

Replace Wi-FiSSID, Wi-Fi Password, CavyIoT username, Password and Device Name with your own credentials.



3. After sending command with correct credentials CavyIoT-DevBoard will try to connect to your wifi and will start sending random values for dummy sensors to cavyIoT server. And start receiving Control Signals from **Control Panel** with default Button Lables as shown below:



Now You can control this Dev-Board from anywhere,anytime! Log in to your account and operate Control Panel. and try to..
Turn ON/OFF LED with the help of buttons.
Change the mode of operation (ie from 'AUTO' to 'MANUAL') and a lot.
You will see the output on serial monitor changes, the state of LED connected changes, and live charts and gauges for sensor data!

4. Click Here for more command reference for Dev-Board.

5. Click Here to know more about control panel.

Troubleshooting

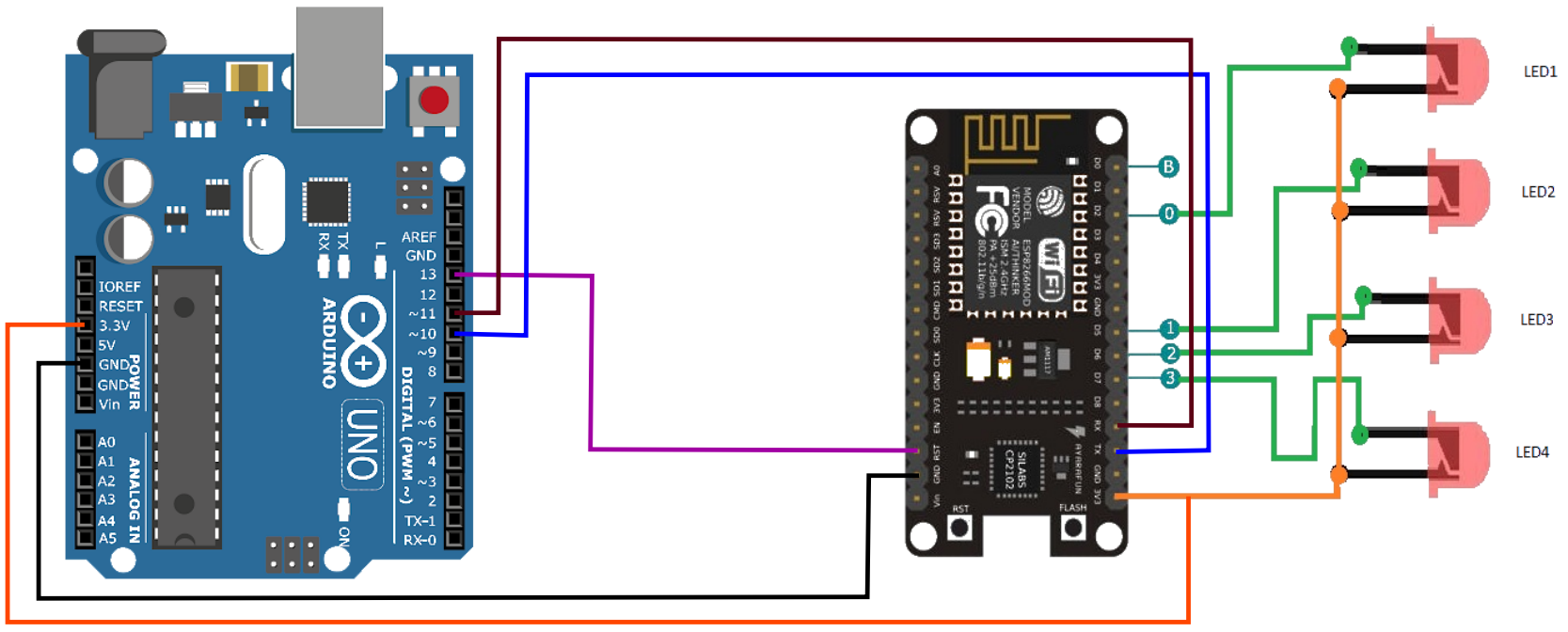
1. If you have troubles in flashing firmware check your wifi connectivity or internet connection.
2. Check your Wi-Fi SSID and Password is correct.
3. Make sure you selected board and COM port properly.

Interfacing Arduino UNO with CavyIoT-DevBoard

Step by step guide for intefacing with Arduino

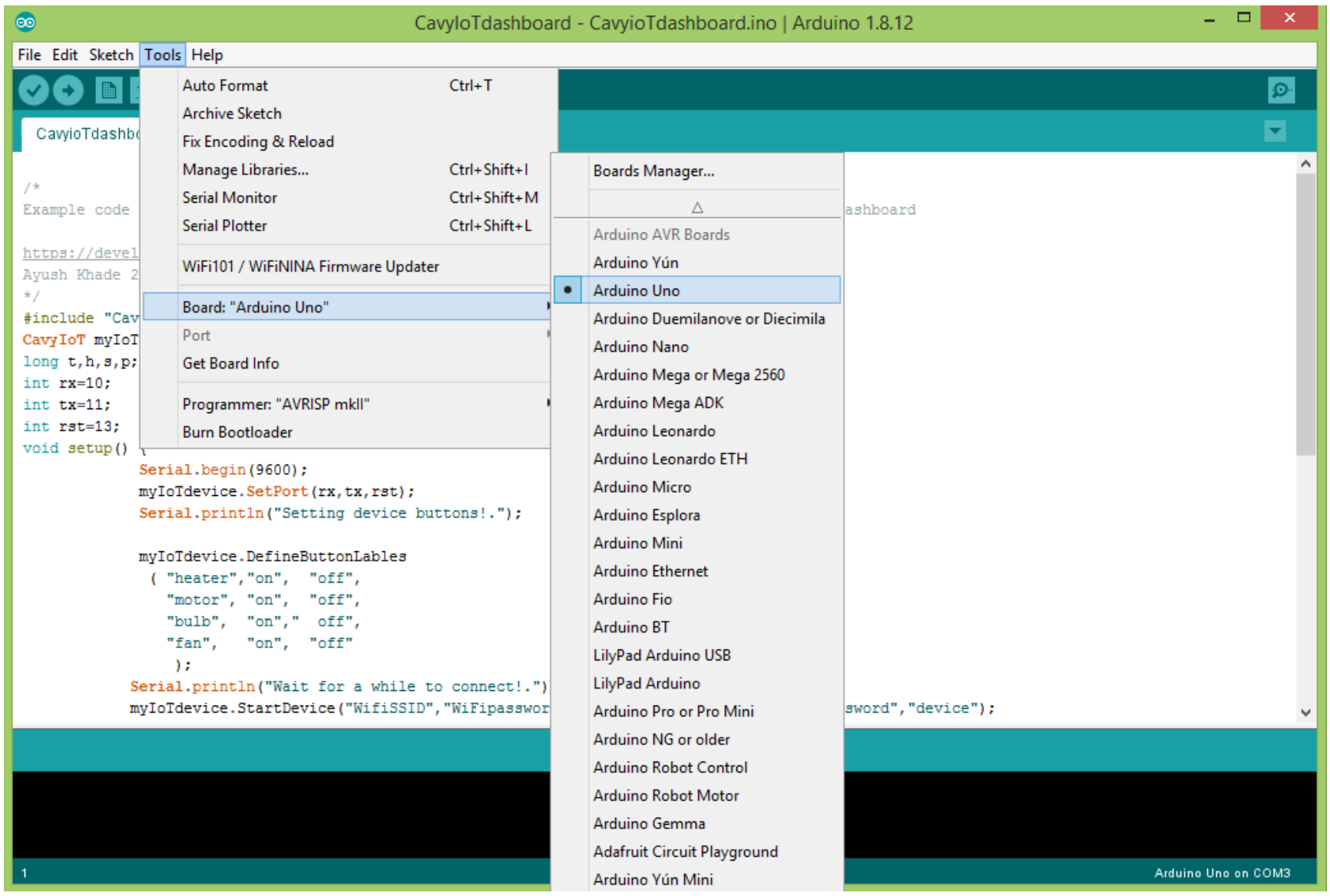
Connection details for interfacing

1. Make connection as below.

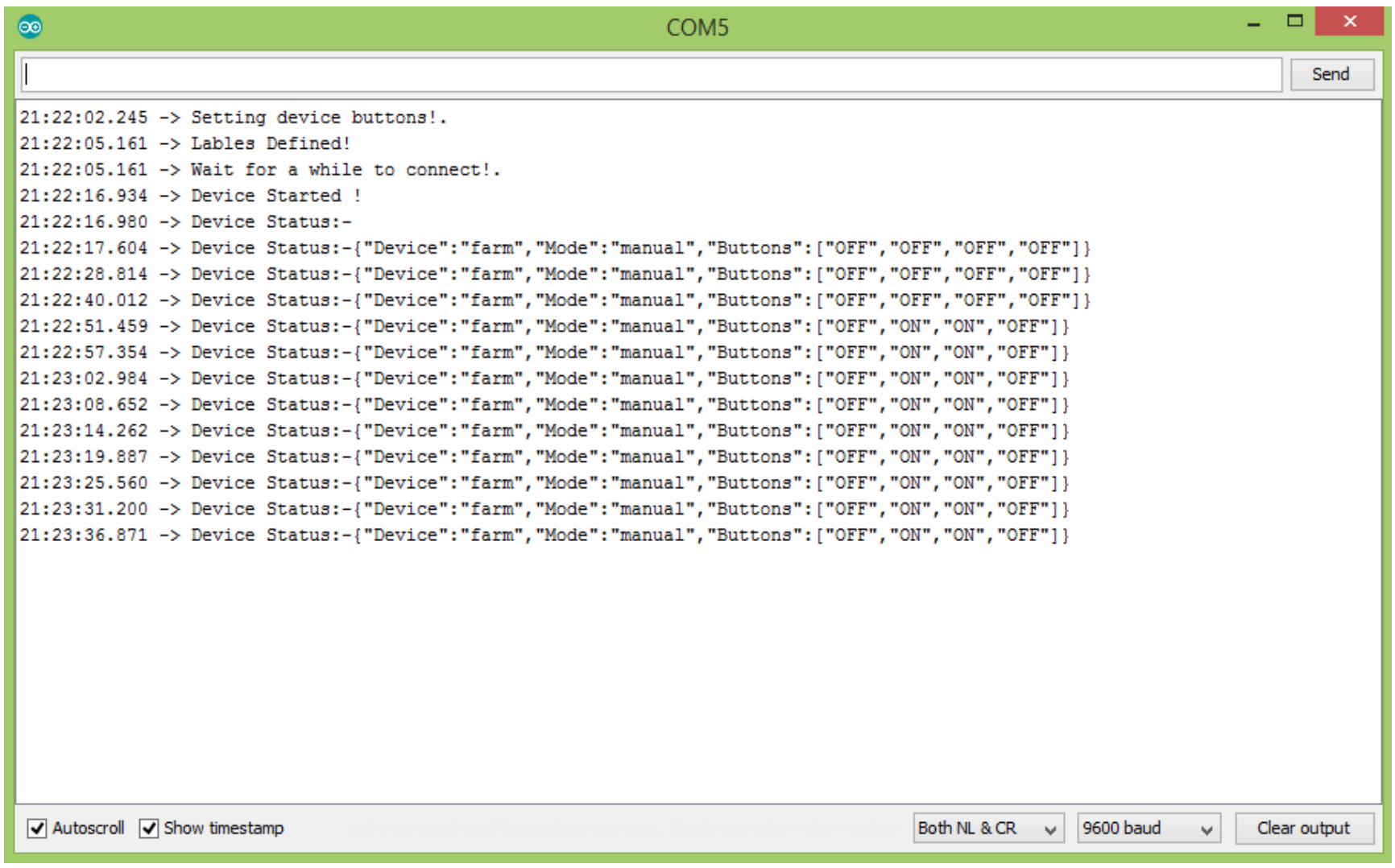


2. In the Arduino IDE, navigate to **File > Examples > CavyIoTdevelopmentBoard-master > CavyIoTdashboard** and open this example sketch.

Then select board and COM port properly. (Arduino-UNO is selected in this example)



3. Before uploading **CavyIoTdashboard.ino** sketch you need to replace Wi-Fi SSID and Password with your own. After Done Uploading open serial monitor **(Baudrate:9600,Both NL & CR)** and **Reset** the Arduino UNO and wait for a while. You will see output on serial monitor as shown below:



Congratulations! Your Arduino UNO is successfully interfacing with CavyIoT-DevBoard.

Now Login and to to CavyIoT Control Panel and Enjoy CavyIoT!

Troubleshooting

1. If you have troubles check your wifi connectivity or internet connection.
2. Check your Wi-Fi SSID and Password is correct.

3. Make sure you selected Arduino UNO board and COM port properly.
4. Check your connections.
5. Try resetting Arduino UNO.

Important links

1. How to set Triggers?
2. How to Edit Gauge?
3. How to Set Ping Interval from Control Panel?
4. How to Set Recording Interval for Data Backup from Control Panel?
5. How to Reset Device from Control Panel?
6. How to switch between AUTO-MANUAL modes of working?
7. How to convert Data Backup file (.CSV) downloaded from CavyIoT-DevBoard to PDF using CavyIoT Online tools?