

Bộ Giáo Dục Và Đào Tạo

Trường Đại Học Hoa Sen

Báo Cáo Cuối Kỳ Bộ Môn

Thực Hành Công Nghệ Thông Tin (2)

Đề tài: Chương trình dò nồng độ cồn

Giảng viên hướng dẫn: Trần Thị Trường Thi

Sinh viên thực hiện: Trần Gia Nguyên Phong

Mã số sinh viên: 22002575

Mục Lục

[Lời Mở Đầu 3](#_Toc143253926)

[Nội dung 4](#_Toc143253927)

[I. Chuẩn bị (Khái quát) 4](#_Toc143253928)

[1. Yêu cầu thiết bị và môi trường 4](#_Toc143253929)

[II. Cài đặt môi trường (1 vài ứng dụng) 10](#_Toc143253930)

[1. Nodejs 10](#_Toc143253931)

[2. Postman 12](#_Toc143253932)

[3. Mysql 13](#_Toc143253933)

[4. Arduino ide 14](#_Toc143253934)

[5. Nối link kiện IoT 16](#_Toc143253935)

[Code mẫu 17](#_Toc143253936)

[I. Iot (Arduino code) 17](#_Toc143253937)

[II. Back-end 23](#_Toc143253938)

[1. Restful API 23](#_Toc143253939)

[2. Mysql Query 26](#_Toc143253940)

[III. Front-end 28](#_Toc143253941)

[Kết luận 29](#_Toc143253942)

[Nguồn tham khảo 30](#_Toc143253943)

# Lời Mở Đầu

Em xin cảm ơn giảng viên bộ môn thực hành công nghệ thông tin (cô Trần thị Trường Thi) đã tạo cơ hội cho em tiếp xúc với những công nghệ mới và những trải nghiệm mới.

Bài báo cáo này sẽ có sự sai sót do đó mong cô thông cảm và góp ý để giúp em rút kinh nghiệm và cải thiện cho những lần báo cáo sau.

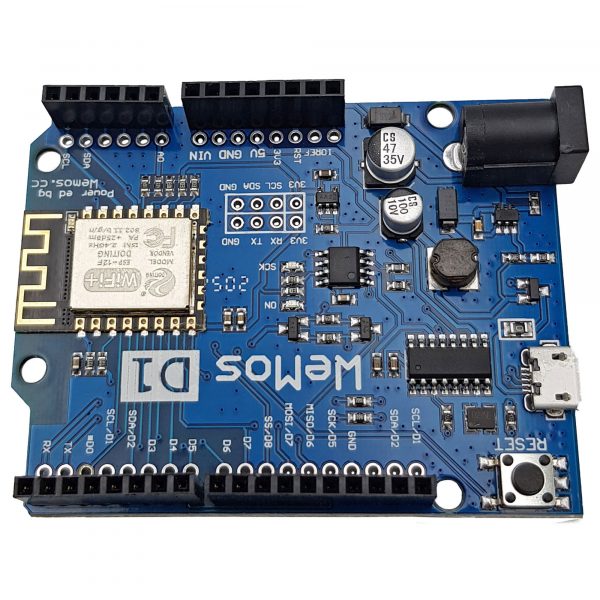
# Nội dung

## Chuẩn bị (Khái quát)

### Yêu cầu thiết bị và môi trường

#### Thiết bị

* Wemos D1 R2 (ESP 2866)



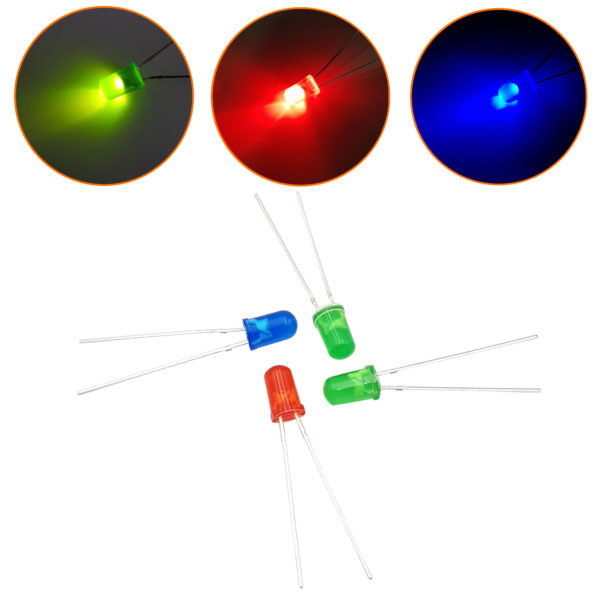
* Cảm biến nồng độ cồn MQ3



* Dây nối (đực cái, cái đực, cái cái)



* Đèn led 5mm



* Board test



* Còi 5v



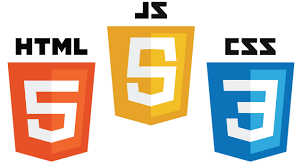
* Dây usb to micro-usb



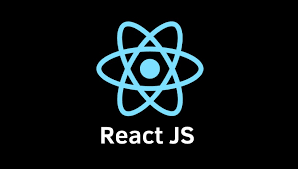
#### Môi trường

##### Front-end

* Html/css, js



* React js

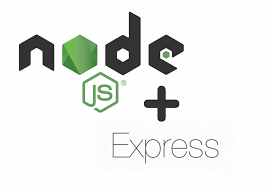


* Bootstrap 5



##### Back-end

* Nodejs
* Express js và những npm ‘s packages (như nodemon, cors, body-parser…)



* Postman



##### Database

* Mysql (và mysql2 cho npm’s package)



##### IoT

* Arduino IDE



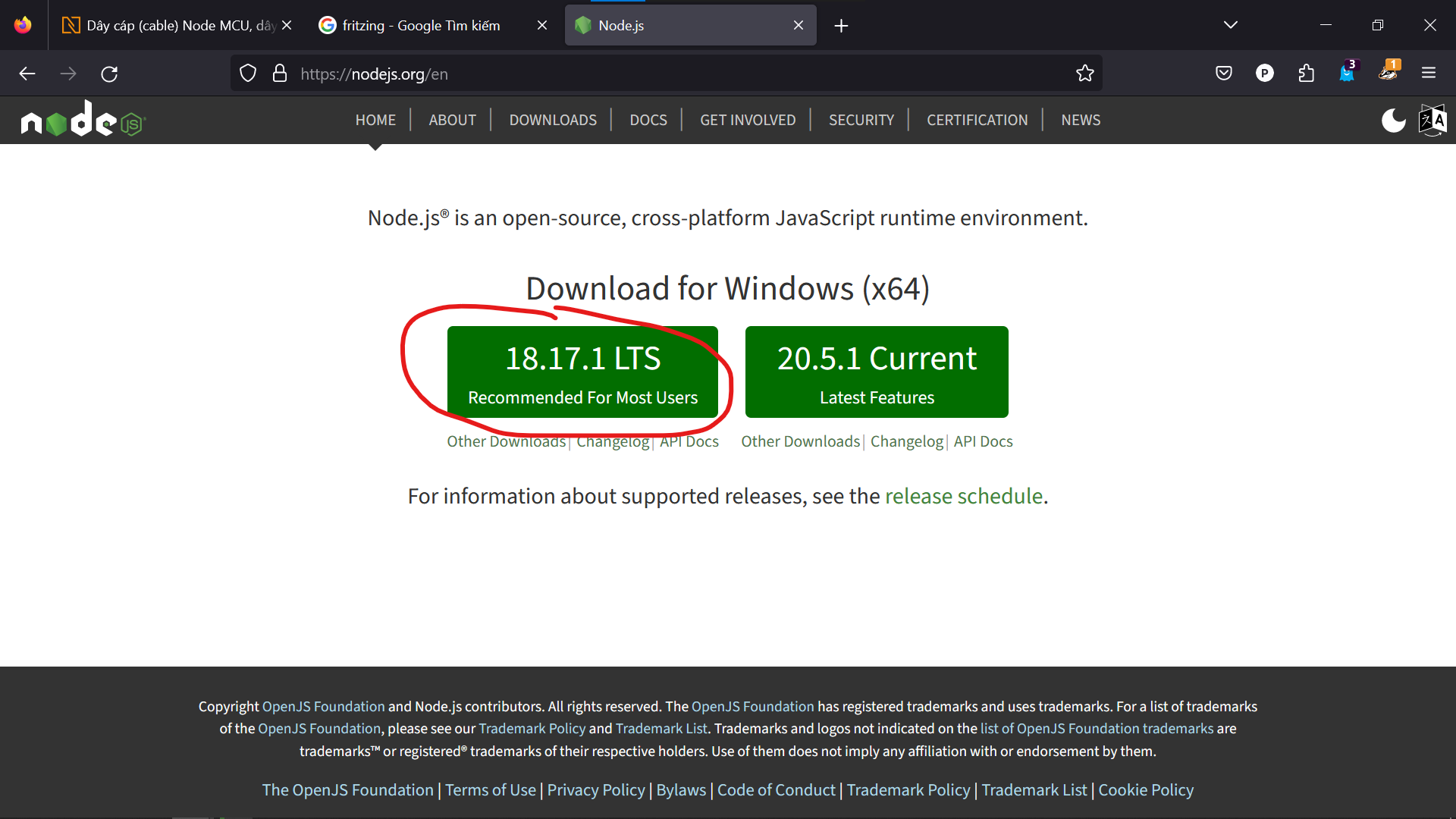
* Fritzing



## Cài đặt môi trường (1 vài ứng dụng)

### Nodejs

Link tải: <https://nodejs.org/en> (tải bản LTS)



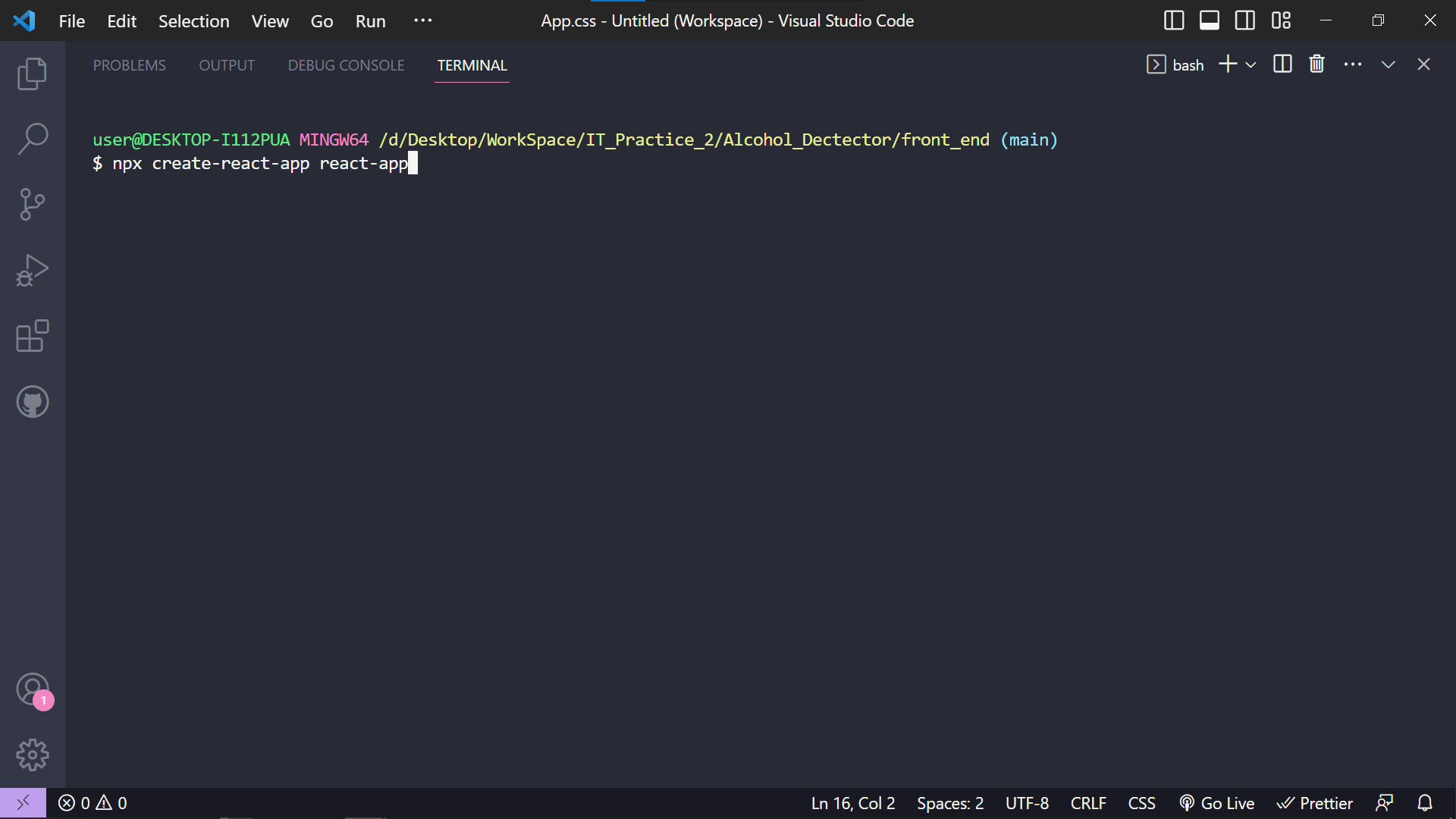
Bắt đầu cài đặt => Sau khi cài đặt, vào PowerShell để kiểm tra bằng lệnh:

**node -v**



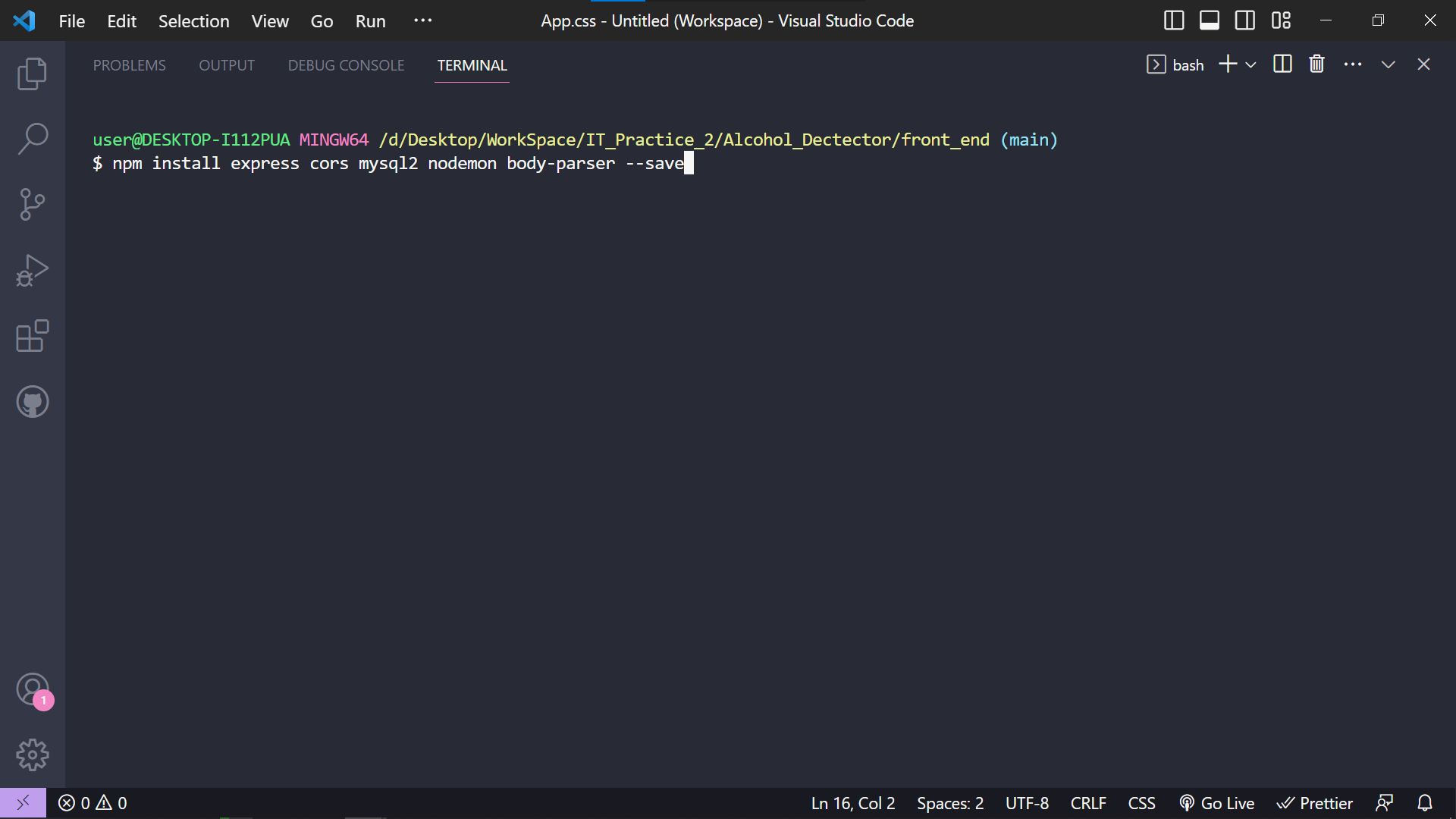
Để tạo 1 reactjs project dung lệnh như sau (xài git hoặc powershell cái nào cũng được)

**npx create-react-app <filename>**



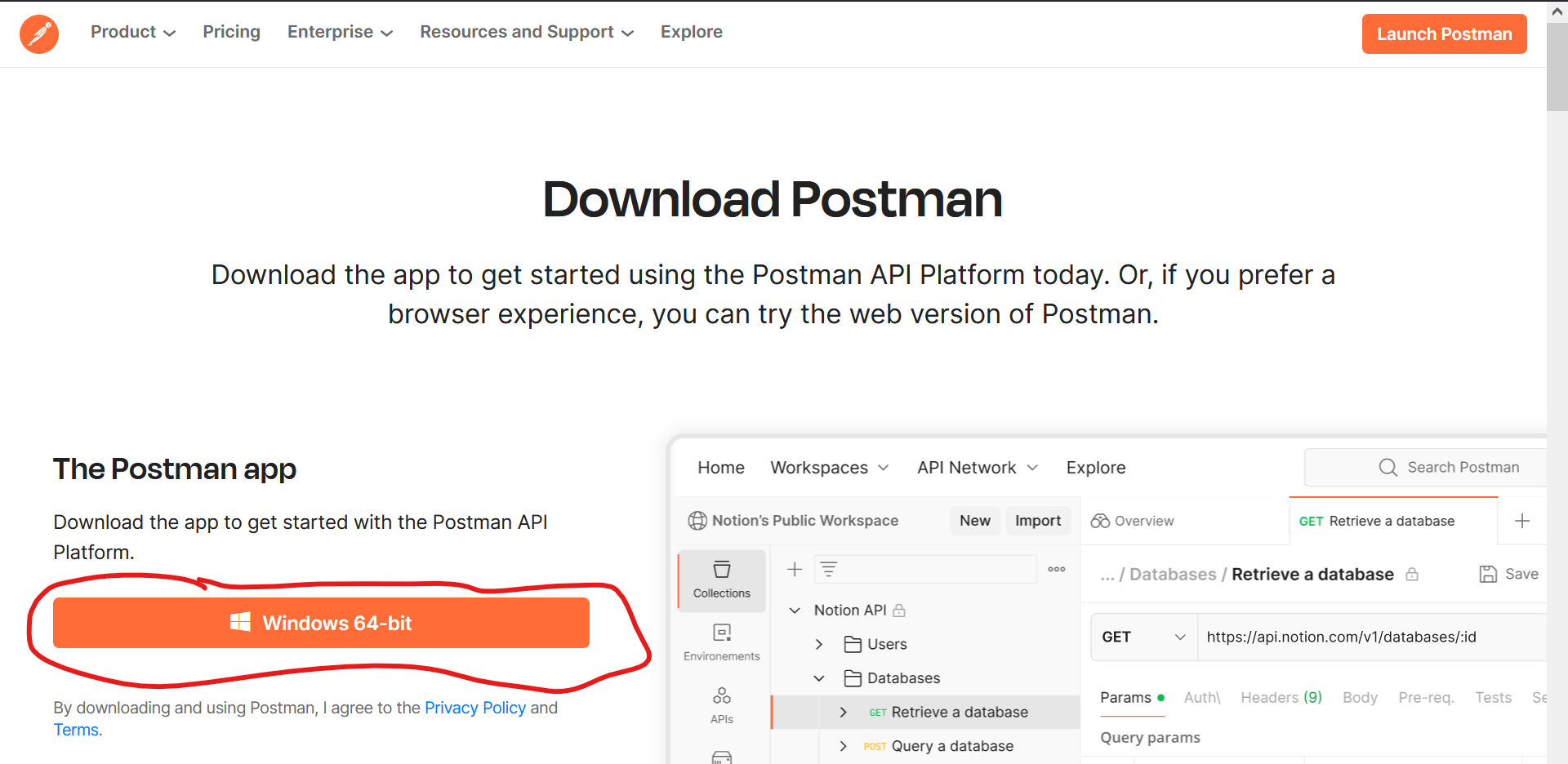
Để tải express cũng như những packages hỗ trợ thì dùng câu lệnh sau (dùng ở thư mục để phục vụ cho back-end):

**npm install express cors mysql2 nodemon body-parser –save**



### Postman

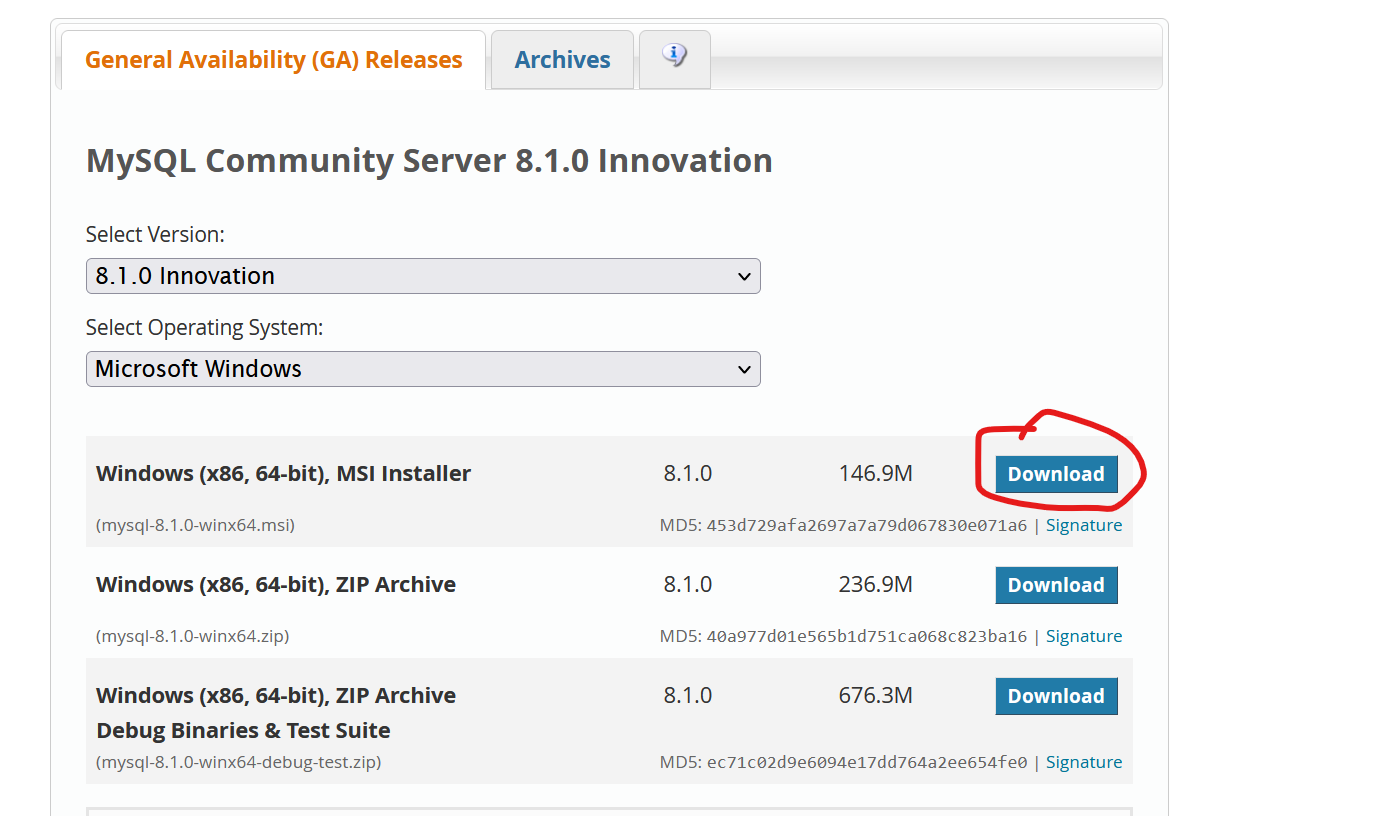
Link tải: <https://www.postman.com/downloads/>



### Mysql

Đầu tiên, tải sql server (bản community).

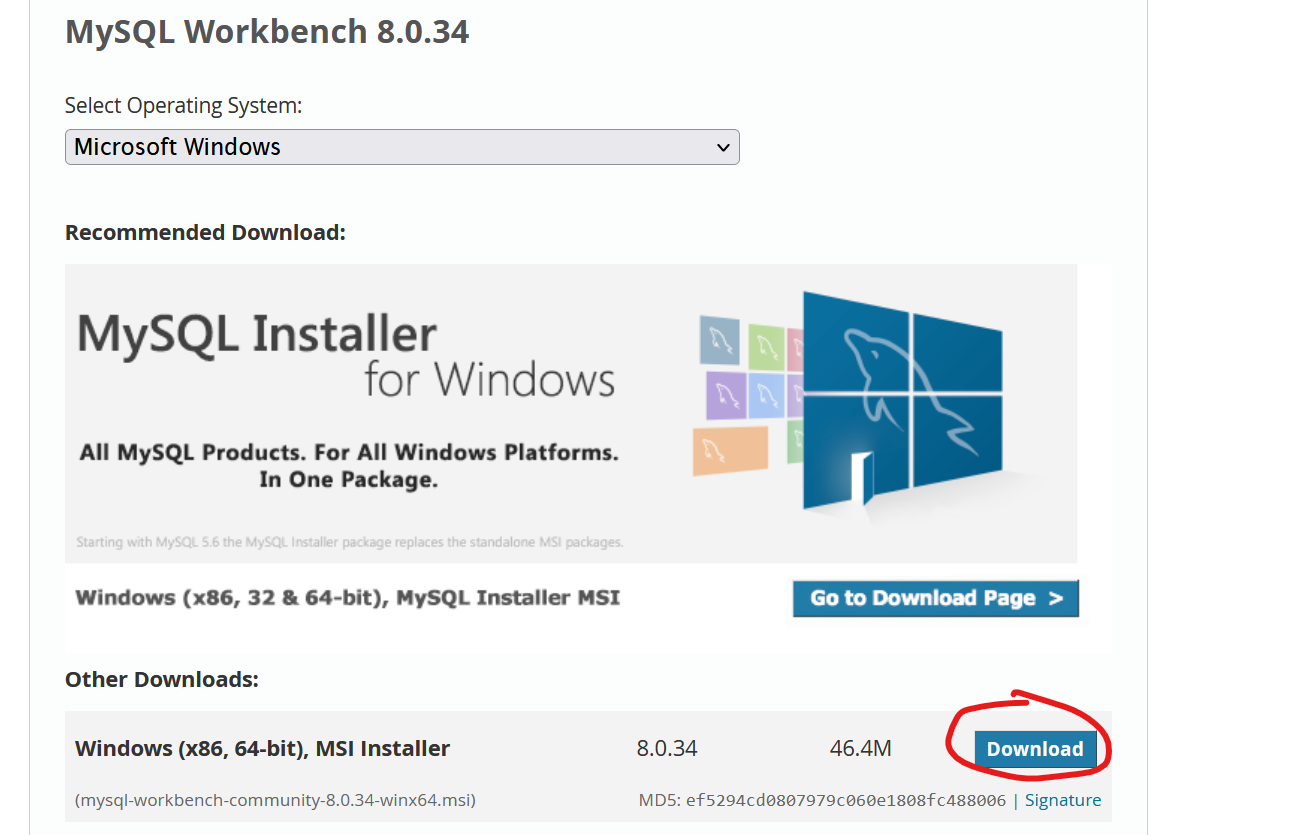
Link tải: <https://dev.mysql.com/downloads/mysql/>



\*\*Chú ý: Khi tải xong, phải tải thêm visual studio rồi tiến hành cài đặt

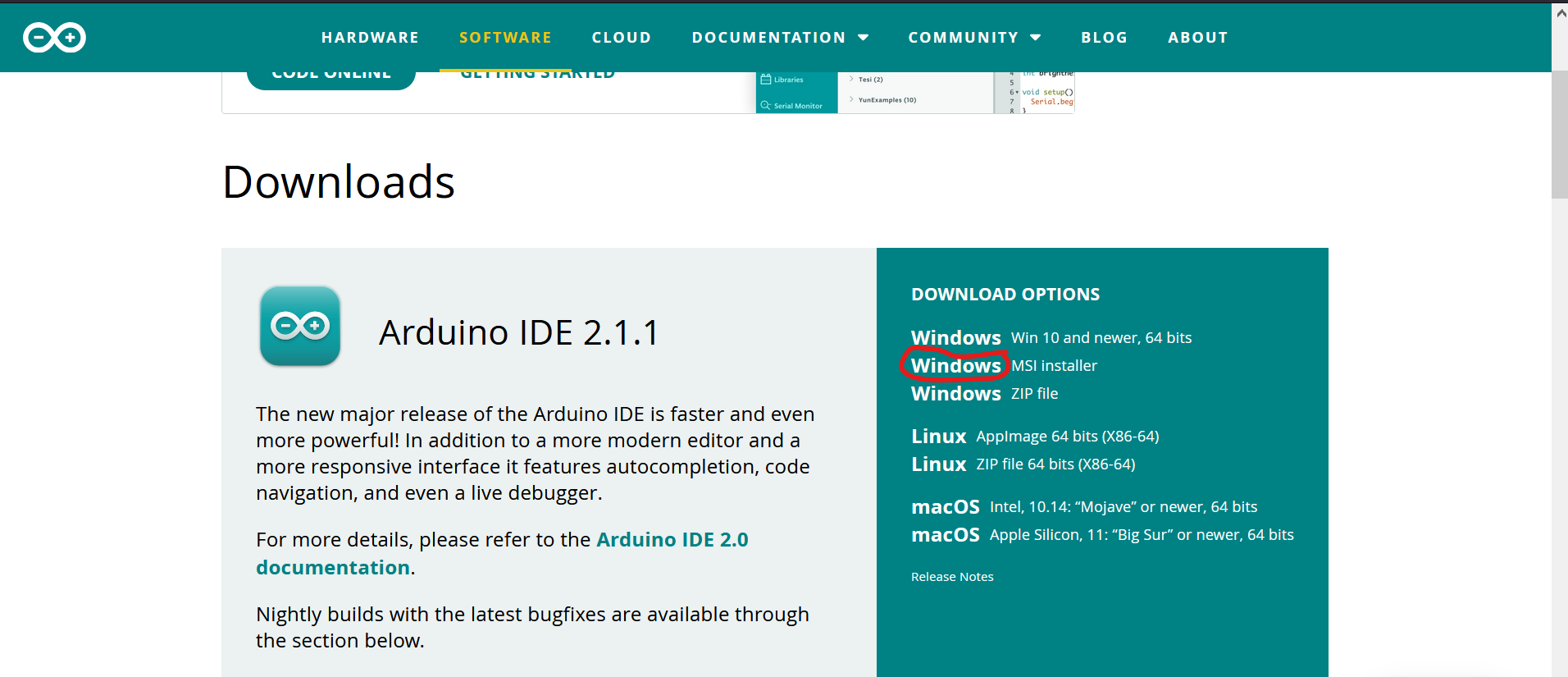
Sau đó, tải mysql workbench

Link tải: <https://dev.mysql.com/downloads/workbench/>



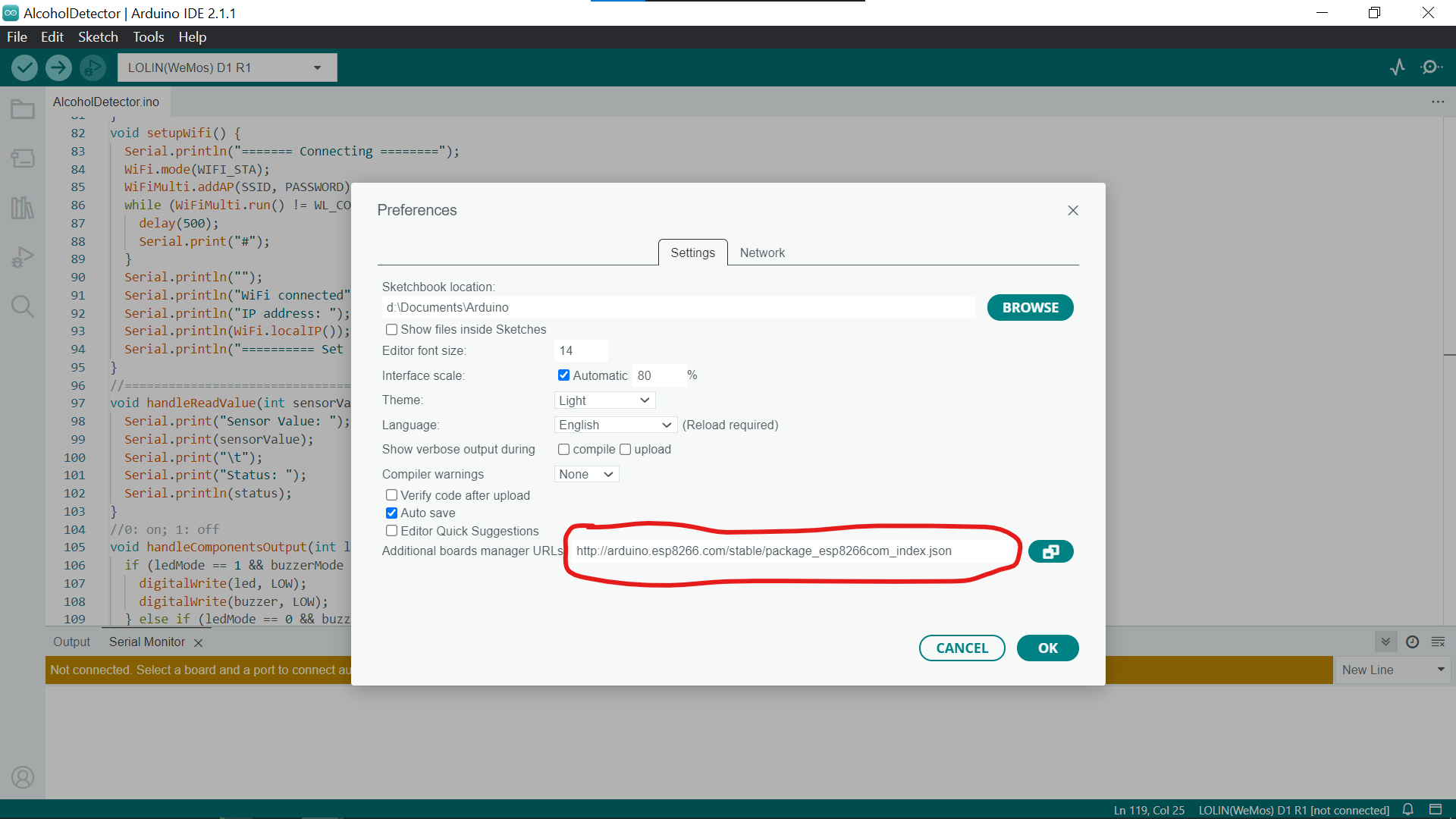
### Arduino ide

Link tải: <https://www.arduino.cc/en/software>

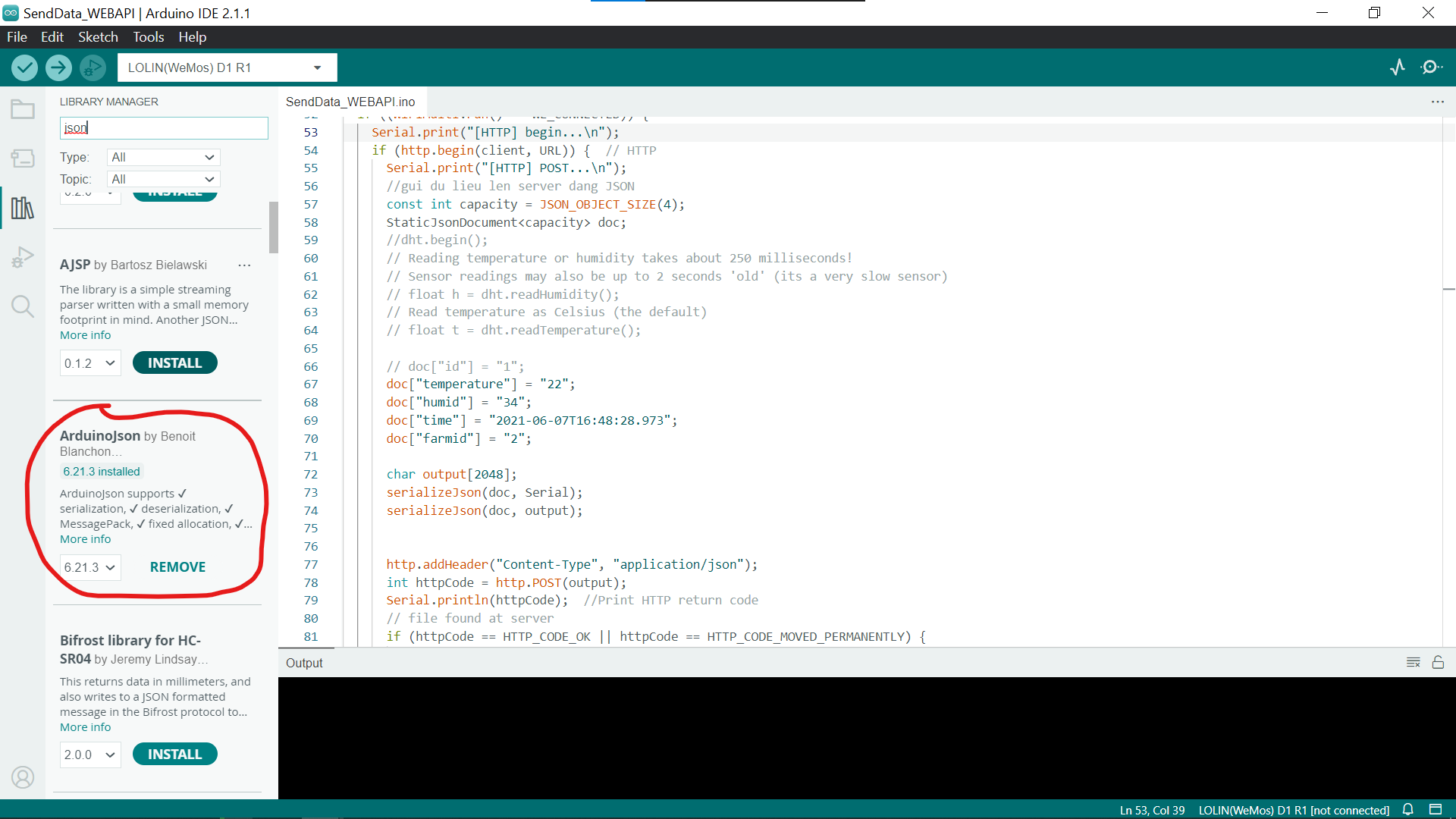


Cài đặt thư viện đối với wemos trong mục preferences (pass đường link vào như hình hướng dẫn):

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



Tải thư viện Arduino JSON (như hình dẫn dưới đây)



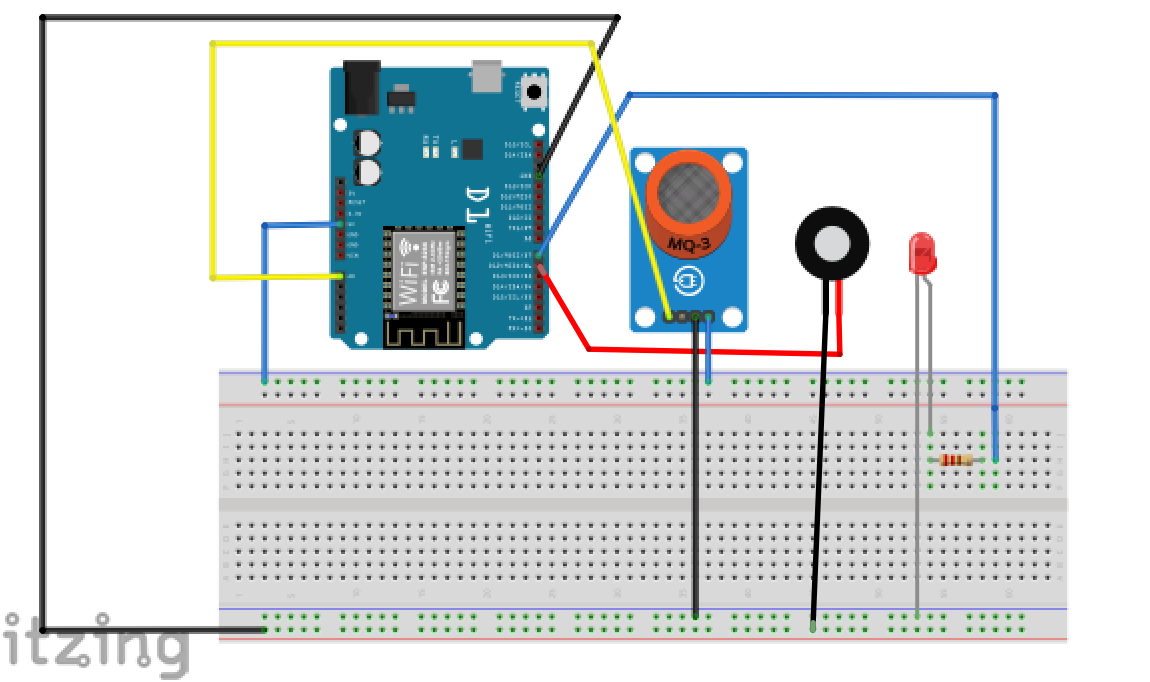
Tải Fritzing

Link tải: <https://arduinofactory.fr/en/download-fritzing-for-free/#Download_Fritzing_093b_for_FREE>

Rồi sau đó giải nén ra rồi xài (xài bản 64 bits).

### Nối link kiện IoT

Như hình vẽ:



# Code mẫu

## Iot (Arduino code)

1. #include <ESP8266WiFi.h>
2. #include <ESP8266WebServer.h>
3. #include <ESP8266mDNS.h>
4. #include <ESP8266WiFi.h>
5. #include <ESP8266WiFiMulti.h>
6. #include <ESP8266HTTPClient.h>
7. #include <WiFiClient.h>
8. #include <Arduino.h>
9. #include <ArduinoJson.h>
10. ESP8266WiFiMulti WiFiMulti;
11. WiFiClient client;
12. HTTPClient http;
13. #define ssid "thienloc"
14. #define password "Pie@2468"
15. #define url "http://192.168.2.8:8080/person"
16. const char \*SSID = ssid;
17. const char \*PASSWORD = password;
18. const char \*URL = url;
19. //settings
20. #define led D7
21. #define buzzer D6
22. #define sensorAnalog A0
23. #define air 650     //Input air value here
24. #define normal 700  //Input normal value here
25. #define drunk 800   //Input drunk value here
26. //json data
27. float p\_StatusIndex;
28. String p\_LName = "Input Here";
29. String p\_MName = "Input Here";
30. String p\_FName = "Input Here";
31. //============ Main Code ================
32. void setup() {
33. // put your setup code here, to run once:
34. Serial.begin(115200);
35. setupComponents();
36. setupWifi();
37. }
38. void loop() {
39. // put your main code here, to run repeatedly:
40. //testSensor();
41. int sensorValue = analogRead(sensorAnalog);
42. if (sensorValue > air) {
43. if (sensorValue <= normal) {
44. handleReadValue(sensorValue, "Normal!");
45. handleComponentsOutput(1, 1);
46. p\_StatusIndex = sensorValue / 100;
47. postJSONDataNormal();
48. } else if (sensorValue > normal && sensorValue <= drunk) {
49. handleReadValue(sensorValue, "Drinking but within legal limits!!");
50. handleComponentsOutput(0, 1);
51. p\_StatusIndex = sensorValue / 100;
52. postJSONDataDrinking();
53. } else {
54. handleReadValue(sensorValue, "DRUNK!!!");
55. handleComponentsOutput(0, 0);
56. p\_StatusIndex = sensorValue / 100;
57. postJSONDataDrunk();
58. }
59. delay(10000);
60. refreshUntilReturnAir();
61. } else {
62. handleReadValue(sensorValue, "Air");
63. delay(500);
64. }
65. }
66. //====================================
67. void setupComponents() {
68. pinMode(led, OUTPUT);
69. pinMode(buzzer, OUTPUT);
70. for (uint8\_t t = 5; t > 0; t--) {
71. Serial.printf("[SETUP SENSOR] WAIT %d...\n", t);
72. Serial.flush();
73. delay(5000);
74. }
75. Serial.println("SETUP COMPLETE!!!");
76. }
77. void setupWifi() {
78. Serial.println("======= Connecting ========");
79. WiFi.mode(WIFI\_STA);
80. WiFiMulti.addAP(SSID, PASSWORD);
81. while (WiFiMulti.run() != WL\_CONNECTED) {
82. delay(500);
83. Serial.print("#");
84. }
85. Serial.println("");
86. Serial.println("WiFi connected");
87. Serial.println("IP address: ");
88. Serial.println(WiFi.localIP());
89. Serial.println("========== Set Up Complete ============== ");
90. }
91. //=====================================
92. void handleReadValue(int sensorValue, String status) {
93. Serial.print("Sensor Value: ");
94. Serial.print(sensorValue);
95. Serial.print("\t");
96. Serial.print("Status: ");
97. Serial.println(status);
98. }
99. //0: on; 1: off
100. void handleComponentsOutput(int ledMode, int buzzerMode) {
101. if (ledMode == 1 && buzzerMode == 1) {
102. digitalWrite(led, LOW);
103. digitalWrite(buzzer, LOW);
104. } else if (ledMode == 0 && buzzerMode == 1) {
105. digitalWrite(led, HIGH);
106. digitalWrite(buzzer, LOW);
107. } else if (ledMode == 0 && buzzerMode == 0) {
108. digitalWrite(led, HIGH);
109. digitalWrite(buzzer, HIGH);
110. }
111. }
112. void refreshUntilReturnAir() {
113. int sensorValue = analogRead(sensorAnalog);
114. int updateSensorValue;
115. int airCheck = air - 50;
116. int count = 0;
117. handleComponentsOutput(1, 1);
118. Serial.println("======================= REFRESH!!! ======================");
119. while (true) {
120. updateSensorValue = analogRead(sensorAnalog);
121. if (updateSensorValue > airCheck) {
122. Serial.printf("=== Loading...%d... ===\n", count++);
123. } else {
124. break;
125. }
126. delay(500);
127. }
128. Serial.println("================= REFRESH COMPLETE!!!!! =================");
129. delay(2500);
130. return;
131. }
132. //POST JSON Data
133. void postJSONDataNormal() {
134. Serial.print("connecting to ");
135. if ((WiFiMulti.run() == WL\_CONNECTED)) {
136. Serial.print("[HTTP] begin...\n");
137. if (http.begin(client, URL)) {  // HTTP
138. Serial.print("[HTTP] POST...\n");
139. //gui du lieu len server dang JSON
140. const int capacity = JSON\_OBJECT\_SIZE(6);
141. StaticJsonDocument<capacity> doc;
142. doc["p\_LName"] = p\_LName;
143. doc["p\_MName"] = p\_MName;
144. doc["p\_FName"] = p\_FName;
145. doc["p\_StatusIndex"] = p\_StatusIndex;
146. doc["p\_Status"] = "Normal!";
147. char output[2048];
148. serializeJson(doc, Serial);
149. serializeJson(doc, output);
150. http.addHeader("Content-Type", "application/json");
151. int httpCode = http.POST(output);
152. Serial.println(httpCode);  //Print HTTP return code
153. // file found at server
154. if (httpCode == HTTP\_CODE\_OK || httpCode == HTTP\_CODE\_MOVED\_PERMANENTLY) {
155. String payload = http.getString();
156. Serial.println(payload);
157. Serial.println("done");
158. } else {
159. Serial.printf("[HTTP] POST... failed, error: %s\n", http.errorToString(httpCode).c\_str());
160. }
161. http.end();  //Close connection Serial.println();
162. Serial.println("closing connection");
163. }
164. }
165. }
166. void postJSONDataDrinking() {
167. Serial.print("connecting to ");
168. if ((WiFiMulti.run() == WL\_CONNECTED)) {
169. Serial.print("[HTTP] begin...\n");
170. if (http.begin(client, URL)) {  // HTTP
171. Serial.print("[HTTP] POST...\n");
172. //gui du lieu len server dang JSON
173. const int capacity = JSON\_OBJECT\_SIZE(6);
174. StaticJsonDocument<capacity> doc;
175. doc["p\_LName"] = p\_LName;
176. doc["p\_MName"] = p\_MName;
177. doc["p\_FName"] = p\_FName;
178. doc["p\_StatusIndex"] = p\_StatusIndex;
179. doc["p\_Status"] = "Drinking but within legal limits!!";
180. char output[2048];
181. serializeJson(doc, Serial);
182. serializeJson(doc, output);
183. http.addHeader("Content-Type", "application/json");
184. int httpCode = http.POST(output);
185. Serial.println(httpCode);  //Print HTTP return code
186. // file found at server
187. if (httpCode == HTTP\_CODE\_OK || httpCode == HTTP\_CODE\_MOVED\_PERMANENTLY) {
188. String payload = http.getString();
189. Serial.println(payload);
190. Serial.println("done");
191. } else {
192. Serial.printf("[HTTP] POST... failed, error: %s\n", http.errorToString(httpCode).c\_str());
193. }
194. http.end();  //Close connection Serial.println();
195. Serial.println("closing connection");
196. }
197. }
198. }
199. void postJSONDataDrunk() {
200. Serial.print("connecting to ");
201. if ((WiFiMulti.run() == WL\_CONNECTED)) {
202. Serial.print("[HTTP] begin...\n");
203. if (http.begin(client, URL)) {  // HTTP
204. Serial.print("[HTTP] POST...\n");
205. //gui du lieu len server dang JSON
206. const int capacity = JSON\_OBJECT\_SIZE(6);
207. StaticJsonDocument<capacity> doc;
208. doc["p\_LName"] = p\_LName;
209. doc["p\_MName"] = p\_MName;
210. doc["p\_FName"] = p\_FName;
211. doc["p\_StatusIndex"] = p\_StatusIndex;
212. doc["p\_Status"] = "DRUNK!!!";
213. char output[2048];
214. serializeJson(doc, Serial);
215. serializeJson(doc, output);
216. http.addHeader("Content-Type", "application/json");
217. int httpCode = http.POST(output);
218. Serial.println(httpCode);  //Print HTTP return code
219. // file found at server
220. if (httpCode == HTTP\_CODE\_OK || httpCode == HTTP\_CODE\_MOVED\_PERMANENTLY) {
221. String payload = http.getString();
222. Serial.println(payload);
223. Serial.println("done");
224. } else {
225. Serial.printf("[HTTP] POST... failed, error: %s\n", http.errorToString(httpCode).c\_str());
226. }
227. http.end();  //Close connection Serial.println();
228. Serial.println("closing connection");
229. }
230. }
231. }
232. void testSensor() {
233. while(true) {
234. int sensorValue = analogRead(sensorAnalog);
235. Serial.print("Sensor value: "); Serial.println(sensorValue);
236. delay(500);
237. }
238. }

## Back-end

### Restful API

1. var express = require('express');
2. var bodyParser = require("body-parser");
3. var cors = require('cors');
4. var morgan = require('morgan');
5. var mysql = require('mysql2');
6. var app = express();
7. app.use(cors());
8. app.use(express.static('public'));
9. app.use(bodyParser.json());
10. app.use(bodyParser.urlencoded({ extended: true }));
11. //mysql
12. var con = mysql.createConnection({
13. host: "localhost",
14. port: "3306",
15. user: "root",
16. password: "1234",
17. insecureAuth: true,
18. database: "alcoholdb"
19. });
20. con.connect(function (*err*) {
21. if (*err*) throw *err*;
22. console.log("Connected!!!")
23. });
24. //Test API
25. app.get("/", function (*req*, *res*) {
26. *res*.send("Hello World");
27. });
28. //Restful API
29. app.get("/person", function (*req*, *res*) {
30. // const id = req.params.id;
31. var sql = "select \* from person;";
32. con.query(sql, function (*err*, *results*) {
33. if (*err*) throw *err*;
34. //console.log(results);
35. *res*.send(*results*);
36. });
37. });
38. app.get("/person/:id", function (*req*, *res*) {
39. const id = *req*.params.id;
40. var sql = "select \* from person where id = "+id+";";
41. con.query(sql, function (*err*, *results*) {
42. if (*err*) throw *err*;
43. //console.log(results);
44. *res*.send(*results*);
45. });
46. });
47. app.post("/person", function(*req*, *res*) {
48. const {p\_LName, p\_MName, p\_FName, p\_StatusIndex, p\_Status} = *req*.body;
49. var sql = "insert into person (p\_LName, p\_MName, p\_FName, p\_StatusIndex, p\_Status) values ('"+p\_LName+"','"+p\_MName+"','"+p\_FName+"',"+p\_StatusIndex+",'"+p\_Status+"');";
50. con.query(sql, function (*err*, *results*) {
51. if (*err*) throw *err*;
52. //console.log(results);
53. // res.send("Add complete!!");
54. // res.send(results);
55. *res*.send("Add Complete ("+p\_LName+", "+p\_MName+", "+p\_FName+", "+p\_StatusIndex+", "+p\_Status+")");
56. });
57. });
58. app.put("/person/:id", function(*req*, *res*) {
59. const {id} = *req*.params.id;
60. const {p\_ID, p\_LName, p\_MName, p\_FName} = *req*.body;
61. var sql = "update person set p\_ID = '"+p\_ID+"', p\_LName = '"+p\_LName+"', p\_MName = '"+p\_MName+"', p\_FName = '"+p\_FName+"' where ID = "+id+";";
62. con.query(sql, function (*err*, *results*) {
63. if (*err*) throw *err*;
64. //console.log(results);
65. *res*.send("Update complete!!");
66. //res.send(results);
67. });
68. });
69. app.delete("/person/:id", function(req, res) {
70. const id = req.params.id;
71. var sql = "delete from person where ID = "+id+";";
72. con.query(sql, function (err, results) {
73. if (err) throw err;
74. //console.log(results);
75. res.send("Delete complete!!");
76. //res.send(results);
77. });
78. });
79. //server
80. const port = 8080;
81. var server = app.listen(port, function () {
82. var host = server.address().address;
83. *console*.log(`Server is listening at http://${host}:${port}`)
84. });

### Mysql Query

create database alcoholdb;

use alcoholdb;

create table person(

ID int not null auto\_increment,

p\_ID int null,

p\_LName varchar(255),

p\_MName varchar(255),

p\_FName varchar(255),

p\_StatusIndex float,

p\_Status varchar(255),

-- s\_id int,

constraint pk\_person primary key (ID),

constraint uc\_person unique (p\_ID)

);

-- create table sensor(

-- id int not null auto\_increment,

-- s\_id int null,

-- s\_normal float,

-- s\_drunklegal float,

-- s\_drunkillegal float,

-- constraint pk\_sensor primary key (id, s\_id)

-- );

select \* from person;

insert into person (p\_ID, p\_LName, p\_MName, p\_FName, p\_StatusIndex, p\_Status)

values ('p\_ID','p\_LName','p\_MName','p\_FName',p\_StatusIndex,'p\_Status');

update person set p\_ID = 'p\_ID', p\_LName = 'p\_LName', p\_MName = 'p\_MName', p\_FName = 'p\_FName' where id = id;

delete from person where id = id;

delete from person where ID = 3;

## Front-end

Sẽ cập nhật sau

# Kết luận

Thông qua dự án trên em đã học được rất nhiều kiến thức mới như lần đầu tiếp xúc với reacjs, node & express cũng như IoT…

Em cảm ơn sự đồng hành của cô trong những buổi học qua và cũng như mong muốn gặp lại cô trong quá trình học để học hỏi them

Em xin cảm ơn.

# Nguồn tham khảo

Những bài hướng dẫn lab trong lớp.

Arduino document: <https://www.arduino.cc/reference/en/>

Stack Overflow: <https://stackoverflow.com/>

Bootstrap documentation: <https://getbootstrap.com/docs/5.3/getting-started/introduction/>

MDB bootstrap: <https://mdbootstrap.com/>

Npm documentation: <https://docs.npmjs.com/>