|  |  |
| --- | --- |
| **โค้ด** | **ความหมาย** |
| #include <TinyGPS++.h> | ไลบรารีสำหรับจัดการข้อมูล GPS |
| #include <HardwareSerial.h> | ไลบรารีสำหรับใช้งานพอร์ตอนุกรมของ ESP32 |
| #include <WiFi.h> | ไลบรารีสำหรับเชื่อมต่อ WiFi |
| #include <BlynkSimpleEsp32.h> | ไลบรารีสำหรับเชื่อมต่อ Blynk (แพลตฟอร์ม IoT) |
| #include <HTTPClient.h> | ไลบรารีสำหรับใช้ HTTP requests |
|  |  |
| static const int RXPin = 16, TXPin = 17; | กำหนดขา RX และ TX สำหรับเชื่อมต่อ GPS กับ ESP32 |
| static const uint32\_t GPSBaud = 9600; | กำหนด Baud Rate ของ GPS (9600 เป็นค่ามาตรฐาน) |
|  |  |
| TinyGPSPlus gps; | สร้างอ็อบเจ็กต์ gps เพื่อเก็บข้อมูลที่ได้จากโมดูล GPS |
| HardwareSerial gpsSerial(1); | สร้างอ็อบเจ็กต์ gpsSerial เพื่อใช้พอร์ตอนุกรมเสริม (Serial1) สำหรับ GPS |
| BlynkTimer timer; | ใช้ Blynk Timer สำหรับจัดการโค้ดให้ทำงานเป็นระยะ |
| WidgetMap myMap(V0); | สร้าง Widget Map บน Blynk (V0 คือ Virtual Pin ที่ใช้) |
|  |  |
| char auth[] = "grvTmlmgkmEhAZcuBjgDQA4m7eVR\_6B4"; | โทเค็นสำหรับเชื่อมต่อ Blynk |
| char ssid[] = "Piriya\_B3"; | ชื่อ WiFi ที่จะเชื่อมต่อ |
| char pass[] = "piriya19052023"; | รหัสผ่านของ WiFi |
|  |  |
| const String serverUrl = "http://192.168.0.104/gps/insert\_gps\_data.php"; | URL ของเซิร์ฟเวอร์ในเครือข่ายท้องถิ่นสำหรับบันทึกค่าพิกัด GPS |
| const String discordWebhookUrl = "https://discordapp.com/api/webhooks/1344771500804407389/kaTFHB-e1UWa7ZGp3CpkOA1-qgjtzPE6ZY1ACXorB6Zt2y5XD63W9a9Z1QBfyge8Ghev";  // <== ใส่ Webhook URL | URL Webhook ของ Discord ที่ใช้ส่งข้อมูลพิกัด GPS ไปยัง Discord Channel |
|  |  |
| void setup() { | เป็นฟังก์ชันที่ ESP32 จะรันเพียงครั้งเดียวตอนเริ่มต้น |
| Serial.begin(115200); | กำหนดให้ Serial Monitor ใช้ Baud Rate **115200** เพื่อใช้ดูข้อมูลที่ส่งออกทาง Serial Monitor |
| gpsSerial.begin(GPSBaud, SERIAL\_8N1, RXPin, TXPin); | เริ่มต้นการสื่อสารกับโมดูล GPS ผ่าน Serial1 |
|  | GPSBaud = 9600 (ค่ามาตรฐานของโมดูล GPS) |
|  | SERIAL\_8N1 = กำหนดโหมดการสื่อสาร (8 บิตข้อมูล, ไม่มี parity, 1 stop bit) |
|  | RXPin, TXPin = กำหนดขา **RX (16)** และ **TX (17)** สำหรับรับส่งข้อมูลกับ GPS |
| WiFi.begin(ssid, pass); | เชื่อมต่อ WiFi โดยใช้ **SSID** และ **รหัสผ่าน** ที่กำหนดไว้ |
| while (WiFi.status() != WL\_CONNECTED) { | เช็กสถานะการเชื่อมต่อ WiFi โดยใช้ WiFi.status() |
| delay(500); | ถ้ายังไม่เชื่อมต่อ (WL\_CONNECTED ไม่ถูกตรวจพบ) จะรอและพิมพ์ . ทุก 500 มิลลิวินาที |
| Serial.print("."); |  |
| } |  |
| Serial.println("\nWiFi Connected"); | เมื่อเชื่อมต่อสำเร็จ จะพิมพ์ "WiFi Connected" ลงใน Serial Monitor |
| Blynk.begin(auth, ssid, pass, "blynk.iot-cm.com", 8080); | เริ่มต้นการเชื่อมต่อ Blynk โดยใช้  auth = โทเค็นของ Blynk  ssid = ชื่อ WiFi  pass = รหัสผ่าน WiFi  "blynk.iot-cm.com" = เซิร์ฟเวอร์ของ Blynk  8080 = พอร์ตที่ใช้สื่อสาร |
| timer.setInterval(5000L, sendData); // ส่งข้อมูลทุก 5 วินาที | ตั้งค่า **Blynk Timer** ให้เรียกใช้ฟังก์ชัน sendData() **ทุกๆ 5 วินาที (5000 มิลลิวินาที)** |
| } |  |
|  |  |
| void sendData() { | ฟังก์ชัน sendData() ถูกเรียกตามระยะเวลาที่ตั้งค่าใน timer.setInterval() (ทุก 5 วินาทีในที่นี้) |
| if (gps.charsProcessed() < 10) { | ฟังก์ชัน gps.charsProcessed() ใช้ตรวจสอบจำนวนตัวอักษรที่ถูกประมวลผลจากโมดูล GPS  ถ้าจำนวนตัวอักษรที่ถูกประมวลผลน้อยกว่า 10 ตัว แสดงว่า GPS ยังไม่ได้รับข้อมูลหรือสัญญาณยังไม่ชัดเจน |
| Serial.println("GPS ERROR: No signal."); | ถ้าจำนวนตัวอักษรที่ถูกประมวลผลน้อยกว่า 10 ตัว (ไม่มีสัญญาณ GPS หรือสัญญาณอ่อน) จะแสดงข้อความ **"GPS ERROR: No signal."** บน Serial Monitor |
| Blynk.virtualWrite(V4, "GPS ERROR"); | ส่งข้อความ "GPS ERROR" ไปยัง Virtual Pin V4 บน Blynk App เพื่อแสดงข้อความแจ้งเตือนว่าไม่พบสัญญาณ GPS |
| return; | ถ้าหากไม่พบสัญญาณ GPS (ข้อมูลน้อยกว่า 10 ตัวอักษร) ให้หยุดการทำงานของฟังก์ชันนี้ (ไม่ทำงานต่อไป) |
| } |  |
|  |  |
| if (gps.location.isUpdated()) { | ฟังก์ชันนี้ตรวจสอบว่าโมดูล GPS ได้รับข้อมูลตำแหน่งใหม่หรือไม่ ถ้ามีการอัพเดทตำแหน่งใหม่ |
| float latitude = gps.location.lat(); | ใช้เพื่อดึงค่าละติจูด (latitude) จากข้อมูล GPS ซึ่งจะถูกเก็บไว้ในตัวแปร latitude เป็นประเภท float (มีจุดทศนิยม) |
| float longitude = gps.location.lng(); | ใช้เพื่อดึงค่าลองจิจูด (longitude) จากข้อมูล GPS ซึ่งจะถูกเก็บไว้ในตัวแปร longitude เป็นประเภท float |
| float speed = gps.speed.kmph(); | ใช้เพื่อดึงความเร็ว (speed) ที่คำนวณจาก GPS ซึ่งจะถูกแสดงเป็นกิโลเมตรต่อชั่วโมง (km/h) และเก็บในตัวแปร speed เป็นประเภท float |
| int satellites = gps.satellites.value(); | ใช้เพื่อดึงจำนวนดาวเทียมที่โมดูล GPS ใช้งานอยู่ในขณะนั้น (ซึ่งมีผลต่อความแม่นยำของข้อมูล) และเก็บไว้ในตัวแปร satellites เป็นประเภท int |
| float bearing = gps.course.deg(); | ใช้เพื่อดึงค่าทิศทาง (bearing หรือ course) ที่แสดงเป็นมุมที่วัดจากทิศเหนือ (degree) โดยค่า bearing จะแสดงเป็น float |
|  |  |
| Serial.print("LAT: "); Serial.println(latitude, 6); | พิมพ์ข้อความ "LAT: " ลงใน Serial Monitor  พิมพ์ค่าละติจูด (latitude) โดยมีการกำหนดให้แสดง 6 ตำแหน่งหลังจุดทศนิยม (กำหนดโดยพารามิเตอร์ 6) |
| Serial.print("LON: "); Serial.println(longitude, 6); | พิมพ์ข้อความ "LON: " ลงใน Serial Monitor  พิมพ์ค่าลองจิจูด (longitude) โดยมีการกำหนดให้แสดง 6 ตำแหน่งหลังจุดทศนิยม |
| Serial.print("Speed: "); Serial.println(speed); | พิมพ์ข้อความ "Speed: " ลงใน Serial Monitor  พิมพ์ค่าความเร็ว (speed) ลงใน Serial Monitor โดยไม่มีการกำหนดตำแหน่งทศนิยม |
| Serial.print("Satellites: "); Serial.println(satellites); | พิมพ์ข้อความ "Satellites: " ลงใน Serial Monitor  พิมพ์จำนวนดาวเทียม (satellites) ที่โมดูล GPS ใช้งานอยู่ในปัจจุบัน |
| Serial.print("Heading: "); Serial.println(bearing); | พิมพ์ข้อความ "Heading: " ลงใน Serial Monitor  พิมพ์ทิศทาง (bearing หรือ heading) ลงใน Serial Monitor |
|  |  |
| Blynk.virtualWrite(V1, latitude); | ส่งค่า **ละติจูด (latitude)** ไปยัง **Virtual Pin V1** ในแอป Blynk ซึ่งจะทำให้แอป Blynk สามารถแสดงค่าละติจูดนี้ได้ |
| Blynk.virtualWrite(V2, longitude); | ส่งค่า **ลองจิจูด (longitude)** ไปยัง **Virtual Pin V2** ในแอป Blynk |
| Blynk.virtualWrite(V3, speed); | ส่งค่า **ความเร็ว (speed)** ไปยัง **Virtual Pin V3** ในแอป Blynk |
| Blynk.virtualWrite(V4, satellites); | ส่งค่า **จำนวนดาวเทียม (satellites)** ที่ใช้งานในขณะนั้นไปยัง **Virtual Pin V4** ในแอป Blynk |
| Blynk.virtualWrite(V5, bearing); | ส่งค่า **ทิศทาง (bearing)** ไปยัง **Virtual Pin V5** ในแอป Blynk |
| myMap.location(1, latitude, longitude, "GPS Tracker"); | ใช้ฟังก์ชัน location() ของ Widget Map บนแอป Blynk  1: คือ ID ของตำแหน่งที่จะถูกแสดงบนแผนที่  latitude: ค่าละติจูดที่ได้รับจาก GPS  longitude: ค่าลองจิจูดที่ได้รับจาก GPS  "GPS Tracker": ข้อความที่จะปรากฏบนตำแหน่งบนแผนที่ (เช่น ใช้แสดงว่าเป็นตำแหน่งของ GPS Tracker) |
|  |  |
| sendToServer(latitude, longitude, speed, satellites, bearing); | ฟังก์ชัน sendToServer() ใช้เพื่อส่งข้อมูลตำแหน่ง GPS (ละติจูด, ลองจิจูด), ความเร็ว, จำนวนดาวเทียมที่ใช้งาน, และทิศทางไปยังเซิร์ฟเวอร์เพื่อบันทึกหรือประมวลผลต่อ  ข้อมูลทั้งหมดจะถูกส่งไปในรูปแบบของพารามิเตอร์ที่ถูกระบุในฟังก์ชันนี้ |
|  |  |
| // แจ้งเตือนผ่าน Discord |  |
| sendDiscordAlert(latitude, longitude, speed, satellites, bearing); | ฟังก์ชัน sendDiscordAlert() ใช้เพื่อส่งข้อมูลตำแหน่ง GPS ไปยัง Discord ผ่าน Webhook |
| } |  |
| } |  |
| void sendToServer(float latitude, float longitude, float speed, int satellites, float bearing) { | ฟังก์ชัน sendToServer() ถูกออกแบบมาเพื่อส่งข้อมูลที่ได้รับจาก GPS (ละติจูด, ลองจิจูด, ความเร็ว, จำนวนดาวเทียม, และทิศทาง) ไปยังเซิร์ฟเวอร์ |
| HTTPClient http; | สร้างอ็อบเจกต์ของคลาส HTTPClient ชื่อว่า http ซึ่งจะใช้ในการส่งคำขอ HTTP ไปยังเซิร์ฟเวอร์ |
| String url = serverUrl + "?latitude=" + String(latitude, 6) + | สร้าง URL สำหรับส่งคำขอ HTTP ซึ่งประกอบไปด้วย: |
| "&longitude=" + String(longitude, 6) + | serverUrl (URL ของเซิร์ฟเวอร์) จะถูกเชื่อมต่อกับพารามิเตอร์ที่ต้องการส่งไป |
| "&speed=" + String(speed) + | ใช้ String(latitude, 6) เพื่อแปลงค่าละติจูดเป็นสตริงและกำหนดให้แสดงผล 6 ตำแหน่งทศนิยม |
| "&satellites=" + String(satellites) + | ใช้ String(longitude, 6) เพื่อแปลงค่าลองจิจูดเป็นสตริงและกำหนดให้แสดงผล 6 ตำแหน่งทศนิยม |
| "&bearing=" + String(bearing); | ใช้ String(speed), String(satellites), และ String(bearing) เพื่อแปลงค่าความเร็ว, จำนวนดาวเทียม และทิศทางเป็นสตริง |
|  |  |
| http.begin(url); | เริ่มต้นการเชื่อมต่อ HTTP โดยใช้ URL ที่ได้สร้างขึ้นในบรรทัดก่อนหน้านี้ |
| int httpCode = http.GET(); | ส่งคำขอ HTTP แบบ GET ไปยังเซิร์ฟเวอร์ และเก็บรหัสสถานะที่ได้รับจากเซิร์ฟเวอร์ในตัวแปร httpCode   * ถ้า httpCode เป็นค่าบวก (200 หรืออื่นๆ) แสดงว่าการเชื่อมต่อสำเร็จ * ถ้า httpCode เป็นค่าลบ (เช่น -1) แสดงว่าเกิดข้อผิดพลาดในการเชื่อมต่อ |
|  |  |
| if (httpCode > 0) { | ตรวจสอบค่าของ httpCode: |
| Serial.println("Data sent successfully to the server."); | ถ้าค่ามากกว่า 0 (การเชื่อมต่อสำเร็จ) จะแสดงข้อความ **"Data sent successfully to the server."** |
| } else { | ถ้าค่าน้อยกว่าหรือเท่ากับ 0 (เกิดข้อผิดพลาดในการเชื่อมต่อ) จะแสดงข้อความ **"Error in sending data."** |
| Serial.println("Error in sending data."); |  |
| } |  |
|  |  |
| http.end(); | ปิดการเชื่อมต่อ HTTP เพื่อให้แน่ใจว่าการใช้งานทรัพยากรทั้งหมดเสร็จสิ้น |
|  |  |
| void sendDiscordAlert(float latitude, float longitude, float speed, int satellites, float bearing) { | ฟังก์ชัน sendDiscordAlert() รับพารามิเตอร์ที่เป็นข้อมูลจาก GPS ได้แก่ ละติจูด (latitude), ลองจิจูด (longitude), ความเร็ว (speed), จำนวนดาวเทียม (satellites), และทิศทาง (bearing) ฟังก์ชันนี้จะส่งข้อมูลเหล่านี้ไปยัง Discord ผ่าน Webhook |
| HTTPClient http; | สร้างอ็อบเจกต์ของคลาส HTTPClient ชื่อว่า http เพื่อใช้ในการส่งคำขอ HTTP ไปยัง Discord Webhook URL |
| http.begin(discordWebhookUrl); | เริ่มต้นการเชื่อมต่อ HTTP โดยใช้ Webhook URL ที่กำหนดไว้ในตัวแปร discordWebhookUrl ซึ่งคือ URL ที่เชื่อมต่อกับ Discord API |
| http.addHeader("Content-Type","application/json"); | เพิ่มหัวข้อ **Content-Type** ที่มีค่าเป็น **application/json** เพื่อบอกให้ Discord API รู้ว่าเราจะส่งข้อมูลในรูปแบบ JSON |
|  |  |
| String jsonPayload = "{\"content\": \"🚀 \*\*GPS Alert!\*\* 🚀\\n" | สร้างข้อความในรูปแบบ **JSON** ที่จะส่งไปยัง Discord โดยประกอบไปด้วยข้อมูลที่ต้องการแจ้งเตือน |
| "\*\*Latitude:\*\* " + String(latitude, 6) + "\\n" | latitude**,** longitude**,** speed**,** satellites**,** bearing: ค่าที่ได้จาก GPS ถูกแปลงเป็นข้อความและใส่ในข้อความ JSON |
| "\*\*Longitude:\*\* " +String(longitude, 6) +"\\n" | ใช้ String() เพื่อแปลงค่าตัวแปรเหล่านี้ให้เป็นสตริง และแสดงผลด้วยทศนิยม 6 ตำแหน่งสำหรับละติจูดและลองจิจูด |
| "\*\*Speed:\*\* " + String(speed) + " km/h\\n" | ข้อความ JSON นี้จะถูกส่งไปในคำขอ POST ไปยัง Discord Webhook |
| "\*\*Satellites:\*\* " + String(satellites) + "\\n" |  |
| "\*\*Bearing:\*\* " + String(bearing) + "°" |  |
| "\"}"; |  |
|  |  |
| int httpCode = http.POST(jsonPayload); | ส่งคำขอ **POST** ไปยัง Discord Webhook โดยส่งข้อมูล **JSON** ที่สร้างขึ้นในตัวแปร jsonPayload ไปด้วย |
|  | ค่าผลลัพธ์จากคำขอ HTTP จะถูกเก็บไว้ในตัวแปร httpCode |
| if (httpCode > 0) { | ตรวจสอบสถานะของการส่งคำขอ: |
| Serial.println("Discord Alert Sent Successfully!"); | ถ้า httpCode > 0 แสดงว่าได้รับการตอบกลับที่ดีจาก Discord (เช่นรหัส HTTP 200), จะแสดงข้อความ **"Discord Alert Sent Successfully!"** |
| } else { | ถ้า httpCode <= 0 แสดงว่าเกิดข้อผิดพลาดในการส่งคำขอไปยัง Discord, จะแสดงข้อความ **"Error Sending Discord Alert."** |
| Serial.println("Error Sending Discord Alert."); |  |
| } |  |
|  |  |
| http.end(); | ปิดการเชื่อมต่อ HTTP หลังจากที่ส่งคำขอเสร็จสิ้นแล้ว เพื่อให้แน่ใจว่าไม่มีการใช้ทรัพยากรที่ไม่จำเป็น |
|  |  |
|  |  |
| void loop() { | ฟังก์ชัน loop() เป็นฟังก์ชันหลักที่ทำงานในลูปซ้ำ (infinite loop) ของโปรแกรม Arduino หรือ ESP32  โค้ดในฟังก์ชันนี้จะทำงานซ้ำ ๆ ตลอดเวลาจนกว่าอุปกรณ์จะถูกปิดหรือรีเซ็ต |
| Blynk.run(); | ฟังก์ชัน Blynk.run() จะเรียกใช้งาน Blynk Library เพื่อให้การเชื่อมต่อกับเซิร์ฟเวอร์ Blynk ทำงานได้อย่างต่อเนื่อง  จะทำการอัปเดตข้อมูลที่เชื่อมต่อกับแอป Blynk หรือจัดการการสื่อสารกับเซิร์ฟเวอร์ Blynk |
| timer.run(); | ฟังก์ชัน timer.run() จะตรวจสอบและเรียกใช้ฟังก์ชันที่ถูกตั้งเวลา (เช่น sendData()) ในระยะเวลาที่กำหนดไว้ใน BlynkTimer  ฟังก์ชันนี้จะทำงานโดยอัตโนมัติตามเวลาที่กำหนดใน timer.setInterval() ซึ่งส่งข้อมูล GPS ทุก 5 วินาทีในกรณีนี้ |
|  |  |
| while (gpsSerial.available()) { | ตรวจสอบว่า **Serial Port** (ที่เชื่อมต่อกับโมดูล GPS) มีข้อมูลที่จะอ่านหรือไม่ |
| gps.encode(gpsSerial.read()); | ถ้ามีข้อมูลใหม่จากโมดูล GPS มันจะทำการอ่านข้อมูลเหล่านั้น |
| } | gpsSerial.read(): อ่านข้อมูลที่เข้ามาจากพอร์ต **gpsSerial** (ข้อมูลที่ได้รับจากโมดูล GPS) |
| } | gps.encode(): ส่งข้อมูลที่ได้รับจากโมดูล GPS ไปให้ไลบรารี **TinyGPS++** เพื่อทำการประมวลผลข้อมูล GPS |