

Instrukcja uruchomienia upladera Nightscout wprost z modułu ESP32 dla Dexcom G6.

Poniższa instrukcja zawiera szczegółowe kroki jakie należy wykonać, aby zrobić niezależny od telefonu z aplikacją Dexcom G6 uploader bezpośrednio z nadajnika G6. Urządzenie komunikuje się drugim torem bluetooth i nie zakłóca odczytów w aplikacji na telefonie. Jest to dodatkowe narzędzie, które może pozwolić na odczyty glikemii w czasie kiedy mamy problem z aplikacją / telefonem / brakami odczytów w telefonie oraz w celach diagnostycznych samego nadajnika i przede wszystkim w celach edukacyjnych dla zainteresowanych. Sam moduł jest powszechnie dostępny i dość tani, można go kupić np. tu:

<https://kamami.pl/esp32/570606-plytka-rozwojowa-iot-wi-fibt-z-ukladem-esp32-i-wyswietlaczem-oled-096.html>

Pliki źródłowe z kodem zostały umieszczone na moim github:

<https://github.com/sarunia/Dexcom-G6-Nightscout-uploader>

po pobraniu repozytorium pliki należy wypakować do jednego folderu i zmodyfikować kilka wpisów pod własne ustawienia w pliku ESP32 Reader.ino otwierając go w Arduino IDE:

linia 70: `static std::string transmitterID = "8U1234";`

wpisać w cudzysłów swój numer ID nadajnika Dexcom G6,

linia 176: `const char *AP_SSID = "nazwa sieci WiFi";`

wpisać w cudzysłów własną nazwę sieci WiFi,

linia 177: `const char *AP_PWD = "hasło sieci WiFi";`

wpisać w cudzysłów hasło do sieci WiFi,

linia 222:

`https.begin("https://teststrony.herokuapp.com/api/v1/devicestatus");`

wpisać własną frazę strony Nightscout zamiast przykładu "teststrony" dla heroku, dla innych serwerów pewnie jest analogicznie, może być tu nawet adres IP własnego serwera Nightscout,

linia 224: `doc["secret"] = "kjsdjskvjxk8lsdkjkh7yruf4r97yicdjquwdbjj3";`

wpisać w cudzysłów hasło API_secret do strony Nightscout, ważne: hasło musi zostać skonwertowane do postaci SHA-1, można to zrobić konwerterem online, np.:

<http://www.sha1-online.com/>

[Home Page](#) | [SHA1 in JAVA](#) | [Secure password generator](#) | [Linux](#)

SHA1 and other hash functions online generator

haslostrony	hash
<div>sha-1 ▼</div>	

Result for sha1: **bf7bd24a4fb1c33d863b3c9d4fd3291e57f113c4**

linia 254: `https.begin("https://teststrony.herokuapp.com/api/v1/entries");`

jak wyżej podstawić własną nazwę,

linia 256: `doc["secret"] = "kjsdjskvjxk8lsdkjkh7yruf4r97yicdjquwdbjj3";`

jak wyżej podstawić własne hasło w postaci sha1,

linia 364: `advertisedDevice.haveName() && advertisedDevice.getName() == ("DexcomXY"))`

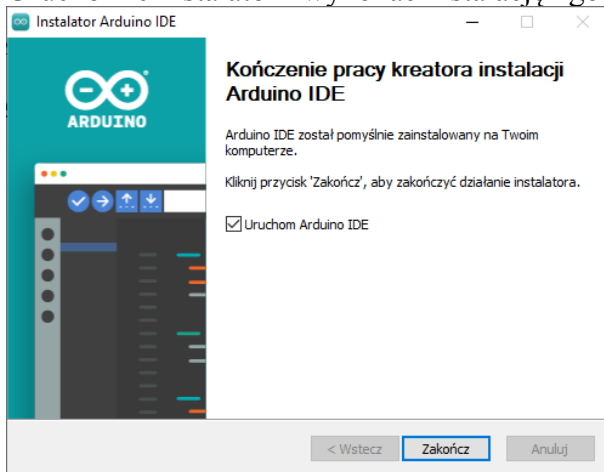
wstawić swoje dwa znaki zamiast liter XY, są to ostatnie dwa znaki ID nadajnika G6, pod taką nazwą rozgłasza się nadajnik przez bluetooth,

A teraz niezbędne kroki do zainstalowania wszystkich środowisk:

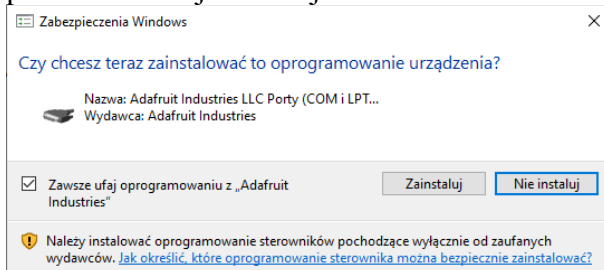
Zainstalować oprogramowanie ArduinoIDE pobierając stąd:

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

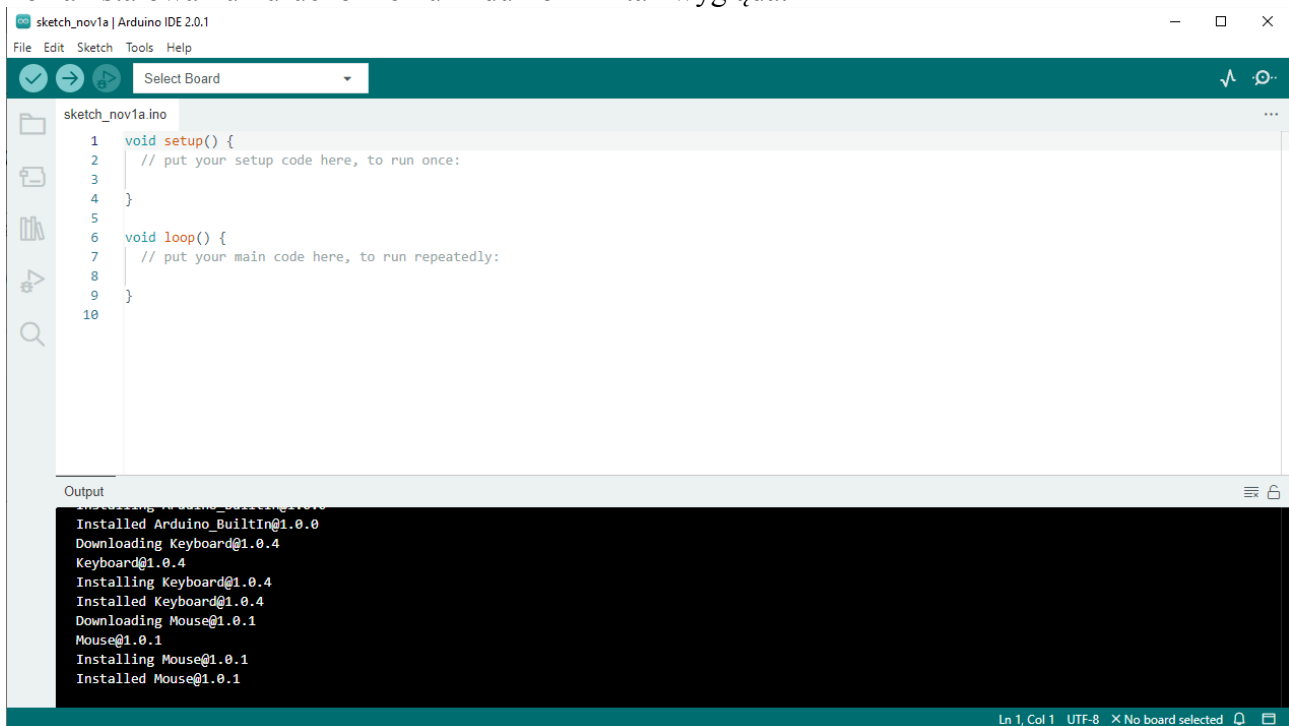
Uruchomić instalator i wykonać instalację zgodnie z domyślnymi ustawieniami:



podczas dalszej instalacji zezwolić na odblokowanie zapory i zainstalować potrzebne sterowniki:



Po zainstalowaniu i uruchomieniu Arduino IDE tak wygląda:



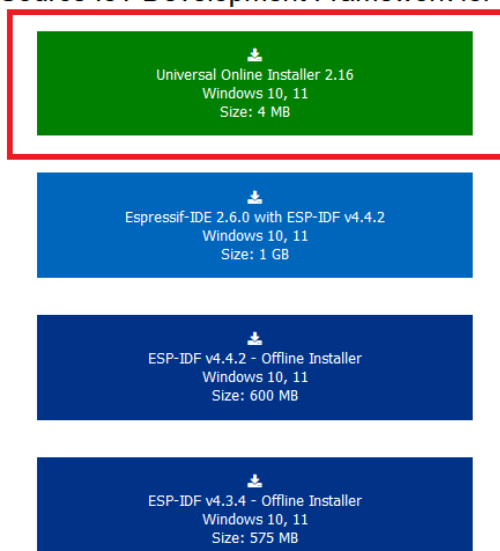
Zainstalować potrzebne narzędzie do obsługi modułów ESP w Arduino IDE, link prowadzi do online-nowego instalatora, co znacznie upraszcza całość instalacji składników ESP-IDF:

<https://dl.espressif.com/dl/esp-idf/?idf=4.4>

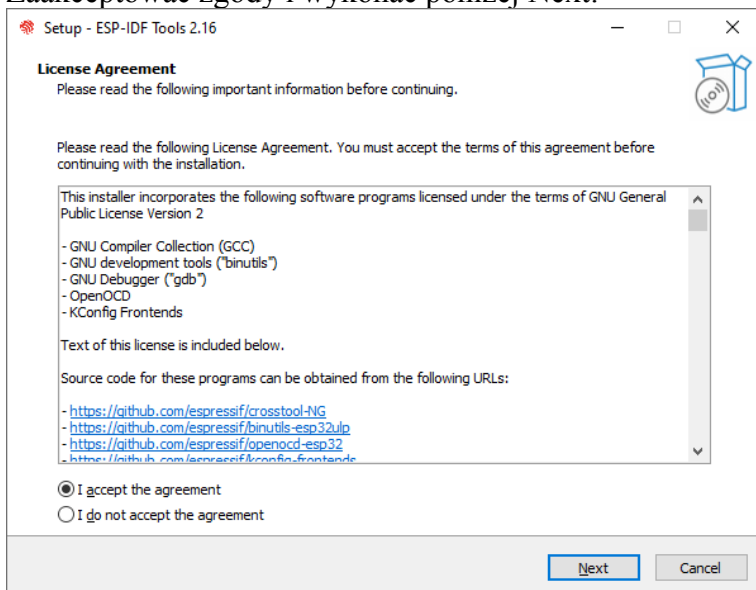
będąc na stronie wybrać pierwszą opcję:

ESP-IDF Windows Installer Download

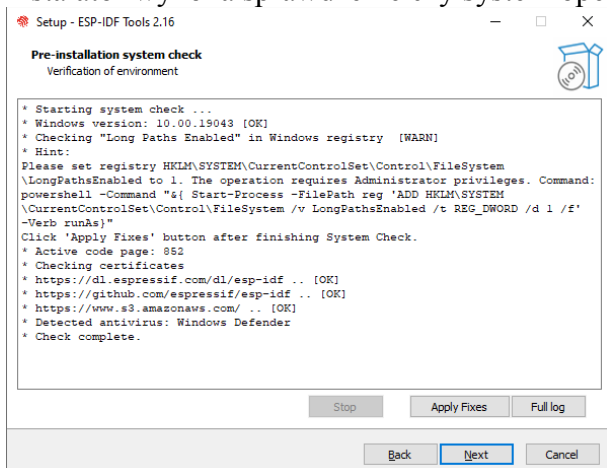
Open Source IoT Development Framework for ESP32



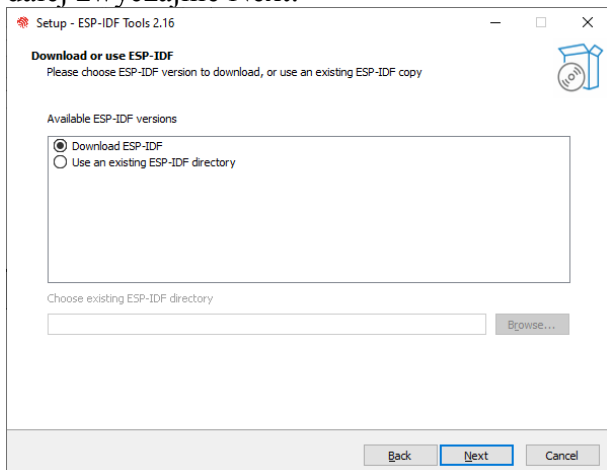
Zaakceptować zgody i wykonać poniżej Next:



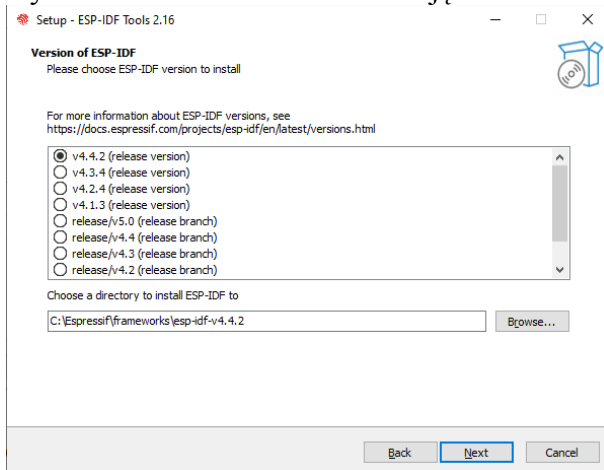
instalator wykona sprawdzenie czy system operacyjny nadaje się do obsługi ESP-IDF:



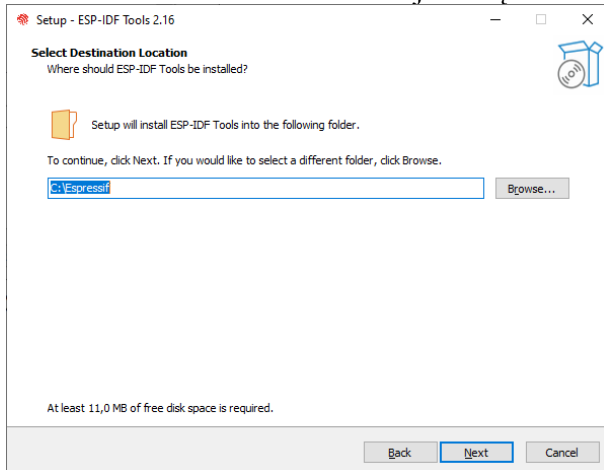
dalej zwyczajnie Next:



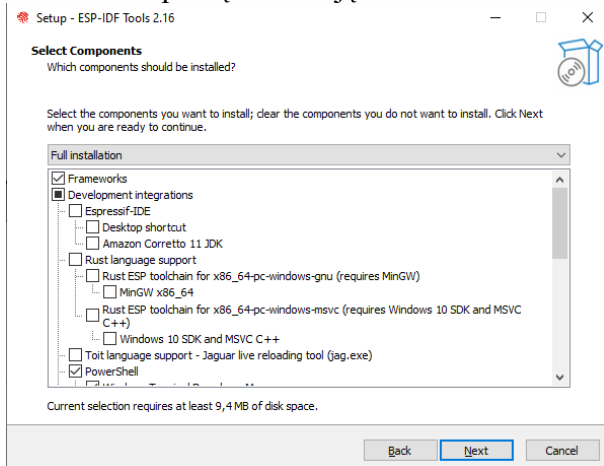
wybrać ostatni release i lokalizację:



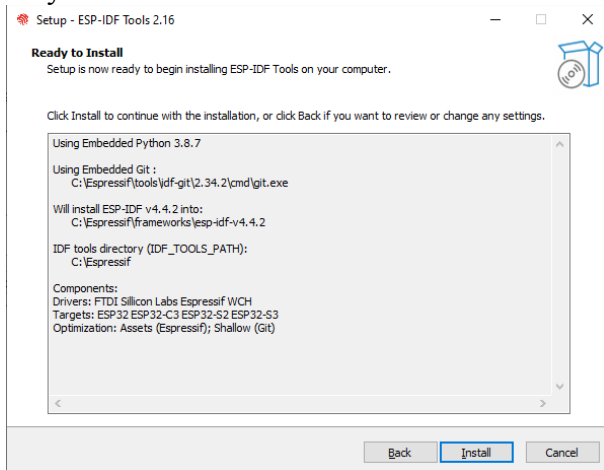
oraz wskazać folder dla instalacji narzędzia:



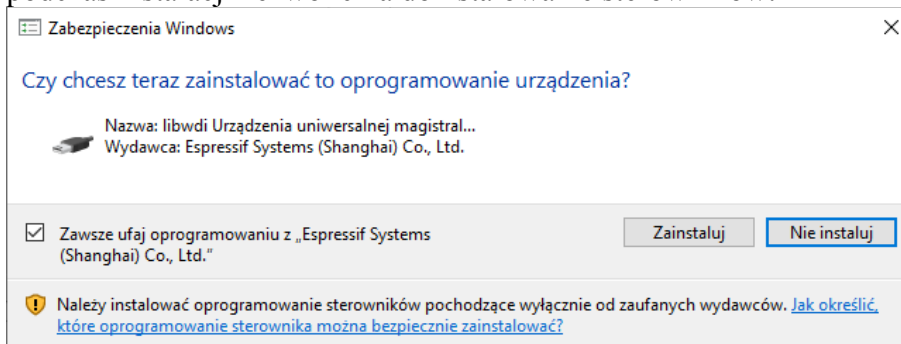
zatwierdzić pełną instalację:



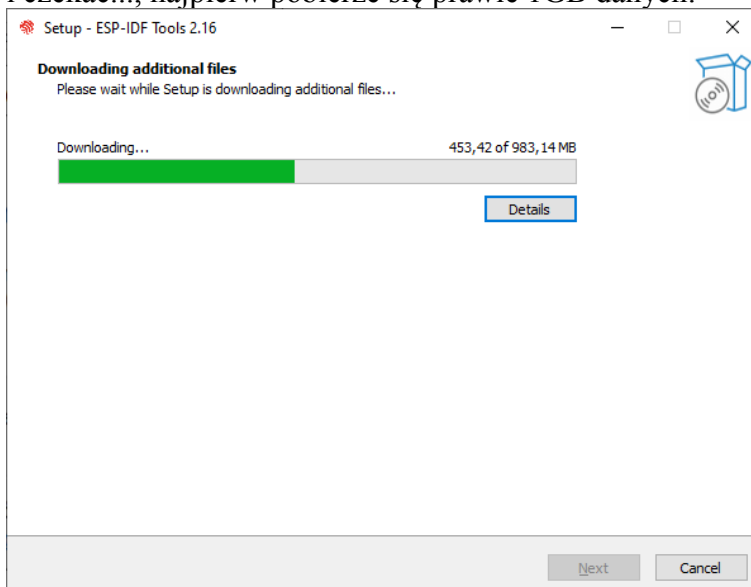
i wykonać Install:



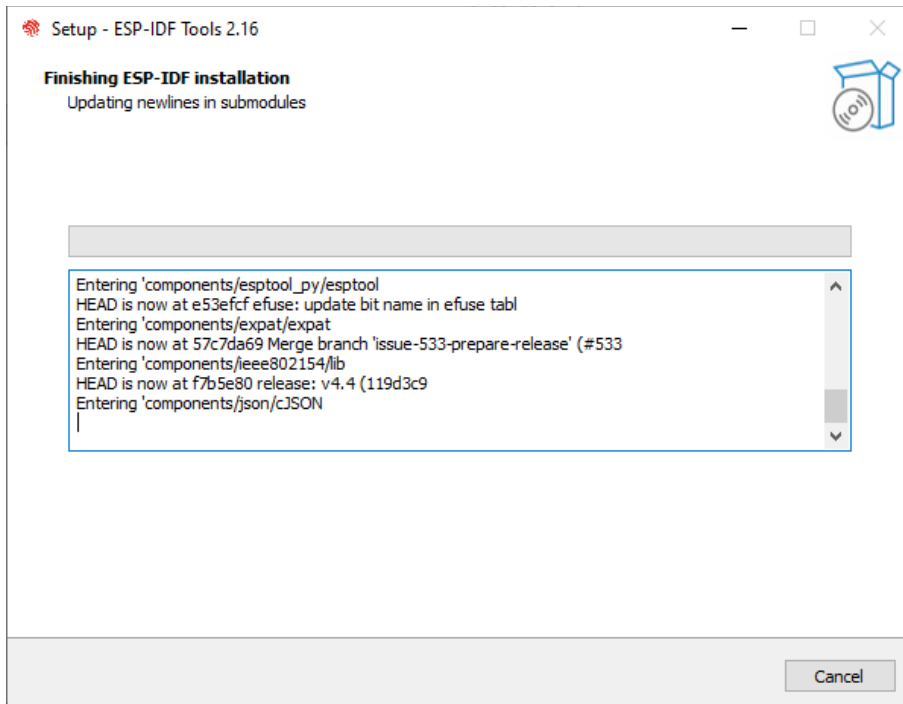
podczas instalacji zezwolić na doinstalowanie sterowników:



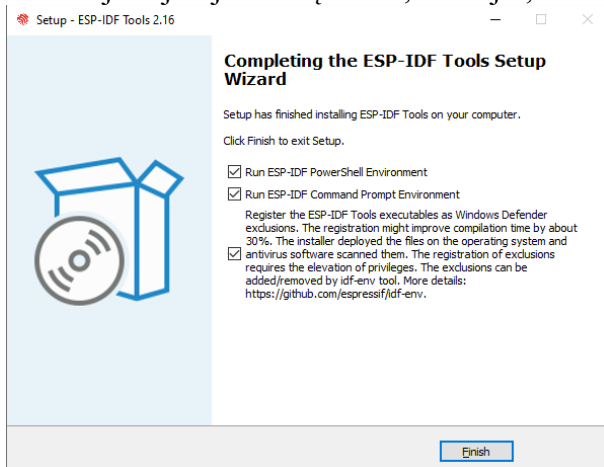
i czekać..., najpierw pobierze się prawie 1GB danych:



Po pobraniu pakietu instalacyjnego rozpocznie się automatyczne instalowanie wszystkich modułów:



instalacja zajmuje trochę czasu, czekaj..., na końcu powinno pojawić się okno:



Po kliknięciu na Finish instalator jeszcze uruchomi wiersz poleceń i wykona skrypty:

```
ESP-IDF 4.4 CMD - "C:\Espressif\idf_cmd_init.bat" esp-idf-e91d384503485fbb54f6ce3d11e841fe
Python 3.8.7
Using Git in C:\Espressif\tools\idf-git\2.34.2\cmd\
git version 2.34.1.windows.1
Setting IDF_PATH: C:\Espressif\frameworks\esp-idf-v4.4.2

Adding ESP-IDF tools to PATH...
C:\Espressif\tools\xtensa-esp32-elf\esp-2021r2-patch3-8.4.0\xtensa-esp32-elf\bin
C:\Espressif\tools\xtensa-esp32s2-elf\esp-2021r2-patch3-8.4.0\xtensa-esp32s2-elf\bin
C:\Espressif\tools\xtensa-esp32s3-elf\esp-2021r2-patch3-8.4.0\xtensa-esp32s3-elf\bin
C:\Espressif\tools\riscv32-esp-elf\esp-2021r2-patch3-8.4.0\riscv32-esp-elf\bin
C:\Espressif\tools\esp32ulp-elf\2.28.51-esp-20191205\esp32ulp-elf-binutils\bin
C:\Espressif\tools\esp32s2ulp-elf\2.28.51-esp-20191205\esp32s2ulp-elf-binutils\bin
C:\Espressif\tools\cmake\3.23.1\bin
C:\Espressif\tools\openocd-esp32\v0.11.0-esp32-20220411\openocd-esp32\bin
C:\Espressif\tools\ninja\1.10.2\
C:\Espressif\tools\idf-exe\1.0.3\
C:\Espressif\tools\ccache\4.3\ccache-4.3-windows-64
C:\Espressif\tools\dfu-util\0.9\dfu-util-0.9-win64
C:\Espressif\frameworks\esp-idf-v4.4.2\tools

Checking if Python packages are up to date...
Python requirements from C:\Espressif\frameworks\esp-idf-v4.4.2\requirements.txt are satisfied.

Done! You can now compile ESP-IDF projects.
Go to the project directory and run:

idf.py build

C:\Espressif\frameworks\esp-idf-v4.4.2>
```

oraz :

```
ESP-IDF 4.4 PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

Using Python in C:\Espressif\python_env\idf4.4_py3.8_env\Scripts
Python 3.8.7
Using Git in C:\Espressif\tools\idf-git\2.34.2\cmd\
git version 2.34.1.windows.1
Setting IDF_PATH: C:\Espressif\frameworks\esp-idf-v4.4.2
Adding ESP-IDF tools to PATH...

Name                                Value
----                                -
OPENOCD_SCRIPTS                     C:\Espressif\tools\openocd-esp32\v0.11.0-esp32-20220411\openocd-esp32\share\openocd\s...
IDF_CCACHE_ENABLE                   1
IDF_PYTHON_ENV_PATH                 C:\Espressif\python_env\idf4.4_py3.8_env

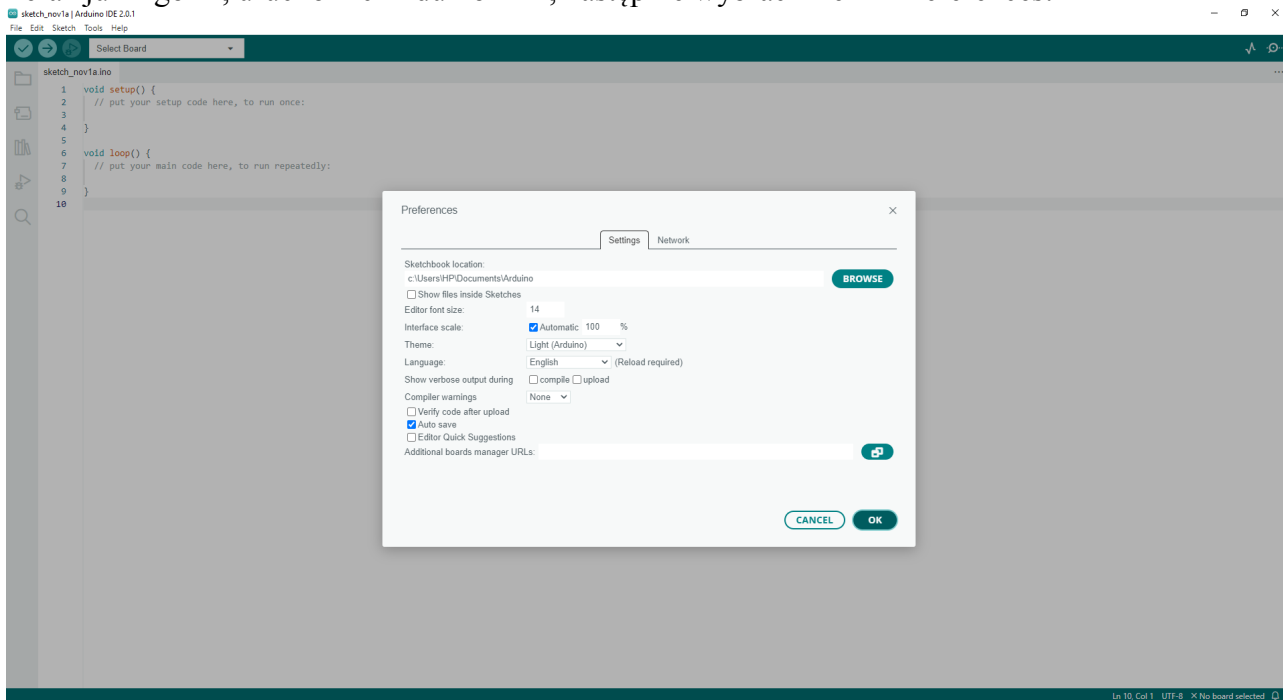
Added to PATH
-----
C:\Espressif\frameworks\esp-idf-v4.4.2\components\esptool_py\esptool
C:\Espressif\frameworks\esp-idf-v4.4.2\components\app_update
C:\Espressif\frameworks\esp-idf-v4.4.2\components\espcoredump
C:\Espressif\frameworks\esp-idf-v4.4.2\components\partition_table
C:\Espressif\tools\xtensa-esp32-elf\esp-2021r2-patch3-8.4.0\xtensa-esp32-elf\bin
C:\Espressif\tools\xtensa-esp32s2-elf\esp-2021r2-patch3-8.4.0\xtensa-esp32s2-elf\bin
C:\Espressif\tools\xtensa-esp32s3-elf\esp-2021r2-patch3-8.4.0\xtensa-esp32s3-elf\bin
C:\Espressif\tools\riscv32-esp-elf\esp-2021r2-patch3-8.4.0\riscv32-esp-elf\bin
C:\Espressif\tools\esp32ulp-elf\2.28.51-esp-20191205\esp32ulp-elf-binutils\bin
C:\Espressif\tools\esp32s2ulp-elf\2.28.51-esp-20191205\esp32s2ulp-elf-binutils\bin
C:\Espressif\tools\cmake\3.23.1\bin
C:\Espressif\tools\openocd-esp32\v0.11.0-esp32-20220411\openocd-esp32\bin
C:\Espressif\tools\ninja\1.10.2\
C:\Espressif\tools\idf-exe\1.0.3\
C:\Espressif\tools\ccache\4.3\ccache-4.3-windows-64
C:\Espressif\tools\dfu-util\0.9\dfu-util-0.9-win64
C:\Espressif\frameworks\esp-idf-v4.4.2\tools
Checking if Python packages are up to date...
Python requirements from C:\Espressif\frameworks\esp-idf-v4.4.2\requirements.txt are satisfied.

Done! You can now compile ESP-IDF projects.
Go to the project directory and run:

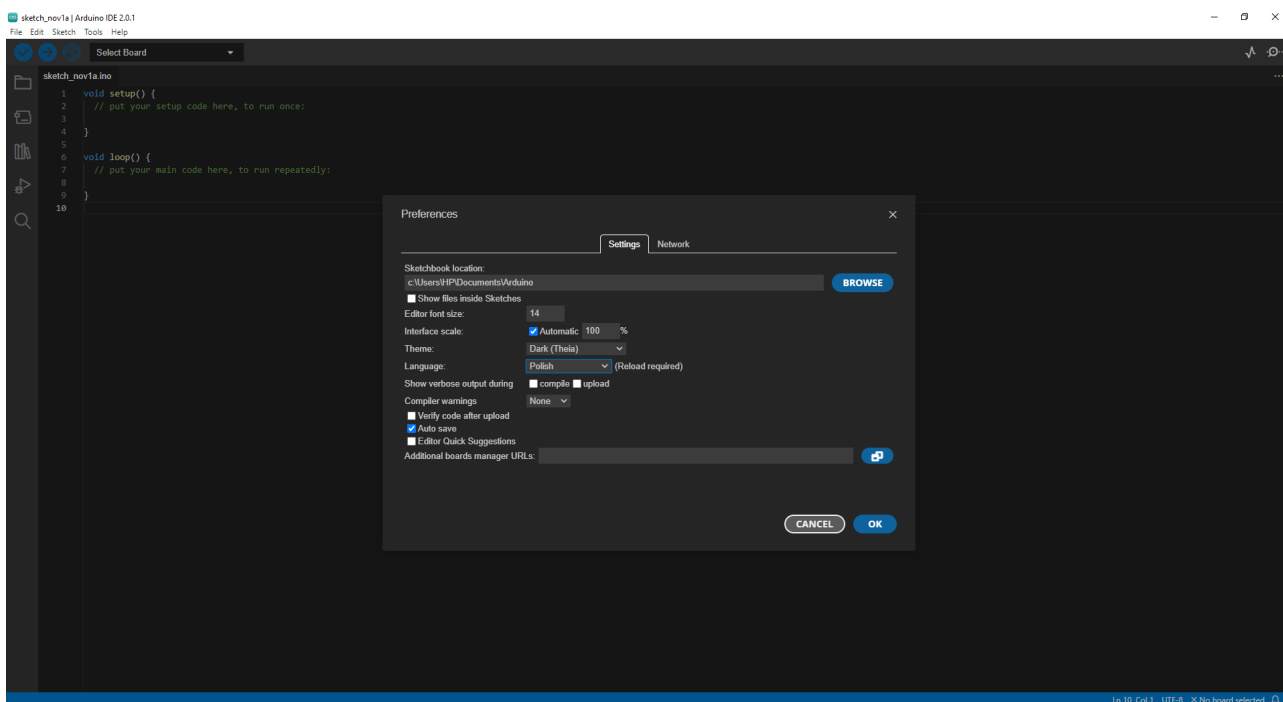
idf.py build

PS C:\Espressif\frameworks\esp-idf-v4.4.2>
```


Teraz już z górki, uruchomić Arduino IDE, następnie wybrać File--->Preferences:



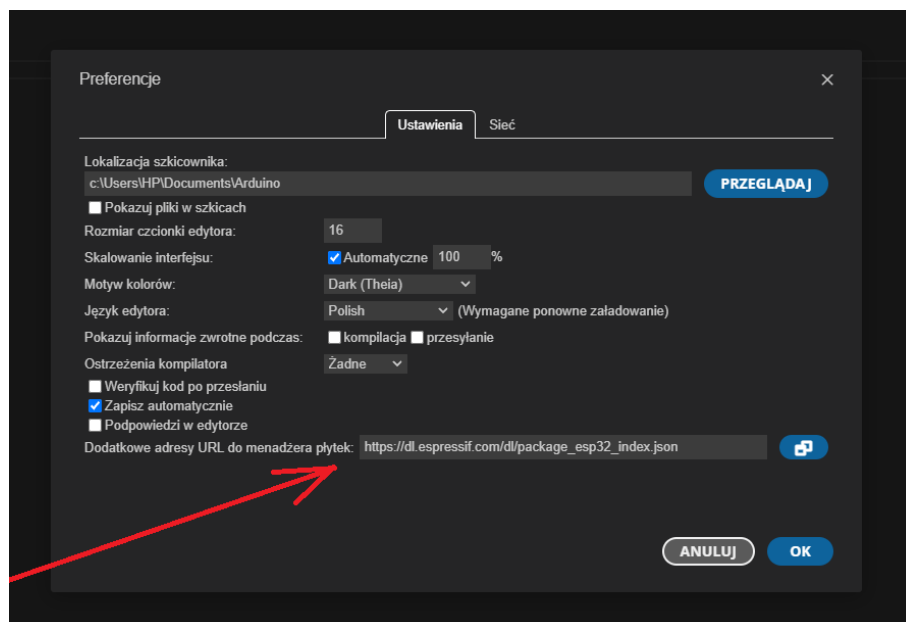
i żeby nie raziło po oczach i w obcym języku to można od razu wybrać ciemny tryb i język PL:



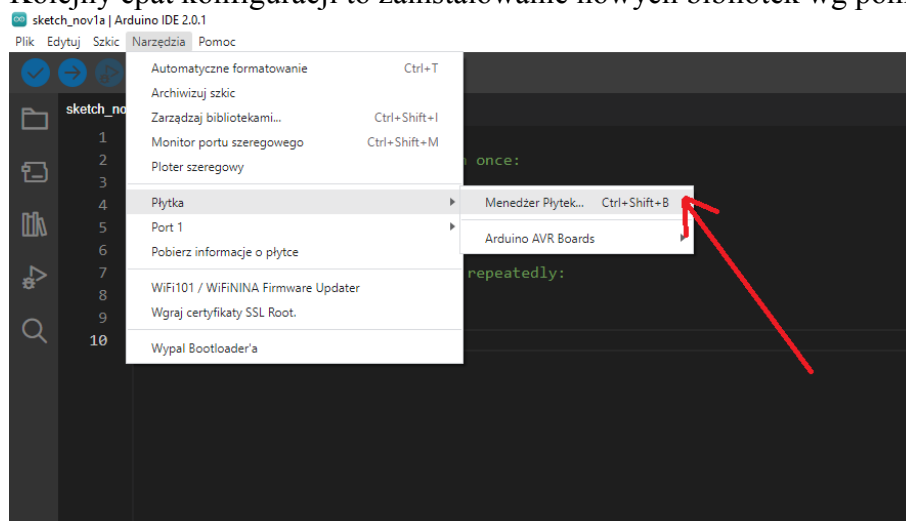
W ostatniej linii dokleić link z adresami menedżera płytek ESP32:

https://dl.espressif.com/dl/package_esp32_index.json

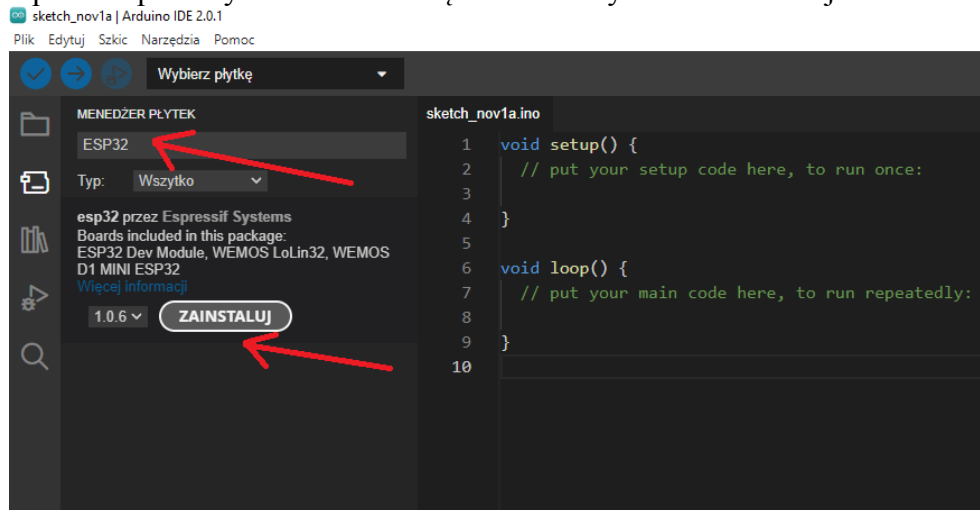
i zatwierdzić zmiany OK



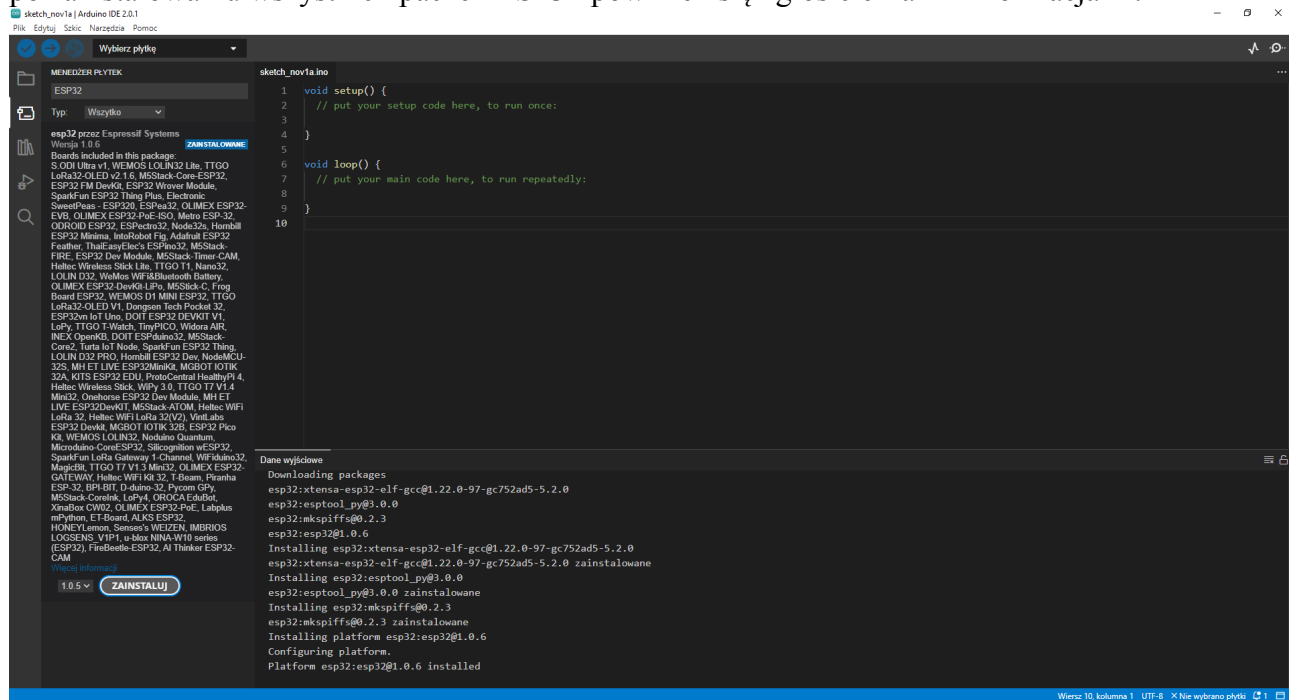
Kolejnym etapem konfiguracji to zainstalowanie nowych bibliotek wg poniższego screena:



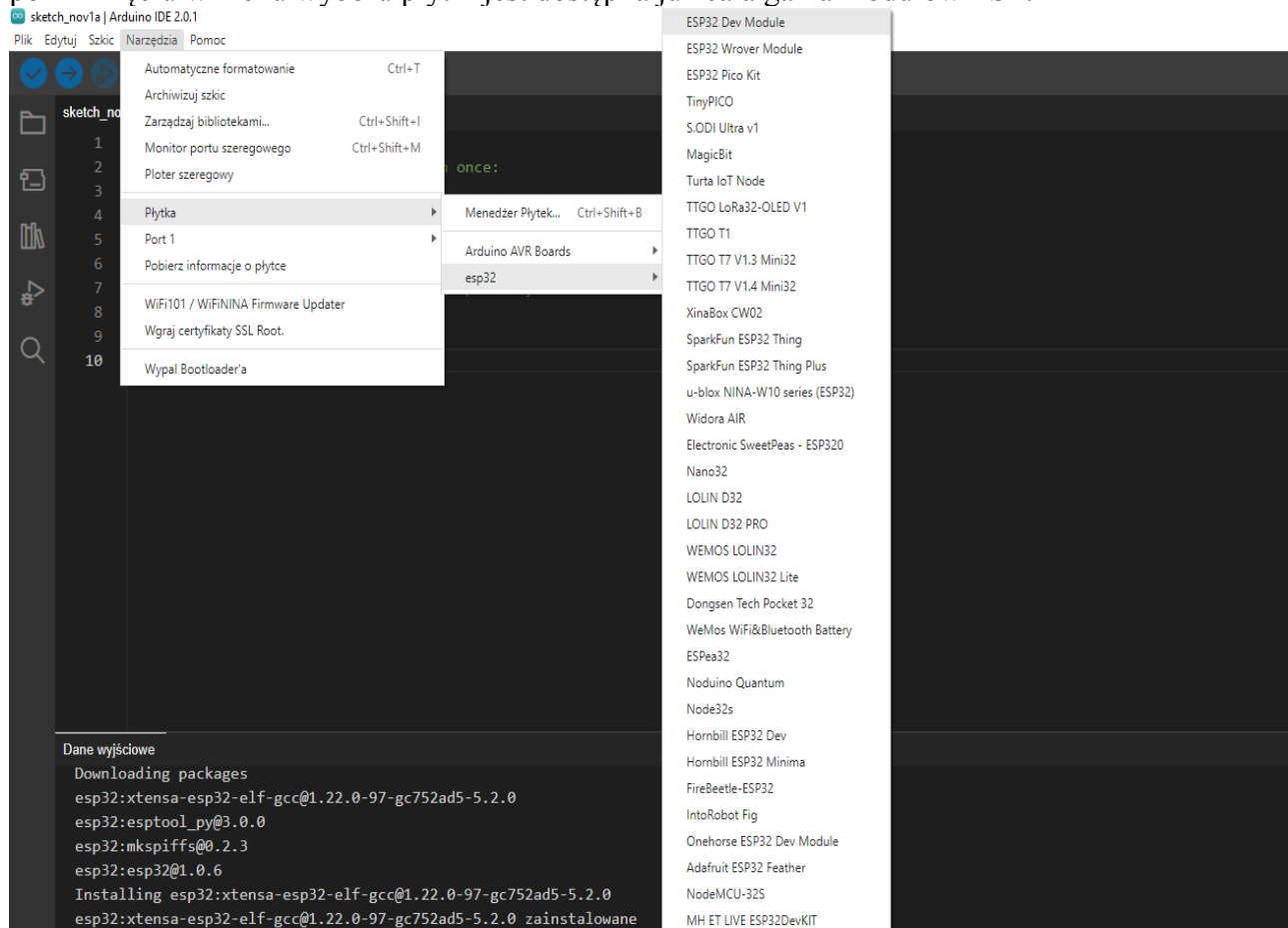
wpisać w pole wyszukiwania frazę: ESP32 i wykonać Zainstaluj:



po zainstalowaniu wszystkich paczek ESP32 powinien się zgłosić ekran z informacjami:



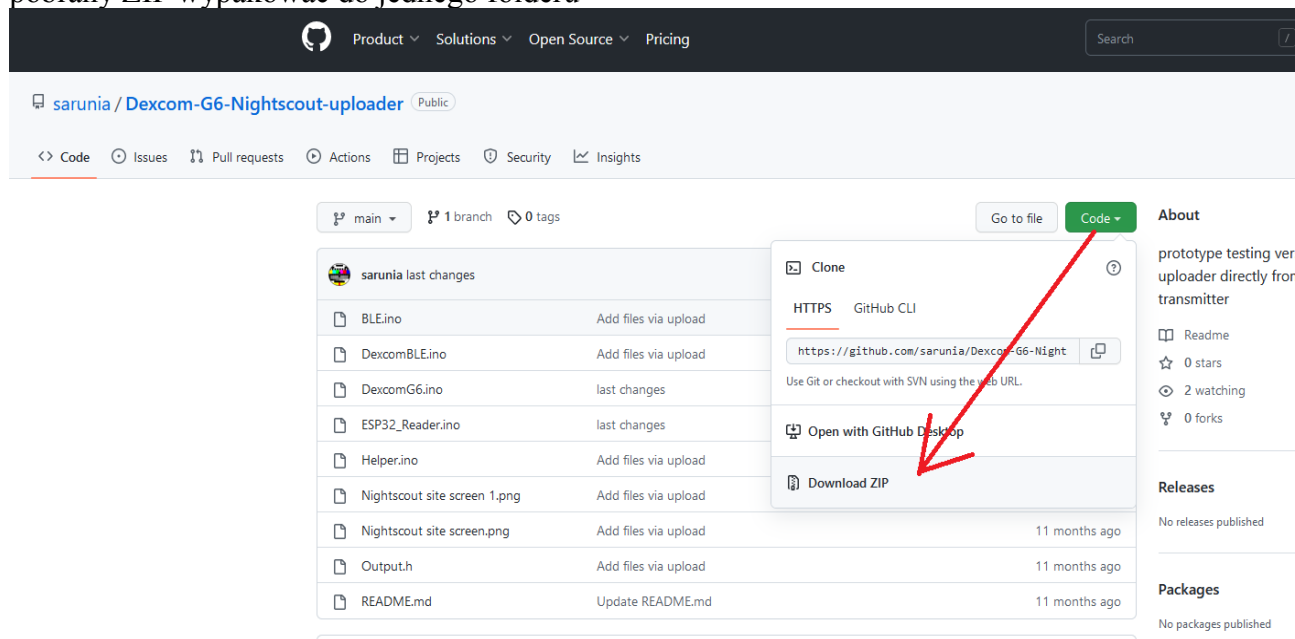
po kliknięciu w menu wyboru płytki jest dostępna już cała gama modułów ESP:



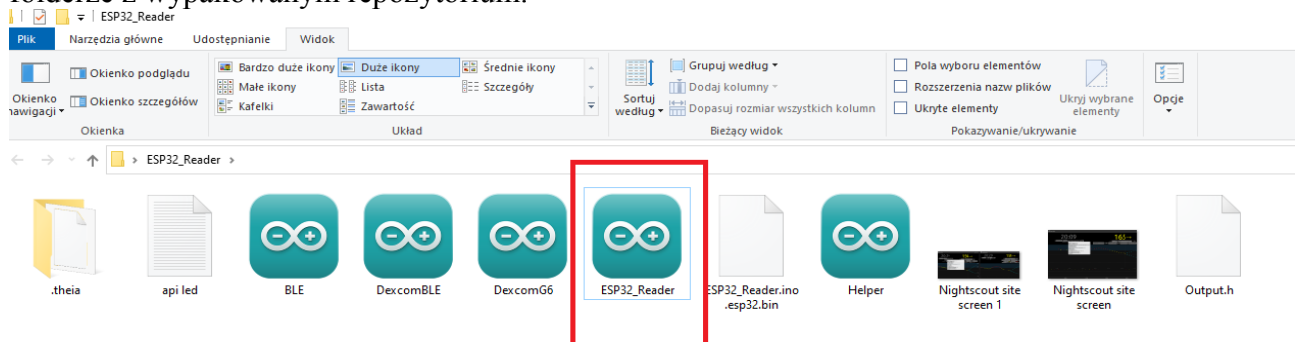
Pobrać pliki z mojego repozytorium dla uploadera Nightscout dla nadajnika Dexcom G6:

<https://github.com/sarunia/Dexcom-G6-Nightscout-uploader>

pobrane ZIP wypakować do jednego folderu



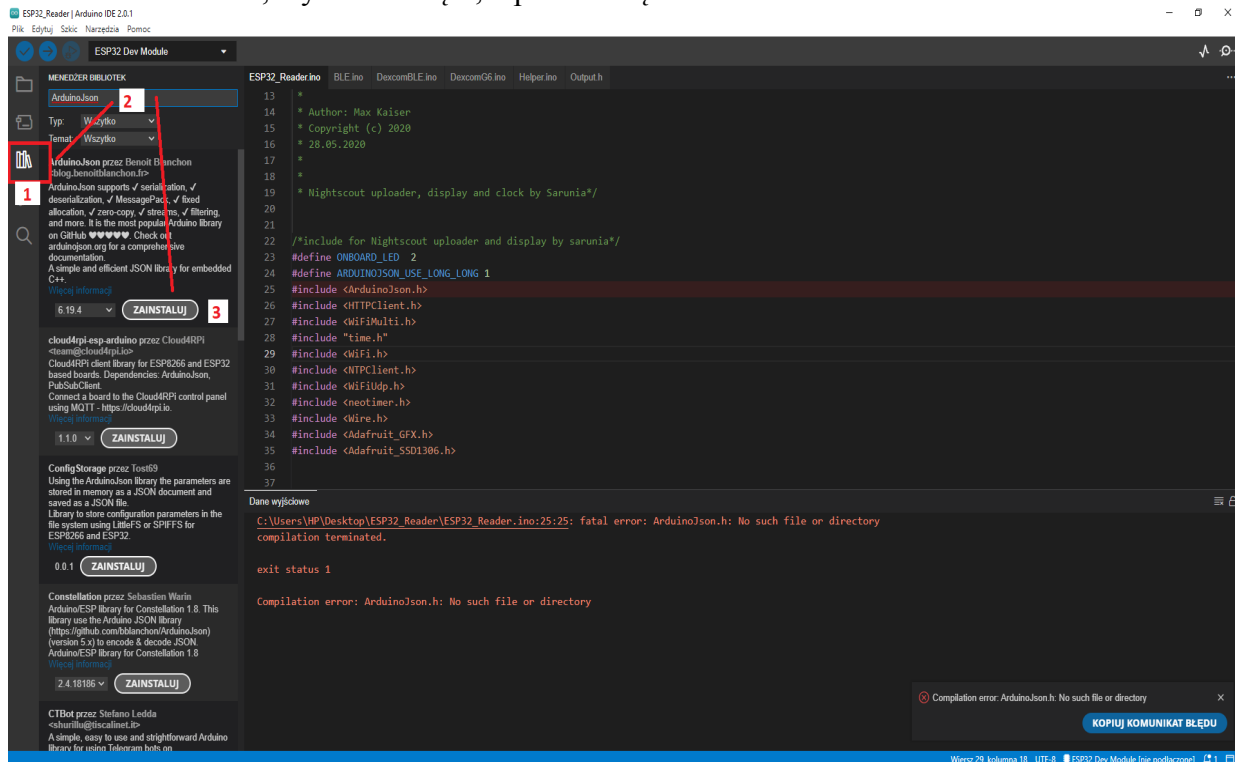
Zamknąć ArduinoIDE i ponownie je uruchomić klikając bezpośrednio w zaznaczony plik w folderze z wypakowanym repozytorium:



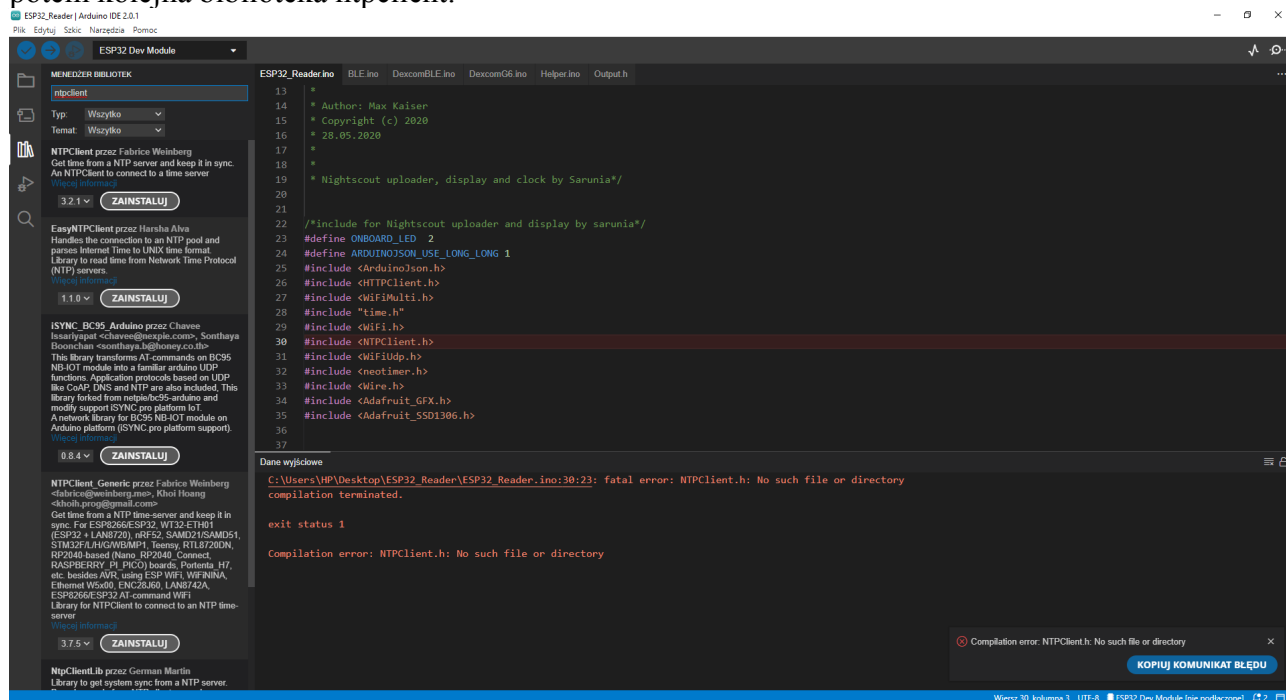
Do skompilowania pliku wsadowego do modułu ESP32 będą potrzebne biblioteki, które są wołane w plikach nagłówkowych kodu:

```
#include <ArduinoJson.h>
#include <HttpClient.h>
#include <WiFiMulti.h>
#include "time.h"
#include <WiFi.h>
#include <NTPClient.h>
#include <WiFiUdp.h>
#include <neotimer.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
```

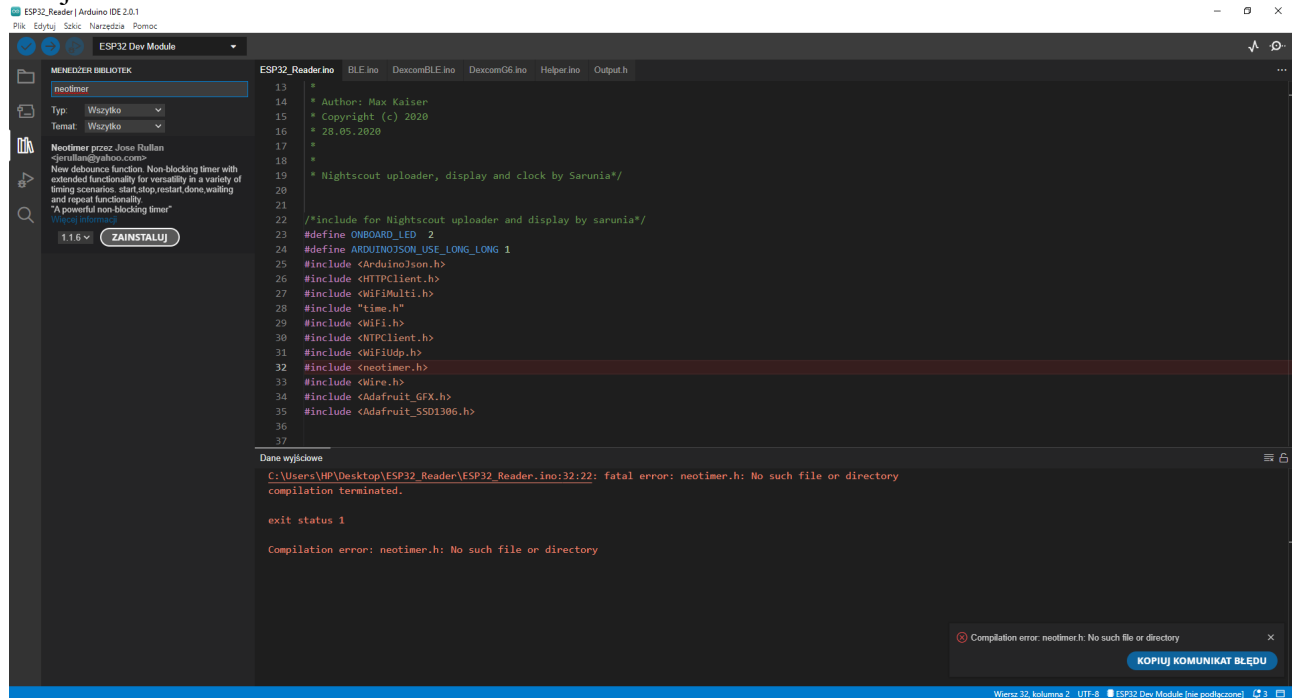
W tym celu trzeba uruchomić narzędzie pobierania i instalowania bibliotek, przykład na pobranie plików ArduinoJSON, wybrać ikonę 1, wpisać frazę ArduinoJson i zainstalować:



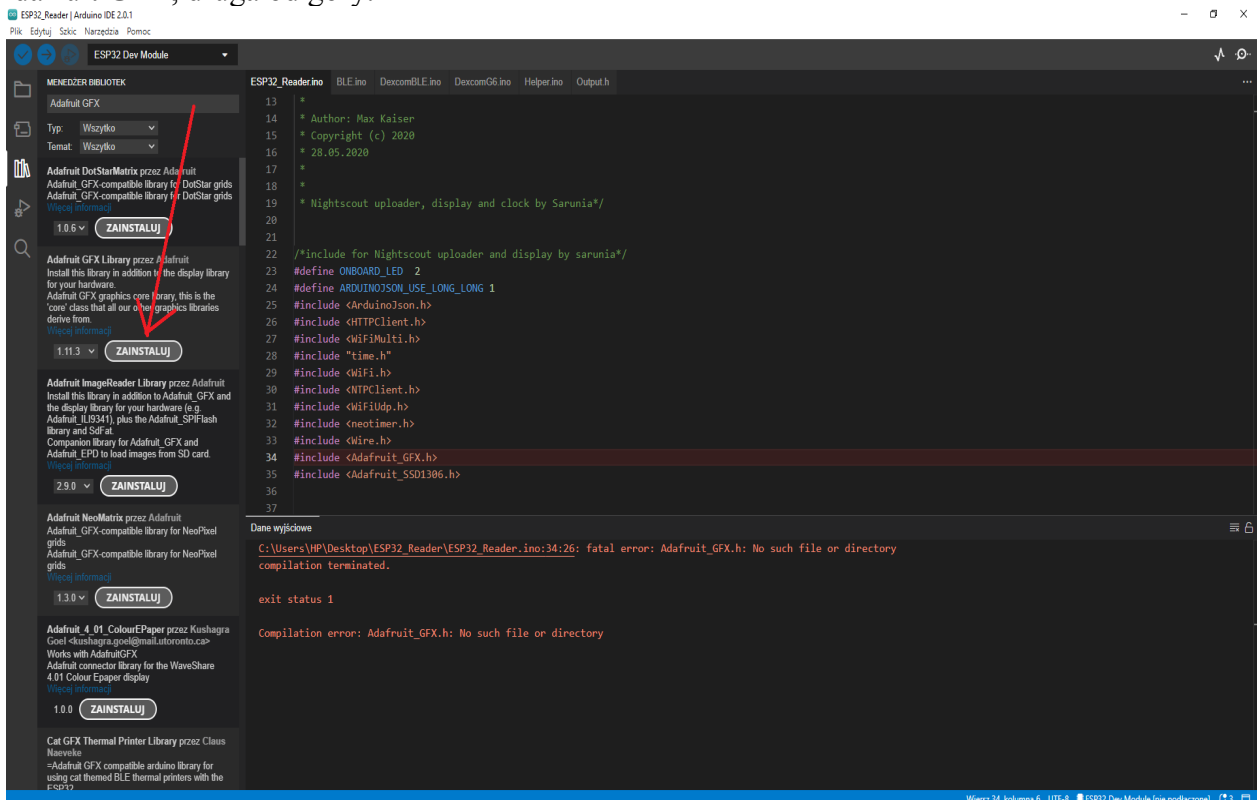
potem kolejna biblioteka ntpclient:



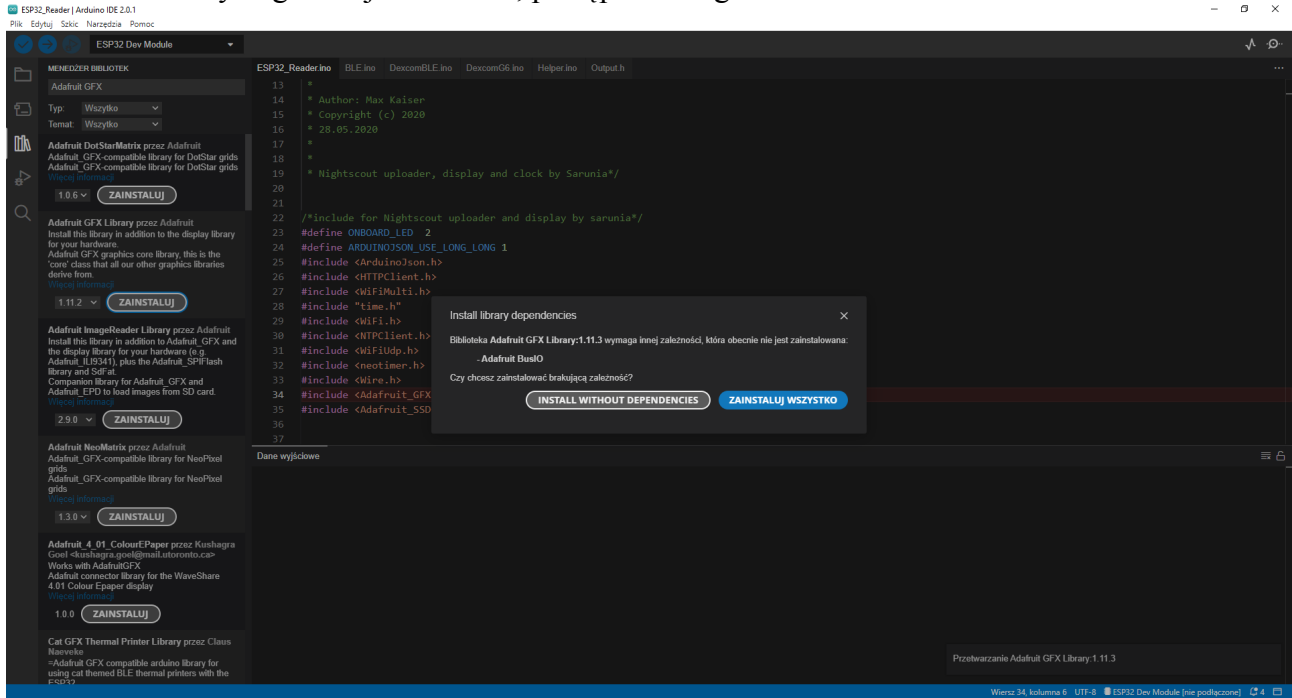
dalej neotimer:



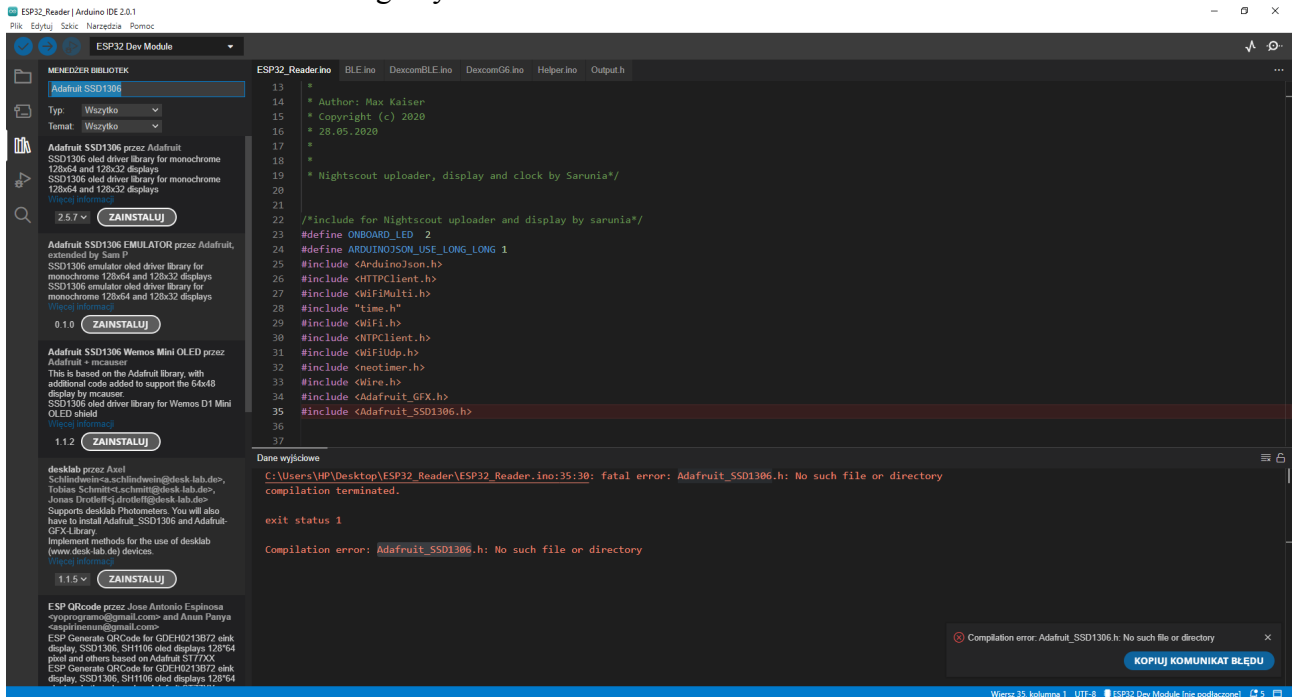
Adafruit GFX, druga od góry:



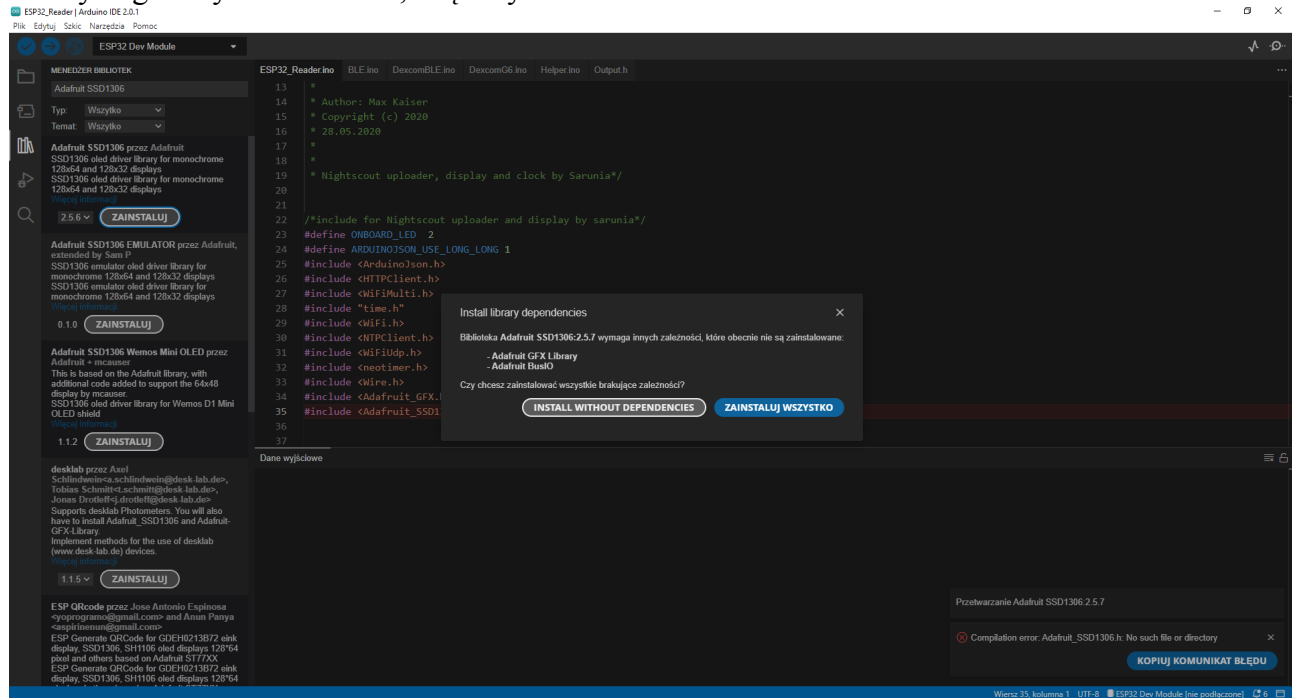
Adafruit GFX wymaga innej zależności, postępować wg ekranu i ZAINSTALUJ WSZYSTKO:



i ostatnia biblioteka do obsługi wyświetlacza Adafruit SSD1306 modułu ESP32:

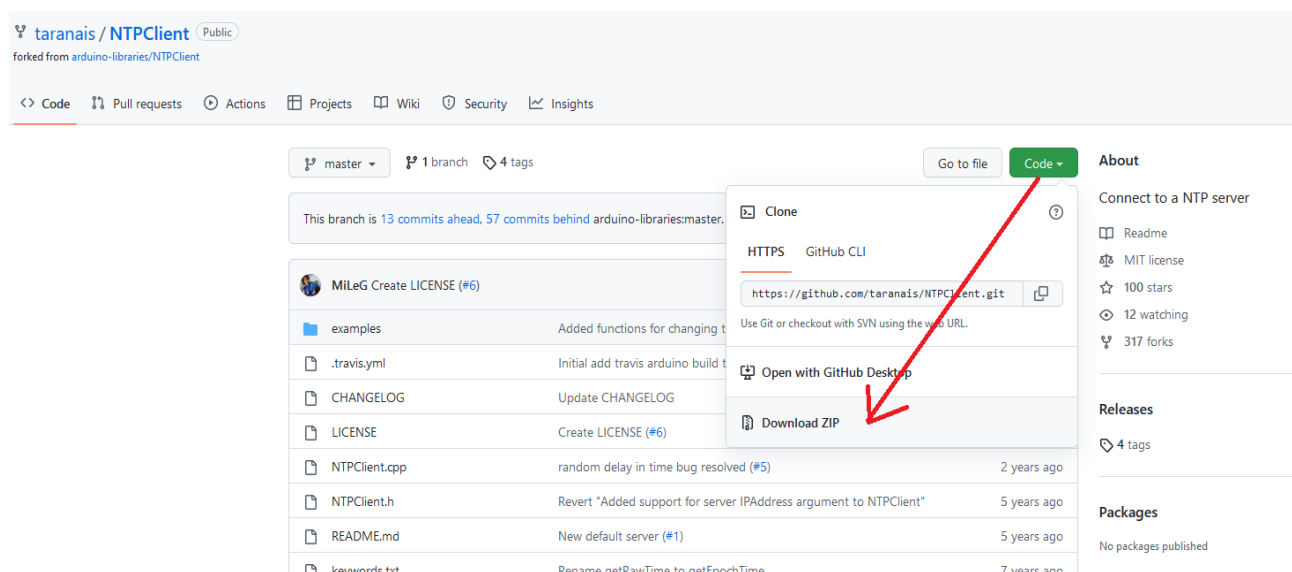


też wymaga innych zależności, więc wykonać **ZAINSTALUJ WSZYSTKO**:

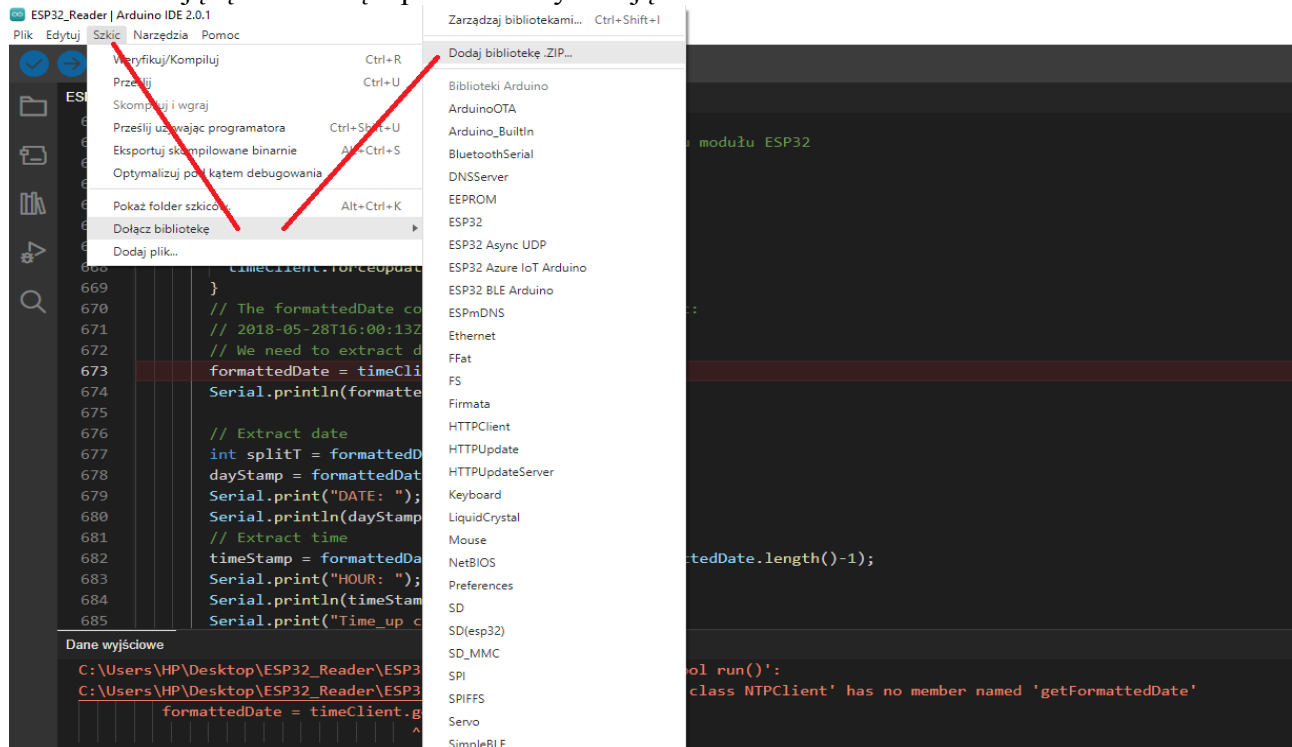


Podczas próby kompilacji na końcu dostajemy info, że biblioteka NTPClient nie zawiera obsługi naszego kodu, a więc należy zmienić bibliotekę NTPClient. W tym celu pobrać plik ZIP z repozytorium na github:

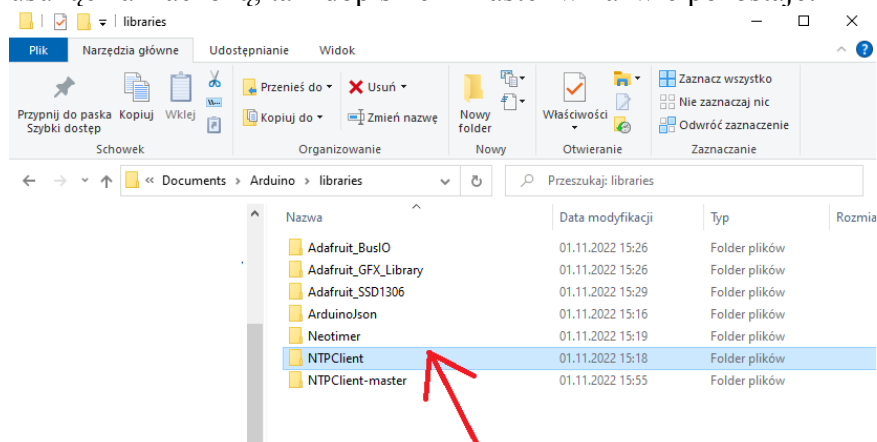
<https://github.com/taranais/NTPClient>



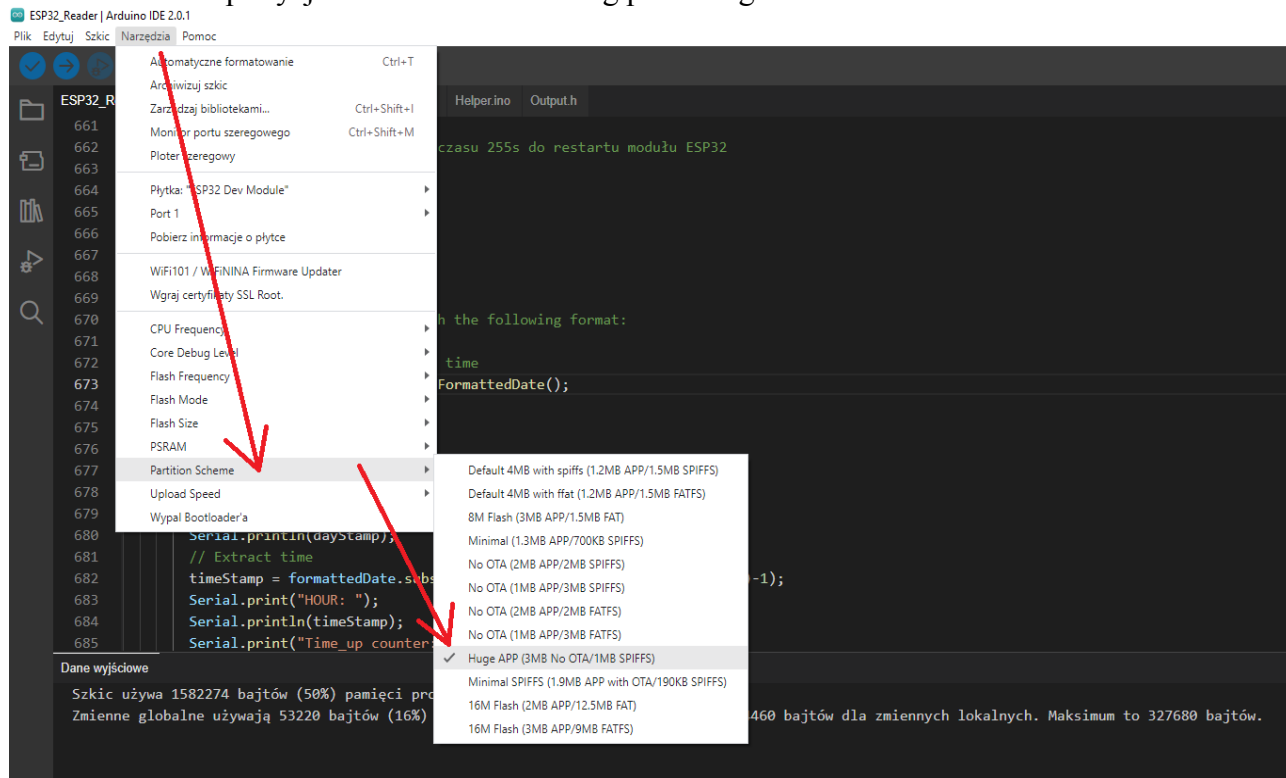
i dodać brakującą bibliotekę z pliku ZIP wybierając z menu:



Następnie koniecznie trzeba usunąć poprzednią bibliotekę NTPClient, najlepiej w pasek wyszukiwania w Windows wpisać frazę ntpclient i przejść do lokalizacji zainstalowanych bibliotek, usunąć zaznaczoną, ta z dopiskiem master w nazwie pozostaje:

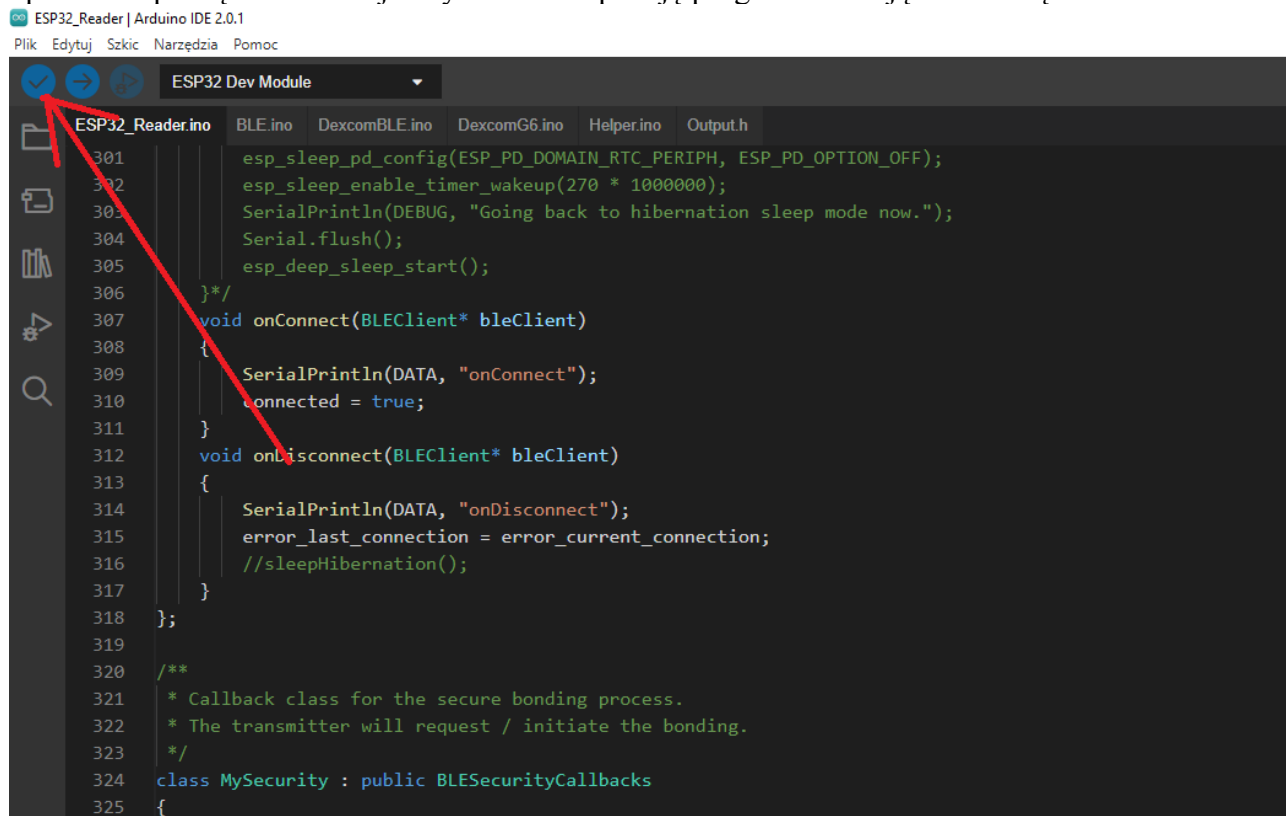


Przy ponownej próbie kompilacji wyskoczy jeszcze jeden błąd, że szkic jest zbyt duży, rada na to zmienić rozmiar partycji dla modułu ESP32 wg poniższego screenu:



jak widać w końcu kompilacja się udała!

Teraz przejść do zmian jakie należy zrobić pod własne ustawienia w pliku ESP32_Reader.ino opisane na początku instrukcji. Wykonać kompilację programu klikając w ikonkę:



Jeśli wszystko przebiegło pomyślnie to na dolnym ekranie powinno się pojawić:

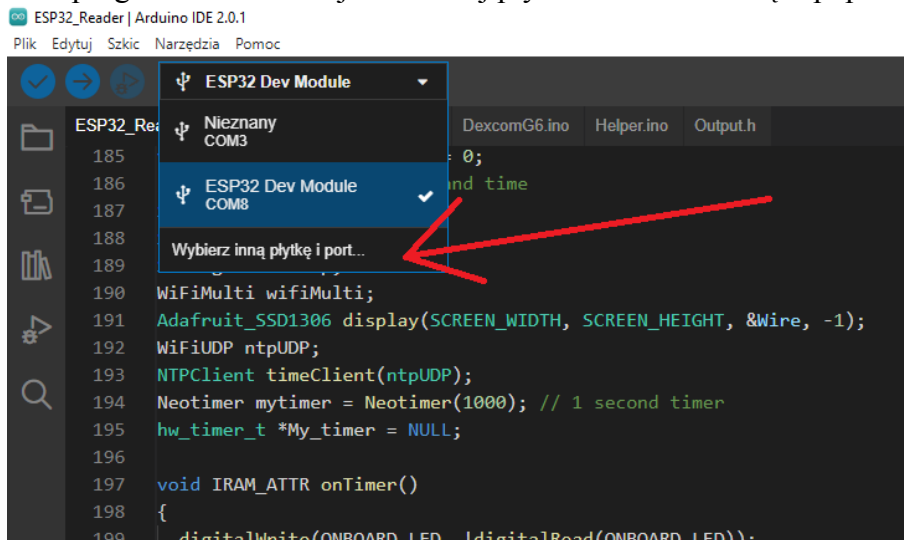
```
190 WiFiMulti wifiMulti;
191 Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
192 WiFiUDP ntpUDP;
193 NTPClient timeClient(ntpUDP);
194 Neotimer mytimer = Neotimer(1000); // 1 second timer
195 hw_timer_t *My_timer = NULL;
196
197 void IRAM_ATTR onTimer()
198 {
199   digitalWrite(ONBOARD_LED, !digitalRead(ONBOARD_LED));
200   //timer_1s = 1;
201   time_up++;
202 }
203
204 unsigned long long getTime() // Function that gets current epoch time
205 {
206   time_t now;
207   struct tm timeinfo;
208   if (!getLocalTime(&timeinfo))
209   {
210     return(0);
211   }
212 }
213 // End of main function
```

Dane wyjściowe Monitor portu szeregowego

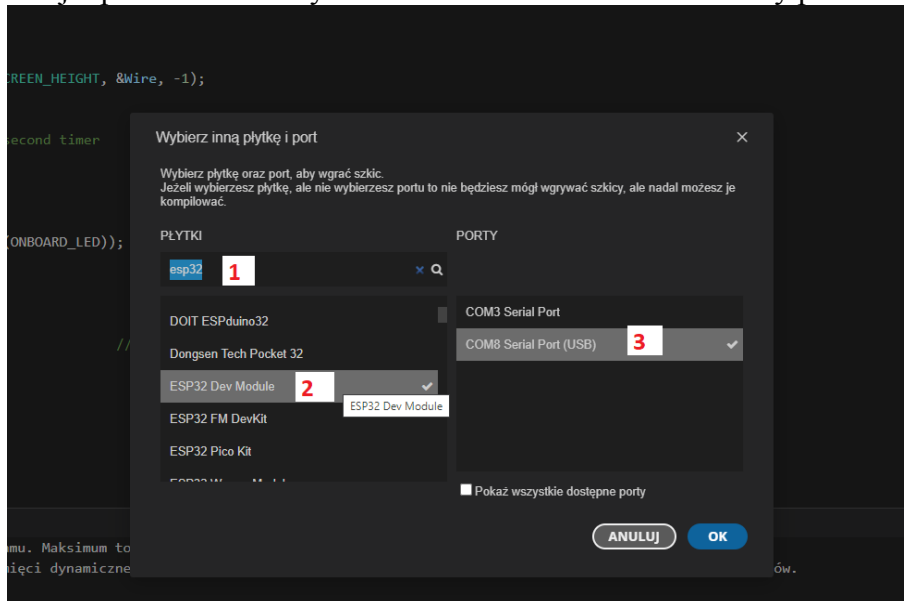
Szkic używa 1585462 bajtów (50%) pamięci programu. Maksimum to 3145728 bajtów.
Zmienne globalne używają 53236 bajtów (16%) pamięci dynamicznej, pozostawiając 274444 bajtów dla zmiennych lokalnych. Maksimum to 327680 bajtów.

Zamknąć ArduinoIDE, podłączyć płytkę ESP32 pod port USB komputera, w Menedżerze urządzeń wy badać pod którym portem COM zgłasza się nasze urządzenie. Następnie ponownie uruchomić ArduinoIDE i poszukać płytki ESP32.

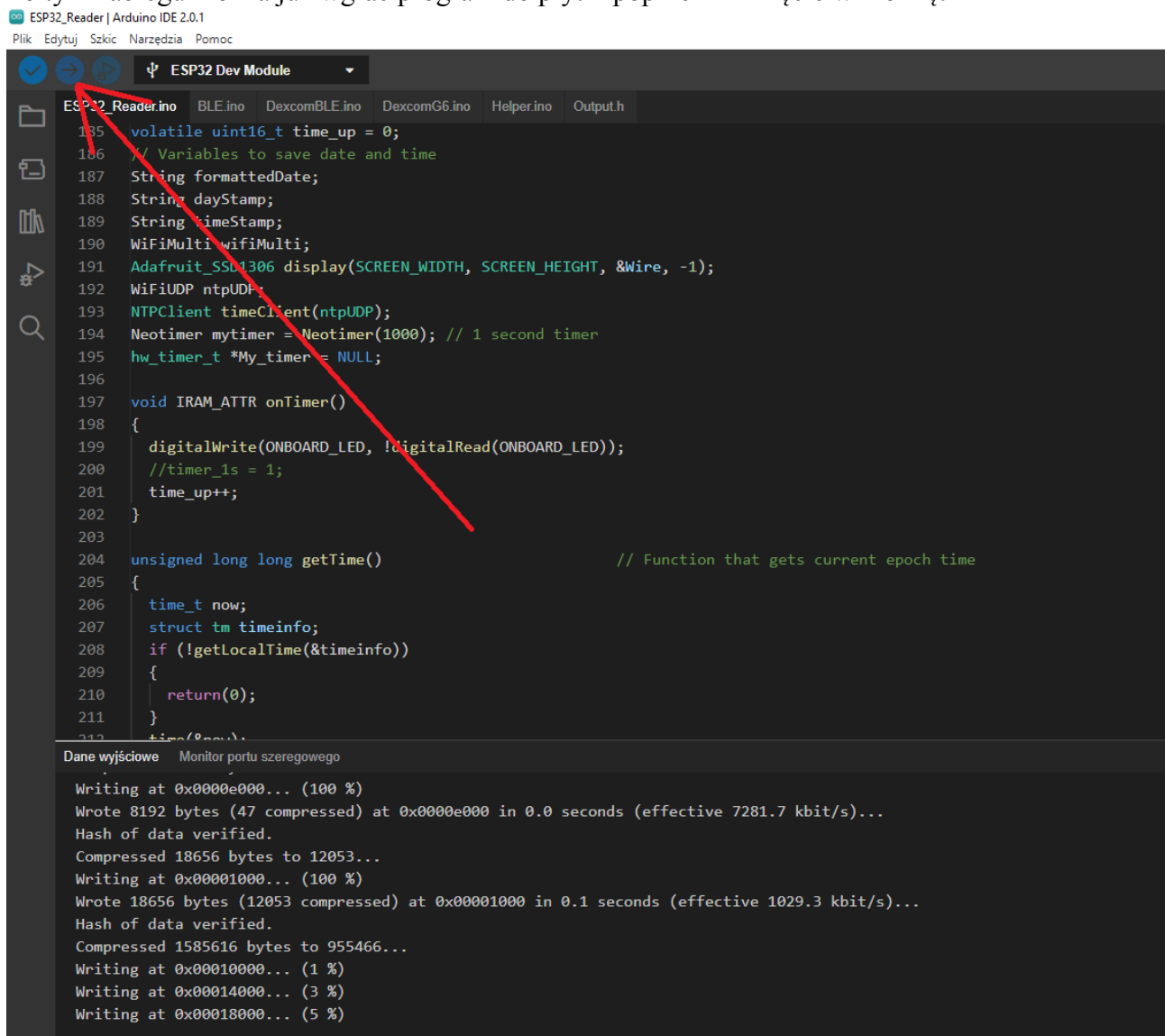
Jeśli program sam nie znajdzie naszej płytki to zrobić to z ręki poprzez rozwijane menu:



dalej wpisać ESP32 i wybrać ESP32 Dev Module i właściwy port COM:

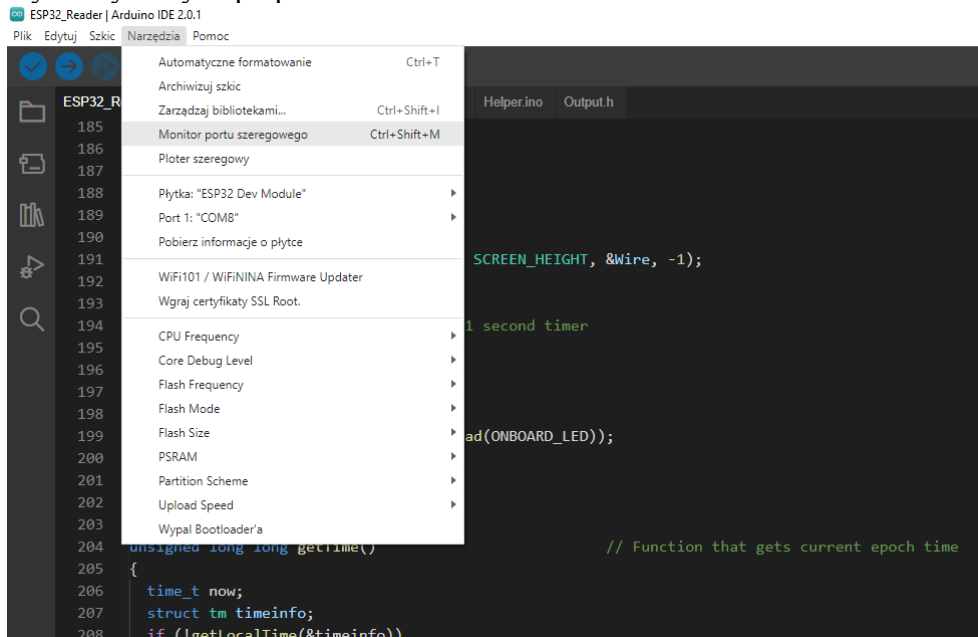


Po tym zabiegu można już wgrać program do płytki poprzez kliknięcie w ikonkę:

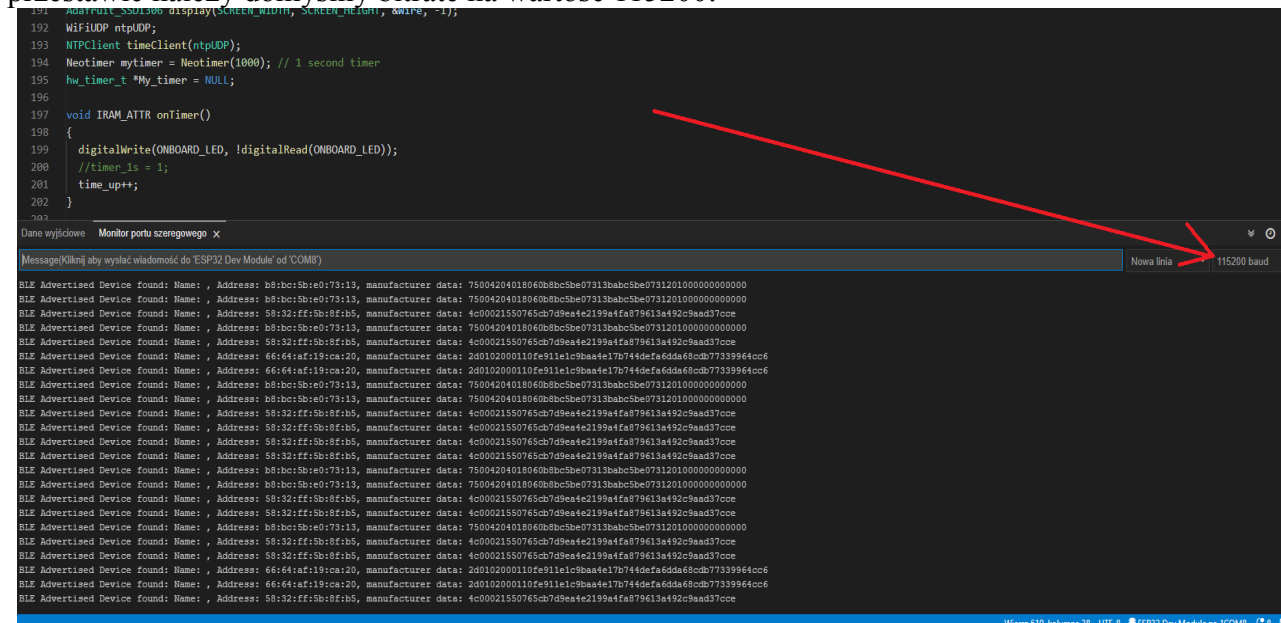


W dolnym oknie "Dane wyjściowe" można zaobserwować proces wgrywania wsadu.

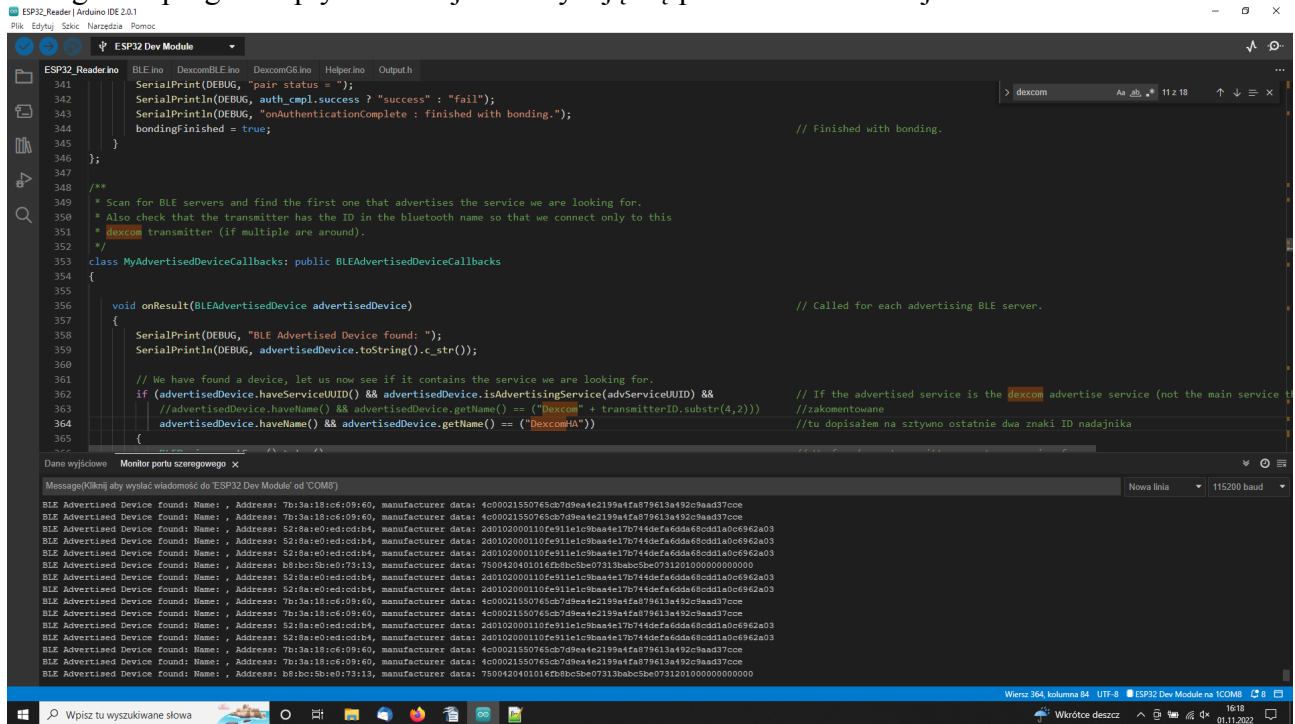
Przydatną funkcją jest też uruchomienie Monitora portu szeregowego, czyli zrobienie nasłuchu wymiany danych po porcie COM:



przestawić należy domyślny bitrate na wartość 115200:



Po wgraniu programu płytką startuje i zaczynają się poszukiwania nadajnika Dexcom G6:



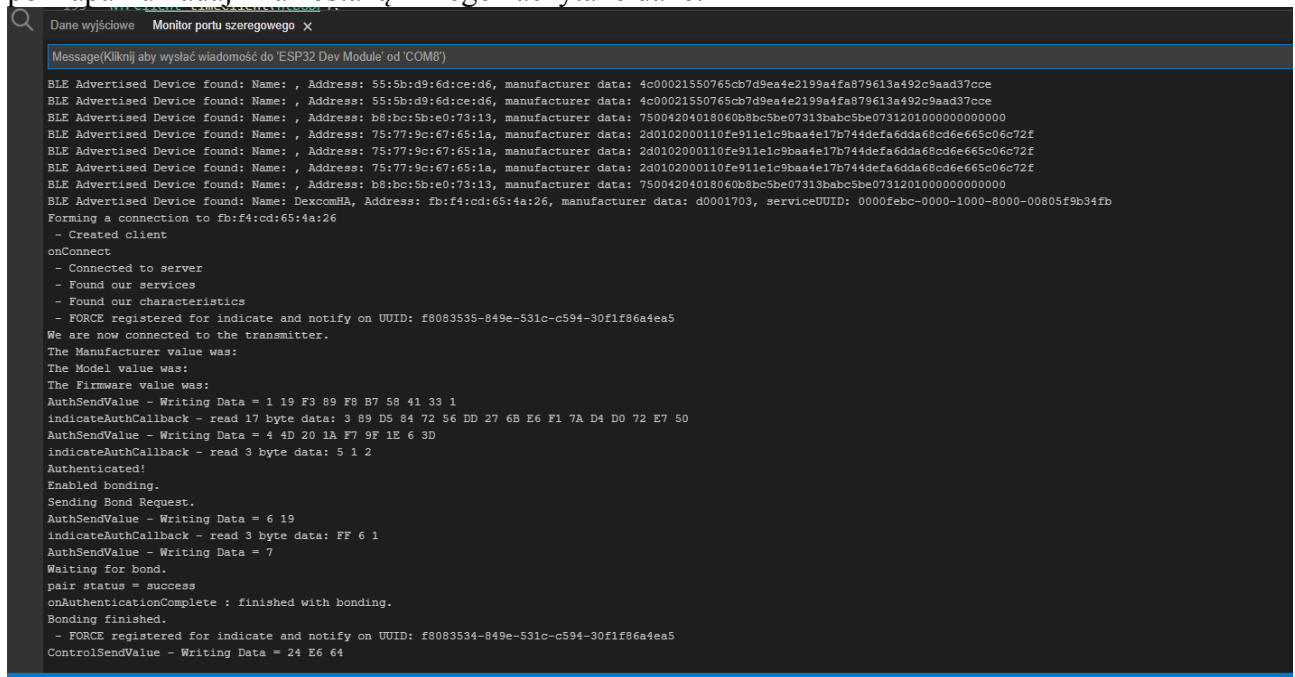
The screenshot shows the ESP32 Reader IDE with the following code:

```
341 SerialPrint(DEBUG, "pair status = ");
342 SerialPrintIn(DEBUG, auth_cmpl.success ? "success" : "fail");
343 SerialPrintIn(DEBUG, "onAuthenticationComplete : finished with bonding.");
344 bondingFinished = true;
345 }
346 };
347
348 /**
349  * Scan for BLE servers and find the first one that advertises the service we are looking for.
350  * Also check that the transmitter has the ID in the bluetooth name so that we connect only to this
351  * * Dexcom transmitter (if multiple are around).
352  */
353 class MyAdvertisedDeviceCallbacks: public BLEAdvertisedDeviceCallbacks
354 {
355
356     void onResult(BLEAdvertisedDevice advertisedDevice) // Called for each advertising BLE server.
357     {
358         SerialPrint(DEBUG, "BLE Advertised Device found: ");
359         SerialPrintIn(DEBUG, advertisedDevice.toString().c_str());
360
361         // We have found a device, let us now see if it contains the service we are looking for.
362         if (advertisedDevice.haveServiceUUID() && advertisedDevice.isAdvertisingService(advServiceUUID) && // If the advertised service is the Dexcom advertise service (not the main service
363             //advertisedDevice.haveName() && advertisedDevice.getName() == ("Dexcom" + transmitterID.substr(4,2))) //zakomentowane
364             advertisedDevice.haveName() && advertisedDevice.getName() == ("DexcomHA")) //tu dopisałem na czystym ostatnie dwa znaki ID nadajnika
365         {
```

The serial monitor shows the following output:

```
Message(Kliknij aby wysłać wiadomość do 'ESP32 Dev Module' od 'COM8')
Nowa linia
115200 baud
BLE Advertised Device Found: Name: , Address: 7b3a18:c6:09:40, manufacturer data: 4c00021550765cb7d9ea4e2199a4fa879613a492c9aad37c0e
BLE Advertised Device Found: Name: , Address: 7b3a18:c6:09:40, manufacturer data: 4c00021550765cb7d9ea4e2199a4fa879613a492c9aad37c0e
BLE Advertised Device Found: Name: , Address: 52:8a:e0:ed:cd:b4, manufacturer data: 2d0102000110fe911e1c9baa4e17b744defa6dda68cd6e665c06c72f
BLE Advertised Device Found: Name: , Address: 52:8a:e0:ed:cd:b4, manufacturer data: 2d0102000110fe911e1c9baa4e17b744defa6dda68cd6e665c06c72f
BLE Advertised Device Found: Name: , Address: 52:8a:e0:ed:cd:b4, manufacturer data: 2d0102000110fe911e1c9baa4e17b744defa6dda68cd6e665c06c72f
BLE Advertised Device Found: Name: , Address: b8:bc:5b:e0:73:13, manufacturer data: 750042040106fb8bc5be07313babcsbe0731201000000000000
BLE Advertised Device Found: Name: , Address: 52:8a:e0:ed:cd:b4, manufacturer data: 2d0102000110fe911e1c9baa4e17b744defa6dda68cd6e665c06c72f
BLE Advertised Device Found: Name: , Address: 52:8a:e0:ed:cd:b4, manufacturer data: 2d0102000110fe911e1c9baa4e17b744defa6dda68cd6e665c06c72f
BLE Advertised Device Found: Name: , Address: 7b3a18:c6:09:40, manufacturer data: 4c00021550765cb7d9ea4e2199a4fa879613a492c9aad37c0e
BLE Advertised Device Found: Name: , Address: 7b3a18:c6:09:40, manufacturer data: 4c00021550765cb7d9ea4e2199a4fa879613a492c9aad37c0e
BLE Advertised Device Found: Name: , Address: 52:8a:e0:ed:cd:b4, manufacturer data: 2d0102000110fe911e1c9baa4e17b744defa6dda68cd6e665c06c72f
BLE Advertised Device Found: Name: , Address: 52:8a:e0:ed:cd:b4, manufacturer data: 2d0102000110fe911e1c9baa4e17b744defa6dda68cd6e665c06c72f
BLE Advertised Device Found: Name: , Address: 7b3a18:c6:09:40, manufacturer data: 4c00021550765cb7d9ea4e2199a4fa879613a492c9aad37c0e
BLE Advertised Device Found: Name: , Address: 7b3a18:c6:09:40, manufacturer data: 4c00021550765cb7d9ea4e2199a4fa879613a492c9aad37c0e
BLE Advertised Device Found: Name: , Address: b8:bc:5b:e0:73:13, manufacturer data: 750042040106fb8bc5be07313babcsbe0731201000000000000
```

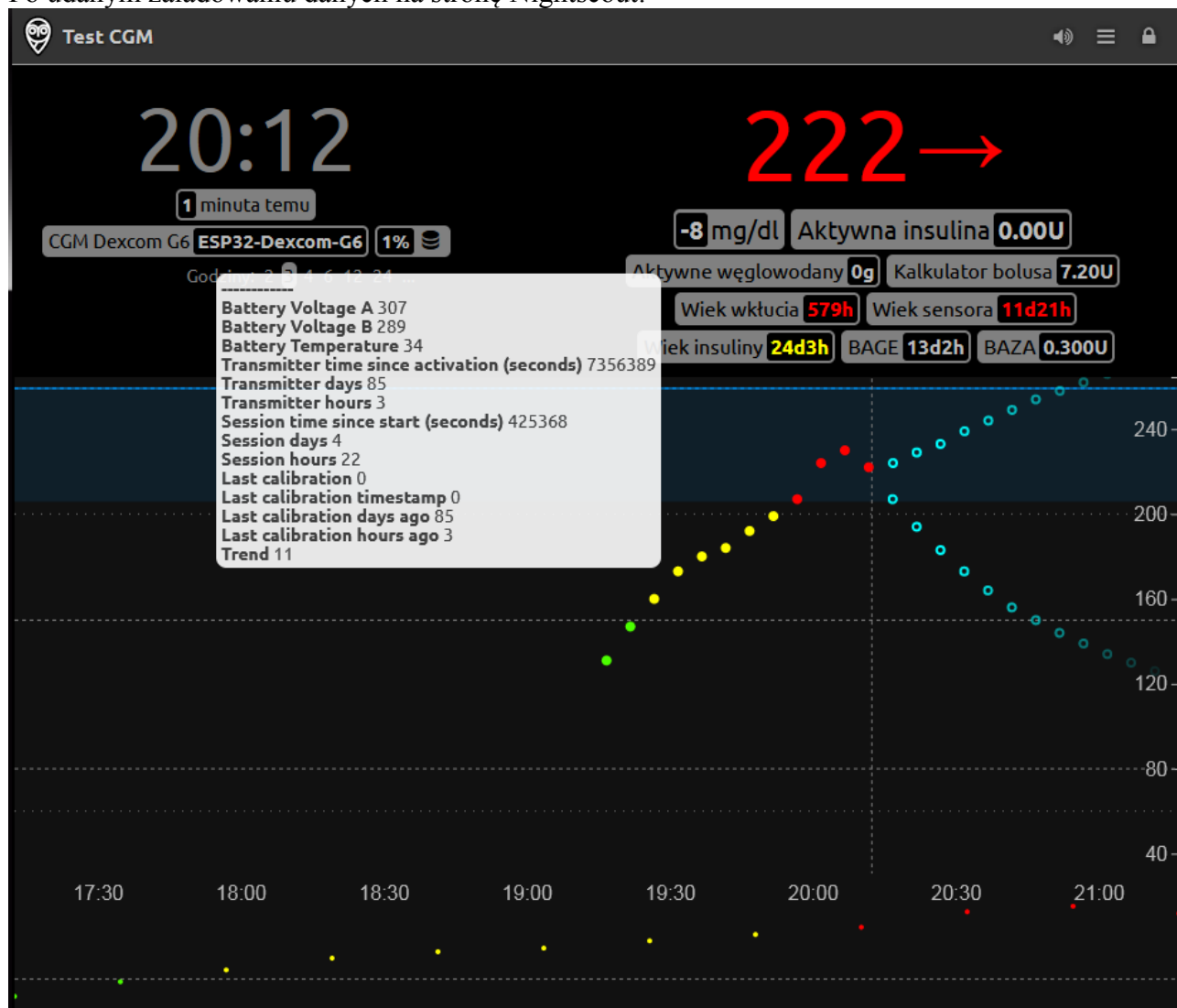
po złapaniu nadajnika zostaną z niego zacytowane dane:



The serial monitor shows the following output:

```
Message(Kliknij aby wysłać wiadomość do 'ESP32 Dev Module' od 'COM8')
BLE Advertised Device Found: Name: , Address: 55:5b:d9:6d:ce:d6, manufacturer data: 4c00021550765cb7d9ea4e2199a4fa879613a492c9aad37c0e
BLE Advertised Device Found: Name: , Address: 55:5b:d9:6d:ce:d6, manufacturer data: 4c00021550765cb7d9ea4e2199a4fa879613a492c9aad37c0e
BLE Advertised Device Found: Name: , Address: b8:bc:5b:e0:73:13, manufacturer data: 750042040106fb8bc5be07313babcsbe0731201000000000000
BLE Advertised Device Found: Name: , Address: 75:77:9c:67:65:1a, manufacturer data: 2d0102000110fe911e1c9baa4e17b744defa6dda68cd6e665c06c72f
BLE Advertised Device Found: Name: , Address: 75:77:9c:67:65:1a, manufacturer data: 2d0102000110fe911e1c9baa4e17b744defa6dda68cd6e665c06c72f
BLE Advertised Device Found: Name: , Address: 75:77:9c:67:65:1a, manufacturer data: 2d0102000110fe911e1c9baa4e17b744defa6dda68cd6e665c06c72f
BLE Advertised Device Found: Name: , Address: b8:bc:5b:e0:73:13, manufacturer data: 750042040106fb8bc5be07313babcsbe0731201000000000000
BLE Advertised Device Found: Name: DexcomHA, Address: fb:f4:cd:65:4a:26, manufacturer data: d0001703, serviceUUID: 0000febc-0000-1000-8000-00805fb34fb
Forming a connection to fb:f4:cd:65:4a:26
- Created client
onConnect
- Connected to server
- Found our services
- Found our characteristics
- FORCE registered for indicate and notify on UUID: f8083535-849e-531c-c594-30f1f86a4ea5
We are now connected to the transmitter.
The Manufacturer value was:
The Model value was:
The Firmware value was:
AuthSendValue - Writing Data = 1 19 F3 89 F8 B7 58 41 33 1
indicateAuthCallback - read 17 byte data: 3 89 D5 84 72 56 DD 27 6B E6 F1 7A D4 D0 72 E7 50
AuthSendValue - Writing Data = 4 4D 20 1A F7 9F 1E 6 3D
indicateAuthCallback - read 3 byte data: 5 1 2
Authenticated!
Enabled bonding.
Sending Bond Request.
AuthSendValue - Writing Data = 6 19
indicateAuthCallback - read 3 byte data: FF 6 1
AuthSendValue - Writing Data = 7
Waiting for bond.
pair status = success
onAuthenticationComplete : finished with bonding.
Bonding finished.
- FORCE registered for indicate and notify on UUID: f8083534-849e-531c-c594-30f1f86a4ea5
ControlSendValue - Writing Data = 24 E6 64
```

Po udanym załadowaniu danych na stronę Nightscout:



opracowanie: Sławomir Malinowski v.0.1