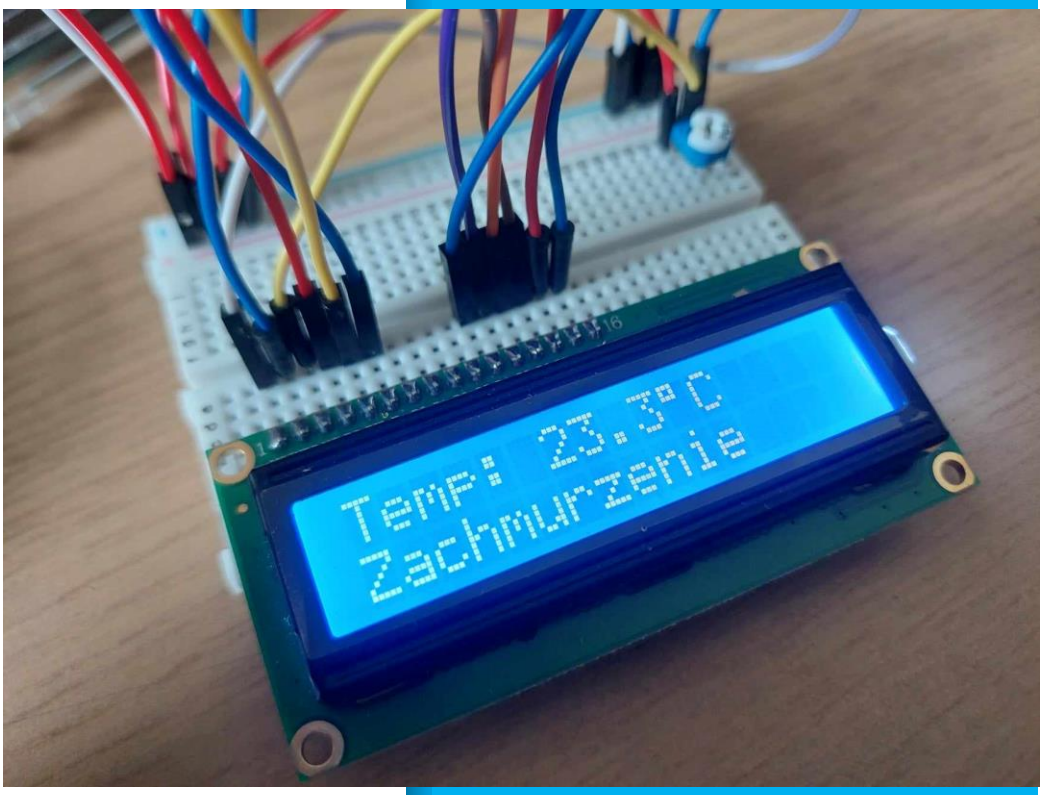
Projekt Arduino

Pogoda na wyświetlaczu LCD



github.com/lbasista

Spis treści

Wstęp

W tym projekcie utworzymy program, który będzie pobierał informacje o bieżącej pogodzie oraz o prognozie na kolejny dzień. Jest to wersja dla Arduino bez modułu umożliwiającego bezpośrednie połączenie internetowe, więc do działania potrzebne jest połączenie z komputerem za pomocą kabla USB. To właśnie program napisany w Pythonie i uruchamiany na komputerze będzie odpowiadał za dostarczanie danych do płytki Arduino.

Rzeczy potrzebne do wykonania projektu:

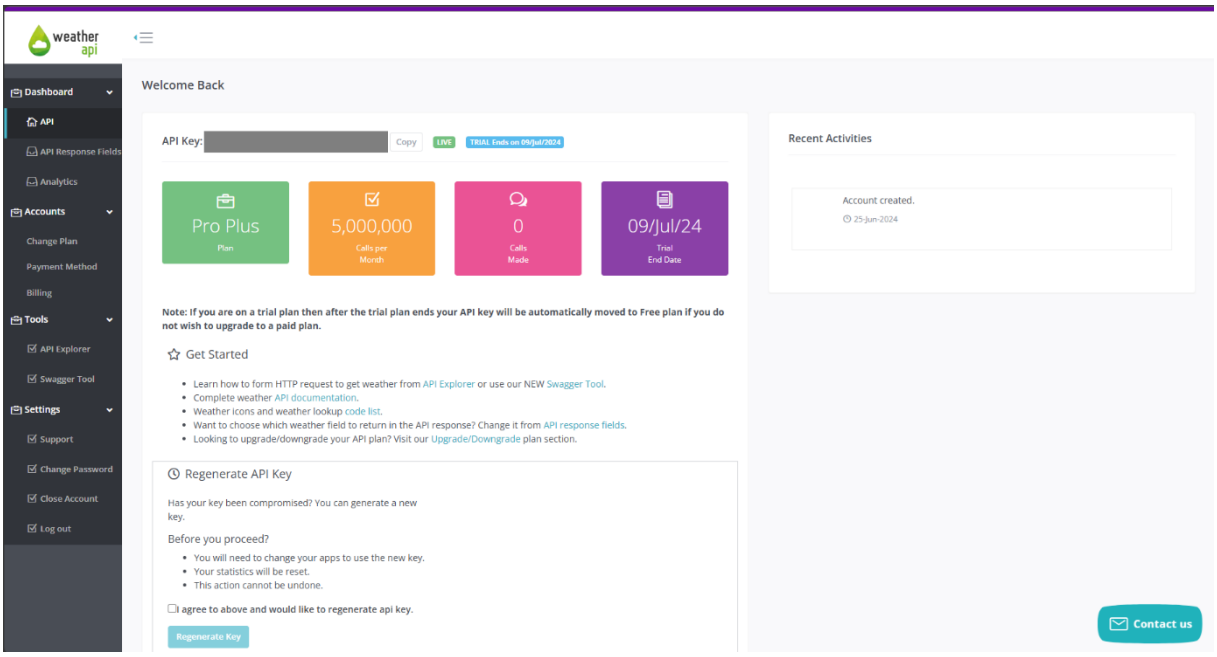
- Płytka Arduino Uno,
- Płytka stykowa,
- Wyświetlacz LCD,
- Potencjometr,
- Przewody połączeniowe,
- Kabel USB – USB B,
- Komputer z połączeniem Internetowym,
- Środowisko do programowania w języku Python,
- Arduino IDE.

Pobieranie pogody z Internetu

Źródłem danych pogodowych do naszego urządzenia będzie strona weatherapi.com.

Generowanie klucza API

Po rejestracji na stronie i zweryfikowaniu maila logujemy się do serwisu. Przed nami powinien pojawić się panel jak na zdjęciu poniżej:



Obraz 1 - Wygląd panelu weatherapi.com

Nasz klucz API jest widoczny na górze ekranu powyżej kolorowych bloków. Teraz potrzebujemy go skonfigurować.

Konfiguracja klucza

Aby otrzymywać tylko informacje pogodowe które nas interesują, czyli temperaturę oraz warunki pogodowe (na przykład *słonecznie* lub *małe zachmurzenie*). W tym celu w menu widocznym po lewej stronie w zakładce *Dashboard* wybieramy pozycje *API Response Fields*.

Domyślnie wszystkie opcje są zaznaczone. Musimy odznaczyć prawie wszystkie opcje. Jedyne opcje które zostawimy włączone to:

- W sekcji *Current Weather*:
  - temp_c,
  - text,
- W sekcji *forecastDay*:
  - date,
- W sekcji *Day*:
  - avgtemp_c,
  - text

Następnie na samym dole klikamy zielony przycisk *Save*.

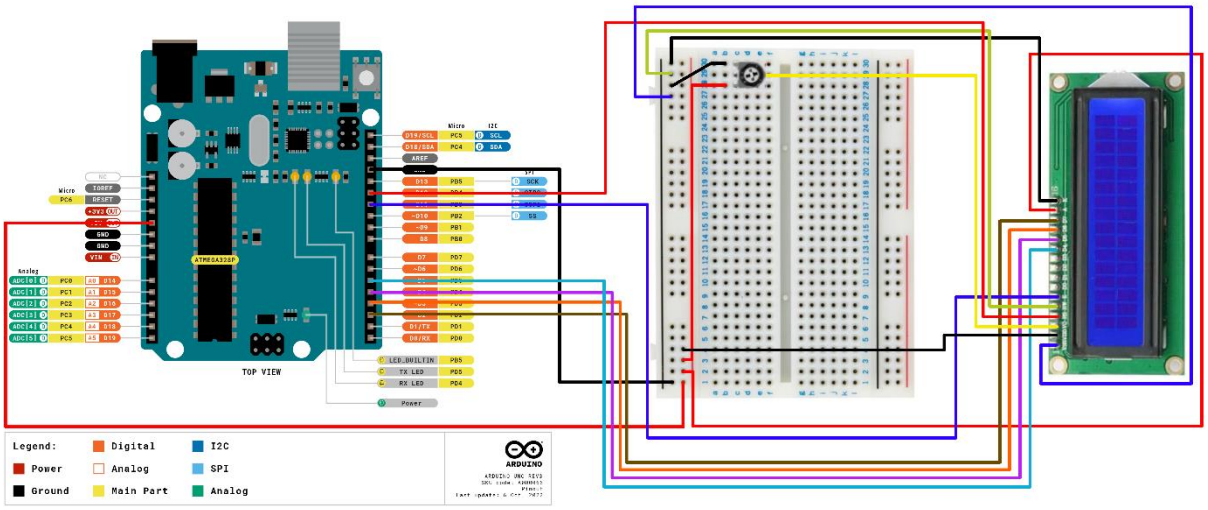
Podłączanie Arduino i wyświetlacza LCD

Aby wszystko działało poprawnie trzeba poprawnie połączyć ze sobą wszystkie komponenty.

W dzisiejszym przykładzie nie użyjemy konwertera I2C dla wyświetlacza LCD, który pozwala zredukować liczbę przewodów potrzebnych do połączenia ekranu z Arduino.

Aby ułatwić sobie łączenie urządzeń ze sobą wykorzystamy płytkę stykową. Na płytce oprócz ekranu i połączeń z Arduino umieścimy również potencjometr. Służy on do regulacji kontrastu na wyświetlaczu.

Poniżej widnieje schemat połączeń między komponentami. Kolory przewodów są wyłącznie symboliczne, by łatwiej było odróżnić je od siebie na schemacie.



Obraz 2 - Schemat połączeń

Schemat w większej rozdzielczości dołączony jest w pliku *Schemat.png*.

Możemy podłączyć teraz płytkę Arduino do komputera za pomocą przewodu USB – USB B. Po zrobieniu tego wyświetlacz powinien się zaświecić.

Programowanie

[Python] Pobieranie danych pogodowych

Pierwszym programem jaki będzie zrobiony to aplikacja w języku Python który pobierze dane z Internetu za pomocą wygenerowanego wcześniej klucza API. Jeśli potrafisz inny język programowania możesz napisać go samemu. Tutaj jednak skupimy się tylko na wspomnianym wcześniej Pythonie.

Otwieramy nasze IDE i tworzymy nowy projekt. Prawie cały kod programu dostępny jest w pliku *Pogoda.py*, wystarczy tylko wpisać swój kod API oraz ID miasta.

Przedstawiona kolejność omawianych części kody nie odzwierciedla faktycznej kolejności w pliku z kodem.

Importujemy biblioteki do łączności Internetowej (*request*), obsługi portów szeregowych (*serial*) oraz czasu (*time*).

```
import requests
import serial
import time
```

Następnie deklarujemy 2 zmienne:

```
api_key = ""
city = ""
```

Jako wartość *api_key* wpisujemy wygenerowany wcześniej klucz API. Zmienna *city* ma przyjąć wartość ID miasta które nas interesuje. Najprawdopodobniej będzie to nazwa w języku angielskim na przykład jeżeli chcemy uzyskać informacje o Krakowie musimy wpisać *Cracow*.

Następnie deklarujemy kolejną zmienną zawierającą adres URL który będzie wywoływał dane pogodowe. Nic tutaj nie zmieniamy, program sam wprowadzi tutaj wartości zmiennych które stworzyliśmy w poprzednim kroku.

```
url = f"http://api.weatherapi.com/v1/forecast.json?key={api_key}&q={city}&days=1&aqi=no&alerts=no"
```

W kolejnym kroku w programie sprawdzamy czy port szeregowy do połączenia z Arduino jest dostępny. W moim przypadku jest to *COM6*. Musisz sprawdzić jaki będzie w Twoim przypadku.

Gdy program wykryje że nie można otworzyć portu wyświetli odpowiedni komunikat.

```
try:
    ser = serial.Serial('COM6', 9600) #Ustaw port COM dla Arduino
    print("Port szeregowy otwarty")
    time.sleep(2)

except Exception as e:
    print(f"Nie można otworzyć portu szeregowego: {e}")
```

Gdy port będzie dostępny wyświetlony zostanie komunikat o powodzeniu. Następnie zapisujemy pobrane dane pogodowe w formacie *json*.

```
while True:

      try:

            print("Pobieranie danych pogodowych...")

            pogoda = (requests.get(url)).json()
```

Następnie po sprawdzeniu czy odpowiedź ze strony nie jest pusta przypisujemy odpowiednim zmiennym wartości:

```
if pogoda.get('current'):

      #Temperatura

      temp = pogoda['current']['temp_c']

      temp_jutro = pogoda['forecast']['forecastday'][0]['day']['avgtemp_c']

      #Opis

      opis = pogoda['current']['condition']['text']

      opis_jutro = pogoda['forecast']['forecastday'][0]['day']['condition']['text']
```

W chwili obecnej otrzymujemy temperaturę z dwoma miejscami po przecinku. Jeśli chcemy zredukować tą ilość dopiszmy te dwie linijki:

```
temp = round(temp, 1)

temp_jutro = round(temp_jutro, 1)
```

Teraz zostaje nam jeden problem – komunikaty są w języku angielskim oraz ich długość często jest większa niż 16 znaków, czyli tyle co w przykładowym wyświetlaczu LCD 16x2. Rozwiążemy to tworząc słownik, który nazwiemy *tlumacz*. Jeśli nie chcesz tego robić możesz pominąć ten krok.

Wywołanie:

```
opis = tlumacz.get(opis, opis)

opis_jutro = tlumacz.get(opis_jutro, opis_jutro)
```

Słownik, który deklarujemy zaraz po importach na początku pliku:

```
tlumacz = {
    'Sunny': 'Slonecznie',
    'Clear': 'Bezchmurne niebo',
    'Partly cloudy': 'M. zachmurzenie',
    'Cloudy': 'Zachmurzenie',
    'Overcast': 'Pochmurnie',
    'Mist': 'Mgla',
    'Patchy rain nearby': 'Miejscowe opady',
    'Patchy snow nearby': 'M. opady sniegu',
    'Patchy sleet nearby': 'M. opady sniegu',
    'Patchy freezing drizzle nearby': 'Miejscowa mzawka',
    'Thundery outbreaks in nearby' : 'Gwaltowna burza',
    'Blowing snow' : 'Wiatr i snieg',
    'Blizzard' : 'Sniezyca',
    'Fog' : 'Mgla',
    'Freezing fog' : 'Mrozna mgla',
    'Patchy light drizzle' : 'Lekka mzawka',
    'Light drizzle' : 'Lekka mzawka',
    'Freezing drizzle' : 'Mrozna mzawka',
    'Heavy freezing drizzle' : 'C. mrozna mzawka',
    'Patchy light rain' : 'Miejscowe opady',
    'Light rain' : 'Lekki deszcz',
    'Moderate rain at times' : 'Srednie opady',
    'Moderate rain' : 'Srednie opady',
    'Heavy rain at times' : 'Duze opady',
    'Heavy rain' : 'Duze opady',
    'Light freezing rain' : 'L. mrozny deszcz',
    'Moderate or heavy freezing rain' : 'Mrozny deszcz',
    'Light sleet' : 'Snieg z deszczem',
    'Moderate or heavy sleet' : 'Snieg z deszczem',
    'Patchy light snow' : 'M. lekki snieg',
    'Light snow' : 'Lekki snieg',
    'Patchy moderate snow' : 'M. opady sniegu',
    'Moderate snow' : 'Opady sniegu',
    'Patchy heavy snow' : 'M. mocny snieg',
    'Heavy snow' : 'M. opady sniegu',
    'Ice pellets' : 'Mokry snieg',
    'Light rain shower' : 'Przelotny deszcz',
    'Moderate or heavy rain shower' : 'Przelotna ulewa',
    'Torrential rain shower' : 'P. nawalnica',
    'Light sleet showers' : 'P.deszcz i snieg',
    'Moderate or heavy sleet showers' : 'P.deszcz i snieg',
    'Light snow showers' : 'L. opady sniegu',
    'Moderate or heavy snow showers' : 'S. opady sniegu',
    'Light showers of ice pellets' : 'L. mokry snieg',
    'Patchy light rain in area with thunder' : 'S. mokry snieg',
    'Moderate or heavy rain in area with thunder' : 'Mala burza',
    'Patchy light snow in area with thunder' : 'M. burza i snieg',
    'Moderate or heavy snow in area with thunder' : 'D. burza i snieg'
}
```

Gdy już mamy dane w odpowiednim dla nas formacie to wysyłamy je przez port USB:

```
data_to_send = f"{temp} {opis}\n{temp_jutro} {opis_jutro};"
print(f"Dane do wysłania: {data_to_send}")


ser.write(data_to_send.encode())
```

Gdy już część główną mamy napisaną to dopisujemy częstotliwość pobierania danych (w tym przypadku 600 sekund, czyli 10 minut) oraz opcję jak wystąpi błąd w trakcie działania programu.

```
        time.sleep(600) #Odświeżanie co 10min
except Exception as e:
        print(f"Błąd: {e}")
        time.sleep(60) #Jeśli błąd odczekaj 1min
```

[C] Programowanie płytki Arduino

Gdy już wysyłamy za pomocą USB dane o pogodzie musimy je odebrać za pomocą Arduino i wyświetlić je na wyświetlaczu. W tym celu w Arduino IDE piszemy kod. Jeśli nie masz ochoty przepisywać kodu, jego całość jest dostępna w pliku *Pogoda.ino*.

Na początku zaimportujmy bibliotekę *LiquidCrystal*, która jest odpowiedzialna za obsługę wyświetlacza LCD. Następnie zadeklarujmy do których pinów na płytce Arduino podpięty jest ekran. W naszym przypadku są to 12, 11, 5, 4, 3 i 2.

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
```

Deklarujemy zmienne, które będą przechowywać odebrane z napisanego rozdział wcześniej programu.

```
String temp = "";
String opis = "";
String temp_jutro = "";
String opis_jutro = "";
```

Jeśli chcemy by ekran odświeżał się częściej niż tylko co 10 minut zadeklarowanych w aplikacji napisanej w Pythonie, to musimy chcianą częstotliwość zadeklarować w tym miejscu. Nazwiemy tą zmienną *interwal*.

```
unsigned long o_odswiezenie = 0;
const unsigned long interwal = 10000;
```

W funkcji *setup*, która uruchamia się razem z płytką Arduino określamy prędkość transmisji (w bitach na sekundę) oraz wielkość wyświetlacza LCD (w tym przypadku 16 kolumn i 2 wiersze).

Opcjonalnie można również dodać komunikat o uruchamianiu. Funkcja *setCursor* ustawia kursor w miejscu, które wpiszemy w nawiasie.

```
void setup() {
  Serial.begin(9600);
  lcd.begin(16, 2);
  lcd.setCursor(0, 0);
  lcd.print("Uruchamianie");
  lcd.setCursor(0, 1);
  lcd.print("Prosze czekac");
}
```

Przechodzimy do funkcji *loop*. Najpierw sprawdźmy czy nowe dane przychodzące przez USB są dostępne. Zrobimy to używając funkcji warunkowej *if*.

Jeśli warunek zostanie określony, to trzeba określić znak do jakiego program ma odczytywać odbierane informacje – u nas jest to średnik. Musimy też określić separatory, które oddzielają przychodzące dane. W naszym przypadku temperaturę od opisu pogody oddziela spacja, a dni znak nowej linii (\n).

W kolejnych liniach kodu, stworzonym kilka kroków temu zmiennym, przypisujemy odpowiednie fragmenty pomiędzy separatorami.

```
if(Serial.available()>0)
  {
    //Podział danych na kategorie
    String prognoza = Serial.readStringUntil(';');

    int separatordzis = prognoza.indexOf(' ');
    int nlinia = prognoza.indexOf('\n');
    int separatorjutro = prognoza.indexOf(' ', nlinia + 1);

    temp = prognoza.substring(0, separatordzis);
    opis = prognoza.substring(separatordzis + 1, nlinia);
    temp_jutro = prognoza.substring(nlinia + 1, separatorjutro);
    opis_jutro = prognoza.substring(separatorjutro + 1);
  }
```

Przechodzimy do części odpowiedzialnej za wyświetlanie naszej pogody. Aby dane wyświetlane były cały czas tworzymy zmienną *uruchomiony*, która zawiera to ile czasu uruchomiony jest ekran.

Następnie w warunku *if* sprawdzamy czy czas od ostatniego odświeżenia ekranu nie przekracza wartości ustalonego wcześniej interwału.

Gdy warunek jest spełniony musimy przypisać zmiennej odpowiedzialnej za przetrzymywanie informacji kiedy nastąpiło ostatnie odświeżenie wyświetlacza.

```
unsigned long uruchomiony = millis();
if (uruchomiony - o_odswiezenie >= interwal)
{
  o_odswiezenie = uruchomiony;
}
```

W dalszej części spełnionej instrukcji warunkowej przechodzimy do funkcji które już bezpośrednio pokażą na wyświetlaczu tekst.

Zaczynamy od wyczyszczenia ekranu i ustawieniu kursora na początku pierwszej linii. Wtedy używając funkcji *print* wyświetlamy dzisiejszą temperaturę. Żeby wyświetlić znak stopni w nawiasach musimy dopisać *(char)223*. Przechodzimy do linii niżej i drukujemy opis pogody. Następnie w funkcji *delay* określamy w milisekundach jak długo dzisiejsza pogoda ma być wyświetlana.

Powtarzamy to samo z prognozą na kolejny dzień. Dla rozróżnienia jaka pogoda się wyświetla przed jutrzejszą temperaturą dodałem znak >.

```
  //Dzisiaj
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Temp: " + temp + (char)223 + "C");
  lcd.setCursor(0,1);
  lcd.print(opis);
  delay(7500);
  //Jutro
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print(">Temp: " + temp_jutro + (char)223 + "C");
  lcd.setCursor(0,1);
  lcd.print(opis_jutro);
  delay(7500);
```

Uruchamianie

Gdy już oba programy mamy napisane oraz odpowiednio podłączone Arduino z komputerem i wyświetlaczem możemy przejść do uruchomienia programu. Najpierw wgrywamy plik *Pogoda.ino* do płytki. Na ekranie powinien pojawić się już napis informujący o uruchamianiu (jeśli go dodaliśmy w funkcji *setup*). Następnie uruchamiamy program pobierający dane pogodowe z Internetu. Po chwili na wyświetlaczu zobaczymy naszą pogodę.

Porada:
Możesz ustawić automatyczne uruchamianie programu wraz z włączaniem komputera. Sposób zrobienia tego różni się w zależności jakiego systemu operacyjnego używasz. W przypadku Windowsa robi się to w aplikacji *Task Scheduler*, a odpowiednikiem na systemie Linux będzie *Cron*.