# PROJEKT WSYW Warszawa 17.12.2022 r.

# Informacje o wykonawcach

|  |  |  |
| --- | --- | --- |
| **Imię** | **Nazwisko** | **Numer albumu** |
| Konrad | Krupski | 310 729 |
| Julia | Polak | 310 965 |

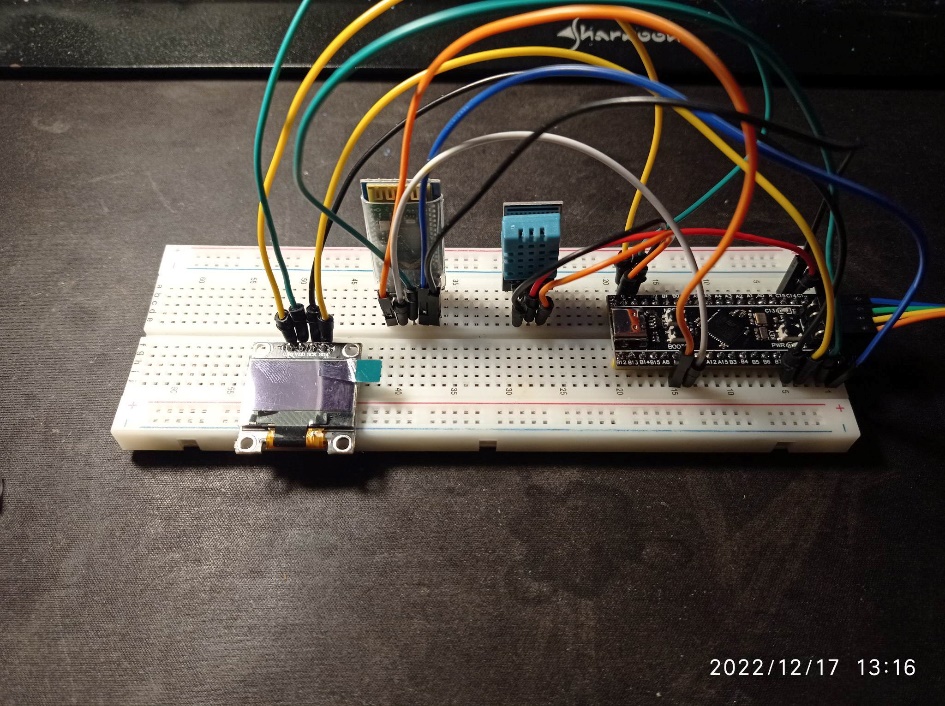
# Opis projektu

Celem projektu jest zbudowanie układu służącego do pomiaru temperatury i wilgotności otoczenia. Układ będzie działał w temperaturach otoczenia od 0°C do 50 °C oraz dla wilgotności od 20% do 90%. Dodatkową funkcjonalnością systemu będzie możliwość zdalnego połączenia się z urządzeniem mobilnym np. telefonem poprzez interfejs bluetooth. Przy użyciu aplikacji na system android będziemy w stanie kontrolować działanie czujnika poprzez wysyłanie odpowiednich komend. W celach jest zrealizowanie komend do wyłączenia i włączenia czujnika oraz do wysyłania informacji, które dane urządzenie będzie pobierało.

# Spis użytych urządzeń

|  |  |  |  |
| --- | --- | --- | --- |
| **Lp.** | **Nazwa urządzenia** | **Zasilanie** | **Dokumentacja** |
| 1 | Płytka Blackpill z procesorem STM32F401CCU6 | 3.3V | [LINK](https://pdf1.alldatasheet.com/datasheet-pdf/view/882025/STMICROELECTRONICS/STM32F401CCU6.html) |
| 2 | Czujnik DHT11 | 3.3V | [LINK](https://pdf1.alldatasheet.com/datasheet-pdf/view/1440068/ETC/DHT11.html) |
| 3 | Programator/debugger do modułów z procesorami | - | - |
| 4 | Płytka stykowa z modułem zasilającym i przewodami połączeniowymi | - | - |
| 5 | Moduł Bluetooth HC-05 | 3.3V | [LINK](https://components101.com/sites/default/files/component_datasheet/HC-05%20Datasheet.pdf) |
| 6 | Wyświetlacz OLED SSD1306 | 3.3V | [LINK](https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf) |

# Zbudowany układ



Rysunek Zbudowany układ

Czujnik DH11 został podłączony do napięcia 3.3V oraz jego wyjście zostało podłączone do pinu C13. Pin ten jest połączony z diodom płytki, dlatego łatwo będzie można zobaczyć kiedy czujnik pobiera i przetwarza informację o temperaturze i wilgotności powietrza.

Moduł Bluetooth HC-05 został podłączony do napięcie 3.3V oraz jego porty zostały podłączone w następujący sposób:

* RX -> A9
* TX -> A10
* EN -> C14
* STATE – port ten nie został podłączony, ponieważ w tym projekcie nie była używana jego funkcjonalność

Ekran OLED SSD1306 został podłączony do napięcia 3.3 V oraz jego porty zostały podłączone w następujący sposób:

* SCK -> B8
* SDA -> B9

Do programowania płytki został użyty programator ST-LINK V2

# Konfiguracja CUBE-IDE

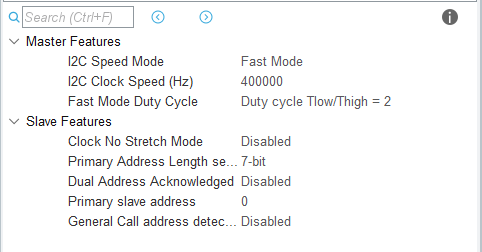
Na początku utworzono projekt postępując według standardowej procedury. Wybrano odpowiedni procesor oraz zdecydowano się na niewprowadzanie domyślnej konfiguracji płytki zadanej przez producenta.

## Konfiguracja I2C

1. Uruchamiamy interfejs I2C1 korzystając z zakładki Connectivity

2. Ustawiamy parametr Speed Mode -> Fast Mode

3. Upewniamy się, że częstotliwość zegara wynosi 400 kHz



Rysunek Parametry I2C

## Konfiguracja USART

1. Uruchamiamy interfejs USART1 korzystając z zakładki Connectivity

2. Tryb Asynchronus

3. W ustawieniach parametrów zmieniamy Baud Rate na 9600 Bits/s

Graphical user interface, text, application

Description automatically generated

Rysunek Parametry UASRT

4. Uruchamiamy przerwania typu global



Rysunek Uruchomienie przerwań UASRT

## Konfiguracja Timer’a

1. Uruchamiamy TIM2 korzystając z zakładki Timers

2. Wybieramy Clock Source -> Internal Clock, czyli wyzwalanie zegarem wewnętrznym

Graphical user interface, text, application, email

Description automatically generated

Rysunek Ustawienia TIM2

3. Ustawiamy częstotliwość zegara na 1 MHz poprzez ustawienie parametru Prescaler -> 15

Graphical user interface, text, application

Description automatically generated

Rysunek Parametry TIM2

## Przypisanie pinów

A picture containing text, electronics

Description automatically generated

Rysunek Schemat płytki

# Zawarcie plików projektowych

W projekcie zostały użyte zewnętrzne bibliotek do obsługi czujnika oraz wyświetlacza OLED. W tej części sprawozdania opiszę jak odpowiednio je załączyć.

## Czujnik DHT11

1. Klonujemy repozytorium

<https://github.com/mesutkilic/DHT11-STM32-Library.git>

2. Załączamy pliki mk\_dht11.c oraz mk\_dht11.h do naszego projektu odpowiednio do folderu Src i Inc

## OLED SSD1306

1. Klonujemy repozytorium

<https://github.com/afiskon/stm32-ssd1306.git>

2. Dla tego przypadku posłużę się funkcja załączenia plików konfiguracyjnych poprzez środowisko CUBE-IDE.

3. Uruchamiamy właściwości projektu

4. C/C++ Build -> Settings -> MCU GCC Compiler -> Include Paths -> Dodajemy folder OLED

Graphical user interface, text, application

Description automatically generated

5. Konfigurujemy plik ssd1306\_conf\_temp.h według tego co jest napisane na stronie github

# Opis kodu

Text

Description automatically generated

Rysunek Załączenie plików

Na początek załączone są wszystkie pliki potrzebne do obsługi peryferiów. Pliki xxxx\_bit.h zawierają bitmapy, które będą wyświetlane na ekranie wyświetlacza.

Text

Description automatically generated

Rysunek Definicja portów

Zdefiniowany również port, do którego dołączony jest czujnik DHT11.

Text

Description automatically generated

Rysunek Zmienne

W tym miejscu mamy zmienne służące do obsługi peryfeirów. Tablice text\_temp i text\_hum przechowując znaki, które są wysyłane przez interfejs UART do urządzenia mobilnego np. telefonu. Zmienna dht jest zdefiniowana w celu obsługi czujnika według wymagań zewnętrznej biblioteki. Zmienne temp i hum przechowują wartości temperatury i wilgotności w formacie bitowym.

Zmienna ParseBuffer przechowuje wartość bitową, która jest przesyłana z urządzenia mobilnego do modułu bluetooth prze interfejs UART. Tablica znaków data przechowuje wartości przesyłane z odułu bluetooth do urządzenia mobilnego w celi ich wyświetlenia np. temperaturę czy wilgotność. Zmienna flaga jest zmienną pomocniczą służącą do włączenia i wyłączenia czujnika.

Text

Description automatically generated

Rysunek Inicjalizacja

Teraz inicjalizowane są peryferia, na początek czujnik a potem moduł bluetooth. Pin C14 jest podłączony do pinu ENABLE modułu bluetooth, dlatego należy ustawić go w stan wysoki, aby uruchomić urządzenie.

Text

Description automatically generated

Rysunek Nieskończona pętla programu

Uruchamiamy timer, który definicuje co, ile czujnik pobiera dane oraz następnie włączamy moduł bluetooth w stan gotowości w celu odebrania nadchodzących danych, ostatecznie uruchamiamy czujnik dht11 za pomocą odpowiedniej funkcji.

W linii 132 widzimy zastosowanie flagi do obsługi czujnika. Flaga domyślnie przyjmuje wartość 1 i pozwala to na pobranie odczytu z czujnika oraz zapisanie go do odpowiednich zmiennych. W przeciwnym wypadku sytuacja ta nie ma miejsca.

Text

Description automatically generated

Rysunek Funkcja printScreen()

Funkcja służy do wyświetlania na ekranie OLED informacji odczytanych z czujnika w zależności od flagi. Jeżeli flaga przyjmuje wartość 1 to na ekranie wyświetlamy informacje, jeżeli to to wszystkie piksele ekranu ustawiane są na kolor czarny. Z powodu tego, że jest to ekran typu OLED to czarny kolor piksela jest równoważny jego wyłączeniu, dlatego pobór mocy jest znikomy.

Text

Description automatically generated

Rysunek Callback do obsługi modułu Bluetooth

W odpowiednim Callback’u zdefiniowano obsługę komunikacji moduł bluetooth -> urządzenie mobilne. Na początek sprawdzamy z jakiego interfejsu UART przychodzi wywołanie funkcji, dalej w zależności od odebranego znaku wysyłana jest odpowiednia informacja. Ten callback jest wywoływany, gdy moduł bluetooth skończy odbierać informacje z urządzenia mobilnego.

# Działanie układu

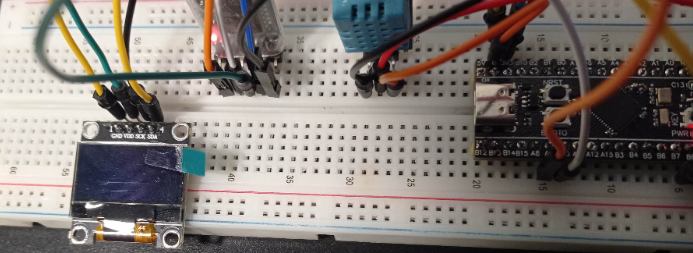
A close-up of a circuit board

Description automatically generated with low confidence

Rysunek Uruchomiony układ



Rysunek Wysyłamy polecenie do wyłączenia ekranu



Rysunek Ekran uległ wyłączeniu

Graphical user interface

Description automatically generated with low confidence

Rysunek Pomiar temperatury



Rysunek Pomiar wilgotności