Urządzenia peryferyjne

Prowadzący zajęcia: dr inż. Jan Nikodem
Termin zajęć: Czwartek TN, godz. 14:10

Osoby wykonujące ćwiczenie: Mateusz Gawłowski, Bartosz Szymański	**Numer grupy:** 2
Tytuł ćwiczenia: System on Chip (SoC) ESP32. Transmisja BLE i akwizycja sygnału. Instalacja FreeRTOS, środowisko Arduino IDE.	**Data wykonania ćwiczenia:** 26.10.2023

Spis treści

1. Cel ćwiczenia

- Na telefon komórkowy, pobrać i uruchomić program obsługi BLE (np. BLE Scanner)
- Podłączyć ESP32 do komputera. Skonfigurować Arduino IDE. Uruchomić program Hello World
- Skonfigurować układ ESP32 do transmisji BLE. Uruchomić przykładowy program do obsługi BLE (klient-serwer, advertising). Zaobserwować transmisje BLE pomiędzy ESP32 a telefonem komórkowym.
- Zainstalować system FreeRTOS. Uruchomić wątki z synchronizacją i komunikacją między nimi. Na jednym z wątków uruchomić transmisję BLE (z punktu 3).
- Odczyt danych z czujnika temperatury przy jednoczesnym wysyłaniu tych danych przez BLE.

2. Wstęp teoretyczny

ESP32 to mikrokontroler, który został opracowany przez firmę Espressif Systems. Jest to układ z rodziny ESP (Espressif System Platform), który integruje mikroprocesor Tensilica LX6, moduł Wi-Fi oraz Bluetooth. ESP32 oferuje szeroki zakres zastosowań, takie jak programowanie IoT, projekty związane z komunikacją bezprzewodową, sterowanie urządzeniami elektronicznymi i wiele innych. Jego elastyczność, niewielkie rozmiary i dostępność interfejsów komunikacyjnych sprawiają, że jest popularnym wyborem w projektach DIY oraz profesjonalnych aplikacjach embedded.

BLE (Bluetooth Low Energy) to bezprzewodowa technologia komunikacyjna, która umożliwia niski pobór energii w urządzeniach komunikujących się ze sobą przez interfejs Bluetooth. Została wprowadzona jako część specyfikacji Bluetooth 4.0 i jest często używana w kontekście IoT (Internet of Things) oraz aplikacji związanych z monitorowaniem zdrowia, śledzeniem fitness, lokalizacją i sterowaniem zdalnym

W sieci Bluetooth Low Energy, węzły komunikują się w paśmie częstotliwości ISM 2,4 GHz, które zostało podzielone na 40 kanałów o szerokości 2 MHz. Wyróżniamy dwa rodzaje kanałów: data (łączność dwukierunkowa) i advertising (3 kanały, używane do wyszukiwania urządzeń BLE, nawiązywania połączenia i transmisji rozgłoszeniowej). Aby zapewnić jakość sygnału, konieczne jest ograniczenie wpływu typowych problemów w łączności bezprzewodowej, takich jak zaniki selektywne i wielodrogowość.

W celu osiągnięcia tego celu, wykorzystuje się technikę przełączania częstotliwości, gdzie 37 kanałów jest używanych naprzemiennie przez określony czas. BLE wykorzystuje modulację GFSK (Gaussian Frequency Shift Keying) do przesyłania informacji.

W przypadku transmisji rozgłoszeniowej, nadawcy korzystają z każdego kanału advertising na przemian, podczas gdy w komunikacji dwukierunkowej konieczne jest nawiązanie połączenia.
Bluetooth Low Energy zużywa znacznie mniej energii, nawet do 20 razy mniej w porównaniu z tradycyjnym Bluetoothem. Wynika to z kilku czynników:

- W sieci Bluetooth LE węzły muszą nasłuchiwać jedynie trzech kanałów advertising, co zajmuje od 0,6 ms do 1,2 ms. W przypadku standardowego Bluetootha konieczne jest nasłuchiwanie ciągle na 32 kanałach, co zazwyczaj zajmuje 22,5 ms. Dzięki temu pobór mocy ogranicza się od 10 do nawet 20 razy.

- Operacje takie jak skanowanie, nawiązywanie połączenia, transmisja danych i potwierdzenia odbioru zajmują w przypadku urządzeń BLE typowo jedynie 3 ms, podczas gdy w "zwykłym" Bluetooth czas ten wynosi setki ms, co wiąże się z wyższym zużyciem energii. Jest to możliwe m.in. dzięki krótszym pakietom danych w Bluetooth Low Energy.

- W sieci BLE węzły komunikują się w paśmie częstotliwości ISM 2,4 GHz, które podzielono na 40 kanałów o szerokości 2 MHz. Wyróżnia się dwa rodzaje kanałów: data i advertising. Aby zapewnić jakość sygnału, konieczne jest ograniczenie typowych problemów w łączności bezprzewodowej, takich jak zaniki selektywne i wielodrogowość.

- Węzły slave w sieci BLE przez większość czasu pozostają w trybie sleep, a tylko okresowo wychodzą z niego, aby odebrać pakiety przesłane przez węzeł master.

Akwizycja sygnału w ESP32. ESP32 posiada wbudowany przetwornik analogowo-cyfrowy(ADC), który został wykorzystany do odczytu temperatury oraz wilgotności z czujnika o nazwie DHT11. Zamiana sygnału analogowego na cyfrowy wyglądała następująco:

- Próbkowanie: Sygnał analogowy jest próbkowany w określonych odstępach czasu, które nazywane są częstotliwością próbkowania. Im częstotliwość próbkowania jest wyższa, tym dokładniejsza reprezentacja sygnału.

- Pomiar napięcia: W momencie próbkowania, przetwornik analogowo-cyfrowy mierzy wartość napięcia na wejściu analogowym. To napięcie jest porównywane z referencyjnym napięciem ustalonym w przetworniku ADC.

- Kwantyzacja: Napięcie próbkowane jest przekształcane na cyfrową liczbę za pomocą procesu zwanego kwantyzacją. Wartość napięcia jest dzielona na dyskretne poziomy, a każdy poziom jest przypisany do konkretnego kodu cyfrowego. Rozdzielczość przetwornika ADC, wyrażona w bitach, określa, ile poziomów jest dostępnych dla reprezentacji sygnału. Im większa rozdzielczość, tym dokładniejsza reprezentacja sygnału, ale wymaga to więcej bitów na każdą próbkę.

- Wyjście cyfrowe: Po dokonaniu kwantyzacji, wynikowy kod cyfrowy jest przesyłany do mikrokontrolera lub mikroprocesora, który może go przetwarzać i wykorzystywać w kodzie programu. W ten sposób sygnał analogowy jest reprezentowany w postaci cyfrowej i może być dalszym używany w obliczeniach, analizie i przechowywaniu

W skrócie sygnał napięcia z czujnika jest próbkowany, przekształcany na cyfrowy kod i następnie można odczytać tę cyfrową wartość w programie mikrokontrolera ESP32, aby odczytać temperaturę.

FreeRTOS (Real-Time Operating System) to otwarte oprogramowanie do zarządzania czasem rzeczywistym, które jest wykorzystywane w wielu aplikacjach, szczególnie w systemach wbudowanych. Jest to niewielki, lekki system operacyjny czasu rzeczywistego, który zapewnia planowanie zadań, zarządzanie wątkami oraz mechanizmy synchronizacji, umożliwiające projektowanie i implementację systemów czasu rzeczywistego.

3. Użyte oprogramowanie

- Serial Bluetooth Terminal
- Adruino IDE

4. Wykonanie zadania

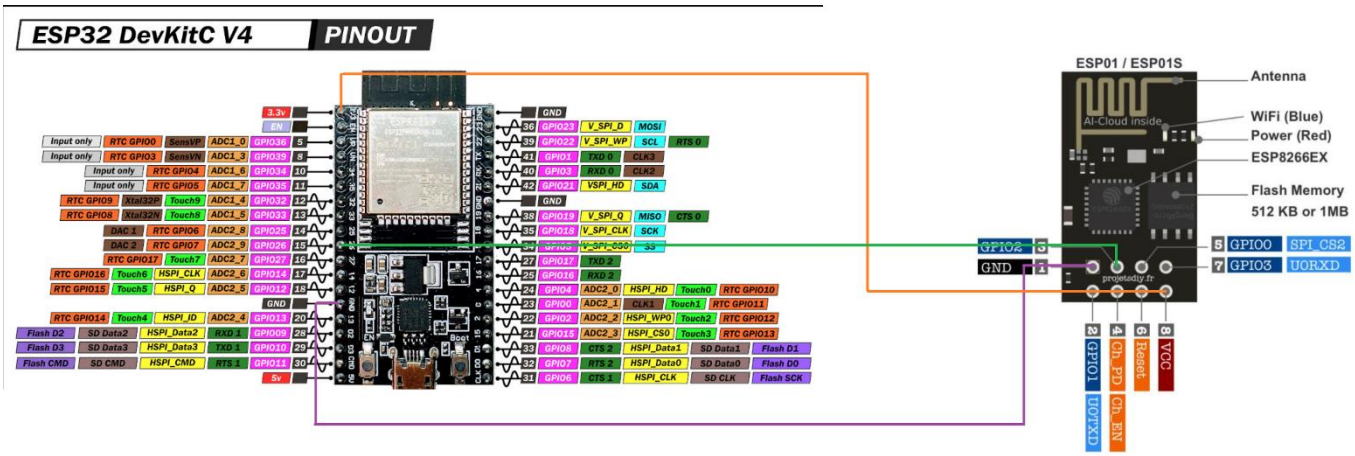
1. Kod programu wraz z krótkim opracowaniem

```
#include <string>
#include <iostream>
#include "DHT.h"
#include "Arduino.h"
#include "BluetoothSerial.h"
String device_name = "Pomiary";
BluetoothSerial SerialBT;
#define DHTPIN 2
#define DHTTYPE DHT11

DHT dht(26, DHTTYPE);
float h, t, f;
void temp(void * parametrs){
  for(;;){
   h = dht.readHumidity();
   t = dht.readTemperature();
  }
```

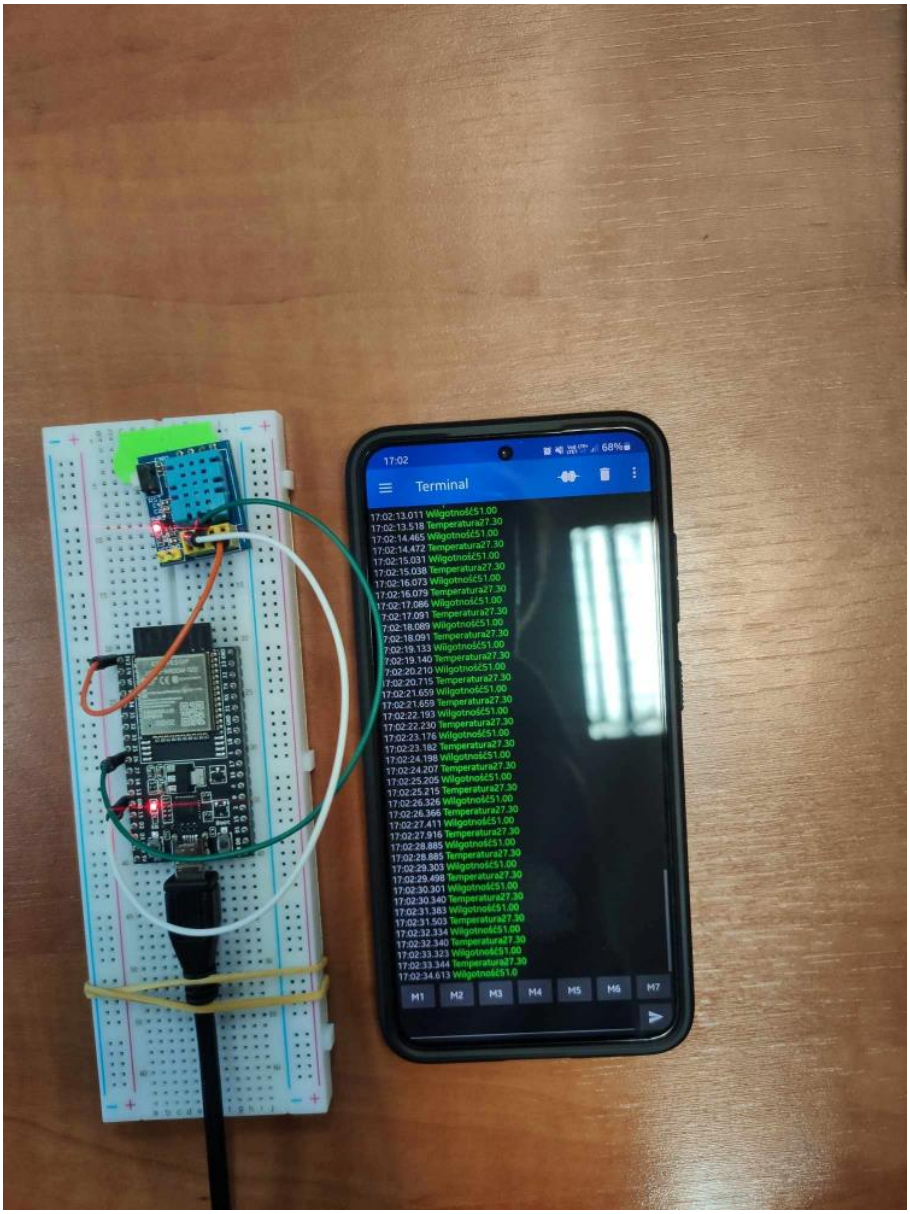
```
}

void blu(void * parametrs){
  for(;;){
    SerialBT.print("Wilgotność ");
    SerialBT.println(h);
    SerialBT.print("Temperatura ");
    SerialBT.println(t);
    vTaskDelay(1000/portTICK_PERIOD_MS);
  }
}
void setup() {
  Serial.begin(115200);
  SerialBT.begin(device_name);
  dht.begin();
     xTaskCreatePinnedToCore(
    temp,
    "Task 1",
    5000,
    NULL,
    1,
    NULL,
    1
  );
     xTaskCreatePinnedToCore(
    blu,
    "Task 2",
    10000,
    NULL,
    1,
    NULL,
    0
  );
}
void loop() {
}
```

Ogólnie rzecz biorąc, program ten inicjalizuje komunikację szeregową, moduł Bluetooth i czujnik DHT11. Następnie tworzy dwie osobne funkcje, które działają w osobnych zadaniach (wątkach). Jedna funkcja (temp) odczytuje temperaturę i wilgotność, a druga funkcja (blu) przesyła te dane przez Bluetooth co sekundę. Cała logika programu jest obsługiwana w zadaniach, dlatego główna pętla (loop) jest pusta.

2. Schemat połączenia czujnika DHT11 oraz ESP32



3. Efekt końcowy



5. Wnioski

Realizacja ćwiczenia w znaczący sposób wpłynęła na nasz poziom rozumienia tematu w obszarze sposobu pracy z ESP32, akwizycją sygnału oraz komunikacji Bluetooth pomiędzy urządzeniami. Tym razem również udało nam się opanować temat FreeRTOS, co sprawiło, że nasze ESP32 było w stanie jednocześnie pobierać dane z czujnika oraz wysyłać ja nam na telefon.

6. Literatura

Poprzez literaturę należy rozumieć wszelkie materiały potrzebne do zrozumienia założeń laboratoriów, sposobu realizacji ćwiczenia oraz źródeł pomocy podczas rozwiązywania problemów napotkanych w trakcie realizacji zajęć.

- Instalacja i konfiguracja środowiska Arduino IDE dla układu ESP32: https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/
- Zasady działania BLE: https://www.bluetooth.com/blog/exploring-bluetooth-5-going-the-distance/
- Wprowadzenie łączności Bluetooth pomiędzy układem ESP32 a telefonem komórkowym https://randomnerdtutorials.com/esp32-bluetooth-classic-arduino-ide/
- Wprowadzenie do FreeRTOS – część 1: https://www.youtube.com/watch?v=F321087yYy4&ab_channel=DigiKey
- Wprowadzenie do FreeRTOS – część 2: https://www.youtube.com/watch?v=JIr7Xm_riRs
- Wprowadzenie do FreeRTOS – część 3: https://www.youtube.com/watch?v=95yUbClyf3E