Design Laboratory Projekt

Raport projektu

System monitorowania temperatury i sterowania wentylatorami

Wykonawcy:

Hubert Durnaś

Dawid Makowski

Hubert Meszko

Prowadzący: **dr inż. Jacek Stępień**

Spis treści:

1. **Temat projektu:**

Fan Controller: System monitorowania temperatury i sterowania wentylatorami.

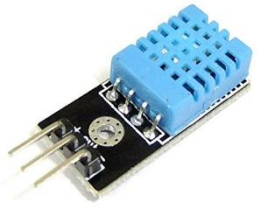
2. **Opis projektu:**

Projekt realizuje system, który mierzy temperaturę otoczenia, wyświetla ją na ekranie LCD i steruje prędkością wentylatorów w zależności od temperatury. Oparty jest na czujniku temperatury DHT11, wyświetlaczu LCD oraz interfejsie RS485. Dodatkowo system przesyła informacje o temperaturze za pomocą protokołu RS485, co umożliwia zdalne monitorowanie.

ANALIZA UKŁADU

1. **Lista komponentów:**
   - **Czujnik DHT11**: Do pomiaru temperatury.

     https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf



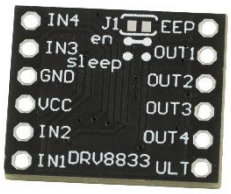
   - **Wyświetlacz LCD z interfejsem I2C**: Do wyświetlania danych o temperaturze.

     Konwerter oparty na układzie PCF8574: https://www.nxp.com/docs/en/data-sheet/PCF8574_PCF8574A.pdf

     https://www.digikey.sg/htmldatasheets/production/2410219/0/0/1/tc1602a-01t.html

   - **Silnik DRV8833.**

     https://www.ti.com/lit/ds/symlink/drv8833.pdf?ts=1734289784009&ref_url=https%253A%252F%252Fwww.google.com%252F



   - **Konwerter UART RS485**: Do przesyłania danych o temperaturze.

https://www.ti.com/lit/ds/symlink/sn65hvd485e.pdf?ts=1732292175646&ref_url=https%253A%252F%252Fwww.google.com%252F

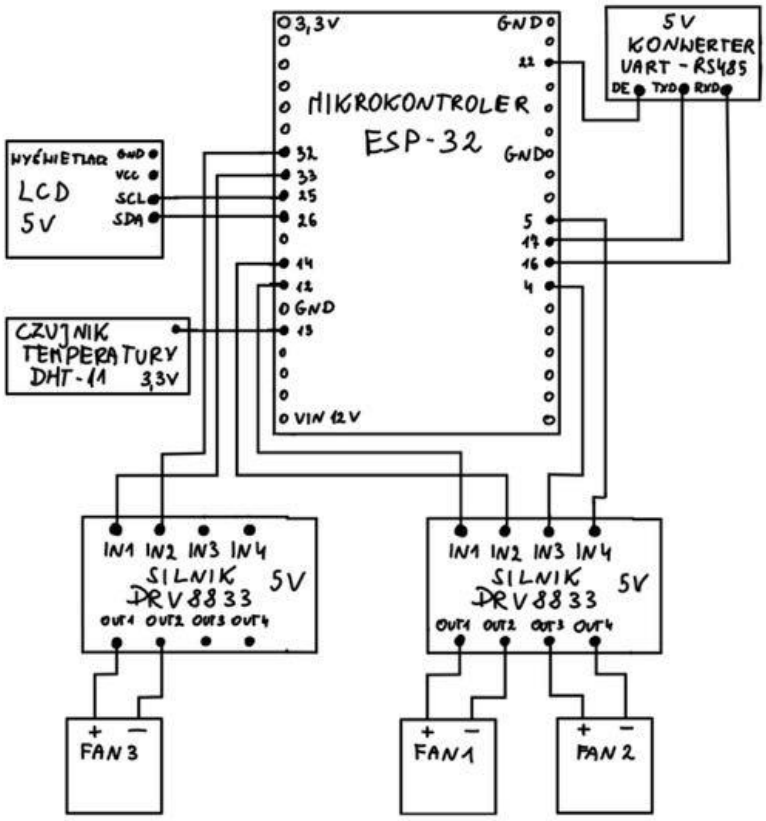
- **Trzy wentylatory 5V**.

https://botland.com.pl/wentylatory-montazowe/9872-wentylator-5v-30x30x10mm-2-przewody-5904422338541.html

- **Płytka ESP32**: Jednostka sterująca.

https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf

2. **Schemat blokowy:**



OPIS WYBRANYCH TECHNOLOGII UŻYTYCH W PROJEKCIE

1. **Komunikacja I2C:**

I2C (Inter-Integrated Circuit) to protokół komunikacyjny, który umożliwia wymianę danych pomiędzy różnymi układami elektronicznymi za pomocą dwóch linii: SDA (dane) i SCL (zegar). Linie te są współdzielone przez urządzenia master i slave. Komunikacja rozpoczyna się, gdy urządzenie master wysyła sygnał startowy, obniżając linię SDA i podnosząc linię SCL. Następnie wysyła osiem impulsów zegarowych, przesuwając bajt z 7-bitowym adresem i bitem odczytu/zapisu. Jeśli urządzenie slave o tym adresie jest aktywne, potwierdza ono odbiór, obniżając linię SDA przy dziewiątym impulsie zegarowym.

Linia SDA zmienia się tylko wtedy, gdy linia SCL jest niska. Po zakończeniu komunikacji, urządzenie master wysyła sygnał stop, podnosząc linię SDA, podczas gdy linia SCL jest już wysoka.

Główne cechy I2C:

- Struktura master-slave: Jeden urządzenie (master) zarządza komunikacją i kontroluje zegar, a pozostałe urządzenia (slave) reagują na polecenia mastera.
- Adresowanie urządzeń: Każde urządzenie slave ma unikalny adres, co umożliwia masterowi komunikację z różnymi urządzeniami na tej samej magistrali.
- Dwukierunkowa komunikacja: Dane mogą być przesyłane w obu kierunkach, zarówno z mastera do slave'a, jak i odwrotnie.
- Protokół wielourządzeniowy: Możliwość podłączenia wielu urządzeń na tej samej magistrali, co czyni go idealnym do integracji różnych sensorów, akcesoriów i innych modułów.

2. UART:

UART (Universal Asynchronous Receiver/Transmitter) to technologia, która umożliwia komunikację między urządzeniami elektronicznymi. Jest szeroko stosowana do komunikacji szeregowej w systemach embedded i komputerach.

Główne cechy UART:

- Dwukierunkowa transmisja danych: UART przesyła dane między dwoma urządzeniami (nadajnik i odbiornik) w obu kierunkach.
- Asynchroniczna komunikacja: Nie wymaga sygnału zegarowego, co oznacza, że urządzenia mogą przesyłać dane niezależnie od siebie.
- Ramy danych: Każdy przesyłany bajt jest opakowany w ramki zawierające bity startu, dane i stopu:
  Bit startu: Sygnał początku transmisji.
  Bity danych: Zazwyczaj od 5 do 9 bitów danych.
  Bit parzystości (opcjonalny): Do wykrywania błędów transmisji.
  Bity stopu: Sygnał końca transmisji.
- Prędkość transmisji (baud rate): Określa liczbę bitów przesyłanych na sekundę.

Zastosowania UART:

- Komunikacja z mikrokontrolerami: Podłączenie do komputerów, modułów GPS, modułów Bluetooth i innych urządzeń peryferyjnych.
- Debugowanie i programowanie: Często używany do debugowania systemów embedded i programowania mikrokontrolerów.
- Połączenia szeregowe: Komunikacja między komputerami i urządzeniami peryferyjnymi.

UART jest często używany w połączeniu z innymi protokołami sieciowymi. Na przykład, dane mogą być przesyłane za pomocą UART do modułu bezprzewodowego, który następnie wykorzystuje protokół Wi-Fi lub Bluetooth do dalszej transmisji danych.

3. Komunikacja RS485:

RS485 to technologia komunikacji szeregowej, która zapewnia niezawodne połączenia na większe odległości i w trudnych warunkach środowiskowych. Jest szczególnie popularna w zastosowaniach przemysłowych, systemach sterowania i automatyce.

Kluczowe cechy RS485:

- Dwukierunkowa komunikacja: RS485 obsługuje komunikację półdupleksową, co oznacza, że dane mogą być przesyłane w obu kierunkach, ale nie jednocześnie.
- Wielopunktowość: Możliwość podłączenia wielu urządzeń do jednej magistrali (maksymalnie 32 urządzenia), co sprawia, że jest idealna do sieciowych systemów sterowania.
- Odporność na zakłócenia: RS485 wykorzystuje różnicowe sygnały napięciowe, co minimalizuje wpływ zakłóceń elektromagnetycznych i umożliwia transmisję danych na odległości do 1200 metrów.

Nasz kod zawiera podstawowe informacje dotyczące komunikacji RS485 przy użyciu portu szeregowego i biblioteki SoftwareSerial.

- Biblioteka SoftwareSerial: Tworzy dodatkowy port szeregowy do obsługi RS485 przy użyciu pinów RX (16) i TX (17).
- Funkcja setup(): Inicjalizuje port szeregowy oraz ustawia pin RS485_DE_PIN do kontrolowania trybu nadawania/odbioru. W przypadku nadawania, pin ten jest ustawiany na wysoki stan logiczny, a w przypadku odbioru na niski.
- Funkcja loop(): Co sekundę wysyła wiadomość tekstową przez RS485, przełączając RS485_DE_PIN między trybem nadawania i odbioru. Wysyłana jest wiadomość "Testowa wiadomość przez RS485".

KOD PROGRAMU

1. **Użyte biblioteki:**

- **Wire.h**: Biblioteka do komunikacji I2C. Pozwala na komunikację z urządzeniami podłączonymi do magistrali I2C.
- **Adafruit_Sensor.h**: Standardowa biblioteka Adafruit dla różnych sensorów, która umożliwia jednolite odczyty danych z różnych typów czujników.
- **DHT.h**: Biblioteka do obsługi czujników wilgotności i temperatury DHT.
- **DHT_U.h**: Rozszerzenie biblioteki DHT, wspiera interfejs Adafruit Unified Sensor.
- **LiquidCrystal_I2C.h**: Biblioteka do obsługi wyświetlaczy LCD podłączonych przez I2C.
- **SoftwareSerial.h**: Biblioteka do tworzenia dodatkowych portów szeregowych na dowolnych pinach mikrokontrolera.

2. **Opis kodu:**

2.1 Zmienne i definicje

- **I2C_SDA i I2C_SCL**: Piny używane do komunikacji I2C.
- **DHT_PIN i DHT_TYPE**: Pin i typ czujnika DHT (DHT11).
- **MOTOR1_IN1, MOTOR1_IN2, MOTOR2_IN1, MOTOR2_IN2, MOTOR3_IN1, MOTOR3_IN2**: Piny sterujące trzema silnikami.
- **RS485_TXD_PIN, RS485_RXD_PIN, RS485_DE_PIN**: Piny używane do komunikacji RS485.
- **rs485**: Tworzy dodatkowy port szeregowy do komunikacji RS485.
- **dht**: Obiekt czujnika DHT.
- **lcd**: Obiekt wyświetlacza LCD.
- **motor3Enabled**: Zmienna do kontrolowania stanu trzeciego silnika (wentylatora).
- **updateTemperature**: Zmienna kontrolująca potrzebę aktualizacji temperatury.
- **timer**: Obiekt timera.

2.2 Funkcja setup()

- Inicjalizacja portu szeregowego i wyświetlenie "System test" na konsoli.

```
Serial.begin(9600);
Serial.println("System test");
```

- Inicjalizacja magistrali I2C.

```
Wire.begin(I2C_SDA, I2C_SCL);
Serial.println("I2C test");
```

- Inicjalizacja czujnika DHT. Uruchamia czujnik DHT na pinie 13.

dht.begin();
Serial.println("Czujnik test");

- Inicjalizacja wyświetlacza LCD oraz wyświetlenie komunikatu "Inicjalizacja...".

  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Inicjalizacja...");
  delay(1000);
  lcd.clear();

- Konfiguracja PWM dla trzech silników za pomocą funkcji ledcSetup i ledcAttachPin. Konfiguruje PWM dla trzech silników na różnych kanałach z częstotliwością 5000 Hz i rozdzielczością 8-bitową.

  ledcSetup(0, 5000, 8);
  ledcAttachPin(MOTOR1_IN1, 0);
  ledcAttachPin(MOTOR1_IN2, 1);

  ledcSetup(2, 5000, 8);
  ledcAttachPin(MOTOR2_IN1, 2);
  ledcAttachPin(MOTOR2_IN2, 3);

  ledcSetup(4, 5000, 8);
  ledcAttachPin(MOTOR3_IN1, 4);
  ledcAttachPin(MOTOR3_IN2, 5);

- Konfiguracja pinu sterującego trybem nadawania/odbioru dla RS485 oraz inicjalizacja komunikacji RS485. Ustawia pin do kontrolowania trybu nadawania/odbioru, inicjalizuje komunikację RS485 z prędkością 9600 bodów.

  pinMode(RS485_DE_PIN, OUTPUT);
  digitalWrite(RS485_DE_PIN, LOW);
  RS485Serial.begin(9600);
  Serial.println("RS485 Initialized");

2.3 Pętla główna (loop())

- Odczyt komend z portu szeregowego i kontrola stanu trzeciego silnika (START_FAN/STOP_FAN).

```
if (rs485.available()) {
  String command = rs485.readStringUntil('\n');
  command.trim();

  if (command == "START_FAN") {
    motor3Enabled = true;
    printRS485("Komenda: START_FAN - Wentylator 3 włączony.");
  } else if (command == "STOP_FAN") {
    motor3Enabled = false;
    printRS485("Komenda: STOP_FAN - Wentylator 3 wyłączony.");
  } else {
    printRS485("Nieznana komenda: " + command);
  }
}
```

- Odczyt temperatury z czujnika DHT i wyświetlenie jej na wyświetlaczu LCD oraz sterowanie PWM dla silników.

```
if (updateTemperature) {
  updateTemperature = false;

  float temperature = dht.readTemperature();
  if (isnan(temperature)) {
    Serial.println("Błąd odczytu z czujnika DHT");
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Blad odczytu");
  } else {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Temp: ");
    lcd.print(temperature);
    lcd.print(" C");

    int pwmValue = 0;
    if (temperature >= 20 && temperature <= 30) {
      pwmValue = map(temperature, 20, 30, 238, 255);
    } else if (temperature >= 31) {
```

```
            pwmValue = 255;
        } else {
            pwmValue = 0;
        }

        ledcWrite(0, pwmValue);
        ledcWrite(1, 0);
        ledcWrite(2, pwmValue);
        ledcWrite(3, 0);

        if (motor3Enabled) {
            ledcWrite(4, pwmValue);
            ledcWrite(5, 0);
        } else {
            ledcWrite(4, 0);
            ledcWrite(5, 0);
        }

        printRS485("Temperatura: ");
        printRS485(temperature);
        printRS485(" *C, PWM: ");
        printRS485(pwmValue);
    }
}
```

2.4 Podsumowanie kodu

Program steruje trzema silnikami na podstawie odczytów temperatury z czujnika DHT, wyświetla temperaturę na wyświetlaczu LCD oraz wysyła te dane przez interfejs RS485. Umożliwia również włączanie i wyłączanie trzeciego silnika (wentylatora) za pomocą komend wysyłanych przez port szeregowy. Całość jest skonfigurowana tak, aby reagować dynamicznie na zmiany temperatury oraz komendy użytkownika.

System monitorowania temperatury i sterowania wentylatorami (Fan Controller) działa zgodnie z założeniami, skutecznie zarządzając temperaturą w oparciu o dane z czujnika DHT11. Komponenty systemu, w tym czujnik, wyświetlacz LCD, sterowniki PWM oraz interfejs RS485, zostały poprawnie zintegrowane, zapewniając stabilną i wydajną pracę całego układu. Wentylatory reagują na zmiany temperatury zgodnie z zaprogramowanym algorytmem, a dane są przesyłane w sposób niezawodny przez RS485, co jest istotne w zastosowaniach, które wymagają długozasięgowej transmisji danych.

Zintegrowanie systemu z interfejsem RS485 umożliwia łatwą komunikację z innymi urządzeniami, co jest korzystne w przypadku rozbudowanych systemów monitorowania lub automatyki. Dzięki prostemu i intuicyjnemu sposobowi sterowania prędkością wentylatorów na podstawie temperatury, system może być wykorzystywany w szerokim zakresie, od wentylacji w małych pomieszczeniach po zaawansowane systemy klimatyzacyjne w przemysłowych instalacjach.

W kontekście codziennych czynności oraz przy wprowadzeniu mocniejszych wentylatorów z lepszymi silnikami projekt może znaleźć szerokie **zastosowanie** w wielu dziedzinach, ułatwiając codzienne życie i pracę. **Na przykład:**

- Inteligentne zarządzanie klimatem w domu: Wentylatory uruchamiają się automatycznie, gdy temperatura przekroczy określony próg, co zapewnia komfort bez konieczności manualnego sterowania.
- Przemysł i serwerownie: W środowiskach przemysłowych lub serwerowniach, gdzie odpowiednia temperatura jest kluczowa dla utrzymania sprzętu w dobrym stanie, system może zapewniać automatyczne chłodzenie. Może działać w połączeniu z innymi urządzeniami monitorującymi, umożliwiając pełną automatyzację sterowania temperaturą i wentylacją, minimalizując ryzyko przegrzania urządzeń.
- Hodowla roślin i zwierząt: W sektorze rolniczym system może być wykorzystywany do monitorowania i utrzymywania odpowiednich warunków w szklarni, stajni lub innych przestrzeniach przeznaczonych do hodowli roślin i zwierząt.
- Przechowywanie wrażliwych materiałów: W magazynach, gdzie przechowywane są materiały wrażliwe na temperaturę, jak elektronika, leki czy żywność, system może pełnić rolę nadzorcy, który reaguje na zmiany temperatury. Może to pomóc w zmniejszeniu strat i zapewnieniu bezpieczeństwa przechowywanych materiałów.

Projekt ten, dzięki swojej uniwersalności, może zatem znaleźć zastosowanie w wielu dziedzinach życia codziennego, przemysłowego i naukowego, przyczyniając się do poprawy komfortu, efektywności oraz bezpieczeństwa w wielu różnych warunkach.