



Kierunek: Elektronika i Telekomunikacja
Specjalność: Teleinformatyka

Praca dyplomowa inżynierska

Stacja meteorologiczna oparta o ESP8266

Damian Zaręba
Nr albumu 8389

Promotor:
dr inż. Tadeusz Leszczyński

Mława 2019r.

Spis treści

Spis treści	2
1 Wstęp	4
2 Elementy składowe projektu	5
2.1 ESP8266EX	5
2.2 BME280	10
2.3 PMS7003	12
3 Wykorzystane protokoły komunikacyjne	14
3.1 I ² C	14
3.2 UART	18
4 Schemat funkcjonalny	20
5 Schemat elektryczny	21
5.1 Ogólny przegląd	21
5.2 Panel słoneczny	22
5.3 Zewnętrzne źródło zasilania	22
5.4 Ładowanie akumulatora LiFePO4	23
5.5 Konwersja z 9V do 5.6V	24
5.6 Konektory dla sensorów pogodowych	25
5.7 Wewnętrzny system dystrybucji zasilania	26
5.8 Pomiar napięcia baterii pinem ADC	27
5.9 Rezystory podciągające dla magistrali I ² C	27
5.10 Pamięć Flash dla ESP8266EX	28
5.11 Linie wejśc/wyjśc ESP8266EX	28
5.12 Sieć dopasowująca dla anteny Wi-Fi	29
5.13 Filtracja zasilania ESP8266EX	29
5.14 Piny programowania ESP8266	30
6 Kod źródłowy	31
6.1 Środowisko programistyczne	31
6.2 Kod przez rozpoczęciem	31
6.3 Kod rozpoczynający	35
6.4 Kod w pętli	37
7 Infografika	38
7.1 Zdjęcia urządzenia	38

7.2 Pomiar sekcji zasilania ESP8266EX	40
8 Streszczenie	41
Spis rysunków	42
Spis tabel	43
Bibliografia	44

1. Wstęp

Celem pracy jest zaprojektowanie i zrealizowanie stacji meteorologicznej opartej o mikroprocesor ESP8266, złożonej z kilku modułów. Aby zrealizować założony cel, zrealizowane zostały następujące zadania:

- Zaprojektowanie i wykonanie płyty głównej z mikrokontrolerem ESP8266EX dla przetwarzania informacji z sensorów oraz sterowania zasilaniem całego urządzenia;
- Zaimplementowanie sensora firmy BOSCH o nazwie BME280, który służy do odczytu temperatury, ciśnienia i wilgotności powietrza;
- Zaimplementowanie czujnika firmy PLANTOWER o nazwie PMS7003, który mierzy ilość pyłu zawieszonego w powietrzu, o wielkości PM1.0 oraz PM10, mierzone w $\mu\text{g}/\text{m}^3$.

W kolejnych rozdziałach pracy przedstawiono schemat blokowy urządzenia oraz ogólny opis poszczególnych modułów wykorzystanych do zbudowania tego urządzenia, wliczając w to charakterystyki głównych komponentów dla każdego modułu. Udokumentowane zostało m.in. konfiguracja środowiska, które zostało wykorzystana do stworzenia tego projektu.

Przeanalizowano szczegółowo schemat urządzenia, a konkretnie płyty głównej, sekcji zasilania dla wykorzystanych sensorów oraz innych elementów niezbędnych do realizacji projektu. Poddana dokładnej analizie będzie każda z części schematu, takie jak sekcja zasilania czy połączeniowa między płytą główną a sensorami.

W przedostatnim rozdziale przedstawiono i opisano algorytmy oraz kod źródłowy programu sterującego stacją pogodową oraz omówiono protokoły komunikacyjne. Wykonuje on wiele zadań, m.in. odczytuje dane z sensorów czy kontroluje układy zasilania poszczególnych części.

W ostatnim rozdziale przedstawione zostały wyniki badań wykonanego modelu stacji pogodowej.

2. Elementy składowe projektu

2.1 ESP8266EX

ESP8266EX to mikroukład z pełnym stosem TCP/IP oraz mikrokontrolerem wyprodukowanym przez Espressif w Szanghaju, Chiny.

Istnieje jego odmiana o nazwie ESP8285 z 1 MiB wbudowanej pamięci typu flash, co umożliwiało wykorzystanie go jako pojedynczego układu zdolnego do podłączenia się do sieci Wi-Fi, po podłączeniu zasilania. W odróżnieniu od rodziny mikrokontrolerów AVR nie może być zasilany napięciem 5V, jedynie 3.3 volta.



Rysunek 2.1: Zdjęcie przedstawiające układ ESP8266EX

Źródło: [3]

ESP8266EX [1] posiada 32 bitowy procesor oparty o rdzeń Xtensa Diamond Standard 106Micro (LX106) firmy Tensilica o nominalnej wartości zegara wynoszącym 80 MHz. Charakteryzuje się on następującymi funkcjami:

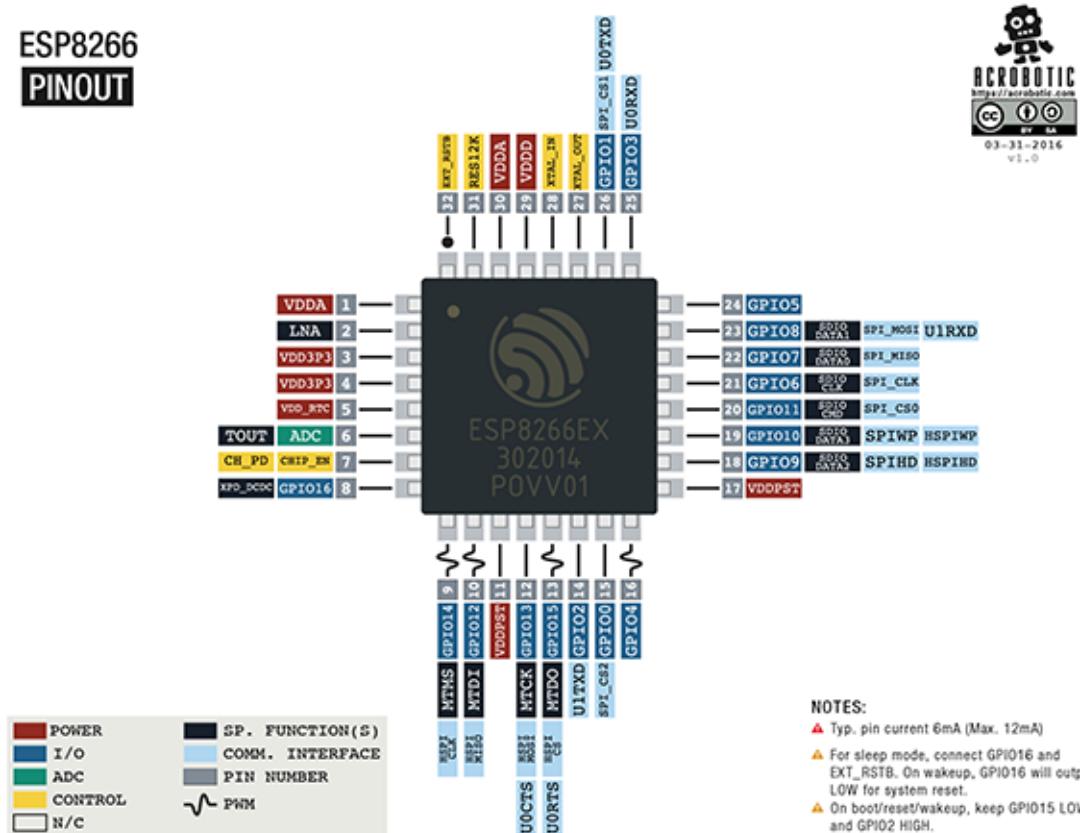
- 16 pinów GPIO
- SPI
- I²C (programowa implementacja)
- I²S z funkcją Direct Memory Access (współdzieli piny z GPIO)
- UART na wyznaczonych pinach GPIO oraz dodatkowy UART na GPIO2 służący jedynie do wysyłania danych
- 10-bitowy ADC oparty o sukcesywną aproksymację.

- Wbudowana obsługa Wi-Fi o standardach b/g/n według IEEE 802.11 z wbudowanym przełącznikiem TR,LNA,Balunem, wzmacniaczem mocy oraz siecią dopasowującą oraz możliwością podłączenia się lub tworzenia sieci z zabezpieczeniami WEP lub WPA/WPA2

Pamięć ulotna tego mikrokontrolera jest podzielona w następujący sposób :

- 32 KiB RAM dla instrukcji
- 32 KiB RAM typu cache dla instrukcji
- 80 KiB RAM dla danych użytkownika
- 16 KiB RAM typu ETS dla danych "systemowych"

Obsługuje pamięć nieulotną typu flash po protokole SPI do pojemności 16 MiB, choć zazwyczaj korzysta się z pamięci o rozmiarach 512 KiB lub 4 MiB.



Rysunek 2.2: Zdjęcie przedstawiające wyprowadzenia dla układu ESP8266EX

Źródło: [4]

Układ ESP8266EX posiada kilka linii zasilania:

- 2x VDDA (Zasilanie sekcji analogowej)- 2.5 wolt do 3.6 wolt
- VDDD (To samo, co VDDA) - 2.5 wolt do 3.6 wolt.
- 2x VDD3P3 (Zasilanie wzmacniacza sygnału Wi-Fi) - 2.5 wolt do 3.6 wolt.
- 2x VDDPST (Zasilanie sekcji cyfrowej i wejść/wyjść) - 1.8 wolt do 3.6 wolt.

Używa się pinów GPIO2, GPIO0 oraz GPIO15/MTDO dla ustawienia trybu uruchamiania układu.

	GPIO0	GPIO2	GPIO15
Tryb programowania	L	H	L
Uruchamianie z pamięci flash	H	H	L
Uruchamianie z karty SD	X	X	H

Tabela 2.1: Tabela przedstawiająca tryby uruchamiania dla układu ESP8266EX

Źródło: [2]

ESP8266EX obsługuje 14 kanałów w łączności Wi-Fi

Numer kanału	Częstotliwość [MHz]
1	2412
2	2417
3	2422
4	2427
5	2432
6	2437
7	2442
8	2447
9	2452
10	2457
11	2462
12	2467
13	2472
14	2484

Tabela 2.2: Tabela przedstawiająca dostępne kanały łączności Wi-Fi dla układu ESP8266EX

Źródło: [2]

Maksymalny pobór prądu przez sam układ, nie licząc łączności Wi-Fi wynosi 12mA. Pobór wzrasta w zależności od mocy transmisji lub odbioru pakietów za pomocą Wi-Fi. Pomiary były wykonane dla napięcia 3 wolt w temperaturze 25°C. Wszystkie pomiary przesyłu danych były bazowane na 50% długości cyklu

Parametry łączności	Minimalny	Typowy	Maksymalny	Jednostka
TX 802.11b CCK 11 Mbps $P_{OUT} = +15 \text{ dBm}$	-	170	-	mA
TX 802.11g OFDM 54 Mbps $P_{OUT} = +17 \text{ dBm}$	-	140	-	mA
TX 802.11n MCS7 $P_{OUT} = +13 \text{ dBm}$	-	120	-	mA
RX 802.11b 1024 bajtów -80 dBm	-	50	-	mA
RX 802.11g 1024 bajtów -70 dBm	-	56	-	mA
RX 802.11n 1024 bajtów -65 dBm	-	56	-	mA

Tabela 2.3: Tabela przedstawiająca pobór mocy dla łączności Wi-Fi przez ESP8266EX

Źródło: [2]

2.2 BME280

BME280 to sensor temperatury, wilgotności i ciśnienia wykonany przez firmę Bosch. Jego rozmiary to 2.5 milimetra na 2.5 milimetra w obudowie typu LGA (Land Grid Array). Komunikuje się z mikrokontrolerem za pomocą protokołu I²C (do 3.4 MHz) lub SPI (3 lub 4 przewodowego, do 10 MHz)



Rysunek 2.3: Zdjęcie przedstawiające układ BME280

Źródło: [6]

Zasilany jest w projekcie napięciem 3.3V, choć jego specyfikacja podaje, że pin zasilania VDD samego układu toleruje napięcia od 1.71 wolt do 3.6 wolt, a pin zasilania wejść/wyjść VDDIO od 1.2 wolt do 3.6 wolt.



Rysunek 2.4: Zdjęcie przedstawiające moduł zawierający BME280, komunikujący się przez protokół I²C

Źródło: [7]

Pobiera w zastosowanej aplikacji około $3.6\mu\text{A}$, czyli pomiaru wilgotności, ciśnienia i temperatury z częstotliwością 1Hz, choć może być niższy, ponieważ pomiary będą dokonywane rzadziej, około co pół godziny lub godzinę, w zależności od wymagań. Będzie możliwość ustawienia tego w kodzie źródłowym programu dla tej aplikacji.

Kluczowe parametry sensora wilgotności:

- Czas odpowiedzi ($\tau_{63\%}$) wynosi 1 sekundę
- Tolerancja dla dokładności pomiaru wynosi $\pm 3\%$ relatywnej wilgotności
- Histereza pomiaru wynosi $\pm 1\%$ relatywnej wilgotności

Kluczowe parametry części pomiarowej dla ciśnienia:

- Średni poziom szumów wynosi 0.2 Pa, co jest odpowiednikiem dla 1.7 cm
- Współczynnik odchylenia temperaturowego wynosi $\pm 1.5 \text{ Pa/K}$, co jest równoważne dla $\pm 12.6 \text{ cm}$ przy zmianie temperatury o 1°C

2.3 PMS7003

Sensor PMS7003 służy do pomiaru ilości cząsteczek w powietrzu o gradiacji PM1.0, PM2.5 oraz PM10 (odpowiednio: cząsteczki o średnicy około $1\mu\text{m}$, $2.5\mu\text{m}$ oraz $10\mu\text{m}$). Powszechnie korzysta się z niego do pomiaru jakości powietrza. Wykorzystuje on do tego laserowy czujnik pyłków.

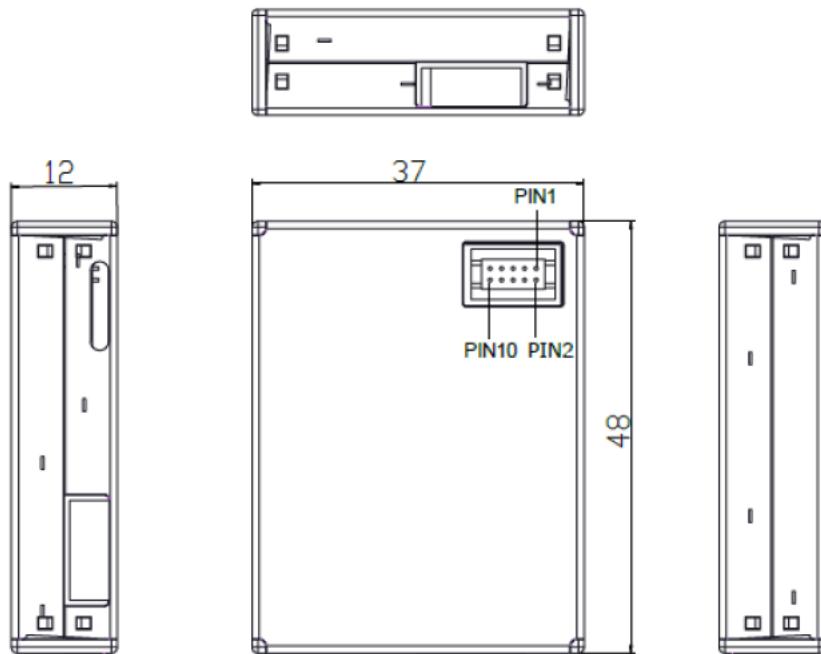


Rysunek 2.5: Zdjęcie przedstawiające moduł PMS7003, komunikujący się przez protokół UART

Źródło: [9]

Czujnik ten charakteryzuje się następującymi parametrami:

- Pojedynczy czas odpowiedzi wynosi mniej niż 1s
- Całkowity czas odpowiedzi wynosi do 10 sekund
- Zasilany jest napięciem 5 wolt
- Pobór prądu podczas pomiarów wynosi do 100 mA
- Stały pobór prądu, kiedy czujnik nie wykonuje pomiaru wynosi do 200 μA
- Napięcie operacyjne dla przesyłu i odbioru danych to 3.3 volta.
- Pracuje w temperaturze od -10 do +60 stopni Celcjusza.
- "Pojemność" czujnika wynosi 0.1L



Rysunek 2.6: Zdjęcie przedstawiające wyprowadzenia czujnika PMS7003

Źródło: [8]

Lista wyprowadzeń tego sensora jest następująca:

Wyprowadzenia PMS7003		
Pin 1	VCC	Zasilanie 5V
Pin 2	VCC	Zasilanie 5V
Pin 3	GND	Masa zasilania
Pin 4	GND	Masa zasilania
Pin 5	RESET	Pin resetu modułu
Pin 6	NC	-
Pin 7	RX	Pin odbierania danych
Pin 8	NC	-
Pin 9	TX	Pin wysyłania danych
Pin 10	SET	Stan wysoki - normalna operacja Stan niski - uśpienie układu

Tabela 2.4: Tabela przedstawiająca listę wyprowadzeń modułu PMS7003

Źródło: [8]

3. Wykorzystane protokoły komunikacyjne

3.1 I²C

I²C [10] to protokół szeregowy zaprojektowany przez firmę Phillips Semiconductor, która dziś działa jako NXP Semiconductors w 1982 roku. Ma możliwość podpięcia wielu urządzeń w trybie *Master* oraz w trybie *Slave*. Istnieją dwie dodatkowe wersje I²C - SMBus (System Management Bus), wykorzystywany w komputerach klasy PC do zarządzania podzespołami oraz PMBus (Power Management Bus) - do kontroli urządzeń związanych z zasilaniem.

Wykorzystuje on dwie obustronne linie z otwartym kolektorem lub otwartym drenem, Serial Data Line (SDA) i Serial Clock Line (SCL), z rezystorami podciągającymi do zasilania. Zazwyczaj używa się napięć +5V lub +3.3V, ale inne napięcia są również dozwolone.

Magistrala I²C posiada kilka trybów prędkości:

- Low-speed - 10 kbit/s
- Standard - 100 kbit/s
- Fast - 400 kbit/s
- Fast mode plus (Fm+) - 1 Mbit/s
- High speed - 3.4 Mbit/s

Z tych najszybszych korzysta się w systemach wbudowanych a nie w komputerach osobistych.

Magistrala I²C posiada dwie ważne cechy - rozszerzanie zegara (Clock Stretching) oraz Arbitraż (Arbitration). Rozszerzanie zegara polaga na tym, że urządzenie typu *Slave* może utrzymywać linię zegara SCL w stanie niskim po otrzymaniu lub wysłaniu bajtu informacji, która wskazuje, że nie jest jeszcze gotowe do przetwarzania kolejnej porcji danych. Urządzenie typu *Master* komunikujące się z danym modułem nie może wtedy zakończyć transmisji tylko musi czekać aż linia zegara SCL będzie w stanie wysokim. Jest to jedyna sytuacja, gdzie urządzenie typu *Slave* ma kontrolę nad linią zegara. Urządzenie typu *Master* musi odczekać dodatkowe 4 µs po prześciu linii SCL w stan wysoki zanim będzie mógł podciągnąć linię zegara do stanu niskiego.

Magistrala I²C posiada deterministyczny system arbitrażu. Arbitraż polega na tym, że każde urządzenie typu *Master* monitoruje linię, wyszukując bitów START i STOP i nie zacznie komunikacji, póki linie są zajęte przez inne urządzenia typu *Master*. Jednakże, dwa urządzenia typu *Master* mogą zacząć transmisję w tym samym czasie - wtedy następuje arbitraż. Każdy transmitter sprawdza poziom linii danych (SDA) i porównuje do oczekiwanej; jeśli się nie zgadza to urządzenie transmitujące traci arbitrację i wyłącza się z danej interakcji z protokołem I²C.

Status zajętości magistrali oraz bitów START i STOP.					
Typ	Linia Nieaktywna (N)	Start (S)	Oczekiwanie (I)	Stop (P)	Rozszerzanie zegara (CS)
Opis	Arbitraż wolny do zajęcia	Zajmowanie linii (Master)	Linia zajęta (Master)	Zwalnianie linii (Master)	Zastopowanie (Slave)
SDA	Podciąganie pasywne	Zbocze opadające (Master)	Stan niski (Master)	Zbocze wzrastające (Master)	Nie ma znaczenia
SCL	Podciąganie pasywne	Podciąganie pasywne	Pasywne podciąganie	Pasywne podciąganie	Stan niski (Slave)

Tabela 3.1: Tabela przedstawiająca stany zajętości linii w magistrali I²C

Źródło: [10]

Typ	Wysłanie jednego bitu danych (I) (0) (Linia SDA jest ustawiona po SCL w celu uniknięcia wykrycia fałszywego stanu linii) Ustawienie bitów (Bs)		Odpowiedź odbiorcy za pomocą bitu ACK (Bajt odebrany od nadawcy) Ustawienie bitów (Bs)		Odpowiedź odbiorcy za pomocą bitu NACK (Bajt nie odebrany od nadawcy) Ustawienie bitów (Bs)	
	Gotowość do próbowania (Bx)	ACK (A)	NACK(A')			
Opis	Nadawca ustawia bit (Master/Slave)	Odbiorca próbuje bit (Master/Slave)	Nadawca ustawia stan wysokiej impedancji	Nadawca widzi linię SDA jako w stanie niskim	Nadawca ustawia stan wysokiej impedancji	Nadawca widzi stan wysoki
SDA	Ustawia bit (Po opadnięciu SCL)	Odbiera bit (Po wzroście SCL)	Utrzymane w stanie niskim (Po opadnięciu SCL)		Sterowane stanem wysokim (Lub pasywnie wysokim) przez odbiorcę (Po opadnięciu SCL)	
SCL	Opadające zbocze (Master)	Wzrastające zbocze (Master)	Opadające zbocze (Master)	Wzrastające zbocze (Master)	Opadające zbocze (Master)	Wzrastające zbocze (Master)

Tabela 3.2: Tabela przedstawiająca stany linii w magistrali I²C przy wymianie informacji między odbiorcą a nadawcą

Źródło: [10]

Magistrala I²C posiada 7 bitową adresację oraz 10 bitową adresację, która jest wykorzystywana jako rozszerzenie podstawowej, 7 bitowej adresacji. 7 bitowa adresacja posiada 2 zarezerwowane grupy adresów - *0000 XXX* oraz *1111 XXX*:

- *0000 000 0* - Generalne odwołanie
- *0000 000 1* - bajt START
- *0000 001 X* - Adres CBUS
- *0000 010 X* - Zarezerwowane dla innego formatu magistrali
- *0000 011 X* - Zarezerwowane na przyszły użytk
- *0000 1XX X* - Kod trybu High Speed
- *1111 1XX 1* - Kod identyfikacyjny urządzenia
- *1111 0XX X* - 10 bitowa adresacja urządzenia typu *Slave*

7-bitowa adresacja składa się z 7 bitów identyfikacji oraz ósmego bitu - oznaczenia zapisu/odczytu danych. Długość części MSB to 4 bity, a LSB - 3 bity.

Pole	S	Adres I ² C								R/W	A	Sekwencje danych I ² C				P
Typ		Pierwszy bajt														
Pozycja bitu w bajcie X		7 6 5 4 3 2 1 0														
Pozycja 7-bitowego adresu	START	7 6 5 4 3 2 1 0									ACK	Kolejny bajt itd. Dalsze wiadomości zapisu lub odczytu idą tutaj.				STOP
Opis		MSB				LSB		0 = Zapis 1 = Odczyt								

Tabela 3.3: Tabela przedstawiająca strukturę 7-bitowej adresacji w magistrali I²C

Źródło: [10]

10-bitowy adres jest podzielony na 2 segmenty - górny i dolny, które są poprzedzone specjalnym adresem, który komunikuje, że wykorzystywana jest 10-bitowa adresacja w magistrali - *1111 0XX X*.

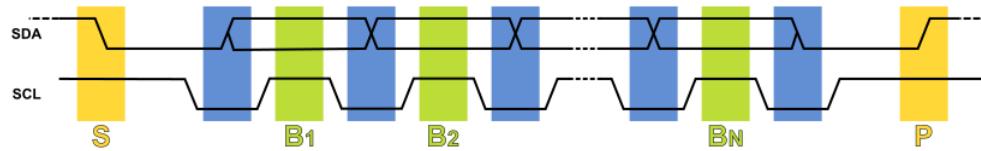
Pole:	S	Wskaznik 10-bitowej Adresacji								Górny adres	R/W	A	Dolny adres								Sekwencje danych I ² C	P
Typ		Pierwszy bajt											Drugi bajt									
Pozycja bitu W bajcie X		7	6	5	4	3	2	1	0				7	6	5	4	3	2	1	0		
Wartość bitu	START	1	1	1	1	0	x	x	x			ACK	x	x	x	x	x	x	x	x		
Pozycja 10-bitowego Adresu							10	9					8	7	6	5	4	3	2	1		
Opis		Wskazuje 10-bitową Adresację								MSB			0 = Zapis 1 = Odczyt							LSB		

Rysunek 3.1: Tabela przedstawiająca strukturę 10-bitowej adresacji w magistrali I²C

Źródło: [10]

Przebieg czasowy protokołu I²C wygląda następująco:

1. Przesył danych jest inicjowany za pomocą bitu START (S) zasygnalizowane przez podcięcie gniazdecie linii SDA w stan niski, kiedy linia SCL jest w stanie wysokim;
2. Linia SCL jest podciagnięta do stanu niskiego, a linia SDA ustawa poziomu pierwszych bitów danych utrzymując linię SCL w stanie niskim (cała długość niebieskiego prostokąta);
3. Dane są próbkowane (odbierane) kiedy linia SCL jest w stanie wzrastającego zbocza dla pierwszego bitu. Źeby bit był poprawny, SDA nie może zmienić swojego stanu między wzrostającym zboczem linii SCL oraz następującym opadem zbocza (cała długość zielonego prostokąta);
4. Powyższa czynność jest powtarzana dla każdego kolejnego bitu;
5. Ostatni bit jest poprzedzony pulsem zegarowym, podczas którego linia SDA jest podciagnięta do stanu niskiego do przygotowania się do bitu STOP;
6. Bit STOP (P) jest sygnałowany kiedy linia SCL wzrasta, poprzedzona wzrostaniem linii SDA.



Rysunek 3.2: Obrazek przedstawiający diagram czasowy w magistrali I²C

Źródło: [10]

3.2 UART

UART (Universal asynchronous receiver-transmitter) [11] to asynchroniczna, szeregowa magistrala komunikacyjna, w której format danych oraz prędkość przesyłu informacji jest konfigurowalna. Poziomy logiczne sygnałów oraz metody są kontrolowane przez zewnętrzną elektronikę podłączoną do kontrolera UART. Podobny protokół, USART (Universal synchronous and asynchronous receiver-transmitter) wspiera komunikację synchroniczną.

UART bierze bajty danych i przesyła indywidualne bity sekwencyjnie. W miejscu docelowym, urządzenie odbiorcze UART "buduje" dane na nowo z przybyłych bitów informacji w całe bajty. Każdy układ UART zawiera rejestr przesuwny, który odpowiada za przetwarzanie informacji między równoległą a szeregową formą danych. Komunikacja może odbywać się w jednym z 3 trybów:

- *Simplex* - Tylko w jedną stronę, bez nadzoru dla urządzenia odbiorczego do wysyłu danych zwrotnych do urządzenia transmitującego;
- *Half duplex* - Urządzenia zmieniają się miejscami przy przesyiale danych - raz jedno wysyła, raz drugie;
- *Full duplex* - Urządzenia wysyłają i odbierają dane w tym samym czasie.

W stanie oczekiwania, kiedy nie są wysyłane dane, linie są w stanie logicznym wysokim. Jest to pozostałość historyczna z telegrafów, gdzie linia była utrzymywana w stanie wysokim, żeby pokazać, iż linia i transmiter nie były uszkodzone. Każda część informacji jest opakowana jako bit START (logiczne 0), bity danych, opcjonalnie bit parzystości oraz jeden lub kilka bitów STOP.



Rysunek 3.3: Obrazek przedstawiający diagram czasowy w magistrali UART

Źródło: [11]

Bit START sygnalizuje urządzenie odbiorcze o tym, że nowa część informacji nadchodzi. Kolejne 5 do 9 bitów, w zależności od odu który przyjęto, reprezentuje tą część danych. Jeśli bit parzystości został użyty, będzie umieszczony po wszystkich bitach danych. Kolejne 1 lub 2 bity zawsze są w oznaczonej pozycji (Logiczna jedynka) i nazwane bitem (lub bitami) STOP. Informują one moduł odbiorczy o tym, że ciąg znaków został ukończony.

UART składa się z kilku kluczowych elementów oraz ewentualnych, opcjonalnych dodatków, które rozszerzają możliwości magistrali w różny sposób. Urządzenie do obsługi protokołu UART wymaga:

- Generator sygnału zegara, zazwyczaj wielokrotność prędkości przesyłu danych dla umożliwienia próbkowania w środku czasu trwania bitu;
- Wejściowe i wyjściowe rejestyry przesuwne;
- Kontrola przesyłu/odbioru danych;
- Logika kontrolująca zapis/odczyt danych.

Dodatkowo, można wyposażyć protokół w dodatkowe bloki funkcyjne, takie jak:

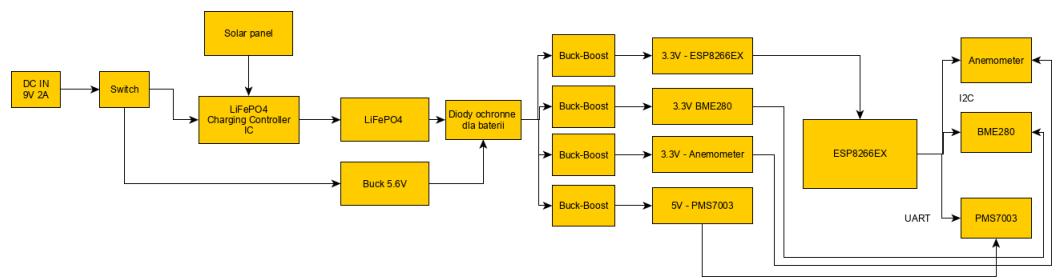
- Bufory przesyłu/odbioru danych;
- Bufor magistrali danych systemowych;
- Bufor pamięci typu FIFO (First-in, first-out);
- Sygnały potrzebne dla zewnętrznego kontrolera DMA (Direct Memory Access);
- Zintegrowana magistrala zarządzająca kontrolerem DMA.

UART posiada kilka specjalnych zdarzeń, które mogą wystąpić podczas transmisji. Są to:

- Błąd przekroczenia (Overrun error) - Odbiornik nie może przetworzyć znaku który przybył przed kolejnym znakiem, który nadchodzi;
- Błąd niedociągnięcia (Underrun error) - Nadajnik skończył wysyłać dane i bufor przesyłu jest pusty;
- Błąd obramowania (Framing error) - Urządzenie nie "widzi" bitu STOP w oczekiwany czasie jego występowania;
- Błąd parzystości (Parity error) - Bit parzystości nie zgadza się między odbiornikiem a nadajnikiem;
- Zerwanie transmisji (Break condition) - Wejście odbiornika jest w „odstępie” (Stan niski) dłużej niż przewidywany czas oczekiwania, zazwyczaj dłużej niż długość znaku.

4. Schemat funkcjonalny

Zaprojektowane i wykonane urządzenie składa się z 3 głównych części - Dwóch różnych, wybieranych przez użytkownika źródeł zasilania (Zasilaczem lub baterią), sekcji zasilania mikrokontrolera i sensorów, oraz samego ESP8266EX i połączeń do wyprowadzeń dla czujników. Zasilanie jest czterokanałowe dla zapewnienia stabilności napięcia dla każdego ważnego układu w projekcie. Przełącznik pozwala na przełączanie między ładowaniem baterii a zasilaniem całej płyty głównej. Panel słoneczny jest bezpośrednio połączony do kontrolera ładowania.



Rysunek 4.1: Obrazek przedstawiający schemat blokowy urządzenia

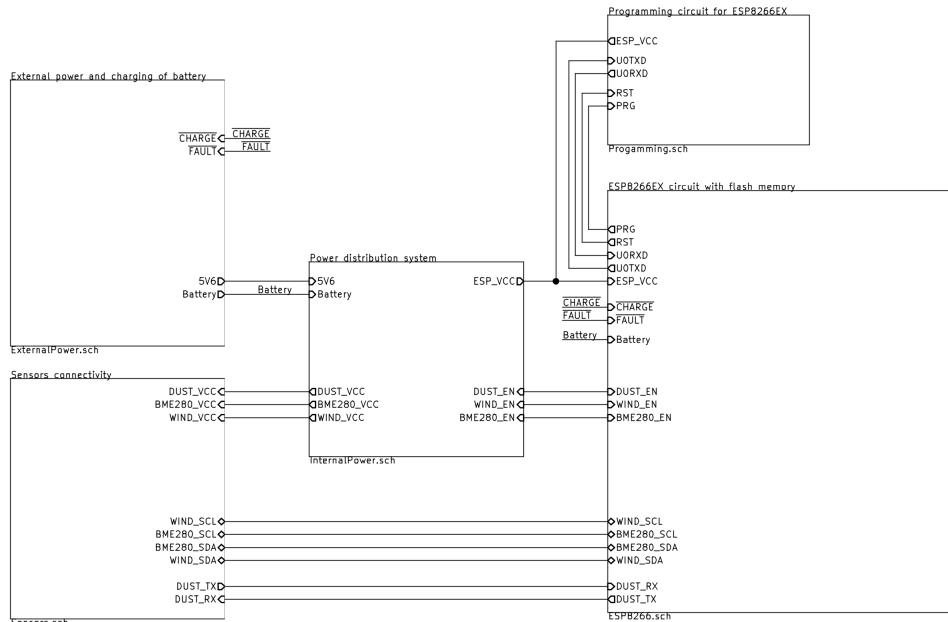
Źródło: Opracowanie własne

Wykorzystano baterię LiFePO4, które pozwalają na bardziej wydajną pracę od Li-Ion czy LiPo w temperaturach poniżej zera, ponieważ jedynie tracą około 10% pojemności niżej zera stopni Celsjusza, nie 30+ % jak Li-Ion czy LiPo. Wykorzystano przetwornice DC-DC typu buck-boost (push-pull) dla stabilizacji napięć z baterii LiFePO4. Mimo dużej ilości zakłóceń generowanych przez przetwornice DC-DC, posiadają o wiele wyższą sprawność od liniowych kontrolerów napięć. Diody Schottky między baterią a przetwornicami uniemożliwi ładowania baterii napięciem 5.6V z kontrolera DC-DC który służy do zasilania całego układu przy odpowiednim ustawieniu przełącznika w tryb zasilania. Wybrano zasilanie 9V i 3A dla dostarczenia odpowiedniej ilości mocy zarówno do ładowania baterii, jak i do samodzielniego zasilania całego układu. Napięcie z przetwornicy podpiętej do zewnętrznego zasilania 9V i 3A wynosi 5.6V, ponieważ po wykorzystaniu diody typu Schottky napięcie nie przekracza maksymalnego dla wykorzystanych przetwornic DC-DC, które wynosi 5.5V. Konwertery napięć typu push-pull dają na wyjściu 3.3V dla sensorów i mikrokontrolera, oraz 5V dla czujnika cząsteczek stałych w powietrzu. Wykorzystany panel fotowoltaiczny musi być na napięcie wynoszące 12V pod obciążeniem.

5. Schemat elektryczny

Schemat został podzielony na bloki funkcyjne, które zostały szczegółowo omówione podczas ich prezencji. Wykorzystane oprogramowanie do narysowania schematu oraz zaprojektowania płyt drukowanej nazywa się KiCAD [12]. Jest to multiplatformowe, darmowe i otwartoźródłowe (FOSS) oprogramowanie do projektowania elektroniki na Windowsa, macOS i Linuksa.

5.1 Ogólny przegląd

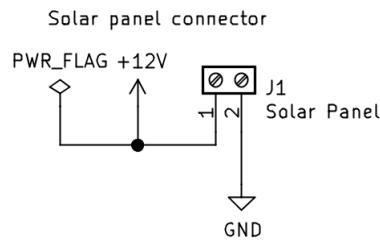


Rysunek 5.1: Obrazek przedstawiający ogólny przegląd schematu projektu

Źródło: Opracowanie własne

Przedstawiono ogólny zarys całego projektu. Jest to schemat nadzędny. Wykorzystano tzw. schematy hierarchiczne dla uporządkowania całości oraz dodania możliwości wykorzystania powstałych „modułów” w innych projektach, co znacząco przyspiesza projektowanie przyszłych urządzeń, które wykorzystują np. ESP8266EX zastosowany w tym projekcie.

5.2 Panel słoneczny

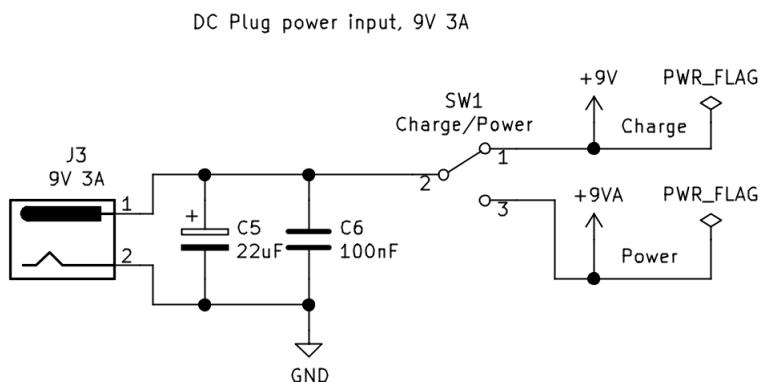


Rysunek 5.2: Obrazek przedstawiający wycinek schematu z konektorem dla panelu słonecznego

Źródło: Opracowanie własne

Wykorzystano 2 pinowy konektor do podłączenia przewodów do panelu słonecznego. Zaciska się połączenie między płytą drukowaną a przewodami za pomocą śrub w konektorze, który może obsługiwać maksymalnie 300V i 18A. Oznaczono również go jako źródło zasilania dla 12V linii, prowadzącej do układu ładowania baterii LiFePO4 za pomocą energii słonecznej.

5.3 Zewnętrzne źródło zasilania

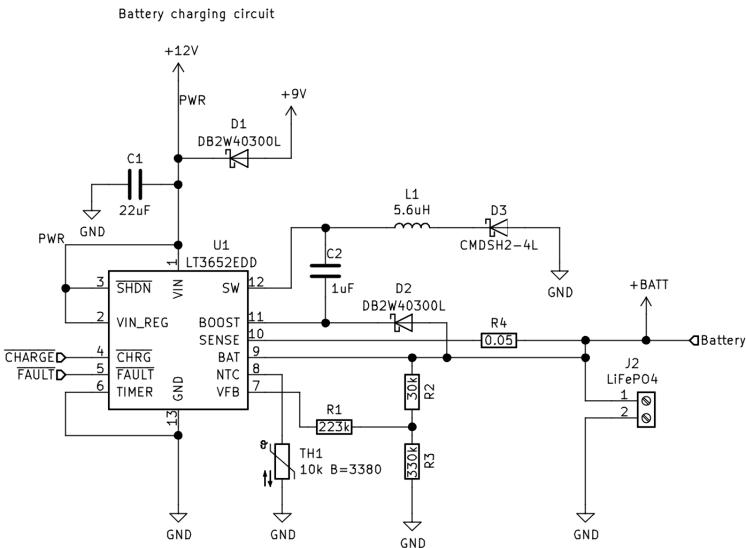


Rysunek 5.3: Obrazek przedstawiający wycinek schematu z konektorem dla zewnętrznego zasilacza do ładowania lub zasilania układu

Źródło: Opracowanie własne

Zasilanie 9V 3A to standardowy, okrągły konektor typu DC, 5.5mm, przystosowany do napięć rzędu 24V i prądu rzędu 8A, z dodatkową osłoną na gniazdo. Podłączone zostały kondensatory do filtracji zasilania, a po nich - przełącznik do wyboru między zasilaniem całości układu a ładowaniem baterii.

5.4 Ładowanie akumulatora LiFePO4



Rysunek 5.4: Obrazek przedstawiający wycinek schematu z układem ładowania akumulatora Li-FePO4

Źródło: Opracowanie własne

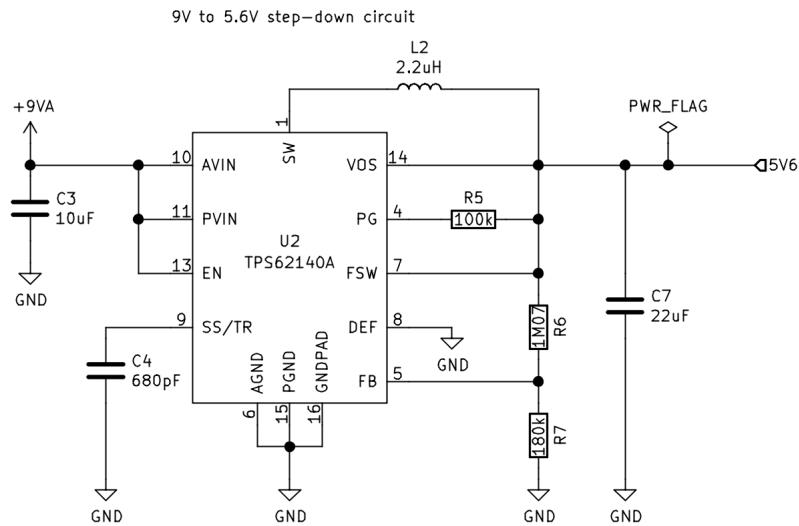
Dioda typu Schottky D1 wykorzystana została jako rozdzielenie między ładowaniem akumulatora za pomocą zewnętrznego zasilacza a panelem słonecznym. Kondensator C10 o wartości $22\mu\text{F}$ posłużył jako odsprzęganie napięcia dla głównego układu ładowającego, LT3652EDD firmy Linear Technology. Jest on specjalnie przystosowany do ładowania baterii wykonanych w różnych technologiach, typu akumulator kwasowy, Li-Ion czy LiFePO4 za pomocą ogniwa słonecznego prądem do 2A. Nie został wykorzystany pin TIMER do ustalania czasu ładowania akumulatora.

Termistor TH1 posłużył do mierzenia temperatury ogniwa, przez co układ LT3652 mógł kontrolować cykle ładowania akumulatora. Rezystory R16, R14 i R17 posłużyły do ustalenia maksymalnego napięcia ogniwa; w ten sposób wybiera się rodzaj akumulatora podłączonego do kontrolera. Rezystor R12 wykorzystano do pomiaru prądu pobieranego przez baterię, co umożliwiło kontrolę cykłów ładowania akumulatora. Wyporwadzone zostały dwa sygnały, $\sim\text{FAULT}$ i $\sim\text{CHARGE}$ do mikrokontrolera ESP8266EX, dla indykacji statusu układu ładowania ogniwa.

Stan pinów	Status kontrolera ładowania	
$\sim\text{CHARGE}$	$\sim\text{FAULT}$	
WYŁ.	WYŁ.	Nie ładuje — oczekивание lub wyłączenie.
WYŁ.	WŁ.	Uszkodzona bateria
WŁ.	WYŁ.	Ładowanie
WŁ.	WŁ.	Błąd termistora

Tabela 5.1: Tabela przedstawiająca sygnały wysyłane przez kontroler ładowania do mikrokontrolera.

Źródło: [13]



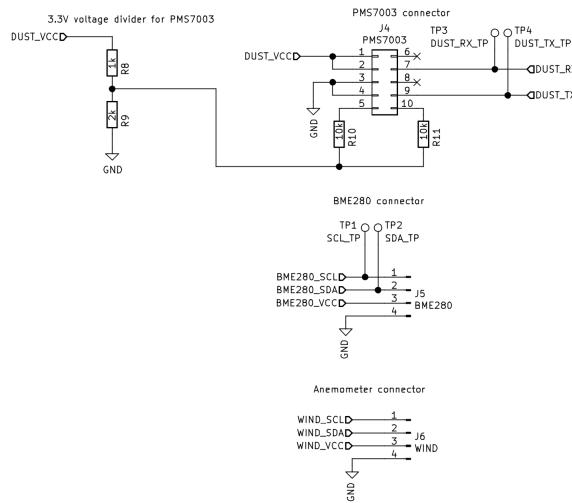
Rysunek 5.5: Obrazek przedstawiający wycinek schematu z konwersją napięcia 9V do 5.6V dla zewnętrznego zasilania lub ładowania płytki głównej.

Źródło: Opracowanie własne

5.5 Konwersja z 9V do 5.6V

Kondensator $10\mu\text{F}$ wykorzystano jako kondensator odsprzęgający dla układu konwertera DC-DC. Kolejny, o wartości 680pF posłużył do zapewnienia poprawnego działania przetwornicy, tak jak rezystor o wartości 100kOhm podłączony do pinu PG (Power Good). Rezystory 1.07 MOhm oraz 180kOhm ustawiły napięcie wyjściowe konwertera typu buck. Kondensator o wartości $22\mu\text{F}$ był niezbędny do prawidłowego obciążenia przetwornicy i filtracji wyjściowego napięcia. Cewka o wartości $2.2\mu\text{H}$ to element przechowujący energię dla układu do konwersji napięć.

5.6 Konektory dla sensorów pogodowych



Rysunek 5.6: Obrazek przedstawiający wycinek schematu z konektorami dla sensorów.

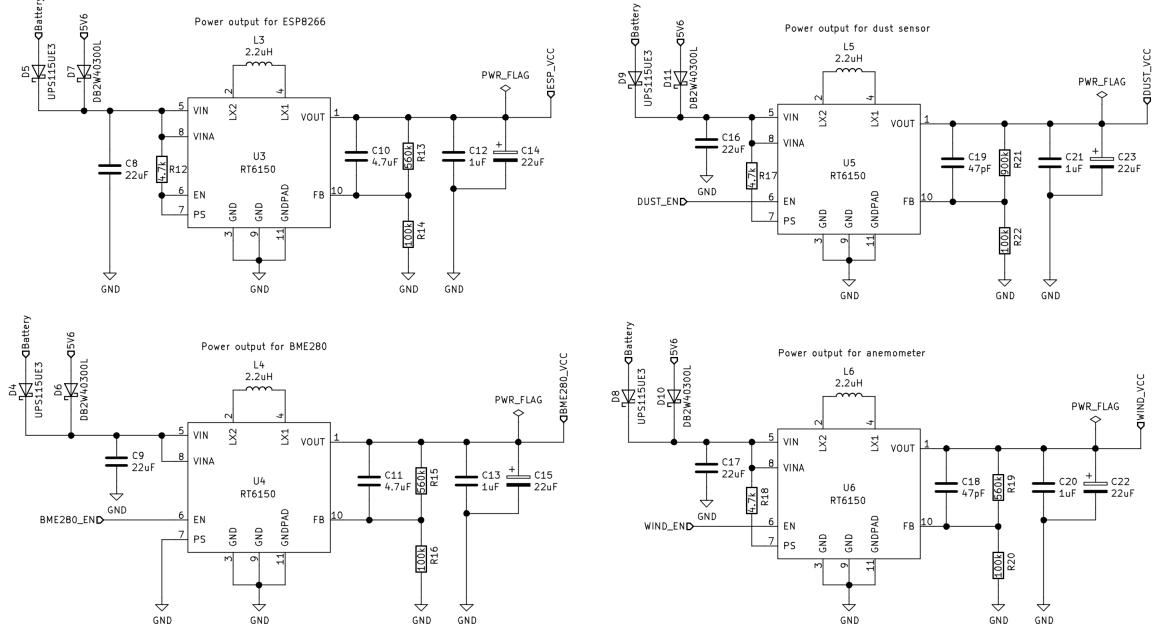
Źródło: Opracowanie własne

Przedstawiono tutaj połączenia między ESP8266EX a sensorami. Wykorzystano konektor 2-rzędowy, 5-kolumnowy dla PMS7003 oraz dwa 4 pinowe konektory 2.54mm. Dzielnik rezystorowy wykorzystano dla zmniejszenia napięcia zasilania PMS7003 z 5V do 3.3V, ponieważ komunikacja odbywa się na napięciu niższym niż napięcie zasilania czujnika. Wykorzystano również rezystory podciągające o wartości 10kOhm dla linii SET i RESET, co zostało podane w nocy katalogowej tego sensora [8].

Sensory wykorzystujące protokół I²C nie wymagają tego typu zabiegów - zasilane są i komunikują się tym samym napięciem, wynoszącym 3.3V.

Dodano również punkty testowe dla obu protokołów dla obserwacji komunikacji między ESP8266EX a sensorami.

5.7 Wewnętrzny system dystrybucji zasilania



Rysunek 5.7: Obrazek przedstawiający wycinek schematu z układem dystrybucji zasilania.

Źródło: Opracowanie własne

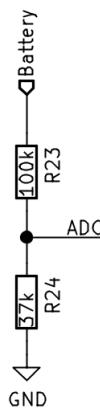
Zastosowano 4 kanały zasilania - 3 kanały z napięciem wyjściowym 3.3V oraz 1 kanał z napięciem wyjściowym 5V. Linie 3.3V dostarczają napięcie dla ESP8266EX, anemometru oraz sensora BME280. Rozdzieleno je dla zapewnienia maksymalnej stabilności napięć i prądów. Wykorzystano diody typu Schottky dla separacji linii wejściowych zasilania dla konwerterów DC-DC typu push-pull. Kondensatory o wartości $22\mu F$ działały jako odsprzęgające dla poprawnego działania przetwornic buck-boost.

Wyprowadzono sygnały kontrolujące konwertery do mikroprocesora, oprócz jednego, który zasila ten mikrokontroler. Przetwornica dla BME280 jest wprowadzona w stan oszczędzania energii, ponieważ sensor BME280 pobiera bardzo małe ilości prądu. Zrobiono to dla podniesienia sprawności kontrolera.

Wykorzystane networking to RT6150 firmy Richtek [14]. Ich maksymalne napięcie wejściowe wynosi 5.5V, a maksymalne napięcie wyjściowe - 5.5V. Maksymalny prąd wyjściowy wynosi 800 mA. Dzielnik rezystorowy na wyjściu przetwornicy jest potrzebny do ustalenia napięcia wyjściowego dla wykorzystywanych układów.

5.8 Pomiar napięcia baterii pinem ADC

ADC voltage reading for battery



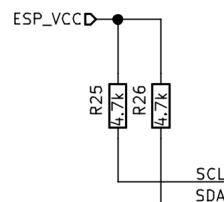
Rysunek 5.8: Obrazek przedstawiający wycinek schematu z pomiarem napięcia baterii przez ESP8266EX.

Źródło: Opracowanie własne

Zastosowano prosty dzielnic rezystorowy do pomiaru napięcia baterii przez ESP8266EX. Maksymalne napięcie wejściowe pinu ADC w tym mikrokontrolerze to 1V [2]. Zastosowane wartości to 100 kOhm i 37 kOhm.

5.9 Rezystory podciągające dla magistrali I²C

I2C interface pull-ups

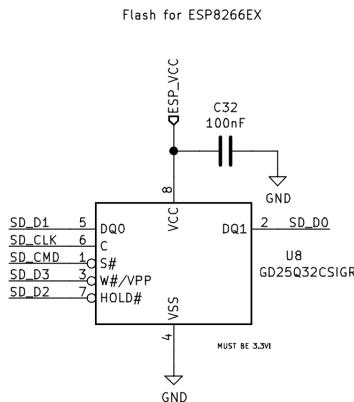


Rysunek 5.9: Obrazek przedstawiający wycinek schematu z rezystorami podciągającymi dla magistrali I²C

Źródło: Opracowanie własne

Zastosowano rezystory o wartości 2 kOhm jako podciągające do linii zasilania w celu zapewnienia maksymalnej możliwej prędkości transmisji danych w magistrali I²C.

5.10 Pamięć Flash dla ESP8266EX

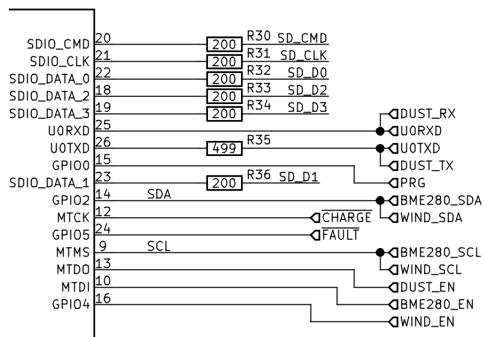


Rysunek 5.10: Obrazek przedstawiający wycinek schematu z pamięcią flash dla ESP8266EX

Źródło: Opracowanie własne

Podłączony został kondensator odsprzęgający o wartości 100nF dla zwiększenia stabilności jego działania. Komunikuje się po magistrali SPI z ESP8266EX. Wykorzystana pojemność to 32 MiB, czyli 4 MB. Umożliwia to napisanie rozbudowanego oprogramowania wewnętrznego do obsługi czujników oraz strony WWW.

5.11 Linie wejścia/wyjścia ESP8266EX



Rysunek 5.11: Obrazek przedstawiający wycinek schematu z komunikacją ESP8266EX z resztą płytka drukowanej.

Źródło: Opracowanie własne

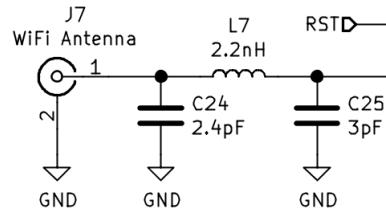
Wykorzystano trzy protokoły komunikacyjne - I²C dla anemometru i BME280, UART dla PMS7003 oraz SPI dla pamięci flash. Zastosowano rezystory terminacyjne 200 ohm dla linii SPI, a 499 ohm dla UART - zgodnie z zaleceniami z Hardware Design Guidelines [15].

Pamięć flash komunikuje się z ESP8266EX w trybie QIO, najszybszym z możliwych. Dostępne są również inne tryby, takie jak QOUT,DIO oraz DOUT.

Wyprowadzono również pin PRG, który umożliwia wprowadzenie ESP8266EX w tryb programowania pamięci flash.

Sygnały ~CHARGE i ~FAULT pochodzą z kontrolera ładowania akumulatora LT3652EDD.

5.12 Sieć dopasowująca dla anteny Wi-Fi

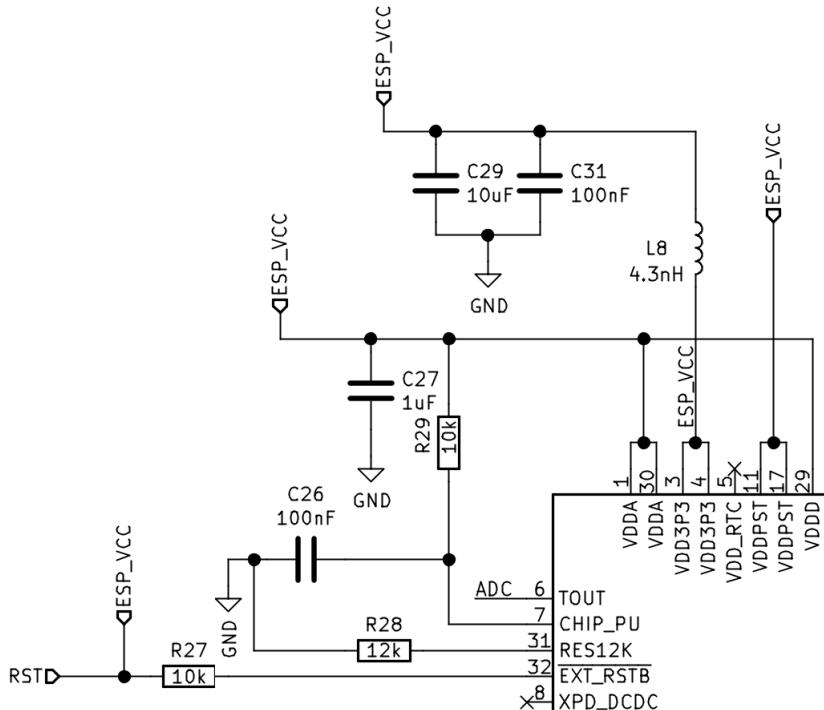


Rysunek 5.12: Obrazek przedstawiający wycinek schematu z siecią dopasowującą dla anteny Wi-Fi

Źródło: Opracowanie własne

Wartości elementów wykorzystano z tzw. Hardware Design Guidelines dla ESP8266EX [15] dla dopasowania anteny Wi-Fi do wykorzystanego układu. Wykorzystano konektor SMA dla anteny Wi-Fi dla możliwości podpięcia różnych anten, co daje więcej możliwości w zakresie regulacji zasięgu.

5.13 Filtracja zasilania ESP8266EX

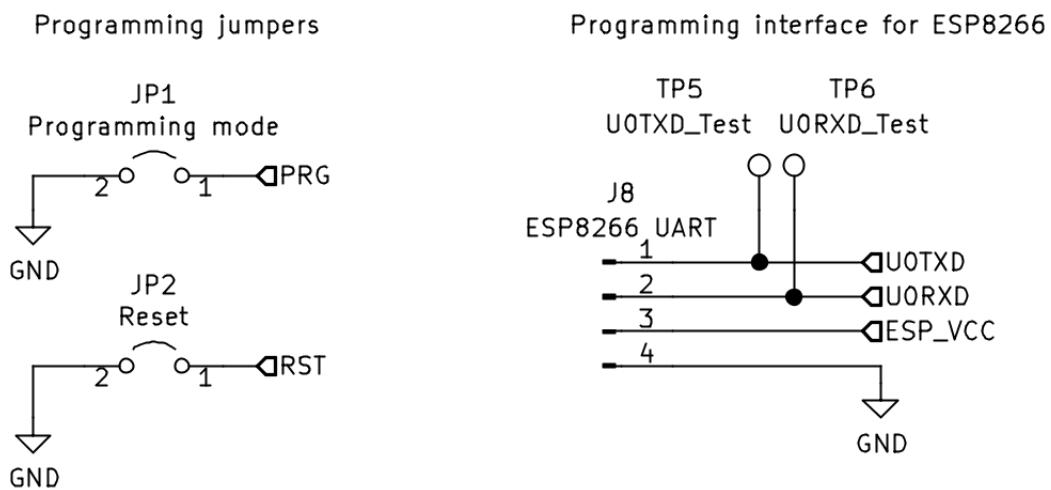


Rysunek 5.13: Obrazek przedstawiający wycinek schematu z filtracją zasilania dla ESP8266EX.

Źródło: Opracowanie własne

ESP8266EX posiada 3 różne stopnie zasilania. VDD3P3 wymaga 2 kondensatorów i cewki, VDDA i VDDD jedynie kondensator, a VDDPST - nic. Rezystor podłączony do linii VDDA służy do utrzymania ESP8266EX w stanie działania - jest to rezystor podciągający.

5.14 Piny programowania ESP8266



Rysunek 5.14: Obrazek przedstawiający wycinek schematu z pinami dla programowania ESP8266EX

Źródło: Opracowanie własne

Wykorzystano 2 zworki, jedna dla resetu ESP8266EX, druga dla przełączenia mikrokontrolera w tryb programowania.

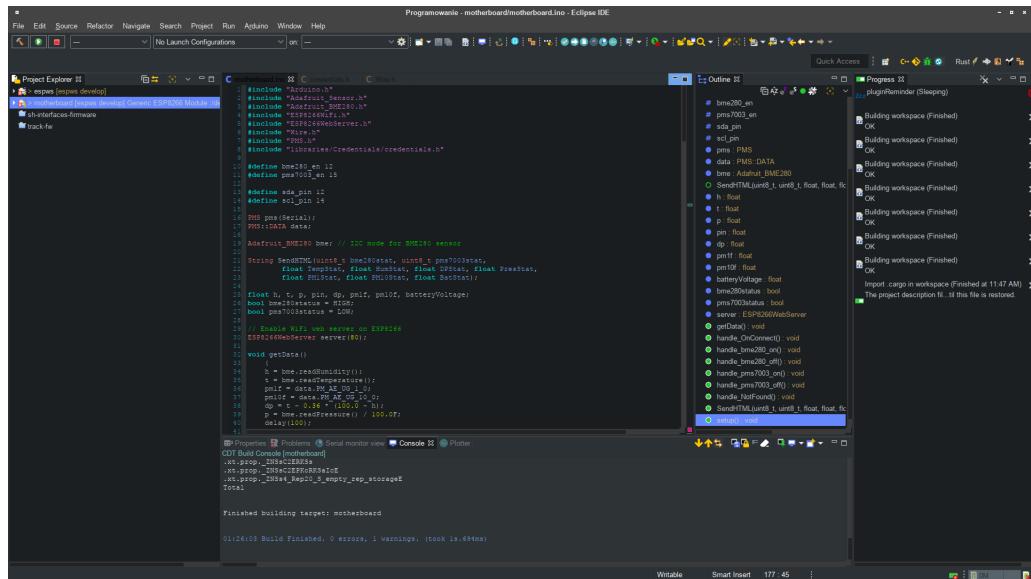
Zastosowano 4 pinowy konektor dla programowania ESP8266EX za pomocą magistrali UART, wykorzystując napięcie 3.3V.

Podpięto również 2 testowe pady dla inspekcji przebiegu programowania np. za pomocą analizatora stanów logicznych.

6. Kod źródłowy

6.1 Środowisko programistyczne

Wykorzystano środowisko programistyczne o nazwie Eclipse IDE. Doinstalowano do niego wtyczkę Sloeber dla możliwości programowania za pomocą bibliotek Arduino.



Rysunek 6.1: Obrazek przedstawiający okno programu Eclipse IDE, wykorzystanego do napisania programu.

źródło: Opracowanie własne

Eclipse IDE jest najpopularniejszym zintegrowanym środowiskiem programistycznym wśród programistów urządzeń wbudowanych czy też „Internetu Rzeczy”.

6.2 Kod przez rozpoczęciem

```
#include "Arduino.h"
#include "Adafruit_Sensor.h"
#include "Adafruit_BME280.h"
#include "ESP8266WiFi.h"
#include "ESP8266WebServer.h"
#include "Wire.h"
#include "PMS.h"
#include "libraries/Credentials/credentials.h"

#define bme280_en 12
```

```

#define pms7003_en 15

#define sda_pin 12
#define scl_pin 14

PMS pms(Serial);
PMS::DATA data;

Adafruit_BME280 bme; // I2C mode for BME280 sensor

String SendHTML(uint8_t bme280stat, uint8_t pms7003stat,
float TempStat, float HumStat, float DPStat, float PresStat,
float PM1Stat, float PM10Stat, float BatStat);

float h, t, p, pin, dp, pm1f, pm10f, batteryVoltage;
bool bme280status = HIGH;
bool pms7003status = LOW;

```

Zainicjalizowano niezbędne biblioteki do obsługi wykorzystanych sensorów oraz plik nagłówkowy z wypisanyem SSID sieci Wi-Fi oraz hasłem do niej.

```

// Enable WiFi web server on ESP8266
ESP8266WebServer server(80);

void getData()
{
    h = bme.readHumidity();
    t = bme.readTemperature();
    pm1f = data.PM_AE_UG_1_0;
    pm10f = data.PM_AE_UG_10_0;
    dp = t - 0.36 * (100.0 - h);
    p = bme.readPressure() / 100.0F;
    delay(100);
}

```

Serwer WWW zaczął działać, a w funkcji getData pobierane są pomiary z BME280 oraz PMS7003.

```

void handle_OnConnect()
{
    bme280status = HIGH;
    pms7003status = LOW;
    getData();
    server.send(200, "text/html",
SendHTML(bme280status, pms7003status, t, h, dp, p, pm1f,
pm10f, batteryVoltage));
}

void handle_bme280_on()
{
    bme280status = HIGH;
    server.send(200, "text/html",
SendHTML(true, pms7003status, t, h, dp, p, pm1f, pm10f,
batteryVoltage));
}

```

```

}

void handle_bme280_off()
{
    bme280status = LOW;
    server.send(200, "text/html",
    SendHTML(false, pms7003status, t, h, dp, p, pm1f, pm10f,
    batteryVoltage));
}

void handle_pms7003_on()
{
    pms7003status = HIGH;
    server.send(200, "text/html",
    SendHTML(bme280status, true, t, h, dp, p, pm1f, pm10f,
    batteryVoltage));
}

void handle_pms7003_off()
{
    pms7003status = LOW;
    server.send(200, "text/html",
    SendHTML(bme280status, false, t, h, dp, p, pm1f, pm10f,
    batteryVoltage));
}

void handle_NotFound()
{
    server.send(404, "text/plain", "Not found");
}

```

Występuje tutaj odświeżanie pomiarów oraz statusu pinów sterujących przetwornicami przy każdym odświeżeniu strony przez zmianę statusu pinu sterującego zasilaniem innych sekcji płytki. Ustawiono również stany początkowe pinów.

```

String SendHTML(uint8_t bme280stat, uint8_t pms7003stat,
float TempStat, float HumStat, float DPStat, float PresStat,
float PM1Stat, float PM10Stat, float BatStat)
{
    String ptr = "<!DOCTYPE html> <html>\n";
    ptr += "<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0,
    user-scalable=no\">\n";
    ptr += "<meta charset='utf-8' />";
    ptr += "<title>Stacja pogodowa </title>\n";
    ptr += "</head>\n";
    ptr += "<body>\n";
    ptr += "<h1>Stacja pogodowa oparta o ESP8266</h1>\n";
    ptr += "<h3>Tryb klienta.</h3>\n";
    ptr += "<table border=\"2\" width=\"456\" cellpadding=\"10\"><tbody>";

    ptr += "<tr><td>";
    ptr += "</h3><h3>Temperatura = ";
    ptr += "</td><td>";
    ptr += TempStat;

```

```

ptr += "&deg;C</td></tr>";

ptr += "<tr><td>";
ptr += "</h3><h3>ŚWILGOTNO =";
ptr += "</td><td>";
ptr += HumStat;
ptr += "%</td></tr>";

ptr += "<tr><td>";
ptr += "</h3><h3>PUNKT ROSY = ";
ptr += "</td><td>";
ptr += DPStat;
ptr += "&deg;C</td></tr>";

ptr += "<tr><td>";
ptr += "</h3><h3>ŚCINIENIE = ";
ptr += "</td><td>";
ptr += PresStat;
ptr += "hPa ";
ptr += "</td></tr>";

ptr += "<tr><td>";
ptr += "</h3><h3>PM 1.0 (ug/m3) = ";
ptr += "</td><td>";
ptr += PM1Stat;
ptr += "</td></tr>";

ptr += "<tr><td>";
ptr += "</h3><h3>PM 10 (ug/m3) = ";
ptr += "</td><td>";
ptr += PM10Stat;
ptr += "</td></tr>";

ptr += "<tr><td>";

ptr += "</h3><h3>NAPICIE BATERII =";
ptr += "</td><td>";
ptr += BatStat;
ptr += "</td></tr>";
ptr += "</tbody></table>";

if (bme280stat)
{
ptr +=
"<p>Stan BME280: ŁWY</p><a class=\"button button-off\""
    href="/bme280_off">ŁWY.</a>\n";
}
else
{
ptr +=

```

```

"<p>Stan BME280: ŁW</p><a class=\"button button-on\"
    href=\"/bme280_on\">ŁW.</a>\n";
}

if (pms7003stat)
{
ptr +=

"<p>Stan PMS7003 - ŁW.</p><a class=\"button button-off\"
    href=\"/pms7003_off\">ŁWY.</a>\n";
}
else
{
ptr +=

"<p>Stan PMS7003: ŁWY</p><a class=\"button button-on\"
    href=\"/pms7003_on\">ŁW.</a>\n";
}

ptr += "</body>\n";
ptr += "</html>\n";
return (ptr);
}

```

Znajduje się tutaj główne ciało strony wraz z polami, w których znajdują się wartości z pobranych sensorów oraz statusy pinów kontrolujących przetwornice.

6.3 Kod rozpoczętający

```

void setup()
{
Serial.begin(9600); // GPIO1, GPIO3 (TX/RX pin for PMS7003
Serial1.begin(9600); // GPIO2; for debugging purposes.
Wire.begin(sda_pin, scl_pin);
Wire.setClock(100000); // Set I2C clock.
pinMode(bme280_en, OUTPUT); // Enable pin for BME280
pinMode(pms7003_en, OUTPUT); // Enable pin for PMS7003
Serial1.println();
Serial1.print("Lacze sie do: ");
Serial.println(mySSID);
WiFi.begin(mySSID, myPassword); // Start Wi-Fi transmission.

while (WiFi.status() != WL_CONNECTED)
{
delay(500);
Serial1.print(".");
}
Serial1.println();
Serial1.println("Wi-Fi podlaczone!!!");
// Set multiple pages for toggling GPIOs

server.on("/", handle_OnConnect);

```

```
server.on("/bme280_on", handle_bme280_on);
server.on("/pms7003_on", handle_pms7003_on);
server.on("/bme280_off", handle_bme280_off);
server.on("/pms7003_off", handle_pms7003_off);
server.onNotFound(handle_NotFound);

// Start of web server
server.begin();
Serial1.println("Serwer WWW dziala - czekanie na IP dla ESP8266EX... ");
delay(5000);
// Printing ESP IP address
Serial1.println(WiFi.localIP());
Serial1.println(F("Test BME280"));

if (!bme.begin())
{
Serial1.println(
"Nie odnaleziono BME280, \u0142sprawd \u0142apoczenia sensor - \u0142pytka!");
while (1)
;
}
Serial1.println("BME280 dziala!");
}
```

Następuje tutaj ustawienie prędkości komunikacji ESP8266 z PMS7003, ustawienie pinów dla protokołu I²C, zmiana trybu działania pinów kontrolujących przetwornice na wyjścia, zestawienie wielu odnośników dla odświeżania i zmiany statusu pinów oraz drugi port UART dla celów diagnostycznych.

6.4 Kod w pętli

```
void loop()
{
server.handleClient();
getData();
int analogValue = analogRead(A0) / 1000; // Setup analog pin
batteryVoltage = (analogValue * 137) / 37;
if (bme280status)
{
digitalWrite(bme280_en, HIGH);
}
else
{
digitalWrite(bme280_en, LOW);
}

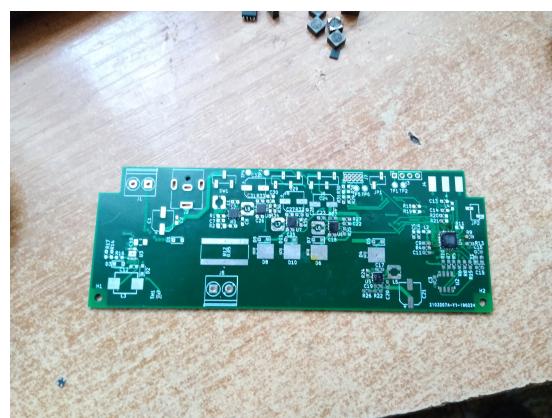
if (pms7003status)
{
digitalWrite(pms7003_en, HIGH);
}
else
{
digitalWrite(pms7003_en, LOW);
}
```

Dokonano tutaj utrzymania połączenia z klientem, pobór danych analogowych jak i cyfrowych oraz logika zmiany stanu pinów kontrolujących przetwornice DC-DC.

Kod zajął około 260 linijek, przez co jest dość rozbudowanym programem.

7. Infografika

7.1 Zdjęcia urządzenia



Rysunek 7.1: Zdjęcie przedstawiające płytę z przylutowanymi układami scalonymi

Źródło: *Opracowanie własne*



Rysunek 7.2: Zdjęcie przedstawiające płytę z przylutowanymi wszystkimi elementami. Do tego odwrotnie przylutowany kondensator, który naprawiono.

Źródło: *Opracowanie własne*



Rysunek 7.3: Zdjęcie przedstawiające wnętrze projektu z przykręconym akumulatorem

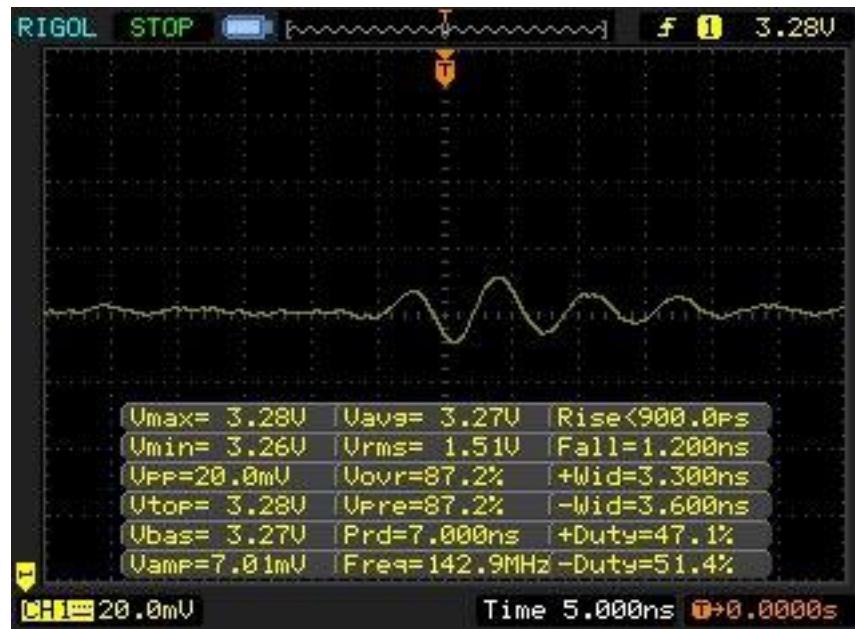
Źródło: Opracowanie własne



Rysunek 7.4: Zdjęcie przedstawiające część zewnętrzną projektu, złożoną i gotową do pracy.

Źródło: Opracowanie własne

7.2 Pomiar sekcji zasilania ESP8266EX



Rysunek 7.5: Zdjęcie przedstawiające pomiar linii zasilania ESP8266

Źródło: Opracowanie własne

Wykorzystana przetwornica DC-DC RT6150 dała znakomite rezultaty jeśli chodzi o zasilanie projektu. Pomiar szumu pokazał, że jest on marginalny i nie wpływa na działanie całości. Zasilanie za pomocą tychże układów dało stabilne rezultaty.



Rysunek 7.6: Zdjęcie przedstawiające stronę główną aplikacji.

Źródło: Opracowanie własne

8. Streszczenie

Wykorzystano do projektu mikrokontroler ESP8266EX, czujnik cząsteczek stałych PMS7003 oraz sensor temperatury, wilgoci oraz ciśnienia BME280. Wyrowadzono również k넥ektor dla czujnika pr dko ci i kierunku wiatru dla przysz ej rozbudowy. BME280 wykorzystuje protok l I C, a PMS7003 - UART. Anemometr powinien by  zasilany napi ciem 3.3V oraz wykorzystywa  protok l I C do komunikacji z mikroprocesorem.

Zaprojektowana i wykonana zosta a p ytka drukowana (PCB). Zaprojektowana zosta a za pomoc  KiCAD [12], a wykonana przez firm  JLCPCB - jlcpcb.com w cenie 200 PLN, lic ac przesy k  firm  kuriersk  UPS. Elementy zam wiono na stronie DigiKey - digkey.pl, kt re przysz y z USA firm  kuriersk  UPS.

Oprogramowanie zawiera obsług  czujnika PMS7003, BME280 oraz kontrol  przetwornicami DC-DC RT6150 firmy Richtek dla lepszego zar dzania energ i  ca ego uk adu. Wykorzystano r wnie  pin analogowy ESP8266EX dla pomiaru stanu baterii przez dzielniczk  rezystorowy.

Uk ad mo e by  zasilany na 2 sposoby - z zewn trznego zasilacza o napi ciu 9V i pr adzie 3A oraz za pomoc  baterii LiFePO4 o napi ciu nominalnym 3.2V. Akumulator mo na  adowa  panelem s onecznym o napi ciu 12V podczas obci zenia oraz zewn trznym, 9V 3A zasilaczem. Istnieje mo liwo c wyboru, czy zewn trzny zasilacz ma  adowa  akumulator, czy zasila  ca y uk ad.

Do programowania ESP8266EX wykorzystywany jest protok l UART oraz napi cie zasilania o wartości 3.3V. Domys lna pr dko c komunikacji ESP8266EX ze „swiatem” wynosi 74880 bodów dla rezonatora kwarcowego 26 MHz. Dla 40 MHz rezonatora ta pr dko c wynosi 115200 bodów. ESP8266EX obs uguje  aczno c Wi-Fi w trybach b/g/n - jako punkt dost pu, klient oraz tryb pod sluchu.

ESP8266EX nie posiada wbudowanej pami ci flash dla przechowywania programów - niezb edna jest zewn trzna pami c. Jego odmiana, ESP8285 - posiada wbudowane 1 MiB pami ci typu flash. Wykorzystano zintegrowane s rodowisko programistyczne Eclipse IDE oraz biblioteki Arduino.

Spis rysunków

2.1	ESP8266EX	5
2.2	ESP8266EX - opis wyprowadzeń	6
2.3	BME280	10
2.4	BME280 - Moduł	10
2.5	PMS7003	12
2.6	PMS7003 - pinout	13
3.1	I ² C - Struktura 10-bitowego adresu	16
3.2	I ² C - Diagram czasowy	17
3.3	UART - Diagram czasowy	18
4.1	Schemat blokowy	20
5.1	Schemat - ogólny przegląd	21
5.2	Schemat - konektor panelu słonecznego	22
5.3	Schemat - konektor zewnętrznego ładowania/zasilania	22
5.4	Schemat - ładowanie akumulatora	23
5.5	Schemat - konwersja do 5.6V	24
5.6	Schemat - konektory dla sensorów	25
5.7	Schemat - układ dystrybucji zasilania	26
5.8	Schemat - sekcja ADC	27
5.9	Schemat - rezystory interfejsu I ² C	27
5.10	Schemat - pamięć flash	28
5.11	Schemat - komunikacja ESP8266EX z resztą płytki	28
5.12	Schemat - sieć dopasowania dla anteny	29
5.13	Schemat - filtracja zasilania ESP8266EX	29
5.14	Schemat - programowanie wewnętrzukładowe	30
6.1	IDE - Okno programu	31
7.1	Zdjęcie - układy scalone	38
7.2	Zdjęcie - ukończona płytka	38
7.3	Zdjęcie - wnętrze projektu	39
7.4	Zdjęcie - wygląd projektu	39
7.5	Pomiar - zasilanie ESP8266	40
7.6	Aplikacja - strona główna	40

Spis tabel

2.1	ESP8266EX - tryby uruchamiania	7
2.2	ESP8266EX - kanały Wi-Fi	8
2.3	ESP8266EX - pobór mocy przy łączności Wi-Fi	9
2.4	PMS7003 - Lista wyprowadzeń	13
3.1	I ² C - Stany zajętości magistrali	15
3.2	I ² C - Stany magistrali przy przesyłaniu danych	15
3.3	I ² C - Struktura 7-bitowego adresu	16
5.1	LT3652 - Sygnały statusu	23

Bibliografia

- [1] „ESP8266”, *Wikipedia*, Zebrane 3 marca 2019, <https://en.wikipedia.org/wiki/ESP8266>
- [2] Nota katalogowa ESP8266, *Espressif*, Zebrane 25 lutego 2019, https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf
- [3] Zdjęcie ESP8266EX, *alphamicrowireless.com* Zebrane 3 marca 2019, http://www.alphamicrowireless.com/media/562039/esp8266ex_370px.gif
- [4] Pinout ESP8266EX, *acrobotic.com*, Zebrane 3 marca 2019, https://learn.acrobotic.com/uploads/esp8266_pinout.png
- [5] Nota katalogowa BME280, *Bosch*, Zebrane 25 lutego 2019, https://ae-bst.resource.bosch.com/media/_tech/media/datasheets/BST-BME280-DS002.pdf
- [6] Zdjęcie przedstawiające układ BME280, *dnatechindia.com* Zebrane 3 marca 2019, <http://www.dnatechindia.com/image/cache/catalog/bme280%201-500x500.jpg>
- [7] Zdjęcie przedstawiające gotowy moduł z układem BME280, *gunook.com* Zebrane 3 marca 2019, <http://img.gunook.com/upload/a/1d/a1d0568812635b491d33f680db52a587.jpg>
- [8] Przetłumaczona nota katalogowa PMS7003, *github.com* Zebrane 26 lutego 2019, <https://raw.githubusercontent.com/eleparts/PMS7003/master/data%20sheet/PMS7003%20datasheet.pdf>
- [9] Zdjęcie sensora PMS7003, *amazon.com* Zebrane 3 marca 2019, https://images-na.ssl-images-amazon.com/images/I/41KI%2BnbtAGL._SX342_.jpg
- [10] „I²C”, *Wikipedia* Zebrane 3 marca 2019, <https://en.wikipedia.org/wiki/I%C2%B2C>
- [11] „Universal asynchronous receiver-transmitter”, *Wikipedia* Zebrane 3 marca 2019, https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter
- [12] *KiCAD EDA*, Zebrane 3 marca 2019, <http://kicad-pcb.org/>
- [13] Nota katalogowa LT3652, *Linear Technology*, Zebrane 3 marca 2019, <https://www.analog.com/media/en/technical-documentation/data-sheets/3652fe.pdf>
- [14] Nota katalogowa RT6150, *Richtek* Zebrane 3 marca 2019, https://www.richtek.com/assets/product_file/RT6150A=RT6150B/DS6150AB-05.pdf
- [15] Zalecenia dla projektowania dla ESP8266EX, *Espressif* Zebrane 3 marca 2019, https://www.espressif.com/sites/default/files/documentation/esp8266_hardware_design_guidelines_en.pdf