|  |  |
| --- | --- |
| **Politechnika Śląska** | **Gliwice** |
| **Wydział Automatyki Elektroniki i Informatyki**  **Kierunek: Automatyka i Robotyka sem. 5** | **Rok akademicki 2019/2020** |
|  | **Semestr zimowy** |

Systemy Mikroprocesorowe

**Domowe wielofunkcyjne urządzenie komunikujące się głosowo**



Autorzy: Dawid Mudry

Karol Marcinak

Grupa 1 Sekcja 2

Kierujący pracą: dr inż. Andrzej Kozyra

Spis treści

[1. Wstęp 3](file:///C:\Users\Acer\Desktop\repoGitHub\SM\example1.docx#_Toc282945161)

[2. Cel i zakres projektu 3](file:///C:\Users\Acer\Desktop\repoGitHub\SM\example1.docx#_Toc282945162)

[3. Kosztorys 0](file:///C:\Users\Acer\Desktop\repoGitHub\SM\example1.docx#_Toc282945177)

[4. Harmonogram 4](file:///C:\Users\Acer\Desktop\repoGitHub\SM\example1.docx#_Toc282945163)

[5. Aplikacja sprzętowa 4](file:///C:\Users\Acer\Desktop\repoGitHub\SM\example1.docx#_Toc282945164)

[5.1. Opis części sprzętowej 4](file:///C:\Users\Acer\Desktop\repoGitHub\SM\example1.docx#_Toc282945165)

[5.2. Schemat ideowy oraz płytka drukowana 6](file:///C:\Users\Acer\Desktop\repoGitHub\SM\example1.docx#_Toc282945169)

[5.3. Problemy w trakcie realizacji 6](file:///C:\Users\Acer\Desktop\repoGitHub\SM\example1.docx#_Toc282945169)

[6. Aplikacja programowa 7](file:///C:\Users\Acer\Desktop\repoGitHub\SM\example1.docx#_Toc282945171)

[6.1. Opis części programowej 7](file:///C:\Users\Acer\Desktop\repoGitHub\SM\example1.docx#_Toc282945172)

[6.2. Schemat blokowy 11](file:///C:\Users\Acer\Desktop\repoGitHub\SM\example1.docx#_Toc282945173)

[6.3. Problemy w trakcie realizacji 11](file:///C:\Users\Acer\Desktop\repoGitHub\SM\example1.docx#_Toc282945174)

[7. Podsumowanie 21](file:///C:\Users\Acer\Desktop\repoGitHub\SM\example1.docx#_Toc282945178)

**1. Wstęp**

W dzisiejszych czasach coraz więcej urządzeń zaczyna komunikować się z ludźmi za pomocą dźwięku przypominającego ludzki głos. Rozwiązania takie stosowane są abyśmy mniej czasu musieli poświęcać na samodzielne sprawdzanie wiadomości, potrzebnych informacji czy docierających do nas komunikatów. Systemy audio obecne są w naszych domach od dziesiątek lat, głównie stosowane do odtwarzania muzyki, audycji radiowych, ale coraz częściej w celu ułatwiania nam codziennego życia. Najnowocześniejsze systemy (oprogramowania) połączone z systemem audio potrafią odtwarzać spersonalizowane wiadomości informujące nas o naszych planach np. rozkładzie zajęć, wizycie u lekarza czy spotkaniu ze znajomymi, poza tym podając podstawowe informacje np. temperaturę i ciśnienie czy też pogodę na drugim końcu świata. Zastosowanie takich „inteligentnych” urządzeń pozwala zaoszczędzić nam czas, głownie dzięki braku konieczności załączania telefonu, bądź komputera i przedzieraniu się przez nadmiar informacji w poszukiwaniu jednej konkretnej. Takie rozwiązania szczególne mogą być pomocne dla osób starszych, którym ten proces zajmuje więcej czasu niż przeciętnemu młodemu człowiekowi oraz osobom niedowidzącym, dla których komunikacja głosowa jest podstawowym sposobem porozumiewania się. Realizacja takich projektów możliwa jest dzięki odpowiedniemu oprogramowaniu, systemie audio, takim aby komunikaty były przejrzyste (niezakłócone), podstawowym czujnikom, podającym wszelaki dane np. o pogodzie. Sercem takiego układu jest mikrokontroler, przetwarzający wszystkie sygnały zarówno wychodzące jak i przychodzące z „zewnątrz”.

**2. Cel i zakres projektu**

Celem naszego projektu jest zbudowanie układu, który przy użyciu odpowiedniego sprzętu i zestawu czujników informował będzie o stanie bieżącym w danym pomieszczeniu za pomocą głośnika z odpowiednim układem wzmacniającym i filtrującym. Dane o otoczeniu zbierane będą za pomocą podstawowych czujników typu: czujnik temperatury, wilgotności, ciśnienia. Dane te będą przetwarzane za pomocą mikrokontrolera ATMega, odpowiednio rozpoznawane oraz przesyłane na stronę internetową, która jest integralną częścią tego projektu. Dźwięki wydobywające się z głośnika tworzone będą przy pomocy zaprogramowanego syntezatora mowy. Komunikacja kontrolera ze stroną internetową będzie odbywać się w sposób bezprzewodowy za pomocą modułu WiFi. Mikrokontroler ATMega oraz moduł WiFi będą miały wspólny interfejs komunikacyjny, który pozwoli im wymieniać ze sobą informacje. Celem wysyłania danych na stronę , będzie możliwość ich późniejszego przeanalizowania na podstawie wygenerowanych wykresów oraz zostanie zaimplementowana funkcja podglądu tych danych na bieżąco. Głównym celem projektu jest skupienie się na poprawnej budowie układu z głośnikiem, tak aby komunikaty głosowe były przejrzyste tzn. nie zawierały żadnych szumów czy innych „niechcianych” sygnałów oraz zaprogramowanie mikrokontrolera w taki sposób, aby w całym układzie nie zachodziły opóźnienia, tj. całość działała w sposób płynny. Sygnały dźwiękowe wymuszane będą za pomocą fizycznych przycisków, do których zaprogramowania zostaną użyte przerwania systemowe. Projekt pod względem technicznym będzie obejmował zaprojektowanie i budowę własnej płytki PCB - drukowanie mozaiki ścieżek na papierze kredowym, przeprasowanie jej na warstwę miedzi, trawienie w kwasie oraz ostatecznie przylutowaniu na niej wszystkich komponentów potrzebnych do realizacji projektu. W skład tych komponentów będzie wchodził mikroprocesor ATMega328p będący sercem układu, zestaw czujników składających się na domową „stację pomiarową” , wśród nich pojawią się: czujnik temperatury i wilgotności DHT11 o wystarczającym zakresie w warunkach domowych, czujnik barometryczny BMP180. Czujniki te zostaną odpowiednio przylutowane przy użyciu lutownicy wraz z odpowiednio dobranym układem rezystorów (4,6kOhm). Wyjściem układu będzie głośnik, do którego zostanie doprowadzony układ wzmacniający zbudowany przy użyciu tranzystora oraz układu cewek. Układ ten w dodatku będzie odpowiednio filtrować sygnał w celu redukcji wszelakich szumów oraz zmiany sygnału PWM na sygnał analogowy(sinusoidalny). Wymiana informacji z/do układu zostanie przeprowadzona przy użyciu modułu WiFi ESP8266, który zostanie połączony ze stroną internetową, na którą będą przesyłane dane o temperaturze, wilgoci i ciśnieniu. Całość zostanie zaprogramowana w środowisku ArduinoIDE, przy użyciu odpowiednich bibliotek pozwalających obsłużenie ww. czujników i modułów. Wywołanie odpowiednich komunikatów głosowych realizowane będzie przez naciśnięcie odpowiedniego fizycznego przycisku przylutowanego do płytki PCB. Poprawność wywołania danego sygnału i aktywności (tzn. wytwarzanie sygnału dźwiękowego) głośnika sygnalizowana będzie przy użyciu paru diod LED przylutowanych wraz z rezystorami. Sygnały dźwiękowe generowane będą przez mikrokontroler, za pomocą zaprogramowanego syntezatora mowy. W celu uzyskania odpowiedniego napięcia zasilającego moduł WiFi użyty zostanie stabilizator 3.3V

**3. Kosztorys**

ATmega328p-pu : 8 zł

Laminat PCB: 4 zł

Papier kredowy: 0.30 zł

Rezystory: 1.5 zł

Kondensatory elektrolityczne/ceramiczne: 3 zł

Osylatory 16 Mhz: 0,7 zł

Diody LED: 1 zł

Głośnik 8 Ohm 0.5 W: 3 zł

Tranzystor: 1.5 zł

Przyciski : 1 zł

Czujnik DHT11: 3 zł

Czujnik BMP180: 3 zł

Czujnik MQ7: 5 zł

Moduł wifi ESP: 8 zł

Port MicroUsb: 3zł

Stabilizator 3V3: 3zł

Zasilacz 5V: 0.01 zł

Obudowa: 4 zł

Konektory żeńskie: 1 zł

**Suma: 54.01 zł**

**4.Harmonogram**

**21.10.19.**

Przedstawienie projektu. Określenie jego koncepcji, dyskusja na temat możliwych zastosowanych rozwiązaniach. Spis potrzebnych elementów do budowy układu.

**22.10.19 - 04.11.19**

Pisanie oprogramowania umożliwiającego obsługę wszystkich zastosowanych czujników oraz elementów wyjściowych. Prototyp hardware'u, zbudowany na płytce stykowej, czyli przyłączenie wszystkich zastosowanych w projekcie czujników, wstępna budowa układu głośnika z wzmacniaczem. Wstępne testowanie.

**05.11.19 - 25.11.19**

Stworzenie płytki PCB, do której zostanie przylutowane większość elementów wraz z oprogramowanym kontrolerem. Optymalizacja kodu programu w celu zmniejszenia zajmowanej przez niego pamięci i płynniejszego działania układu.

**26.11.19 - 16.12.19:**

Testy modułu WiFi – przesyłanie danych do/z komputera (strony internetowej). Debbugowanie oprogramowania, w celu znalezienia nieoczekiwanych błędów i zwiększenia płynności. Test całego układu, sprawdzania odpowiedzi na różne sygnały

**5.Aplikacja sprzętowa**

**5.1 Opis części sprzętowej**

Centralnym elementem naszego projektu jest mikroprocesor ATMega328P-PU. Jest to 16 bitowy procesor zawierający 16 KB pamięci Flash oraz posiadającym 28 linie I/O. W naszym układzie jest on taktowany zewnętrznym oscylatorem o częstotliwości 16 MHz.

Warstwa czujników składa się z 3 układów, odpowiednio MQ-7,BMP-180, DHT-11.

Czujnik MQ-7 jest połączony do napięcia zasilania 5 oraz komunikuje się z procesorem za pomocą pin numer 25.

Czujnik BMP-180 jest połączony do napięcia zasilania 3,3V oraz komunikuje się z procesorem za pomocą interfejsu I2C.

Czujnik DHT-11 jest połączony do napięcia zasilania 3,3V oraz komunikuje się z procesorem za pomocą pin numer 12

Napięcie zasilania jest dostarczane z zasilacza 5V, 2,5A. przez port microUSB typu B, napięcie zasilanie 3,3V dostajemy używając stabilizatora napięcia LD1117 oraz dwóch kondensatorów 100nF oraz 22uF połączonych według dokumentacji produktu dostarczonej przez producenta elementu. Praca urządzenia jest sygnalizowana przez czerwona diodę LED, która jest podłączona do napięcia zasilania 5V przez rezystor 1k Ohm.

Głównym elementem obwodu głośnika jest tranzystor BD137, który wzmacnia sygnał PWM wystawiany przez procesor ATmega328P na pinie numer 5, sygnał ten jest filtrowany przez 47uF kondensator, baza tranzystora jest połączona z kolektorem za pomocą 2k Ohm rezystora.

Programowanie ATmegi328P odbywa się za pomocą interfejsu UART, aby zaprogramować procesor korzystamy z gotowej przejściówki FT232RL microUSB, na płytce PCB znajduje się konektor cztero-pinowy odpowiednio RX, TX, RST, GND. Do Sygnału RST jest dołączony obwód składający się kondensatora 100nF i rezystora 10k.

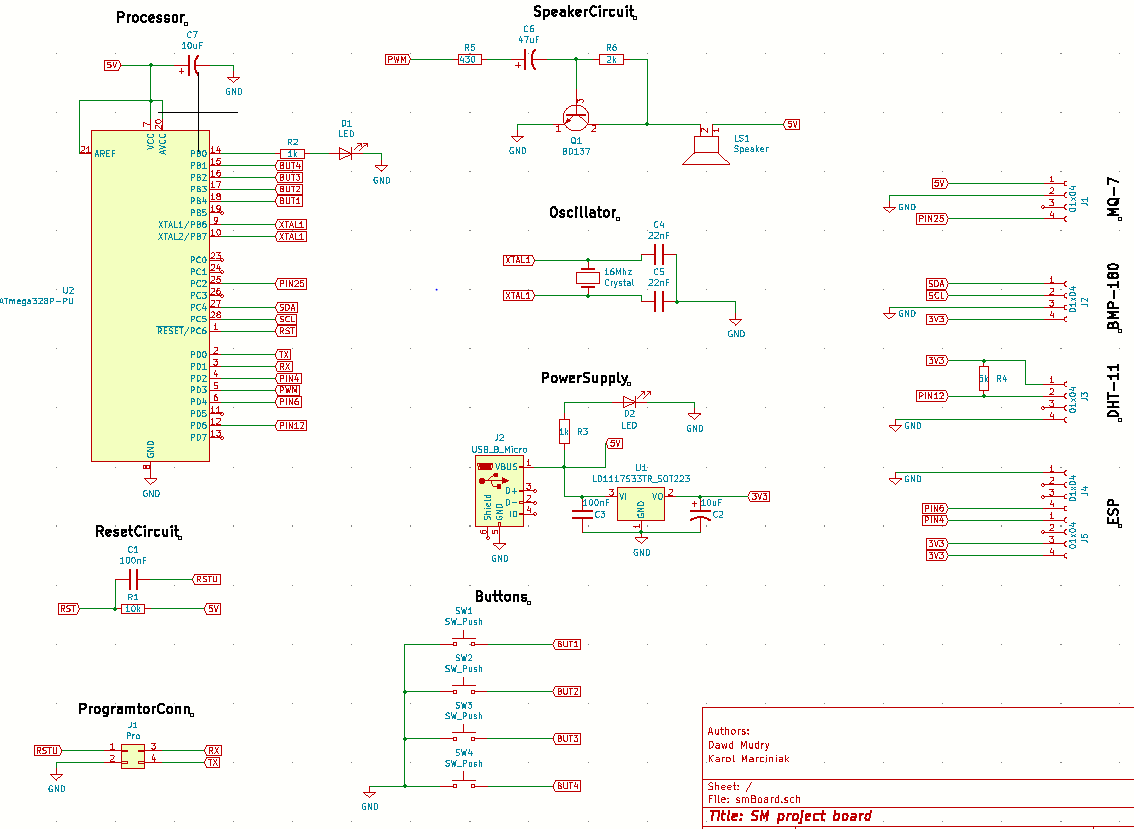
Procesor ATmega328P komunikuje się z modułem WiFi ESP8266 za pomocą interfejsuUART, procesory dzielą ze sobą dwie linie podłączone do pinów 4 oraz 6 odpowiednio RX i TX. ESP8266 wysyła informacje na stronę serwer www za pomocą wbudowanej anteny WiFi.

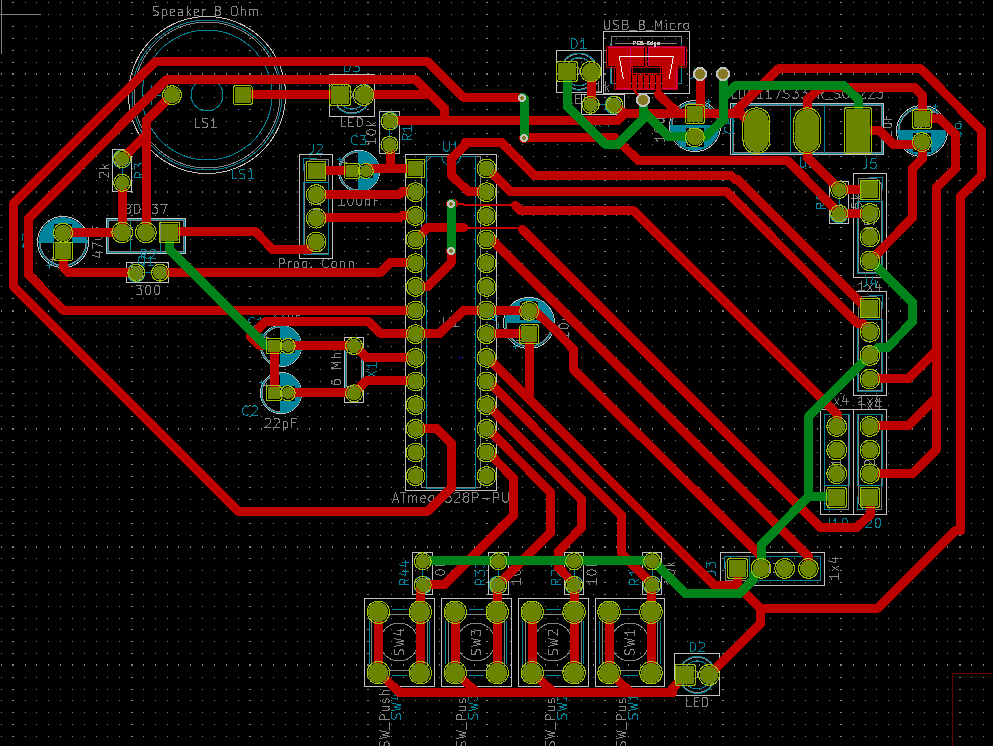
Do płytki PCB dołączone są cztery przyciski łączące masę oraz piny od 15-18, każdy przycisk aktywuje odpowiedni pomiar danej wielkości wywołanie pomiaru odbywa się za pomocą przerwań, a wciśniecie przycisku jest informowane za pomocą żółtej diody LED która jest połączona bezpośrednio do pinu numer 14, prąd płynący przez diodę jest ograniczony za pomocą 1k Ohm rezystora.

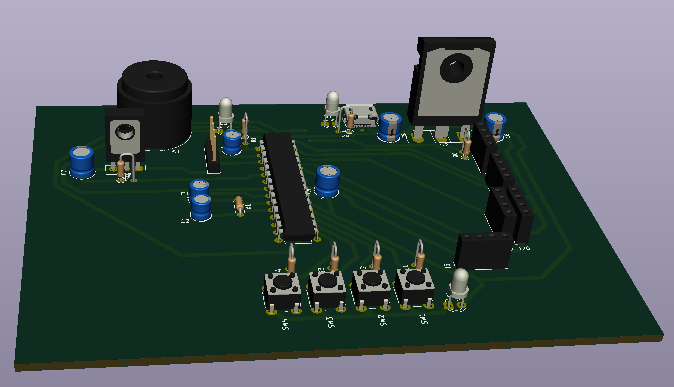
**5.1 Schemat ideowy oraz płytka drukowana**

Schemat ideowy oraz schematpłytki PCB został wykonany za pomocą oprogramowania KiCad. Schemat PCB został wydrukowany na drukarce laserowej firmy HP, następnie wydruk został przetransportowany na laminat za pomocą techniki transferu chemicznego użyliśmy acetonu. Wytrawianie płytki odbywało się w kwasie siarkowym. Po wytrawieniu płytka była sprawdzona pod możliwością zwarcia ścieżek i przeczyszczona.

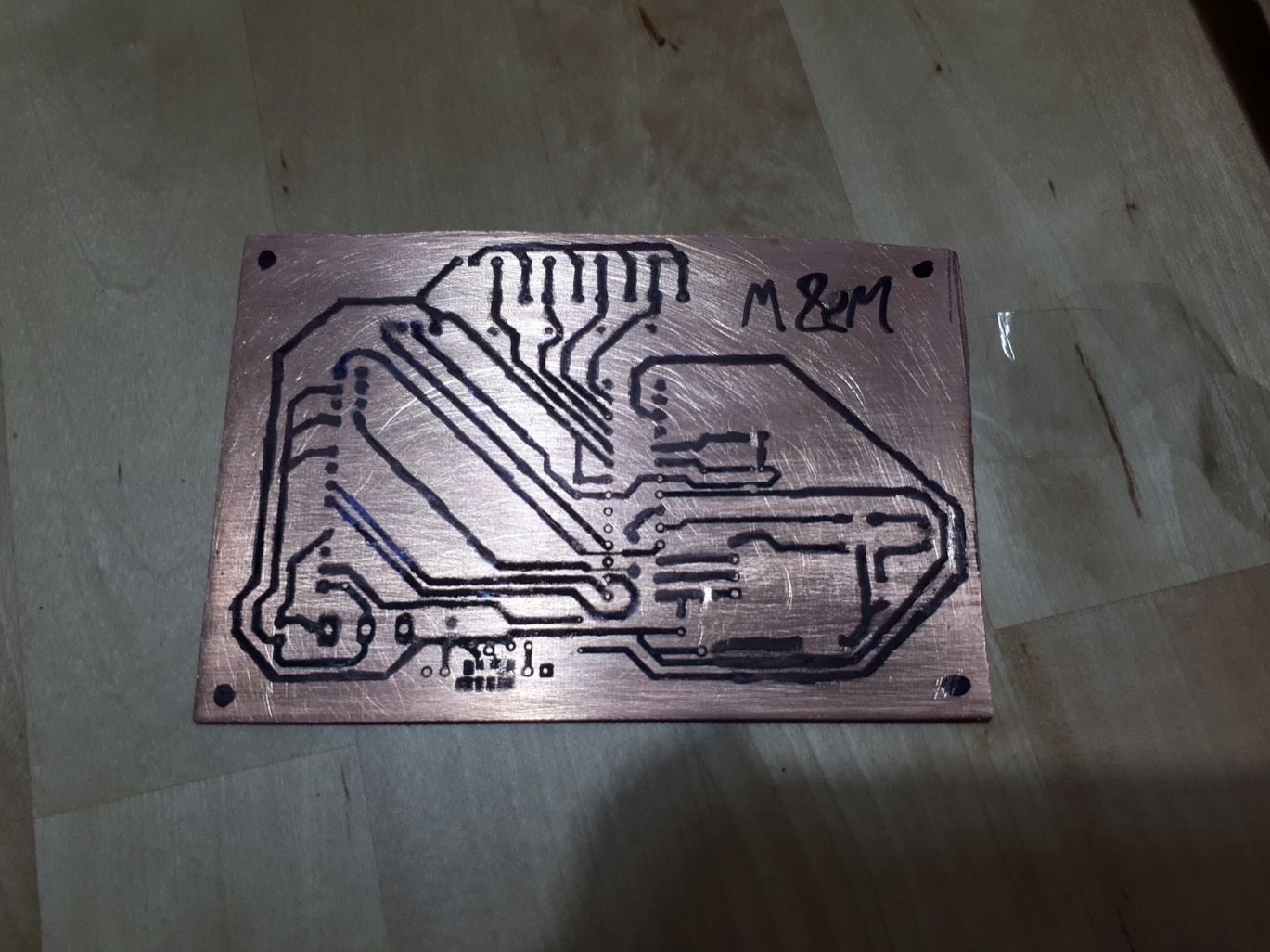
*Schemat Ideowy*

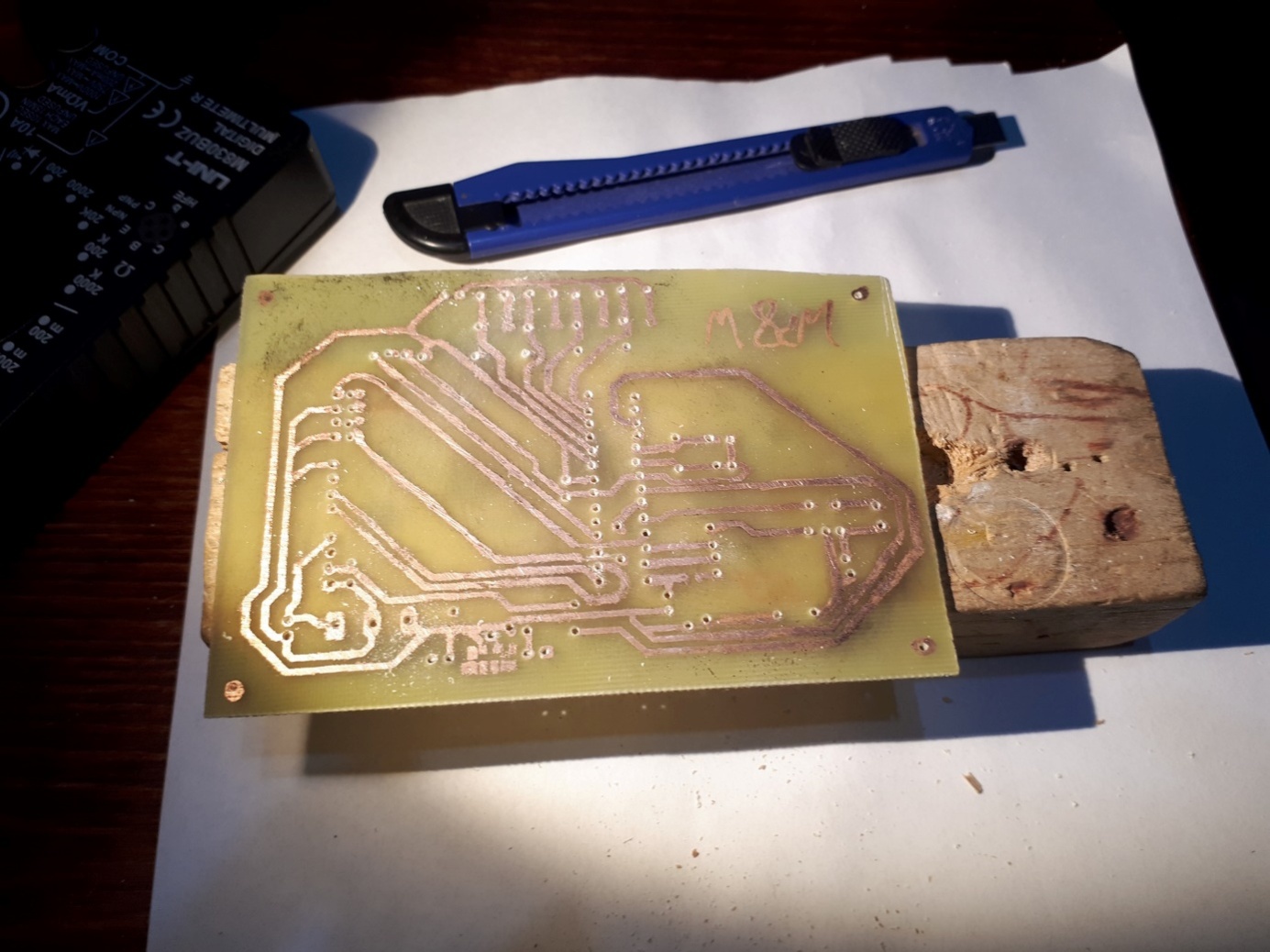


 *Schemat PCB*

 *Podgląd 3D*

*Laminat przed wytrawieniem*

**

* Laminat po wytrawieniu*

 *Efekt końcowy bez obudowy*

 *Efekt Koncowy w obudowie*

**5.3 Problemy napotkane podczas realizacji**

Podczas przenoszenia wydruku na płytkę wiele razy zakończyło się to niepowodzeniem, dopiero za 5 próbą oraz poprawienie mazakiem ścieżek udało się wytrawić płytkę.

Problem z przylutowaniem procesora do płytki, schemat został źle wydrukowany przez to wyjścia I/O nie zgadzały się z płytką PCB, rozwiązaliśmy ten problem przylutowując procesor po stronie ścieżek.

Problem z połączeniem masy na płytce, wynikający z jedno warstwowego laminatu. Problem został rozwiązany za pomocą taśmy izolacyjnej i kilku kabelków.

Problem ze zbyt dużą ilością prądu płynącą z pinu PWM. Problem został rozwiązany przez zwiększenie rezystora z 215 Ohm na 430 Ohm.

Problem z pomyleniem pinów w czujniku BMP-180 przez co cały program zawieszał się. Problem został rozwiązany po 2h obróceniem o 180 stopni czujnika.

Problem ze źle przylutowanym konektorem do czujnika DHT-11. Rozwiązaniem problemu było szybkie poprawienie lutu.

**6.Aplikacja Programowa**

**6.1 Opis części programowej**

Pierwszym etapem programu jest załączenie wszystkich koniecznych bibliotek, stworzenie obiektów oraz koniecznych zmiennych globalnych, w kolejnym etapie wykonuje się pętla setup(). Tworzone inicjalizowane są tam wszystkie używane piny oraz tworzone prototypy funkcji przerwań. Najważniejszym etapem tej pętli jest tworzenie komunikacji pomiędzy procesorem atmega a modułem wifi esp8266. Następnie moduł ten jest konfigurowany, tzn. ustawiany jest tryb pracy – tryb klienta i sposób obsługi połączeń – tryb wielo połączeniowy. Na koniec moduł łączy się z wybraną siecią wifi. Całość inicjalizacji esp realizowana jest za pomocą komend AT, ich poprawną interpretację zapewnia wgrany bezpośrednio na moduł firmware.

Kolejnym etapem jest główna funkcja programu czyli funkcja loop(), na początku pętli odczytujemy czas, który minął od podłączenia urządzenia za pomocą wbudowanej funkcji millis(). Następnie odczytywane są wartości ze wszystkich podłączonych czujników. W przypadku czujnika DHT odczyt zachodzi jedynie w przypadku gdy wartość jest różna od 0, warunek taki jest bezpieczny gdyż zakres tego czujnika wynosi 5-95%, a wartość zero zwraca w przypadku błędu. W przypadku czujnika czadu MQ-7 odczytywana jest wartość analogowa, zostaje ona następnie przeliczona na „prawdziwą” wartość, część wzorów użyta do tego wynika z jego charakterystyki dostarczonej od producenta i jego wcześniejszej przybliżonej „kalibracji”.

Następnym etapem programu jest sprawdzenie flag z przerwań. W czasie naciśnięcia przycisku zostaje ustawiona flaga w postaci zmiennej boolean na wartość „true” oraz zapalona dioda. Jeżeli wartość wynosi „true” to w głównej pętli programu zostanie wywołana funkcja sayNumber(), zgaszona dioda oraz wyzerowana flaga. Postawiono na takie rozwiązanie gdyż wywołanie funkcji sayNumber() w przerwaniu powodowało wywoływanie przerwania aż do zawieszenia się programu. Wynika to najprawdopodobniej z faktu, ze w bibliotece syntezatora mowy funkcja sayNumber() również wywołuje przerwanie. W samym programie funkcja ta działa w taki sposób, że przyjmuje wartość long int, następnie sprawdza ją, jeżeli jest mniejsza od 0 to dodaje minus jako wymawiane słowo, jeżeli równa zero, to wymawia zero, a jeżeli większa od zera to dzieli tą liczbę na tysiące, setki, dziesiątki i cyfry z przedziału <0,19>, zależnie od jej wielkości, a następnie za pomocą syntezatora mowy generuje odpowiedni sygnał PWM i przekazuje go do głośnika. Obsługa liczb z zakresu <-9999, 9999> została zaimplementowana na początku programu.

Końcowym etapem głównej funkcji programu – loop() jest sprawdzenie jaki czas upłynął od ostatniego wysłania danych do modułu esp. Dane zależnie od czujnika wysyłane są co 61,62,63 skeundy. W każdym warunku sprawdzany jest ten czas, przypisywany nowy czas do zmiennych, a następnie konwertowanie są dane z czujników na zmienne typu String. Dane te są argumentami w funkcjach wysyłania, odpowiednio updateDHT(), updateBMP(), updateMQ(). W funkcjach tych tworzona jest odpowiednia komenda zgodna z protokołem HTTP, następnie za pomocą komend AT:  
 – rozpoczęcie połączenia z serwerem - podawane jest IP, rodzaj połączenia (TCP), kanał oraz port   
– inicjalizacja wysłania danych - podawany jest kanał oraz rozmiar wcześniej przygotowanej komendy w bajtach, odczytywany za pomocą funkcji length()

– wysłanie komendy  
– zamknięcie połączenia

Po wysłaniu danych, program powraca do pętli głównej i kolejne wartości wysyłane są po minucie. Dane na serwerze odczytywane są poprzez „łapanie” tych requestów, interpretowanie ich, a następnie wstawiane są wartości do lokalnej bazy danych. Kod przechwytujący został napisany w języku PHP. Dzięki tym danym możliwy jest podgląd wartości „na żywo” na stronie oraz analizowanie wartości na podstawie generowanych wykresów kolumnowych i liniowych.

**6.2 Schemat blokowy**

**6.3 Problemy w trakcie realizacji**

**-** Błędne wartości odczytywane z czujnika DHT – pomogło przeczytanie dokumentacji technicznej, zgodnie z którą czujnik ten daje poprawne odczyty raz na sekundę, a w przypadku błędnych wystawia 0. Wystarczył warunek if, który odrzuca błędne wartości.

- Błędne wartości z czujnika MQ-7 - wartości odczytywane po podpięciu modułu prosto do mikroprocesora były dalekie od rzeczywistych, niezdatne do funkcjonowania. Pomogło zastosowanie wstępnej kalibracji, przez odczytanie napięcia na wyjściu, przeliczeniu go na wartość rezystancji oraz podstawieniu do wzorów zgodnych z charakterystyką wyjściową tego czujnika.

- Brak komunikacji między esp a serwerem - czynników było wiele. Pierwszym problemem okazał się stary firmware modułu, który miał problem z interpretacją znaku rozpoczęcia nowej linii ‘\n’, oprogramowanie nie interpretowało tego znaku i przepisywało do wysyłanego zapytania GET, do którego konieczne jest rozpoczęcie nowej linii przy podawaniu adresu ip host’a. Pomogło zaktualizowanie firmare. Kolejnym problemem był brak opóźnień w funkcji wysyłającej dane. Pomogło odczytywanie logów z serwera i zastosowanie minimalnych opóźnień w programie.

- Brak kompatybilności pomiędzy biblioteką SoftwareSerial.h potrzebną do komunikacji pomiędzy mikroprocesorem a esp, a biblioteką PinChangeInterrupt.h potrzebną do obsługi przerwań PCINT (Pin Change INTerrupt) - producent atmegi dzieli wszystkie piny na 3 wektory, jako że komunikacja między esp a atmegą odbywa się na zasadzie przerwań to biblioteka SoftwareSerial zawłaszcza wszystkie 3 wektory do swojej dyspozycji, mimo że w naszym przypadku używany jest tylko jeden. W wyniku czego następował konflikt między bibliotekami, objawiający się w postaci komunikatu podczas próby kompilowania programu: „multiple definition of `\_\_vector\_3'”. Pomogła modyfikacja biblioteki i usunięcie części kodu przywłaszczającej wektor do którego należały piny obsługujące przyciski, i podmiana części kodu w deklaracjach pozostałych wektorów (gdyż używały one usuniętego).

-„ Zamrażanie” się programu przez funkcję delay() – pomogło przejście na funkcję millis() i sprawdzanie różnic czasowych pomiędzy czasem załączenia urządzenia, a czasem ostatniego wywołania danej funkcji.

**7. Podsumowanie**