



Akademia Górnictwo-Hutnicza
im. Stanisława Staszica w Krakowie.
Wydział Informatyki, Elektroniki i Telekomunikacji

Sensory w aplikacjach wbudowanych

SPRAWOZDANIE

Rok I, Systemy wbudowane, Elektronika i Telekomunikacja IIst.

Temat: Moduł z sensorami oparty na procesorze STM32F103, zgodny z wyprowadzeniami Mikrobus™

Zespół: 1. Mateusz Kozyra 2. Mirosław Wiącek 3. Radosław Sajdak	Oceny indywidualne: 1. 2. 3.
Data oddania sprawozdania:	Ocena sprawozdania:
Uwagi prowadzącego zajęcia:	Informacje dodatkowe:

Spis treści

1. Wstęp.....	3
1.1. mikroBUS™.....	3
1.2. STM32F103.....	4
1.3. STS30	4
1.4. BMP280.....	5
1.5. MQ2.....	5
1.6. Makefile	5
1.7. Github	5
2. Wykonanie projektu	7
2.1. Projekt płytka	7
2.1.1. Schemat.....	7
2.1.2. Layout	13
2.1.3. Popełnione błędy oraz wykonane przeróbki	16
2.2. Oprogramowanie	20
2.2.1. Biblioteki peryferiów	20
2.2.2. API	20
2.2.3. Niewykorzystane możliwości	20
2.2.4. System kontroli wersji	20
3. Prezentacja działania.....	21
4. Wnioski	23

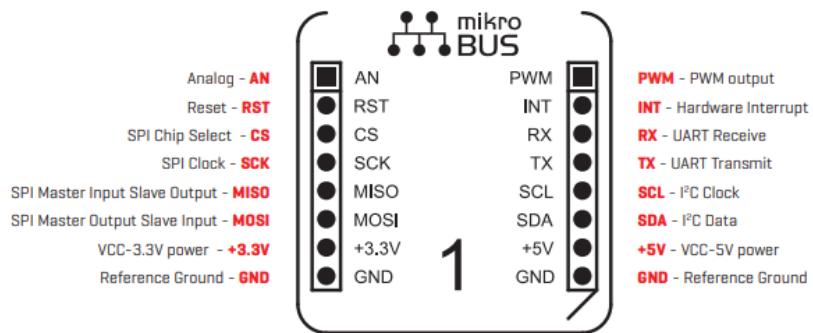
1. Wstęp

Założeniem projektu, było stworzenie modułu sensora, zgodnego ze standardem mikroBUS™. Zespół, korzystając z okazji na zdobycie nowych umiejętności, zdecydował się rozszerzyć założenia projektowe o dwa dodatkowe sensory (co daje łącznie trzy układy pomiarowe na jednej płytce). Układy te, komunikują się z mikroprocesorem STM32F103 znajdującym się na tej samej płycie. W procesorze, dane są przetwarzane, nakładając dodatkową warstwę abstrakcji dla obsługi sensorów. Z płytą, można komunikować się przy użyciu interfejsu UART. Dodatkowym założeniem projektu, było różne podejście do tworzenia oprogramowania, każdego z członków zespołu. Zdecydowano, że jeden będzie tworzyć kod, testując go z mikroprocesorem STM32F4, kolejny bezpośrednio na wykonanej płycie, a ostatni, bez bezpośredniej styczności ze sprzętem. Podejście to, pozwoliło spojrzeć na projekt z innej perspektywy, jak gdyby zespół był większy i rozproszony. Cały proces tworzenia był kontrolowany przy użyciu platformy Github, pozwalającej na wygodną kontrolę wersji oraz dyskusję nad kolejnymi zmianami w kodzie. W niniejszym sprawozdaniu, skupiono się przede wszystkim na procesie tworzenia modułu oraz popełnionych błędach, nie pomijając krótkiego wstępu teoretycznego. Całość, zakończona jest wnioskami na temat procesu oraz zdobytych umiejętności.

1.1. mikroBUS™

MikroBus™ jest standardem definiującym układ gniazd oraz wyprowadzeń na płytce, zawierającej układ scalony (np. sensor). Standard opisuje również dokładne wymiary płytka, a także narzuca warstwę nadruków na płytce. Założeniem mikroBUS™, jest umożliwienie szybkiego i łatwego rozbudowywania układów na rzecz prototypowania. Dostarczane przez wielu producentów płytki z gotowymi gniazdami, pozwalają w prosty sposób uruchomić dowolny sensor, a następnie stworzyć dla niego oprogramowanie. Tym samym, mikroBUS™ okazuje się świetną alternatywą dla niestabilnych płyt stykowych. Na rysunku 1.1 przedstawiono zdefiniowany w standardzie pinout.

Ze względu na zmienione założenia projektu, stworzona płytka zgadza się ze standardem co do wyprowadzeń i wymiarów, jednak nie posiada naniesionych nadruków. Wynika to wprost z braku miejsca, na płytce. Dodatkowo, kilka sensorów na jednej płytce, również nie jest zgodne ze standardem.



Rys. 1.1. Specyfikacja wyprowadzeń mikroBUS™[1]

1.2. STM32F103

Do stworzenia projektu, wybrano procesor STM32F103C8 z rdzeniem ARM®Cortex®-M3. Wybór, podyktowany był przede wszystkim dostępnością, ponieważ układ ten, w obudowach LQFP48 znajdował się w prywatnych zasobach zespołu. Obecnie, na rynku dostępne jest niewiele procesorów, a części elektroniczne są drogie. Z tego powodu, zdecydowano się korzystać z tego, co było dostępne. Nie oznacza to jednak, że procesor ten był pod jakimkolwiek względem ograniczający. Najistotniejszymi z jego cech są:

- 2 interfejsy I²C
- 3 interfejsy USART
- 2 interfejsy SPI
- Zasilanie 2.0 - 3.6V
- Zewnętrzny zegar 4 - 16MHz
- 12-bitowe, wielokanałowe przetworniki A/D

Układ, programowany jest przy użyciu interfejsu SWD oraz programatora ST-link. Fakt ten, znaczco ułatwiał wykonanie działającego układu. Ze względu na przyjęte założenia i wybrane sensory, wybrany procesor bardzo dobrze wpasowywał się w projekt. Dla wykonania oprogramowania, istotne były również dostarczone przez ST biblioteki oraz oprogramowanie STM32CubeMX, wprowadzające wysoki poziom abstrakcji w tworzeniu oprogramowania. Pozwoliło to uniknąć często czasochłonnego, ręcznego pisania do rejestrów procesora, w celu konfiguracji periferii. Były również momenty, w których HAL okazał się problemem, co wspomniane zostało w dalszej części sprawozdania.

1.3. STS30

Krótki opis sensora Mirka

1.4. BMP280

Krótki opis sensora Matiego

1.5. MQ2

Krótki opis sensora Matiego

1.6. Makefile

Jak wspomniano we wstępnie, jednym z założeń, było tworzenie oprogramowania równolegle, na dwóch różnych procesorach. STM32F1 i STM32F4. Procesory te, należąc do zupełnie różnych rodzin, korzystają z innych bibliotek, a do ich konfiguracji niejednokrotnie wymagane są różne kroki. Z tego powodu, w projekcie wykorzystano program *Make*. Pozwala on, na podstawie pliku konfiguracyjnego *makefile*, zdefiniować sposób komplikacji oraz ścieżki bibliotek wykorzystywanych przez kod. Narzędzie to, jest stosunkowo proste w konfiguracji, a jej przykłady są ogólnodostępne w internecie. Dodatkowym argumentem przemawiającym za jego wykorzystaniem, była chęć jego przetestowania w praktyce.

1.7. Github

Github, to serwis internetowy pozwalający przechowywać projekty z wykorzystaniem systemu kontroli wersji. Korzysta on z systemu Git opublikowanego na licencji GNU GPL 2 (wolne i otwarte oprogramowanie). System ten, jest obecnie wykorzystywany przez wielu programistów, a jego znajomość często wymagana jest przez pracodawców. Github, poza kontrolą wersji, dostarcza wygodne i przejrzyste środowisko w przeglądarce, pozwalające programistom na np. przegląd kodu współpracowników. Fakt ten, został wykorzystany podczas tworzenia projektu. Pozwoliło to nie tylko na poprawienie jakości tworzonego oprogramowania, ale również skonfrontowanie różnych podejść do współpracy nad kodem.

2. Wykonanie projektu

2.1. Projekt płytki

2.1.1. Schemat

Projekt płytki, naturalnie rozpoczęto od wykonania schematu ideowego. W tym celu przyjęto następujące założenia:

- Układ, musi być zasilany zewnętrznie (5V oraz 3.3V)
- Wykorzystany zostanie konwerter UART-USB
- Układ, będzie można wyłączyć tranzystorem, sterującym przetwornicami
- Wykorzystany zostanie zewnętrzny generator

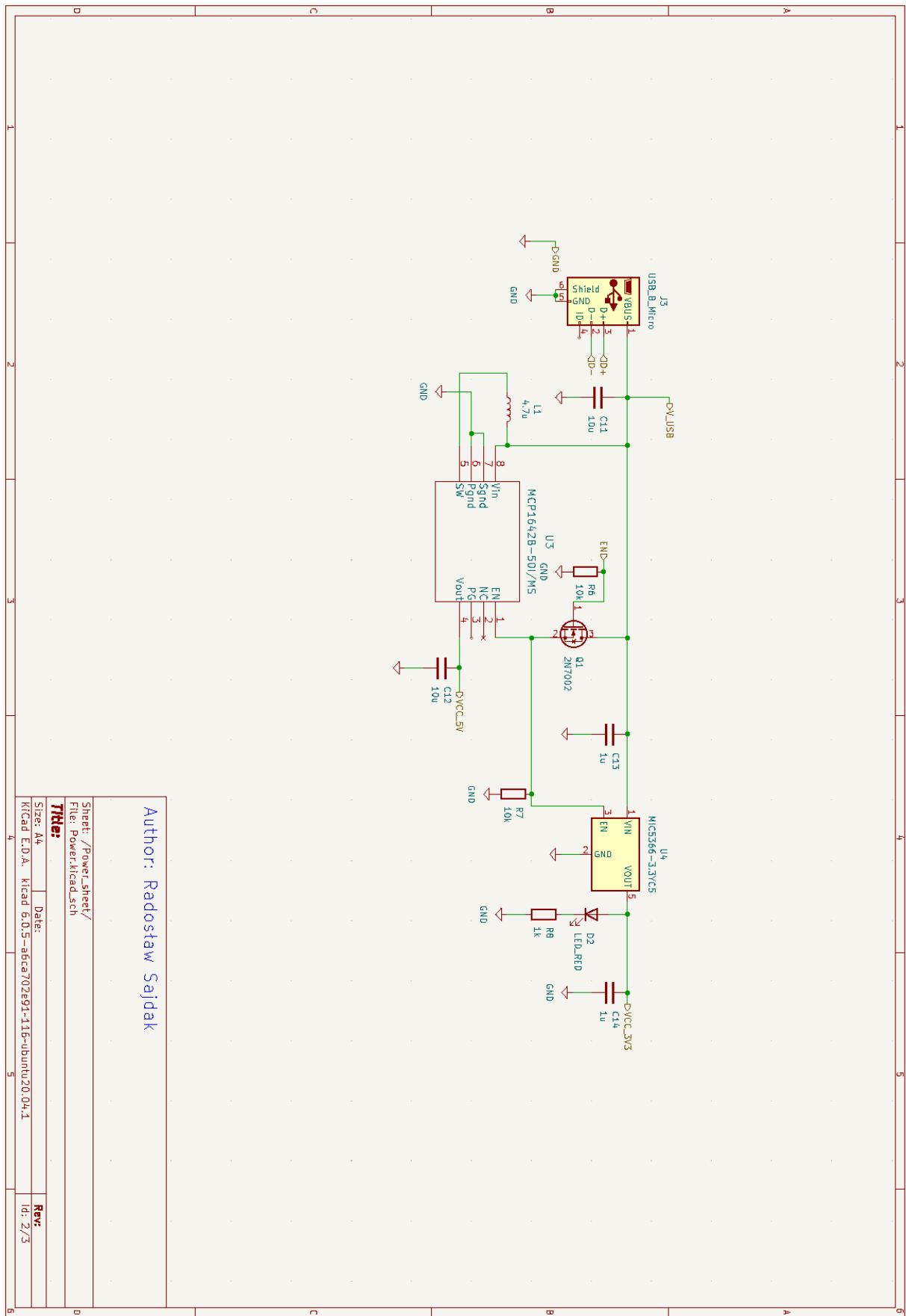
Według standardu mikroBUSTTM, płytka zasilana jest przez układ w który jest wpięta. Często jest to zasilanie procesora, na którym tworzone jest oprogramowanie. Ponieważ w opisywanym projekcie, wykorzystany został procesor, oraz trzy sensory (w tym jeden pracujący z napięciem 5V), zdecydowano się na zasilanie zewnętrzne. Miało ono zminimalizować prawdopodobieństwo nieprawidłowej pracy układu. Jednocześnie, czujnik MQ2 pobierając setki mA prądu, mógłby uszkodzić przetwornicę układu w który została wpięta płytka. Najprostszym rozwiązaniem, było wyprowadzenie portu USB i zasilanie z niego układu. Zwrócono uwagę na fakt, że USB w komputerach, nie ma stabilnego napięcia 5V, a często wręcz 4.7-4.8V. Jest to zachowanie zdefiniowane w standardzie USB 2.0 [2] w rozdziale 7. Z tego powodu, należało wykorzystać regulator napięcia.

Wybór tego i wszystkich kolejnych komponentów, podkutowany był w dużej mierze dostępnością na rynku. Kolejnym kryterium, były parametry układu, opisywane w dostarczanych przez producentów dokumentacjach. Jako układ do zasilania sensora MQ2, wykorzystano przetwornicę typu BOOST - MCP1642 o wyjściowym napięciu właśnie 5V. Pozwoliła ona na stabilne zasilenie czujnika. Jej ważną cechą, jest maksymalny prąd na wyjściu, o wartości 800mA. Wysoki prąd jest tutaj niezbędny, ponieważ zasilanie wspomnianego czujnika wymaga nawet 200mA prądu stałego. Do zasilania reszty elementów, wykorzystano regulator LDO MIC5365 o wyjściowym napięciu 3.3V. Gwarantowany prąd wyjściowy regulatora to 150mA, co jest wartością wystarczającą do zasilenia reszty elementów z zapasem. Całość, z założenia włączana miała być tranzystorem NMOS 2N7002. Bramka tego tranzystora, połączona jest z

pinem RESET standardu mikroBUS™, pozwalając użytkownikowi wyłączyć układ. W przypadku tego elementu, pojawiły się dwa błędy, wymagające przerobienia gotowej płytki. Problem, szerzej opisano w podrozdziale 2.1.3.

Schemat oraz layout, wykonane zostały w programie KiCad. Opensourcowym oprogramowaniu dla różnych systemów operacyjnych. Program ten, umożliwia podział schematu na bloki, co wykorzystano przenosząc sekcję zasilania do osobnego schematu. Na schemacie 2.1 przedstawiono całą, wspomnianą sekcję. Zasilanie, doprowadzone jest z USB do obu przetwornic. Gdy procesor jest zasilony, a więc działa przetwornica 3.3V, włączona zostaje czerwona dioda LED. Widoczny na schemacie rezystor R10, nie jest uwzględniony w layoucie i został dołożony ręcznie do płytki. Więcej na ten temat, opowiada podrozdział 2.1.3.

Na schemacie, widoczna jest również znacząca ilość kondensatorów. Zostały one dodane do schematu, jako wskazane przez producenta jako konieczne dla poprawnego działania układów. Bardzo istotny jest również rezystor R7, gwarantujący ustalony stan niski po wyłączeniu tranzystora.



Rys. 2.1. Schemat części zasilającej moduł

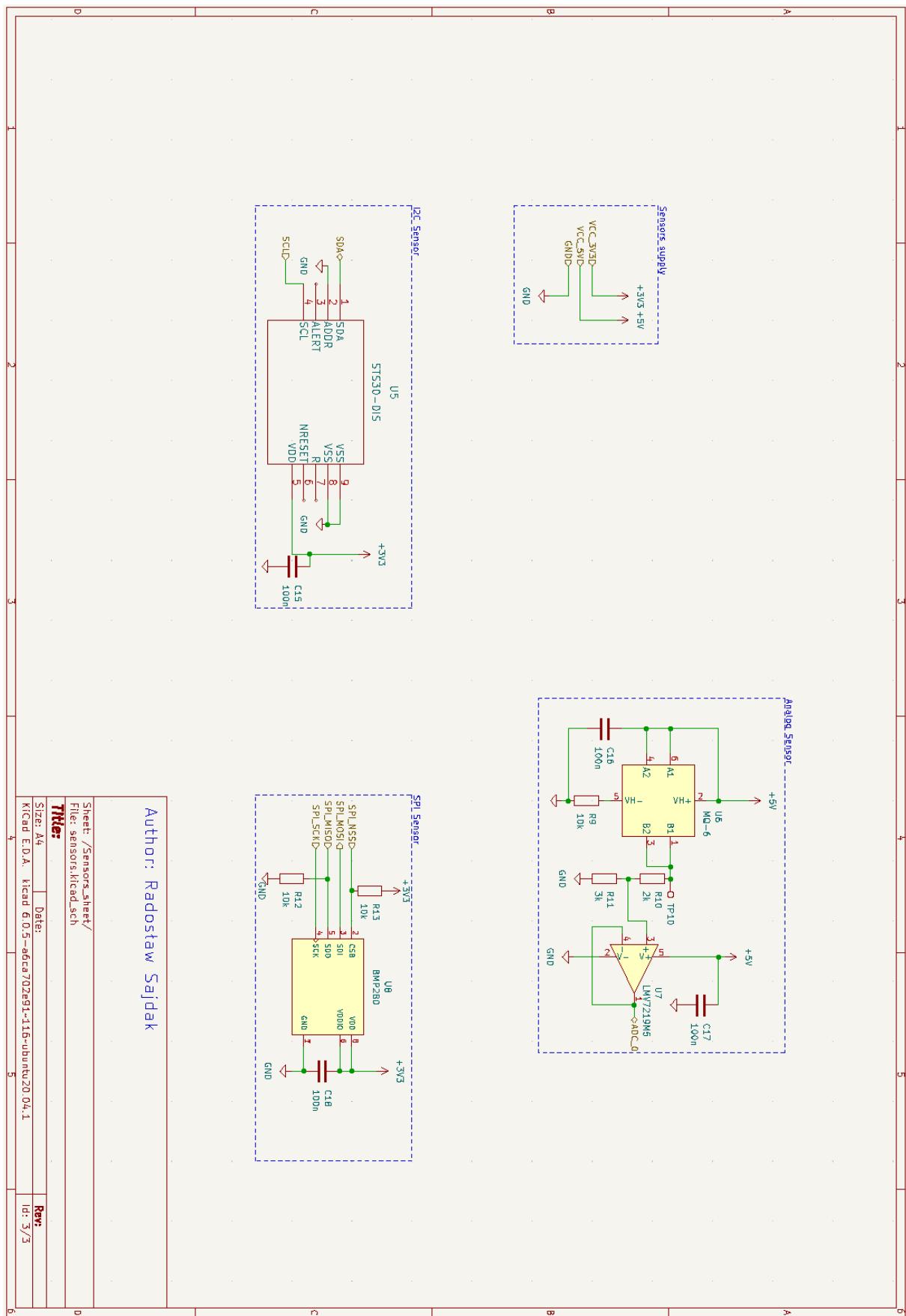
Kolejnym blokiem, jest schemat 2.2, zawierający w sobie wszystkie czujniki. Podobnie jak wcześniej, producenci w swoich dokumentacjach zalecają stosowanie kondensatorów, jak najbliżej zasilania układu, dla zapewnienia jego prawidłowego działania. W przypadku sensora działającego z użyciem SPI - BMP280 - zastosowano rezystor pull-down na linii MISO. Dzięki temu, gdy linia nie jest używana, zagwarantowany jest stan niski. Podobnie w przypadku linii NSS. Zastosowanie rezystora podciągającego do zasilania, pozwala zagwarantować, że układ nie będzie aktywny gdy nie jest to pożądane.

Wartym uwagi jest układ czujnika analogowego MQ2. Ze względu na zastosowany procesor, mierzone przetwornikiem napięcie, musi mieścić się w przedziale 0-3.6V. Ponieważ wybrany czujnik działa w zakresie 0-5V, konieczne jest zastosowanie dzielnika napięcia. W tym przypadku, układ rezystorów oraz czujnika, staje się źródłem prądu, który mógłby uszkodzić procesor. Z tego powodu, zastosowano wzmacniacz operacyjny w konfiguracji nieodwracającego wtórnika napięciowego. Wzmacniacz operacyjny, mający bardzo małą rezystancję wyjściową, stanowi w przybliżeniu źródło napięcia równe co do wartości spadkowi napięcia na R11.

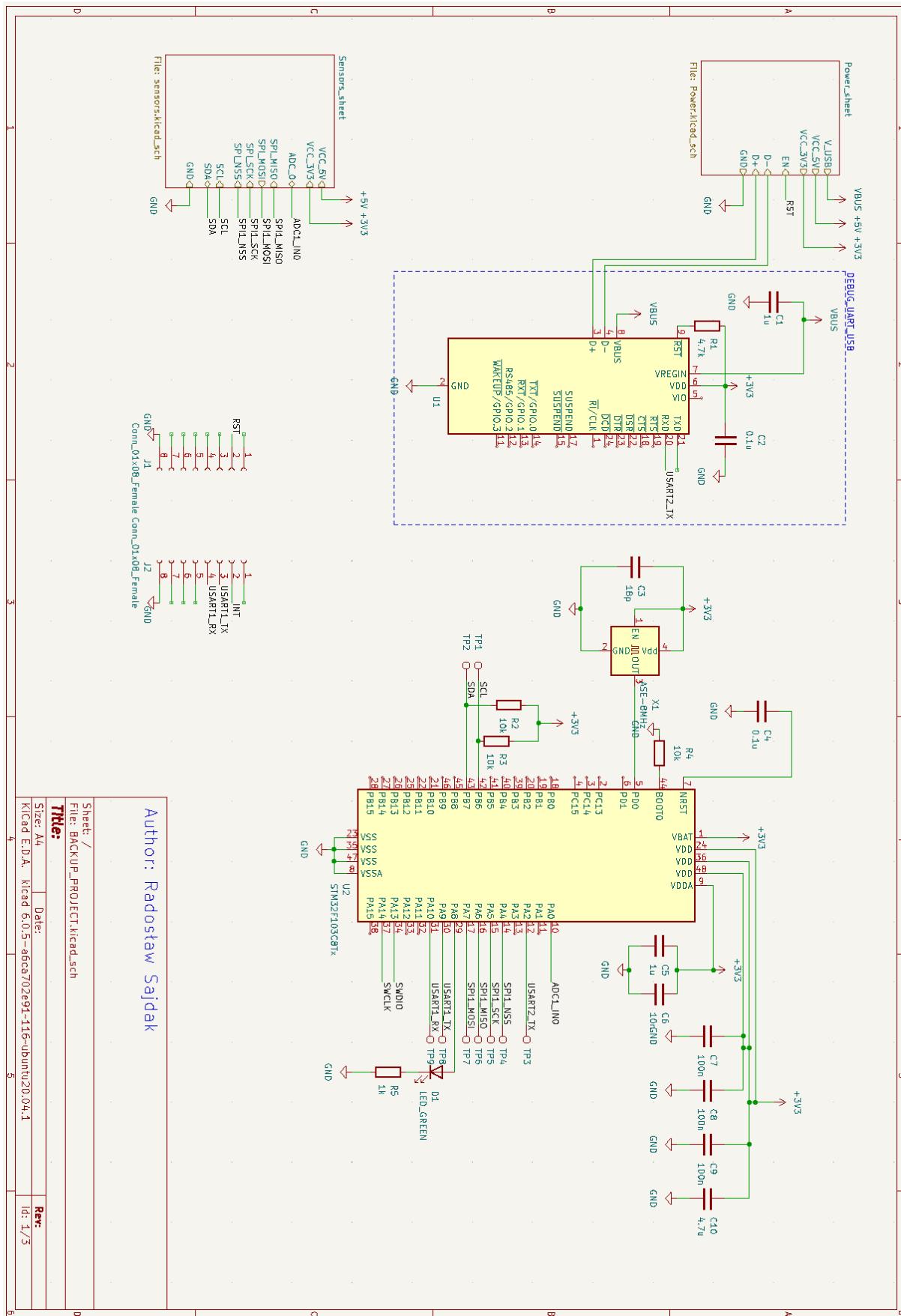
Ostatnim, a zarazem najistotniejszym, jest schemat 2.3. Po jego lewej stronie, zaznaczono fragment odpowiadający za konwersję UART-USB. Wykorzystany układ konwertera to CP2102. Układ ten, pozwala obserwować logi mikroprocesora przy użyciu tego samego przewodu, którym zasilamy płytę. W trakcie tworzenia schematu, rozważano również tylko wyprowadzenie testpointów, pozwalających na podejrzenie logów zewnętrznym konwerterem. Ponieważ jednak planowany jest również inny projekt, wykorzystujący wiele takich konwerterów, zdecydowano się na jego użycie w celach przede wszystkim edukacyjnych i testowych. Głównym zagadnieniem z nim związanym, było użycie pary różnicowej, wymaganej dla prawidłowego działania USB.

Po prawej stronie schematu 2.3, widoczny jest blok mikroprocesora STM32F103C8. Zastosowano tutaj zewnętrzny generator o taktowaniu 8MHz, zastępując wewnętrzny zegar mikroprocesora. Istotnym zaganiem, wymaganym przez mikroprocesor według dokumentacji, jest montaż osobnych kondensatorów o wartości 100n przy każdym z pinów zasilania, możliwe jak najbliżej układu. Dla zapewnienia prawidłowego działania magistrali I²C, dodano rezystory podciągające linie do zasilania. Wszystkie piny posiadają też testpointy, mające ułatwić tworzenie oprogramowania dzięki szybkiemu wykryciu błędów w transmisji, a więc np. w złutowaniu układu. Dodatkowo, na pinie 8, wyprowadzono LED. Zgodnie z przyjętymi przez zespół założeniami, wskazuje on gotowość układu do pracy, a więc przejście wszystkich kroków inicjalizacyjnych.

Wartym zaznaczenia jest tutaj fakt nie wyprowadzenia pinów SWDIO oraz SWCLK. Ponieważ są to piny wymagane tylko na etapie tworzenia oprogramowania, podjęto decyzję o ich niewykorzystaniu. Ważną w tym wypadku kwestią, był również brak miejsca na ścieżki, pozwalające na taki zabieg. Niemniej jednak, osoba składająca PCB, bez większych problemów była w stanie włutować przewody programatora bezpośrednio do pinów mikroprocesora.



Rys. 2.2. Schemat połączeń czujników

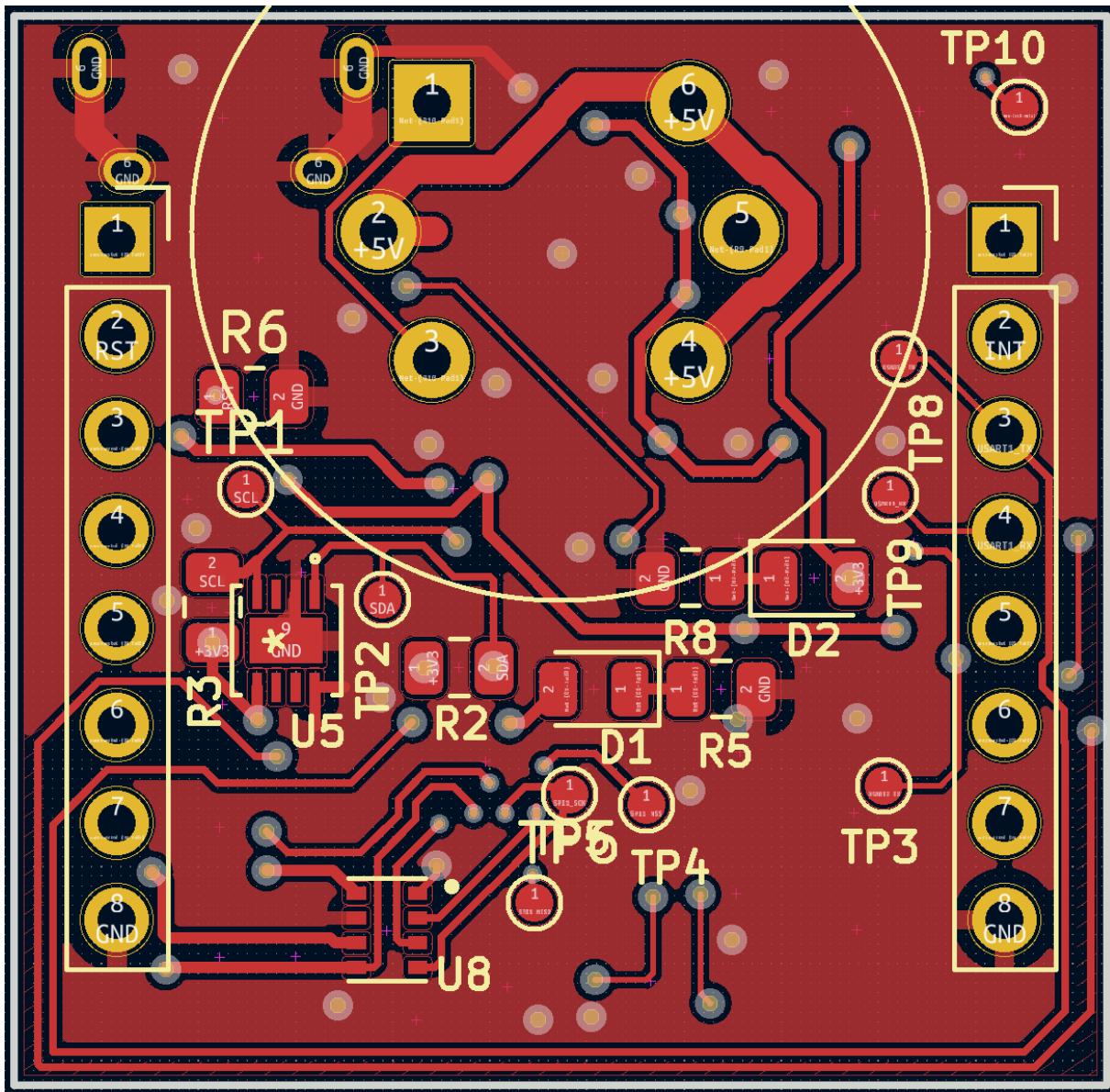


Rys. 2.3. Schemat połączeń konwertera UART-USB oraz mikroprocesora

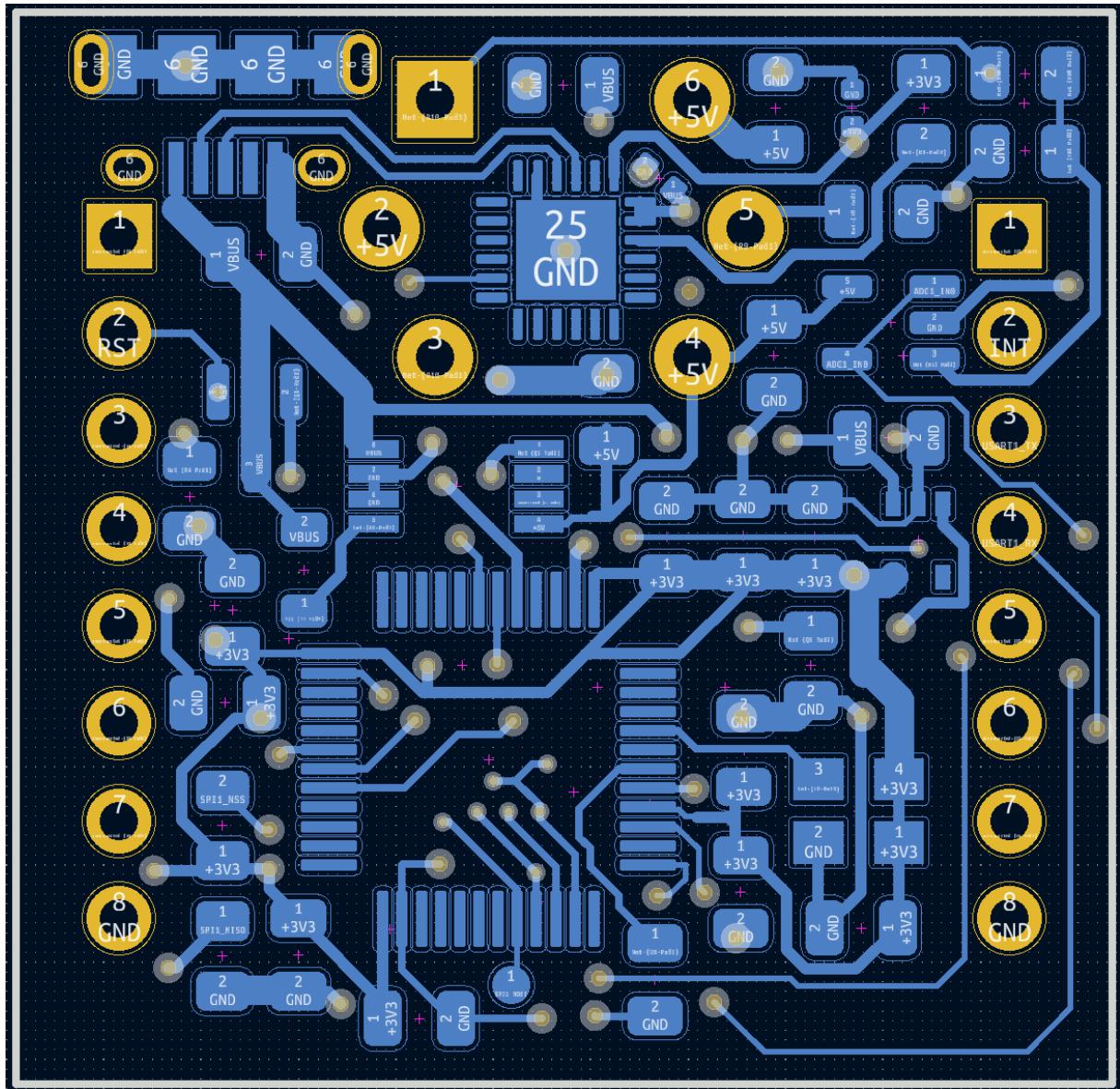
2.1.2. Layout

W kolejnym kroku, wykonany został layout płytki. Obrazek 2.4 przedstawia frontową warstwę elektryczną płytki oraz nadruki na niej. Z założenia, jest to warstwa w większości wypełniona masą (GND). Niestety, ze względu na brak miejsca na warstwie tylnej, masa układu jest znacznie pocięta, choć pozostała jednolita. W przypadku linii 5V, zasilającej czujnik MQ2, zastosowano znacznie szersze ścieżki o szerokości 30mils. Pozostałe, mają szerokość 12 oraz 7,87mils. Druga z tych wartości, jest wartością minimalną, domyślną dla programu KiCad. Mimo ręcznego ustawiania mniejszej szerokości, zgodnej z zaleceniami JLCPCB[3], program automatycznie zmieniał szerokość ścieżki na większą. Podobnie było w przypadku przelotek. Niestety, zorientowano się o tym pod koniec tworzenia płytki, a mogłyby to pomóc oszczędzić miejsce. Szczególnie istotne, jest to dla warstwy sygnałowej, przedstawionej na rysunku 2.5. Ze względu na czytelność, rysunek przedstawia jedynie połączenia elektryczne, bez nadruku. Jako elementy pasywne, wykorzystano te w obudowach 0806 oraz pojedyncze elementy 0402. Również w przypadku tej warstwy, znacznie pogrubiono ścieżki doprowadzające zasilanie. Ma to na celu poprawę przewodności ścieżek, która zależy od ich szerokości. Jak można zauważyć, na płytce jest bardzo gęsto, a część elementów znajduje się na granicy spełnienia warunków DRC (np. odstępów między padami). Ponieważ jednak był to pierwszy projekt płytki autora, popełniono tutaj kilka błędów. Najważniejszym z nich, jest użycie komponentów 0806. Zdecydowano się na nie, ponieważ wszystko lutowane było ręcznie (z użyciem szablonu do pasty, jednak w trakcie wykonywania projektu, większość układów była wielokrotnie ręcznie przelutowywana). Niestety, zajmują one bardzo dużo miejsca na płytce, a obudowy 0603 również dają się łatwo lutować ręcznie. Dobrą zmianą, byłaby również bardziej zwarta sekcja zasilania. Dodatkowe miejsce, mogłyby pozwolić wygodniej umiejscowić komponenty. Również szerokość ścieżek sygnałowych, mniejsza o 1,87mils (0,0254mm) mogłyby pozwolić zaoszczędzić nieco miejsca i lepiej poprowadzić ścieżki.

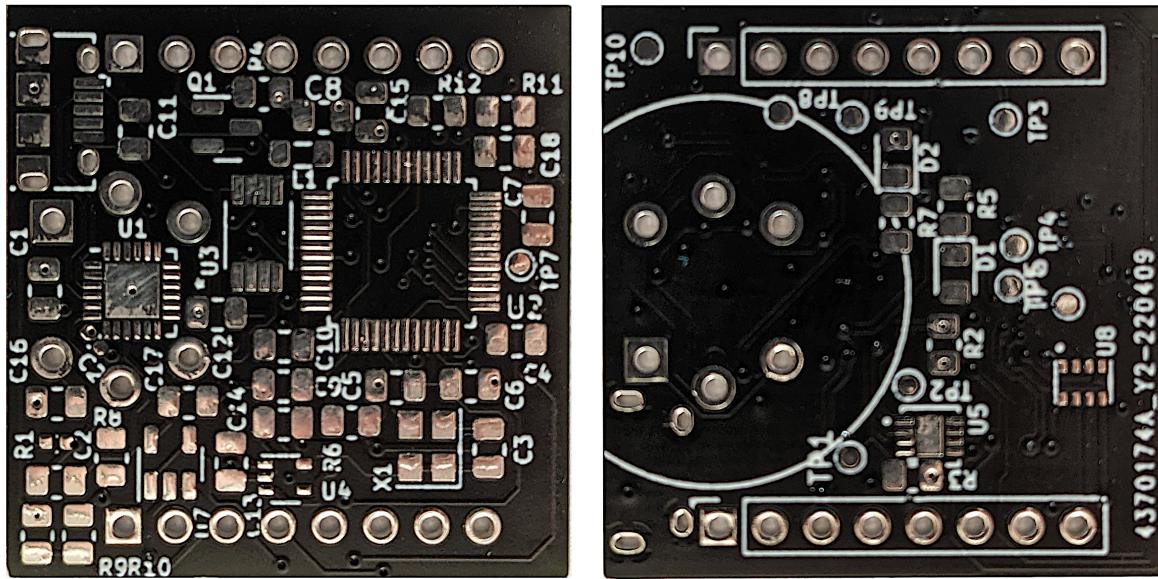
Po wygenerowaniu plików, płytę oraz szablon, zamówiono w serwisie JLCPCB. Jednocześnie, części zamówione w sklepie Mouser. Obrazek 2.6 przedstawia gotową, wykonaną płytę przed przygotowaniem elementów. Natomiast obrazek 2.7 przedstawia zlutowaną, gotową płytę przed wykonaniem niezbędnych przeróbek, omówionych w następnym podrozdziale.



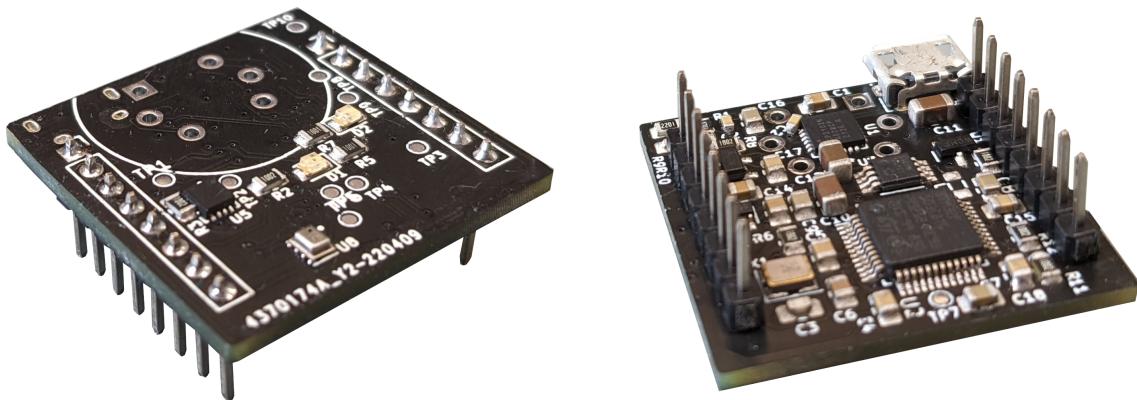
Rys. 2.4. Frontowa warstwa elektryczna płytki



Rys. 2.5. Tylna warstwa sygnałowa płytki



Rys. 2.6. Zdjęcie wykonanej płytki przed lutowaniem elementów



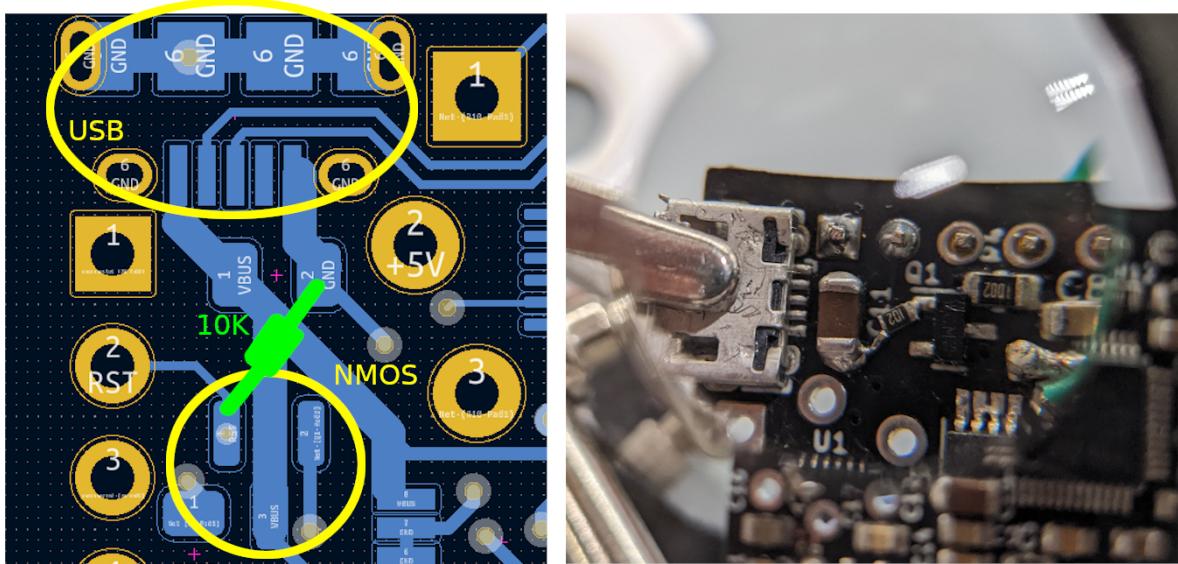
Rys. 2.7. Zdjęcie wykonanej płytki po lutowaniu elementów

2.1.3. Popełnione błędy oraz wykonane przeróbki

Niestety, ze względu na zupełny brak doświadczenia w projektowaniu układów elektronicznych oraz zwykłe niedopatrzenia, popełniony został szereg błędów, wymagający zmian w celu uruchomienia płytki.

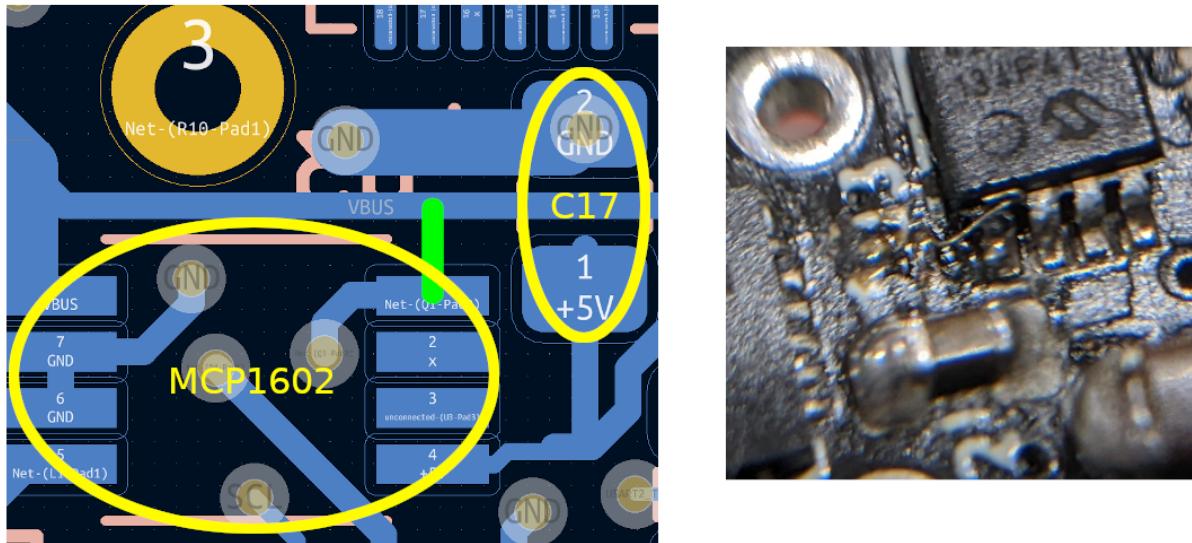
Pierwszym z wykrytych błędów, była bramka tranzystora włączającego przetwornice wisząca w powietrzu. Zaznaczony na schemacie 2.1 rezistor R6, dodany został już po wyprodukowaniu płytak. Ze względu na jego brak, bramka znajdowała się w niaustalonym stanie, a mówiąc kolokwialnie - pływała. Powodowało to włączanie i wyłączenie układu w krótkich odstępach czasu, co uwidoczniło się przez mruganie diody LED po podpięciu zasilania, ale nie doprowadzeniu zasilania na bramkę

tranzystora. Uruchomiony raz układ, nie wyłączał się, ponieważ zgromadzone w bramce ładunki, nie miały gdzie ujść. Rozwiążaniem było właśnie dodanie rezystora R6 ściągającego do masy. Rozwiązanie na fragmencie layoutu oraz złożonej płytce, przedstawia rysunek 2.8. Kolorem żółtym, zaznaczono istotne elementy, pozwalające odnaleźć miejsce na schemacie oraz layoutcie, a kolorem zielonym, dodany element.



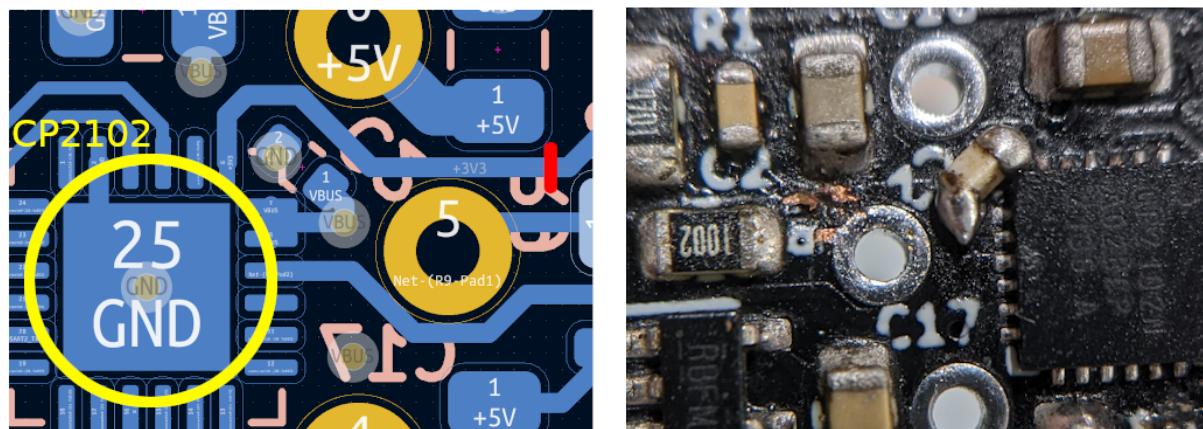
Rys. 2.8. Rezystor ściągający bramkę do masy

Jak się później okazało, wspomniany mosfet generował znacznie więcej problemów. Najważniejszym z nich, było nie włączanie przetwornicy MCP1642. Aby przetwornica mogła być włączona, napięcie na pinie EN, powinno wynosić $0.75 * \text{Vin}$. W przypadku tego układu, wartość ta wynosi 3.5V. Niestety, w stworzonej konfiguracji, napięcie nie przekracza 1.5V. W trakcie realizacji projektu, podjęto kilka różnych prób przerobienia układu w taki sposób, aby przetwornica mogła zadziałać poprawnie. Wszystkie z nich, opisane zostały poniżej. Ostatecznie jednak, zdecydowano podłączyć pin EN na stałe do zasilania USB (Vbus). Nie niesie to za sobą znaczących konsekwencji, ponieważ przetwornica ta, zasila jedynie czujnik MQ2, a ten, jako czujnik analogowy, nie wymaga resetu. Obrazek 2.9 przedstawia zastosowaną poprawkę. Zielonym kolorem zaznaczono miejsce zwarcia ścieżki z pinem przetwornicy.



Rys. 2.9. Zwarcie pinu EN ze ścieżką Vbus

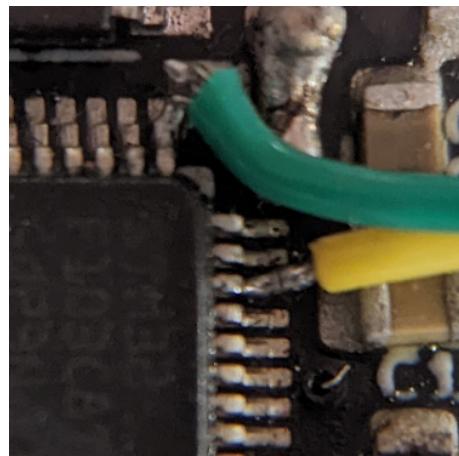
Dopisać o tych innych zmianach które próbowałem zrobić Kolejnym z problemów, tym razem wynikającym z niedoczytania dokumentacji, było połączenie wewnętrznego LDO konwertera CP2102, z linią 3.3V. W efekcie, mimo wyłączonej przetwornicy MIC5365, układ utrzymywał zasilanie. W tym przypadku, prostym i szybkim rozwiązaniem było odcięcie ścieżki, łączącej konwerter z wyjściem przetwornicy MIC5365. Zmiana, widoczna jest na obrazku 2.10. Na zdjęciu widać również delikatne naruszenie ścieżki obok, linii rezystora ściągającego MQ2 do masy. Niestety, powstało ono podczas cięcia, ze względu na brak miejsca między komponentami. Nie wpływa ono jednak na działanie układu. Na tym samym zdjęciu, widać również kondensator z bardzo dużą ilością cyny. Jest to kondensator w obudowie 0603, który nie mieściąc się na padach 0402 (niedostępny element), został dolutowany „rozszerzając” pady cyną. Było to spore wyzwanie, aby nie zewrzeć wspomnianych padów, z pinami CP2102.



Rys. 2.10. Przecięcie ścieżki 3.3V na wyjściu LDO CP2102

Tu jeszcze coś innego będzie

Jak wcześniej wspomniano, brak miejsca oraz inne czynniki, spowodowały, że nie wyprowadzono pinów programatora. O ile jednokrotne przylutowanie przewodów bezpośrednio do procesora, nie było szczególnym problemem, o tyle uciążliwym stało się podczas innych przeróbek. Przewody, wielokrotnie odpadały, gdy płytka została nagrzana. Tym samym w kolejnym projekcie, zdecydowanie należy wyprowadzić pady programatora możliwie daleko od reszty komponentów. Zdjęcie 2.11 przedstawia przylutowane przewody.



Rys. 2.11. Przewody SWD wlutowane w procesor. Zielony - SWDCLK, Żółty - SWDIO

- OK!!! Pokazać odcięcie LDO konwertera
- OK!!! Bramka wisząca w powietrzu
- Zła przetwornica 5V ($V_{in} > V_{out}$)
- Brak cewki = zwarcie xD
- Źle wsadzony mosfet. Pokazać możliwe przeróbki (OPAMP, rezystory, zdjęcia z prób i cięć).
- OK!!! Bypass przetwornicy
- OK!!! Ręczne lutowanie SWD do procka

2.2. Oprogramowanie

2.2.1. Biblioteki peryferiów

Krótko opisać jakie były założenia przy tworzeniu peryferiów (rozbicie drzewka kodu dla porządku, ustandaryzowane kody błędów, osobne libki na wszystko itp.)

2.2.2. API

Założenia API. Jak działa, wrzucić grafy, listę komend(?), mechanizmy rejestracji komend jako coś, co pozwala łatwo rozbudować soft do innej aplikacji.

2.2.3. Niewykorzystane możliwości

Nie wiem, tutaj mógłbym napisać że zrobiłem kozak timery, ale w sumie to nie wiadomo po co bo API jest bezobsługowe xD No i może o tym, że Mati nie wiadomo po co pisał na F4...

2.2.4. System kontroli wersji

IMHO spoko też napisać, o tym, że korzystaliśmy z tego gitu mocno, dbaliśmy o porządek, robiliśmy review itp. To jednak zjadło kupę czasu.

3. Prezentacja działania

Wrzucić tutaj fotki z cutecoma i pokazać, że działa. Może jakieś zdjęcia że przed initem lampka nie świeci, a po świeci xD

4. Wnioski

No na pewno napisać, że inaczej bym PCBka zrobił. Że dużo było czytania dokumentacji, trzeba było sobie przypomnieć sporo z elektroniki, a i tak pojawiły się błędy. Na pewno, że 0805 są za duże. Że robienie review pomaga obu stronom czegoś się nauczyć. Że sensory na PCB muszą być przemyślane (MQ + temperatury). Że mimo tego, że N osób widziało schematy i layout, to błędy przeszły produkcję i dużo czasu straciło się na debug zasilaczami, oscyloskopami itp.. Ale trzeba też powiedzieć, że robiliśmy projekt 4fun i dużo się nauczyliśmy.

Bibliografia

- [1] MikroElektronika. *MikroBUS standard specification*.
- [2] USB organization. *USB 2.0 specification*.
- [3] JLCPCB. *PCB Capabilities*.