AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ INFORMATYKI, ELEKTRONIKI,
I TELEKOMUNIKACJI

KATEDRA ELEKTRONIKI

Praca dyplomowa inżynierska

Opracowanie urządzenia wykrywającego wypadek rowerowy z powiadamianiem GSM

Development of a bicycle accident detection device with GSM notification

Autor: *Radosław Sajdak*
Kierunek studiów: *Elektronika i Telekomunikacja*
Opiekun pracy: *dr inż. Łukasz Krzak*

Kraków, 2022

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystycznego wykonania albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.", a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.): „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej «sądem koleżeńskim».", oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i że nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

Serdecznie dziękuję . . . tu ciąg dalszych podziękowań np. dla promotora, żony, sąsiada itp.

Spis treści

1. Wstęp

Obecny rozwój mikroprocesorów, pozwala na tworzenie coraz bardziej złożonych urządzeń. Rozwój układów o niskim zużyciu energii, popycha elektronikę w kierunku małych, wielofunkcyjnych urządzeń. Połączenie tych dwóch procesów pozwala na stworzenie elastycznych urządzeń, których zastosowanie może zmieniać się jedynie dzięki oprogramowaniu.

1.1. Cele pracy

Rowerzyści górscy, podczas samotnych wypraw rowerowych, wielokrotnie zastanawiają się, co w sytuacji, gdy ulegną wypadkowi podczas samotnej wyprawy? Jak długo nikt nie wezwie pomocy? W ten sposób, powstał pomysł stworzenia urządzenia asystującego.
Celem pracy było stworzenie innowacyjnego urządzenia wykrywającego upadek na rowerze. Urządzenie miało być przymocowane do ramy roweru. Urządzenie, informację o wypadku wysyła przy użyciu modemu LTE-M opartego o interfejs szeregowy UART. Za lokalizowanie urządzenia, odpowiadać będzie moduł GPS oparty o magistralę I^2C. Całość, sterowana będzie przy użyciu mikrokontrolera.

1.2. Analiza wymagań technicznych i dobór komponentów

Docelowo, urządzenie miało zwiększać bezpieczeństwo podczas wypraw rowerowych. Musiało więc być bardzo energooszczędne. Minimalne wymaganie, to 24 godziny pracy na jednym ładowaniu baterii. Jednocześnie, nie może być zbyt duże, aby w łatwy sposób można było zamontować je na rowerze. Pobierana lokalizacja, miała mieć dokładność około 100m. Jest to dokładność wystarczająca, aby zobaczyć ranną osobę, leżący rower, lub usłyszeć wołanie o pomoc. Ważnym było, aby urządzenie, było w pełni niezależne od innych układów, jak np. telefon.
Planując pracę, zdecydowano się wykorzystać trzy moduły:

- Akcelerometr
- GPS
- LTE

Dodatkowo, ze względu na łatwy w użyciu stos Bluetooth, wykorzystano również Bluetooth Low Energy, celem stworzenia bezprzewodowego interfejsu sterowania urządzeniem.
Ponieważna rynku dostępnych było wiele różnych modułów, poniżej dokonano ich porównania oraz wyboru układów, najbardziej pasujących do stworzonego rozwiązania.

1.2.1. Akcelerometr

Akcelerometry to układy, mierzące przyspieszenie. Mogą dokonać pomiaru przyspieszenia statycznego (np. Ziemskiego), lub dynamicznego, działającego z sił, działających na układ. W przypadku dostępnych na rynku akcelerometrów należy pamiętać, że w stanie spoczynku wskazują one przyspieszenie około $9.81\frac{m}{s^2}$. Można więc było, wykorzystać ten fakt, do implementacji algorytmu.
Obecnie, większość układów to układy integrujące akcelerometr i żyroskop w jednym układzie scalonym. Coraz częściej, można też spotkać magnetometr. Dla obecnych na rynku układów, wyróżniamy dwa najważniejsze parametry:

– Zakres pracy akcelerometru - określany jako $\pm X_g$, a więc przyspieszenie w trzech kierunkach, podane jako wielokrotność przyspieszenia Ziemskiego. Zazwyczaj, wartość ta, mieści się w przedziale od kilku, do kilkunastu g.

– Zakres pomiarów żyroskopu - określony jako *dps (degrees per second)*. Jeśli prędkość kątowa będzie większa, niż wybrany zakres, układ ulegnie nasyceniu

Głównym wymaganiem dotyczącym akcelerometru, była jego energooszczędność. Był to jedyny układ, który działa przez cały czas. Z tego powodu, akcelerometr powinien był nie tylko zużywać mało prądu, ale również posiadać różne tryby pracy. Dodatkowym atutem, była wbudowana pamięć, pozwalająca na buforowanie danych.
Spośród dostępnych na rynku układów, wybrane zostały trzy, dostępne w trakcie tworzenia pracy.

1.2.1.1. MPU-6050

Wybrany układ, jest 3-osiowym akcelerometem i żyroskopem. Korzysta on z magistrali I^2C. Zgodnie z dokumentacją układu, w normalnym trybie pracy, można spodziewać się ok. 3.8mA prądu, pobieranego przez układ.[1] Wartość tę, można zredukować do nawet 10μA, ograniczając częstotliwość próbkowania akcelerometru do 1.25Hz i wyłączając żyroskop. Ponadto, układ posiada tryb niskiego zużycia energii, pozwalający uśpić nieaktywny układ. Sam akcelerometr, pracuje w zakresie $\pm$2g, $\pm$4g, $\pm$8g oraz $\pm$16g. Dodatkowo, układ posiada tzw. Digital Motion Pocessor (DMP), czyli układ wspomagający przetwarzanie danych w kierunku wykrywania gestów. Wbudowane FIFO, pozwala na buforowanie danych. Zaletą akcelerometru, są programowalne przerwania oraz przerwanie "High-G", wyzwalane w momencie przekroczenia zdefiniowanego przyspieszenia.

1.2.1.2. LSM9DS1

Układ ten, nie różni się znacząco od MPU-6050. Zgodnie z dokumentacją, jest on dodatkowo wyposażony w magnetometr. Największą z różnic, jest pobierany przez niego prąd. W przypadku LSM9DS1, akcelerometr w trybie normalnym, pobiera około 600μA.[2] Niestety, wykorzystanie żyroskopu, dodaje kolejne 4mA. Żyroskop, posiada tryb niskiego zużycia energii, pozwalający ograniczyć zużycie energii do 1.9mA. Tym samym, układ nie jest w stanie zejść poniżej 1.96mA, co znacząco przekraczało domniemany pobór prądu.

1.2.1.3. LSM6DSOX

Ostatni z wybranych układów, był układem producenta ST. Jest on dedykowany do rozwiązań, o niskim zużyciu energii. Według dokumentacji, jego zużycie energii to 550μA.[3] Wartość ta, jest kilkukrotnie niższa, niż w przypadku MPU-6050. Co więcej, układ posiada tryb LowPower, w którym zużycie energii można ograniczyć do nawet 4μA. LSM6DSOX, posiada również 9kB FIFO, po którego napełnieniu, wystawiane jest przerwanie. Dodatkową zaletą, jest szesnaście programowalnych maszyn stanów. Pozwalają one na maksymalne ograniczanie zużycia energii, dzięki możliwości wyłączenia mikrokontrolera w trybie analizy danych. LSM6DSOX, do komunikacji wykorzystuje I^2C. Pozostałe jego parametry, są zbliżone do opisywanych wcześniej.

1.2.1.4. Wybór akcelerometru

Spośród trzech dostępnych układów, wybrany został LSM6DSOX. Jak pokazuje tabela 1.1, układy te, są stosunkowo podobne. Decydującym elementem, okazały się być maszyny stanów, wbudowane w akcelerometr. Pozwoliły one znacząco ograniczyć zużycie energii całego układu oraz przyspieszyć proces tworzenia aplikacji.

	MPU-6050	LSM9DS1	LSM6DSOX
FIFO	1kB	128B	9kB
Prąd pracy	3.8 mA	600 μA	550 μA
Low Power	10-110 μA	1.9-3.1 mA	4-20 μA
Zakres pracy	2-16g	2-16g	2-16g

Tabela 1.1. Porównanie dostępnych akcelerometrów

1.2.2. Mikrokontroler

Mikrokontroler, jest mózgiem układu. z tego powodu, wybrany układ musiał być bardzo wydajny, ale Jednocześnie energooszczędny. Kolejnym z wymagań, była obsługa interfejsu I^2C oraj UART. Ponieważ wybrany akcelerometr posiada dwa piny przerwań zewnętrznych, mikrokontroler musiał być gotowy je obsłużyć.

1.2.2.1. ATmega328p

Mikrokontroler ATmega328p, to jeden z najpopularniejszych układów na rynku. Jego niska cena i łatwość użycia, pozwala budować najróżniejsze aplikacje. Układ wspiera zewnętrzny zegar o częstotliwości do 16MHz. Posiada on UART, I^2C oraz SPI, co pozwoliło rozważyć go w pracy. Dodatkowo, w dokumentacji, można znaleźć informację o dwóch przerwaniach zewnętrznych, obsługiwanych przez układ.[4] Dla najwyższej częstotliwości, podczas pracy układ pobiera 9-14mA. Wartość ta, może zostać ograniczona do 2.8mA w trybie czuwania. W trybie zupełnego uśpienia, mikrokontroler pobiera pomiędzy 44, a 66μA.

1.2.2.2. STM32F303K8

Układ STM32F303K8 oparty jest na rdzeniu ARM Cortex M4. Posiada on wbudowany układ czasu rzeczywistego z funkcją wybudzania z kalendarzem. 11 wbudowanych liczników, pozwala na budowanie wyjątkowo złożonych aplikacji. Mikrokontroler wyposażono w SPI, I^2C, SPI, oraz 3 UARTy. W przypadku rdzeni Cortex, każdy z pinów może być skonfigurowany jako przerwanie, co daje nam aż 60 przerwań. [5] Układ, może być taktowany zewnętrznym zegarem o częstotliwości do 32MHz, a więc dwukrotnie większym, niż w przypadku kontrolera ATmega. STM32F303K8 pobiera (w zależności od konfiguracji) 12.9 - 34mA w trybie pracy oraz 0.93-18.57μA w trybie głębokiego uśpienia. Warto zaznaczyć, że wartości maksymalneukład osiąga, gdy wszystkie peryferia są uruchomione. W przypadku tej pracy, dążono do minimalizacji ilości działających podzespołów.

1.2.2.3. nRF52840

Mikrokontroler nRF52840 jest najmocniejszym spośród wybranych do tego porównania. Oparty na rdzeniu ARM Cortex M4, taktowany zegarem 64MHz, posiada 1MB pamięci flash oraz 256kB RAM. Wspiera on I^2C, 4x SPI, 2x UART i 3xRTC. Obsługuje też Bluetooth, co wyróżnia go na tle dwóch pozostałych układów. Mimo, że posiada on tak dużo peryferiów, jego zużycie energii w trybach głębokiego uśpienia mieści się pmiędzy 0.4uA, a 17.37uA. Są to wartości zbliżone do kontrolra STM32F303K8, opartego na tym samym rdzeniu. Pracujący układ, w zależności od konfiguracji, zużywa 2.8-6.3mA. Dodatkowo, nRF52840 posiada wbudowane PMU (Power Management Unit), które automatycznie zarządza zużywaną energią, bez ingerencji użytkownika. Zaletą, jest również bardzo obszerna dokumentacja[6], dedykowane fora wsparcia oraz powszechność stosowania tego układu w urządzeniach.

1.2.2.4. Wybór mikrokontrolera

Porównanie układów pokazało, że prosta ATmega328p znacząco odstaje od pozostałych układów. Mimo, że jest najprostszym z układów, nie mógł zostać wykorzystany w pracy, ze względu na wysokie zużycie energii. Problemem mogła okazać się również niewielka liczba liczników. W tym przypadku, najlepiej wypadł STM32F303K8. Niestety, jego wadą było wysokie zużycie energii w trakcie pracy. Z tego powodu, ostateczny wybór padł na nRF52840. Podsumowanie porównania, obrazuje poniższa tabela.1.2

	ATmega328p	STM32F303K8	nRF52840
FLASH	32kB	64kB	1MB
Prąd pracy	9.3-14 mA	12.9-34 mA	2.8-6.3 mA
Low Power	40-66 μA	0.93-18.57 μA	0.4-17.37 μA
Liczniki	2x8bit, 1x16bit	1x32bit, 2x24bit, 8x16bit	5x32bit
I^2C	✓	✓	✓
UART	1	3	2
Przerwania zewnętrzne	2	16	48
System operacyjny	Brak	Free RTOS	Zephyr RTOS

Tabela 1.2. Porównanie dostępnych mikroprocesorów

1.2.3. Modem GSM/LTE

Modem GSM/LTE to układ pozwalający na komunikację przez sieci komórkowe 4G lub GSM. Zastosowanie tego typu komunikacji, pozwoliło zapewnić dostęp do sieci niemal niezależnie od miejsca w którym znajdowało się urządzenie. Ponieważ procedura autentykacji do sieci jest czasochłonna i wymaga radzia, działającego z dużą mocą. Zgodnie z przewidywaniami, był jest to układ o największym zużyciu energii. Minimalnym wymaganiem dotyczącym tego układu, jest obsługa SMS. Podczas rozważań nad wyborem modemu, dodatkową zaletą była obsługa zapytań HTTP. Pozwoliła ona na przetestowanie alternatywnego podejścia do problemu powiadamiania.

1.2.3.1. SIM800L

SIM800L to jeden z najpopularniejszych układów do amatorskich rozwiązań. Jego największymi zaletami są cena i mnogość dostępnych bibliotek czy przykładów. Niestety, głównie dedykowanych do mikrokontrolera ATmega328p. Zastosowanie go z innymi układami nie jest jednak problemem, ponieważ działa on w oparciu o tak zwane komendy AT, wysyłane przez UART. Pobór prądu tego modemu, mieści się pomiędzi 150, a 450mA podczas transmisji danych. oraz 50-60μ w trybie PowerDown. Co ważne, podczas uruchamiania, układ pobiera do 2A prądu, co jest wartością bardzo dużą, w przypadku układów zasilanych bateryjnie. Niestety, układ ten działa z przestarzałymi już sieciami GSM i nie obsługuje zapytań HTTP.[7]

1.2.3.2. Quectel EC20 R2.1

Jest to jeden z łatwo dostępnych na rynku układów, działających w oparciu o technologię LTE. komunikacja z nim, przebiega przy użyciu komend AT. Układ, wspiera możliwość wysyłania zapytań HTTP. Pobór prądu modemu, osiąga 300mA, jednak w tybie PowerDown, wynosi jedynie 11μA. Jest to więc wartość ponad pięciokrotnie mniejsza, niż w przypadku SIM800L. Jest to ważne, ponieważ projektowane urządzenie miało pracować większość czasu w trybie całkowitego uśpienia. W przypadku tego układu, problemem było również znalezienie pełnej dokumentacji.

1.2.3.3. SKY66430

Jest to produkt firmy Skyworks, specjalizującej się w tworzeniu półprzewodników do układów radiowych i komunikacji mobilnej. Wybrany modem, działa w technologii LTE-M, co wymagało zakup specjalnych kart SIM. Układy LTE-M, zużywają znacznie mniej energii niż układy komunikujące się przez zwykłą sieć 4G. Są też w stanie transmitować z miejsc, o bardzo słabym zasięgu. SKY66430 podczas transmisji pobiera 150-250mA prądu, a w trybie PowerDown jedynie 1μA. Wartości te, uczyniły go więc najlepszym z rozważanych układów. Dodatkowym jego atutem jest obsługa zapytań HTTP. [8]

	SIM800L	Quectel EC20 R2.1	SKY66430
Prąd pracy	150-450 mA	300 mA	150-250 mA
PowerDown	50-60 μA	11 μA	1 μA
SMS	✓	✓	✓
HTTP	✗	✓	✓

Tabela 1.3. Porównanie dostępnych mikroprocesorów

1.2.3.4. Wybór modemu

Analizując powyższe dane, nie sposób było wybrać układ inny, niż SKY66430. Jest to układ o zdecydowanie najniższym zużyciu energii. Co więcej, wspiera on zapytania HTTP, co pozwoliło na przeprowadzenie dodatkowych testów projektu.

1.2.4. GPS

GPS, czyli Global Positioning System, to system nawigacji oparty na lokalizacji satelitarnej. Jego zadaniem, jest dostarczenie użytkownikom informacji o ich położeniu. W przypadku projektowanego urządzenia, kluczowym było pozyskanie możliwie dokładnej lokalizacji. Tak jak w przypadku pozostałych komponentów, układ powinien był charkteryzować się jak najniższym zużyciem energii. Ponieważ układ ten mógł okazać się najbardziej energochłonny, dodatkowym wymaganiem była możliwość jego wyłączenia. Istotnym elementem była również dokładność określanego położenia. Jako wystarczająca, uznana została dokładność 100m.

1.2.4.1. Beitian BN-220

Jest to jeden z najtańszych układów dostępnych na rynku. Układ, komunikuje się przy użyciu interfejsu UART, przesyłając wiadomości NMEA. Są to zdania, zapisane kodem ASCII, ustandaryzowanej strukturze. Zgodnie z dokumentacją, układ pobiera około 50mA i jest zasilany napięciem 5V.[9]

1.2.4.2. Mediatek MT3339

Kolejny układ, zasilany jest napięciem 3.3V i pobiera około 3mA prądu. Tak, jak BN-220, komunukuje się przy użyciu wiadomości NMEA, jednak korzystając z magistrali I^2C. Dodatkowo, układ ten,

posiada wewnętrzny zegar czasu rzeczywistego, pozwalający na wybudzenie układu ze stanu uśpienia. Ciekawą funkcjonalnością tego modułu, jest obsługa EPO. Jest to predykcja położenia satelit, na podstawie czasu i zapisanego w pamięci pliku. Dzięki temu rozwiązaniu, można znacząco skracać czas określania lokalizacji. Dokładność urządzenia to około 3m, co jest wartością znacznie większą niż zakładana.[10]

1.2.4.3. Mediatek MT3333

MT3333 to układ bardzo zbliżony do wspomnianego wcześniej MT3339. Różnią się one zużyciem energii. W przypadku MT3333, pobór prądu wynosi od 5 do 19mA. Podobnie jak poprzednik, komunikuje się on przy użyciu wiadomości NMEA, przesyłanych magistralą I^2C.[11]

1.2.4.4. Wybór modułu GPS

Pomimo, że układ MT3339 cechuje się najmniejszym zużyciem energii oraz posiada interesujące funkcjonalności, zdecydowano się na wybór modułu Mediatek MT3333. Wynikło to bezpośrednio z faktu, że w momencie tworzenia pracy, dostępna była tylko dokumentacja do układu MT3333. Ponieważ jednak układy te są zgode ze standardem oraz korzystają z tego samego interfejsu komunikacji, możliwą byłaby podmiana układów.

	BN-220	MT3339	MT3333
Prąd pracy	50 mA	3 mA	5-19 mA
Interfejs	UART	I^2C	I^2C

Tabela 1.4. Porównanie dostępnych modułów GPS

1.3. Podobne rozwiązania obecne na rynku

Podobne rozwiązanie, zaproponowała firma Specialized. Inżynierowie stworzyli urządzenie o nazwie ANGI, widoczne na rysunku 1.1. Jest to mały układ, wyposażony w akcelerometr i żyroskop. Komunikuje się on przy użyciu Bluetooth z ich autorską aplikacją na telefon z iOS lub Androidem. Urządzenie, zamontowane na kask, wysyła powiadomienie do aplikacji, gdy rowerzysta uderza kaskiem w przeszkodę. Rozwiązanie to, ma szereg zalet, takich jak prostota budowy i bardzo niskie zużycie energii. Jest ono jednak uzależnione od telefonu, który w przypadku wycieczek górskich, czasem wygodniej zostawić w domu.

Rys. 1.1. Urządzenie ANGI, stworzone przez firmę Specialized.[12]

2. Prototyp urządzenia

Głównym zagadnieniem pracy było stworzenie kompletnego urządzenia, które wykryje niebezpieczne zdarzenie i poinformuje o tym zdefiniowanego użytkownika. W niniejszym rozdziale, przeprowadzono testy, mające pomóc stworzyć odpowiednie algorytmy. Opisano również kolejne kroki procesu tworzenia urządzenia, a także rozważono różne podejścia w rozwiązaniu problemów. Omówione zostały również poszczególne fragmenty działającego urządzenia.

Tę część pracy, rozpoczęto od przeprowadzenia podstawowych eksperymentów, mających pokazać, jakich przyspieszeń można spodziewać się w przypadku rzeczywistego zderzenia na rowerze. Testy, rozpoczęo od stworzenia urządzenia, mającgo zbierać surowe dane z akcelerometru. Ze względu na łatwość i szybkość implementacji, wykorzystano platformę Raspberry Pi Zero W, z systemem operacyjnym Raspbian Lite. Korzystając z gotowch bibliotek, napisany został skrypt, który po uruchomieniu tworzył plik tekstowy, do którgo zapisywał surowe dane, pobrane z akcelerometru. Sam akcelerometr, ustawiony był na częstotliwość próbkowania 416Hz i skalę $\pm$8g.
Ze względów czysto praktycznych, tj. w celu zabezpieczenia płytek przed uszkodzeniem, wykonany został model 3D prostej obudowy, który został następnie wydrukowany na drukarce 3D. Rysunek 2.1a przedstawia Raspberry wykonanej na potrzeby projektu obudowie. Zgodnie z rysunkami technicznymi płytki, wykorzystano otwory montażowe. Następnie, stworzono dodatkową warstwę izolacyjną między płytkami, do której przymontowano płytkę z akcelerometrem.(Rysunek 2.1b) Gotowy układ, zasilany był przy użyciu powerbanku, połączonego przez USB. Obudowa, zamknięta była drukowaną pokrywką. Rozwiązanie przedstawia rysunek 2.1c.

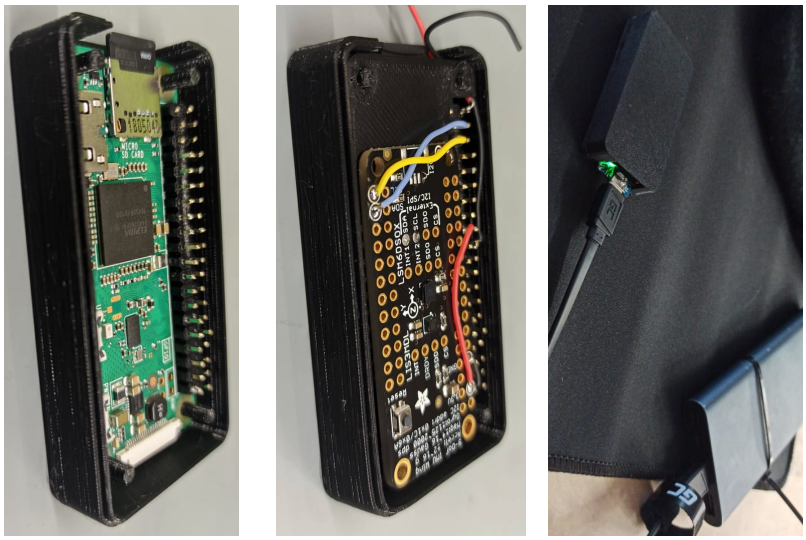
2.1. Opracowanie algorytmu wykrywania kolizji

Cały układ, zasilany był przy użyciu zewnętrznej baterii, zamontowanej na siodełku roweru. Ze względu na bezpieczeństwo, testy przeprowadzane były na rowerze bez rowerzysty. Sam układ, zamontowany został na rurze podsiodłowej. Ponieważ jest to prawie centralny punkt roweru, oczekiwano, że przyspieszenia w przypadku silnego wyrzucenia koła w powietrze, będą mniejsze. Eksperyment przeprowadzono przez rozpędzenie roweru do prędkości około $10\frac{km}{h}$. Kolejno wykonano:

– Zderzenie roweru z drzewem

– Upadek na kamieniach

– Upadek ze stromego zbocza

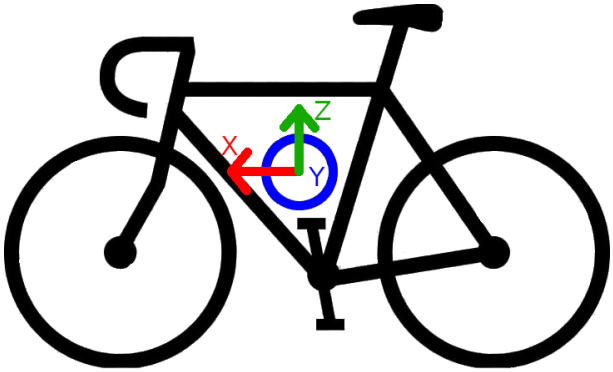


(a) Raspberry Pi Zero w drukowanej obudowie (b) Płytka z akcelerometrem w obudowie (c) Urządzenie testowe zasilane zewnętrzną baterią

Rys. 2.1. Urządzenie testowe do zbierania danych

2.1.1. Analiza zebranych danych

W celu analizy danych, stworzono prosty skrypt w języku Python. Pobierał on surowe dane z pliku, przetwarzał, a następnie rysował zależności czasowe $a(t)$ dla każdej z osi. Ponieważ akcelerometr zwraca liczbę 16-bitową ze znakiem, należało uwzględnić ten fakt w konwersji, ponieważ Python posiada dynamiczne typy danych, co mogło zakłamać wyniki. Zebrane dane, okazały się być znacząco zaszumione. Z tego powodu, przefiltrowane zostały filtrem dolnoprzepustowym. Rysunek 2.2 przedstawia orientację urządzenia na rowerze. W trakcie analizy danych, przyjęto zaznaczone na nim kierunki osi. Podczas każdego z testów, rower z przymocowanym urządzeniem, rozpędzono do



Rys. 2.2. Orientacja urządzenia zamontowanego na rowerze

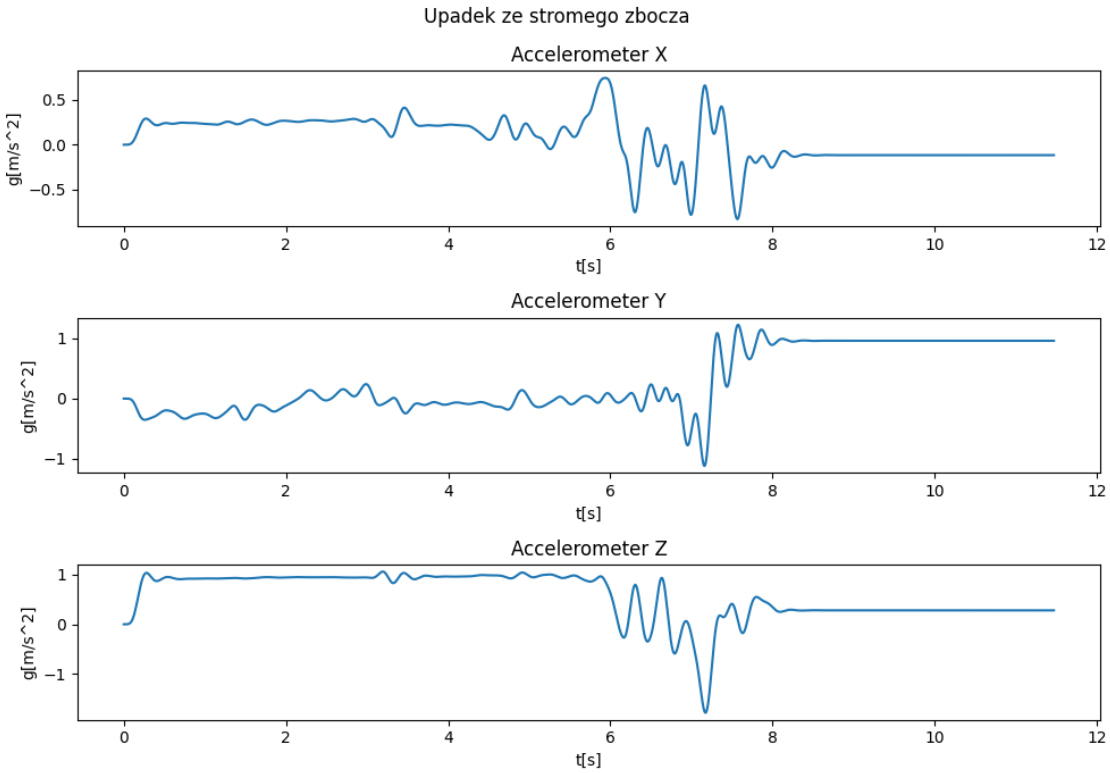
około $1\frac{m}{s}$. Zgodnie z przewidywaniami, w większości przypadków, przyspieszenia układu sięgały ok. 1g. Należało również zwrócić uwagę na zwroty przyspieszeń, które zmieniały się po obrocie roweru. Poniżej przedstawiono trzy analizowane przebiegi, na podstawie których utworzone zostały algorytmy.

Rysunek 2.3 przedstawia zapis upadku ze stromego zbocza. Jest to przypadek gdy rowerzysta górski podczas uskoku, traci kontrolę nad rowerem, lub wypada z trasy. Przez około pół sekundy rower swobodnie spadał, powoli przechylając się w kierunku przedniego koła. Około szóstej sekundy uderzył nim o zbocze, co widać wyraźnie zarówno na osi X, jak i Z. Następnie odbity rower ponownie uderzył w ziemię, tym razem pochylając się po odbiciu na bok. Kolejne uderzenie, było więc uderzeniem bocznym, widocznym jako głęboki dół na osi Y. Niemal natychmiast, rower obrócił się kołami do góry, co widać na osi Z jako przyspieszenie równe -1g. Po siódmej sekundzie rower zaczął sunąć bokiem wzdłuż zbocza i szybko wyhamowywać. akcelerometr zarejestrował to jako oscylajcje na osi Y pomiędzy 7, a 8 sekundą. Na wykresie, można było również zauważyć, że po zdarzeniu, rower zatrzymał się na jednym z boków, ponieważ oś Y miała stałą wartość bliską 1g.

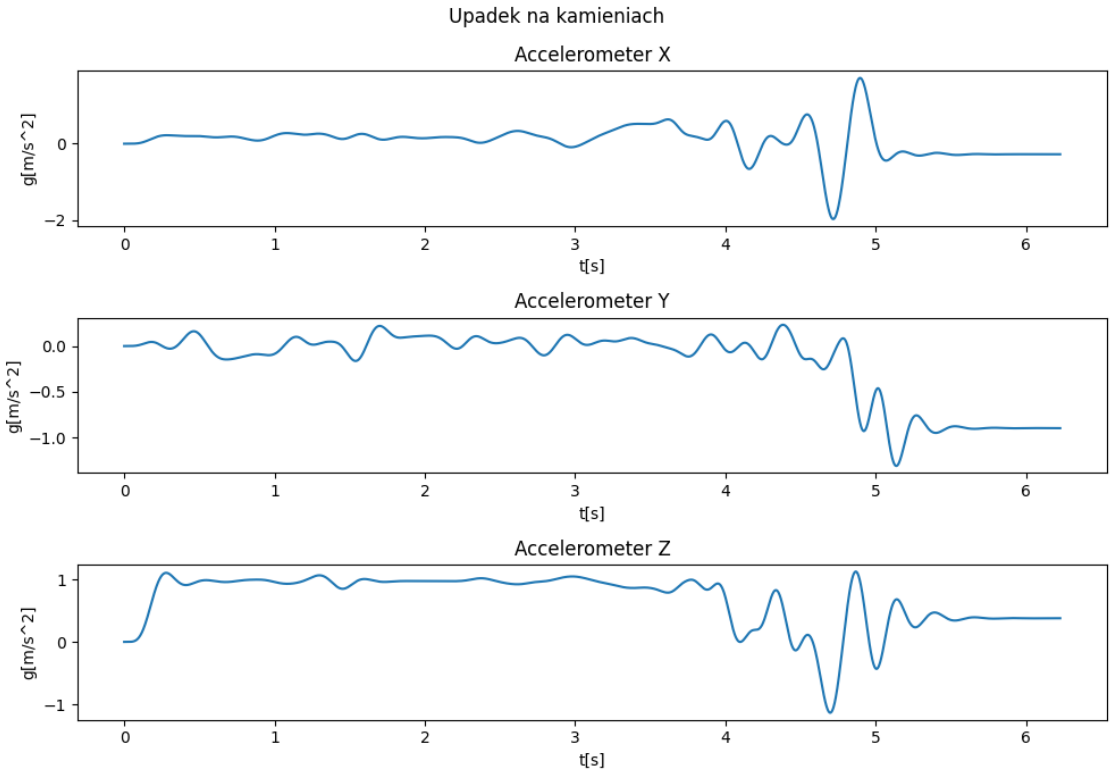
Kolejny z rysunków (2.4), przedstawia sytuację gdy rowerzysta jadąc po kamieniach, uderza tylnym kołem o jeden z nich i traci kontrolę nad rowerem. Sytuacja ta charakterysuje się silnymi oscylacjami osi X i Z, przy względnie stabilnej osi Y. W czwartej sekundzie, rower uderzył tylnym kołem w kamień. Koło zostało wyrzucone do góry, z przyspieszeniem około 1g. Po 500ms rowerzysta rower uderzył z całą energią przednim kołem w ścieżkę, osiągając przyspieszenie prawie 2g. W tej chwili pojazd testowy jeszcze raz odbił się od, a następnie bokiem uderzył w drzewo przy scieżce.

Rysunek 2.5 przedstawia sytuację, gdy rowerzysta tracąc kontrolę nad rowerem, uderza w drzewo. W tym przypadku rower po około 4.5s uderza w drzewo, co odczytane zostało jako wzrost do 0.5g na osi X. Niestety, nie było to uderzenie czyste, ponieważ duża część energii rozłożyła się na odbicie roweru od drzewa. Zapis osi Y pokazuje, że po uderzeniu, rower zaczął obracać się. Ostatecznie jednak, zatrzymał się na jednym z boków.

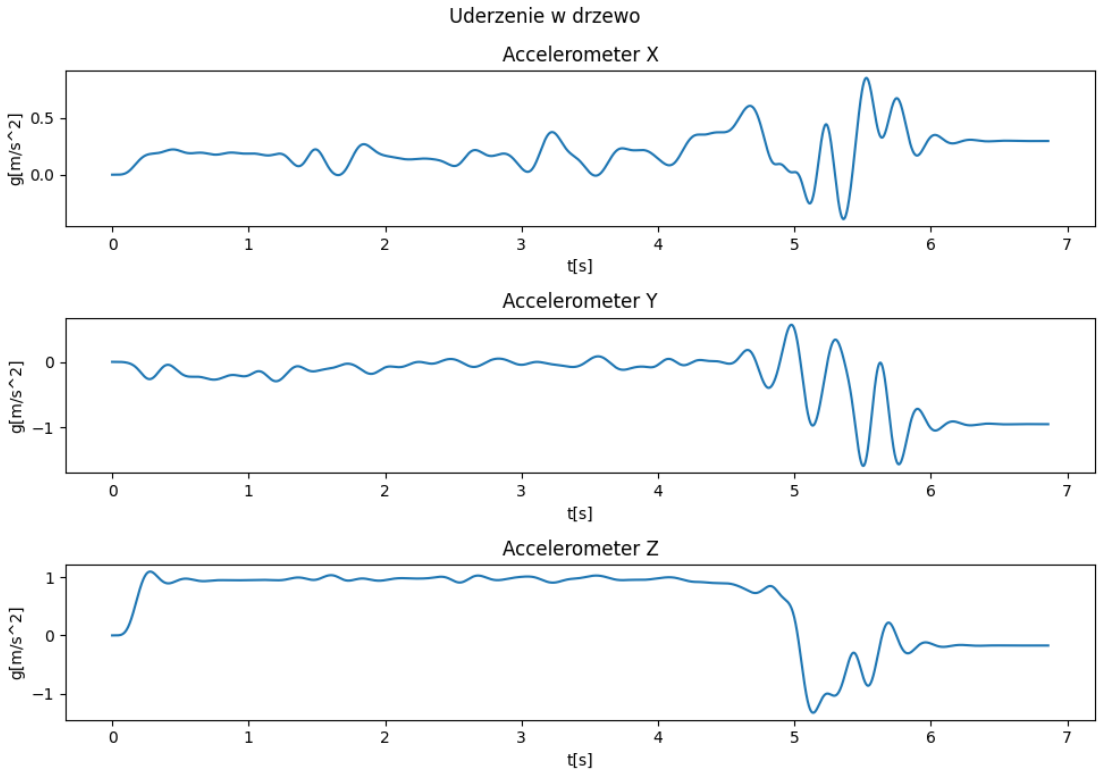
Przeprowadzone testy, pozwoliły na sprawdzenie przeciążeń działających na rower w trakcie wypadku. Pierwszym z nasuwających się wniosków był fakt, że niepotrzebnie ograniczona została częstotliwość próbkowania akcelerometru. W przypadku Raspberry Pi, magistrala I^2C, pozwala na transmisję 12,5kB/s. Ponieważ jeden zestaw danych zawierał 12 bajtów, częstotliwość próbkowania mogła wynosić aż 1kHz. Tymczasem, ustawiona została częstotliwość 416Hz. Ustawienie to wynikało z obawy, że kod w języku skryptowym nie nadąży za większą częstotliwością. Kolejnym z wniosków, była konieczność stosowania filtrów dolnoprzepustowych. Surowe zapisy z akcelerometru, były silnie zaszumione, na co również wpływ mogła mieć mała częstotliwość próbkowania względem szybkości zmian podczas wypadku. Mimo tego, testy dostarczyły cennych danych na podstawie których stworzono algorytmy, opisane w podrozdziale 2.1.2



Rys. 2.3. Przefiltrowany zapis upadku roweru ze stromego zbocza



Rys. 2.4. Przefiltrowany zapis upadku roweru na kamieniach



Rys. 2.5. Przefiltrowany zapis uderzenia roweru w drzewo

2.1.2. Maszyny stanów wykrywające wypadek

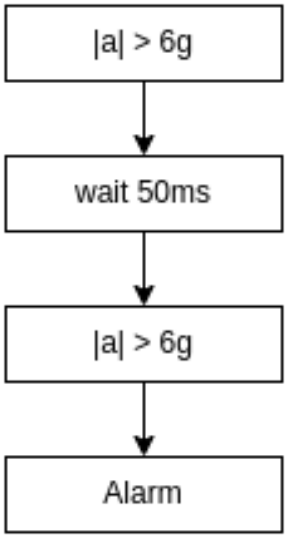
Na podstawie analizy danych z podrozdziału 2.1.1 zdecydowano się na implementację czterech, niezależnych maszyn stanów. Każda z nich, odpowiadać będzie analizowanym przypadkom. Istotnym elementem pracy było możliwie największe uproszczenie maszyn, aby przyspieszyć działanie układu i zminimalizować zużycie energii, bez utraty skuteczności.

2.1.2.1. Wykrywanie przeciążenia w dowolnej osi

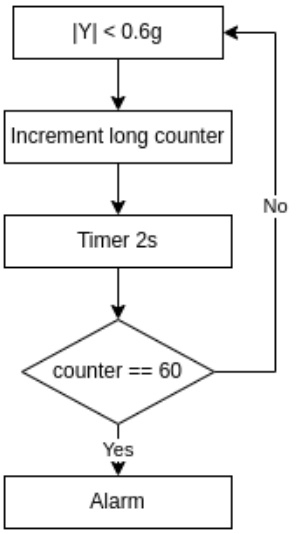
Pierwsza z maszyn stanów, służy wykryciu znaczących przeciążeń na dowolnej z osi. Podczas testów zauważono, że podczas każdego z wydarzeń, wysoka wartość przeciążenia odróżnialna była od szumu na podstawie czasu trwania. Określono, że jeśli po około 50ms próbka utrzymuje wysoką wartość, to nastąpiło silne przeciążenie. Ponieważ zdecydowano się skorzystać z wbudowanych w akcelerometr maszyn stanów, można było skorzystać z ich wewnętrznych liczników. Opisywana maszyna stanów, gdy dowolna z osi przekroczy wartość $\pm$6g, ustawia licznik na 50ms. Gdy przez ten czas, nie wystąpi próbka o wartości mniejszej niż 6g, to wyzwolony zostaje alarm. Duża wartość przyspieszenia, pozwoliła uniknąć fałszywych alarmów podczas bezkolizyjnej jazdy. Maszynę przedstawia diagram 2.6a.

2.1.2.2. Maszyna stanów wykrywająca przewrócony rower

Dwie kolejne maszyny stanów, przedstawione na diagramie 2.6b, to algorytmy wykrywające przewrócenie roweru. Podczas analizy danych zauważono, że każdy wypadek, kończył się przechyleniem roweru na jeden z boków. Z tego powodu, zaimplementowano licznik, który co 2 sekundy próbkuje wartość na osi Y i sprawdza, czy nie przekracza ona wartości bezwzględnej 0.6g. Jeśli przez dwie minuty, rower nie wróci do pozycji jazdy, wyzwolony zostaje alarm. W praktyce, opisana maszyna musiała zostać zaimplementowana w akcelerometrze jako dwie, niezależne, z uwagi na brak funkcji wartości bezwzględnej w układzie.



(a) Wykrycie silnego przeciążenia

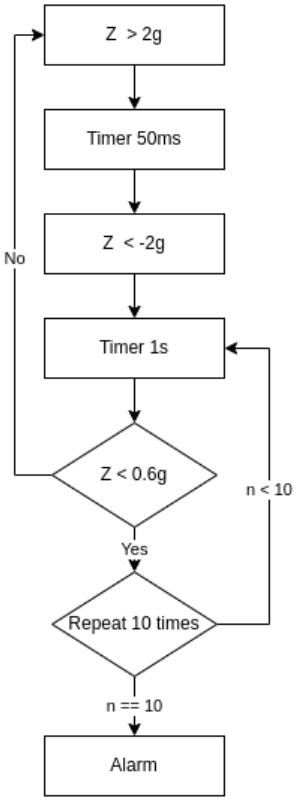


(b) Wykrycie przewrócenia roweru

Rys. 2.6. Maszyny 1-3

2.1.2.3. Maszyna stanów wykrywająca uderzenie i upadek

Ostatnia z maszyn stanów (rysunek), wykrywa uderzenie w osi Z, czyli np. najechanie na korzeń. Następnie, przez 60 sekund sprawdza, czy rower nie wrócił do wyjściowej pozycji, czyli zwykłej jazdy. W przypadku, gdy przez 10 sekund rower nie powróci do pionu, uruchomiony zostanie alarm. Maszyna ta, okazała się być najdokładniejszą z prezentowanych. Wynika to bezpośrednioz faktu, że oś Z jest osią najbardziej stabilną w trakcie jazdy. Pozostałe maszyny, stanowią więc jej uzupełnienie.



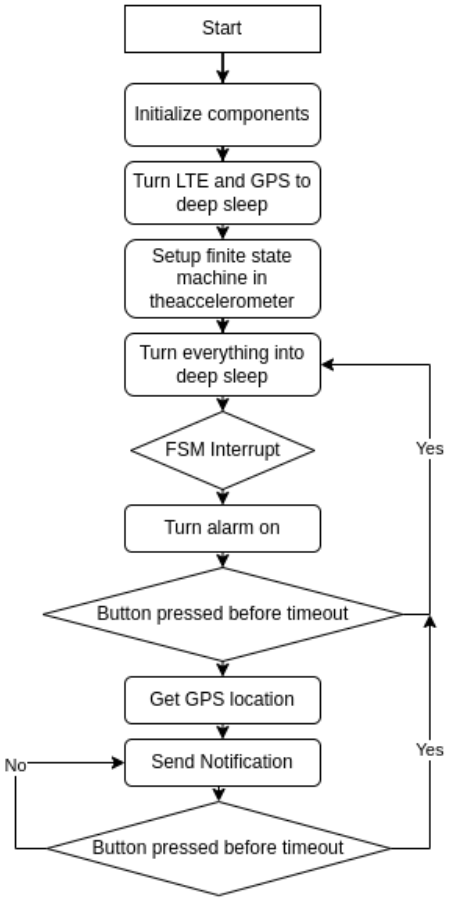
Rys. 2.7. Czwarta maszyna stanów

2.2. Opracowanie logiki urządzenia

2.2.1. Wysokopoziomowy algorytm działania urządzenia

Głównym zagadnieniem budowy urządzenia, było maksymalne wydłużenie czasu pracy na baterii. Z tego powodu, założono, że lokalizacja oraz połączenie z siecią, będą zestawiane tylko w momencie wystąpinia wypadku. Na podstawie rozważań oraz testów przeprowadzonych w podrozdziale 2.2.2, zdecydowano się na wykorzystanie maszyny stanów wbudowanej w akcelerometr. Dzięki temu, możliwym było ustawienie urządzenia w tryb głębokiego uśpienia, już po inicjalizacji urządzenia. Rysunek 2.8 obrazuje założoną logikę urządzenia. Po uruchomieniu układu, inicjalizowane są wszystkie komponenty. W przypadku LTE, wykonywana jest rejestracja do sieci. Ponieważ jest to urządzenie odpowiedzialne

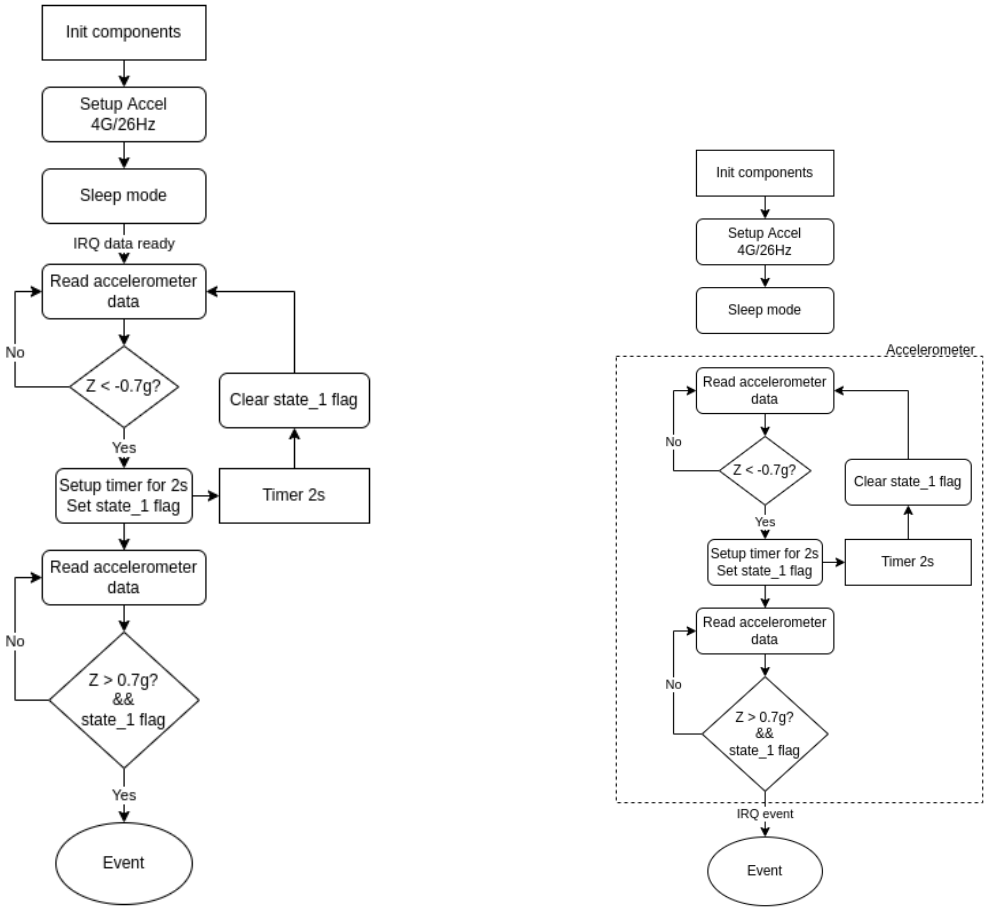
za bezpieczeńswo, rejestracja przy uruchomieniu może wykryć nieprawidłowości jak brak środków na koncie, brak karty SIM czy urwana antena modemu. Ta funkcjonalność, nie została jednak zaimplementowana na tym etapie rozwoju urządzenia. Po inicjalizacji podzespołów, wyłączony został zarówno modem oraz moduł GPS. Zgodnie z założeniami, są one uruchamiane w momencie wystąpienia alarmu, aby oszczędzić energię. Kolejnym krokiem, jest wgranie zrzutu pamięci akcelerometru, celem ustawienia maszyn stanów. Po jego wykonaniu, mikrokontroler przechodzi w tryb uśpienia. W tle, uruchomiony pozostaje stos Bluetooth, pozwalając połączyć się z urządzeniem i wpisać numer telefonu, na który wysłane zostanie powiadomienie. Wyzwolenie dowolnej z maszyn stanów, uruchamia przerwanie. Wyzwolenie przerwania, skutkuje uruchomieniem modułu GPS, a po poprawnym pobraniu lokalizacji, również modemu. Powiadomienie, jest wysyłane periodycznie, do momentu wyłączenia alarmu przyciskiem. Pomiędzy alarmami, wszystkie podzespoły pozostają wyłączone, celem oszczędzania anergii.



Rys. 2.8. Diagram przedstawiający logikę działania urządzenia

2.2.2. Wybór metody analizy danych przez urządzenie

Ponieważ jednym z najważniejszych elementów pracy, było osiągniecie jak najniższego zużycia energii, zdecydowano się na porównanie podejścia klasycznego, opartego na czytaniu danych i ich analizie przez mikrokontroler, z maszynami stanów realizowanymi przez akcelerometr. W tym celu, skonfigurowano prosty algorytm, wykrywający obrót urządzenia w osi Z. Akcelerometr skonfigurowano w tryb niskiego zużycia energii, a następnie wprowadzono mikrokontroler w tryb uśpienia. W tym stanie, układ oczekiwał na przerwanie, informujące o gotowości danych. Po jego wystąpieniu, następowało zczytanie danych z akcelerometru, a następnie porównanie ich do zdefiniowanej stałej. Po spełnieniu warunku obrotu o 180°, ustawiano licznik na 2 sekundy, resetujący stan maszyny. Jeśli w tym czasie, odczytano dane o przeciwnym zwrocie, maszyna stanów uznana była za zrealizowaną, o czym informowano przez wiadomość wypisaną na UART. Całość, zobrazowana jest na diagramie 2.9.



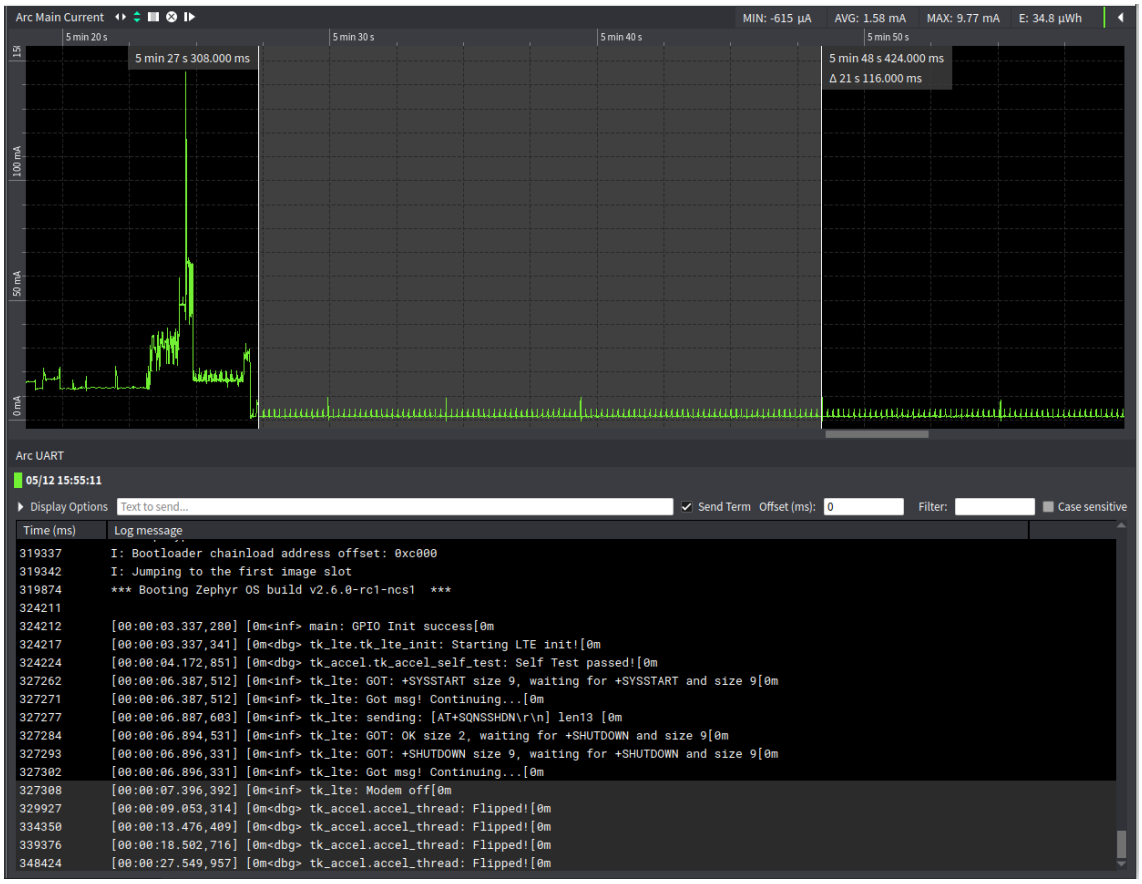
(a) Wykrywanie zdarzenia przez mikrokontroler

(b) Wykrywanie zdarzenia przez akcelerometr

Rys. 2.9. Analiza zdarzenia przez mikrokotroler lub akcelerometr

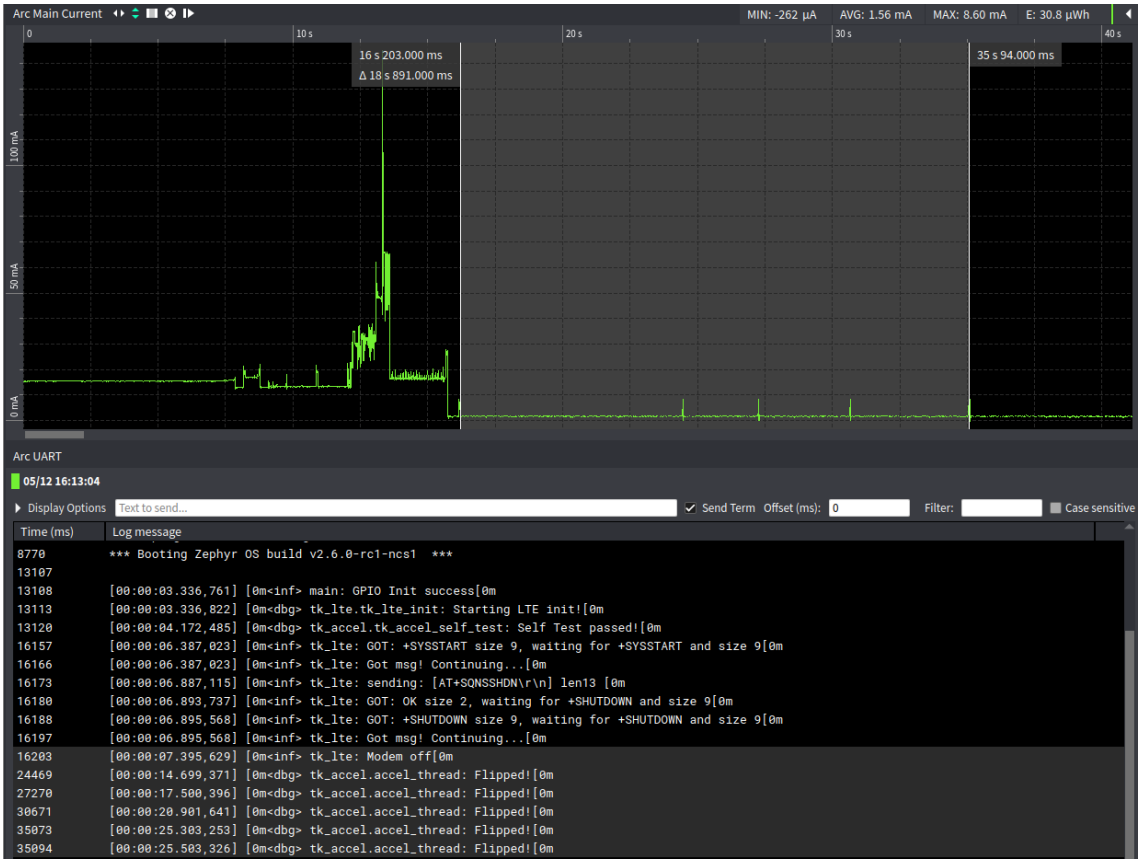
Każde rozwiązań, zostało zmierzone przy użyciu urządzenia Otii. Jest to urządzenie służące do precyzyjnej analizy energii pobieranej przez układ.[13] Na zapisie z programu(rys. 2.10) wyraźnie widać inicjalizację modemu LTE, który w trakcie przeprowadzania testów był już zintegrowany na płyce PCB.

Jest ona widoczna w postaci znaczącego wzrostu pobieranego przez układ prądu, sięgającego nawet 200mA. Gdy modem przechodzi w tryb głębokiego uśpienia, wyraźnie widać odczyty i przerwania, w postaci drobnych pików. Sam skok, wynikał z wywołania funkcji logującej, jednak w wyraźny sposób zaznaczał moment wystąpienia przerwania. W przypadku tego podejścia, pobór prądu wyniósł 1.58mA. Należy jednak zaznaczyć, że zastosowana tutaj maszyna stanów, była maszyną bardzo prostą. Jej rozbudowa, wiązałaby się z bardzo szybkim wzrostem zużycia energii. Kolenym z minusów była ilość generowanych przez akcelerometr przerwań. Każdy odczyt danych nie tylko zajmuje linię danych I^2C, ale również czas procesora.



Rys. 2.10. Pobór prądu w czasie, podczas analizy przez mikrokontroler

Podejście drugie, z maszyną stanów skonfigurowaną na akcelerometrze, okazało się zużywać średnio aż o 200μA mniej. Tak duża różnica, przy tak prostej funkcjonalności, była silnym argumentem za wykorzystaniem wbudowanej maszyny stanów. Dodatkowo, na rysunku 2.11 widać, że zniknęły gęste piki prądu. Wynikało to wprost z tego, że akcelerometr wystawia przerwanie tylko wtedy, gdy spełniona wykonana zostanie cała maszyna stanów. Powyższe rozważanie, pokazało więc zasadność stosowania akcelerometru analizującego dane niezależnie od mikrokontrolera. Dodatkowo, STMelectronics, stworzyło zestaw narzędzi, ułatwiających budowę urządzeń, w oparciu o ich układy. Przykładem jest program Unico-GUI[14], pozwalający programować akcelerometry w oparciu o płytki rozwojowe. Podczas tworzenia tej pracy, wykorzystano zestaw STEVAL-MKI109V3 + STEVAL-MKI197V1. Program Unico,

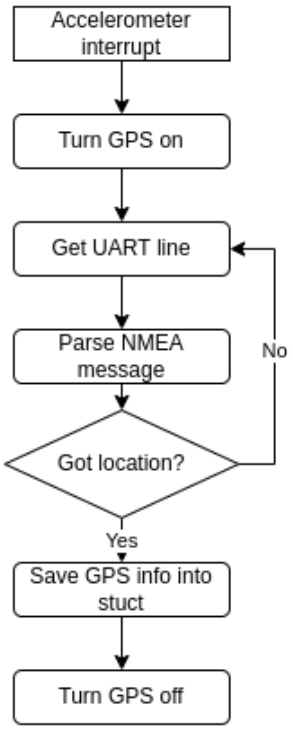


Rys. 2.11. Pobór prądu w czasie, podczas analizy przez akcelerometr

pozwala projektować maszyny stanów w oparciu o interfejs graficzny. Było to znaczące ułatwienie, ze względu na możliwość zapisania zrzutu pamięci akcelerometru do kodu w języku C. Dzięki temu, w łatwy sposób można było testować różne konfiguracje układu, unikając ręcznego nadpisywania rejestrów. Pomocną okazała się również nota aplikacyjna, wyjaśniająca sposób, w jaki maszyny stanów powinny być programowane.[15]

2.2.3. Implementacja pobierania lokalizacji

Zasada działania modułu MT3333 jest stosunkowo prosta. Zgodnie z dokumentacją[11], należało w odpowiedniej kolejności uruchomić przetwornice układu, zachowując właściwe odstępy czasowe. Gdy zostaną one uruchomione poprawnie, układ zacznie szukać lokalizacji, jednocześnie cały czas wysyłając wiadomości NMEA. Wyzwaniem okazało się tutaj parsowanie przychodzących wiadomości. Ostatecznie, do bufora dopisywane były bajty, do momentu otrzymania znaku nowej linii. Przychodzące linie były testowane przez algorytm. Po otrzymaniu wiadomości zawierającej lokalizację, wiadomość parsowana była do struktury, zawierającej szerokość i wysokość geograficzną w postaci tablic bajtów. Po otrzymaniu lokalizacji, moduł był wyłączany, aby nie marnować energii. Schemat działania, przedstawiono na diagramie 2.12



Rys. 2.12. Działanie modułu GPS

2.2.4. Implementacja powiadamiania o zdarzeniu

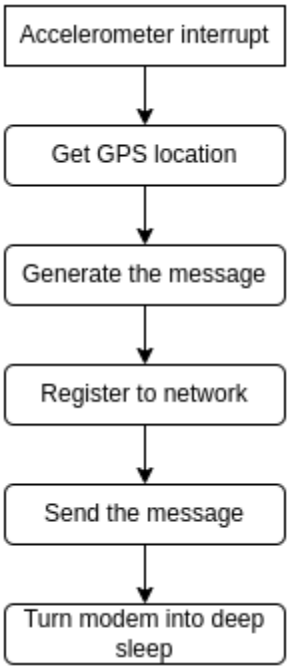
Dzięki zastosowaniu układu SKY66430, podczas tworzenia prototypu, przetestowano dwa, zupełnie różne podejścia do problemu powiadamiania:

– Zapytanie HTTP

– Powiadomienie SMS

Każde z podejść, zakładało logikę, przedstawioną na rysunku 2.13. Całość komunikacji z modemem, realizowana była przy użyciu komend AT. Ważnym problemem, była kolejność wysyłanych do niego poleceń. Z tego powodu, zaimplementowana została funkcja oczekująca na odpowiedź od modemu, potwierdzającą przetworzenie wysłanej komendy. Dzięki jej zastosowaniu, zminimalizowana została szansa niepowodzenia transmisji, ponieważ w przypadku zbyt długiego oczekiwania, modem był resetowany przeznaczonym do tego pinem.

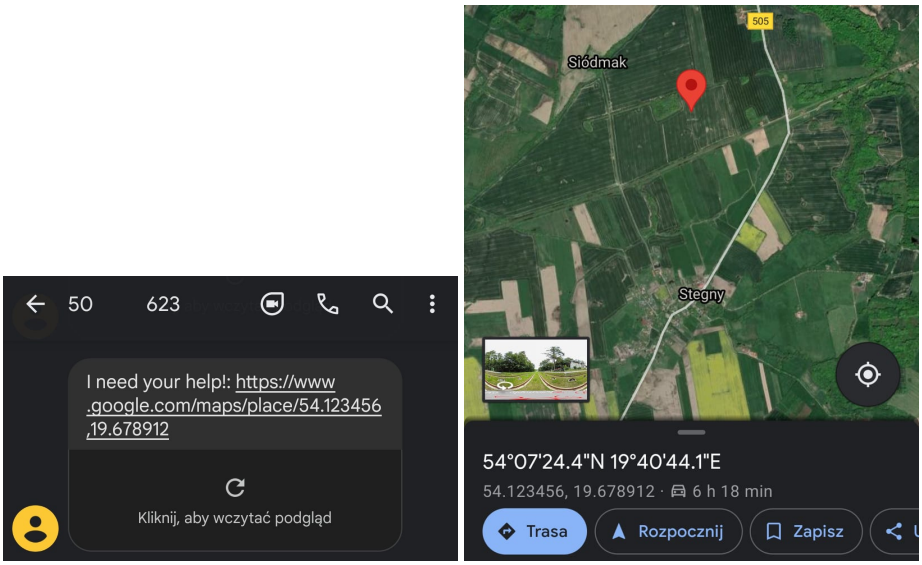
Na potrzeby przetestowania zapytań HTTP, stworzone zostało proste gniazdo TCP w JavaScript. Następnie, wykorzystując gotowe biblioteki, zaprogramowano bota na platformę Discord. Jest to platforma oparta o otwarty kod źródłowy, na której użytkownicy tworząc kanały głosowe i tekstowe, komunikują się w zorganizowany sposób. Bot, jest wirtualnym użytkownikiem, wykonującym zaprogramowane akcje. Mogą one obsługiwać specjalne funkcje serwera, lub monitorować zachowanie użytkowników. W



Rys. 2.13. Procedura wysłania powiadomienia o zdarzeniu

omawianym podejściu, bot mógł wysłać powiadomienie z otrzymaną lokalizacją, do wszystkich użytkowników serwera. Ponieważ platforma posiada bardzo dobrze rozwiniętą aplikację mobilną, powiadomienie mogłoby być szybko odczytane przez jednego z użytkowników. Jednocześnie, nie byłoby wymagane definiowanie jednego, numeru docelowego. Ostatecznie jednak, rozwiązanie to zostało uznane za niepasujące do urządzenia na tym etapie. Takie podejście, wymagało stworzenia stabilnego kodu zewnętrznej aplikacji (bota). Jednocześnie, zapytanie HTTP przesyła dużo danych, w porównaniu do SMS. Mogło to okazać się problemem w miejscach o słabym zasięgu.

Finalnie, zdecydowano się na wysłanie prostego powiadomienia tekstowego, przy użyciu wiadomości SMS. Za tym rozwiązaniem, przemawiała stabilność i niezawodność sieci komórkowych, mających obecnie niemal całkowite pokrycie, nawet w górach. Wysłanie SMS, pozwoliło również uniezależnić urządzenie od innych aplikacji. Ponieważ współrzędne geograficzne nie są szczególnie użyteczne bez mapy, wysyłane powiadomienie SMS zawiera zawiera link do map Google. Link ten jest tworzony w kodzie układu, wykorzystując funkcje bibliotek standardowych. Rysunek 2.14 przedstawia powiadomienie, wysyłane przez urządzenie. Numer telefonu, na który powinno zostać wysłane powiadomienie, wpisywany jest do urządzenia przy użycu Bluetooth. W przypadku, gdy użytkownik nie wpisze żadnego numeru, urządzenie wyśle powiadomienie na ostatni znany mu numer. Jeśli pamięć przeznaczona na numer telefonu będzie pusta, w trakcie uruchamiania urządzenia wyzwolony zostanie buzzer który pozostanie włączony tak długo, aż użytkownik nie wpisze numeru.

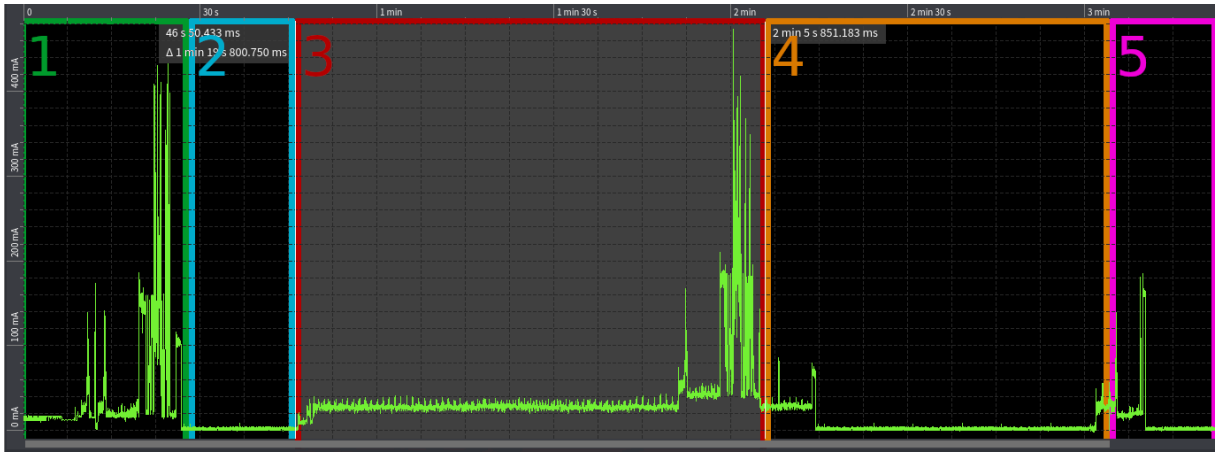


(a) Odebrana wiadomość SMS z lokalizacją (b) Lokalizacja na mapie Google

Rys. 2.14. Wysyłane powiadomienie z lokalizacją

3. Testy urządzenia

3.1. Analiza poboru prądu układu



Rys. 3.1. 1. inicjalizacja urządzenia. 2.Tryb uśpienia. 3. Wyzwolenie alarmu oraz wysłanie lokalizacji. 4.Oczekiwanie wysłanie kolejnego powiadomienia. Buzzer włączony. 5.Przerwanie alarmu przyciskiem, uśpienie układu.

Bibliografia

[1] *Dokumentacja układu MPU-6050*.

[2] *Dokumentacja układu LSM9DS1*.

[3] *Dokumentacja układu LSM6DSOX*.

[4] *Dokumentacja układu ATmega328p*.

[5] *Dokumentacja układu STM32F303K8*.

[6] *Dokumentacja układu nRF52840*.

[7] *Dokumentacja układu SIM800L*.

[8] *Dokumentacja układu SKY66430*.

[9] *Dokumentacja układu Beitian BN-220*.

[10] *Opis producenta układu MT3339*.

[11] *Dokumentacja układu MT3333*.

[12] *Strona producenta rozwiązania ANGI*.

[13] *Strona producenta Otii*.

[14] *Program Unico-GUI*.

[15] *Nota aplikacyjna maszyn stanów układu LSM6DSOX*.