

Politechnika Poznańska
Wydział Elektryczny
Instytut Automatyki i Inżynierii Informatycznej

Praca dyplomowa inżynierska

**UKŁAD POMIAROWY DO BOLIDU KLASY FORMUŁA STUDENT
(PROJEKT ZESPOŁOWY)**

Marcin Aftowicz
Jakub Baranowski

Promotor
dr inż. Dariusz Janiszewski

Poznań, 2015 r.

Podziękowania:

W tym miejscu można wstawić podziękowania.

Imię i Nazwisko autora 1

Lub nie wstawiać żadnych.

Imię i Nazwisko autora 2

Spis treści

| | | |
|----------|--|-----------|
| 1 | Wstęp oraz motywacja | 1 |
| 1.1 | Struktura pracy | 1 |
| 1.2 | PUT Motorsport | 1 |
| 1.3 | Analiza problemu | 1 |
| 1.4 | Podział prac | 2 |
| 2 | Omówienie protokołów komunikacyjnych | 3 |
| 2.1 | Główna magistrala komunikacyjna - CAN | 3 |
| 2.1.1 | CAN w modelu ISO/OSI | 3 |
| 2.1.2 | Budowa ramki CAN | 5 |
| 2.1.3 | Filtry akceptacyjne | 5 |
| 2.2 | Archiwizacja danych - SD | 6 |
| 2.2.1 | FatFs | 7 |
| 2.2.2 | SD Bus | 7 |
| 2.2.3 | Serial Peripheral Interface (SPI) | 8 |
| 2.2.4 | Direct Memory Acces (DMA) | 8 |
| 2.3 | Komunikacja bezprzewodowa - XBee | 8 |
| 2.3.1 | Universal Asynchronous Receiver and Transmitter - UART | 8 |
| 3 | Realizacja projektu | 9 |
| 3.1 | Model systemu | 9 |
| 3.2 | Główny komputer pokładowy | 9 |
| 3.2.1 | Mikrokontroler | 9 |
| 3.2.2 | Obsługa magistrali CAN | 11 |
| 3.2.3 | Obsługa karty SD | 11 |
| 3.2.4 | Obsługa modułu XBee | 11 |
| 3.2.5 | Zasilanie | 11 |
| 3.2.6 | System przerwań (NVIC) | 11 |
| 3.3 | Rozproszone jednostki pomiarowe - HUB | 11 |
| 3.3.1 | Wprowadzenie | 11 |
| 3.3.2 | Układ główny | 11 |
| 3.3.3 | Separacja sygnałów analogowych | 11 |
| 3.3.4 | Zasilanie | 11 |
| 3.4 | Zdalny interfejs użytkownika | 11 |
| 3.4.1 | Wymagania sprzętowe | 11 |
| 3.4.2 | Moduł odbiorczy XBee | 11 |
| 3.4.3 | Grphical User Interface (GUI) | 11 |
| 4 | Badania eksperymentalne | 13 |

| | |
|-----------------------|-----------|
| 5 Podsumowanie | 15 |
| Spis tablic | 17 |
| Spis rysunków | 19 |
| Literatura | 21 |

Streszczenie

Niniejsza praca inżynierska stanowi dokumentację projektu realizowanego przez grupę elektryczną w zespole PUT Motorsport w ramach konkursu *Formula Student*. Jako studenci IV roku Automatyki i Robotyki Wydziału Elektrycznego Politechniki Poznańskiej oraz członkowie Koła Naukowego Sensorzadeklarowaliśmy chęć uczestnictwa w konkursie i przyłączenie się do zespołu.

Rozdział 1

Wstęp oraz motywacja

1.1 Struktura pracy

Jak wygląda rozkład rozdziałów i co w nich opisano.

1.2 PUT Motorsport

Formula Student (FS) to najbardziej prestiżowy, europejski konkurs w dziedzinie Motorsport, prowadzony przez *Institution of Mechanical Engineers*. Wspierany przez przemysł, konkurs ma na celu inspirowanie i rozwijanie przedsiębiorczości i innowacyjności u młodych inżynierów. Uczelnie z całego świata mają za zadanie zaprojektować i zbudować w pełni funkcjonalny samochód wyścigowy, który ukończy statyczne i dynamiczne konkurencje testujące zarówno wiedzę studentów, jak i wydajność pojazdu.

Zespół wyścigowy Politechniki Poznańskiej - PUT Motorsport to drużyna składająca się z 23 studentów tworzących 6 grup projektowych, która podjęła się wyzwania wystartowania w konkursie i skonstruowania własnego bolidu. Wyróżnia się 6 technicznych grup projektowych odpowiedzialnych za: poszycie, zawieszenie, aerodynamikę, napęd, materiały oraz elektronikę. Wszystkie grupy pracują równolegle i dbają o zachowanie spójności oraz kompatybilności projektowanych elementów. Specyfika takiej pracy nakłada ograniczenia i wymusza elastyczność rozwiązań, tak aby mogły one zostać dopasowane do całości [3][8].

1.3 Analiza problemu

W ramach projektu PUT Motorsport zdecydowano się na utworzenie systemu elektronicznego, który umożliwi analizę pracy poszczególnych podzespołów pojazdu. We wstępnych fazach projektu konstruktorzy nie są w stanie sprecyzować swoich wymagań, dlatego opracowywane rozwiązanie musi być elastyczne i uniwersalne. Istnieje szereg cech, które musi spełniać system i szereg ograniczeń, które muszą zostać wzięte pod uwagę. W celu monitorowania pracy podzespołów, potrzebne jest urządzenie zbierające dane z rozproszonych w pojeździe czujników. Aby móc korzystać z tych danych należy je wyświetlić, a najlepiej zarchiwizować. System musi być odporny na zakłócenia, które powstaną np. w chwili zapłonu. Musi być modyfikowalny, tak aby spełnić możliwie wiele potencjalnych potrzeb konstruktorów (przy maksymalizacji możliwości ingerencji w jego strukturę, minimalizacja zmian w oprogramowaniu). Musi być kompatybilny ze sterownikiem silnika, który zostanie

użyty w celu zmian charakterystyki działania silnika. Musi posiadać łatwe i wygodne w obsłudze, intuicyjne oraz estetyczne GUI.

1.4 Podział prac

Podział prac nad projektem był równomierny. Każdy element był konsultowany między autorami.

Rozdział 2

Omówienie protokołów komunikacyjnych

2.1 Główna magistrala komunikacyjna - CAN

Wybór magistrali CAN spełnia wszystkie założenia projektu zawarte w [Sekcji 1.3: Analiza problemu](#)). Jest to protokół komunikacyjny wykorzystywany przez sterowniki silnika ECU serii PE3[7], który został wybrany przez konstruktorów pojazdu. Jest to powszechnie stosowany standard w systemach automatyki przemysłowej i samochodowej. Charakteryzuje się wysokim bezpieczeństwem (odpornością na błędy) transmisji. Pozwala na przesyłanie mikrostrumieni danych, takich jak uzyskiwane dane z czujników, składających się z 1 do 8 bitów na komunikat.

2.1.1 CAN w modelu ISO/OSI

Controller Area Network to standard przemysłowej sieci transmisyjnej, stworzonej na początku lat osiemdziesiątych przez niemiecką firmę Bosch. Jak każdy powszechnie stosowany protokół komunikacyjny, tak i CAN został w roku 1993 zstandaryzowany i opisany przez *International Standard Organisation (ISO)* na warstwach modelu ISO/OSI i przyjęty za normę ISO-11898 [5] ([Rysunek 2.1](#)). Standard CAN miał obejmować warstwy 1. (fizyczną) 2. (łącza danych) oraz 7. (aplikacji) [2].

W warstwie fizycznej istnieją dwie wersje protokołu:

- *Low Speed CAN* od 5 kb/s do 125 kb/s
- *High Speed CAN* do 1 Mb/s

Wersje te jednak nie opisują bezpośrednio realizacji fizycznej transmisji sygnału. Powstało wiele dokumentów, które uściślają to zagadnienie. Sygnał musi być sygnałem różnicowym, a najczęściej stosowanym medium jest skrętka dwóch przewodów, ekranowanych lub nie. Sygnał różnicowy zapobiega zniekształceniu sygnału przez zakłócenia. Topologia sieci to magistrala, co oznacza, że wszystkie elementy sieci podłączone są do wspólnej pary przewodów.

W nowszej wersji specyfikacji (oznaczanej CAN 2.0), która jest odpowiedzią na rosnące zapotrzebowanie, warstwa łącza danych podzielona jest na dwie części:

- *Logical Link Control (LLC)* odpowiedzialną za retransmisję danych, zarządzanie filtrami identyfikatorów oraz sygnalizację przepełnień skrzynek odbiorczych i nadawczych.
- *Media Access Control (MAC)* odpowiedzialną za dostęp do medium, kodowanie i enkapsulację danych oraz wykrywanie błędów transmisji.

| Model warstwowy ISO/OSI | Warstwa magistrali CAN | | |
|--|---|------------------------|--------------------------------|
| Warstwa 8 Aplikacja: "Urządzenie na magistrali" | CANopen | DeviceNet | Smart Distributed System (SDS) |
| Warstwa 7 "Warstwa aplikacji" | CAL: CAN Warstwa aplikacji dla aplikacji przemysłowych | Specyfikacje DeviceNet | Specyfikacje SDS |
| Warstwy 3..6 | Puste!!! | | |
| Warstwa 2 "Warstwa łączy danych" | LLC: Logical Link Control MAC: Medium Access Control zgodnie z ISO 11898 <u>Rezultat:</u> Specyfikacje CAN 2.0A, CAN 2.0B | | |
| Warstwa 1 "Warstwa fizyczna" | "Low-Speed CAN" ISO 11519-2 | | "High-Speed CAN" ISO 11898 |

Rysunek 2.1: CAN w modelu ISO/OSI [2]

Dostęp do medium realizowany jest poprzez wyróżnienie dwóch stanów magistrali: dominującego i recesywnego. (Wartości napięć przedstawiono na [Rysunku 2.2](#). Standard ISO-11898 może być stosowany również w sieciach o niższych prędkościach, dlatego zaprezentowano go jako uniwersalny).

| Napięcie na magistrali | Stan magistrali | |
|--|---------------------------|------------------------------|
| | recesywny {ustępujący} | dominujący {przeważający} |
| CANH | 2,5V | 3,5V |
| CANL | 2,5V | 1,5V |
| dopuszczalne napięcie różnicowe $U_0 = \text{CANH} - \text{CANL}$ | 0 - 0,5V | 0,9 - 2,0V |

Rysunek 2.2: Poziomy bezwzględne linii magistrali w odniesieniu do (lokalnej) masy zgodnie z ISO-11898 [2]

Jeżeli urządzenia magistrali wymuszają jednocześnie stan recesywny i dominujący, to na linii ustabilizuje się stan dominujący. System dostępu do medium jest potocznie zwany "iloczynem na drucie" (więcej informacji w [Sekcji 2.1.3: Filtry akceptacyjne](#)) Warstwa łączy danych często obsługiwana jest sprzętowo przez kontrolery magistrali CAN, które spotykane są jako integralne części niektórych mikrokontrolerów.

Istnieje bardzo wiele różnych standardów opartych na warstwie aplikacji. Każdy producent opracowuje swój standard. Istnieją:

- CANopen oparty na standardzie grupy CiA (CAN in Automation - standard

DS 301). Bardzo popularny protokół, używany w systemach wbudowanych.

- CAN Areospace - standard wprowadzony przez NASA (National Aeronautic and Space Administration). Używany do systemu kontrolno-nawigacyjnego.
- CAN Kingdom - specyfikacja warstwy aplikacji stworzona przez szwedzka firmę Kvaser AB. Daje on projektantom swobodę w tworzeniu własnego systemu, otwierając możliwość do projektowania systemu modułowego.
- Device Net - szeroko stosowany w aplikacjach automatyki przemysłowej.
- SDS - (Smart Distributed System) - specyfikacja stworzona przez firmę Honeywell, zajmującą się systemami sterującymi oraz kontrolno-pomiarowymi.
- SafetyBus - standard opracowany przez grupę Safety Network International e.V. Stosowany w przemyśle transportowym, i automatyce przemysłowej.
- SAE - standard zdefiniowany przez grupę Society of Automotive Engineers. Stosowany jest jako system komunikacji urządzeń kontrolnych, pomiarowych w samochodach osobowych (J1850) i ciężarowych (J1939) [4].

Oraz wiele innych, będących wariacjami powyższych.

2.1.2 Budowa ramki CAN

Wyróżnia się podział standardu CAN na dwie kolejne grupy, wewnątrz warstwy łączy danych:

- CAN 2.0 A - podstawową
- CAN 2.0 B - rozszerzoną

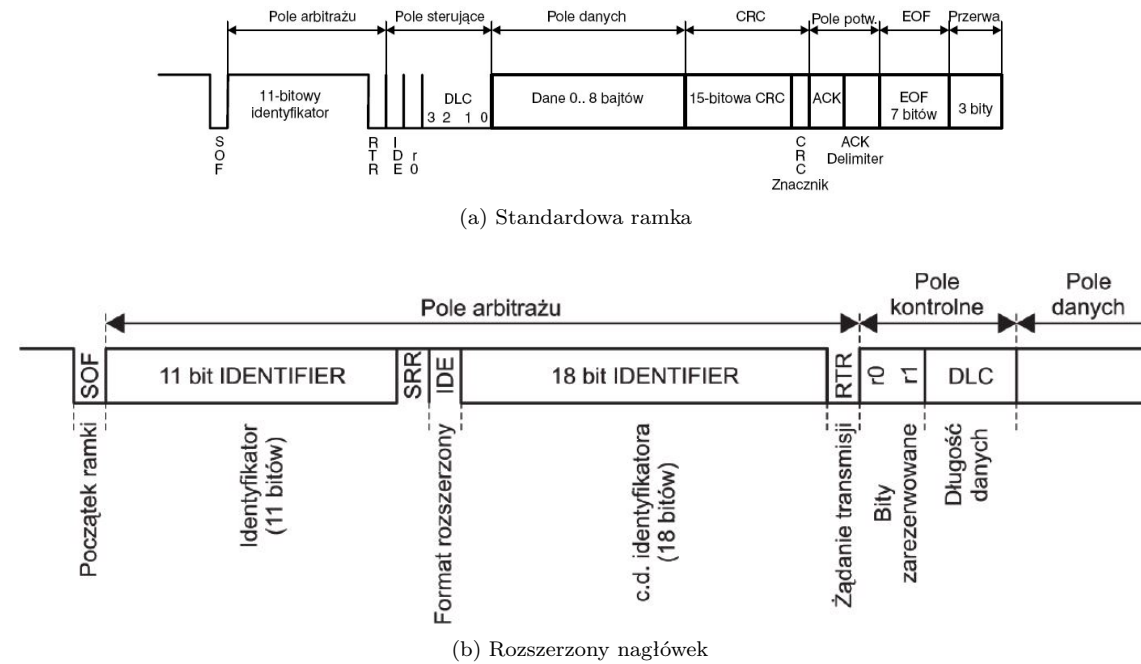
Podział ten ogranicza się do budowy ramki, a przede wszystkim do długości pola arbitrażu wiadomości. Podstawowa wersja ramki posiada 11-bitowy identyfikator ([Rysunek 2.3a](#)), natomiast rozszerzona 29-bitowy ([Rysunek 2.3b](#)). Dobrze zaprojektowany system może skutecznie łączyć w sobie obie wersje protokołu.

Po polu arbitrażu następuje pole kontrolne, w którym zapisana jest informacja o ilości przesyłanych danych. Kod DLC (Data Length Code) to nic innego, tylko zapis binarny liczby bajtów przesłanych w polu danych. Maksymalna liczba to 8, czyli zakres wartości DLC wynosi 0b0000, 0b1000j [14].

Pole danych jest opcjonalne, gdyż istnieją ramki, które są go pozbawione, jak ramka żądania transmisji, czy ramka przepełnienia.

2.1.3 Filtry akceptacyjne

Ogromną zaletą systemu opartego na protokole CAN jest obecność filtrów wiadomości. W sieci CAN identyfikator wiadomości jest jednocześnie jej priorytetem. Im niższy identyfikator, tym wyższy priorytet. Wynika to z faktu, że za stan logiczny 0 odpowiada bit dominujący na magistrali. Dlatego nagłówek ramki zawierający identyfikator nazywany jest polem arbitrażu. Węzeł, który chce wysłać wiadomość o najmniejszym identyfikatorze, uzyska dostęp do magistrali jako pierwszy. Wszystkie węzły monitorują sieć, również w trakcie nadawania. Gdy wykryją, że wiadomość którą nadają, nie pokrywa się z tą na magistrali, przestają nadawać i oczekują na koniec ramki. Wtedy ponawiają próbę nadania wiadomości [2][5].



Rysunek 2.3: Ramka CAN [5]

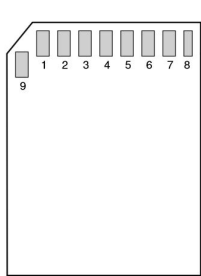
Ważnym spostrzeżeniem oraz znaczącą różnicą między protokołem CAN, a innymi protokołami, jest fakt iż protokół nie posiada adresów. Identyfikator jest powiązany z wiadomością, a nie z urządzeniem. Jako, że każde urządzenie odczytuje stan na magistrali, wysyłanie wiadomości odbywa się w trybie broadcast. W wersji podstawowej, dzięki 11-bitowemu identyfikatorowi istnieje 2048 różnych ramek, w rozszerzonej ponad 500 milionów. Nie ma potrzeby aby wszystkie ramki były przetwarzane przez wszystkie węzły magistrali [2]. Kontrolery CAN umożliwiają filtrację ramek na poziomie sprzętowym, bez potrzeby angażowania jednostki centralnej. Istnieją dwa podstawowe sposoby filtracji wiadomości:

- Tryb maskowania. Definiujemy w nim maskę, która określa, które bity identyfikatora będą porównywane z wzorcowym identyfikatorem. Dzięki temu trybowi, możemy w łatwy sposób zadeklarować zbiór interesujących nas identyfikatorów.
- Tryb listy identyfikatorów. Tworzymy listę identyfikatorów, które będą akceptowane przez węzeł. Jest to wygodne rozwiązanie w przypadku małej ilości pożądanych wiadomości.

2.2 Archiwizacja danych - SD

Mimo iż dane są na bieżąco wysyłane do zdalnego interfejsu użytkownika, trzeba wziąć pod uwagę awaryjność takiego przesyłu oraz możliwość gubienia pakietów danych przy dużych odległościach. Potrzebny jest stabilny i szybki system zapisu zebranych danych, który będzie niezależny od bezprzewodowej komunikacji. Wybrano zapis danych na kartę SD podłączoną bezpośrednio do głównego komputera pokładowego. Standard kart SD jest standardem opracowanym przez trzech producentów: Toshiba, SanDisk i MEI [9], który wyewoluował ze starszego standardu MultiMedia-Card (MMC). Zarówno budowa samej karty, połączenia elektryczne jak i protokół są częścią specyfikacji SD Card (SDC), podzielonej na wiele mniejszych dokumentów [10][11]. SDC oferuje zaawansowany interfejs 9 linii elektrycznych (zegarowej,

komend, 4 linie danych i 3 linie zasilania), który może pracować z maksymalną częstotliwością 50 MHz [10].(Rysunek 2.4).



| Pin | Name | Function (SD Mode) | Function (SPI Mode) |
|-----|----------|--------------------|-------------------------------|
| 1 | DAT3/CS | Data Line 3 | Chip Select/Slave Select (SS) |
| 2 | CMD/DI | Command Line | Master Out Slave In (MOSI) |
| 3 | VSS1 | Ground | Ground |
| 4 | VDD | Supply Voltage | Supply Voltage |
| 5 | CLK | Clock | Clock (SCK) |
| 6 | VSS2 | Ground | Ground |
| 7 | DAT0/DO | Data Line 0 | Master In Slave Out (MISO) |
| 8 | DAT1/IRQ | Data Line 1 | Unused or IRQ |
| 9 | DAT2/NC | Data Line 2 | Unused |

Rysunek 2.4: Diagram Karty SD [9]

Z Rysunku 2.4 wynika, że karty SD wspierają dwa fizyczne protokoły komunikacyjne: SD Bus (Sekcja 2.2.2: SD Bus) oraz SPI (Sekcja 2.2.3: Serial Peripheral Interface (SPI)).

Protokół komunikacyjny kart SD opiera się na prostym systemie komend i odpowiedzi. Wszystkie komendy są inicjowane przez mastera. Karta SD odpowiada na zapytanie ramką odpowiedzi, po której może nastąpić przesył danych, jeżeli taka była komenda, lub zgłoszenie błędu. Cały protokół służy do obsługi systemu plików zawartego na karcie.

2.2.1 FatFs

Z perspektywy systemu plików każdy nośnik danych podzielony jest na klastry i sektory. Sektory są zazwyczaj długości 512 bajtów, natomiast klastry przyjmują różne wartości, w zależności od pojemności dysku i rodzaju systemu plików. Pliki zapisywane są w klastrach, zajmując je całkowicie. Oznacza to, że gdy plik jest mniejszy od pojedynczego klastra, cały klaster zostanie przypisany do tego pliku. System plików FAT opiera się na tablicy alokacji plików FAT (File Allocation Table). Jest to tablica, która stanowi katalog plików znajdujących się na danej partycji/dysku [6].

FatFs to biblioteka implementująca system plików FAT dla systemów wbudowanych. Jest to pomost łączący warstwę sprzętową z warstwą aplikacji. Niezależnie od platformy sprzętowej, po zdefiniowaniu podstawowych funkcji, system zadziała na wybranej platformie sprzętowej. Minimalna aplikacja zakłada, że użytkownik napisze funkcje odpowiedzialne za wysłanie i odbiór wiadomości oraz inicjalizację karty. Dokładny opis przewidywanego działania tych funkcji dostępny jest na głównej stronie, z której pobrano bibliotekę [1]. Dodatkowo na stronie podane są źródła, z których można pobrać biblioteki oparte na FatFs implementujące ją na wybranych platformach sprzętowych. Jedną z takich bibliotek, autorstwa Tilen'a Majerle [16], użyto w projekcie.

2.2.2 SD Bus

Protokół SD Bus dzieli się na dwie wersje. Wyróżnia się wersję 1-bitową oraz 4-bitową.

SD Bus w wersji 1-bitowej to synchroniczny, szeregowy protokół z jedną linią komend, jedną danych i jedną zegarową.

SD Bus w wersji 4-bitowej różni się od niego tylko szerokością linii danych, których

jest 4. Przy dobrej implementacji może być czterokrotnie szybszy niż jego uboższa wersja.

Protokół SD Bus wymaga obliczania sumy CRC, która zapobiega błędom transmisji. W przypadku wersji 4-bitowej, CRC liczone jest dla każdej linii danych z osobna. SD Bus jest domyślnym protokołem do obsługi kart SD, aby przełączyć kartę w tryb SPI należy podczas inicjalizacji użyć specjalnej komendy i przekazać odpowiedni dla niej kod CRC [10].

2.2.3 Serial Peripheral Interface (SPI)

Interfejs SPI służy do dwukierunkowej (full duplex) , synchronicznej, szeregowej komunikacji i składa się z trzech linii:

- MISO - Master Input Slave Output, jednokierunkowa linia danych służąca do odbierania danych przez mastera.
- MOSI - Master Output Slave Input, jednokierunkowa linia danych służąca do wysyłania danych przez mastera.
- SCK - linia zegarowa służąca synchronizacji komunikacji [6].

Do aktywacji wybranego układu peryferyjnego służy dodatkowo linia SS (Slave Select - wybór układu podrzędnego).

Jako że podstawą komunikacji z kartami SD jest wymiana komend i danych, a SPI nie dysponuje linią komend, wszystkie komendy i dane są szeregowo wysyłane po linii MOSI i odbierane na linii MISO. Tryb SPI wspiera większość komend używanych w komunikacji z kartami SD. Implementacja tego protokołu jest dużo łatwiejsza niż specyficznego SD Bus, dlatego jest to popularniejsze rozwiązanie i zdecydowanie lepiej udokumentowane. Większość dzisiejszych mikrokontrolerów posiada konfigurowalne peryferium SPI. W przypadku jego braku, można łatwo zaimplementować komunikację na zwykłych wyjściach cyfrowych [10].

2.2.4 Direct Memory Acces (DMA)

Bardzo wiele operacji wykonywanych na blokach danych polega tylko na ich kopiowaniu. Nie ma potrzeby angażować do tego procesu rejestrów CPU (jednostki sterującej). Na potrzeby kopiowania danych, bez użycia procesora stworzono blok DMA (Direct Memory Acces). Jeżeli rozpatrywać peryferia jako zmapowaną pamięć, można używać DMA do kopiowania danych z peryferiów do bloków pamięci wewnętrznej lub odwrotnie. Obsługa karty SD może odbywać się przy użyciu modułu DMA, dzięki czemu można wskazać kontrolerowi DMA blok pamięci, który ma zostać skopiowany do karty, a zapis odbędzie się bez użycia procesora.

2.3 Komunikacja bezprzewodowa - XBee

2.3.1 Universal Asynchronous Receiver and Transmitter - UART

Rozdział 3

Realizacja projektu

3.1 Model systemu

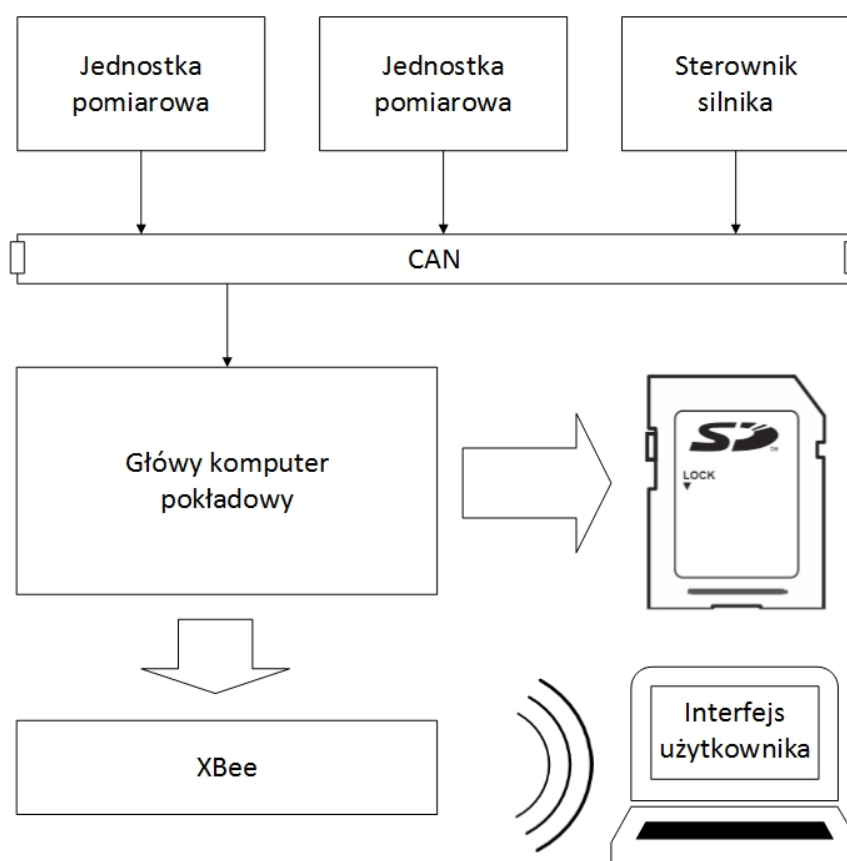
Zdecydowano się na stworzenie systemu wbudowanego, który spełni wymagania przedstawione w ([Sekcji 1.3: Analiza problemu](#)). W celu zbierania danych potrzebny jest osobny układ, który będzie odpowiedzialny za konkretną grupę czujników, dalej zwany jednostką pomiarową (HUB). Pozwala to na większą elastyczność podczas doboru czujników oraz protokołów komunikacyjnych. Niweluje to również problem przesyłu sygnału z czujników na duże odległości do jednego centralnego urządzenia, zmniejszając również ilość przewodów w pojeździe. Centralne urządzenie jest dalej zwane głównym komputerem pokładowym. Komputer pokładowy ma za zadanie zebranie danych, wyświetlanie ich oraz archiwizację. Komunikacja między jednostkami pomiarowymi, a głównym komputerem pokładowym musi być odporna na zakłócenia, gdyż odległości między tymi urządzeniami mogą być znaczne, a dowolność ułożenia przewodów ograniczona przez konstrukcję pojazdu. Komunikacja musi zapewnić wszystkim jednostkom pomiarowym czas na wysłanie wiadomości, a komputerowi pokładowemu na archiwizację i przesył do interfejsu użytkownika. Komunikacja musi być szybka i niezawodna. Wybrano do tego celu platformę sprzętową z wbudowanym kontrolerem magistrali Controller Area Network (CAN), który autonomicznie wykonuje część operacji, odciażając jednostkę centralną. W celu archiwizacji danych użyto karty SD, która zapewnia możliwość obsługi zarówno przez wbudowany system, jak i przez komputer osobisty. Umożliwia również duże prędkości zapisu danych, a dzięki wbudowanemu kontrolerowi dostępu do pamięci (DMA), odciaża jednostkę centralną. Do komunikacji bezprzewodowej wybrano moduł radiowy XBee, który umożliwia komunikację na znaczące odległości, co gra znaczącą rolę podczas przejazdów pojazdu na długich trasach. Jest obsługiwany przez zintegrowany układ UART.

3.2 Główny komputer pokładowy

Komputer pokładowy składa się z dwóch elementów: płytki ewaluacyjnej STM32-F4Discovery oraz nakładki rozszerzającej jej możliwości. Głównym podzespołem komputera pokładowego jest mikrokontroler serii STM32F4.

3.2.1 Mikrokontroler

Nazwa serii reprezentuje kolejno nazwę producenta - STMicroelectronics, wielkość pojedynczego rejestru - 32 bity oraz wersję rdzenia - CortexTM-M4 CPU firmy ARM [12][13]. Jest to podrodzina rdzeni zoptymalizowana pod kątem minimalizacji



Rysunek 3.1: Model systemu pomiarowego

ceny przy zachowaniu dużej wydajności, przeznaczona do zastosowań konsumenc-
kich i przemysłowych [6]. Powodem wyboru tej platformy sprzętowej było spełnienie
przez nią wszystkich założeń projektu odnośnie jednostki sterującej oraz posiada-
nych układów peryferyjnych [15]. Układ musiał

- być szybki - 168 MHz,
- być niezawodny - dwa timery typu watchdog,
- posiadać rozszerzenie SDIO do obsługi kart SD (Seksja ?? : Secure Digital Card Input Output (SDIO)),
- posiadać magistralę CAN do komunikacji z układami pomocniczymi (Seksja 2.1: Główna magistrala komunikacyjna - CAN),
- posiadać układ UART do komunikacji z nadajnikiem XBee (Seksja 2.3.1: Uni-
versal Asynchronous Receiver and Transmitter - UART)
- być łatwy w programowaniu, co umożliwiła rozbudowana biblioteka dostar-
czana przez producenta,

Zastosowanie nakładki rozszerzającej funkcjonalność Discovery pozwoliło znacznie
ułatwić proces projektowania skracając go znacznie.

3.2.2 Obsługa magistrali CAN

Transceiver CAN

3.2.3 Obsługa karty SD

3.2.4 Obsługa modułu XBee

3.2.5 Zasilanie

3.2.6 System przerwań (NVIC)

3.3 Rozproszone jednostki pomiarowe - HUB

3.3.1 Wprowadzenie

Główne zadanie Rozproszonej Jednostki Pomiarowej(Hub) to dokonywanie pomiaru wielkości fizycznych mierzonych przez czujniki rozmieszczone w bolidzie. Za standard analogowego sygnału wejściowego przyjęto 0-12V.

Układ Jednostki Pomiarowej składa się z 3 odseparowanych galwanicznie części: pomiarowej, mikrokontrolerowej i transmisji CAN.

Układ realizujący działanie Jednostki to STM32f103, zapewniając peryferia pomiarowe jak i komunikacyjne.

3.3.2 Układ główny

Mikrokontroler STM32f103 pochodzi z rodziny układów o rdzeniu CortexTM-M3, dostępny od paru lat na rynku, sprawdza się w rozwiązaniach wymagających małego i prostego kontrolera. Szereg peryferiów w które wyposażony jest ten układ stawia go w kategorii uniwersalności nie osiągalnej przez inne układy na rynku tej klasy. Najważniejsze peryferia wykorzystane w Hubie to:

- Dwa 12 bitowe przetworniki analogowo-cyfrowe, potrafiące obsłużyć do 16 kanałów
- Interfejs komunikacyjny CAN 2.0B

Układ dostarcza także 7 timerów sprzętowych, które mogą pracować w wielu zaawansowanych trybach. Do wykonywania pomiarów z określonym próbkowaniem wystarczą podstawowe tryby działania dostarczonych timerów.

3.3.3 Separacja sygnałów analogowych

3.3.4 Zasilanie

3.4 Zdalny interfejs użytkownika

3.4.1 Wymagania sprzętowe

3.4.2 Moduł odbiorczy XBee

3.4.3 Grphical User Interface (GUI)

Rozdział 4

Badania eksperymentalne

Rozdział 5

Podsumowanie

Spis tablic

Spis rysunków

| | | |
|-----|--|----|
| 2.1 | CAN w modelu ISO/OSI [2] | 4 |
| 2.2 | Poziomy bezwzględne linii magistrali w odniesieniu do (lokalnej) masy zgodnie z ISO-11898 [2] | 4 |
| 2.3 | Ramka CAN [5] | 6 |
| 2.4 | Diagram Karty SD [9] | 7 |
| 3.1 | Model systemu pomiarowego | 10 |

Literatura

- [1] ChaN. FatFs - FAT file system module R0.10c . http://elm-chan.org/fsw/ff/00index_e.html, 2014.
- [2] Elektor Electronics. Magistrala CAN. *ELEKTRONIKA PRAKTYCZNA*, 2000.
- [3] Formula Student. <http://events.imeche.org/formula-student/about-us>.
- [4] Magistrala CAN - Warstwa aplikacyjna. <http://canbus.pl/index.php?id=5>.
- [5] Paweł Moll. Sieci CAN. *ELEKTRONIKA PRAKTYCZNA*, 2005.
- [6] Krzysztof Paprocki. *Mikrokontrolery STM32 w praktyce*. Wydawnictwo BTC, Legionowo, 2011.
- [7] Performance ELEKTRONICS. *AN400 Rev B- Application Note CAN Bus Protocol for PE3 Series ECUs*, November 2011.
- [8] PUT Motorsport. <http://motorsport.put.poznan.pl>.
- [9] SanDisk. *Secure Digital Card Product Manual - Revision 1.7*, September 2003.
- [10] SD Card Association. *SD Specifications Part 1: Physical Layer Simplified Specification*. SD Group, wydanie 2.00, September 2006.
- [11] SD Card Association. *SD Specifications Part E1 - SDIO Simplified Specification*, wydanie 2.00, February 2007.
- [12] Tomasz Starak. Tanio, taniej, STM32F4Discovery... Cortex-M4 w pełnej okazałości. *ELEKTRONIKA PRAKTYCZNA*, Listopad 2011.
- [13] STMicroelectronics. *UM1472 User Manual*, January 2012.
- [14] STMicroelectronics. *RM0090 Reference manual*, February 2013.
- [15] STMicroelectronics. *STM32F405xx STM32F407xx Datasheet*, June 2013.
- [16] Majerle Tilen. Library 21 - Read SD card with FatFs on STM32F4. <http://stm32f4-discovery.com/2014/07/library-21-read-sd-card-fatfs-stm32f4xx-devices/>, July 2014.



© 2015 Marcin Aftowicz, Jakub Baranowski

Instytut Automatyki i Inżynierii Informatycznej, Wydział Elektryczny
Politechnika Poznańska

Skład przy użyciu systemu L^AT_EX.

Bib_TE_X:

```
@mastersthesis{ key,  
  author = "Marcin Aftowicz \and Jakub Baranowski",  
  title = "Układ pomiarowy do Bolidu klasy Formuła Student (projekt zespołowy)",  
  school = "Poznan University of Technology",  
  address = "Poznań, Poland",  
  year = "2015",  
}
```