

Spis treści

1.Wstęp.....	3
1.1 Systemy informatyczne.....	4
1.2 Infrastruktura systemów informatycznych.....	5
1.2.1 Klasyfikacja sieci.....	6
1.2.2 Topologie.....	6
1.2.3 Media transmisyjne.....	9
1.3 Protokoły komunikacyjne.....	10
2.Cel i założenia pracy.....	18
3.Analiza istniejących rozwiązań.....	20
3.1 Płatność przez telefon.....	20
3.2 Automaty samoobsługowe.....	23
3.2.1 Parkometry.....	23
3.2.2 Parkomaty.....	24
3.2.3 Automaty wjazdowe.....	26
3.3 Rezerwacja internetowa.....	27
3.4 Zestawienie rozwiązań.....	28
4.Projekt koncepcyjny systemu.....	31
4.1 Funkcja główna.....	31
4.2 Środowisko pracy.....	31
4.3 Eksploatacja.....	31
4.4 Struktura systemu.....	31
4.5 Schemat blokowy funkcjonalny.....	32
4.6 Koncepcja działania systemu.....	34
5.Integracja E2LP w sieci LAN.....	35
5.1 Płyta rozszerzająca FMC NXP ARM.....	35
5.2 Mikrokontroler LPC2368 z procesorem ARM7TDMI.....	35
5.3 Połączenie mikrokontrolera LPC z chipem karty sieciowej.....	36
5.3.1 Media Independent Interface.....	37
5.3.2 Reduced Media Independent Interface.....	38
5.3.3 Konwerter RMII-MII.....	38
5.3.4 Maszyna stanów MDIO.....	41
5.4 Implementacja komunikacji w standardzie Ethernet na LPC2368.....	41
5.4.1 Programowa konfiguracja rejestrów urządzenia EMAC.....	42
5.4.2 Pętla główna programu LPC2368.....	43
5.5 Podsumowanie.....	44
6.Internetowy system rezerwacji.....	45
6.1 Wykorzystane oprogramowanie.....	45
6.2 Projekt bazy danych.....	45

6.3 Projekt witryny internetowej.....	48
6.3.1 Schemat przejść pomiędzy stronami.....	48
6.3.2 Szablon pojedynczej strony.....	49
6.4 Implementacja witryny.....	50
6.4.1 Logowanie.....	51
6.4.2 Rejestracja.....	54
6.4.3 Zarządzanie pojazdami.....	54
6.4.4 Pozostałe strony.....	66
6.5 Podsumowanie.....	67
7.Program pośredniczący w komunikacji między mikrokontrolerem a bazą danych.....	68
7.1 Koncepcja aplikacji.....	68
7.2 Wykonanie.....	68
7.3 Protokół komunikacji z mikrokontrolerem LPC.....	72
7.4 Weryfikacja kodu wjazdowego z bazą danych.....	72
7.5 Odnotowanie wjazdu i wyjazdu.....	75
7.6 Podsumowanie.....	75
8.Program mikrokontrolera PicoBlaze.....	76
8.1 Emulacja terminalu do wpisywania kodów.....	76
8.2 Algorytm działania.....	77
8.3 Wyświetlanie wiadomości użytkownikowi.....	79
9.Integracja szlabanu automatycznego 615 BPR z systemem.....	81
10.Model systemu oraz testy.....	84
11.Podsumowanie oraz wnioski.....	85
Spis ilustracji.....	86
Spis tabel.....	87

1. Wstęp

Niniejsza praca dyplomowa jest wynikiem prowadzonej przez autora w trakcie studiów magisterskich pracy przejściowej [], która jak nazwa wskazuje dotyczyła analizy różnych sposobów komunikacji z wykorzystaniem układu logiki programowalnej FPGA Spratan6 znajdującego się na platformie edukacyjnej E2LP. Praca przejściowa ograniczała się jednak głównie do komunikacji z urządzeniami peryferyjnymi takimi jak pamięć flash oraz wyświetlacz LCD znajdującymi się na platformie E2LP. Sposobami komunikacji pozwalającymi na wymianę danych z urządzeniami znajdującymi się poza nią były port podczerwieni oraz UART. Rozwiązania te mimo swojej mającej wiele zalet prostoty mają także dość duże ograniczenia, którymi są np. podatność na zakłócenia jak w przypadku podczerwieni jak również niewielki zasięg transmisji danych oraz możliwość połączenia tylko z jednym urządzeniem poprzez port szeregowy w przypadku UART. Dużą chęcią autora było wykorzystanie innego cennego urządzenia znajdującego się na platformie E2LP jakim jest karta sieciowa umożliwiająca komunikację w sieci LAN jak również w internecie. Dążenia te były jak najbardziej zgodne z zagadnieniami jakie poruszane są na specjalności informatyka przemysłowa w ramach której studiował autor.

W dalszej kolejności podjęte zostały próby integracji platformy E2LP z siecią LAN poprzez ową kartę sieciową. Szybko okazało się, że jest to zadanie znacznie bardziej skomplikowane niż testowane w pracy przejściowej sposoby komunikacji. Z pomocą przyszła płyta rozszerzająca platformę E2LP z mikrokontrolerem z procesorem w architekturze ARM, który wyposażony jest w sprzętowy sterownik EMAC. Mając dostęp do przykładowych aplikacji wraz z ich dokumentacją opublikowanych na stronach producenta oprogramowania do programowania i testowania mikrokontrolerów Keil stało się jasne, że komunikacja platformy w sieci opartej o technologię Ethernet jest jak najbardziej możliwa. Zanim udało jednak to zrobić pojawiły rozważania w jaki sposób można by wykorzystać taką funkcjonalność. W ten sposób właśnie narodził się temat tej pracy.

Rozwijając go należy sprecyzować, że zarządzanie ma być w dużym stopniu zautomatyzowane. Zawierać się w nim będą m.in. możliwość rejestracji w systemie, logowanie, dodawanie pojazdów oraz autoryzacja i sterowanie szlabanem wjazdowym. Podkreślić należy, że E2LP jest istotnym elementem owego systemu informatycznego pełniącym w dużej mierze rolę części sprzętowej. Ważnym też jest że komunikuje się ono z pozostałymi komponentami systemu poprzez Ethernet. Inną ściśle sprzętową częścią jest szlaban automatyczny, którego dokumentacja zostanie wykorzystana.

W dalszej części wstępu opisane zostaną systemy informatyczne, a dalej ich infrastruktura. Przystawione zostaną protokoły komunikacyjne w oparciu o które komunikują się ich elementy składowe. Opisany zostanie w tym celu model odniesienia OSI. Będzie to wprowadzenie teoretyczne do implementowanych w dalszych rozdziałach protokołów komunikacyjnych.

W kolejnym rozdziale przeanalizowane zostaną istniejące rozwiązania pełniące podobną funkcję. Analiza ta będzie dotyczyła różnych aspektów tych rozwiązań takich jak koszt zainstalowania, obszar zastosowań, łatwość obsługi. W wyniku tej analizy przedstawiona zostanie ocena każdego z nich, a uzyskane w ten sposób wnioski staną się cennym źródłem informacji do rozpoczęcia własnego projektu systemu.

Projekt ten będzie opisany w następnym rozdziale i opierać on będzie się głównie na założeniach zawartych w karcie pracy oraz własnych autora. Opracowane zostaną schematy ideowy i funkcjonalny. Następnie przedstawione i uzasadnione zostaną wybrane rozwiązania takie jak oprogramowanie.

Powstały w ten sposób projekt stanie się punktem odniesienia do wykonania części

praktycznej jaką będzie głównie programowanie w wielu językach oraz na różnych platformach, ale również integracja sprzętowa z wykorzystaniem platformy E2LP. Opis działań związanych z pisanem kodu tudzież konfiguracją poszczególnych elementów systemu przedstawiony zostanie w kilku kolejnych rozdziałach.

Wynikający z nich rozdział przedstawi model służący prezentacji funkcjonowania systemu. Głównie będą to komponenty softwarowe, ale także takie, które ze względów praktycznych zostaną zasymulowane. Opisane w tym rozdziale zostaną również testy stworzonego modelu. Będą one odzwierciedleniem rzeczywistego działania systemu.

Ostatni rozdział pracy będzie podsumowaniem oraz wnioskami płynącymi z otrzymanych rezultatów.

1.1 Systemy informatyczne

Rozwój nowych technologii z różnych dziedzin takich jak elektronika oraz IT stwarza możliwości udoskonalania wielu dziedzin działalności człowieka. Przykładem mogą być tu właśnie systemy informatyczne, które w dzisiejszych czasach odgrywają coraz większą rolę w funkcjonowaniu nowoczesnych przedsiębiorstw jak również instytucji. W tym miejscu należy zdefiniować czym jest system informatyczny oraz jak jest jego struktura.

Na podstawie dostępnych powszechnie informacji w literaturze oraz m.in. w art. 7 pkt 2a ustawy o ochronie danych osobowych można przedstawić następującą definicję: system informatyczny to zbiór powiązanych ze sobą elementów, których wspólną funkcją jest przetwarzanie danych w sposób cyfrowy mające na celu usprawnienie działania organizacji w jakiej pracuje. Elementy składowe systemu informatycznego to:

- Personel korzystający z systemu – w jego skład wchodzi osoby zarządzające systemem, pełniące w nim funkcje administracyjne oraz zwykli użytkownicy.
- Przetwarzane informacje – są to dane zebrane w postaci elektronicznej zebrane we wspólnej bazie lub rozproszone, są to często dane poufne zawierające informacje o osobach, ale także bazy wiedzy opisujące obiekty w postaci zbioru atrybutów.
- Infrastruktura – jest to sprzęt wykorzystany do przechowywania, zabezpieczenia oraz udostępniania danych, a także technologie informacyjne pozwalające na komunikację z systemem. Głównym elementem infrastruktury są komputery zarówno osobiste jak i serwery o dużej mocy obliczeniowej jak również sieci komputerowe, które pozwalają na integrację ich ze sobą.
- Oprogramowanie – jest to część systemu informatycznego odpowiadająca za formę przetwarzania danych, ale również sposób komunikacji z użytkownikami tworząc tak zwane interfejsy. Od niego często zależy bezpieczeństwo danych a także ich modyfikacja i aktualizacja. Ważną funkcją oprogramowania jest także uwierzytelnianie użytkowników korzystających z systemu.
- Organizacja – jest to zbiór procedur, przepisów w oparciu, o które dany system funkcjonuje. Wyliczyć tu można: zarządzenie, rozporządzenia, wytyczne, regulacje, przepisy prawne, strategię rozwoju, przewidywane scenariusze wykorzystania systemu, działania w wypadku sytuacji awaryjnych oraz działania ochronne.

Systemy informatyczne wspomagające funkcjonowanie organizacji podzielić można na różne kategorie (na podstawie <http://platinumit.pl/systemy-informatyczne/rodzaje-systemow/bmp-zarzadzanie-procesami-biznesowymi> i Wikipedia). Scharakteryzowanych zostanie tutaj kilka z

nich:

- ERP – Planowanie zasobów przedsiębiorstwa (ang. *Enterprise Resource Planning*). Pozwalają na współdzielenie, synchronizację i wykonywanie operacji na danych w całej organizacji. Zapewnia to efektywne planowanie wykorzystania zasobów przedsiębiorstwa. Dają możliwość zarówno gromadzenia informacji jak i przetwarzania - wykonywanie skomplikowanych operacji, generowanie dokumentów i wykresów, zarządzanie kodem źródłowym, procesami i kwerendami za pomocą interfejsów w postaci specjalistycznych programów.
- BPM – Zarządzanie procesami biznesowymi (ang. *Business Process Management*). Opiera się głównie na zwiększeniu efektywności procesów biznesowych w przedsiębiorstwie poprzez jego optymalizację. Głównymi celami stosowania BPM są: uwolnienie i odpowiednie kierowanie przepływem informacji oraz nadzór nad poszczególnymi działaniami, mapowanie, wizualizacja i analiza procesów, odzwierciedlenie przebiegu procesów w elektronicznych formularzach zapisu i kontroli danych, budowa i nadawanie obiegu elektronicznych formularzy, spójnie z przebiegiem procesu, projektowanie i nadzorowanie ścieżki zadań (ang. *workflow*), kontrola poszczególnych etapów realizacji działań.

Tego typu systemy są często elementem składowym systemów ERP.

- CRM – Zarządzanie relacjami z klientem (ang. *Consumer Relationships Management*). Wspomaga procesy zachodzące między organizacją a klientem. Automatyzuje wymianę danych a nim. Jego głównym zadaniem jest określenie potrzeb klienta i zaproponowanie mu najlepszej oferty a w dalszej perspektywie utrzymanie określonych relacji z nim. Systemy te w dużej mierze wykorzystują internet. Poprzez witryny internetowe pozwalają użytkownikom przeglądać oferty i je wybierać. Z tego rodzaju systemów korzystają m.in. banki, sklepy internetowe, portale usługowe jak np. hotele z możliwością rejestracji, rezerwacje biletów.
- MRP – Planowanie zapotrzebowania materiałowego (ang. *Material Requirements Planning*). Na podstawie analizy danych, zebranych we wskazanym okresie czasu, pomagają określić wysokość zapasów magazynowych. Dzięki temu pozwalają uniknąć problemu zalegającego towaru, uwalniając tym samym dodatkowy kapitał. Głównymi celami stosowania tych systemów są: zmniejszenie kosztów zarządzania, redukcja zapasów magazynowych, precyzyjne określenie czasów dostaw produktów, optymalizacja infrastruktury magazynowej, natychmiastowe reagowanie na potrzeby klientów.

Wymienić tutaj także można systemy zarządzania łańcuchem dostaw, ruchem produktów w magazynach, realizacji produkcji, prawami dostępu do informacji. Innym określeniem cechującym systemy informatyczne jest Zintegrowany System Informatyczny. Jest to modułowo zorganizowany system informatyczny obsługujący wszystkie obszary działalności przedsiębiorstwa. Jego zadaniem jest optymalizacja przepływu informacji oraz kontrolowania i zarządzania działalnością organizacji poprzez udostępnienie współdzielenia danych oraz narzędzia do ich prezentacji. Do systemów takich zaliczyć można wcześniejsze ERP.

1.2 Infrastruktura systemów informatycznych

Ważną częścią systemów informatycznych z punktu widzenia pisanej pracy dyplomowej jest ich infrastruktura. W podrozdziale tym opisane zostaną ważne pojęcia związane z ową infrastrukturą.

Siecią komputerową nazywa się grupę komputerów lub innych urządzeń np..

mikrokontrolerów lub telefonów komórkowych połączonych ze sobą za pomocą dowolnego medium transmisyjnego w celu wymiany danych lub współdzielenia zasobów, którym może być urządzenia peryferyjne i sieciowe (skaner, drukarka), oprogramowania jak np. bazy danych, przesyłania danych (jak poczta elektroniczna, pliki itp.).

1.2.1 Klasyfikacja sieci

Sieci komputerowe klasyfikować można ze względu na sposób dostępu do zasobów na dwa rodzaje:

- *Klient-serwer* — sieć, w której znajduje się jeden centralny serwer udostępniający dane. Przykładem może być tu system oparty o bazę danych, ale także LDAP oraz oparte na nim Active Directory służące do autoryzacji dostępu użytkowników np. do komputera. Najbardziej powszechnym zastosowaniem tego rodzaju dostępu do zasobów to serwery Http.
- *Peer-to-peer* (sieć równoprawna) — sieć, w której wszystkie urządzenia są równoprawne. Każdy z komputerów pełni zarówno rolę klienta (pobierając dane z innego urządzenia), jak i serwera (dla innych urządzeń korzystających z udostępnionych zasobów).

Sieci komputerowe klasyfikować można także ze względu na obszar działania:

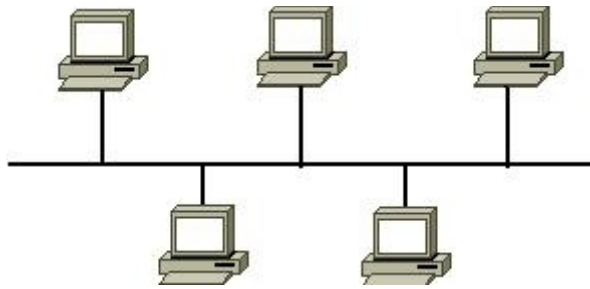
- *LAN (ang. Local Area Network)* — sieć lokalna działająca na niewielkim, ograniczonym obszarze. Przykładem sieci lokalnej LAN jest sieć obejmująca swoim zasięgiem budynek szkoły. Najczęściej spotyka się sieci zbudowane z wykorzystaniem technologii *Ethernet*.
- *MAN (ang. Metropolitan Area Network)* — sieć działająca na większym obszarze, np. miasta — LODMAN.
- *WAN (ang. Wide Area Network)* — sieć rozległa działająca na dużym obszarze, Sieciami rozległymi można nazwać sieci dużych firm łączące oddziały na terenie całego kraju. Mianem sieci rozległej określamy również internet, który swoim zasięgiem obejmuje cały świat. Tego rodzaju sieci korzystają z wielu technologii transmisji na dalekie odległości, takich jak *Frame Relay*, *ATM*, *DSL*.

1.2.2 Topologie

Topologie sieci lokalnych mogą być podzielone na fizyczne oraz logiczne. Topologia fizyczna określa organizację okablowania strukturalnego, topologia logiczna opisuje dostęp do medium fizycznego oraz reguły komunikacji. Nie mniej jednak są one ze sobą powiązane.

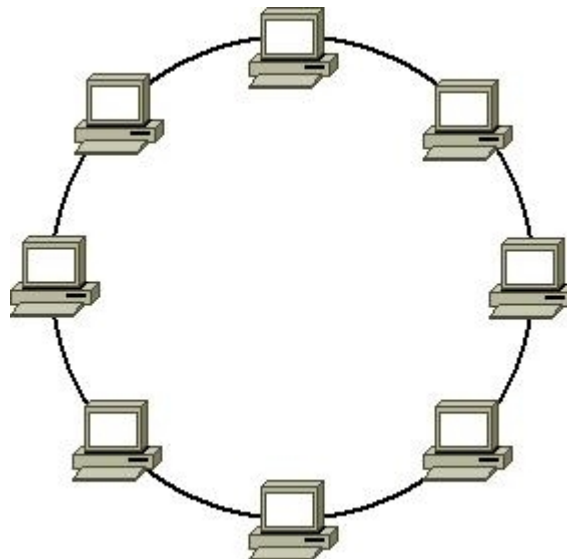
Topologie fizyczne dzieli się na:

- *Magistrala* – charakteryzuje się tym że wszystkie elementy sieciowe podłączone są do pojedynczego, otwartego kabla zwanego szyną lub magistralą, którego zadaniem jest rozprowadzanie sygnału. Wymagane jest aby oba końce magistrali były zakończone opornikami ograniczającymi, zwanymi terminatorami. Rysunek (<http://swiatlan.pl/topologie/topologia-magistrali.html>)



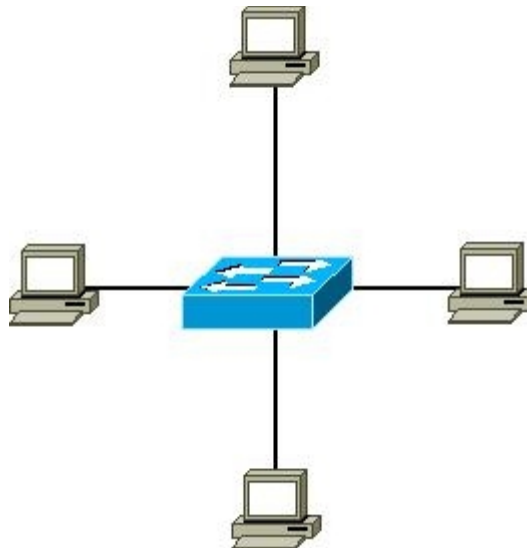
Ilustracja 1.2.1: Topologia magistrali

- Pierścień – komputery w tej topologii połączone są ze sobą za pomocą jednego nośnika informacji w układzie zamkniętym. Każda znajdująca się w sieci stacja robocza posiada dwa połączenia, po jednym ze swoich najbliższych sąsiadów. Połączenie takie tworzone było podobnie jak w przypadku magistrali przy wykorzystaniu kabla koncentrycznego jednak z tą różnicą, że w przypadku pierścienia okablowanie tworzy układ zamknięty, czyli nie ma żadnych zakończeń. Dane przesyłane były wokół pierścienia w jednym kierunku. Każda stacja robocza napotykana na drodze sygnału pełniła rolę wzmacniacza. Poza tym pobierała ona i odpowiadała na pakiety do niej zaadresowane, a także przysyłała dalej pozostałe pakiety do następnej stacji roboczej wchodzącej w skład sieci. Główna wada takiego rozwiązania polegała na tym, że uszkodzenie jednej stacji roboczej najczęściej unieruchamiało całą sieć pierścieniową.



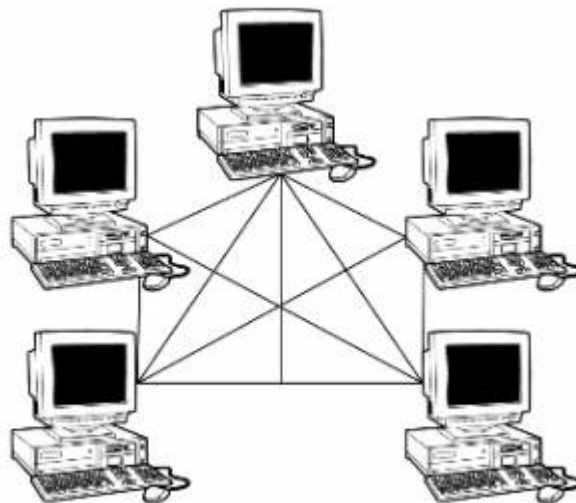
Ilustracja 1.2.2: Topologia pierścienia

- Gwiazda – charakteryzuje się tym iż wszystkie punkty końcowe sieci, zwane terminalami, podłączone są do węzła centralnego którym jest serwer, koncentrator lub przełącznik, natomiast nie są już łączone między sobą. Rozwiązanie takie jest wielką zaletą sieci o topologii gwiazdy, stawiającą ją na wyższym miejscu niż topologie szyny czy pierścienia. Pozwala to zarządzać połączeniem każdego komputera z osobna, a także zapewnia, że sieć będzie działała nawet w przypadku gdy kilka komputerów ulegnie awarii. Wprowadzenie elementu centralnego powoduje także zwiększenie łatwości wykrywania uszkodzeń oraz monitorowania i zarządzania siecią. Podstawową wadą sieci opartych na topologii gwiazdy jest fakt, że uszkodzenie węzła centralnego powoduje przerwanie działania całej sieci.



Ilustracja 1.2.3: Topologia gwiazdy

- Siatka – polega na zapewnieniu wszystkim urządzeniom połączeń ze wszystkimi innymi urządzeniami w sieci. Oznacza to, że każdy host ma własne połączenie z pozostałymi. Jest to rozwiązanie najbardziej odporne na uszkodzenia rozwiązanie. Pozwala także na wykorzystanie maksymalnej prędkości transmisji łącza pomiędzy połączonymi urządzeniami.



Ilustracja 1.2.4: Topologia siatki

Topologie logiczne dzielą się na dwa rodzaje:

- **Topologia rozgłaszania** polega na tym, że host wysyła dane do wszystkich hostów podłączonych do medium. Kolejność korzystania z medium według reguły „kto pierwszy wysze, pierwszy zostanie obsłużony” (ang. *first come, first serve*). Przykładem są tutaj sieci Ethernet, gdzie realizowany jest najczęściej przez protokół CSMA/CD (ang. *Carrier Sense Multiple Access/Collision Detection*), który jest przykładem topologii rozgłaszania. Protokół ten wykrywa, czy łącze jest dostępne, a także reaguje na występujące kolizje. Jeśli węzeł wykryje, że sieć jest zajęta, będzie oczekiwał przez losowo wybrany czas przed

ponowieniem próby. Jeśli węzeł wykryje, że medium nie jest zajęte, rozpocznie nadawanie i nasłuchiwanie. Celem nasłuchiwania jest upewnienie się, że żadna inna stacja nie nadaje w tym samym czasie. Po zakończeniu transmisji danych urządzenie powróci do trybu nasłuchiwania. Jeśli dwa urządzenia rozpoczęły nadawanie w tym samym czasie, występuje kolizja, która jest wykrywana przez urządzenia nadawcze. Transmisja danych zostaje wówczas przerwana. Węzły zatrzymują nadawanie na losowo wybranym czasie, po którym jest podejmowana kolejna próba uzyskania dostępu do medium.

- **Topologia przekazywania tokenu** (żetonu) – polega na kontrolowaniu dostępu do sieci poprzez przekazywanie elektronicznego tokenu. Żeton (ang. *token*) dostępu jest określoną sekwencją bitów zawierającą informację kontrolną. Przejęcie żetonu przez urządzenie sieciowe zezwala na rozpoczęcie transmisji danych. Każda sieć ma tylko jeden żeton dostępu przekazywany między kolejnymi węzłami sieci. Jeśli komputer ma dane do wysłania, usuwa żeton z pierścienia i rozpoczyna transmisję. Dane wędrują po kolejnych węzłach sieci aż trafią do adresata. Komputer odbierający wysyła do komputera nadającego komunikat o odebraniu danych. Po weryfikacji komputer wysyłający tworzy nowy żeton dostępu i wysyła go do sieci. Przykładem może tu być Token Ring.

1.2.3 Media transmisyjne

Medium transmisyjne w sieciach komputerowych to nośnik informacji w postaci sygnałów określonego typu. Parametry transmisji zależą od parametrów użytego medium. Wyróżnia się media przewodowe i bezprzewodowe.

Media przewodowe to:

- **Kabel koncentryczny** – Kabel koncentryczny składa się z miedzianego przewodnika (rdzenia) otoczonego warstwą elastycznej izolacji, która z kolei otoczona jest splecioną miedzianą taśmą lub folią metalową działającą jak drugi przewód oraz ekran dla znajdującego się wewnątrz przewodnika. Ta druga warstwa lub ekran zmniejsza także liczbę zewnętrznych zakłóceń elektromagnetycznych. Podłączenie urządzeń do sieci komputerowej zbudowanej przy użyciu kabla koncentrycznego umożliwia łącza typu BNC. Ten rodzaj okablowania jest wykorzystywany w sieciach budowanych w topologii pierścienia lub magistrali. Obecnie praktycznie nie stosuje się go w sieciach komputerowych. Maksymalna prędkość transmisji dla kabla koncentrycznego wynosi 10 Mb/s, a maksymalna długość sieci to 500 m.
- **Kabel skręcany (skrętka)** składa się z zestawu 4 par żył miedzianych skręconych ze sobą. Skręcenie przewodów pozwala na wyeliminowanie zakłóceń elektromagnetycznych. Jest stosowany w topologii gwiazdy. W skrętce każda żyła oznaczona jest osobnym kolorem: zielonym, pomarańczowym, niebieskim, brązowym oraz biało-zielonym, biało-pomarańczowym, biało-niebieskim, biało-brązowym.

Ze względu na rodzaje stosowanego ekranowania wyróżnia się następujące kable typu skrętka (rysunek 3.4):

- U/UTP (ang. *Unshielded /Unshielded Twisted Pair*) — kabel skręcany nieekranowany. Stanowi najpopularniejszy środek transmisji danych, jest stosowany w pomieszczeniach.
- F/UTP (ang. *Foiled/Unshielded Twisted Pair*) — kabel skręcany ekranowany folią z przewodem uziemiającym. Stosuje się go na korytarzach lub na zewnątrz budynków.
- S/FTP (ang. *Shielded/ Foiled Twisted Pair*) — kabel skręcany z ekranem wykonanym w postaci foliowego opłotu każdej pojedynczej pary i dodatkowo zewnętrznej siatki.

- SF/UTP (ang. *Shielded/Foiled Twisted Pair*) — kabel skręcany z podwójnym zewnętrznym ekranem w postaci foliowego opłotu i siatki.

Kabel typu skrętka podłączany jest do gniazd i końcówek typu RJ45 (złącze 8P8C). Maksymalna długość połączenia za pomocą kabla typu skrętka pomiędzy dwoma urządzeniami wynosi 100 m. Po przekroczeniu tej odległości należy użyć aktywnych urządzeń sieciowych w celu wzmocnienia sygnału.

- Światłowód – Coraz większą popularność jako typ okablowania zdobywa światłowód. Jest to spowodowane przede wszystkim jego dużą przepustowością, odpornością na zakłócenia, możliwością transmisji na dalekie odległości (dla standardu 100-BaseFX do 2000 m). Najważniejszym elementem systemu światłowodowej transmisji danych jest światło, które może być emitowane przez diody LED lub diody laserowe. Światłowód składa się z płaszcza zewnętrznego (ochronnego), zbioru włókien, natomiast każde włókno to zbiór trzech elementów: bufora, płaszcza i rdzenia. Promień świetlny, który trafia do rdzenia pod odpowiednim kątem, jest określany jako mod światłowodowy. Ze względu na liczbę równocześnie transmitowanych modów rozróżnia się światłowody:
 - jednomodowe — transmitujące jeden mod (promień) światła,
 - wielomodowe — transmitujące wiele modów (promieni) światła.

W mediach bezprzewodowych medium transmisyjnym jest fala elektromagnetyczna. W praktycznej realizacji może to być podczerwień, gdzie źródłem promieniowania jest dioda LED lub laserowa. są stosowane na otwartym terenie bądź wewnątrz budynków. Bardziej popularne są jednak fale radiowe. do transmisji wymagają planowania przydziału częstotliwości z uwzględnieniem maksymalnej dopuszczalnej mocy nadajników, rodzaju modulacji oraz innych zaleceń Międzynarodowej Unii Telekomunikacji (ITU). Sieci WLAN mogą pracować w dwóch trybach: ad-hoc, w którym urządzenia łączą się bezpośrednio ze sobą, lub w trybie infrastruktury z wykorzystaniem punktów dostępowych (ang. *access point*).

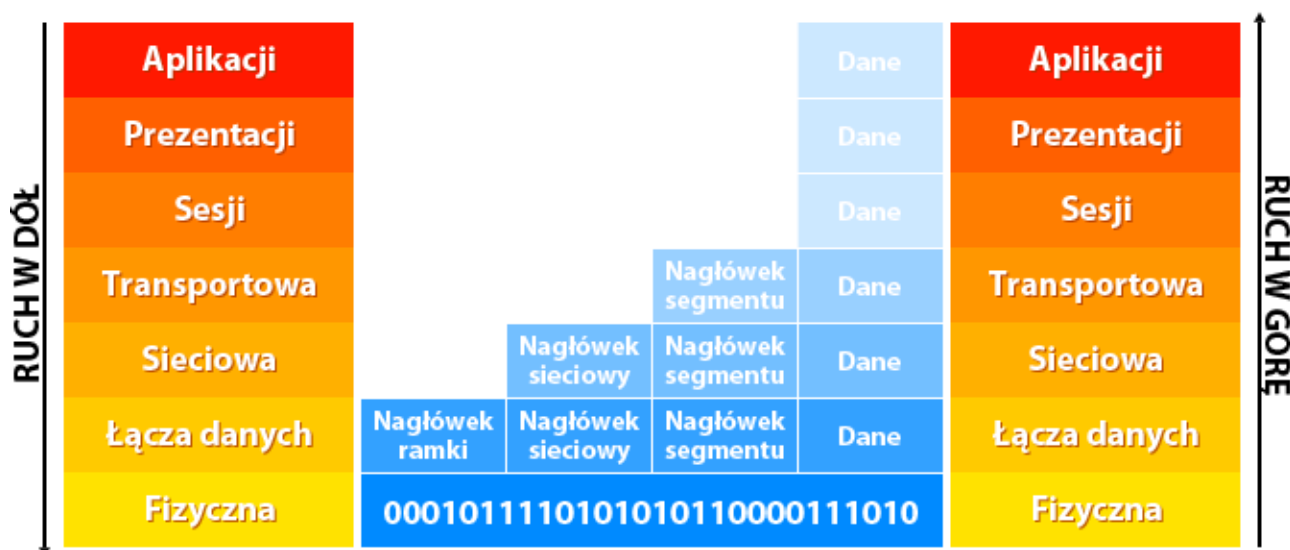
1.3 Protokoły komunikacyjne

Protokół komunikacyjny to zbiór reguł umożliwiający wymianę informacji pomiędzy połączonymi urządzeniami. Do opisu przeznaczenia oraz obszaru działania poszczególnych protokołów stosowany jest tak zwany model odniesienia OSI ang. *Open System Interconnection Reference Model*). Jest on odwzorowaniem transmisji danych w sieciach komputerowych, a także wzorcem używanym do reprezentowania mechanizmów przesyłania informacji w sieci. Pozwala wyjaśnić, w jaki sposób dane pokonują różne warstwy w drodze do innego urządzenia w sieci, nawet jeśli nadawca i odbiorca. Celem stosowania go jest podział procesu komunikacji sieciowej na mniejsze, łatwiejsze do zarządzania procesy.

Model ten jest podzielony na 7 warstw, których hierarchia przedstawiona jest poniżej

[<http://zstzbaszynek.pl/wp-content/uploads/2013/09/warstwy.png>].

Warstwy w modelu odniesienia OSI



Ilustracja 1.3.1: Model OSI

Warstwy aplikacji, prezentacji i sesji zajmują się współpracą z oprogramowaniem. Nazywane są warstwami wyższymi. Pozostałe to warstwy niższe.

Warstwa fizyczna – jest ona fundamentem całej komunikacji. odbiera dane z warstwy łączy danych i przesyła je w medium transmisyjnym jako bity zaprezentowane jako sygnał elektryczny, radiowy lub optyczny.

Warstwa łączy danych – jej zadaniem jest przetwarzanie danych pomiędzy warstwą fizyczną, a wyższymi. Na podstawie bitów odebranych przez warstwę niższą tworzy tak zwane ramki, które są najmniejszą jednostką informacji w modelu OSI. Tworzą one pakiety istniejące w wyższych warstwach. Analogicznie jest w przypadku wysyłania danych. Pakiety dzielone są na ramki. Te z kolei na bity warstwy fizycznej. Każdej ramce przypisane zostaje szereg danych technicznych. Do odebranego z warstw wyższych pakietu dodawany jest nagłówek, w którym znajduje się suma kontrolna CRC. Odbiorca pakietu, na podstawie sumy kontrolnej jest w stanie sprawdzić poprawność przesłanych danych i zdecydować czy dane zostaną przesłane do wyższych warstw. Jeżeli suma kontrolna obliczona po odebraniu pakietu nie jest zgodna z sumą kontrolną zapisaną w nagłówku – pakiet jest odrzucany.

Warstwa ta jest podzielona na dwie podwarstwy: kontroli łączy logicznego (LLC – Logical Link Control) i kontroli dostępu do nośnika (MAC – Media Access Control). Podwarstwa kontroli dostępu do nośnika (MAC) umieszcza adresy fizyczne nadawcy i odbiorcy pakietu w nagłówku. Adres MAC jest 48 bitowym, szesnastkowym, niepowtarzalnym adresem sprzętowym urządzenia sieciowego. Podwarstwa LLC jest przede wszystkim odpowiedzialna za rozdzielanie, zwielokrotnianie danych transmitowanych przez podwarstwę MAC (podczas transmitowania) oraz łączenie ich (podczas odbierania), jeżeli zachodzi taka potrzeba, sterowanie przepływem, detekcję i retransmisję zgubionych pakietów.

W tej warstwie tej działają takie urządzenia sieciowe jak np. most i przełącznik. [<http://swiatlan.pl/protokoly/model-osi.html>]

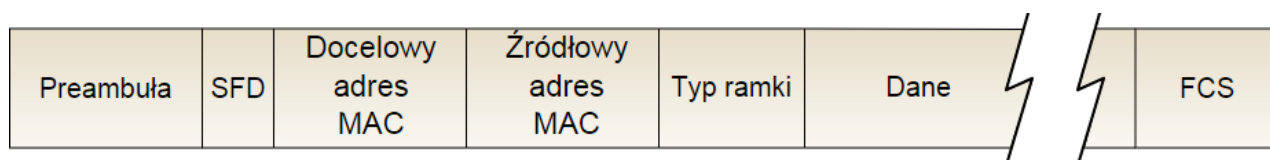
Technologią działającą w opisanych powyżej warstwach jest Ethernet, będący najpopularniejszym standardem w sieciach lokalnych. Opisuje specyfikację przewodów oraz

przesyłanych nimi sygnałów jak również formaty ramek i protokoły w tych warstwach. Całość tej specyfikacji znajduje się w standardzie IEEE 802. Działa w oparciu o topologię magistrali w której każdy z przyłączonych węzłów (urządzeń) ma przypisany niepowtarzalny adres MAC (ang. *Media Access Control*).

Istnieją 4 standardy ramek w technologii Ethernet:

- Ethernet wersja 1 – już nieużywana,
- Ethernet wersja 2 (Ethernet II) – zwana też ramką DIX od firm DEC, Intel i Xerox, które opracowały wspólnie ten typ ramki i opublikowały w 1978. Jest ona w tej chwili najczęściej stosowana,
- IEEE 802.x LLC,
- IEEE 802.g LLC.

Poniżej ilustracja ramki Ethernet (źródło: Wiki)



Ilustracja 1.3.2: Ramka Ethernet

Preambuła – składająca się z 7 bajtów złożonych z naprzemiennych jedynek i zer.

SFD – (ang. *start frame delimiter*), czyli znacznik początkowy ramki w postaci sekwencji 8 bitów (1 bajt).

Adres MAC odbiorcy (6 bajtów), adres MAC nadawcy (także 6 bajtów).

Typ (2 bajty) – jeżeli wartość jest równa lub większa od 1536 (w zapisie szesnastkowym 0x0600), to określa typ protokołu który jest używany, jeżeli mniejsza to oznacza długość danych w ramce. Na przykład, wartość od 0x0800 oznacza, że rama zawiera datagramu IPv4. Podobnie od 0x0806 wskazuje ramkę ARP, 0x8100 wskazuje ramki IEEE 802.1Q i 0x86DD wskazuje ramkę IPv6.

Dane (46 – 1500 bajtów) – jeżeli dane mniejsze niż 46 bajtów, to uzupełniane są zerami.

FCS – (ang. *Frame Check Sequence*) suma kontrolna (4 bajty).

Warstwa sieciowa - jest odpowiedzialna za adresowanie i trasowanie w sieci. Jako jedyna dysponuje wiedzą dotyczącą jej fizycznej topologii. Zajmuje się kontrolą przepływu danych – zapewnia odpowiednie skierowanie transmisji pakietów do urządzeń, które nie są ze sobą bezpośrednio połączone. Warstwa ta nie posiada żadnych mechanizmów kontroli poprawności przesyłanych danych. Nie musi zapewniać pewności transmisji, więc w razie błędu pomija niepoprawne pakiety danych.

Protokołami działającymi w tej warstwie są IP, ICMP, ARP, RARP jak również protokoły trasowania (ang. routing), takie jak **RIP** (Routing Information Protocol), **OSPF** (Open Shortest Path First), **BGP** (Border Gateway Protocol), **EIGRP** (Enhanced Interior Gateway Routing Protocol) oraz inne. Ich zadaniem jest wyznaczenie następnego urządzenia do którego ma zostać przekazany pakiet, tak aby trafił do odbiorcy. [źródło <http://swiatlan.pl/protokoly/protokol-ip.html>]

Do adresowania pakietów w warstwie sieciowej służy protokół IP (ang. *Internet Protocol*), który umożliwia przesyłanie danych w formie pakietów. Pakiety zawierają adresy odbiorcy, nadawcy oraz całość lub fragment przesyłanych danych. Głównym zadaniem tego protokołu jest wybór trasy oraz przesłanie nią pakietów z danymi. W razie awarii IP będzie starał się dostarczyć pakiety trasami alternatywnymi. Jest czasami określany jako zawodny protokół przesyłania pakietów, ponieważ nie daje żadnej gwarancji, że pakiety dotrą do odbiorcy. Pakiet podczas przesyłania może zostać zgubiony, zatrzymany, dotrzeć w innej kolejności niż podczas wysyłania lub zostać przekazany z błędami. Protokół IP jest protokołem bezpołączeniowym - przed rozpoczęciem transmisji nie jest zestawiana wirtualna sesja komunikacyjna pomiędzy dwoma hostami, które nie komunikowały się ze sobą wcześniej. Każdy pakiet obsługiwany jest oddzielnie. Pakiety z danymi jednej transmisji mogą zostać przekazane zupełnie innymi trasami, część z nich może zostać zgubiona lub dotrzeć z opóźnieniem. Ponadto nie ma obsługi powiadamiania nadawcy i odbiorcy o błędach w transmisji. Wszystkie te funkcje muszą być obsługiwane przez inne warstwy modelu OSI. [źródło <http://swiatlan.pl/protokoly/protokol-ip.html>]

Aby dwa urządzenia mogły między sobą przysyłać dane konieczne było wprowadzenie **adresów IP** – unikalnych numerów przyporządkowanych interfejsom sieciowym urządzeń znajdujących się w danej sieci fizycznej. Adresy IP urządzeń jednej sieci nie mogą się powtarzać. W Internecie przyznawaniem adresów IP zajmuje się organizacja **IANA** (Internet Assigned Numbers Authority).

Nagłówek protokołu IP przedstawiony jest poniżej.

Offsets	Bity 0-3	4-7	8-15	16-18	19-23	24-31
0	Wersja	IHL	Typ usługi / DS (6b) + ECN (2b)	Długość całkowita		
32	Identyfikator			Flagi	Przemieszczenie fragmentacji	
64	TTL		Protokół	Suma kontrolna nagłówka		
96	Adres źródłowy					
128	Adres docelowy					
160	Opcje					Dopełnienie

Ilustracja 1.3.3: Zobrazowanie ramki protokołu IP

Wersja

Określa wersję protokołu, obecnie 4 lub 6.

IHL (*Internet Header Length*)

Długość nagłówka wyrażona w liczbie 4-bajtowych części (np. wartość 5 oznacza 20 bajtów).

Typ Obsługi (*Type of Service*)

Znaczenie tego pola jako ToS zostało zdefiniowane w RFC 1349 - określało, jaki priorytet powinien mieć pakiet. Oprogramowanie routerów najczęściej ignoruje to pole. Obecnie obowiązują normy RFC 2474 i RFC 2475 definiujące podział tego pola na odpowiednio 6-bitowe DS - *Differentiated Services* i 2-bitowe ECN - *Explicit Congestion Mode*. Pierwsze z nich określa sposób przekazywania danych w węźle sieciowym (PHB - *Per Hop Behaviour*), drugie zawiadamia o zatorach.

Długość całkowita

Zawiera długość pakietu w bajtach (maksimum 65535 bajtów – maksymalna wartość liczby 16-bitowej; minimum 20 bajtów, bo taka jest długość nagłówka).

Identyfikator

Pomaga poskładać pakiet, który został podzielony na części.

Flagi

Trzy bity, pierwszy to aktualnie zawsze zero, drugi to flaga do-not-fragment (DF), określająca czy pakiet może być fragmentowany, trzeci to flaga more-fragments-following (MF), informująca o tym czy istnieją dalsze fragmenty pakietu.

Przemieszczenie fragmentu

Pole służy do złożenia w całość pakietu, określając miejsce danego fragmentu w całym pakiecie.

TTL (*Time To Live*)

Liczba przeskoków, przez które może przejść, zanim zostanie zignorowany (routery i komputery zmniejszają tę wartość o 1, gdy przesyłają pakiet), przykładowo TTL = 16 pozwala na przejście przez 16 routerów, zanim zostanie zignorowany).

Protokół

Określa jaki protokół warstwy transportowej będzie wykorzystany do dostarczenia pakietu na miejsce.

Suma kontrolna nagłówka

Wartość używana do sprawdzania poprawności pakietu.

Adres źródłowy i adres docelowy

32-bitowe adresy IP.

[źródło wikipedia]

ICMP (ang. *Internet Control Message Protocol*, internetowy protokół komunikatów kontrolnych), pełni funkcje kontrolne, diagnostyczne i informacyjne. Jest on używany przez polecenia sprawdzające poprawność połączenia. Korzystają z niego programy takie jak ping oraz traceroute.

IGMP (ang. *Internet Group Management Protocol*) – wykorzystywany przez komputery do powiadamiania routerów w swojej sieci o chęci przyłączenia się do lub odejścia z określonej grupy multicastowej.

ARP (ang. *Address Resolution Protocol*) – jego zadaniem jest mapowanie adresów fizycznych MAC na adresy logiczne IP. Jest stosowany w technologiach Ethernet, Token Ring oraz FDDI. Dzięki niemu systemy operacyjne mogą budować tak zwaną tablicę ARP zawierającą pary adresów (MAC i IP). Protokół ARP nie jest niezbędny do działania sieci komputerowych, może zostać zastąpiony przez statyczne wpisy w tablicy ARP, przyporządkowujące adresom warstwy sieciowej adresy fizyczne na stałe. Jednak gdy adresy nadawane są dynamicznie i nowe urządzenie pojawia

się w sieci wykorzystuje polecenia tego protokołu do pozyskania adresów fizycznych urządzeń z którymi chce się komunikować.

Warstwa transportowa – najwyższa z warstw niższych modelu OSI. Zapewnia kontrolę komunikacji pomiędzy urządzeniami na poziomie logicznym. Protokoły warstwy transportowej stosowane są w przypadku urządzeń końcowych, nie mają natomiast zastosowania w urządzeniach infrastruktury sieciowej. Warstwa transportowa zajmuje się również kontrolą poprawności i integralności danych. To tutaj są zdefiniowane mechanizmy, które odpowiadają za ponowienie transmisji danych w przypadku gdy dane nie mogły zostać dostarczone do odbiorcy. Warstwa transportowa rejestruje informacje o utraceniu połączenia pomiędzy urządzeniami i pozwala na bezpieczne zakończenie transmisji. Dane zbyt duże do przesłania za jednym razem są dzielone na tzw. segmenty, które są kolejgowane i wysyłane wg nadanych im priorytetów. Protokołami używanymi w tej warstwie są TCP i UDP.

TCP (ang. Transmission Control Protocol) – jest protokołem odpowiadającym za zapewnienie niezawodności wymiany danych pomiędzy aplikacjami na różnych maszynach, której nie zapewniają protokoły niższych warstw poprzez zestawienie tak zwanego wirtualnego połączenia. Połączenie to działa w architekturze klient-serwer. Serwer oczekuje na nawiązanie połączenia na określonym porcie. Port jest jedynie jednym z parametrów zapisywanym w nagłówku pakietu w postaci numeru. Samo połączenie znajdować może się w różnych stanach (maszyna stanów TCP) opisanych w dokumencie RFC793.

Poniżej ramka protokołu.

Offset	Oktet	0								1								2								3							
Oktet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Port nadawcy																Port odbiorcy															
4	32	Numer sekwencyjny																															
8	64	Numer potwierdzenia (jeżeli flaga ACK jest ustawiona)																															
12	96	Długość nagłówka				Zarezerwowane				N S	C W R E	E C G	U R K	A C H	P S H	R S T	S Y N	F I N	Szerokość okna														
16	128	Suma kontrolna																Wskaźnik priorytetu (jeżeli flaga URG jest ustawiona)															
20	160	Opcje (jeżeli długość nagłówka > 5, to pole jest uzupełniane "0")																															
...																															

Ilustracja 1.3.4: Zobrazowanie ramki protokołu TCP

Port nadawcy – 16-bitowy numer identyfikujący port nadawcy.

Port odbiorcy – 16-bitowy numer identyfikujący port odbiorcy.

Numer sekwencyjny – 32-bitowy identyfikator określający miejsce pakietu danych w pliku przed fragmentacją (dzięki niemu, można "poskładać" plik z poszczególnych pakietów).

Numer potwierdzenia – 32-bitowy numer będący potwierdzeniem otrzymania pakietu przez

odbiorcę, co pozwala na synchronizację nadawanie-potwierdzenie.

Długość nagłówka – 4-bitowa liczba, która oznacza liczbę 32-bitowych wierszy nagłówka, co jest niezbędne przy określaniu miejsca rozpoczęcia danych. Dlatego też nagłówek może mieć tylko taką długość, która jest wielokrotnością 32 bitów.

Zarezerwowane – 3-bitowy ciąg zer, zarezerwowany dla ewentualnego przyszłego użytku.

Flagi – 9-bitowa informacja/polecenie dotyczące bieżącego pakietu. Poszczególne flagi oznaczają:

- **NS** – (ang. Nonce Sum) jednobitowa suma wartości flag ECN (ECN Echo, Congestion Window Reduced, Nonce Sum) weryfikująca ich integralność
- **CWR** – (ang. Congestion Window Reduced) flaga potwierdzająca odebranie powiadomienia przez nadawcę, umożliwia odbiorcy zaprzestanie wysyłania echa.
- **ECE** – (ang. ECN-Echo) flaga ustawiana przez odbiorcę w momencie otrzymania pakietu z ustawioną flagą CE
- **URG** – informuje o istotności pola "Priorytet"
- **ACK** – informuje o istotności pola "Numer potwierdzenia"
- **PSH** – wymusza przesłanie pakietu
- **RST** – resetuje połączenie (wymagane ponowne uzgodnienie sekwencji)
- **SYN** – synchronizuje kolejne numery sekwencyjne
- **FIN** – oznacza zakończenie przekazu danych

Szerokość okna – 16-bitowa informacja o tym, ile danych może aktualnie przyjąć odbiorca. Wartość 0 wskazuje na oczekiwanie na segment z innym numerem tego pola. Jest to mechanizm zabezpieczający komputer nadawcy przed zbyt dużym napływem danych.

Suma kontrolna – 16-bitowa liczba, będąca wynikiem działań na bitach całego pakietu, pozwalająca na sprawdzenie tego pakietu pod względem poprawności danych. Obliczana jest z całego nagłówka TCP z wyzerowanymi polami sumy kontrolnej oraz ostatnich ośmiu pól nagłówka IP stanowiących adresy nadawcy i odbiorcy pakietu.

Wskaźnik priorytetu – jeżeli flaga URG jest włączona, informuje o ważności pakietu.

Opcje – czyli ewentualne dodatkowe informacje i polecenia:

- **0** – koniec listy opcji
- **1** – brak działania
- **2** – ustawia maksymalną długość segmentu

W przypadku opcji 2 to tzw. **Uzupełnienie**, które dopełnia zerami długość segmentu do wielokrotności 32 bitów (patrz: informacja o polu "Długość nagłówka")

Protokół TCP wymusza aby wysyłany pakiet wyposażony był w dodatkowe dane, które zapewniając ich dotarcie do odbiorcy. Powoduje to zwiększenie jego rozmiarów, a ewentualna

retransmisja danych znacznie zwiększa ruch sieciowy. Czasem jednak nie jest konieczne aby mieć pewność czy wszystkie wysłane dane dotarły ale ich aktualność np. transmisja obrazu na żywo. Sprawdza się w takich wypadkach protokół nie zestawiający połączenia UDP.

UDP (ang. *User Datagram Protocol* – protokół pakietów użytkownika) – podobnie jak IP jest bezpołączeniowy i nie gwarantuje dotarcia wysłanej wiadomości. W stosunku do TCP jest on znacznie uproszczony. Nagłówek zawiera jedynie adresy i numery portów nadawcy i odbiorcy, sumę kontrolną oraz długość danych.

Warstwa sesji – zarządza sesjami transmisyjnymi poprzez nawiązywanie i zrywanie połączeń między aplikacjami. Zarządza danymi sesji wysyłanymi i odbieranymi przez nie.

Warstwa prezentacji – odpowiada za prezentację danych użytkownikowi, obsługę znaków narodowych, formatów graficznych oraz kompresję i szyfrowania. Gdy dane mają zostać wysłane dzięki niej niższe warstwy zawsze otrzymują dane w tym samym formacie. Obsługuje na przykład formaty plików graficznych takich jak MPEG, JPG, GIF itp. Protokoły w niej występujące to MIME, XDR, TLS, SSL.

Warstwa aplikacji – obszar jej działania to aplikacje użytkownika. Komunikuje się z warstwami niższymi poprzez interface API (ang. *Application Programming Interface*), które umożliwia tworzenie tzw. gniazd sieciowych (ang. *socket*) pracujących w architekturze klient-serwer. Protokoły działające w tej warstwie to m.in. HTTP, SMTP, IMAP, POP, DHCP, FTP, DNS, Telnet oraz wiele innych. Użytkownik ma możliwość tworzenie własnych protokołów na dowolnych portach powyżej nr 1024.

2. Cel i założenia pracy

Celem pracy jest opracowanie projektu koncepcyjnego oraz wykonie podstawowych elementów systemu informatycznego umożliwiającego automatyczne zarządzanie dostępem do płatnego parkingu. Ma on być przede wszystkim rozwiązaniem ułatwiającym funkcjonowanie instytucji, pozwalającym także na uczciwe regulowanie kwestii finansowych związanych z korzystaniem z parkingu. Docelowym miejscem zastosowania systemu będzie parking przy wydziale mechatroniki, więc system powinien uwzględniać, że obsługiwany ma również być szlaban blokujący wjazd, a zezwolenie na wjazd ma być weryfikowane przed szlabanem tak jak to ma miejsce obecnie. Mimo to w rozdziale trzecim opisane zostaną także rozwiązania obsługujące parkingi otwarte.

Jak już wspomniano we wstępie w zakresie zarządzania parkingiem jakim zajmować będzie się system wchodzić będą takie czynności jak:

- Rejestracja nowych użytkowników podczas, której podawać one będą swoje dane osobowe.
- Tworzenie sesji logowania w systemie ze antykadencją.
- Dodawanie pojazdów, którymi użytkownicy będą wjeżdżać.
- Autoryzacja przy wjeździe na teren parkingu.
- Sterowanie szlabanem.

Przed przystąpieniem do wykonywania projektu powziąć należy szereg założeń. Najistotniejsze z nich zawarte zostały w karcie pracy. Są to jej zakres oraz podstawowe wymagania. Przedstawione są poniżej.

Zakres pracy:

- Przegląd istniejących rozwiązań realizujących podobne funkcje.
- Schemat ideowy i blokowy systemu.
- Opracowanie oprogramowania na platformę E2LP i do sterowania szlabanem automatycznym.
- Projekt bazy danych do przechowywania informacji niezbędnych do realizacji funkcji systemu.
- Opracowanie witryny internetowej do rejestracji i autoryzacji użytkowników.
- Opracowanie i przeprowadzenie testów wybranych elementów systemu.

Podstawowe wymagania:

- Wykorzystanie platformy E2LP wraz z zewnętrznym modulem mikroprocesorowy opartym na mikrokontrolerze LPC2368
- Wykorzystanie dokumentacji szlabanu automatycznego 615 BRP
- Implementacja komunikacji z platformą E2LP w standardzie Ethernet

Istotne są także założenia wysnute przez autora. Oddzielne tyczą się części softwarowej oraz sprzętowej. Odnoszące się do tej softwarowej znajdują się poniżej:

- I. System ma posiadać bazę danych przechowującą informacje o użytkownikach i samochodach oraz rejestrować wszystkie przejazdy przez szlaban.
- II. Ma udostępniać internetowy interfejs dla użytkowników chcących korzystać z parkingu. Niezbędne będzie więc opracowanie witryny internetowej umożliwiającej logowanie oraz rejestrowanie nowych użytkowników, a także dodawanie nowych samochodów.
- III. Osoby mają mieć możliwość rejestrowania kilku pojazdów, a także deklarowanie wjazdu jednym samochodem kilku użytkowników. Jednocześnie powinno być w sposób jednoznaczny określone jaka osoba i jakim samochodem przejeżdża przez szlaban.
- IV. Powinny istnieć różne formy zezwolenia na wjazd. Dla zwykłych użytkowników: abonament oraz wjazd jednorazowy. Dla osób uprzywilejowanych natomiast brak konieczności posiadania uregulowania finansowego za możliwość przejazdu. Egzekwowanie opłat nie jest w zakresie tej pracy, więc użytkownicy posiadać będą jedynie atrybut określający na jakich zasadach korzystają z parkingu.
- V. Wykorzystane oprogramowanie powinno być bezpłatne.

Założenia do części sprzętowej są następujące:

1. Wykorzystanie komponentów platformy E2LP. Zwłaszcza niezbędny będzie moduł rozszerzający NXP ARM Extension Board.
2. Opracowanie sprzętowego interfejsu użytkownika wjeżdżającego na teren parkingu. Mają być udostępnione dwa terminale do wpisywania kodów – jeden dla wjazdu drugi dla wyjazdu. Oprócz tego wykorzystanie zastany wyświetlacz LCD znajdujący się na płycie czołowej E2LP. Był on z powodzeniem testowany w pracy przejściowej [] i jest to założenie będące zgodne z pierwszym.
3. Do implementacji logiki w układzie FPGA wykorzystany zostanie znany z pracy przejściowej [] procesor softwarowy PicoBlaze.
4. Na podstawie ukończonej integracji platformy E2LP z siecią LAN opracowana zostanie dokumentacja pozwalająca innym osobom chcącym wykorzystać taką funkcjonalność na proste jej użycie.

3. Analiza istniejących rozwiązań

Istnieje wiele rozwiązań realizujących podobną funkcję do projektowanego systemu. W tym rozdziale przedstawiona zostanie analiza wybranych. Przeglądając te rozwiązania podzielić je można we względu na sposób uiszczania opłaty na trzy grupy:

- Płatność poprzez telefon
- Automaty samoobsługowe
- Rezerwacja internetowa

3.1 Płatność przez telefon

Telefon komórkowy jest dzisiaj tak powszechny, że posiada go praktycznie każda osoba. Większość z nich to smartfony, a każdy udostępnia możliwość wysyłania SMS oraz dzwonienia. Stwarza to więc możliwość rezerwacji różnych usług oraz płacenia za nie właśnie korzystając z telefonu. Usługę pozwalającą płacić za miejsca parkingowe w tak zwanych strefach płatnego parkowania udostępnia firma MobiParking.

Jest to usługa pozwalająca na płacenie telefonem komórkowym za postój w strefach płatnego parkowania. Komunikacja pomiędzy Użytkownikiem, a usługą jest realizowana za pomocą 3 kanałów: SMS, aplikacji mobilnej oraz numerów specjalnych. MobiParking został wyłoniony w drodze przetargu ogłoszonego przez Zarządu Dróg Miejskich w Warszawie, stając się jedynym obowiązującym kanałem płatności komórką za parkowanie w stolicy. Stosowany jest on również w innych polskich miastach.

[<http://di.com.pl/warszawa-mobiparking-czyli-placenie-za-parking-przez-telefon-42296>]

[<http://skycash.com/mobiparking.html>]

[<http://arch.um.bialystok.pl/1048-mobiparking/default.aspx>]

Z usługi mobiParking mogą korzystać wszyscy posiadacze komórek i smartfonów, bez względu na sieć GSM. Po dokonaniu rejestracji należy zasilić utworzone konto środkami pieniężnymi oraz zaopatrzyć się i umieścić w pojeździe identyfikator mobiParking. Jest nim specjalna naklejka dostępna w kilkunastu punktach odbioru lub wydruk, którego wzór można znaleźć na stronie internetowej (www.mobiparking.pl), w zakładce „Aktywacja”.

Aplikację mobilną można pobrać bezpośrednio na telefon z tej samej strony. Obsługa aplikacji sprowadza się do wyboru gotowych komend, które są prezentowane w formie menu na ekranie telefonu. Oprócz podstawowych funkcji (tj. parkowanie pojazdu w wybranym mieście) aplikacja posiada możliwość sprawdzenia stanu dostępnych środków pieniężnych, czy też sprawdzenia informacji odnośnie kosztów parkowania w danym mieście.

Główne funkcjonalności systemu mobiParking:

- Regulowanie należności z dowolnego miejsca, bez konieczności szukania kart postojowych.
- Naliczanie opłaty za czas parkowania zgodnie z aktualnymi stawkami opłat w strefie bez konieczności przewidywania czasu parkowania.
- Samodzielna kontrola czasu parkowania i uzupełnianie środków wykorzystywanych do parkowania.
- Informacja o stanie swoich „środków parkingowych” w czasie rzeczywistym.

W celu uruchomienia usługi **mobiParking** należy wykonać 3 czynności, które zostały opisane poniżej.

- **Aktywacja usługi** polega na wysłaniu SMS z numerem rejestracyjnym samochodu o treści **A(spacja)Nr_Rejestracyjny**(np. **A BIA45651**) lub też dodaniu numeru rejestracyjnego pojazdu z poziomu zainstalowanej aplikacji mobilnej.
- **Uzupełnienie środków parkingowych.** Dostępne są różne oferty dla klientów indywidualnych oraz biznesowych.
- **Oznaczenie pojazdu identyfikatorem mobiParking.** Należy przykleić naklejkę informującą o korzystaniu z usługi mobiParking na przednią szybę samochodu (od wewnątrz w prawym dolnym rogu przedniej szyby pojazdu). Naklejki wydawane są w Biurze Strefy Płatnego Parkowania w Urzędzie Miejskim danego miasta bądź innej oddelegowanej do tego jednostce administracyjnej. Identyfikator usługi można pobrać w wersji elektronicznej do samodzielnego wydruku.

Korzystanie z usługi przewiduje dwa warianty parkowania.

I. Gdy nie wiadomo ile czasu będzie trwało parkowanie

Dodatkowo jak wspomniano we wstępnym opisie tej usługi przewidziane są trzy sposoby płatności:

A. Parkowanie z wykorzystaniem kodu SMS:

Rozpoczęcie parkowania polega na wysłaniu SMS na numer **82002** o treści:

[skrót_nazwy_miasta_lub_strefy_parkingowej][kolejny numer rejestracyjny samochodu przypisany do danego numeru telefonu]

np. WAR2.

System weryfikuje uruchomienie usługi oraz potwierdza rozpoczęcie pobierania należności za parkowanie. SMS zwrotny informuje użytkownika o stanie „środków parkingowych”.

Zakończenie parkowania polega na wysłaniu SMS treści **K**. Po zakończeniu parkowania system wysyła użytkownikowi informację o czasie postoju, pobraniu należności oraz stanie dostępnych środków. Aby zakończyć parkowanie kolejnego pojazdu z listy dostępnych pojazdów należy wysłać SMS z komendą **K1, K2, itd.** W odpowiedzi przesyłany jest SMS potwierdzający zakończenie parkowania.

B. Parkowanie z wykorzystaniem szybkich kodów:

Rozpoczęcie *lub* zakończenie parkowania poprzez Krótkie kody wygląda bardzo podobnie.

Jeżeli Użytkownik wpisze poniższy kod dokona rozpoczęcia wniesienia opłaty za parkowanie w Białymstoku. Ponowne wprowadzenie tego samego kodu doprowadzi do zakończenia pobierania opłaty.

Należy wpisać poniższy kod z klawiatury aparatu (tak jakby wybierało się numer telefonu) a następnie zatwierdzić go przyciskiem inicjującym połączenie. Kod wygląda

następująco:

**[sekwencja_znaków_zależna_od_strefy_lub_miasta]#* rozpoczęcie parkowania pierwszego na liście samochodu

**[sekwencja_znaków_zależna_od_strefy_od_miasta]*[numer_samochodu]#* rozpoczęcie parkowania wybranego samochodu System weryfikuje uruchomienie usługi oraz potwierdza rozpoczęcie pobierania należności za parkowanie.

C. Parkowanie za pomocą aplikacji mobilnej

Aby zaparkować pojazd należy uruchomić aplikację, a następnie:

1. wybrać miasto, w którym ma być zaparkowany pojazd
2. wybrać opcję START
3. wybrać NUMER REJESTRACYJNY, który ma zostać zaparkowany
4. ZATWIERDŹ KODEM PIN, chęć zaparkowania pojazdu
5. na wyświetlaczu pojawi się potwierdzenie rozpoczęcia parkowania ze wskazaniem numeru rejestracyjnego pojazdu oraz lokalizacji, w jakiej pojazd został zaparkowany

II. Gdy wiadomo ile czasu będzie trwało parkowanie

A. Parkowanie za pomocą SMS:

Rozpoczęcie parkowania polega na wysłaniu na numer **82002** SMS o treści:

[przewidywana_liczba_minut][skrót_nazwy_miasta_lub_strefy_parkingowej][kolejny_numer_rejestracyjny_samochodu_przypisany_do_danego_numeru_telefonu]

np. 30WAR2

Należy pamiętać, że liczba minut może być tylko wielokrotnością liczby 30. Przez ten czas osoba będzie mogła pozostawić swój pojazd na danym parkingu. Jeżeli czas postoju będzie dłuższy i personel weryfikujący to odnotuje - automatycznie pobrana zostanie z konta kwota zapewniająca parkowanie przez kolejne pół godziny lub czas przez jaki samochód pozostawał na parkingu.

B. Parkowanie za pomocą szybkich kodów:

Należy z klawiatury aparatu telefonicznego wpisać kod tak jak w przykładzie (tak jakby wybierało się numer telefonu) a następnie zatwierdzić go przyciskiem inicjującym połączenie.

**[sekwencja_znaków_zależna_od_strefy_lub_miasta]*[liczba_minut]#* - rozpoczęcie parkowania pierwszego na liście samochodu

**[sekwencja_znaków_zależna_od_strefy_od_miasta]*[numer_samochodu]*[liczba_minut]#* - rozpoczęcie parkowania wybranego samochodu System weryfikuje uruchomienie usługi oraz potwierdza rozpoczęcie pobierania należności za parkowanie.

C. Parkowanie za pomocą aplikacji mobilnej

Aby zapłacić za czasowe parkowanie pojazdu należy uruchomić aplikację, a następnie:

1. wybrać miasto, w którym ma być zaparkowany pojazd
2. wybrać opcję CZASOWE

3. wybrać NUMER REJESTRACYJNY, który ma zostać zaparkowany
4. ZATWIERDŹ KODEM PIN, chęć zaparkowania pojazdu
5. na wyświetlaczu pojawi się potwierdzenie rozpoczęcia parkowania ze wskazaniem numeru rejestracyjnego pojazdu oraz lokalizacji, w jakiej pojazd został zaparkowany oraz informacja o wykupionym czasie parkowania

Podsumowanie

Niewątpliwą zaletą systemu mobiParking są dwa warianty określania czasu parkowania oraz aż trzy dotyczące realizacji płatności. Wadą jest to, że obsługuje on jedynie otwarte parkingi nie korzystające ze szlabanów. Wspólną cechą mobiParking oraz innych podobnych rozwiązań z opłatą za parkowanie przez telefon jest to, że użytkownik nie musi rezerwować miejsca z wyprzedzeniem, ale także nie ma pewności czy w ogóle znajdzie wolne miejsce na danym parkingu. Różnorodność form opłat jest bardzo wygodna, ale korzystanie z telefonu wymaga dodatkowych opłat związanych z wysyłaniem SMS lub dzwonieniem. Aplikacja mobilna korzysta co prawda z internetowej transmisji danych wykorzystując technologię LTE bądź GPRS jednak wiąże się to także z dodatkowymi opłatami, a dostęp do internetu poprzez Wi-Fi nie zawsze jest możliwy.

3.2 Automaty samoobsługowe

Idea działania tej formy płatności jest następująca. W tzw. strefie płatnego parkowania w odpowiednim miejscu znajdują się automaty umożliwiające opłacenie miejsca parkingowego. Stosowane są dwa rodzaje takich urządzeń. Parkometry oraz parkomaty.

3.2.1 Parkometry

Jest urządzeniem, które działa podobnie do parkomatu lecz nie wydaje ono biletów, a jest przypisane do konkretnego miejsca na parkingu. Jest to kosztowne rozwiązanie, gdyż liczba miejsc na parkingu a co za tym idzie tego typu urządzeń może być bardzo duża.

Historia parkometrów jest znacznie dłuższa niż parkomatów. Pierwszy raz zastosowane zostały w Oklahoma City w Stanach Zjednoczonych w 1932 roku. Ich potrzeba wynikała z powodu rosnącej liczby samochodów w mieście. Właściciele sklepów w centrum miasta narzekali, że miejsca parkingowe dla ich potencjalnych klientów zajmowane są przez pracowników pobliskich biur. Prawnik i dziennikarz Carl Magee. Zaproponował on, aby za postój na miejscu parkingowym kierowca był zobowiązany płacić, a opłatę tą powinien uiszczać w specjalnym urządzeniu. Po pewnym czasie sam skonstruował pierwszy parkometr.

Zasada działania była bardzo prosta. Po wrzuceniu odpowiedniej monety uruchamiany był specjalny mechanizm sprężynowy. Na panelu czołowym widoczna była tarcza pokazująca ile czasu pozostało do końca parkowania. Mechanizm działał bardzo podobnie do klasycznego zegarka. Tarcza po osiągnięciu skrajnej pozycji sygnalizowała koniec czasu parkowania.

Rozwiązanie na owe czasy okazało się zaskakująco skuteczne. Wprowadzono strefy z krótkim i długim okresem parkowania. Handlowcy mieli udostępnione miejsca parkingowe dla klientów parkujących przez kilka bądź kilkadziesiąt minut, a pracownicy biur parkowali na innych parkingach przez cały dzień. Doprowadziło to do uporządkowania kwestii parkowania w mieście, a z faktu iż właścicielem parkometrów było miasto zwiększyły się jego dochody.

Parkometry są dość przyjazne dla kierowców, którzy nie muszą zastanawiać się w jaki sposób należy uregulować postój na danym parkingu. Wsiadając z pojazdu mają przed sobą to urządzenie. W obsłudze jest ono również bardzo proste. Dodatkowo jeżeli jeden kierowca zakończy parkowanie wcześniej niż planował to inny może wykorzystać jego czas. Oczywiście nie działa to

w drugą stronę. Parkometr nie zwraca pieniędzy przed upływem czasu parkowania.

Jednak z powodu dużych kosztów w z wynikających z dużej liczby parkometrów związanych z każdym miejscem parkingowym są one praktycznie całkowicie wyparte z powodu bardziej opłacalnych parkomatów.

3.2.2 Parkomaty

Są to urządzenia zainstalowane na parkingu bądź w pobliżu niego. Umożliwiają opłatę za zarezerwowane miejsce na danym parkingu w różnych formach. Poniżej opisany zostanie konkretny przykład wykorzystania takich urządzeń.

W Gdyni wprowadzone zostały strefy płatnego parkowania wykorzystujące parkomaty Strada. Parkomaty te wyposażone są w panele słoneczne co rozwiązuje problem zasilania. [<http://www.sppgdynia.pl/jakdziala.php>].

Parkomaty solarne Strada zainstalowane w Gdyni umożliwiają wnoszenie opłat na dwa sposoby:

- gotówką za pomocą monet o nominałach od 10 groszy do 5 złotych,
- specjalną kartą strefową.

Bilet wykupiony w parkomacie, bez względu na jego lokalizację uprawnia do postoju na obszarze całej strefy w ramach opłaconego czasu. Zgodnie z zapisami obowiązującej Uchwały Rady Miasta Gdyni, kierowca parkujący swój pojazd na obszarze strefy zobowiązany jest do wniesienia opłaty niezwłocznie po rozpoczęciu parkowania, zatem niezbędne jest posiadanie przy sobie „drobnych” w ilości wystarczającej na opłacenie parkowania.

Jak już wspomniano wcześniej, obsługa urządzenia jest niezwykle prosta i nawet osoby, które nigdy wcześniej nie korzystały z tego rodzaju urządzeń, nie powinny mieć z nią najmniejszych problemów.

Na obudowie parkomatu znajdują się następujące elementy:

- Etykiety informacyjne: na jednej z nich znajduje się instrukcja w formie diagramu, na drugiej informacja o obowiązujących w strefie stawkach opłat.
- Ekran – wielofunkcyjny wyświetlacz służący informowaniu użytkownika na bieżąco podczas wnoszenia opłaty o stanie transakcji
- Przyciski obsługowe (od lewej):
 - Szary – przycisk służący zmianie języka komunikacji. Parkomat obsługuje cztery języki: polski jako język domyślny, angielski, niemiecki i rosyjski. Każdorazowe naciśnięcie szarego przycisku powoduje zmianę języka na następny w kolejności. Przy wyjściu ze stanu czuwania, zawsze zobaczymy komunikaty w języku polskim.
 - Żółty – w chwili obecnej przycisk ten nie ma przypisanej żadnej funkcji.
 - Niebieskie – dwa przyciski oznaczone symbolami „+” i „-”. Służą do podwyższania, bądź obniżania kwoty opłaty wnoszonej za pomocą karty strefowej.
 - Zielony (większy niż pozostałe) – służy do potwierdzenia transakcji przez użytkownika
 - Czerwony – przerwanie transakcji w dowolnym jej momencie przed ostatecznym zatwierdzeniem. W czasie wydruku biletu użycie czerwonego przycisku jest ignorowane.
- Wlot monet (poniżej przycisków) – oznaczony symbolem graficznym, takim samym jak na etykiecie informacyjnej.

- Głowica czytnika kart
- Kieszon zwrotu monet – zakryta czerwoną półprzezroczystą klapką – kierowca może tam odebrać monety zwrócone w wyniku anulowania transakcji, bądź odrzucone przez parkomat z jakiegokolwiek innej przyczyny
- Kieszon biletów – zakryta częściowo zieloną półprzezroczystą klapką – z niej należy odebrać bilet i umieścić go w samochodzie za przednią szybą w sposób umożliwiający odczytanie danych przez kontrolerów.



Ilustracja 3.2.1: Panel czołowy parkomatu

Oплата за pomocą monet

Parkomat w stanie czuwania wyświetla datę i godzinę. Wsunięcie monety w szczelinę wlotu monet powoduje wyjście ze stanu czuwania (po zmroku ekran jest podświetlany) a na ekranie pojawia się informacja o wniesionej kwocie oraz minimalnej i maksymalnej opłacie możliwej do wniesienia.

Zgodnie z zapisami Uchwały Rady Miasta Gdyni, opłata minimalna została ustalona na kwotę 80 gr i odpowiada 30 minutom parkowania, zaś opłata maksymalna wynosi aż 96 zł i odpowiada opłacie za niecałe 4 dni.

Kierowca chcący wnieść opłatę powinien wrzucić do urządzenia monety o łącznej wartości co najmniej 80 groszy i nie większej niż 96 zł. W trakcie wnoszenia opłaty na wyświetlaczu urządzenia pokazywana jest wartość już wniesionej opłaty oraz odpowiadająca jej godzina, do której parkowanie zostało opłacone. Kierowca decyduje, w którym momencie zatwierdzić transakcję. Jeżeli wniesiona kwota będzie większa niż opłata do końca danego dnia, zostanie ona przeniesiona na następny dzień, w którym obowiązują opłaty. Na przykład opłata w kwocie 5 zł wniesiona o godzinie 17:10 w piątek będzie skutkowałą wydaniem biletu, którego ważność upływa w najbliższy poniedziałek o godzinie 09:56.

Dla wygody kierowców, szczególnie przyjezdnych i turystów, parkomaty przyjmują opłaty we wszystkie dni tygodnia 24 godziny na dobę. Można zatem przyjechać w nocy z niedzieli na poniedziałek, wnieść opłatę i nie być zmuszonym do ранnego wstawania.

Należy pamiętać, że jednorazowo urządzenie może przyjąć 25 monet, zatem wysokie kwoty opłat powinny być wnoszone za pomocą monet o wysokich nominałach.

Wrzucenie kwoty większej niż maksymalna powoduje automatyczne zwrócenie pieniędzy, a bilet nie zostanie wydany. Wrzucenie kwoty mniejszej niż minimalna i zatwierdzenie jej przynosi analogiczny skutek.

Oplata za pomocą karty

Karta strefowa jest dystrybuowana przez Biuro Obsługi Strefy znajdujące się w Gdyni przy ul. Białostockiej 3. Użytkownicy nabywający kartę, powinni przeznaczyć pewną kwotę (maksymalnie 300 zł), którą karta zostanie „załadowana”. Środki zgromadzone na karcie mogą być wykorzystywane wyłącznie do opłat za parkowanie we wszystkich parkomatach zainstalowanych w Strefie Płatnego Parkowania w Gdyni.

Kierowca wnoszący opłatę za pomocą karty posługuje się nieco inną procedurą aniżeli ma to miejsce w przypadku bilonu.

Parkomat w stanie czuwania wyświetla datę i godzinę. Za pomocą niebieskich przycisków należy wyprowadzić urządzenie ze stanu czuwania, a następnie określić kwotę opłaty oraz odpowiadający jej czas końca parkowania. Przycisk oznaczony „+” powoduje skokowe zwiększenie kwoty, zaś oznaczony symbolem „-” jej zmniejszenie. Każde użycie jednego z przycisków powoduje zmianę kwoty o 50 groszy. Po osiągnięciu satysfakcjonujących parametrów, kierowca zatwierdza transakcję za pomocą zielonego przycisku, po czym proszony jest o zbliżenie karty do głowicy czytnika. Parkomat weryfikuje kartę i „zdejmuje” zdefiniowane przez kierowcę środki oraz wydaje bilet. Zaletą takiego rozwiązania jest chociażby brak konieczności wyjmowania karty z portfela, czy nawet z kieszeni. Po zakończonej transakcji na wyświetlaczu urządzenia pokazywany jest stan pozostałych środków na karcie.

Jeżeli kierowca zechce zrezygnować z transakcji, to w przypadku opłaty kartą może to zrobić nawet po jej zatwierdzeniu – wystarczy, że nie zbliży karty to głowicy czytnika. Transakcja zostanie automatycznie zakończona po ok. 30 sekundach.

Zaleca się, aby kierowcy weryfikowali zgodność danych wydrukowanych na bilecie. Prosimy o zgłaszanie do Biura Obsługi Strefy każdej stwierdzonej nieprawidłowości.

Podczas korzystania z parkomatów, kierowcy mogą spotkać się z komunikatem o treści „autodiagnostyka” lub „proszę czekać”. Komunikaty te są wyświetlane, kiedy urządzenie wykonuje test wewnętrzny podzespołów lub przesyła dane do centrum monitoringu. Operacje takie trwają maksymalnie kilkanaście sekund i w czasie ich trwania parkomat nie przyjmuje opłat. Należy poczekać, aż komunikat zniknie.

Uszkodzone urządzenia przy próbie rozpoczęcia opłaty wyświetlają komunikat o treści „Parkomat uszkodzony”.

3.2.3 Automaty wjazdowe

Ciekawe rozwiązania są w ofercie firmy ParkSystem. Znajdują się tam urządzenia podobne do parkomatów – tak zwane automaty wjazdowe. Ich przeznaczeniem są parkingi zamknięte ze szlabanem wjazdowym. Przykładowy system wyposażony jest w dwa szlabany – jeden wjazdowy drugi wyjazdowy. Związane z nimi terminale: wjazdowy, który wydaje bilety oraz wyjazdowy-płatniczy. Terminal wjazdowy posiada także czytnik kart abonamentowych. System wyposażony

jest również w pętlę indukcyjną ułatwiającą osobą niedosłyszącym korzystanie z parkingu. [<http://www.parksystem.pl/pl/oferta,1>]

Klient który chce wjechać na parking, podjeżdża pod terminal wjazdowy. Czujniki (pętle indukcyjne lub fotokomórki) rozpoznają pojazd. Opłaty nie są pobierane przed wjazdem na parking. Po przyciśnięciu przycisku na panelu automatu klient odbiera wydrukowany bilet wjazdowy. Po pobraniu biletu szlaban się otwiera i samochód wjeżdża na parking. Klienci posiadający wykupione karty abonamentowe, wjeżdżają na parking po przyłożeniu ich do czytnika.

Rozliczanie biletu w kasie automatycznej

Po przyłożeniu biletu do czytnika (skanera) znajdującego się w kasie automatycznej system sprawdzi datę oraz godzinę wjazdu. Na wyświetlaczu LCD pojawi się opłata za parkowanie (zgodnie z taryfą opłat). Klient uiszcza opłatę za pomocą monet bądź banknotów (opcjonalnie kart płatniczych lub zbliżeniowych). Automat wydaje resztę monetami. Po przyciśnięciu przycisku „dowód opłaty” kierowca może otrzymać paragon z wyszczególnionym podatkiem VAT. Kierowca zabiera bilet, który został opłacony i automatycznie przekodowany jako wyjazdowy z którym udaje się do pojazdu. Zwyczajowo stosuje się minimalny czas przeznaczony na wyjazd (tak aby kierowca miał wystarczająco dużo czasu na dojście do samochodu i wyjazd z parkingu). Jeżeli ten czas zostanie przekroczony to Klient będzie musiał dopłacić należność za rozpoczętą godzinę według ustalonej taryfy.

Rozliczanie w kasie ręcznej

Sposób rozliczania klienta i wszystkie czynności są podobne jak przy kasie automatycznej. Różnicę stanowi fakt, że bilet parkingowy podajemy pracownikowi parkingu i to on rozlicza nas z należności za postój. Po otrzymaniu od Pracownika Parkingu biletu wyjazdowego kierujemy się do samochodu. Kasa ręczna może spełniać funkcję przyjmowania opłat w sytuacjach awaryjnych, możliwość zarządzania abonamentami, funkcje rozliczenia zagubionego biletu przez klienta, zarządzanie kartami abonamentowymi (sprzedaż, kontrola ważności, kodowanie, blokowanie np. w przypadku zagubienia kradzieży), raportowanie przychodów w skali wybranego dnia, tygodnia, miesiąca, potencjalnie możliwa obsługa płatności przy użyciu kart kredytowych.

Podsumowanie

Parkomaty mogą być umiejscowione zarówno na parkingach otwartych jak i zamkniętych - przed szlabanem. Wadą tego rozwiązania są natomiast koszty utrzymania parkomatu. Z powodu przetwarzania i przesyłania krytycznych danych osobowych jakimi są m.in. parametry logowania np. w serwisie bankowym wymaga on zainstalowania wyjątkowo bezpiecznego sprzętu i oprogramowania. W przypadku płacenia gotówką musi być ona odpowiednio zabezpieczona przed złodziejami jak również cały automat przed zwykłymi chuliganami. Wymaga to więc dodatkowego dozoru takiego urządzenia przez służby porządkowe.

3.3 Rezerwacja internetowa

Rozwiązanie to jest najbliższe temu, które ma zostać opracowane w tej pracy. Ideą jest, aby osoba chcąc zaparkować swoje auto najpierw zarezerwowała dane miejsce w konkretnym czasie i ewentualnie z góry uiściła opłatę za tą usługę. Rozwiązania te są zwłaszcza na lotniskach gdzie samochody pozostawiane mogą być w formie depozytu. Przykładowo osoba wylatująca na dłuższy czas może przyjechać samochodem na lotnisko i pozostawić go na parkingu.

Taka możliwość parkowania udostępniona jest na lotnisku Schonefeld w Berlinie. Ruch pojazdów jest tam bardzo duży. Powoduje to liczne korki. System internetowy [<http://parken.berlin->

airport.de/pl/searchresult/] pozwala na rezerwacje miejsca parkingowego na kilka minut lub na dłuższy czas.

Najpierw należy wybrać czas w jakim auto ma znajdować się na parkingu. Jeżeli w tym czasie dostępne są wolne miejsca to użytkownik wprowadza dane swojego pojazdu. Najważniejszy jest nr rejestracyjny. Po uiszczeniu opłaty za zarezerwowane parkowanie przyznawane jest miejsce na parkingu. Służby lotniskowe przepuszczają na parking auto na podstawie nr rejestracyjnego weryfikując go poprzez odpowiednie oprogramowanie.

Bardzo podobne rozwiązanie znajduje się na lotnisku w Katowicach [<https://www.katowice-airport.com/pl/pasazer/parkingi-rezerwacja#>]. Po wejściu na stronę należy wybrać parking na którym chce się zostawić samochód. Następnie podać czas w jakim ma on tam być parkowany. Podobnie jak w opisywanym lotnisku berlińskim wymagane jest podanie nr rejestracyjnego oraz pozostałych danych pojazdu. Jeśli miejsce w danym czasie na wybranym parkingu jest dostępne to użytkownik jest zobowiązany uiścić opłatę. Jeżeli wszystkie powyższe czynności wykonane zostały poprawnie osoba ma wstęp na teren parkingu.

Inne bardzo ciekawe rozwiązanie zaproponowała chicagowska firma ParkWiz. Ich rozwiązanie sprawdza się nie tylko na lotniskach, które są oblegane praktycznie cały czas, ale również w miejscach obleganych w szczególnych okazjach np. koncerty [<http://www.komputerswiat.pl/nawosci/programy/2010/30/mobilna-rezerwacja-miejsc-parkingowych.aspx>].

Model biznesowy tego rozwiązania jest następujący. W systemie zdefiniowane są dwa rodzaje użytkowników. Jednym jest instytucja, która chce w danym czasie udostępniać miejsca parkingowe. Drugim osoby chcące zaparkować swoje auto. Za zamieszczenie miejsca parkingowego w serwisie instytucja nie płaci nic. Dopiero w momencie gdy jakaś osoba decyduje się zarezerwować miejsce pobierane są opłaty w następujących proporcjach:

- 75% opłaty trafia do instytucji udostępniającej miejsce;
- 25% do firmy ParkWiz (10% to umownie opłata pochodząca bezpośrednio od osoby parkującej, 15% od instytucji).

Pomimo dość sporej opłaty jaką ponosi instytucja oferująca miejsce parkingowe bo aż 25% to działalność firmy ParkWiz jest godna uznania, gdyż jest ona pomostem między udostępniającym, a parkującym.

Podsumowanie

Rezerwacja internetowa co należy podkreślić w porównaniu do opisywanych wcześniej sposobów płatności jest znacznie bardziej wygodne ze względu właśnie na sposób płacenia gdyż stwarza znacznie więcej jego możliwości. Oprócz przelewu może to być również płacenie przez telefon.

3.4 Zestawienie rozwiązań

Opisane w powyższych podrozdziałach rozwiązania zostaną teraz zestawione we wspólnej tabeli w której porównane zostaną pod kątem dodatkowych istotnych cech.

	Płacenie przez telefon	Automaty samoobsługowe	Rezerwacja internetowa
Obsługa parkingów zamkniętych	Nie	Tak	Tak, ale wymaga dodatkowej weryfikacji – zautomatyzowanej lub wykorzystując personel
Obsługa parkingów otwartych	Tak	Tak	Istnieje taka możliwość, ale rzadko stosowana
Koszt zainstalowania i eksploatacji	Infrastruktura do przechowywania danych o osobach i weryfikacji uprawnień	W przypadku parkomatów oraz parkometrów jest to ich utrzymanie	Infrastruktura do przechowywania danych o osobach i weryfikacji uprawnień, interface użytkownika
		Używanie szlabanu (jeśli jest wykorzystywany)	
Koszt dla użytkownika (oprócz tych ponoszonych za samo parkowanie)	Opłaty za wysyłanie SMS, dzwonienie lub korzystanie z internetu	Zarządca parkingu może przenieść na użytkownika koszty związane z drukowaniem biletów wjazdowych oraz samej eksploatacji automatu	Praktycznie żadne gdyż dostęp do internetu jest dzisiaj powszechny szczególnie w miastach
Obsługa	Ogranicza się do korzystania z telefonu komórkowego bądź smartfona	Automaty są dzisiaj powszechnymi urządzeniami, korzystanie z nich nie sprawia nikomu problemu – szczególnie parkometry, dodatkowo użytkownik nie musi nigdzie wstępnie deklarować korzystania	Interface przez przeglądarkę, łatwy i dostępny, wymaga wstępnej deklaracji korzystania, ale przez to umożliwia różne formy korzystania – abonament bądź jednorazowo
Obszar zastosowań	Strefy płatnego parkowania wprowadzone w kilku polskich miastach	Strefy płatnego parkowania wprowadzone w kilku polskich miastach a także parkingi prywatnych instytucji	Parkingi wymagające wstępnej deklaracji chęci korzystania tam gdzie istnieje duże natężenie ruchu jak na lotniskach, ale również imprezy masowe, możliwość depozytowania pojazdów
Udział personelu	Obsługa i administracja systemu, weryfikacja zezwolenia na postój	Uzupełnianie bileterki, wymiana gotówki, dozór automatu,	Obsługa i administracja systemu, weryfikacja uprawnień do wjazdu

	zaparkowanego pojazdu	czasem windykacja opłat	przed szlabanem
Podatność na awarie	Dane przechowywane mogą być w sposób redundantny lub dzielone na wiele serwerów, więc awaria jednego z nich nie oddziałuje negatywnie na cały system	Uszkodzony automat może zablokować możliwość korzystania z parkingu jeżeli ustawiony jest on przed wjazdem, uszkodzenie pojedynczego parkometru nie oddziałuje na pozostałe	Tylko w przypadku awarii całego systemu jednak dane również mogą być redundantne
		Szlaban w razie zepsucia może być sterowany ręcznie	
Odporność na nieuczciwość użytkowników / bezpieczeństwo danych i pieniędzy	Kradzież telefonu innego użytkownika bądź atak hakerski na serwer danych, ewentualny wyciek danych o użytkownikach	Możliwość zniszczenia automatu, kradzieży pieniędzy, użytkownik w razie takiego zdarzenia nic przez to nie traci	Atak hakerski na serwer danych, wyciek danych o użytkownikach,
Uciążliwość wprowadzania modyfikacji	Praktycznie nieodczuwalna dla użytkowników, łatwość zmian w oprogramowaniu	Wymaga niekiedy wyłączenia z użytku na dłuższy czas całego parkingu, szczególnie uciążliwe w przypadku parkometrów	Praktycznie nieodczuwalna dla użytkowników, łatwość zmian w oprogramowaniu

Tabela 3.1: Zestawienie porównawcze rozwiązań parkingowych

4. Projekt koncepcyjny systemu

4.1 Funkcja główna

Funkcją główną projektowanego systemu jest zarządzanie dostępem do płatnego parkingu. Ma on zapewniać możliwość rejestracji osób i pojazdów przez internet, a także weryfikację czy osoba ma możliwość wjazdu danym pojazdem po wpisaniu kodu wjazdowego na terminalu przy szlabanie.

4.2 Środowisko pracy

Użytkownikiem systemu będzie osoba chcąca skorzystać z możliwości parkowania na płatnym parkingu, gdzie wjazd blokowany i udostępniany jest poprzez sterowany szlaban przed którym następuje także weryfikacja.

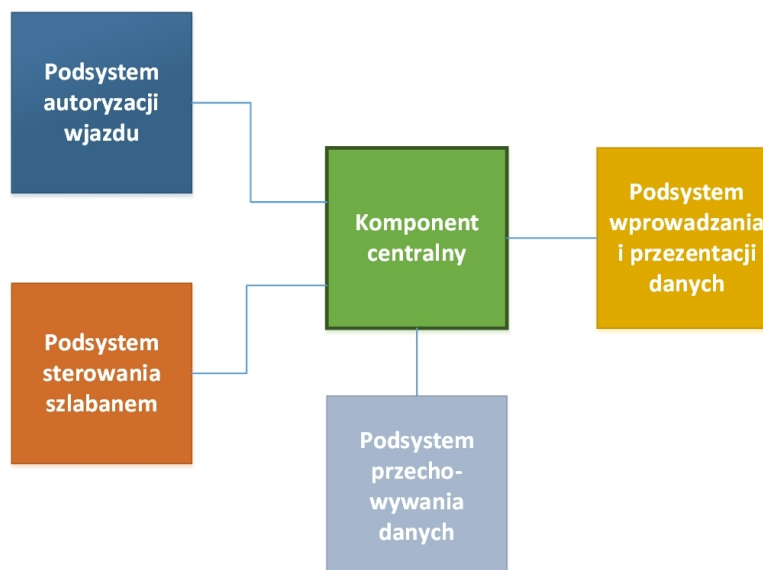
System będzie rozproszony. Część elementów znajdować będzie się przy samym szlabanie, inne wymagać będą zapewnienia infrastruktury do przechowywania danych w wersji cyfrowej.

4.3 Eksploatacja

System wymagać będzie udziału człowieka. Potrzebny będzie administrator zajmujący się kontrolowaniem zbieranych danych i raportujący je. Elementy wykonawcze wymagać będą konserwatora.

4.4 Struktura systemu

Poniższy schemat ideowy przedstawia podstawowe elementy, które wchodzić będą w skład projektowanego systemu.



Ilustracja 4.4.1: Schemat ideowy systemu

- Komponent centralny

Będzie to najważniejsza część systemu na której znajdować się będzie większość oprogramowania. Jej zadania to integracja, zarządzanie i przekazywanie danych pomiędzy innymi podsystemami.

- Podsystem przechowywania danych

Jego rolę będzie pełniła baza danych. Przechowywane będą tam informacje o użytkownikach i pojazdach. Tam też będą weryfikowane uprawnienia użytkowników do wjazdu.

- Podsystem wprowadzania i prezentacji danych

Jego rolę pełnić będzie witryna internetowa, której funkcjami będą rejestrowanie i logowanie użytkowników oraz dodawanie nowych pojazdów.

- Podsystem autoryzacji wjazdu

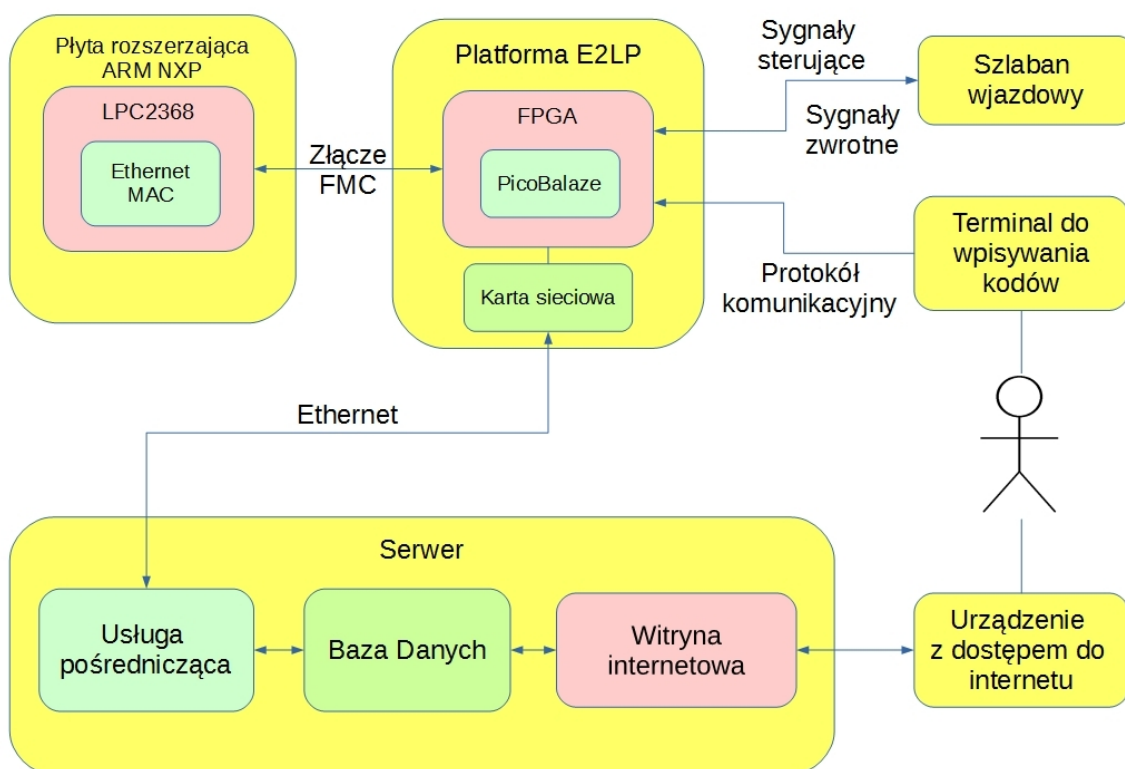
Zadaniem tego podsystemu będzie umożliwienie wprowadzania kodu wjazdowego za pomocą specjalnego terminalu i przesłanie go do podsystemu przechowywania danych w którym zostanie zweryfikowany. Terminal powinien więc posiadać klawiaturę do wpisywania kodów oraz wyświetlać informacje zwrotne.

- Podsystem sterowania szlabanem

Jego zadaniem będzie sterowanie szlabanem tj. otwieranie i zamykanie. Posiadać on będzie układ sterowania komunikujący się z nim za pomocą odpowiednich sygnałów.

4.5 Schemat blokowy funkcjonalny

Na podstawie opisanej w poprzednim rozdziale koncepcji, wymagań oraz założeń do pracy powstał poniższy schemat.



Ilustracja 4.5.1: Schemat blokowy funkcjonalny

Ważnym elementem projektowanego systemu jest platforma E2LP, której wykorzystanie było jednym z wymagań do pracy. Z zawartych na niej komponentów użyty będzie oczywiście układ FPGA Spartan6, ale także karta sieciowa do komunikacji w standardzie Ethernet, której implementacja jest również jednym z wymagań. Wyświetlacz LCD posłuży do wyświetlania informacji użytkownikom, a port podczerwieni do odbierania danych z terminalu, którego rolę pełnić będzie pilot od telewizora, co do celów demonstracyjnych w zupełności wystarczy.

Znajdujące się na płycie E2LP konektory tzw. SnapWires posłużyć mogą do sterowania binarnego szlabanem. Przykładowym szlabanem będzie szlaban automatyczny 615 BRC znajdujący się na wjeździe do parkingu przy wydziale mechatroniki. Wykorzystanie jego dokumentacji jest jednym z wymagań do pracy.

Sama karta sieciowa nie wystarczy do poprawnej komunikacji w standardzie Ethernet. Potrzebny więc będzie odpowiedni kontroler wysyłający i odbierający poprzez nią dane, a że implementacja takiego układu jest skomplikowana nawet na FPGA - wykorzystana zostanie płytka rozszerzająca platformę E2LP. Jej pełna nazwa to NXP FMC ARM Extension Board. Znajdujący się na niej mikrokontroler LPC2368 ma wbudowany kontroler EMAC, który wykorzystany zostanie do obsługi sieci. Połączenie płyty rozszerzającej z płytą główną realizowane jest poprzez złącze FMC.

Powyższe akapity opisywały część hardwarową systemu. Równie ważna jest część softwarowa. Zawierać będzie ona serwer bazy danych, ale również Http, przez który wysyłane będą strony. Tę rolę spełniać będzie zespół oprogramowania LAMP – opisany w kolejnym rozdziale[].

Istotnym elementem części softwarowej jest usługa pośrednicząca w komunikacji pomiędzy platformą E2LP a bazą danych. Napisana zostanie ona w języku Java. Technologia ta umożliwia implementację komunikacji sieciowej w standardzie TCP oraz obsługę bazy danych poprzez tak zwanego drivera.

4.6 Koncepcja działania systemu

Użytkownik chcący skorzystać z projektowanego systemu najpierw będzie musiał zarejestrować się na stronie internetowej połączonej z bazą danych. Na stronie tej będzie on miał możliwość oprócz rejestracji także logowanie do sesji. Oprócz tego będzie on mógł rejestrować swój pojazd i uzyskiwać dla niego kod wjazdowy. Użytkownik będzie mógł z poziomu strony udostępniać swoje pojazdy, a także swój abonament innym użytkownikom poprzez prosty interfejs. Tam też będzie miał możliwość przeglądania swoich pojazdów oraz tych które komuś udostępnił lub zostały udostępnione jemu.

Centralną częścią systemu będzie baza danych przechowująca wszystkie powyższe informacje. Strona internetowa pisana w różnych technologiach (PHP, JavaScript) będzie jednym z komponentów systemu wymieniających dane z bazą. Drugim równie istotnym będzie aplikacja napisana w Javie pośrednicząca pomiędzy nią, a platformą E2LP.

E2LP natomiast należeć będzie do części hardwarowej systemu. Pełnić będzie jednocześnie rolę podsystemu sterowania szlabanem oraz autoryzacji wjazdu.

Podsystem autoryzacji wjazdu w założeniu ma składać się z dwóch terminali do wpisywania kodów – jeden dla pojazdów wjeżdżających drugi dla wyjeżdżających. Umieszczone one powinny być w miejscu umożliwiającym wpisywanie kodów bez konieczności wysiadania z pojazdu. W celu demonstracyjnym ich rolę pełnić będzie pilot od telewizora z klawiaturą numeryczną. Ponieważ będzie on jeden - dodatkowe klawisze przekazywały będą informację o kierunku przejazdu. Po przekazaniu kodu do E2LP prześle ono go do aplikacji Javy, która określi na podstawie zapytań wysłanych do bazy czy użytkownik ten ma uprawnienia do wjazdu na parking (lub wyjazdu) i odeśle informację zwrotną, która wyświetlona zostanie na wyświetlaczu LCD. W informacji tej zawarty będzie powód ewentualnego brak zezwolenia na przejazd.

W przypadku gdy użytkownik chcący wjechać na parking będzie miał do tego uprawnienia podsystem sterowania szlabanem otworzy go. Sterowanie szlabanem nie będzie jednak implementowane fizycznie a powstanie jedynie projekt układu sterowania takim urządzeniem. Konkretnie będzie to szlaban automatyczny 615 BRP zainstalowany na wjeździe do parkingu wydziału Mechatroniki.

5. Integracja E2LP w sieci LAN

W poprzednim rozdziale wspomniano, że do implementacji sterownika karty sieciowej w układzie FPGA nie jest prostym zadaniem. Z tego powodu wykorzystana zostanie rozszerzająca platformę E2LP płyta z mikrokontrolerem posiadającym taki sterownik w postaci sprzętowej. W rozdziale tym zostanie opisana ta płyta oraz znajdujące się na niej komponenty. Następnie opisana będzie integracja mikrokontrolera z tą płytą z chipem karty sieciowej znajdującym się na płycie głównej E2LP. Dokonana ona zostanie w układzie FPGA Spartan6. Następnie na przykładzie aplikacji ze strony Keil przedstawiona będzie programowa konfiguracja mikrokontrolera LPC2368, aby mógł on obsługiwać działać w sieci LAN.

5.1 Płyta rozszerzająca FMC NXP ARM

Twórcy platformy E2LP zapewnili możliwość jej rozwoju m.in. poprzez dołączenie do niej rozszerzającej płyty FMC NXP ARM. Jest ona podobnie jak główna płyta E2LP zestawem ewaluacyjnym stworzonym w celach edukacyjnych. W budowanym systemie będzie ona wykorzystana jako rozszerzenie E2LP jednak z powodzeniem może być ona wykorzystana jako autonomiczny zestaw uruchomieniowy. Zgodnie z dokumentacją **od serbów** [], zasilana może być poprzez kabel USB lub z płyty głównej E2LP. Zależy to od pozycji zworki zasilającej. W skład komponentów zainstalowanych na płycie wchodzi:

- Mikrokontroler LPC2368 z procesorem ARM7TDMI
- Termometr cyfrowy DS12S20
- Niskonapięciowy wzmacniacz audio LM386
- Akcelerometr cyfrowy z interfejsem I2C
- 8 konektorów snapwire
- Potencjometr obrotowy
- Enkoder z przyciskiem
- Port CAN TJA1040
- 5 diod monochromatycznych oraz jedna RGB

Połączenie płyty rozszerzającej realizowane jest poprzez konektory FMC. Kompletna mapa pinów łączących dwie płyty z uwzględnieniem pinów Spartan 6 oraz LPC2368 znajduje się w tabeli [].

Istotnym elementem niezbędnym do programowania mikrokontrolera na płycie rozszerzającej jest programator/debuger ULINK-ME firmy Keil. Pozwala ona na programowanie oraz debugowanie programu w mikrokontrolerze co znacznie ułatwia tworzenie programów. Działa on w oparciu o protokół JTAG przez, który jest dołączany do płyty FMC ARM NXP oraz do komputera poprzez port USB.

5.2 Mikrokontroler LPC2368 z procesorem ARM7TDMI

Głównym elementem płyty rozszerzającej jest mikrokontroler LPC2368FBD100 z procesorem ARM7TDMI-S, którego maksymalne taktowanie może wynosić 72 MHz. Dostępnych jest w nim do 512 kB pamięci flash z możliwością programowania tzw. w systemie (ISP) co jest w

dzisiejszych czasach standardem, ale również w aplikacji (IAP). Oznacza to, że wewnętrzna pamięć flash może być reprogramowana dzięki czemu możliwe jest w niej przechowywanie danych które będą niezmiennie dla różnych wczytywanych do mikrokontrolera programów, ale również po wyłączeniu zasilania układu.

Kolejnym wyposażeniem omawianego układu jest spory zasób pamięci SRAM. 32 kB wykorzystywane przez procesor w celu zwiększenia jego funkcjonalności, 16 kB takiej samej pamięci dedykowane do obsługi Ethernetu, może ona być wykorzystana także przez CPU. Kolejne 8 kB tej pamięci przeznaczone jest dla kontrolera USB poprzez DMA.

Z innych interesujących funkcji/urządzeń należy wymienić m.in. to, że zapewnia do 32 przerwań dzięki Advanced Vectored Interrupt Controller (VIC), 6 przetworników ADC o rozdzielczości 10-bitowej, jeden DAC, wbudowane porty USB oraz CAN, 70 wejść-wyjść ogólnego przeznaczenia oraz zapewnienie możliwości komunikacji w różnych standardach: RS-232, I2C, SPI, CAN, USB. Wiele innych funkcji opisanych jest w podręczniku do mikrokontrolerów serii LPC23xx [um].

Najistotniejszym z punktu widzenia budowanego systemu urządzeniem jest kontroler EMAC (Ethernet Media Access Controller). Zapewnia on implementację niskopoziomowej komunikacji z kartą sieciową w warstwie łącza danych (Data-Link) wykorzystując protokół RMII. W rozdziale 11 podręcznika [1] do mikrokontrolera LPC znajduje się pełna instrukcja do EMACa. Pozwala on na komunikację w standardzie Ethernet z prędkością 10 oraz 100 Mbps w trybie full oraz half duplex zgodnie ze standardem IEEE 802.3. Dzięki zastosowaniu technologii DMA może on korzystać z pamięci RAM wykorzystywanej przez procesor. Pojawia się tu problem dostępu do pamięci współdzielonej gdyż EMAC jak i sam CPU mogą w tej samej chwili próbować odczytać lub zapisać dane zapisane w tym obszarze pamięci znajdujące się pod tym samym adresem. Projektanci mikrokontrolerów, a dokładnie firma ARM Ltd. już od dawna starała się rozwiązać ten problem czego efektem stało się wprowadzenie do owych układów szyny AHB. Jest to rozwiązanie sprzętowe bardzo podobne do znanego z programowania komputerów semafora lub też mutexu. Zasób współdzielony jest chroniony poprzez multiplexer zarządzający dostępem do niego z różnych urządzeń. AHB jest wykorzystywane także do innych celów m.in. do komunikacji z kontrolerem USB.

Wbudowane urządzenie EMAC stanowi kompletny sterownik służący do komunikacji w standardzie Ethernet. Jest to rozwiązanie sprzętowe odpowiadające programowemu sterownikowi na komputerach klasy PC. Oczywiście do komunikacji sieciowej potrzebne jest urządzenie warstwy fizycznej (PHY), przekazujące dane jako sygnał do medium, którym może być skrętka, fala radiowa w technologii Wi-Fi lub światłowód. Mowa tu o tzw. karcie sieciowej. Interface'm jakim komunikuje się EMAC z urządzeniem PHY jest RMII (zredukowany MII). Nie wymusza to jednak aby interface'm karty sieciowej było także RMII. Przyłączenie nastąpi bowiem poprzez układ FPGA na którym z łatwością można zaimplementować konwerter interface'ów. Okaze się, że jest to potrzebne gdyż karta sieciowa znajdująca się na platformie E2LP posiada zwykły interface MII.

5.3 Połączenie mikrokontrolera LPC z chipem karty sieciowej

Opisywany w poprzednim rozdziale mikrokontroler LPC2368 posiada co prawda wbudowany kontroler Ethernet jednak na płycie ARM NXP Extension board nie znajduje się żadna karta sieciowa. Znajduje się ona na głównej płycie E2LP. Dokładnie jest to Intel® LXT972M

umożliwiająca komunikację w standardzie 10/100 Mbps, w trybie Full Duplex. Połączenie urządzenia PHY z mikrokontrolerem zostanie wykonane poprzez odpowiednią konfigurację FPGA. Dokładnie będzie to implementacja interface'u MII (ang. Media Independent Interface) oraz RMII.

5.3.1 Media Independent Interface

Jest to protokół będący pomostem pomiędzy warstwą łącza danych a warstwą fizyczną modelu OSI. W projektowanym podsystemie służy do połączenia kontrolera EMAC z urządzeniem PHY. Tłumaczenie na polski jego nazwy to „interface niezależny od medium”. Niezależność ta oznacza, oznacza że może on być stosowany tam gdzie medium jest zarówno tzw. skrętka jak i światłowód. W podstawowej wersji interface dzieli się na dwie grupy sygnałów: odpowiedzialne za nadawanie i odbieranie danych oraz zarządzanie urządzeniem PHY – MDIO.

Pierwsza grupa to następujące sygnały:

Nadawanie danych (transmission):

- TX_CLK – Taktowanie (PHY -> MAC)
- TXD0 – Dane bit 0 (MAC -> PHY) (transmitowany jako pierwszy)
- TXD1 – Dane bit 1 (MAC -> PHY)
- TXD2 – Dane bit 2 (MAC -> PHY)
- TXD3 – Dane bit 3 (MAC -> PHY)
- TX_EN – Zezwolenie na transmisję (MAC -> PHY)
- TX_ER – Błąd transmisji (MAC -> PHY, sygnał opcjonalny)

Odbieranie danych (receive):

- RX_CLK – Taktowanie (PHY -> MAC)
- RXD0 – Dane bit 0 (PHY -> MAC) (odbierany jako pierwszy)
- RXD1 – Dane bit 1 (PHY -> MAC)
- RXD2 – Dane bit 2 (PHY -> MAC)
- RXD3 – Dane bit 3 (PHY -> MAC)
- RX_DV – Odebrane poprawne (data valid)(PHY -> MAC)
- RX_ER – Błąd odbierania (PHY -> MAC)
- CRS – Wykryta fala nośna (PHY -> MAC)
- COL – Wykryta kolizja (PHY -> MAC)

W standardzie 100 Mbps jaki zostanie zaimplementowany w opisywanym systemie do taktowania (sygnały RX_CLK oraz TX_CLK) wykorzystać należy wbudowany w chip karty sieciowej oscylator 25 MHz.

Druga grupa sygnałów to tzw. sygnały zarządzające (MDIO, ang Management Data Input/Output). Służą one do zarządzania, ale również diagnostyki urządzenia PHY. Wyróżniamy tu dwa sygnały:

- MDIO – dwukierunkowy sygnał danych
- MDC – taktowanie interface'u (MAC -> PHY)

Patrząc na strukturę sygnałów interface'u MDIO dostrzec można powinowactwo do opisywanego w pracy przejściowej [] interface'u SPI. Różnicą jest jednak to, że wymiana danych przebiegu tu na współdzielonej, dwukierunkowej linii MDIO, a nie na dwóch oddzielnych

jednokierunkowych.

Opisany powyżej MII jest interface'm karty sieciowej Intel® LXT972M. Kontroler EMAC mikrokontrolera LPC2368 jest jednak wyposażony w jego inną odmianę - tzw. zredukowany (RMII).

5.3.2 Reduced Media Independent Interface

Różnicą tego interface'u w stosunku do MII jest jak nazwa wskazuje redukcja sygnałów służących do wymiany danych. Zamiast 4 służących do nadawania i odbierania są tutaj po 2. Wobec czego zmianie musiała ulec także częstotliwość taktowania z 25 MHz na 50 MHz aby utrzymać prędkość transmisji 100 Mbps. Usunięte są także inne sygnały m.in. RX_DV oraz CRS zostały zredukowane do jednego sygnału, a COL został usunięty.

Zgodnie z dokumentacją UM10211 LPC23XX User manual interface RMII reprezentują następujące piny mikrokontrolera:

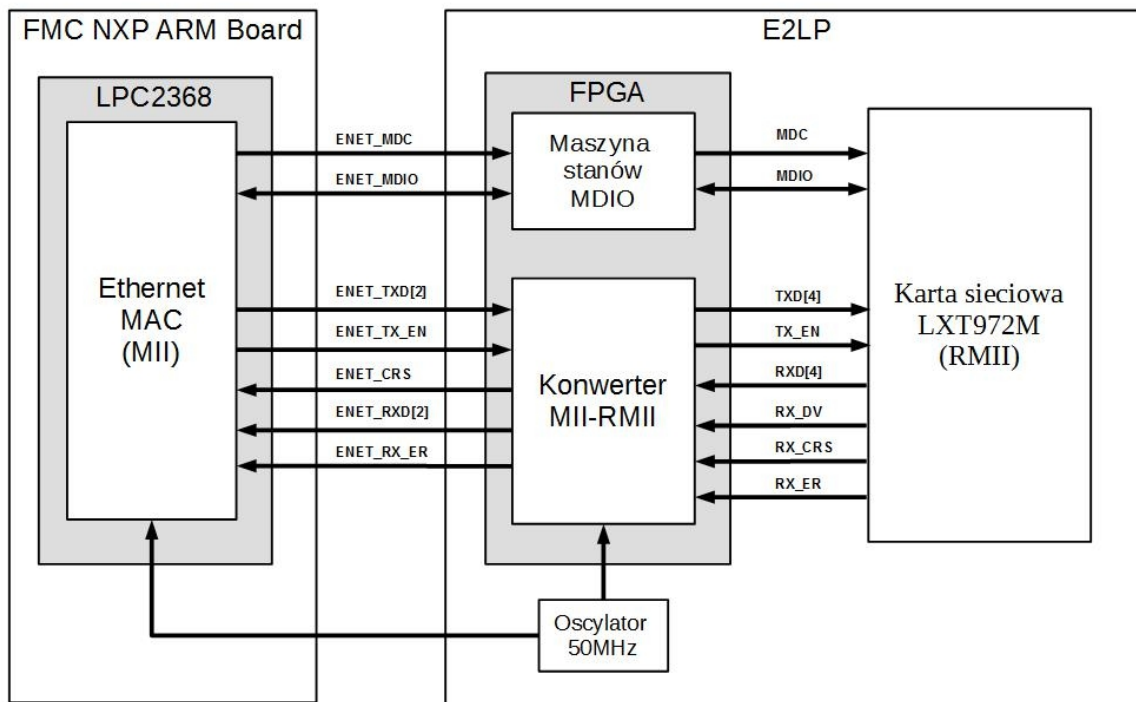
Pin	Oznaczenie	Kierunek	Opis
P1[0]	ENET_TXD0	EMAC->PHY	Dane do transmisji bit 0
P1[1]	ENET_TXD1	EMAC->PHY	Dane do transmisji bit 1
P1[4]	ENET_TX_EN	EMAC->PHY	Dane gotowe do transmisji
P1[8]	ENET_CRS	PHY->EMAC	Wykrycie fali nośnej
P1[9]	ENET_RXD0	PHY->EMAC	Odebrane dane bit 0
P1[10]	ENET_RXD1	PHY->EMAC	Odebrane dane bit 1
P1[14]	ENET_RX_ER	PHY->EMAC	Błąd danych odbieranych
P1[15]	ENET_REF_CLK/ENET_RX_CLK	PHY->EMAC	Referencyjny zegar danych odbieranych
P1[16]	ENET_MDC	EMAC->PHY	Taktowanie interface'u MDIO
P1[17]	ENET_MDIO	dwukierunkowy	Dane interface'u MDIO

Tabela 5.1: Zestawienie pinów mikrokontrolera LPC2368 odpowiedzialnych za komunikację z urządzeniem PHY

W związku z tym, że kontroler EMAC oraz chip karty sieciowej posiadają różne interface'y, konieczne będzie opracowanie odpowiedniego konwertera.

5.3.3 Konwerter RMII-MII

Budowany konwerter w układzie Spratan 6 jest przykładem ogromnej elastyczności logiki programowalnej FPGA. Opracowany on zostanie w języku VHDL. Poniżej schemat ideowy działania konwertera w układzie.



Ilustracja 5.3.1: Schemat sygnałów komunikacyjnych

Konwerter podzielony zostaje na dwa oddzielne procesy: dane odbierane oraz nadawane. Odbieranie danych zrealizuje proces poniżej:

```
process (clk) --RX
begin
    if falling_edge(clk) then
        if rx_sel = '0' then
            ENET_CR_S <= Ethernet_Lite_RX_DV;
            if Ethernet_Lite_RX_DV = '1' then
                ENET_RXD <= Ethernet_Lite_RXD(1 downto 0);
                rx_sel <= '1';
            end if;
        else
            ENET_RXD <= Ethernet_Lite_RXD(3 downto 2);
            rx_sel <= '0';
        end if;
    end if;
end process;
```

Taktowaniem tego procesu jest sygnał 'clk' podłączony do oscylatora 50 MHz znajdującego się na płycie E2LP. Ten sam oscylator znajdzie się na tzw. liście wrażliwości drugiego procesu

odpowiadającego za nadawanie danych. Najpierw zadeklarowany zostanie typ wyliczeniowy reprezentujący trzy stany transmisji:

```
type tx_state_type is (IDLE, TX0, TX1);
```

Następnie instancja tego typu:

```
signal tx_state : tx_state_type := IDLE;
```

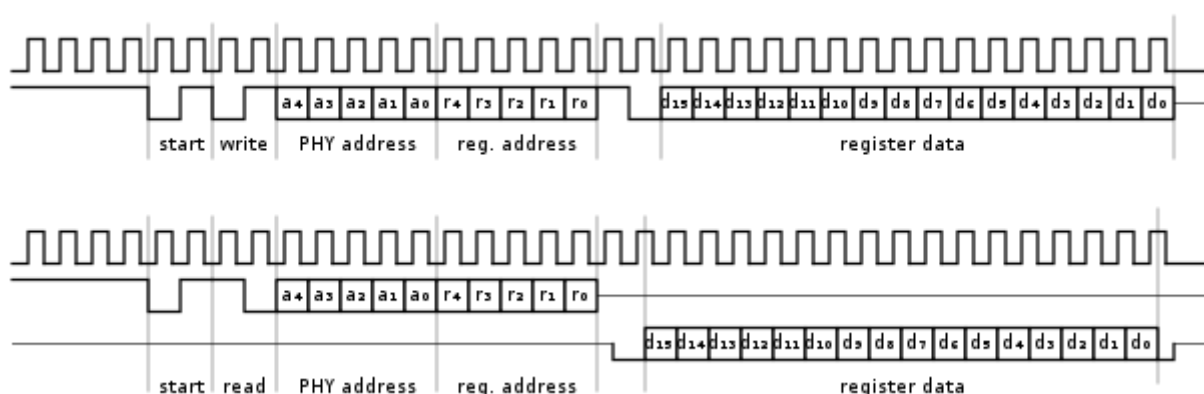
I w końcu proces realizujący konwersję:

```
process(clk)--TX
begin
    if falling_edge(clk) then
        case tx_state is
            when IDLE =>
                if ENET_TX_EN = '1' and ENET_TXD /= "00" then
                    tx0_data <= ENET_TXD;
                    tx_state <= TX1;
                end if;
            when TX0 =>
                tx0_data <= ENET_TXD;
                tx_state <= TX1;
            when TX1 =>
                Ethernet_Lite_TXD <= ENET_TXD & tx0_data;
                Ethernet_Lite_TX_EN <= ENET_TX_EN;
                tx_state <= TX0;
                if ENET_TX_EN = '0' then
                    tx_state <= IDLE;
                end if;
            end case;
        end if;
    end process;
```


5.3.4 Maszyna stanów MDIO

Bardziej kłopotliwy w implementacji niż konwerter RMII – MII okazał się wspomniany tylko do tej pory interface zarządzający MDIO. Wydawałoby się, że wystarczy bezpośrednio połączyć odpowiednie sygnały między EMAC a PHY i jest to sprawa prosta jednak w układzie FPGA nie jest możliwe bezpośrednie, fizyczne połączenie dwóch pinów poprzez wewnętrzną logikę. Należy jasno określić kierunkowość takiego sygnału. I z tego właśnie powodu w Spartanie 6 zaimplementowany musi zostać w pełni protokół MDIO jako maszyna stanów widoczna na rysunku [poprzednia strona]. Należy bowiem dokładnie określić moment transmisji w którym powinno się zmienić kierunek sygnału.

Poniżej zobrazowanie ramki protokołu MDIO pochodzący ze strony Wikipedia.



Ilustracja 5.3.2: Ramka protokołu MDIO

Pierwsza rodzaj ramki to wysyłająca polecenia do chipu, a druga odczytująca. W obu przypadkach całkowita długość ramki to 32 bity. Pierwsze dwie to sygnał startu transmisji, dwie kolejne określają rodzaj ramki, pięć następnych adres urządzenia PHY, kolejne pięć to adres rejestru, 16 ostatnich to dane. Adres urządzenia PHY ustala się poprzez odpowiednią konfigurację pinów ADDR0 i ADDR1 chipu. Według dodatku D [manualu E2LP] jest to 0x0.

Ramka odczytująca dane rejestrów chipu karty sieciowej wymaga opisanej zmiany kierunkowości. Bity 3 oraz 4 identyfikują ją i pozwalają określić moment zmiany kierunku sygnału.

5.4 Implementacja komunikacji w standardzie Ethernet na LPC2368

W rozdziale II opisano sprzętową realizację komunikacji Ethernet możliwą w mikrokontrolerach z serii LPC23xx realizowaną przez urządzenie EMAC jednak wymagana jest także programowa konfiguracja komunikacji jak również implementacja stosu protokołów wyższych warstw. Firma Keil przygotowała wiele przykładów wykorzystania urządzeń dostępnych w mikrokontrolerze LPC. Jednym z takich przykładów jest projekt [LPC2300 EASYWEB] możliwy do ściągnięcia ze strony Keil [1]. Opisana jest w nim konfiguracja odpowiednich rejestrów oraz wiele funkcji i definicji będących API do kontrolera EMAC. Opis tego API znajduje się w dokumencie **EasyWeb: Tiny TCP/IP Stack and Web Server** także możliwym do pobrania ze strony Keil. Przykład ten opisuje sposób konfiguracji oraz użycie odpowiednich funkcji API do realizacji prostego serwera http umożliwiającego połączenie jednego na raz klienta. W

projektowanej aplikacji nie będzie realizowany serwer lecz konfiguracja pozwalająca na łączenie się ze zdalnym serwerem jako klient.

5.4.1 Programowa konfiguracja rejestrów urządzenia EMAC

Podrozdział ten jest opisem działań niezbędnych do przystosowania LPC2368, ale także innych mikrokontrolerów z serii LPC23xx do możliwości komunikacji w standardzie Ethernet.

Program główny projektu [] znajduje się w pliku main.c. Program działa jak standardowy program na mikrokontroler w oparciu o pętlę while(). Najpierw jednak należy skonfigurować wszystkie rejestry, aby móc używać kontrolera EMAC do obsługi sieci używając do tego celu wspomnianych już funkcji API opisanych także w dokumencie [xxxx].

Jako pierwsza z funkcji API wywołana jest TCPLowLevelInit() zdefiniowana w pliku tcpip.c. W niej to w pierwszej kolejności ustawiany i uruchamiany jest timer odpowiadający za ewentualną retransmisję pakietu w standardzie TCP.

Zaraz po konfiguracji timera retransmisji wywołana jest Init_EMAC() z pliku EMAC.c konfigurująca sam kontroler EMAC. Pierwszym poleceniem jest włączenie jego zasilania wykorzystując służący do tego rejestr PCONP. Bardzo istotnym w tym momencie jest, aby do pinu P1[15] pełniącego funkcję ENET_REF_CLK/ENET_RX_CLK podłączone było taktowanie 50 MHz. Inaczej program zatrzyma się na miejscu podłączenia zasilania do EMAC'a poprzez rejestr PCONP.

Następnie konfigurowane są funkcje odpowiednich pinów interface'u RMII przy pomocy rejestrów PINSEL. Potem resetowane są wszystkie wewnętrzne moduły urządzenia EMAC przy modyfikacji rejestrów MAC_MAC1 oraz MAC_COMMAND. Zaraz po nich w rejestrze MAC_MCFG ustalana jest częstotliwość taktowania sygnału MDC. Z reguły częstotliwość ta wynosi 2 MHz. Maksymalna z jaką może pracować chip Intel® LXT972M to 8 MHz. W naszym przypadku będzie to 2 MHz.

Następnie inicjowane w odpowiedni sposób są rejestry EMAC co skutkuje między innymi rozpoczęciem taktowania MDC.

Wywołana zostaje funkcja write_PHY (PHY_REG_BMCR, 0x3100);

Przekazane jej argumenty zgodnie z dokumentacją chipu karty sieciowej [] skutkują konfiguracją jej logiki. Jest to więc wysłanie danych poza sam mikrokontroler. PHY_REG_BMCR zdefiniowany jest jako adres rejestru 0x00 odpowiedzialny za podstawową konfigurację PHY firmy Intel. Znaczenie poszczególnych bitów rejestru opisane jest szczegółowo w tabeli 40 rozdziału 8.0 dokumentacji LXT972M. Wysłanie wartości 0x3100 skutkuje konfiguracją prędkości transmisji na 100 Mbps, ustawienie trybu Full Duplex, aktywowanie auto-negocjacji tych parametrów z siecią oraz reset logiki chipu.

W dalszej kolejności konfigurowane są rejestry EMAC w sposób zapewniający kompatybilność z parametrami transmisji współpracującego z nim PHY poprzez rejestry: MAC_MAC2, MAC_COMMAND, MAC_IPGT – tryb full duplex oraz MAC_SUPP – prędkość 100 Mbps. Następnie funkcje rx_descr_init () oraz tx_descr_init () inicjalizują deskryptory danych odbieranych i nadawanych poprzez zaalokowanie dla nich odpowiednich bloków pamięci oraz nadanie EMAC'owi bezpośredniego dostępu do nich z pominięciem CPU (DMA). Potem w rejestrze MAC_RXFILTERCTRL ustawiane odbieranie wszystkich pakietów (Broadcast). W rejestrze MAC_INTENABLE aktywowane są przerwania EMAC'a, po czym w MAC_INTCLEAR są one resetowane. Na koniec modyfikowane są ponownie MAC_COMMAND, MAC_MAC1, aby ostatecznie uruchomić transmisję i odbieranie danych w kontrolerze EMAC.

Następnie program wraca do funkcji TCPLowLevelInit() w których inicjowane są zmienne

globalne:

```
TransmitControl = 0;           // rodzaj wysyłanej ramki
TCPFlags = 0;                  // flagi niskopoziomowe TCP
TCPStateMachine = CLOSED;     // stany połączenia TCP zgodne z RFC793
SocketStatus = 0;              // status gniazda sieciowego
```

Są to ostatnie procedury funkcji TCPLowLevelInit(). Jest w niej wszystko co potrzebne do zapewnienia sprzętowej konfiguracji LPC2368. Dalej program przechodzi do powtarzanej w nieskończoność pętli while().

5.4.2 Pętla główna programu LPC2368

Cała pętla wygląda następująco:

```
while (1)
{
    DoNetworkStuff();
    TCPActiveOpen();
    PicoBlazesThread();
}
```

Funkcja DoNetworkStuff() spełnia wiele ważnych funkcji w kontekście programowej obsługi sieci. Między innymi kontroluje statusy urządzenia EMAC, gniazda sieciowego, a także połączenia TCP. W ten sposób komunikuje się z innymi funkcjami pętli while. Oprócz tego dekoduje odebrane ramki Ethernet, obsługuje polecenia protokołów zaimplementowanych w API: ARP, ICMP, IP, TCP.

Kolejna funkcja TCPActiveOpen() zajmuje się uruchomieniem tzw. aktywnego połączenie TCP. Jest to odpowiednik klienta z architektury klient-serwer. Oznacza to że mikrokontroler będzie klientem serwera na innej maszynie zbindowanego na porcie ustalonym przez zmienną TCPRemotePort. TCPLocalPort z kolei określa lokalny port z którego wysyłane będą dane. Wykorzystane API pozwala także na użycie mikrokontrolerów LPC serii 23xx jako urządzenie zachowujące się jak serwer. Pozwala na to funkcja TCPPassiveOpen(). Ograniczeniem jej jest jednak to, że umożliwia ona aktywne połączenie z jednym tylko klientem. Przykładowy projekt [EasyWeb] demonstruje wykorzystanie tej funkcji do utworzenia mini serwera http wyświetlającego aktualną wartość z potencjometru znajdującego się na płycie NXP ARM. Projekt ten jest co prawda dedykowany dla zestawu MCB2300 jednak po niewielkich przeróbkach może z powodzeniem być użyty w E2LP. Różnicą jest w zasadzie jedynie to że MCB2300 posiada dwa potencjometry, a ARM NXP jeden, podłączony do innego przetwornika ADC. Połączenie ze skonstruowanym w ten sposób serwerem jest bardzo proste. Wystarczy wpisać adres IP mikrokontrolera w dowolnej przeglądarce i zostanie w niej wyświetlona prosta strona zawierająca wskaźnik informujący o aktualnej wartości na potencjometrze. Opisane jest to dokładniej w dokumencie [EasyWeb: Tiny TCP/IP Stack and Web Server] .

Funkcja PicoBlazesThread(), a także funkcje przez nią wywoływane napisane zostały z myślą o komunikacji pomiędzy software'owym mikrokontrolerem PicoBlaze zaimplementowanym w układzie FPGA, a programem pośredniczącym w wymianie danych z bazą danych. Są to funkcje, których logika działania została w całości opracowana przez autora pracy. O mikrokontrolerze PicoBlaze oraz jego możliwościach można przeczytać w pracach [michał zych, moja, tamtego

gością]. Program pośredniczący opisany zostanie w kolejnym rozdziale.

Jedną z funkcji wywoływanych przez `PicoBlazesThread()` jest `SendEcho()`. Jej zadaniem jest konfiguracja odbierania danych pomiędzy samym LPC, a programem z którym będzie się on docelowo łączył. Program ten to wspomniany już pośrednik w komunikacji z bazą danych. Napisany on zostanie w technologii Java w której do komunikacji po TCP wykorzystane są tzw. strumienie, które to z kolei wykorzystują serjalizację. Ta polega na wysyłaniu obiektów jako dane. W protokołach tekstowych obiektem takim będzie łańcuch znaków. W praktyce wygląda to tak, że aplikacja taka po nawiązaniu połączenia TCP wysyła najpierw określony ciąg znaków konfiguracyjnych a następnie oczekuje na otrzymanie w zwrocie od odbiorcy takiego samego ciągu znaków. Tym właśnie zajmuje się funkcja `SendEcho()`. Dodatkowo każde kolejne odebrane ciągi znaków poprzedzone są innymi danymi konfiguracyjnymi, które funkcja ta obsługuje, jak również wysyła powiadzenia odebrania danych przychodzących. O protokole wymiany danych implementowanym między innymi przez tą funkcję powiedziane zostanie przy okazji opisu aplikacji pośredniczącej **[rozdział]**.

W ciele `PicoBlazesThread()` znajdują się procedury odpowiedzialne za niskopoziomową komunikację z mikrokontrolerem `Picoblaze`. Dokładniej za odbieranie danych od niego. Komunikacja ta wygląda analogicznie do opisanej w **[moja pp]** komunikacji z driverem LCD. Wektor danych zajmuje się na pinach portu 2 (od 2 do 9). Natomiast sygnały konfiguracyjne na pinach 2.10 i 2.12. ponieważ wektor danych wykorzystywany przez tą funkcję składa się z 8 bitów dane mogą być odbierane bezpośrednio w kodzie ASCII.

Kolejną z wywoływanych funkcji jest `ProcessData()`. Jej zadaniem jest obsługa danych odebranych przez mikrokontroler LPC, znajdujących się w buforze danych wejściowych `_RxTCPBuffer` i przekazanie ich do `PicoBlaze'a` poprzez kolejną funkcję – `SendToPico(int)`. Ta z kolei zajmuje się podobnie jak procedury opisane w poprzednim akapicie niskopoziomową komunikacją poprzez piny FMC. Korzysta ona z wektora danych znajdującego się na pinach 0.16, 1.27, 1.28, 1.29 [tabela z pinami] oraz dwóch bitów konfiguracyjnych na pinach 0.11 i 0.15.

Funkcja `SendFromPico()` także wywoływana z funkcji `PicoBlazesThread()` tak jak jej nazwa wskazuje wysyła dane pochodzące z `PicoBlaze'a` poprzez TCP. Samo wysyłanie przez nią danych polega na skopiowaniu ich do bufora wyjściowych `TCP_TX_BUF[]`, wywołaniu funkcji z API `[] TCPTransmitTxBuffer()`, a w dalszej kolejności wyzerowaniu odpowiednich flag.

5.5 Podsumowanie

6. Internetowy system rezerwacji

6.1 Wykorzystane oprogramowanie

Biorąc pod uwagę powyższe założenia, a przede wszystkim to ostatnie najbardziej sprawdzonym rozwiązaniem jest LAMP – Linux, Apache, MySQL, PHP.

Powyższy akronim określa zbiór oprogramowania umożliwiający tworzenie stron internetowych połączonych z bazą danych. Wszystkie komponenty tego zestawu są udostępnione na zasadzie otwartego oprogramowania, co oznacza że deweloperzy mają możliwość dostępu do ich kodów źródłowych. Umożliwia to modyfikację takiego oprogramowania w dowolny sposób. Żadna ze składowych LAMP nie był projektowany z myślą o kompatybilności z którąkolwiek z pozostałych. Mimo to ten zestaw stał się bardzo popularny wśród twórców stron internetowych, głównie ze względu na elastyczność i dostępność. Poniżej zostaną scharakteryzowane poszczególne komponenty:

- Serwer Http Apache. Najpopularniejszy z używanych obecnie na całym świecie. Jego głównymi zaletami jest bezpieczeństwo, ale także praca wielowątkowa oraz skalowalność
- System zarządzania relacyjnymi bazami danych MySql. Jego zaletami oprócz dostępności jest obsługiwanie wielu funkcji wykorzystywanych w bazach danych takich jak wyzwalacze, transakcje, widoki, procedury składowane oraz wiele innych. Istnieją także narzędzia służące obsłudze bazy danych MySql takie jak MySql-Workbench. Rolę bazy danych w LAMP może pełnić także zbliżone oprogramowanie o nazwie MarinaDB.
- Język skryptowy PHP. Jest to obiektowy język zaprojektowany do generowania akcji wykonywanych po stronie serwera. Umożliwia implementację ważnych z punktu widzenia użytkownika funkcji jak ciasteczka oraz sesje. Umożliwia tworzenie własnych funkcji rozszerzających jego możliwości. Ma m.in. wbudowaną obsługę MySql. Można go mieszać z kodem html, a jego ciągi znakowe mogą być bezpośrednio wyświetlane użytkownikowi. W zestawie LAMP język PHP może być zastąpiony przez Pearl lub Python.
- System operacyjny Linux. Rolę tą może pełnić także Windows. Wtedy akronom zmienia się w WAMP. Programiści preferują jednak system Linux z uwagi na to, że także jest dostępny bezpłatnie. Mimo, iż instalacja oprogramowania na Linuxie bywa bardziej skomplikowane niż na Windowsie to przy pomocy narzędzi takich jak taskel istnieje możliwość instalacji komponentów AMP znacznie szybciej niż na produkcie Microsoftu. Narzędzie to służy do instalacji kolekcji pakietów złożonego oprogramowania.

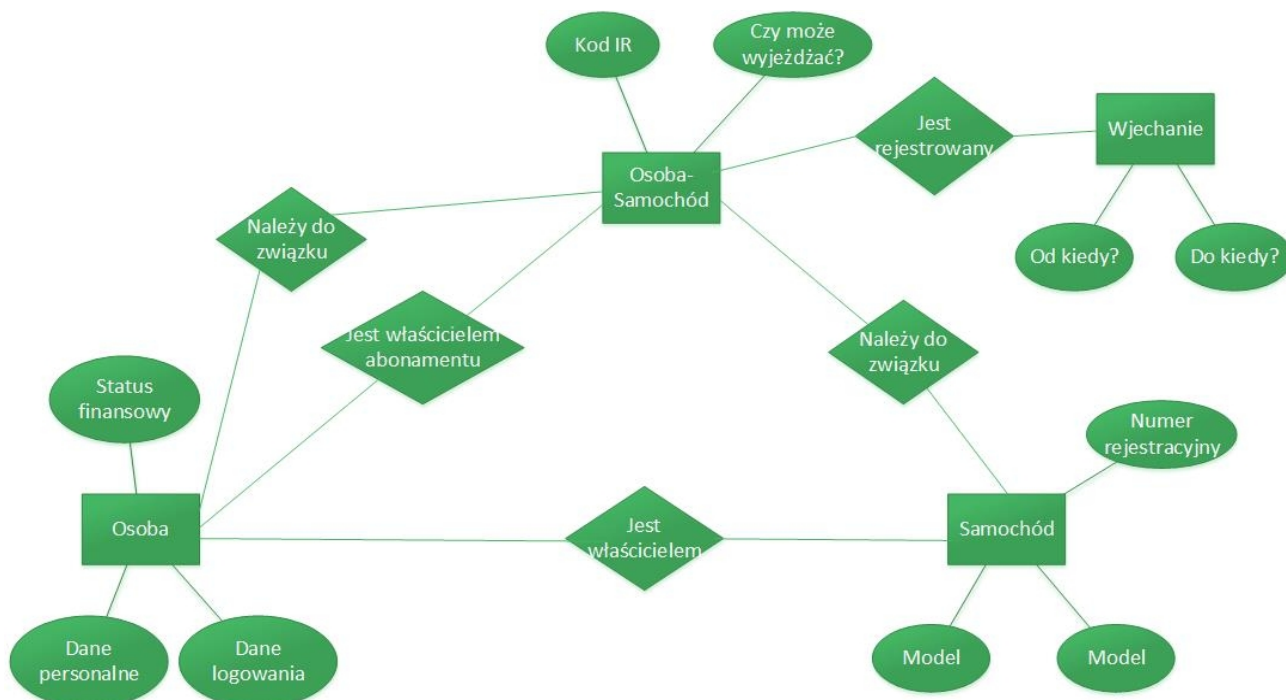
Gdy wszystkie komponenty LAMP zostaną już zainstalowane można bez problemu udostępniać utworzone strony poprzez serwer Http w sieci. Na systemie Linux wystarczy jedynie skopiować je do katalogu /etc/var/www/html. Domyślnie znajduje się tam plik index.html który zawiera informacje o serwerze Apache. Można dodawać w nim również swoje katalogi. Aby wyświetlić stronę poprzez serwer w sieci lokalnej należy w przeglądarce wpisać adres IP maszyny z serwerem Apache. Dla lokalnego komputera wystarczy wpisać 127.0.0.1. Jeżeli strona znajduje się w katalogu należy wpisać po adresie IP znak „/”, a po nim nazwę katalogu.

6.2 Projekt bazy danych

Centralnym elementem projektowanego systemu będzie relacyjna baza danych. Zanim zostanie ona zaprojektowana należy przyjąć pewne założenia odnośnie jej działania.

- Baza powinna pozwalać na jednoznaczną identyfikację osób oraz samochodów,
- Rejestrować każde wjechanie oraz wyjechanie na parking,
- Definiować jednoznacznie encję będącą związkiem między samochodem oraz osobą i do każdego takiego związku przypisywać indywidualny kod wjazdowy,
- Dla każdego związku określać tak zwanego właściciela abonamentu co jest realizacją założenia funkcjonalności projektowanego systemu – osoba może udostępniać innej osobie swój abonament oraz swój pojazd.

W oparciu o powyższe założenia opracowany został model związków encji przedstawiony poniżej.

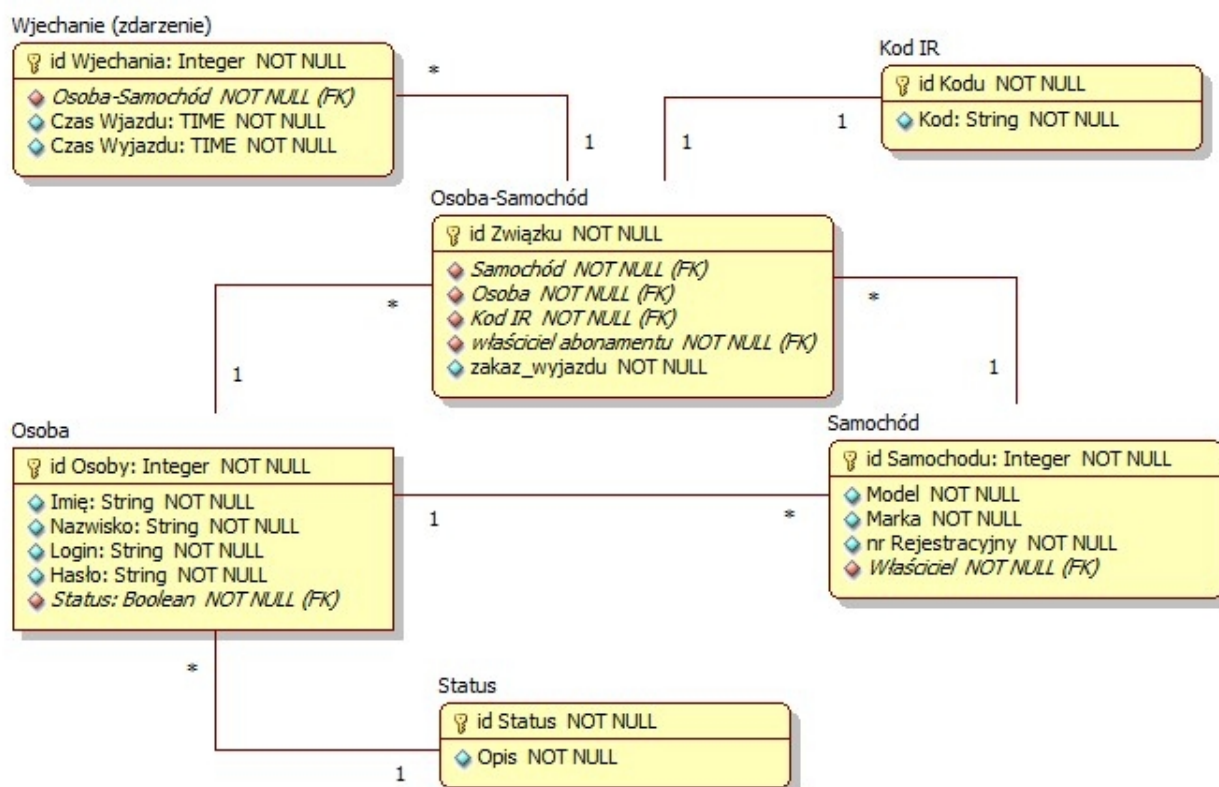


Ilustracja 6.2.1: Model związków encji projektowanej bazy danych

Model ten uwzględnia atrybuty jakie posiadać będą encje. Osoba posiadać będzie dane personalne jakimi będą imię i nazwisko oraz dane logowania (ang. *credentials*). Istotny będzie także status określający na jakich zasadach osoba wjeżdżać będzie (lub nie) na parking. Statusy opisane zostaną w dalszym akapicie. Samochód opisany będzie przede wszystkim nr rejestracyjnym oraz modelem i marką.

W związku z założeniem mówiącym, że osoba będzie mogła wjeżdżać więcej niż jednym pojazdem oraz jednym pojazdem wjeżdżać więcej niż jedna osoba – między tymi dwoma encjami istnieć będzie związek wiele do wielu. Z tego względu powstać musi pośrednia encja reprezentująca związek między samochodem i osobą. Atrybutami tego związku będą przede wszystkim indywidualny kod wjazdowy, właściciel abonamentu oraz flaga określająca czy dana osoba może wyjeżdżać danym pojazdem z parkingu. Właściciel abonamentu jest atrybutem, którego pojawienie się wynikało z założenia o możliwości udostępniania abonamentu między użytkownikami.

Encja 'Wjechanie' będzie logiem archiwizującym każde zdarzenie wjechania oraz wyjechania z parkingu z czasami jako atrybutami.



Ilustracja 6.2.2: Model fizyczny bazy

Następnie w oparciu o powyższy model związków encji opracowany został model fizyczny bazy danych przedstawiony poniżej.

Powyższy model bazy został zaimplementowany na serwerze MySQL. Narzędzia graficzne wspomagające tworzenie relacyjnych baz danych takie jak mysqlmyadmin oraz MySQL Workbench pozwalają w wygodny sposób zdefiniować tabele bazy.

Zdefiniowane zostały ograniczenia unikalne. W tabeli 'Samochód' jest nią nr rejestracyjny pojazdu. W tabeli 'Osoba' Login użytkownika. W tabeli 'Kod_IR' jest to Kod. Najbardziej złożonym, bo składające się aż z trzech kolumn ograniczenie znajduje się w tabeli 'Osoba-Samochód'. Kolumnami tymi są Osoba, Samochód i właściciel abonamentu. To ostatnie ograniczenie powstało w wyniku opisanych w dalszej części pracy działań związanych z tworzeniem witryny internetowej. Użytkownicy będą mogli bowiem udostępniać swoje samochody, a także udostępniać swój abonament innym użytkownikom na ich samochody. Ponieważ przeglądając stronę mogą oni wielokrotnie próbować wykonać tą samą czynność, ograniczenie to zapobiega takim zdarzeniom.

Tabela 'Status' jest istotna z punktu widzenia przechowywania informacji o uprawnieniach jak też uregulowaniu finansowym użytkowników. Jedynym atrybutem tej tabeli jest opis słowny statusu danej osoby. Ustalono następujące statusy użytkownika opisane w tej tabeli:

- Osoba uprzywilejowana – taki status przyznawa osobą lub pojazdom specjalnym, które nie mają obowiązku uiszczania opłat za wjazd. Przykładem mogą być tutaj pracownicy techniczni uczelni, którzy w godzinach pracy mają nieograniczony wstęp na teren parkingu używanym przez nich pojazdem.
- Abonament terminowy – oznacza to, że użytkownik ma zagwarantowane prawo do korzystania z parkingu po uiszczeniu odpowiedniej opłaty przez z góry określony czas np. 30 dni.

- Wjazd jednorazowy – osobie przyzwała jednorazowa możliwość wjazdu na parking, po wyjechaniu z niego status jest zmieniany na 'Zakaz wjazdu'
- Zakaz wjazdu – oznacza to, że zarejestrowana osoba nie może korzystać z parkingu z powodu nie opłacania tej usługi i nie jest osobą uprzywilejowaną.

6.3 Projekt witryny internetowej

Dla osób chcących korzystać z parkingu najważniejszym elementem systemu będzie witryna internetowa. Powinna ona zapewniać opisane w założeniach funkcjonalności. Będzie ona musiała współpracować z bazą danych dlatego więc, wymagane będzie korzystanie z opisanego w **poprzednim rozdziale** języka PHP. Skrypty tego języka wykonywane są po stronie serwera. Ma on w sobie wbudowaną obsługę baz danych MySQL.

Po stronie klienta – przeglądarki znajdować będą się elementy z których bezpośrednio korzystać będzie użytkownik. Wykorzystane będzie na pewno język HTML, który tworzy podstawową strukturę strony. Jednak jego funkcjonalność nie zapewnia obsługi interakcji niezbędnych do spełnienia wszystkich wymagań. HTML pozwala natomiast deklarowanie skryptów, które interpretować może przeglądarka. Takie skrypt pisać można w języku JavaScript.

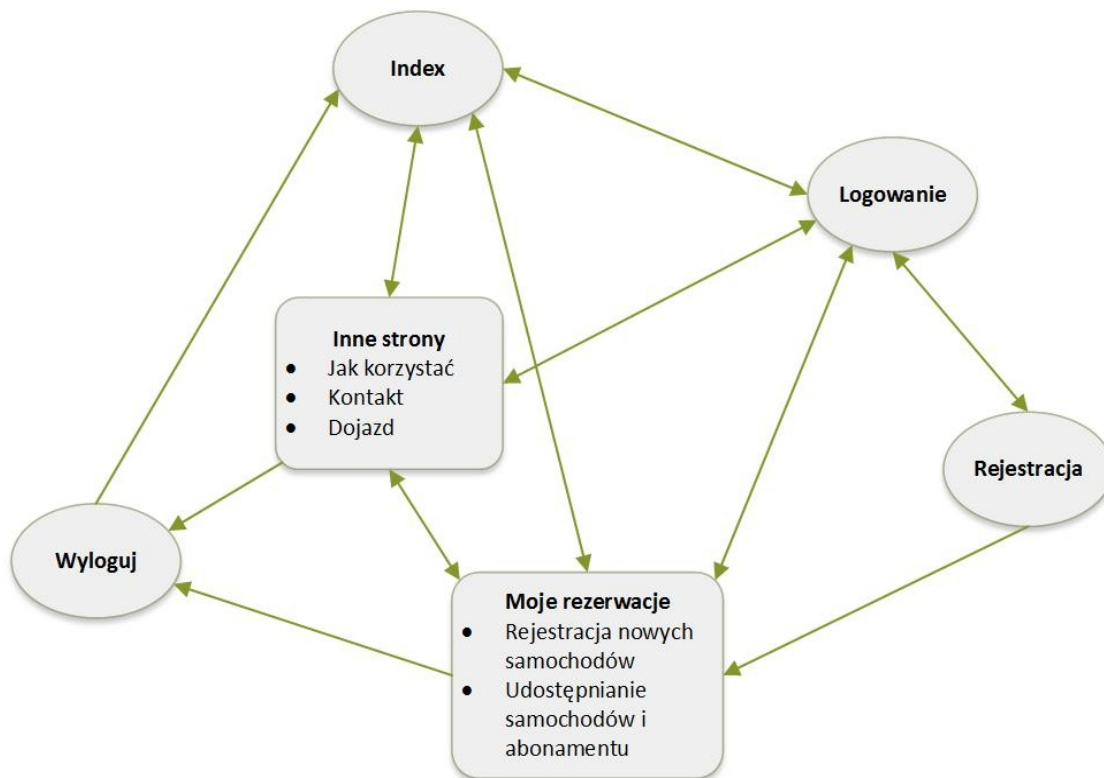
JavaScript jest skryptowym językiem programowania co oznacza, że nie jest on kompilowany do postaci kodu wynikowego, a interpretowany przez inny program. W tym przypadku jest to przeglądarka internetowa. Przykłady mi są tu Google Chrome, Mozilla Firefox, Internet Explorer czy też Safari. Najnowsze wersje każdej z tych przeglądarek mają wbudowany interpreter JavaScript.

Celem stosowania tego języka jest obsługa zdarzeń pochodzących od użytkownika takich jak reagowanie na kliknięcia na stronie, ale również wywoływanie innych skryptów, przechwytywanie wysyłanych przez nie danych np. w postaci XML i prezentowanie ich na stronie. Innymi funkcjami tego języka jest np. zapewnianie różnych usług takich jak wykorzystanie API Google lub OpenStreetMap do wyświetlania mapy terenu z naniesionymi elementami.

Istotną zaletą korzystania z JavaScript jest możliwość asynchronicznego przeładowywania dynamicznych elementów na stronie co w połączeniu z wykorzystaniem dokumentów XML określa technikę AJAX (ang. Asynchronous JavaScript And XML). Będzie to niezmiernie przydatne podczas tworzenia interfejsu użytkownika.

6.3.1 Schemat przejść pomiędzy stronami

Poniżej schemat przejść pomiędzy kolejnymi stronami witryny w formie diagramu.



Ilustracja 6.3.1: Schemat przejść pomiędzy ekranami

Oprócz podstawowych elementów wynikających z założeń w witrynie znajdują się także inne strony takie jak opis korzystania z usługi, informacje kontaktowe oraz wskazówki dojazdu. Na powyższym diagramie przedstawione są one jako jeden element, aby nie zmniejszać jego czytelności. Strona „Moje rezerwacje” zawierać będzie większość funkcjonalności o których mowa w założeniach. Będzie to prostsze niż tworzenie oddzielnych stron dla każdej funkcji.

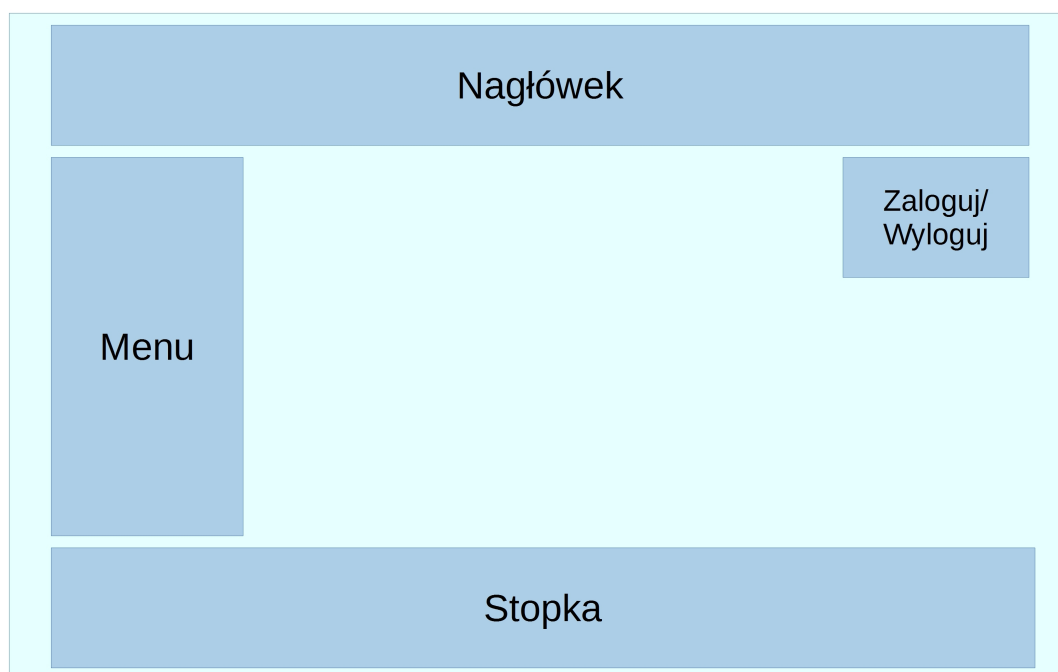
Ponieważ jak wynika z diagramu praktycznie z każdego miejsca witryny ma istnieć możliwość przejścia do każdego innego. Z tego powodu wygodnie będzie zastosować stałe elementy na wszystkich stronach. Opracować należy, więc szablon strony.

6.3.2 Szablon pojedynczej strony

Stalymi elementami na stronach będą:

- Nagłówek
- Menu
- Stopka
- Przycisk logowanie i wylogowywania

Poniższy rysunek przedstawia rozmieszczenie tych elementów na szablonie strony.



Ilustracja 6.3.2: Szablon pojedynczej strony witryny

W pierwszej kolejności utworzona zostanie strona, której kopia zapasowa będzie punktem startowym do tworzenia innych stron. Nagłówek oraz Menu będą elementami zawierającymi hiperłącza do innych stron. Stopka zawierać będzie najprawdopodobniej jedynie stały tekst. Wykonane zostaną one w postaci skryptów JavaScript w oddzielnych plikach. Będzie to rozwiązanie zwiększające skalowalność witryny. Dodając kolejne strony nie będzie potrzeby dodawania w menu na każdej stronie nowego hiperłącza, gdyż wystarczy wtedy jedynie aktualizować jeden plik z którego korzystać będą wszystkie strony.

Aby wykorzystywać kod JavaScript napisany w zewnętrznych plikach należy w kodzie HTML zadeklarować nowy skrypt, a wewnątrz niego typ oraz źródło, czyli ścieżkę do pliku.

```
<script type="text/javascript" src="menu.js"></script>
```

Skrypty te można rozmieszczać w tabelach, które są podstawowym elementem języka HTML. Do zdefiniowania stałych elementów na stronach utworzone zostały kolejno skrypty: logo.js, menu.js, stopka.js.

Przycisk Zaloguj/Wyloguj będzie generowany dynamicznie z serwera w zależności od tego czy użytkownik jest zalogowany czy nie. Zadanie to spełniać będzie kod w języku PHP. Aby możliwe było jego wykonywanie na serwerze pliki stron muszą być zapisane z rozszerzeniem .php. Kod taki nie jest widoczny dla przeglądarki, gdyż jak już było to powiedziane PHP wykonuje się na serwerze, a do klienta trafia jedynie jego echo, którym może bez przeszkód być kod HTML. Stąd korzystając z PHP mówi się o dynamicznym generowaniu stron WWW.

6.4 Implementacja witryny

Wszystkie pliki z których składa się witryna znajdują się w katalogu /etc/var/www/html/Rezerwacje. Są nimi skrypty napisane w HTML, CSS, JavaScript oraz PHP. Nazwy oraz funkcje tych plików opisane są w tabeli poniżej.

Nazwa pliku	Funkcja
Zaloguj.php	strona logowania
Zarezerwuj.php	strona zarządzania samochodami
Jak.php	strona zawierająca opis korzystania z systemu
NewUser.php	strona rejestracji nowego użytkownika
Dojazd.php	strona z informacjami o dojeździe na wydział
Kontakt.php	strona z informacjami kontaktowymi
index.php	strona powitalna witryny
logo.js	skrypt budujący logo szablonu stron
menu.js	skrypt budujący menu szablonu stron
stopka.js	skrypt budujący stopkę szablonu stron
myStyle.css	plik definiujący style stron
addCar.php	skrypt dodający do bazy samochód
changeCode.php	skrypt zmieniający kod wjazdowy
choosePerson.php	skrypt wskazujący osobę, której udostępni się abonament
get_cars.php	skrypt pobierający z bazy listę samochodów użytkownika
get_shared_Abo_cars.php	skrypt pobierający z bazy listę samochodów użytkownika na które udostępnił swój abonament
get_shared_cars.php	skrypt pobierający z bazy listę udostępnionych samochodów
logOut.php	skrypt wylogowujący użytkownika z serwisu
phpsqlajax_dbinfo.php	skrypt zawierający informacje związane z bazą
searchPeople.php	skrypt wyszukujący w bazie innych użytkowników
shareCar.php	skrypt udostępniający samochód użytkownika

Tabela 6.1: Zestawienie plików zaimplementowanych na serwerze będących częścią witryny

Zaraz po wejściu na witrynę tj. wpisaniu w przeglądarce adresu *adres_serwera/Rezerwacje* użytkownik zostanie przekierowany na stronę *index.php*. Jest ona stroną domową witryny. Gdy użytkownik nie jest zalogowany wyświetlona zostanie informacja z prośbą o zalogowanie się lub zarejestrowanie, a wraz z nimi odpowiednie hiperłącza.

6.4.1 Logowanie

Poniższa ilustracja prezentuje wygląd strony logowania. Głównym elementem tej strony jest formularz w którym użytkownik może wpisać swoje dane logowania.

Rezerwacje

Menu serwisu

[Moje rezerwacje](#)

[Jak działa rezerwacja](#)

[Wskazówki dojazdu](#)

[Informacje kontaktowe](#)

Username:

Password:

Login

Jeżeli nie posiadasz konta w serwisie [Zarejestruj się](#)

Warszawa 2015

Ilustracja 6.4.1: Strona logowania witryny

Plik implementujący tą stronę zawiera kod HTML, który jest wyświetlany w przeglądarce użytkownikowi oraz skrypt PHP zawierający sekwencję procedur obsługujących bazę danych w celu weryfikacji danych logowania użytkownika.

Po wpisaniu loginu i hasła użytkownik klika na przycisk „Login” co powoduje uruchomienie skryptu PHP znajdującego się w tym pliku. Dzieje się tak gdyż w części HTML zapisana była odpowiednia deklaracja w formularzu wyświetlanym w przeglądarce.

```
<FORM NAME = "form1" METHOD = "POST" ACTION = "Zaloguj.php">
```

Dodatkowo elementom wejściowym w których wpisywane będą dane logowania przez użytkownika nadawane są odpowiednie identyfikujące ich nazwy.

```
<INPUT TYPE = 'TEXT' Name = 'username' maxlength="20">
```

```
<INPUT TYPE = 'TEXT' Name = 'password' maxlength="16">
```

Deklaracja ta oznacza wywołanie metody „POST” na wskazanej stronie, a właściwie pliku, którym jest tutaj ten sam z którego pochodzi wyświetlona strona. Można, więc powiedzieć, że po kliknięciu przycisku „Login” strona wywołuje samą siebie.

Metoda „POST” pozwala na przekazanie danych do strony na serwerze. Skrypt PHP w pliku „Zaloguj.php” rozpoznaje czy wywołana została na nim metoda „POST” w wyrażeniu warunkowym `if ($_SERVER['REQUEST_METHOD'] == 'POST')` i odczytując dane wpisane w formularzu rozpoczyna ich weryfikację z bazą danych. Najpierw podejmowana jest próba połączenia z odpowiednią bazą na serwerze korzystając ze zmiennych z pliku `phpsqlajax_dbinfo.php`. Zmienne te to adres serwera, nazwa bazy, nazwa i hasło użytkownika, który loguje się do bazy. Ponieważ skrypt wykonuje się na serwerze użytkownikiem tym jest root. Pozostałe zmienne to połączenie z serwerem `$db_handle` oraz referencja do pożądanej bazy `$db_found`. Generowane są one poprzez odpowiednie wywołania poniższych funkcji.

```
$db_handle = mysql_connect($server, $user_name, $pass_word);  
$db_found = mysql_select_db($database, $db_handle);
```

Skrypt następnie pobiera dane logowania użytkownika poprzez argumenty metody „POST” pochodzące z pól formularza na tej stronie.

```
$uname = $_POST['username'];  
$pword = $_POST['password'];
```

Dane te są weryfikowane przez funkcję `quote_smart`, co jest wymagane w celu uniknięcia przekazania do zapytania SQL nie poprawnych znaków. Znaki te mogą być błędnie interpretowane przez silnik bazy lub też może to być atak na bazę tak zwane SQL injection.

```
$uname = quote_smart($uname, $db_handle);  
$pword = quote_smart($pword, $db_handle);
```

Następnie korzystając z powyższych zmiennych budowane jest wspomniane już zapytanie SQL.

```
$SQL = "SELECT * FROM Rejestracje.Osoba WHERE Login = $uname AND Haslo = $pword";
```

Jest ono następnie wykonywane na bazie, a jej wynik przekazywany jest do zmiennej `$result` wykorzystując poniższą funkcję.

```
$result = mysql_query($SQL);
```

Z wyniku zapytania można następnie wyłuskać liczbę zwróconych wierszy za pomocą kolejnej wbudowanej w PHP funkcji.

```
$num_rows = mysql_num_rows($result);
```

Jeżeli liczba zwróconych wierszy jest większa niż zero to oznacza to, że dane logowania są poprawne i użytkownik może zostać zalogowany. Wykonywane są poniższe procedury.

```
session_start(); // utworzenie nowej sesji  
$_SESSION['login'] = "1"; // zapisanie w zmiennych sesji parametrów które  
$_SESSION['user'] = $uname; // wykorzystywane będą przez inne skrypty
```

W celu wyłuskania danych z zapytania, które potrzebne będą do zainicjowania innych parametrów sesji wywołać należy poniższą funkcję pobierającą jeden wiersz z otrzymanego wyniku zapytania przekazując jej ten wynik, tak jak poniżej.

```
$row = mysql_fetch_array($result);
```

Następnie ze zmiennej reprezentującej ten wiersz w prosty sposób pobrać można wartość kolumny zawierającej interesujące nas dane. Jest ona bowiem kontenerem asocjacyjnym w którym przechowywane są pary wartości: nazwa kolumny będąca kluczem oraz odpowiadającą jej wartość pola.

```
$_SESSION['userId'] = $row['idOsoba'];  
$_SESSION['status'] = $row['status'];
```

Po zapisaniu potrzebnych parametrów sesji użytkownik przenoszony zostaje na stronę rezerwacji wjazdów wykorzystując poniższą procedurę.

```
header ("Location: Zarezerwuj.php");
```

W wypadku braku rezultatu zapytania wykonywane jest kolejne zapytanie określające czy wpisany login w ogóle istnieje w bazie. Oczywiście bez względu na wynik tego dodatkowego zapytania użytkownik nie zostanie zalogowany i wyświetlona zostaje mu stosowna informacja

stwierdzająca przyczynę niepowodzenia logowani pod formularzem, czerwoną czcionką.

Ostatnią procedurą wykonywaną przez ten skrypt jest zamknięcie połączenia z bazą.

```
mysql_close($db_handle);
```

6.4.2 Rejestracja

Jeżeli użytkownik nie posiada konta w serwisie będzie musiał się zarejestrować. Strona zawierająca odpowiedni do tego formularz oraz skrypt w języku PHP znajduje się w pliku NewUser.php. Formularz przedstawiony jest poniżej.

Formularz rejestracji nowego użytkownika. Ma on sześć pól tekstowych: Imię, Nazwisko, Telefon, Login, Hasło i Potwierdź hasło. Każde pole ma gwiazdkę obok siebie, co sugeruje, że jest wymagane. Na dole formularza znajduje się przycisk o napisie "Register".

Ilustracja 6.4.2: Formularz rejestracji nowego użytkownika

Działanie skryptu PHP na tej stronie jest analogiczne jak na stronie Zaloguj.php. Różnicą jest to, że zamiast zapytania wybierającego wykonywane jest zapytanie definiujące dane – dodające nowego użytkownika. Jeżeli uda się zarejestrować nową osobę to następuje przekierowanie na stronę zarządzania pojazdami – opisaną w kolejnym podrozdziale, a jeżeli nie to wyświetlona zostanie informacja o przyczynie niepowodzenia.

6.4.3 Zarządzanie pojazdami

Plik Zarezerwuj.php stanowi najbardziej złożony, ale przez to najbardziej istotny element całej witryny. Generowana przez niego strona realizuje bowiem większość z zakładanej funkcjonalności witryny, którymi są rejestrowanie nowych pojazdów, generowanie dla nich kodów przejazdowych oraz udostępnianie pojazdów i abonamentu.

Jest to znacznie bardziej złożony zbiór linii kodu niż opisywana w poprzednim podrozdziale Zaloguj.php. Użytkownik może wejść na tą stronę niezależnie od tego czy jest zalogowany czy nie. Skrypt PHP sprawdza to w pierwszej kolejności w poniższym wyrażeniu warunkowym sprawdzając parametry sesji co wymaga oczywiście uprzedniej jej aktywacji.

```
session_start();  
if (!(isset($_SESSION['login']) && $_SESSION['login'] != ''))
```

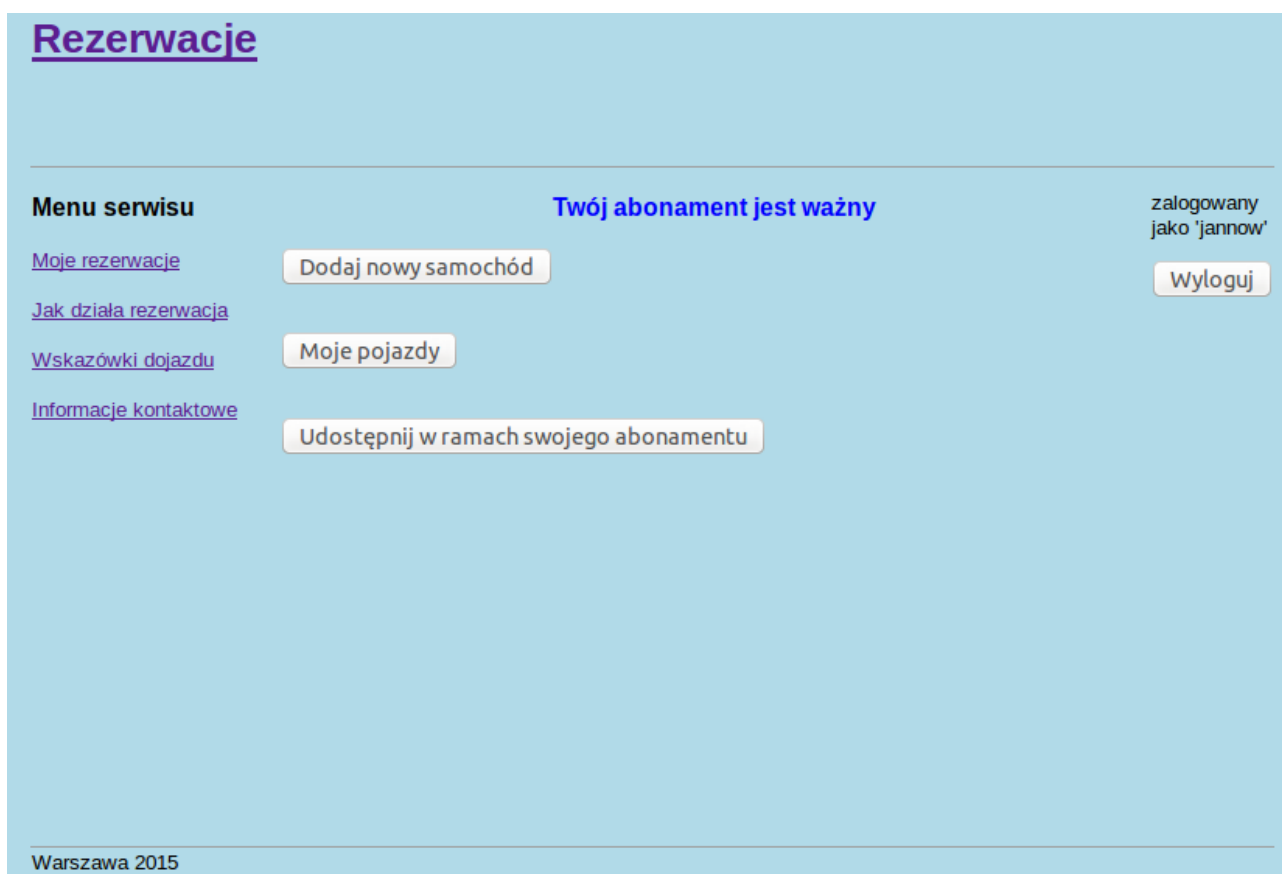
Pozytywny wynik tego wyrażenia oznacza że użytkownik nie jest zalogowany i inicjowane są zmienne, które odczytywane są między innymi przez inny skrypt PHP generujący przycisk z napisem „Zaloguj”, nazwie strony na jaką przeniesiony zostanie użytkownik po kliknięciu na ten przycisk oraz informację o tym, że jest on niezalogowany.

W treści strony wyświetlane są wskazówki co zrobić, aby móc korzystać z oferowanych usług. Są tam hiperłącza do stron pozwalających na zalogowanie. Przedstawia to rysunek poniżej.



Ilustracja 6.4.3: Strona zarządzania pojazdami, gdy użytkownik nie jest zalogowany

Jeżeli jednak użytkownik jest zalogowany strona ta wygląda zupełnie inaczej i udostępnia użytkownikowi pożądane przez niego usługi. Generowany dynamicznie przycisk zmienia nazwę na „Wyloguj”, a co za tym idzie łatwy do przewidzenia rezultat jego kliknięcia. Widniejący nad nim napis wyświetla nazwę zalogowanego użytkownika. W górnej części wygenerowanej strony wyświetlana jest informacja o statusie w bazie użytkownika.



Ilustracja 6.4.4: Strona zarządzania pojazdami po zalogowaniu

W zależności od statusu użytkownika – uregulowania finansowego - na stronie tej wyświetlane mogą być różne informacje. Powyższy zrzut ekranu przedstawia sytuację, gdy osoba ma opłacony terminowy abonament. W przypadku abonamentu jednorazowego wyświetlona zostanie informacja: „Masz prawo do jednorazowego wjazdu”. Osoba uprzywilejowana zobaczy napis: „Osoba uprzywilejowana”, a w razie braku zezwolenia na wjazd wyświetlone zostanie: „Nie masz uprawnień do wjazdu na teren parkingu. Bardziej szczegółowy opis znaczenia każdego ze statusów opisany został w **poprzednim podrozdziale**.

Zaraz po załadowaniu strony użytkownik pod opisaną w poprzednim akapicie etykietą zobaczy trzy przyciski. Pierwszy z nich służy do dodawania nowych pojazdów. Drugi do przeglądania pojazdów. Trzeci do udostępniania swojego abonamentu innym użytkownikom. Wszystkie te przyciski jak i etykieta statusu oraz inne obiekty znajdujące się w tej części strony takie jak tabele i inne przyciski pojawiające się po kliknięciu na któryś z tych przycisków wygenerowane zostały dynamicznie jako echo skryptu PHP.

Po kliknięciu na pierwszy z nich wyświetlony zostanie formularz jak na zrzucie poniżej.

Ilustracja 6.4.5: Formularz dodawania nowego pojazdu

Po wpisaniu odpowiedniego numeru rejestracyjnego, marki oraz modelu pojazdu i kliknięciu przycisku „Zatwierdź” auto zostanie dodane do bazy, a informacja o tym pojawi się po formularzem. Jego właścicielem (jednym z atrybutów tabeli 'Samochód') będzie oczywiście osoba, która tego dokonała. W przypadku niepowodzenia dodania pojazdu pod formularzem wyświetlona zostanie stosowna informacja czerwonym kolorem oznajmiająca jego powód.

Zdarzenia asynchroniczne (AJAX)

Z punktu widzenia użytkownika dodanie pojazdu jest bardzo łatwe lecz za prostym formularzem kryje się złożony ciąg operacji na bazie danych, a także na stronie. W pierwszej kolejności wykorzystywany jest język JavaScript. Po kliknięciu na przycisk „Zatwierdź” wywoływana jest funkcja `OnAddCarToDatabaseClicked()`. Jej zadaniem jest wywołanie kolejnej funkcji `addCarToDatabase(callback)`, która wykorzystując technologię AJAX próbuje dodać do bazy samochód deklarowany przez użytkownika. Argumentem tej funkcji jest delegat – wskaźnik na funkcję. Kod tej funkcji może być wywołany w dowolnym miejscu funkcji, która ją przechwytuje.

W pierwszej kolejności funkcja `addCarToDatabase(callback)` tworzy obiekt-żądanie (request), który służyć będzie do komunikacji z serwerem. W zależności od przeglądarki obiekt ten będzie innego typu. Poniższa linia kodu jest dobrym przykładem zastosowania typowania dynamicznego języka JavaScript:

```
var request = window.ActiveXObject ? new ActiveXObject('Microsoft.XMLHTTP') : new XMLHttpRequest;
```

W następnej linii deklarowane jest przechwytywanie danych zwróconych przez serwer wykorzystując delegat jako funkcję anonimową.

```
request.onreadystatechange = function() {  
    if (request.readyState == 4) {  
        request.onreadystatechange = doNothing;  
        callback(request, request.status);  
    }  
};
```

Jest to na razie deklaracja zachowania w wypadku przyszłych wydarzeń. W dalszej kolejności w funkcji `addCarToDatabase` sprawdzane są warunki poprawności danych wejściowych odczytując je bezpośrednio z pól formularza.

```
document.forms[0].marka.value
```

W przypadku poprawności danych na obiekcie request wykonywana jest metoda `open()`, której zadaniem jest utworzenie żądania do serwera. Jej argumenty przedstawione są poniżej.

```
request.open("nazwa metody protokołu http", "url skryptu z przekazaniem parametrów", true)
```

Wykorzystaną metodą protokołu http może tu być GET, POST lub REQUEST. Wykorzystanie którejkolwiek z tych funkcji jest sprawą umowną i nie ma to większego znaczenia, ponieważ żądania te obsługiwane będą przez skrypty będące częścią budowanego systemu a interpretacja tego żądania będzie opisana na kolejnej stronie. Drugim argumentem tej metody jest adres url strony a w zasadzie pliku, który ma obsłużyć żądanie razem z przekazanymi argumentami. Są nimi odczytane z formularza atrybuty dodawanego pojazdu (marka, model, nr rejestracyjny). Przekazanie tych argumentów jest bardzo proste. Po nazwie pliku dopisać należy znak „?”, a zaraz po nim nazwę pierwszego argumentu, znak „=” i wartość tego argumentu. Nie należy wpisywać spacji pomiędzy żadnymi znakami. Kolejny argument dodawany jest po dodaniu (zaraz po wartości poprzedniego argumentu) znaku „&”, a zaraz po nim analogicznie nazwy, znaku „=” i wartości kolejnego argumentu. Wszystkie następne argumenty dopisywane są w taki sam sposób tzn. po

znaku „&”. Wartości przekazanych argumentów pochodzą tu bezpośrednio z pól formularza. Wywołanie metody open na obiekcie request wygląda następująco. Wskazany plikiem jest skrypt addCar.php, którego zadaniem jest dodanie do bazy nowego samochodu.

```
request.open("GET", "addCar.php?nr_rej="+document.forms[0].nr_rej.value+"&marka="+
+document.forms[0].marka.value + "&model="+document.forms[0].model.value+"&zid=1", true);
```

Trzecim argumentem funkcji addCarToDatabase jest zezwolenie przeglądarce na asynchroniczne oczekiwanie na odpowiedź z serwera. Dzięki temu może zająć się ona innymi zadaniami w czasie oczekiwania na dane. Faktyczne wysłanie żądania następuje po wykonaniu na obiekcie request metody send()

```
request.send().
```

Dla obiektu request już jest zadeklarowane przechwytywanie danych z serwera. Jest nim wywołanie przekazanego do funkcji addCarToDatabase() delegatu. Ciało tego delegatu wykonuje szereg procedur pozwalających odczytać odebrane dane w formacie XML. Reprezentowany jest jako zmienna 'data' będąca argumentem delegatu. Plik w formacie XML znajduje się w polu 'responseXML'.

```
var xml = data.responseXML; // utworzenie zmiennej
                             // będącej referencją do
                             // odebranego dokumentu
var res = xml.documentElement.getElementsByTagName("nazwa elementu"); // wyłuskanie elementu
                             // po nazwie
```

Następnie ze zmiennej res odczytywany jest rozmiar odebranych danych. W przypadku gdy będą one równe zero dane nie zostały poprawnie przekazane z serwera. Jeżeli nie są puste podejmowana jest próba odczytania ich i w zależności od wartości jaką przekazują podejmowane są dalsze akcje. W przypadku skryptu addCar.php jest to pojedyncza wartość, ale mogą to być także całe tabele danych.

Zastosowanie techniki AJAX pozwala wykorzystywać skrypty napisane np. w PHP tak samo jak zwykłe funkcje. Analogicznie jak w funkcjach można przekazywać do nich argumenty tak jak to opisane zostało powyżej, ale także odbierać dane w postaci dokumentu o strukturze obiektowej w formacie XML. Taki sposób reprezentacji danych nazywa się DOM (ang. *Document Object Model*). Utworzony obiekt-żądanie request jak już było to poprzednio opisane jest przygotowany na odbieranie danych zwróconych przez serwer. Aby to się stało dane muszą być najpierw skonstruowane w wywołanym skrypcie addCar.php.

DOM

Argumenty przesłane do skryptu odczytywanie są w następujący sposób.

```
$zmienna = $_GET['argument']
```

gdzie \$_GET[] jest kontenerem przechowującym odebrane dane z metody GET. Następnie tworzony jest obiekt DOM.

```
$dom = new DOMDocument("1.0");
```

Następnie dla dokumentu tego tworzony jest znacznik języka XML.

```
$node = $dom->createElement("nazwa znacznika");
```

Obiekt ten może być rodzicem dla innych obiektów (znaczników) lub też istnieć w dokumencie autonomicznie. Można tworzyć dla niego kolejne znaczniki jak poniżej.

```
$newnode = $dom->appendChild($node);
```

W ten sam sposób można tworzyć kolejne znaczniki na tym samym poziomie lub analogicznie jego potomków. Zapisanie konkretnej wartości (atrybutu) w takim obiekcie wykonuje się korzystając z poniższej metody.

```
$newnode->setAttribute("nazwa atrybutu", "wartość ");
```

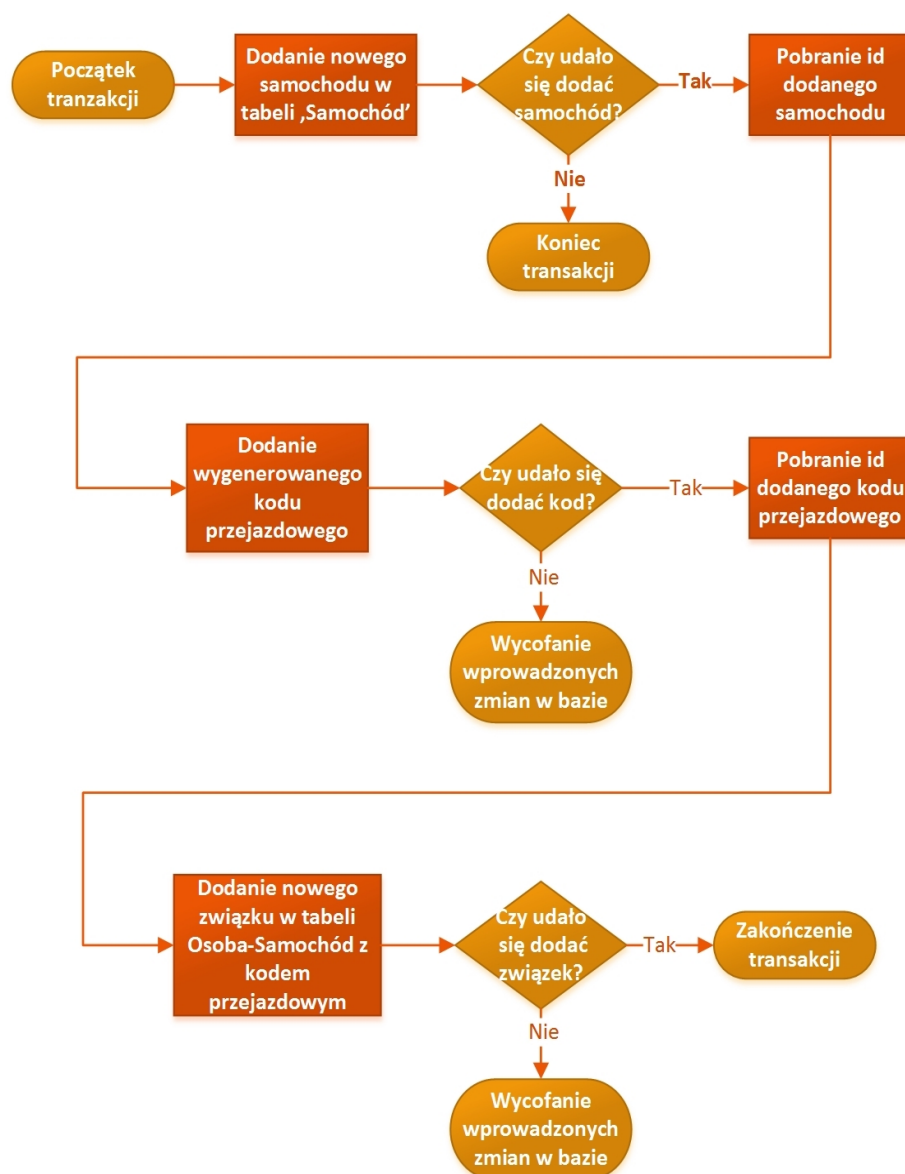
Wysłanie gotowego dokumentu odbywa się poprzez echo skryptu PHP tak jak poniżej.

```
echo $dom->saveXML();
```

Opisane w tym podrozdziale wykorzystanie DOM służy w skrypcie addCar.php jedynie do przekazania wyniku transakcji tzn. czy się udało, a jeżeli nie to z jakiego powodu. Zwracana jest więc tutaj tak naprawdę tylko jedna wartość, jednak, jak już było to powiedziane, w kolejnych skryptach będą to całe tabele danych. Najpierw jednak należy opisać operacje jakie wykonuje skrypt addCar.php.

Dodanie samochodu (PHP)

Skrypt addCar.php implementuje sekwencję zapytań definiujących i wybierających mających na celu dodanie nowego samochodu użytkownika i wygenerowanie dla niego kodu przejazdowego. Ponieważ jest to złożony ciąg operacji wykorzystany tutaj zostanie mechanizm transakcji udostępniony przez silnik MySQL. Najpierw zostanie jednak sporządzony algorytm zapytań wykonywanych na bazie przez ten skrypt.



Ilustracja 6.4.6: Algorytm transakcji dodania nowego pojazdu

Zastosowanie mechanizmu transakcji pozwala kontrolować dane wprowadzane do tabel. Jeżeli któreś z kolejnych zapytań modyfikujących nie powiedzie się to wszystkie poprzednie tego rodzaju zostają wycofane. Dzięki temu w bazie nie są przechowywane niepotrzebne dane.

Rozpoczęcie transakcji wymaga jedynie wysłania do bazy zapytania o treści: `START TRANSACTION`. Zakończenie transakcji jest wynikiem wykonania zapytania `COMMIT`, a wycofanie zmian `ROLLBACK`.

Przeglądanie pojazdów

Po kliknięciu na przycisk „Moje pojazdy” wyświetlone zostaną różne listy z samochodami. Może być ich maksymalnie 5:

- „Auta, których jestem właścicielem” - lista samochodów dodanych przez użytkownika, których jest właścicielem.

>auta, których jestem właścicielem

Marka	Model	Nr rejestracyjny	Kod wjazdowy	
FSO	Polonez	wr54645	3269125	<div>Zmień kod</div> <div>Udostępnij</div>

Ilustracja 6.4.7: Przykładowy widok tabeli aut użytkownika, których jest właścicielem i wjeżdża nimi w ramach swojego abonamentu

Przycisk „Zmień kod” służy do zmiany wygenerowanego kodu wjazdowego.

Przycisk „Udostępnij” pozwala na udostępnienie pojazdu innym użytkownikom co opisane będzie w kolejnym podrozdziale.

- „Udostępnione mi w ramach mojego abonamentu” - lista pojazdów, których właścicielem jest inny użytkownik, ale zostały udostępnione aktualnie przeglądającemu swoje pojazdy użytkownikowi.

> Udostępnione mi w ramach mojego abonamentu

Marka	Model	Nr rejestracyjny	Kod wjazdowy	Osoba udostępniająca samochód	
Peugeot	106	wa454350	3622758	Jacek Placek	<div>Zmień kod</div>

Ilustracja 6.4.8: Przykładowy widok tabeli aut, których użytkownik nie jest właścicielem, ale może nimi wjeżdżać w ramach swojego abonamentu

Przycisk „Zmień kod”, podobnie jak w poprzedniej tabeli służy do zmiany wygenerowanego kodu wjazdowego.

- „Udostępnione innym w ramach ich abonamentu” - lista samochodów dodanych przez użytkownika, których jest właścicielem, ale zezwolenie na wjazd nimi mają także inni użytkownicy zarejestrowani w systemie. Zezwolenie to pochodzi z woli użytkownika, który jest właścicielem tego pojazdu.

> Udostępnione innym w ramach ich aboanamentu

Marka	Model	Nr rejestracyjny	Osoba, której udostępniono samochód
FSO	Polonez	wr54645	Jan Nowak

Ilustracja 6.4.9: Przykładowy widok tabeli aut, których użytkownik jest właścicielem i udostępnił je innym użytkownikom w ramach ich abonamentu

- „Udostępnione od innych w ramach ich abonamentu” - lista samochodów dodanych przez użytkownika, których jest właścicielem, ale wjazd na parking tym pojazdem zapewniony jest w ramach abonamentu innego użytkownika.

> Udostępnione od innych w ramach ich abonamentu

Marka	Model	Nr rejestracyjny	Kod wjazdowy	Osoba udostępniająca abonament	
FSO	Polonez	wr54645	7374548	Michael Schumacher	Zmień kod

Ilustracja 6.4.10: Przykładowy widok tabeli aut, na które możliwość wjazdu użytkownik dostał od innych użytkowników

- „Udostępnienie mojego abonamentu innym” - lista samochodów, których właścicielem jest inny użytkownik i może on wjeżdżać tym samochodem w ramach abonamentu bieżącego użytkownika, które mu ten abonament uprzednio udostępnił.

> Udostępnienie mojego abonamentu innym

Marka	Model	Nr rejestracyjny	Kod wjazdowy	Osoba, której udostępniono abonament
Nissan	Micra	wh78843	9951861	Wacek Kobyłka

Ilustracja 6.4.11: Przykładowy widok tabeli aut, na które możliwość wjazdu użytkownik udostępnił innym użytkownikom w ramach swojego abonamentu

Tabela ta może zawierać pojazdy, których właścicielem jest sam użytkownik lub osoba, której abonament jest udostępniony.

Za wyświetlaniem tabel z pojazdami kryje się nieco skomplikowana logika. Za reagowanie na kliknięcia na przyciski na stronie odpowiadają funkcje zaimplementowane w skrypcie języka JavaScript. Kliknięcie na przycisk „Moje pojazdy” powoduje wywołanie funkcji `OnMyCarsClick()`. Do głównych jej zadań należy wywoływanie z odpowiednimi argumentami funkcji `getCars()`. Ta z kolei działa bardzo podobnie do opisywanej wcześniej `addCarToDatabase()`, która zgłasza żądanie do serwera o dodanie nowego samochodu do bazy i wygenerowanie dla użytkownika kodu wjazdowego na ten samochód. `getCars()` z kolei ma za zadanie wyciągnąć z bazy informacje o samochodach konkretnego użytkownika. Jednak jej argumentem jest oprócz delegata reagującego na dane z serwera, także konkretny url skryptu do którego się ma odwołać razem z argumentami tego. W `OnMyCarsClick()` wywoływana jest ona w sumie 5 razy – dla każdej z możliwych do wyświetlenia tabel.

Dla tabeli „Auto, których jestem właścicielem” jest to skrypt `getCars.php`. Dla tabel „Udostępnione mi w ramach mojego abonamentu” oraz „Udostępnione innym w ramach ich abonamentu” jest to skrypt `get_shared_cars.php`. A dla tabel „Udostępnione od innych w ramach ich abonamentu” i „Udostępnienie mojego abonamentu innym” jest to skrypt `get_shared_Abo_cars.php`.

Jak już było wspomniane delegat przekazywany zarówno do funkcji `addCarToDatabase()` i `getCars()`, zajmuje się obsługą danych zwróconych przez serwer. W przypadku tej drugiej jest to jednak proces znacznie bardziej złożony. Funkcja ta musi bowiem przenieść te dane do tabel na stronie. Tabele te są tak naprawdę deklarowane na stronie razem przyciskami które powodują ich wygenerowanie. Pochodzą z wygenerowanego dynamicznie echa skryptu PHP tej strony. Przypisane są im odpowiednie identyfikatory, które mogą być zwykłą niepowtarzalną nazwą.

Wszystkie odebrane dane zawarte są w poniższej zmiennej xml. Do zmiennej `cars`,

zdefiniowanej poniżej, przekazywane są wszystkie dane pod znacznikiem o tej samej nazwie.

```
var cars = xml.documentElement.getElementsByTagName("car")
```

Zmienna ta jest kontenerem zawierającym wiersze będące wynikiem zapytania wykonanego przez wywołany skrypt. Następnie po sprawdzeniu czy wynik zapytania nie jest pusty.

```
if (cars.length>0) ,
```

Jeżeli nie są one puste rozpoczynane jest wypełnianie wskazanej tabeli danymi. Najpierw zapisany zostanie jej nagłówek. Wskazanie tabeli ze strony w JavaScript wykonuje się w następujący sposób.

```
var table = document.getElementById("nazwa tabeli")
```

Utworzenie nagłówka wymaga stworzenia nowego wiersza tabeli tak jak poniżej.

```
var header_row = document.createElement("TR")
```

Następnie nadawany jest mu identyfikator.

```
header_row.setAttribute("id", "id nagłówka"),
```

Potem można go dodać do tabeli w następujący sposób.

```
document.getElementById("myCarsToAlienTable").appendChild(table)
```

W nowo utworzonym wierszu można dodawać teraz kolumny.

```
var new_collumn = document.createElement("TD");           // utworzenie nowej kolumny
var new_text_field = document.createTextNode("nazwa");      // utworzenie nowego pola tekstowego
new_collumn.appendChild(new_text_field);                   // dodanie pola tekstowego do kolumny
header_row.appendChild(new_collumn);                        // dodanie kolumny do wiersza
```

Po dodaniu nagłówka można wypełnić tabelę danymi. Należy je wyciągnąć z kontenera *cars*. Najlepiej zrobić to w pętli po wszystkich jego elementach. W każdej iteracji odczytywany jest jeden wiersz reprezentujący dane pojazdu i dodawany do tabeli. Odczytanie pojedynczej wartości z wiersza, którą może być model lub marka pojazdu wygląda następująco.

```
var marka = cars[i].getAttribute("marka");
```

Następnie należy dodać nowy wiersz analogicznie jak dla nagłówka. Dodanie odczytanej wartości do wiersza wygląda prawie tak samo jak dodanie nazwy nagłówka. Zamiast stałego tekstu dodać należy utworzoną zmienną z odczytaną daną.

```
var new_text_field = document.createTextNode(marka);
```

W niektórych tabelach pojawiają się przyciski służące np. do udostępnienia pojazdu innym użytkownikom lub zmiany kodu wjazdowego. Dodanie takiego przycisku przebiega bardzo podobnie jak do dodania zwykłego pola z tekstem lecz musi mieć określony właściwy typ.

```
var new_button = document.createElement("BUTTON");
```

Aby z takiego przycisku był jakikolwiek pożytek musi zostać mu przypisana właściwa akcja. Zajmują się tym odpowiednie funkcje. Do zadeklarowania akcji zmiany kodu wjazdowego dla danego pojazdu służy funkcja `setCodeChangeButton(„id przycisku”)`. Jej argumentem jest identyfikator danego przycisku. Powinien on być unikatowy w skali całej strony, aby akcja zdefiniowana była jednoznacznie dla danego przycisku. W funkcji tej najpierw tworzona jest zmienna reprezentująca dany przycisk na podstawie przekazanego identyfikatora.

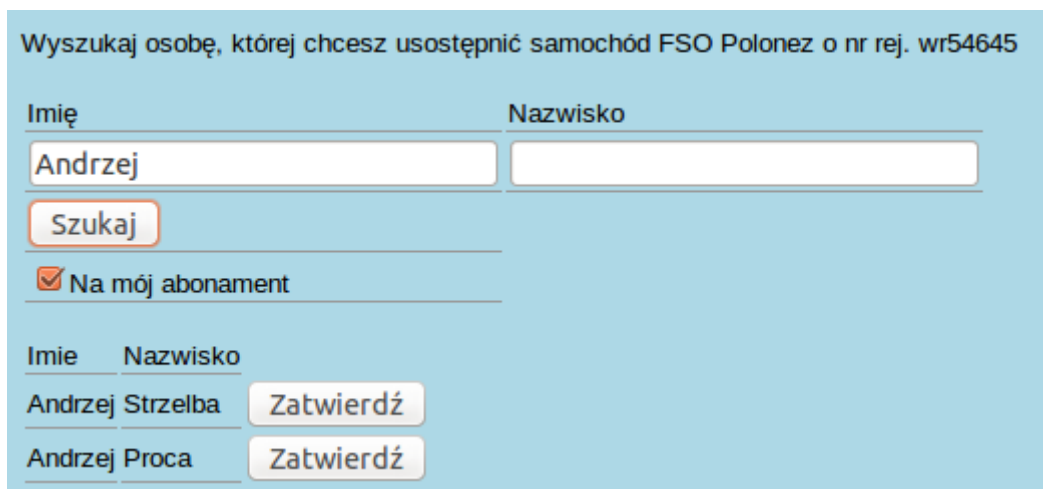
```
var button = document.getElementById(„przekazany identyfikator”);
```

Następnie dla przycisku tego deklarowany jest tak zwany action listener reagujący na kliknięcie na przycisk. Przekazywany jest mu delegat który wyzwalany jest właśnie takiej interakcji ze strony użytkownika. Procedury egzekwowane w tej funkcji są analogiczne jak w poprzednio opisywanych `addCarToDatabase()` i `getCars()`. Najpierw tworzony jest obiekt-żądanie, a potem wywoływana funkcja, która to żądanie egzekwuje, analogicznie jak funkcje `addCarToDatabase()` i `getCars()`. Dla akcji zmiany kodu w dalszej kolejności zgłaszane jest poprzez request zgłaszane jest żądanie wywołania skryptu `changeCode.php`.

Udostępnianie pojazdów i abonamentu

W założeniach do funkcjonalności systemu jest możliwość udostępniania między użytkownikami samochodów oraz abonamentu. To pierwsze założenie przewiduje sytuację, gdy jednym samochodem może wjeżdżać więcej niż jedna osoba. Może się tak zdarzyć gdy np. małżeństwo korzysta z jednego parkingu tym samym pojazdem i nie zawsze wjeżdżają oni razem. To drugie tzn. udostępnianie abonamentu ma miejsce, gdy to przykładowe małżeństwo podobnie wjeżdża na parking tym samym pojazdem, ale nie chcą oni płacić dwóch oddzielnych abonamentów za tą usługę. W witrynie będzie istniała, więc możliwość udostępnienia swojego samochodu w ramach swojego abonamentu, a także udostępnienia swojego abonamentu innej osobie na jej samochód.

Aby udostępnić swój samochód innej osobie należy w tabeli „Auta, których jestem właścicielem” kliknąć przy wybranym samochodzie na przycisk „Udostępnij”. Wyświetlony zostanie następujący formularz umożliwiający wyszukiwanie innych użytkowników po imieniu i nazwisku.



Wyszukaj osobę, której chcesz udostępnić samochód FSO Polonez o nr rej. wr54645	
Imię	Nazwisko
<input type="text" value="Andrzej"/>	<input type="text"/>
<input type="button" value="Szukaj"/>	
<input checked="" type="checkbox"/> Na mój abonament	
Imię	Nazwisko
Andrzej Strzelba	<input type="button" value="Zatwierdź"/>
Andrzej Proca	<input type="button" value="Zatwierdź"/>

Ilustracja 6.4.12: Formularz do wyszukiwania osób

CheckBox „Na mój abonament” pozwala użytkownikowi zdecydować czy swój samochód udostępnia w ramach swojego abonamentu czy też na jego koszt. Kliknięcie przycisku „Zatwierdź” spowoduje udostępnienie danego samochodu. Jeżeli przebiegnie to prawidłowo wyświetlona zostanie informacja jak na zrzucie poniżej.

☒ Na mój abonament

Twój samochód został pomyślnie udostępniony

Imie	Nazwisko	
Andrzej	Strzelba	Zatwierdź
Andrzej	Proca	Zatwierdź

Ilustracja 6.4.13: Informacja o udostępnieniu pojazdu

W wypadku nie powodzenia, które może być spowodowana próbą udostępnienia tego samego samochodu tej samej osobie wyświetlona zostanie informacja jak poniżej.

☒ Na mój abonament

Błąd wewnętrzny. Prawdopodobnie już wykonałeś taką operację

Imie	Nazwisko	
Andrzej	Strzelba	Zatwierdź
Andrzej	Proca	Zatwierdź

Ilustracja 6.4.14: Informacja o niepowodzeniu operacji z możliwą przyczyną

Kolejną funkcjonalnością jest możliwość przekazania swojego abonamentu innej osobie co pozwoli jej wjeżdżać swoim pojazdem, ale na koszt osoby udostępniającej. W tym celu należy kliknąć przycisk „Udostępnij w ramach swojego abonamentu”. Pod nim wyświetlony zostanie bardzo podobny formularz jak w przypadku udostępniania pojazdów. Także pozwala on na wyszukiwanie osób, które wyświetlą się w tabeli jak poniżej.

Udostępnij w ramach swojego abonamentu

Imię Nazwisko

Ja

Szukaj

Osoba		Samochody		
Imie	Nazwisko	Marka	Model	Nr rejestracyjny
Jacek	Placek	Samochody	Peugeot	106 wa454350
Jan	Nowak	Samochody		

Zatwierdź

Ilustracja 6.4.15: Formularz do wyszukiwania osób oraz przeglądania ich pojazdów

Przy każdej osobie znajduje się przycisk „Samochody”, który po kliknięciu pokazuje drugą tabelę wyświetlającą samochody danej osoby. Przy każdym samochodzie z kolei znajduje się przycisk „Zatwierdź” dokonujący faktycznego udostępnienia na dany samochód innego użytkownika. Gdy udostępnienie przebiegnie poprawnie w formularzu zostanie wyświetlona informacja jak poniżej.

Samochód osoby został pomyślnie udostępniony w ramach Twojego abonamentu

Gdy operacja nie zakończy się powodzeniem, np. podobnie jak w przypadku udostępniania samochodów z powodu próby wykonania ponownie tej samej czynności, pojawi się informacja:

Błąd wewnętrzny. Prawdopodobnie już wykonałeś taką operację

6.4.4 Pozostałe strony

Serwis budują także inne strony. Są to strona do której hiperłącze znajduje się w menu strony pod nazwą „Jak działa rejestracja”. Zawiera ona opis korzystania z serwisu. Jej implementacja znajduje się w pliku Jak.php. Dostępna jest każdemu kto odwiedzi witrynę bez konieczności logowania się.

Strona pod hiperłączem „Wskazówki dojazdu” wyświetla mapę z lokalizacją parkingu na którym możliwy jest wjazd. Obecna mapa wyświetla położenie wydziału Mechatroniki.



Ilustracja 6.4.16: Przykład wyświetlenia mapy na stronie „Wskazówki Dojazdu”

Strona „Informacje kontaktowe” zawiera m.in. numery telefonów do osób mogących pomóc

w korzystaniu z parkingu. Zawiera także adresy stron związanych np. strona główna wydziału Mechatroniki.

6.5 Podsumowanie

Jak już było wspomniane na wstępie tego rozdziału przedstawione w nim opisy nie dotyczyły kompletnych zabiegów związanych z programowaniem elementów systemu. Nie opisuje ona wszystkich stron jakie zostały zaimplementowane lecz ich najważniejsze elementy na pojedynczych przykładach, aby nie opisywać kilkakrotnie tych samych działań. Krótki opis wszystkich stron (plików) znajduje się w tabeli [], co jest wystarczające do zrozumienia ich funkcji.

Rozdział ten przedstawił opis implementacji witryny internetowej będącą internetowym interface'm użytkownika. Witryna ta została w pełni zaimplementowana spełniając wszystkie wstępne założenia. Rozdział 8 przedstawi jej testy.

7. Program pośredniczący w komunikacji między mikrokontrolerem a bazą danych

7.1 Koncepcja aplikacji

LPC2368 docelowo łączy się z bazą danych MySQL. Jednak implementacja poleceń protokołu MySQL na mikrokontrolerze byłaby ogromnym wysiłkiem. Z tego powodu znacznie łatwiej będzie napisać aplikację pośredniczącą w wymianie danych z bazą. Dostępnych bezpłatnie jest wiele bibliotek implementujących kompletnie protokół MySQL. Są to tak zwane drivery. Przykładem może tu być framework Qt, ale także Java w której to właśnie napisana zostanie opisywana aplikacja. Potrzebna do tego będzie jednak wspomniana biblioteka o nazwie: `mysql-connector-java`. Dostępna jest ona na stronie `dev.mysql.com`, oczywiście bezpłatnie.

7.2 Wykonanie

Aby możliwe było korzystanie z drivera należy dodać go do projektu. W środowisku Eclipse trzeba wejść do właściwości (properties) projektu. Następnie przejść do edytora ścieżek budujących projekt (Java Build Path) i wybrać zakładkę bibliotek (Libraries). W niej należy wybrać dodawanie zewnętrznych bibliotek (Add External JAR's) i wskazać pobraną bibliotekę `mysql-connector-java`.

Kolejnym krokiem jest utworzenie nowej klasy która będzie połączeniem do bazy danych poprzez pobraną i dodaną bibliotekę. Klasa ta nazwana będzie `DBConnector`. W pliku implementującym ją na samym wstępie należy zaimportować niezbędne przestrzenie nazw zawierające klasy, które będą później wykorzystywane:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
```

W klasie należy zadeklarować następnie zmienne niezbędne do dalszej pracy.

```
private Connection con; // obiekt będący bezpośrednim połączeniem z bazą
private Statement st; // zapytanie, które zostanie wysłane do bazy
private ResultSet rs; // rezultat wykonanego zapytania
```

W dalszym ciągu niezbędnych czynności zaimplementować należy obiekt `con` klasy `Connection` wykorzystując konstruktor tej klasy. Po utworzeniu bloku try – catch zainicjowany będzie ten obiekt.

```
con = DriverManager.getConnection("jdbc:mysql://ip_serwera:port/nazwaBazy",
"uzytkownik","haslo");
```

Jako `IP_serwera` należy wpisać to na którym znajduje się baza MySQL. Jeżeli baza znajduje się na tej samej maszynie co program łączący się z nią można jako IP wpisać `127.0.0.1`. Port domyślny protokołu MySQL to `3306`.

Następnie zainicjować trzeba obiekt `st` klasy `Statement`, a więc zapytanie wysyłane do bazy danych, wykorzystując metodę `createStatement()` obiektu-połączenia `con`.

```
st = con.createStatement();
```

W bloku catch jako argument należy podać standardowy obiekt klasy Exception. Metoda odpytująca bazę danych w celu weryfikacji kodu opisana jest w **rozdziale[]**.

Kolejną klasą, która będzie utworzona w ramach naszej aplikacji pośredniczącej jest Server. Jest to klasa implementująca gniazdo sieciowe do komunikacji z mikrokontrolerem LPC. Jest to bardziej złożona klasa niż DBConnector, więc nie będzie ona opisywana tak szczegółowo. Jej implementacja rozpoczyna się od importowania przestrzeni nazw z których istotne są java.io.* oraz java.net.* gdyż one zawierają klasy niezbędne do komunikacji sieciowej z wykorzystaniem protokołu TCP.

Klasa Server dziedziczy po klasie JFrame, która tworzy okno wyświetlane użytkownikowi w którym umieścić można przyciski, pola tekstowe oraz inne elementy graficznego interfejsu użytkownika (GUI). Posiada ona kilka pól prywatnych niezbędnych do komunikacji sieciowej. Są to:

```
private ObjectOutputStream output;    // strumień danych wyjściowych
private ObjectInputStream input;      // strumień danych wejściowych
private ServerSocket server;         // serwer TCP
private Socket connection;           // gniazdo sieciowe
```

Pozostałe pola to elementy GUI i nie mają one większego związku z komunikacją sieciową. ChatWindow wypisuje logi komunikacji. Bardzo ważnym polem jest tutaj instancja omawianej w poprzednich akapitach klasy DBConnector.

```
private DBConnector con;
```

W konstruktorze klasy Server oprócz inicjalizacji związanych z GUI znajduje się też zaalokowanie pamięci dla naszego łącznika z bazą MySQL:

```
con = new DBConnector();
```

Metodą która inicjuje serwer TCP z którym komunikować się będzie mikrokontroler LPC jest startRunning(). Poniżej całe jej ciało.

```
public void startRunning(){
    try{
        server = new ServerSocket(50000, 100);
        while(true){
            try{
                waitForConnection();
                setupStreams();
                whileChatting();
            }catch EOFException eofException{
                showMessage("\n Server ended the connection! ");
                System.out.println("Rzucony wyjatek");
            }finally{
                close();
            }
        }

        }catch(IOException ioException){
            ioException.printStackTrace();
        }
}
```

W pierwszej kolejności inicjowane jest pole server, którego konstruktor przyjmuje dwa

argumenty. Pierwszy to port na którym będzie nasłuchiwał połączeń budowany serwer, a drugi to liczba dopuszczalnych połączeń – klientów. Efektem tej linii powyższego kodu jest zaalokowanie pamięci dla gniazda sieciowego do komunikacji TCP.

Dalej w bloku try wywoływane są trzy metody. Pierwszą z nich jest `waitForConnection()`, której całe ciało znajduje się poniżej.

```
private void waitForConnection() throws IOException{
    showMessage("Waiting for some to connect...\n");
    connection = server.accept();
    showMessage(" Now connected to "+
        connection.getInetAddress().getHostName());
}
```

Program zatrzymuje się w linii `connection = server.accept()` do momentu aż nie pojawi się nowe połączenie od klienta, którym w tym wypadku będzie mikrokontroler LPC. Kiedy połączenie zostanie nawiązane program jest wykonywany dalej. Przy okazji wyświetlane są informacje dla użytkownika. Następnie program przechodzi do funkcji `setupStreams()`.

```
private void setupStreams() throws IOException{
    output = new ObjectOutputStream(connection.getOutputStream());
    output.flush();
    input = new ObjectInputStream(connection.getInputStream());
    showMessage("\n Streams are now setup\n");
}
```

Rolą tej funkcji jest synchronizacja strumieni. Strumienie to obiekty służące do komunikacji w Javie, która wykorzystuje serializację. Serializacja to zamiana obiektów w dane i wysłanie ich. W naszym przypadku z punktu widzenia programowania w Javie wysyłanym oraz odbieranym obiektem będzie łańcuch znaków, czyli zwykły tekst.

W funkcji `setupStreams()` najpierw alokowana jest pamięć na strumień wyjściowy. Następnie wykonywana jest metoda `flush()`, której efektem jest wysłanie danych synchronizacyjnych do klienta. W kolejnej linii inicjowany jest strumień wejściowy (`input`). Program zatrzymuje się w tym miejscu i oczekuje na odebranie danych synchronizacyjnych od klienta, tych samych, które wysłane zostały w poprzedniej linii. Kiedy klient odeśle echo strumienie są w pełni zsynchronizowane i program przechodzi dalej do funkcji `whileChatting()`, której uproszczona forma przedstawiona jest poniżej.

```
private void whileChatting() throws IOException {
    .
    .
    do {
    try {
        message = (String) input.readObject(); // Odczytanie wiadomości
        .
        if (message.contains("REQ"))
            continue;
        else if (message.substring(0, 1).contains("D"))
            con.sendLog(); // Zapisanie w bazie
        else if (message.charAt(7) != ')' && message.charAt(7) != '+') {
            this.sendMessage("zzzzzzzz"); // Niewłaściwy format kodu
        } else {
            results =
                con.allowOpen(message); // Weryfikacja kodu
        }
    }
    }
}
```

```

        if (results == 0)                // Można wjeżdżać
        {
            this.sendMessage("bbbbbbbb");
        } else {
            if (results == 2)            // Błędny kod
                this.sendMessage("rrrrrrrr");
            else if (results == 3)        // Kod w użyciu
                this.sendMessage("ggggggg");
            else if (results == 4)        // Brak zezwolenia
                this.sendMessage("ccccccc");
            else if (results == 5)        // Inny samochód tego użytkownika
                this.sendMessage("aaaaaaa");
            else if (results == 6)        // Zakaz wyjazdu
                this.sendMessage("ttttttt");
            else                          // Błąd wewnętrzny
                this.sendMessage("ppppppp");
        }
    } catch (ClassNotFoundException classNotFoundException) {
        showMessage("\n idk what LPC has send!");
    }
} while (!message.equals("END")); // gdy klient coś takiego napisze to
                                // połączenie zostaje rozłączone
}

```

Funkcja ta działa w oparciu o pętlę do – while. W bloku try poprzez metodę readObject() strumienia wejściowego odczytywane są odebrane dane, a właściwie następuje ich deserializacja. W przypadku błędu deserializacji rzuca wyjątek i program znów próbuje odczytać dane wejściowe w tej samej linii. Jeżeli odczytane są one prawidłowo wywołana jest na obiekcie con klasy DBConnector metoda allowOpen, której przekazywana jest odebrana od mikrokontrolera wiadomość. Decyduje ona odpytując bazę danych czy przekazany kod upoważnia do przejazdu przez szlaban. Opis działania tej metody oraz zwracane przez nią wartości opisane są w kolejnym **podrozdziale**.

Każda z opisanych powyżej funkcji wywoływanych w funkcji startRunning() rzuca wyjątek typu IOException. Z tego powodu znajdują się one w blokach try-catch. Jest to dość powszechna praktyka programowania w języku Java. Umożliwia ona kontynuowanie pracy programu nawet w wypadku poważnych błędów i to nie tylko dotyczących operacji wejścia-wyjścia jak w tym przypadku.

Klasą której instancja inicjuje działanie serwera pośredniczącego jest Program. W niej to w metodzie main() tworzona jest instancja klasy Server. Poniżej funkcja main() Programu.

```

public static void main(String[] args) {
    Server server = new Server();
    server.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    server.startRunning();
}

```

Najpierw zaalokowana jest pamięć dla obiektu server, a w następnej linii deklarowane jest zachowanie się obiektu po naciśnięciu przycisku zamknij okna servera. Należy tutaj przypomnieć że klasa Server dziedziczy po klasie JFrame, która reprezentuje okno graficznego interface'u użytkownika (GUI). Ostatnim poleceniem programu głównego jest wywołanie opisanej już metody startRunning() na obiekcie server.

Sporo powiedziane było o alokacji pamięci dla obiektów, nic jednak o zwalnianiu jej. W języku Java nie jest to jednak problemem, gdyż w przeciwieństwie do innych języków spełniających obiektowy, strukturalny a także proceduralny paradygmat programowania takich jak

C czy C++ programista jest zwolniony z obowiązku zarządzania pamięcią. Wyręcza go w tym tzw. odśmieczacz (Garbage Collector), który działa w kontekście maszyny wirtualnej Javy i zajmuje się zarządzaniem pamięcią.

7.3 Protokół komunikacji z mikrokontrolerem LPC

Aplikacja pośrednicząca pomiędzy LPC a bazą danych do komunikacji z mikrokontrolerem wykorzystuje protokół TCP. Wymiana danych odbywa się w oparciu o architekturę klient-serwer. Serwer jest zaimplementowany w klasie 'Server' aplikacji pośredniczącej, natomiast klientem jest platforma E2LP w której komunikacją sieciową zajmuje się LPC.

Serwer nasłuchuje na porcie 50000. Po nawiązaniu połączenia TCP serwer wysyła inicjującą wiadomość którą mikrokontroler zapisuje w swojej pamięci. Staje się ona dla niego szablonem, który z odpowiednimi modyfikacjami odsyła on do serwera przesyłając informacje np. kody przejazdowe.

Ze względu na stosowaną w technologii Java serializację ramka danych odbieranych przez aplikację w ten sposób napisaną musi mieć odpowiednią postać, aby mogła być odpowiednio zdekodowana przez obiekt klasy `ObjectInputStream`, którego zadaniem jest zrzutowanie danych na odpowiedni obiekt. Kodowaniem danych zajmuje się obiekt klasy `ObjectOutputStream`.

W pierwszych 3 bajtach zawarte są dane na temat serializowanego obiektu, którym w tym przypadku jest łańcuch znaków. Kolejne 7 bajtów to miejsce na wpisany kod przejazdowy. Następny bajt to informacja o kierunku przejazdu, którą można interpretować także jako identyfikator terminala – wjazdowy lub wyjazdowy.

Kiedy samochód przejeżdża przez szlaban i rejestruje to czujnik optyczny to wysyłana jest ramka tej samej długości z tym, że dane użyteczne to oprócz tych wymaganych do serializacji także znak „D”, który jest interpretowany w odpowiedni sposób przez serwer. Zajmuje się tym opisana poprzednio funkcja `whileChatting()`.

7.4 Weryfikacja kodu wjazdowego z bazą danych

Do weryfikacji kodów wjazdowych służy funkcja `allowOpen()`. Wykonuje ona sekwencję zapytań wybierających mających na celu wyłuskanie informacji jaka osoba i jakim samochodem próbuje przejechać przez szlaban oraz czy może przez niego przejechać, a jeżeli nie to z jakiego powodu.

Poniżej została przedstawiona w dużym uproszczeniu funkcja `allowOpen()`.

```
public int allowOpen(String message) {
    try {
        String query = "zapytanie sql";           // Utworzenie zapytania
        rs = st.executeQuery(query);               // Wykonanie zapytania

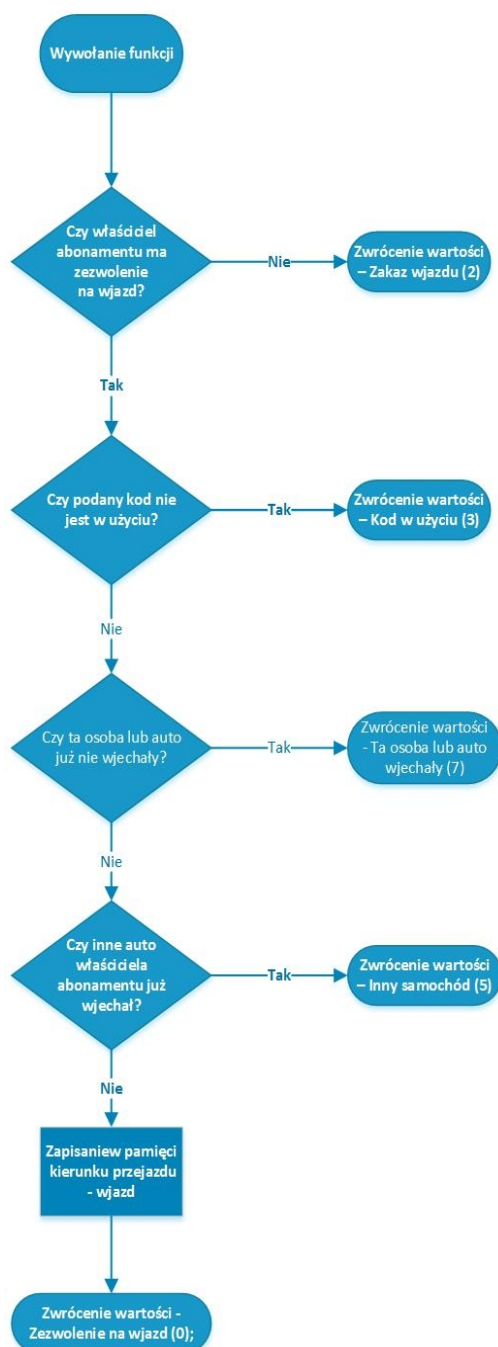
        while (rs.next()) {
            String data = rs.getString("pożądana kolumna");
            // Przetwarzanie odebranych danych
        }

    } catch (Exception ex) {
        System.out.println(ex);
    }
}
```


W bloku try tworzone jest zapytanie jako zwykły String. Następnie wykorzystując metodę `executeQuery()` z `rs` jest ono wykonywane w bazie, a jego rezultat przekazywany jest do obiektu `rs`. Warto pamiętać, że metoda `executeQuery()` wykonuje tylko zapytania wybierające dane, a nie definiujące czy też modyfikujące. Do tego służy metoda `executeUpdate()`.

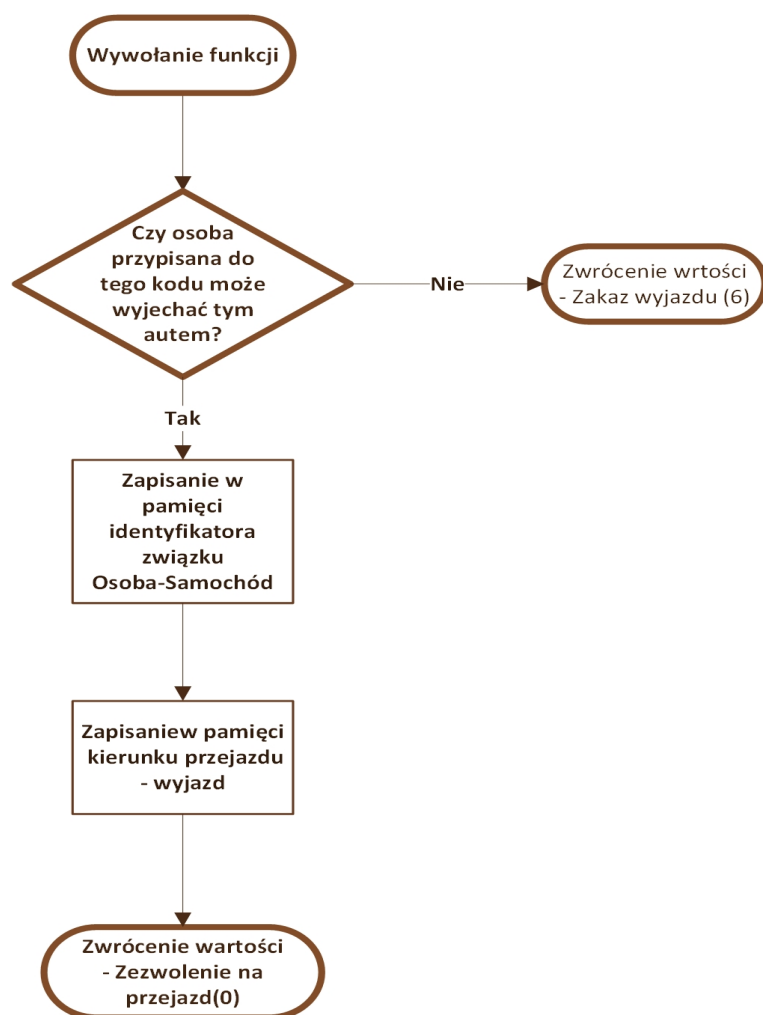
Zapytania wykonywane są w odpowiedniej sekwencji w zależności od tego czy osoba próbuje wjechać czy też wyjechać z parkingu. Algorytmy tych zapytań w zależności od kierunku przejazdu znajdują się poniżej. W nawiasach wypisane są wartości zwracane przez funkcję.

- Wjazd



Ilustracja 7.4.1: Algorytm odpytywania bazy danych gdy użytkownik próbuje wjechać

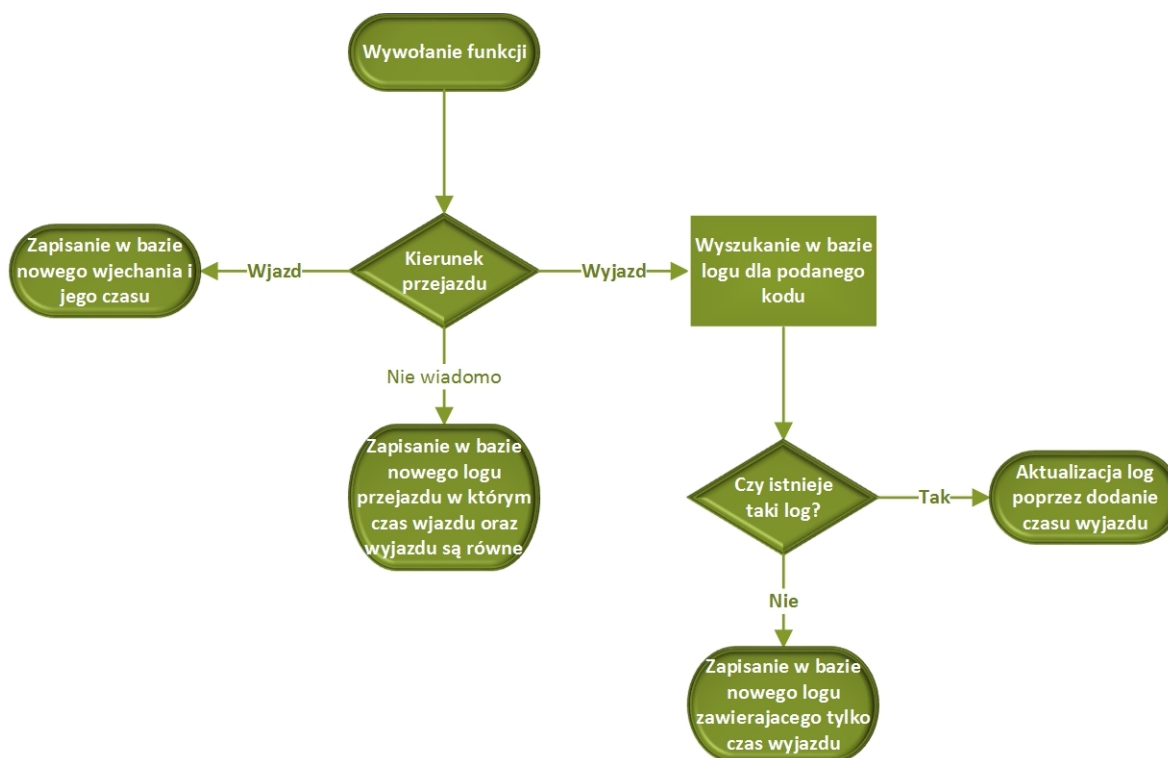
- Wyjazd



Ilustracja 7.4.2: Algorytm odpytywania bazy danych gdy użytkownik próbuje wyjechać

7.5 Odnotowanie wjazdu i wyjazdu

W tabeli 'Wjechanie' zapisywane są czasy wjazdu i wyjazdu każdego parkowania. Następuje to nie w momencie wpisania kodu lecz kiedy czujnik optyczny pod szlabanem wykryje że auto przejechało. Jeżeli przesłana ramka danych zawiera znak „D” to wywoływana jest funkcja odpowiedzialna za aktualizację danych w tabeli 'Wjechanie' – sendLog(). Algorytm działania tej funkcji przedstawiony jest poniżej.



Ilustracja 7.5.1: Algorytm dodawania logu wjazdu lub wyjazdu

Zapisywanie wjazdów na parking jest bardzo istotną funkcją bazy, gdyż pozwala określić czy samochód danego użytkownika znajduje się na parkingu oraz przede wszystkim dokumentować każde wjechanie co pozwala na generowanie raportów, a także prowadzenie statystyk korzystania z parkingu.

7.6 Podsumowanie

Napisany w języku Java program jest ważną częścią systemu, gdyż nie tylko jest pomostem pomiędzy E2LP, ale również to on odpowiada de facto za weryfikację uprawnień osoby próbującej wjechać lub wyjechać z parkingu. Mikrokontroler wysyła do tej aplikacji jedynie kod wpisany przez użytkownika oraz kierunek przejazdu i oczekuje na informację zwrotną. Cała logika odpytywania bazy danych znajduje się właśnie w niej.

Opracowana w tym rozdziale oraz w poprzednich część praktyczna pracy jest podstawa do przedstawienia modelu systemu opisanego w kolejnym rozdziale.

8. Program mikrokontrolera PicoBlaze

Ważnym elementem aplikacji FPGA jest 8-bitowy procesor softwaerowy PicoBlaze. Udostępniony na zasadach wolnej licencji przez firmę Xilinx. Programuje się go w assemblerze. W projektowanym układzie służyć będzie do obsługi urządzeń zewnętrznych takich jak wyświetlacz LCD, port podczerwieni, port UART oraz mikrokontroler LPC poprzez konektory FMC. Pełna dokumentacja assemblera PicoBlaze'a jest udostępniona przez firmę Xilinx w **dokumencie** [1].

8.1 Emulacja terminalu do wpisywania kodów

Jednym z elementów projektowanego systemu zgodnie z założeniami są dwa terminale umożliwiające weryfikację kodów przejazdowych. Budowa własnych terminali lub zakup gotowych są działaniami zbyt czasochłonnymi lub kosztownymi. Z tego powodu należy poszukać prostszych rozwiązań. Z pomocą przychodzą tu peryferia znajdujące się na platformie E2LP. Konkretnie mowa tu o porcie podczerwieni. Konfiguracja portu pozwalająca na odczytywanie danych wysyłanych bezprzewodowo opisana w pracy przejściowej[2]. Rolę terminala wjazdowego pełnić będzie pilot od telewizora nadający w standardzie RC-5. Z racji tego, że klawiatura na pilocie posiada w zasadzie tylko znaki numeryczne od 0 do 9 to kody wjazdowe systemu także znajdować się będą w tym zakresie. Dla celów demonstracyjnych w zupełności to wystarczy. Jeden pilot pełnić będzie zarówno rolę terminala wjazdowego jak i wyjazdowego. Poniżej zdjęcie pilota z opisem wykorzystanych przycisków.

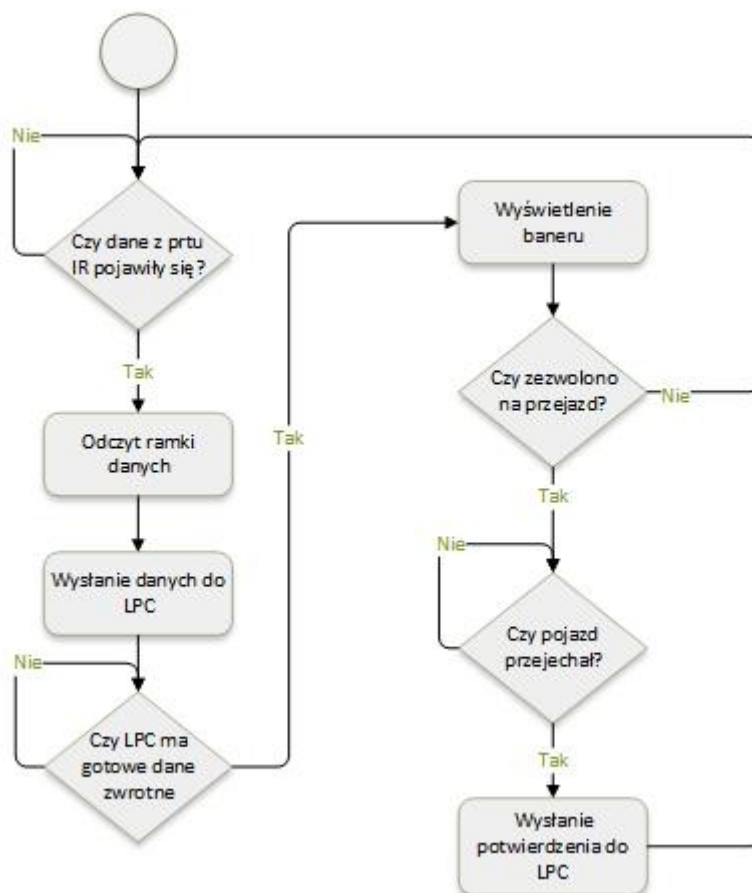


Ilustracja 8.1.1: Pilot emulujący terminal wjazdowy z opisem przycisków

Kod przejazdowy składać będzie się z 7 znaków. Ponieważ jeden pilot reprezentować będzie terminal wjazdowy i wyjazdowy po wpisaniu kodu przycisnąć należy odpowiedni przycisk wskazujący kierunek – wjazd lub wyjazd. Sekwencja w sumie 8 znaków zaraz po wpisaniu jest przesyłana z PicoBlaze do mikrokontrolera, a następnie do aplikacji LPCToMySQLConnector, która posługując się tym kodem odpytuje bazę danych w celu jego weryfikacji.

8.2 Algorytm działania

Algorytm działania mikrokontrolera przedstawiony jest poniżej.



Ilustracja 8.2.1: Algorytm działania mikrokontrolera PicoBlaze

Program główny znajduje się w pliku `program.psm`. Po inicjalizacji następuje ustaleniu początkowych wartości rejestrów. W etykiecie `IR_petla1` wpisywana jest do rejestru `s7` wartość dziesiętna 8 co odpowiada 8 znakom wpisywanym na terminalu. W `IR_petla2` zerowane są rejestry `sA`, `sB`, `sC` przechowujące odebrane dane z jednej ramki kodu RC-5. Większość czasu PicoBlaze oczekuje na pojawienie się danych na porcie podczerwieni pod etykietą `.IR_czekaj`. Gdy pojawi się transmisja na porcie podczerwieni, co rozpoczyna się odczytaniem stanu niskiego, program przechodzi kolejno do etykiet `IR_rx1`, `IR_rx2`, `IR_rx3`, które odczytują poszczególne segmenty ramki kodu RC-5 opisanego w **praca przejściowa[]** i zapisują je kolejno do rejestrów `sA`, `sB`, `sC`. Rejestr `sA` zawiera bit startu oraz tzw. „turn around bit”, który przechodzi w stan przeciwny z każdym naciśnięciem przycisku na pilocie. Jeżeli przycisk wciśnięty jest na stałe to bit ten się nie zmienia co jest wygodne np. gdy chcemy zmienić poziom głośności w telewizorze. W `sB` znajduje się odebrany adres urządzenia docelowego, nie jest brany pod uwagę w naszej aplikacji, a rejestr `sC` to kod odebranego przycisku.

Po sprawdzeniu danych przez filtr zakłóceń pod etykietą `'Filter_zero'` dane z rejestru `sC` konwertowane są na standard ASCII oraz wysyłane do LPC i wyświetlane na LCD w podprogramie `'convert_send'` znajdującym się w pliku `banners_converts_send.psm`. Wartości na jakie konwertowane są przyciski przedstawia poniższa tabela.

PRZYCISK	KOD	ASCII hex
0	3F	30
1	3D	31
2	3B	32
3	39	33
4	37	34
5	35	35
6	33	36
7	31	37
8	2F	38
9	2D	39
Page-	29	bez konwersji ('-')
Page+	2B	bez konwersji ('+')

Tabela 8.1: Kodowanie przycisków z klawiatury pilota

Przyciski „Page+” i „Page-” odpowiadające za wskazanie kierunku przejazdu nie muszą być konwertowane, gdyż ich wartość może być bezpośrednio interpretowana w aplikacji LPCToMySQLConnector.

Następnie dekrementowana jest wartość rejestru s7, który jest licznikiem odebranych znaków. Gdy po dekrementacji będzie ona równa zero to flaga Z PicoBlaze'a zostanie ustawiona w stan niski. Jeżeli jeszcze nie będzie ona równa się zero program wraca do odczytywania kolejnego znaku skacząc do etykiety IR_petla2.

Gdy odebrane zostanie 8 znaków program nie skacze do odczytywania kolejnych znaków lecz przechodzi dalej. Pod etykietą LPC_THREAD oczekuje na dane zwrotne z LPC. Gdy pojawią się one wysyłane jest potwierdzenie odbioru pod etykietą LPC_DV_down oraz wywoływany jest podprogram which_banner znajdujący się w pliku banners_convers_send.psm, który decyduje jaką informację dla użytkownika wyświetlić na LCD, a przede wszystkim przekazać informację czy może on przejechać przez szlaban czy nie.

Jeżeli osoba ma zezwolenie na przejazd program oczekuje na odebranie potwierdzenia, że pojazd przejechał przez szlaban co zasymulowane jest poprzez odbiór jakiegokolwiek znaku na porcie podczerwieni w etykiecie w IR_czekaj_na_przejazd. Gdy już zostanie odebrane potwierdzenie przejazdu PicoBlaze wysyła do LPC potwierdzenie o tym, że pojazd znalazł się na parkingu co jest istotne z punktu widzenia przetwarzania danych w systemie bo samo wpisanie kodu na terminalu nie powoduje zarejestrowania wjazdu samochodu na parking, a jedynie przekazanie informacji o możliwości przejazdu i otwarciu szlabanu.

Jeżeli osoba nie ma zezwolenia na przejazd to na wyświetlaczu LCD wyświetlany jest powód odmowy. Po tych zabiegach program wraca do swojej pętli głównej. Najpierw z podprogramu which_banner, a następnie skokiem bezwarunkowym do etykiety IR_petla1.

Wykorzystanie rejestrów PicoBlaze przedstawione jest w poniższej tabeli.

s0	Opóźnienie programowe
s1	Opóźnienie programowe
s2	Opóźnienie programowe
s3	Odczyt danych z portu podczerwieni
s4	Licznik bitów odebranych z podczerwieni
s5	Odbieranie i wysyłanie danych przez UART, wyświetlenie baneru
s6	Filtr danych z portu podczerwieni, wyświetlenie baneru
s7	Licznik ramek odebranych z portu podczerwieni
s8	Konwersja na znak ASCII
s9	Konwersja na znak ASCII
sA	Ramka danych RC-5, wyświetlenie baneru
sB	Ramka danych RC-5, wyświetlenie baneru
sC	Ramka danych RC-5
sD	Konwersja na znak ASCII
sE	Konwersja na znak ASCII
sF	Wysyłanie danych do LPC i na LCD

Tabela 8.2: Opis wykorzystania rejestrów procesora PicoBlaze

8.3 Wyświetlanie wiadomości użytkownikowi

Na płycie czołowej E2LP znajduje się wyświetlacz LCD, który doskonale nadaje się do prezentacji użytkownikowi informacji. Poniższe zdjęcie przedstawia ową płytę czołową na, której zaznaczone jest umiejscowienie odbiornika podczerwieni oraz alfanumeryczny wyświetlacz LCD.



**Odbiornik
podczerwieni**

Ilustracja 8.3.1: Widok panelu czołowego E2LP z przykładowym banerem na LCD oraz zaznaczonym umiejscowieniem odbiornika podczerwieni

Liczba znaków wyświetlana na tym LCD może najwyżej wynosić 32. Informacja na powyższym zdjęciu wyświetlana jest zanim zostanie wpisany kod. Po wpisaniu go użytkownik oczekuje na wiadomość zwrotną, której treść jak i zachowanie szlabanu może być różne w zależności od wyniku weryfikacji, której dokonuje wspomniana aplikacja LCPToMySQLConnector. Przewidziane są następujące wiadomości zwrotne wyświetlane przez LCD.

- "Proszę przejeżdżać" – kiedy podany kod jest poprawny i użytkownik ma zezwolenie na przejazd przez szlaban, który w momencie tym zaczyna się podnosić.
- "Niepoprawny format kodu" – kiedy wpisany kod zawiera błędne znaki np. wśród pierwszych 7 znaków znajdują się znaki określające kierunek przejazdu.
- "Nieważny kod" – kiedy wprowadzony kod nie został odnaleziony w bazie.
- "Twój abonament nie jest ważny" – kiedy kod zostanie odnaleziony w bazie, ale osoba do której jest on przypisany nie uregulowała finansowo swojej możliwości wjazdu.
- "Kod w użyciu" – kiedy dany kod jest poprawny i osoba ma możliwość wjazdu, ale kod ten jest już wykorzystany.
- "Inne Twoje auto już wjechało" – kiedy użytkownik ma uprawnienia do wjazdu, ale jedno jego auto już znajduje się na parkingu. Obecnie system pozwala wjeżdżać tylko jednym pojazdem. Nie dotyczy to jednak samochodów specjalnych.
- „Ta osoba lub auto już wjechały” - kiedy osoba, która wjechała już na parking próbuje ponownie wjechać używając innego kodu. Inną sytuacją gdy ten komunikat może się wyświetlić nastąpi gdy użytkownik próbuje wjechać na parking samochodem, który już się tam znajduje ale używając innego kodu.
- "Błąd wewnętrzny Przepraszamy" – kiedy nastąpi błąd w przetwarzaniu danych przez aplikację LCPToMySQLConnector.
- "Zakaz wyjazdu" – kiedy użytkownik próbuje wyjechać z parkingu lecz z jakiegoś powodu nie jest mu to zabronione.

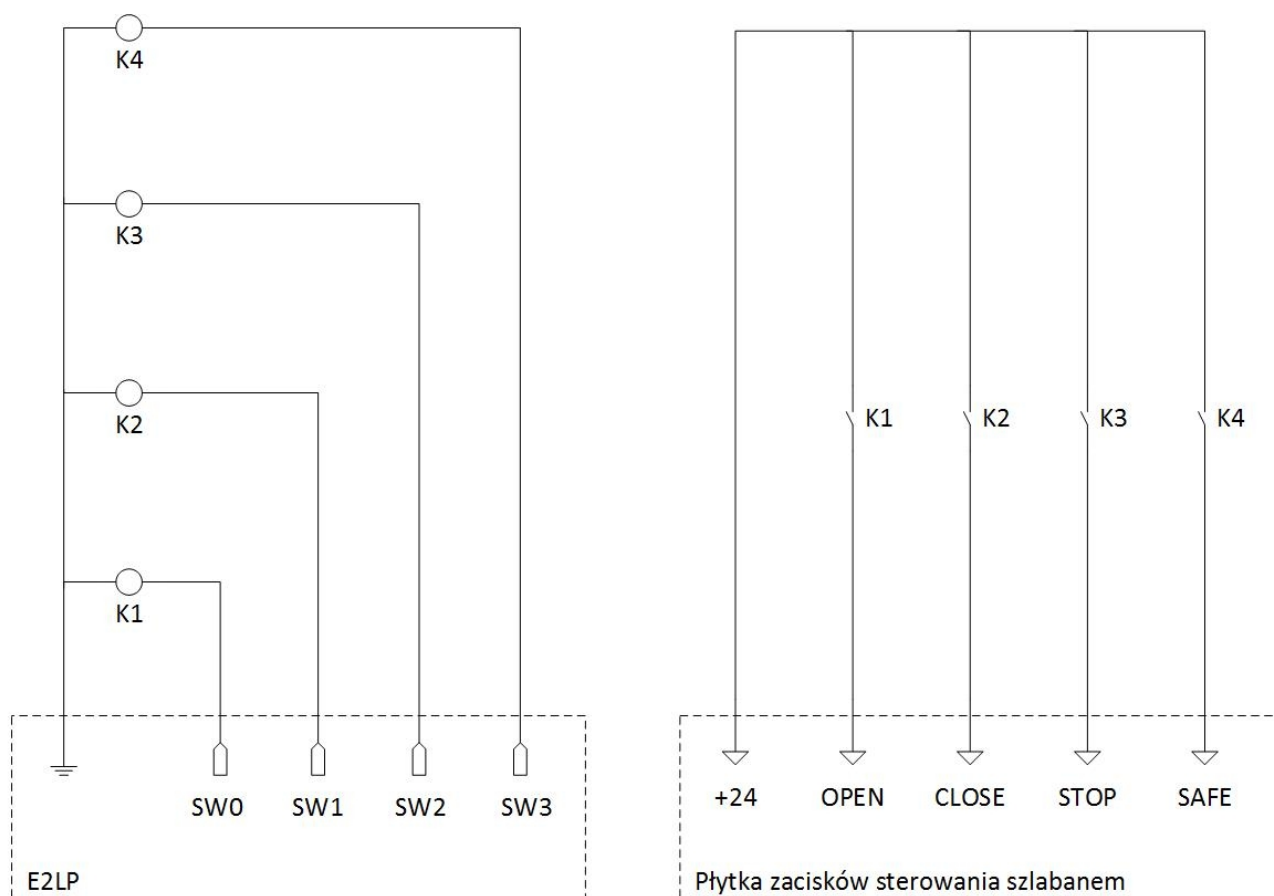
9. Integracja szlabanu automatycznego 615 BPR z systemem

Szlaban jest jedynym urządzeniem wykonawczym projektowanego systemu. Korzystając z dokumentacji[] szlabanu automatycznego 615 BPR zainstalowanego przy wjeździe do parkingu wydziału mechatroniki można zaprojektować układ sterowania. Powyższy szlaban wykorzystuje w sumie 4 sygnały.

- Open/Otwórz – normalnie otwarty,
- Close/Zamknij – normalnie otwarty,
- Stop – normalnie zamknięty,
- Safe – normalnie zamknięty.

Działanie tych sygnałów jest zależne od aktywnego trybu sterowania. Jest ich w sumie 7, a opis tych trybów oraz funkcje poszczególnych sygnałów zawiera się z rozdziale „Zespół sterujący szlabanu” na stronie 10 dokumentacji.

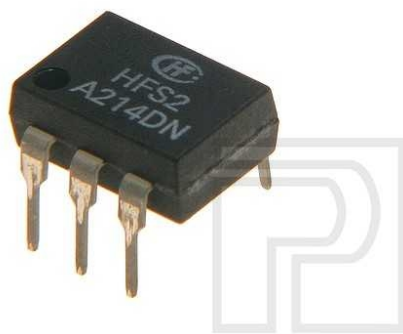
Z platformy E2LP możliwe jest odczytywanie i wysyłanie sygnałów binarnych za pomocą Snapwire connectors. Do sterowania szlabanem potrzebny będzie jednak stopień pośredni, którego rolę mogą pełnić przekaźniki. Poniżej schemat układu sterowania.



Ilustracja 9.1: Schemat przyłączenia sygnałów sterujących między E2LP a szlabanem

Lewa strona schematu przedstawia połączenia zasilania cewek elektromagnesów przekaźników. Prawa natomiast ich elementy wykonawcze – styki. Są one połączone z zaciskami z płytki sterowania szlabanem. Styki połączone z zaciskami STOP i SAFE są normalnie zamknięte tak jak te sygnały co wynika z dokumentacji szlabanu [1].

Jako przekaźniki wykorzystane mogą być HFSA-A21-4DN oferowane przez firmę Piekarz Części Elektroniczne.



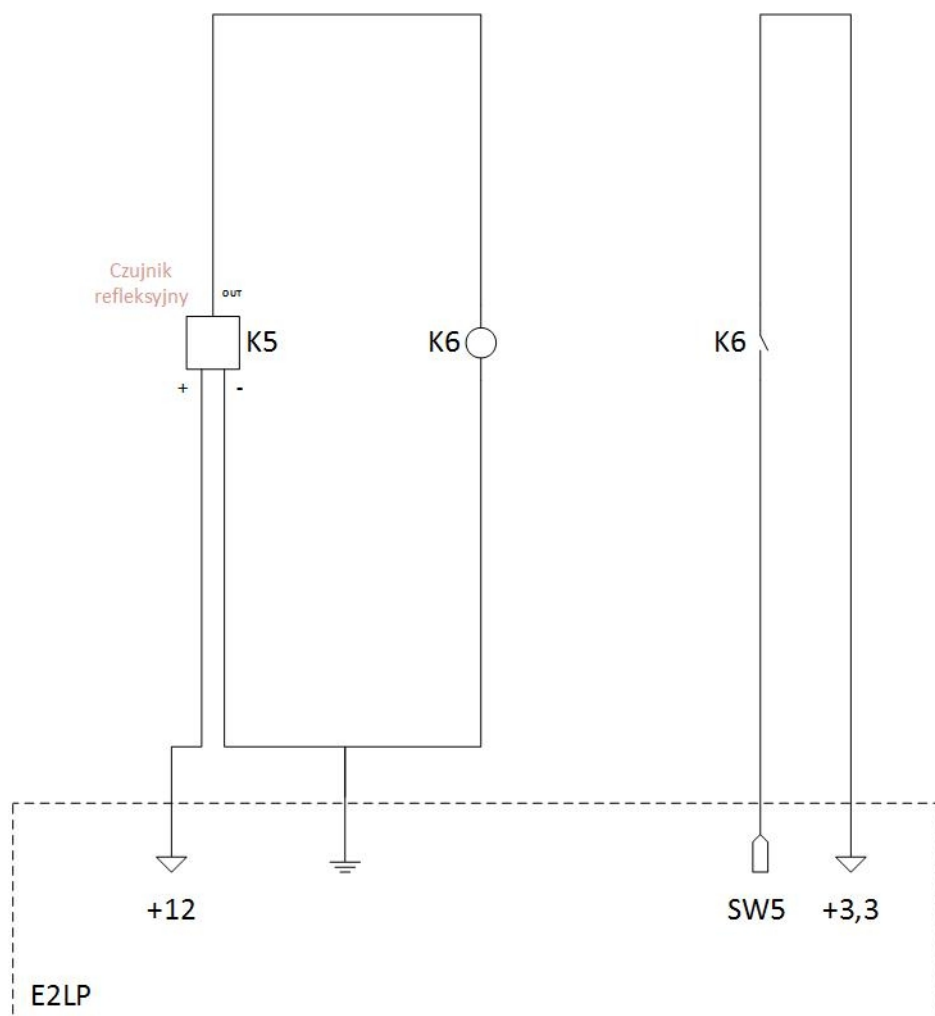
Ilustracja 9.2: Przekaźnik półprzewodnikowy

Napięcie sterujące tego przekaźnika wynosi 3V, prąd operacyjny to 5mA, więc bez problemu może być zasilana z wyjścia układu Spartan6, który operuje na napięciach poziomu TTL/CMOS (3,3 V, 24mA obciążenie pinu). Dokładna specyfikacja tego przekaźnika znajduje się w dokumentacji **f_02989.pdf**.

W programie mikrokontrolera PicoBlaze przewidziany jest krok w którym oczekuje on na informację zwrotną o tym czy pojazd przejechał. Funkcję tę pełnić będzie czujnik optyczny refleksyjny. Nadajnik i odbiornik takiego czujnika umieszczone są we wspólnej obudowie. Równoległe osie optyczne nadajnika i odbiornika skierowane są w końcowy punkt zasięgu, w którym jest umieszczony specjalny reflektor odblaskowy (lusterko). Wiązka promieni świetlnych wysyłana przez nadajnik po odbiciu od reflektora wraca do odbiornika. Przesłonięcie wiązki promieni świetlnych przez obiekt powoduje przerwanie transmisji i przełączenie obwodu wyjściowego czujnika.

Rolę czujnika pełnić będzie TOR30-4000R oferowany przez TWT Automatyka. Jego specyfikacja opisana jest w dokumentacji **odbiciowe-too_10.pdf**. Współpracować on będzie z przekaźnikiem półprzewodnikowym CX380D5 firmy CRYDOM.

Schemat połączenia czujnika z przekaźnikiem przedstawiony jest poniżej.



Ilustracja 9.3: Schemat podłączenia czujnika optycznego do E2LP

10. Model systemu oraz testy

11. Podsumowanie oraz wnioski

Cele oraz założenia pracy zostały zrealizowane. Dokonana została w niej integracja różnych podsystemów. Stopień złożoności pracy najlepiej określa liczba języków w jakich były programowane jej elementy. Mikrokontroler w języku C, układ FPGA Spartan6 w VHDL, procesor softwarowy PicoBlaze w Assemblerze, aplikacja pośrednicząca w komunikacji pomiędzy mikrokontrolerem a bazą danych w Javie, do obsługi relacyjnej bazy danych wykorzystany został SQL, do stworzenia witryny internetowej zgodnej z założeniami wykorzystane zostały HTML, PHP oraz JavaScript. Daje to w sumie 8 języków programowania. Na dodatek są one niekiedy bardzo odległe od siebie pod względem zastosowań. Z jednej strony język opisu sprzętu VHDL oraz bezpośrednio tłumaczony na kod maszynowy Assembler. Z drugiej wysokopoziomowe języki jak Java, ale również interpretowane, skryptowe PHP i JavaScript.

Praca pozwoliła także autorowi na poznanie wielu nowych zagadnień z zakresu programowania. Oprócz utrwalenia znanej już z pracy przejściowej techniki programowania układów FPGA pozwoliła na zdobycie doświadczenia w używaniu (programowanie oraz debugowanie) mikrokontrolera z procesorem w architekturze ARM. Integracja mikrokontrolera z kartą sieciową wymagało rzetelnego korzystania z dokumentacji tych układów co także było kształcące.

Bardzo dużą część pracy stanowiło tworzenie witryny internetowej. Podstawą merytoryczną dla autora przy tej części pracy był przedmiot Techniki i Bezpieczeństwo w Internecie prowadzony przez dr inż. Pawła Wnuka. Wykonanie witryny jak również scharakteryzowanie systemów informatycznych we wstępie pozwoliło znacznie pogłębić wiedzę o działaniu nowoczesnych systemów informatycznych. Istotną częścią było również sporządzenie wstępnych projektów: całego systemu, ale również w późniejszych etapach bazy danych, witryny internetowej, sposobu integracji sprzętowej i programowej komponentów oraz algorytmów działania. Pozwoliło to na przećwiczenie i utrwalenie analitycznego sposobu myślenia.

Poza osobistymi korzyściami dla autora należy podkreślić, że w części sprzętowej praca ta jako pierwsza na wydziale mechatroniki umożliwiła integrację platformy E2LP w sieci LAN. Sporządzona w wyniku tego dokumentacja z pewnością przyda się osobom chcącym wykorzystać platformę w ten sposób. Dodatkowo praca była pierwszą, która opisała wykorzystanie płyty rozszerzającej do E2LP z oddzielnie programowanym mikrokontrolerem.

Spis ilustracji

Ilustracja 1.2.1: Topologia magistrali.....	7
Ilustracja 1.2.2: Topologia pierścienia.....	7
Ilustracja 1.2.3: Topologia gwiazdy.....	8
Ilustracja 1.2.4: Topologia siatki.....	8
Ilustracja 1.3.1: Model OSI.....	11
Ilustracja 1.3.2: Ramka Ethernet.....	12
Ilustracja 1.3.3: Zobrazowanie ramki protokołu IP.....	13
Ilustracja 1.3.4: Zobrazowanie ramki protokołu TCP.....	15
Ilustracja 3.2.1: Panel czołowy parkomatu.....	25
Ilustracja 4.4.1: Schemat ideowy systemu.....	29
Ilustracja 4.5.1: Schemat blokowy funkcjonalny.....	31
Ilustracja 5.3.1: Schemat sygnałów komunikacyjnych.....	37
Ilustracja 5.3.2: Ramka protokołu MDIO.....	39
Ilustracja 6.2.1: Model związków encji projektowanej bazy danych.....	44
Ilustracja 6.2.2: Model fizyczny bazy.....	45
Ilustracja 6.3.1: Schemat przejść pomiędzy ekranami.....	47
Ilustracja 6.3.2: Szablon pojedynczej strony witryny.....	48
Ilustracja 6.4.1: Strona logowania witryny.....	50
Ilustracja 6.4.2: Formularz rejestracji nowego użytkownika.....	52
Ilustracja 6.4.3: Strona zarządzania pojazdami, gdy użytkownik nie jest zalogowany.....	53
Ilustracja 6.4.4: Strona zarządzania pojazdami po zalogowaniu.....	54
Ilustracja 6.4.5: Formularz dodawania nowego pojazdu.....	54
Ilustracja 6.4.6: Algorytm transakcji dodania nowego pojazdu.....	58
Ilustracja 6.4.7: Przykładowy widok tabeli aut użytkownika, których jest właścicielem i wjeżdża nimi w ramach swojego abonamentu.....	59
Ilustracja 6.4.8: Przykładowy widok tabeli aut, których użytkownik nie jest właścicielem, ale może nimi wjeżdżać w ramach swojego abonamentu.....	59
Ilustracja 6.4.9: Przykładowy widok tabeli aut, których użytkownik jest właścicielem i udostępnił je innym użytkownikom w ramach ich abonamentu.....	59
Ilustracja 6.4.10: Przykładowy widok tabeli aut, na które możliwość wjazdu użytkownik dostał od innych użytkowników.....	60
Ilustracja 6.4.11: Przykładowy widok tabeli aut, na które możliwość wjazdu użytkownik udostępnił innym użytkownikom w ramach swojego abonamentu.....	60
Ilustracja 6.4.12: Formularz do wyszukiwania osób.....	62
Ilustracja 6.4.13: Informacja o udostępnieniu pojazdu.....	63
Ilustracja 6.4.14: Informacja o niepowodzeniu operacji z możliwą przyczyną.....	63
Ilustracja 6.4.15: Formularz do wyszukiwania osób oraz przeglądania ich pojazdów.....	63
Ilustracja 6.4.16: Przykład wyświetlenia mapy na stronie „Wskazówki Dojazdu”.....	64
Ilustracja 7.4.1: Algorytm odpytywania bazy danych gdy użytkownik próbuje wjechać.....	71
Ilustracja 7.4.2: Algorytm odpytywania bazy danych gdy użytkownik próbuje wyjechać.....	72
Ilustracja 7.5.1: Algorytm dodawania logu wjazdu lub wyjazdu.....	73
Ilustracja 8.1.1: Pilot emulujący terminal wjazdowy z opisem przycisków.....	74
Ilustracja 8.2.1: Algorytm działania mikrokontrolera PicoBlaze.....	75
Ilustracja 8.3.1: Widok panelu czołowego E2LP z przykładowym banerem na LCD oraz zaznaczonym umiejscowieniem odbiornika podczerwieni.....	77
Ilustracja 9.1: Schemat przyłączenia sygnałów sterujących między E2LP a szlabanem.....	79
Ilustracja 9.2: Przekaznik półprzewodnikowy.....	80

Ilustracja 9.3: Schemat podłączenia czujnika optycznego do E2LP.....	81
--	----

Spis tabel

Tabela 3.1: Zestawienie porównawcze rozwiązań parkingowych.....	30
Tabela 5.1: Zestawienie pinów mikrokontrolera LPC2368 odpowiedzialnych za komunikację z urządzeniem PHY.....	38
Tabela 6.1: Zestawienie plików zaimplementowanych na serwerze będących częścią witryny.....	51
Tabela 8.1: Kodowanie przycisków z klawiatury pilota.....	78
Tabela 8.2: Opis wykorzystania rejestrów procesora PicoBlaze.....	79