PROTOTYP SYSTEMU DO REJESTRACJI MIEJSC PARKINGOWYCH

european student magazine
Numer 1 / 2025

Mateusz **Mindewicz**[1], Aleksandra **Sikora**[2*]

[1] Politechnika Świętokrzyska, Wydział Elektrotechniki, Automatyki i Informatyki, al. Tysiąclecia Państwa Polskiego 7, 25-314 Kielce, Polska, s092664@student.tu.kielce.pl
[2] Politechnika Świętokrzyska, Wydział Elektrotechniki, Automatyki i Informatyki, al. Tysiąclecia Państwa Polskiego 7, 25-314 Kielce, Polska, asikora@tu.kielce.pl
[*] Corresponding author

Streszczenie – W artykule omówiono funkcjonalności i zbudowanie autorskiego prototypu systemu parkingowego do monitorowania dostępności miejsc parkingowych. Wykorzystano technologie Flutter, Flask oraz narzędzia Raspberry Pi 4, ESP32 i IR Sensor. W ramach pracy powstała strona internetowa oraz aplikacja na urządzenia mobilne do zarządzania sesjami parkingowymi. Zabezpieczenie danych zrealizowano z wykorzystaniem tokenów JWT oraz haszowania haseł. Przeprowadzone testy backendu oraz aplikacji potwierdziły poprawność działania prototypu.

Słowa kluczowe – ESP32, IR Sensor, system parkingowy, Raspberry Pi 4, Flutter

WSTĘP

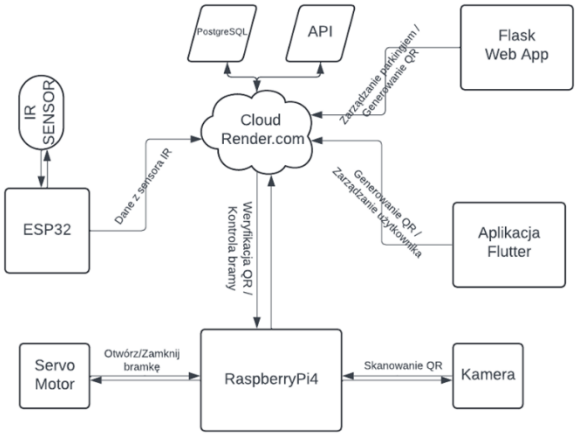
Rozwój technologii Internetu Rzeczy (IoT) umożliwia tworzenie zaawansowanych systemów, takich jak inteligentne parkingi, które dzięki czujnikom i mikrokontrolerom ułatwiają znalezienie wolnych miejsc i efektywne zarządzanie ruchem. Odpowiadają one na problemy związane z urbanizacją, rosnącą liczbą pojazdów i zanieczyszczeniem powietrza w miastach.

W artykule opisano prototyp systemu parkingowego pozwalającego na sprawdzanie dostępności miejsc, historię sesji oraz generowanie kodów QR do wjazdu.

System opiera się na technologii Flask [1-2], Flutter [3], Render.com [4], PostgreSQL [5] oraz urządzeniach Raspberry Pi 4 [6], ESP32 [7] i sensorach IR [8]. Użytkownicy korzystają z niego przez stronę internetową lub aplikację.

I. SCHEMAT POŁĄCZEŃ W SYSTEMIE

Schemat przedstawia system parkingowy oparty na technologii chmurowej, IoT oraz aplikacjach, mobilnej i webowej. ESP32 z sensorem IR monitoruje status miejsc parkingowych, a dane przesyłane są do chmury Render.com. Chmura obliczeniowa przetwarza dane, przechowuje je w bazie PostgreSQL i obsługuje aplikacje oraz komunikację z RPi4 zarządzającą wjazdem i wyjazdem. Backend API umożliwia rejestrację użytkowników, generowanie kodów QR oraz zarządzanie miejscami i sesjami parkingowymi. Raspberry Pi 4 skanuje kody QR i steruje bramą. Aplikacja mobilna wyświetla dostępne miejsca, generuje kody QR i łączy się z serwerem, a aplikacja webowa pozwala zarządzać systemem, integrując się z bazą danych. System zapewnia efektywne zarządzanie parkingiem.

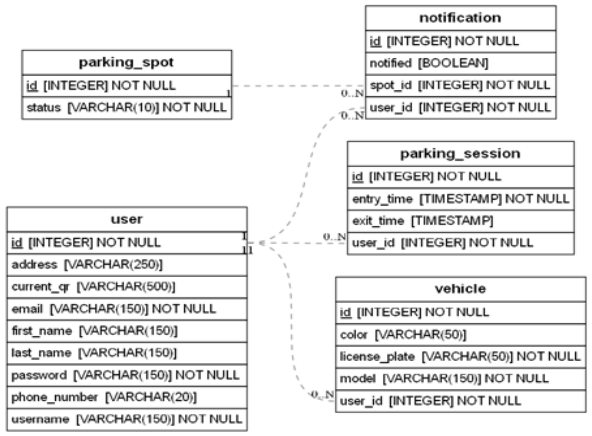


Rysunek 1. Schemat projektu

II. MODELE I SCHEMAT BAZY DANYCH

Struktura bazy danych systemu parkingowego składa się z pięciu tabel: USER, VEHICLE, PARKING_SPOT, PARKING_SESSION i NOTIFICATION. Tabela USER przechowuje dane użytkowników (np. imię, email, hasło) i jest powiązana z tabelami VEHICLE, PARKING_SESSION i NOTIFICATION, umożliwiając przypisanie jednemu użytkownikowi wielu pojazdów, sesji parkingowych i powiadomień. VEHICLE zawiera informacje o pojazdach użytkowników i ma relację wiele do jednego z USER. PARKING_SPOT przechowuje status miejsc parkingowych (wolne/zajęte) i jest połączona z NOTIFICATION, umożliwiając przypisanie powiadomień do miejsc

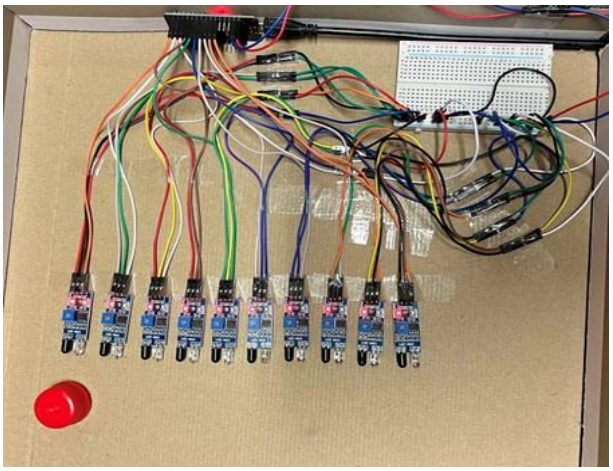
parkingowych. PARKING_SESSION gromadzi dane o sesjach parkingowych, z relacją wiele do jednego z USER, a NOTIFICATION zarządza powiadomieniami, łącząc dane użytkowników i miejsc parkingowych. Całość umożliwia śledzenie dostępności miejsc parkingowych, zarządzanie użytkownikami i ich pojazdami oraz wysyłanie powiadomień, zapewniając integralność danych dzięki relacjom i kluczom.



Rysunek 2. Schemat ERD Bazy Danych

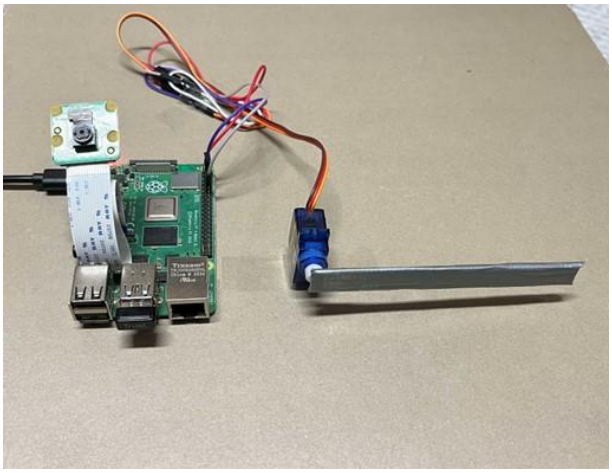
III. PROJEKT SYSTEMU

ESP32 zbiera dane z czujników IR monitorujących status miejsc parkingowych. Czujniki podłączone do wyprowadzeń cyfrowych ESP32 wykrywają przeszkody, generując sygnał HIGH lub LOW. Zebrane dane są przekształcane w parametry URL i przesyłane na serwer w chmurze. Po połączeniu z siecią Wi-Fi ESP32 cyklicznie aktualizuje informacje o stanie miejsc parkingowych.



Rysunek 3. Czujniki IR podłączone do ESP32

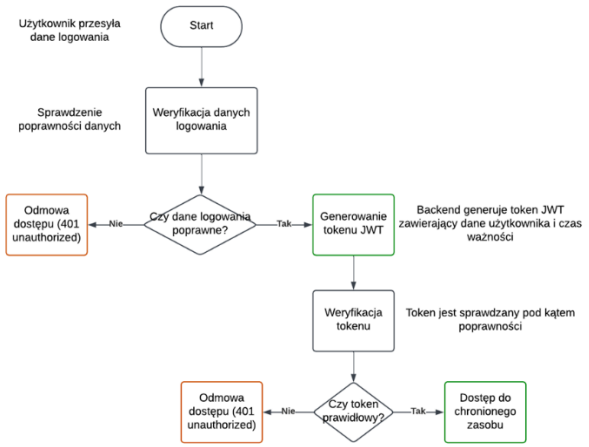
Raspberry Pi 4 odpowiada w projekcie za sterowanie serwomotorem, odczytywanie kodów QR przy użyciu podłączonej kamery oraz przesyłanie danych na serwer w chmurze. We wcześniejszej wersji projektu RPi4 służył jako agregator informacji. Na tym mini komputerze był wdrożony serwer oraz baza danych, jednakże te funkcje zostały przeniesione do chmury.



Rysunek 4. Kamera oraz serwomotor podłączone do Raspberry Pi 4

IV. SYSTEM AUTORYZACJI I LOGOWANIA

System oparty na Flask działa zarówno dla aplikacji webowej, jak i mobilnej, zapewniając wygodną obsługę użytkowników. Flask umożliwia generowanie stron i dostarczanie informacji z bazy danych przy użyciu funkcji render_template. Dodatkowo API zwraca dane w formacie JSON, co umożliwia korzystanie z aplikacji mobilnej. Serwer wykorzystuje bazę PostgreSQL oraz obsługuje logowanie i sesje użytkowników za pomocą Flask-Login i JWT. Token JWT [9] pozwala na autoryzację żądań API, eliminując potrzebę utrzymywania sesji na serwerze. Po zalogowaniu użytkownik otrzymuje token, który przechowuje lokalnie w aplikacji mobilnej i przesyła w nagłówkach kolejnych żądań API. Serwer weryfikuje token, sprawdzając poprawność podpisu, istnienie użytkownika oraz ważność tokenu. W przypadku nieautoryzowanego dostępu aplikacja zwraca komunikat o błędzie (401 Unauthorized). Schemat działania obejmuje kilka kluczowych etapów: weryfikację danych logowania, generowanie tokenu JWT, przesyłanie tokenu w żądaniach API oraz weryfikację tokenu przed przyznaniem dostępu do chronionych zasobów. Dzięki temu system obsługuje użytkowników zarówno korzystających ze strony internetowej, jak i aplikacji mobilnej, zapewniając bezpieczeństwo i efektywność działania

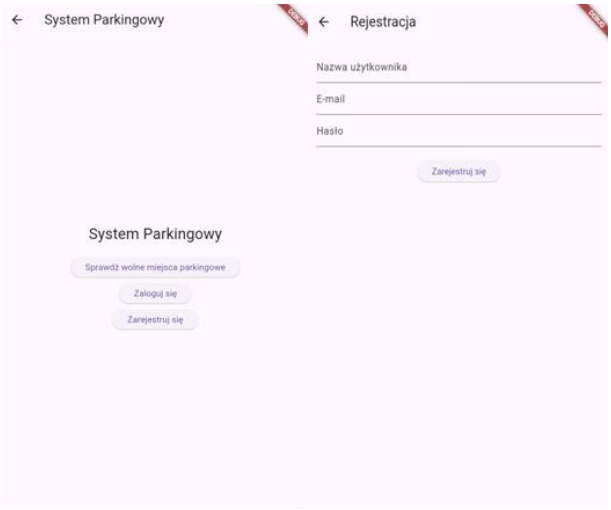


Rysunek 5. Schemat blokowy funkcjonalności systemu

V. PREZENTACJA APLIKACJI

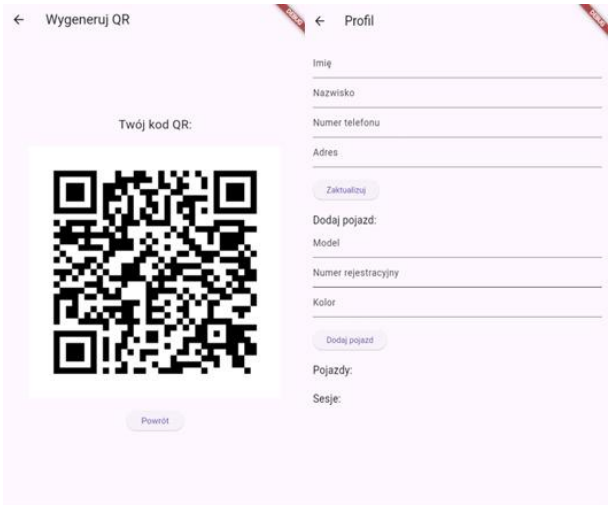
Aplikacja mobilna i strona internetowa systemu parkingowego zostały zaprojektowane tak, aby oferować identyczny zakres funkcji, co zapewnia spójność działania oraz umożliwia korzystanie z systemu na różnych urządzeniach. Dzięki temu użytkownicy mogą wybrać najdogodniejsze dla siebie środowisko – aplikację mobilną lub stronę internetową – przy zachowaniu pełnego dostępu do kluczowych funkcjonalności.

System umożliwia m.in. korzystanie ze strony startowej, rejestrację i logowanie, podgląd statusu miejsc parkingowych, zarządzanie danymi osobowymi i pojazdami, przegląd historii sesji parkingowych oraz generowanie i skanowanie kodów QR. Takie podejście zapewnia wygodę użytkowania, uniwersalność i łatwy dostęp do wszystkich możliwości systemu, niezależnie od używanego urządzenia.



Rysunek 6. Główny ekran oraz panel rejestracji

Rysunek 7. Status miejsc oraz sesje parkingowe



Rysunek 8. Generowanie kodu QR oraz profil użytkownika

VI. WNIOSKI

W ramach projektu opracowano prototyp systemu parkingowego, który obejmuje schemat działania, implementację oprogramowania oraz integrację z chmurą. System korzysta z Raspberry Pi 4 do obsługi wjazdu i wyjazdu, ESP32 i sensorów IR do monitorowania miejsc parkingowych oraz serwera opartego na Flask do generowania strony internetowej i API. Aplikacja mobilna została stworzona w Flutterze, a dane przechowywane są w bazie PostgreSQL.

Zrealizowano kluczowe funkcje, takie jak obsługa kodów QR, monitorowanie miejsc, powiadomienia o wolnych miejscach, rejestracja użytkowników, historia sesji parkingowych oraz integracja z chmurą. Dzięki dekoratorowi api_or_login_required system obsługuje zarówno użytkowników webowych, jak i mobilnych, zapewniając elastyczność i bezpieczeństwo. Generowanie kodów QR oparto na unikalnych identyfikatorach, a dane są szyfrowane i uwierzytelniane za pomocą JWT.

Testy potwierdziły poprawność działania systemu, w tym odczyty kodów QR, aktualizację statusu miejsc w czasie rzeczywistym oraz synchronizację między aplikacją, stroną i serwerem.

DESIGN AND IMPLEMENTATION OF A SYSTEM FOR RECORDING PARKING SPACES USING RASPBERRY PI, ESP32

The purpose of the study was to design, program the functionality and build a prototype parking system for monitoring parking space availability. The technologies used were Flutter, Flask and Raspberry Pi 4, ESP32 and IR Sensor tools. The work included a website and a mobile device application for managing parking sessions. Data security was implemented using JWT tokens and password hashing. Tests of the backend and the application confirmed the correctness of the prototype.

Key words: ESP32, IR Sensor, Parking system, Raspberry Pi 4

BIBLIOGRAFIA

[1] Dokumentacja Flask: https://flask.palletsprojects.com/en/stable/ [Data dostępu: 15.08.2024]
[2] M. Grinberg: Flask. Tworzenie aplikacji internetowych w Pythonie. Wydanie II. Wydawnictwo Helion, 10.03.2020
[3] Dokumentacja Flutter: https://docs.flutter.dev/ [Data dostępu: 15.08.2024]
[4] Dokumentacja Render.com: https://render.com/docs [Data dostępu: 15.08.2024]
[5] Dokumentacja PostgreSQL: https://www.postgresql.org/docs/ [Data ostatniej aktualizacji: 21.11.2024, Data dostępu: 20.12.2024]
[6] A. Gromczyński: W labiryncie IoT. Budowanie urządzeń z wykorzystaniem układów ESP8266 i ESP32. Wydawnictwo Helion, 11.12.2024
[7] S. Monk: Raspberry Pi. Receptury. Wydanie III. Wydawnictwo Helion, 21.08.2020
[8] D. Jost: What is an IR sensor, https://www.fierceelectronics.com/sensors/what-ir-sensor [Data utworzenia: 29.07.2019, Data dostępu: 07.01.2025]
[9] Dokumentacja JWT: https://jwt.io/introduction [Data dostępu: 09.01.2025]