POLITECHNIKA
RZESZOWSKA
im. IGNACEGO ŁUKASIEWICZA

Projektowanie systemów wbudowanych

Projekt

Parking samochodowy ze szlabanem

Maksymilian Celtner 169416

Spis treści

1. Cel i założenia projektu

Celem projektu było zaprojektowanie i wykonanie systemu mikroprocesorowego bazującego na mikrokontrolerze ESP32 i reprezentującego prostą makietę parkingu samochodowego. Wjazd na parking umożliwiać miał szlaban podnoszony i opuszczany przez serwomechanizm. Pierwotna wersja projektu zakładała podnoszenie się szlabanu po odczytaniu karty przez czytnik RFID oraz jego opuszczanie po wykryciu obiektu (samochodu) przez czujnik odległościowy umieszczony za szlabanem. Ze względu na nieprawidłowe działanie czytnika, pomysł niestety został porzucony. Nowa wersja projektu obejmowała zastąpienie czytnika RFID drugim czujnikiem odległościowy. Tym samym szlaban podnosić się miał po wykryciu samochodu chcącego wjechać na parking i opuszczać po przejeździe auta i wykryciu go przez drugi czujnik. Funkcjonalność szlabanu zakładała także działanie w odwrotnej sytuacji, kiedy to auto zamierzało opuścić parking. Do całości dodany miał zostać wyświetlacz OLED pokazujący informację o wolnych miejscach parkingowych.

W ramach projektu, w środowisku EasyEDA zaprojektowany miał zostać schemat ideowy wraz z obwodem drukowanym PCB. Rozwiązanie sprzętowe zrealizowane miało zostać na bazie płytki stykowej, bez wykonywania płytki PCB. Cel projektu zakładał napisanie i uruchomienie programu sterującego dla zmontowanego układu.

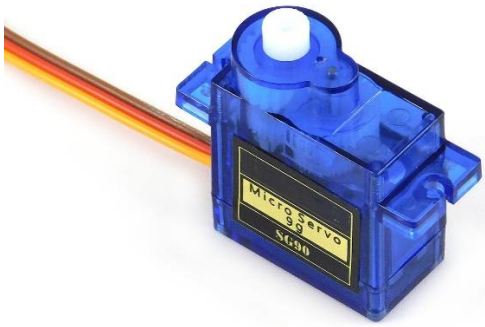
2. Potrzebne komponenty

W celu zrealizowania projektu zakupione zostały następujące elementy:

- ESP32 - mikrokontroler z wbudowanym Wi-Fi i Bluetooth, oferujący liczne interfejsy (GPIO, ADC, SPI, I2C), idealny do projektów IoT. W projekcie zarządza wszystkimi komponentami i logiką działania.

- Czujnik ultradźwiękowy HC-SR04 - czujnik odległości mierzący dystans w zakresie 2-400 cm z dużą dokładnością. W projekcie wykrywa pojazdy przed i za szlabanem.

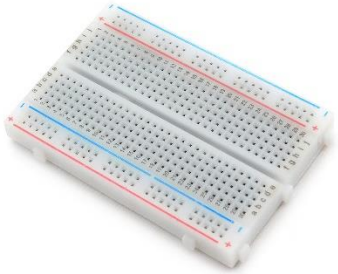
- Serwomechanizm SG90 - lekki serwomechanizm z kątem obrotu 180°, sterowany sygnałem PWM. W projekcie odpowiada za ruch szlabanu.



- Wyświetlacz OLED 0.96” – wyświetlacz o rozdzielczości 128x64 pikseli, komunikujący się przez I2C, umożliwia wyświetlanie liczby dostępnych miejsc i komunikatów parkingowych.

- Czytnik RFID RC522 - moduł odczytujący karty RFID, pierwotnie planowany do identyfikacji pojazdów w projekcie, ale nieużyty w finalnej wersji.

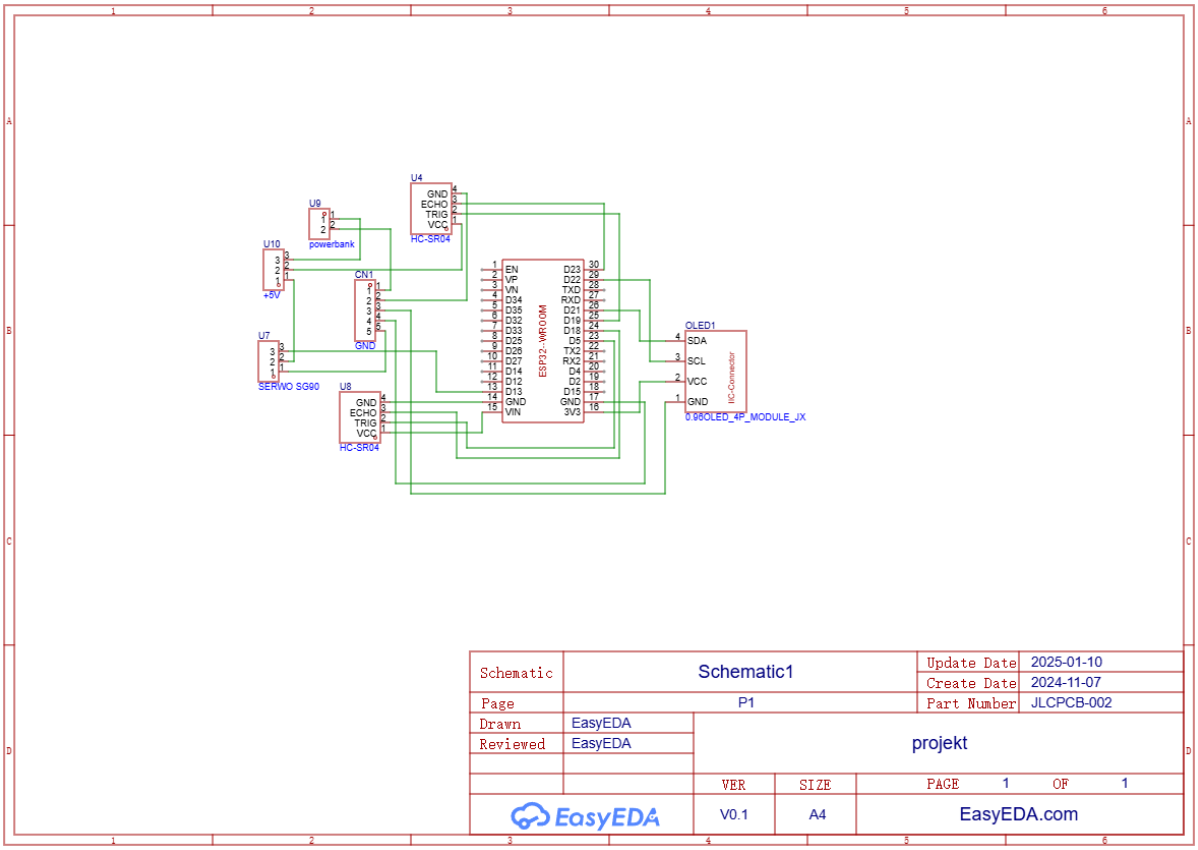
- Płytka stykowa - umożliwia bezlutowe połączenie komponentów, ułatwiając testowanie i budowę prototypu.



- Przewody połączeniowe - przewody Female-Female, Male-Male i Male-Female służą do elastycznego łączenia komponentów na płytce stykowej i mikrokontrolerze.

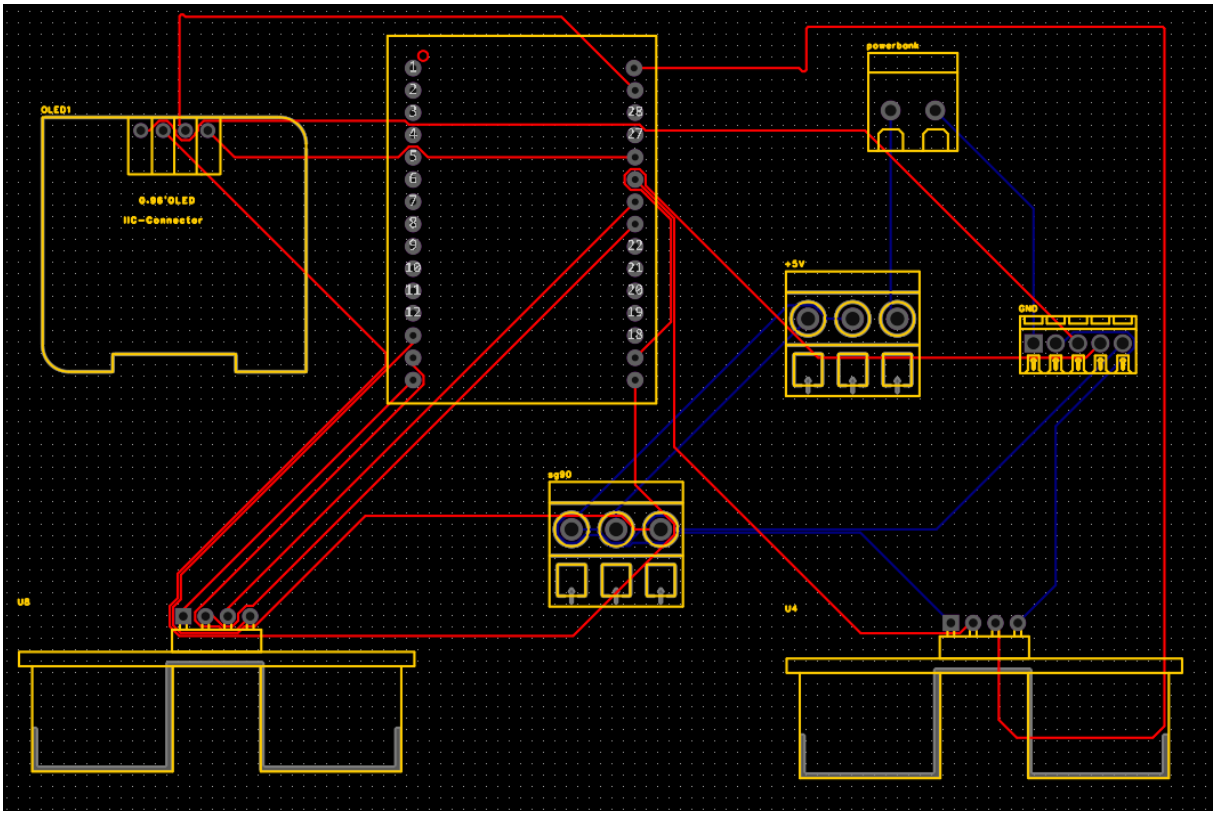
3. Schemat połączeń w EasyEDA

W środowisku EasyEDA zaprojektowany został schemat ideowy przedstawiony na poniższym rysunku. Serwomechanizm SG90 oraz jeden czujników odległościowych HC-SR04 to dwa elementy zasilane z zewnętrznego źródła 5V (powerbank). Do schematu dodane zostały 3 terminale przyłączeniowe reprezentujące powerbank (U9), sygnały plusowe, +5V (U10) oraz sygnały minusowe, GND (CN1). Wyświetlacz OLED zasilany jest z pinu 3V3 mikrokontrolera ESP32, a drugi czujnik odległościowy HC-SR04 z pinu VIN mikrokontrolera ESP32.



4. Projekt płytki PCB w EasyEDA

Na podstawie utworzonego schematu wygenerowana została płytka PCB. Dodane zostały wszystkie połączenia poszczególnych elementów. Czerwone ścieżki odpowiadają połączeniom na górnej warstwie płytki, zaś niebieskie ścieżki to połączenia na dolnej warstwie. Cały obwód drukowany widoczny jest na poniższym rysunku.



5. Implementacja oprogramowania

Poniżej zamieszczone i omówione zostały poszczególne fragmenty kodu sterującego utworzonego dla zmontowanego układu. Pełna wersja programu znajduje się w przesłanym katalogu.

Kod korzysta z bibliotek:

- *ESP32Servo.h*, odpowiadającej za sterowanie serwomechanizmami za pomocą PWM na ESP32
- *Wire.h*, odpowiedzialnej za obsługę komunikacji I2C
- *Adafruit_GFX* i *Adafruit_SSD1306*, służących do obsługi graficznego wyświetlacza OLED

trigPin i *echoPin* to piny używane przez czujniki HC-SR04 do wysyłania i odbierania fal ultradźwiękowych. Utworzony został obiekt serwomechanizmu sterujący ruchem szlabanu. Dodane zostały ustawienia wyświetlacza oraz parametry parkingu definiujące liczbę maksymalnych i aktualnych wolnych miejsc na parkingu.

```
const int trigPin1 = 19;  // Trig pierwszego czujnika (przed szlabanem)
const int echoPin1 = 23;  // Echo pierwszego czujnika (przed szlabanem)
const int trigPin2 = 5;   // Trig drugiego czujnika (za szlabanem)
const int echoPin2 = 18;  // Echo drugiego czujnika (za szlabanem)

Servo szlaban;
const int servoPin = 13;

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

const int maxMiejsca = 3; // Maksymalna liczba miejsc
int wolneMiejsca = maxMiejsca; // Liczba wolnych miejsc
```

Funkcja *setup()* odpowiada za konfigurację czujników, inicjalizację serwa oraz inicjalizację wyświetlacza (ustawienie OLED do wyświetlania liczby wolnych miejsc). Piny dla HC-SR04 ustawiane są jako wyjścia (Trig) i wejścia (Echo). Szlaban początkowo znajduje się w pozycji poziomej (zamkniętej).

```
void setup() {
  pinMode(trigPin1, OUTPUT);
  pinMode(echoPin1, INPUT);
  pinMode(trigPin2, OUTPUT);
  pinMode(echoPin2, INPUT);

  szlaban.attach(servoPin, 500, 2400); // Pin, minimalny impuls, maksymalny
impuls
  szlaban.write(0); // Szlaban w pozycji poziomej (zamkniętej)
```

```
  if (!display.begin(SSD1306_PAGEADDR, 0x3C)) {
    Serial.println(F("Błąd inicjalizacji OLED"));
    for (;;); // Zatrzymaj program w razie błędu
  }
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);

  Serial.begin(115200);
  displayStatus();
}
```

Poniżej przedstawiony został fragment kodu odpowiadający za główną logikę działania systemu parkingowego i sterowanie ruchem pojazdów przy wjeździe i wyjeździe.

```
void loop() {
   if (detectCar(trigPin1, echoPin1)) {
    if (wolneMiejsca > 0) { // Jeśli są wolne miejsca
      Serial.println("Auto wykryte przed szlabanem. Otwieram szlaban...");
      otworzSzlaban(); // Otwórz szlaban

      bool autoPrzejechalo = false;
      unsigned long startTime = millis(); // Licz czas
      while (millis() - startTime < 10000) { // Czekaj maksymalnie 10 sekund
        if (detectCar(trigPin2, echoPin2)) {
          autoPrzejechalo = true;
          break;
        }
      }

      if (autoPrzejechalo) {
        Serial.println("Auto przejechało przez szlaban.");
        wolneMiejsca--; // Zmniejsz liczbę wolnych miejsc
        displayStatus(); // Zaktualizuj wyświetlacz
      } else {
        Serial.println("Nie wykryto auta za szlabanem.");
      }

      zamknijSzlaban(); // Zamknij szlaban po przejechaniu auta
    } else {
      Serial.println("Brak wolnych miejsc. Szlaban nie otwiera się.");
      displayNoSpace(); // Wyświetl komunikat "Brak wolnych miejsc!" na 5
sekund
      delay(5000); // Opóźnienie przed kolejną pętlą
      displayStatus(); // Przywróć normalny komunikat na wyświetlaczu
    }
  }
```

```
  if (detectCar(trigPin2, echoPin2)) {
    Serial.println("Auto wykryte przed wyjazdem. Otwieram szlaban...");
    otworzSzlaban(); // Otwórz szlaban

    bool autoWyjechalo = false;
    unsigned long startTime = millis(); // Licz czas
    while (millis() - startTime < 10000) { // Czekaj maksymalnie 10 sekund
      if (detectCar(trigPin1, echoPin1)) {
        autoWyjechalo = true;
        break;
      }
    }

    if (autoWyjechalo) {
      Serial.println("Auto wyjechało z parkingu.");
      if (wolneMiejsca < maxMiejsca) { // Unikaj przekraczania maksymalnej
liczby miejsc
        wolneMiejsca++; // Zwiększ liczbę wolnych miejsc
        displayStatus(); // Zaktualizuj wyświetlacz
      }
    } else {
      Serial.println("Nie wykryto auta wyjeżdżającego przez pierwszy
czujnik.");
    }

    zamknijSzlaban(); // Zamknij szlaban po wyjeździe auta
  }

  delay(100); // Krótka przerwa w pętli
}
```

W pierwszej części pętli sprawdzany jest stan pierwszego czujnika (umieszczonego przed szlabanem), który wykrywa pojazdy próbujące wjechać na parking. Jeśli pierwszy czujnik wykryje obecność pojazdu, a liczba wolnych miejsc parkingowych jest większa od zera, szlaban zostaje podniesiony. Następnie system czeka maksymalnie 10 sekund, aż pojazd przejedzie przez drugi czujnik (umieszczony za szlabanem). Jeśli drugi czujnik potwierdzi przejazd pojazdu, liczba wolnych miejsc zostaje zmniejszona o jeden, a wyświetlacz OLED aktualizuje informację o dostępnych miejscach. W przypadku, gdy pojazd nie zostanie wykryty przez drugi czujnik w wyznaczonym czasie, system zamyka szlaban bez zmiany stanu liczby wolnych miejsc. Jeśli na parkingu nie ma wolnych miejsc, szlaban pozostaje zamknięty, a na wyświetlaczu pojawia się komunikat "Brak wolnych miejsc!" przez 5 sekund.

W drugiej części pętli sprawdzany jest stan drugiego czujnika (umieszczonego za szlabanem), który wykrywa pojazdy opuszczające parking. Jeśli drugi czujnik wykryje pojazd, szlaban zostaje podniesiony, a system czeka maksymalnie 10 sekund na potwierdzenie wyjazdu przez pierwszy czujnik (umieszczony przed szlabanem). Jeśli pierwszy czujnik potwierdzi, że pojazd opuścił parking, liczba wolnych miejsc zostaje zwiększona o jeden, a wyświetlacz OLED aktualizuje informację o dostępnych miejscach. W przypadku, gdy pojazd nie zostanie wykryty

przez pierwszy czujnik w wyznaczonym czasie, szlaban zostaje zamknięty bez zmiany liczby wolnych miejsc.

Pętla kończy się krótkim opóźnieniem (100 ms), które zapobiega zbyt częstemu wywoływaniu funkcji i poprawia stabilność działania systemu.

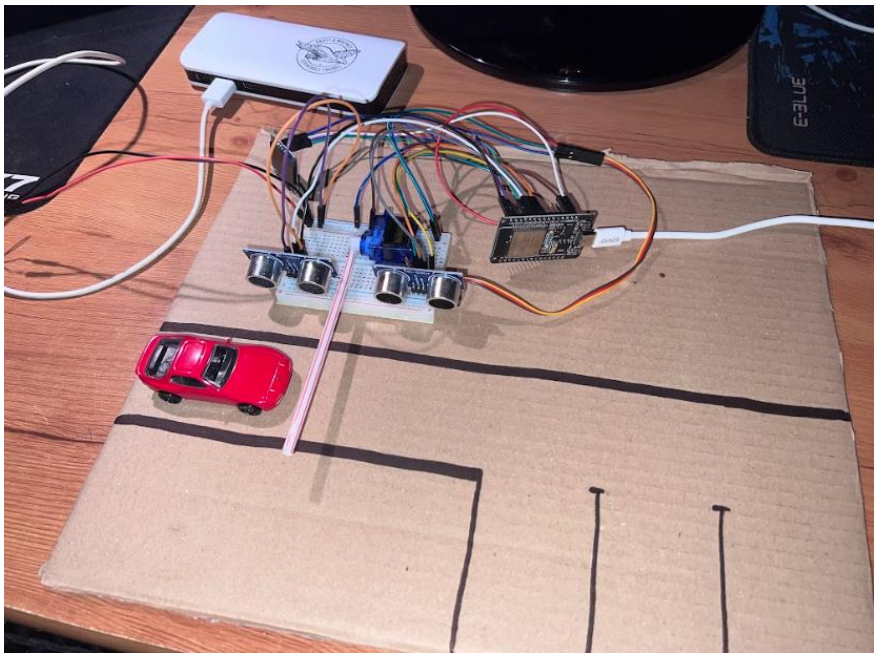
W kodzie zawartych jest kilka funkcji pomocniczych przedstawionych poniżej:

```
float measureDistance(int trigPin, int echoPin) {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  long duration = pulseIn(echoPin, HIGH); // Czas trwania sygnału w
mikrosekundach
  float distance = (duration * 0.0343) / 2; // Oblicz odległość w cm
  return distance;
}

bool detectCar(int trigPin, int echoPin) {
  float distance = measureDistance(trigPin, echoPin);
  return (distance > 0 && distance < 5); // Wykrywanie w zakresie <5 cm
}

void otworzSzlaban() {
  for (int angle = 0; angle <= 90; angle++) { // Od 0° do 90°
    szlaban.write(angle);
    delay(10); // Opóźnienie między krokami
  }
  Serial.println("Szlaban otwarty.");
}

void zamknijSzlaban() {
  for (int angle = 90; angle >= 0; angle--) { // Od 90° do 0°
    szlaban.write(angle);
    delay(10); // Opóźnienie między krokami
  }
  Serial.println("Szlaban zamknięty.");
}
```

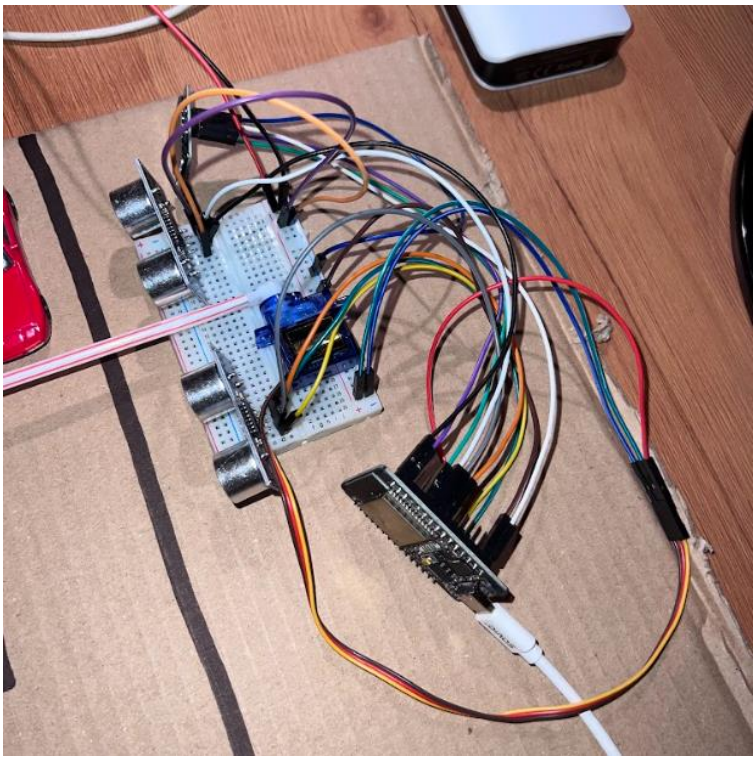
Funkcja measureDistance() mierzy odległość przy użyciu czujnika HC-SR04, na podstawie czasu powrotu echa. detectCar() wykrywa auto w odległości mniejszej niż 15 cm. otworzSzlaban() i zamknijSzlaban()stopniowo otwierają i zamykają szlaban, symulując jego ruch. Dodatkowo kod posiada funkcję do aktualizacji wyświetlacza OLED oraz funkcję wyświetlającą komunikat "Brak wolnych miejsc!" w przypadku braku miejsc parkingowych.

Podsumowując działanie kodu, steruje on systemem parkingowym, otwierając i zamykając szlaban w zależności od wjazdu i wyjazdu aut. Wyświetlacz OLED informuje o liczbie wolnych miejsc, a czujniki HC-SR04 dokładnie monitorują ruch pojazdów. Serwo SG90 odpowiada za realistyczny ruch szlabanu.

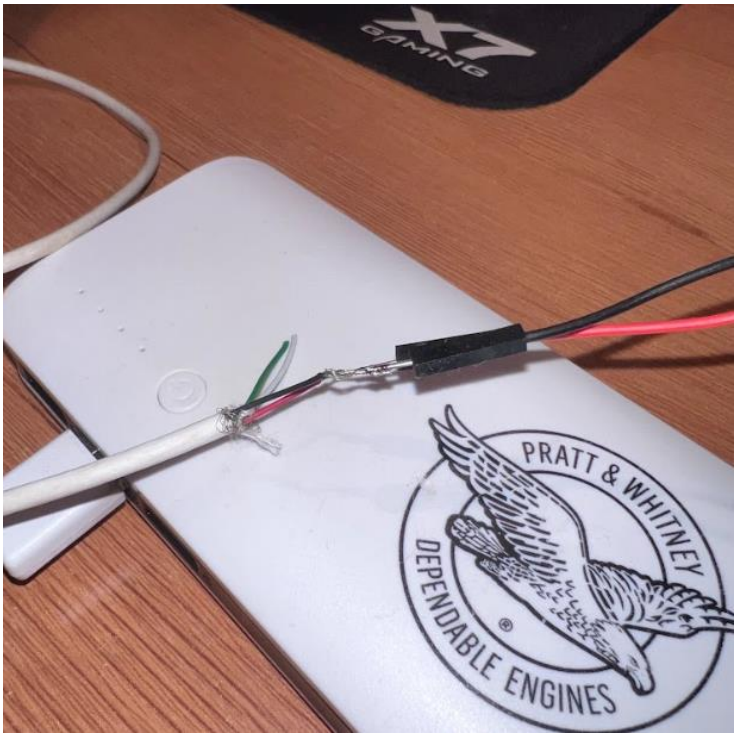
6. Zdjęcia



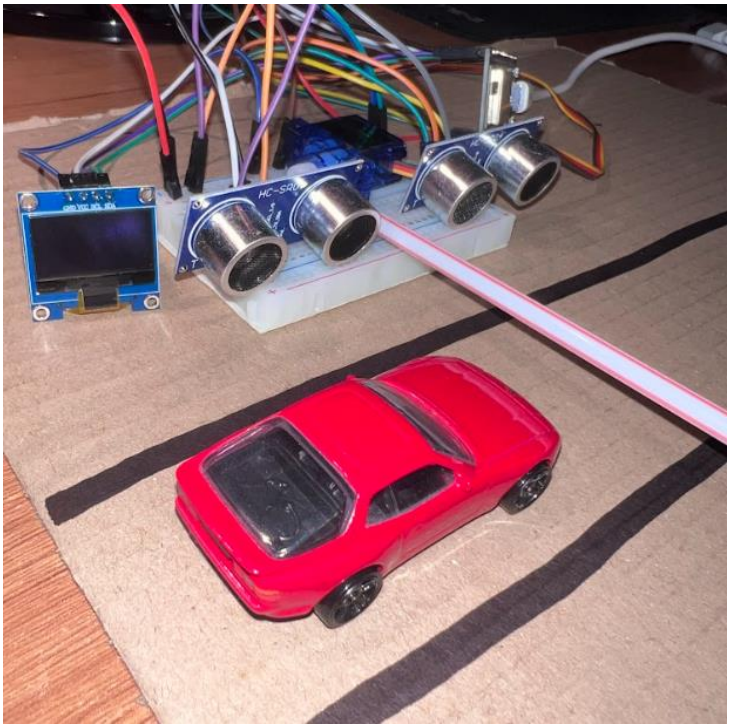
Rys. 1. Projekt parkingu samochodowego



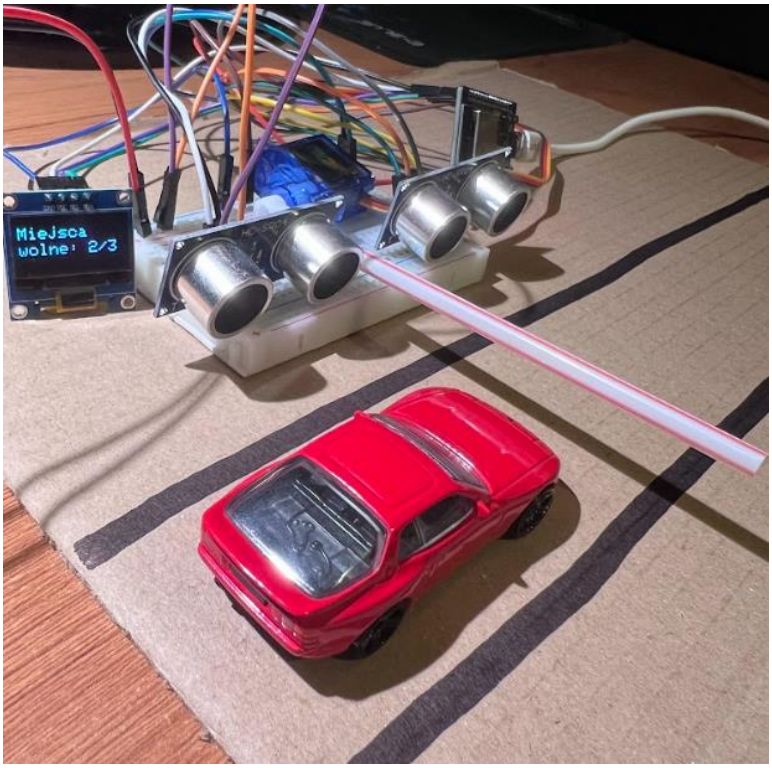
Rys. 2. Połączenia przewodów



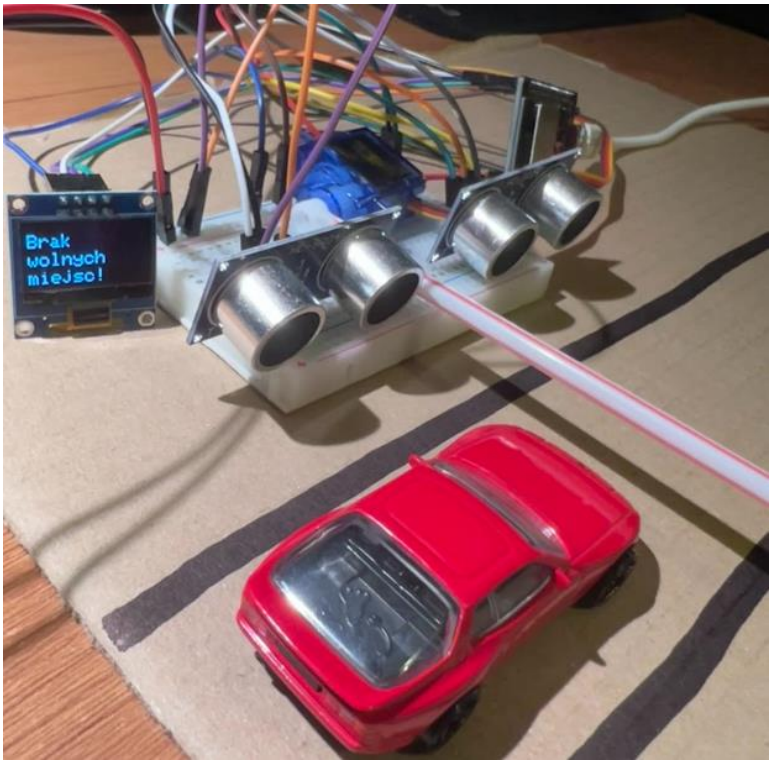
Rys. 3. Lutowanie przewodów do kabla powerbanka



Rys. 4. Auto stojące przed szlabanem



Rys. 5. Wyświetlanie ilości dostępnych miejsc parkingowych



Rys. 6. Wyświetlanie informacji o braku wolnych miejsc parkingowych

Wszystkie zamieszczone zdjęcia oraz dodatkowy film przedstawiający działanie układu znajdują się na dysku Google pod poniższym linkiem:

https://drive.google.com/drive/folders/1--orA6bVg4I1_K14_IkSLDixngsKn6DL?usp=sharing

7. Podsumowanie

W ramach projektu zaprojektowano i zrealizowano prostą makietę parkingu samochodowego sterowaną za pomocą mikrokontrolera ESP32. System umożliwia automatyczną kontrolę wjazdu i wyjazdu pojazdów, monitorując dostępność miejsc parkingowych oraz wyświetlając ich liczbę na wyświetlaczu OLED. Wykorzystano czujniki ultradźwiękowe HC-SR04 do detekcji pojazdów, serwomechanizm SG90 do sterowania szlabanem oraz powerbank jako dodatkowe, zewnętrzne źródło zasilania.

Przy użyciu EasyEDA zaprojektowano schemat połączeń i projekt płytki PCB, co umożliwiło łatwe odwzorowanie i montaż układu w rzeczywistości. Oprogramowanie, napisane w Arduino IDE, zarządzało logiką wjazdu i wyjazdu, reagując na dane z czujników i sterując szlabanem oraz wyświetlaczem.

Projekt działa zgodnie z założeniami – wykrywa pojazdy, otwiera i zamyka szlaban oraz aktualizuje stan wolnych miejsc parkingowych. Dokumentacja obejmuje zdjęcia z realizacji, pełne schematy oraz opis fragmentów kodu źródłowego.

Autor za wkład własny pracy uważa:

- Zakup odpowiednich komponentów potrzebnych do realizacji projektu
- Połączenie ze sobą elementów w celu stworzenia logicznie działającego układu
- Odpowiednie przylutowanie przewodów do kabla USB w celu stworzenia zasilania zewnętrznego w postaci powerbanka dla serwomechanizmu
- Pisanie oraz modyfikowanie kodu sterującego w Arduino IDE
- Stworzenie schematu ideowego oraz zaprojektowanie obwodu drukowanego PCB w EasyEDA
- Utworzenie prostej makiety parkingu samochodowego