Laboratorium z przedmiotu Systemy wbudowane (SW)

Karta projektu – zadanie 7

Nazwa projektu:

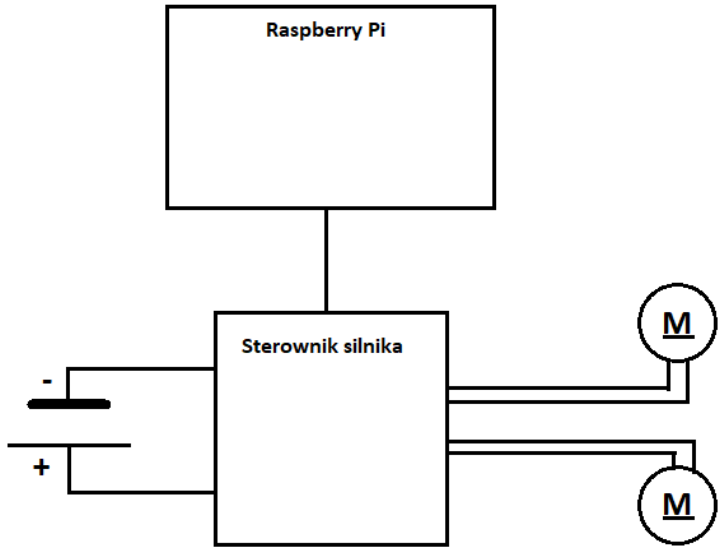
Zdalnie sterowany pojazd

Prowadzący:	*Autorzy (tylko nr indeksu):*	*Grupa dziekańska:*	I1.1
A. Antonowicz	142179, 136722	***Ocena:***	

Cel projektu:

Celem projektu jest zaprogramowanie na platformie Raspberry Pi pojazdu sterowanego zdalnie z komputera poprzez SSH, wykorzystując połączenie z siecią WiFi.

Schemat:



Wykorzystana platforma sprzętowa, czujniki pomiarowe, elementy wykonawcze:

- *Raspberry Pi 3B*
- *Sterownik silnika*
- *2 silniki prądu stałego*
- *Akumulator litowo-polimerowy*

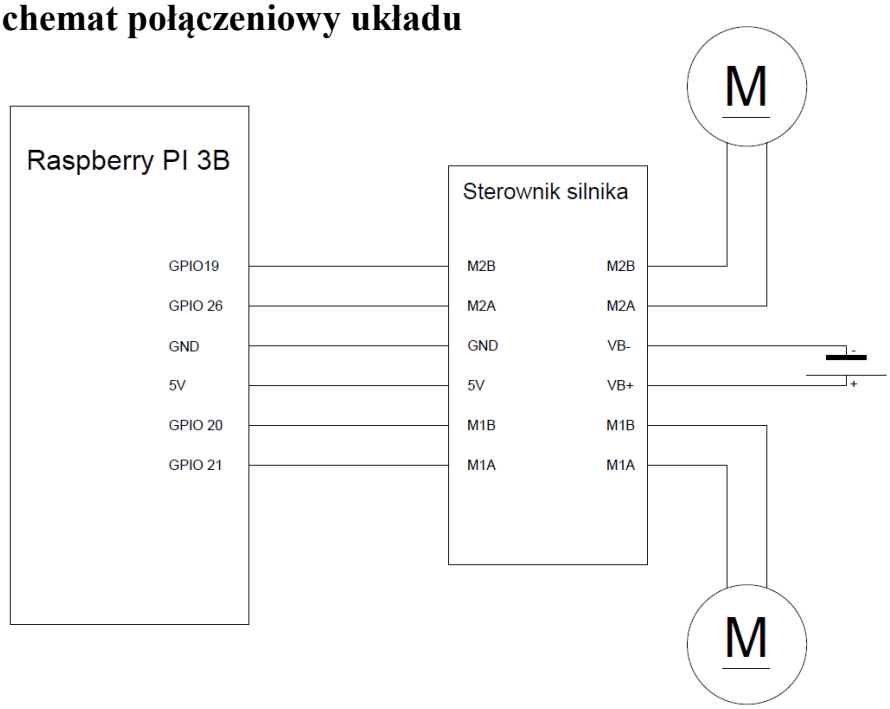
1. Cel i zakres projektu

Projekt miał na celu stworzenie modelu pojazdu z możliwością zdalnego sterowania z aplikacji z komputera.

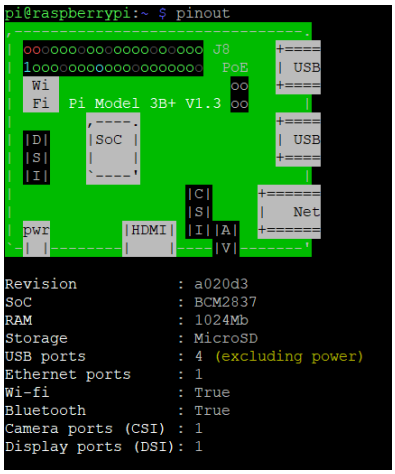
Pojazd to układ składający się z podwozia, trzech kół, z czego dwóch napędzanych silnikami prądu stałego, ze sterownika silników oraz z platformy Raspberry Pi. Do realizacji niezbędne było wykonanie podwozia oraz zakup pozostałych elementów, a także wypożyczenie Raspberry Pi.

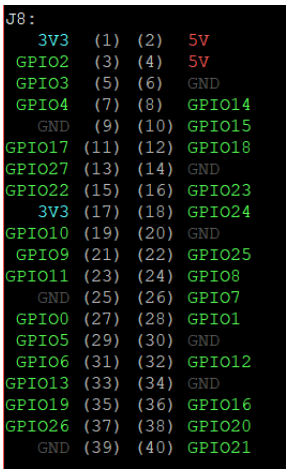
Zdalne sterowanie to przesyłanie danych za pomocą protokołu SSH, przy wykorzystaniu połączenia z siecią WiFi obu urządzeń. Do realizacji niezbędne było skonfigurowanie na Raspberry Pi protokołu SSH oraz możliwości bezprzewodowego łączenia się urządzenia z siecią.

2. Schemat połączeniowy układu



GPIO Raspberry Pi 3B:





3. Projekt a realizacja

Projekt udało się zrealizować zgodnie z założeniami.

Trudności pojawiły się przy konfiguracji protokołu SSH na Raspberry Pi – okazało się, iż domyślnie nie pozwala on na używanie X11 Forwarding, czyli przesyłania sesji graficznej, dodatkowo system Windows wymaga instalacji osobnego serwera oraz odpowiedniej konfiguracji PuTTY (IDE Pycharm rozwiązania tego nie wspiera w ogóle). Sposób rozwiązania problemu:

- uruchomienie programu serwera Xming,
- skonfigurowanie ustawień SSH->X11 w PuTTy.

Ustawione dane do połączenia SSH:

Login: pi ; hasło: raspberry

Oprócz tego nie pojawiły się dalsze trudności.

Projekt okazał się być prosty w realizacji, jednak dający możliwości ciekawego rozbudowania go dalej. Efektowną opcją mogłaby być kamerka rejestrująca obraz z pojazdu i przekazująca go na komputer z uruchomioną aplikacją. Innym sposobem rozbudowy mogłoby być dołączenie czujnika odległości, umożliwiającego wykrycie przeszkody i wyhamowanie przed nią pojazdu w celu uniknięcia zderzenia.

4. Najważniejsze fragmenty kodu

W kodzie programu można wyróżnić dwie główne części: pierwszą, odpowiedzialną za wygląd aplikacji oraz drugą, odpowiedzialną za zdalne sterowanie silnikami pojazdu.

Wykorzystane biblioteki

```
import RPi.GPIO as IO
import time
import pigpio
import os, sys
import pygame
```

Wygląd aplikacji – biblioteka pygame

```
def set_text(font_settings, text,color, X=100,Y=100):
   final_text = font_settings.render(text, True, color, white)
   final_text_rectangle = final_text.get_rect()
   final_text_rectangle.center = (X,Y)
   return final_text,final_text_rectangle


pygame.init()
screen = pygame.display.set_mode((500, 300))
pygame.display.set_caption('Remote Car Controller')
```

```
instruction_font = pygame.font.Font('freesansbold.ttf', 16)
text_w, text_w_rectangle = set_text(instruction_font,'w - forward',
green, 250,70)
text_s, text_s_rectangle = set_text(instruction_font,'s- backward',
blue, 250,90)
text_a, text_a_rectangle = set_text(instruction_font,'a - left', green,
250,110)
text_d, text_d_rectangle = set_text(instruction_font, 'd - right', blue,
250,130)
```

Sterowanie silnikiem

```
HIGH = 110
MEDIUM=95
LOW = 70
def motor_speed(direction,
m1a_speed=0,m1b_speed=0,m2a_speed=0,m2b_speed=0):
   if direction=="FORWARD":
      m1b_speed = MEDIUM       #Right motor forward
      m2a_speed = MEDIUM       #Left motor forward
   elif direction=="BACKWARD":
      m1a_speed = MEDIUM       #Right motor backward
      m2b_speed = MEDIUM       #Left motor backward
   elif direction=="LEFT":
      m1b_speed = HIGH         #Right motor strongly forward
      m2a_speed = LOW             #Left motor weakly forward
   elif direction=="RIGHT":
      m1b_speed = LOW             #Right motor weakly forward
      m2a_speed = HIGH         #Left motor strongly forward

   pi.set_PWM_dutycycle(M1A, m1a_speed)
   pi.set_PWM_dutycycle(M1B, m1b_speed)
   pi.set_PWM_dutycycle(M2A, m2a_speed)
   pi.set_PWM_dutycycle(M2B, m2b_speed)
```

Pętla główna programu:

```
while (run):
   screen.fill(white)
   screen.blit(text_w, text_w_rectangle)
   screen.blit(text_s, text_s_rectangle)
   screen.blit(text_a, text_a_rectangle)
   screen.blit(text_d, text_d_rectangle)
   time.sleep(.02)
   key_state = pygame.key.get_pressed()

   if key_state[pygame.K_w]:  # ride forward
      motor_speed("FORWARD")


   elif key_state[pygame.K_s]:  # ride backward
      motor_speed("BACKWARD")
      text = "s pressed"

   elif key_state[pygame.K_a]:  # turn left
```

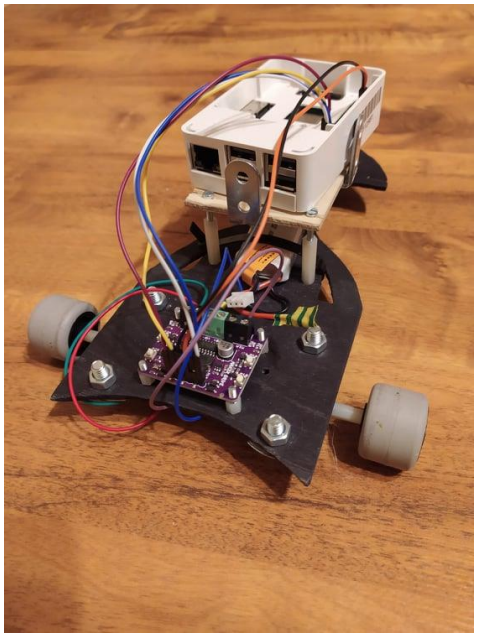
```
            motor_speed("LEFT")
            text = "a pressed"

        elif key_state[pygame.K_d]:  # turn left
            motor_speed("RIGHT")
            text = "d pressed"

        else:
            motor_speed("STOP")
            text = ""

        text_pressed, text_pressed_rectangle = set_text(instruction_font,
text, pink, 250, 200)
        screen.blit(text_pressed, text_pressed_rectangle)
        pygame.display.update()

        for event in pygame.event.get():
            if event.type == pygame.KEYDOWN:
                if event.key == K_ESCAPE:  # end game
                    motor_speed("STOP")
                    run=False
                    pygame.quit()
                    break
```

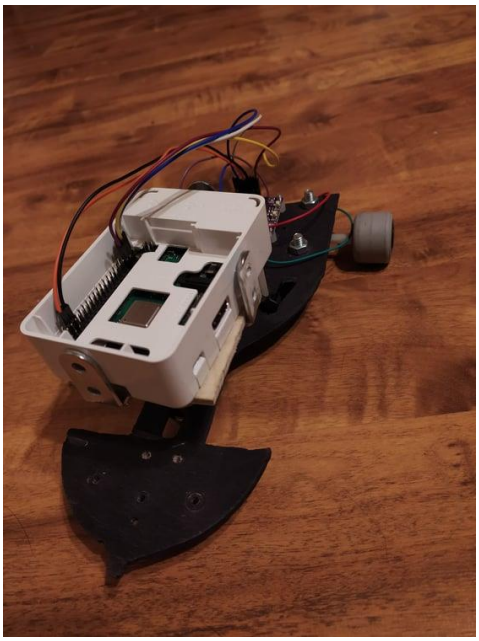
5. Prezentacja urządzenia i połączeń

5.1 Specyfikacja elementów układu

Silniki prądu stałego	Sterownik silnika	Akumulator
Silnik N20-BT12 micro 10:1 1300RPM - 9V Napięcie zasilania: 3-9V Prąd bez obciążenia: 35mA Prędkość bez obciążenia: 1300 obr/min Moment obrotowy: 0,014Nm Przełożenie: 10:1 Średnica wału: 3mm	**Cytron Maker Drive MX1508** Układ: MX1508 Ilość kanałów: 2 Napięcie zasilania: 2,5-9,5 V Napięcie wyjściowe: 5V Prąd wyjściowy: 200 mA Prąd ciągły: 1A Prąd chwilowy: 1,5A Mostek H	**Pakiet Li-Pol Dualsky 800mAh 25C 2S 7.4V ECO-S** Dwa ogniwa Li-pol Napięcie nominalne: 7,4 V Pojemność: 88 mAh Prąd rozładowania ciągły: 20A

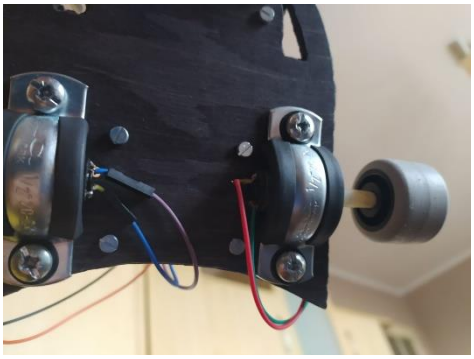
5.2 Prezentacja pojazdu



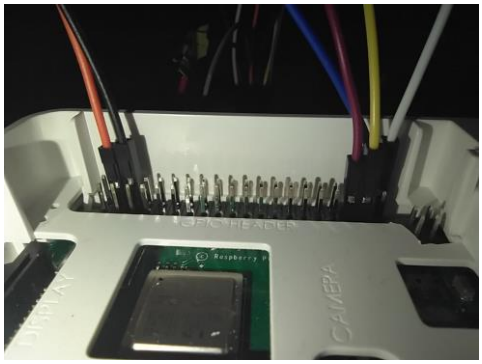


5.3 Realizacja połączeń

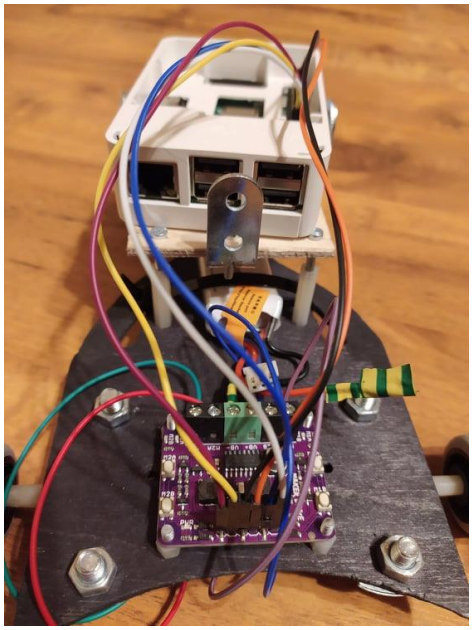
Mocowanie silników:



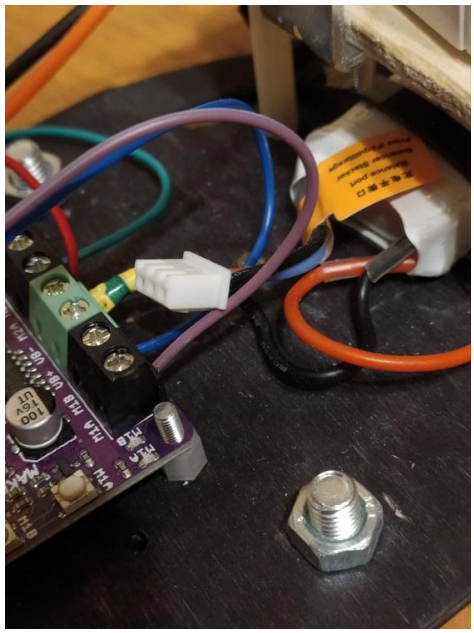
Podłączenie pinów do Raspberry:



Podłączenia sterownika silników:



Podłączenie akumulatora do sterownika:



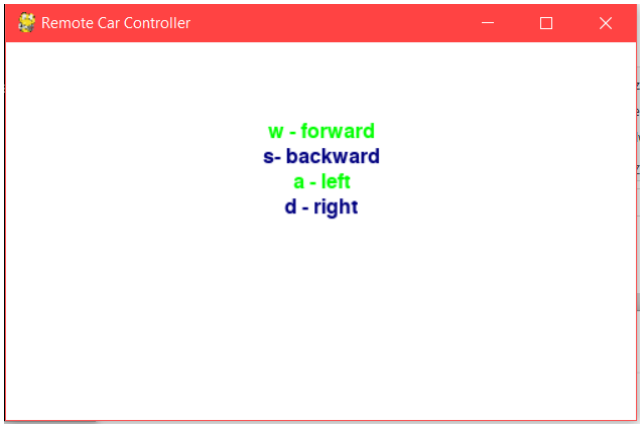
6. Działanie aplikacji

Aplikacja to proste okno graficzne wyświetlane przy użyciu bibliotegi **pygame**.

Sterowanie pojazdem odbywa się przy użyciu klawiatury:

- w – jazda prosto
- s – jazda w tył
- a – skręt w lewo
- d – skręt w prawo

Aplikacja wykrywa wciśnięty klawisz i wyświetla go w oknie oraz przekazuje do pętli głównej programu, gdzie na jej podstawie wybierane są odpowiednie instrukcje silnika.



7. Podsumowanie i wnioski

1. Pojazd udało się zrealizować zgodnie z założeniami oraz schematem wstępnym. Dobrane silniki oraz sterownik zostały odpowiednio dopasowane przez co uniknięto awarii sprzętu. Szczególną uwagę należało zwracać na odpowiednie zabezpieczenie elementów, aby nie dopuścić do spięć przewodów, które mogłoby prowadzić do uszkodzenia wszystkich podłączonych elementów.
2. Platforma Raspberry Pi 3B posiada wbudowany moduł WiFi, dzięki czemu nie trzeba było dołączać go osobno. Do łączenia z Raspberry Pi poprzez WiFi konieczne było połączenie przewodowe z siecią lokalną i następnie skonfigurowanie możliwości łączenia bezprzewodowego na platformie poprzez protokół SSH. Po poprawnej konfiguracji, możliwe było łączenie bez podłączania przewodowego. Do nawiązywania połączenia SSH wykorzystywano w projekcie program PuTTy z użyciem serwera Xming oraz środowisko PyCharm.
3. Zadanie projektowe pozwoliło poznać kolejną, obok BeagleBone Black, platformę z systemem wbudowanym. Praca z użyciem Raspberry Pi nie różniła się szczególnie od sposobu wykonywania zadań na BeagleBone podczas laboratoriów.