





## Milestone 1





### 1. Zwięzły opis robota

Projekt zakłada stworzenie modelu pojazdu autonomicznego opartego na mikrokontrolerze STM32, który w trybie półautomatycznym realizuje polecenia operatora, a w pełni automatycznym samodzielnie wybiera trasę i reaguje na otoczenie. Dzięki zestawowi czujników ultradźwiękowych i optycznych pojazd potrafi wykrywać i omijać przeszkody oraz precyzyjnie podążać za namalowaną linią trasy. Sterowanie napędem realizowane jest poprzez generowanie sygnałów PWM, a całość logiki działania opiera się na wbudowanych timerach i (opcjonalnie) systemie zarządzania zadaniami w czasie rzeczywistym. Użytkownik komunikuje się z pojazdem poprzez interfejs UART (np. moduł Bluetooth HC-05), co umożliwia zdalne wydawanie komend, odczyt stanu sensorów oraz zmianę trybu jazdy. Mam pomysł, żeby na płytce STM32F3Discovery zaimplementować diodowy system informacyjny: podczas skrętu w prawo migotałyby diody zamontowane po prawej stronie platformy, a przy skręcie w lewo – analogiczne diody po stronie lewej.

### 2. Elementy wybrane do budowy robota

Lp.	Komponent	Ilość	Zdjęcie
1	HC-05 – bezprzewodowy moduł Bluetooth	1	
2	HC-SR04 czujnik ultradźwiękowy	2	
3	Koszyk na 6 baterii typu AA (R6)	1	
4	L298N – płyta sterownika silników DC	1	



5	Płytki stykowa justPi – 400 otworów	1	
6	Silniki DC TT z przekładnią i kołami (3–6 V, wał podwójny, 65 mm)	4	
7	STM32F3-Discovery (STM32F3DISCOVERY)	1	
8	TCRT5000 – czujnik odblaskowy IR	2	

### 3. Mechanika robota

#### Deseczka jako podstawa robota

- Wymiary: 14,5 cm × 22 cm × 0,5 cm, cienka sklejka.
- Wywierciłem w niej otwór pod kable prowadzące do czujników TCRT5000 – dzięki temu przewody nie płaczą się pod robotem.

#### Mocowanie silników DC

- Przyklejone bezpośrednio do spodniej strony deseczki klejem Pattex (do klejenia drewna i innych tworzyw sztucznych).
- Klej trzyma bardzo mocno, a nie dodaje praktycznie żadnej masy.

#### Sterownik silników (L298N)

- Przytwierdziłem do kawałka dokładnie dociętego styropianu – Całość przymocowałem do deseczki taśmą dwustronną.

#### Płytki STM32F3-Discovery

- Zamontowana jest na kawałku plastikowego bloczka z zestawu Jengi (oczywiście przyklejona z dwóch stron).

#### Czujniki TCRT5000

- Zamocowałem je na przedniej lewej i prawej stronie robota, korzystając z taśmy dwustronnej i szarej taśmy. uformowałem z dwustronnej taśmy małe „bloczki”, które trzymają płytki czujników.
- Przewody poprowadzone przez otwór w deseczce, dzięki czemu nie przeszkadzają w jeździe.

#### Czujniki ultradźwiękowe HC-SR04

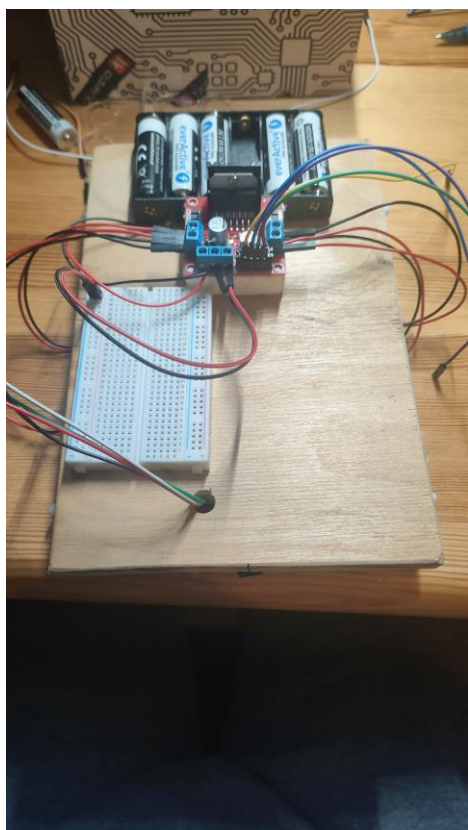
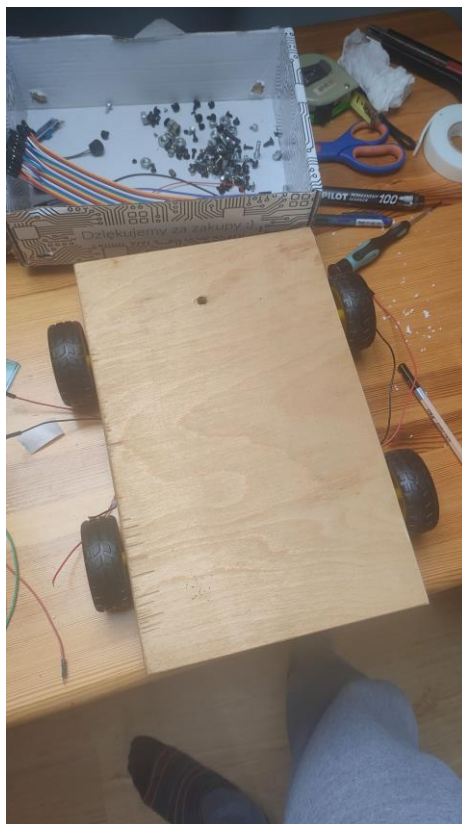
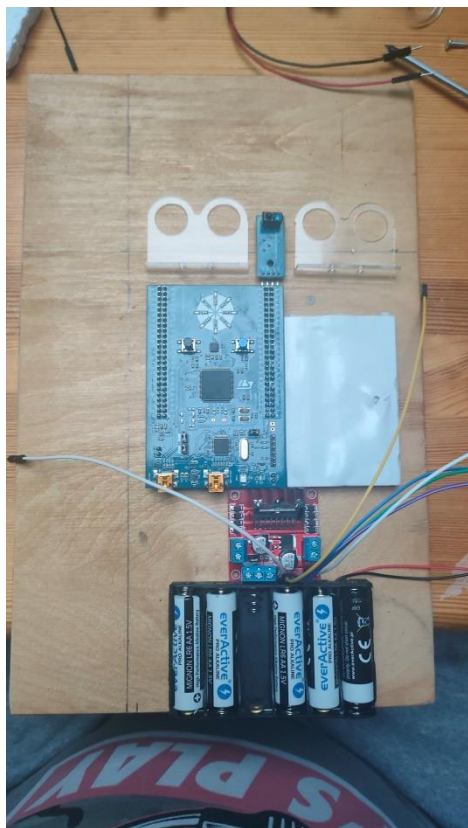
- Przyklejona podstawa specjalnie do tych czujników za pomocą taśmy dwustronnej i żeby nie spadały przykleiłem je klejem.

#### Okablowanie

- Wszystkie przewody sygnałowe i zasilające zakończyłem złączkami żeńsko-męskimi, żeńsko-żeńskimi i męsko-męskimi.
- Prowadzenie kabli wzdłuż krawędzi deseczki zabezpieczyłem punktowo taśmą, żeby nic się nie plątało pod kołami.

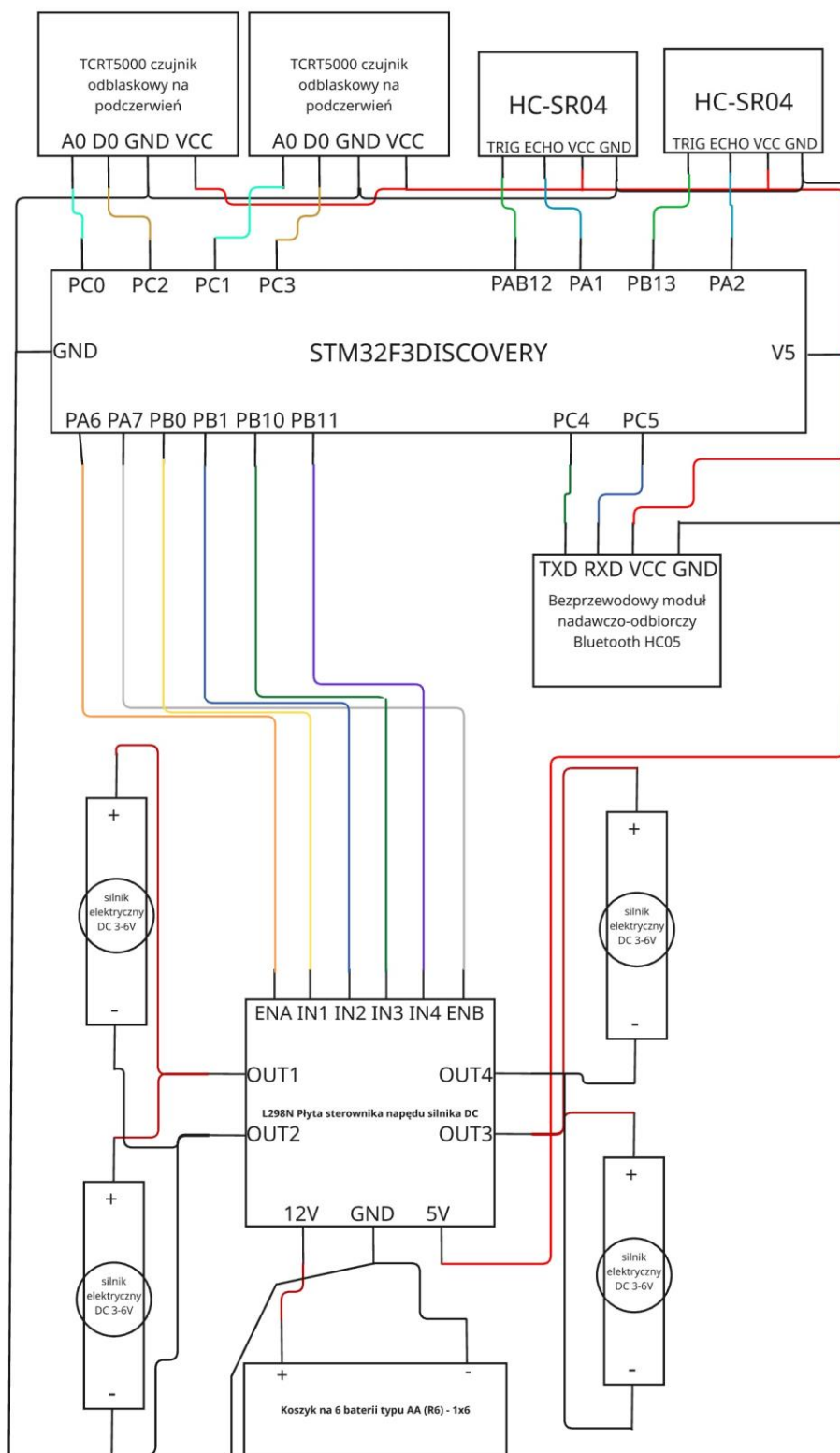


Poniżej kilka zdjęć podczas budowy robota:

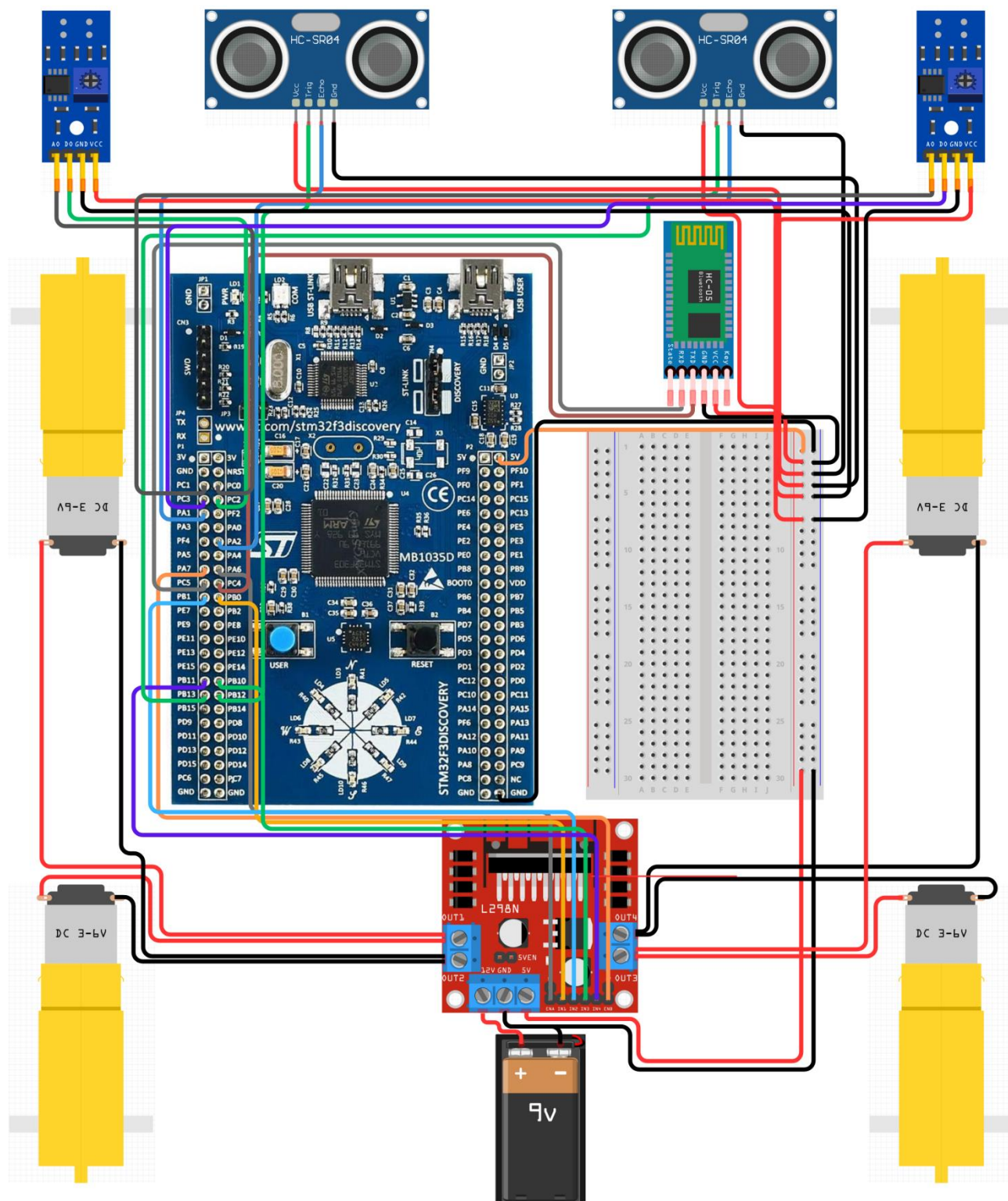




#### 4. Schemat elektroniczny robota



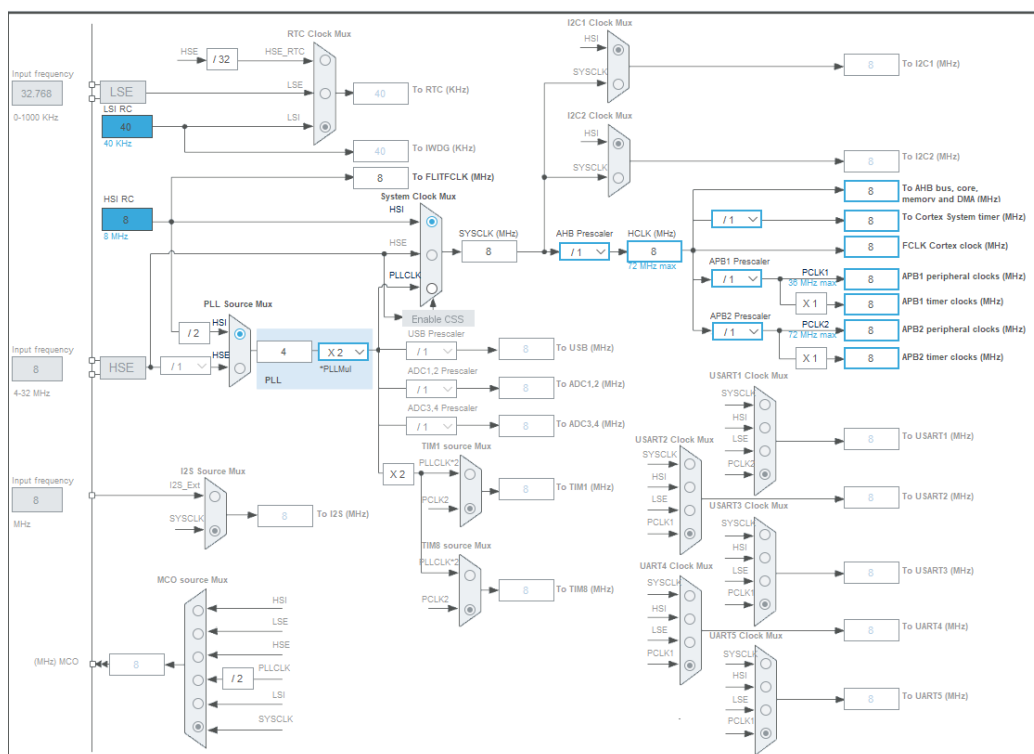
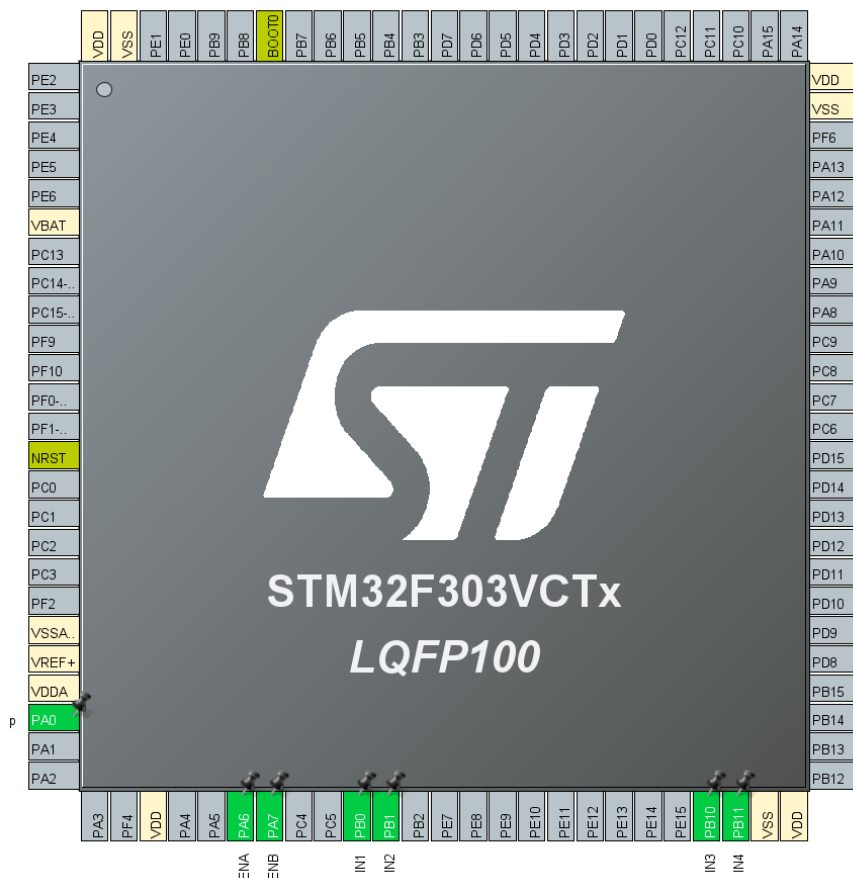
Rys. 4.1: Schemat uproszczony układu elektronicznego robota (zrobiony za pomocą programu Microsoft Paint).



Rys. 4.2: Schemat montażowy robota z płytką stykową (zrobiony za pomocą programu Canva).



## 5. Oprogramowanie sterujące



The image displays the STM32CubeMX configuration interface for a TIM3 timer. The left sidebar shows the 'Timers' category selected, with TIM3 highlighted. The main area shows the 'Mode' configuration for TIM3, including Slave Mode (Disable), Trigger Source (Disable), Clock Source (Internal Clock), and Channel settings (Channel1: PWM Generation CH1, Channel2: PWM Generation CH2, Channel3: Disable, Channel4: Disable). The 'Combined Channels' are set to Disable. Below the Mode configuration, there are checkboxes for 'ETR IO as Clearing Source', 'XOR activation', and 'One Pulse Mode'. The bottom section shows the 'Configuration' tab with 'Parameter Settings' selected, displaying the 'Counter Settings' for TIM3, including Prescaler (PSC - 16 bits value), Counter Mode (Up), Counter Period (AutoR... 99), Internal Clock Division (No Division), and auto-reload preload (Disable). The 'Trigger Output (TRGO) Param...' section shows Master/Slave Mode (Disable) and Trigger Event Selectio... (Reset). The 'Clear Input' section shows Clear Input Source (Disable). The 'PWM Generation Channel 1' section shows Mode (PWM mode 1), Pulse (16 bits value) (0), Output compare preload (Enable), and Fast Mode (Disable).





Fragmenty kodu pochodzące z pliku main.c:

```
#define CZAS_JAZDY      1000  
#define CZAS_OBROTU    300
```

CZAS\_JAZDY = 1000 oznacza, że robot jedzie przez 1 sekundę. CZAS\_OBROTU\_90 = 300 sprawia, że obrót o prawo czy w lewo trwa 0,3 sekundy.

```
static void zatrzymajSilniki(void)  
{  
    HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_1, 0);  
    HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_2, 0);  
}
```

Funkcja zatrzymajSilniki() wyłącza sygnał PWM na obu kanałach silników (TIM\_CHANNEL\_1 i TIM\_CHANNEL\_2), ustawiając wartość porównania na 0. W praktyce oznacza to 0 % wypełnienia sygnału PWM, czyli silniki przestają dostawać sygnał i natychmiast się zatrzymują.

```
void jedzNaprzod(void)  
{  
    uint32_t predkosc = (htim3.Init.Period+1)*0.3;  
    // ustawienie kierunku  
    HAL_GPIO_WritePin(IN1_GPIO_Port, IN1_Pin, 1);  
    HAL_GPIO_WritePin(IN2_GPIO_Port, IN2_Pin, 0);  
    HAL_GPIO_WritePin(IN3_GPIO_Port, IN3_Pin, 1);  
    HAL_GPIO_WritePin(IN4_GPIO_Port, IN4_Pin, 0);  
    // ustawienie prędkości  
    __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_1, predkosc);  
    __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_2, predkosc);  
    // jazda  
    HAL_Delay(CZAS_JAZDY);  
    zatrzymajSilniki();  
}
```

Funkcja jedzNaprzod() najpierw oblicza wartość odpowiadającą 30 % pełnej mocy timera, a następnie ustawia piny sterujące tak, aby oba silniki kręciły się do przodu. Po uruchomieniu PWM na zadanej prędkości funkcja czeka przez czas określony makrem CZAS\_JAZDY, a na koniec przerywa sygnał i zatrzymuje silniki.

```
void jedzDoTyłu(void)  
{  
    uint32_t predkosc = (htim3.Init.Period + 1)*0.3;  
    HAL_GPIO_WritePin(IN1_GPIO_Port, IN1_Pin, 0);  
    HAL_GPIO_WritePin(IN2_GPIO_Port, IN2_Pin, 1);  
    HAL_GPIO_WritePin(IN3_GPIO_Port, IN3_Pin, 0);  
    HAL_GPIO_WritePin(IN4_GPIO_Port, IN4_Pin, 1);  
    __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_1, predkosc);  
    __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_2, predkosc);  
    HAL_Delay(CZAS_JAZDY);  
    zatrzymajSilniki();  
}  
void skrecWLevo(void)  
{  
    uint32_t predkosc = (htim3.Init.Period + 1)*0.3;  
    HAL_GPIO_WritePin(IN1_GPIO_Port, IN1_Pin, 1);  
    HAL_GPIO_WritePin(IN2_GPIO_Port, IN2_Pin, 0);  
    HAL_GPIO_WritePin(IN3_GPIO_Port, IN3_Pin, 0);  
    HAL_GPIO_WritePin(IN4_GPIO_Port, IN4_Pin, 1);  
    __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_1, predkosc);  
    __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_2, predkosc);  
}
```



```
HAL_Delay(CZAS_OBROTU_90);  
zatrzymajSilniki();  
}
```

Funkcja jedzDoTyłu() oblicza wartość PWM odpowiadającą 30 % pełnej skali timera, po czym ustawia piny sterujące tak, aby oba silniki kręciły się w kierunku odwrotnym. Następnie wpisuje tę wartość do kanałów PWM, wywołuje opóźnienie równe CZAS\_JAZDY, a na koniec przerywa sygnał i zatrzymuje silniki.

```
void skrecWLevo(void)  
{  
    uint32_t predkosc = (htim3.Init.Period + 1)*0.3;  
    HAL_GPIO_WritePin(IN1_GPIO_Port, IN1_Pin, 1);  
    HAL_GPIO_WritePin(IN2_GPIO_Port, IN2_Pin, 0);  
    HAL_GPIO_WritePin(IN3_GPIO_Port, IN3_Pin, 0);  
    HAL_GPIO_WritePin(IN4_GPIO_Port, IN4_Pin, 1);  
    __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_1, predkosc);  
    __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_2, predkosc);  
    HAL_Delay(CZAS_OBROTU_90);  
    zatrzymajSilniki();  
}
```

Funkcja skrecWLevo() także wylicza 30 % maksymalnej mocy PWM, a potem ustawia piny kierunkowe tak, żeby lewy silnik jechał naprzód, a prawy do tyłu. Po uruchomieniu PWM zadaną prędkością funkcja czeka przez czas CZAS\_OBROTU\_90, a potem wyłącza sygnał i zatrzymuje silniki.

```
void skrecWPrawo(void)  
{  
    uint32_t predkosc = (htim3.Init.Period + 1)*0.3;  
    HAL_GPIO_WritePin(IN1_GPIO_Port, IN1_Pin, 0);  
    HAL_GPIO_WritePin(IN2_GPIO_Port, IN2_Pin, 1);  
    HAL_GPIO_WritePin(IN3_GPIO_Port, IN3_Pin, 1);  
    HAL_GPIO_WritePin(IN4_GPIO_Port, IN4_Pin, 0);  
    __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_1, predkosc);  
    __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_2, predkosc);  
    HAL_Delay(CZAS_OBROTU_90);  
    zatrzymajSilniki();  
}
```

Funkcja skrecWPrawo() działa analogicznie, tyle że kierunek na pinach jest odwrócony: lewy silnik kręci się do tyłu, a prawy naprzód, co powoduje obrót robota w prawo. Po upływie CZAS\_OBROTU\_90 PWM jest zatrzymywany, a silniki przestają się obracać.

```
void zatrzymajRobota(void)  
{  
    HAL_GPIO_WritePin(IN1_GPIO_Port, IN1_Pin, 0);  
    HAL_GPIO_WritePin(IN2_GPIO_Port, IN2_Pin, 0);  
    HAL_GPIO_WritePin(IN3_GPIO_Port, IN3_Pin, 0);  
    HAL_GPIO_WritePin(IN4_GPIO_Port, IN4_Pin, 0);  
    HAL_TIM_PWM_Stop(&htim3, TIM_CHANNEL_1);  
    HAL_TIM_PWM_Stop(&htim3, TIM_CHANNEL_2);  
    __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_1, 0);  
    __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_2, 0);  
}
```

Funkcja zatrzymajRobota() ustawia wszystkie piny sterujące w stan niski, przerywa sygnał PWM na obu kanałach i zeruje wartości porównania. Dzięki temu robot całkowicie traci zasilanie na wyjściach i nie wykonuje żadnego ruchu.



```
void sekwencja(void)
{
    HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_1);
    HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_2);
    jedzNaprzod();
    skrecWLewo();
    jedzNaprzod();
    skrecWPrawo();
    jedzDoTylu();
    jedzNaprzod();
    skrecWPrawo();
    skrecWLewo();
    jedzDoTylu();
    skrecWPrawo();
    jedzNaprzod();
    zatrzymajRobota();
}
```

Funkcja sekwencja() rozpoczyna pracę od uruchomienia sygnału PWM na obu kanałach timera, co przygotowuje silniki do wykonywania ruchów. Następnie kolejno wywołuje funkcje jedzNaprzod(), skrecWLewo(), jedzNaprzod(), skrecWPrawo(), jedzDoTylu(), jedzNaprzod(), skrecWPrawo(), skrecWLewo(), jedzDoTylu(), skrecWPrawo() oraz ponownie jedzNaprzod(), realizując w ten sposób zaprogramowaną ścieżkę manewrów. Po wykonaniu wszystkich operacji ruchu funkcja kończy działanie, wywołując zatrzymajRobota(), które zatrzymuje sygnał PWM i całkowicie wyłącza silniki.

```
int main(void)
{
    /* USER CODE BEGIN 1 */

    /* USER CODE END 1 */

    /* MCU Configuration-----*/

    /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
    HAL_Init();

    /* USER CODE BEGIN Init */

    /* USER CODE END Init */

    /* Configure the system clock */
    SystemClock_Config();

    /* USER CODE BEGIN SysInit */

    /* USER CODE END SysInit */

    /* Initialize all configured peripherals */
    MX_GPIO_Init();
    MX_TIM3_Init();
    /* USER CODE BEGIN 2 */

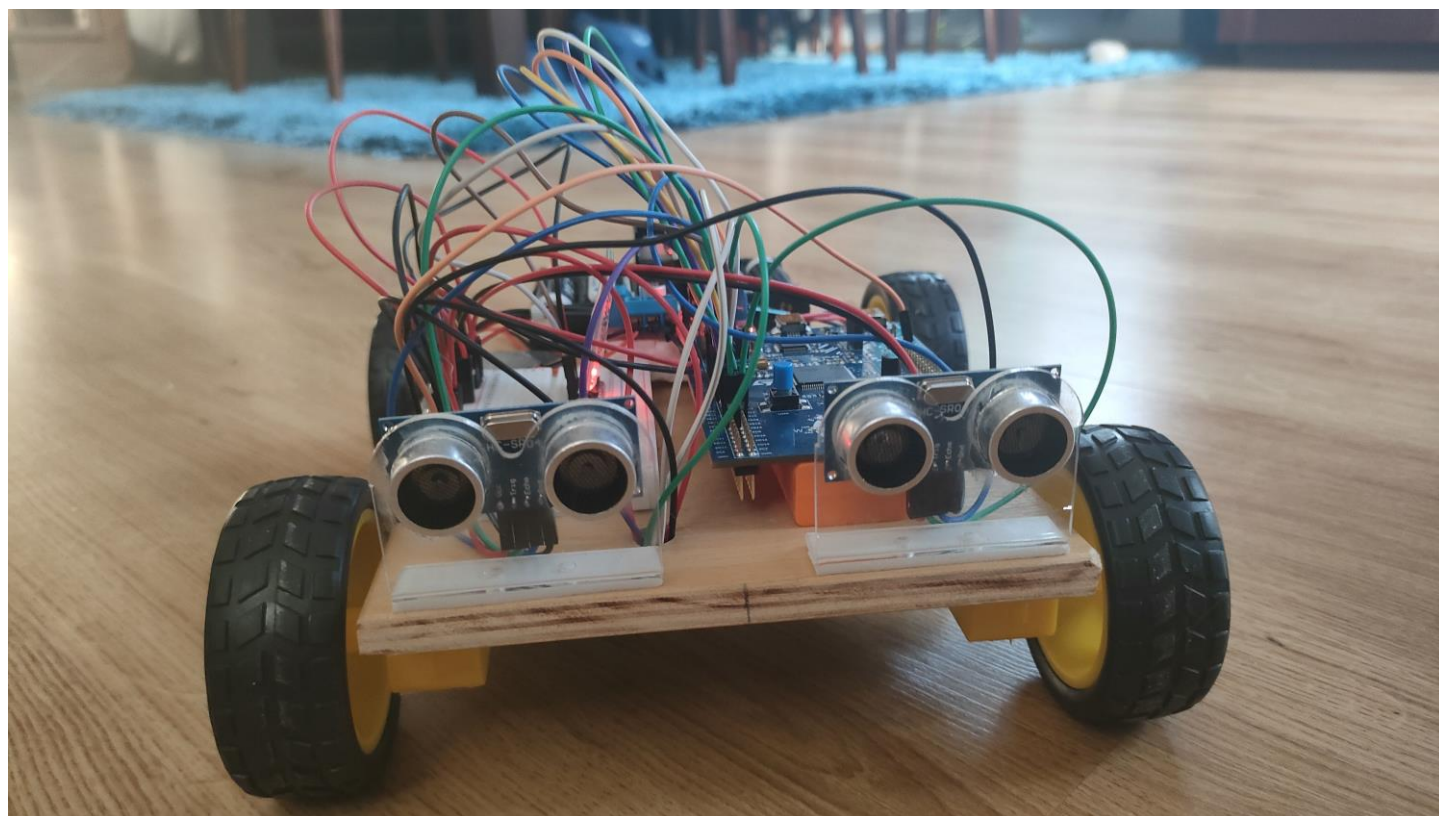
    /* USER CODE END 2 */

    /* Infinite loop */
```

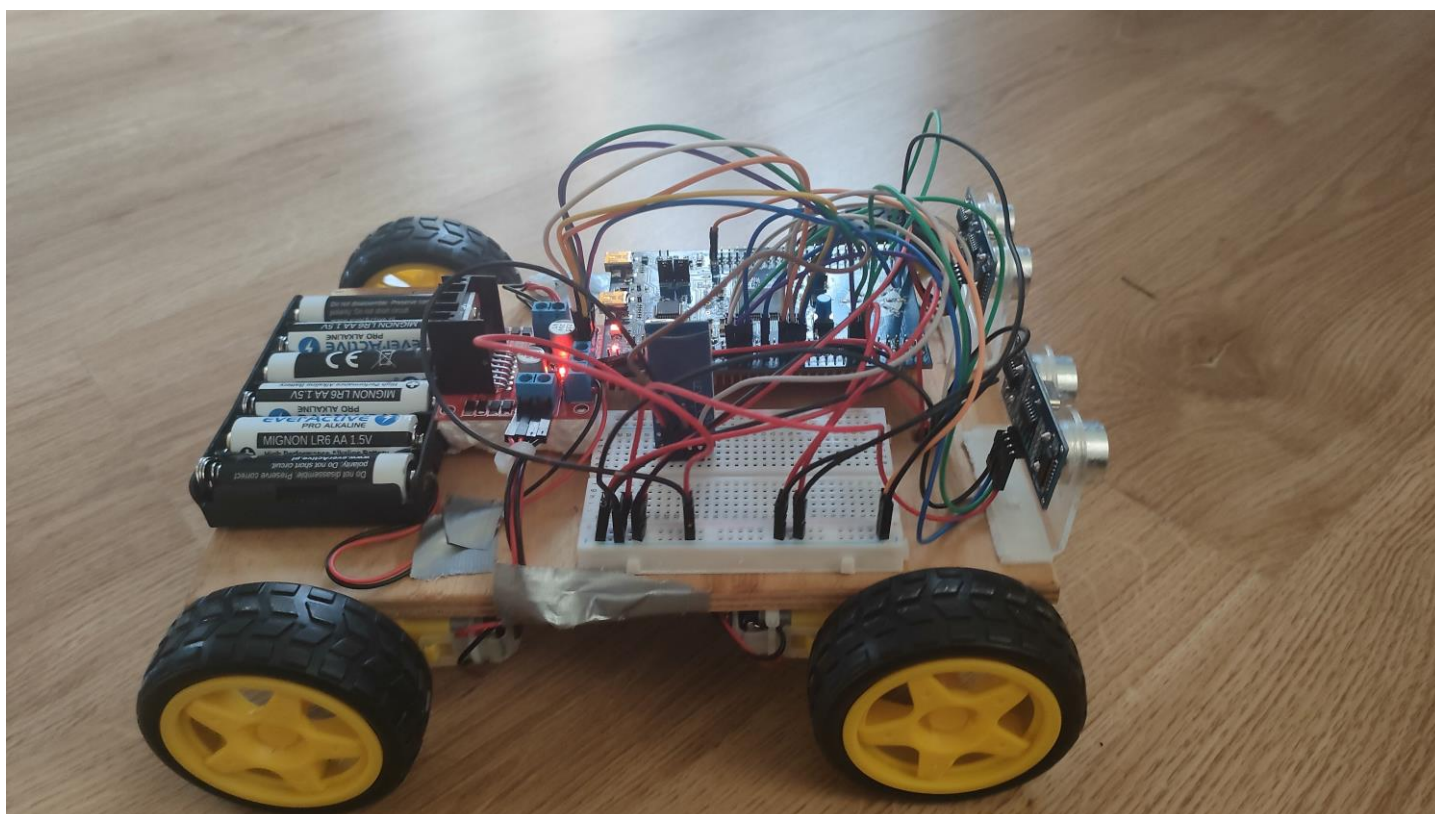
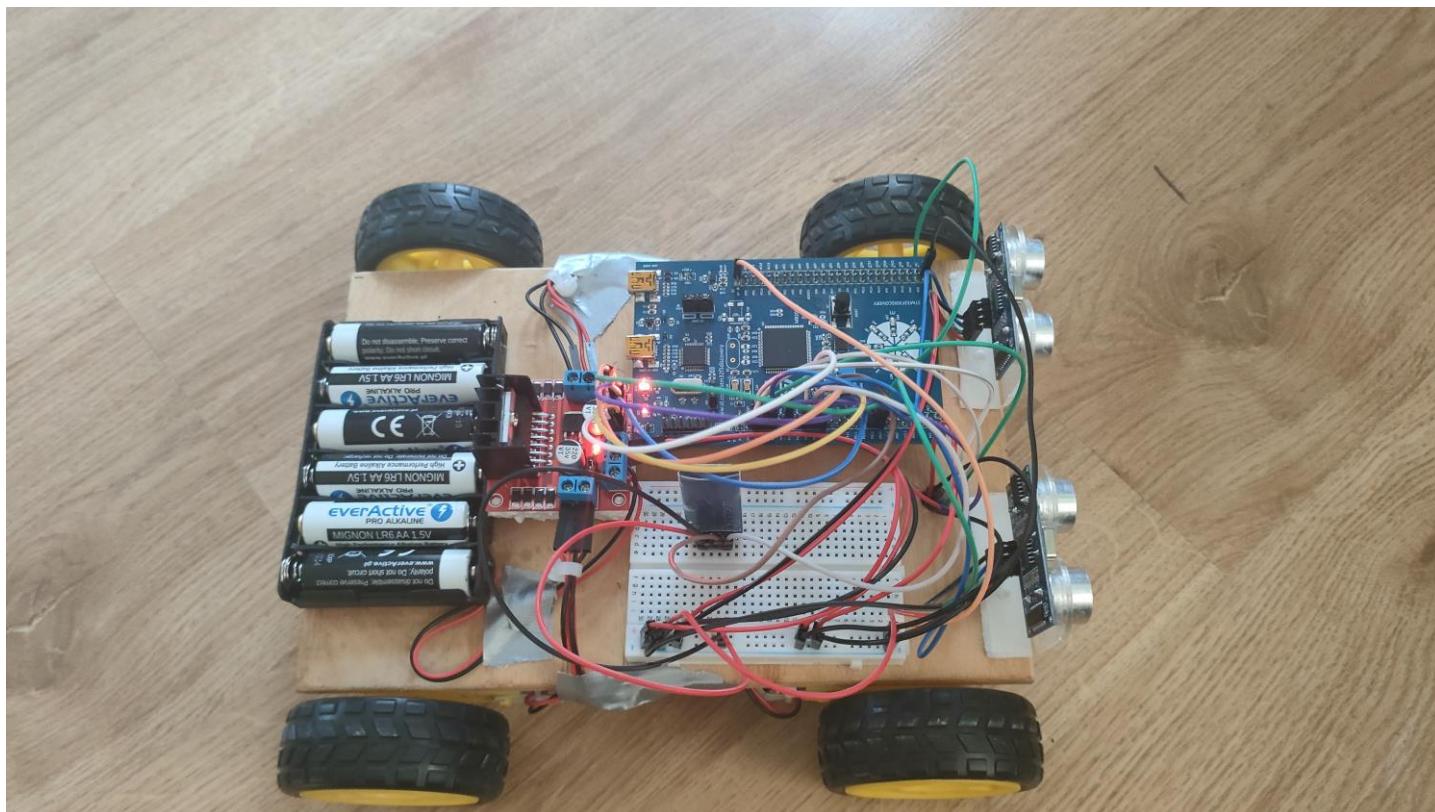
```
/* USER CODE BEGIN WHILE */
uint8_t guzikoSprawdz = 0;
uint8_t guzikoSprawdzPrzed = 0;
while (1){
    guzikoSprawdz = HAL_GPIO_ReadPin(p_GPIO_Port, p_Pin);
    if(guzikoSprawdz == 1 && guzikoSprawdzPrzed == 0){
        wykonajSekwencjeRobota();
    }
    guzikoSprawdzPrzed = guzikoSprawdz;
}
/* USER CODE END WHILE */
/* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
}
```

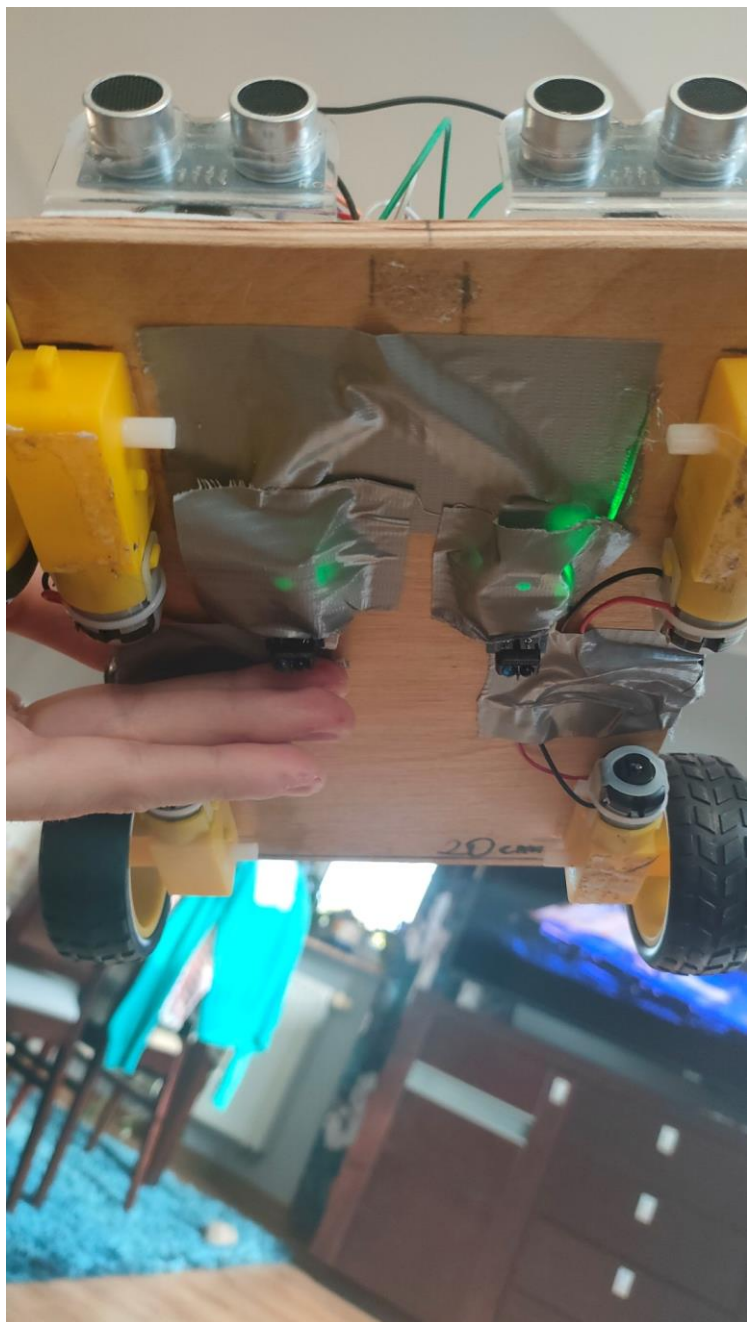
Funkcja main() rozpoczyna działanie od inicjalizacji HAL, zegara systemowego oraz skonfigurowania GPIO i timera. W pętli nieskończonej kod odczytuje stan przycisku i kiedy wykryje naciśnięcie, wywołuje sekwencję ruchów robota. Po wykonaniu sekwencji program wraca do oczekiwania na kolejne naciśnięcie przycisku.

## 6. Zdjęcia opracowanego robota









Link do filmiku jak jedzie: <https://youtu.be/2g-l-w1x-kc>