



## Milestone 2

Zmiany wprowadzonych usprawnień, modyfikacji sprzętu zostały podkreślone reszta została bez zmian z poprzedniej części Milestone.

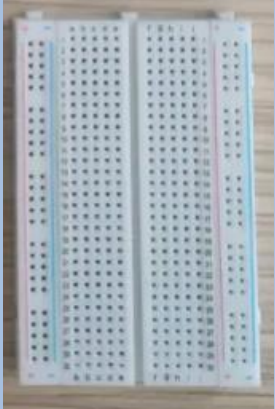



### 1. Zwięzły opis robota

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 odbłaskowy 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 plączą 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

- Przykleiłem je na przedniej lewej i prawej stronie robota, za pomocą kleju Pattex (do klejenia drewna i innych tworzyw sztucznych).
- Przewody poprowadzone przez otwór w deseczce, dzięki czemu nie przeszkadzają w jeździe.

#### Czujniki ultradźwiękowe HC-SR04

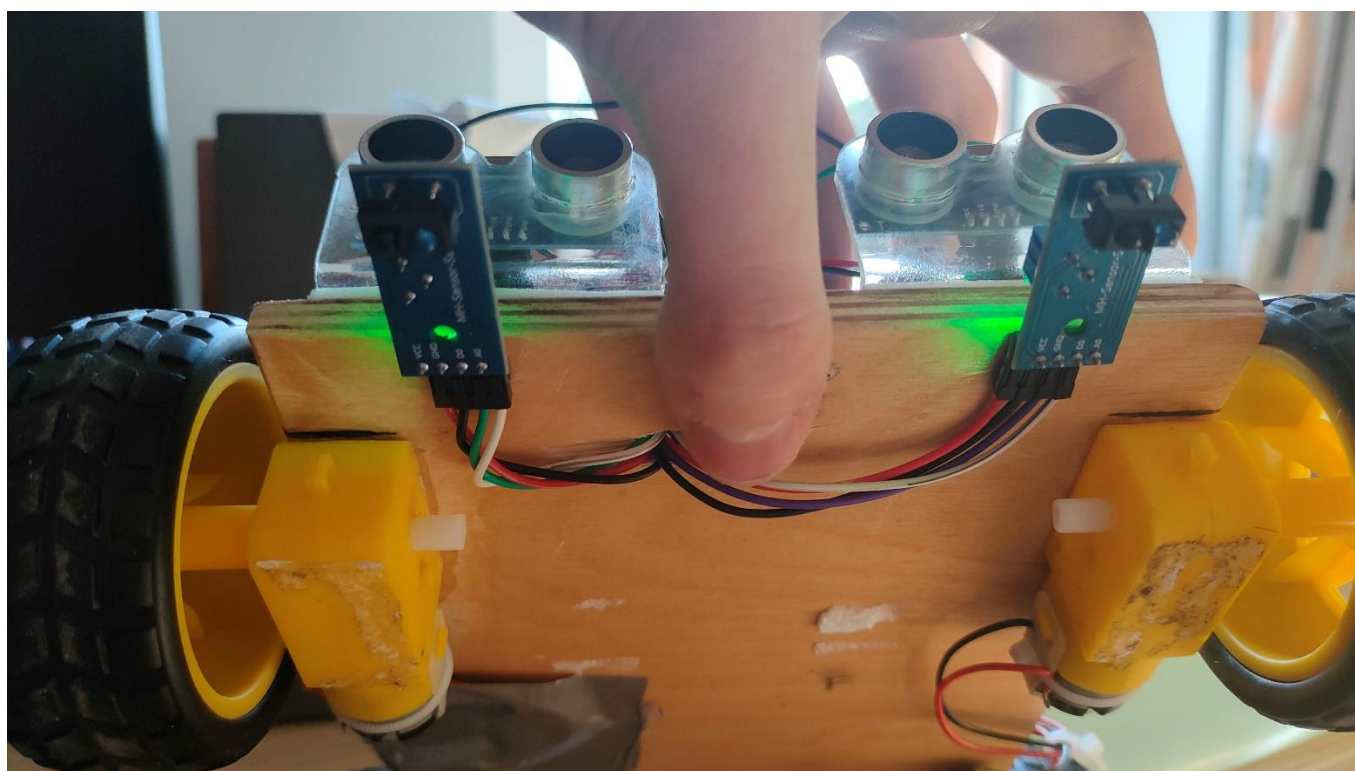
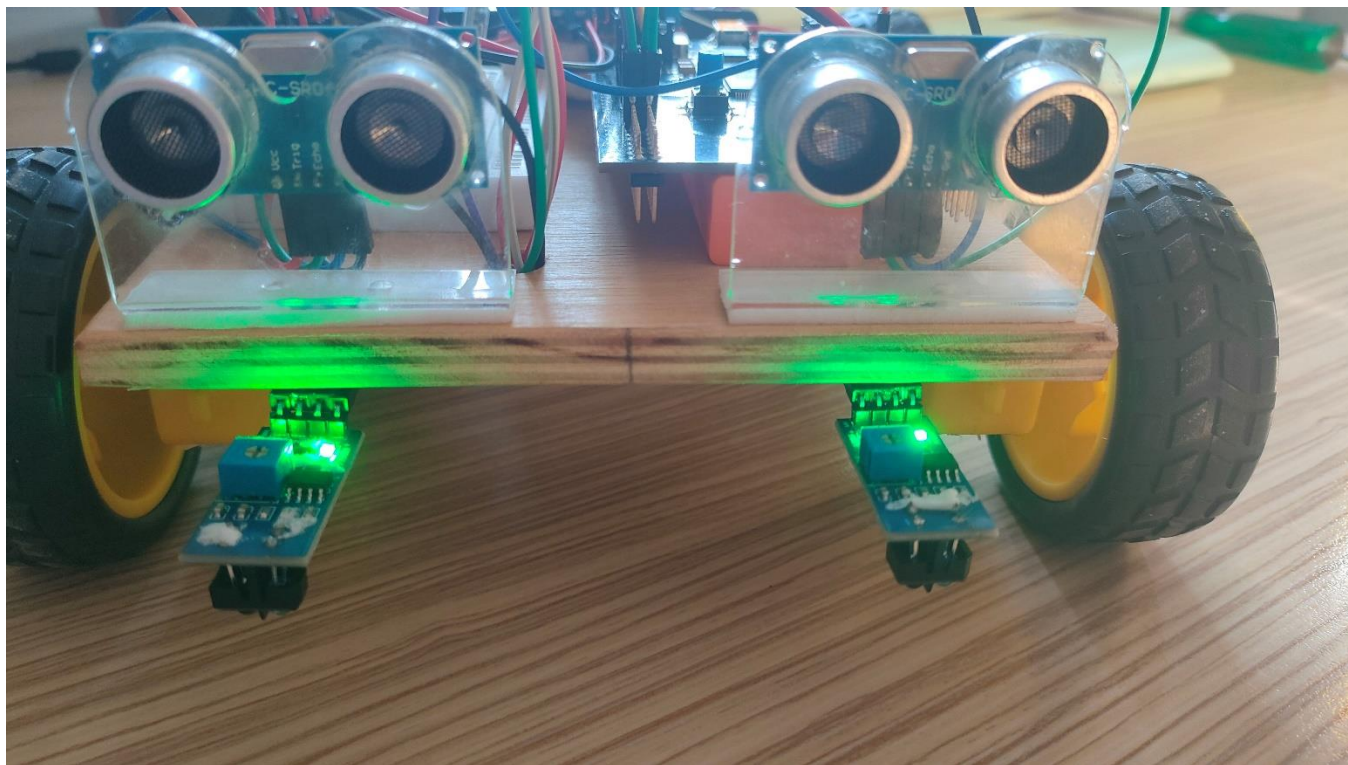
- Przyklejona podstawa specjalnie do tych czujników za pomocą taśmy dwustronnej i żeby nie spadały przykleiłem je klejem.

#### Okablowanie

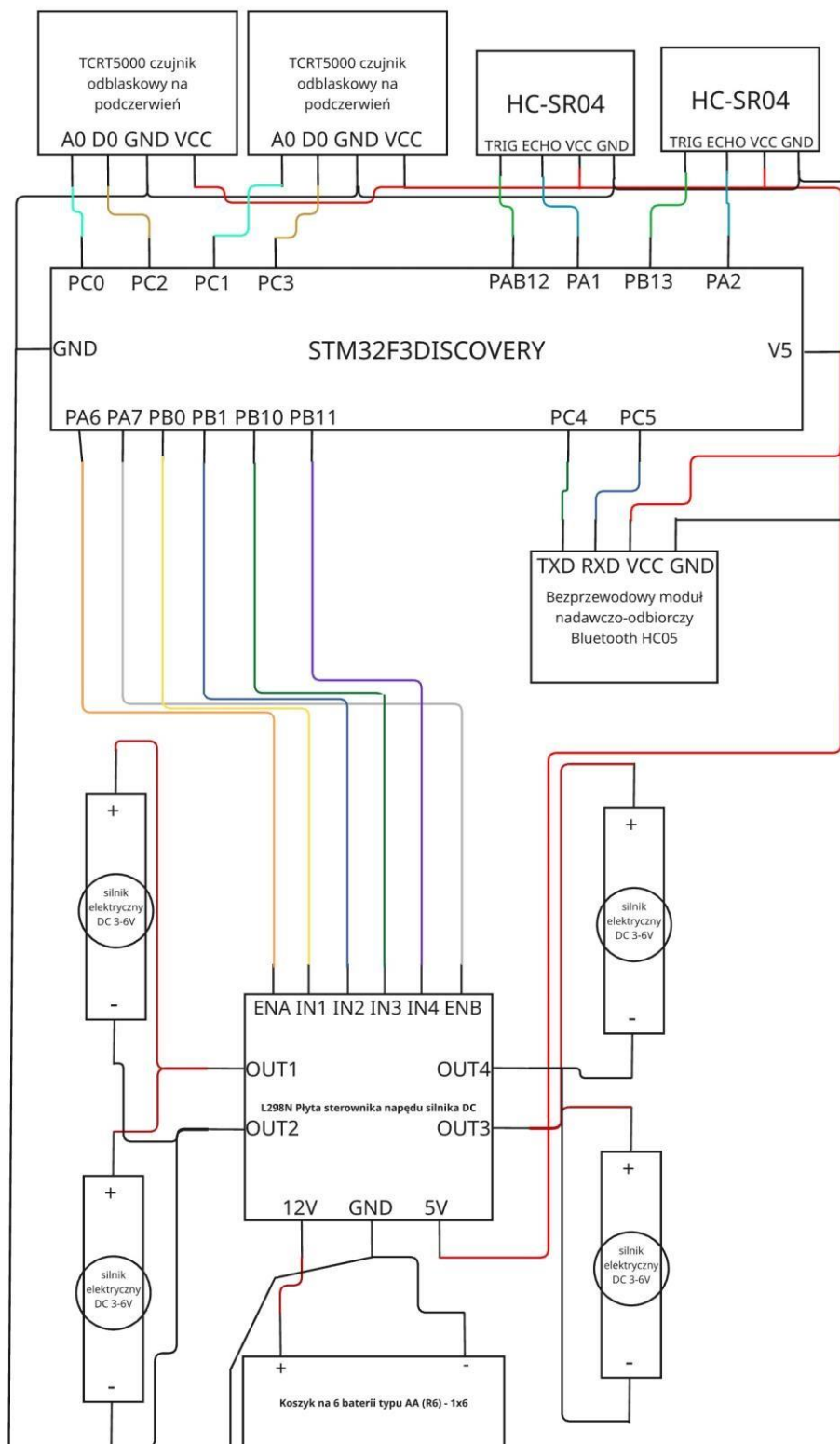
- Wszystkie przewody sygnałowe i zasilające zakończyłem złączkami żeńsko-męskimi, żeńsko-żeńskimi i męsko-męskimi.
- Prowadzenie kabli wzdłuż krawędzi deseczki zabezpieczyłem punktowo taśmą, żeby nic się nie plątało pod kołami.



Poniżej zdjęcia modyfikacji:

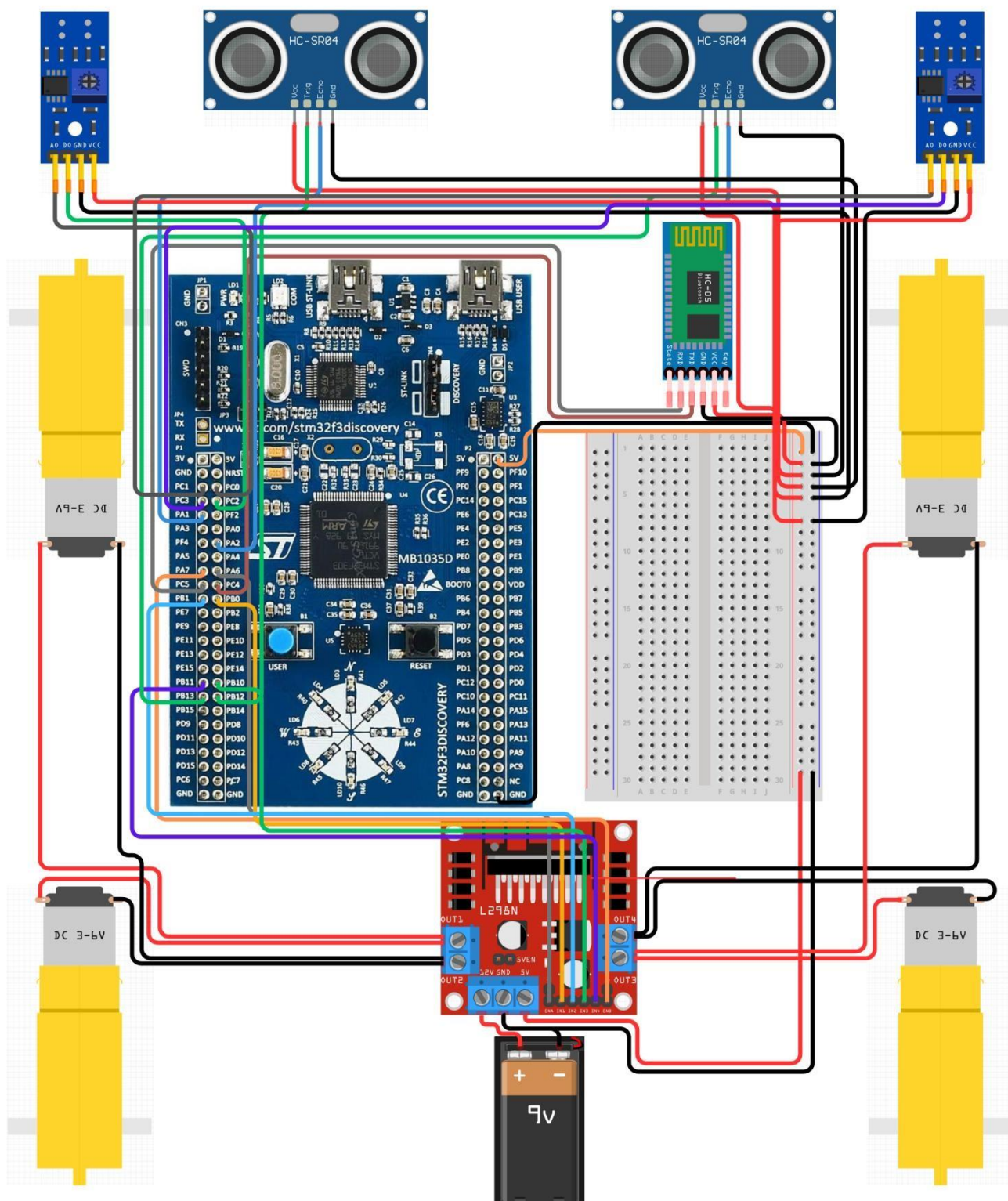


#### 4. Schemat elektroniczny robota



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

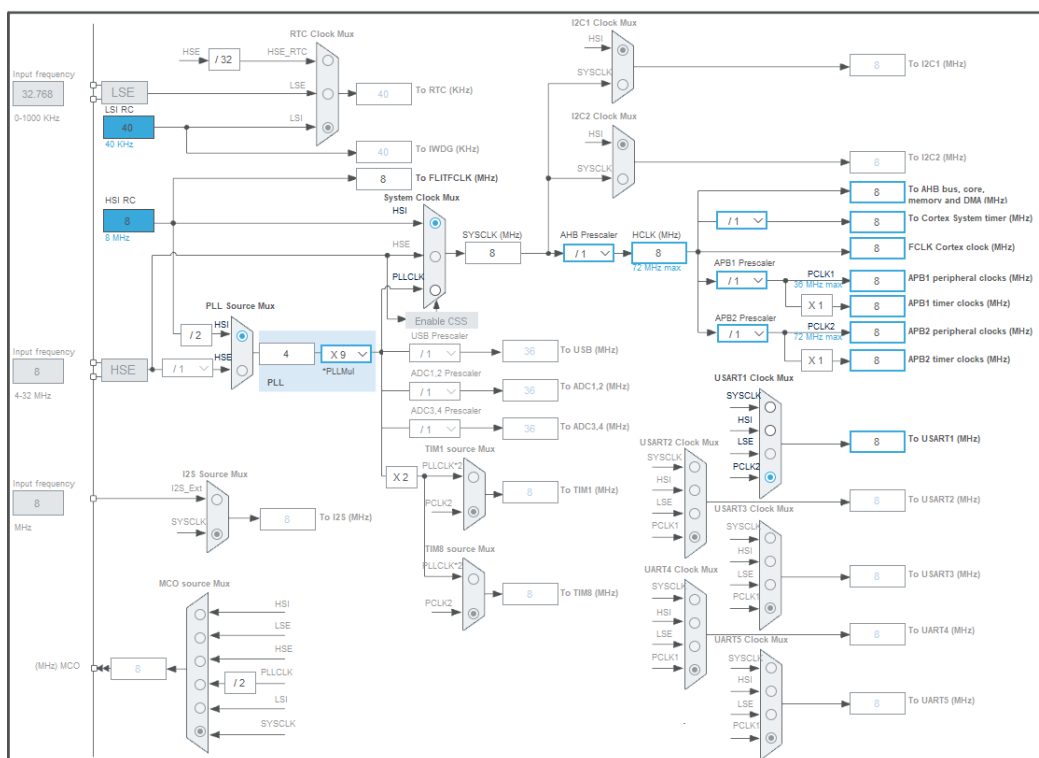
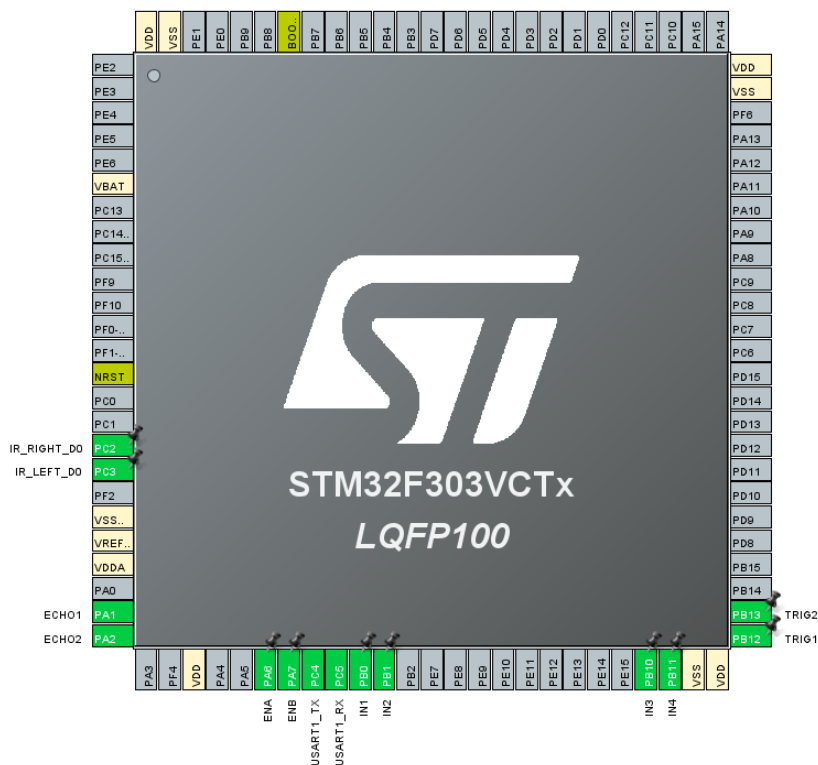




Rys. 4.2: Schemat montażowy robota z płytą stykową (zrobiony za pomocą programu Canva).



## 5. Oprogramowanie sterujące



Categories

A-Z

System Core >

Analog >

Timers >

RTC

TIM1

TIM2

**TIM3**

TIM4

TIM6

TIM7

TIM8

TIM15

TIM16

TIM17

Connectivity >

CAN

I2C1

I2C2

IRTIM

SPI1

SPI2

SPI3

UART4

UART5

USART1

USART2

USART3

USB

Multimedia >

Computing >

Middleware and Software Packs >

Mode

Slave Mode

Disable

Trigger Source

Disable

Clock Source

Internal Clock

Channel1

PWM Generation CH1

Channel2

PWM Generation CH2

Channel3

Disable

Channel4

Disable

Combined Channels

Disable

☐ ETR IO as Clearing Source

☐ XOR activation

☐ One Pulse Mode

Configuration

Reset Configuration

**DMA Settings**

**GPIO Settings**

**Parameter Settings**

**User Constants**

**NVIC Settings**

Configure the below parameters :

Search (Ctrl+F)

Counter Settings

Prescaler (PSC - 16 bi...

7

Counter Mode

Up

Counter Period (AutoR...

99

Internal Clock Division ...

No Division

auto-reload preload

Disable

Trigger Output (TRGO) Param...

Master/Slave Mode (M...

Disable (Trigger input effect not d...

Trigger Event Selectio...

Reset (UG bit from TIMx\_EGR)

Clear Input

Clear Input Source

Disable

PWM Generation Channel 1

Mode

PWM mode 1

Pulse (16 bits value)

0

Output compare preload

Enable

Fast Mode

Disable





USART1 Mode and Configuration

Categories A-Z

System Core

- DMA
- ✓ GPIO
- IWDG
- ✓ NVIC
- RCC
- ✓ SYS
- ⚠ TSC
- WWDG

Analog

Timers

- RTC
- TIM1
- ✓ TIM2
- ✓ TIM3
- TIM4
- TIM6
- TIM7
- TIM8
- TIM15
- TIM16
- TIM17

Connectivity

- CAN
- I2C1
- I2C2
- IRTIM
- SPI1
- SPI2
- SPI3
- UART4
- UART5
- ✓ USART1
- USART2
- USART3
- USB

Multimedia

Computing

Mode

Mode Asynchronous

Hardware Flow Control (RS232) Disable

☐ Hardware Flow Control (RS485)

Configuration

Reset Configuration

✓ NVIC Settings

✓ DMA Settings

✓ GPIO Settings

✓ Parameter Settings

✓ User Constants

Configure the below parameters :

Search (Ctrl+F)

Basic Parameters

- Baud Rate 9600 Bits/s
- Word Length 8 Bits (including Parity)
- Parity None
- Stop Bits 1

Advanced Parameters

- Data Direction Receive and Transmit
- Over Sampling 16 Samples
- Single Sample Disable

Advanced Features

- Auto Baudrate Disable
- TX Pin Active Level Inver... Disable
- RX Pin Active Level Inve... Disable
- Data Inversion Disable
- TX and RX Pins Swapping Disable
- Overrun Enable



Fragmety kodu pochodzące z pliku main.c:

```
#define PREDKOSC_MIN    0
#define PREDKOSC_MAX    100
#define KROK_PREDKOSCI  10
volatile uint8_t Auto=0;
volatile uint8_t procentPredkosci=10;
char znak;
```

Na początku ustawiono trzy makra: PREDKOSC\_MIN i PREDKOSC\_MAX określają minimalny i maksymalny procent wypełnienia PWM (0–100 %), a KROK\_PREDKOSCI definiuje, o ile procent zmienia się prędkość przy każdej komendzie. Zadeklarowano dwie zmienne typu volatile uint8\_t: Auto (flaga trybu automatycznego) i procentPredkosci (bieżący procent wypełnienia), oraz bufor na pojedynczy znak.

```
void aktualizujPredkosc(void)
{
    uint32_t predkosc = (htim3.Init.Period + 1)*procentPredkosci/100;
    __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_1, predkosc);
    __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_2, predkosc);
}
```

Funkcja aktualizujPredkosc() przelicza wartość rejestru porównawczego: (htim3.Init.Period + 1) \* procentPredkosci / 100, co zapewnia liniową zależność wypełnienia sygnału od procentu. Wynik przypisywany jest jednocześnie do obu kanałów timera TIM3 (kanały 1 i 2) przez makra HAL:

```
__HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_x, predkosc);
```

```
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)
{
    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);
    aktualizujPredkosc();
}
```

Funkcja jedzNaprzod() ustawia piny kierunkowe tak, aby oba silniki jechały naprzód (IN1=1, IN2=0; IN3=1, IN4=0), a następnie wywołuje aktualizujPredkosc(), która oblicza i uruchamia PWM o zadanym procentowym wypełnieniu – w efekcie robot rusza do przodu.



```
void jedzDoTyłu(void)
{
    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);
    aktualizujPredkosc();
}
```

Funkcja jedzDoTyłu() ustawia piny kierunkowe odwrotnie (IN1=0, IN2=1; IN3=0, IN4=1), dzięki czemu oba silniki kręcą się w tył, a potem wywołuje aktualizujPredkosc(), aby nadać im zadaną moc – robot jedzie do tyłu.

```
void skrecWLewo(void)
{
    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);
    aktualizujPredkosc();
}
```

Funkcja skrecWLewo() Ustawia lewy silnik do przodu (IN1=1, IN2=0) i prawy do tyłu (IN3=0, IN4=1), a następnie wywołuje aktualizujPredkosc(), co powoduje obrót robota w lewo z aktualnie ustawioną prędkością.

```
void skrecWPrawo(void)
{
    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);
    aktualizujPredkosc();
}
```

Funkcja skrecWPrawo() ustawia lewy silnik do tyłu (IN1=0, IN2=1) i prawy do przodu (IN3=1, IN4=0), a potem uruchamia aktualizujPredkosc(), wprawiając robota w obrót w prawo.

```
void zatrzymajRobota(void)
{
    zatrzymajSilniki();
    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);
}
```

Funkcja zatrzymajSilniki() ustawia wszystkie piny kierunkowe IN1–IN4 na 0, odcinając zasilanie silników; nie wywołuje PWM, dzięki czemu robot natychmiast się zatrzymuje.



```
void jedzPoLinii(void)
{
    aktualizujPredkosc();
    uint8_t lewySensor = HAL_GPIO_ReadPin(IR_LEFT_D0_GPIO_Port, IR_LEFT_D0_Pin);
    uint8_t prawySensor = HAL_GPIO_ReadPin(IR_RIGHT_D0_GPIO_Port, IR_RIGHT_D0_Pin);

    if(lewySensor==0&&prawySensor==0)
    {
        jedzNaprzod();
    }
    else if(lewySensor==1&&prawySensor==0)
    {
        skrecWLewo();
    }
    else if(lewySensor==0&&prawySensor==1)
    {
        skrecWPrawo();
    }
    else
    {
        zatrzymajRobota();
    }
}
```

W funkcji jedzPoLinii() odczytywane są dwa cyfrowe wejścia IR: lewySensor = HAL\_GPIO\_ReadPin(IR\_LEFT\_D0\_GPIO\_Port, IR\_LEFT\_D0\_Pin) oraz analogicznie prawySensor. Gdy oba czujniki widzą biały (0/0), jedzie na wprost, jak lewy widzi czarny (1) a prawy biały, skręca w lewo analogicznie kiedy warunek jest odwrotny, skręca w prawo. W pozostałych przypadkach (1/1) robot zatrzymuje się.

```
void obslugaMenu(char znak)
{
    switch (znak)
    {
        case 'M':
            if (procentPredkosci + KROK_PREDKOSCI <= PREDKOSC_MAX) {
                procentPredkosci += KROK_PREDKOSCI;
            } else {
                procentPredkosci = PREDKOSC_MAX;
            }
            aktualizujPredkosc();
            break;

        case 'N':
            if (procentPredkosci >= KROK_PREDKOSCI) {
                procentPredkosci -= KROK_PREDKOSCI;
            } else {
                procentPredkosci = PREDKOSC_MIN;
            }
            aktualizujPredkosc();
            break;
        case 'G':
            jedzNaprzod();
            break;
        case 'T':
```





```
    jedzDoTyłu();  
    break;  
case 'L':  
    skrecwLewo();  
    break;  
case 'P':  
    skrecwPrawo();  
    break;  
case 'S':  
    zatrzymajRobota();  
    break;  
case 'A':  
    Auto = 1;  
    break;  
case 'a':  
    Auto = 0;  
    zatrzymajRobota();  
    break;  
default:  
    break;  
}  
}
```

Obsługa poleceń z UART została wykonana za pomocą funkcji obsługiMenu(char znak) w którym znajduje się switch, reagujący na znaki:

- M zwiększa procentPredkosci o KROK\_PREDKOSCI, aż do PREDKOSC\_MAX;
- N zmniejsza o krok, aż do minimum;
- G, T, L, P, S wywołują odpowiednio: jazdę przód, tył, skręt w lewo, skręt w prawo oraz zatrzymanie;
- A i a włączają lub wyłączają tryb automatyczny (Auto = 1/0) i zatrzymują robot w przypadku wyłączenia.

```
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();  
}
```



```
MX_TIM2_Init();
MX_USART1_UART_Init();
/* USER CODE BEGIN 2 */
HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_1);
HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_2);

aktualizujPredkosc();

HAL_TIM_IC_Start_IT(&htim2, TIM_CHANNEL_2);
HAL_TIM_IC_Start_IT(&htim2, TIM_CHANNEL_3);
HAL_UART_Receive_IT(&huart1, (uint8_t*)&znak, 1);

char *w = "Robot Działa\r\n";
HAL_UART_Transmit(&huart1, (uint8_t*)w, strlen(w), HAL_MAX_DELAY);
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    if (Auto)
    {
        jedzPoLinii();
    }
}
/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */
/* USER CODE END 3 */
}
```

Rozpoczyna działanie od wywołań inicjalizujących HAL (HAL\_Init()), ustawienia zegara systemowego (SystemClock\_Config()) oraz konfiguracji peryferiów (MX\_GPIO\_Init(), MX\_TIM3\_Init(), MX\_USART1\_UART\_Init()). Następnie uruchamia PWM na kanałach 1 i 2 timera TIM3 i wywołuje aktualizujPredkosc(), aby nadać początkową wartość wypełnienia. Aktywuje odbiór pojedynczego bajtu UART w trybie przerwań oraz wysyła komunikat "Robot Działa\r\n". W nieskończonej pętli while(1) sprawdza flagę Auto – jeśli jest włączona (Auto == 1), wywołuje jedzPoLinii(), realizując automatyczne podążanie za linią.

```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
    if (huart->Instance == USART1)
    {
        obslugaMenu(znak);
        HAL_UART_Receive_IT(&huart1, (uint8_t*)&znak, 1);
    }
}
```

Funkcja HAL\_UART\_RxCpltCallback(UART\_HandleTypeDef \*huart) wyłapuje przerwanie po odebraniu znaku na USART1, przekazuje go do obslugaMenu(znak) i ponownie uruchamia odbiór kolejnego bajtu przerwami, co zapewnia ciągłą komunikację.



[https://play.google.com/store/apps/details?id=de.kai\\_morich.serial\\_bluetooth\\_terminal&hl=pl](https://play.google.com/store/apps/details?id=de.kai_morich.serial_bluetooth_terminal&hl=pl)

## Serial Bluetooth Terminal

Kai Morich  
Zakup w aplikacji

4.6★  
3,14 tys. opinii

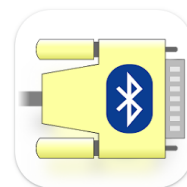
1 min+  
Pobrania

PEGI 3

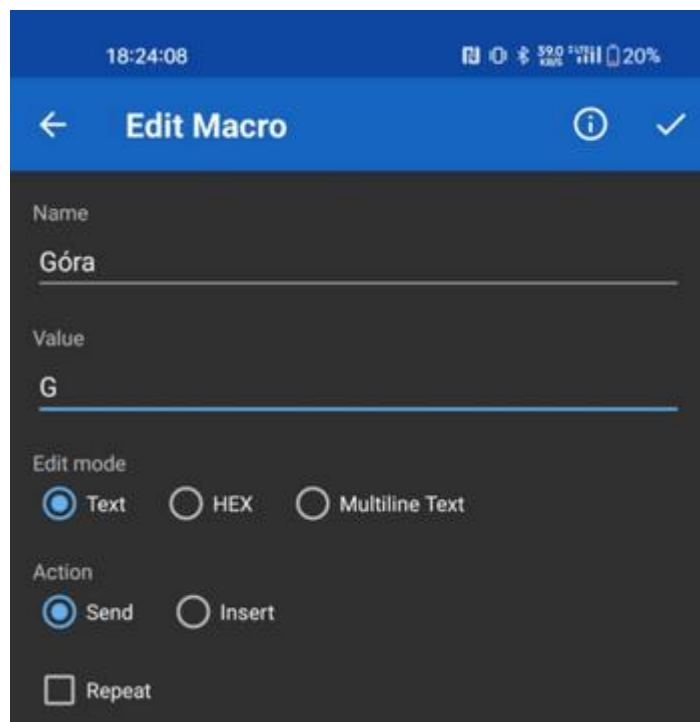
Zainstaluj

Udostępniaj

Dodaj do listy życzeń



**Przykładowa konfiguracja przycisku powyżej aplikacji:**

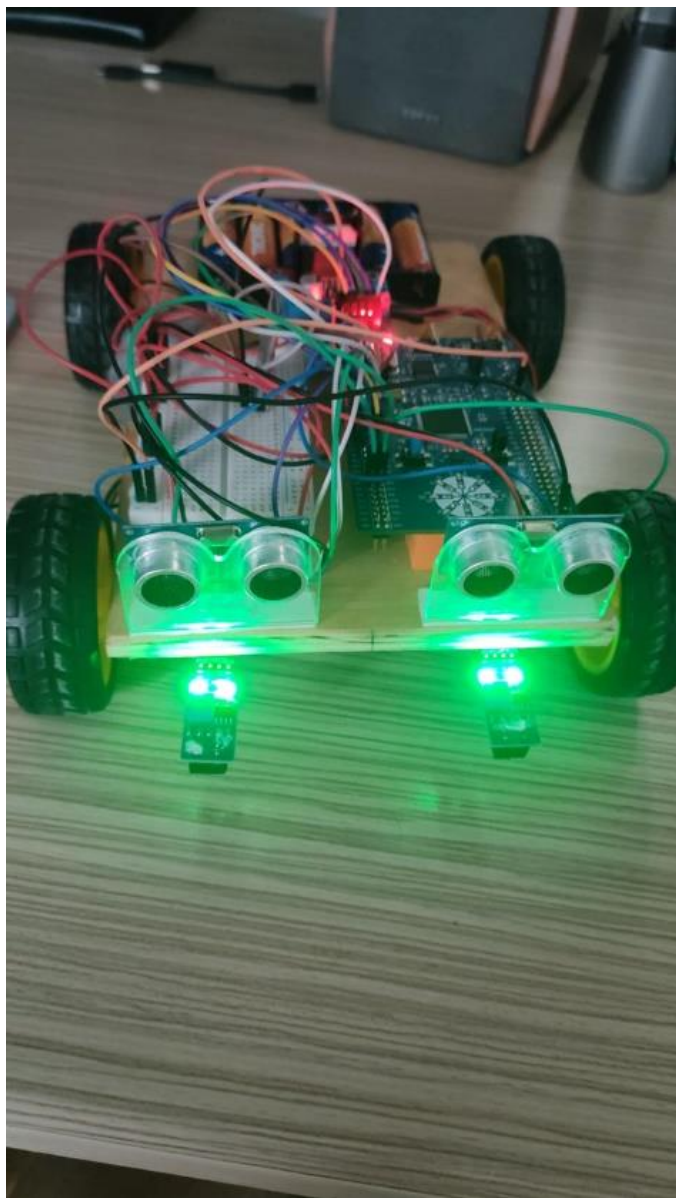


W Serial Bluetooth Terminal przyciski makr zdefiniowano tak, by na każdy tap wysyłały jeden znak ASCII odpowiadający komendzie robota (np. „G” dla ruchu do przodu, „T” dla do tyłu, „L”/„P” dla skrętów, „S” dla stop, „A”/„a” dla włączenia/wyłączenia trybu automatycznego oraz „M”/„N” dla zmiany prędkości). Każde makro w edycji ma ustawiony tryb „Text” i działanie „Send”, bez opcji powtarzania.





## 6. Zdjęcia opracowanego robota





7. Filmy opracowanego robota:

