Systemy wbudowane i mikroprocesory [2024/2025]

Sprawozdanie z pierwszego kroku milowego

Wykonali:

Zuzanna Orzechowska 21284

Adrian Popielarczyk 21295

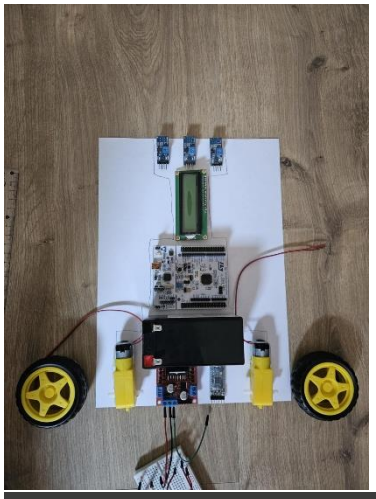
Wymagania podstawowe:

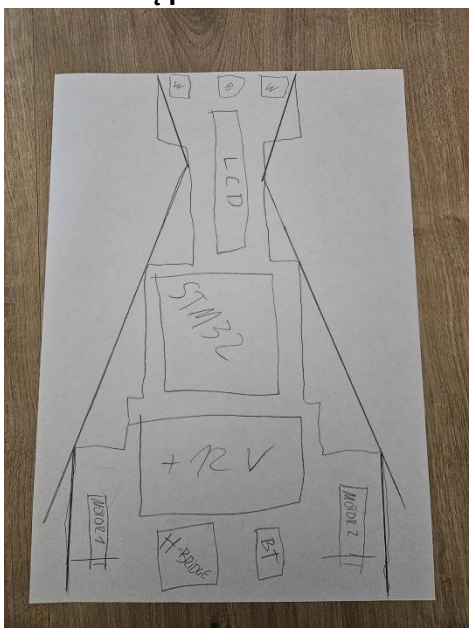
Ukończenie mechaniki robota.

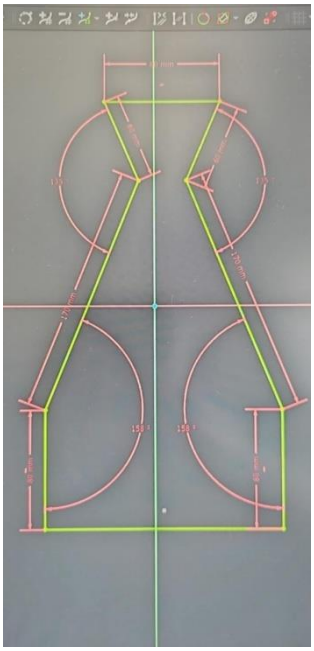
Jazda zaprogramowanej sekwencji złożone z 10 rozkazów (przód, tył, lewo, prawo).

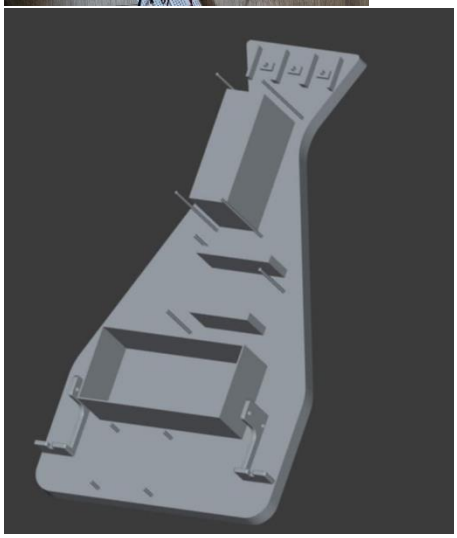
Dokumentacja projektu robota:

Model koncepcyjny wraz z projektem i budową podwozia:

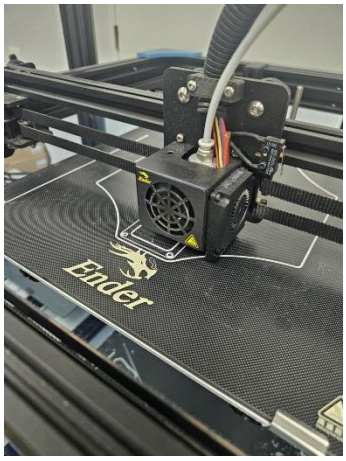


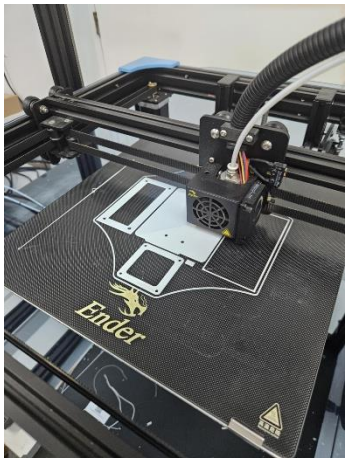


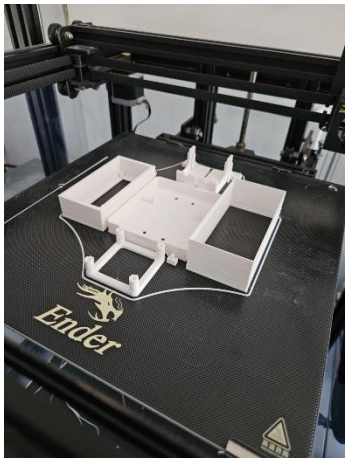




Druk elementów trzymających:







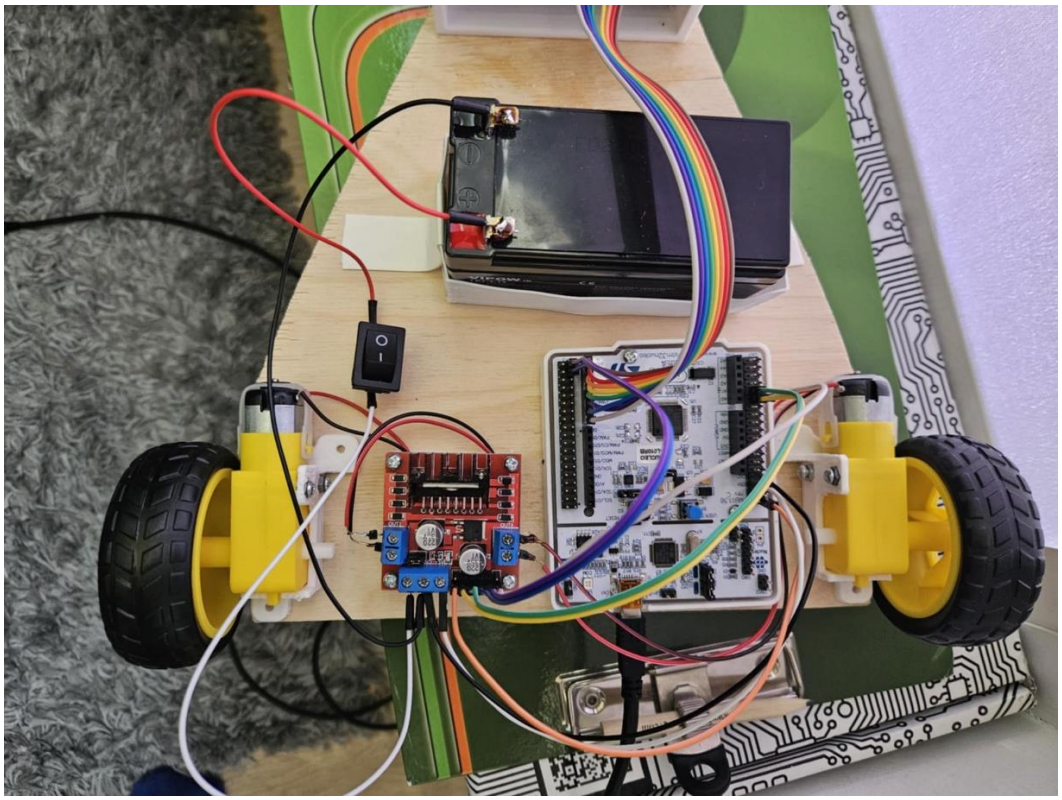
Elementy poza podkładkami pod sensory odbicia, zostały wzięte z poniższych źródeł, autorzy elementów wskazani są również na tych samych stronach:
https://www.printables.com/model/59983-16x2-lcd-with-d1-mini-case/files

https://www.thingiverse.com/thing:6121113/files

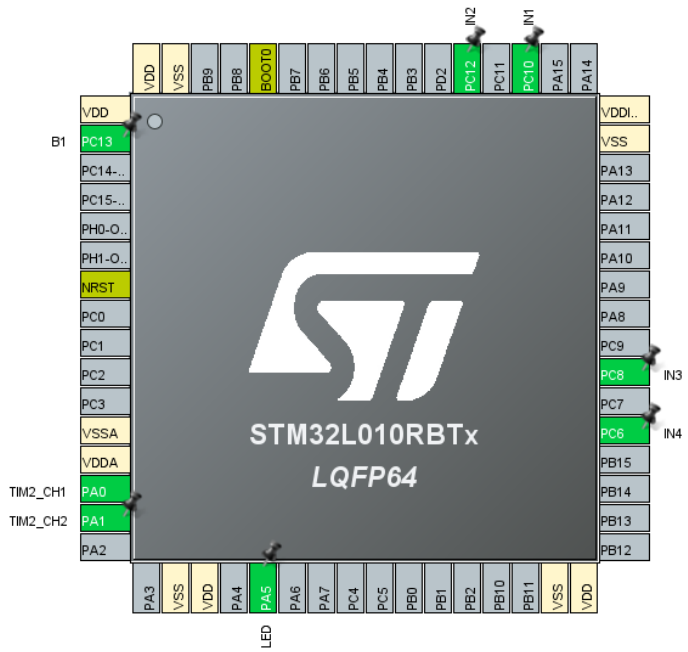
https://www.printables.com/model/161725-din-rail-case-for-stm32-nucleo-64-boards/files

https://www.printables.com/model/552244-sparky-tt-motor-mounts

Efekt prac, po umieszczeniu elementów oraz przymocowaniu kabli:



Konfiguracja pinów:



Użyte części:

- **1x Płytka STM32 Nucleo L010RB**
- **2x Silnik z przekładnią 48:1, o zakresie pracy 3-6V**
- **1x Mostek typu H, model L298n**
- **1x Akumulator żelowy 12V, 1.3Ah**
- **1x Przełącznik On/Off**
- **Kable do łączeń**
- **Połowa korka po winie**

Wyjścia pinów podłączono do odpowiadających deskrypcją pinów na mostku H.

Płytka działa z zegarem 16 MHz, prescaler timerów ustawiono na 0, a maksymalne wypełnienie na 1600. Uzyskano dzięki temu częstotliwość PWM ok. 10 000Hz.

Kod programu głównego, wykonujący 10 sekwencji po kolei:

Kod w sekcji USER CODE WHILE:

```
while (1)
{

while(HAL_GPIO_ReadPin(GPIOC, B1_Pin) == GPIO_PIN_SET) //następuje odczytanie stanu przycisku na porcie C, jeśli on zostanie
//wciśnięty i zaczyna rozpoczynanie sekwencji
{
HAL_GPIO_TogglePin(GPIOA, LED_Pin); // po uruchomieniu - włączeniu przycisku dioda LED zaczyna świecić z opóznieniem 200ms
HAL_Delay(200);
}

//następuje reset wszystkich pinów dla kierunków kanalu mostka H na porcie C - początkowo motory sterujące ruchem kół zostają wyłączone
//stąd stan GPIO_PIN_RESET
HAL_GPIO_WritePin(GPIOC, IN1_Pin, GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOC, IN2_Pin, GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOC, IN3_Pin, GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOC, IN4_Pin, GPIO_PIN_RESET);
```

```
//następuje inicjalizacja PWM na dwóch kanałach Timera - pierwszym i drugim, dla dwóch motorów
HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_2);

// po inicjalizacji początkowe wartości wartości dla obu kanałów ustawione zostały na 0,
// timer i PWN są aktywowane, ale początkowo koła nie dostają zadnych sygnałów, tak zwany startowy stan bezpieczny
__HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, 0);
__HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_2, 0);

//po udanym odpaleniu zaświeca się dioda Led na pinie PA5 - wskazuje na gotowość działania robota i poruszania się
HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_SET);

uint32_t duty = 1200; //aby koła mogły się poruszać ustawiono sygnał dla PWM na 1200 - wskazuje jak szybko mają się poruszać koła, w tym przypadku będą się one poruszały z 75% prędkości maksymalnego wypelnienia = 1600
for (int i = 0; i < 10; ++i) // pętla przechodzi przez wszystkie 10 sekwencji
{
switch(i)
{
//ponizej znajdują się kody wykonujące 10 rozkazów sekwencji ruchu pojazdu
case 0: // pierwsza sekwencja - jazda do przodu, IN1 i IN2 są odpowiedzialne za lewy motor, IN3 i IN4 są odpowiedzialne za prawy motor
HAL_GPIO_WritePin(GPIOC, IN1_Pin, GPIO_PIN_SET); //IN 1 ma wartość 1
HAL_GPIO_WritePin(GPIOC, IN2_Pin, GPIO_PIN_RESET); // IN2 ma wartość 0, zgodnie z zasadą działania mostka H
//taka kofiguracja wskazuje ze sygnał PWM "idzie" w jednym kierunku, co sprawia, ze motor obraca się do przodu
HAL_GPIO_WritePin(GPIOC, IN3_Pin, GPIO_PIN_SET); // taka sama kofnigracja występuje dla prawego motoru
HAL_GPIO_WritePin(GPIOC, IN4_Pin, GPIO_PIN_RESET);
__HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, duty); //ustawienie wypełnienia sygnału PWM na 1200 - koła nabierają 75% maksymalnej prędkości
__HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_2, duty); // to samo dzieje się dla drugiego motora podłączonego do kanału 2
HAL_Delay(2500); //taka ppjedyncza sekwencja trwa 2,5 sekundy
break;
case 1: //druga sekwencja - jazda do tyłu - wartości są odwrotne co do tych z jazdy do przodu, czyli
HAL_GPIO_WritePin(GPIOC, IN1_Pin, GPIO_PIN_RESET); // wartość 0 oraz 1 sprawia, ze sygnal PWM "idzie" w drugim kierunku kanału - następuje obrót kół do tyłu
HAL_GPIO_WritePin(GPIOC, IN2_Pin, GPIO_PIN_SET);
HAL_GPIO_WritePin(GPIOC, IN3_Pin, GPIO_PIN_RESET); // to samo ustawienie dla drugiego motoru
HAL_GPIO_WritePin(GPIOC, IN4_Pin, GPIO_PIN_SET);
__HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, duty); //wypełnienie pozostaje to samo - 1200
__HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_2, duty);
HAL_Delay(2500);
break;
case 2: //trzecia sekwencja - skręt w lewo, tylko jedne pin IN2 od lewego motoru jest uwstawiony na 1
HAL_GPIO_WritePin(GPIOC, IN1_Pin, GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOC, IN2_Pin, GPIO_PIN_SET); //sterowanie drugim tranzystorem dla lewego motoru, silnik nie porusza się, ale obraca w odpowiednią stronę
HAL_GPIO_WritePin(GPIOC, IN3_Pin, GPIO_PIN_RESET); //prawy motor nie dostaje zadnych sygnałow
HAL_GPIO_WritePin(GPIOC, IN4_Pin, GPIO_PIN_RESET);
__HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, 0); //lewy motor nie dostaje sygnału PWM który sprawiłby, ze ten zacznie sie obracac
__HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_2, duty);
HAL_Delay(1000); // sekwencja trwa 1 sekunde
break;
case 3: // czwarta sekwencja - skręt w prawo - działa to na podobnej zasadzie co skręt w lewo
HAL_GPIO_WritePin(GPIOC, IN1_Pin, GPIO_PIN_SET);
HAL_GPIO_WritePin(GPIOC, IN2_Pin, GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOC, IN3_Pin, GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOC, IN4_Pin, GPIO_PIN_RESET);
__HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, duty);
```

```
__HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_2, 0);
HAL_Delay(1000);
break;
case 4: //sekwencja HARD STOP - robot zatrzymuje sie, wszystkie piny otrzymują sygnał 0
HAL_GPIO_WritePin(GPIOC, IN1_Pin, GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOC, IN2_Pin, GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOC, IN3_Pin, GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOC, IN4_Pin, GPIO_PIN_RESET);
__HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, duty);
__HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_2, duty);

HAL_Delay(3000); //zatrzymanie trwa 3 sekundy
break;
case 5: // sekwencja - lewy silnik do przodu, prawy do tyłu – obrót w miejscu (lewo)
HAL_GPIO_WritePin(GPIOC, IN1_Pin, GPIO_PIN_SET); //lewy silnik porusza się do przodu, z racji iz sygnał z PWM
przebiega w jednym kierunku
HAL_GPIO_WritePin(GPIOC, IN2_Pin, GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOC, IN3_Pin, GPIO_PIN_RESET); // prawy silnik obraca się do tylu, sygnał z PWM przebiega w
odwrotnym kierunku
HAL_GPIO_WritePin(GPIOC, IN4_Pin, GPIO_PIN_SET);
__HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, duty);
__HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_2, duty);
HAL_Delay(2500);
break;
case 6: // sekwencja lewy silnik do tyłu, prawy do przodu – obrót w miejscu (prawo)
HAL_GPIO_WritePin(GPIOC, IN1_Pin, GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOC, IN2_Pin, GPIO_PIN_SET);
HAL_GPIO_WritePin(GPIOC, IN3_Pin, GPIO_PIN_SET);
HAL_GPIO_WritePin(GPIOC, IN4_Pin, GPIO_PIN_RESET);
__HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, duty);
__HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_2, duty);
HAL_Delay(2500);
break;
case 7: // sekwencja jazdy do przodu wolniej
HAL_GPIO_WritePin(GPIOC, IN1_Pin, GPIO_PIN_SET);
HAL_GPIO_WritePin(GPIOC, IN2_Pin, GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOC, IN3_Pin, GPIO_PIN_SET);
HAL_GPIO_WritePin(GPIOC, IN4_Pin, GPIO_PIN_RESET);
__HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, duty / 2); // prędkość jest dwukrotnie ograniczona, stąd
motory poruszają się dwa razy wolniej
__HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_2, duty / 2);
HAL_Delay(4000); // sekwencja traw 4 sekundy
break;
case 8: // sekwencja jazdy do przodu szybciej
HAL_GPIO_WritePin(GPIOC, IN1_Pin, GPIO_PIN_SET);
HAL_GPIO_WritePin(GPIOC, IN2_Pin, GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOC, IN3_Pin, GPIO_PIN_SET);
HAL_GPIO_WritePin(GPIOC, IN4_Pin, GPIO_PIN_RESET);
__HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, duty*1.20); // prędkość jest zwiększona do ok 95%
makwymalnego wypełnienia PWM
__HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_2, duty*1.20);
HAL_Delay(1000); // sekwencja trwa 1 sekunde
break;
case 9: // sekwencja łagodnego SOFT STOP zatrzymania z mrugnięciem LEDem
HAL_GPIO_WritePin(GPIOC, IN1_Pin, GPIO_PIN_SET); // wszystkie piny dla obu motorów otrzynują sygnał 1
HAL_GPIO_WritePin(GPIOC, IN2_Pin, GPIO_PIN_SET);
HAL_GPIO_WritePin(GPIOC, IN3_Pin, GPIO_PIN_SET);
HAL_GPIO_WritePin(GPIOC, IN4_Pin, GPIO_PIN_SET);
__HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, 0); // oba timery nie dostają zadnego sygnału - PWM
ustawiony na 0
__HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_2, 0);
HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_SET); // Led zapala się i po 0.2 sekundach zgasza się
```

```
HAL_Delay(200);
HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_RESET); // po mrugnięciu LED zostaje wyłączony
break;
}



}

/* USER CODE END WHILE */

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

Wnioski: Robot bez problemu wykonuje wszystkie 10 sekwencji po podłączeniu prostownika do akumulatora żelowego. Akumulator żelowy użyty do zasilania jest najprawdopodobniej mocno zużyty gdyż dopiero po podłączeniu go pod prostownik z napięciem 13,5V i korzystaniu z napięcia odpowiadającemu jego znamionowej, umożliwia to ruch robota bez problemu. Próbowano zastosować dodatkowe osobne zasilanie dla płytki STM32, jednakże nie przyniosło to oczekiwanych rezultatów, w związku z czym potrzebny będzie inny akumulator żelowy, o większym napięciu, najlepiej w granicach 14V. Zaprogramowane sekwencje będą mogły zostać użyte do dalszych prac nad robotem.