Systemy Wbudowane i Mikroprocesory 2024/2025

Raport z budowy robota

Szymon Muttka, Michał Mazurek, Szymon Makutonowicz

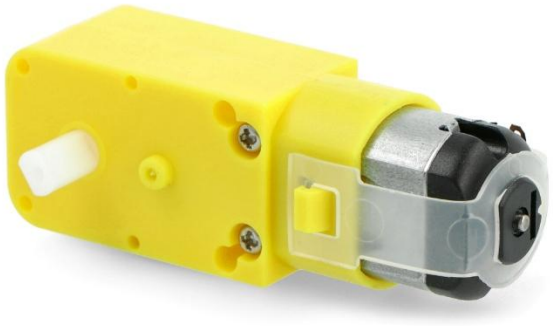
1. Opis robota

Projekt zakładał stworzenie pojazdu strażackiego o nazwie „Jelcz”.
Niestety nie udało nam się zrealizować wszystkich celów, które wyznaczyliśmy sobie na początku.
Początkowo projekt wydawał się prosty, jednak z czasem przerósł nasze umiejętności i wiedzę.
Dodatkowo, błędy organizacyjne – takie jak brak dokładnego planowania poszczególnych etapów czy niewłaściwe zarządzanie czasem – sprawiły, że projekt napotykał liczne trudności, a niektóre jego elementy musieliśmy wykonywać od nowa.
Mimo tych problemów, wynieśliśmy z pracy nad projektem wiele cennych doświadczeń:
Poszerzyliśmy swoją wiedzę na temat działania elektroniki oraz typowych błędów, których należy unikać przy jej projektowaniu.
Opanowaliśmy podstawy obsługi i programowania płytki elektronicznej STM32 Nucleo.
Rozwinęliśmy umiejętność dzielenia złożonych zadań na mniejsze, łatwiejsze do rozwiązania problemy, stosując podejście „dziel i zwyciężaj”.
Nauczyliśmy się pracy zespołowej oraz lepszej organizacji stanowiska pracy i systematyczności.

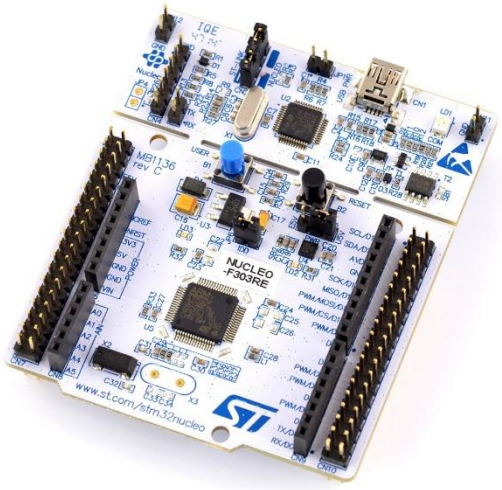
2. Elementy wybrane do budowy robota

Gotowa płyta podłogowa stworzona z pleksi

4x silniki dc
Opis:
- zasilanie dc 3V-9V,
- Koła z oponą o średnicy 65 mm i szerokości 26 mm,
- Przekładnia 48:1,
- Prędkość obrotowa 80 obr/min



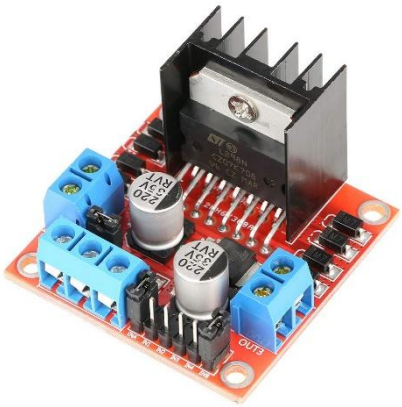
Płytka STM 32F303RE
Link do dokumentacji płytki:
https://www.st.com/resource/en/datasheet/stm32f303re.pdf



2x Sensor optyczny - TCRT5000
Opis:
- Sensor wysyła sygnał niski bądź wysoki, Wysoki-Zanleziono czarny kolor Niski-Nie znaleziono
- Moduł posiada zarówno wyjście analogwe jak i cyfrowe
- Zasilanie 3,3V
- Zasięg 3mm-12mm

Sensor ultradźwiękowy - HC-SR04
- Zasilanie 5V
- Zasięg 2cm do 200cm

Mostek H L298N
- Zasilanie +12V
- Wyjścia 5V
- Możliwość sterowania PWM oraz dwoma kanałami za pomocą wejść logicznych



Regulator napięcia L7805 (Miał on zostać wykorzystany w bazowej wersji projektu)
- Zasilanie +12v
- Wyjście 5V 1.2A

3. Mechanika robota

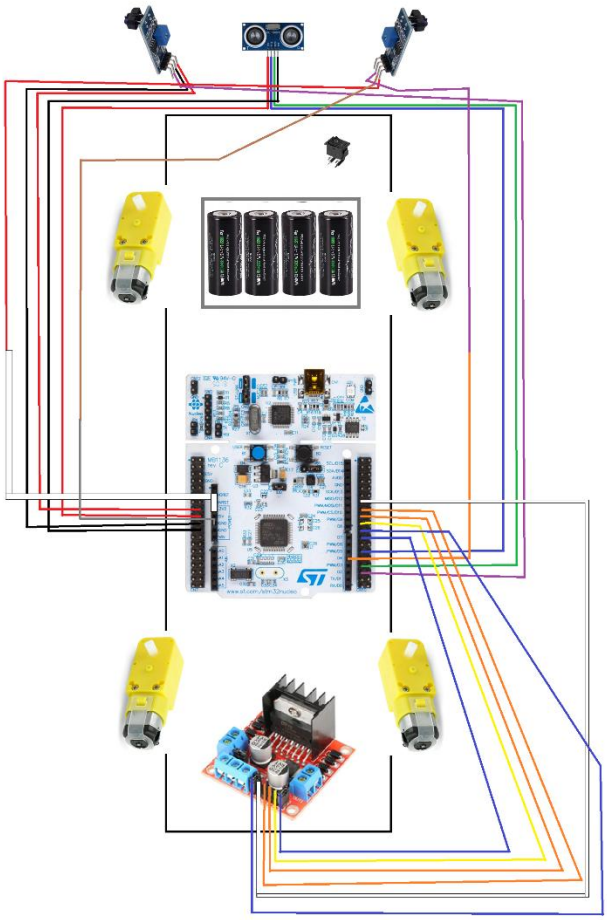
Choć projekt nie wygląda tak, jak początkowo go sobie wyobrażaliśmy, dołożyliśmy starań, aby spełniał wszystkie wymogi zawarte w instrukcji przekazanej przez prowadzącego zajęcia.
Finalnie stworzyliśmy czterokołowy pojazd, który potrafi poruszać się, wykorzystując niezależne sterowanie dwiema parami kół po jednej na każdej stronie. Skręcanie pojazdem, które początkowo wydawało się proste, okazało się znacznie trudniejsze w realizacji, niż przypuszczaliśmy, i to właśnie ten element zajął nam najwięcej czasu. Ostatecznie zdecydowaliśmy się uprościć mechanizm skrętu do minimum. Pojazd skręca dzięki obrotowi jednej pary kół w przeciwnym kierunku do drugiej.
Każdy z silników zasilany jest napięciem około 7–8V, pochodzącym z akumulatorów. Choć jest to nieco poniżej ich maksymalnej sprawności (9V), pozwala to na stabilne działanie. Sterowanie odbywa się za pośrednictwem mostka H.
Na froncie pojazdu zamontowaliśmy czujnik ultradźwiękowy. Dzięki temu udało się zaimplementować funkcję omijania przeszkód znajdujących się na trasie pojazdu. Dodatkowo zamontowaliśmy dwa czujniki optyczne, które wykrywają czarną linię. Umożliwiło to zaprogramowanie funkcji poruszania się po wyznaczonej trasie w oparciu o linię na podłożu.

Pojazdem można także sterować przewodowo za pomocą programu PuTTY.
Projekt zasilany jest szeregowym układem akumulatorów litowo-jonowych 4V, co daje łącznie około 17V. Z kolei płytka STM32 Nucleo zasilana jest osobno z powerbanku podłączonego przez kabel USB.
Chociaż projekt musi już zostać oddany do oceny, planujemy kontynuować jego rozwój. Chcemy dodać funkcjonalności, których nie udało się zrealizować z powodu problemów technicznych i naszych umiejętności, takie jak:
komunikacja Bluetooth,
pompa wodna,
działko wodne sterowane za pomocą serwomechanizmu,
w pełni funkcjonalne podwozie.
obudowa pojazdu zaprojektowana w fusion 360.

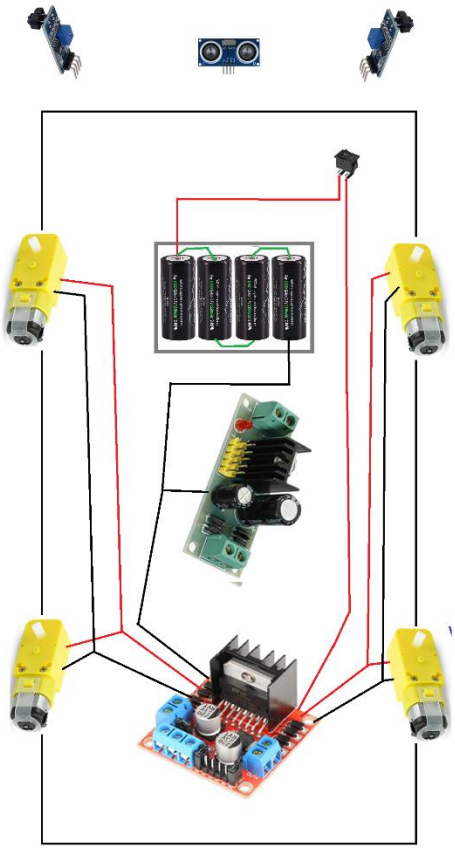
4. Schemat elektroniczny robota

Połączenia fizyczne:

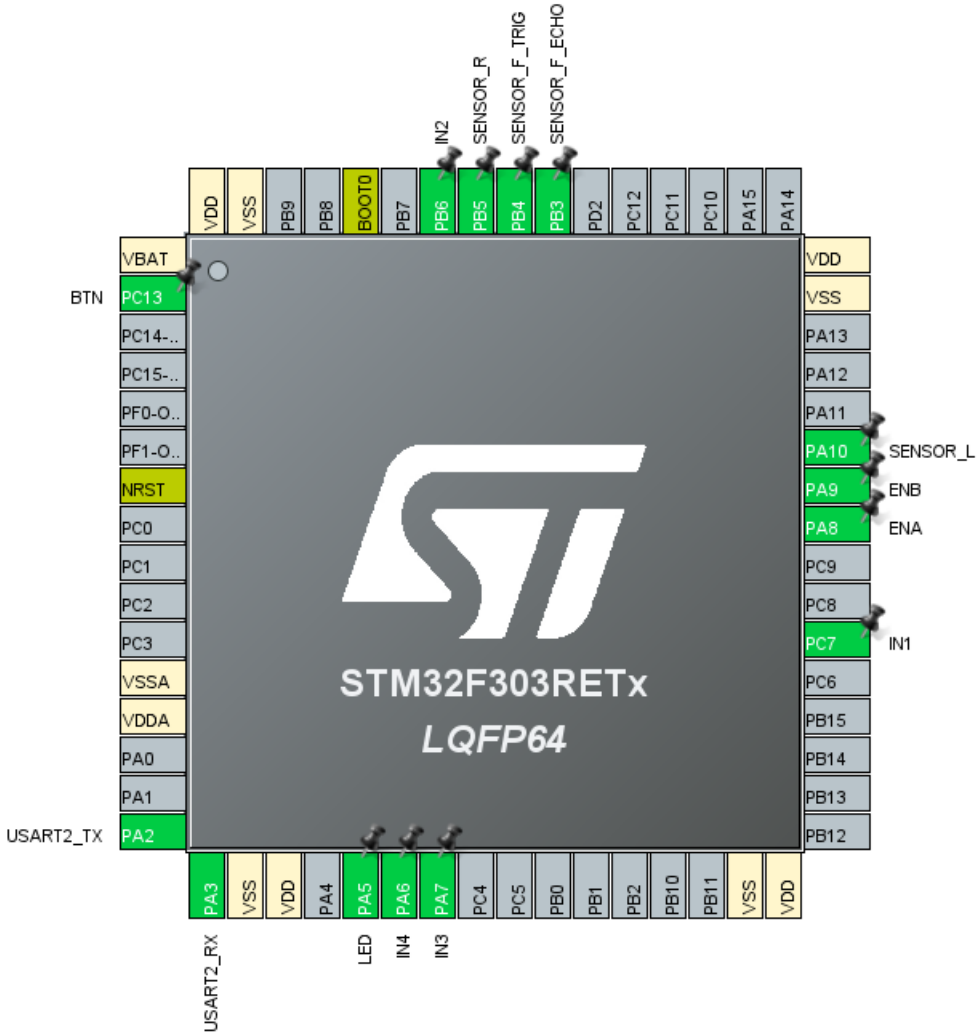
połączenia z płytką



zasilanie



Konfiguracja pinów na płytce:



5. Oprogramowanie sterujące

main.c

Prywatne definicje funkcji

```
/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_TIM1_Init(void);
static void MX_TIM2_Init(void);
static void MX_USART2_UART_Init(void);
/* USER CODE BEGIN PFP */
void L_WHL_Forward();
void L_WHL_Reverse();
void L_WHL_Stop();

void R_WHL_Forward();
void R_WHL_Reverse();
void R_WHL_Stop();

void ALL_WHL_Forward();
void ALL_WHL_Reverse();
void ALL_WHL_Stop();

void Turn_Left();
void Turn_Right();

void WHL_Debug();
/* USER CODE END PFP */
```

Treść funkcji

```
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
void L_WHL_Forward()
{
    HAL_GPIO_WritePin(IN3_GPIO_Port, IN3_Pin, 0);
    HAL_GPIO_WritePin(IN4_GPIO_Port, IN4_Pin, 1);
}

void L_WHL_Reverse()
{
    HAL_GPIO_WritePin(IN3_GPIO_Port, IN3_Pin, 1);
    HAL_GPIO_WritePin(IN4_GPIO_Port, IN4_Pin, 0);
}

void L_WHL_Stop()
{
    HAL_GPIO_WritePin(IN3_GPIO_Port, IN3_Pin, 0);
    HAL_GPIO_WritePin(IN4_GPIO_Port, IN4_Pin, 0);
}

void R_WHL_Forward()
{
    HAL_GPIO_WritePin(IN1_GPIO_Port, IN1_Pin, 0);
    HAL_GPIO_WritePin(IN2_GPIO_Port, IN2_Pin, 1);
}

void R_WHL_Reverse()
{
    HAL_GPIO_WritePin(IN1_GPIO_Port, IN1_Pin, 1);
    HAL_GPIO_WritePin(IN2_GPIO_Port, IN2_Pin, 0);
}

void R_WHL_Stop()
{
    HAL_GPIO_WritePin(IN1_GPIO_Port, IN1_Pin, 0);
    HAL_GPIO_WritePin(IN2_GPIO_Port, IN2_Pin, 0);
}
```

```c
void ALL_WHL_Forward()
{
    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);
}

void ALL_WHL_Reverse()
{
    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);
}

void ALL_WHL_Stop()
{
    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);
}

void Turn_Left()
{
    L_WHL_Reverse();
    R_WHL_Forward();
}

void Turn_Right()
{
    L_WHL_Forward();
    R_WHL_Reverse();
}



void WHL_Debug()
{
    R_WHL_Forward();
    HAL_Delay(1000);
    R_WHL_Stop();
    HAL_Delay(1000);
    R_WHL_Reverse();
    HAL_Delay(1000);
    R_WHL_Stop();
    HAL_Delay(1000);

    L_WHL_Forward();
    HAL_Delay(1000);
    L_WHL_Stop();
    HAL_Delay(1000);
    L_WHL_Reverse();
    HAL_Delay(1000);
    L_WHL_Stop();
    HAL_Delay(1000);

    ALL_WHL_Forward();
    HAL_Delay(1000);
    ALL_WHL_Stop();
    HAL_Delay(1000);
    ALL_WHL_Reverse();
    HAL_Delay(1000);
    ALL_WHL_Stop();
    HAL_Delay(1000);
}
/* USER CODE END 0 */
```

Początkowe ustawienie silników

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_2); //uruchomienie timerów dla silników
__HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_1, 3000);
__HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_2, 3000); //ustawienie prędkości silników (30%)
HAL_TIM_Base_Start(&htim2);
ALL_WHL_Stop();
mode = 0; //tryb pracy robota
```

Pętla while

```
while (1)
{
    //WHL_Debug();
    switch(mode)
    {
        case 0: //sterowanie pojazdem przez UART przewodowo
            while(mode==0)
            {
                uint8_t CMD;
                HAL_UART_Receive(&huart2, &CMD, 1, HAL_MAX_DELAY); //odczytaj wysłaną komende
                switch(CMD)
                {
                    case 'W':
                        HAL_UART_Transmit(&huart2, (uint8_t*)"Command: Forward\r\n", 18, HAL_MAX_DELAY); //wydrukuj wykonaną komende
                        ALL_WHL_Forward();
                        HAL_Delay(300);
                        ALL_WHL_Stop();
                        break;
                    case 'A':
                        HAL_UART_Transmit(&huart2, (uint8_t*)"Command: Turn left\r\n", 20, HAL_MAX_DELAY);
                        Turn_Left();
                        HAL_Delay(300);
                        ALL_WHL_Stop();
                        break;
                    case 'S':
                        HAL_UART_Transmit(&huart2, (uint8_t*)"Command: Reverse\r\n", 18, HAL_MAX_DELAY);
                        ALL_WHL_Reverse();
                        HAL_Delay(300);
                        ALL_WHL_Stop();
                        break;
                    case 'D':
                        HAL_UART_Transmit(&huart2, (uint8_t*)"Command: Turn right\r\n", 21, HAL_MAX_DELAY);
                        Turn_Right();
                        HAL_Delay(300);
                        ALL_WHL_Stop();
                        break;
                    default:
                        HAL_UART_Transmit(&huart2, (uint8_t*)"Unknown command\r\n", 17, HAL_MAX_DELAY);
                        ALL_WHL_Stop();
                        break;
                }
            }
```

```
      case 1: //jazda do przodu aż do wykrycia przeszkody
          while(mode==1)
          {
              HAL_GPIO_WritePin(SENSOR_F_TRIG_GPIO_Port, SENSOR_F_TRIG_Pin, GPIO_PIN_SET);
              HAL_Delay(10);
              HAL_GPIO_WritePin(SENSOR_F_TRIG_GPIO_Port, SENSOR_F_TRIG_Pin, GPIO_PIN_RESET); //rozpoczęcie mierzenia odległości

              while(HAL_GPIO_ReadPin(SENSOR_F_ECHO_GPIO_Port, SENSOR_F_ECHO_Pin)==GPIO_PIN_RESET);
              uint32_t start = __HAL_TIM_GET_COUNTER(&htim2); //czas startu mierzenia odległości

              while(HAL_GPIO_ReadPin(SENSOR_F_ECHO_GPIO_Port, SENSOR_F_ECHO_Pin)==GPIO_PIN_SET);
              uint32_t end = __HAL_TIM_GET_COUNTER(&htim2); //czas końca mierzenia odległości

              uint32_t duration = end - start; //czas wędrówki sygnału od przeszkody z powrotem do czujnika
              float dist_cm = duration * 0.034 / 2; //wzór na dystans od przeszkody w centymetrach

              if(dist_cm <= 25) //cofnij się i skręć w prawo
              {
                  HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, 1);
                  ALL_WHL_Stop();
                  HAL_Delay(500);
                  ALL_WHL_Reverse();
                  HAL_Delay(500);
                  Turn_Right();
                  HAL_Delay(300);
                  ALL_WHL_Stop();
                  HAL_Delay(500);
              }
              else
              {
                HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, 0);
                ALL_WHL_Forward();
              }
          }
          break;

      case 2: //podążanie za czarną linią
          while(mode==2)
          {
              uint8_t SENSOR_L = HAL_GPIO_ReadPin(SENSOR_L_GPIO_Port, SENSOR_L_Pin);
              uint8_t SENSOR_R = HAL_GPIO_ReadPin(SENSOR_R_GPIO_Port, SENSOR_R_Pin);

              if(((SENSOR_L==0) && (SENSOR_R==0)) || ((SENSOR_L==1) && (SENSOR_R==1))) //oba sensory wykrywają lub nie wykrywają linii
              {
                  ALL_WHL_Forward();
                  HAL_Delay(40);
                  ALL_WHL_Stop();
                  HAL_Delay(90);
              }
              if((SENSOR_L==1) && (SENSOR_R==0)) //lewy sensor widzi linię
              {
                  Turn_Left();
              }
              if((SENSOR_L==0) && (SENSOR_R==1)) //prawy sensor widzi linię
              {
                  Turn_Right();
              }
          }
          break;
  }

/* USER CODE END WHILE */
```

stm32f3xx_it.c

Zmiana trybu poprzez wysłanie przerwania naciśnięciem przycisku

```
/**
  * @brief This function handles EXTI line[15:10] interrupts.
  */
void EXTI15_10_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI15_10_IRQn 0 */
  //zmień tryb po naciśnięciu przycisku
    if(HAL_GPIO_ReadPin(BTN_GPIO_Port, BTN_Pin)!=GPIO_PIN_RESET)
    {
        if(mode+1==3)
            mode=0;
        else
            mode++;
    }
  /* USER CODE END EXTI15_10_IRQn 0 */
  HAL_GPIO_EXTI_IRQHandler(BTN_Pin);
  /* USER CODE BEGIN EXTI15_10_IRQn 1 */

  /* USER CODE END EXTI15_10_IRQn 1 */
}
```

6. Zdjęcia opracowanego robota