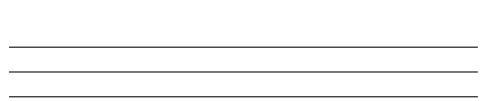


PROJEKT ZESPOŁOWY - ROBOT ZWIADOWCZY

SEKCJA ELEKTRONICZNA

Robot zwiadowczy - Czwarty kamień milowy (06.06.2022)

Igor Giziński, Michał Mendelak



Prowadzący:
dr hab. inż Elżbieta Roszkowska

Katedra Cybernetyki i Robotyki
Wydziału Elektroniki, Fotoniki i
Mikrosystemów
Politechniki Wrocławskiej

Spis treści

1	Charakterystyka kamienia milowego	1
2	Wykonane zadania	1
3	Specyfikacja finalnego produktu	1
4	Cele niezrealizowane	1

1 Charakterystyka kamienia milowego

Czwartym kamieniem milowym dla sekcji elektroników było stworzenie funkcji pozwalających na sterowanie robotem oraz opracowanie sposobu na odbieranie sygnałów z aplikacji przesyłanych poprzez bluetooth.

2 Wykonane zadania

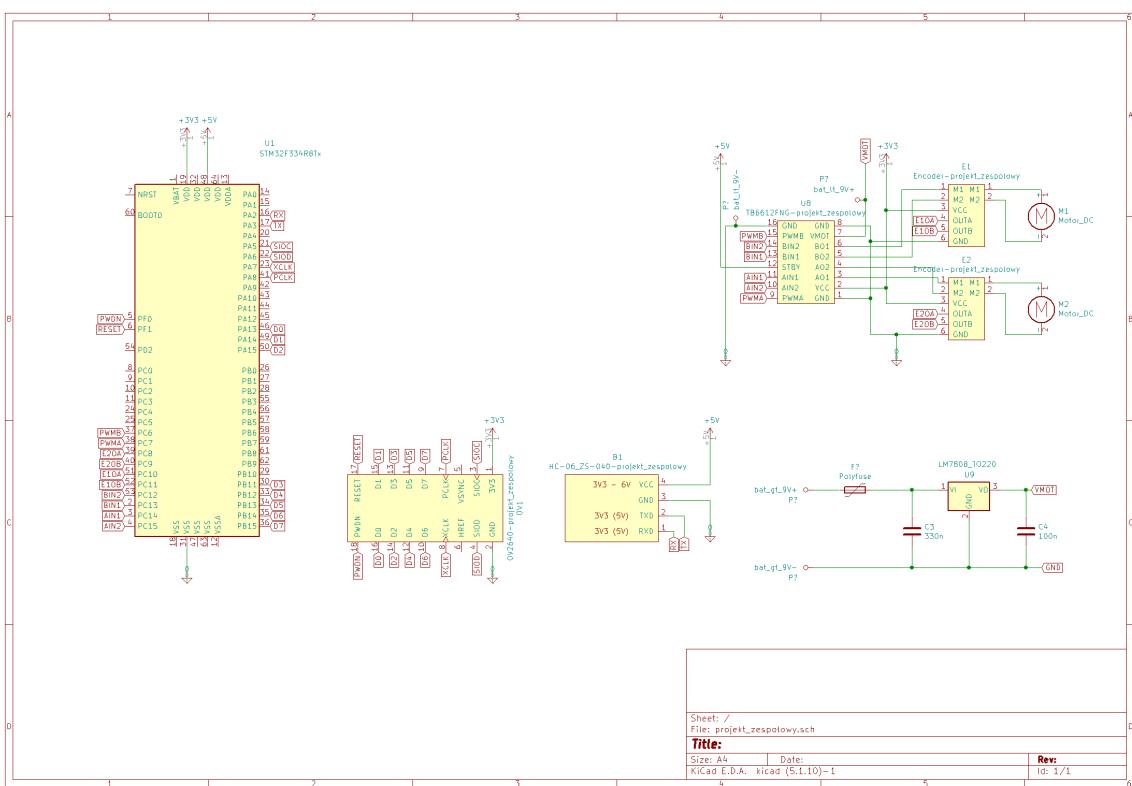
Zgodnie z założeniami na ostatni etap prac utworzono funkcję pozwalającą na sterowanie dwoma silnikami w robocie (rys.3). W celu odbierania sygnałów wysyłanych z aplikacji skorzystano z kodu, którym zespół programistów sprawdzał połączenie bluetooth. Dodatkowo w trakcie etapu wykonano serwis płytka ze względu na nieznaczne uszkodzenia, które nastąpiły podczas transportu co niestety spowodowało opóźnienia w realizacji projektu. Podczas serwisu przyjęto również płytka tak aby mieściła się ona wewnątrz obudowy (rys.4).

3 Specyfikacja finalnego produktu

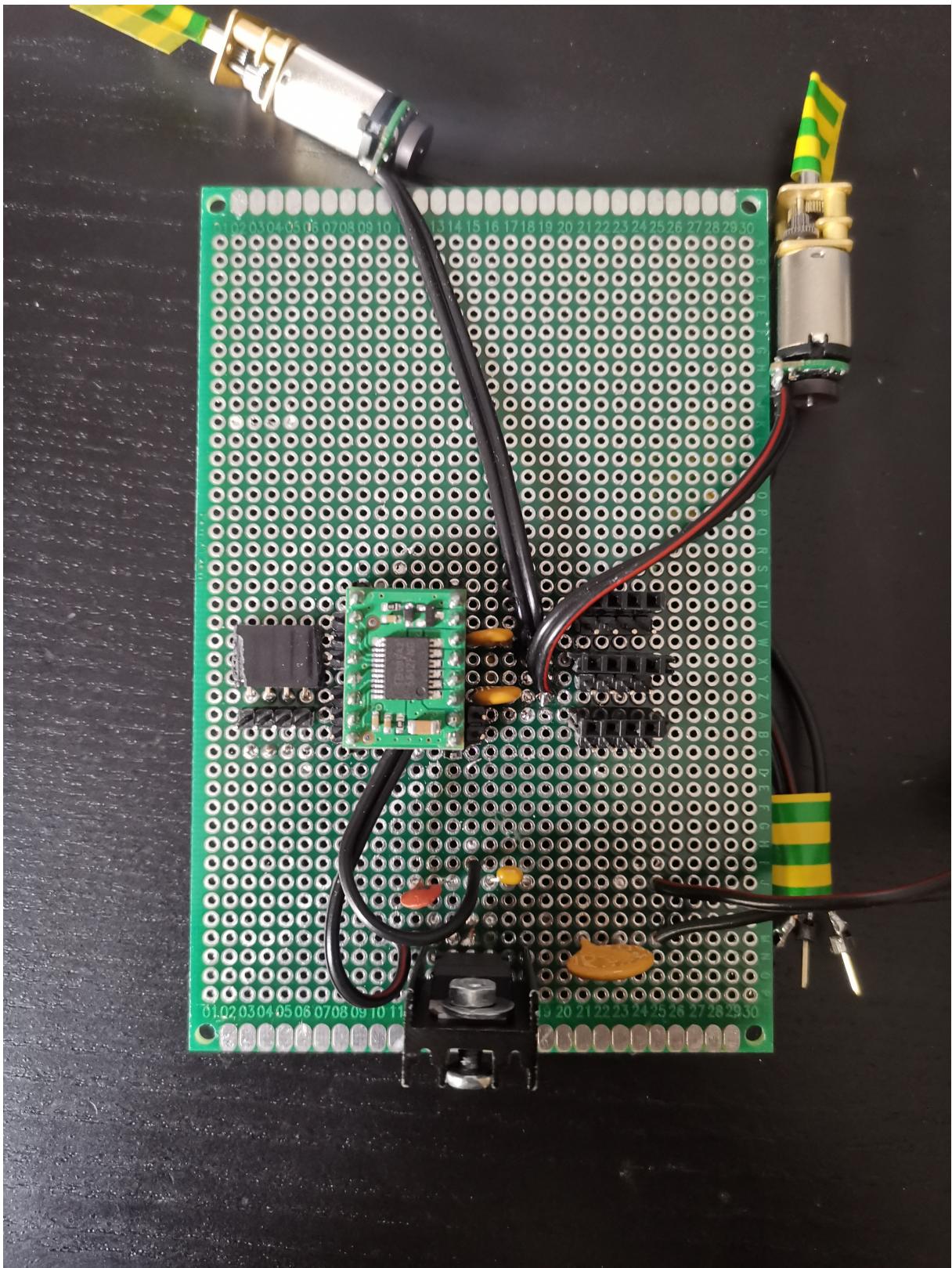
Końcowym efektem prac zespołu elektroników jest płytka (rys.2) pozwalająca na sterowanie silnikami robota za pomocą dwóch sygnałów pwm sterujących prędkością obrotów silników oraz czterech sygnałów logicznych sterujących kierunkiem obrotów silników. Do sterowania silnikami wykorzystywano sterownik silników TB6612FNG firmy pololu. Płytnka posiada wejścia, na które możliwe jest nałożenie płytki stm. Na płytce występują dodatkowo 3 zestawy pinów (5V, 3.3V i GND) pozwalające na podłączanie do płytka dodatkowych peryferiów co pozwala na ewentualny rozwój konstrukcji w przyszłości. Płytnka posiada dwa wejścia zasilania silników, jedno przeznaczone do podłączania baterii typu LiPol o napięciu większym niż 9V oraz drugie dla baterii poniżej 9V. Schemat połączeń elektronicznych w zbudowanym robocie przedstawiony jest na schemacie poniżej (rys.1).

4 Cele niezrealizowane

Problemem, którego do tej pory nie udało się rozwiązać jest to, że jeżeli źródło zasilania zostanie podpięte do układu obniżającego napięcie z niewiadomych przyczyn pomimo dostarczenia prądu do sterownika silników nie powoduje to ruchu silników, a na samych silnikach napięcie wynosi 0V. Problem ten nie ma jednak wpływu na końcowy efekt ponieważ silniki można również zasilać z napięć poniżej 9V bez wykorzystania układu obniżającego napięcie.



Rysunek 1: Schemat połączeń elektronicznych w robocie



Rysunek 2: Zbudowana płytka

```

if(recieve_flag )
{
    recieve_flag=0;

    __HAL_DMA_DISABLE_IT(&hdma_usart2_rx, DMA_IT_HT);

    memcpy(MainBuff, RxBuff, size_recieved);
    // printf("Recieved: %s\r\n",MainBuff);
    sscanf(MainBuff, "%c %d %d", &state, &pwm_1, &pwm_2);
    printf("Recieved: %c %d %d\r\n", state, pwm_1, pwm_2);

    //HAL_UART_Receive_DMA(&huart2, data, 100);
    //HAL_UART_Transmit(&huart2, data, strlen(data), 100);

    if (pwm_1 > 0) //do przodu
    {
        HAL_GPIO_WritePin(MOTOR_1_GPIO_1_GPIO_Port, MOTOR_1_GPIO_1_Pin,
                           1);
        HAL_GPIO_WritePin(MOTOR_1_GPIO_2_GPIO_Port, MOTOR_1_GPIO_2_Pin,
                           0);
        pwm_1 = 1000.0 * pwm_1 / 100.0;
        __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_1, pwm_1);

    } else // do tyłu
    {
        HAL_GPIO_WritePin(MOTOR_1_GPIO_1_GPIO_Port, MOTOR_1_GPIO_1_Pin,
                           0);
        HAL_GPIO_WritePin(MOTOR_1_GPIO_2_GPIO_Port, MOTOR_1_GPIO_2_Pin,
                           1);
        pwm_1 = -1000.0 * pwm_1 / 100.0;
        __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_1, pwm_1);

    }

    if (pwm_2 > 0) //do przodu
    {
        HAL_GPIO_WritePin(MOTOR_2_GPIO_1_GPIO_Port, MOTOR_2_GPIO_1_Pin,
                           1);
        HAL_GPIO_WritePin(MOTOR_2_GPIO_2_GPIO_Port, MOTOR_2_GPIO_2_Pin,
                           0);
        pwm_2 = 1000.0 * pwm_2 / 100.0;
        __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_2, pwm_2);

    } else // do tyłu
    {
        HAL_GPIO_WritePin(MOTOR_2_GPIO_1_GPIO_Port, MOTOR_2_GPIO_1_Pin,
                           0);
        HAL_GPIO_WritePin(MOTOR_2_GPIO_2_GPIO_Port, MOTOR_2_GPIO_2_Pin,
                           1);
        pwm_2 = -1000.0 * pwm_2 / 100.0;

        __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_2,
                               pwm_2);

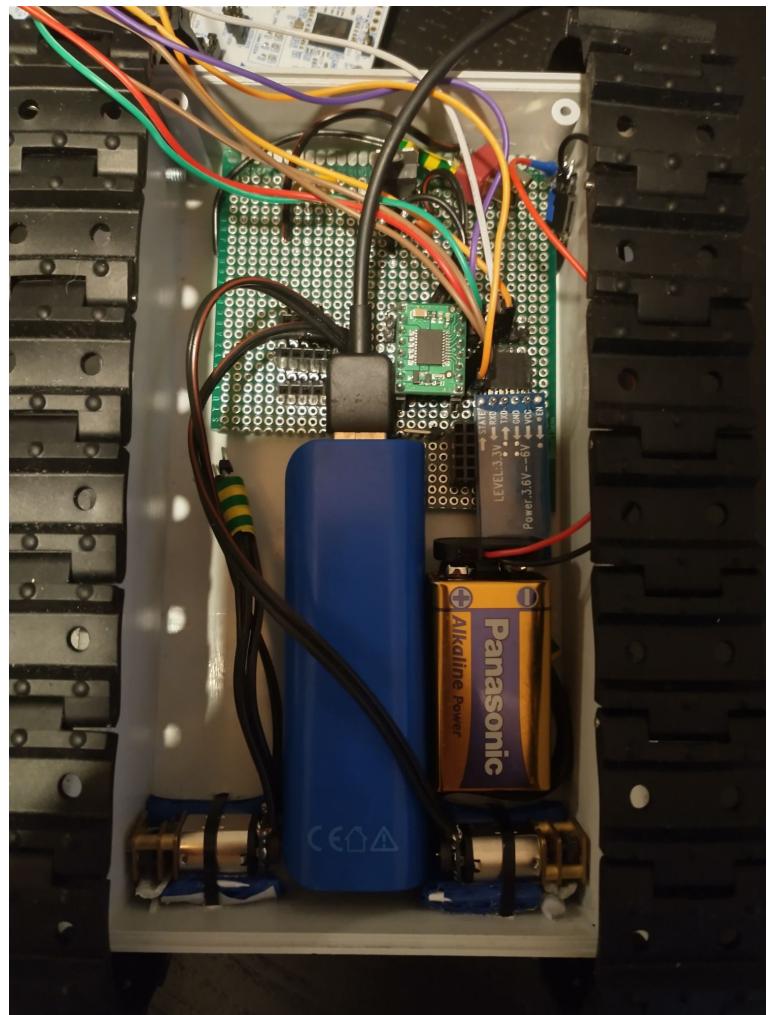
    }

    if (state == 'r') //jeżeli puszczone
    {
        memset(MainBuff, 0, sizeof(MainBuff));
        memset(RxBuff, 0, sizeof(RxBuff));
        //soft stop
        HAL_GPIO_WritePin(MOTOR_1_GPIO_1_GPIO_Port, MOTOR_1_GPIO_1_Pin,
                           0);
        HAL_GPIO_WritePin(MOTOR_1_GPIO_2_GPIO_Port, MOTOR_1_GPIO_2_Pin,
                           0);
        HAL_GPIO_WritePin(MOTOR_2_GPIO_1_GPIO_Port, MOTOR_2_GPIO_1_Pin,
                           0);
        HAL_GPIO_WritePin(MOTOR_2_GPIO_2_GPIO_Port, MOTOR_2_GPIO_2_Pin,
                           0);
    }

    HAL_UARTEx_ReceiveToIdle_DMA(&huart2, RxBuff,RxBuff_SIZE);
}

```

Rysunek 3: Fragment kodu sterujący silnikami na podstawie danych otrzymywanych z aplikacji



Rysunek 4: Połączenia wewnętrz robota