Systemy Wbudowane laboratoria

Adam Friedensberg, Piotr Zaraś

17 czerwca 2019

1 Opis

1.1 Cel

Celem projektu jest stworzenie systemu umożliwiającego wymiarowanie pomieszczeń przy założeniach:

1. Wszystkie ściany w pomieszczeniu przecinają się pod kątem prostym
2. W pomieszczeniu nie ma nieregularności w postaci wnęk (np. na grzejniki)
3. Dostęp do ścian na pozioimie podłogi nie jest utrudniony

1.2 Komponenty

Głównym elementem systemu jest wcześniej złożone autko, którego komponentami są:

1. Od 2 do 4 niezależnych silników
2. Źródła zasilania w postaci 2 akumulatorów
3. Dwóch czujników HC-SR04 umieszczonych na froncie i u lewego boku autka - służących do ultradźwiękowego pomiaru odległości
4. Mikrokontrolera arduino Uno
5. Modułu służącego do zapisu danych na kartę mikroSD do arduino
6. Czujniki szczelinowe zamontowane na kołach autka służące do pomiaru odległości

1.3 Działanie

Autko stawiamy tak by jego lewa strona znajdowała się w odległości 10-20 cm od ściany. Po następnym uruchomieniu autko stara się utrzymać przy ścianie i jechać prosto aż napotka przeszkodę lub jego lewy czujnik odległości wykryje jej znaczny wzrost. Autko zakłada w tedy, że należy zakręcić odpowiednio w prawą lub lewą stronę. Na każdym zakręcia autko robi dwie rzeczy, po pierwsze zapisuje przejechaną odległość i kierunek skrętu na kartę microSD oraz sprawdza czy nie znajduje się w miejscu startowym. Gdy autko stwierdzi, że znalazło się w punkcie startowym przerywa ono wykonywanie programu.

2 Implementacja

2.1 Predefinicje i setup

W programie używamy kilku predefiniowanych wartości (nie będących pinami):

1. HORIZON - horyzont wydarzeń nic co jest w odległości większej niż ta wartość nie jest uznawane za widoczne. Stała wprowadzona ze względu na niedokładności pomiaru dla dużych odległości.
2. MIN_DIST_L / MAX_DIST_L - minimalna/maksymalna odległość autka od lwej ściany dopuszczalna przed podjęciem próby wyrównania.
3. MIN_DIST_F / MAX_DIST_F - minimalna/maksymalna dopuszczalna odległość autka od przeszkody z przodu (z założenia ściany) przed którą autko ma wykonać zakręt w prawo.
4. MIN_TURN_DIST - minimalny dystans z lewej strony autka przy, którym ma zostać wykonany zakręt w lewo.
5. CIRC - obówd koła podany w mm.

Na etapie setupu podłączamy odpowiednie piny z arduino do biblioteki Wheels zapewnionej przez wykładowcę. Ustawiamy prędkości poszczególnych osi. Zerujemy również wszystkie zmienne globalne odpowiedzialne za mierzenie odległości (funkcje *restartGaps*() i *restartRot*()). I usuwamy poprrzednio istniejący plik z karty SD.

2.2 Main loop

Główna pętla służy tutaj tylko do wywoływania funkcji *checkPath*() w interwałach co 150 ms.

2.3 Kontrola jazdy (funkcja *checkPath*())

Funkcja *checkPath*() odpowiada za kontrolowanie jazdy wzdłuż ściany oraz zakręcanie tam gdzie jest to potrzebne. Wszystko bazuje na predefiniowanych odległościach, które są porównywane z aktualnymi pomiarami z czujników.

Algorithm 1: checkPath()

```
fDist ← getDistF();
lDist ← getDistL();
if fDist < MIN_DIST_F then
    Ustaw prędkości kół na domyślną prędkość;
    Zapisz dane do pliku;
    Wykonaj skręt w prawo;
else
    Jedź prosto;
end

if lDist < MIN_DIST_L then
    Ustaw prędkość lewego koła na domyślną prędkość;
    Zmniejsz prędkość prawej strony autka w stosunku do domyślnej;        // odbij od ściany
else if lDist > MIN_TURN_DIST then
    Ustaw prędkości kół na domyślną prędkość;
    Zapisz dane do pliku;
    Wykonaj skręt w lewo;
else if lDist > MAX_DIST_L then
    Ustaw prędkość prawego koła na domyślną prędkość;
    Zmniejsz prędkość lewej strony autka w stosunku do domyślnej;        // wyrównaj do ściany
else
    Ustaw prędkości kół na domyślną prędkość;
end
```

2.4 Pomiar odległości (funkcje *getDistL*() oraz *getDistF*())

Obie funkcje pomocnicze wykorzystywane są wewnątrz *checkPath*(), a ich zadaniem jest zwrócenie wartości odległości (w centymetrach) od czujników. Poniżej algorytm działania *getDistF*(), który jest prawie identyczny jak *getDistL*(), a różnica polega jedynie na wykorzystaniu innych definicji dla pinów pod które podłączone są czujniki.

Algorithm 2: getDistF()

```
tot;                                          // time of travel, czyli czas powrotu sygnału
distance;                                     // wartość zwracana, oznacza odległość ściany od czujnika

pin TRIG_F ← HIGH;                                                    // digitalWrite()
delay(10);
pin TRIG_F ← LOW;                                                     // digitalWrite()

tot ← pulseIn(ECHO_F, HIGH);
distance ← tot/58;

return distance;
```

2.5 Zapisywanie do pliku (funkcja *writeToAFile*(*charc*))

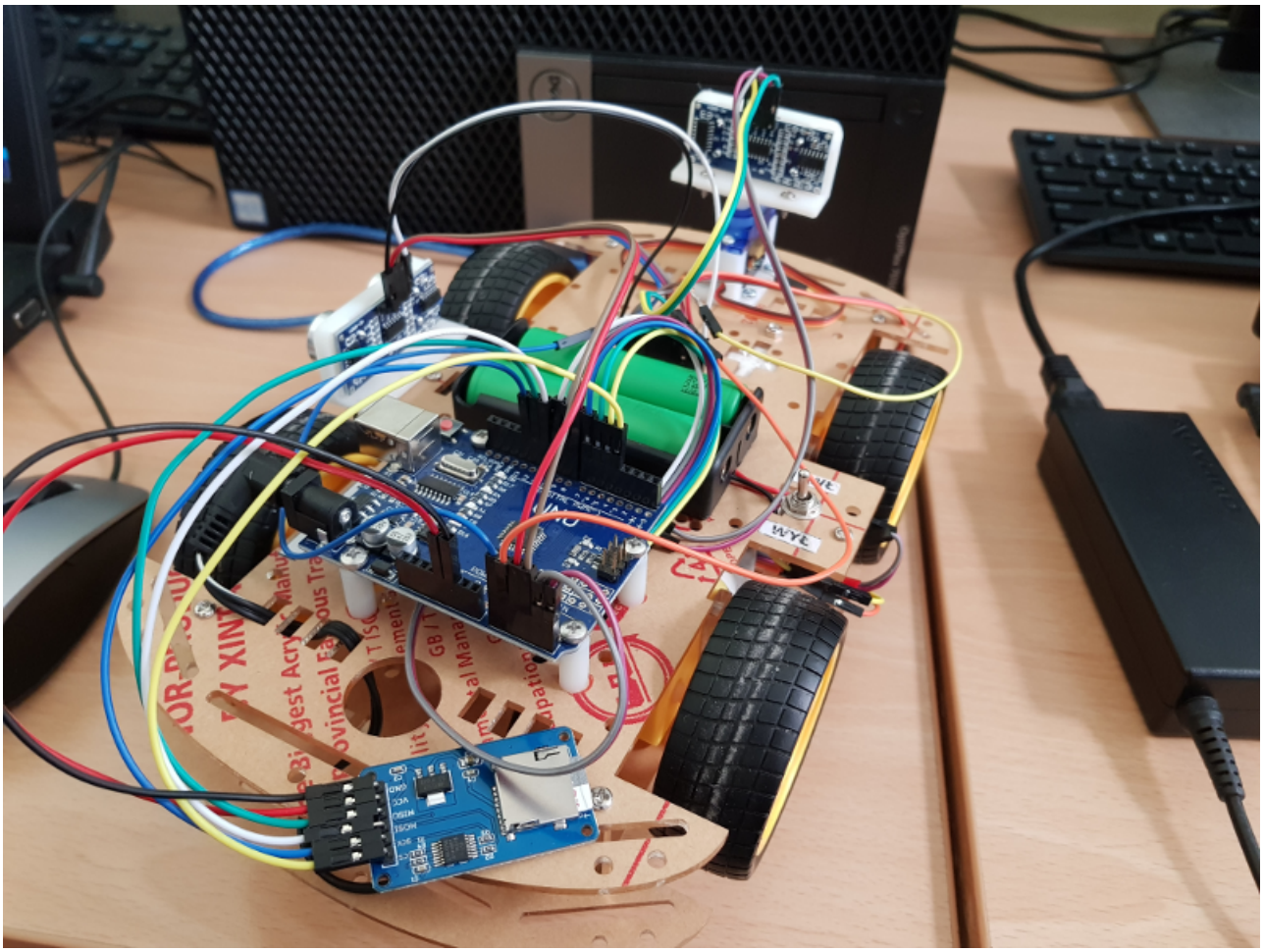
Przy zapisywaniu do pliku korzystamy z biblioteki *SD.h* oraz adaptera kart SD.

Algorithm 3: writeToAFile(char c)

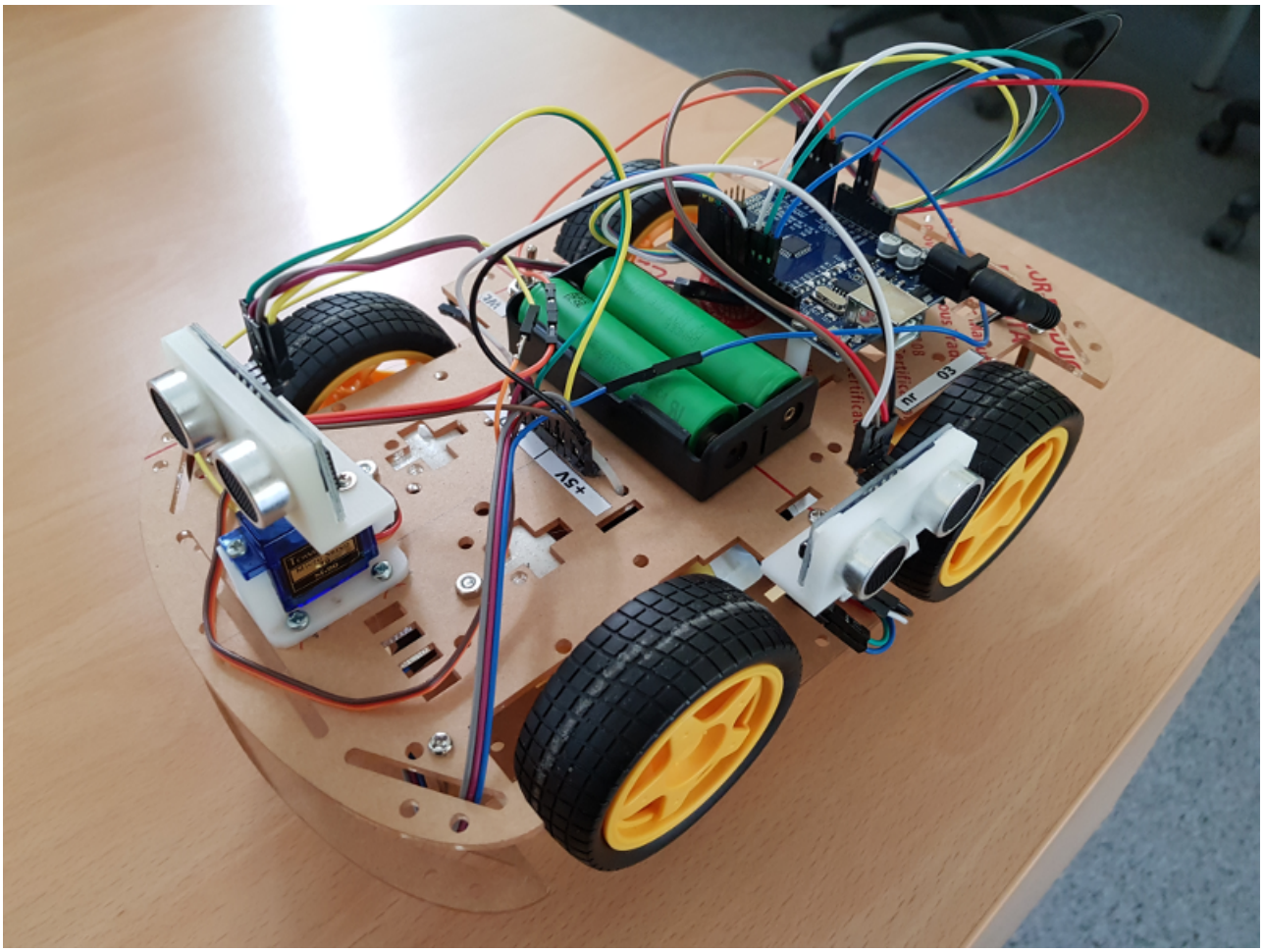
Data: c - kierunek skrętu R lub L

```
1 plik.open();
2 if plik jest dostępny then
3 |   len ← (((lGap + rGap)/2)/40)*CIRC;      // gdzie lGap i rGap to ilość naliczonych szczelin
4 |   plik ← len++c;
5 plik.close();
6 restartGaps();                                                    // wyzeruj naliczanie szczelin
```

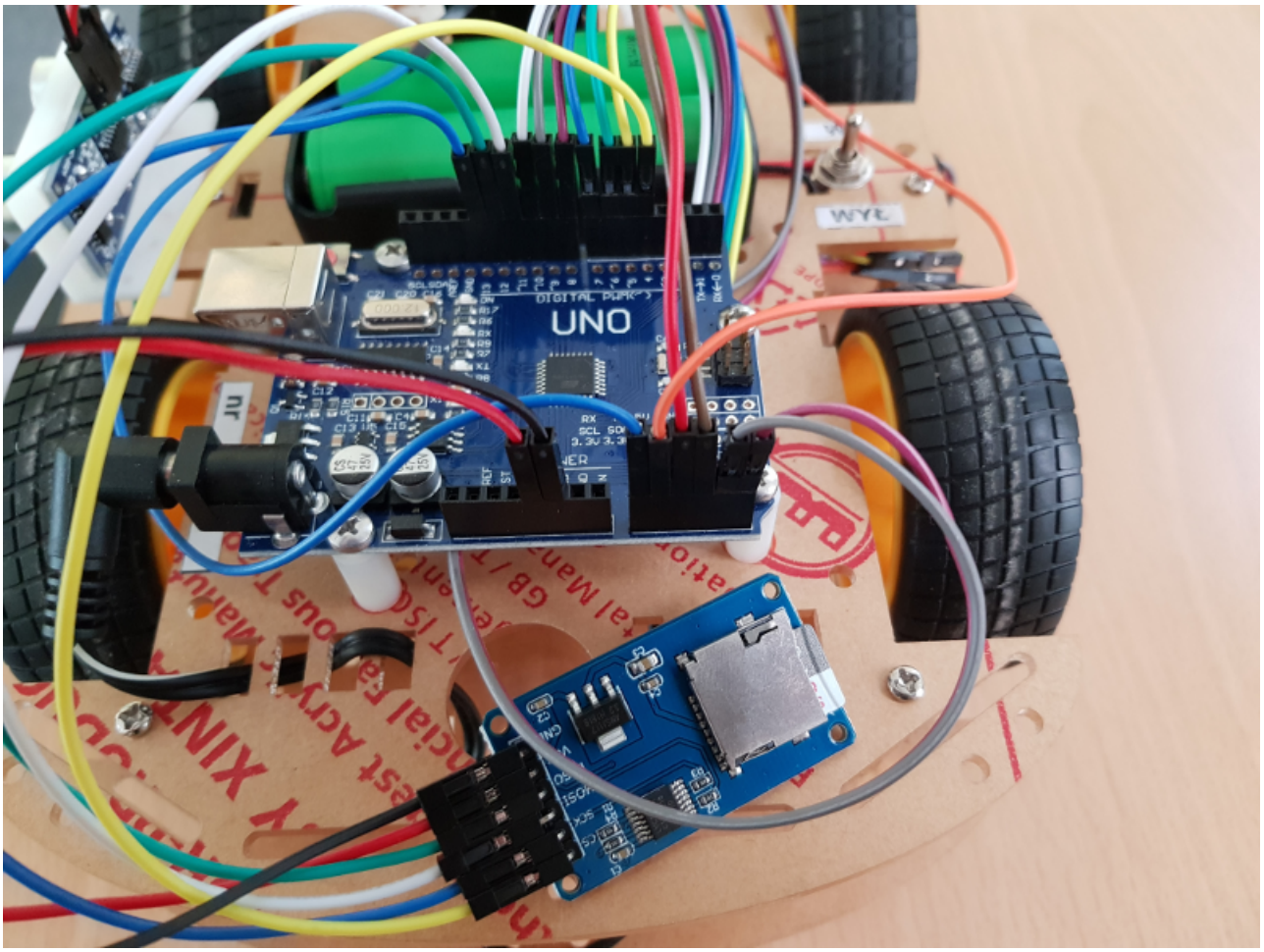
3 Zdjęcia projektu



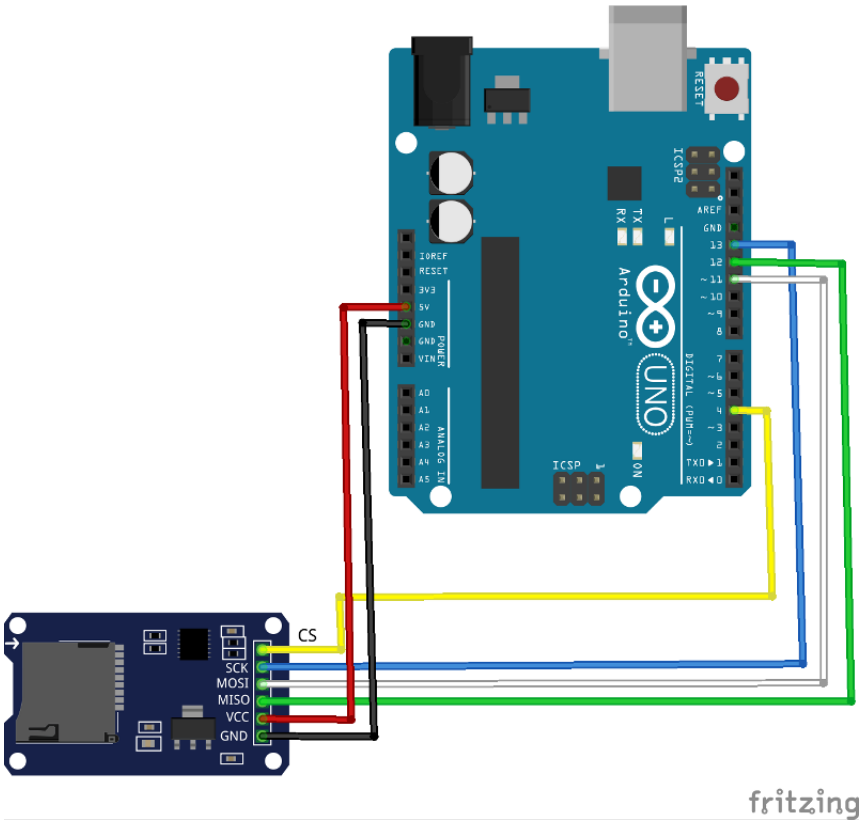
Rysunek 1: Podłączonie - zdjęcie z tyłu autka



Rysunek 2: Podłączonie - zdjęcie z przodu autka



Rysunek 3: Podłączonie kabelków - zdjęcie płytki Arduino



Rysunek 4: Podłączonie - schemat podłączenia MicroSD Card Adapter