Biblioteka Compass

USER MANUAL

1. Użycie

DODAWANIE PLIKOW

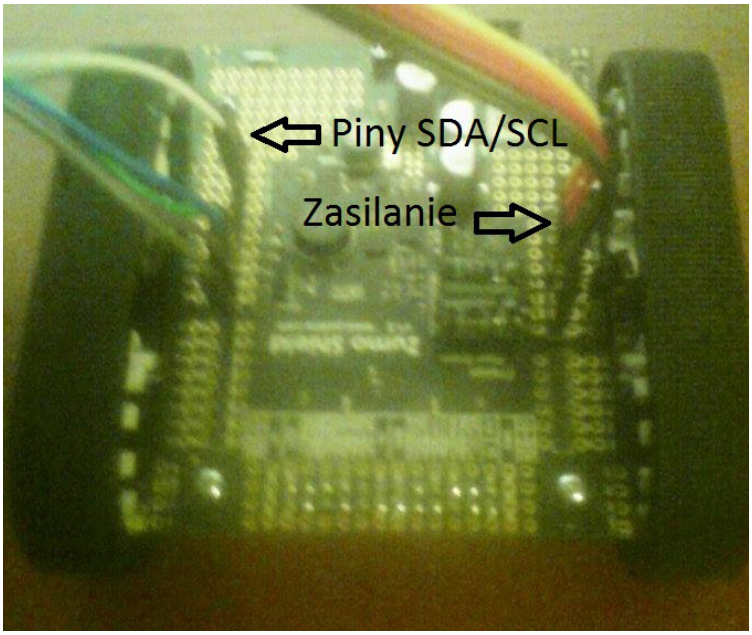
Pobierz i dodaj do swojego projektu pliki:

compass.c ,compass.h– funkcje wysokiego poziomu obsługi kompasu
mag.c, mag.h – funkcje niskiego poziomu obsługi kompasu z płytki FRDM
mag2.c, mag2.h – funkcje niskiego poziomu obsługi kompasu z ZUMO
i2c.c, i2c.h – funkcje obsługi komunikacji i2c0
i2c1.c, i2c1.h – funkcje obsługi komunikacji i2c1

W pliku main.c dodaj ***#include "compass.h"***

PODŁĄCZENIE

Dla poprawnego działania biblioteki konieczne jest poprawne podłączenie robota do płytki. Należy podłączyć piny PORTE1 do SCL oraz PORTE0 do SDA na robocie. Konieczne jest także podłączenie do zasilania dla kompasu na ZUMO. W tym celu podłączamy piny GND oraz 3.3V



Ilustracja 1: Sposób podłączenia ZUMO do płytki

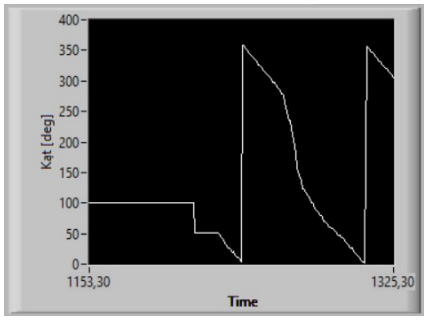
INICJALIZACJA

Po poprawnym podłączeniu możliwa jest udana inicjalizacja kompasów. Za pomocą funkcji ***eCompass_init()*** inicjalizujemy od razu dwa kompasy.

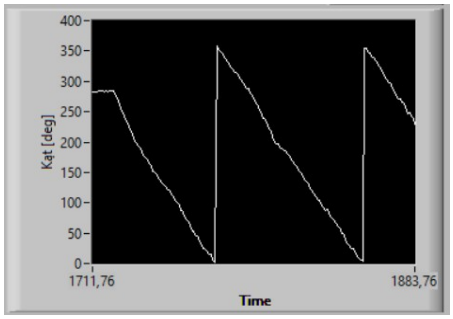
KALIBRACJA

Następnym krokiem jest kalibracja kompasów. Na ilustracjach 2 oraz 3 można zauważyć że charakterystyka wartości obrotu odczytywanej przez kompas od rzeczywistej przed kalibracją jest daleka od liniowej przez co dane z takiego kompasu nie odpowiadają rzeczywistości. Po kalibracji dane są bardzo zbliżone do liniowych. Kalibracje może być przeprowadzona raz, ponieważ dane offsetu są zapisaywane w kompasach. Nie jest zapisywany natomiast współczynnik AR – stosunek maxymalnych wartości na osiach X i Y. Jak widać na ilustracjach 4 i 5 ma on znaczenie tylko kiedy

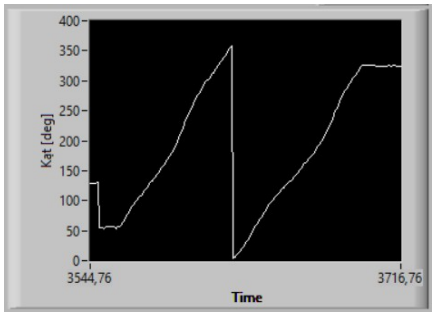
wartości na tych osiach się różnią – np. płytka jest krzywo ustawiona.



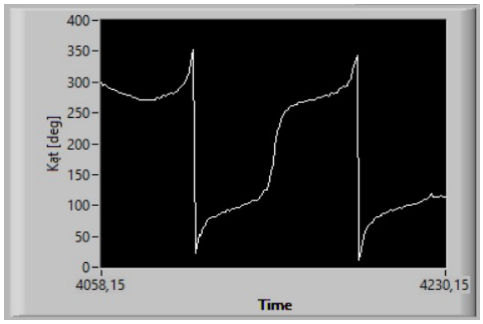
Ilustracja 3: Kompas na robocie. Przed kalibracją.



Ilustracja 2: Kompas na robocie. Po kalibracji



Ilustracja 4: Kompas na płytce krzywo ustawionej. Uwzględziony współczynnik AR.



Ilustracja 5: Kompas na płytce krzywo ustawionej. Brak współczynnika AR.

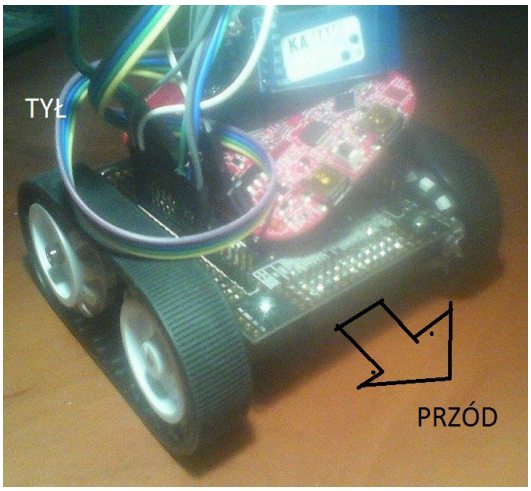
Kalibracje zaczynamy poprzez wywołanie funkcji ***`startCalibration()`***. Następnie należy obrócić robota i płytkę o co najmniej 360 stopni w pozycji poziomej. Im wolniejsze obracanie tym dokładniejsza kalibracja. Kalibracje kończy się poprzez wywołanie funkcji ***`stopCalibration()`***.

WYBÓR KOMPASU

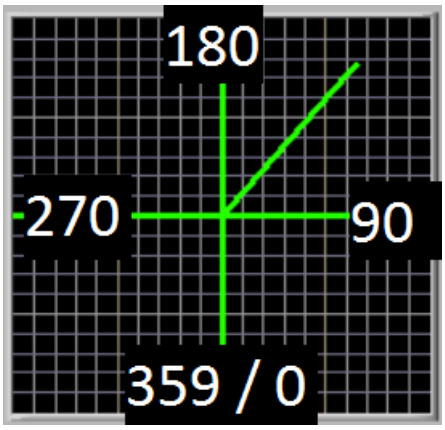
Po udanej kalibracji możemy zdecydować z którego kompasu chcemy odczytywać dane. Zmianę kompasu dokonujemy wywołując funkcję ***`selectCompass(int num)`***. Numer należy wybrać z przedziały 0 do 3. 0 oznacza korzystanie z dwóch kompasów. 1 to kompas na płytce. 2 to kompas na robocie. Jest on najdokładniejszy dlatego jest on używany domyślnie. Opcja 3 wskazuje kąt który jest średnią arytmetyczną kątów otrzymanych z dwóch kompasów.

ODCZYTYWANIE KĄTA

Kąt odczytujemy poprzez funkcję ***`eCompass()`***. Zwraca ona kąt w stopniach od 0 do 359. Przód robota został zaznaczony na ilustracji 6. Na ilustracji 7 można odczytać jaka wartość odpowiada jakiemu obróceniu robota. Dla północy jest to 180stopni. Zwiększa się wraz z obrotem przeciwnym do wskazówek zegara. Kompas na płytce wskazuje takie same wartości o ile jest położony tak aby przód (strona na której znajduje się wyświetlacz była skierowana w dół). Jeżeli płytka jest obrócona wyświetlaczem do góry to wartości wskazywane są inne. Zwiększają się wraz z obrotem zgodnym ze wskazówkami zegata. Odczytywanie wspólnych wartości kompasów (opcja 0 lub 3) jest możliwa tylko wtedy kiedy płytka FRDM jest położona wyświetlaczem do dołu i przodem (od strony wyswietlacza) zgodnym z przodem robota ZUMO.



Ilustracja 6: Robot ZUMO



Ilustracja 7: Wartości zwracana przez eCompass()

JAZDA PROSTA

W pliku motorDriver.c znajdują się funkcje do skrętów robota o określony kąt lub do określonego kierunku (turnLefrC(int kat), turnLeftDir(int dir)). Znajduje się tam także funkcja jedzProsto(), której celem jest prowadzenie robota po prostej linii i przeciwdziałał nierównej jeździe prawej i lewej gąsienicy. Należy ją wywołać w momencie od którego chcemy by robot zapamiętał obecny kierunek i jechał cały czas w tym kierunku. W praktyce robot trzymając jeden kierunek jeździ bardzo krzywą drogą, ponieważ pole magnetyczne w różnych miejscach jest różne. Doświadczalnie różnice osiągały nawet około 70 stopni. Ten błąd nie ma związku z kompasem na robocie, a jest wynikiem innego pola magnetycznego w różnych miejscach. Dlatego funkcję tę można traktować jako ciekawostkę, a nie jako narzędzie wyprostowujące tor robota.

2. Funkcje

eCompass_init() **–**inicjalizuje działanie kompasu, bez wywołanie tej funkcji, żadne inne nie będą działać. Inicjalizuje obydwa kompasy i transmisje na I2C0 i I2C1

startCalibration() **–** wywołaj, aby rozpocząć kalibracje.

stopCalibration()**–** wywołaj, aby zakończyć kalibracje.

eCompass() **–** zwraca wartość obrotu kompasu.

setAvaraging(uint8_t num) **–** Określa ile wykonać pomiarów, aby zwrócić jeden wynik (domyslnie ustawione na 1). Zmiana niezalecana.

selectCompass(uint8_t num) **–** Wybór kompasu z którego korzystamy. 0-obydwa kompasy, 1-kompas na płytce FRDM, 2-kompas na robocie, 3-średnia kątów z dwóch kompasów

3. Użyte zasoby w bibliotece kompas

Sprawdź, czy nie kolidują z innymi zasobami użytymi przez ciebie!

Komunikacja **I2C0**

Adres kompasu : 0001110b

Port przyłączenia kompasu: PORTE 24, PORTE25

Komunikacja **I2C1**

Adres kompasu : 0011101b
Port przyłączenia kompasu: PORTE0, PORTE1

Timer **PIT chanel 0**

Dodatkowo w plikach motorDriver.c są używane

Timer **TPM 0** → sterowanie obrotami silniczków
Timer **TPM 1 chanel 0**→ wolniejszy timer, do skręcania
PORTY: PORTA13 , PORTC9, PORTD2, PORTD4

4. Dodatkowe użycie akcelerometru

Dodaj pliki
acc.c, acc.h
W pliku ***compass.c*** odkomentuj ***accelInit()*** (23 wiersz)
W pliku ***compass.h*** odkomentuj ***#include „acc.h"*** (2 wiersz)

Akcelerometr łączy się przez to samo I2C, co kompas. Jego adres to 0011101b.