ROBOT MAȘINĂ PE 2 ROȚI

2 MOTOARE, MODUL BLUETOOTH ȘI SENZOR ULTRASUNETE

DRAGOȘ-ȘTEFAN PERJU



REZUMAT

Acest referat va discuta realizarea ca proiect a unui robot mașină de dimensiuni mici

pe 2 roți, având câte un motor pentru fiecare roată, controlate indirect folosind un microcontroller Freescale/NXP FRDM KL25Z. Robotul mașină este controlat la distanță prin intermediul unui modul Bluetooth și se oprește când întâlnește un obstacol frontal, folosind un senzor de ultrasunete. Este alimentat la 5V, în această situație printr-o baterie externă USB. În referat se va discuta conexiunea dintre aceste părți componente și detaliile tehnice ale fiecăruia.

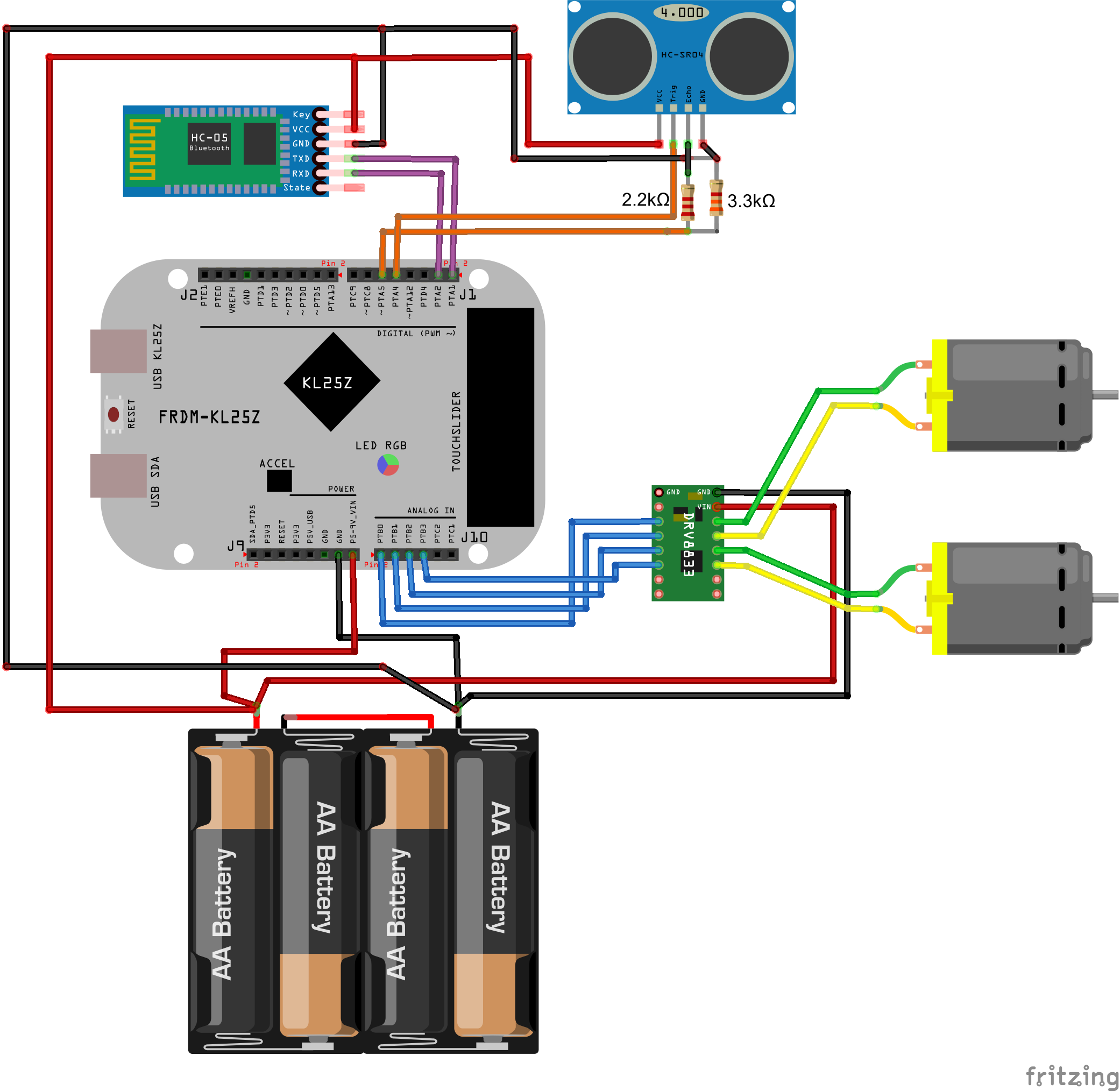
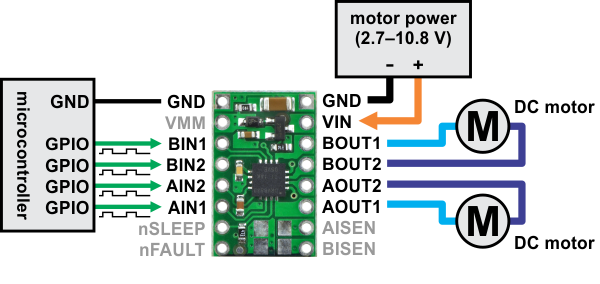
DESCRIEREA COMPONENTELOR FIZICE

În realizarea acestui proiect, s-au folosit următoarele componente:

* Microcontroller Freescale/NXP FRDM KL25Z, pe platforma mbed
* Driver de motoare dual Pololu DRV8833
* Modul Bluetooth master-slave HC-05
* Senzor de ultrasunete HC-SR04
* 2 motoare DC 3-6V
* Baterie USB externă 4800mAh 5V

Schema de conectare între aceste componente se poate studia pe pagina următoare. Se vor menționa câteva caracteristici tehnice pentru fiecare componentă în parte și câteva particularități la conectarea lor.

În ceea ce privește microcontrollerul FRDM KL25Z, se poate alimenta cu o tensiune de la 5 la 9V, iar cât despre pinii de intrare-ieșire, primește și difuzează o tensiune de 3.3V. Este important acest lucru, deoarece trebuie să ținem cont și de capacitățile pinilor de intrare-ieșire și la celelalte componente auxiliare. Dacă driverul de motor și modulul Bluetooth nu au probleme în acest sens, senzorul de ultrasunete HC-SR04 difuează un semnal numit ECHO care este de 5V. Acest lucru ar afecta microcontrollerul, dacă nu am realiza un divizor de tensiune folosind rezistențe de 2.2k și 3.3k, cum este în schemă, care să transpună semnalul ECHO la 3.3V la intrarea microcontrollerului.



Modului Bluetooth HC-05 comunică prin comunicație serială cu FRDM KL25Z. Prin comenzi AT, se poate configura numele acestuia, PINul utilizat precum și baud rate-ul folosit la UART. Motoarele DC se alimentează la o tensiune între 3-6V. La 5V, oferă 200 de rotații pe minut și consumă 100mA curent. Reducția făcută pe motor de către cutia de viteze este de 1:48. Combinat cu driverul de motor dual, reprezintă un ansamblu bun de a controla un robot mașină de dimensiuni mici.

Driverul de motoare dual Polulu DRV8833 poate fi alimentat în intervalul de tensiune 2.7-10.8V. Curentul de ieșire pentru motoare ajunge astfel să fie 1.2A fiecare. Având două H-bridge-uri, protecție împotriva surplusului de curent și a supraîncălziri, reprezintă o alegere potrivită pentru a alimenta cele 2 motoare DC din cadrul proiectului, folosind semnale PWM setate la o frecvență de 50 Hz (20 ms). Cu ajutorul lui, se poate controla și un singur motor pas-cu-pas bipolar.

Întregul circuit al proiectului folosește doar 80mA când este pe stand-by și așteaptă un dispozitiv să se conecteze prin Bluetooth. Doar 60mA după ce s-a conectat unul, 340 mA la folosirea ambelor motoare la viteză minimă și 440mA la viteză maximă. Folosirea doar a unui singur motor la viteza maximă reprezintă un consum de curent de 280mA. Știind că bateria externă USB utilizată pentru acest proiect are o autonomie de 4800mAh, o împărțire simplă ne spune că proiectul propus poate rezista între 10 și 14 ore pe baterie, cu folosirea motoarelor.

DESCRIEREA PROGRAMULUI REALIZAT

Programul a fost realizat pe scheletului RTOS pentru platforma mbed. Are 4 fire de execuție:

1. Preluarea comenzilor Bluetooth
2. Controlul motoarelor
3. Raportarea distanței percepute de senzorul de ultrasunete utilizatorului
4. Procesarea datelor pentru auto-oprirea robotului în caz de obstacol

Bineînțeles, pentru folosirea unei astfel de arhitecturi cu mai multe fire de execuție, în cadrul programului s-a folosit un ansamblu de obiecte pentru excluziune mutuală și partajare de memorie. Acest lucru asigură flexibilitate pentru viitor, dacă se dorește adăugarea de facilități noi la programarea robotului, cum ar fi autopilot.

Demonstrăm acest tip de programare prin studierea unei secvențe de cod:

void SensorWrap::run(void const\* args) {

while (1) {

bool ab;

ABMutex.lock();

ab = autobreak;

ABMutex.unlock();

RepMutex.lock();

if (autobreak || report) {

RepMutex.unlock();

sensor.updateDistance();

}

else {

RepMutex.unlock();

}

if (ab) {

sensor.autobreakBrain();

}

}

}

Secvența de cod reprezintă funcția principală pentru firul de execuție cu privire la auto-oprirea în caz de obstacol. Se cere o actualizare a distanței percepute de senzorul de ultrasunete, după care se apelează „creierul” ce decide oprirea robotului mașină dacă este nevoie.

Observăm comenzile de blocare și deblocare pentru excluziune mutuală. Folosim ABMutex când setăm sau citim flagul boolean global autobreak, la fel fiind și pentru RepMutex și flagul report. Aceste două flaguri globale țin setările utilizatorului pentru a folosi facilitatea de auto-oprire și de raportare a distanței. În cazul în care nu se folosește niciuna dintre facilități, nu se cere actualizarea distanței percepute de senzorul de ultrasunete fără folos. În acest fir, dacă este activat, se procesează datele pentru a decide oprirea robotului în caz de un obstacol.

Existând situația în care se poate seta de către firul de comenzi Bluetooth aceste flaguri menționate, în același timp cu citirea acestor flaguri de către firul de execuție pentru procesarea informațiilor de pe senzor, s-au folosit obiecte de exclusiune mutuală. Observăm că această funcție dispusă în secvența de cod devine, deci, complicată – dorindu-ne în acest program blocarea resurselor critice doar pentru un scurt timp, astfel nepunând comenzile de blocare și deblocare doar la începutul și sfârșitul funcției, ci presărat după cum este subliniat.

Discutând despre posibilitatea extinderii facilităților, cum ar fi cel de autopilot, folosirea scheletului RTOS și a paradigmii de programare cu mai multe fire de execuție este una bună, deoarece rămâne foarte ușor să implementăm ideile bazate pe, spre exemplu, folosirea senzorului de ultrasunete, deoarece informațiile dedicate acestuia sunt deja preluate și procesate de către un singur fir de execuție. Astfel, nu ne vom afla în situația în care trebuie să cerem de mai multe ori același lucru, cu fiecare facilitate adăugată, pentru a fi siguri de prospețimea informațiilor cât timp se execută singurul fir de execuție. Aici, putem mări doar prioritatea firului de execuție dacă dorim o îmbunătățire în acest sens sau să facem un fir nou de execuție pentru decizia auto-opririi – lăsând un fir pentru actualizarea senzorului.

CONCLUZIE

Un astfel de proiect a ajutat la învățarea a multor elemente ce a fost nevoie pentru crearea lui. De la folosirea letconului, la folosirea specificațiilor fiecărei componente folosite cu atenție și programarea pe paradigma firelor multiple de execuție, a ajutat la explorarea, ansamblarea și programarea unui alt tip de computer decât cel cunoscut și utilizat de utilizatori obișnuiți. Produsul final, astfel, dovedește a fi un exemplu robust de proiect educativ și interesant de implementat.

Dragoș-Ștefan Perju

Facultatea de Automatică și Calculatoare

Universitatea Tehnică „Gheorghe Asachi” Iași

anul 3, grupa 1306A, 2016