Procesarea semnalelor primite de la senzori pe microcontroler

Student: Simina Dan-Marius

Proiect Structura Sistemelor de Calcul

Universitatea Tehnică din Cluj-Napoca

Cuprins

1. Introducere

1.1 Context

Scopul acestui proiect este eficientizarea implementărilor de procesare a datelor de la senzori pe microcontroler. Microcontrolerele sunt folosite în produse și dispozitive controlate automat, cum ar fi sistemele de control al motoarelor automobilelor, dispozitivele medicale implantabile, aparatele electrocasnice, uneltele electrice, jucării și alte sisteme încorporate. Prin reducerea dimensiunii și costului în comparație cu un design care utilizează un microprocesor separat, memorie și dispozitive de intrare/ieșire, microcontrolerele fac practic posibil controlul digital al unui număr mai mare de dispozitive și procese. În ceea ce privește microcontrolerele senzorii joacă un rol crucial, permițând microcontrolerelor să colecteze informații despre mediu și să răspundă în consecință.

Implementările eficiente ale procesării datelor de la senzori pe microcontrolere sunt importante din mai multe motive, două dintre ele fiind:

Constrângeri de Resurse: Microcontrolerele au adesea putere de procesare, memorie și sursă de energie limitate. Procesarea eficientă a datelor asigură utilizarea eficace a acestor resurse, maximizând performanța fără a supraîncărca sistemul.

Procesare în Timp Real: Multe aplicații, cum ar fi robotică sau sisteme auto, necesită procesare a datelor în timp real. Algoritmii eficienți pot ajuta la asigurarea unor răspunsuri rapide la intrările senzorilor, îmbunătățind fiabilitatea și performanța sistemului.

1.2 Obiective

Simularea unui giroorizont și a unui radioaltimetru folosind o plăcuță Arduino UNO împreună cu un giroscop(MPU 6050) și a unui senzor de distanță(HC-SR04). Microcontrolerul va prelua și prelucra datele de la senzori, iar apoi vor transmite datele prin intermediul interfeței seriale unei aplicații care se va ocupa de partea de afișare.

Mai întâi, se va realiza o implementare inițială bazată doar pe partea de funcționalitate, iar mai apoi se vor căuta metode de optimizare atât din punct de vedere al timpului de execuție cât și al dimensiunii programului. În final se vor compara cele două soluții obținute și se vor prezenta diferențele de performanță obținute.

2. Studiu bibliografic

2.1 Specificații Arduino UNO

Arduino UNO este o placă de dezvoltare populară, ideală pentru începători și profesioniști, bazată pe microcontrolerul ATmega328P. Designul său simplu și versatil o face potrivită pentru o gamă largă de proiecte electronice.

Specificații tehnice:

- Este un microcontroler bazat pe ATmega328P.
- Tensiunea de funcționare a Arduino-ului este de 5V.
- Tensiunea de intrare recomandată variază între 7V și 12V.
- Tensiunea de intrare (limita) este între 6V și 20V.
- Pinii de intrare și ieșire digitali - 14.
- Pinii de intrare și ieșire digitali (PWM) - 6.
- Pinii de intrare analogici sunt 6.
- Curentul continuu pentru fiecare pin I/O este de 20 mA.
- Curentul continuu utilizat pentru pinul de 3.3V este de 50 mA.
- Memoria Flash - 32 KB, iar 0.5 KB din memorie este utilizată de bootloader.
- SRAM este de 2 KB.
- EEPROM este de 1 KB.
- Viteza CLK este de 16 MHz.
- LED integrat.
- Dimensiunile Arduino-ului sunt 68.6 mm x 53.4 mm.

Nu există suport pentru multiprocesare sau multithreading pe Arduino.

2.2 Cum se poate realiza multitasking cu Arduino

1) **Implementarea ca o mașină cu stări finite:**

   O mașină de stare finită este un model de calcul care include stările în care poate fi o mașină și modul în care mașina trece de la o stare la alta; mașina poate fi doar într-o singură stare la un moment dat.

   Totodată ar trebui îndeplinite urmatoarele condiții:

   1. Menținerea timpul de execuție al tuturor funcțiilor foarte scurt.
   2. Nu folosim delay().
   3. Nu se așteptă pentru a primi ceva.

2) Folosirea librăriei Prothreads:

Protothreads este o bibliotecă pur C. Cu Protothreads poți „simula” multithreading-ul pentru sisteme bazate pe evenimente, astfel că este destul de utilă pentru programele Arduino mai complexe.

3) Multitasking cu millis():

Funcția millis returnează numărul de milisecunde de la momentul în care placa Arduino a început să ruleze programul curent, iar astfel o putem folosi pentru a gestiona mai multe sarcini fără a bloca programul. Aceasta permițând să se execute diferite funcții în funcție de timpul scurs.

4) Utilizarea întreruperilor externe:

Unii pini de pe Arduino suportă întreruperi hardware. Practic se creează o funcție care este declanșată de un buton sau alt actuator de pe un pin hardware. Când întreruperea este declanșată, programul va fi întrerupt, iar funcția va fi executată. Odată ce funcția s-a terminat, programul continuă de unde a fost întrerupt. Desigur, funcția ar trebui să fie foarte rapidă, pentru a nu opri execuția principală "thread"-ului prea mult timp.

5) Folosirea librăriei TaskScheduler

Suportă: execuția periodică a sarcinilor (cu perioadă de execuție dinamică în milisecunde sau microsecunde – frecvența execuției), număr de iterații (număr limitat sau infinit de iterații), execuția sarcinilor într-o secvență prestabilită, modificarea dinamică a parametrilor de execuție ai sarcinilor (frecvență, număr de iterații, metode callback), economisirea energiei prin intrarea în modul de repaus IDLE când sarcinile nu sunt programate să ruleze, invocarea sarcinilor bazată pe evenimente prin intermediul obiectului Status Request, ID-uri de sarcini și Puncte de Control pentru gestionarea erorilor și temporizatorul watchdog, pointerul Local Task Storage (care permite utilizarea aceluiași cod callback pentru mai multe sarcini), prioritizarea stratificată a sarcinilor, funcții std::functions (unde sunt suportate), timeout global al sarcinilor, legarea statică și dinamică a metodelor callback.

2.3 Cum funcționează senzorul de accelerometru si giroscop MPU-6050

Senzorul MPU6050 are integrat un accelerometru pe 3 axe și un giroscop pe 3 axe pe un singur cip.

Giroscopul măsoară viteza de rotație sau rata de schimbare a poziției unghiulare în timp, pe axele X, Y și Z. Acesta utilizează tehnologia MEMS(microfabricated systems comprising both electrical and mechanical components) și efectul Coriolis(forța Coriolis este o forță aparentă, de inerție, care

acționează asupra unui corp când acesta este situat într-un sistem de referință aflat în mișcare de rotație, iar din punct de vedere fizic, ea este o urmare a conservării momentului cinetic a mișcării rotative) pentru măsurare. Ieșirile giroscopului sunt exprimate în grade pe secundă, așa că, pentru a obține poziția unghiulară, trebuie doar să integrăm viteza unghiulară.

2.4 Cum funcționează senzorul de distanță HC-SR04

Senzorul HC-SR04 emite un ultrasunet la 40.000 Hz care călătorește prin aer, iar dacă există un obiect sau un obstacol pe calea sa, acesta va reveni la modul. Luând în considerare timpul de călătorie și viteza sunetului se poate calcula distanța.

2.5 Filtrarea semnalelor primite de la senzori

Măsurările din lumea reală conțin adesea zgomot. În termeni generali, zgomotul este doar partea semnalului pe care nu o dorești. De exemplu, vibrațiile de la motor adaugă zgomot dacă măsori accelerația unui kart. Filtrarea este o metodă de a elimina o parte din semnalul nedorit pentru a obține un rezultat mai neted.

Una dintre cele mai simple modalități de a filtra datele zgomotoase este prin calcularea mediei.

Calculul mediei funcționează prin adunarea unui număr de măsurători și împărțirea totalului la numărul de măsurători adunate. Cu cât incluzi mai multe măsurători în medie, cu atât mai mult zgomot va fi eliminat. Totuși, există o returnare în scădere: media a 101 măsurători nu va fi cu mult mai puțin zgomotoasă decât media a 100 măsurători.

2.6 Metode de optimizare a programului

Optimizarea codului înseamnă îmbunătățirea codului pentru a produce un program eficient. Un cod bine optimizat trebuie să fie concis, să ocupe mai puțină memorie, să aibă un timp de execuție mai rapid, un randament mai mare, un consum redus de energie și, cel mai important, rezultatul trebuie să fie același ca și rezultatul codului neoptimizat.

Tehnici pentru optimizarea codului:

1) **Eliminarea codului inutil**

   Codul inutil se referă la orice variabilă, funcție sau bibliotecă neutilizată pe care ai putea să o fi inclus în schița ta. Aceste „coduri moarte” provin adesea din etapa inițială a dezvoltării.

2) Folosirea tipurilor de date mai mici

Un tip de date indică ce poate conține o variabilă în programare. O variabilă este un nume pe care îl atribui unei porțiuni de date din memorie. Există câteva tipuri de date disponibile în programarea Arduino.

Arduino efectuează cele mai rapide operațiuni pe tipurile de date întregi (cum ar fi int și unsigned int), dar mai ales pe tipurile de date de dimensiune mică, precum byte și char. Aceasta se datorează faptului că procesorul din majoritatea plăcilor Arduino, cum ar fi ATmega328P (utilizat în Arduino Uno), are o arhitectură de 8 biți, ceea ce înseamnă că este optimizat pentru a lucra cu date de 8 biți (cum ar fi byte sau char).

3) Folosirea funcțiilor

Funcțiile sunt secțiuni denumite ale unui program care îndeplinesc o sarcină specifică. Ele previn repetarea codului și economisesc memorie, deoarece CPU-ul încarcă codul din memorie doar atunci când funcția este apelată.

4) Folosirea variabilelor locale în loc de variabile globale

Variabilele globale sunt declarate înainte de funcția void setup() și pot fi apelate în întreaga schiță, în timp ce variabilele locale sunt disponibile doar în funcția lor părinte. Variabilele globale sunt încărcate de fiecare dată când programul rulează pe Arduino-ul tău, ceea ce înseamnă că ocupă resurse și contribuie la timpul de execuție, chiar dacă bucla principală nu le folosește. În schimb, variabilele locale sunt încărcate doar când funcția lor părinte este apelată.

5) F() Strings

F() Strings oferă o altă modalitate de a afișa textul în monitorul serial sau pe un ecran. Imprimarea tradițională a șirurilor consumă o cantitate semnificativă de RAM. O modalitate de a remedia acest lucru este să salvăm șirurile în memoria flash. Pentru a face acest lucru, adăugăm F() la începutul variabilei șir.

6) Mutarea constantelor în PROGMEM

PROGMEM este un cuvânt cheie în Arduino IDE care stochează datele în memoria programului sau în memoria flash, în loc de RAM. Este recomandat să stochezi date care nu

se schimbă, precum constantele, în memoria flash, deoarece aceasta are o capacitate mai mare. Totuși, trebuie menționat că memoria flash este mai lentă la încărcare.

7) Direct Port Manipulation

Porturile sunt coduri care reprezintă registrele dintr-un microcontroller. Acestea îți permit să controlezi și să citești direct starea pinilor.

8) Eliminarea bootloader-ului

În final, poți elimina bootloader-ul Arduino pentru a elibera spațiu. Microcontrolerele sunt de obicei programate printr-un programator, cu excepția cazului în care ai un firmware în microcontroler care permite instalarea de firmware nou fără a folosi un programator extern. Acest firmware se numește bootloader, conform site-ului oficial Arduino.

Din păcate, acest software ocupă aproximativ 2000 de octeți din memoria flash. Ia în considerare eliminarea bootloader-ului și programarea cu un programator extern sau prin ISP dacă ești limitat de memorie.

2.7 Implementare aplicație pentru preluarea și afișarea datelor

Pentru implementarea aplicație care va afișa datele preluate de la microcontroler se va folosi Windows Forms. Windows Forms este un framework UI pentru crearea de aplicații desktop pe Windows. Acesta oferă una dintre cele mai productive metode de a crea aplicații desktop, bazându-se pe designerul vizual furnizat în Visual Studio. Funcționalitatea, precum plasarea prin drag-and-drop a controalelor vizuale, face ușor procesul de construire a aplicațiilor desktop.

Cu Windows Forms, dezvolți aplicații grafic bogate, ușor de distribuit, actualizat și care funcționează atât offline, cât și conectate la internet. Aplicațiile Windows Forms pot accesa hardware-ul local și sistemul de fișiere al calculatorului pe care rulează.

Comunicarea serială poate fi stabilită și în procesul de proiectare a aplicației, făcând astfel mai ușoară și rapidă interacțiunea cu Arduino printr-un PORT COM specificat.

Bibliografie

https://en.wikipedia.org/wiki/Microcontroller

-Data vizitării: 19.10.2024

https://roboticsbackend.com/how-to-do-multitasking-with-arduino/

- Autor: Edouard Renard, Data vizitării: 24.10.2024

https://roboticsbackend.com/arduino-protothreads-tutorial/

- Autor: Edouard Renard, Data vizitării: 24.10.2024

https://medium.com/@araffin/simple-and-robust-computer-arduino-serial-communication-f91b95596788

- Autor: Antonin RAFFIN, Data vizitării: 24.10.2024

https://docs.arduino.cc/hardware/uno-rev3/

- Data vizitării: 24.10.2024

https://howtomechatronics.com/tutorials/arduino/arduino-and-mpu6050-accelerometer-and-gyroscope-tutorial/

- Autor: Dejan, Data vizitării: 24.10.2024

https://ro.wikipedia.org/wiki/For%C8%9Ba_Coriolis

- Autori: Petre Pascu, 2A02:8071:3E90:6200:18E7:5F2C:16DA:988F, 2A02:8071:3E90:6200:844B:A5F4:B612:9D89, 2A02:8071:3E90:6200:78EC:74BA:AA43:7079, FoxBot, 2A02:8071:3E90:6200:58A6:C44A:D055:E50E, Miehs, 92.230.230.246, Falseufals Nicolae Coman, Data vizitării: 24.10.2024

https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/

- Autor: Dejan, Data vizitării: 24.10.2024

https://www.megunolink.com/articles/coding/3-methods-filter-noisy-arduino-measurements/

- Data vizitării: 24.10.2024

https://roboticsbackend.com/how-to-do-multitasking-with-arduino/#Lets_start_multitasking

- Autor: Edouard Renard, Data vizitării: 24.10.2024

https://jonblack.me/blog/2015/arduino-multitasking-using-finite-state-machines/

-Autor: Jon Black, Data vizitării: 24.10.2024

https://docs.arduino.cc/libraries/taskscheduler/

-Data vizitării: 24.10.2024

https://learn.circuit.rocks/nine-techniques-to-optimize-your-arduino-code

-Data vizitării: 24.10.2024

Plan de lucru

Întâlnire proiect 1:

- Alegere proiect

Întâlnire proiect 2:

- Scris partea de introducere si design

Întâlnire proiect 3:

- Scris partea de analiză și design
- Schema interfață grafică
- Implementare parțială program pentru Arduino și aplicație

Întâlnire proiect 4:

- Scris partea de implementare și testare + validare
- Terminat interfață grafică și prima variantă de program pentru Arduino

Întâlnire proiect 5:

- Optimizat program arduino
- Modificări documentație(dacă apar pentru implementarea nouă) și scris partea de concluzii

Întâlnire proiect 6:

- Testare și eventuale modificări

Întâlnire proiect 7:

- Prezentare