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 3](#_Toc180620433)

[1.1 Context 3](#_Toc180620434)

[1.2 Obiective 3](#_Toc180620435)

[2. Studiu bibliografic 4](#_Toc180620436)

[2.1 Specificații Arduino UNO 4](#_Toc180620437)

[2.2 Cum se poate realiza multitasking cu Arduino 4](#_Toc180620438)

[2.3 Cum funcționează senzorul de accelerometru si giroscop MPU-6050 5](#_Toc180620439)

[2.4 Cum funcționează senzorul de distanță HC-SR04 5](#_Toc180620440)

[2.5 Filtrarea semnalelor primite de la senzori 6](#_Toc180620441)

[2.6 Metode de optimizare a programului 6](#_Toc180620442)

[2.7 Comunicarea cu aplicatia de pe calculator 6](#_Toc180620443)

[Bibliografie 7](#_Toc180620444)

# 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 radio 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

O primă variantă de a realiza multitasking cu Arduino:

1. Menținerea timpul de execuție al tuturor funcțiilor foarte scurt. Nu spun că funcțiile tale ar trebui să aibă un număr maxim de linii de cod. Ceea ce spun este că ar trebui să monitorizezi timpul de execuție și să te asiguri că este destul de scăzut.
2. Nu folosim delay(). Această funcție va bloca complet programul tău. Așa cum vom vedea mai jos cu un exemplu de cod, există alte moduri de a obține același comportament ca și cu funcția delay(). Asta merită repetat: nu folosi delay().
3. Nu se așteptă pentru a primi ceva. Dacă programul tău ascultă un input de la utilizator, de exemplu, un mesaj text prin comunicare serială, înseamnă că nu controlezi când va avea loc acest eveniment, deoarece provine dintr-o sursă externă. Cea mai simplă modalitate de a obține input-ul utilizatorului este să aștepți pentru el și apoi să continui execuția programului când primești datele. Ei bine, nu face asta. Așa cum vom vedea mai târziu, există alte modalități de a menține comunicarea externă non-blocantă pentru restul programului.
4. Folosește ceva de genul unei mașini de stări pentru procese mai lungi. Să zicem că ai un proces care necesită cu adevărat multe acțiuni diferite și o anumită perioadă de așteptare între 2 acțiuni. În acest caz, ar fi mai bine să separi acest proces în mai multe funcții mici (vezi primul punct de mai sus) și să creezi o mașină de stări în programul tău principal pentru a le apela pe rând, atunci când este necesar. Acest lucru îți va permite, de asemenea, să calculezi orice altă parte a programului între 2 pași ai procesului.

O altă variantă este folosirea librăriei Prothreads.

Protothreads este o bibliotecă pur C. Necesită mai multe cunoștințe și este mai complexă de gestionat pentru începători și programatori de nivel mediu.

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

Multitasking cu millis(): Folosește funcția millis() pentru a gestiona mai multe sarcini fără a bloca programul. Aceasta îți permite să execuți diferite funcții în funcție de timpul scurs.

Unii pini de pe Arduino (nu toți, fii atent la asta) suportă întreruperi hardware. Practic, creezi 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 ta va fi executată. Odată ce funcția s-a terminat, programul continuă de unde a fost întrerupt. Desigur, funcția ta ar trebui să fie foarte rapidă, pentru a nu opri execuția principală "thread"-ului prea mult timp.

Aceasta poate fi o soluție excelentă pentru a executa un cod în funcție de unele input-uri externe și pentru a te asigura că nu pierzi niciun input, dacă frecvența semnalului de intrare este foarte mare.

## 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 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

int

## 2.7 Comunicarea cu aplicatia de pe calculator

# Bibliografie

<https://en.wikipedia.org/wiki/Microcontroller> - 19.10.2024

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

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

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

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

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

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

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

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

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

Plan de lucru