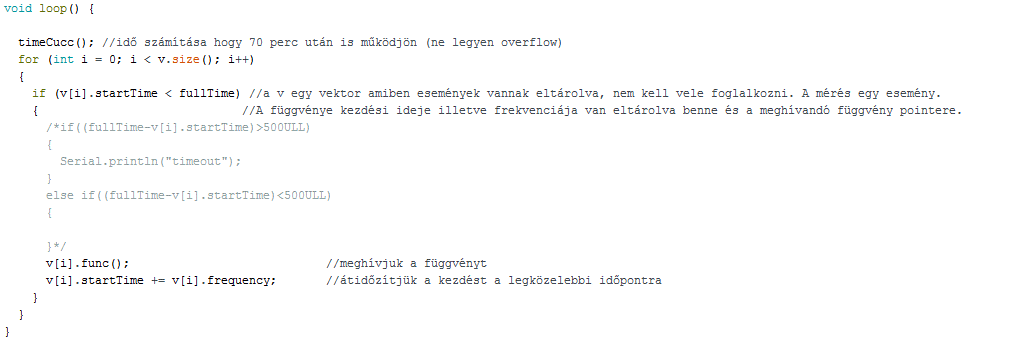
Árammérés

Használt vezérlő: Arduino nano

Microchip: ATmega328p (old bootloader)

A kód alapja egy for ciklus ami folyamatosan nézi az időt, és azt hogy az adott időpillanatban el kellett volna-e kezdődnie egy mérésnek.



A mérés 4 lépésből áll

1. idő kiértékelés megkezdése
2. mintavétel
3. kiértékelés
4. teljesítési idő kiértékelése

1: Ez annyit takar, hogy az első mérés futása előtt eltároljuk a mérés indulásának idejét ezredmásodpercekben.

2: Analog read

3: A valós értékek számítása a jelenlegi maximum-minimum keresése

4: ha lefutott 1000 mérés (10 jelzés küldés) megnézzük mennyi idő telt el az indtástól számítva ezredmásodpercekben.

Csúszóablakos megoldás index tárolása nélkül

Leírás: A max-min keresésegy 100 elemű tömbben történik az aktuális értéket egy számláló fogja jelezni és az adott helyen álló értéket felülírjuk az újjal. Ha a számláló eléri a százat nullázzuk és a számlálás újraindul. A max-min keresés minden érték beérkezésére megtörténik. 100 mérésenként kiküldjük a minimum és maximum értékeket.

Eredmény: A legnagyobb mérési frekvencia amit el tudtam vele érni az a 950 milliomod másodpercenként történő. (kb 1000 mérés/s)

Csúszóablakos megoldás index tárolásával

Leírás: Itt is egy 100 elemű tömbbünk van de eltároltuk a min és maxérték indexét. Ha a min vagy a max értéket kitoljuk a tömbből, akkor újraellenőrizzük a mérések eredményeit, egyébként elég csak az újonnan beérkezett értéket összehasonlítani az előző max és minimum értékkel. 100 mérésenként kiküldjük a minimum és maximum értékeket.

Eredmény: Gond nélkül tartja a 400 milliomod másodpercenkénti mérést (kb 2500 mérés/sec)

Blokkos megoldás

Leírás: A méréseket blokkokba osztjuk. Csak a min és maxértéket tároljuk el. A blokk elején újraírjuk őket az első méréssel. 100 mérésenként kiküldjük a minimum és maximum értékeket.

Eredmény: Gond nélkül tartja a 400 milliomod másodpercenkénti mérést (kb 2500 mérés/sec)

Valódi eredménytől való eltérés a méréskor

1000 mérés per másodperc esetén, 50hz-es hálózati frekvenciával számolva hullámonként 20 mérés jut. A max eltérés a valódi maximum és minimum értékektől ez esetben fél mérés távolság. A tévedés számításához Csak a szinusz függvényre van szükség. 360/20=18

18 egységenként (fokonként) van egy mérés.

Ennek a fele 9.

sin(99)=0.9876883

Szóval a legnagyobb tévedés a valódi maximumhoz képest kb 1.3%

Összegezve:

1-sin(90+(360/f\*50/2))=h

ahol f a frekvencia, h a maximum hiba.

|  |  |
| --- | --- |
| Mérési frekvencia | Max tévedés a valós maximumhoz és minimumhoz képest |
| 1000 Hz | 1.231% |
| 1500 Hz | 0.548% |
| 2000 Hz | 0.308% |
| 2500 Hz | 0.197% |

Max tévedéshez mérési frekvencia számítása:

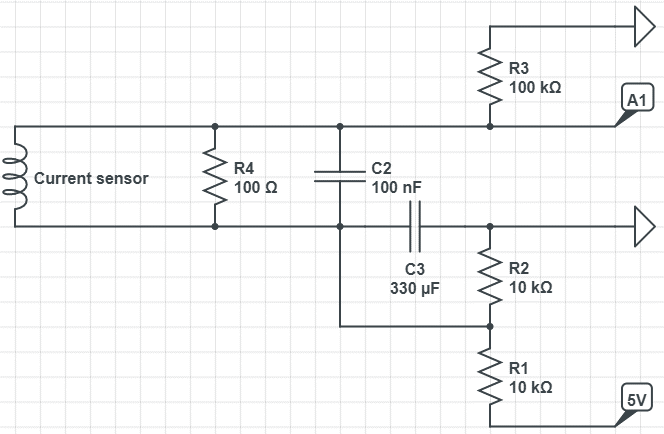
f=abs(1/(2\*(arcsin(-h+1)-90)/50/360))

Nullérték vizsgálat:

Leírás: ha megtaláltuk a nullértékeket, és megvizsgáljuk az aktuális előjeleket, akkor kitalálható, hogy lokális minimumot vagy maximumot kell keresni a függvényben. Ezzel minden hullám maximuma és minimuma kiszámítható.

Eredmény:

Áramköri felépítés



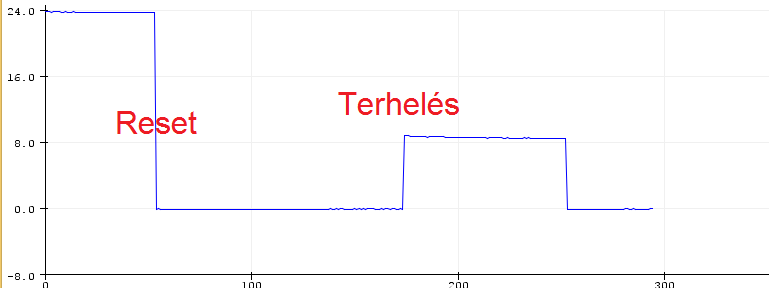
Az áramkör működése

Az áramerősség mérő 1000:1 arányban generál áramot. Ez R4 ellenálláson pontosan I\*R feszültséget eredményez. R2 és R1 feszültségosztóként működik. Mivel az arduino nem képes negatív áramot mérni, így eltoltuk a mérési tartományt. A 0 érték 2,5V az R4 ellenálláson eső feszültség ehhez képest fog változni. C3 kondenzátor biztosítja, hogy esetleges negatív áram hullám ne keltsen negatív feszültséget az áramkörben, C2 kondenzátor pedig segít a hurkok által összeszedett zaj csökkentésében. 100khz illetve 200khz-nél csúcsosodnak ki a zajok ilyen értékeknél ez a kondenzátor kb ½-es feszültségosztást végez, míg az általunk mért 50hz-re ≈34200 ohmos ellenállásként viselkedik. Ez (mivel áramosztóként viselkedik) azt jelenti, hogy az R4 ellenálláson folyó árammérőből származó áramot 1/342-ed részt csökkenti. Ez belekalkulálható a programba.

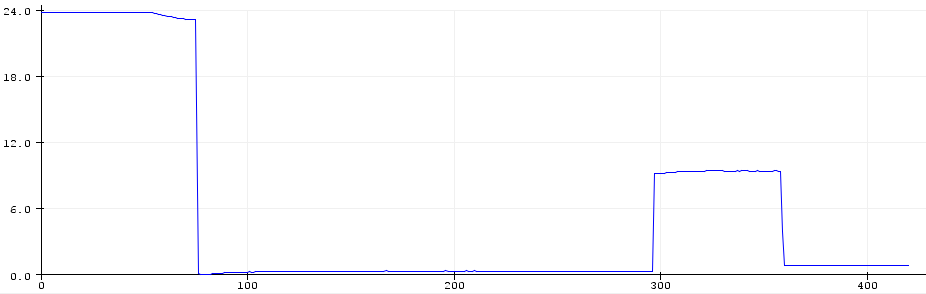
A kondenzátor impedanciája: X=1/(2\*pi\*f\*C)

Galvanikus leválasztás

Leválasztott földdel:



Leválasztott föld nélkül:



A különbség a két grafikon között az, hogy míg az elsőnél reset után stabilan beállt nullába, addig a második esetben a offset reset (beállítja az aktuális offsetet a mért értékre) gomb felengedése után elkezdett emelkedni a mért áramerősség annak ellenére, hogy nem tettünk terhelést a hálózatra, a mérés befejezése után pedig még inkább elmászott az érték. Valószínűleg a laptop (amivel használtam az arduino-t) tápja valamilyen módon beleszennyezett a mérésbe.

Ezen kívül, breadboarddal egy pontos mérésre relatívan alkalmatlan eszközt kaptunk. A beforrasztással jelentősen javítható a pontosság.

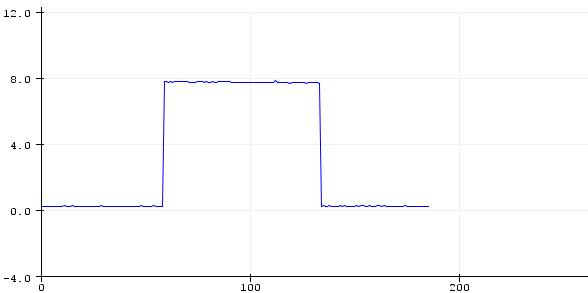
ESP 32

Használt vezérlő: FireBeetle ESP32

A kód nagyjából maradt változatlan, csak figyelni kellett, hogy melyik pinen lehet analóg jeleket olvasni, mivel ESP-ből rengeteg féle létezik. Amit itt használtam, annak a gpio lábai nagy többségén lehetett analóg bemenetet olvasni. Azonban az eredmény sokkal csúnyább mint arduino esetében. Illetve figyelni kellett, hogy az esp 3,3V –os lábbal rendelkezik kizárólag, 5V-ossal nem illetve az analóg mérési tartománya 0V-3,3V-ig terjed, és nem 10 bites, hanem 12 bites az A/D konvertere.

Szoftveres szűrés

20 mintavételből kidobom azokat amik az átlagtól 2%-nál jobban eltérnek és azokat az értékeket átlagolom. Ezzel már elérhető a kellő pontosság esp esetében is.



MQTT szerverre való adatküldés

Az mqtt szerverre való adatküldést kizárólag esp-n tudtam már megoldani, mivel ennek van beépített wifi modulja, ami lehetővé teszi az internetre való csatlakozást.

Felhasznált könyvtár:

https://github.com/knolleary/pubsubclient/blob/master/examples/mqtt\_esp8266/mqtt\_esp8266.ino

Előjött problémák:

Mqtt-re feliratkozott kliensek néha igen alacsony frissítési frekvenciával rendelkeznek, így nem feltétlenül a valós adatokat láthatjuk. Azonban a megfelelő klienssel ez a probléma kezelhető.

Pontosság növelése

A projekt végeztével összeszedtem, hogy milyen pontosság növelési lehetőségek elérhetőek. Első az áramkör optimalizálása, a második a szoftveres szűrés. Mivel nem vagyok villamosmérnök így az első lehetőséget már kimerítettnek éreztem a frekvenciaszűrő beépítésével, így a második opció mellett döntöttem. Ezzel viszont óhatatlanul jött az igény a gyorsabb mérési metódusok keresésére, mivel ezek a boardok idejük egy igen nagy részét az analóg mérés elvégzésével töltik, mely idő kihasználatlan marad a processzor ezt nem találtam optimálisnak, főleg azt tekintve, hogy minél nagyobb mintavételi frekvenciát is szeretnénk, hogy nagyobb multisample szintet el tudjunk érni.

Regiszterprogramozás

A probléma megoldását az jelenti, hogy egy kicsit mélyebben bele kell néznünk a vezérlő felépítésébe.

Az adc konverziót vezérlő regisztereket ADCSRA illetve ADCSRB-nek nevezik.

ADCSRA

Bit7: ADC bekapcsolása (1)

Bit 6: ADC mérés kezdése. Ezt a bitet abban az esetben használjuk, ha egyesével kezdünk mérésekbe. 1-re kell állítani a mérés kezdéséhez és automatikusan visszaáll nullába a mérés végeztével.

Bit 5: ADC konverzió automatikus aktiválása pozitív élnél 1-es értéknél.

Bit 4: Interrupt flag: 1-re változik az értéke, ha kész a konverzió.

Bit 3: Interrupt enable: Engedélyezzük az interruptok küldését, ha kész a konverzió, amennyiben a bit értéke 1.

Bit 2-0: Órajel szerinti előskálázás. Be lehet állítani a mintavételi frekvenciát.

Az ADCSRA regiszter kívánt tartalma „10101100” vagyis 0xAC.

ADCSRB

Bit 7, 5:3 semmi

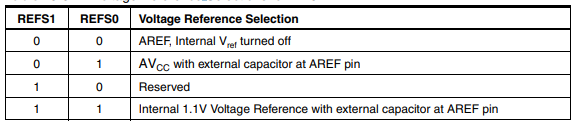
Bit 6: negatív bemeneti feszültség

Bit 2-0 Free running mode enable. Önmagától nem kezdi el a mérést, ezt külön kell elindítani Az ADCSRA megfelelő bitjének beállításával.

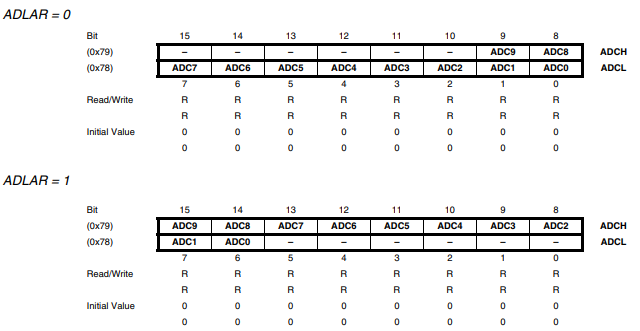
Ennek a regiszternek a kívánt állapota 0x40

ADMUX

Bit 6-7: referencia feszültség kiválasztása



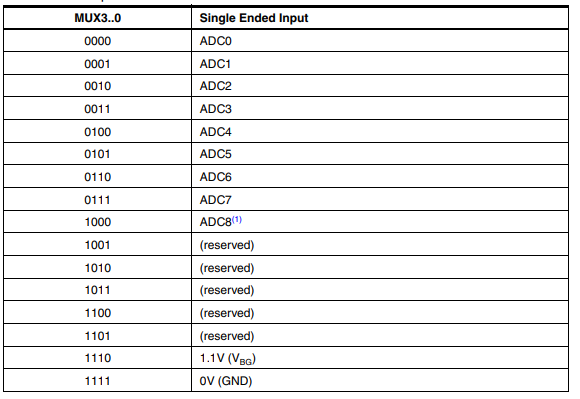
Bit 5



Az eredmény regiszterbe íródásának módját határozza meg.

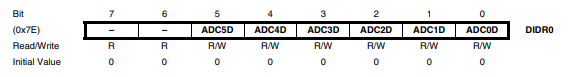
Bit 4: semmi

Bit 3:0 Mérés forrása



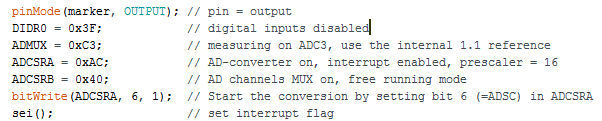
Az admux kívánt értéke 0xC3.

DIDR0

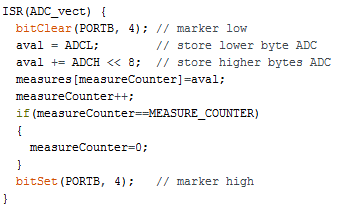


A digitális inputokat lehet kikapcsolni az analóg lábakon, 1-es érték beírásával. Ajánlott így tenni ha nem használjuk, mivel fogyasztása van. A DIDR0 kívánt értéke 0x3F

A kód az inicializáláshoz:

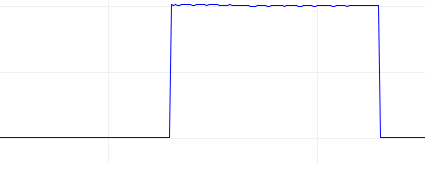


Interrupt függvény:

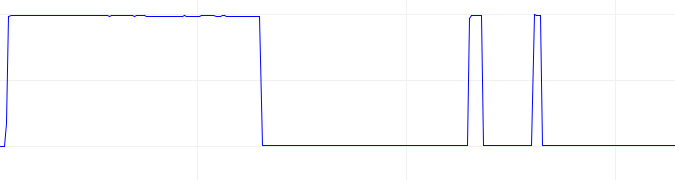


Ezzel a kóddal folyamatosan mehet az adatfeldolgozás, az egyetlen hibája, hogy nem lesz determinisztikus, mikor kapunk új adatot.

Normál megvalósítással:



Interrupttal:



Igazából különbség csak a mintavételi frekvenciában volt, de ettől több adatot tudtunk felhasználni multisample-re, így pontosabbak is lettek az adataink.