Mätning av realtidselförbrukning samt gränssnitt för smarta elmätare

Referat

todo

Innehållsförteckning

Inledning

När vi har ett riktigt kallt dygn i Finland måste vi köpa till och med trettio procent av vår elenergi från våra grannländer som Norge, Sverige, Estland och Ryssland [1]. På Energiateollisuus nätsidor kan man räkna att under de senaste tio åren importerades en femtedel av vår el från grannländer [2]. I pengar blir det proportionellt dyrare, då elpriset i regel är högt då det behövs extra el. Problemet förvärras ytterligare av att användningen av fossila bränslen minskar i landet [3], då fossila bränslen är lätta att använda för snabb justering av elproduktionen. Det vill säga när det behövs extra energi kan man starta ett ”pensionerat” kraftverk. Dessutom kommer dessa kalla dygn inte jämnt som förr, utan de kommer plötsligt nästan utan förvarning []. Alla dessa pengar flödar rakt ut ur landet, vilket kunde minskas med hjälp av optimering av elförbrukningen. Speciellt gäller det under dessa kalla dagar, men även på en vardaglig nivå.

För att som småförbrukare kunna optimera elförbrukningen under förbrukningstopparna måste man först kunna mäta sin momentana elförbrukning. Elöverföringsbolagens databaser utan extra hårdvara klarar endast av att ge förbrukningen med en timmes noggrannhet i efterhand [], vilket inte är tillräckligt för att kunna justera sin egen förbrukning. I något skede kommer nästa generationens elmätare, som redan finns i bruk i Sverige och Norge. Dessa elmätare erbjuder praktisk uppföljning med hjälp av normala Ethernetkablar (RJ45) []. Dock är det svårare att veta när dessa får börja användas i Finland, då det hänger på vem som ska administrera de nya data som produceras om elförbrukningen [].

Dessutom kommer nästa generationens elmätare troligen med vettigare uppföljning med hjälp av normala Ethernetkablar (RJ45) att komma ut inom en inte allt för lång tid.

Än så länge är således det lättaste sättet att i realtid följa med sin egen elförbrukning att installera någon typ av extern modul på sin elmätare. Syftet med den här avhandlingen är att utforska möjligheterna till att följa med sin egen elförbrukning samt gränssnitten för det här.

Olika kommunikationsgränssnitt på elmätarna

I teorin finns det alternativa sätt att få reda på sin elförbrukning. Elbolagen skickar troligtvis oftast sitt data via 2G nätverk med hjälp av simkortslösa lösningar. Dock är det säkerligen krypterat, så att inte vem som helst kan spionera på andra till exempel. Vidare kan data skickas frekvensmodulerat via själva elkablarna.

Ett sätt vore att få samma data som elbolagen får av dig. Men som nämnts fås dessa data endast en timme i efterhand. Oberoende på vilket sätt det kan tänkas att få samma data som elbolagen använder uppstår problemet att ifall man kommer åt samma data som bolagen, finns risk till att man kommer för nära själva tekniken och implementationen av elmätare, vilket kan äventyra korrektheten av data som elbolaget får.

Blinkande lysdiod

Teorin bakom den blinkande lysdioden är relativt simpel; den blinkar en gång per förbrukad Wattimme (Wh). Vidare räknar/mäter elmätaren naturligtvis jämfört med föregående blinkning. Det vill säga, när lysdioden blinkar till har det förbrukats en Wattimme sedan lysdioden blinkade till senast.

Det framtida gränssnittet HAN (Home Area Network)

todo

Externt mätsystem

todo

Mätning av realtidselförbrukning

För att i praktiken pröva på hur det går att följa med sin egen elförbrukning ämnar jag först använda ett infrarött gränssnitt som finns på elmätaren. Dock verkar det som om det här gränssnittet endast är till för elbolaget, då de inte kunde aktivera gränssnittet på distans. Detta borde vara möjligt enligt tillverkaren. Däremot använder sig elbolaget av gränssnittet då de fysiskt kommer till elmätaren för att kalibrera den vid behov [].

Eftersom man inte kommer åt elbolagens elmätare, eller mer specifikt deras utdatakanaler ur elmätaren, måste man använda den typ av gränssnitt som än så länge finns till hands. Således måste den blinkande lysdioden (Light Emitting Diode, LED) användas som praktiskt gränssnitt i denna studie.

Metod

Två mätkretsar med analog elektronik gjordes för att pröva teorin. Först gjordes en krets med ett ljuskänsligt motstånd samt kondensator, på grund av att det fanns så mycket teori om det på nätet. Senare gjordes också en annan krets med fototransistor samt utan kondensator.

Uppföljning av röd blinkande lysdiod

Grunden till experimentet hittades från Simon Auburys blogg. Hans projekt ger ett tydligt exempel på hur man kan detektera ljusimpulserna från sin elmätare [4]. Vidare får han även reda på temperatur och luftfuktighet med sitt system, vilket bortses här.

Hårdvaran i Auburys system består huvudsakligen av en enkel elkrets samt en Raspberry Pi-miniatyrdator (RPi- miniatyrdator). Elkretsen har ett ljuskänsligt motstånd (Light Dependent Resistor, LDR) som detekterar förändringar i ljusstyrka. När det blir ljusare har motståndet lägre resistans och vice versa. Det finns även ljuskänsliga motstånd vars resistans varierar den andra vägen, det vill säga resistansen ökar när det blir ljusare. Dock krävs även en kondensator i kretsen

för att mäta resistansen i motståndet. Det här görs genom att man mäter tiden det tar för att kondensatorn blir full och släpper elspänningen.

I sitt Pythonprogram importerar och använder Aubury Gpiozeros högnivåfunktioner för att lättare kunna läsa av RPis digitala pinnar för indata och utdata för allmän användning (General Purpose Input Output, GPIO) [5].

Ljuskänsligt motstånd och kondensator mha referenser

todo

Fototransistor och motstånd (spänningsfördelning)

todo

Beräkning av momentana elförbrukningen

todo

Hårdvara

På lägre nivå är den hårdvara som används ihopkopplat med hur vi faktiskt skapar data. Spänningen mellan en jordningspinne samt mätpinne ger data angående ifall vi har hög eller låg spänning.

I början fördes en skärm samt mus ner till källaren för att på plats bredvid elmätaren justera koden samt det ljuskänsliga motståndet. Senare kommunicerades det i stället via LAN, då det är kallt att jobba i källaren längre tider. En VNC-server startades via SSH kontakt för att sedan i praktiken kunna använda RPi som vanligt men via bordsdator i uppvärmda utrymmen.

Raspberry Pi

Raspberry Pi (RPi) är en miniatyrdator med digitala portar samt pinnar för inmatning. Den är till för att förmånligt använda en mikrokontroller för till exempel programmeringsprojekt. Ursprungligen tänktes den som ett hjälpmedel för att lära ut grundläggande datavetenskap i skolor samt i utvecklingsländer. []

RPi används i mätningarna eftersom den behändigt kan: hantera elkretsar med hjälp av digitala pinnar, köra program på ett Linux operativsystem samt spara data. Pinnarna på en RPi är digitala jämfört med det vanligare analoga systemet i mikrokontrollers. Dessa pinnar är till för indata och utdata för allmän användning, vilket tas senare noggrannare upp under rubriken Mjukvara. Sparandet av data tas upp i underrubriken Datauppsamling.

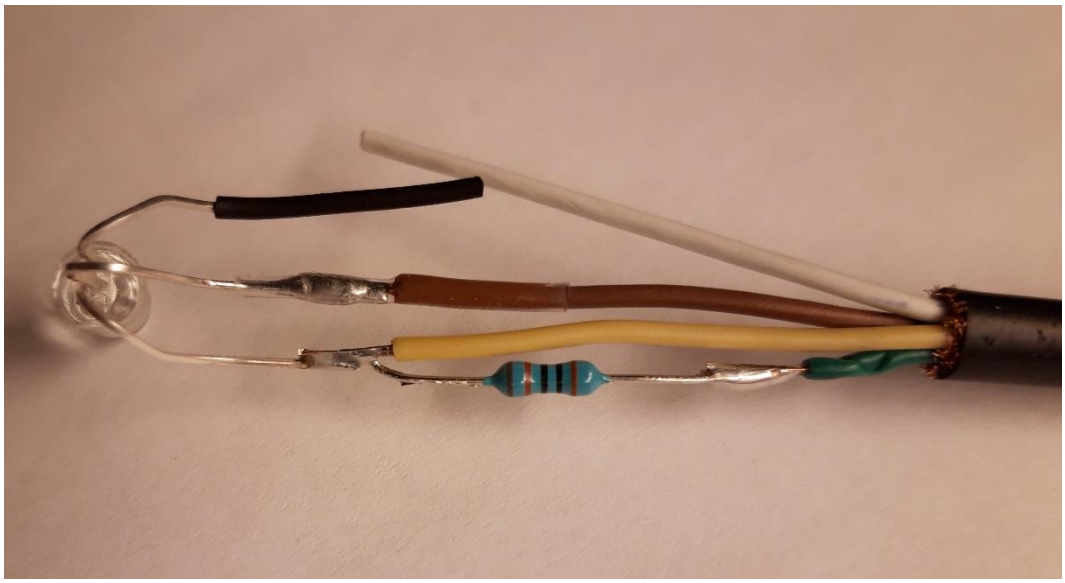
Elkretsar

För att kunna följa med och registrera källarens elmätarens lysdiodens blinkningar måste en krets lödas ihop som bland annat innehåller det ovan diskuterade ljuskänsliga motståndet.

För att kunna testa koden som detekterar/följer upp lysdiodens blinkningar från elmätaren i källaren, samt för att öva lödandet, lödde jag först ihop en gammal kretsbyggsats vars funktion är att få en lysdiod att blinka.

Följande komponenter köpptes för att löda elkretsarna: tre meter kabel innehållande fyra mindre kablar, tre olika styrkors ljuskänsliga motstånd, flera pinnar samt deras motstycken, fyra kondensatorer á 0,1; ca 0,25; ca 0,47; och 1 µF samt några röda lysdioder. Lödningarna förbereddes först genom att ändorna av kabeln skalades av så att metalltråden kommer fram. Komponenterna löddes ihop så att ena ändan fick hon typens pinnända. Sedan löddes kondensatorns minus ända till den gröna sladden i kabeln. Strömtillförselns valda sladd på 3,3 V löddes fast i positiva benet på det ljuskänsliga motståndet. Mätningssladden löddes fast till både positiva ändan av kondensatorn samt det ljuskänsliga motståndets andra ben. Den fjärde sladden som är vit behövdes inte och lämnades oanvänd.

Nedan kan vi se den andra kretsen med fototransistor och utan kondensator. Den bruna strömtillförseln löddes till transistorns x ben. Den gröna jordningssladden samt den gula mätsladden löddes fast i transistorns y ben. Den vita sladden samt transistorns z ben behövdes inte.



Mjukvara

Med tanke på mjukvara behövs sådan som känner igen spänningsskillnader för pinnarna för indata och utdata för allmän användning. Kapitlet börjar med ovannämnda samt fortsätter med programmen som bearbetar data som samlas in av pinnarna. Därefter diskuteras hur data som genererats sparas.

Indata och utdata för allmän användning

Pinnarna för indata och utdata för allmän användning (GPIO, General Purpose Input OutPut) på en RPi kan användas för att ändra analoga signaler till digitala data.

GPIOzero är ett högnivågränssnitt för programmering av pinnarna i RPis inmatnings- samt utmatningskanaler för generell användning.

Mest intressanta klassen för mig som GPIOzero har är LightSensor (LDR) [6]. Med hjälp av metoder som vänta_på_ljus (wait_for_light()) samt vänta_på_mörker (wait_for_dark()) verkar det som om man kunde få till stånd någonting.

Som alternativ kan man välja att inte använda högnivågränssnittet GPIOzero och istället använda direkt samma som GPIOzero, det vill säga importera RPi.GPIO. Då används som grundläggande programmeringsaxiom att inmatningen för en viss pinne (GPIO.input(pinNumber, True)) är sann, alltså att den pinnen har det digitala inmatningsvärdet högspänning, booleska värdet ett eller det synonyma booleska värdet Sant (True). Då har RPi.GPIO systemet detekterat att den analoga strömmen

som matas in till pinne p är över ett visst gränsvärde y. Motsvarande kan naturligtvis konstateras för utmatningen av samma eller en annan pinne.

Pythonprogram

Jag skrev ett Pythonprogram som använder kretsen och som körs på en Raspberry Pi. Programmet kan följa med momentan elförbrukning med några sekunders mellanrum.

Det första programmet väntar på att antingen en programmerbar tid har gått, eller på att värdet på RPis digitala indatapinne går från låg spänning till hög spänning (ungefär 1,4V) []. Därefter tar programmet reda på vilketdera argument som triggades genom att hålla reda på ifall föregående trigger(uppladdning av kondensatorn/tidskrav) var en äkta blinkning av lysdioden.

Det andra programmet hade som största skillnad till det första att det inte använde GPIOzeros högnivåfunktioner.

Senare skrevs en modifierad kod för fototransistorkretsen. Fototransistorn släpper igenom ström överhuvudtaget endast då när den detekterar ljus. Det finns inte samma risk som med det ljuskänsliga motståndet, då en instans av en uppladdad kondensator kunde ge en ”falsk” blinkning. Således blev den här kretsen mycket enklare.

Datauppsamling

Eftersom insamlingshastigheten på data som samlas är relativt långsamt samlas data primärt på samma minneskort (mirco SD) som RPi är installerat på. Vidare bestyrks det här av att mängden data per sampel är även lågt. Senare kan möjligtvis en databas upprättas för att göra data mer åskådligt och användbart.

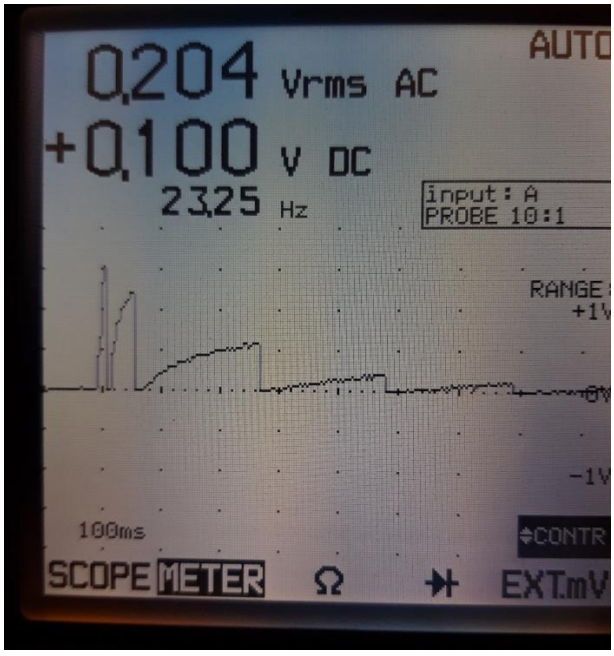
Resultat

Eftersom Pythonprogrammen vid mätningen av den momentana elförbrukningen använder tiden sedan lysdioden blinkade den senaste gången, är inte mätvärdena exakta. De ligger proportionellt mer efter nutid då lägre effekter mäts. Detta är på grund av att programmet vid låg effektförbrukning får vänta längre tid på att en

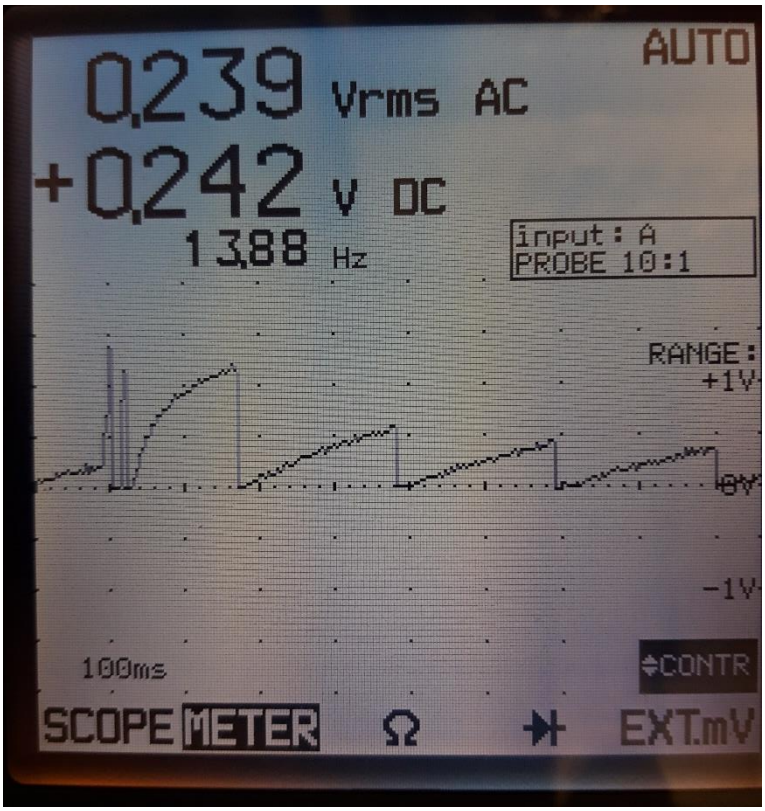
wattimme har förbrukats. Vidare dominerar det här felet det faktum att en RPi med sitt Linux-operativsystem inte är lämplig för mycket tidskritiska beräkningar, då Linux håller på med flera saker på samma gång. Således kan det hända att operativsystemet gör någonting helt annat precis då när en blinkning från lysdioden kommer [7].

För att förstå GPIOzeros högnivåfunktioner måste deras källkod läsas. Vid närmare granskning av källkoden märktes att programmet är hårdkodat till att vänta en tiondelssekund på att kondensatorn töms. Det här är för länge med tanke på att en ljusimpuls sker under ungefär drygt trettio millisekunder. Då kan programmet missa en ljusimpuls delvis eller helt. Således skrevs en ny version av programmet som inte använde GPIOzero.

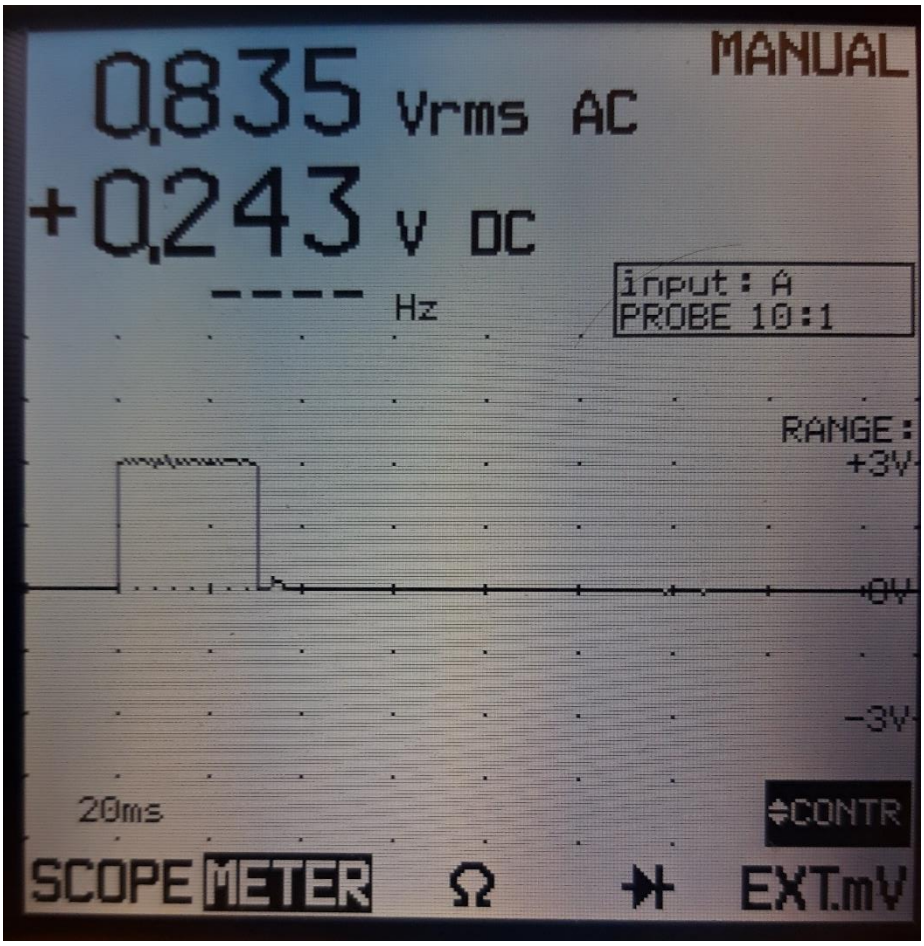
En sak som orsakade problem med brytningstiden för kretsen med det ljuskänsliga motståndet var ljuset i källaren. På bild z nedan är lampan i källaren släckt och det syns att det tar länge för uppladdningen av kondensatorn efter att ljuspulsen försvunnit.



På bilden x nedan används 200 ms som brytningstid. Under en ljusimpuls hinner kondensatorn fyllas tre gånger och mätningen kommer upp i cirka 1,4 V.



Kretsen med fototransistor är bättre då den kan reglera strömtillförsel absolut sätt, det vill säga den släpper inte igenom någon ström ifall det är mörkt och lysdioden lyser inte. Vidare är den även lättare att koda och det här ser vi även från Scope metern i bilden nedan. [].



Analys av resultat

Efter att ha studerat Gpiozeros dokumentationssidor [] samt experimenterat med deras kod blev det klart att det blir lättare och effektivare att förbigå deras funktioner och istället använda RPis egna lågnivåfunktioner (som också Gpiozero använder i sin kod). Orsaken var att GPIOzero använde sig av en inbyggd väntetid som väntade på att kondensatorn skulle tömmas. Dock ville jag slippa den här väntetiden, för att annars kanske man missar blinkningen helt eller delvis.

Det opraktiska med ljuskänsliga motstånd är att det även behövs en kondensator för att detektera resistansen i motståndet genom att mäta tiden det tar för kondensatorn att fyllas. Således är det en bättre idé med fototransistor.

Diskussion

todo

Utvecklingsmöjligheter

todo

Referenser

1. https://energia.fi/uutishuone/materiaalipankki/sahkon_hankinta_energialahteittain_2007-2019.html#material-view

2. https://www.fingrid.fi/sahkomarkkinat/sahkojarjestelman-tila/

3. https://energiavirasto.fi/toimitusvarmuus

4. https://simon-aubury.medium.com/home-power-monitoring-65d0fded7769

5. https://github.com/saubury/power-pi/blob/master/power-pi.py

6. https://gpiozero.readthedocs.io/en/stable/#

7. https://pypi.org/project/RPi.GPIO/

https://gpiozero.readthedocs.io/en/stable/api_input.html?highlight=lightsensor

https://www.raspberrypi.org/

Bilagor

Pythonskript

Resultat textfiler