IA2109EM Energie Monitor

Lees eerst de voorgaande hoofdstukken over de Voltmonitor, Amperemeter en Wattmeter.

Inleiding. De 'slimme meter' is alleen handig voor de netwerkbeheerder en de energieleverancier. Voor de gebruiker is er niets slim aan de meter. Daarnaast; de cyber-beveiliging en privacy-waarborg is zeer twijfelachtig.

Wel willen diverse commerciële partijen graag hun aanvullende energie-monitor-systeem 'verkopen'; het liefst met de nodige extra's zoals 'gemakkelijke website' , 'apps' en abonnementen. Wat krijg je voor je geld?, Wat kun je er mee? Bespaart het energie? Krijg je inzicht in je energie-verbruik? Wie is de data-koe die gemolken wordt?

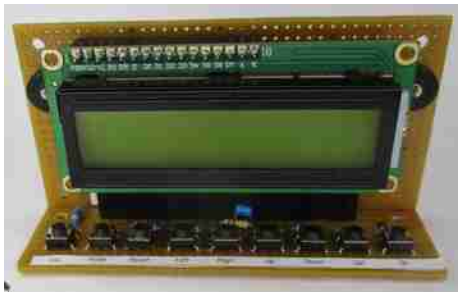
	Realtime inzicht	Inzicht zonne energie	Verplicht abonnement	Inzicht historie >12 maanden	Draadloos	Prijs	Incl. uitbreiding zon/abbo
EnergyFlip	✓	✓	Nee	✓	✓	€ 99,-	€ 154,-
BeeClear	✓	✗	Nee	✓	✓	€ 150,-	n.v.t.
CEMM Basic	✓	✓	Nee	✓	✗	€ 179,-	€ 239,-
EARN-E	✓	✓	€ 1,- p.m.	✓	✓	€ 30,-	€ 2,50 p.m.
Enelogic YouLess	✓	✗	Optioneel	✗	✗	€ 99,-	€ 19,- p.m.
Energy Logger BLUE	✓	✗	Nee	✗	✓	€ 47,-	n.v.t.
Geo Trio II	✓	✗	Nee	✗	✓	€ 125,-	n.v.t.
HomeWizard	✓	✗	Optioneel	Optioneel	✓	€ 30,-	€ 1,- p.m.
IUNGO	✓	✓	Nee	✗	✓	€ 189,-	€ 305,-
Meternmd	✓	✗	Optioneel	Optioneel	✓	€ 250,-	vanaf € 1,50,-
Plugwise Smile	✓	✗	Nee	✓	✓	€ 119,-	n.v.t.
Qubida SEM-1000	✓	✗	Nee	✓	✗	€ 100,-	n.v.t.
Slimme Meter Wifi Adapter	✓	✓	Nee	✓	✓	€ 70,-	n.v.t.
Solarwatt Manager Flex	✓	✓	Nee	✓	✓	niet bekend	niet bekend
Toon®	✓	✓	€ 4,50 p.m.	✓	✓	€ 200,-	n.v.t.

In de praktijk is de slimme meter vaak stommemetergedoe; de communicatie en doorgeven van meetwaarden is afhankelijk van de netbeheerder, en ligt regelmatig plat. De actueel gemeten waarden worden met grote vertraging afgerond tot kwh. Onthutsend hoeveel systemen de opgewekte zonne-energie en totaal energieverbruik niet eens kunnen monitoren, terwijl een energie-monitor-systeem helemaal niet zo gecompliceerd en duur hoeft te zijn.

Overigens, de ouderwetse CV-klok-thermostaat wordt tegenwoordig ook al als 'slimme meter' verkocht; dat is slim!

Bij het meten van het energiegebruik (of energieopbrengst van een zonnestroominstallatie) wordt de gemeten spanning vermenigvuldigt met de gemeten stroom, ofwel een energie-meter kent eigenlijk twee meetingangen; één voor het meten van de spanning en één voor de stroom.

Een ***Wattmeter (energy meter, power meter)*** wordt gebruikt om het werkelijke momentele (-hier en nu-) energiegebruik te meten in Watt, niet te verwarren met de kWh-meter die een verbruikdoorteller is.

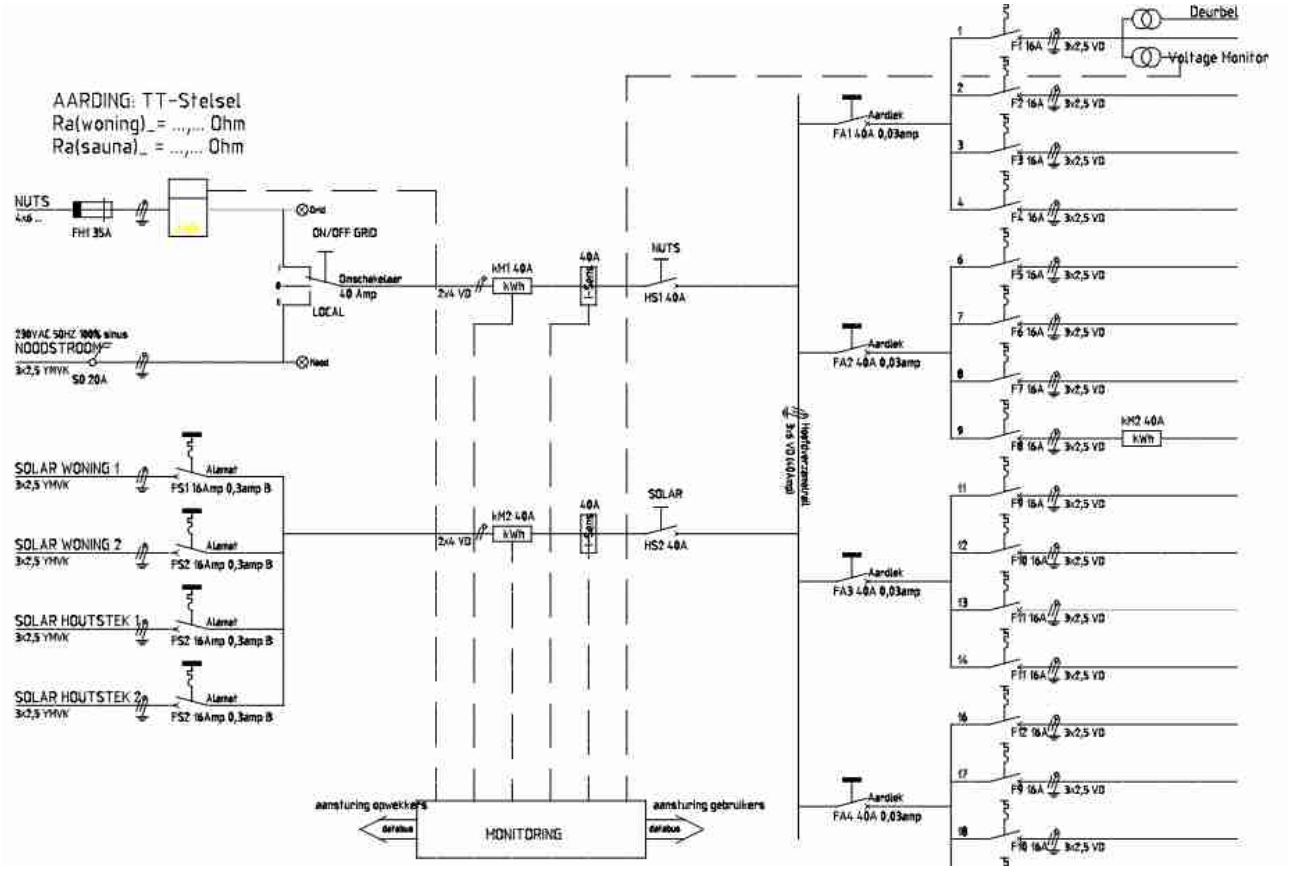


De hier gepresenteerde Energie Monitor is het IA2109-shield voorzien van een passende voorzet voor het gelijktijdig meten van spanning- en twee stroom-meetingangen; één voor de netstroom en één voor de zonnestroom. Voor een grotere nauwkeurigheid is de meetvoorzet voorzien van een analoge spanningsreferentie.

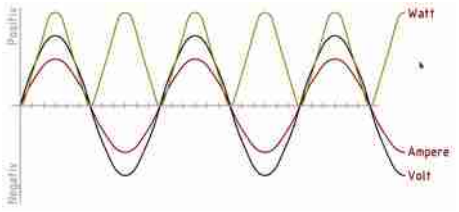
In de software wordt gebruik gemaakt van automatische kalibratie, voedings-spanningcontrole, temperatuur-compensatie, en foutcode-afhandeling. Met een mode-drukknop kunnen de de verschillende instellingen en metingen op het display weergegeven worden.

Voor communicatie is de meter voorzien van een UART-aansluiting met een RS232-module. Via deze aansluiting kan de meter de actueel gemeten waarden aan een datalogger of PC doorgeven. Deze gegevens kunnen weer gebruikt worden voor het aansturen van een modulair gebalanceerde PV-installatie.

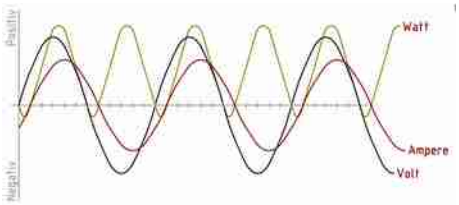
Hieronder is aangegeven hoe en waar een energiemonitor-systeem in een schakel-en verdeelinrichting ingebouwd kan worden.



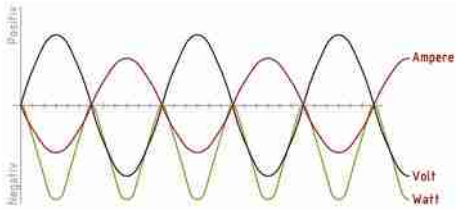
Energie, Watt, Volt en Ampere. Voor veel mensen is ***energie*** een vanzelfsprekend containerbegrip. Iedereen gebruikt het, maar slechts weinigen kunnen het goed omschrijven. De gebruiker zegt dat hij 'stroom gebruikt', terwijl er in feite ***energie*** wordt verbruikt.



Energie wordt uitgedrukt in Joules. Elektrische energie wordt uitgedrukt in ***Watt***, waarbij 1 Watt hetzelfde is als 1 Joule. De afgenomen energie is gelijk aan de momentele spanning maal de momentele stroom, ofwel P=U*I. Toch, bij wisselspanning is de gemiddelde spanning nul volt, en de gemiddelde stroom nul ampere. Wanneer spanning en stroom synchroon (=positief in fase) variëren (zoals bij een zuivere ohmse belasting), zal bij het weergeven van spanning, stroom èn het vermogen (=energie) in één-en-dezelfde grafiek, de vermogenslijn totaal positief zijn. Overigens, bij spanning en stroom in fase, is de vermogenslijn eigenlijk een dubbelfasig gelijkgericht signaal.



We weten dat spanning en stroom niet in fase zijn bij gebruik van bepaalde apparaten, zoals wasmachine, centrifuge, stofzuiger, lasapparaat, enz., en dat er dan sprake kan zijn van schijnbaar vermogen (VA) en werkelijk vermogen (in Watt). (Inductief; stroom loopt voor. Capacitief; stroom loopt na.) Bij een reactieve (=niet ohmse) belasting, zien we de vermogenslijn van uitsluitend positief, een klein stukje door de nulllijn zakken naar 'negatieve energie'. Deze 'gedeeltelijke negatieve energie' wordt ***blindstroom*** genoemd.



Bij een normale huisinstallatie wordt altijd energie afgenomen, en is de energierichting meestal één kant op, de positieve richting genoemd. Bij een zonnestroom-installatie wordt energie opgewekt, en is de energierichting tegengesteld (niet 180° in fase verschoven, maar stroomrichting is negatief!). Afhankelijk van het momentele huishoudelijk gebruik, kan er dan zelfs energie naar het elektriciteitsnet teruggeleverd worden.

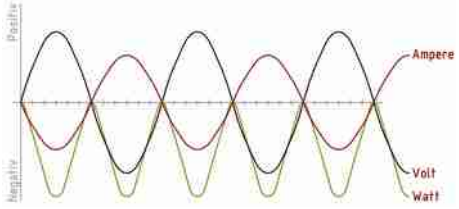
Aan de hand van de grafieken kan opgemaakt worden, dat wanneer de spanningstop U_{max} samenvalt met de stroomtop I_{max} er energie afgenomen wordt, en dat wanneer de stroomtop 180 graden ***verschoven lijkt*** ten opzichtte van de spanningstop er energie terugeleverd wordt. Bij een zuiver ohmse zonnestroom-installatie ***lijkt*** de stroom 180 graden uit fase (maar is in werkelijkheid synchroon tegengesteld (ofwel negatief)); de vermogenslijn zakt in zijn geheel tot onder de nullijn, en er wordt energie aan het net teruggeleverd.

Overigens, in werkelijkheid zijn spanning en stroom helemaal geen zuivere sinusen, zodat ook de vermogensgrafiek geen mooie sinusgolf zal zijn. Toch, aan het principe van afnemen en terugleveren van energie verandert er niets.

Kortom; het ***schijnbaar vermogen*** is de spanning maal de stroom; 230 VAC * 16 ampere = 3680 VoltAmpere, ofwel ***VA***.
Het ***blindvermogen*** is de spanning maal de blindstroom; 230 VAC * 1 ampere = 230 VoltAmpereReactief, ofwel in ***Var***.
Het ***werkvermogen*** is de effectieve spanning maal de effectieve stroom; 230 VAC * 15 ampere = 3450 ***Watt***.

Bij energie-monitoren met zonnepanelen zijn de Watt-metingen het belangrijkst. De rest is eigenlijk alleen maar bijvangst. Een belangrijke bijvangst bij zonnepanelen is de monitoring van de netspanning.

Energierichting bepalen. Het omslagpunt tussen energie afnemen en energie terugleveren zit in de positie en grootte van I_{max} (en dus ook I_{min}.) ten opzichtte van U_{max}. Wanneer in één periode I_{max} samenvalt met U_{max} is de energierichting positief en wordt energie afgenomen.



Wanneer in één periode I_{min} samenvalt met U_{max} is de energierichting negatief en wordt energie teruggeleverd.

Blindstroom en zonnepanelen. Grootverbruikers zoals industriële installaties met motoren, veroorzaken door fase-verschuiving grote blindstromen. ***Blindstroom*** is energie dat weliswaar niet gemeten maar wel degelijk de installatie belast, en daardoor ongemerkt zorgt voor warmteontwikkeling en elektrische beveiligingen ontregelt. Grootverbruikers moeten zelf zorgen voor ***cos-phi-***, danwel ***blindstroom-compensatie*** met condensatoren en/of spoelenbanken, en worden beboet wanneer ze dat nalaten. De reden daarvoor is dat grootverbruikers met hun blindspanning- en stromen het elektriciteitsnet en de energiecentrale's kunnen ontregelen. En omdat effectieve blindstroom-beveiliging nauwelijks te realiseren is, kunnen zonnepaneel-installaties op explosieve wijze defect raken.

Ook bij zonnestroom-installaties kan er sprake zijn van blindstroom, ofwel ongewenste gedeeltelijke fase-verschuiving. Bij een steeds groter gebruik van particuliere zonnestroom-installaties, neemt ook hun gezamelijke invloed op de net-blindstroom toe. Voor particulier gebruik is een blindstroomfactor van 0,85...1,00 acceptabel.

Aanmelden zonnepanelen bij netbeheerder. Vanaf 27 april 2021 moeten ook particuliere zonnestroom-installaties voldoen aan de europese eisen **Netcode Elektriciteit**. Dit betekent dat alle met het elektriciteitsnet verbonden zonnestroom-installaties via de website www.energieleveren.nl moeten worden aangemeldt.

Alleen goedgekeurde typen zonnestroom-omvormers mogen gebruikt worden; zie https://www.netbeheernederland.nl/_upload/Files/ Regulering_ 20_421e9b124c.pdf.

Vanaf april 2022 geldt hetzelfde voor alle aan het net gekoppelde netbatterijen en/of -accu's.

Salderen, voor of achter de meter. Bij de meeste woningen ('kleinverbruikers', ofwel < 3x80amp) wordt de zonnepaneel-installatie als een 'eindgroep' ***achter de kWh-meter*** aangesloten, zodat eerst de zelf opgewekte energie gratis en belastingvrij kan worden gebruikt. Een eventueel overschot aan met zonnepanelen opgewekte energie zal zo ook aan het net teruggeleverd worden. ***Vóór de meter*** invoeden van teruglevering wordt vaak gedaan door bedrijven als aparte zonnestroom-netaansluiting. Dit zijn dus geen kleinverbruikers, en de teruglevering gaat met een aparte kWh-meter en overeenkomst met de netbeheerder en energiehandelaar.

Bij de oudste mechanische ferraris(draaischijf)-meter kan het telwerk vaak ook terugdraaien, zodat de kWh-meter de aan het net teruggeleverde energie tegen de kostprijs inclusief belastingen en heffingen verrekend. In de zomer bij een zonnige dag draait de meter vol terug (en wordt er lekker verdiend), en in de winter tijdens de avond en nacht registreert het telwerk de verbruikte energie. Een zeer groot voordeel van de ferraris-meter is dat deze in feite de teruggeleverde energie tegen dezelfde prijs verkoopt als dat de opgenomen energie wordt ingekocht, en dat eenmaal per jaar afgerekend wordt.

'Het net' wordt zo door veel zonnepaneel-eigenaren gebruikt als 'accu'; 's zomers opladen en 's winters ontladen. De netbeheerder (Enexis, Liander, Stedin, enz) bezit alleen het netwerk (de 'draden') tussen de opwekkers (energiecentrale's, windparken, zonneparken, enz.) en de verbruikers. Het aanleggen, onderhouden, en heen en weer sturen van energie over het netwerk is niet gratis. Het aandeel aan zonnestroom is inmiddels zo groot en weersafhankelijk, dat de stabiliteit van het elektriciteitsnet in gevaar komt. Kleine zonnestroom-opwekkers kunnen nog steeds gebruik maken van het net als gratis jaarverbruik-accu. Vanaf 2024 zal dit niet meer gratis zijn.

Kleinverbruikers mogen niet méér terugleveren dan hun netaansluiting aan kan, met een maximum van 5000 kWh per jaar. De energiehandelaar (Essent, Eneco, BudgetEnergie, Oxxio, enz.) bepaalt niet alleen de prijs van de afgenomen energie, maar ook die voor de teruggeleverde energie (feed-in tarief). Vanaf 2020 moet over elke geleverde kWh netbeheerkosten worden betaald, zodat ook de teruggeleverde energie apart geregistreerd moet worden.

Omdat de terugleverkosten door de energiehandelaar verrekend moeten worden, moet de kleinverbruiker-ferrarismeter daarom vervangen worden door een kWh-meter met een apart teruglevertelwerk. Een slimme meter is niet verplicht!

Het verrekenen van de afgenomen en teruggeleverde energie voor kleinverbruikers wordt ***salderen*** genoemd, en de wijze waarop is wettelijk vastgelegd. Eénmaal per jaar (met de jaarnota) ***moet*** de energiehandelaar deze verrekening toepassen.

Omdat het aandeel zonnestroom steeds groter wordt (en daardoor ook de netbeheerkosten) wordt de salderingsregeling afgebouwd, ofwel vanaf 2025 is salderen nog slechts mogelijk over een beperkt aantal kWh's (tot een bepaald maximum) tegen een vastgestelde vergoeding.

Meer stroom afgenomen dan teruggeleverd

Afgenomen vs teruggeleverd	normaal/dal	kWh	prijs per kWh	kosten
afgenomen	normaaltarief	1000	0,06821	€ 68,21
teruggeleverd	normaaltarief	2000	-0,06821	€ -136,42
afgenomen	daltarief	2000	0,05432	€ 108,64
teruggeleverd	daltarief	500	-0,05432	€ -27,16
Subtotaal		**500**		**€ 13,27**
Belastingen (overheid)		**kWh**	**tarief per kWh**	**kosten**
Energiebelasting		500	0,09428	€ 47,14
Opslag Duurzame Energie		500	0,03000	€ 15,00
Subtotaal				**€ 75,41**
btw (21%)				€ 15,08
TOTAAL				**€ 90,49**

Afnemen, opwekken, verbruiken en *terugleveren* worden vaak met elkaar verward. De zonnepanelen wekken energie op, waarvan een deel zelf gebruikt wordt en een deel teruggeleverd ***kan*** worden. Het energieverbruik is het totaal van de zelf opgewekte en verbruikte energie ***plus*** de via het net afgenomen energie. De aan het net teruggeleverde energie is het overschot aan zelf opgewekte energie, op tijdstippen dat het eigen verbruik te klein was.

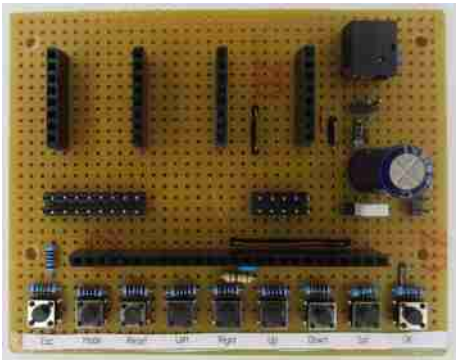
Stel de kWh-meter heeft 2500 kWh geteld, en het terugtelwerk heeft 1000 kWh geteld. Bij sommige energiehandelaren betaal je dan voor 1500 kilowattuur, ofwel saldering volgens de ferraris-verrekenmethode. Wanneer er in totaal meer stroom opgewekt werd dan verbruikt is, wordt het verschil slechts vergoed tegen een veel lager tarief (producententarief).

Ofwel, niet alle teruggeleverde energie telt mee voor volledige saldering. Afhankelijk van het contract met de energie-handelaar kan het zelfs voordeliger zijn om van salderen helemaal af te zien, en de zonnepanelen uit te richten naar de meest optimale midwinter-situatie.

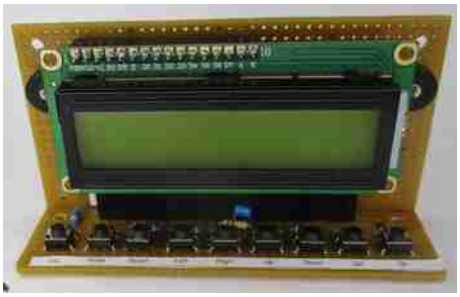
Energie-Monitor-modules. Voor het monitoren van een huisinstallatie met netaansluiting en zonnestroom-installatie, wordt voor de netspanning gebruik gemaakt van spanningmeet-voorzetmodule IAVM-01B. Voor het meten van de net- en zonne-stroom twee voorzetmodules IACC-01B. Voor een betere en stabiele meetnauwkeurigheid wordt refentiespanning-module IAVR-01A gebruikt.

IAAM-01. Omdat het project nog in ontwikkeling is, wordt gebruik gemaakt van een 'analoge-module-bus-print' gecodeerd als IAAM-01, bedoeld voor het plaatsen van vier analoge insteekmodules. Eventuele meetvoorzet-aanpassingen kunnen dan als module worden uitgeprobeerd. Pas wanneer de Energie Monitor goed werkt, kan een definitieve versie als één print worden gemaakt.

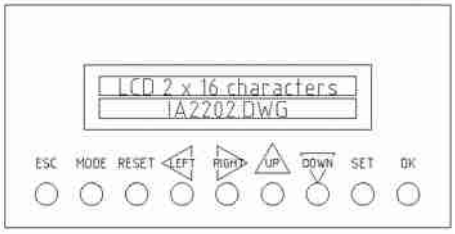
Om de busprint in de toekomst ook voor andere schakelingen te kunnen gebruiken, zijn ook de niet gebruikte digitale poorten naar diverse pinheaders doorverbonden.



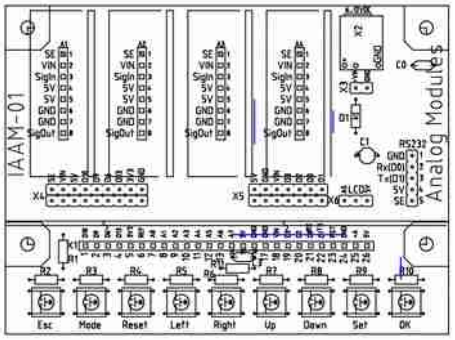
Keypad, 1x9. Het IA2109-shield is oorspronkelijk ontwikkeld als meetinstrument-display-front-end, waarbij de front-afmetingen zo compact mogelijk zijn. Een 3-maal-3 negen-toets analoge keypad neemt nogal wat ruimte in beslag, terwijl bij vaste meetinstrumenten een keypad zelden gebruikt zal worden.



Het keypad kan ook minimalistisch uitgevoerd worden als een één rij negentoets-keypad direct onder het display. Op de analoge-modules-busprint is het keypad daarom direct vooraan onder het IA2109-schield, als negen printdruktoetsen geplaatst. In een definitieve versie kunnen deze vervangen worden door haakse printdrukknoppen. Begin met het plaatsen en vastzetten van alle printdruk-knoppen. Helaas is de passing door de metrische knoppenmaatvoering niet echt optimaal te noemen.



X1. Plaats X1, de 26-polige IA2109-pinheader voor het IA2109-shield.



X4-I/O. Plaats X4, de 2 maal 9 pinheader voor de I/O D6, D9, D10 en D13.

X5-I/O. Plaats X5, de 2 maal 7 pinheader voor de I/O D0, D1, D2 en D3.

X6-jumper. Plaats X6. Met een jumper op X6 kan de achtergrond-verlichting van het IA2109-shield omgeschakeld worden van +5VDC naar uitgang A5 voor PWM-sturing.

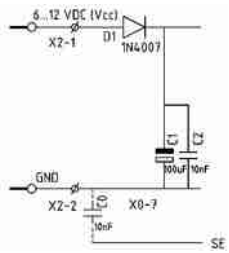
RS232. De busprint is ook voorzien van een pinheader voor aansluiting op een RS232-module. Wanneer deze gebruikt wordt, zijn de I/O-ingangen D0 en D1 al in gebruik.

X2-, X3-voeding. Plaats X2 en X3. De print is voorzien van een DC-jackplug-chassisdeel voor de externe voeding, die ergens tussen 6...12 volt DC mag zijn. Voor het aansluiten van een batterij/accu-voeding kan X3 gebruikt worden.

Analoge bus A1...A4. Plaats de female pinheaders A1, A2, A3 en A4. Omdat de pinning van elke pinheader hetzelfde is, onstaat een klein analoog bussysteempje voor vier analoge voorzetmodules. A0 wordt gebruikt voor het keypad, en A5 voor aansturing van de LCD-achtergrondverslichting.

Bruggen. Wanneer alle male en female pinheaders geplaatst zijn, kan op de componentzijde de vier draadbruggen worden aangegeven.

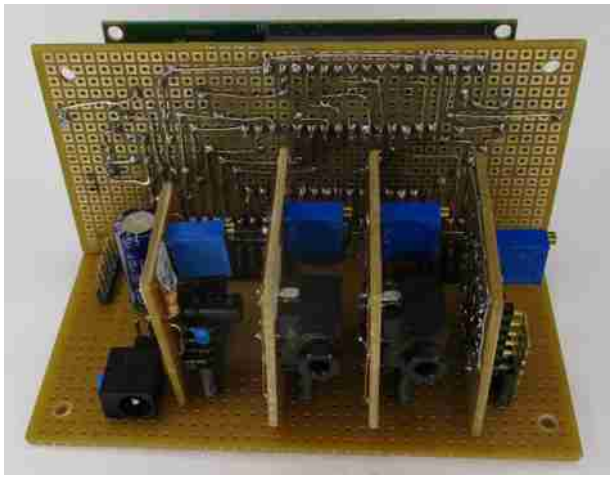
Voeding. De eerste schakeling is de voeding. X2 is een DC-jackplug (***ton-bus***, ofwel ***barrel-jack***) zodat de Energie-Monitor gevoed kan worden met een externe voedingsadapter van 6...12 VDC. Diode D1 (1N4007) beveiligt de busprint tegen een verkeerdom aangesloten voeding.

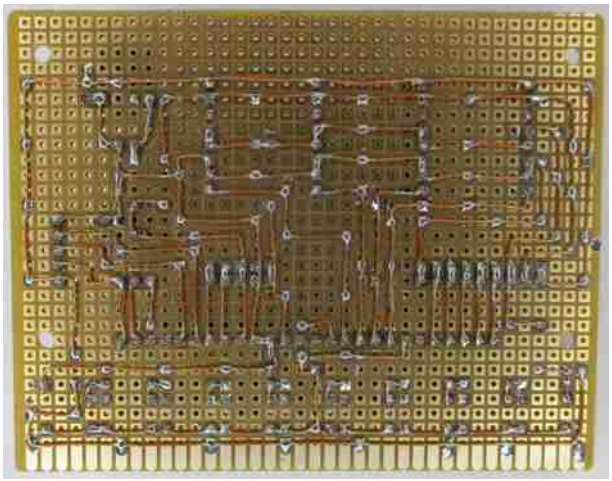


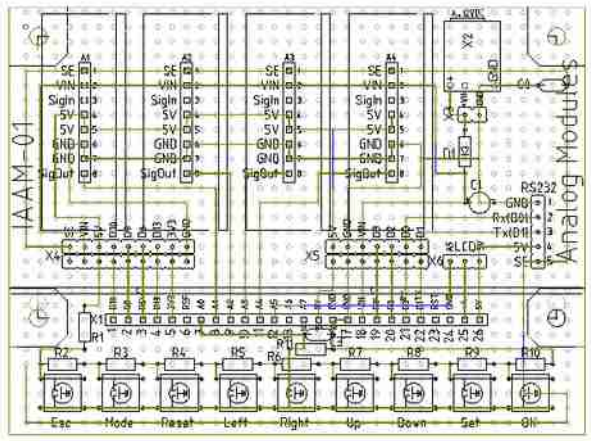
Condensator C1 (100uF/25 Volt) zorgt voor extra afvlakking en schakelpower wanneer deze gevraagd is. De voedingsspanning wordt aan alle analoge voorzetschakelingen, en via X1-18 VIN aan de Arduino doorgegeven. Arduino's eigen spanningsregeling maakt er dan 5 VDC van, welke dan weer op X1-15 en -26 beschikbaar is.

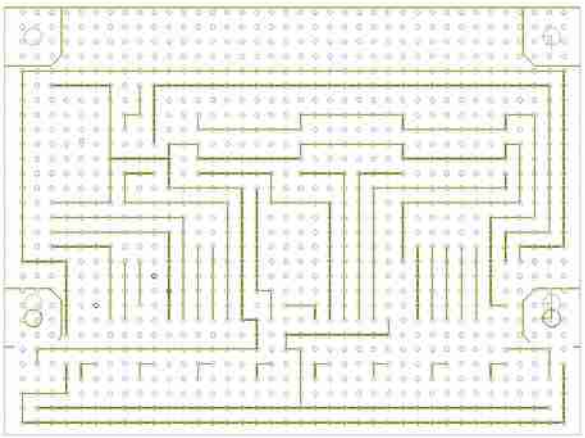
Keypad-weerstanden. Plaats eerst de draadbrug onder R10, en plaats dan de weerstanden R1 tot en met R10 (alle 1 K). Alleen weerstand R11 is 100 K. Condensator van 10 nF (C3) tussen A5 en GND werkt de contactdender weg.

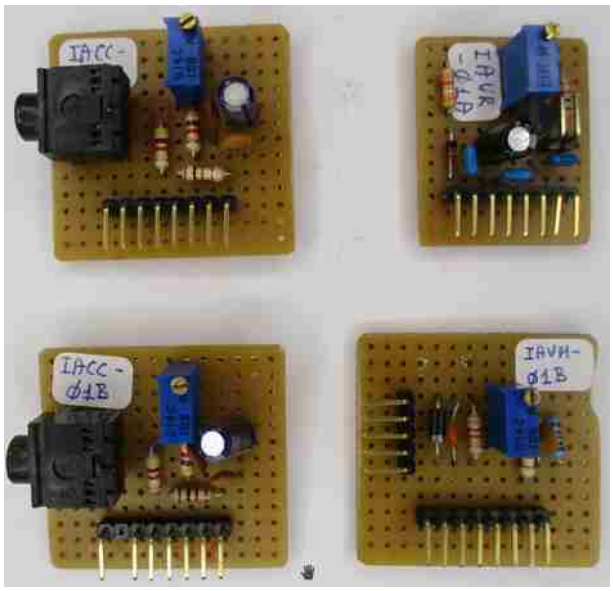
Bedrading. De rest van de busprint is bedrading. Als laatste moet de print gecontroleerd worden op ongewenste kortsluitingen. Ook belangrijk is de controle dat de ene module geen kortsluiting met een ander module kan maken.









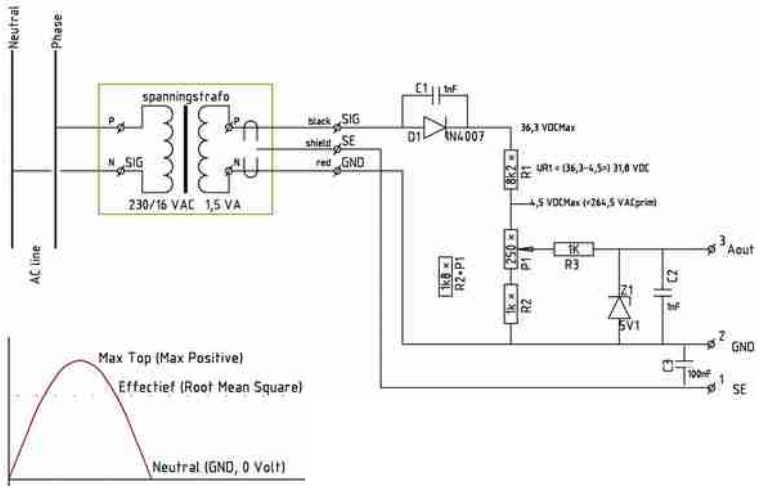


IA2109EM, Werking Energie-Monitor

Spanning-meetingang. Voor het meten danwel monitoren van de netspanning wordt voorzetschakeling IAVM-01B gebruikt aangesloten op A1. Zie de eerder behandelde voltmonitor voor keuze van de juiste spannings-deler-weerstanden.

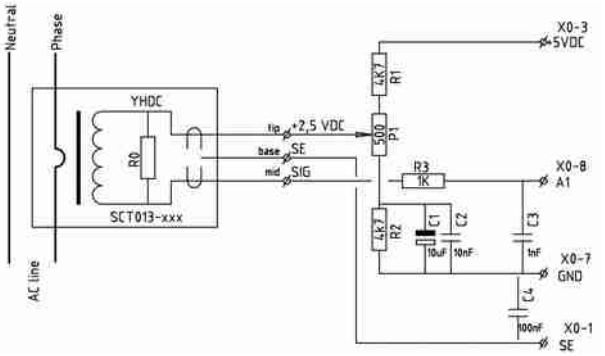
Om voor de 'spanningsloep' de grootst mogelijke ingangsgevoeligheid te realiseren, is het meetsignaal een enkelzijdig gelijkgerichte gelijkspanning, die onafhankelijk is van Arduino's Vcc.

Merk op dat met een dubbelzijdig gelijkgericht spanningssignaal het begin van een complete periode anders niet herleid kan worden.



Stroom-meetingang. Voor het meten van de stroom wordt een vereenvoudigde voorzetschakeling IACC -01B gebruikt van de eerder behandelde stroommeter. Als opnemers worden twee ongewijzigde SCT013/030's gebruikt, aangesloten op A2 en op A3.

De benodigde 2,5 VDC voor de ***voorinstelling (bias offset)*** van de stroom-nullijn wordt van Arduino's 5V afgenomen. Bij een daling in Vcc wijzigt daardoor ook de ADC-nullijn-instelling (ca. 512) automatisch mee. De grootte van het meetsignaal zelf is afhankelijk van de stroomopnemer, en dus onafhankelijk van Arduino's Vcc.

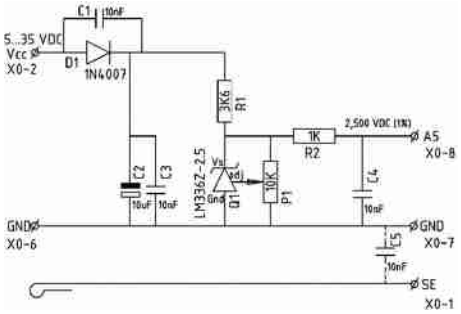


Met C1 wordt de voedingsspanning ontkoppeld en met C2 HF-ontstoord. Eventuele spikes op de signaalleiding zelf worden met R3, C3 weggewerkt. De weerstands- en condensatorwaarde (R3,C3) hiervan moeten dezelfde zijn als voor de spanningsmeting, zodat een eventuele tijdvertraging voor beide signalen exact hetzelfde is.

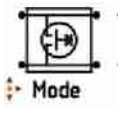
De Energie Monitor wordt voorzien van twee stroommeet-ingangen, zodat zowel de net-aansluiting als ook de zonnestroom-installatie gelijktijdig gemonitord kunnen worden.

Analoge Referentie. Voor een grotere stabiliteit en meetnauwkeurigheid wordt gebruik gemaakt van een externe analoge referentie-spanning van 2,500 VDC van module IAVR-01A aangesloten op analoge ingang A4.

Let op; de functie `analogReference()` wordt ***niet*** gebruikt; de Arduino functioneert inwendig dus met de normale eigen standaard ARef.
Bij onregelmatige meetwaarden kan het plaatsen van een HF-ontstoorcondensator over Aref en GND een oplossing zijn.



Mode- en Reset-knop. Hoewel de busprint de beschikking heeft over de negen Keypad-drukknoppen, wordt voor de Energie-Monitor uitsluitend en alleen de Mode- drukknop gebruikt. Een energiemonitor wordt geplaatst voor uw en andermans veiligheid. Omdat de functie het bewaken is van de maximaal toelaatbare waarden, is een 'on-the-fly' wijzigen van deze instellingen absoluut niet toegestaan!



Met de Mode-knop kunnen de diverse instellingen en huidige meetwaarden worden ingezien. De energie-monitor onthoudt de laagste en hoogste gemeten waarden. Wanneer de gebruiker met de Mode-knop de Auto Reset-mode op het scherm laat staan, zullen enige tijd later alle geregistreerde max- en mimaal-registraties gereset worden.

Testen IA2109EM. Controleer alle printbanen met een loep. Nadat busprint IAAM-01 visueel gecontroleerd is, wordt deze aangesloten op een externe voeding. Dit kan zijn met een adapter via de DC-jackplug X2, of met (backup) batterij via X3 met een spanning groter dan 6 VDC.

Externe voeding VIN. Controleer met een universeelmeter met massa aan GND, dat de externe voedingsspaning VIN alleen op de daarvoor bedoelde aansluitingen gemeten KAN worden. Controleer of tussen X1-17 en X1-18 de voedingsspanning aanwezig is, en dat X1-18 inderdaad +VDC is.

IA2109LCD. Shield IA2109 kan voorzien zijn van een transistor voor sturing van de achtergrond-verlichting. Omdat de energiemonitor niet continu geraadpleegd wordt is continue achtergrondverlichting ook niet nodig, en kan op X6 met een jumper deze verbonden worden met A5. Met de jumper direct naar 5 VDC blijft de verlichting continue branden.

Wijzig in sketch IA2109LCD-mk1 (”Hello World”) `BackLed=A5` , en upload de sketch naar het shield, en plaats deze op X1. Aangesloten op een externe adapter moet het display nu knipperend actief worden. Gebruik de IA2109-potmeters om de weergave optimaal af te regelen.

```
#include <LiquidCrystal.h>
const int rs=12, en=11, d4=8, d5=7, d6=5,
d7=4;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

int BackLed = A5;

void setup()
  { lcd.begin(16, 2);
    lcd.print("hello, world!");
    pinMode(BackLed, OUTPUT);
  }
void loop()
  { digitalWrite(BackLed, HIGH);
    lcd.noDisplay();
    delay(500);
    lcd.display();
    delay(500);
    digitalWrite(BackLed, LOW);
    delay(1000);
  }
```

Keypad. De Energie Monitor gebruikt van het keypad alleen de ‘Mode’- en ‘Reset’-knop.

Omdat de analoge busprint IAAM-01 ook voor andere apparaten gebruikt kan worden, is het wel zo zinvol om het keypad op de juiste werking te testen.

Plaats de hiernaast weergegeven sketch IA2109AK mk5 in het shield, en test het keypad. Merk op dat de toetsvolgorde een andere is dan die van het 3x3-keypad.

Testen is zinvol. In dit geval bleek de Reset-drukknop een contact-probleem te hebben, zodat deze naar aanleiding van deze test vervangen is.

```
#include <LiquidCrystal.h>
const int rs=12, en=11, d4=8, d5=7, d6=5, d7=4;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

int BackLed = A5;
int keyVal;

void setup()
  {
    digitalWrite(BackLed, HIGH);
    lcd.begin(16, 2);
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("  IA2109AK mk5");
    lcd.setCursor(0,1);
    lcd.print("   Keypad 1x9 ");
    delay(2000);
  }
void loop()
  {
    keyVal = analogRead(0);
    lcd.setCursor(0,0);
    lcd.print("  Press a key     ");
    lcd.setCursor(2,1);
    if (keyVal < 90)
      { lcd.print("No key pressed ");}
      else
      if (keyVal < 110)
        { lcd.print ("OK");}
        else
          if (keyVal < 210)
            { lcd.print ("Set");}
            else
              if (keyVal < 310)
                { lcd.print ("Down");}
                else
                  if (keyVal < 410)
                    { lcd.print ("Up");}
                    else
                      if (keyVal < 510)
                        { lcd.print ("Right");}
                        else
                          if (keyVal < 610)
                            { lcd.print ("Left");}
                            else
                              if (keyVal < 710)
                                { lcd.print ("Reset");}
                                else
                                  if (keyVal < 815)
                                    { lcd.print ("Mode");}
                                    else
                                      if (keyVal < 920)
                                        { lcd.print ("Esc");}
    // End nested If-Then-Else
    if (keyVal > 100)
      { lcd.print(" (");
        lcd.print(keyVal);
        lcd.print(")           ");
      }
  }
```

Spanningsreferentie. Plaats een op 2,500 VDC afgeregelde IAVR-01A-module op female pinheader A4.

Plaats de hiernaast weergegeven sketch in het IA2109-shield. Ontkoppel de usb-kabel, en sluit de externe voedingsadapter aan.

Aan de hand van de gemeten referentiespanning van 2,500 VDC wordt nu Arduino's voedingsspanning (en de interne ARef-spanning) herleid.

In dit voorbeeld wordt de monitor gevoed vanuit een 9V blok-batterij, en is de externe referentiespanning op de module afgeregeld op exact 2,500 Volt.

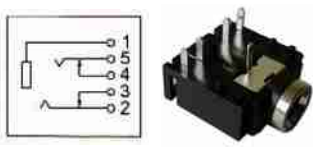
De referentiespanning wordt aangeboden op ingang A4, en dit geeft een ADC-waarde van 508, terwijl dit 512 moet zijn. Dit betekent dat de Arduino's Vcc iets hoger dan gebruikelijk is.

Nacalculatie leert dat Vcc 5,034 VDC is, hetgeen dus ook de Aref voor alle analoge ingangen is.

Conclusie; een afwijking in de voedingsspanning en dus van Aref wordt keurig opgemerkt.

```
// IA2109AR_mk1_AnalogReference


#include <LiquidCrystal.h>
const int rs=12, en=11, d4=8, d5=7, d6=5,
d7=4;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

int AnalogValue = 512;
int BackLed = A5;
const int VRefSensor = A4;
float VRefVcc = 5.0;
float VRefSense = VRefVcc / 1024;
// Default VrefSense = 5/1024 = 0,0048828
V/step

void setup()
  { //analogReference(DEFAULT);
    digitalWrite(BackLed, HIGH);
    pinMode(VRefSensor,INPUT);
    lcd.begin(16,2);
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("  IA2109AR mk1");
    lcd.setCursor(0,1);
    lcd.print("Analog Reference... ");
    delay(2000);
  }

void loop()
  { AnalogValue = analogRead(VRefSensor);
    VRefSense = 2.500 / AnalogValue;
    VRefVcc = VRefSense * 1023;
    lcd.setCursor(0,1);
    lcd.print("RSense :");
    lcd.print(VRefSense,6);
    lcd.print("                ");
    lcd.setCursor(0,0);
    lcd.print("ADCzero:");
    lcd.print(AnalogValue);
    lcd.print("           ");
    delay(4000);
    lcd.setCursor(0,0);
    lcd.print("VRef:");
    lcd.print((AnalogValue*VRefSense),3);
    lcd.print(" V           ");
    lcd.setCursor(0,1);
    lcd.print("Vcc :");
    lcd.print(VRefVcc,3);
    lcd.print(" V             ");
    delay(4000);
  }
```

Stroom-meetingang2. Plaats een stroommeet-module IACC-01B in female pinheader A3. Voor het aansluiten van de stroommeetspoel SCT013 wordt een 'plat type' 3,5 mm audio jack pcb chassisdeel gebruikt, met ingebouwde schakelaars. De print is zodanig ontworpen, dat wanneer geen stroomsensor aangesloten is, de signaalleiding niet open blijft hangen, maar naar de voorspanningsdeler wordt geschakeld.



Laad sketch `IA2109AM mk1 Ampere-Meter` in het shield, met de juiste instelling voor `AmpSensor` en `BackLed`. Wanneer geen meetspoel aangesloten is, moet de voorinstelling `Analog Zero` aangepast worden naar 512. Na enige tijd zal het display nul ampere aangeven.

```
// IA2109AM mk1 Ampere Meter Harm J Seef, The I
//
// 2022 Feb 02   H2S     - AC Current Ampere m
//
// ------------------------------------------

#include <LiquidCrystal.h>
const int rs=12, en=11, d4=8, d5=7, d6=5, d7=4;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

const int AmpSensor = A3;

int BackLed = A5;
```

Maak de net-draad spanningsloos, plaats dan pas de stroommeetspoel, en sluit deze aan op de spanningsloze EnergieMonitor. Nadat de EnergieMonitor opgestart is, kan een verbuiker worden ingeschakeld en kan diens stroom gemeten worden. Alleen de grootte van de stroom wordt gemeten, niet de richting. Het display geeft de theoretische waarde met de initiële gevoeligheid weer. De afregeling wordt later uitgevoerd.

Stroom-meetingang1. Plaats een stroommeet-module IACC-01B in female pinheader A2. De werkwijze is dezelfde als bij de vorige stroom-meetingang.

Spanning-monitoringang. Plaats spanningmeetmodule IAVM-01B in female pinheader A1, sluit de meetkabel aan, en laad sketch `IA2109VM mk1 VoltMonitor` in het shield. Het display moet nu 0,00 VAC meten. Na het inschakelen van de meettransformator zal het display een netspanning laten zien gebaseerd op de initiële gevoeligheid.

IA2109EM mk2 EnergieMonitor. In de vorige paragrafen is de EnergieMonitor-hardware opgebouwd en zijn de diverse functionele deelschakelingen getest. De eerste sketch `IA2109EM mk1 EnergyMonitor` is vanwege tegenvallende resultaten en ervaringen komen te vervallen, en vervangen door IA2109EM mk2.

```
IA2109EM_mk2_EnergyMonitor
// IA2109EM_mk2_EnergyMonitor
```

Bedrijfs-parameters. Voor het correct meten en functioneren van de monitor, is het belangrijk dat de juiste werk-condities zijn ingevoerd. Grid-Voltage in Volt, -Deviatie in procent, en -Frequentie in Herz, spreken voor zich. In bepaalde landen mag de onderspanningsafwijking groter zijn dan de bovenspanningsafwijking.

```
// ---Operating parameters-----
float GridVoltRms = 230;
float GridVoltDevPos = 10;
float GridVoltDevNeg = 10;
float GridFrequency = 50.0;
float GridMaxCurrent = 35;
float SolarMaxCurrent = 35;
float CabinetMaxCurrent = 40;
```

Zeer belangrijk voor ieders veiligheid zijn de diverse toelaatbare maximaal-stromen (limieten). Bij `GridMaxCurrent` dient de waarde van de net-hoofdzekering te worden aangegeven. Bij opname van netstroom en/of teruglevering van zonnestroom mag deze waarde niet overschreden worden.

Bij `SolarMaxCurrent` dient de waarde van de maximaal toelaatbaar op te wekken zonnestroom te worden opgegeven. LET OP; wanneer maximaal zonnestroom wordt opgewekt, en niets daarvan wordt verbruikt of teruggeregeld, dan mag `SolarMaxCurrent` niet groter zijn dan de nethoofdzekerings-waarde!

CabinetMaxCurrent. Schakel-en-verdeelinrichtingen en hun componenten (-zoals zekering-automaten, schakelaars, bedrading, rails, enz.-) zijn ontworpen, ontwikkeld en getest vanuit een 'één-voeding-één-stroomrichting-gebruik'; een tweede voeding en/of teruglevering is nog steeds geen ontwerp-voorwaarde (al wordt inmiddels wel gewerkt aan het aanpassen van IEC61439). De meeste hoofdschakelaars, aardlek-automaten, installatie-automaten, enz. voor woningen kunnen een maximaalstroom verwerken van 16, 25, 32 of 40 Ampere. Zelfs met een volgens NEN1010 aangelegde schakel- en verdeel-inrichting ('meterkast'), kan door de 'groen-elektrificering' van het huishouden met de E-auto, E-fiets, E-warmtepomp, E-verwarming, E-koken, E-boiler, Vaatwasser, Wasmachine, Droger, enz., enz., de zogeheten **gelijktijdigheidsfactor *(Rated Diversity Factor)*** voor de **hoofdrail** te groot worden, en kunnen de te verdelen stromen de ontwerpstroom gemakkelijk vèr overschrijden.

Stel, een standaard max. 40 ampere huis-verdeelinstallatie wordt gevoed met een netstroom van 35 ampere (een zeer gebruikelijke situatie), en daarop wordt een PV-installatie aangesloten van twee eindgroepen van 16 ampere. Bij een gelijktijdig gebruik van alle zware verbruikers, -of kortsluiting-, kan de gezamelijke stroom door de verdeelinstallatie dan 67 ampere zijn!! De maximaal toelaatbare component-ontwerpstroom `CabinetMaxCurrent` van 40 ampere wordt dan ruimschoots overschreden, met alle kans op oververhitting, overslag en brand!


Toegepaste bekabeling was te dun waardoor de bekabeling te heet werd.

Een juist gedimensioneerde en correct geïnstalleerde **hoofdrail** is bij gebruik van zonnepanelen en/of thuisaccu's fataal belangrijk! Inmiddels berucht is de temperatuurverhoging in de meterkast veroorzaakt door teruglevering bij zonnige dagen. De functie van de `CabinetMaxCurrent`-parameter is hiermee verklaard.

Seriële communicatie. Onderdeeel van de sketch is een optie om de diverse meetwaarden seriëel te melden naar een PC of controller, zodat deze in een logbestand kunnen worden geplaatst, of worden gebruikt voor het regelen en besturen van diverse gebruikers en energieopwekkers. Met `SerialMonitor` kan deze optie worden in- of uitgeschakeld, en bij `SerialInterval` kan in minuten de tijdsinterval tussen de meldingen worden aangegeven. Bij een modulair gebalanceerde PV-installatie is eenmaal per 5 minuten een gebruikelijke instelling. `SerialTimer` wordt gebruikt voor het detecteren van het juiste meldmoment.

```
bool SerialMonitor = true;
int SerialInterval = 5;
unsigned long SerialTimer = millis();
bool AlarmOnError = true;
```

Wanneer bepaalde bedrijfslimieten overschreden wordt, wordt de gebruiker met een 'fout-alarm' daarop geattendeerd. Met `AlarmOnError = false` wordt deze alarmering uitgeschakeld. Dit is vooral handig tijdens het inregelen of bij onderhoud van de EnergieMonitor.

I/O-parameters. Vervolgens worden I/O parameters vastgelegd. Gebruikelijk is om de Input- en Output-parameters als constanten vast te leggen, maar dat hoeft niet! Eén stroomsensor is aangesloten op A2 voor meten van de netstroom, en een andere identieke sensor aangesloten op A3 voor de zonnestroom. In `Setup()` wordt alleen 'een AmpSensor aangesloten'.

```
pinMode(AmpSensor,INPUT);                    // current sensor
```

```
//---I/O parameters-------------
const int KeyPad = A0;
const int VoltSensor = A1;
// const int GridAmpSensor = A2;
// const int SolarAmpsensor = A3;
int AmpSensor = A2;
const int VRefSensor = A4;
const int BackLed = A5;
const int LedRed = 10;
const int LedOrange = 9;
const int LedGreen = 6;
```

De sketch is geprogrammeerd voor het stroommeten van één sensor. Door softwarematig omschakelen van variabele `AmpSensor` wordt eerst de ene (A2) en dan de andere (A3) stroom gemeten.

```
if (AmpSensor == A2)
  { AmpSensor = A3; }
  else
  { AmpSensor = A2; }
```

Display-verlichting. Direct na het inschakelen wordt de achtergrondverlichting `BackStatus` ingeschakeld, en laat de EnergieMonitor de belangrijkste bedrijfsinstellingen zien, die de ingestelde `InitShowDelay`-tijd worden weergegeven. Wanneer de Mode-toets gedurende de ingestelde `BackOnTime` -tijd niet wordt gebruikt, zal de achtergrondverlichting worden uitgeschakeld, en zo weer een beetje energie besparen.

```
//---Auxiliary vars-------------
int InitShowDelay = 500; //2000;
int KeyVal = 0;
bool BackStatus = true;
int long BackOnTime = 60;
int long BackStartTime = 0;
bool KeyState = false;
bool KeyPrev = false;
```

Mode-knop. De Mode-knop heeft een dubbelfunctie. De eerste maal dat de knop bediend wordt, zal de achtergrond-verlichting worden ingeschakeld, zodat het display goed af te lezen is. Pas wanneer de achtergrondverlichting ingeschakeld is, is de knop daadwerkelijk als Mode-schakelaar te gebruiken. Houdt de knop-ingedrukt totdat de volgende mode verschijnt, en laat dan de knop los. Bij een storing kan de Mode-knop ook gebruikt worden als Storing Reset.

Initiële gevoeligheid. De EnergieMonitor kan de meetgevoeligheid indien nodig zelf aanpassen naar temperatuurverloop, veroudering, enz. Om dit mogelijk te maken wordt het programma eerst gestart met de meest waarschijnlijke gevoeligheids-instellingen voor de netspanning `VoltSense` , -stroom `CurrSense` , en externe referentiespanning `VRefSense`.

`GridCurrentRms` is de gemeten netstroom, en `SolarCurrentRms` is de gemeten zonnestroom.

```
float VRefVcc = 5.0;
float VRefSense = VRefVcc / 1024;

// float VoltSense = 0.2868764;


float VoltSense = 0.455;
float VoltSenseCalibCorr = VoltSense / 0.2868764;

float CurrSense = 0.1463415;

float CurrRms = 0;
float GridCurrentRms = 0.0;
float SolarCurrentRms = 0.0;
```

Meetproces. Voor een modulair geregelde zonnestroom-installatie, is de vermogensmeting in Watt het belangrijkste; hoeveel energie wordt er opgewekt, wat is het huidig verbruik, hoeveel net-energie wordt er opgenomen, danwel aan het net teruggeleverd? De vermogensmeting van IA2109WM mk2 is daar een goede basis voor.

Voor de metingen zelf wordt één functie gebruikt die zowel de effectieve spanning als de effectieve stroom meet. Bij een eerste meetcyclus wordt de netstroom gemeten, en alle netstroom-variabelen herleid. Bij de volgende meetcyclus wordt de solarstroom gemeten, en alle solar-variabelen herleid.

```
if (AmpSensor == A2)
  { AmpSensor = A3; }
  else
  { AmpSensor = A2; }
```

```
const int VoltSensor = A1;
// const int GridAmpSensor = A2;
// const int SolarAmpsensor = A3;
int AmpSensor = A2;


//---Global parameters----------------------
int AdcVoltMax = 0;
float GridVoltUpper = GridVoltRms +
    ((GridVoltRms/100) * GridVoltDevPos);
float GridVoltLower = GridVoltRms -
    ((GridVoltRms/100) * GridVoltDevNeg);
float GridVoltHighest = GridVoltLower;
float GridVoltLowest = GridVoltUpper;

int AdcAmpMax = 0;
float GridCurrentHighest = 0;
float GridCurrentLowest = 0;
float SolarCurrentHighest = 0;
float SolarCurrentLowest = 0;

float RealPow = 0;
int   GridApparantPower = 0;
int   GridRealPower = 0;
float GridPowerFactor = 1.0;

int   SolarApparantPower = 0;
int   SolarRealPower = 0;
float SolarPowerFactor = 1.0;

float CabinetCurrent = 0;
float CabinetCurrentHighest = 0;
float CabinetCurrentLowest = 0;
```

Wanneer eventuele gemeten waarden te hoog of te laag is, moeten ze worden vastgelegd, of moet er ingegrepen worden. Hetzelfde geldt voor de bewaking van de maximaal-stromen.

Voor het monitoren van de installatie-prestaties zijn daarvoor de nodige extra parameters aangemaakt, die bij storing en onderhoud geraadpleegd kunnen worden.

Errorprocedure. Het monitoren van de diverse grenswaarden en limieten is weinig zinvol wanneer er geen foutmelding verschijnt. Variabele `ErrorCode` wordt gebruikt om overschrijdingen of fouten te melden.

```
int   ErrorCode = 0;
```

```
//  0 = no error
//  1 = No Voltage Reference Module found
//  2 = Auto adjusting sensor sensitivity
//  3 = Vcc too low
//  4 = Volt ADC overload
//  5 = Voltage below lower limit
//  6 = Voltage above upper limits
//  7= Grid ADC overload
//  8 = Solar ADC overload
//  9 = Grid current max!
// 10 = Grid sensor overload
// 11 = Solar current max!
// 12 = Solar sensor overload
// 13 = Cabinaet overload
// 14 = Solar Switched off
```

Indien nodig wordt in een speciale routine ErrorProc de juiste actie uitgevoerd.

DisplayMode. De EnergieMonitor bewaakt diverse kritische waarden, ofwel limieten. Voor onderhoud -en storingzoeken na een foutmelding-, zijn diverse hoogst en laagst gemeten waarden vastgelegd. Helaas heeft het display slechts 2 regels tekst van 16 karakters. Met de Mode-knop kan de display-weergave worden aangepast voor het uitlezen van de vastgelegde gegevens, en voor het detailmeten van spanning en stromen. De weergave van het display is afhankelijk van de teller DisplayMode.

```
//---Display-------
int DisplayMode = 0;
```

Standaard (`DisplayMode = 0`) laat de Energiemonitor alleen het huidige energiegebruik zien.

```
//  0 = Show solar and grid power (Monitor mode)
//  1 = Grid monitor mode; U RMS,I RMS
//  2 = Solar monitor mode; U RMS, I RMS
//  3 = Lowest and highest voltage
//  4 = Show cabinet highest and lowest amp
//  5 = Grid lowest and highest current
//  6 = Solar lowest and highest current
//  7 = Grid power and power factor
//  8 = Solar power and power factor
//  9 = Solar control (trafic light)
// 10 = Analog Reference mode
// 11 = Vcc (Aref) and ADC sensitivity
// 12 = Grid current sensitivity
// 13 = Solar current sensitivity
// 14 = VAC Calibration mode
// 15 = IAC Grid Calibration mode
// 16 = IAC Solar Calibration mode
// 17 = Reset Lowest/Highest records
```

Wanneer dat nodig is kan met de Mode-knop een detail-weergave worden gekozen. Dit is vooral handig bij het instellen, afregelen of foutzoeken met de EnergieMonitor.

De Mode-knop zelf bedient een `DisplayMode`-teller, zodat het scherm de gewenste informatie op het display geeft. Wanneer de gebruiker de monitor in een andere detail-modus laat staan, zal na verloop van tijd teruggeschakeld worden naar de normale Energie-monitor-modus.

```
void loop()
  { //---Back Light Control----------
    if (BackStatus == true)
      { if ((BackStartTime + BackOnTime) <
          millis())
        { BackStatus = false;
          DisplayMode = 0;
          digitalWrite(BackLed,LOW);
        }
      }
```

Keypad Control. Het eerste deel van het programma omvat het aansturen van de achtergrondverlichting, en het afvangen en verwerken van de Mode-knop-bediening. Voor de Keypad-bediening wordt dus een analoge ingang gebruikt.

```
//---Keypad Control-----------
KeyVal = analogRead(KeyPad);
if (KeyVal < 710)
  { KeyState = LOW; }
  else
  if (KeyVal < 815)
    { KeyState = HIGH;
    }
if ((KeyState == HIGH) &&
    (BackStatus == false))
  { delay(500);
    BackStatus = true;
    digitalWrite(BackLed,HIGH);
    BackStartTime = millis();
    KeyState = LOW;
  }
if ((KeyState == HIGH) &&
    (KeyPrev == false))
  { BackStartTime = millis();
    DisplayMode++;
    if (DisplayMode > 17)
      { DisplayMode = 0; }
  }
if((KeyState == LOW) &&
   (KeyPrev == true))
  { KeyPrev = false;
    ErrorCode = 0;
  }
```

Bij een foutmelding kan met de Mode-knop de Errorcode gereset worden.

Uiteraard zal wanneer de fout niet verholpen wordt, de fout actief blijven.

Wanneer in DisplayMode 17 de Mode-knop niet gebruikt wordt om door te schakelen, zullen even later diverse registratie-variabelen gereset worden.

```
if ((ErrorCode > 0) &&
    (BackStatus == true) &&
    (KeyState == HIGH))
 { lcd.setCursor(0,0);
   lcd.print(" Resetting...      ");
   delay(100);
   ErrorCode = 0;
   KeyState = LOW;
 }
if (DisplayMode == 17)
  { if ((BackStartTime + BackOnTime - 200) <
      millis())
    { GridVoltHighest = GridVoltLower;
      GridVoltLowest = GridVoltUpper;
      GridCurrentHighest = 0.0;
      GridCurrentLowest = 0.0;
      SolarCurrentHighest = 0.0;
      SolarCurrentLowest = 0.0;
      ErrorCode = 0;
      DisplayMode = 0;
      //Serial.println("AutoReset");
    }
  }
```

Sensor-Kalibratie. De EnergieMonitor is voorzien van een automatische kalibratie en temperatuur-compensatie voor de spanning- en stroom-meetgevoeligheid. Voor een grotere meetnauwkerigheid maakt de energiemonitor gebruik van spannings-referentie-module IAVR-01A aangesloten op A4. Sluit de monitor aan op de te gebruiken externe voedingsspanning. Bij het testen van de referentiemodule is de referentiespanning al afgeregeld op 2,500 VDC.

Na het afkoppelen van de USB-kabel en inschakelen van de externe voeding, zal de EnergieMonitor zichzelf eerst instellen (-initïeel-) op de gewenste meetgevoeligheden voor spanning, stroom en referentiespanning per ADC-stap.

```
float VRefVcc = 5.0;
float VRefSense = VRefVcc / 1024;
```

Analoge Referentie De spanningsreferentie van 2,500 VDC wordt doorgegeven aan analoge ingang `VRefSensor`. Bij een Vcc van 5 VDC zal deze een waarde teruggeven van 512 ADC-stappen, ofwel een gemeten referentiespanning van 2,500 VDC. Wanneer Vcc verloopt, verloopt via ARef ook de ADC-waarde. Zodra de gemeten referentiespanning beneden de 2,490 VDC -of groter wordt dan 2,510 VDC- zal het programma dit als een error zien,

```
//---Sensitivity & Reference----------
AnalogValue = analogRead(VRefSensor);
if ((AnalogValue < 472) ||
    (AnalogValue > 552))
  { ErrorCode = 1; }
  else
  if (((AnalogValue * VRefSense) > 2.510) ||
      ((AnalogValue * VRefSense) < 2.490))
    { ErrorCode = 2; }
```

LET OP: De automatische kalibratie kan alleen correct werken wanneer de USB-kabel losgekoppeld is, en een externe voeding voor de EnergieMonitor wordt gebruikt.

Wanneer het programma een spanningsreferentie-fout ziet, voert de energiemonitor een zelfkalibratie uit.

De referentiespanning-gevoeligheid `VRefSense` wordt aangepast naar de daadwerkelijk gemeten ADC-waarde. Daarmee kan Vcc worden herleid en is ook Arduino's interne ARef vast te stellen. Wanneer Arduino's voedingsspanning `VRefVcc` te laag wordt verschijnt er een foutmelding.

De netspannings-meetgevoeligheid `VoltSense` en de stroom-meetgevoeligheid `CurrSense` worden met de daadwerkelijke referentiewaarden bijgeregeld. VoltSense moet vanwege de RMS-sampling-methode met een `VoltSenseCalibCorr`-factor worden gecorrigeerd.

```
if (ErrorCode == 2)
  { // 512 = 2.500 volt
    // 534 (=2.500 Volt)
    // VRefSense = 2.500 / 534 = 0.0046816
    VRefSense = 2.500 / AnalogValue;
    VRefVcc = VRefSense * 1023;
    // 0.0046816 * 1023 = 4.7893258 VDC
    if (VRefVcc < 4.500)
      { ErrorCode = 3; }

    // VoltSense = 0.2868764;

    VoltSense = (264.5 /
      ((4.5/VRefVcc) * 1024)) ;
    VoltSense = VoltSense * VoltSenseCalibCorr;

    // CurrSense = 0.1463415;
    // CurrSense = 30.0 /
    //      (717 - 512 );
    // CurrSense = 30/(703-512)=0.1195219
    CurrSense = 30.0 /
      (((3.5/VRefVcc) * 1024) - 512);
  }
```

Kalibreren spanningsmonitor. `DisplayMode 14` is de VAC-kalibratie-modus.

```
if (DisplayMode == 14)
  { lcd.setCursor(0,0);
    lcd.print("Volt calibration");
    lcd.setCursor(0,1);
    lcd.print(GridVoltRms) ;
    lcd.print(" VAC RMS ");
    lcd.print("          ");
    ErrorCode = 0;
    delay(1);
  }
```

Als het goed is, is de meetvoorzet afgeregeld met behulp van IA2109VM mk1, bij een berekende gewenste meet-gevoeligheid van 0,287 Volt/step.

```
float VoltSense = 0.2868764;
```

De RMS-sample-meetmethode heeft een behoorlijk negatieve invloed op de meetgevoeligheid, en vanwege de verschillende manieren van programmeren laat deze zich niet berekenen. Met onderstaande procedure kan de gewenste initiële gevoeligheid eenvoudig herleid worden.

1. Stel in de sketch de initiële sensorgevoeligheid in op 0,5, en upload de sketch met `DisplayMode= 14`.

```
float VoltSense = 0,5;
```

2. Draai instelpotmer P2 helemaal rechtsom totdat de spanning niet lager wordt. Noteer deze waarde; bijvoorbeeld 204 VAC.
3. Draai instelpotmeter P2 helemaal linksom totdat de spanning niet hoger wordt. Noteer deze waarde; Bijvoorbeeld 304 VAC.
4. Regel met instelpotmeter P2 de spanning af op ongeveer het gemiddelde van de twee metingen. (204 + 304)/2 = ca 254 VAC. De loper van de instelpotmeter zal nu ongeveer in het midden staan.
5. Gebruik een gecertificeerde spanningsmeter ter controle van de netspanning, en vergelijk deze met de gemeten waarde. Is de monitorwaarde te hoog, dan moet `VoltSense` verkleind worden. Is ze te laag dan moet `VoltSense` verhoogd worden.

Wanneer `VoltSense` eenmaal achterhaald is (± 1 Volt), hoeft deze nooit meer gewijzigd te worden. Alleen P2 wordt nu nog gebruikt voor de afregeling, en eventuele latere correctieregeling.

```
float VoltSense = 0.455;
```

Met potmeter P2 op spanningsmeetnodule IAVM-01B wordt de spanningsweergave nu zo exact mogelijk op de juiste spanning ingesteld (± 0,5 V is vrij normaal). Merk op dat tijdens het afregelen op een zonnige dag de netspanning al snel een 18 Volt hoger is dan de norm van 230 VAC.

In principe hoeft dit slechts éénmaal afgeregeld te worden. Eénmaal goed afgesteld, zal in de toekomst de automatische kalibratie de meetgevoeligheid per ADC-stap indien nodig automatisch aanpassen.

```
float VoltSense = 0.455;
float VoltSenseCalibCorr = VoltSense / 0.2868764;
```

Meet-procedure. De spanning-, stroom-meting is elkaar geschoven en als één vermogens-meting/functie/procedure vastgelegd. Eerst wordt de spanning gemeten, direct daarop één stroom (netstroom of zonnestroom), en aan de hand van de gemeten waarden wordt de energie herleid.

```
//---Measuring Real Power RMS--
void MeasureRealPower(void)
  { int AverageLoop = 0;
    int MeaSamples = 0;
    int AdcAmpPrev = 0;
    int AdcVolt = 0;
    int AdcAmp = 0;
    float MeaVolt = 0;
    float MeaCumVolt = 0;
    float MeaVoltRms = 0;
    float MeaAmp = 0;
    float MeaCumAmp = 0;
    float MeaRealPow = 0;
    float MeaCumRealPow = 0;
    float CumVoltRms = 0;
    float AmpRms = 0;
    float CumAmpRms = 0;
    //float RealPow = 0;
    float CumRealPow = 0;
    float AppPow = 0;
    float CumAppPow = 0;
    float PowerFactor  = 0;
    //float GridVoltRms = 0;
    AverageLoop = 0;
    while(true)
```

Voor een correcte werking worden eerst de nodige lokale variabelen aangemaakt.

De werking van de vermogensmeting is al eerder behandeld.

```
while(true)
  { MeaSamples = 0;
    AdcVoltMax = 0;
    AdcAmpMax = 0;
    AdcAmpPrev = analogRead(AmpSensor);
    AdcVolt = analogRead(VoltSensor);
    AdcAmp = analogRead(AmpSensor);
    if (AdcVolt > 0)
      { while(true)
          { if (AdcVolt > AdcVoltMax)
              { AdcVoltMax = AdcVolt; }
            MeaVolt = AdcVolt * VoltSense;
            MeaCumVolt += (MeaVolt * MeaVolt);
            if (AdcAmp > AdcAmpMax)
              { AdcAmpMax = AdcVolt; }
            MeaAmp = (AdcAmpPrev + AdcAmp) / 2;
            MeaAmp = (AdcAmp -512) * CurrSense;
            MeaCumAmp += (MeaAmp * MeaAmp);
            MeaRealPow = MeaVolt * MeaAmp;
            MeaCumRealPow += MeaRealPow;
            MeaSamples ++;
            AdcAmpPrev = AdcAmp;
            AdcVolt = analogRead(VoltSensor);
            AdcAmp = analogRead(AmpSensor);
            if (AdcVolt < 1)
              { break; }
          }
```

Met `AdcVoltMax` en `AdcAmpMax` worden de maximale ADC-waarden genoteerd.

Wanneer voldoende samples gevonden zijn, worden de werkelijke momentele waarden herleid.

```
if (MeaSamples > 40)
  { AverageLoop ++;
    MeaVoltRms =
      sqrt((MeaCumVolt/MeaSamples));
    CumVoltRms += MeaVoltRms;
    AmpRms = sqrt((MeaCumAmp/MeaSamples));
    CumAmpRms += AmpRms;
    RealPow = MeaCumRealPow/MeaSamples;
    CumRealPow += RealPow;
    AppPow = MeaVoltRms * AmpRms;
    CumAppPow += AppPow;
    MeaCumVolt = 0;
    MeaCumAmp = 0;
    MeaCumRealPow = 0;
```

Uitermate belangrijk is het om Arduino's analoge ingangen te beschermen tegen te hoge spanningen. Het meetbereik voor de Arduino is om clipping te voorkomen beperkt tot maximaal 4,5 VDC op de analoge ingang, ofwel tot een ADC-waarde van 920.

De 'Max'-beveiliging kijkt alleen naar de analoge ADC-waarde, zodat een verkeerde sensor-gevoeligheid daar geen invloed op heeft. Alles boven een ADC-waarde van 1000 wordt als een Error vastgelegd. `ErrorCode 4` voor de spanning-ADC en `ErrorCode 7` voor de stroom-ADC.

```
if (AdcVoltMax > 1000)
      { ErrorCode = 4;
        break;
      }
if (AdcAmpMax > 1000)
  { ErrorCode = 7;
    break;
  }
```

Bepalen stroomrichting. Vanwege de halve periodemeting, is de spanning altijd positief. Het momenteel vermogen `MeaRealPow` wordt berekent als het product van de momentele spanning `MeaVolt` en momentele stroom `MeaAmp`.

Wanneer het herleide werkelijk vermogen `RealPow` negatief is, moet de netto netto stroomrichting wel omgekeerd zijn. Door de gemeten AmpRms met -1 te vermenigvuldigen wordt de stroomsensor richtgevoelig gemaakt.

En kan ook de energie en zijn richting herleid worden.

```
if (AverageLoop >= 25)
  { GridVoltRms = CumVoltRms / 25;
    AmpRms = CumAmpRms / 25;
    if ((AmpRms < CurrNulling) &&
        (AmpRms > (CurrNulling * -1.0)))
      { AmpRms = 0; }
    RealPow = CumRealPow / 25;
    if (RealPow < 0)
      { AmpRms = -1.0 * AmpRms; }
    AppPow = MeaVoltRms * AmpRms;
    PowerFactor = RealPow/AppPow;
    if (AmpRms == 0)
      { RealPow = 0;
        AppPow = 0;
        PowerFactor = 1;
      }
```

```
AdcAmpPrev = analogRead(AmpSensor);
AdcVolt = analogRead(VoltSensor);
AdcAmp = analogRead(AmpSensor);
if (AdcVolt > 0)
  { while(true)
      { if (AdcVolt > AdcVoltMax)
          { AdcVoltMax = AdcVolt; }
        MeaVolt = AdcVolt * VoltSense;
        MeaCumVolt += (MeaVolt * MeaVolt);
        if (AdcAmp > AdcAmpMax)
          { AdcAmpMax = AdcVolt; }
        MeaAmp = (AdcAmpPrev + AdcAmp) / 2;
        MeaAmp = (AdcAmp -512) * CurrSense;
        MeaCumAmp += (MeaAmp * MeaAmp);
        MeaRealPow = MeaVolt * MeaAmp;
        MeaCumRealPow += MeaRealPow;
        MeaSamples ++;
        AdcAmpPrev = AdcAmp;
        AdcVolt = analogRead(VoltSensor);
        AdcAmp = analogRead(AmpSensor);
        if (AdcVolt < 1)
          { break; }
```

Herleiden vermogen. Na de meting wordt afhankelijk van de gebruikte stroomsensor op A2 (grid) of A3 (solar), de herleide stroom `AmpRms` togekend als netstroom `GridCurrentRms` of als zonnestroom `SolarCurrentRms`.

Vervolgens worden stromen gecontroleerd of hun limieten.

Pas daarna wordt het werkelijk en schijnbaar vermogen herleid, en de arbeidsfactor.

Ter afsluiting wordt de stroomsensor omgeschakeld.

```
    AmpSensor = A3;
  }
  else
  { if (ErrorCode == 7)
      { ErrorCode = 8; }
    SolarCurrentRms = AmpRms * SolarSensorCalib;
    if (SolarCurrentRms < SolarCurrentLowest)
      { SolarCurrentLowest = SolarCurrentRms; }
    if (SolarCurrentRms > SolarCurrentHighest)
      { SolarCurrentHighest = SolarCurrentRms; }
    if (SolarCurrentRms > SolarMaxCurrent)
      { ErrorCode = 11; }
    if (SolarCurrentRms > 30.0)
      { ErrorCode = 12; }
    SolarApparantPower =
      GridVoltRms * SolarCurrentRms;
    SolarRealPowr = RealPow;
    SolarPowerFactor = 1;
    if ((SolarRealPower != 0) &&
        (SolarApparantPower != 0))
      { SolarPowerFactor = 1.0 *
          SolarRealPower / SolarApparantPower;
      }
    AmpSensor = A2;
  }
pinMode(AmpSensor,INPUT);
```

```
if (AmpSensor == A2)
  { if (ErrorCode == 7)
      { ErrorCode = 7; }
    GridCurrentRms = AmpRms * GridSensorCalib;
    if (GridCurrentRms < GridCurrentLowest)
      { GridCurrentLowest = GridCurrentRms; }
    if (GridCurrentRms > GridCurrentHighest)
      { GridCurrentHighest = GridCurrentRms; }
    if (GridCurrentRms > GridMaxCurrent)
      { ErrorCode = 9; }
    if (GridCurrentRms > 30.0)
      { ErrorCode = 10; }
    GridApparantPower =
      GridVoltRms * GridCurrentRms;
    GridRealPower = RealPow;
    GridPowerFactor = 1;
    if ((GridRealPower != 0) &&
        (GridApparantPower != 0))
      { GridPowerFactor =
          1.0 * GridRealPower/GridApparantPower;
      }
    AmpSensor = A3;
  }
  else
  { if (ErrorCode == 7)
      { ErrorCode = 8; }
```

Spanning- en stroomlimieten. Na de meetprocedure wordt de netspanning vergeleken met de toegestane minimaalwaarde `GridVoltLower` (van 207 VAC) en de minimale overspanning `GridVoltUpper` (van 253 VAC). ErrorCode 5 of 6 betekent dan dat de netspanning- te hoog, danwel te laag is.

De laagst en hoogst gemeten netspanning worden vastgelegd, zodat de oorzaak van een eventuele storing sneller te achterhalen is.

Hetzelfde geldt voor de maximale installatie-stroom.

```
//---Voltage, Current & Power Measurement--------
MeasureRealPower();

if (GridVoltRms < GridVoltLowest)
  { GridVoltLowest = GridVoltRms; }
if (GridVoltRms > GridVoltHighest)
  { GridVoltHighest = GridVoltRms; }

if (GridVoltRms < GridVoltLower)
  { ErrorCode = 5; }
if (GridVoltRms > GridVoltUpper)
  { ErrorCode = 6; }

//---Cabinet-------------------------------------
CabinetCurrent = GridCurrentRms + SolarCurrentRms;
if (CabinetCurrent > CabinetCurrentHighest)
  { CabinetCurrentHighest = CabinetCurrent; }
if (CabinetCurrent < CabinetCurrentLowest)
  { CabinetCurrentLowest = CabinetCurrent; }
if (CabinetCurrent > CabinetMaxCurrent)
  { ErrorCode = 13; }
```

Afregelen netstroommeting. Voor de stroommeting hoeft alleen de initiële nullijn via de meetvoorzet te worden ingesteld; de stroomsensor hoeft niet eens aangesloten te zijn. Selecteer met de Mode-knop de netstroomkalibratie ‘Grid ADC:’ Op het display moet nu `512` worden aangeven. Wanneer dat niet het geval is, kan met de meerslagpotmeter op de meetvoorzet zo dicht mogelijk op 512 worden afgeregeld.

Afregelen zonnestroommeting. Voor de stroommeting hoeft alleen de initiële nullijn via de meetvoorzet te worden ingesteld; de stroomsensor hoeft niet eens aangesloten te zijn. Selecteer met de Mode-knop de netstroomkalibratie ‘Solar ADC:’ Op het display moet nu `512` worden aangeven. Wanneer dat niet het geval is, kan met de meerslagpotmeter op de meetvoorzet zo dicht mogelijk op 512 worden afgeregeld.

Stroomsensor-kalibratie. Ook een stroomsensor voldoet soms niet aan de specificaties. Wanneer de gemeten waarde continu consequent afwijkt van de weergegeven waarde, -of er worden verschillende soorten/typen stroomsensoren gebruikt, kan de weergegeven waarde met een factor xxxSenssorCalib vermenigvuldigt worden.

```
float GridSensorCalib = 1.0 / 1.0;
float SolarSensorCalib = 1.0 / 1.0;
```

Stroommeting nullen. Wanneer er geen netstroom vloeit (-bij aangesloten gesloten stroomopnemer-), kan de stroomtrafo na een meting zelf een minimum ruissignaal produceren. De nul-stroomwaarde op het display in de stroomweergavemodus springt dan heen en weer van bijvoorbeeld -0,05 naar 0 naar 0,05 enzovoort. Met variabele `CurrNulling` kan een ‘nul-drempelwaarde’ worden ingesteld. Bij een meetwaarde kleiner dan de drempelwaarde, wordt de meetwaarde automatisch genuld.

```
float CurrNulling = 0.0;
//float CurrNulling = 0.12;
```

Energy Monitor. Uiteindelijk komen alle meetwaarden samen als het opgenomen/teruggeleverde vermogen van of naar het net, en het huidige opgewekte zonnestroom-vermogen; beide in Watt.

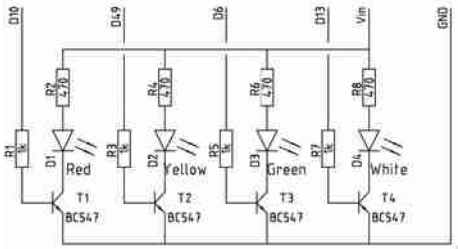
Bij een 32 ampere hoofdzekering mogen beide de (32x230=) 7360 Watt niet overschrijden.

```
//---Diplay modes--------------------------------
lcd.clear();
if (DisplayMode == 0)
  { lcd.setCursor(1,0);
    lcd.print("Usage:");
    dtostrf((GridRealPower + SolarRealPower),
      5,0,CharValue);
    lcd.print(CharValue);
    lcd.print(" W       ");
    lcd.setCursor(0,1);
    lcd.print("G:");
    dtostrf(GridRealPower,5,0,CharValue);
    lcd.print(CharValue);
    lcd.print(" S:");
    dtostrf(SolarRealPower,5,0,CharValue);
    lcd.print(CharValue);
    delay(1);
  }
```

Richting stroomopnemers. Bij de energiemonitor wordt een stroommeetspoel om de grid-fasedraad, en een stroommeetspoel om de solar-fasedraad geplaatst. Het programma is gebaseerd op de aanname dat voeden met netstroom en de solarstroom een positieve waarde is. Wanneer de gemeten waarde negatief blijkt te zijn, moet de stroomspoel omgekeerd worden. Om beschadiging van de meetspoelen te voorkomen, moet eerst de Energiemonitor uitgeschakeld worden en de installatie spanningsloos worden gemaakt! Dan pas kunnen de meetspoelen losgemaakt en ongekeerd worden.

Solar Status. Vanwege de bedrade meetsensoren wordt de energie-monitor in -of direct nabij- de meterkast geplaatst. Om energie besparen gebruiksvriendelijker te maken, is het in de praktijk wel zo handig om op afstand in één oogopslag de huidige 'Solar Status' te kunnen zien. Een 'verkeerslicht'-signalering buiten de meterkast is daarbij de meest handige oplossing. De signalering is dan;

rood	= netstroomvoeding, geen solarvoeding
rood-geel	= netstroomvoeding en te weinig solar-voeding
geel	= netstroomvoeding gelijk aan solarvoeding
groen geel	= solarvoeding groter dan netstroomvoeding
groen	= solarvoeding, en netstroom terugleveren



Voor het aansturen van de signalering wordt gebruik gemaakt van een integere variabele `SolarStatus`, met een waarde van 0 tot en met 5.

```
// Indicator 'trafic light' showing current
// 1 = Red          Grid. No Solar at all
// 2 = Red-Orange   Grid exceeds solar
// 3 = Orange       Solar equals grid
// 4 = Green-Orange Solar exceeds grid
// 5 = Green        Solar surplus to grid
```

Het softwarematig aansturen van de signalering is met `digitalWrite()` de eenvoud zelve.

```
if (SolarStatus == 0)
  { digitalWrite(LedRed,LOW);
    digitalWrite(LedOrange, LOW);
    digitalWrite(LedGreen, LOW);
  }
if (SolarStatus == 1)
  { digitalWrite(LedRed,HIGH);
    digitalWrite(LedOrange, LOW);
    digitalWrite(LedGreen, LOW);
  }
if (SolarStatus == 2)
```

Het vaststellen van de huidige `SolarStatus` is iets gecompliceerder. We maken het onszelf gemakkelijk door eerst de `SolarStatus` naar nul te resetten. Vervolgens wordt in meerdere treden het niveau van `SolarStatus` verhoogd naar de juiste eindwaarde (step up-leveling).

`SolarStatus` kan daardoor ook gebruikt worden voor het aansturen van een modulair gebalanceerde PV-installatie. De 'default'-waarde per bij- en/of af te schakelen solarmodule wordt gesteld op 100 Watt, ofwel de ***zonnestroomdrempel (`SolarThreshold`)*** wordt gesteld op 100 Watt.

```
// int SolarThreshold = 100;
SolarStatus = 0;
if ((RealPowerG > 0) &&
   (RealPowerS <= SolarThreshold))
  { SolarStatus = 1;
  }
if ((RealPowerG > 0) &&
   (RealPowerS > SolarThreshold))
  { SolarStatus = 2;
  }
if ((RealPowerG > 0) &&
   (RealPowerG <
     (RealPowerS - SolarThreshold)))
  { SolarStatus = 3;
  }
```

Een probleem bij elke Aan/Uit- of Modulaire regeling, is het 'pendelen' rond een schakeldrempel. Met een minimum ***zonnestroomdrempel (SolarThreshold)*** van 100 Watt wordt al veel overbodig schakelen voorkomen.

Modulaire regeling. Wanneer terugleveren naar het net niet toegestaan is, kan `SolarStatus` ook gebruikt worden om een modulair gebalanceerde PV-installatie terug te regelen. Zie de eerder behandelde Cascade Counter.

```
if ((RealPowerG < 0) &&
   ((RealPowerS - SolarThreshold) >
   SolarThreshold))

  { SolarStatus = 6;
  }
```

Solar Alarm. Zonnepanelen worden aangeschaft met energiebesparing als doel. Zeer vervelend is dat deze bij een te hoge netspanning uitgeschakeld kunnen worden. Wanneer dat het geval is wordt alarm geslagen.

```
if ((ErrorCode == 6) &&
     (CurrRmsG > CurrRmsS))
  { SolarStatus = 7;
    ErrorCode = 14;
  }
```

Reset Alarm. In de normale bedrijfsmode wordt het vermogensgebruik weergegeven en is de display-verlichting uitgeschakeld. Bij een alarm zal de de display-verlichting gaan knipperen. Met de Mode-knop wordt het alarm bevestigd, en wordt de display-verlichting continu aan gezet. Met een volgende Mode-knop bediening wordt de error gereset. Wanneer de fout niet verholpen is, zal dezelfde foutmelding weer verschijnen.

IA2109EM mk2 EnergyMonitor. Hieronder de vervallen software van versie mk2 (die later vervangen is door mk3).

```
// IA2109EM_mk2_EnergyMonitor, Harm J Seef, The Netherlands
// Proof of concept.  Sketch for energy monitoring 230 VAC solar
// with the IA2109 shield, an analog Voltage Reference,
// and YHDC SCT013/30/1V current transformers.
// Warning; Mount current sensors in right direction!
//
// 2022 may  4  HJS  Designed for the IA2109 shield.
// 2022 aug 25  HJS  Grid current and Grid Power added.
// 2023 feb  9  HJS  Solar current and power added
// 2023 mrt 15  HJS  SolarStatus and Errorproc
// 2023 dec 09  HJS  Sketch redesigned.
//
// ================================================================================================

#include <LiquidCrystal.h>                          // Add library
const int rs=12, en=11, d4=8, d5=7, d6=5, d7=4;     // Set vars for IA2109 interface pinning
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);          // Activate LiquidCrystal
                                                    //
// ---Operating parameters-------------------------- //
float GridVoltRms = 230;                            // Default grid voltage in VAC
float GridVoltDevPos = 10;                          // Allowed positive grid deviation in percent
float GridVoltDevNeg = 10;                          // Allowed negative grid deviation in percent
float GridFrequency = 50.0;                         // Default grid frequency in Hz
float GridMaxCurrent = 35;                          // Max grid load in amp (main fuse)
float SolarMaxCurrent = 35;                         // Max solar load in amp
float CabinetMaxCurrent = 40;                       // Highest allowed current in cabinet
                                                    //
bool SerialMonitor = true;                          // RS232-monitor mode On/Off
int SerialInterval = 5;

. .. . . . . .

        // Combined current exceeds cabinet limits
      if (ErrorCode == 14)                          //
        { lcd.print(">> Solar OFF! <<"); }          // Combined current exceeds cabinet limits
      delay(500);                                   //
  }                                                 //
}                                                   //
```

Conclusie. Op de testbank doet de EnergieMonitor precies wat het doen moet. Nu de praktijk nog.

En de praktijk is soms vervelend irritant.

1. De netspanning blijkt veel hoger te kunnen worden dan 253 VAC,

2. en helaas moet er ergens in de buurt een apparaat aanwezig zijn (autolader?, lastrafo?) die harmonische storingen op het elektriciteitsnet injecteert. De monitor geeft daardoor te veel en te vaak een irritante overspanningsfout.

AverageVoltage(); Maximaalspanning in de praktijk Bij het huidige aantal zonnepaneel-installaties is een netspanning van méér dan 253 VAC helemaal geen bijzonderheid meer. Nader onderzoek heeft geleerd dat netbeheerders pas van overspanning spreken wanneer gedurende meer dan 10 minuten 264 VAC of méér op het net staat!

De schakeling en sketch van de Energiemonitor werken op zich zoals het hoort. Wel is er op de meetlokatie regelmatig sprake van netspannings-harmonischen (veelvouden van 50Hz) die de toegepaste sample-meet-methode verstoren, waardoor deze flinke overspanningen gaat zien, die er in werkelijkheid helemaal niet zijn.

Daar waar de Energiemonitor spanningen ‘ziet’ tot wel 284 VAC, ziet een RMS universeelmeter alleen de normale 230 VAC.

```
//  0 = no error
//  1 = No Voltage Reference Module found
//  2 = Auto adjusting sensor sensitivity
//  3 = Vcc too low
//  4 = Volt ADC overload
//  5 = Voltage below lower limit
//  6 = Voltage above solar limits (253 VAC)
//  7 = Voltage above upper limits (264 VAC)
//  8 = Harmonic error

//  9   7= Grid ADC overload
// 10   8 = Solar ADC overload
// 11   9 = Grid current max!
// 12  10 = Grid sensor overload
// 13  11 = Solar current max!
// 14  12 = Solar sensor overload
// 15  13 = Cabinaet overload
// 16  14 = Solar Switched off
```

Als oplossing wordt eerst de Errorcodelijst aangepast, waardoor de erna volgende errorcodes omgenummerd moeten worden.

Omdat 264 VAC door de netbeheerder blijkbaar als ‘maximumwaarde’ wordt gezien, wordt deze als bedrijfsparameter toegevoegd.

```
// ---Operating parameters-
float GridVoltRms = 230;
float GridVoltDevPos = 10;
float GridVoltDevNeg = 10;
float GridVoltMaxMax = 264;
```

Uitgerekend de gebruikte RMS-sample-meet-methode blijkt gevoelig te zijn voor net-harmonischen. Om een harmonische overspanning te onderscheiden van de werkelijke netspanning, wordt alleen bij een overspanningsmelding de gemeten effectieve waarde vergeleken met een herleide netspanning op basis van een cumulatief gemiddelde spanning `GridVoltAvg`.

```
float GridVoltAvg = 0;
```

Met functie `AverageVoltage()`, wordt op de meest eenvoudigste manier de netspanning herleid. Deze methode wijkt af van de RMS-sample-meet-methode, doordat nu 1. op basis van gemiddelde ADC-waarden de netspanning herleid wordt, en 2. een andere sample-frequentie aangehouden wordt, 3. een ander aantal samples gebruikt wordt, en 4. het aantal meetperioden beperkt wordt tot 10.

De kans dat dezelfde harmonische beide meetmethoden op exact dezelfde wijze verstoren is bijzonder klein, zodat de tweede meetmethode als controle voor de eerste methode gebruikt kan worden .

Om de gemiddelde meetmethode-waarde zo nauwkeurig mogelijk te krijgen (en te houden!), wordt daarom ook gebruik gemaakt van de `VoltSense` - gevoeligheid. Dit levert per definitie een te lage berekende gemiddele spanning op, die nog gecorrigeerd moet worden met de meetmethode-vormfactor van 1,19.

```
void AverageVoltage()
  { int   AvgLoops = 0;
    float AvgCum = 0;
    int   AvgSamples = 0;
    int   AdcVolt = 0;
    while(true)
      { AvgSamples = 0;
        GridVoltAvg = 0.0;
        AdcVolt = analogRead(VoltSensor);
        if (AdcVolt > 0)
          { while(true)
              { GridVoltAvg = GridVoltAvg + AdcVolt;
                AvgSamples ++;
                AdcVolt = analogRead(VoltSensor);
                if (AdcVolt < 1)
                  { break; }
              }
            if (AvgSamples < 95)
              { continue; }
              else
              { GridVoltAvg = (VoltSense *
                  (GridVoltAvg / AvgSamples));
                GridVoltAvg = 1.19 * GridVoltAvg;
                AvgCum = AvgCum + GridVoltAvg;
                AvgLoops++;
                if (AvgLoops > 10)
                  { GridVoltAvg = AvgCum /10;
                    break;
                  }
                  else
                  { continue; }
              }
          }
      }
  }
```

De gebruikte vormfactor van 1,19 bij de gemiddelde meet-methode komt niet uit de lucht vallen. Wanneer als vormfactor 1,00 wordt gebruikt, kan deze met `DisplayMode=17` opgevraagd worden. We zien dat het verschil dan een factor 1.180 maal is.


```
if (DisplayMode == 17)
  { AverageVoltage();
    lcd.setCursor(0,0);
    lcd.print("RMS: ");
    lcd.print(GridVoltRms,1) ;
    lcd.print(" VAC    ");
    lcd.setCursor(0,1);
    lcd.print("AVG: ");
    lcd.print(GridVoltAvg,1) ;
    lcd.print(" ");
    lcd.print((GridVoltRms/GridVoltAvg),3);
    lcd.setCursor(0,1);
    delay(1);
  }
```

De gemiddelde meetmethode-spanning wordt met een vormfactor van 1.19 met opzet nèt iets boven de RMS-spanning afgesteld. Daardoor zal de AVG-meting altijd circa 2 volt boven de RMS-meting uitkomen; de AVG-spanningsdrempel

Uitsluitend en alleen wanneer een RMS-overspanning gemeten wordt, wordt ter controle ook de netspanning herleid volgens de gemiddelde meetmethode. Wanneer de RMS-spanning nu groter is dan de AVG-spanning, is er sprake van een harmonische RMS-meetfout.

In dat geval wordt de RMS-waarde gecorrigeerd met de AVG-waarde minus de AVG-spanningsdrempel. Uiteraard wordt deze harmonische storing wel als errror vastgelegd.

```
//---Voltage, Current & Power Measureme
MeasureRealPower();

if (GridVoltRms > GridVoltUpper)
  { AverageVoltage();
    if (GridVoltAvg < GridVoltRms)
      { GridVoltRms = GridVoltAvg - 2;
        ErrorCode = 8;
      }
  }
```

IA2109EM mk3 EnergyMonitor. Hieronder de software.

```
// IA2109EM_mk3_EnergyMonitor, Harm J Seef, The Netherlands
// Proof of concept.  Sketch for energy monitoring 230 VAC solar
// with the IA2109 shield, an analog Voltage Reference,
// and YHDC SCT013/30/1V current transformers.
// Warning; Mount current sensors in right direction!
//
// 2022 may  4  HJS  Designed for the IA2109 shield.
// 2022 aug 25  HJS  Grid current and Grid Power added.
// 2023 feb  9  HJS  Solar current and power added
// 2023 mrt 15  HJS  SolarStatus and Errorproc
// 2023 dec 09  HJS  Sketch redesigned.
// 2024 apr 09  HJS  - Harmonic error handling added
//                   - GridVoltMaxMax added
//
// ================================================================================================

#include <LiquidCrystal.h>                          // Add library
const int rs=12, en=11, d4=8, d5=7, d6=5, d7=4;     // Set vars for IA2109 interface pinning
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);          // Activate LiquidCrystal
                                                    //
// ---Operating parameters--------------------------- //
float GridVoltRms = 230;                            // Default grid voltage in VAC
float GridVoltDevPos = 10;                          // Allowed positive grid deviation in percent
float GridVoltDevNeg = 10;                          // Allowed negative grid deviation in percent
float GridVoltMaxMax = 264;                         // Maximum allowed grid voltage
float GridFrequency = 50.0;                         // Default grid frequency in Hz
float GridMaxCurrent = 35;                          // Max grid load in amp (main fuse)
float SolarMaxCurrent = 35;                         // Max solar load in amp
float CabinetMaxCurrent = 40;                       // Highest allowed current in cabinet
                                                    //
bool  SerialMonitor = true;                         // RS232-monitor mode On/Off
int   SerialInterval = 5;                           // RS232 log interval time in minutes
unsigned long SerialTimer = millis();               // Start timer
bool  AlarmOnError = false;                         // Alarm on error On/Off
                                                    //
// xxxSensorCalib                                   // Current sensor calibration factor =
                                                    // Measured Amps Rms / Displayed Amps Rms
float GridSensorCalib = 1.0 / 1.0;                  // Calibration factor grid sensor
float SolarSensorCalib = 1.0 / 1.0;                 // Calibration factor grid sensor
float CurrNulling = 0.0;                            // nulling current
//float CurrNulling = 0.12;                         //
                                                    //
//---I/O parameters---------------------------------- //
const int KeyPad = A0;                              // Keypad input
const int VoltSensor = A1;                          // Valid input A1, A2, A3, A4
// const int GridAmpSensor = A2;                    // grid current sensor on A2!  IS CONNECTED!!
// const int SolarAmpsensor = A3;                   // solar current sensor on A3!  IS CONNECTED!!
int AmpSensor = A2;                                 // current sensor input as variable!
                                                    // if (AmpSensor == A2)
                                                    //   { AmpSensor = A3; }
                                                    //   else
                                                    //   { AmpSensor = A2; }
const int VRefSensor = A4;                          // external voltage reference (ExRef)
const int BackLed = A5;                             // Used as digital output (LCD back light)
const int LedRed = 10;                              // Red LED      Not enough solar power available
const int LedOrange = 9;                            // Orange LED   Just enough solar power
const int LedGreen = 6;                             // Green LED    Solar exeeds grid power
                                                    //
//---Global parameters------------------------------- //
int   AdcVoltMax = 0;                               // Highest ADC value
float GridVoltUpper = GridVoltRms +                 // Highest allowed voltage
       ((GridVoltRms/100) * GridVoltDevPos);        //   (230 + 10% = 253.0)
float GridVoltLower = GridVoltRms -                 // Lowest allowed voltage
       ((GridVoltRms/100) * GridVoltDevNeg);        //   (230 - 10% = 207.0)
float GridVoltHighest = GridVoltLower;              // record highest voltage RMS ever
float GridVoltLowest = GridVoltUpper;               // record lowest voltage RMS ever
float GridVoltAvg = 0;                              // measures average grid voltage
                                                    //
int   AdcAmpMax = 0;                                // highest ADC value
float GridCurrentHighest = 0;                       // record highest grid current ever (forward)
float GridCurrentLowest = 0;                        // record lowest grid current ever (backwards)
float SolarCurrentHighest = 0;                      // record highest solar current solar ever (overload warning)
float SolarCurrentLowest = 0;                       // record lowest solar current ever (inverter malfunction)
                                                    //
float RealPow = 0;                                  // Real Power in Watt
int   GridApparantPower = 0;                        // Appearant Power from grid in VA
int   GridRealPower = 0;                            // Grid Real Power in Watt
float GridPowerFactor = 1.0;                        // Calculated power factor (cos phi), default
                                                    //
int   SolarApparantPower = 0;                       // Appearant Power from solar in VA
int   SolarRealPower = 0;                           // Solar Real Power in Watt
float SolarPowerFactor = 1.0;                       // Calculated power factor (cos phi), default
                                                    //
float CabinetCurrent = 0;                           // Maximum cabinet component design specs
float CabinetCurrentHighest = 0;                    //
float CabinetCurrentLowest = 0;                     //
                                                    //
                                                    //
//---Solar Parameters-------------------------------- //
int SolarThreshold = 200;                           // Solar panel module in Watt
int SolarStatus = 1;                                // 'trafic light'; Red, Orange, Green
                                                    // 1 = Red          Grid, No Solar at all               (Up-signal to cascade counter)
                                                    // 2 = Red-Orange   Grid exceeds solar                  (Up signal to cascade counter)
                                                    // 3 = Orange       Solar equals grid                   (Up-signal to cascade counter)
                                                    // 4 = Green-Orange Solar exceeds grid                  (Up-signal to cascade counter)
                                                    // 5 = Green        Solar feed, surplus back to grid    (Solar feeds all; no signal))
                                                    // 6 = Green        Solar feed. Grid back feed overload  (Down-signal to cascade counter)
                                                    //
//---Auxiliary vars---------------------------------- //
```

```
int  InitShowDelay = 2000;                          // Delay(time) in Init loop in ms.
int  KeyVal = 0;                                    // Keypad value
bool BackStatus = true;                             // Back light on
int long BackOnTime = 60;                           // Back light on time in seconds
int long BackStartTime = 0;                         // Back light on start time marker
bool KeyState = false;                              // no key pressed
bool KeyPrev = false;                               // previous key state
                                                    //
float VRefVcc = 5.0;                                // Initial Vcc at startup
float VRefSense = VRefVcc / 1024;                   // Initial VoltMax analog input = Vcc = 5VDC = 1024 steps
                                                    // Initial VrefSense = 5/1024 = 0,0048828 V/step
// float VoltSense = 0.2868764;                     // Default sensitivity 230 VAC
                                                    // Analog input from 0...4,5 VDC measures 0...370 VMaxTop (=264,5 VACrms)
                                                    // StepMax = (4,5/5) * 1024 = 922
                                                    // VoltSense = 264,5/922= 0,2868764
float VoltSense = 0.455;                            // Default sensitivity required by sampling RMS-method
float VoltSenseCalibCorr = VoltSense / 0.2868764;   // Calibration Correction factor = 1.5860489
                                                    //
float CurrSense = 0.1463415;                        // Default current sensor sensitivity at startup
                                                    //
float CurrRms = 0;                                  // Measured current
float GridCurrentRms = 0.0;                         // Measured grid current in Ampere RMS
float SolarCurrentRms = 0.0;                        // Measured solar current in Ampere RMS
                                                    //
int   GridZeroBias = 512;                           // Grid current zero line
int   SolarZeroBias = 512;                          // Solar current zero line
                                                    //
int   AnalogValue = 0;                              // 0...1023
                                                    //
char  CharValue[17] ="1234567890123456";            // char array for right alignment on 2*16 LCD-display
int   ErrorCode = 0;                                //  0 = no error
                                                    //  1 = No Voltage Reference Module found
                                                    //  2 = Auto adjusting sensor sensitivity
                                                    //  3 = Vcc too low
                                                    //  4 = Volt ADC overload
                                                    //  5 = Voltage below lower limit
                                                    //  6 = Voltage above solar limits (253 VAC)
                                                    //  7 = Voltage above maximum limits (264 VAC)
                                                    //  8 = Harmonic error
                                                    //  9 = Grid ADC overload
                                                    // 10 = Solar ADC overload
                                                    // 11 = Grid current max!
                                                    // 12 = Grid sensor overload
                                                    // 13 = Solar current max!
                                                    // 14 = Solar sensor overload
                                                    // 15 = Cabinaet overload
                                                    // 16 = Lower Solar Surplus
                                                    //
///---Display------------------------------------- //
int DisplayMode = 17;                               //  0 = Show solar and grid power (Monitor mode)
                                                    //  1 = Grid monitor mode; U RMS,I RMS
                                                    //  2 = Solar monitor mode; U RMS, I RMS
                                                    //  3 = Lowest and highest voltage
                                                    //  4 = Show cabinet highest and lowest amp
                                                    //  5 = Grid lowest and highest current
                                                    //  6 = Solar lowest and highest current
                                                    //  7 = Grid power and power factor
                                                    //  8 = Solar power and power factor
                                                    //  9 = Solar control (trafic light)
                                                    // 10 = Analog Reference mode
                                                    // 11 = Vcc (Aref) and ADC sensitivity
                                                    // 12 = Grid current sensitivity
                                                    // 13 = Solar current sensitivity
                                                    // 14 = VAC Calibration mode
                                                    // 15 = IAC Grid Calibration mode
                                                    // 16 = IAC Solar Calibration mode
                                                    // 17 = Check voltage RMS and AVG
                                                    // 18 = Reset Lowest/Highest records
                                                    //
//---Measuring Real Power RMS------------------------ //
void MeasureRealPower(void)                         // Measuring RealPower RMS with the max possible samples...
  { int AverageLoop = 0;                            // loop counter
    int MeaSamples = 0;                             // samples counter
    int AdcAmpPrev = 0;                             // previous adc currdent sample
    int AdcVolt = 0;                                // adc voltage sample
    int AdcAmp = 0;                                 // adc current sample
    float MeaVolt = 0;                              // measured voltage
    float MeaCumVolt = 0;                           // cumulated voltage samples
    float MeaVoltRms = 0;                           // calculated voltage rms
    float MeaAmp = 0;                               // measured current
    float MeaCumAmp = 0;                            // cumulated current samples
    float MeaRealPow = 0;                           // measured real power
    float MeaCumRealPow = 0;                        // cumulated real power
    float CumVoltRms = 0;                           // cumulated voltage RMS
    float AmpRms = 0;                               // Current RMS
    float CumAmpRms = 0;                            // cumulated current RMS
    //float RealPow = 0;                            // Real Power in Watt; global
    float CumRealPow = 0;                           // cumulated Real Power
    float AppPow = 0;                               // Apparant Power in VA
    float CumAppPow = 0;                            // cumulated apparant power
    float PowerFactor  = 0;                         // cos phi-factor real/apparant power
    //float GridVoltRms = 0;                        // global var
    AverageLoop = 0;                                // Reset average loop counter
    while(true)                                         // Average loop...
      { MeaSamples = 0;                                 // Reset vars...
        AdcVoltMax = 0;                                 //
        AdcAmpMax = 0;                                  //
        AdcAmpPrev = analogRead(AmpSensor);             // sample current as ADC-value
        AdcVolt = analogRead(VoltSensor);               // sample voltage as ADC-value
        AdcAmp = analogRead(AmpSensor);                 // sample current as ADC-value
        if (AdcVolt > 0)                                // after a zero crossing (positive halve period)
          { while(true)                                     // start measurement loop...
              { if (AdcVolt > AdcVoltMax)                   //
```

```
            { AdcVoltMax = AdcVolt; }                  // update highest ADC-value
            MeaVolt = AdcVolt * VoltSense;             // Calculate sample voltage
            MeaCumVolt += (MeaVolt * MeaVolt);         // add square value to sum of all samples
            if (AdcAmp > AdcAmpMax)                    //
              { AdcAmpMax = AdcVolt; }                 // update highest ADC-value
            MeaAmp = (AdcAmpPrev + AdcAmp) / 2;        // adjust real current
            MeaAmp = (AdcAmp -512) * CurrSense;        // Calculate sample current
            MeaCumAmp += (MeaAmp * MeaAmp);            // add square value to sum of all samples
            MeaRealPow = MeaVolt * MeaAmp;             // Calculate instant real power
            MeaCumRealPow += MeaRealPow;               // add to cumulativ
            MeaSamples ++;                             // raise counter
            AdcAmpPrev = AdcAmp;                       // shift current sample as previous
            AdcVolt = analogRead(VoltSensor);          // read next samples
            AdcAmp = analogRead(AmpSensor);            //
            if (AdcVolt < 1)                           // at end of positive half period
              { break; }                               // quit loop
          }                                            //
        if (MeaSamples > 40)                         // if enough valid samples measured
          { AverageLoop ++;                          // raise average counter
            MeaVoltRms =                             // Calculate
              sqrt((MeaCumVolt/MeaSamples));         // voltage RMS
            CumVoltRms += MeaVoltRms;                // add to cumulative
            AmpRms = sqrt((MeaCumAmp/MeaSamples));   // Caluclate current RMS
            CumAmpRms += AmpRms;                     // add to cumulative
            RealPow = MeaCumRealPow/MeaSamples;      // Calculate real power
            CumRealPow += RealPow;                   // add to cumulative
            AppPow = MeaVoltRms * AmpRms;            // Calculate apparant power
            CumAppPow += AppPow;                     // add to cumulative
            MeaCumVolt = 0;                          // reset vars
            MeaCumAmp = 0;                           //
            MeaCumRealPow = 0;                       //
            //Serial.print(GridVoltRms);             //
            //Serial.print(" VAC \t");               //
            //Serial.print(AmpRms);                  //
            //Serial.print(" Amp \t");               //
            //Serial.print(RealPow);                 //
            //Serial.print( " Watt \t");             //
            //Serial.println(MeaSamples);            //
            //delay(1);                              //
          }                                          //
        if (AdcVoltMax > 1000)                       // (0...1023)
              { ErrorCode = 4;                       // Volt sensor overload
                break;                               //
              }                                      //
        if (AdcAmpMax > 1000)                        // (0...1023)
          { ErrorCode = 9;                           // Amp sensor overload
            break;                                   //
          }                                          //
      }                                              //
    if (AverageLoop >= 25)                           // if enough measurements executed
      { GridVoltRms = CumVoltRms / 25;               // Calculate Real Voltage RMS
        AmpRms = CumAmpRms / 25;                     // Calculate Real Current RMS
        if ((AmpRms < CurrNulling) &&                // if current near to zero
            (AmpRms > (CurrNulling * -1.0)))         //
          { AmpRms = 0; }                            // make zero
        RealPow = CumRealPow / 25;                   // Calculate Real Power
        if (RealPow < 0)                             // if negative
          { AmpRms = -1.0 * AmpRms; }                // change current direction
        AppPow = MeaVoltRms * AmpRms;                // Calculate Apparant power
        PowerFactor = RealPow/AppPow;                // Calculate Power Factor
        if (AmpRms == 0)                             // adjust...
          { RealPow = 0;                             //
            AppPow = 0;                              //
            PowerFactor = 1;                         //
          }                                          //
        CumVoltRms = 0;                              // reset vars
        CumAmpRms = 0;                               //
        CumRealPow = 0;                              //
        CumAppPow = 0;                               //
        //Serial.print(GridVoltRms);                 //
        //Serial.print(" VAC \t");                   //
        //Serial.print(AmpRms);                      //
        //Serial.print(" Amp \t");                   //
        //Serial.print(AppPow);                      //
        //Serial.print( " VA \t");                   //
        //Serial.print(RealPow);                     //
        //Serial.print(" Watt \t");                  //
        //Serial.print("PF: ");                      //
        //Serial.println(PowerFactor);               //
        //delay(1);                                  //
        break;                                       //
      }                                              //
  }  // while average                              //
                                                    //
if (AmpSensor == A2)                                // if measuring grid values
  { if (ErrorCode == 9)                             // => Yes, this is correct
      { ErrorCode = 9; }                            // grid sensor
    GridCurrentRms = AmpRms * GridSensorCalib;      // update grid values...
    if (GridCurrentRms < GridCurrentLowest)         //
      { GridCurrentLowest = GridCurrentRms; }       // record lowest value ever
    if (GridCurrentRms > GridCurrentHighest)        //
      { GridCurrentHighest = GridCurrentRms; }      // record highest value ever
    if (GridCurrentRms > GridMaxCurrent)            // grid current overload
      { ErrorCode = 11; }                           //
    if (GridCurrentRms > 30.0)                      // grid sensor overload
      { ErrorCode = 12; }                           //
    GridApparantPower =                             // Apparant
      GridVoltRms * GridCurrentRms;                 //
    GridRealPower = RealPow;                        // Real
    GridPowerFactor = 1;                            // reset to default
    if ((GridRealPower != 0) &&                     // if a power measured
        (GridApparantPower != 0))                   //
      { GridPowerFactor =                           // calculate power factor
```

```
            1.0 * GridRealPower/GridApparantPower;  //
          }                                         //
        AmpSensor = A3;                             // switch to solar input
      }                                             //
      else                                          //
      { if (ErrorCode == 9)                         //
          { ErrorCode = 10; }                       // solar sensor
        SolarCurrentRms = AmpRms * SolarSensorCalib;  // update solar values...
        if (SolarCurrentRms < SolarCurrentLowest)   //
          { SolarCurrentLowest = SolarCurrentRms; }  // record lowest value ever
        if (SolarCurrentRms > SolarCurrentHighest)  //
          { SolarCurrentHighest = SolarCurrentRms; }  // record highest value ever
        if (SolarCurrentRms > SolarMaxCurrent)      // solar current overload
          { ErrorCode = 13; }                       //
        if (SolarCurrentRms > 30.0)                 // aolar sensor overload
          { ErrorCode = 14; }                       //
        SolarApparantPower =                        // Apparant
          GridVoltRms * SolarCurrentRms;            //
        SolarRealPower = RealPow;                   // Real
        SolarPowerFactor = 1;                       // reset to default
        if ((SolarRealPower != 0) &&                // if a power measured
            (SolarApparantPower != 0))              //
          { SolarPowerFactor = 1.0 *                // PowerFactor
              SolarRealPower / SolarApparantPower;  //
          }                                         //
        AmpSensor = A2;                             // switch to grid input
      }                                             //
    pinMode(AmpSensor,INPUT);                       // Switch current sensor
  }                                                 //
                                                    //
void AverageVoltage()                               //
  { int   AvgLoops = 0;                             // loop counter
    float AvgCum = 0;                               // cummulative
    int   AvgSamples = 0;                           // sample counter
    int   AdcVolt = 0;                              // adc value
    while(true)                                     //
      { AvgSamples = 0;                             //  // reset
        GridVoltAvg = 0.0;                          //
        AdcVolt = analogRead(VoltSensor);           //
        if (AdcVolt > 0)                            //
          { while(true)                             //      // start measurement loop
              { GridVoltAvg = GridVoltAvg + AdcVolt;  //      // add adc voltage to sum of previous measurements
                AvgSamples ++;                      //      // raise sample counter
                AdcVolt = analogRead(VoltSensor);   //      // read next sample
                if (AdcVolt < 1)                    //      // at end of positive half period
                  { break; }                        //      // quit sample loop
              }                                     //      //
            if (AvgSamples < 95)                    //  // if not enough samples
              { continue; }                         //  // restart measurement loop
              else                                  //  //
              { GridVoltAvg = (VoltSense *          //  // calculate average
                  (GridVoltAvg / AvgSamples));      //  // voltage
                GridVoltAvg = 1.19 * GridVoltAvg;   //  // Avg-form factor adjustment (normal above RMS)
                AvgCum = AvgCum + GridVoltAvg;      //  // add to cumulative
                AvgLoops++;                         //  // raise loop counter
                if (AvgLoops > 10)                  //  // after 10 measurements
                  { GridVoltAvg = AvgCum /10;       //  // recalculate
                    break;                          //  //
                  }                                 //  //
                  else                              //  //
                  { continue; }                     //  //
              }                                     //  //
          }                                         //  //
      }                                             //  //
  }                                                 //
                                                    //
//---------------------------------------------------- //
void setup()                                        //
  { Serial.begin(115200);                           // setup RS232 serial communication
    SerialInterval = SerialInterval * 1000;         // update value to millis
    BackOnTime = BackOnTime * 1000;                 // ditto
    pinMode(KeyPad,INPUT);                          // set up and connect KeyPad
    pinMode(VoltSensor,INPUT);                      // voltage sensor
    pinMode(AmpSensor,INPUT);                       // current sensor
    pinMode(VRefSensor,INPUT);                      // external voltage reference
    pinMode(BackLed,OUTPUT);                        // set up as digital output!
    digitalWrite(LedRed,HIGH);                      // Red LED      Not enough solar power available
    digitalWrite(LedOrange,HIGH);                   // Orange LED   Just enough solar power
    digitalWrite(LedGreen,HIGH);                    // Green LED    Solar exeeds grid power
    digitalWrite(BackLed,HIGH);                     // switch back light on
    Serial.println("EnergyMonitor IA2109EM mk3");   //
    lcd.begin(16,2);                                // Initialize LCD...
    lcd.clear();                                    // Reset LCD
    lcd.setCursor(0,0);                             // First line first position
    lcd.print("  IA2109EM mk3");                    // Show sketch name...
    lcd.setCursor(0,1);                             // Second line first position
    lcd.print("Energy Monitor... ");                // Show application...
    delay(InitShowDelay);                           //
    lcd.setCursor(0,0);                             // Show grid specs/operating parameters
    lcd.print("Grid: ");                            //
    lcd.print(GridVoltRms,0);                       //
    lcd.print(" VAC             ");                 //
    lcd.setCursor(0,1);                             //
    lcd.print("Freq:  ");                           //
    lcd.print(GridFrequency,0);                     //
    lcd.print(" Hz              ");                 //
    delay(InitShowDelay);                           //
    lcd.setCursor(0,0);                             //
    lcd.print("Main Max : ");                       //
    lcd.print(GridMaxCurrent,0);                    //
    lcd.print(" A               ");                 //
    lcd.setCursor(0,1);                             //
    lcd.print("Solar Max: ");                       //
```

```
    lcd.print(SolarMaxCurrent,0);                   //
    lcd.print(" A             ");                   //
    delay(InitShowDelay);                           //
    lcd.setCursor(0,0);                             //
    lcd.print("Low : ");                            //
    lcd.print(GridVoltLower,1);                     //
    lcd.print(" VAC           ");                   //
    lcd.setCursor(0,1);                             //
    lcd.print("High: ");                            //
    lcd.print(GridVoltUpper,1);                     //
    lcd.print(" VAC           ");                   //
    delay(InitShowDelay);                           //
    lcd.setCursor(0,0);                             //
    lcd.print("Max volt: ");                        //
    lcd.print(GridVoltMaxMax,0);                    //
    lcd.print( "VAC        ");                     //
    lcd.setCursor(0,1);                             //
    lcd.print("V Sensor Check...");                 //
    delay(InitShowDelay);                           //
    BackStartTime = millis();                       // set back light timer
  }                                                 //
                                                    //
//---------------------------------------------------- //
void loop()                                         // Main program...
  { //---Back Light Control-------------------------- //
    if (BackStatus == true)                         // Only if back light is on
      { if ((BackStartTime + BackOnTime) <          // if back light On time
          millis())                                 // passed by then
        { BackStatus = false;                       // reset back light
          DisplayMode = 0;                          // back to monitor power mode
          digitalWrite(BackLed,LOW);                // switch off
        }                                           //
      }                                             //
                                                    //
    //---Keypad Control------------------------------ //
    KeyVal = analogRead(KeyPad);                    // Read Keypad
    if (KeyVal < 710)                               // if no valid key pressed
      { KeyState = LOW; }                           // set Low
      else                                          //
      if (KeyVal < 815)                             // if Mode key pressed
        { KeyState = HIGH;                          // set High
        }                                           //
    if ((KeyState == HIGH) &&                       // if Mode key pressed AND
        (BackStatus == false))                      // back light is off, then
      { delay(500);                                 // debounce, and repetion keypress time
        BackStatus = true;                          // set var
        digitalWrite(BackLed,HIGH);                 // switch back led on
        BackStartTime = millis();                   // update back light timer
        KeyState = LOW;                             // reset
      }                                             //
    if ((KeyState == HIGH) &&                       // if Mode key pressed AND
        (KeyPrev == false))                         // not pressed before,
      { BackStartTime = millis();                   // then update back light timer
        DisplayMode++;                              // add 1 to mode value
        if (DisplayMode > 18)                       // if max number
          { DisplayMode = 0; }                      // roll over
      }                                             //
    if((KeyState == LOW) &&                         // if key not pressed anymore
       (KeyPrev == true))                           //
      { KeyPrev = false;                            // reset control var
        ErrorCode = 0;                              // reset errorcode
      }                                             //
                                                    //
    if ((ErrorCode > 0) &&                          // if errorproc-mode AND
        (BackStatus == true) &&                     // backlight on again AND
        (KeyState == HIGH))                         // error confirmed
      { lcd.setCursor(0,0);                         //
        lcd.print(" Resetting...      ");           //
        delay(100);                                 //
        ErrorCode = 0;                              // reset error
        KeyState = LOW;                             //
      }                                             //
    if (DisplayMode == 18)                          // if auto reset mode selected
       { if ((BackStartTime + BackOnTime - 200) <   // 200 ms before, back light On time
           millis())                                // passed by then
         { GridVoltHighest = GridVoltLower;         // reset record vars...
           GridVoltLowest = GridVoltUpper;          //
           GridCurrentHighest = 0.0;                //
           GridCurrentLowest = 0.0;                 //
           SolarCurrentHighest = 0.0;               //
           SolarCurrentLowest = 0.0;                //
           ErrorCode = 0;                           //
           DisplayMode = 0;                         //
           //Serial.println("AutoReset");           //
         }                                          //
       }                                            //
                                                    //
    //---Sensitivity & Reference--------------------- //
    AnalogValue = analogRead(VRefSensor);           // Read analog voltage reference (2,500 V should be 512)
    if ((AnalogValue < 472) ||                      // if External Analog Voltage Reference malfunctions
        (AnalogValue > 552))                        //
      { ErrorCode = 1; }                            // set error
      else                                          //
      if (((AnalogValue * VRefSense) > 2.510) ||    // If measured reference voltage too high OR
          ((AnalogValue * VRefSense) < 2.490))      // too low
        { ErrorCode = 2; }                          // set error
                                                    //
    if (ErrorCode == 2)                             // Whenever re-adjust OR
      { // 512 = 2,500 volt                         // theoretical
        // 534 (=2,500 Volt)                        // example measured value External Analog Voltage Reference
        // VRefSense = 2.500 / 534 = 0.0046816      // example calculated new value per ADC step
        VRefSense = 2.500 / AnalogValue;            // update to real ADC sensitivity
        VRefVcc = VRefSense * 1023;                 // Recalculate real Vcc = real ARef
```

```
      // 0,0046816 * 1023 = 4,7893258 VDC          // example calculated real Vcc
      if (VRefVcc < 4.500)                         // if supply voltage to low
        { ErrorCode = 3; }                         // set error
                                                   //
      // VoltSense = 0.2868764;                    // Voltage Startup default sensitivity
                                                   // Analog input from 0...4,5 VDC measures 0...370 VMaxTop (=264,5 VACrms)
                                                   // StepMax = (4,5/5) * 1024 = 922   (default)
                                                   // VoltSense = 264,5/922 = 0,2868764 (default)
                                                   //
      VoltSense = (264.5 /                         // If Vcc falls, then ARef falls, but signal stays 4.5 VDC
        ((4.5/VRefVcc) * 1024)) ;                  // StepMax = (4,5/VRefVcc) * 1024
                                                   // StepMax = (4,5/4,7) * 1024 = 980 (example)
                                                   // VoltSense = 264,5/StepMax = 264,5/980 = 0.269898 (example)
      VoltSense = VoltSense * VoltSenseCalibCorr;  // Calibration Correction factor, required by sampling RMS-method
                                                   //
      // CurrSense = 0.1463415;                    // Startup default.  YHDC Current Transformer SCT013/30A/1V  sensor specs;
                                                   // AnaMinMin  = 1,5 volt = 1,5/5 vdc = 1024 * (1,5/5) = 307 (Ana) = -30 Amp
                                                   // AnaAvgZero = 2,5 volt = 2,5/5 vdc = 1024 * (2,5/5) = 512 (Ana) =   0 Amp
                                                   // AnaMaxMax  = 3,5 volt = 3,5/5 vdc = 1024 * (3,5/5) = 717 (Ana) = +30 Amp
      // CurrSense = 30.0 /                        // number of Amp from 0... 30 , divided by
      //     (717 - 512 );                         // number of ADC steps from 0...30 Amp    (sets 0,1463415 amp per ADC-step)
                                                   //
                                                   // If Vcc falls, then ARef falls.  However,GridZeroBias (512) stays same, signal stays same
                                                   // Example;
                                                   // AnaMinMin  = 1,5 volt =>  1,5/4,7    vdc => 1024 * (1,5 /4,7) = 326 (Ana) = -30 Amp
                                                   // AnaAvgZero = 2,5 volt => (4,7/2)/4,7 vdc => 1024 * (2,35/4,7) = 512 (Ana) =   0 Amp
                                                   // AnaMaxMax  = 3,5 volt =>  3,5/4,7    vdc => 1024 * (3,5 /4,7) = 763 (Ana) =  30 Amp
      // CurrSense = 30/(763-512)=0.1195219        //
      CurrSense = 30.0 /                           //
        (((3.5/VRefVcc) * 1024) - 512);            // Theoretical by specs
  }                                                //
                                                   //
//---Voltage, Current & Power Measurement-------- //
MeasureRealPower();                                // Read Watt RMS
                                                   //
if (GridVoltRms > GridVoltUpper)                   // if Volt RMS above thresshold (& harmonic noise)
  { AverageVoltage();                              // remeasure grid voltage
    if (GridVoltAvg < GridVoltRms)                 // if average below RMS
      { GridVoltRms = GridVoltAvg - 2;             // correct value
        ErrorCode = 8;                             // Harmonic Error
      }                                            //
  }                                                //
                                                   //
if (GridVoltRms < GridVoltLowest)                  // if voltage lower
  { GridVoltLowest = GridVoltRms; }                // record lowest voltage ever ever
if (GridVoltRms > GridVoltHighest)                 // if voltage higher
  { GridVoltHighest = GridVoltRms; }               // record highest voltage ever
                                                   //
if (GridVoltRms < GridVoltLower)                   // if grid voltage to low
  { ErrorCode = 5; }                               // set error
if (GridVoltRms > GridVoltUpper)                   // if grid voltage to high
  { ErrorCode = 6; }                               // set error ( >253 VAC; solar Off)
if (GridVoltRms >  GridVoltMaxMax)                 // if grid exeeds maximum value
  { ErrorCode = 7;                                 //
  }                                                //
                                                   //
//---Cabinet------------------------------------- //
CabinetCurrent = GridCurrentRms + SolarCurrentRms;// calculate current through cabinet
if (CabinetCurrent > CabinetCurrentHighest)        //
  { CabinetCurrentHighest = CabinetCurrent; }      // record highest
if (CabinetCurrent < CabinetCurrentLowest)         //
  { CabinetCurrentLowest = CabinetCurrent; }       // record lowest
if (CabinetCurrent > CabinetMaxCurrent)            // if exeeds specs
  { ErrorCode = 15; }                              //
                                                   //
//---Solar Status-------------------------------- // Indicator 'trafic light' showing current solar status
// SolarStatus = 1                                 // 1 = Red          Grid, No Solar at all   (Up-signal to cascade counter)
                                                   // 2 = Red-Orange   Grid exceeds solar      (Up signal to cascade counter)
                                                   // 3 = Orange       Solar equals grid       (Up-signal to cascade counter)
                                                   // 4 = Green-Orange Solar exceeds grid      (Up-signal to cascade counter)
                                                   // 5 = Green        Solar surplus to grid
                                                   // 6 = Green-Red    Lower Solar surplus     (Down signal to cascade counter)
// int SolarThreshold = 100;                       // Watt Solar panel module
SolarStatus = 0;                                   // reset status
if ((GridRealPower > 0) &&                         // Grid feed in, AND
    (SolarRealPower <= SolarThreshold))            // No solar
  { SolarStatus = 1; }                             //
if ((GridRealPower > 0) &&                         // Grid feed in, AND
    (SolarRealPower > SolarThreshold))             // Solar feed in
  { SolarStatus = 2; }                             //
if ((GridRealPower > 0) &&                         // Grid feed in, AND
    (GridRealPower <                               // Grid feed in smaller as
      (SolarRealPower - SolarThreshold)))          // Solar feed in
  { SolarStatus = 3; }                             //
if ((SolarRealPower - SolarThreshold) >            // Solar feed in exceeds
     GridRealPower)                                // Grid feed in
  { SolarStatus = 4; }                             //
if (GridRealPower < 0)                             // Negative Grid feed
  { SolarStatus = 5; }                             //
if ((GridRealPower < 0) &&                         // Negative Grid feed, AND
   ((SolarRealPower - SolarThreshold) >            // Feed back exeeds
   SolarThreshold))                                // solar module capacity
                                                   // >> No feed back allowed! <<
  { SolarStatus = 6; }                             //
if ((ErrorCode == 6) &&                            // Voltage too high, AND
     (GridCurrentRms > SolarCurrentRms))           // and Grid feed larger as Solar feed
  { SolarStatus = 7; }                             // >> Solar module's switched off! <<
// if (ErrorCode == 16;                            //
                                                   //
if (SolarStatus == 0)                              // No SolarStatus
  { digitalWrite(LedRed,LOW);                      //
    digitalWrite(LedOrange, LOW);                  //
    digitalWrite(LedGreen, LOW);                   //
```

```
  }                                          //
if (SolarStatus == 1)                        // Grid, No solar at all
  { digitalWrite(LedRed,HIGH);               //
    digitalWrite(LedOrange, LOW);            //
    digitalWrite(LedGreen, LOW);             //
  }                                          //
if (SolarStatus == 2)                        // Grid exceeds Solar
  { digitalWrite(LedRed,HIGH);               //
    digitalWrite(LedOrange, HIGH);           //
    digitalWrite(LedGreen, LOW);             //
  }                                          //
if (SolarStatus == 3)                        // Solar equals grid
  { digitalWrite(LedRed,LOW);                //
    digitalWrite(LedOrange, HIGH);           //
    digitalWrite(LedGreen, LOW);             //
  }                                          //
if (SolarStatus == 4)                        // Solar exceeds Grid
  { digitalWrite(LedRed,LOW);                //
    digitalWrite(LedOrange, HIGH);           //
    digitalWrite(LedGreen, HIGH);            //
  }                                          //
if (SolarStatus > 5)                         // Solar surplus to Grid, OR
  { digitalWrite(LedRed,LOW);                //
    digitalWrite(LedOrange, LOW);            //
    digitalWrite(LedGreen, HIGH);            //
  }                                          //
if (SolarStatus == 6)                        // Lower Solar Surplus...
  { digitalWrite(LedRed,HIGH);               //
    digitalWrite(LedOrange, LOW);            //
    digitalWrite(LedGreen, HIGH);            //
  }                                          //
                                             //
//---Serial Monitor----------------------------- //
if (SerialMonitor == true)                   // Serial RS232-monitor mode on
  { if (millis() > SerialTimer)              // if current time after start marker
      { Serial.print(GridVoltRms,1);         //
        Serial.print(" VAC \t");             //
        Serial.print(GridCurrentRms);        //
        Serial.print(" IAC \t");             //
        Serial.print(GridApparantPower);     //
        Serial.print(" VA \t");              //
        Serial.print(GridRealPower);         //
        Serial.print(" WG \t; ");            //
        Serial.print(GridPowerFactor);       //
        Serial.print(" PF \t");              //
        Serial.print(SolarCurrentRms);       //
        Serial.print(" IAC \t");             //
        Serial.print(SolarApparantPower);    //
        Serial.print(" VA \t");              //
        Serial.print(SolarRealPower);        //
        Serial.print(" WS \t");              //
        Serial.print(SolarPowerFactor);      //
        Serial.print(" PF \t");              //
        Serial.print                         //
          (GridRealPower + SolarRealPower);  //
        Serial.print(" Watt \t");            //
        Serial.print                         //
          (GridCurrentRms + SolarCurrentRms);//
        Serial.print(" Amp \t S: ");         //
        Serial.print(SolarStatus);           //
        Serial.print(" \t E: ");             //
        Serial.println(ErrorCode);           //
        //Serial.println();                  //
        delay(1);                            //
        SerialTimer = millis() + SerialInterval;  // set new start time
      }                                      //
  }                                          //
//---Diplay modes------------------------------- //
lcd.clear();                                 //
if (DisplayMode == 0)                        // Energy monitor mode
  { lcd.setCursor(1,0);                      //
    lcd.print("Usage:");                     //
    dtostrf((GridRealPower + SolarRealPower),//
      5,0,CharValue);                        //
    lcd.print(CharValue);                    //
    lcd.print(" W          ");               //
    lcd.setCursor(0,1);                      //
    lcd.print("G:");                         //
    dtostrf(GridRealPower,5,0,CharValue);    //
    lcd.print(CharValue);                    //
    lcd.print(" S:");                        //
    dtostrf(SolarRealPower,5,0,CharValue);   //
    lcd.print(CharValue);                    //
    delay(1);                                //
  }                                          //
if (DisplayMode == 1)                        // Grid Monitor mode
  { lcd.setCursor(0,0);                      //
    lcd.print("Grid: ");                     //
    lcd.print(GridVoltRms,1) ;               //
    lcd.print(" VAC ");                      //
    lcd.setCursor(0,1);                      //
    lcd.print("      ");                    //
    if (GridCurrentRms >= 0)                 //
      { lcd.print(" "); }                    //
    if (GridCurrentRms < 10)                 //
      { lcd.print(" "); }                    //
    lcd.print(GridCurrentRms,1) ;            //
    lcd.print(" Amp ");                      //
    delay(1);                                //
  }                                          //
if (DisplayMode == 2)                        // Solar Monitor mode
  { lcd.setCursor(0,0);                      //
    lcd.print("Solar: ");                    //
```

```
      lcd.print(GridVoltRms,1) ;                  //
      lcd.print(" VAC ");                         //
      lcd.setCursor(0,1);                         //
      lcd.print("       ");                       //
      if (SolarCurrentRms >= 0)                   //
        { lcd.print(" "); }                       //
      if (SolarCurrentRms < 10)                   //
        { lcd.print(" "); }                       //
      lcd.print(SolarCurrentRms,1) ;              //
      lcd.print(" Amp ");                         //
      delay(1);                                   //
    }                                             //
  if (DisplayMode == 3)                           // Voltage lowest/highest mode
    { lcd.setCursor(0,0);                         //
      lcd.print("High Volt: ");                  //
      lcd.print(GridVoltHighest,1);               //
      lcd.setCursor(0,1);                         //
      lcd.print("Low Volt : ");                  //
      lcd.print(GridVoltLowest,1);                //
      delay(1);                                   //
    }                                             //
  if (DisplayMode == 4)                           // Cabinet lowest/highest amp mode
    { lcd.setCursor(0,0);                         //
      lcd.print("Hig Cab: ");                    //
      lcd.print(CabinetCurrentHighest,1);         //
      lcd.print(" A  ");                          //
      lcd.setCursor(0,1);                         //
      lcd.print("Low Cab: ");                    //
      lcd.print(CabinetCurrentLowest,1);          //
      lcd.print(" A  ");                          //
      delay(1);                                   //
    }                                             //
  if (DisplayMode == 5)                           // Grid lowest/highest amp mode
    { lcd.setCursor(0,0);                         //
      lcd.print("Hig Grd: ");                    //
      lcd.print(GridCurrentHighest,1);            //
      lcd.print(" A ");                           //
      lcd.setCursor(0,1);                         //
      lcd.print("Low Grd: ");                    //
      lcd.print(GridCurrentLowest,1);             //
      lcd.print(" A               ");             //
      delay(1);                                   //
    }                                             //
  if (DisplayMode == 6)                           // Solar lowest/highest amp mode
    { lcd.setCursor(0,0);                         //
      lcd.print("Hig Sol: ");                    //
      lcd.print(SolarCurrentHighest,1);           //
      lcd.print(" A ");                           //
      lcd.setCursor(0,1);                         //
      lcd.print("Low Sol: ");                    //
      lcd.print(SolarCurrentLowest,1);            //
      lcd.print(" A ");                           //
      delay(1);                                   //
    }                                             //
  if (DisplayMode == 7)                           // Grid monitor mode
    { lcd.setCursor(0,0);                         //
      dtostrf(GridCurrentRms,4,1,CharValue);      //
      lcd.print(CharValue);                       //
      lcd.print(" A      ");                      //
      lcd.setCursor(9,0);                         //
      dtostrf(GridApparantPower,4,0,CharValue);   //
      lcd.print(CharValue);                       //
      lcd.print(" VA    ");                       //
      lcd.setCursor(0,1);                         //
      dtostrf(GridPowerFactor,4,2,CharValue);     //
      lcd.print(CharValue);                       //
      lcd.print(" GPF    ");                      //
      lcd.setCursor(9,1);                         //
      dtostrf(GridRealPower,4,0,CharValue);       //
      lcd.print(CharValue);                       //
      lcd.print(" W     ");                       //
      delay(1);                                   //
    }                                             //
  if (DisplayMode == 8)                           // Solar monitor mode
    { lcd.setCursor(0,0);                         //
      dtostrf(SolarCurrentRms,4,1,CharValue);     //
      lcd.print(CharValue);                       //
      lcd.print(" A      ");                      //
      lcd.setCursor(9,0);                         //
      dtostrf(SolarApparantPower,4,0,CharValue);  //
      lcd.print(CharValue);                       //
      lcd.print(" VA    ");                       //
      lcd.setCursor(0,1);                         //
      dtostrf(SolarPowerFactor,4,2,CharValue);    //
      lcd.print(CharValue);                       //
      lcd.print(" SPF    ");                      //
      lcd.setCursor(9,1);                         //
      dtostrf(SolarRealPower,4,0,CharValue);      //
      lcd.print(CharValue);                       //
      lcd.print(" W     ");                       //
      delay(1);                                   //
    }                                             //
  if (DisplayMode == 9)                           //
    { lcd.clear();                                //
      lcd.setCursor(0,0);                         //
      lcd.print("SolarStatus: ");                 //
      lcd.print(SolarStatus);                     //
      lcd.print("             ");                 //
    }                                             //
  if (DisplayMode == 10)                          // Whenever in Analog Voltage Reference mode
    { lcd.setCursor(0,0);                         //
      lcd.print("Ref.Zer:");                     // zero line based on 2,500 VDC Analog Ref
      lcd.print(AnalogValue);                     // (lower Vcc means higher zero line)
```

```
      lcd.print(" Adc       ");                  //
      lcd.setCursor(0,1);                        //
      lcd.print("Sense:");                       //
      lcd.print((VRefSense),7);                  // calculated Reference Sensitivity
      lcd.print("          ");                   //
      delay(1);                                  //
    }                                            //
  if (DisplayMode == 11)                         // Whenever ADC step sensitivity
    { lcd.setCursor(0,0);                        //
      lcd.print("Aref  :");                      //
      lcd.print(VRefVcc,3);                      // new Vcc example 4,79 VDC (equals ARef)
      lcd.print(" VDC          ");               //
      lcd.setCursor(0,1);                        //
      lcd.print("Vsense:");                      //
      lcd.print(VoltSense,6);                    // voltage sensitivity per ADC step
      lcd.print(" V            ");               //
      delay(1);                                  //
    }                                            //
  if (DisplayMode == 12)                         // Grid current sensitivity
    { lcd.setCursor(0,0);                        //
      lcd.print(" Grid Amp.Sense  ");            //
      lcd.setCursor(0,1);                        //
      lcd.print(CurrSense,7);                    //
      lcd.print(" V/stp  ");                     //
      delay(1);                                  //
    }                                            //
  if (DisplayMode == 13)                         // Solar current sensitivity
    { lcd.setCursor(0,0);                        //
      lcd.print("Solar Amp.Sense ");             //
      lcd.setCursor(0,1);                        //
      lcd.print(CurrSense,7);                    //
      lcd.print(" V/stp  ");                     //
      delay(1);                                  //
    }                                            //
  if (DisplayMode == 14)                         // Voltage calibration mode
    { lcd.setCursor(0,0);                        //
      lcd.print(" Voltage adjust ");             //
      lcd.setCursor(1,1);                        //
      lcd.print(GridVoltRms) ;                   //
      lcd.print(" VAC RMS ");                    //
      lcd.print("          ");                   //
      ErrorCode = 0;                             // skip Error-handling
      delay(1);                                  //
    }                                            //
  if (DisplayMode == 15)                         // Irms zero current calibration mode
    { AmpSensor = A2;                            //
      pinMode(AmpSensor, INPUT);                 //
      GridZeroBias = analogRead(AmpSensor);      // Set ADC at 512 as current zero line
      lcd.setCursor(0,0);                        //
      lcd.print("Grid ADC : ");                  //
      lcd.print(GridZeroBias);                   //
      lcd.print("              ");               //
      lcd.setCursor(0,1);                        //
      lcd.print("Adjust   : ");                  //
      lcd.print(512-GridZeroBias);               //
      lcd.print("          ");                   //
      delay(1);                                  //
    }                                            //
  if (DisplayMode == 16)                         // Irms zero current calibration mode
    { AmpSensor = A3;                            //
      pinMode(AmpSensor, INPUT);                 //
      SolarZeroBias = analogRead(AmpSensor);     // Set ADC at 512 as current zero lin
      lcd.setCursor(0,0);                        //
      lcd.print("Solar ADC: ");                  //
      lcd.print(SolarZeroBias);                  //
      lcd.print("              ");               //
      lcd.setCursor(0,1);                        //
      lcd.print("Adjust   : ");                  //
      lcd.print(512-SolarZeroBias);              //
      lcd.print("          ");                   //
      delay(1);                                  //
    }                                            //
  if (DisplayMode == 17)                         // Check voltage RMS versus AVG
    { AverageVoltage();                          //
      lcd.setCursor(0,0);                        //
      lcd.print("RMS: ");                        //
      lcd.print(GridVoltRms,1) ;                 //
      lcd.print(" VAC    ");                     //
      lcd.setCursor(0,1);                        //
      lcd.print("AVG: ");                        //
      lcd.print(GridVoltAvg,1) ;                 //
      lcd.print(" ");                            //
      lcd.print((GridVoltRms/GridVoltAvg),3);    //
      lcd.setCursor(0,1);                        //
      delay(1);                                  //
    }                                            //
  if (DisplayMode == 18)                         // Reset records mode
    { lcd.setCursor(0,0);                        //
      lcd.print("Don't press Mode");             //
      lcd.setCursor(0,1);                        //
      lcd.print(" to Auto Reset  ");             //
      delay(250);                                //
    }                                            //
                                                 //
  //---Error procedure------------------------------ //
  if ((AlarmOnError == true) &&                  // if notification required
      (ErrorCode > 0))                           //
    { if (BackStatus == false)                   // Blink light as alarm
        { BackStatus = true; }                   //
      lcd.setCursor(0,0);                        // show error
      lcd.print("  ErrorCode:");                 //
      lcd.print(ErrorCode);                      //
      lcd.print("           ");                   //
```

```
      lcd.setCursor(0,1);                       //
      if (ErrorCode == 1)                       //
        { lcd.print("VRef malfunction"); }      // No Voltage Reference Module found
      if (ErrorCode == 2)                       //
        { lcd.print("Adjust Reference");        // Auto adjusting sensor sensitivity
          ErrorCode= 0;                         //
        }                                       //
      if (ErrorCode == 3)                       //
        { lcd.print("Vcc & ARef Low  "); }      // Vcc/Battery/accu low
      if (ErrorCode == 4)                       //
        { lcd.print("VoltADC overload"); }      //
      if (ErrorCode == 5)                       //
        { lcd.print("Volt Lower Limit "); }     // Grid Voltage too low
      if (ErrorCode == 6)                       //
        { lcd.print("Solar Upper Limit"); }     // Grid Voltage too high (Solar should be switched off
      if (ErrorCode == 7)                       //
        { lcd.print("Volt MaxMax Limit"); }     //
      if (ErrorCode == 8)                       //
        { lcd.print("Harmonic Error   "); }     //
      if (ErrorCode == 9)                       //
        { lcd.print("Grid ADC Max!!  "); }      // Grid ADC overload
      if (ErrorCode == 10)                      //
        { lcd.print("Solar ADC Max!! "); }      // Solar ADC overload
      if (ErrorCode == 11)                      //
        { lcd.print("Grid Amp Max!   "); }      // Grid current exceeds sensor limits
      if (ErrorCode == 12)                      //
        { lcd.print("Grid Sensor Max!"); }      // Grid current exceeds main fuse limit
      if (ErrorCode == 13)                      //
        { lcd.print("Solar Amp Max!   "); }     // Solar current exceeds sensor limits
      if (ErrorCode == 14)                      //
        { lcd.print("Solar Sensor Max!"); }     // Solar current exceeds main fuse limit
      if (ErrorCode == 15)                      //
        { lcd.print("Cabinet Overload "); }     // Combined current exceeds cabinet limits
      if (ErrorCode == 16)                      //
        { lcd.print(">> Solar OFF! <<"); }      // Combined current exceeds cabinet limits
      delay(500);                               //
      ErrorCode = 0;                            //
  }                                             //
}                                               //
```