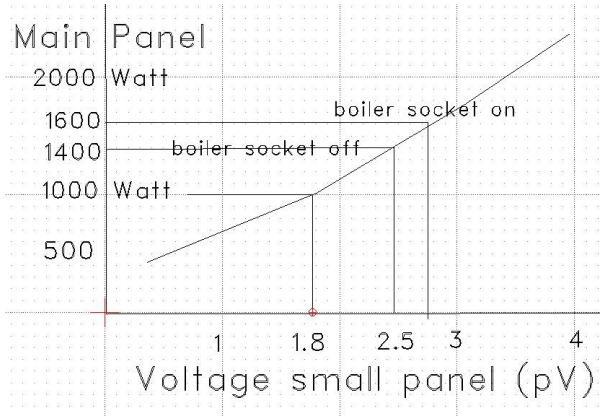
Zonne energie gestuurde stopcontacten - (Boiler als thuisbatterij)
Het opwarmen van een elektrische boiler, terwijl de zonnepanelen de stroom ervoor leveren, spaart het genereren van elektriciteit ervoor en brengt de verspilling van overtollige zonne-energie naar beneden. Het ontlast het elektriciteits-net en spaart de consument geld.
Als idee bij persoonlijk gebruik: Het opwarmen van 50l. water van 20° naar 70° kost ca. 3 kW.
Los van een elektrische boiler kunnen andere voorwerpen geladen of gebruikt worden terwijl de zon de energie ervoor levert: Het laden van auto, fiets, computer en gereedschap batterijen tot het gebruik van de stofzuiger, grasmaaier en airconditioning aan toe.
Het gebruik van een tijd instelbaar stopcontact kan het gebruik verschuiven naar de tijden waarin normaliter de zon energie levert maar gedurende bewolkte periodes gebruik je nog steeds energie uit het net. Daarom meten wij de opbrengst van de panelen en laten een simpele microprocessor diverse plug-in stopcontacten schakelen.

Het meten van de zonne-energie
Sommige omvormers (die de energie van de zonne-panelen omzetten naar de netspanning) hebben een mogelijkheid de opbrengst via een plug af te lezen. In ons geval wil ik echter liever niet de onbekende proprietary software van de omvormer gaan ondervragen. Daarom gebruik ik een klein zonnepaneeltje op het dak, dat naast de hoofdpanelen is gemonteerd. Na het opmeten van het voltage bij diverse opbrengsten van de grote panelen, was een grafiekje op te stellen: Vanaf 2.5 volt of hoger van het kleine paneeltje, levert het grote paneel de benodigde 1400 Watt die nodig is om onze 50 l. boiler op te warmen. We schakelen het stopcontact van de boiler daarom aan bij 2.75V en af bij 2.5V. (Een zekere spreiding is nodig om veelvuldig aan en uit tikken te voorkomen)


Vastgezet aan een, onder de pannen klemmende, lat.



Het aan en afschakelen van de boiler – Solar socket
Wij gebruiken hiervoor een plug-in stopcontact van een niet meer werkend remote gecontroleerd stopcontact. Het relais ervan hebben we vervangen voor een wat zwaarder type. Met een kabeltje naar de microprocessor wordt het relais voorzien van de benodigde 5 V om het relais aan te zetten.

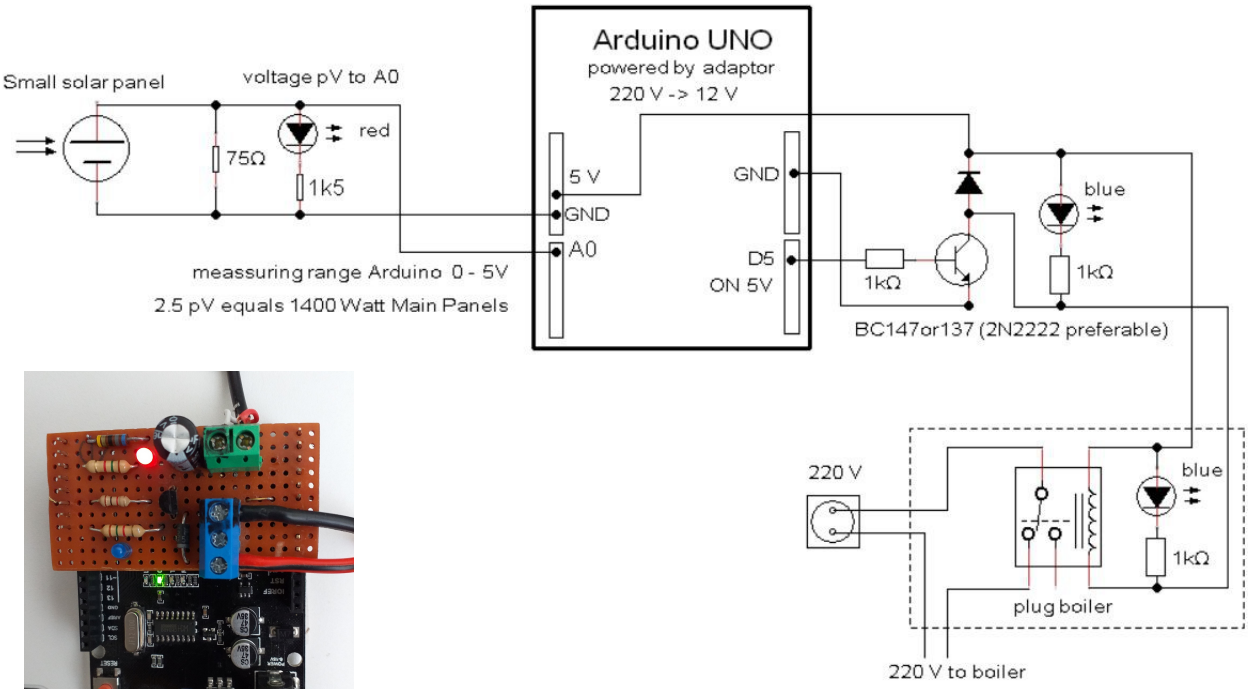
Een andere optie is een remote gecontroleerd stopcontact aan te sturen met een 433Mz zendertje op de microprocessor. Dit wordt later besproken.

De blauwe LED op de foto gaat aan als er spanning op het stopcontact staat. De stekker van de boiler zit ingeplugd. De rood zwarte draad loopt naar de meterkast waar de microprocessor zich bevind.

De boiler blijft tevens geschakeld door zijn eigen thermostaat.

De boiler is een simpele 50 liter boiler die boven de aanrecht is opgehangen. Hij is met T-stukjes aangesloten aan de koud en warm water leidingen van de kraan boven de gootsteen.
Bij het opwarmen expandeert het water; Dit wordt via een ventiel in de koud water leiding uit het systeem gelaten. Wij vangen het op in een glazen vaas voor hergebruik en gebruiksindicatie.
De warm water voorziening van de **gasketel** is via het menu afgezet en de fysieke kraan op de ketel is dichtgedraaid.
Omdat we elektrisch koken kan met deze warmwatervoorziening de gastkraan dicht blijven zolang we geen gasverwarming nodig hebben.

De microprocessor.
Als microprocessor gebruiken wij een Arduino Uno. Onze set-up is hieronder geschetst. De onderdelen, die aan beide zijden van de Arduino microprocessor zijn getekend, (Input links output rechts) zijn gemonteerd op een printje dat op de aansluit pinnen van de Arduino geschoven is. Zie de foto.



De analoge input van de Arduino heeft een bereik van 0-5 V. Het maximale voltage van het kleine paneeltje (pV) is daarom beperkt zodat het de 5V niet overschrijdt. Een rood LEDje gaat branden wanneer een pV van1.8 V is bereikt. In dat geval levert het grote paneel ongeveer 1000 Watt.

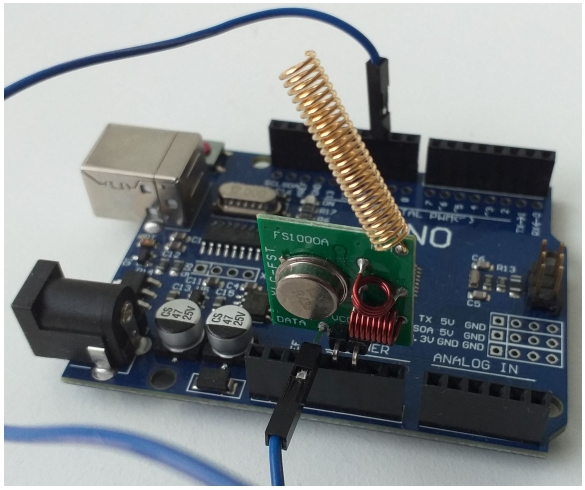
Het opwarmen van water
*$P = m * c * \Delta t / 3600$ (eenheden: kWatt = kg * kJ/kg * graden verschil in C or Kelvin)*
c water 4186 J/kg
*Om 50 l water 50° te verwarmen is 50 * 50 * 4.186 / 3600 = 2.91 kW nodig*

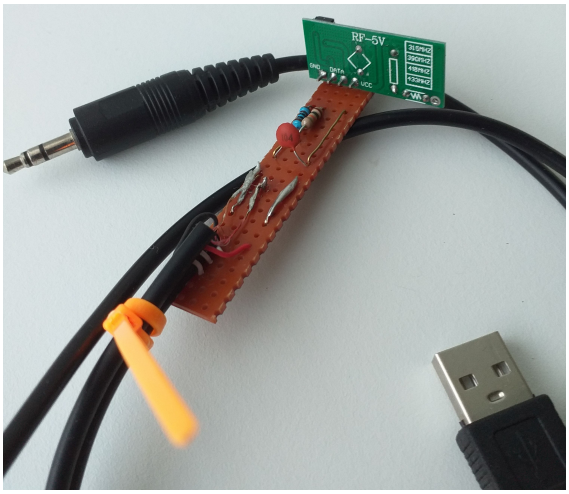
In een gemiddeld huishouden wordt 20% – 25% van het gas gebruikt voor het opwarmen van water. Een verschuiving naar direct door de zon gevoede elektrische boilers, als een soort thuisbatterij, lijkt daarom de moeite waard.

Om tot een meer algemene energie besparing te komen zou het handig zijn als de industrie de, aan de verschillende omvormers aan te sluiten, remote stopcontacten, zou leveren. (geschikt voor boiler en/of airco gebruik)

Het gebruik van remote controlled (RC) stopcontacten
Wij hebben RC stopcontacten aan ons systeem toegevoegd om een kleine kachel of acculader aan te schakelen wanneer de zonnepanelen daar de spanning voor leveren. Het schakelen van de boiler zou hier ook mee kunnen gebeuren. **Let wel op** het maximale vermogen dat deze stopcontacten mogen leveren. Sommige types mogen **slechts 1000W** schakelen.

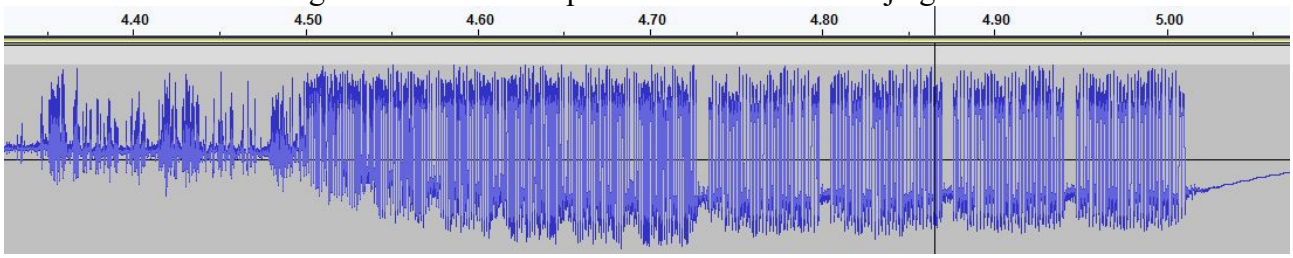
Om de RC stopcontacten te schakelen is de afstandsbediening nodig of een klein 433MHz zendertje dat bediend wordt door de microprocessor. Om het zendertje aan te sturen is het nodig de code te weten waarop de stopcontacten reageren voor elk commando. Om achter de code te komen kan men er op een computer naar luisteren in combinatie met het programma Audacity





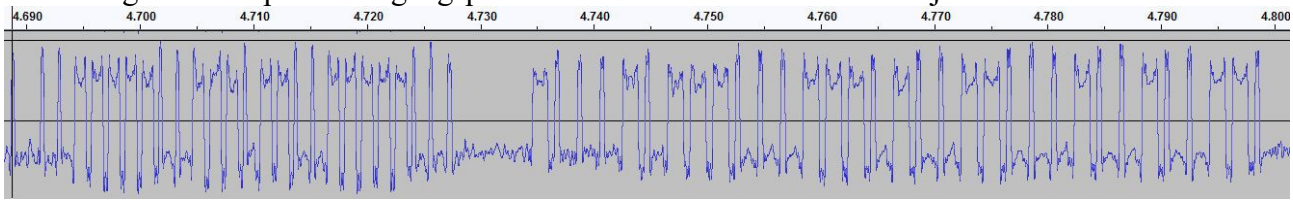
Zender (links) aangesloten aan de 5V en GND van de Arduino. De gebogen data pin is recht gebogen en verbonden met de digitale output 10 aan de andere zijde.
Ontvanger(rechts) De 5V en GND komen van een USB connector. De data en GND gaan naar een audio plug. Beiden gaan naar de computer. Het datasignaal is met een weerstandsbrug 1k-10k verzwakt.
Zender en ontvanger worden doorgaans samen verkocht voor een paar Euro.

Het vinden van de remote commando's
Dit kan met het programma Audacity met de 433 MHz ontvanger op de computer aangesloten. Het indrukken van een commando op de remote geeft dan, naast de reeds aanwezige storing, een hoorbaar en zichtbaat signaal dat men kan opnemen. Dit ziet er in mijn geval uit als dit:

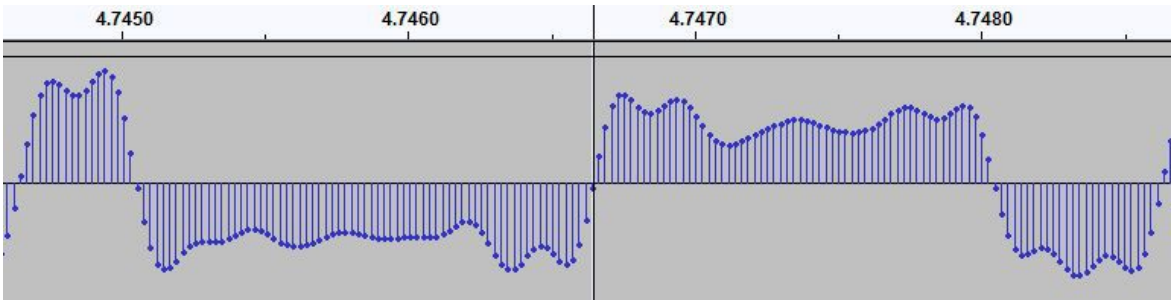


In dit geval ziet het commando eruit als een patroon van 24 bits dat 6 maal gezonden wordt, gevolgd door een patroon van 32 bits dat 4 maal wordt uitgezonden.

Verder ingezoomd op het overgangspunt van 24 naar 32 bits is ook het tijdsverschil zichtbaar:



Bij het nog verder inzoomen worden per “LOW” en “HIGH” bit de sample punten zichtbaar:



De standaard sample rate van Audacity is 44100 Hz. Door de sample punten te tellen is de tijdsduur van het hoog en laag uitzenden te berekenen. 44 sample punten bijvoorbeeld, duren 44 / 44100 = 998 µ seconds;

Om het signaal later in het programma te kunnen simuleren definieer ik in het begin van het programma de tijden van laag en hoog van elk bit type. De code volgorde wordt geteld en opgeslagen in twee multi arrays, één voor het 24 bits patroon en de andere voor het 32 bits patroon.

In het programma ziet het er als volgt uit:

```
// delay times (HIGH or LOW situations) in  µ seconds:

#define Bit24LowUp 295          //13 samplepoints
#define Bit24LowDown 1179       //52 samplepoints
#define Bit24HighUp 998         //44 samplepoints
#define Bit24HighDown 476       //21 samplepoints

#define Bit32LowUp 408          //18 samplepoints
#define Bit32LowDown 1610       //71 samplepoints
#define Bit32HighUp 1429        //66 samplepoints
#define Bit32HighDown 590       //26 samplepoints

#define StartDelay24LowUp 400
#define StartDelay24LowDown 2250
#define StartDelay32LowUp 408
#define StartDelay32LowDown 7188
```

Met 3 stopcontacten, elk met een “ON” en “OFF”, the 6 commando's en arrays zien eruit als:

```
bool S24[6][24] =    {{0,0,1,1,1,0,1,1,1,1,0,0,0,0,1,1,0,1,1,0,1,1,0,0},    //  A_ON
                      {0,0,1,1,1,1,1,1,0,0,1,1,1,1,0,1,1,0,0,1,1,1,1,1,0,0},   //  A_OFF
                      {0,0,1,1,0,0,0,0,1,0,1,1,1,0,1,0,0,0,1,1,0,1,0,1},   // B_ON
                      {0,0,1,1,0,1,1,1,0,0,0,0,1,1,0,1,0,1,0,0,0,1,0,1},   // B_OFF
                      {0,0,1,1,0,1,1,0,1,1,1,0,0,0,0,1,1,0,1,1,1,1,1,0},   // C_ON
                      {0,0,1,1,0,1,0,1,0,1,1,0,1,0,0,0,0,0,0,0,1,1,1,1,0}}; // C_OFF
```

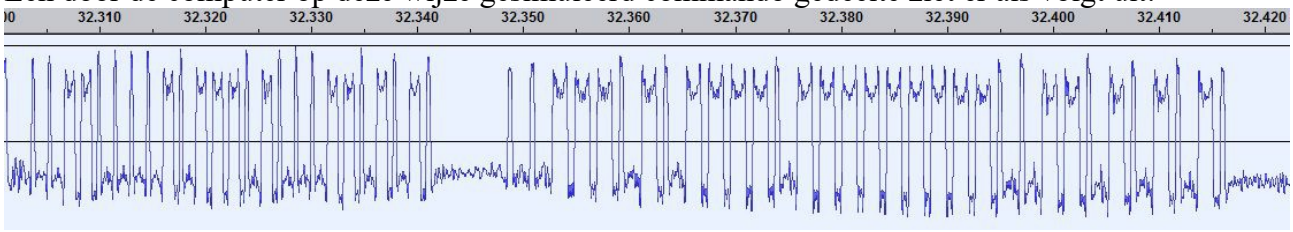
```
bool S32[6][32] =    {{1,1,1,1,0,0,0,1,0,1,0,1,1,1,1,1,1,1,1,1,0,1,0,1,1,0,1,0,0,1,1},    // A_ON
                      {1,0,0,0,1,0,1,1,1,0,0,0,1,1,1,0,1,0,0,1,1,0,0,0,1,0,0,1,0,0,1,1},   // A_OFF
                      {0,0,1,1,1,0,1,0,1,1,1,1,0,1,1,1,1,1,1,1,1,1,0,0,1,1,0,1,0,1,0,1},   // B_ON
                      {1,0,1,1,1,1,0,0,1,1,1,0,1,1,1,0,0,1,1,0,1,1,1,1,1,0,0,1,0,1,0,1},   // B_OFF
                      {1,1,1,1,1,1,1,1,1,1,0,1,1,0,0,1,1,0,1,0,1,0,0,1,1,1,0,1,1,0,1,1},   // C_ON
                      {1,1,0,1,0,0,1,1,0,0,1,0,0,0,0,1,1,0,0,0,0,0,0,1,1,0,0,1,1,0,1,1}}; // C_OFF
```

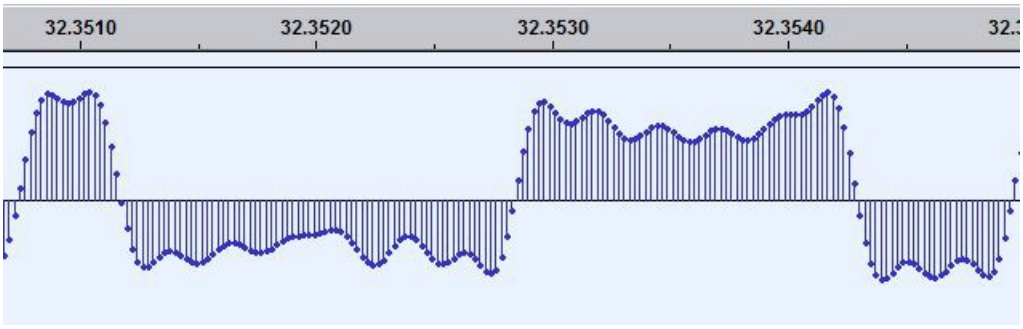
Los van de bit patronen begint elk patroon met een lang LOW. De timing hiervan komt nauwer dan die van de bit patronen. Gedurende het programma roep ik voor elk stopcontact en commando, dezelfde functie aan, met het betreffende commando nummer:

```
void switchRC(int commandNumber)
{
  int bitNumber;
  int bitPattern;
  for (bitPattern = 0; bitPattern < 6; bitPattern++)  // 24 bit part 6x
  {  // start with the extra long LOW part
    digitalWrite(pinRCdata, HIGH);
    delayMicroseconds(StartDelay24LowUp);
    digitalWrite (pinRCdata, LOW);
    delayMicroseconds(StartDelay24LowDown);
    for (bitNumber = 0; bitNumber < 24; bitNumber++)
    {
      if (S24[commandNumber][bitNumber] == HIGH)
      {
        digitalWrite (pinRCdata, HIGH);
        delayMicroseconds(Bit24HighUp);
        digitalWrite (pinRCdata, LOW);
        delayMicroseconds(Bit24HighDown);
      }
      else
      {
        digitalWrite (pinRCdata, HIGH);
        delayMicroseconds(Bit24LowUp);
        digitalWrite (pinRCdata, LOW);
        delayMicroseconds(Bit24LowDown);
      }
    }
  }
  for (bitPattern = 0; bitPattern < 4; bitPattern++)  // 32bits part 4x
  {  //start with  an extra long LOW:
    digitalWrite (pinRCdata, HIGH);
    delayMicroseconds(StartDelay32LowUp);
    digitalWrite (pinRCdata, LOW);
    delayMicroseconds(StartDelay32LowDown);
    for (bitNumber = 0; bitNumber < 32; bitNumber++)
    {
      if (S32[commandNumber][bitNumber] == HIGH)
      {
        digitalWrite (pinRCdata, HIGH);
        delayMicroseconds(Bit32HighUp);
        digitalWrite (pinRCdata, LOW);
        delayMicroseconds(Bit32HighDown);
      }
      else
```

```
      {
       digitalWrite (pinRCdata, HIGH);
       delayMicroseconds(Bit32LowUp);
       digitalWrite (pinRCdata, LOW);
       delayMicroseconds(Bit32LowDown);
      }
    }
  }
}
```

Een door de computer op deze wijze gesimuleerd commando gedeelte ziet er als volgt uit:

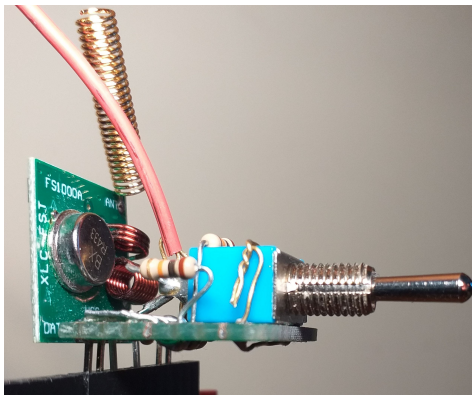


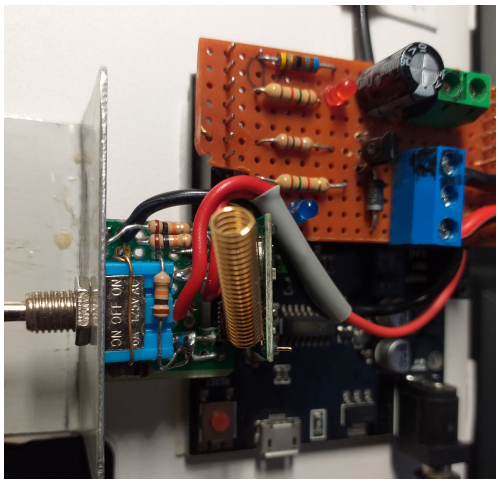


De gesimuleerde commando's werken prima voor alle drie RC stopcontacten.

Het toevoegen van de zender aan de eerder gebouwde setup.
Het zender-printje kan ik met zijn GND en data pin rechtstreeks in de GND en digitale output 12 drukken als ik eerst de 5V pin recht buig. De 5 volt pin verbind ik met de 5V out van de Arduino.

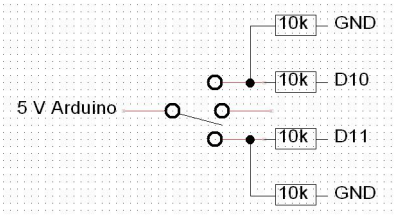
Ik heb tevens een 3 standen schakelaar toegevoegd met het doel de boiler ook aan te kunnen zetten als er geen zon is of af te zetten als de inhoud warm genoeg is.
Met deze 3 standen schakelaar zet ik 5V op digitale inputs D10 of D11 om de keuze aan het Arduino programma duidelijk te maken. Het relais van het boiler-stopcontact blijft, al dan niet bekrachtigd, via output D5. De stroom naar de D10 en D11 is beperkt met een 10k weerstand en er is tevens een 10k pull-down weerstand naar GND toegevoegd. Zonder de pull-down weerstanden blijven de inputs af en toe op high hangen.





Niet volledig ingedrukt om de pinnen zichtbaar te houden.
De twee rchter pinnen zijn niet aangesloten en hebben alleen

een mechanische functie.



De 10 k pull-down weerstanden van D10 en D11 naar GND zijn zichtbaar op de rechter foto.

De volgende programma's voor de microprocessor zijn te openen/ modificeren in het Arduino IDE (Integrated Development Environment). Hiermee zijn de programma's ook te installeren op de diverse microcontrollers. Het IDE bevat tevens een serieel monitor programma waarmee tijdens het runnen van het programma diverse waarden (bijvoorbeeld de meetwaarde van het zonnepaneeltje) in de gaten gehouden kan worden.

De listings zijn down te loaden van de website bootprojecten.nl

_240709_Solar_Switched_BoilerSocket.ino
Simpel programma dat alleen het boiler stopcontact aan of uit zet

_240719_Solar_Switched_Boiler_Socket.ino
Als boven met de 3 standen schakelaar “Altijd aan”, ”Zon geregeld”, ”Altijd uit”

_240828_Solar_Switched_Sockets.ino
Als boven met de RC stopcontacten.

De programma's dienen als voorbeeld/ idee en zijn makkelijk aan te passen aan uw eigen situatie en voorkeuren.

Jeroen Droogh, bootprojecten@gmail.com