Boiler as “home battery“

Heating your electric boiler, while your solar panels are delivering the power for it, saves generating the power and brings down the waste of unused solar power during peak production periods. As a consumer it will probably save you money.

As an idea on personal level: the heating of 50l. of water from 20° to 70° will consume ca. 3 kW.

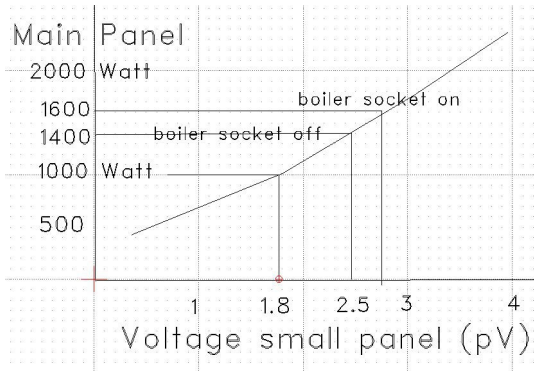
Apart from an electric boiler, other items can be used or charged while your panels are delivering power, like for instance car-, bike- , computer- or other batteries. The same counts for the use of an air-conditioning or lawn mower.

Using a time controlled socket can shift the heating of the boiler to your normally sunny times, but during cloudy conditions the boiler will still draw power from the grid. So we monitor the sun directly and let a simple microprocessor switch the socket of the boiler.

How to measure the solar power input

Some inverters (which convert the power from your panels to AC and the voltage of the grid) offer a way to monitor the solar input. In our case however, I am not directly willing to connect to the proprietary software and interrogate the internal data of the inverter. So we use a small solar panel which is mounted next to the main panels. Monitoring it's voltage we found that from 2.5 volts upwards, the main panels deliver the 1400 Watt needed to heat our (50 l.) boiler. So we switch the boiler's socket “ON” at 2.75 Volt and “OFF” at 2.5 Volt.

Fitted on a baton, stuck under the shingles



How to switch the boiler “ON” or “OFF” - Solar socket

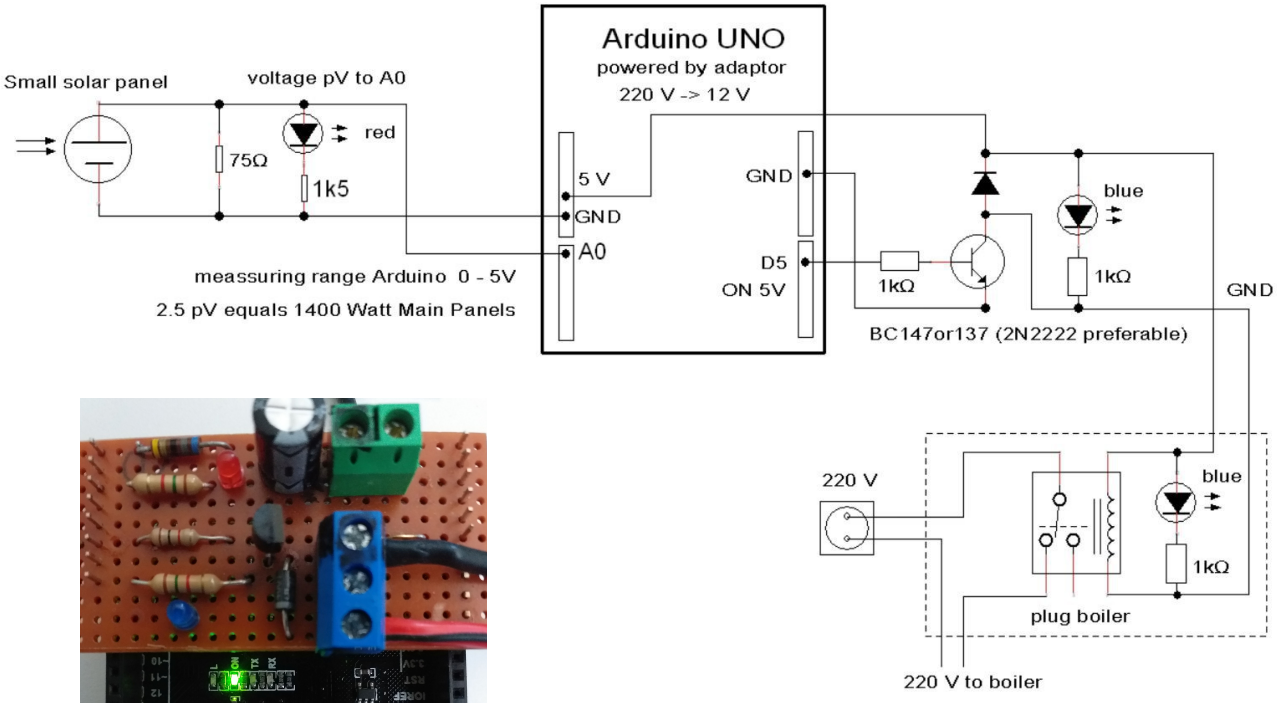
We use a controllable socket for our boiler. We fitted a non-working remote controlled (RC) socket with a heavier relay and run the required 5V to operate the relay with a cable from the microcontroller.

*Another option is the use of remote control (RC) sockets, **with an appropriate power rating**, and control them via a small 433MHz transmitter, this is explained later on.*

The blue LED on the socket indicates that there's power on the socket. The picture shows the plug of the boiler in the socket and the black&red wire coming from the microcontroller. This wire puts 5V on the relay when the solar panels produce at least 1400 Watts.

The boiler is still controlled by its own thermostatic switch and will only switch on when the water content is to be heated. Ours is a simple 50 litre type ,connected with "T"connectors near the piping of the kitchen sink. The warm water system of the central(gas) heating system is switched of in its menu and the physical valve is closed. As we cook electrical (induction), we closed our main gas supply.
We only keep the gas connection for a winter situation we need gas for heating our home.
The during the heating expanding water is collected in the vase for re-cycling and as boiler-use indication.

The microprocessor.
The parts drawn on either side of the Arduino are mounted on the small pictured circuit board:



The analogue input of the Arduino microcontroller can handle 0 -5 V. Therefore the maximum pV is adjusted in order not to exceed 5V. A red LED light will show when the pV reaches 1.8 Volt.
In this situation, the main panels deliver 1000 Watt.
When digital output D5 triggers the relay of the boiler's socket, the blue LED on the print lights up.

Heating of water:
$P = m * c * \Delta t / 3600$ (*units: kWatt = kg * kJ/kg * degree difference C or Kelvin)*
c water 4186 J/kg
*To heat 50 l water 50° will costs 50 * 50 * 4.186 / 3600 = 2.91 kW*

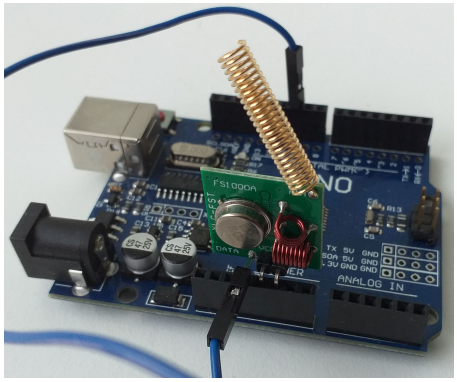
In an average Dutch household, 20% – 25% of gas consumption is used for the heating of water. So a shift to solar powered electric boilers, as a kind of home batteries, looks worth the effort.

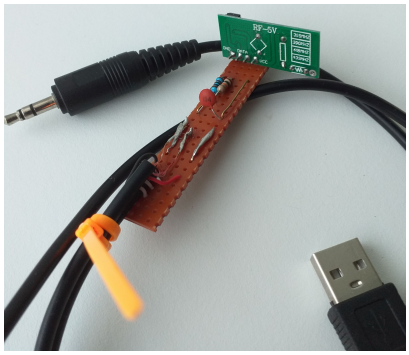
It would be nice if the industry supplied remote controlled sockets, which are directly controlled by the inverter of the solar panels. It would save rigging the small solar panel and it would make the direct use of solar power wider accessible.

Using remote controllable (RC) sockets.

We added RC sockets to our system to be able to use a small heater or a battery charger while the solar panels are producing power.
You can do the same for your boiler but before buying sockets I would recommend to **check their power rating** beforehand. Some RC sockets can only handle 1000 Watts.

To operate the sockets without the remote you need a small 433 MHz transmitter-shield to send the signal. You also have to figure out the signal for each individual command. For this a 433MHz receiver on your computer's sound card, together with the program Audacity can be used:



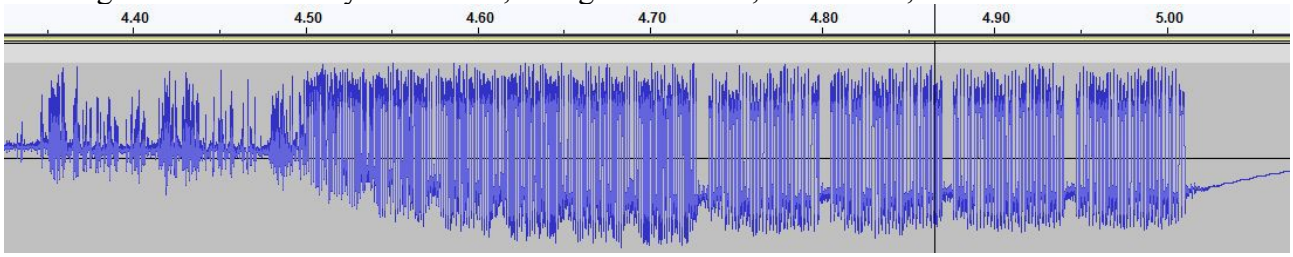


Transmitter (left) connected to the 5V and GND of the |Arduino, the angled data header pin is straightened to evade the female header of the Arduino and is lead to digital output 10 at the other side.
Receiver (right) The 5V and GND are supplied via an USB connection, the data and GND connect to an audio plug to connect to your computer's sound-card. The data signal from the receiver is made weaker with a 1k-10k resistor bridge.
Transmitter and receiver are mostly sold together for a few Euro's.

Finding the commands

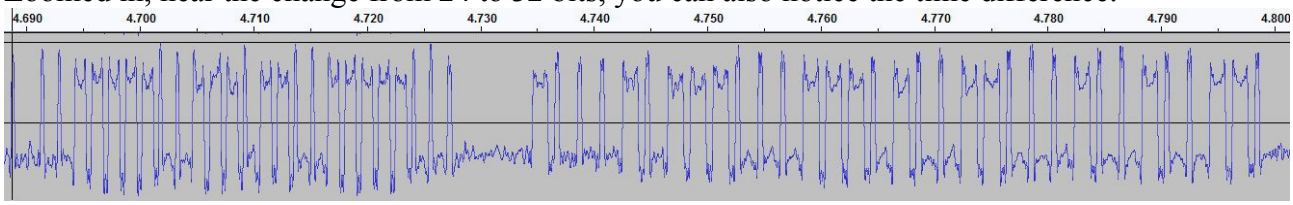
If you run Audicity and push the remote of the RC Sockets, you will receive, apart from a lot of disturbance, the given remote commands. They will be audible and visible on the recording.

Looking at the data send by the remote, a single command, in our case, looks like this:

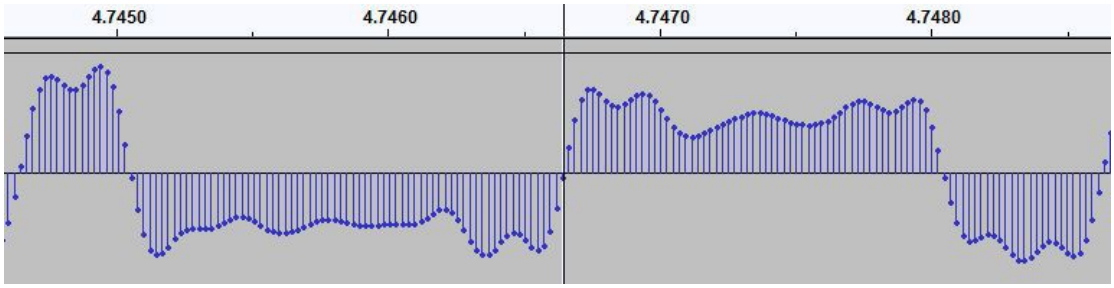


In this case it consists of a pattern of 24 bits, which is send 6 times followed by a pattern of 32 bits, which is send 4 times.

Zoomed in, near the change from 24 to 32 bits, you can also notice the time difference:



Zoomed in to a level that the sample points are visible, a “LOW” and “HIGH” bit look like this:



The standard sample rate of Audacity is 44100 Hz. So, for example, 44 sample points take
44 / 44100 = 998 µ seconds;
To simulate the signal later in the program I define the times of the LOW and HIGH for each bit type, and times of the HIGH/LOW part at the start of the patterns. The code sequences are stored in two multi arrays, one for the 24 bits pattern and the other for the 32 bits pattern.

In the program it looks like this:
// 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
```

Having 3 sockets, each with a “ON” and “OFF”, the 6 commands and arrays look like this:

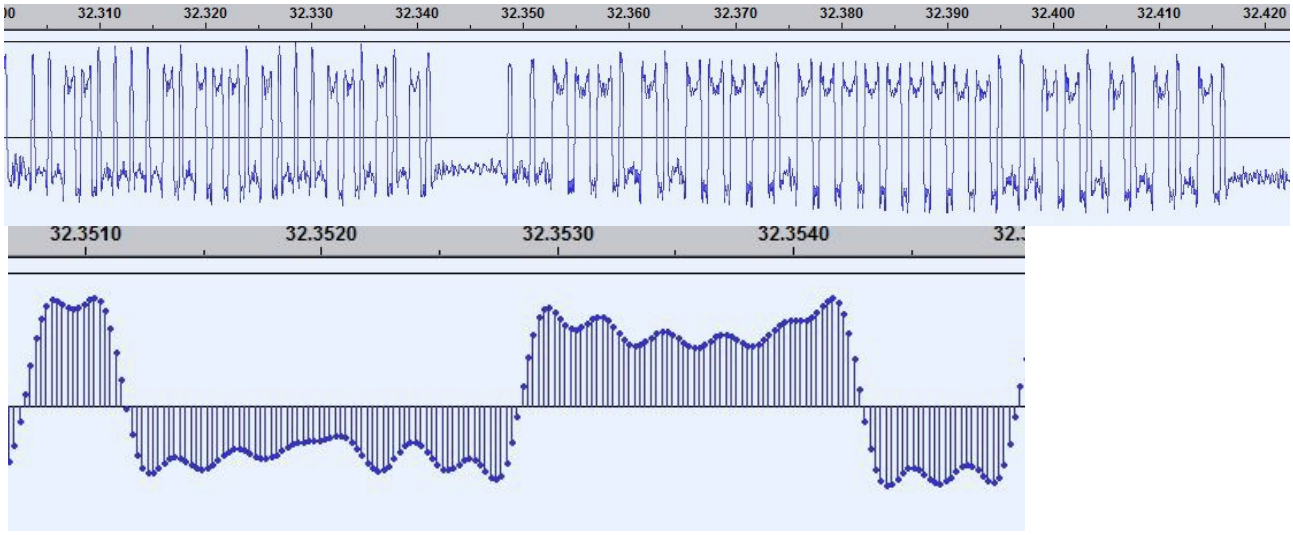
```
bool S24[6][24] =   {{0,0,1,1,1,0,1,1,1,1,0,0,0,0,1,1,0,1,1,0,1,1,0,0},   //  A_ON
                     {0,0,1,1,1,1,1,1,0,0,1,1,1,0,1,1,0,0,1,1,1,1,1,0,0},   //  A_OFF
                     {0,0,1,1,0,0,0,0,1,0,1,1,1,0,1,0,0,0,1,1,0,1,0,1},   // B_ON
                     {0,0,1,1,0,1,1,1,0,0,0,0,1,1,0,1,0,1,0,0,0,1,0,1},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

bool S32[6][32] =   {{1,1,1,1,0,0,0,1,0,1,0,1,1,1,1,1,1,1,1,1,1,0,1,0,1,1,0,1,0,0,1,1},    // 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
```

Apart from the bit patterns you should start each pattern with a long LOW part to make the patterns distinguishable. The timing of these LOW parts at the start is important and shouldn't deviate to much from the original.
During the running of the program, I call the same function to execute the operation of the sockets and only have to supply the required command number:

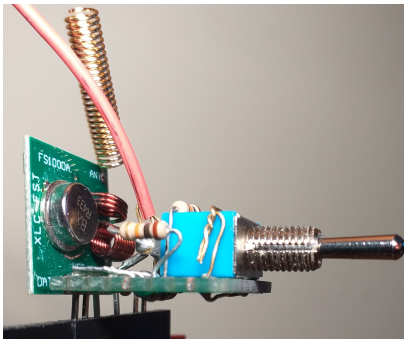
```
void switchRC(int commandNumber)
{
 int bitNumber;
 int bitPattern;
 for (bitPattern = 0; bitPattern < 6; bitPattern++)  // 24 bit part 6x
 {  // start with the extra long LOW part
  digitalWrite(pinRCdata, HIGH);
  delayMicroseconds(StartDelay24LowUp);
  digitalWrite (pinRCdata, LOW);
  delayMicroseconds(StartDelay24LowDown);
  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);
   }
  }
 }
}
```

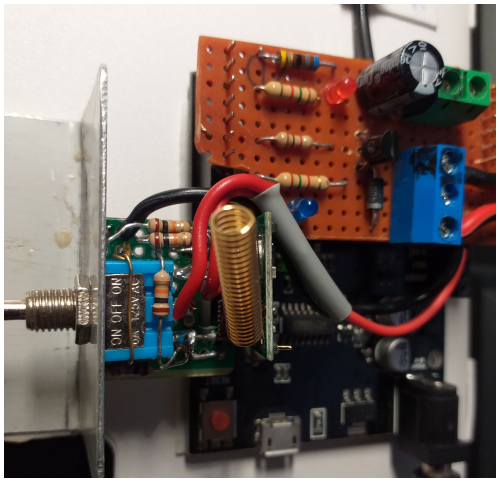
The computer simulated commands and bit patterns look like this:



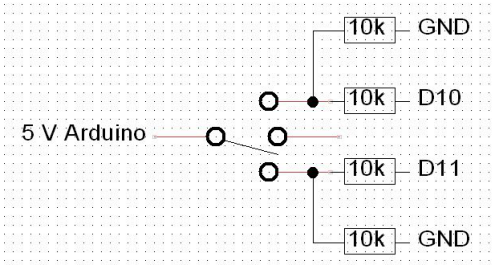
They work fine for all the sockets.

Adding the transmitter to the earlier described setup
After straightening the 5V-pin of the little transmitter shield, its GND and data pin fit straight into GND and D12 of the Arduino The 5V-in of the transmitter is led to the 5V-out of the Arduino.
I also added a 3 way switch, to let me choose between “always on”, “solar controlled” or ” always off”. 5V is put on digital inputs D10 or D11 (via a 10k resistor and with a 10k pull-down resistor to GND). to let the program know what choice is made. The relay in the socket remains controlled via output D5.





Not pushed in yet to make the pins visible -
The two pins to the right are just for mechanical support and are not connected.
The 10 k pull-down resistors from D10 and 11 to GND are visible on the final setup to the right.



program listings (on separate download)

_240709_Solar_Switched_BoilerSocket.ino
Simple boiler socket switching

_240719_Solar_Switched_Boiler_Socket.ino
As above with 3-way switch “Always On”, ”Solar Regulated”, ”Always Off”

_240828_Solar_Switched_Sockets.ino
As above with RC Sockets switching along depending on settings

The programs are pretty much self explanatory and you can easily adjust the settings to your own situation and preferences.

Jeroen Droogh, bootprojecten@gmail.com