Snap circuits activities for 9 - 11

Energy efficiency and smart homes
This activity is about setting up a miniature house and maximise energy efficiency. Kids will be integrating miniature appliances connected to snap circuits, and program the snaps via arduino.
The activity is suitable for groups of children aged 9 to 14 yo.

Difficulty **Easy** | Duration **1 hour(s)** | Categories **Electronics, House** | Cost **20 EUR (€)**

Contents

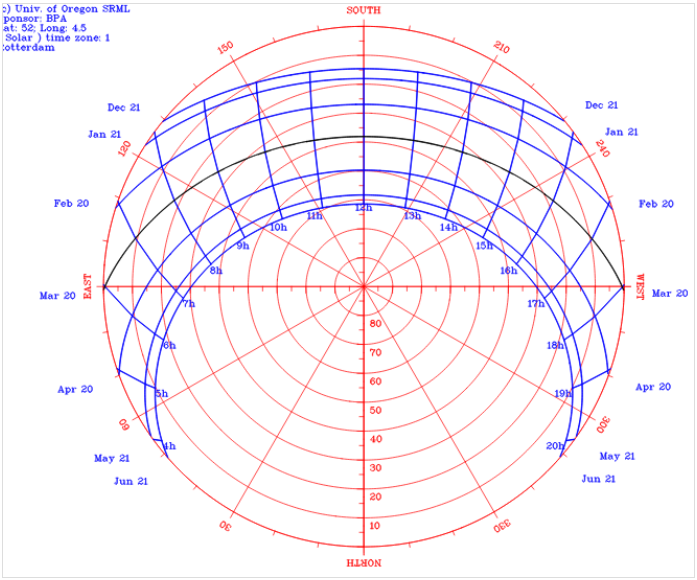
Introduction

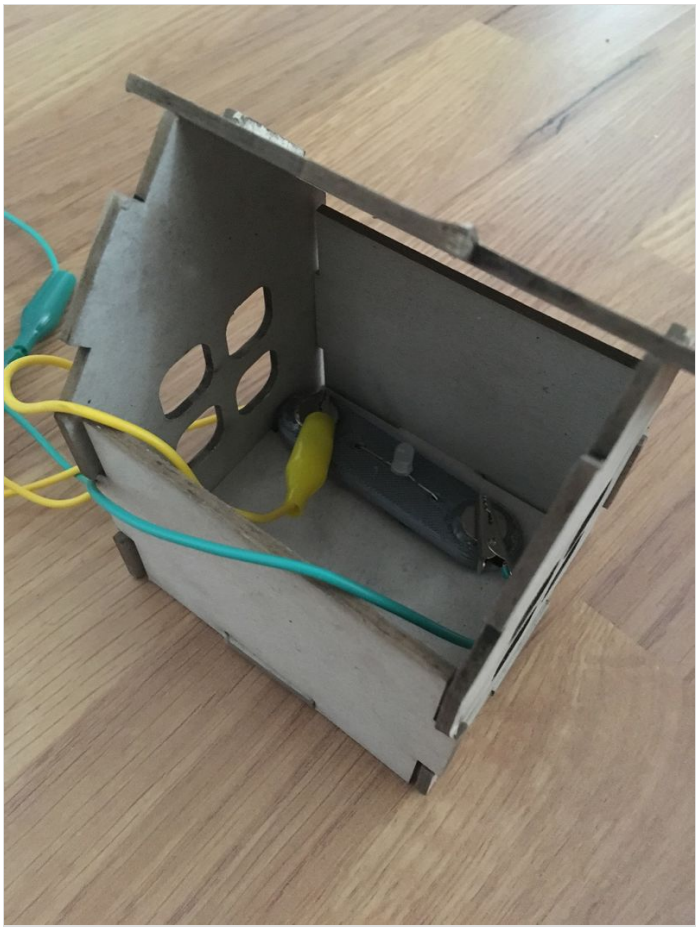
First of all, let's get clear on what energy efficiency is.
A warm living room in winter or a sports stadium brightly illuminated at night – we use energy to achieve a specific benefit. Energy efficiency is a means of measuring the energy-expenditure required to achieve a certain benefit. The lower the losses in energy to achieve a specific purpose are, the higher is the degree of energy efficiency.
Energy demand is increasing worldwide. The energy market situation is heating up and energy prices are on the rise. Instabilities in many exporting and transit countries are a cause for concern and the increased combustion of fossil energy sources is accelerating climate change. An expansion of energy supply options is costly and will take time. On the other hand, increasing energy efficiency curbs energy prices, reduces dependency on energy imports, counteracts energy distribution conflicts and cuts climate-damaging carbon dioxide emissions.
So how can we achieve energy efficiency in our own living environment?
Consider the Known Conditions: Sun, Wind and Light
The orientation of your home is one of the factor that can impact on energy efficiency. Obtain a sun path diagram for your site's location. This will help you determine the orientation of your home by giving a visual of where the sun travels in the sky throughout the day.
Generally speaking, the best tip for minimizing energy consumption is to orient the home facing south to capture solar gain in the winter and block solar gain in the summer.
So how do you read a sun path diagram?
The image above is a sun path diagram for the city of Rotterdam, in the Netherlands.
We can tell that in Rotterdam, on June 21st, at 20h, the sun is located at 310°, which is to say very close to the geographical north. On April 20th, at 18h, the sun is located at about 280°, which is to say towards the geographical west.
Also, it's good to keep in mind that the sky is generally speaking three times brighter at the zenith than at the horizon, so if you live in a predominantly overcast area this is the perfect environment to let in that bright light from above and maximize free illumination.
Other factors to take into consideration in order to maximize energy efficiency are directly related to the appliances you use.
Here's a few tips:

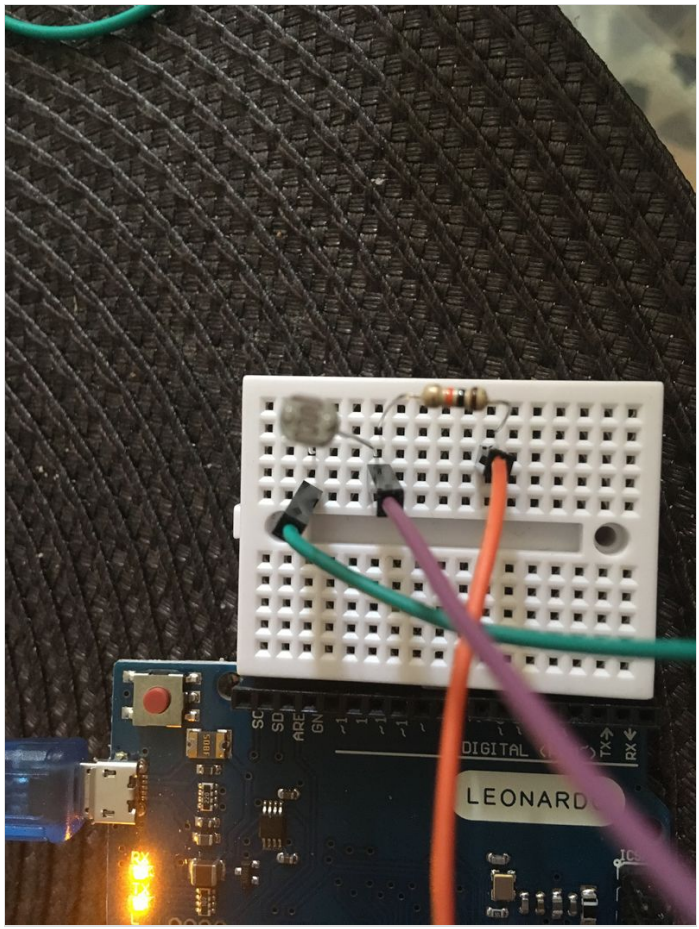
- use smart appliances, for example lights bulbs that go on at night and automatically turn off during the day
- use smart plugs that can be programmed to turn on and off at specific times, that can be remotely controlled via wi-fi, or that display the energy consumption (ex. in kwh) in real time.

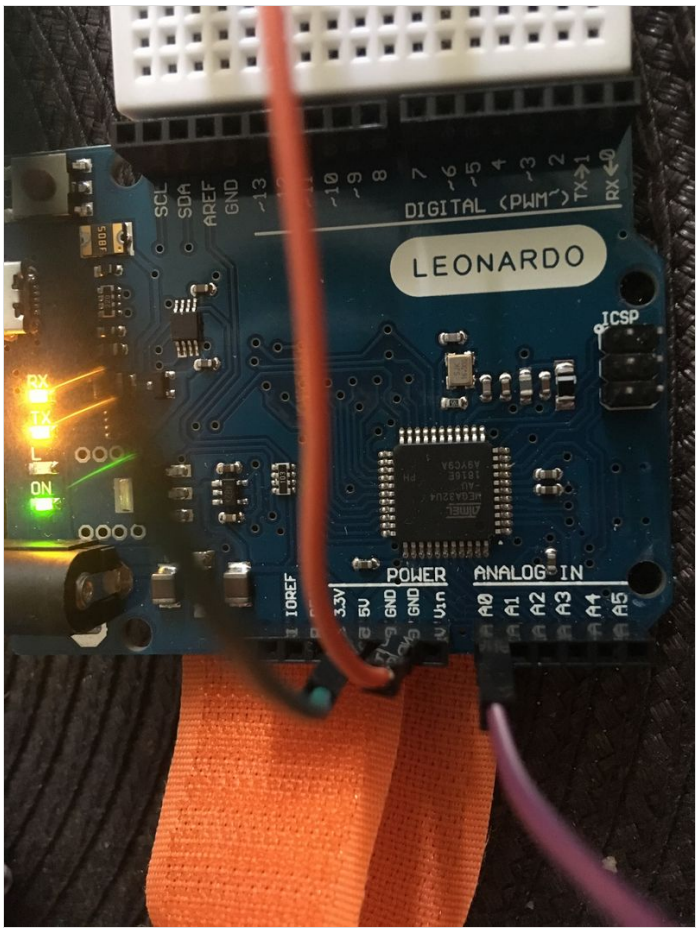
Step 1 - Setting up the miniature smart house

Kids will be using arduino uno boards and snap components to work on the energy efficiency of their miniature house.
First of all, have the kids set up their miniature houses. They can build one using cardboard, or you can laser cut them in advance, using for example a 3mm thick MDF board. Here's the design of a miniature house, ready for laser cut.
Next, ask the participants to place at least one snap LED component inside the house.
At this stage, they will need to program the LED(s) they have installed in the house via arduino, specifically program them to go **on during the day** (that is if a certain light threshold is attained) and **off at night** (that is to say if the amount of light in the environment falls below a given threshold).
They'll need to use an LDR in connection with the LEDs. The LDR can be mounted on a snap support, or simply used as it is.



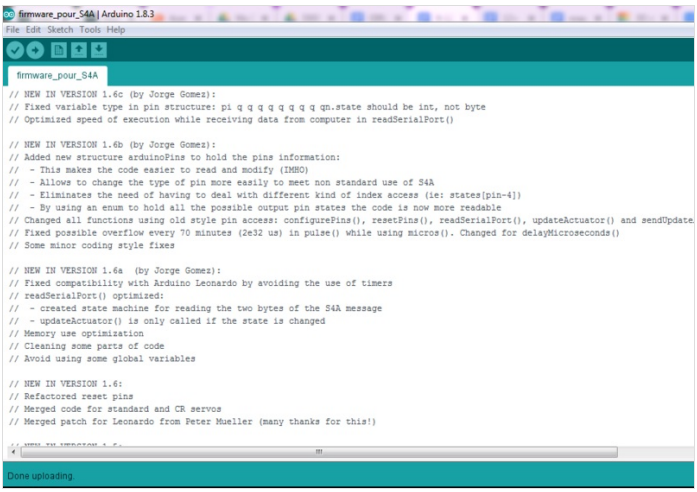


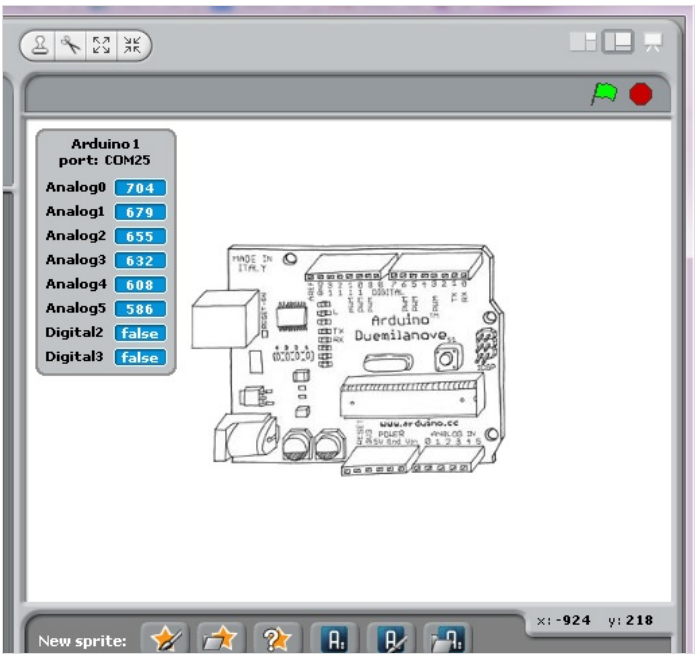

DIGITAL
LEONARD

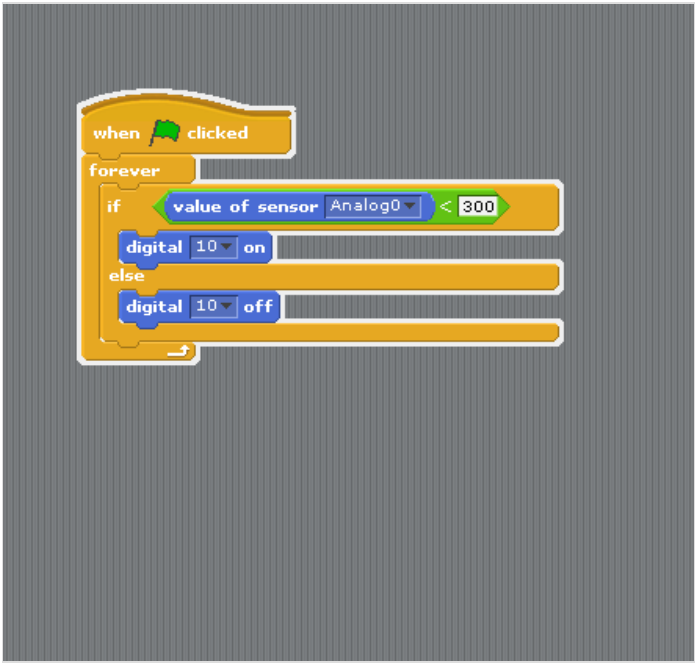


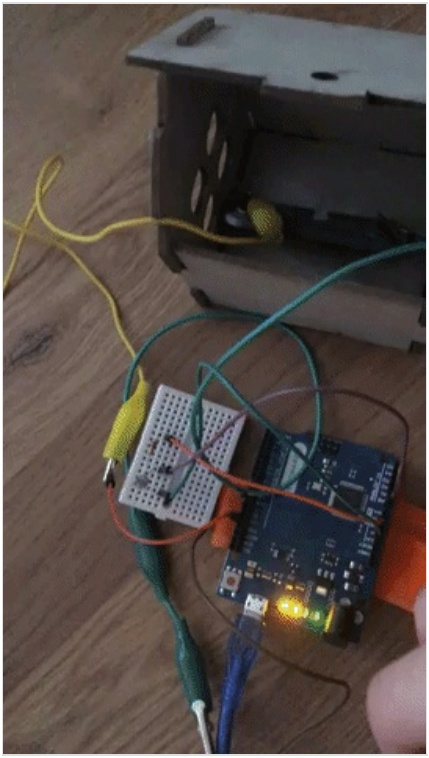
Step 2 - Programming

Instruct the kids that they need to Use Scratch for Arduino to achieve this programming part.
First of all, we need to prepare the arduino board. To do so, you need to upload the S4A firmware onto the board, via arduino IDE.
Launch Arduino IDE and open this code.
Upload the code onto the board.
Next, open Scratch for arduino and wait until the software recognizes the board.
Now program Scratch for Arduino to obtain something like this:
The Led inside the miniature house should go on if the LDR records an amount lower than 300. Otherwise it should stay off.
n.b. you can visualize in real time the amount of light detected by the LDR just by looking at analog 0 under this dialogue window
At this stage, kids can add extra appliances in their house, for example a miniature Aircon device. The main support for the miniature Aircon device can be a vibrating motor snap or an LED snap. Have them place the device inside the miniature house. Next, instruct them to use a temperature /humidity sensor to program the device so that the aircon goes on whenever the temperature rises above, say 28 degrees, or the humidity exceeds say 70%.


firmware_pour_S4A | Arduino 1.8.3
File Edit Sketch Tools Help
firmware_pour_S4A
// NEW IN VERSION 1.6c (by Jorge Gomez):
// Fixed variable type in pin structure: pi q q q q q q q qn.state should be int, not byte
// Optimized speed of execution while receiving data from computer in readSerialPort()
// NEW IN VERSION 1.6b (by Jorge Gomez):
// Added new structure arduinoPins to hold the pins information:
// - This makes the code easier to read and modify (IMHO)
// - Allows to change the type of pin more easily to meet non standard use of S4A
// - Eliminates the need of having to deal with different kind of index access (ie: states[pin-4])
// - By using an enum to hold all the possible output pin states the code is now more readable
// Changed all functions using old style pin access: configurePins(), resetPins(), readSerialPort(), updateActuator() and sendUpdate
// Fixed possible overflow every 70 minutes (2e32 us) in pulse() while using micros(). Changed for delayMicroseconds()
// Some minor coding style fixes
// NEW IN VERSION 1.6a (by Jorge Gomez):
// Fixed compatibility with Arduino Leonardo by avoiding the use of timers
// readSerialPort() optimized:
// - created state machine for reading the two bytes of the S4A message
// - updateActuator() is only called if the state is changed
// Memory use optimization
// Cleaning some parts of code
// Avoid using some global variables
// NEW IN VERSION 1.6:
// Refactored reset pins
// Merged code for standard and CR servos
// Merged patch for Leonardo from Peter Mueller (many thanks for this!)
Done uploading.


Arduino 1
port: COM25
Analog0 704
Analog1 679
Analog2 655
Analog3 632
Analog4 608
Analog5 586
Digital2 false
Digital3 false
Arduino
Duemilanove
New sprite:
x: -924 y: 218


when clicked
forever
if value of sensor Analog0 < 300
digital 10 on
else
digital 10 off




Arduino 1
port: COM25
Analog0 716
Analog1 691
Analog2 666
Analog3 643
Analog4 619
Analog5 597
Digital2 false
Digital3 false