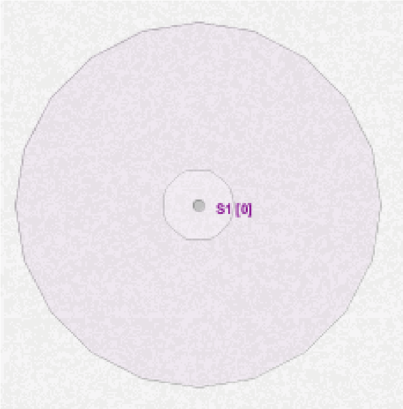
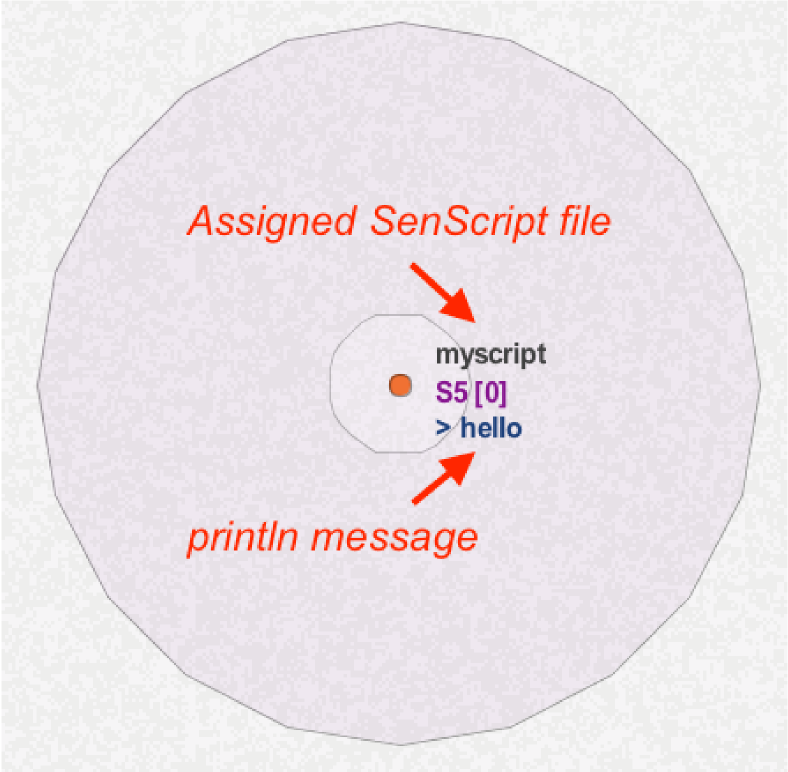
CupCarbon Simulator

CupCarbon

- The sensor node is the main object of CupCarbon ! It contains four main parts :
- radio modules, a sensing unit and a battery.
- A sensor node can be added by clicking on the menu Add → Add Sensor Node and then by clicking on the map, in the place where the sensor node must be added.
- To stop adding sensor nodes, just click on the right button of the mouse, or by typing the escape [esc] button of the keyboard.
- In the center of the sensor node, we find the name S followed by its id.
- For example, if its id is equal to 4, then its name will be S4.
- In the right part of the name, we find a number situated between brackets which is equal to [0] by default.
- This number represents the MY address of this sensor node.

CupCarbon





Assigned SenScript file

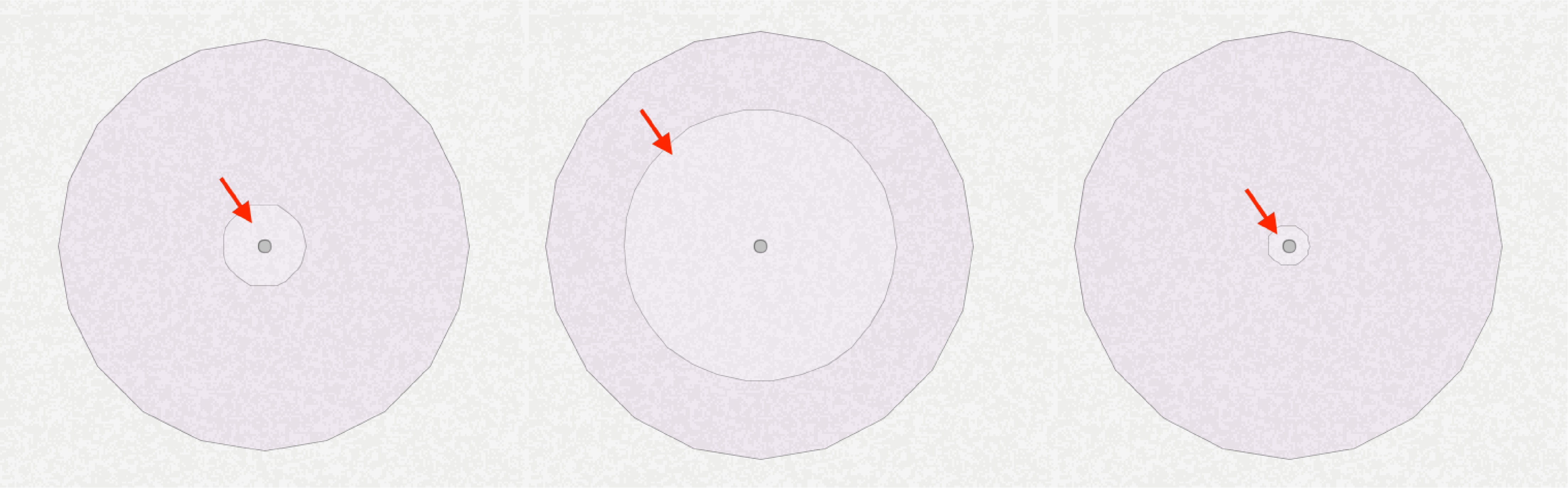
println message

If a SenScript is assigned to it then it will be displayed in a gray color above its name. The print messages will be displayed in blue bellow its name.

CupCarbon

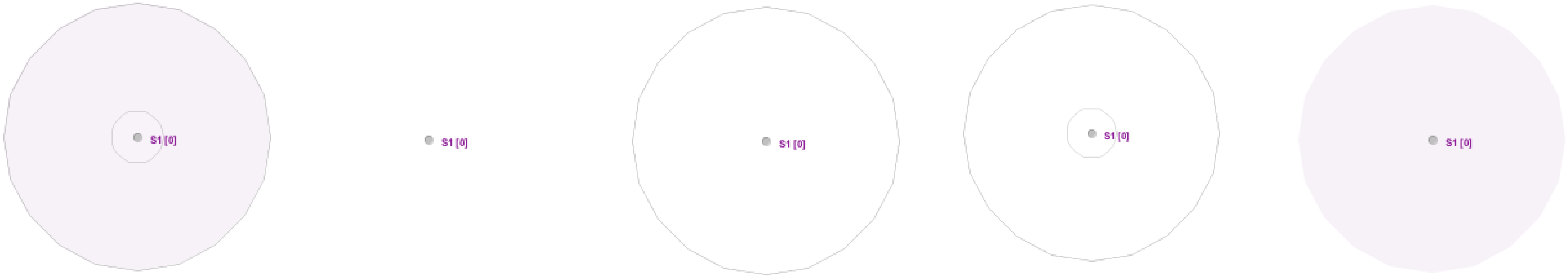
- A sensor node can contain many radio modules with different or the same standards (802.15.4, WiFi or LoRa).
- However, for the communication, only one radio module is considered.
- To change the considered radio module, we select the name of this radio on the Radio Module List of the **Radio Parameters** view situated in the left of the CupCarbon environment (4).
- Then, one click on Current.
- The radio range circle displayed in the purple color represents the radio range of the current radio. This color changes to blue for the LoRa standard and to green for the Wi-Fi standard.
- One can change the radio range using the keys ‘+’ and ‘–‘ of the keyboard.
- A sensor node contains a sensing unit represented by transparent white circle.
- The radius of the area can be changed using the buttons ‘)’ for increasing the radius and ‘(‘ for decreasing the radius.

CupCarbon



CupCarbon

- Other keys of the keyboard can be used to modify some parameters of the sensor node, like, ‘d’ to show/hide the name of the sensor node, ‘D’ to show/hide the message displayed by the sensor node, ‘n’ to show/hide the assigned SenScript file name.
- The key ‘b’ show/hide the battery/buffer levels.
- The key ‘h’ allows to hide some parts of the sensor node.



CupCarbon

- By typing many times we hide in each time the following parts :
- 1. First time: all the parts except the center
- 2. Second time: shows only the radio range circle of the current radio module
- 3. Third time: shows only the radio range of the current radio module and the sensing unit
- 4. Fourth time: shows only the sensing unit
- 5. Fifth time: shows only the radio range area of the current radio module
- 6. Sixth time: coming back to the initial drawing of the sensor node (with all the parts)

CupCarbon

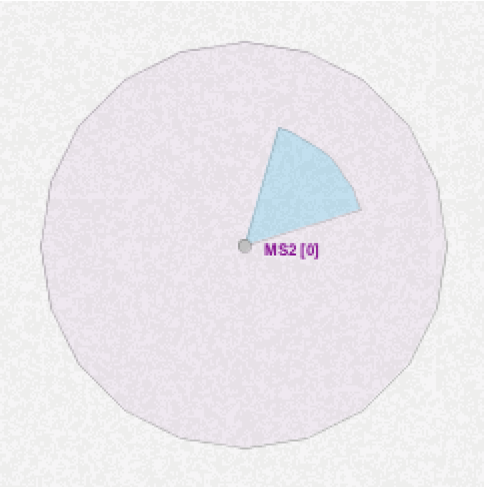
- The key ‘r’ allows to show directly on the sensor node some parameters (name, id, my address, network address, channel, SenScript file name, route file name, and the battery level).

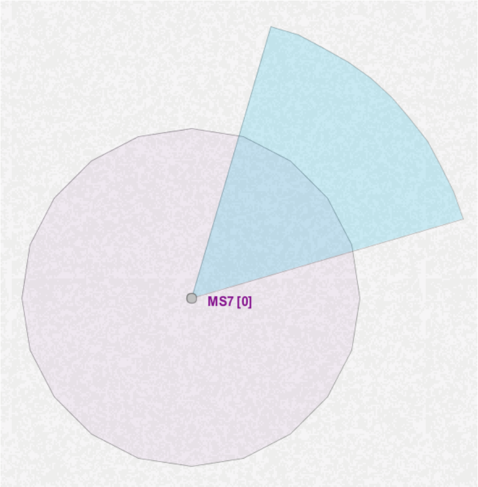
- If the sensor node is mobile, it is possible to test its mobility by clicking on the key ‘s’.
- To stop the mobility, click on ‘q’.
- To duplicate a sensor node, use the key ‘c’.
- Finally, to kill a sensor node, use the key ‘k’.
- Clicking a second time on ‘k’ will lead to the initial sensor node.

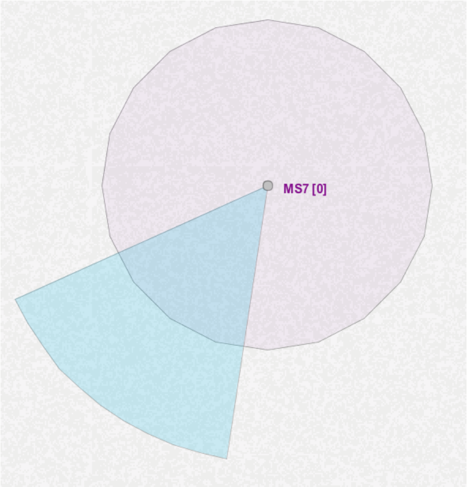
CupCarbon

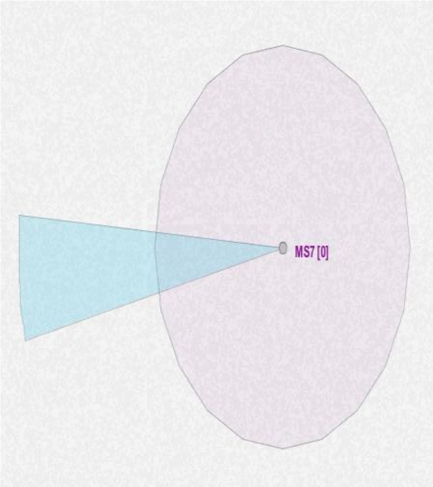
- **Directional Sensor Node**
- The Directional Sensor Node is the same as the classical sensor node with another type of sensing unit, which is directional.
- This last is not circular, it has a form of a cone that can be modified with the SenScript and manually using the following keys of the keyboard:
- 1. ‘)’ and ‘(‘: increase or decrease the sensing unit range (distance). Use ‘]’ and ‘[‘ for more precision.
- 2. ‘p’ and ‘o’: rotate (left or right) the sensing unit range. Click in addition on ALT for more precision.
- 3. ‘P’ and ‘O’ : increase or decrease the areas (angle) of the sensing unit.
- Click in addition on ALT for more precision.

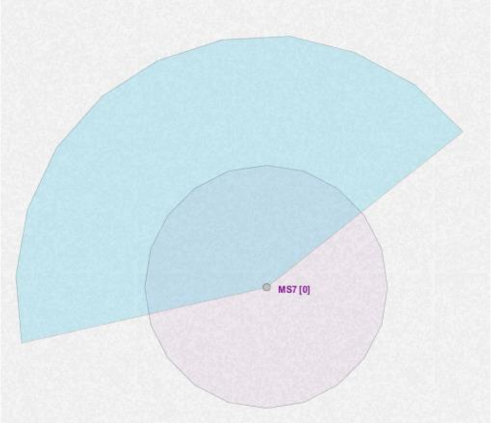
CupCarbon





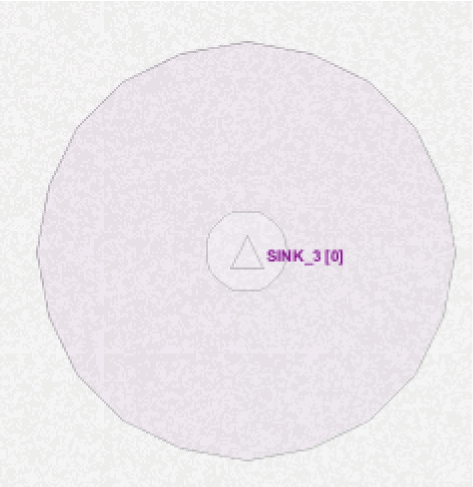






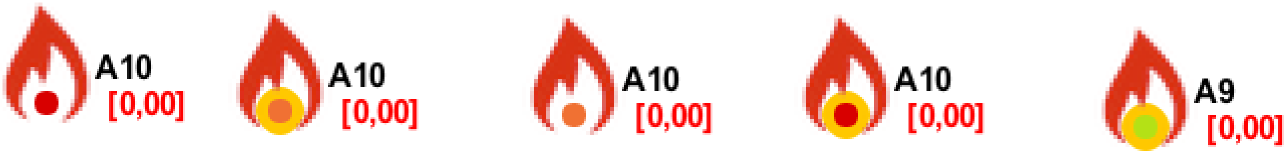
CupCarbon

- **Base Station (Sink)**
- The base station is exactly the same object than the sensor node with the exception that it has an infinite battery.



CupCarbon

- **Gas (Analog events)**



- The Gas or the natural event allows to generate analog values (the value is displayed in the red text). It can be mobile.
- Its objective is to simulate random or given values coming from the environment.
- It is possible to use the Natural Event Generator in order to generate random values based on the Gaussian distribution.
- To take into account the event during the simulation process, the (Mobility/Event) box must be activated in the Simulation Parameters panel.
- A red center means that the event is not mobile, otherwise it will be in orange color.
- The second white circle around becomes yellow when a natural event file is assigned.
- During the simulation, if the event is mobile, then the center becomes green.

SenScript

- SenScript is the script used to program sensor nodes of the CupCarbon simulator. It is a script where variables are not declared and without types, but they can be initialized (set command).
- It is possible to use the instruction function to add complex and additional functions programmed in Java.
- **Example:**
- SenScript: **set x "abcd"** → Java: **x = "abcd";**
- SenScript: **set y x** → Java: **y = x;**
- SenScript: **set z x+y** → Java: **z = x+y;**

SenScript

- **inc / dec**
- **int**
- **max / min**
- **and / or**
- **xor / not**
- **for end**
- **if [else] end**
- **while end**
- **goto**
- **function**
- **math**
- **set**

- **and (logic)**
- **`and x a b`** → x = a & b
- **areadsensor (Analog Read Sensor)**
- **`areadsensor s`**
- s=X or s=S#y#v, where X means that there is no sensed value (or event), S means that a sensed value has been read from the event having id=y, where the read value is equal to v (use the function rdata function to read these values).
- If there is many events then s=S#s1#v1#s2#v2#S3#v3… (use rdata function to read these values).
- Note that, s2, s3, … can be null if there is no many events.

SenScript

- **atch**
- `atch 12` → Change the channel of the current sensor to 12
- **atid**
- `atid 1111` → Change the identifier of the sensor to 1111
- **atpl**
- `atpl 60` → Change the signal power (power level) of the radio of the sensor. In this example, the sensor will use only 60% of its radio range.

- **atmy**
- `atmy 4` → Change the my of the sensor to 4.

SenScript

- **`atnd n`** → n = the number of the neighbors of the sensor.

- **`atnd n v`**
- → n = the number of the neighbors of the sensor and v the vector of the identifiers of each neighbor (use vget to get the values of v)
- **atget**
- **`atget id x`**
- **`atget my x`**
- **`atget ch x`**
- → x = id, my or ch of the sensor node

SenScript

- **battery**
- **`battery x`**
- → x = the level of the battery in Joules.
- **`battery set x`**
- → The level of the battery will be set to x

SenScript

- **dec**
- **`dec x`**
- → x = x - 1
- **delay**
- **`delay 1000`**
- → wait 1 second (1000 ms) before the next instruction
- **distance**
- **`distance x 2`**
- → x = the Euclidean distance between the current sensor and a communicating sensor node having an id=2.
- **dreadsensor**
- **`dreadsensor x`**
- → x=1 if the sensor detects an event (mobile), x=0, otherwise

SenScript

- **charat**
- **`charat c hello 1`**
- → c = ‘e’
- the character situated in the index 1 of hello
- **conc**
- **`conc x a b`**
- → concatenate a and b and assign it to x (=ab). In java this command is written as `x = “a”+”b”;`
- **`conc x a b`**
- → concatenate the value of the variable a with the value of the variable b

SenScript

- **cprint**
- `cprint "hello" x`
- → The same function as print where the result is displayed in the console
- **data**
- `data p 1 4 6`
- → p = 1#4#6

SenScript

- **for end**
- `for i 0 10`
- `print "i = " i`
- `delay 1000`
- `end`
- → the same as the following for on C: for(int i=0; i<10; i+=1)
- `for i 0 10 2`
- `print "i = " i`
- `delay 1000`
- `end`
- → the same as the following for on C: for(int i=0; i<10; i+=2)

SenScript

- **if [else] end**
- `if (x==1)`
- `mark 1`
- `else`
- `mark 0`
- `end`
- `if ((x==1) && ((y>0) || (y<5)))`
- `mark 1`
- `else`
- `mark 0`
- `end`
- In the current version of CupCarbon, it is not possible to write: `if (x==1 && y>0) ...`
- One must write: `if ((x==1) && (y>0)) ...`

SenScript

- inc
- `inc x`
- → x = x + 1
- int
- `int x a`
- → assign to x the integer value of a (if a=3.2 then x=3)
- kill
- `kill 0.3`
- → the current node will be killed (i.e., with empty battery) with a probability of 30%