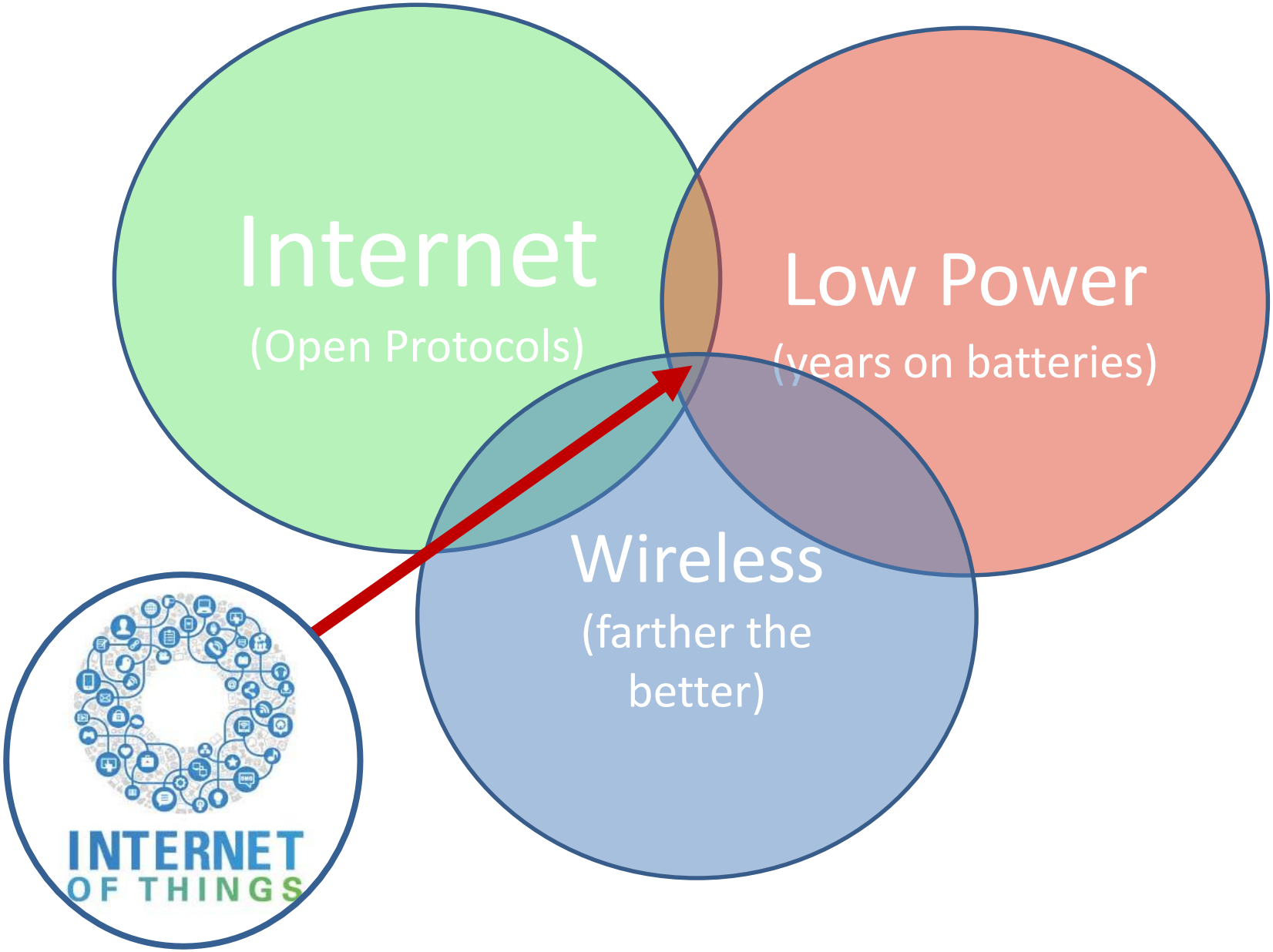
Contiki introduction

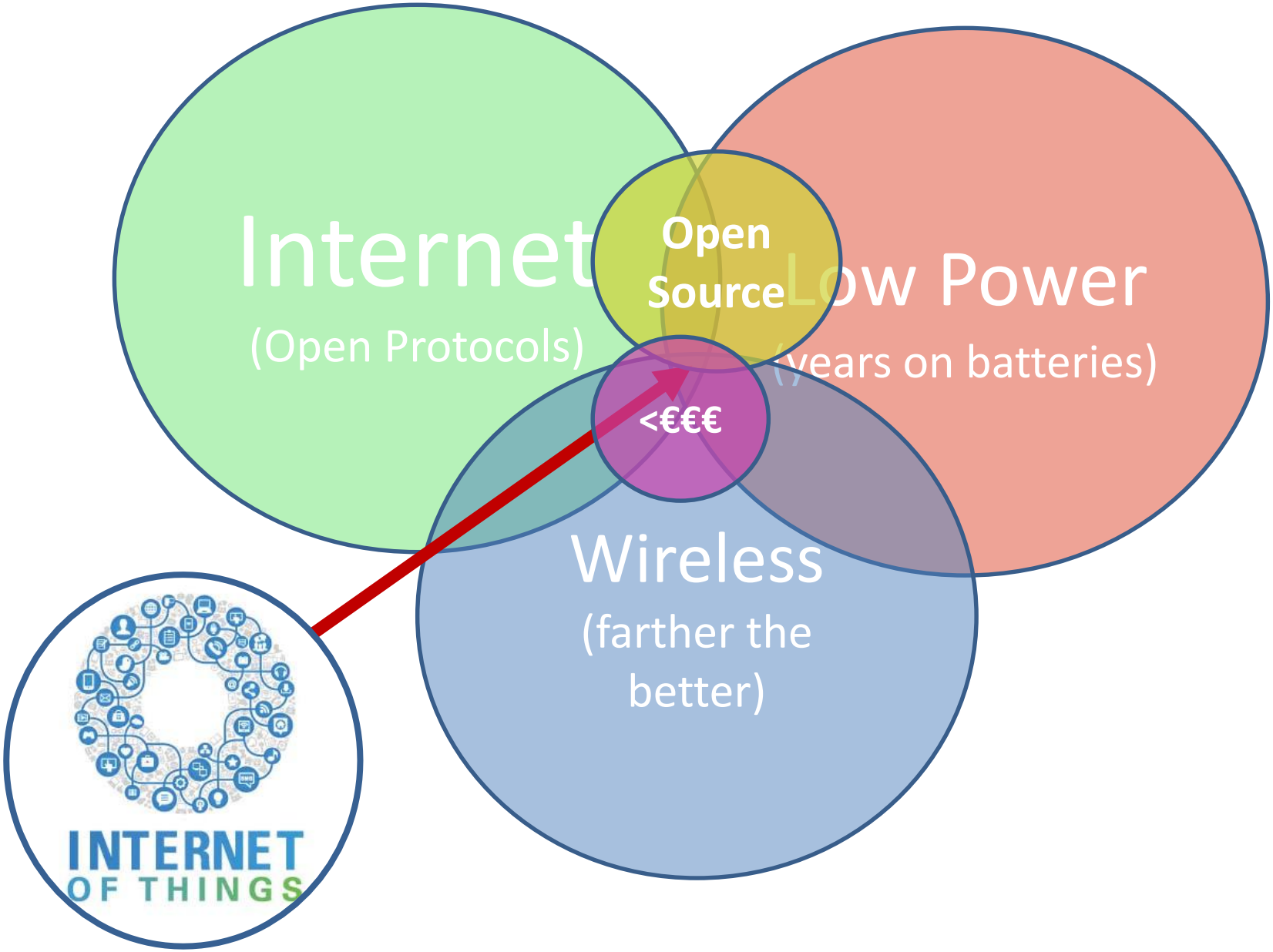
Antonio Liñán Colina,
Zolertia

BY NC SA

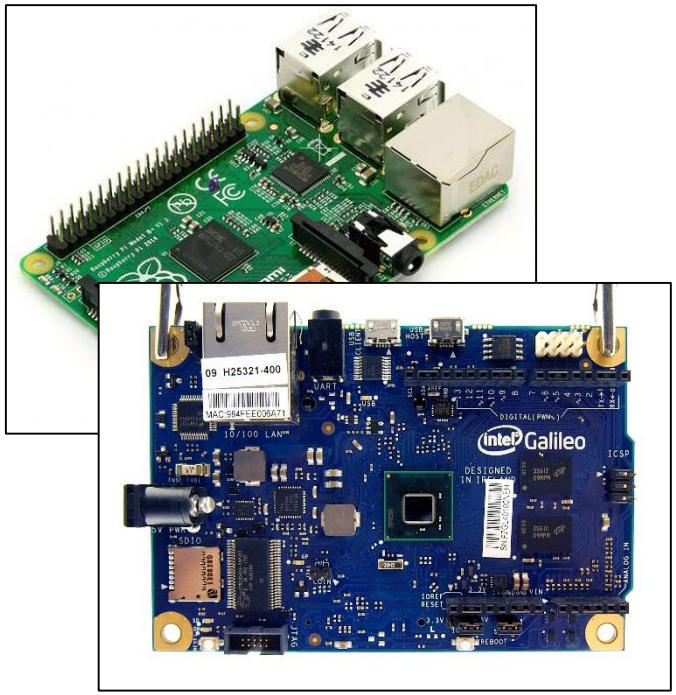
Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0



Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0



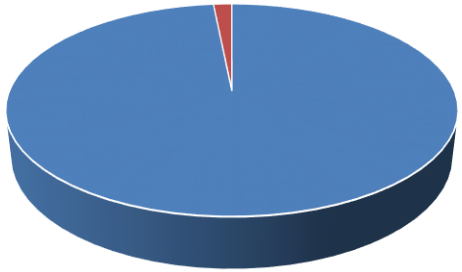
Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0



>

>

Processing power



Unused System Required

200-500 mA @ 5V → 1– 2.5 Watts
USB power bank 7Ah → 19.6 hours

DietPi build → 16MB RAM (1GB available)
Typical application: 50KB (0.005%)

http://fuzon.co.uk/phpbb/viewtopic.php?f=8&t=6
http://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-battery-life

Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0

GPRS GENERAL PACKET RADIO SERVICE

3G

Wi Fi™

<

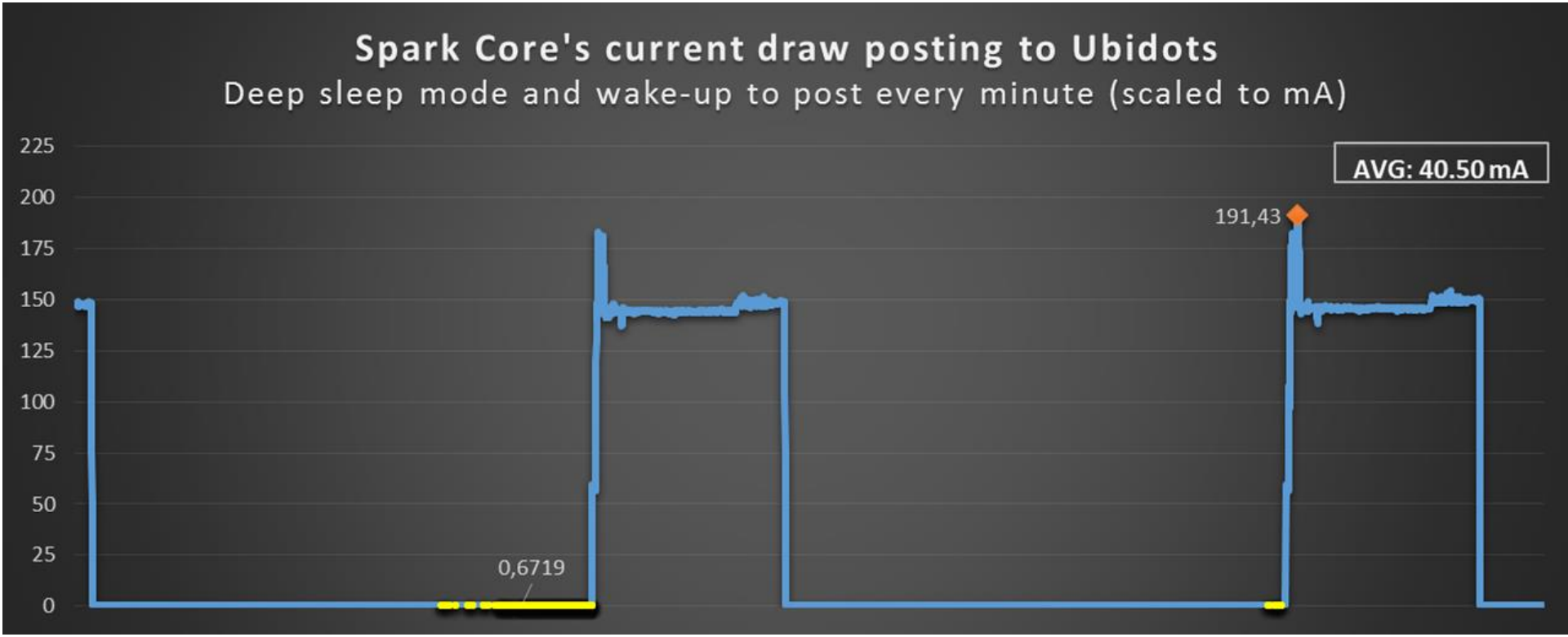
RFID

<

30m

http://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-battery-life

Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0

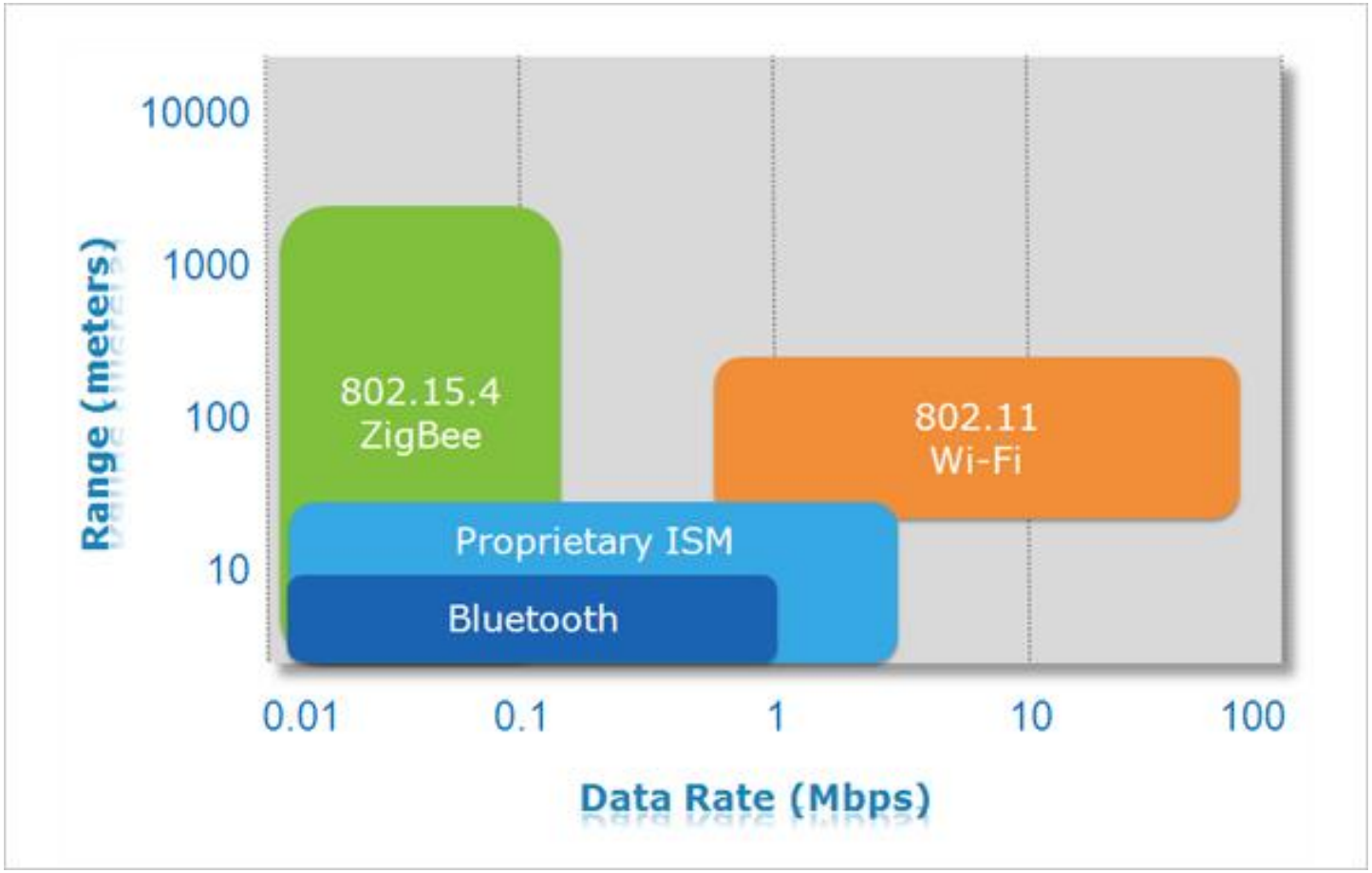


40.5 mA (avg) @ 5V → 225mW

USB power bank 7Ah → 121h (5 days)

http://www.ubidots.com/
https://www.particle.io/
https://www.hackster.io/4354/potato-powered-iot-947b69

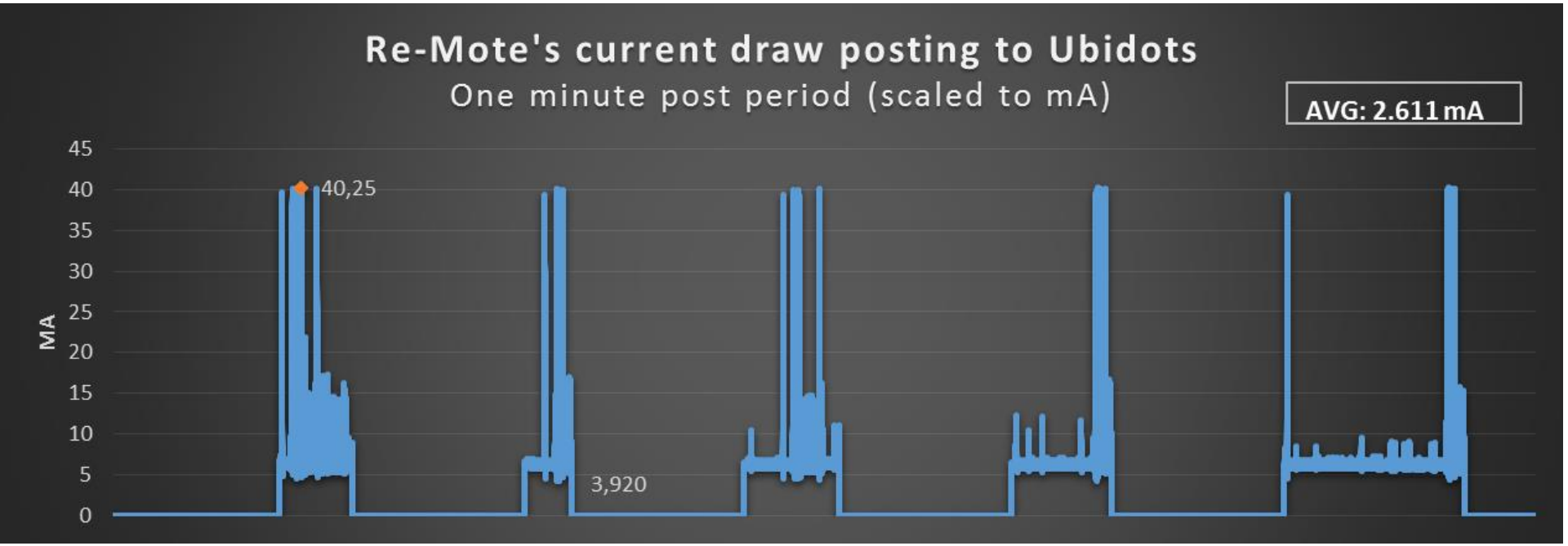
Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0



* Commercial solutions without special antennas

http://www.atmel.com/

Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0



2.6 mA (avg) @ 5V → 13 mW

USB power bank 7Ah → 1884h (78,5 days)

http://zolertia.io/

http://www.ubidots.com/

https://www.hackster.io/4354/potato-powered-iot-947b69

Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0

Zolertia RE-mote prototype A
868MHz, 3.14Km @50Kbps
IEEE 802.15.4g, 2dBi omni

http://zolertia.io/
https://ict-rerum.eu/first-long-range-test-with-the-rerum-re-mote-platform/

Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0

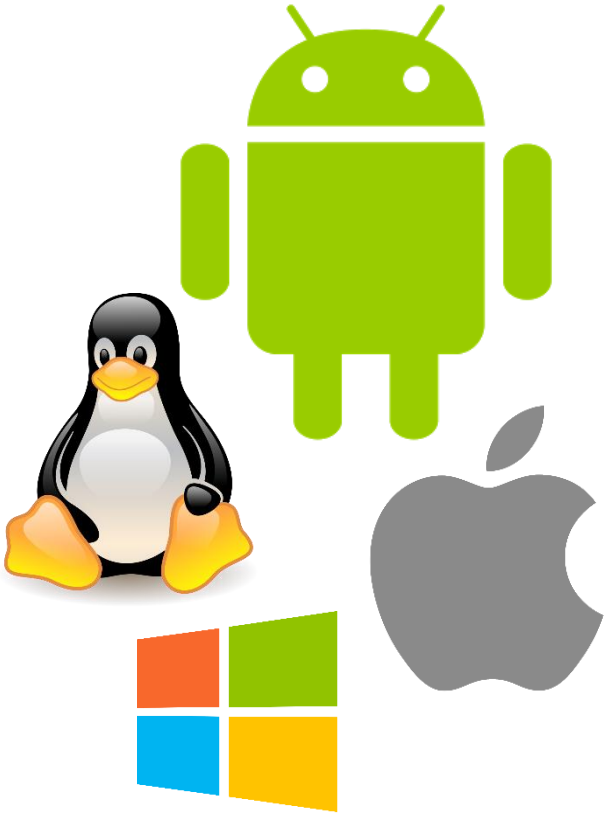
Zolertia Z1 mote
2.4Ghz, 284 m @250Kbps
IEEE 802.15.4
3dBi omni + 12dBi directional

http://zolertia.io/

Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0

RAM
ROM/FLASH
Processor

<



> 1MB RAM/ROM

Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0

Contiki

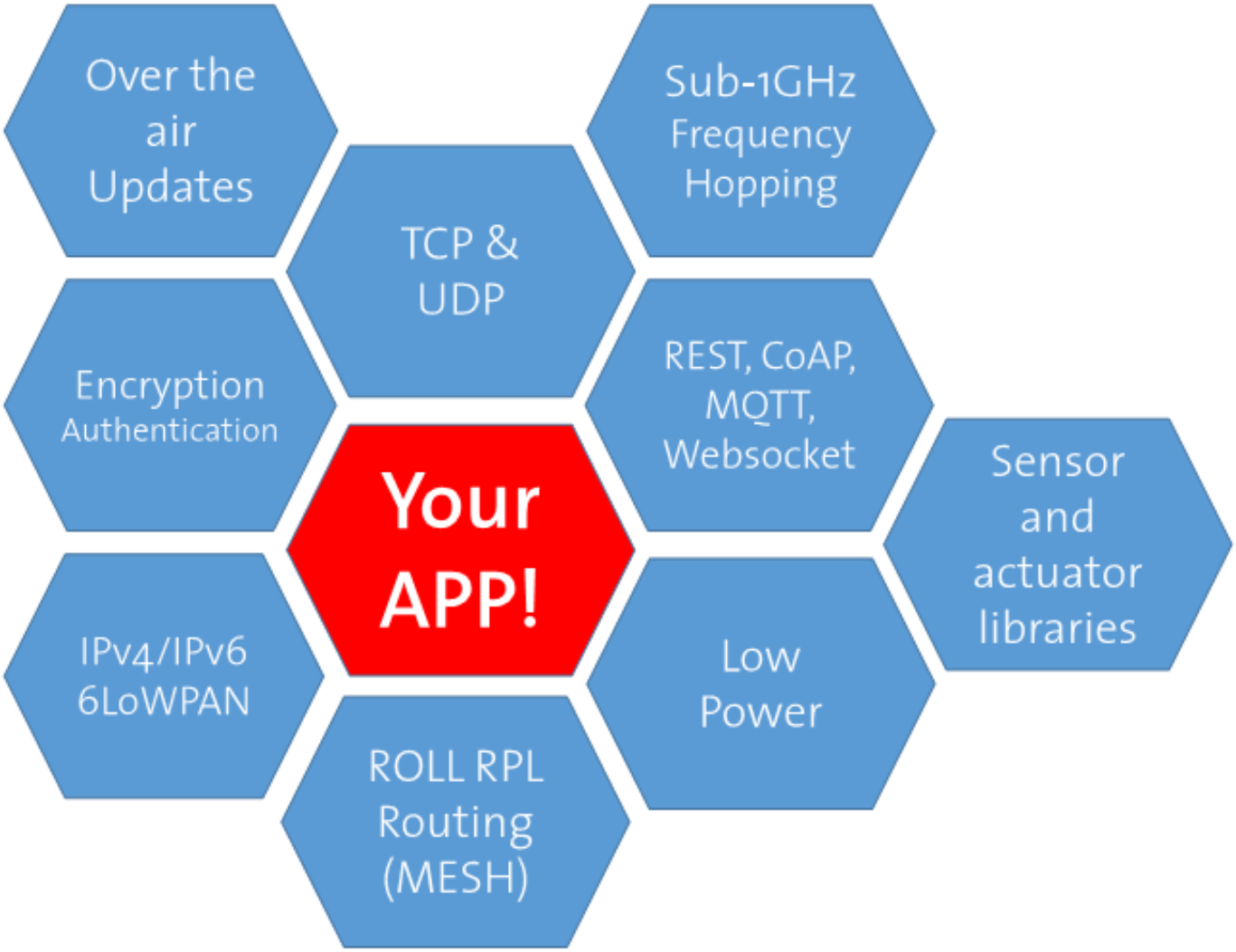
The Open Source OS for the Internet of Things

- Architectures: 8-bit, 16-bit, 32-bit
- Open Source (source code openly available)
- IPv4/IPv6/Rime networking
- Devices with < 8KB RAM
- Typical applications < 50KB Flash
- Vendor and platform independent
- C language
- Developed and contributed by Universities, Research centers and industry

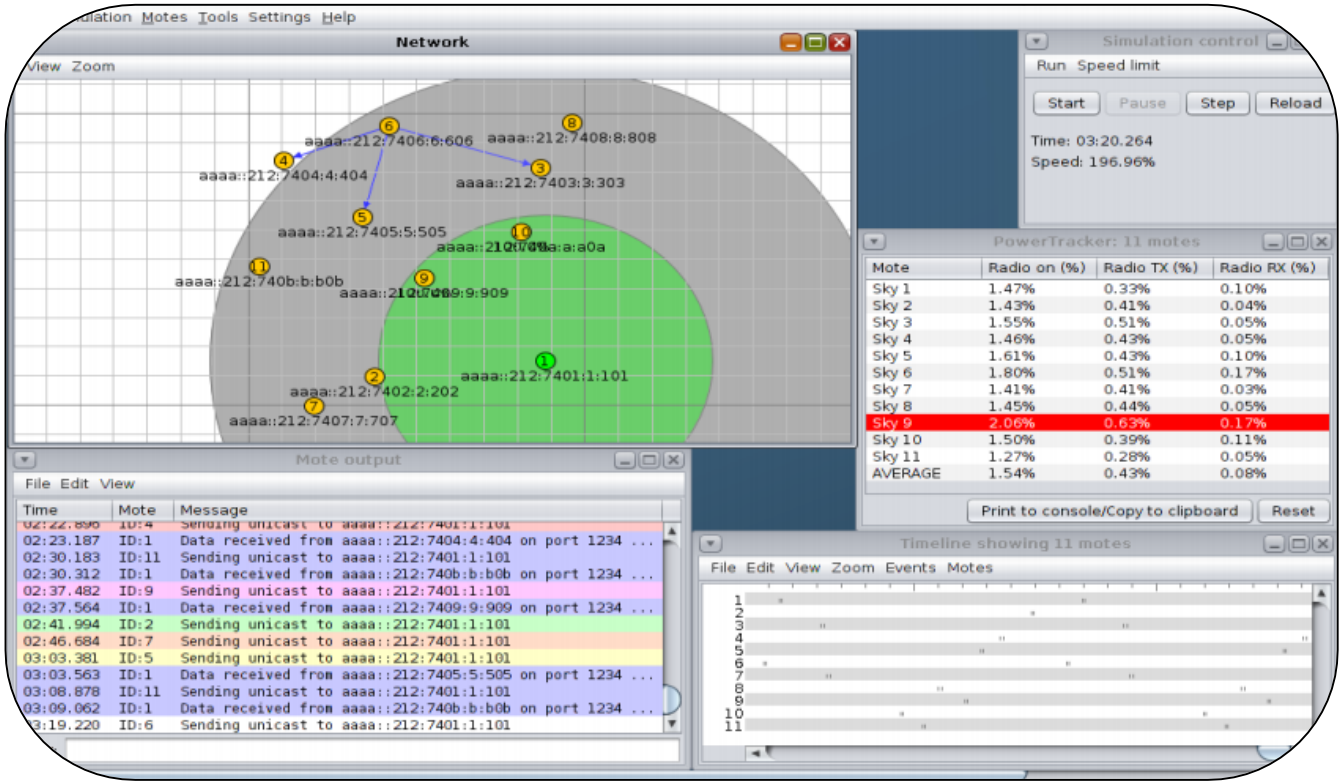
GCC

open source initiative®

Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0

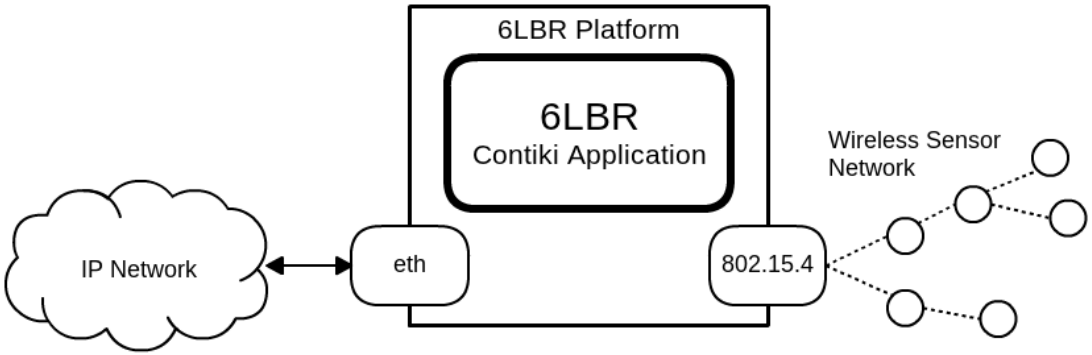


Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0



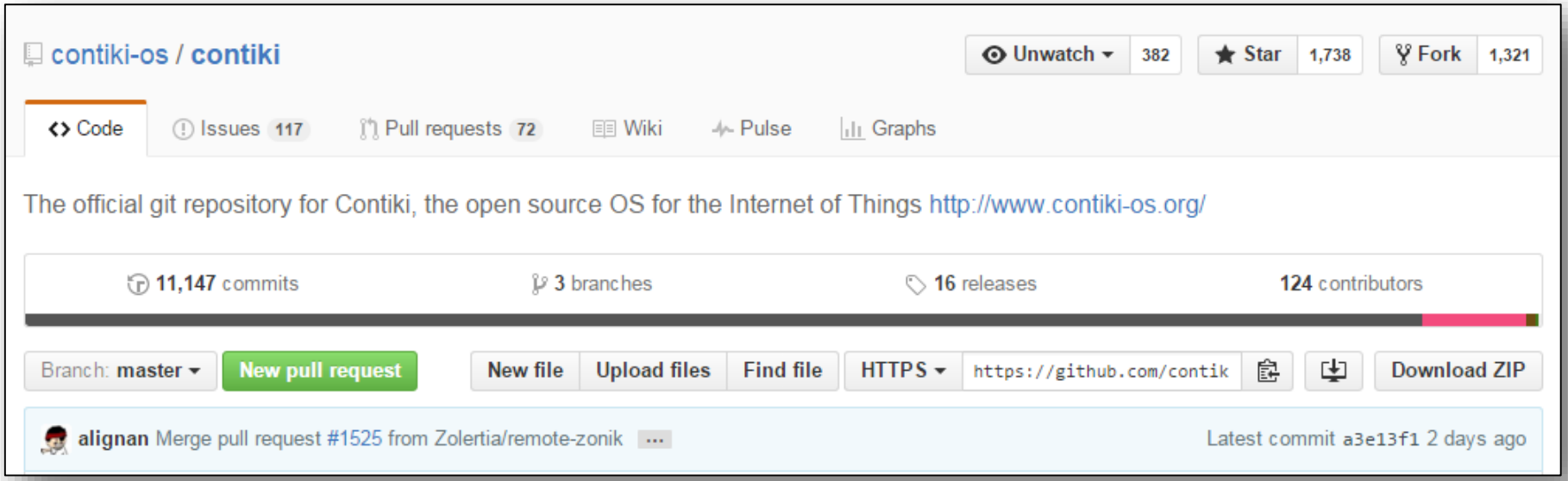
http://www.tado.com

http://www.lifx.com



http://cetic.github.io/6lbr/

Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0



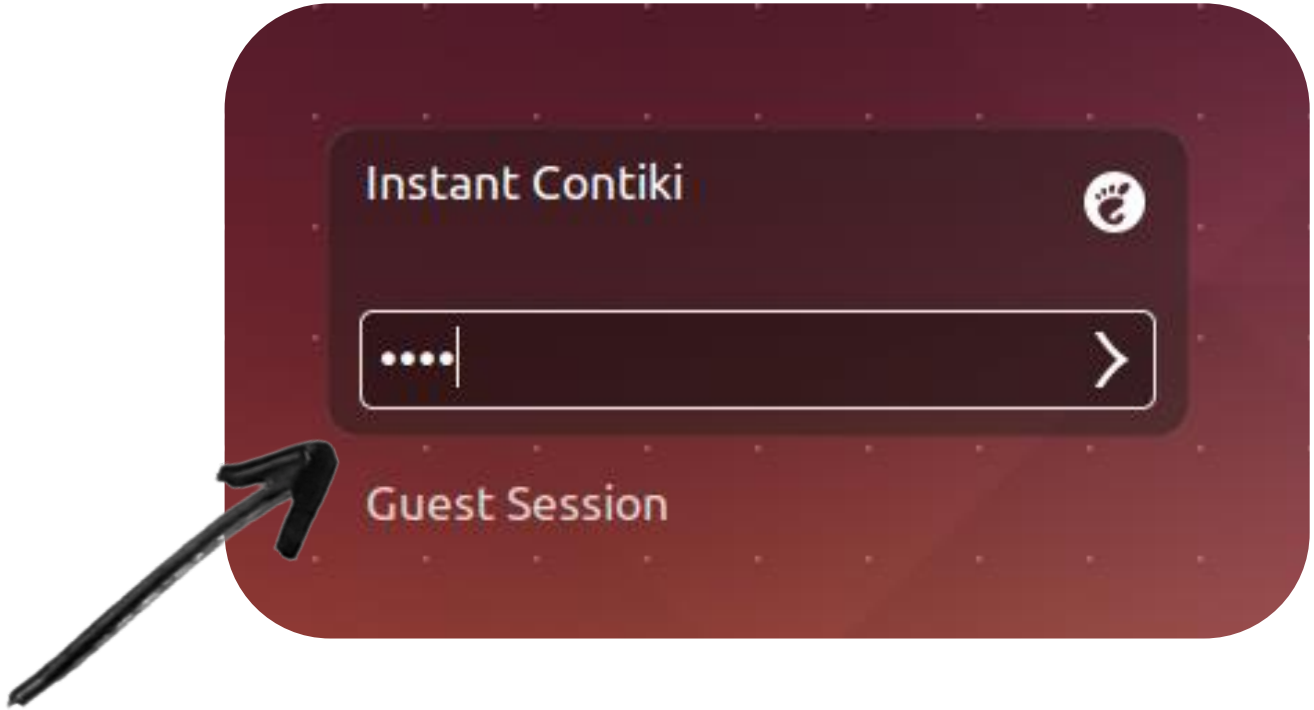
www.contiki-os.org

https://github.com/contiki-os/contiki

Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0

INSTANT CONTIKI

VMWare virtualized develop environment



user

http://www.contiki-os.org/start.html

Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0

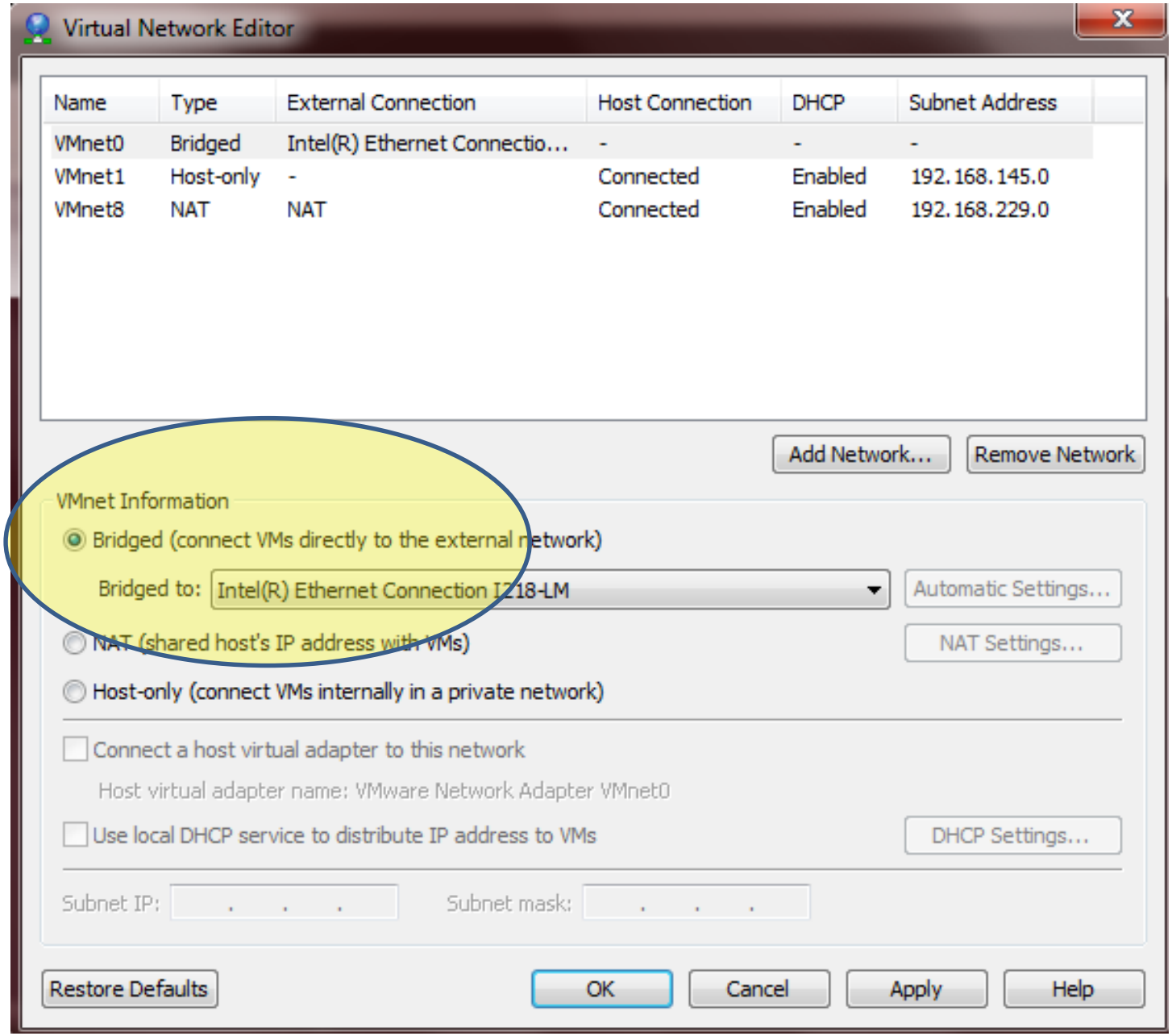
If you are using a laptop/PC and have a 32-bit Linux machine

```
sudo add-apt-repository ppa:wireshark-dev/stable
sudo apt-get –y install git git-core build-essential wireshark
git clone –recursive https://github.com/alignan/contiki
git checkout zolertia-tutorial
```

To install the toolchain (application to convert the source code into an image to program the Zolertia devices)

```
wget "https://sourceforge.net/projects/zolertia/files/Toolchain/msp430-47.tar.gz" -O
$HOME/msp430-47.tar.gz && tar -zxvf $HOME/msp430-47.tar.gz -C $HOME/msp430-47

sudo echo "export PATH=$HOME/msp430-47/bin:$PATH" >> $HOME/.bashrc && source
$HOME/.bashrc
```

Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0



Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0

alignan Added slides for day 2 — Latest commit `a8c2e47` just now

apps	Reworked simple UDP rx/tx example, added binaries and made servreg-ha...	4 days ago
core	ning-free compilation even with fragmentation disabled	6 days ago
cpu	#1293 from simonduq/pr/fix-warnings	7 days ago
dev	king branch 'alignan/z1_sniffer' into walc15	5 days ago
doc	Add support for the CC13xx CPU	3 months ago
examples		19 hours ago
platform		a day ago
regression-tests	Merge pull request #1293 from simonduq/pr/fix-warnings	7 days ago
tools		a day ago

MCU implementation (MSP430)

Devices implementation (radio, etc) (CC2420)

Examples (ipv6, Zolertia, etc)

Platform specific implementation (Z1, Zoul)

Tools (flashing, emulation, visualization)

Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0

alignan Added optional switch to compile for the Rpi — Latest commit `f5ee9e0` a day ago

..		
apps	s. The node-id.h file contains	3 years ago
dev		28 days ago
Makefile.common		a day ago
Makefile.z1	sing CONTIKI...	a year ago
Makefile.z1sp	Merges Z1SP into Z1 platform	a year ago
README.z1sp	Merges Z1SP into Z1 platform	a year ago
cfs-coffee-arch.h	Coffee no longer uses watchdog calls	5 years ago
contiki-conf.h		17 days ago
contiki-z1-main.c		2 months ago
contiki-z1-platform.c	Removed all old RCS tags in the Contiki source tree. Those RCS tags a...	3 years ago
node-id.c	Use the Z1 product ID as MAC/Node ID if no value is found in the XMEM	a year ago
platform-conf.h		a year ago

Specific Z1 applications

Specific sensors and actuators drivers

Makefiles (where the platform specifies the files to use and include)

Specific configuration for Contiki

Main Application, Z1 initialization when booting

Specific platform configuration (pin-out, peripherals)

Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0

Home / Toolchain

Name	Modified	Size	Downloads / Week
Parent folder			
README.txt	2015-11-28	1.5 kB	1
msp430-47-rpi.tar.gz	2015-11-28	129.7 MB	3
msp430-z1.tar.gz	2012-08-17	17.4 MB	2
msp430-47.tar.gz	2012-08-17	97.5 MB	4
msp430-46.tar.gz	2012-08-17	84.8 MB	1
msp430-gcc-4.4.5.tar.gz	2011-09-02	21.1 MB	5
Totals: 6 Items		350.4 MB	16

user@instant-contiki: ~/contiki

File Edit View Search Terminal Help

```
user@instant-contiki:~/contiki$ msp430-gcc --version
msp430-gcc (GCC) 4.7.0 20120322 (mspgcc dev 20120716)
Copyright (C) 2012 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

user@instant-contiki:~/contiki$
```

https://sourceforge.net/projects/zolertia/files/Toolchain/

Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0

01-basics

Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0

Fixes a problem related to being allowed to write to USB ports as the /dev/ttyUSB0 used to flash the nodes

```
sudo usermod -a -G dialout user
```

For the change to take effect, you need to logout and log back in

Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0

```
PROCESS(hello_world_process, "Hello world process"); ❶
AUTOSTART_PROCESSES(&hello_world_process); ❷
```

❶ `hello_world_process` is the name of the process and `"Hello world process"` is the readable name of the process when you print it to the terminal.

❷ The `AUTOSTART_PROCESSES(&hello_world_process)` tells Contiki to start that process when it finishes booting.

```
/*---------------------------------------------------------------------------*/
PROCESS(hello_world_process, "Hello world process");
AUTOSTART_PROCESSES(&hello_world_process);
/*---------------------------------------------------------------------------*/
PROCESS_THREAD(hello_world_process, ev, data) ❶
{
  PROCESS_BEGIN(); ❷
  printf("Hello, world\n"); ❸
  PROCESS_END(); ❹
}
```

❶ You declare the content of the process in the process thread. You have the name of the process and callback functions (event handler and data handler).

❷ Inside the thread you begin the process,

❸ do what you want and

❹ finally end the process.

examples/zolertia/tutorial/01-basics/01-hello-world.c

Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0

To compile an application and program the nodes:

```
make TARGET=z1 hello-world.upload
```

You can save the TARGET so next time you don't have to type it:

```
make TARGET=z1 savetarget
```

This will create a Makefile.target file, for the compiler to know which platform to compile for, if no TARGET argument is added in the compilation command line

Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0

Shows a list of connected devices and USB ports

```
make z1-motelist
```

Restarts the devices

```
make z1-reset
```

Opens a serial connection and prints debug output

```
make login
```

As “make login” but with a timestamp

```
make serialview
```

If there are more than one device connected, use `MOTES=/dev/ttyUSBx` to choose which one to command, else the above commands will be executed on all connected devices

Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0

```
user@instant-contiki:~/contiki/examples/zolertia/tutorial/01-basics$ make TARGET=z1 01-hello-world.
upload
msp430-objcopy 01-hello-world.z1 -O ihex 01-hello-world.ihex
cp 01-hello-world.ihex tmpimage.ihex
/dev/ttyUSB0
make z1-reset z1-upload
make[1]: Entering directory `/home/user/contiki/examples/zolertia/tutorial/01-basics'
make -k -j 20 z1-reset-sequence
make[2]: Entering directory `/home/user/contiki/examples/zolertia/tutorial/01-basics'
../../../../tools/zolertia/z1-bsl-nopic --z1 -c /dev/ttyUSB0 -r
MSP430 Bootstrap Loader Version: 1.39-goodfet-8
Use -h for help
Use --fromweb to upgrade a GoodFET.
Reset device ...
Done
make[2]: Leaving directory `/home/user/contiki/examples/zolertia/tutorial/01-basics'
make -j 20 z1-upload-sequence
make[2]: Entering directory `/home/user/contiki/examples/zolertia/tutorial/01-basics'
+++++ Erasing /dev/ttyUSB0
MSP430 Bootstrap Loader Version: 1.39-goodfet-8
Use -h for help
Use --fromweb to upgrade a GoodFET.
Mass Erase...
Transmit default password ...
+++++ Programming /dev/ttyUSB0
MSP430 Bootstrap Loader Version: 1.39-goodfet-8
Invoking BSL...
Transmit default password ...
Current bootstrap loader version: 2.13 (Device ID: f26f)
Changing baudrate to 38400 ...
Program ...
43055 bytes programmed.
+++++ Resetting /dev/ttyUSB0
MSP430 Bootstrap Loader Version: 1.39-goodfet-8
```

examples/zolertia/tutorial/01-basics/01-hello-world.c

Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0

user@instant-contiki: ~/contiki/examples/zolertia/tutorial/01-basics

File Edit View Search Terminal Help

```
user@instant-contiki:~/contiki/examples/zolertia/tutorial/01-basics$
user@instant-contiki:~/contiki/examples/zolertia/tutorial/01-basics$
user@instant-contiki:~/contiki/examples/zolertia/tutorial/01-basics$ make z1-motelist
using saved target 'z1'
../../../../tools/zolertia/motelist-zolertia -b z1
------------- --------------- ----------------------------------------
Reference     Device          Description
------------- --------------- ----------------------------------------
Z1RC5072      /dev/ttyUSB0    Silicon Labs Zolertia Z1
user@instant-contiki:~/contiki/examples/zolertia/tutorial/01-basics$
```

Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0

```
user@instant-contiki:~/contiki/examples/zolertia/tutorial/01-basics$ make z1-reset && make login
using saved target 'z1'
make -k -j 20 z1-reset-sequence
using saved target 'z1'
make[1]: Entering directory `/home/user/contiki/examples/zolertia/tutorial/01-basics'
../../../../tools/zolertia/z1-bsl-nopic --z1 -c /dev/ttyUSB0 -r
MSP430 Bootstrap Loader Version: 1.39-goodfet-8
Use -h for help
Use --fromweb to upgrade a GoodFET.
Reset device ...
Done
make[1]: Leaving directory `/home/user/contiki/examples/zolertia/tutorial/01-basics'
using saved target 'z1'
../../../../tools/sky/serialdump-linux -b115200 /dev/ttyUSB0
connecting to /dev/ttyUSB0 (115200) [OK]
Rime started with address 193.12.0.0.0.0.19.208
MAC c1:0c:00:00:00:00:13:d0 Ref ID: 5072
Contiki-2.6-3253-g441f988 started. Node id is set to 5072.
CSMA ContikiMAC, channel check rate 8 Hz, radio channel 26
Tentative link-local IPv6 address fe80:0000:0000:0000:c30c:0000:0000:13d0
Starting 'Hello world process'
Hello, world
Hello world, again!
This is a value in hex 0xABCD, the same as 43981
```

examples/zolertia/tutorial/01-basics/01-hello-world.c

Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0

Buttons and LEDs

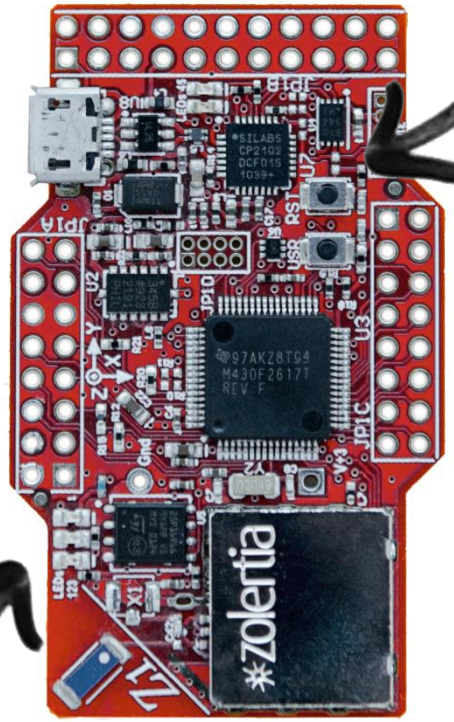
Events and actions can be triggered by pressing the user button: send a message over the radio, take a sensor sample, start a process, etc.

The LEDs (light-emitting diodes) help us to understand what happens in the mote, by using different colours and blinking sequences we know when an event is happening, if there are any errors or what happens in our application.

Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0

```c
PROCESS_THREAD(led_button_process, ev, data)
{
  /* Every process start with this macro, we tell the system this is the start
   * of the thread
   */
  PROCESS_BEGIN();

  /* Start the user button using the "SENSORS_ACTIVATE" macro */
  SENSORS_ACTIVATE(button_sensor);

  /* And now we wait for the button_sensor process to inform us about the user
   * pressing the button.  We create a loop to wait forever, but to save
   * processing cycles, we "pause" the application until the expected event is
   * received and only resume when this event occurs
   */

  while(1) {
    printf("Press the User Button\n");
    PROCESS_WAIT_EVENT_UNTIL(ev == sensors_event && data == &button_sensor);

    /* When the user button is pressed, we toggle the LED on/off... */
    leds_toggle(LEDS_GREEN);

    /* To try different LED combinations try and replace "LEDS_GREEN" with:
     * - LEDS_RED
     * - LEDS_BLUE
     * - LEDS_ALL
     *
     * And we print its status: when zero the LED is off, else on.
     * The number printed when the sensor is on is the LED ID, this value is
     * used as a mask, to allow turning on and off multiple LED at the same
     * time (for example using "LEDS_GREEN + LEDS_RED" or "LEDS_ALL"
     */
    printf("The sensor is: %u\n", leds_get());
  }
```

Buttons: User & Reset



LEDs: Red, Blue, Green

examples/zolertia/tutorial/01-basics/02-led-and-button.c

Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0

Timers

Timers allow to execute actions periodically, like measuring a sensor periodically, waiting a few seconds before executing a function, etc.

Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0

Timers

- Timer: ms, manual
- Stimer: seconds, manual
- Etimer: ms, triggers an event
- Ctimer: ms, callbacks
- Rtimer: us, callbacks

```
static void
rtimer_callback_example(struct rtimer *timer, void *ptr)
{
  uint32_t *rtimer_ticks = ptr;
  printf("rtimer, now: \t%ld\n", *rtimer_ticks);

  /* We can restart the ctimer and keep the counting going */
  (*rtimer_ticks)++;
  ctimer_restart(&ct);
}
/*---------------------------------------------------------------------------*/
/* This is an example of functions called by the ctimer callback timer.  Notice
 * the argument of the function is "void *ptr", meaning you can pass any type
 * of data as pointer
 */
static void
ctimer_callback_example(void *ptr)
{
  uint32_t *ctimer_ticks = ptr;
  printf("ctimer, now: \t%ld\n", *ctimer_ticks);

  /* The real timer allows execution of real-time tasks (with predictable
   * execution times).
   * The function RTIMER_NOW() is used to get the current system time in ticks
   * and RTIMER_SECOND specifies the number of ticks per second.
   */

  (*ctimer_ticks)++;
  rtimer_set(&rt, RTIMER_NOW() + RTIMER_SECOND, 0,
             rtimer_callback_example, ctimer_ticks);
}
```

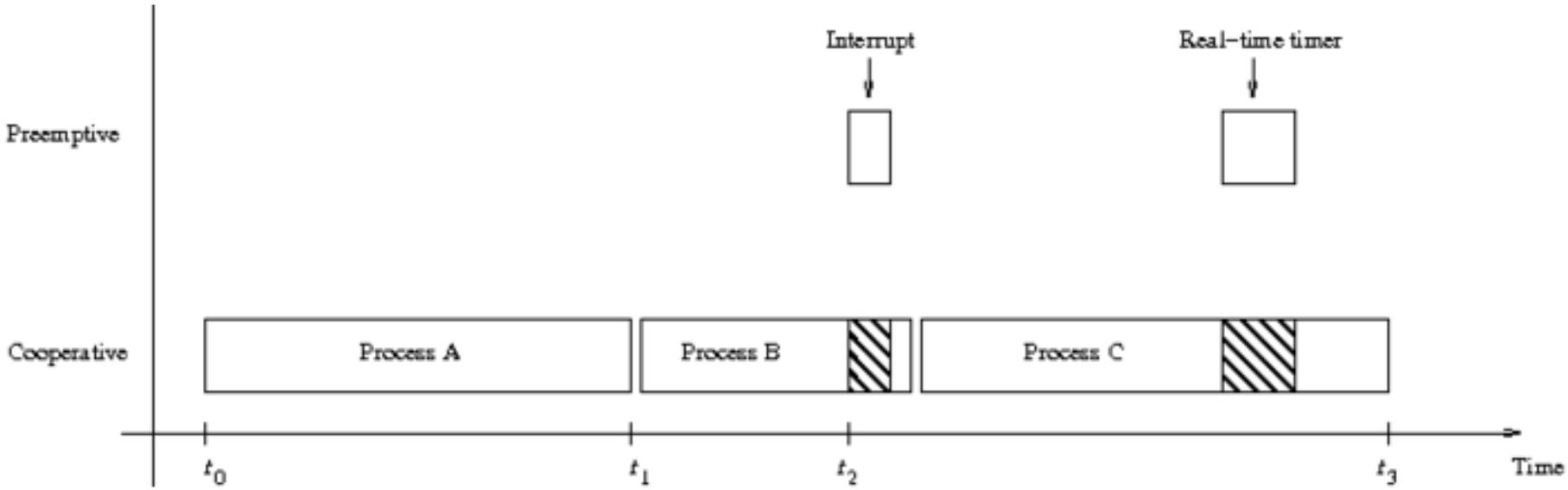
```
void etimer_set(struct etimer *t, clock_time_t interval); // Start the timer.
void etimer_reset(struct etimer *t); // Restart the timer from the previous expiration time
void etimer_restart(struct etimer *t); // Restart the timer from current time.
void etimer_stop(struct etimer *t); // Stop the timer.
int etimer_expired(struct etimer *t); // Check if the timer has expired.
int etimer_pending(); // Check if there are any non-expired event timers.
clock_time_t etimer_next_expiration_time(); // Get the next event timer expiration time.
void etimer_request_poll(); // Inform the etimer library that the system clock has changed.
```

examples/zolertia/tutorial/01-basics/03-timers.c

Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0

Processes

Contiki has two execution contexts: cooperative and preemptive
Processes are cooperative and sequential, interrupts (button, sensors events) and the real-timer are preemptive.



https://github.com/contiki-os/contiki/wiki/Processes

examples/zolertia/tutorial/01-basics/04-processes.c

Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0

```c
printf("Process 1 started\n");

/* Use a random value to wait before starting the process */
etimer_set(&et1, WAIT_TIME);
PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&et1));

/* Start the process and send as data our name */
process_start(&process3, "Process 1");

/* And wait until process3 sends us a message */

while(1) {
  /* This protothread waits for any event, we need to check of
   * ourselves
   */
  PROCESS_YIELD();

  /* We are waiting here for an event from process3 */
  if(ev == event_from_process3) {
    counter = *((uint8_t *)data);
    printf("Process 3 has requested shutdown in %u seconds\n", counter);
    etimer_set(&et1, CLOCK_SECOND);
  }

  /* We are waiting here for a timer event */
  if(ev == PROCESS_EVENT_TIMER) {

    /* When the counter reaches zero, kill process3 */
    if(counter <= 0) {
      process_exit(&process3);

    /* Increment the counter value and restart the timer */
    } else {
      printf("Process 3 will be terminated in: %u\n", counter);
      counter--;
      leds_toggle(LEDS_RED);
      etimer_reset(&et1);
    }
  }
```

```c
struct process {
  struct process *next;
  const char *name;
  int (* thread)(struct pt *,
                 process_event_t,
                process_data_t);
  struct pt pt;
  unsigned char state, needspoll;
};
```

https://github.com/contiki-os/contiki/wiki/Processes

examples/zolertia/tutorial/01-basics/04-processes.c

Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0

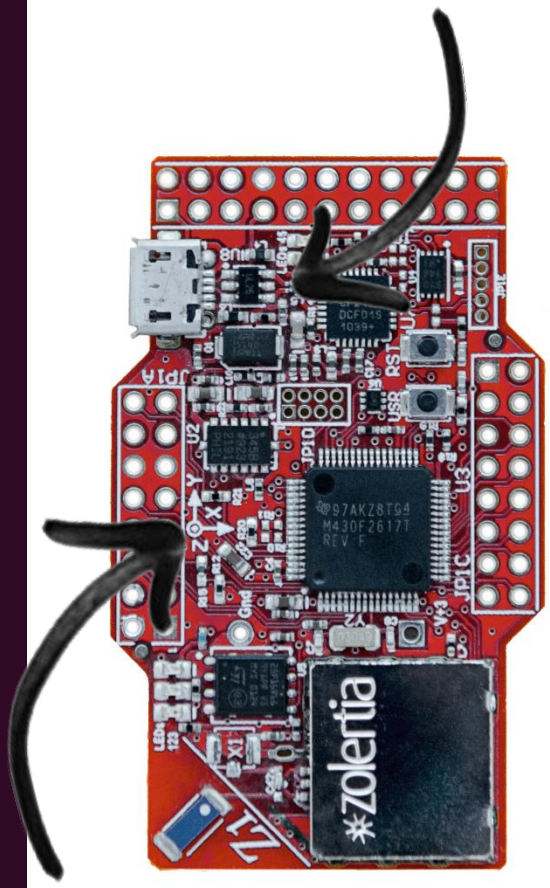
Sensors

A sensor is a transducer whose purpose is to sense or detect a characteristic of its environment, providing a corresponding output, generally as an electrical or optical signal, related to the quantity of the measured variable

Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0

```c
/* Initialize the sensors, the SENSORS_ACTIVATE(...) macro invokes the
 * configure(...) method of Contiki's sensor API
 */
SENSORS_ACTIVATE(adxl345);
SENSORS_ACTIVATE(tmp102);
SENSORS_ACTIVATE(battery_sensor);

/* Spin the timer */
etimer_set(&et, CLOCK_SECOND);

while(1) {

  PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&et));

  /* Read the sensors */
  x_axis = adxl345.value(X_AXIS);
  y_axis = adxl345.value(Y_AXIS);
  z_axis = adxl345.value(Z_AXIS);
  temp   = tmp102.value(TMP102_READ);
  batt   = battery_sensor.value(1);

  /* Print the readings */
  printf("Acceleration: X %d Y %d Z %d\n", x_axis, y_axis, z_axis);
  printf("Temperature: %d.%u\n", temp / 100, temp % 100);

  /* Convert the ADC readings to mV */
  batt *= 5000;
  batt /= 4095;
  printf("Battery: %u\n\n", (uint16_t)batt);

  etimer_reset(&et);
}

PROCESS_END();
```

TMP102: temperature



ADXL345: acceleration(3 axis)

examples/zolertia/tutorial/01-basics/05-onboard-sensors.c

Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0

Antonio Liñán Colina

alinan@zolertia.com
antonio.lignan@gmail.com

Twitter: @4Li6NaN
LinkedIn: Antonio Liñan Colina
github.com/alignan
hackster.io/alinan