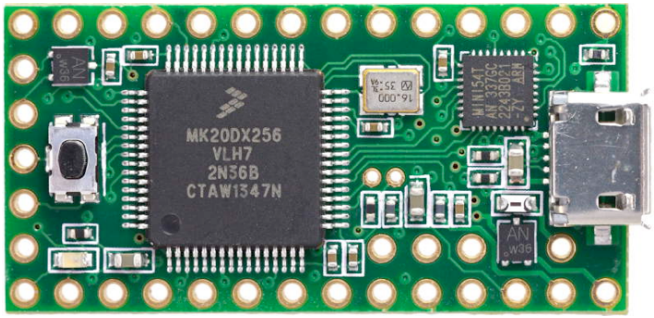
Low-cost LoRa image IoT device for visual surveillance applications

Prof. Congduc Pham
http://www.univ-pau.fr/~cpham
Université de Pau, France

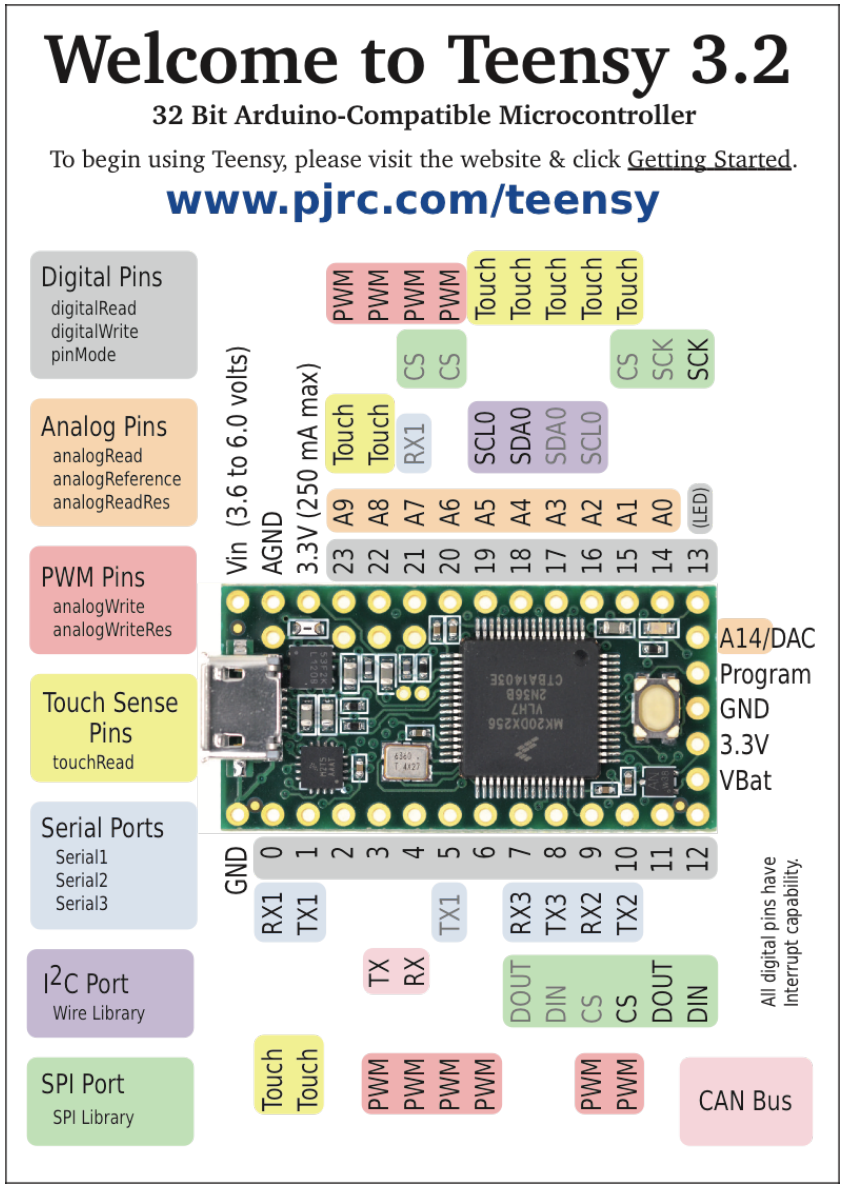
Contents

- We will show how to build a low-cost, long-range image IoT device for visual surveillance applications
- You should read first the tutorials on our low-cost and long-range IoT platform
  - "Low-cost LoRa IoT device: a step-by-step tutorial" to understand the general long-range IoT device concepts
  - "Low-cost LoRa gateway: a step-by-step tutorial" to understand the gateway part
- The target hardware platform for the image IoT device is the Teensy32
- More information can also be found at http://cpham.perso.univ-pau.fr/WSN-MODEL/tool-html/imagesensor.html
- Let's get started…

Assembling the hardware



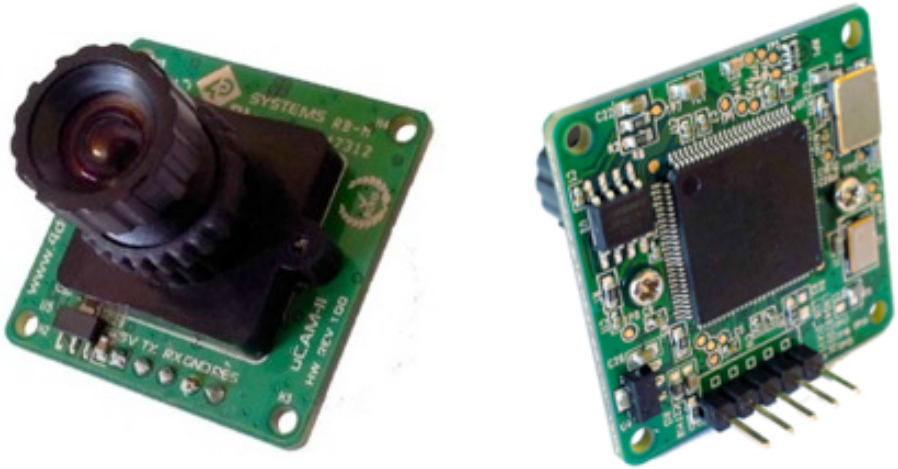
Get the Teensy32 board



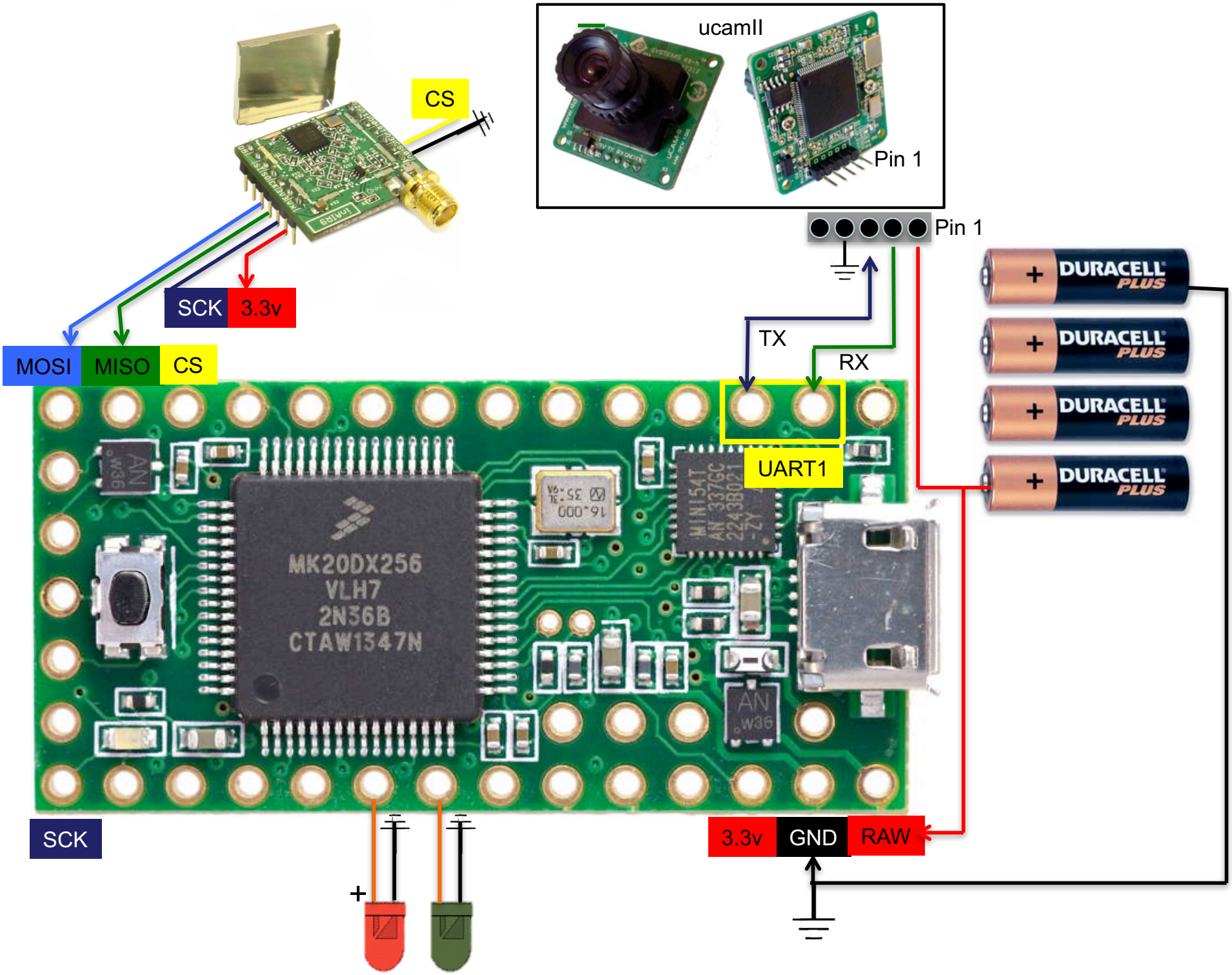
The Teensy 3.2 is an Arduino-compatible board by PJRC. It is a small form factor micro-controller board based on the MK20DX256 Cortex-M4. Its rated speed is 72MHz but it can be overclocked at 96MHz or slowed down to 48MHz and 24MHz. The board has 64KB of SDRAM, making it suitable for storing raw image data for various processing such as image compression and image change detection algorithms.

It can have advanced power-saving mode that will be managed with the `Snooze` library as it will be explained later on in the software section.

Get the 4D System uCamII camera



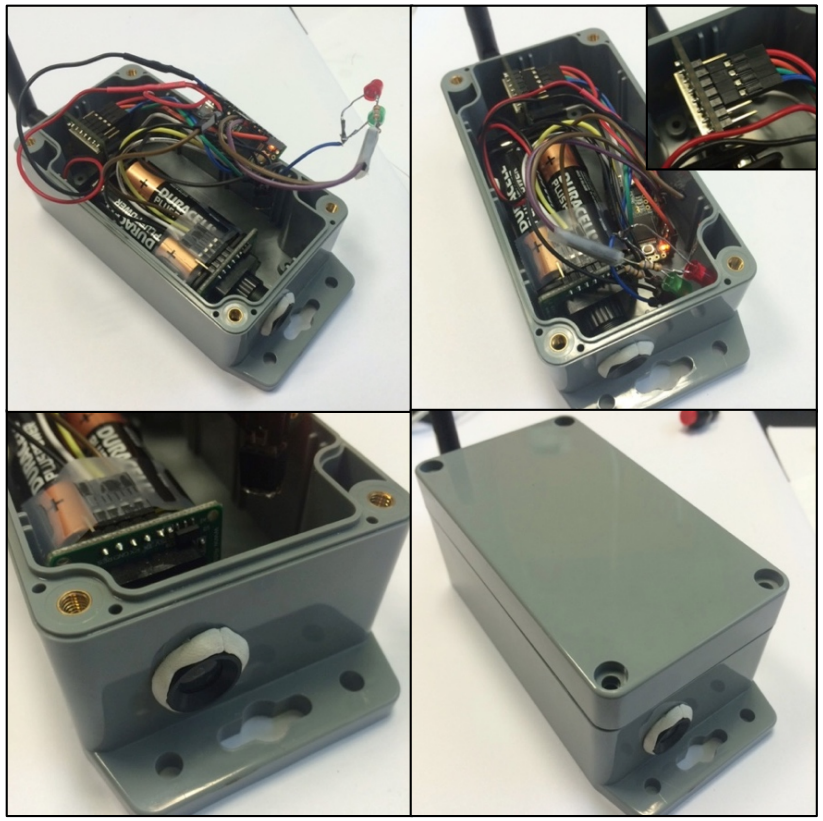
The CMOS uCamII camera is from 4D systems. The uCamII is connected to the Teensy32 board through an UART interface at 115200 bauds. The uCamII is capable of providing both raw and JPEG bit streams but we are not using this last feature as it is impossible from the delivered JPEG bit stream to build a packet stream tolerant to packet losses. Instead, we retrieve raw 128x128 8-bpp grey scale images from the uCamII then we operate image compression on the board.


ucamII
Pin 1
CS
SCK
3.3v
MOSI
MISO
CS
Pin 1
TX
RX
UART1
DURACELL PLUS
DURACELL PLUS
DURACELL PLUS
DURACELL PLUS
MK20DX256
VLH7
2N36B
CTAW1347N
SCK
3.3v
GND
RAW

CONNECTING EVERYTHING

- The radio module
  - MOSI, MISO, CLK and CS of SPI bus are connected to corresponding pins of the Teensy32. VCC for the radio is connected to the 3.3V pin of the Teensy32.
- The uCamII module
  - It is connected to Teensy's UART1 interface
  - VCC for the uCAM needs between 4.5V and 9V that the Teensy cannot deliver. So a wire from the battery pack will be directly connected to uCam's pin 1.
- The battery pack
  - Use 4 AA batteries to deliver about 6V
- The leds
  - 2 leds will be connected to show operation of the board: capture led (red) and sync led (green)
  - Connect red and green to Teensy's pin 18 & 19 respectively

PUTTING IN A BOX AND USING DIFFERENT LENSES



56° lens

76° lens

116° lens

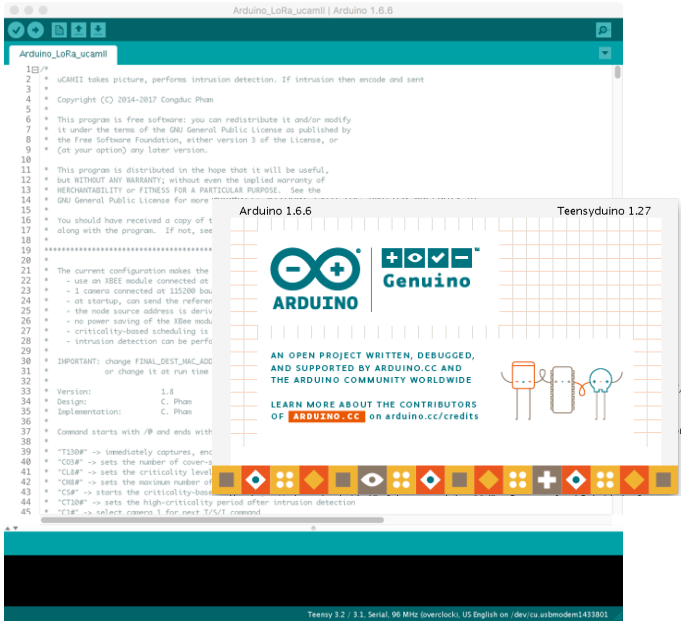
The uCamII is shipped with a 56° angle of view lens but 76° and 116° lenses are also available for various application needs.

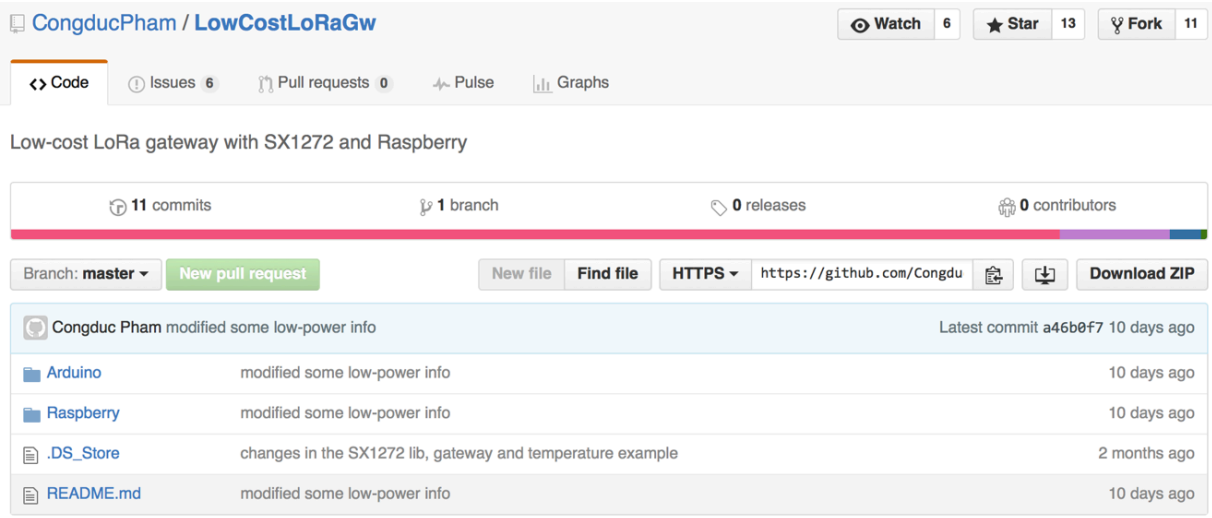
Updating to the uCamIII

- The uCamII is now replaced by the uCamIII which has backward compatibility
- The uCamIII adds a hardware RESET pin that can be used to reset the camera module when synchronization is lost with the camera. This was definitely lacking in the uCamII

Getting, compiling & uploading the software

GETTING THE SOFTWARE

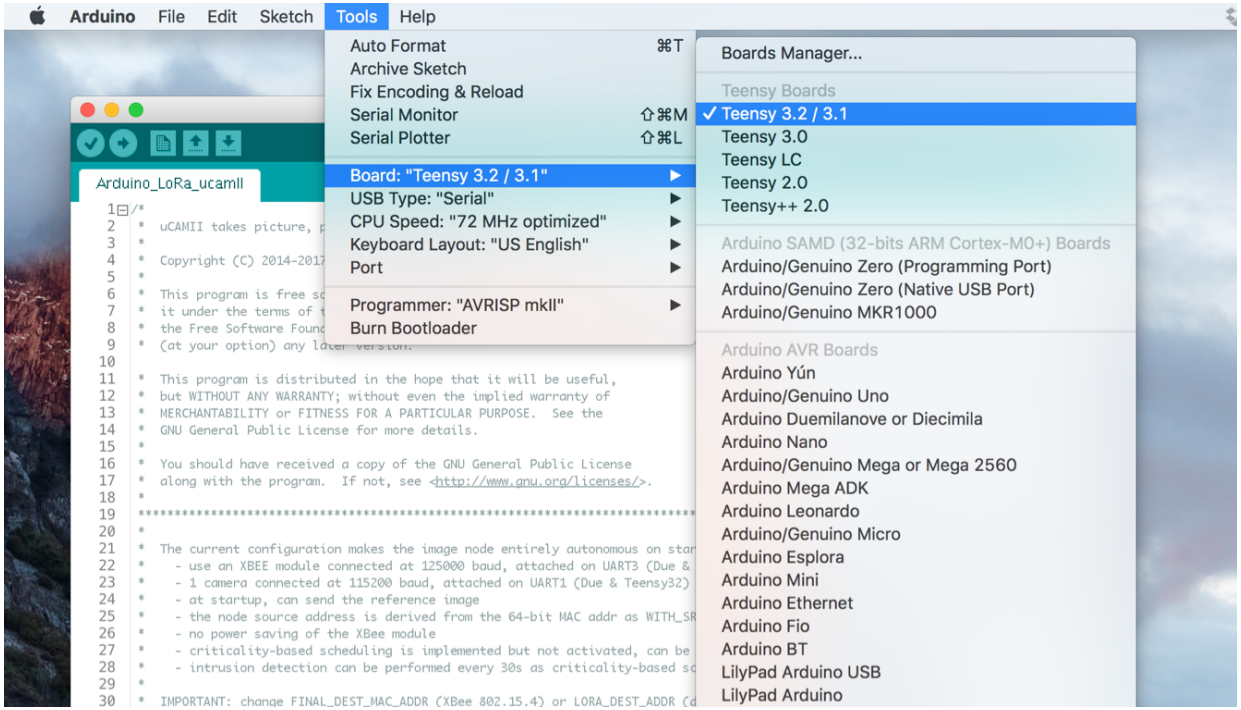




You will need the Arduino IDE. You can take the latest version but it is not mandatory. Then you need to install the Teensyduino add-on to the installed Arduino IDE. You can download all versions of Teensyduino on www.pjrc.com but take care to choose a version that is compatible with you Arduino IDE version. Then go to our github: https://github.com/CongducPham/LowCostLoRaGw .

Go into the Arduino folder and get Arduino_LoRa_ucamII, SX1272 and uCam folder. Copy Arduino_LoRa_ucamII into your “sketch” folder and SX1272 & uCam into “sketch/libraries”

Compiling



Open the Arduino_LoRa_ucamII sketch and select the Teensy32 board. You can use 72MHz version. **Connect the Teensy to your computer**.

Then, click on the « verify » button

Uploading & serial monitor

After compilation, Teensyduino will display the Teensy specific control window indicating that you may have to press the RESET button on the Teensy board to upload the sketch.

Please read the Teensy troubleshooting section at https://www.pjrc.com/teensy/troubleshoot.html

Once the program has been uploaded you can check the image device output on the Arduino Serial Monitor.

In real-world deployment scenario, you have to rely on the leds to know how the image device is operating.

Default behavior

- The default behavior is defined by the following main compilation define statements
  - #define LOW_POWER & #define LOW_POWER_HIBERNATE
  - #define USEREFIMAGE
  - #define GET_PICTURE_ON_SETUP
- Once powered on, the image device will therefore
  1. Take an image and store it as the reference image
  2. Sleep for DEFAULT_INTER_SNAPSHOT_TIME (e.g. 60s) using low-power mode
  3. Wake-up, take an image and compare with reference image
  4. Transmit to the gateway the new image on change detection
  5. Cycle back to step 2
- DEFAULT_INTER_SNAPSHOT_TIME can be easily changed

LED INDICATIONS

- Sync led (green led)
  - The green led shows that the ucam has successfully syncked
- Capture led (red led)
  - When device is initiating an image capture the red led is ON
  - The red led will blink once when transmitting each packet
  - When device is receiving a command, the red led will blink 5 times
- On startup
  - The green led will blink 5 times then the red led will be ON during the sync attempts
  - When ucam has synched, green led will be ON for 1s
  - If the red led stays ON, it means that the ucam is not ready, reset the board by unplugging and plugging again the VCC wire; repeat the process until the green led turns to ON
- Taking first reference image
  - The red led will blink 5 times then the green led will blink 3 times to indicate that raw image is encoded and for each transmitted image packet the red led will blink
- Waking-up for image change detection
  - The red led will be ON and once camera has synched the green led will blink 3 times
  - The red led will then be OFF and if there have been changes, will blink for each image packet transmission

IMAGE ENCODING SCHEME

ADJUSTABLE IMAGE QUALITY FACTOR Q

raw 16384b

Q=100; 9768b (1.67)

PSNR=51.344

Q=90; 5125b (3.2)

PSNR=29.414

Q=80; 3729b (4.4)

PSNR=28.866

Q=70; 2957b (5.5)

PSNR=28.477

Q=60; 2552b (6.4)

PSNR=28.024

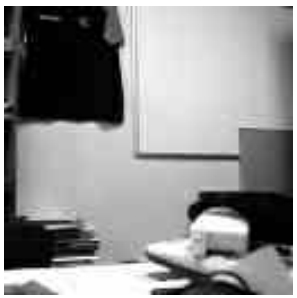
Q=50; 2265b (7.2)

PSNR=27.912

Q=40; 2024b (8.1)

PSNR=27.423

Q=30; 1735b (9.5)



PSNR=26.933

Q=20; 1366b (12)

PSNR=26.038

Q=10; 911b (18)

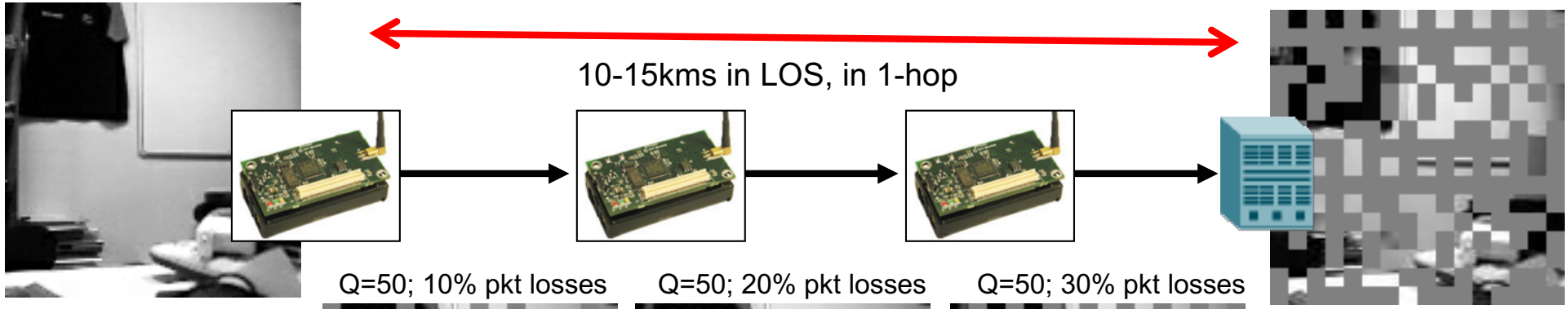
PSNR=25.283

Q=5; 576b (28.44)

PSNR=23.507

Scientific cooperation with V. Lecuire from CRAN laboratory for the optimized image encoding algorithm

PACKET LOSS-TOLERANT BIT STREAM, ANY RECEPTION ORDER



Q=50; 10% pkt losses

Q=50; 20% pkt losses

Q=50; 30% pkt losses

Q=50; 40% pkt losses

Q=50; 50% pkt losses

Q=50; 60% pkt losses

Image change detection for Intrusion detection

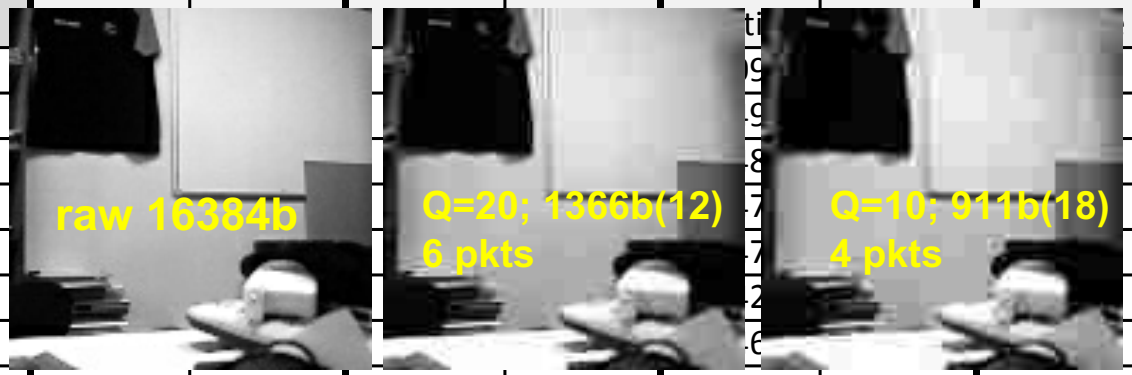
Very lightweight « simple-differencing » method done on-the-fly when reading raw image data. Change detection takes into account modification in image's luminosity:

```
#define LUM_HISTO
```

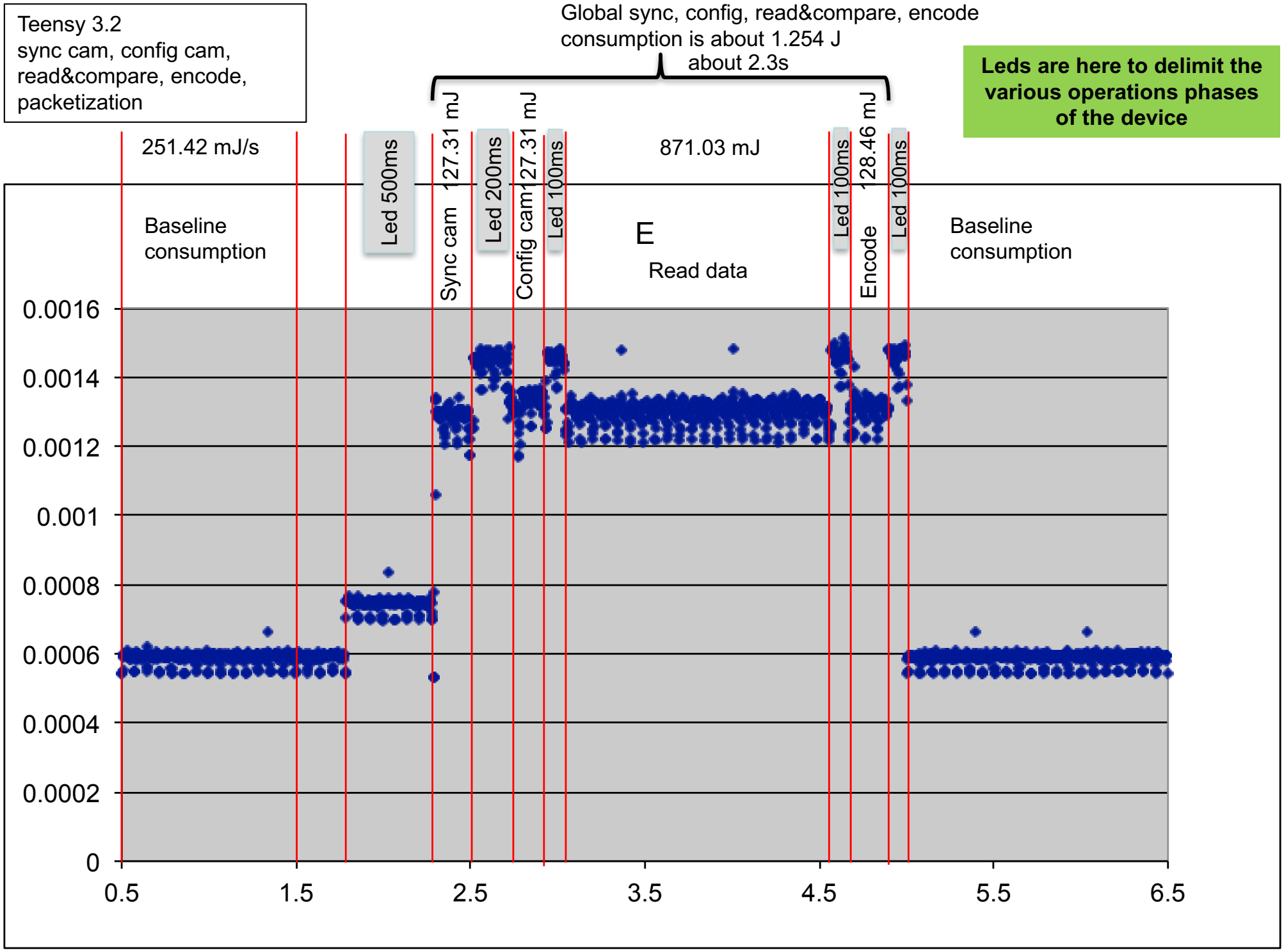
```
// CHANGE HERE THE THRESHOLD TO TUNE THE INTRUSION DETECTION PROCESS
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
#define PIX_THRES                   55
#define INTRUSION_THRES             500
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

Image encoding performances, Q=10 & 20

	96MHz		72MHz		48MHz		24MHz		MSS=240 N	MSS=240 S
Quality Factor Q	encode	packetiza	encode	packetiza	encode	packetiza	encode	packetization time	number of packets	size in bytes (compression ratio)
100								813	47	9982 (1.64)
90								322	23	5090 (3.21)
80								218	16	3595 (4.55)
70								178	13	2842 (5.76)
60								162	11	2461 (6.65)
50								150	10	2129 (7.69)
40								139	9	1898 (8.63)
30	224	33	260	44	345	64	637	127	7	1608 (10.19)
20	223	31	260	39	345	58	636	115	6	1279 (12.81)
10	223	26	260	31	345	50	636	99	4	824 (19.88)
5	223	23	259	31	344	45	635	89	3	503 (32.57)



- Capturing an image and encoding it roughly take 2.3s
  - Time to sync & config ucam is about 400ms
  - Time to read raw image data from ucam is 1512ms
  - Time of compare with reference image is neglectible
  - Time for encoding and packetization is about 300ms


Teensy 3.2
sync cam, config cam, read&compare, encode, packetization
Global sync, config, read&compare, encode
consumption is about 1.254 J
about 2.3s
Leds are here to delimit the various operations phases of the device
251.42 mJ/s
Led 500ms
127.31 mJ
Sync cam
Led 200ms
127.31 mJ
Config cam
Led 100ms
871.03 mJ
Read data
Led 100ms
128.46 mJ
Encode
Led 100ms
Baseline consumption
Baseline consumption
0.0016
0.0014
0.0012
0.001
0.0008
0.0006
0.0004
0.0002
0
0.5
1.5
2.5
3.5
4.5
5.5
6.5

Power consumption

	baseline (mJ/s)	baseline, hibernate (mJ/s)	read (mJ)	encode (mJ)	Read+encode (mJ)
96MHz	251.42	0.834	871.03	128.46	999.49
72MHz	219.54	0.834	834.97	143.58	978.55
48MHz	211.19	0.834	813.30	185.58	998.88
24MHz	160.95	0.834	719.09	302.48	1021.57

- Using the nominal 72MHz mode minimizes consumption
- Hibernate mode when idle consumes about 167uA
- When transmitting the board consumes about 68mA
- Assuming
  - 1 image/hour (2s)
  - Image encoding with Q=10 (300ms)
  - Transmission of 4 packets (8s)
- Then can run for 268 days on 4 AA batteries

INTERACTING WITH DEVICE

- **Must** disable low-power mode to enable receive window at the image device
  - Comment #define LOW_POWER
- By connecting the Teensy to your computer
  - String commands can be passed to the device from the Arduino serial monitor
- By remote LoRa message
  - Use for instance an interactive device to send string commands to the image device
- Command starts with /@ and ends with #

String command list

- Command starts with /@ and ends with #
- Example
  - Set inter-snapshot time to 30s: /@F30000#
  - Make a snapshot: /@I#

Command	Camera command
T	immediately captures, encodes and transmits the image
C1	selects camera 1 for next T/S/I/Q command
F30000	sets inter-snapshot time to 30000ms, i.e. 30s, fps is then 1/30
S0/S1	starts a snapshot, make the comparison with the reference image and transmit the image if S1
I	makes a snapshot and stores a new reference image
Z150	sets the MSS size, default is 240 for LoRa
Q30	sets the quality factor, default is 20 for LoRa
D8	sets the 8-bit destination address to node 8. Default is 1. Use D0 for broadcast
Command	LoRa specific commands
LORAM1	sets to Libelium LoRa mode 1
LORAC12	uses channel 12
LORAPL/H/M	sets power to Low, High or Max
LORAA9	sets node address to 9

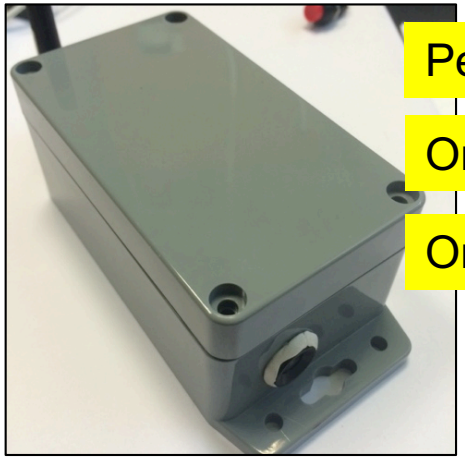
Gateway side

- Support of image processing is in the `lora_gateway/ucam-images` folder
- Read the README at https://github.com/CongducPham/LowCostLoRaGw/tree/master/gw_full_latest/ucam-images for installation instructions
- Received encoded images will be saved by `post_processing_gw.py` in `/home/pi/Dropbox/LoRa-test/images`
- `post_processing_gw.py` will call the image decoding program to decode in BMP the encoded image and will move the BMP file into `/var/www/html/images/uploads/node_x` where `x` is the address of the device
- Connect to http://192.168.200.1/images to view the received images (assuming you connect to the gateway's WiFi network)

RECEIVING IMAGE PACKETS

- When `post_processing_gw.py` receives the first image packet for a new image, it creates a file in `/home/pi/Dropbox/LoRa-test/images`
- For instance if you receive a first image from sensor 3 taken by camera 0 and encoded with a quality factor of 10, then the BMP image will be named: `ucam_0-node_0003-cam_0-Q10.dat`
- A timer will also be set (e.g. 90s in the current distribution) to close the file and start the image decompression procedure
- When decompressing the compressed image data in BMP format, the file will be named: `ucam_0-node_0003-cam_0-Q10-P4-S847.bmp` for instance to indicate that 4 packets has been received and the real compressed size is 847 bytes.

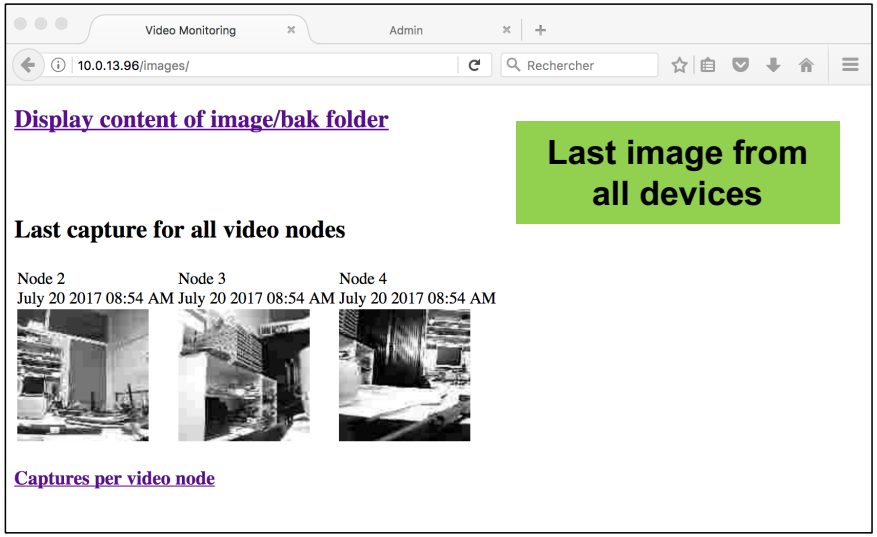
OUT-OF-THE-BOX !



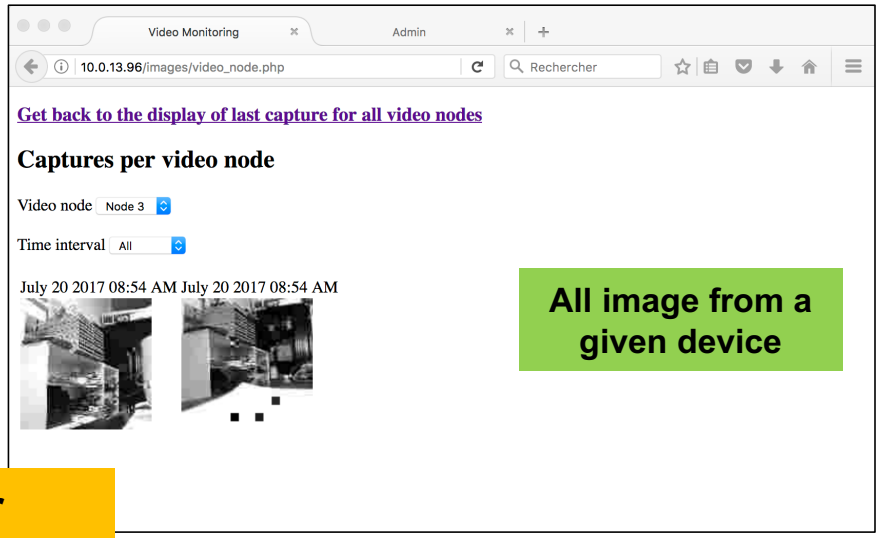
Periodic

On-demand

On event



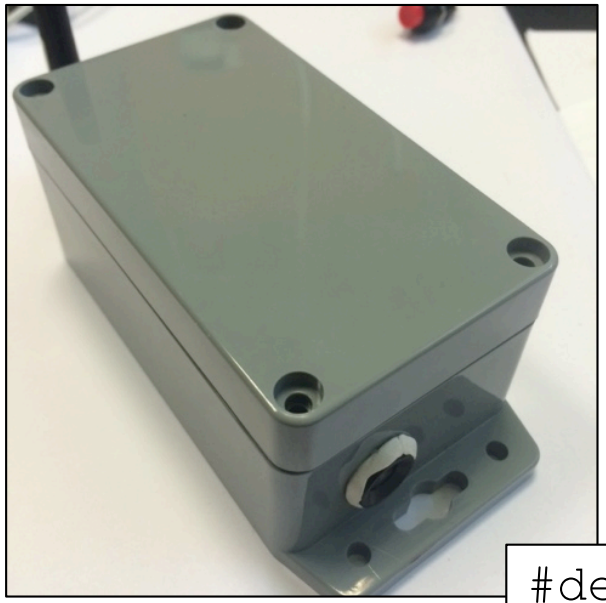
Last image from all devices



All image from a given device

Embedded web server
http://192.168.200.1/images

DEFAULT CONFIGURATION



```
#define LORAMODE  1
#define UCAM_ADDR 4
```

The default configuration in the Arduino_LoRa_ucamII example is:

Send packets to the gateway (one or many if in range)
LoRa mode 1 (BW=125kHz, SF=12, freq=865.2MHz)
Node address is 4