



An open-source alpine web camera

Outline

- What is an “alpine web camera”?
- The Market
- Initial prototypes
- The result
- Other hurdles
- The future

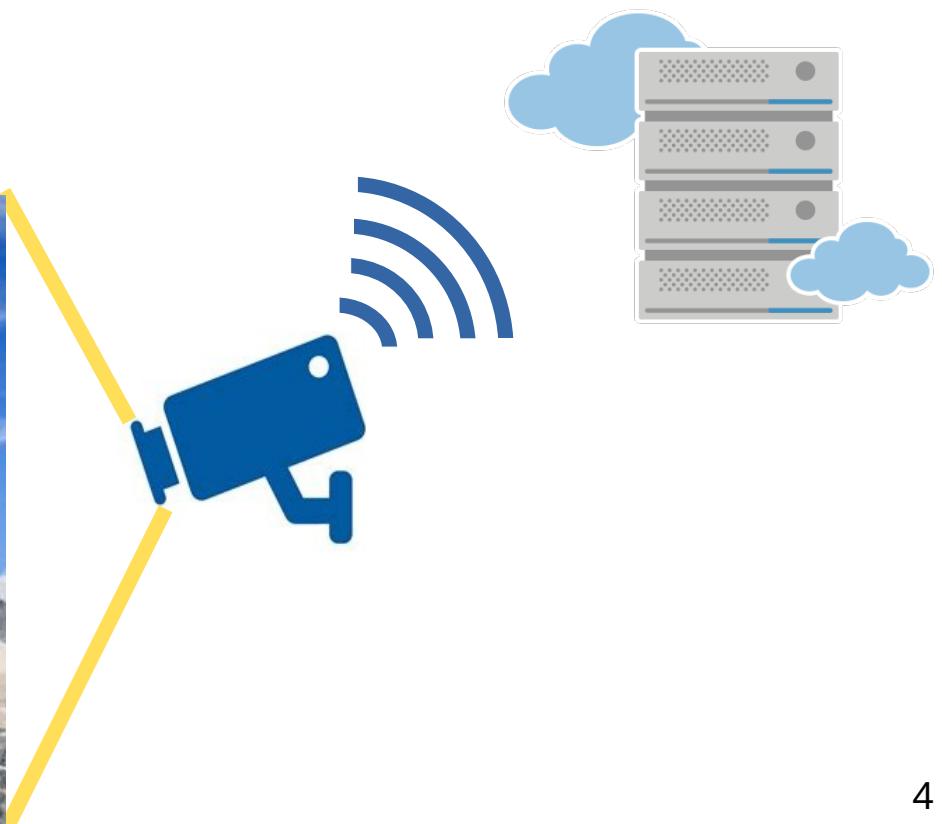


What is an “alpine web camera”?



A quick definition

An “alpine webcam” is a device that can take pictures, send them to a server, and sustain the harsh weather of the Alps for the entire year in a fully autonomous way.



What is an alpine web camera for?

How to know how are the conditions on a mountain before a hike?



Does the snow up there look like this....

What is an alpine web camera for?

How to know how are the conditions on a mountain before a hike?



...or like this?

What is an alpine web camera for?

- They help hikers and alpinists prepare for the conditions of the mountain & decide if a hike is feasible at all
- They help owners monitor damages made by the weather or the animals while the hut is unmanned
- They perform long-term monitoring of the weather and glaciers areas for climate studies
- Used for marketing too



*Beautiful, but don't forget your shovel
(picture taken by a webcam)*

What's so special about them?

These webcams can be very simple to make:

- Most IP-cam on the market can sustain rather harsh weather
- They need only Internet, power and a server to send the pictures to.
 - Even easier with “cloud solutions”
- Many are very easy to install
 - Screw it under the roof
 - Plug it in
 - Turn it on



What's so special about them?

In a few cases however it's not so simple:

- Some huts are empty in the winter and have no Internet connection or stable power throughout the year
- The weather can get extreme and destroy flimsy webcams
- Some webcams should be placed far from the hut itself to be useful



*Wind, snow, and no roof make things harder
(picture taken by a webcam)*

Requirements

Optimally, a candidate device should be:

- Reliable → some locations are inaccessible for most of the year
- Power efficient → to run on batteries and/or solar panels
- Sturdy and small → to resist the weather
- Can access the Internet → to upload the pictures
- Have a good quality camera → to take beautiful pictures
- Configurable → for variable conditions and usage
- User friendly → most hut manager are not digital natives
- Cheap → Hut managers are not rich :)



The Market

The market

The two “mainstream” product category that approach this niche are:

- Outdoor security cameras:
 - Either heavy or flimsy, sometimes both
 - Power-hungry
 - Expensive
 - Focus on video streams
- Trap cams:
 - Often no Internet connectivity
 - Not very configurable
 - Focus on motion detection
 - Focus on IR night pictures
 - Images often low-res



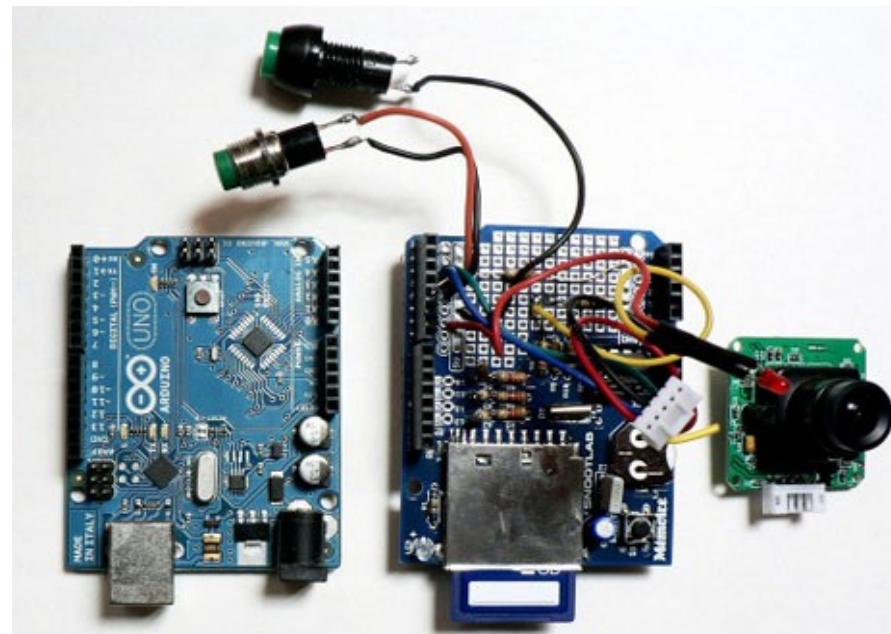
Niche products

We're also not the only people facing this problem!

- Timelapse cameras
- Weather cams
- etc...

The products targeting these niches, even commercial, are rather DIY:

- Not always reliable
- Hand made on request
- Surprisingly shy about their tech
- No specs whatsoever



Niche products

SolarCam.fr

- Mid-priced 4G solar webcam (~400€)
- You can opt-out from the “cloud” (FTP server)
- Rather DIY for a commercial offering
- Not open-source: you get a mystery box, good luck

SolarCam 4G



RifugiInRete.it

- Fully wired webcam (for power and internet)
- You pay a subscription for the “cloud”, no option to own your pictures (~100€/year)
- You can't buy it online: you have to convince them you're a hut manager first
- Also gives you a closed mystery box



Let's try them out

We tried 4 outdoor security cameras ranging from 80€ to 600€, a 350€ trapcam, and one “DIY” niche vendor (Meteoproject, 285€).



Results: barely sufficient in all cases, and generally disappointing.

Some examples



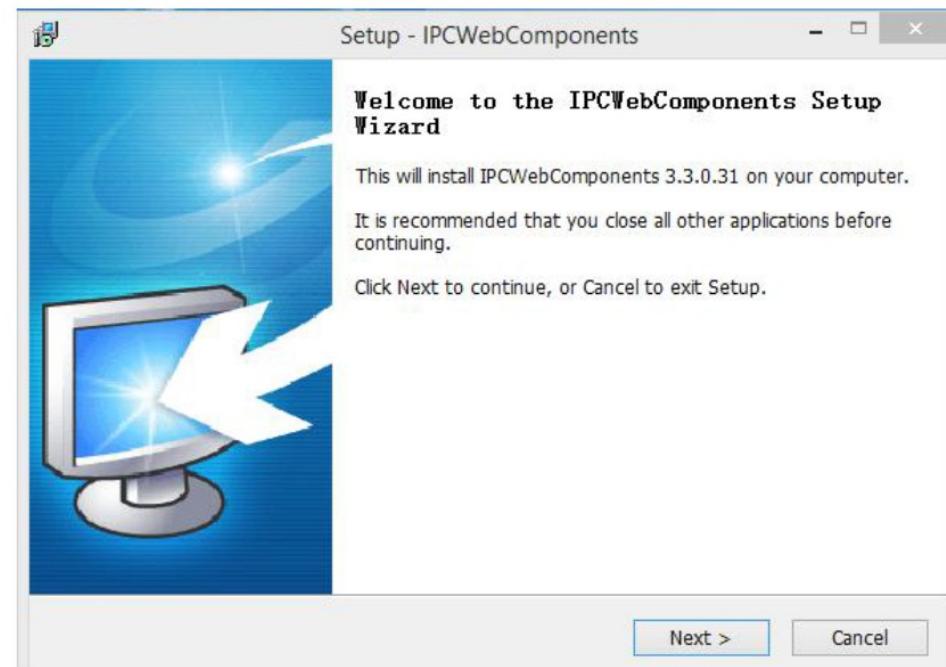
This is the manual for a 140€ security camera. Imagine the content.

Some examples

Another cam's control panel asked for the following:

- Make sure you have a Windows machine with Internet Explorer
- Activate “Allow unsigned ActiveX plugins to run on this browser”
- Visit shadywebsite.com from IE
- Download shadyplugin.exe and install it
- Restart the browser
- Browse to this IP and enjoy your control panel

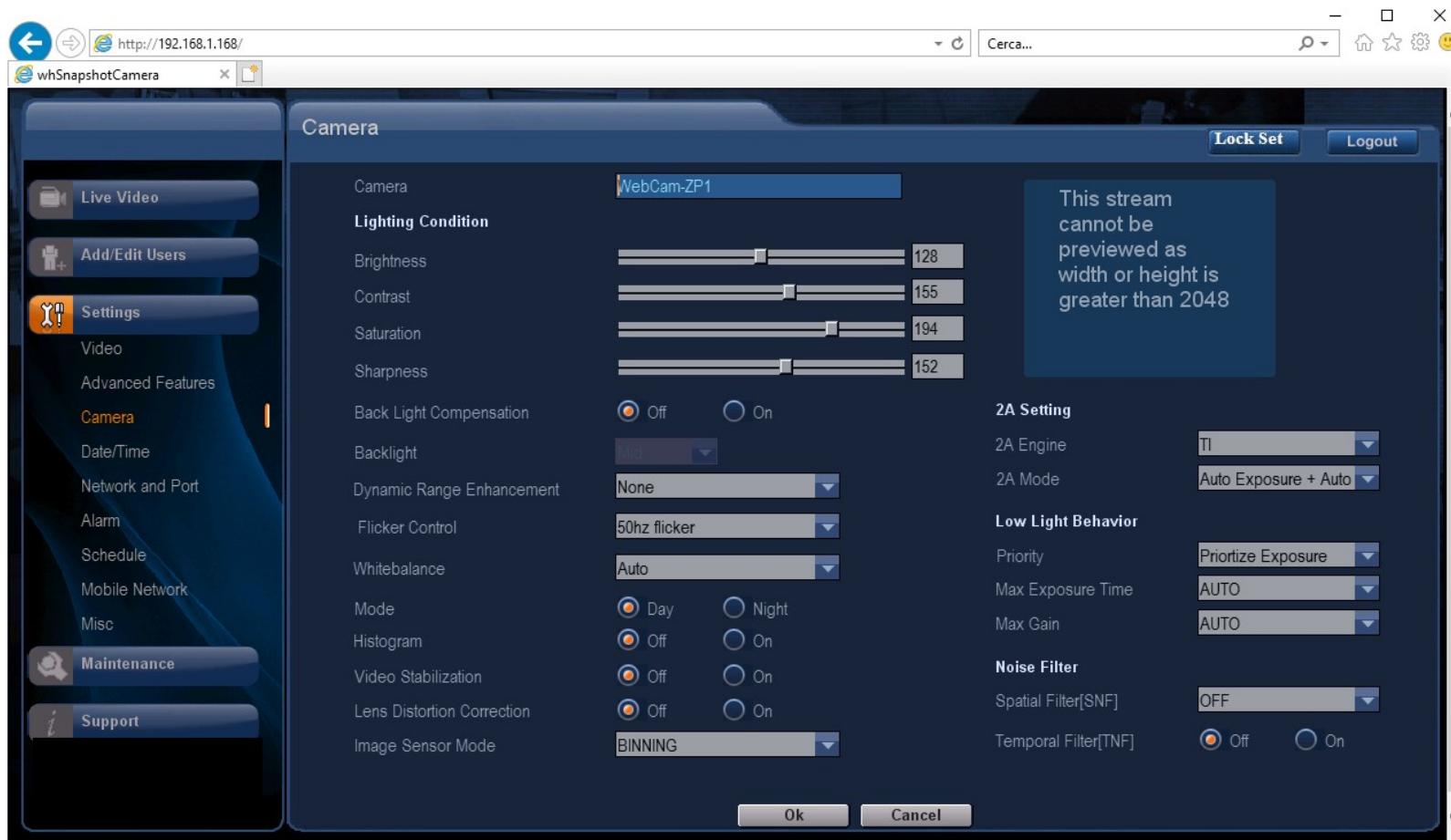
Sure no trouble can come from this, am I right??



Straight out of the user manual

Some examples

Even the most high-end camera we tested (600€) requires a similar setup (IE, ActiveX) just to show you a form!



Some examples

This is one of the most usable panels we came across (the trapcam, 350€):

The screenshot shows a configuration interface titled "MMS / 4G Setup". The interface is divided into several sections:

- Setup**: Includes Mode (Auto), MMS Option (OFF), Country (America), Data Option (OFF), Operator (AT&T), and a checkbox for "Show 4G Parameters".
- MMS APN**: Fields for APN, Account, and Password, all containing placeholder text "*****".
- MMS Server**: Fields for URL (MMSC) and Proxy (IP) with Port, both containing placeholder text "*****".
- Phone Number**: Fields for Phone 1, Phone 2, and Phone 3.
- Data APN**: Fields for APN, Account, and Password, all containing placeholder text "*****".
- SMTP Server**: Fields for Type (default1), Encrypt (SSL), Server, Email, and Password.
- FTP Server**: Fields for Server, Port, Account, Password, and FTP Directory (Root). A note says "Must Existed".
- Recipient Email**: Fields for MailBox 1, MailBox 2, and MailBox 3.

At the bottom are "OK" and "Cancel" buttons.

Unfortunately, it could only open from the SD card of the webcam itself (so forget remote control)

Summary

These products are not even approaching most of our initial requirements.

- Expensive for the features they provide
- Low-quality software and/or hardware
- Hard to configure and to use, for different reasons
- Most are not very reliable

Conclusion: the effort of making our own is comparable to the effort needed to make any of these work for us.

So we went on and made one from scratch.



Initial prototypes



Requirements

Optimally, a candidate device should be:

- Reliable → some locations are inaccessible for most of the year
- Power efficient → to run on batteries and/or solar panels
- Sturdy and small → to resist the weather
- Can access the Internet → to upload the pictures
- Have a good quality camera → to take beautiful pictures
- Configurable → for variable conditions and usage
- User friendly → most hut manager are not digital natives
- Cheap → Hut managers are not rich :)

Requirements

Optimally, a camera

- Reliable → solar panel for most of the year
- Power efficient → solar panels
- Sturdy and durable
- Can access the Internet
- Have a good camera
- Configurable
- User friendly → interface
- Cheap → Hunderds of thousands



Using a phone - the good part

- SolarCam.fr too has a used phone in its black box!
- We found a free, open-source Android app ([mobilewebcam](#)) that worked well
- We deployed two webcam built this way and they've been working like a charm for 5 years



This picture was taken by our second phone-based webcam

Using a phone - the bad part

- Mobilewebcam has been abandoned since a decade now
- More recent Android versions lock the Camera API so hard that implementing some functionality requires expertise that we didn't have
- Many features require rooting the device
- Every device would need unique treatment.

Conclusion: too much work, too many questions.

Mobilewebcam's codebase is now in [this repo](#), getting minimal maintenance from me.



The camera

Where else can we find a good camera?

- Cheap digital cameras
- USB IP Webcams
- etc...

Neither of the solutions above is reliable.
No source, no specs, no documentation,
no drivers...

Whatever we could make is hackish and
might become unfeasible as soon as the
device goes out of production or
upgrades its firmware.

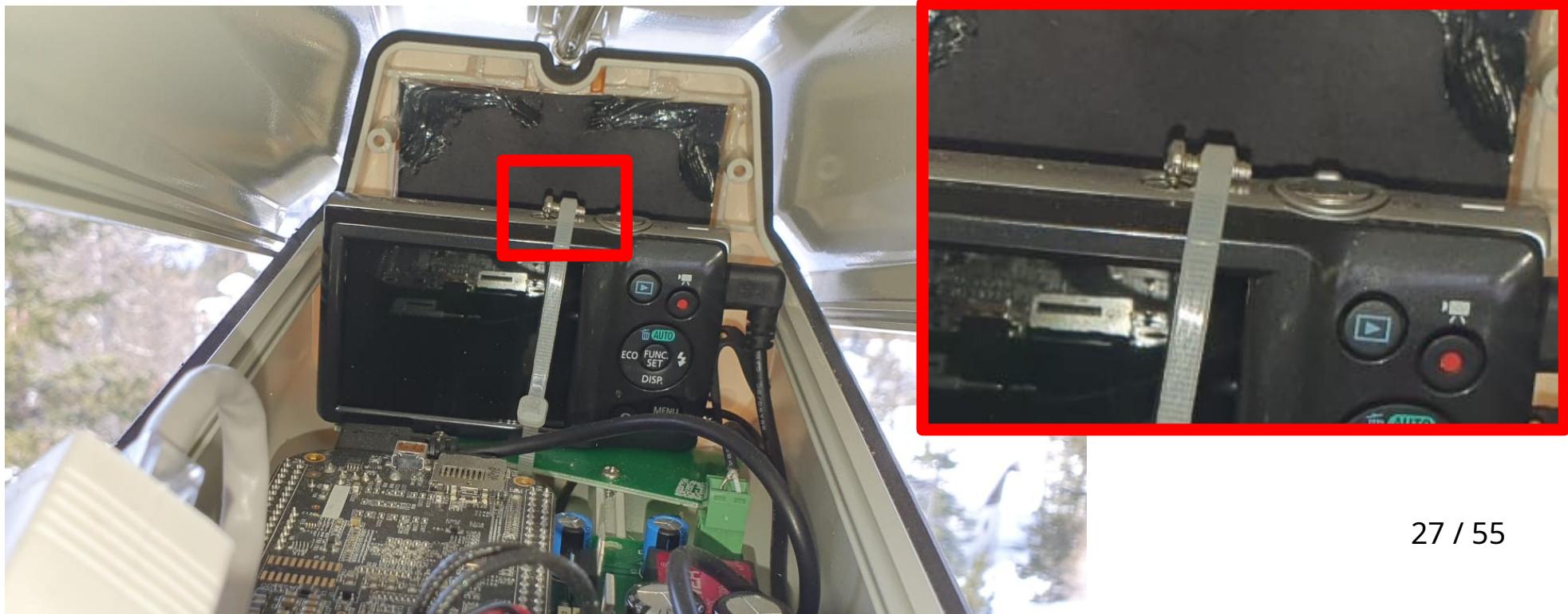


The camera

RifugiInRete chose the digital camera option for their product.

They had to literally tie a screw to the shutter button of the camera to keep it on while plugged in.

That's what "no specs, no docs" means.



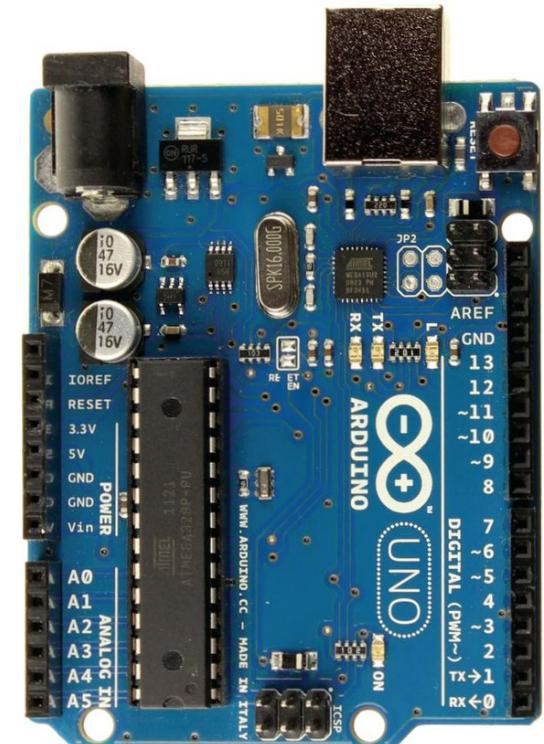
Down the rabbit hole

- We did not want to go down that path. We needed an alternative that could last for long, be reproducible in 5-10 years with small adjustments.
- So we took a leap of faith and decided to build the system from its components: single-board computers, microcontrollers, and their cameras



The Arduino prototype

- We developed a small PoC with an Arduino board and ArduCAM.
- We managed to take full resolution pictures!
- We wanted to be able to add some text over the image... but image processing immediately proved too much for the PoC.
- Conclusion: Arduino is good for simple cameras, but image handling goes too close to its limit and adding any feature is impossible.



The Raspberry Pi prototype

- Specs felt excessive for the task, and the power consumption too!
- Raspberry Pi is not optimized for low-power operations and keeps drawing power even when fully turned off
- The leanest one, RPi Zero W, still draws 90mA when idle and around 250mA when shooting pictures.
- We initially considered it to be too much to survive for long on batteries and we looked for alternatives.



Is there really any alternative?

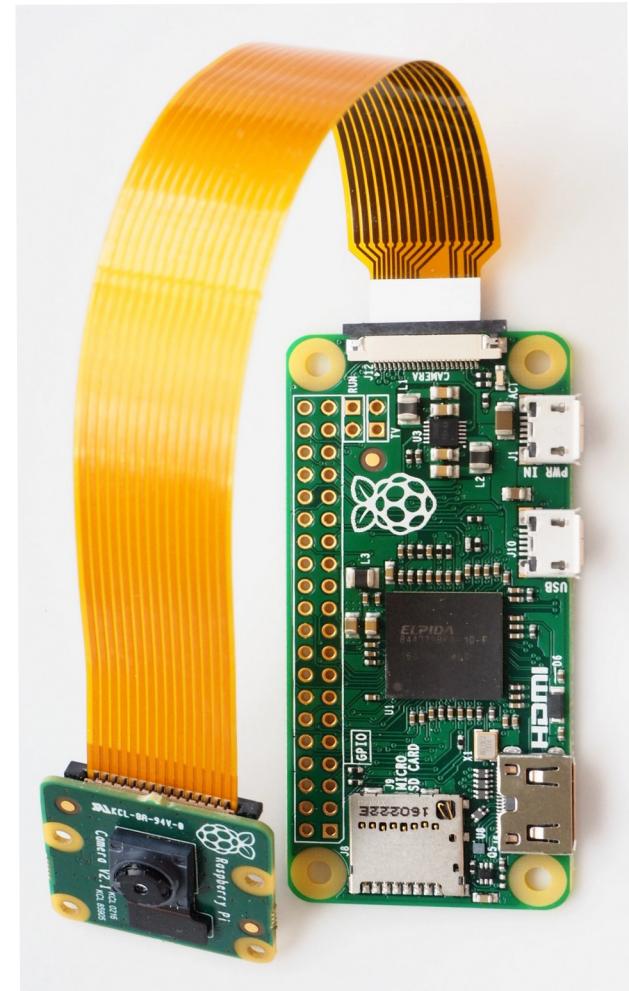
- The research was disappointing:
 - A Canadian producer (CubeCamera): nice, but the shipping costs!!
 - A German producer that promised the chips would be available by Q1 2021 (it was September 2020)
- Alternative boards were **very expensive** or demanded a commercial partner making **orders of thousands of parts**.
- If you know any “small scale” producer in this area I beg you to raise your hand NOW



PHYTEC

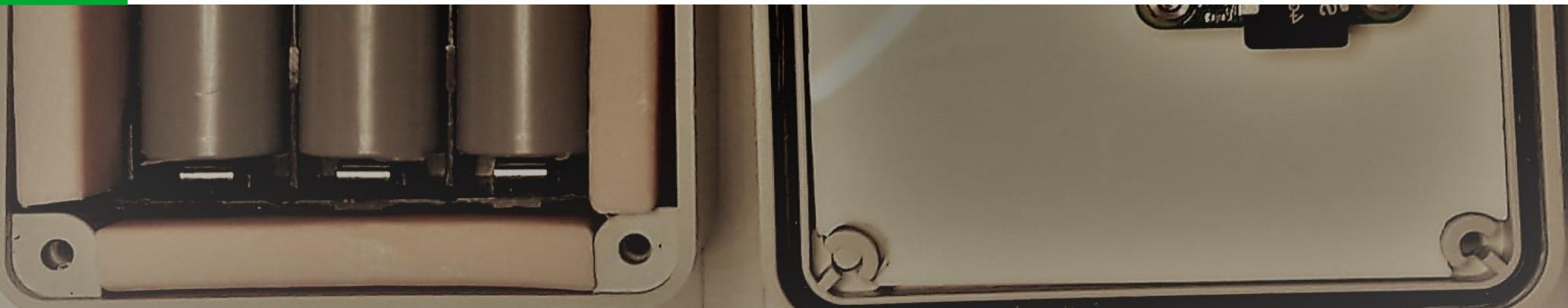
The decision

- Everything considered, it seemed only one way ahead was left: the Raspberry Pi option.
- We decided for the following plan:
 - An external RTC board to try control the power draw
 - Bigger batteries, larger solar panels (even though it means €€€)
 - Keep an eye on the market for better boards





The result



Requirements

How many of these requirements we matched?

- Reliable → to test
- Power efficient → no
- Sturdy and small → yes
- Can access the Internet → yes
- Have a good quality camera → yes
- Configurable locally and remotely → yes
- User friendly → hopefully
- Cheap → so-so

Overview

ZanzoCam is made of

- A customized Raspberry Pi OS image
- A Raspberry Pi Zero W
- A Raspberry Pi Camera v2 (or HQ)
- A “dumb” server (more on this soon)
- A waterproof case (we build our own)

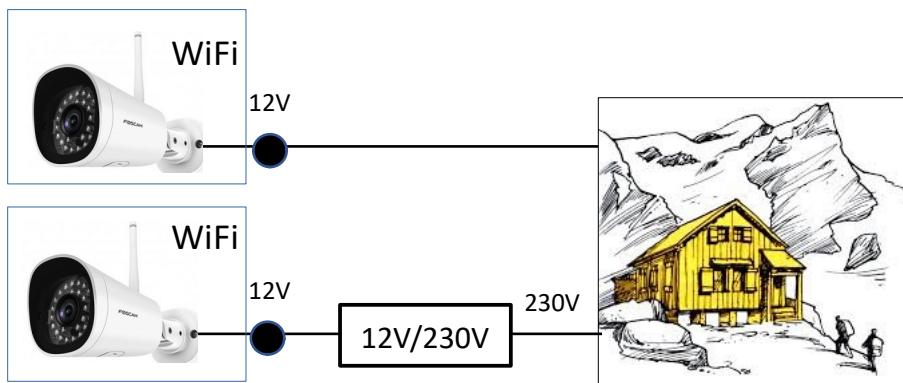
Then, depending on your setup:

- Either a power cable or a solar panel
- A WiFi router, an Ethernet cable or a modem

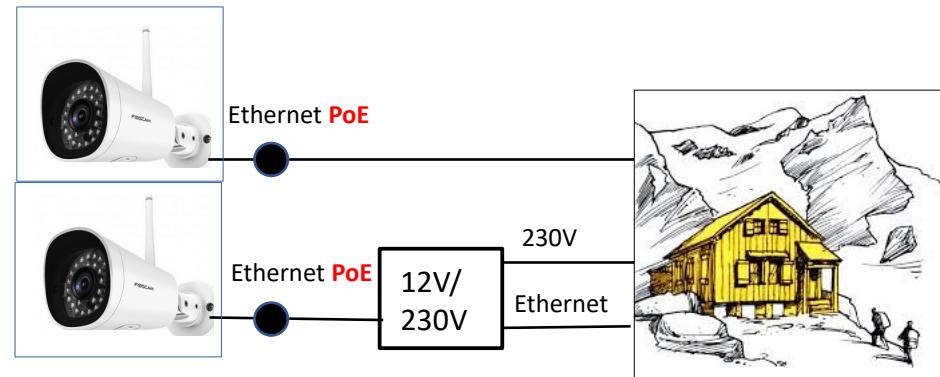


The 4 hardware configurations

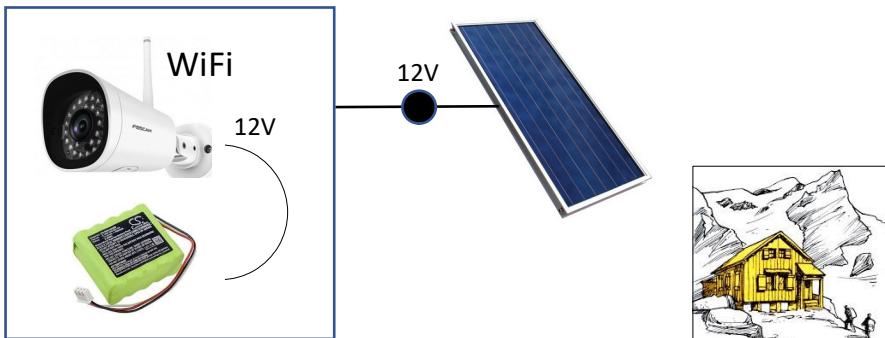
A: WiFi and Power line



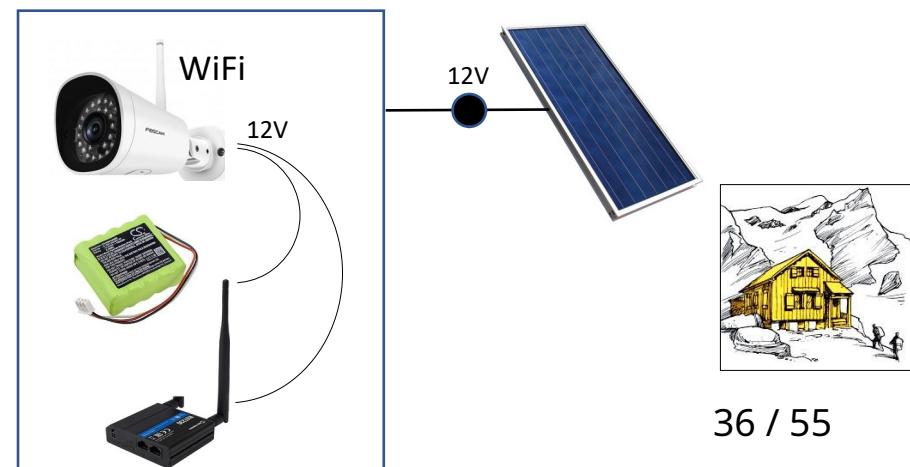
B: Ethernet (PoE)



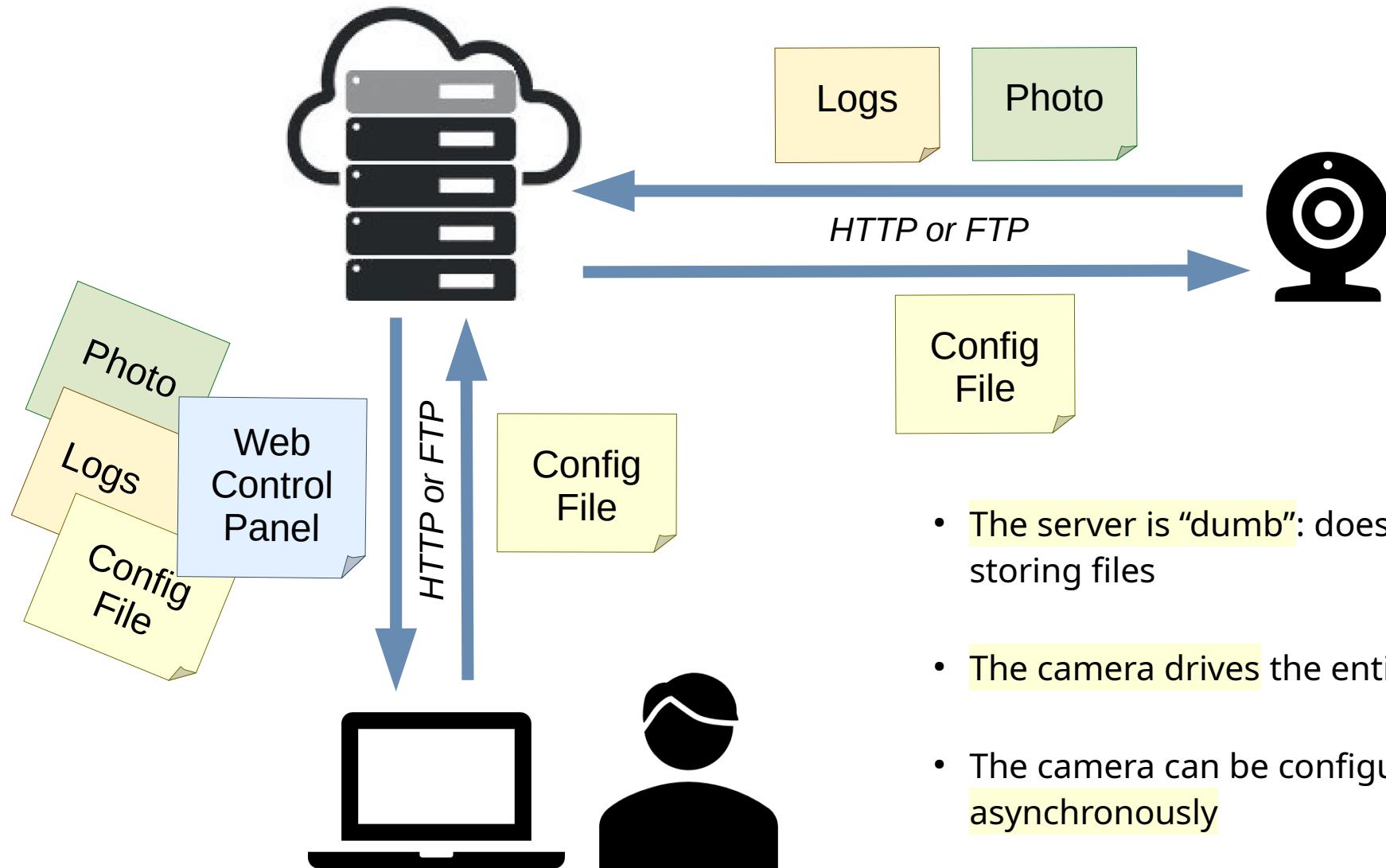
C: WiFi and Solar



D: Router and Solar

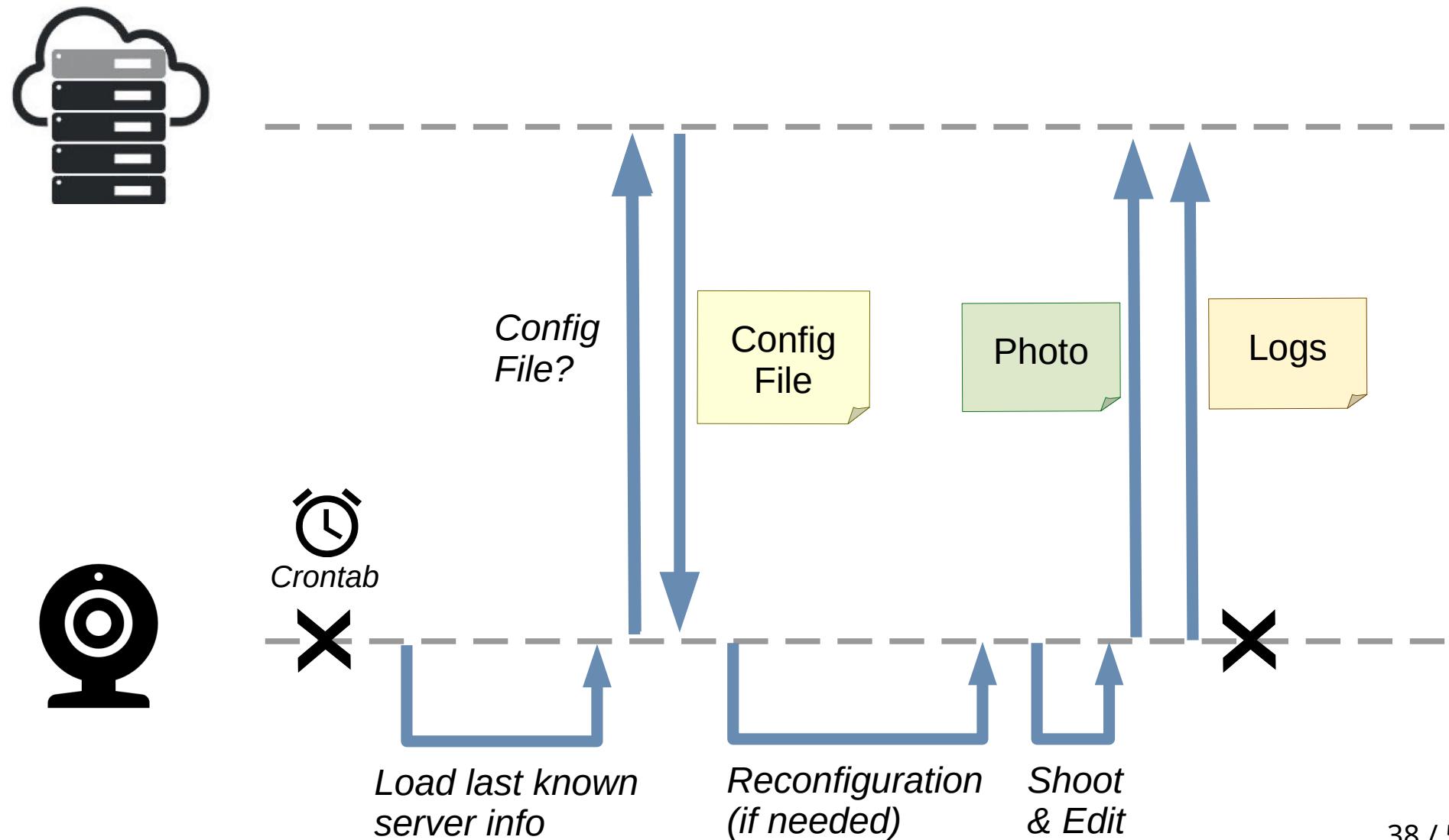


The software

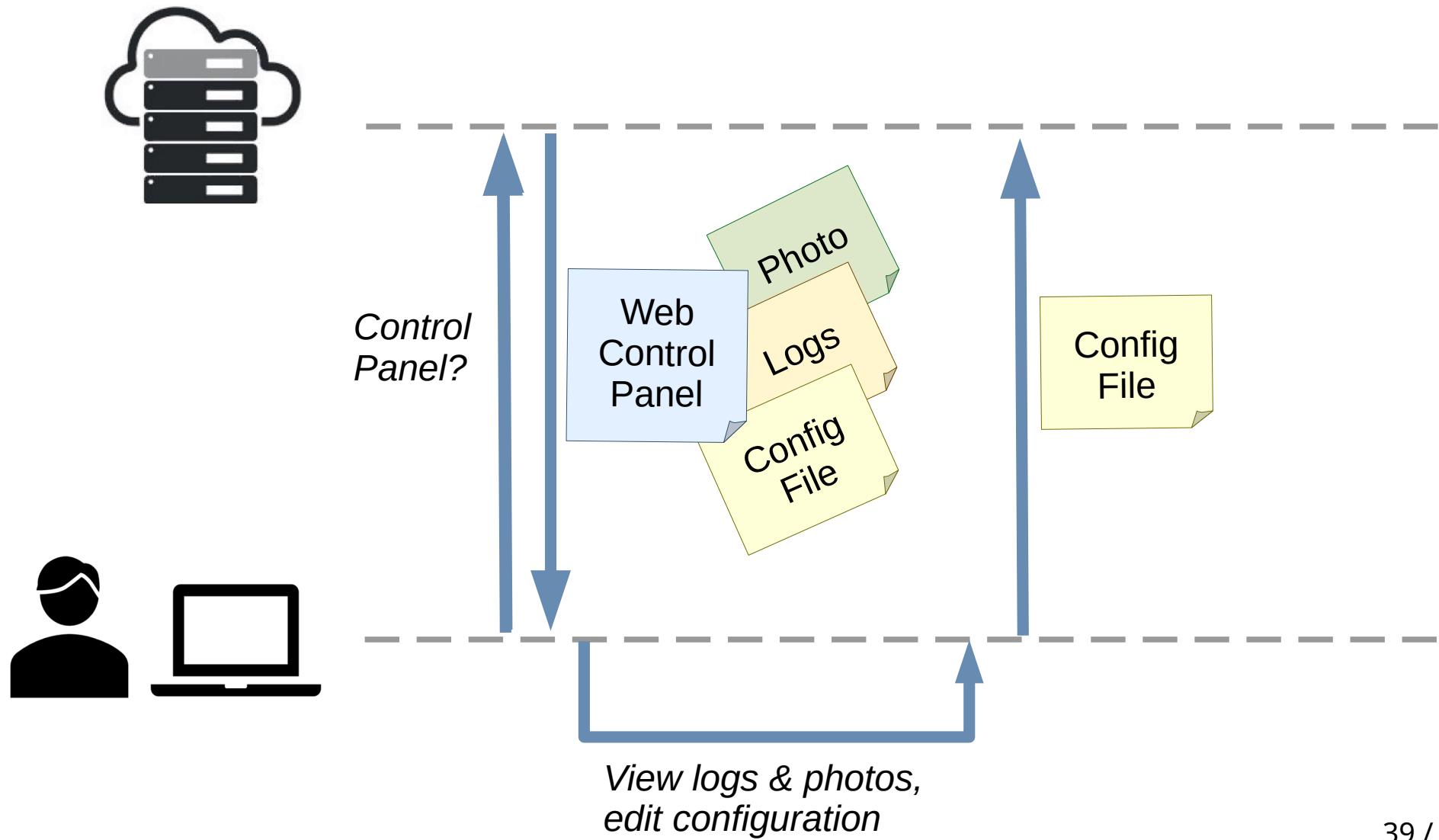


- The server is “dumb”: does nothing but storing files
- The camera drives the entire process
- The camera can be configured asynchronously

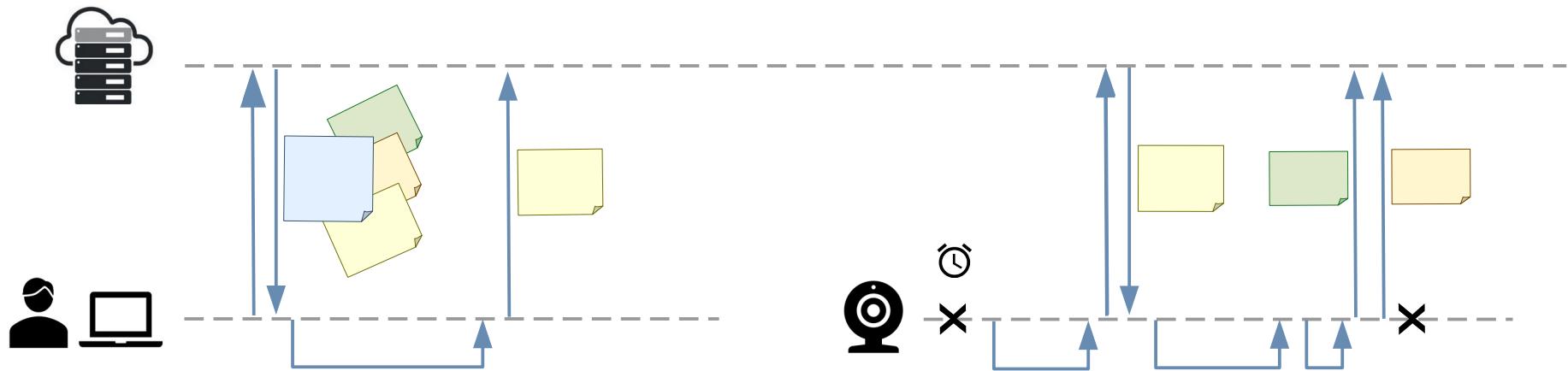
The camera flow



The user flow



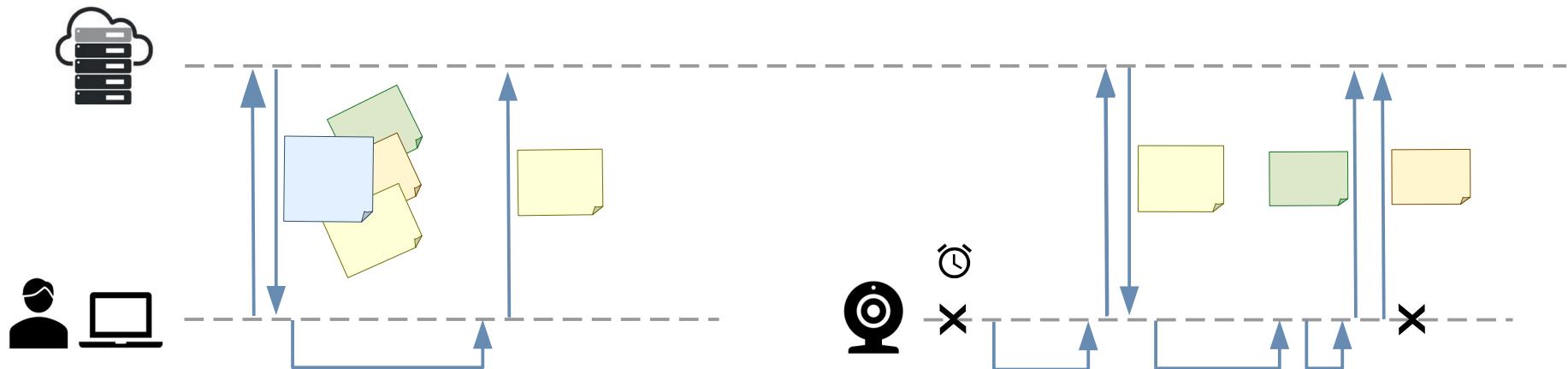
The overall flow



This architecture is the contrary of a “cloud” solution: the server contains no logic at all. Simple FTP folders are sufficient.

The camera is configurable, but not controllable (no static/public IP, no domain name, no VPN, no direct remote access whatsoever is needed) = smaller attack surface too.

The overall flow



PRO:

- Simple (monolithic)
- Decentralized
- Quite secure

CON:

- No direct control, which means:
 - Harder to debug
 - Harder to install
- More power-hungry

The web interface

Simple for the user: most hut owners are not good with tech!

The screenshot shows the Zanzocam web interface. At the top, there is a logo with the letters 'Zc' in a stylized font, followed by 'ZANZOCAM' and a small camera icon. Below the logo, the text 'V0.10.0' is visible. The main title 'Impostazioni' (Settings) is centered above a form. The form is divided into two sections: 'Server:' and 'Orario:'. The 'Server:' section has tabs for 'HTTP' and 'FTP', with 'FTP' selected. It contains fields for 'Hostname' (caisovico.it), 'Sottocartella (se necessaria)' (empty), 'Avanzate' (checkbox 'Usa TLS' is unchecked), 'Username (se necessario)' (zanzocam1@caisovico.it), and 'Password (se necessaria)' (redacted). A 'MOSTRA' (Show) button is next to the password field. The 'Orario:' section includes fields for 'Inizia a scattare alle:' (04:00), 'Smetti di scattare alle:' (22:05), and 'Frequenza scatti:' (Ogni ora). At the bottom of the settings page, there is a link 'Configurazione manuale'.

The web interface

Simple for the developer: ME! Who has time to waste on debugging?

The screenshot shows a web-based configuration interface for a device named 'ZANZOCAM'. The interface is divided into two main sections: a left panel for manual configuration and a right panel for viewing and editing JSON data.

Left Panel (Configuration):

- Server:** A tabbed section with **HTTP** (disabled) and **FTP** (selected).
- Hostname:** caisovico.it
- Sottocartella (se necessaria):** An empty input field.
- Username (se necessario):** zanzocam1@caisovico.it
- Password (se necessaria):** A masked password entry.

Orario:

- Inizia a scattare alle: 04:00
- Smetti di scattare a: 22:05

Bottom: Configurazione manuale

Right Panel (JSON View):

JSON Raw Data Headers

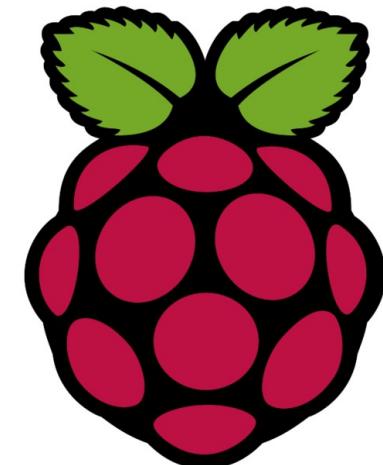
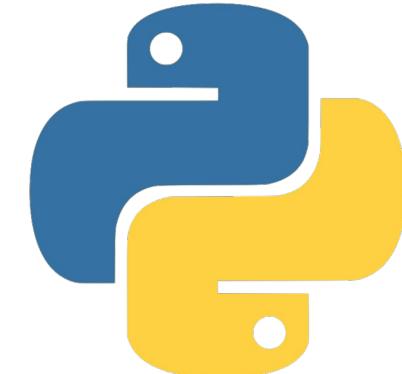
Save Copy Collapse All Expand All Filter JSON

```
configuration:
  server:
    protocol: "FTP"
    hostname: "caisovico.it"
    tls: false
    username: "zanzocam1@caisovico.it"
    password: "zanzocam1"
    max_photos: "10"
  time:
    start_activity: "04:00"
    stop_activity: "22:05"
    frequency: "60"
  image:
    name: "foto-test"
```

The camera software

The system is simple:

- Raspberry Pi OS Lite + a few packages
- One Python package
 - Very quick to develop (it's my free time after all)
 - As simple as it could possibly be, and documented (I forget my own code too)
 - 98% test coverage (never trust interpreted languages in autonomous systems)
- Small customizations (swap disabled, crontabs, etc)



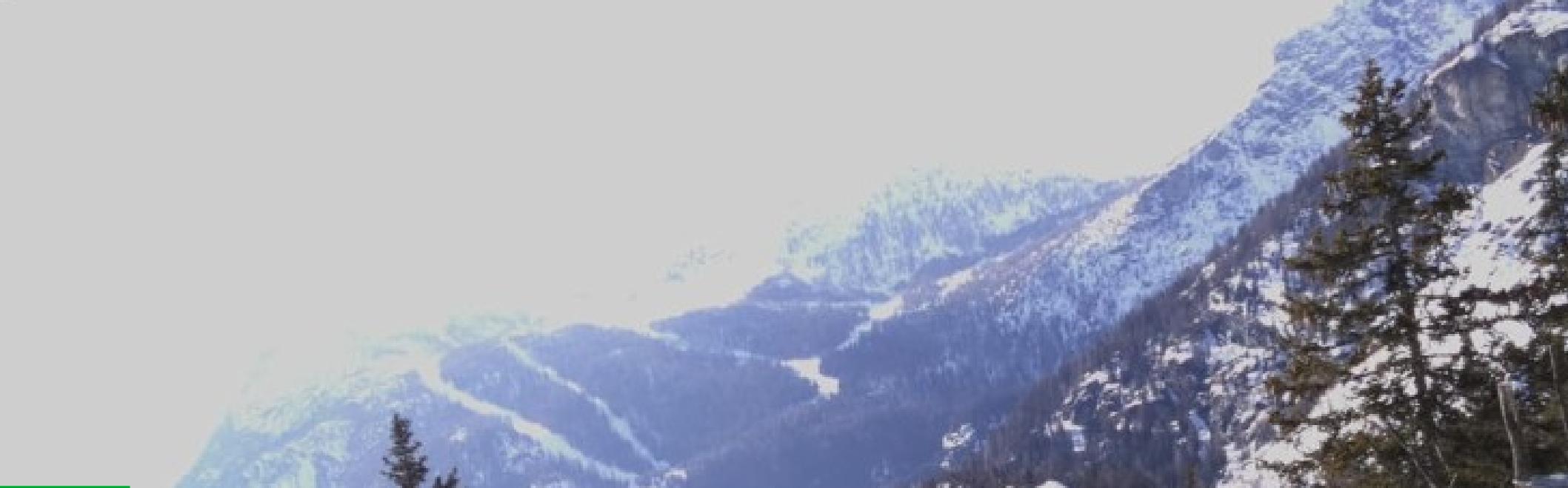
Everything is open source ([enjoy](#)).

It's not perfect yet!

The system can definitely be improved:

- The web interface is a bit of a JS mess
 - Redoing it slowly in Svelte
- Raspberry Pi OS is too fat
 - Make a custom buildroot
- SD cards can't take many writes
 - Use a ramdisk to minimize them
- Improve the initial setup flow
 - Currently a bit awkward
- etc...





Other hurdles



The team

- *Sara*: the only developer
- *Giorgio*: the man with the money
- *Paolo*: the electronics master
- *Michele*: the smart high-schooler



If you've seen the Pokemon you know where this is going...

Communication

- Non-developers have hard time understanding why some tradeoffs have to be done, and communication is tiring.
 - Nobody bothered me for choosing Python (arguable at best)
 - I've been asked why cameras aren't accessible remotely when the RTC turns them off
 - Once the system was complete, the specs changed to include VPN access to the cameras – even the RTC ones
 - I had to add support for HQ cams (12MP) when we have lenses graded for 3MP maximum
 - And many others...
- We learned that the best communication sometimes is no communication

The money

We originally planned to redistribute the project for free to CAI members and beyond.

Then, Giorgio got a small grant from CAI to pay for the raw materials: nice, right?

Now CAI transformed the whole thing into a business:

- Paolo will assemble 20+ webcams by hand for pocket money, and hut owners will pay 200-600€ (above market prices!)
- We can't own the pictures made by the ZanzoCam, but have to send them to their servers "because they paid for it"



The catch

Did I mention that I'm the only software developer involved in the entire project?

- Nobody put any ties on the IP of the ZanzoCam
- Paolo approves the idea of redistributing his instructions on the hardware assembly
- Nobody else knows what Open Source means and have no opinion on it

Result: I am free to open-source the whole thing. Skilled people unwilling to pay will be able to do so with all my support.



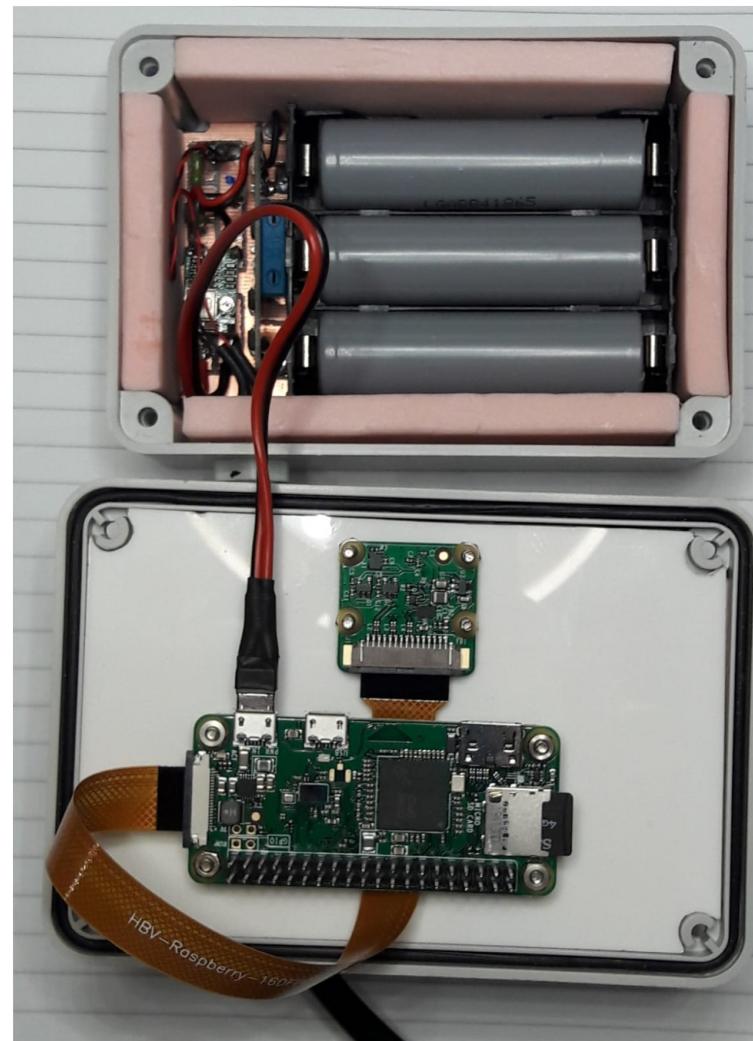
Free as in Freedom

Why caring?

- There seem to be no simple, open-source (or at least not locked down) product in this area
- Plenty of open-source has been used for this product: the least I can do is to give something back
- This niche is very diverse and very small: the best usecase for customizable FOSS software
- I've already met a few people facing the same issue, while hosting mobilewebcam under my GitHub
- I personally believe FOSS is better than closed software, for users and often for companies too – but there's little awareness among consumers

Lessons learned

- If you need one or two remote webcams and you have an old phone, that's still the best option
 - There might be better options than mobilewebcam out there
- The Raspberry Pi was not that much overboard with the specs!
- We could probably afford a more expensive board with better power management
- Do not accept money from strangers...



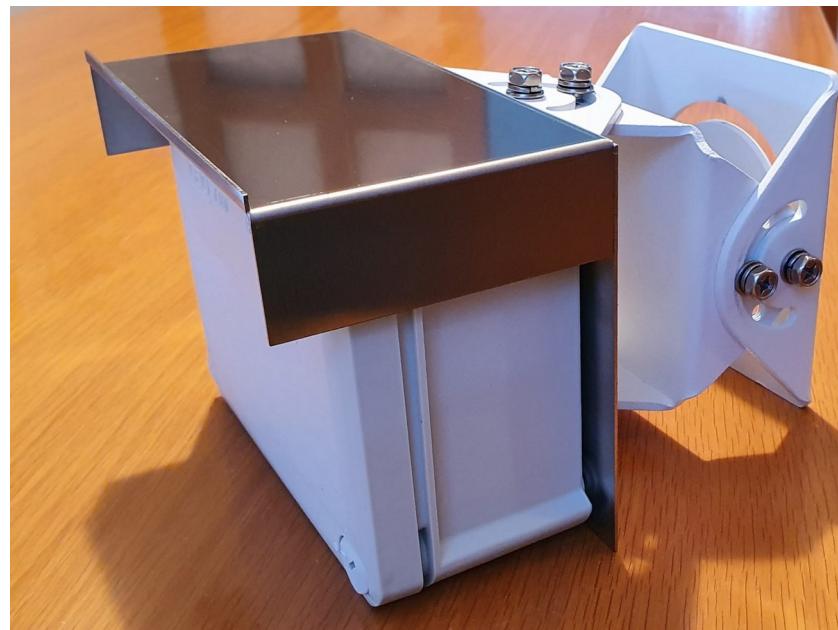
The inside of a battery-powered ZanzoCam



The Future

What now?

- ZanzoCam is not complete!
 - Many features are still missing: RTC, modem, even Ethernet
 - We will start the field tests this summer and winter
 - Many improvements can still be done
- Some aspects have been neglected due to incompetence:
 - Security (*yeah that's bad...*)
- We will train some younger CAI member for the long-term maintenance



A ZanzoCam ready for a field test

The background of the image is a dark, star-filled night sky. At the bottom, there is a silhouette of a mountain range with some snow-capped peaks. The sky transitions from dark blue at the top to a warm orange and yellow glow near the horizon.

That's all folks!