# The Frankencamera: An Experimental Platform for Computational Photography


Andrew Adams, Eino-Ville Talvala, Sung Hee Park,
David E. Jacobs, Boris Ajdin, Natasha Gelfand,
Jennifer Dolson, Daniel Vaquero, Jongmin Baek,
Marius Tico, Hendrik P. A. Lensch, Wojciech Matusik,
Kari Pulli, Mark Horowitz, Marc Levoy

# Viewfinder Alignment

- A pixel-accurate alignment algorithm that runs at 320x240 at 30fps on an N95
- Low-noise viewfinding
  - Align and average a moving window of previous frames
- Panorama capture
  - automatically take new images when the view has moved to a new location
- ...

# Computational Photography

```
for (...) {
  Change camera settings
  Take picture
}
Combine the pictures
```

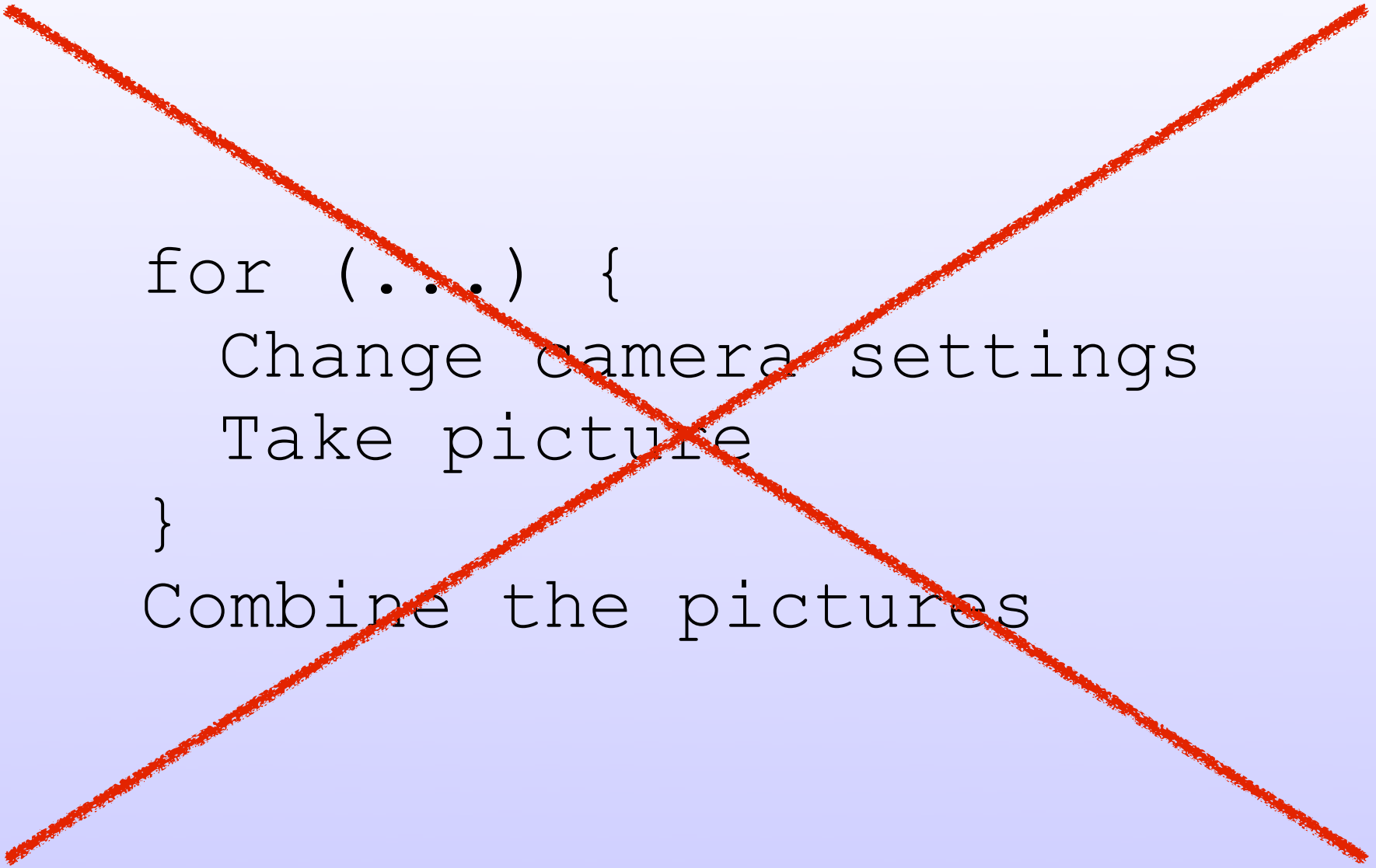
# Problem 1: Platform is closed

- On N95, no control over
  - exposure time
  - white balance
  - focus
  - frame rate
  - image format/resolution
  - post-processing pipeline parameters
  - metering algorithm
  - autofocus algorithm
- iPhone/Android is equivalent or worse
- "Real" cameras can't be reprogrammed at all

# Problem 2: Wrong sensor model

- Real image sensors are pipelined
  - While one frame is post-processing,
  - the next one is exposing,
  - and the sensor is being configured for the one after that
- N95 Viewfinding mode:
  - Pipelined, high frame rate
  - Settings changes take effect an unknown number of frames into the future
- N95 Still Capture mode:
  - Not pipelined
  - Throughput (frame rate) = 1/pipeline latency

# Computational Photography



# A Programmable Camera Platform

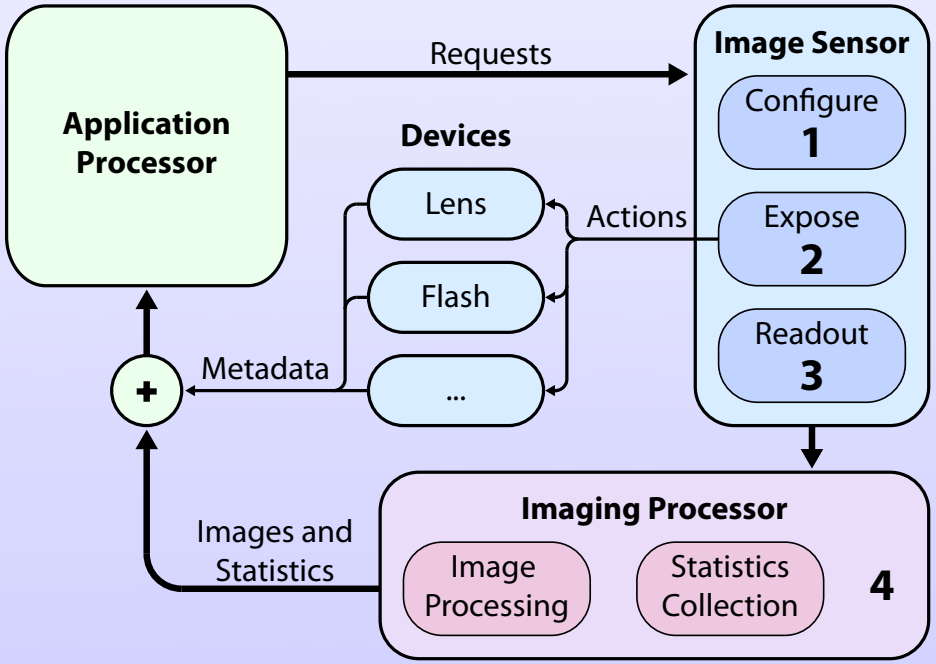
1. Should be open, all the way down
2. Should be able to capture or stream bursts of images with deterministically varying settings at full frame rate
3. Should have enough compute and memory to do computational photography
4. Should be easy to program for using standard tools
5. Should be a credible walking-around camera

# The Frankencamera

- **A system architecture for programmable cameras**
- Two implementations
- An API to program for the architecture (FCam)
- Example applications

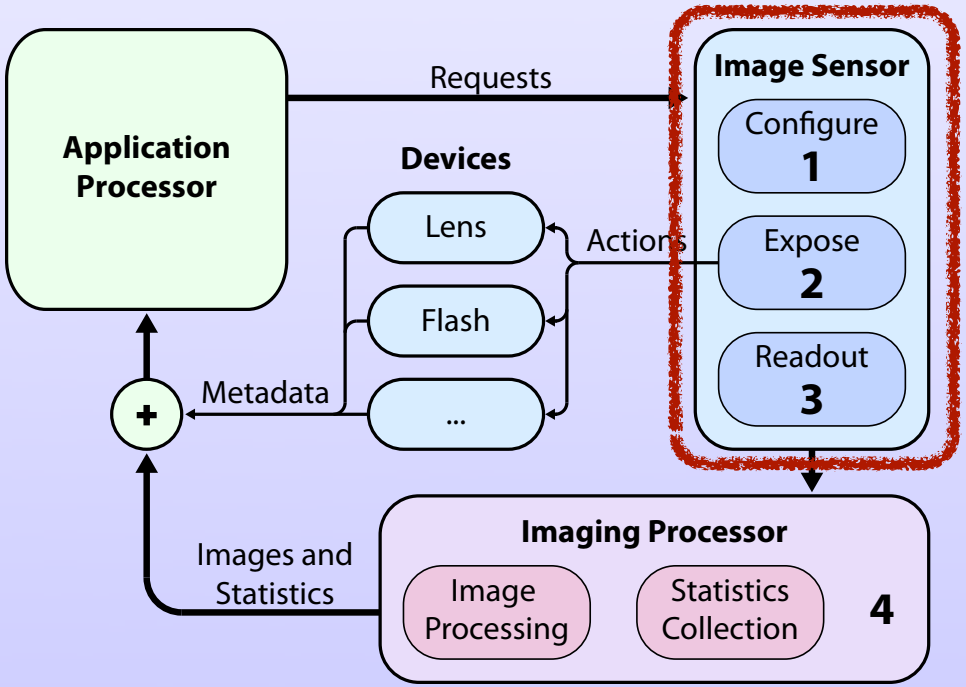
# The Frankencamera Architecture

- A general architecture for programmable cameras
- All settings are embedded in the requests and frames flowing through the imaging pipeline.



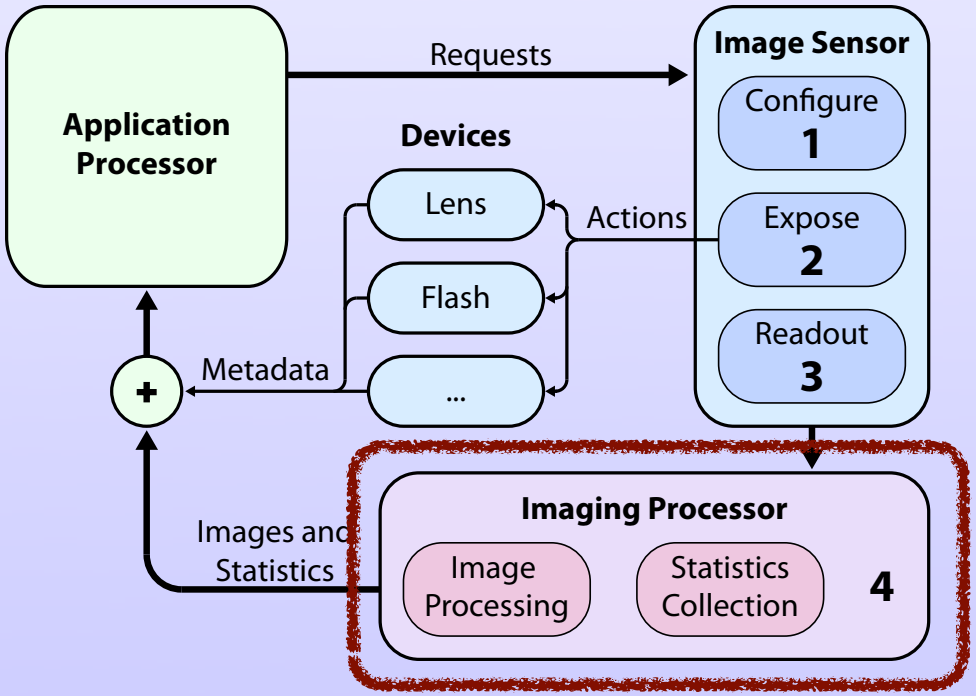
# The Sensor

- The sensor is a pipeline that converts **requests** for images into images
- The sensor has no visible state. A request for an image specifies all parameters to be used in that image's capture.



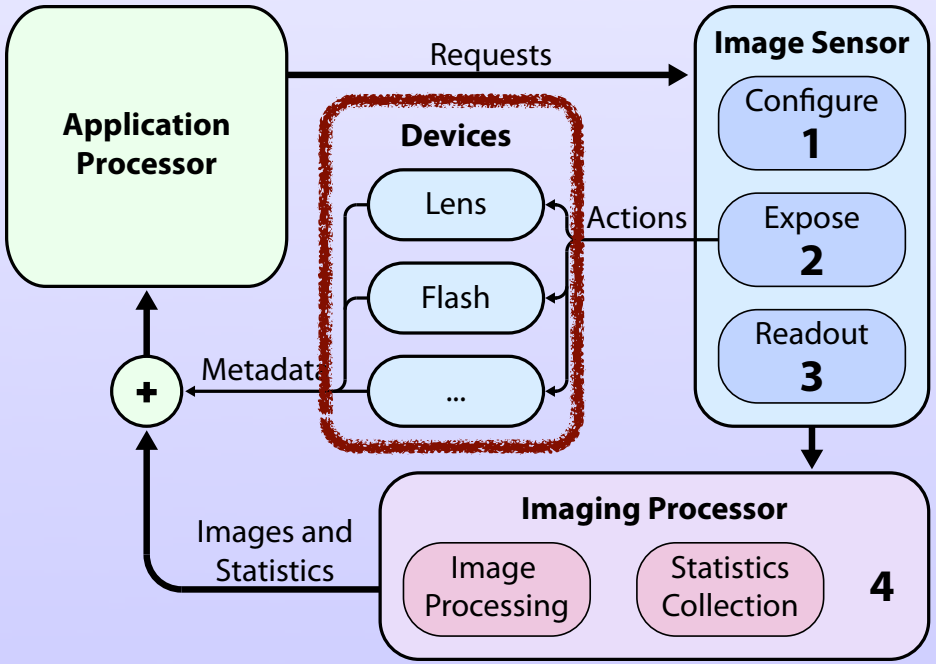
# The Image Signal Processor (ISP)

- Receives sensor data, and optionally transforms it
- Untransformed raw data must be available to the application if requested
- May create histograms, other statistics for each image



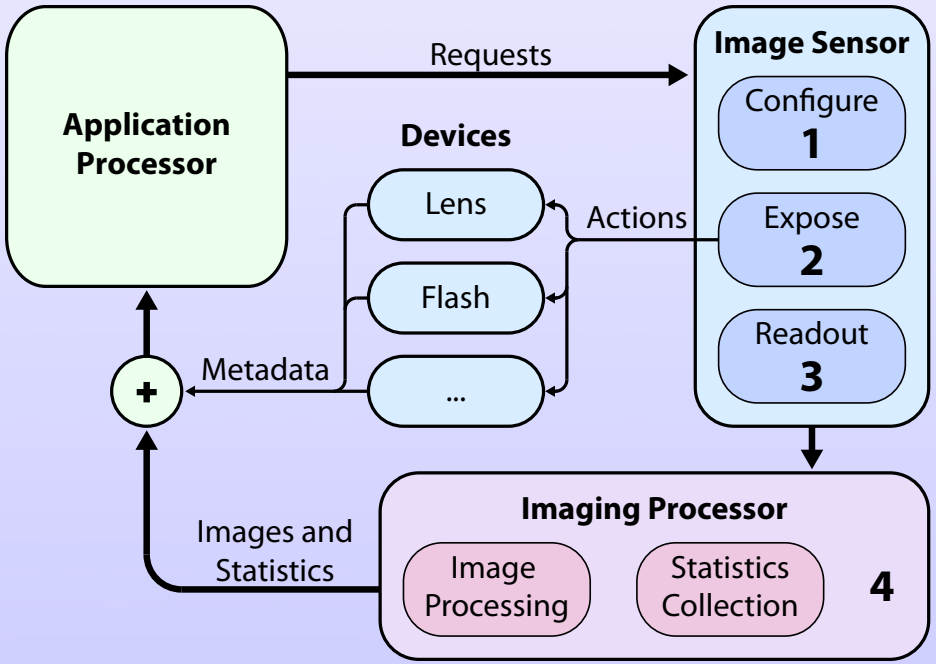
# Other Devices

- Other devices (like the lens and flash) can schedule **Actions** to be triggered at a given time into an exposure, and can **tag** returned images with extra metadata.



# Everything is visible

- No hidden daemon running autofocus/metering
- Programmer has full control over sensor settings, and full access to the supplemental statistics the ISP computes for each frame.



# The Frankencamera

- A system architecture for programmable cameras
- **Two implementations**
- An API to program for the architecture (FCam)
- Example applications

# Implementations

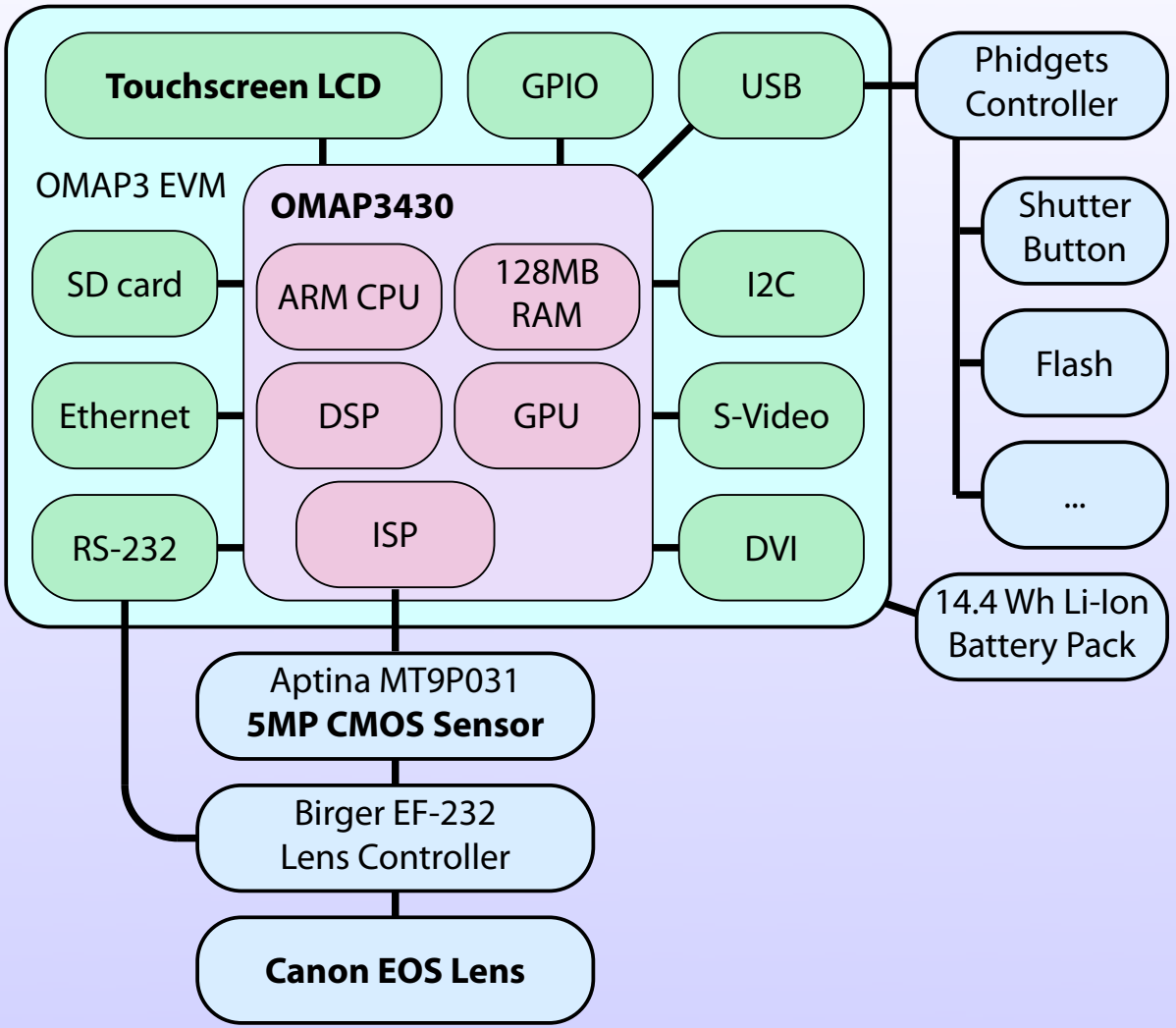
F2 Frankencamera


Nokia N900

# Implementations

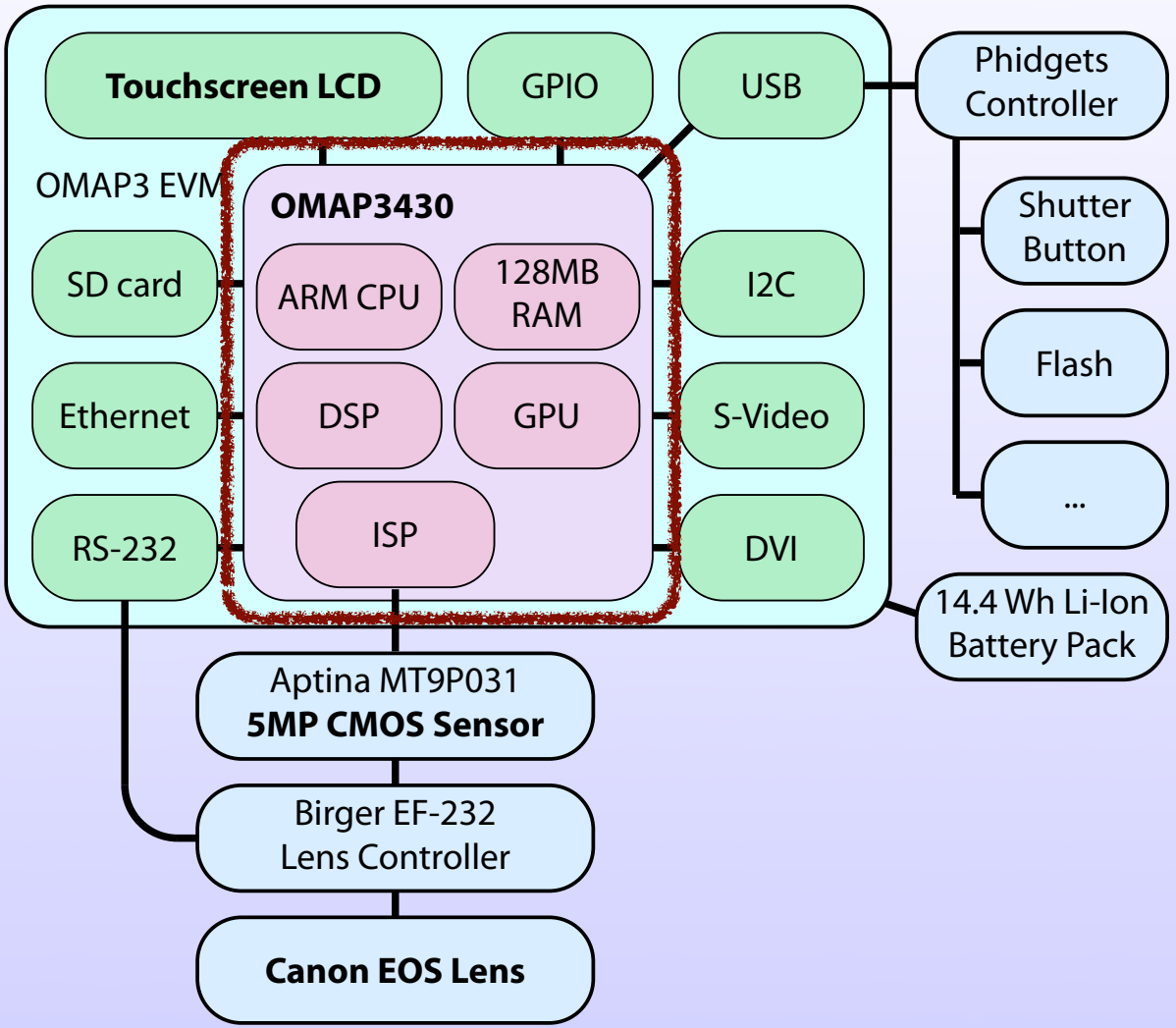
F2 Frankencamera - Internals

# Implementations



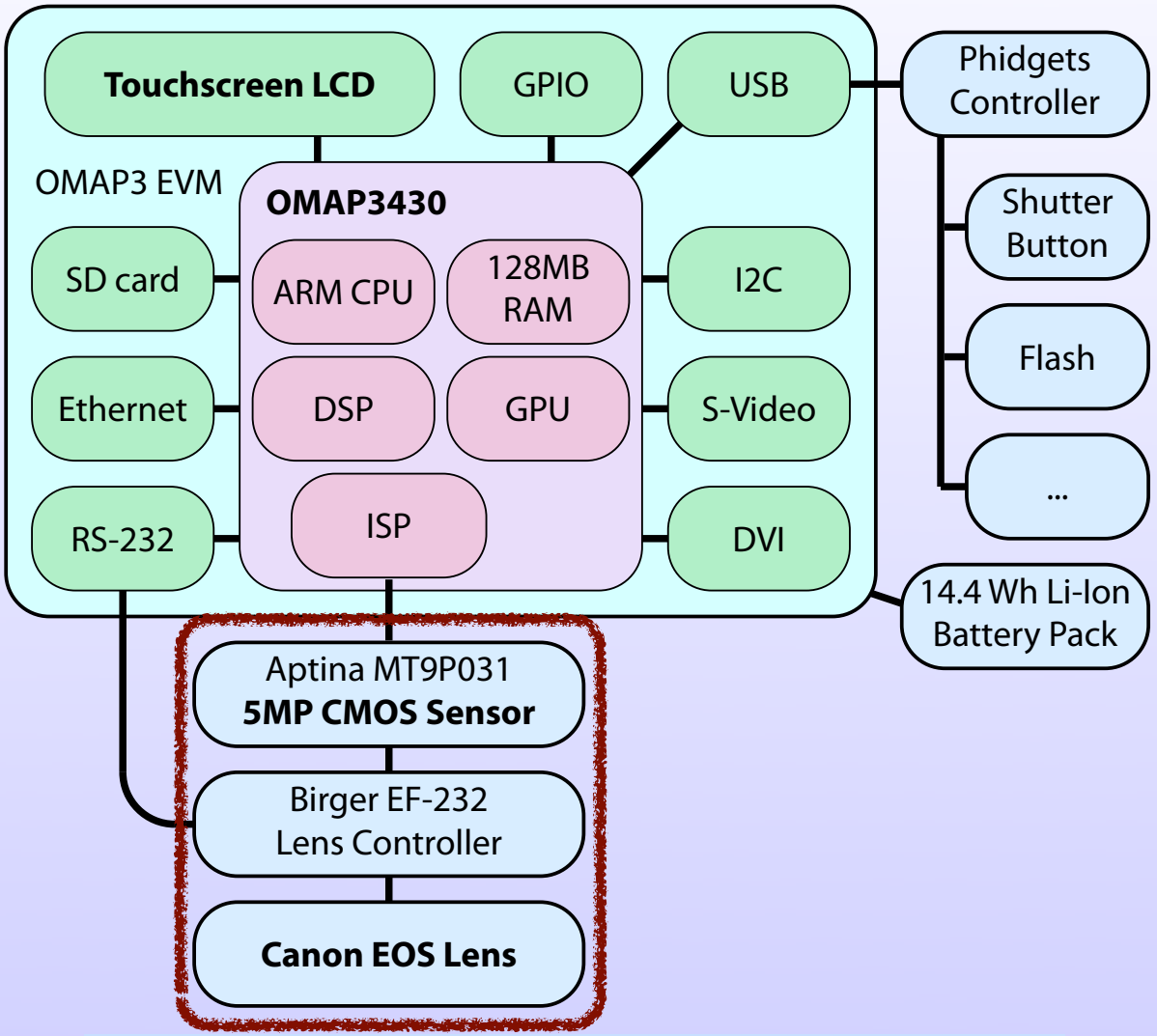

F2 Frankencamera - Internals

# Implementations



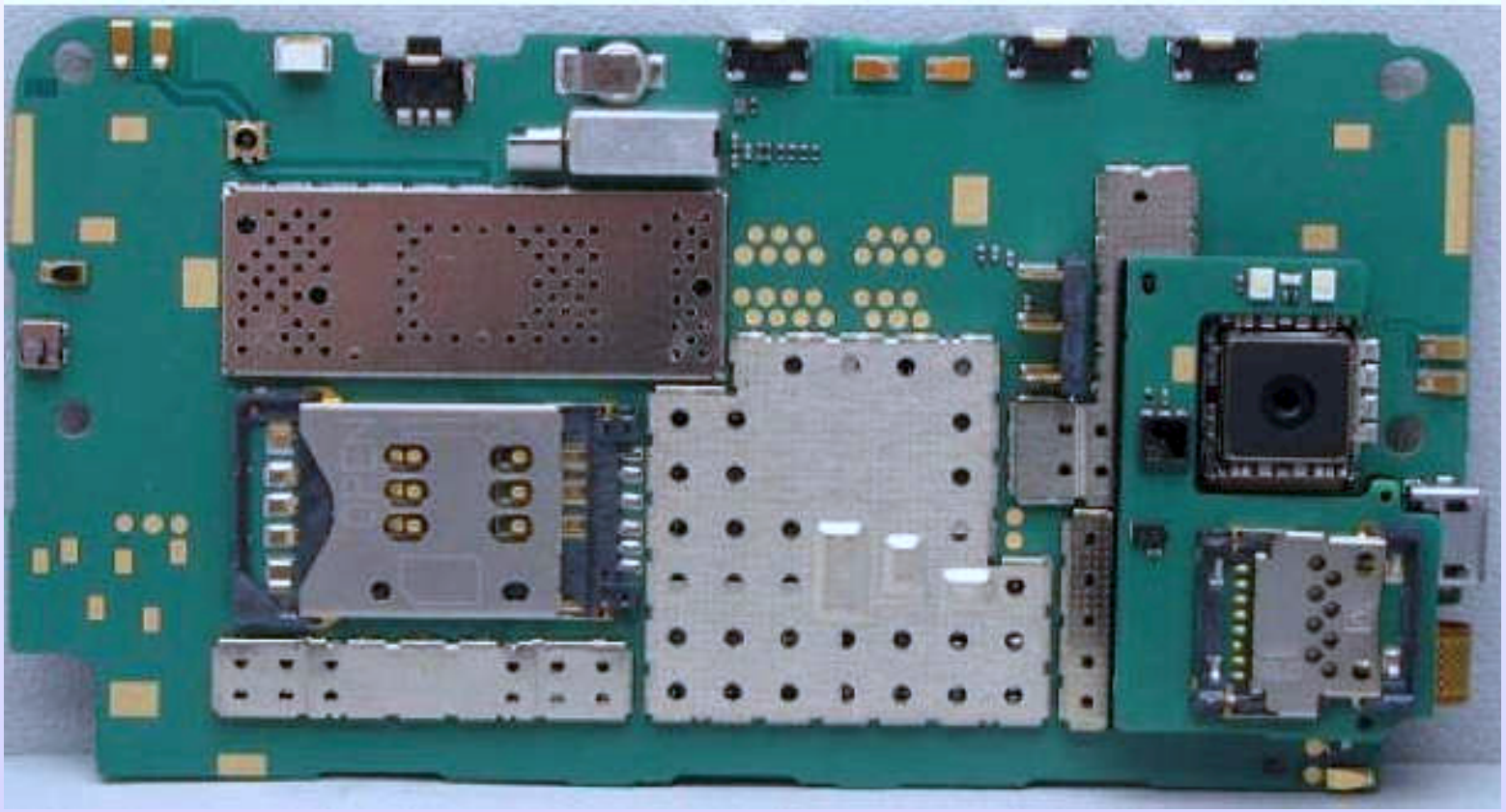

F2 Frankencamera - Internals

# Implementations



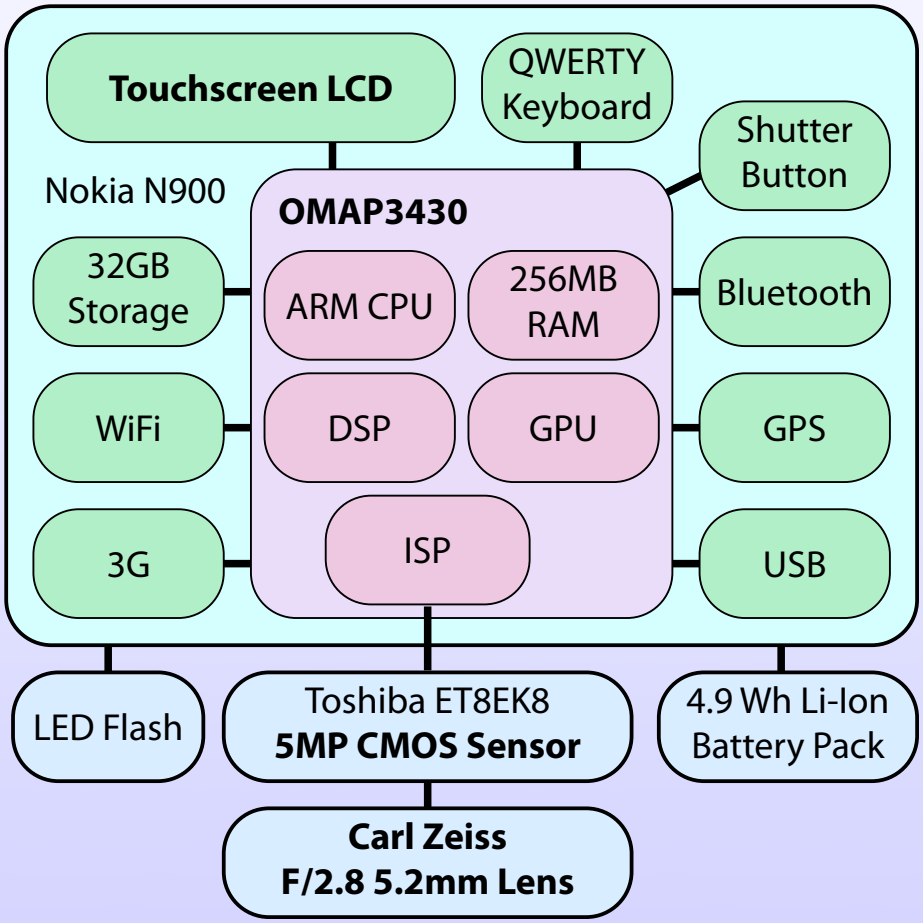

F2 Frankencamera - Internals

# Implementations



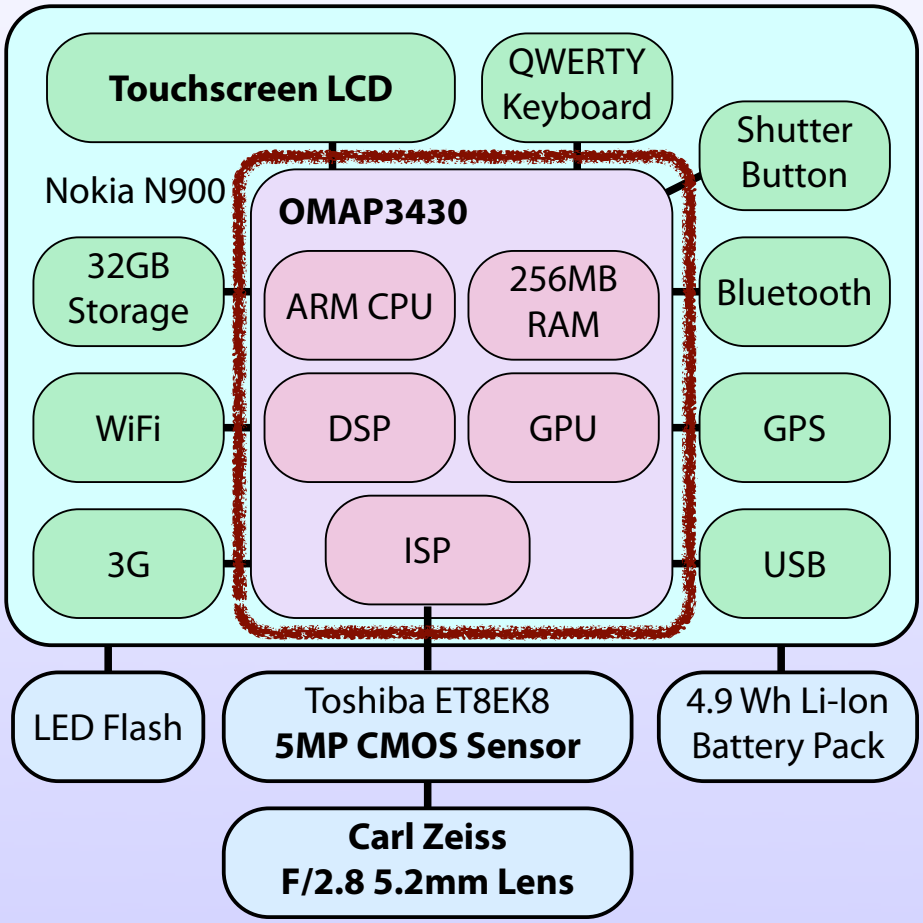
Nokia N900 - Internals

# Implementations



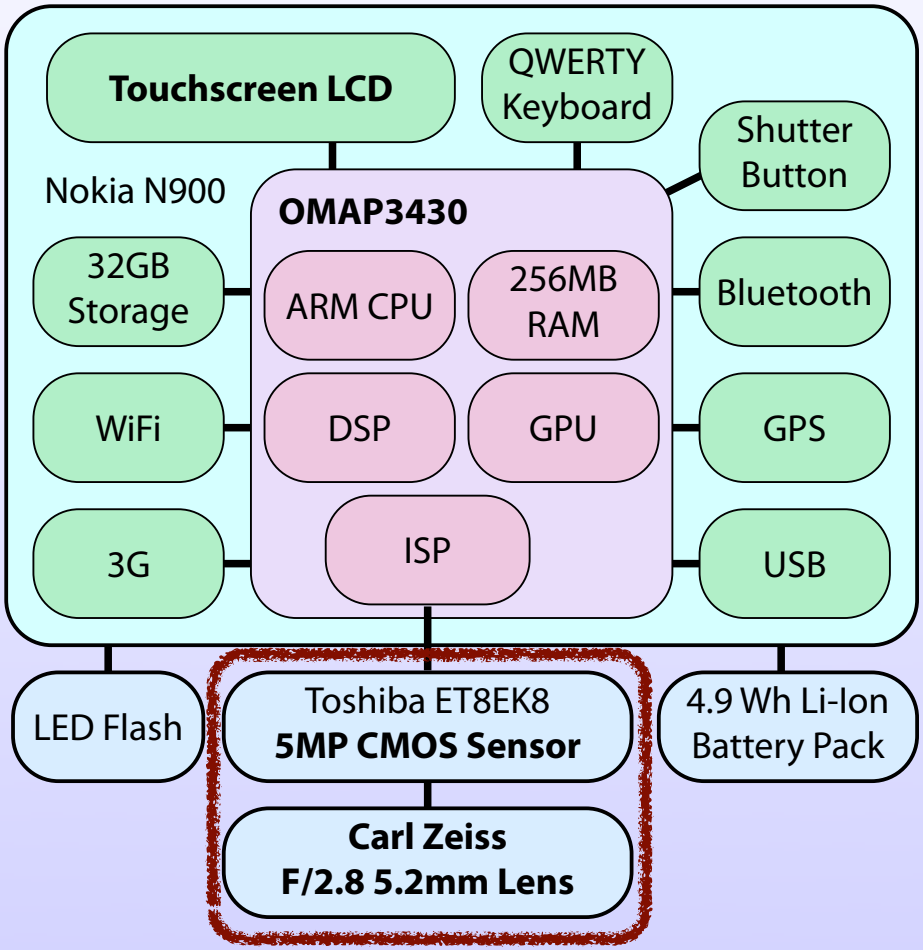

Nokia N900 - Internals

# Implementations




Nokia N900 - Internals

# Implementations




Nokia N900 - Internals

# The Frankencamera

- A system architecture for programmable cameras
- Two implementations
- **An API to program for the architecture (FCam)**
- Example applications

# Simple HDR Burst

```
#include <FCam/N900.h>
...
```

# Simple HDR Burst

```
#include <FCam/N900.h>
...
  Sensor sensor;
  Shot shortReq, midReq, longReq;
  Frame short, mid, long;
```

# Simple HDR Burst

```
#include <FCam/N900.h>
...
    Sensor sensor;
    Shot shortReq, midReq, longReq;
    Frame short, mid, long;

    shortReq.exposure =  10000; // microseconds
    midReq.exposure   =  40000;
    longReq.exposure  = 160000;
    shortReq.image = Image(sensor.maxImageSize(), RAW);
    midReq.image   = Image(sensor.maxImageSize(), RAW);
    longReq.image  = Image(sensor.maxImageSize(), RAW);
```

# Simple HDR Burst

```
#include <FCam/N900.h>
...
    Sensor sensor;
    Shot shortReq, midReq, longReq;
    Frame short, mid, long;

    shortReq.exposure =  10000; // microseconds
    midReq.exposure   =  40000;
    longReq.exposure  = 160000;
    shortReq.image = Image(sensor.maxImageSize(), RAW);
    midReq.image   = Image(sensor.maxImageSize(), RAW);
    longReq.image  = Image(sensor.maxImageSize(), RAW);

    sensor.capture(shortReq);
    sensor.capture(midReq);
    sensor.capture(longReq);
```

# Simple HDR Burst

```
#include <FCam/N900.h>
...
    Sensor sensor;
    Shot shortReq, midReq, longReq;
    Frame short, mid, long;

    shortReq.exposure =  10000; // microseconds
    midReq.exposure   =  40000;
    longReq.exposure  = 160000;
    shortReq.image = Image(sensor.maxImageSize(), RAW);
    midReq.image   = Image(sensor.maxImageSize(), RAW);
    longReq.image  = Image(sensor.maxImageSize(), RAW);

    sensor.capture(shortReq);
    sensor.capture(midReq);
    sensor.capture(longReq);

    short = sensor.getFrame();
    mid = sensor.getFrame();
    long = sensor.getFrame();
```

# HDR Viewfinder with metering

```
#include <FCam/N900.h>
...



...
  while(1) {









  }
```

# HDR Viewfinder with metering

```
#include <FCam/N900.h>
...
  vector<Shot> hdr(2);
  hdr[0].exposure = 40000;
  hdr[1].exposure = 10000;
...
  while(1) {









  }
```

# HDR Viewfinder with metering

```
#include <FCam/N900.h>
...
  vector<Shot> hdr(2);
  hdr[0].exposure = 40000;
  hdr[1].exposure = 10000;
...
  while(1) {
    sensor.stream(hdr);









  }
```

# HDR Viewfinder with metering

```
#include <FCam/N900.h>
...
  vector<Shot> hdr(2);
  hdr[0].exposure = 40000;
  hdr[1].exposure = 10000;
...
  while(1) {
     sensor.stream(hdr);

     Frame longExp = sensor.getFrame();
     Frame shortExp = sensor.getFrame();









  }
```

# HDR Viewfinder with metering

```
#include <FCam/N900.h>
...
  vector<Shot> hdr(2);
  hdr[0].exposure = 40000;
  hdr[1].exposure = 10000;
...
  while(1) {
     sensor.stream(hdr);

     Frame longExp = sensor.getFrame();
     Frame shortExp = sensor.getFrame();

     hdr[0].exposure = autoExposeLong(longExp.histogram(),
                                      longExp.exposure());
     hdr[1].exposure = autoExposeShort(shortExp.histogram(),
                                       shortExp.exposure());


  }
```

# HDR Viewfinder with metering

```
#include <FCam/N900.h>
...
  vector<Shot> hdr(2);
  hdr[0].exposure = 40000;
  hdr[1].exposure = 10000;
...
  while(1) {
     sensor.stream(hdr);

     Frame longExp = sensor.getFrame();
     Frame shortExp = sensor.getFrame();

     hdr[0].exposure = autoExposeLong(longExp.histogram(),
                                      longExp.exposure());
     hdr[1].exposure = autoExposeShort(shortExp.histogram(),
                                       shortExp.exposure());

     overlayWidget.display( blend(longExp, shortExp) )
  }
```

# Firing a second-curtain sync flash

```
...
Shot flashShot;
flashShot.exposure = 100000;  // 0.1 sec
...
```

# Firing a second-curtain sync flash

```
...
Shot flashShot;
flashShot.exposure = 100000;  // 0.1 sec
...
Flash flash;

Flash::FireAction fire(&flash);
```

# Firing a second-curtain sync flash

```
...
Shot flashShot;
flashShot.exposure = 100000;  // 0.1 sec
...
Flash flash;

Flash::FireAction fire(&flash);

fire.duration = 1000; // 1 ms
fire.brightness = flash.maxBrightness();
fire.time = flashShot.exposure - fire.duration;
```

# Firing a second-curtain sync flash

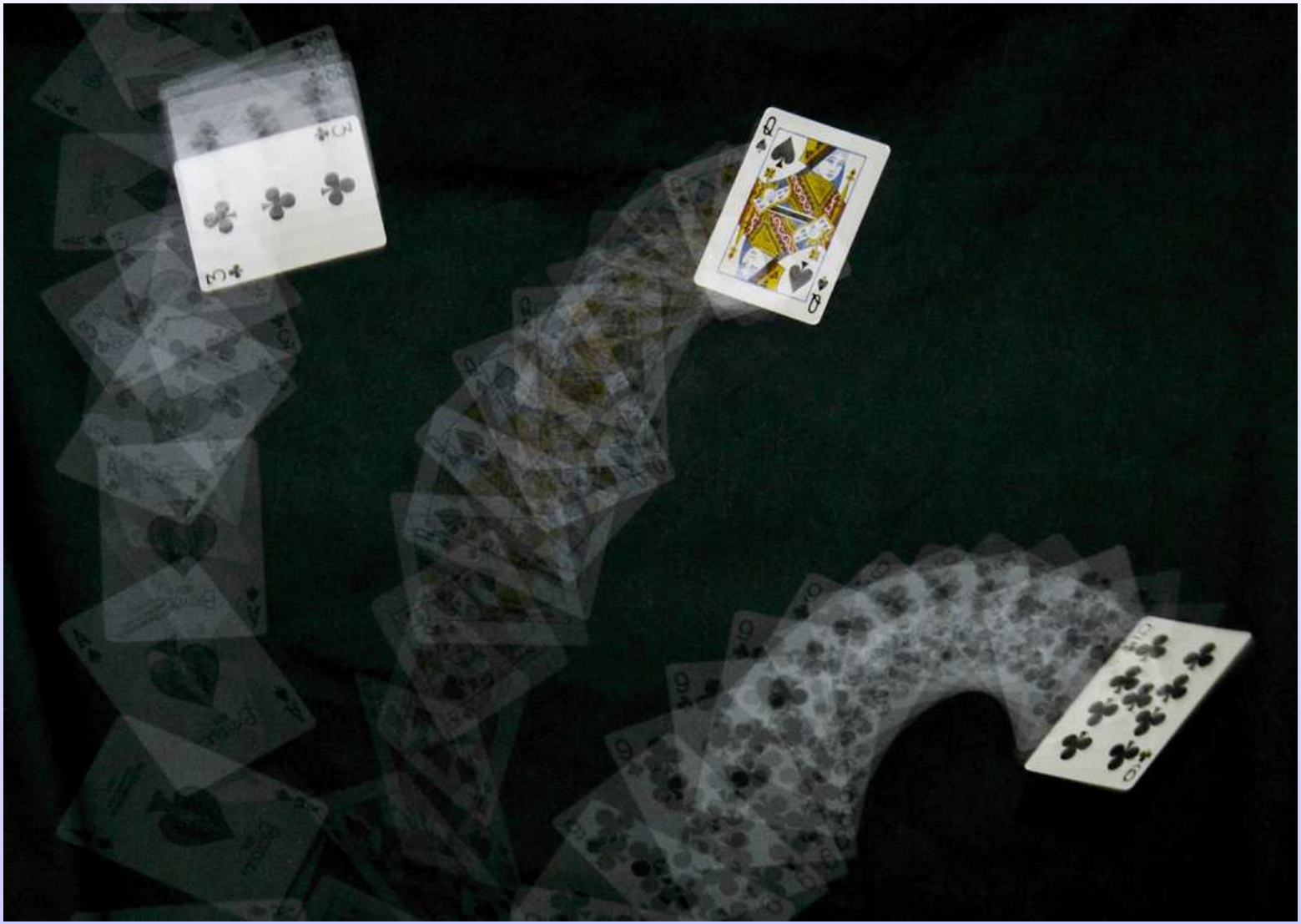
```
...
Shot flashShot;
flashShot.exposure = 100000;  // 0.1 sec
...
Flash flash;

Flash::FireAction fire(&flash);

fire.duration = 1000; // 1 ms
fire.brightness = flash.maxBrightness();
fire.time = flashShot.exposure - fire.duration;

flashShot.addActions(fire);
sensor.capture(flashShot);
Frame flashFrame = sensor.getFrame();
```

# Double-flash example

- Using the F2 Frankencamera and two Canon flash units

# Double-flash example

- Using the F2 Frankencamera and two Canon flash units



# Implementation problems

- Resolution switching is slow
  - Due to underlying ISP driver
  - Roughly 700 ms 'shutter lag'
  - Not fundamental to the architecture, but hard to fix.
- Getting image frames to the GPU seems to have a 300 ms latency
- Hardware bugs also crop up
  - Had to disable vignetting compensation
- F2 Frankencamera issues
  - Small format sensor (1/2.5")
  - Hard to duplicate in large quantities

# The Frankencamera

- A system architecture for programmable cameras
- Two implementations
- An API to program for the architecture (FCam)
- **Example applications**

# Automatic Panorama Capture

capture interface

individual images

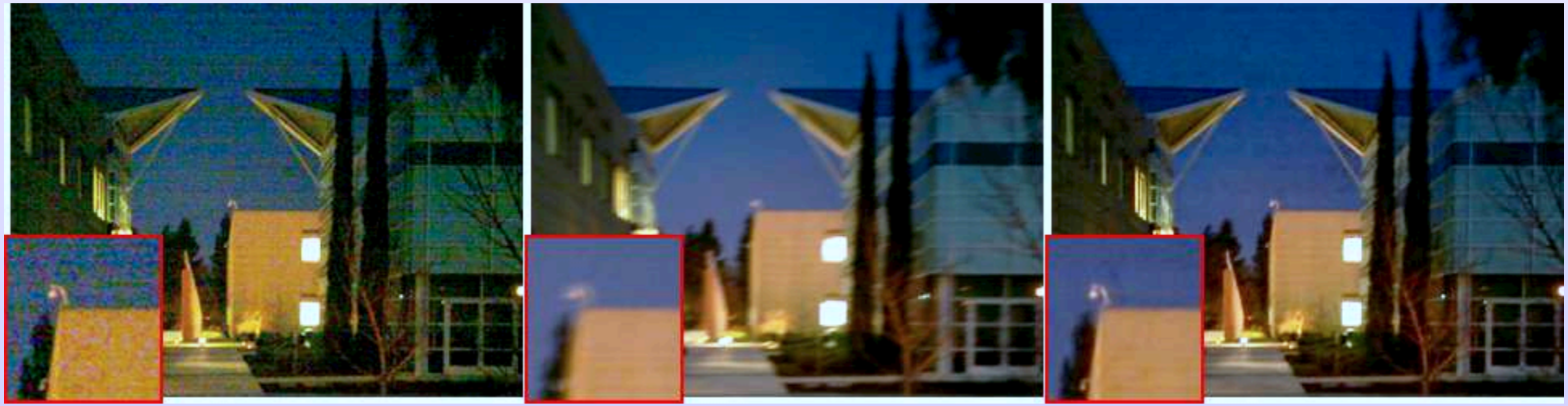
extended dynamic range panorama

# High-resolution HDR Capture

- Created completely on-camera, ~1 minute processing time

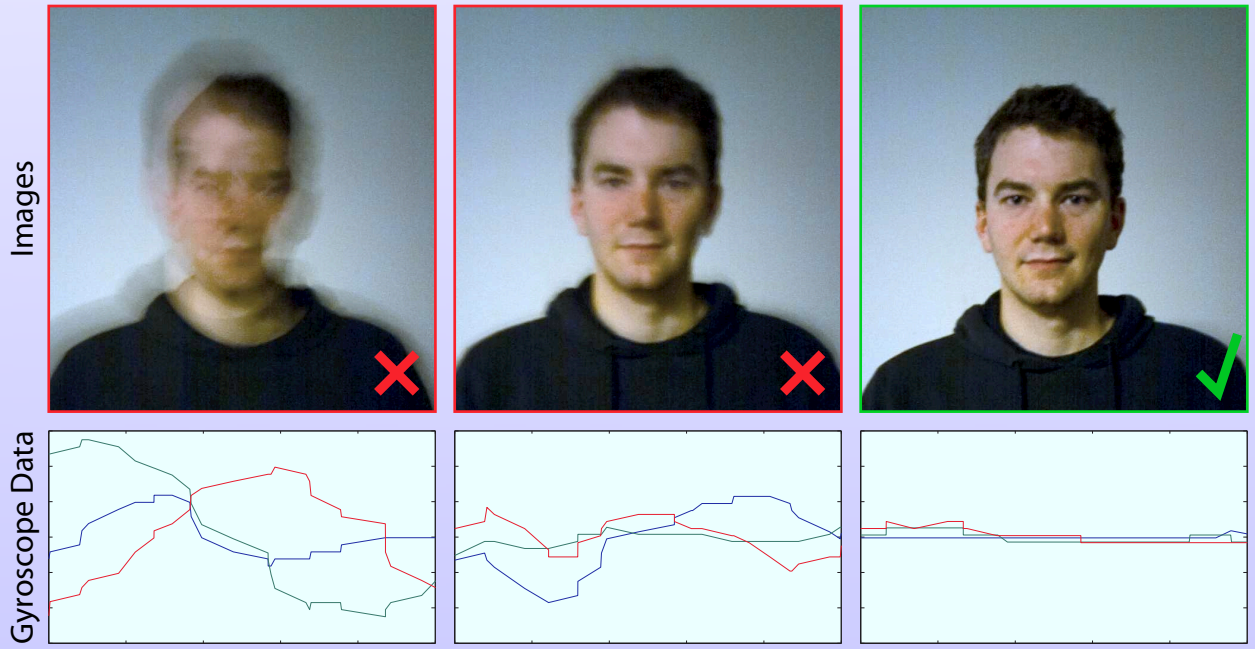
# Low-noise Viewfinder and Capture

- Combines multiple aligned frames in viewfinder mode
- High resolution capture combines two captures:
  - Noisy short exposure
  - Blurry long exposure



# Lucky Imaging

- Attach 3-axis gyroscope to the N900
- Estimate if a captured image suffers from handshake, and keep capturing if it does. Usually done in 10 frames.
- Allows sharp hand-held 1/3 second exposures.

# Applications from CS448

- FCam API just finished before course started
- First assignment: Autofocus - 1 week
  - Robustness was most important, speed a second goal
- Best method: double-sweep
  - Coarse scan through entire focal range
  - Fine scan through sharpest region
- Course projects...

# Remote Flash over Bluetooth

- By Michael Barrientos and David Keeler
- Allows a device action to be sent to some other N900 over Bluetooth, to enable multi-camera coordination.

Figure 4: On-camera flash resulting in a red-eye effect

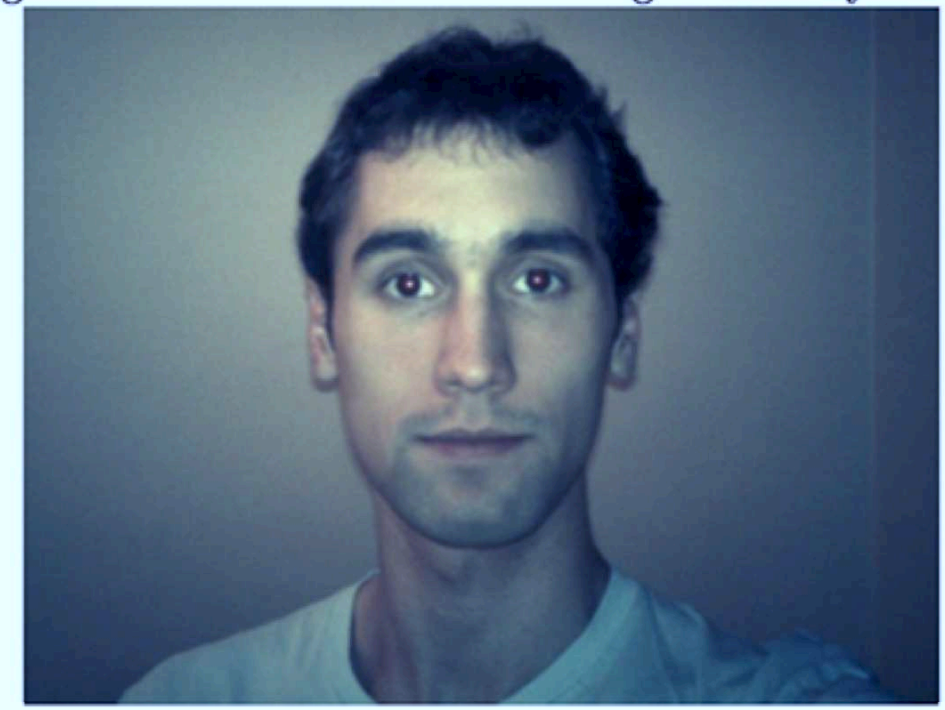
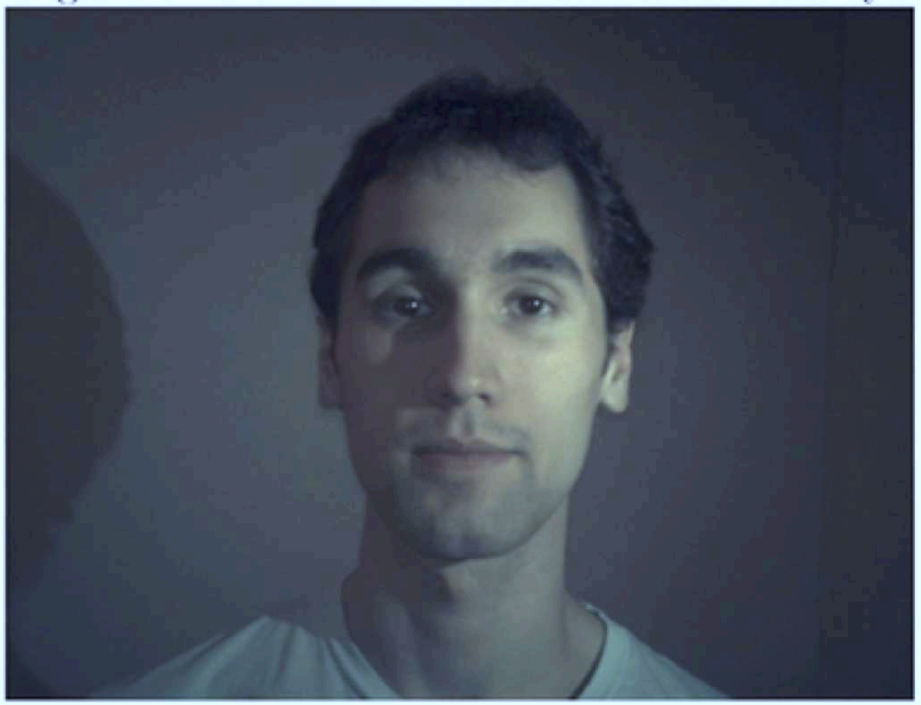


Figure 5: Off-camera flash eliminates red-eye



# Blur-Free Available Light Photography

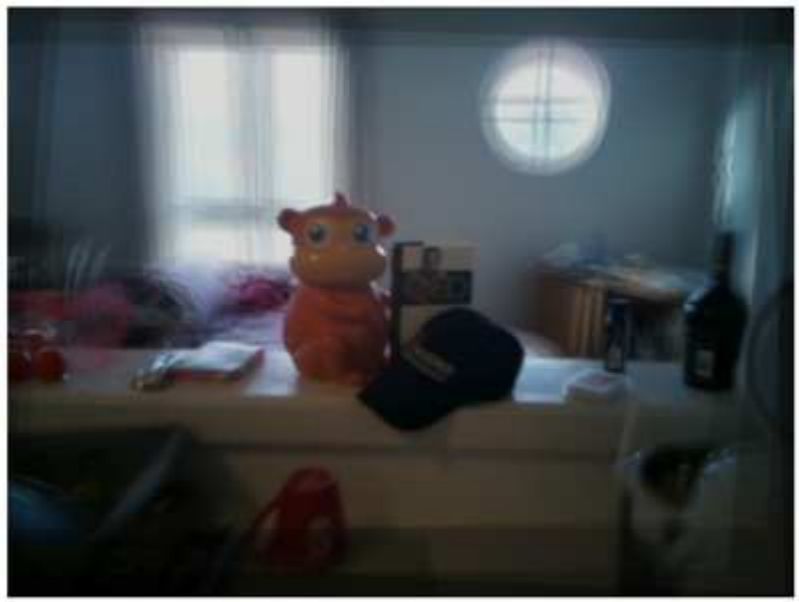
- By Dmitri Makarov and Ben Olson
- Short/long exposure fusion using blind deconvolution.

# Photomontage

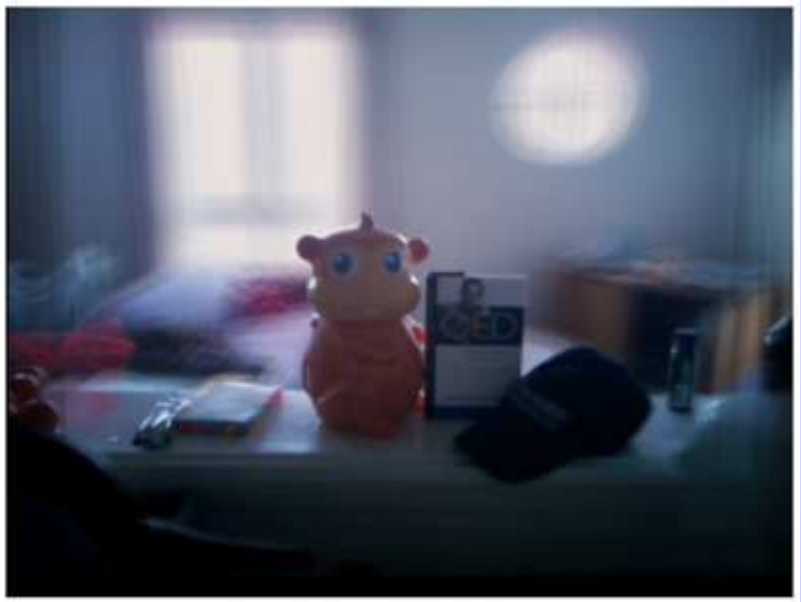
- By Nikhil Gupta and Juan Manuel Tamayo
- Assistant for taking multiple images to be merged later

# Painted Aperture for Portraits

- By Edward Luong
- Combines images from multiple camera positions to nicely blur out the background of a portrait.
- Offline implementation for the merging, using SIFT features and RANSAC. Feature detection is the slow part.



(a) Using 8 images for blur.



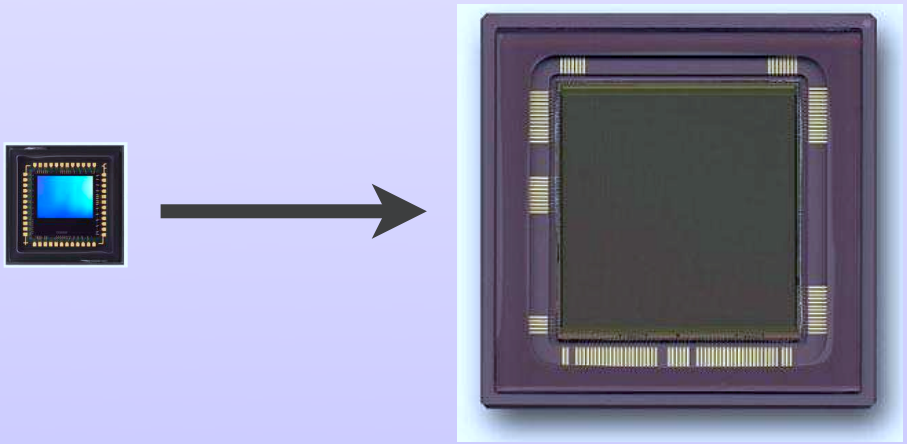
(b) Using 16 images for blur.

# Getting Started with FCam

- http://fcam.garage.maemo.org/
- Includes API, code examples, and FCamera
  - BSD licensed
- From Nokia: HDR, Low-Light
- Feel free to stop us in the halls and ask us to demo these

# Future Work

- Support courses using FCam
  - A bundle of N900s
  - Some courseware
  - An F3
- The F3
  - The F2 uses a cell-phone-quality image sensor
  - We're currently engineering a DSLR-quality replacement



# Conclusion

- Current APIs are bad for computational photography
- Camera platforms should be open