An autonomous sensor platform including a high resolution camera for harsh environments

C. Ferrandez[1], J.-M Friedt[2], K.P. Yew[3], D. Fuin[3], H. Guyennet[3]

[1] FEMTO-ST/LPMO, Besançon
[2] Association Projet Aurore, Besançon
[3] LIFC, Besançon

Slides available at `http://jmfriedt.free.fr`

July 6, 2006

Introduction: objective of this work

- Past experience: uClinux and software picture acquisition
- Last year we presented a purely software solution to low resolution CMOS camera data acquisition [1] $\Rightarrow$ low cost (little additional hardware) but slow, blurred images, requires hard real time (stop all other activities)
- Inappropriate for high resolution camera (data acquisition takes too long, cannot be stored in uClinux memory)
- Provide a portable user interface for inexperienced users (Windows, MacOS X, GNU/Linux)

New requirements for environmental monitoring in harsh environments:

- high resolution ($>$ 3 Mpixels) allowing further image processing
- autonomous for $\simeq$ a year
- data storage or real time transfer (high bandwidths)

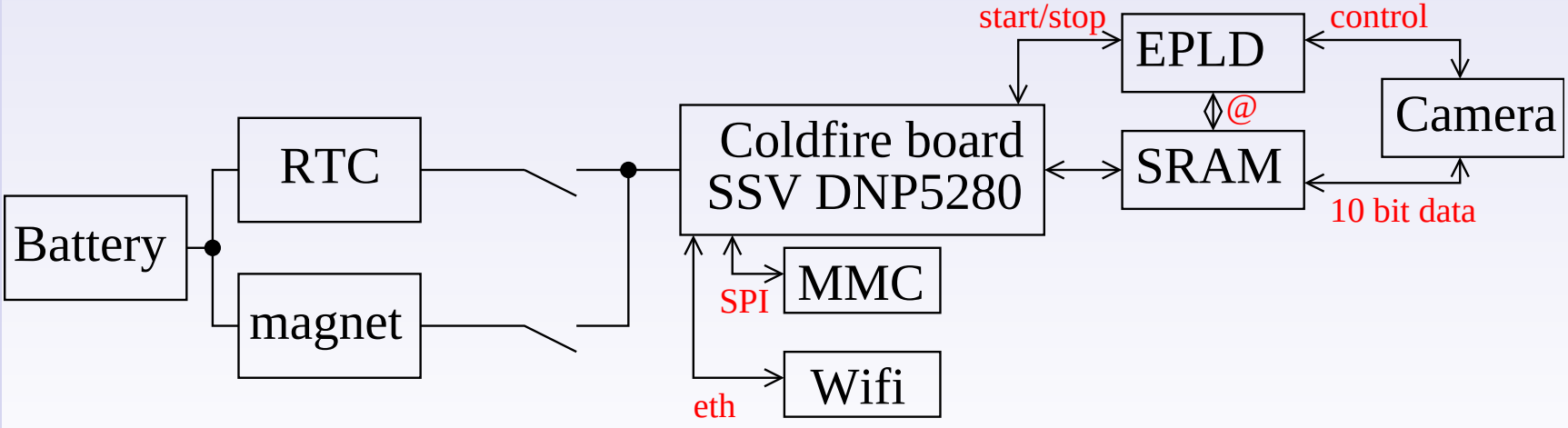
[1] `http://jmfriedt.free.fr/rmll.pdf`

Why use uClinux

A *system* requires many functionalities, including:

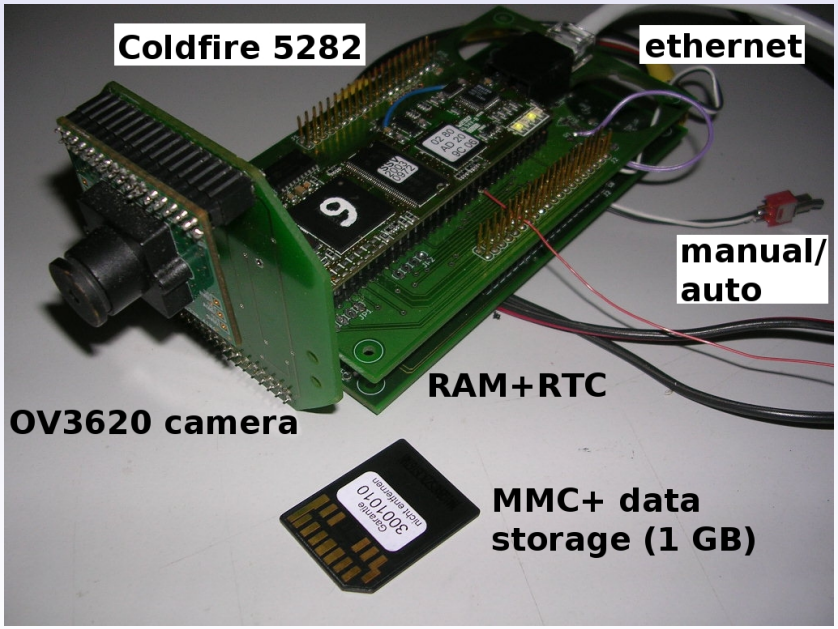
- fast data transfer (ethernet+TCP/IP), NFS during development
- data storage (VFS) and retrieval,
- efficient software development environment (cross gcc, compression tools, C programming with OS calls such as signal management).

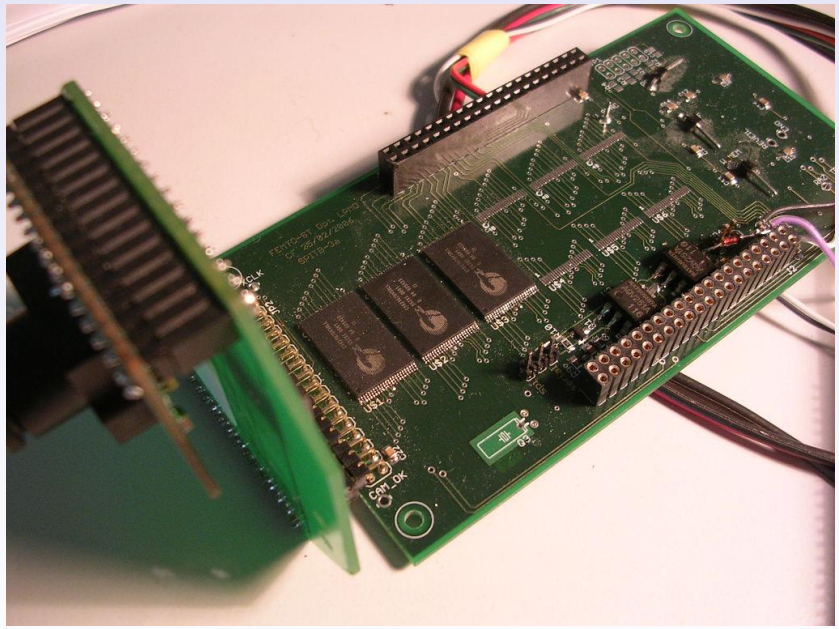
Original requirement of real time data transmission
Most suitable to address the evolutions of the project.



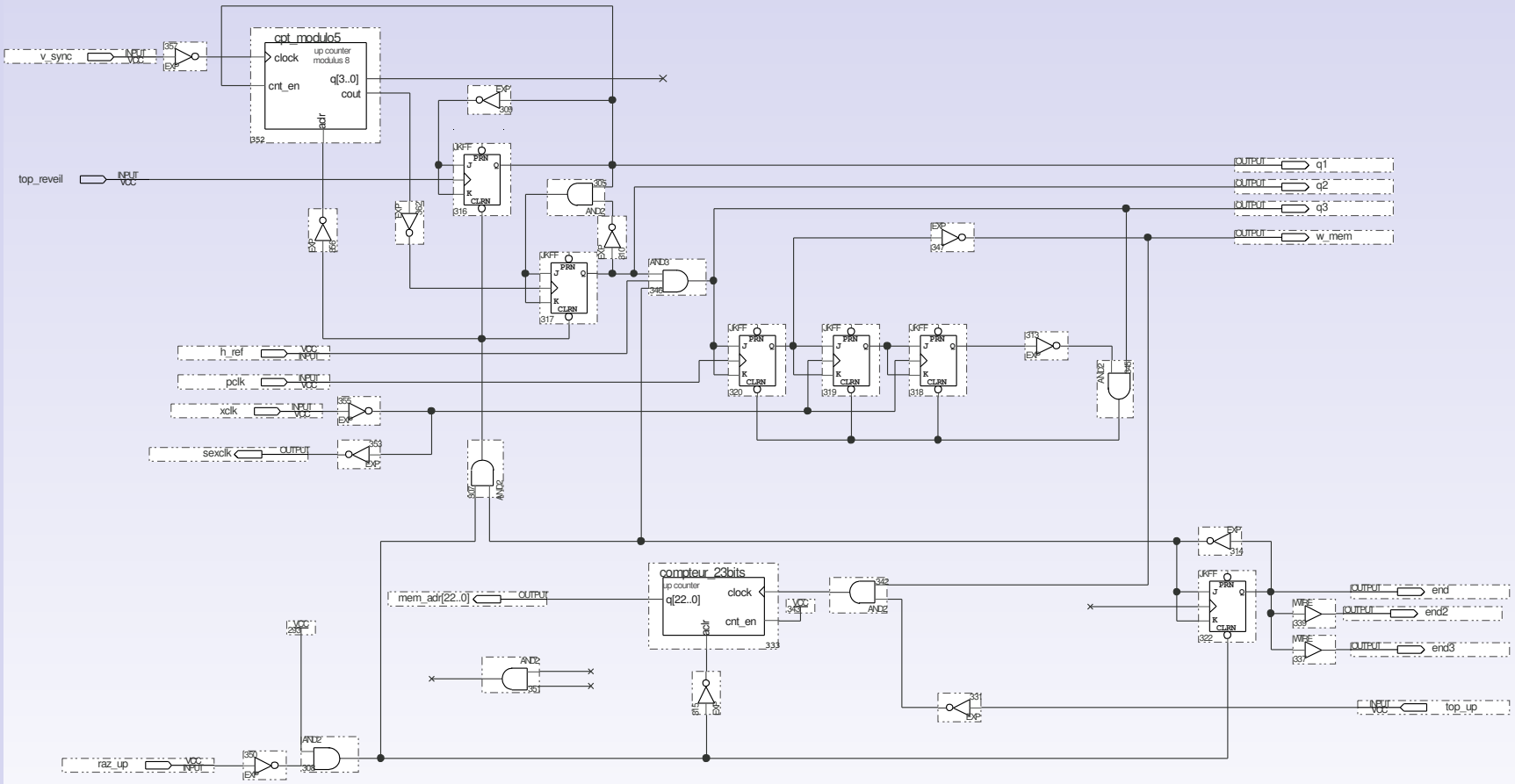
Hardware aspects

- Requires experience with additional hardware (large RAM, EPLD) and power management
  but fast data processing ⇒ reduces the duty cycle.
- Large (3 Mpixel) camera running at 5.55 fps (20 MHz quartz).
- SRAM that large with excessively low power consumption have only recently become available (mobile phone industry): Cypress CY62167DV30.





The solution adopted here: DMA and data retrieval



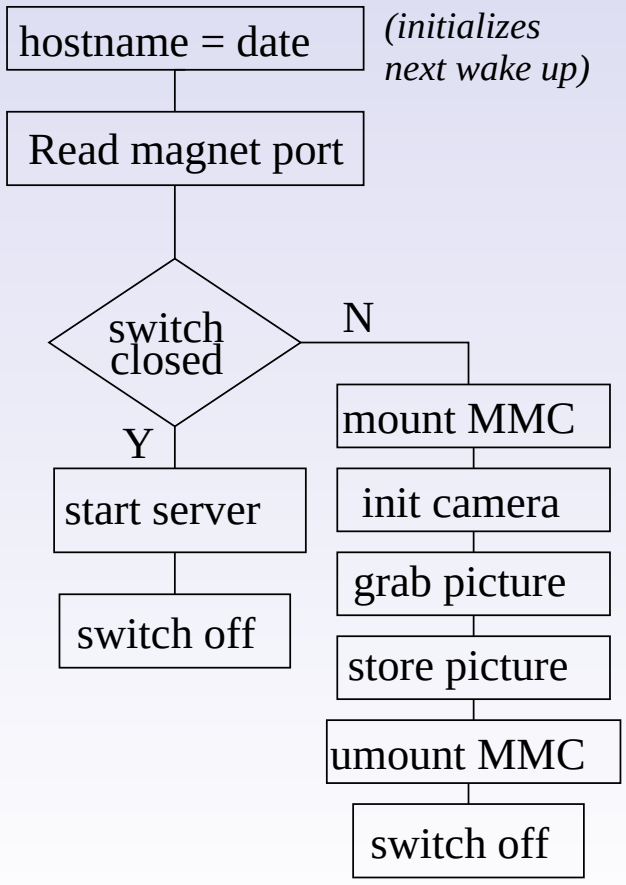
- Extensible to any pixel number assuming RAM is available
- 10 bits/pixel $\Rightarrow$ 16 bit data bus (RAM)

Real time image transfer for selecting the position of the camera

- Upon power up, the processor detects whether it was switched on automatically (see later) or manual: in the latter case, communicate with the user.
- Server running on uClinux, written in C.
- Portable client written in Java: little constraint on the client (most OS).
- Graphical interface for inexperienced users.

Development strategy

1. develop simple programs for testing the ability to access each peripheral (I/O ports, data bus, ADC, RTC ...)
2. once each subprogram is demonstrated, schedule using `/etc/rc`
3. upon condition at power up, select the sequence calling each subprogram



```
#!/bin/sh

IP="`ipdbug ip`"

BC="`ipdbug broadcast`"
NM="`ipdbug netmask`"
NW="`ipdbug network`"
GW="`ipdbug gateway`"

HOME_DIR=/home/
numero="`ipdbug ip | cut -d\. -f4`"

/bin/expand /etc/ramfs.img /dev/ram1
mount -t proc proc /proc
mount -t ext2 /dev/ram1 /var
mkdir /var/tmp
mkdir /var/log
mkdir /var/run
mkdir /var/lock
mkdir /var/empty
ifconfig lo 127.0.0.1
route add -net 127.0.0.0 netmask 255.0.0.0 lo
ifconfig eth0 $IP broadcast $BC netmask $NM up
route add -net $NW netmask $NM eth0
route add -net 192.168.0.0 netmask 255.255.255.0 eth0
portmap &
#mount -t nfs 192.168.0.1:/home/uClinux /usr
cat /etc/motd

insmod /lib/modules/ssvhwa.o
aimant="`/bin/PA3`"

insmod $HOME_DIR/module/i2c_mod.o
cd $HOME_DIR/ov3620
./i2c > /dev/null
$HOME_DIR/ov3620/sleep 5

heure="`$HOME_DIR/ov3620/spi_rtc | cut -d\  -f1`"
h="${numero}_${heure}"
/bin/hostname $h

nom_fichier=`hostname`
$HOME_DIR/ov3620/spi_rtc

# NE PAS ACCEDER A LA RTC QUAND MMC EST MOUNTEE

mknod /var/mmca b 254 0
mkdir /var/mmc

if [ $aimant = 0 ] ;
then
 echo "aimant =0 "
 cd $HOME_DIR/ov3620
 ./server
 ./stop
 ./spi_rtc
else
 echo "aimant !=0 : " $aimant
 insmod $HOME_DIR/mmc_qspi/qspi/module_qspi/qspi_mod.o
 insmod $HOME_DIR/mmc_qspi/mmc/mmc_qspi_mod.o
 $HOME_DIR/ov3620/sleep 1
 insmod $HOME_DIR/mmc_qspi/mmc/mmc_qspi_mod.o
 mount /var/mmca /var/mmc
 $HOME_DIR/ov3620/sleep 4
 cd /var/mmc
 $HOME_DIR/module/i2c_clt
 mv image.pgm $nom_fichier
 # touch $nom_fichier
 echo "picture captured"
 ls -l
 cd /
 umount /var/mmc
 echo "done"
 rmmod mmc_qspi_mod
 rmmod qspi_mod
 rmmod i2c_mod
 $HOME_DIR/ov3620/stop
 $HOME_DIR/ov3620/spi_rtc
fi
```

Module for reading buffer memory

1. software unlocks the VSYNC detection in EPLD
2. fast image transfer from CMOS sensor to RAM
3. a flag indicates to the module that image transfer is completed
4. read from RAM to local memory by 1 KB blocks (data bus single cycle access)
5. each block is processed in user space (store on MMC or send to eth0)
6. repeat until whole buffer RAM is processed

$\Rightarrow$ we never require uClinux to manage more than a few KB

General module structure

```
static int ca88_open(struct inode *inode, struct file *file){
{unsigned long flags;

 image=(unsigned char *)kmalloc(image_size, GFP_KERNEL);

// 1) Phase1 : enregistrement en RAM Tampon
  *((volatile u8 *) GPTAPORT) = 0; //RAZ port BL
  *((volatile u8 *) PORTQA) = cam_ok; //cam_ok=1

  printk (KERN_INFO "OV3620 open\n");

  *((volatile u8 *) GPTAPORT) = raz_up;
  *((volatile u8 *) GPTAPORT) = 0;

  *((volatile u8 *) GPTAPORT) = top_reveil;
  *((volatile u8 *) GPTAPORT) = 0;

  printk (KERN_INFO "END ?\n");

  while ((*((volatile u8 *) PORTASP) & end) == 0)
    { }; // wait for VSYNC

  *((volatile u8 *) PORTQA) = 0; //mise a 1 de cam_ok
  printk (KERN_INFO "Fin d'acquisition\n");

  *((volatile u8 *) GPTAPORT) = raz_up;
  *((volatile u8 *) GPTAPORT) = 0;
  return 0;
}

static ssize_t
ca88_read (struct file *file, char *buffer, size_t count, loff_t * ppos)
{
  unsigned int x = 0;
  unsigned long flags;

  x = 0;
  while (((*((volatile u8 *)PORTASP)&end)==0) && (x<image_siz
    {image[x] =*(volatile u8 *) (cs2 + 6); // lire MSB
     x++;
     image[x] =*(volatile u8 *) (cs2 + 5); // lire LSB
     x++;
     *((volatile u8 *) GPTAPORT) = top_up; // inc @
     *((volatile u8 *) GPTAPORT) = 0;
    };
  if (copy_to_user (buffer, image, image_size))
     return (-EFAULT);
  return (x);
}

static int __init
ca88_init (void)
{
  if (register_chrdev (108, "ppp", &ca88_fops))
    {printk (KERN_ERR "OV3620: unable to get major %d\n",108)
  printk (KERN_INFO "OV3620 module installed.\n");

  *((volatile u8 *) GPTADDR) = 0xff; // Port B[0-3] output
  *((volatile u8 *) DDRAS) = 0x00; // PORT B[4-7] input
  *((volatile u8 *) DDRQA) = 0x0B; // Port A[0-3] output
  *((volatile u8 *) DDRQB) = 0x00; // Port A[4-7] input
  return (0);
}
```

Real Time Clock programming

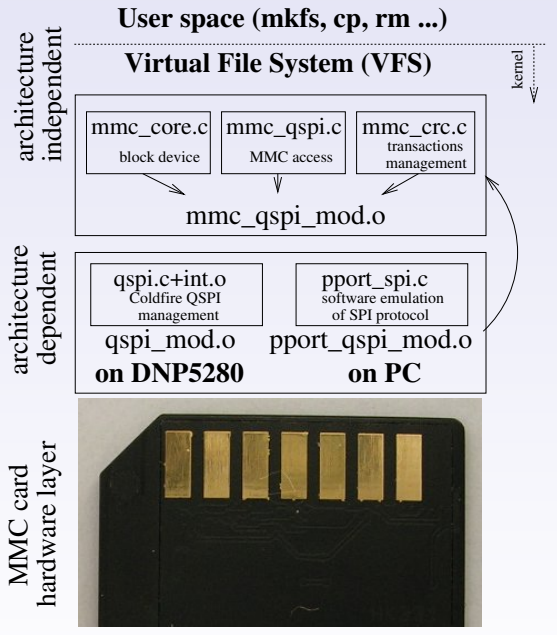
Autonomous mode: the camera wakes up at given intervals every day, captures an image, stores it and goes back to sleep. Good stability.

1. requires SPI port access (shared with MMC)
2. RTC provides some RAM – always powered – which is used for storing time interval between two shots
3. at wakeup, read current date/time, set hostname accordingly
4. reset alarm for next wakeup
5. only component running while all other parts of the system are asleep

$\Rightarrow$ fundamental aspect of energy saving

Data storage: using a MultiMediaCard

- Requirement: store over 6 MB per picture in non volatile memory for more than 200 pictures (as allowed by the battery capacity, see later)
- MMC was selected because it is easy to program (SPI port), well documented and widely available at a reasonable price (digital camera industry).
- However, low bandwidth due to the serial communication mode (parallel communication mode of MMC+ requires full MMC functionalities including CRC which was only recently achieved).



- We wanted a filesystem to be able to attempt a recovery from system crash during autonomous data acquisition, and for managing multiple files at the same time (ADC conversion + pictures).
- Driver [2] linking a low level access to the SPI port of the Coldfire processor and proving raw access to any 512 byte sector of the MMC to the high level VFS provided with the Linux kernel
- provides ext2 and vfat filesystems but no partition management under uClinux (the same driver works under Linux with partition management).
- uClinux: irreversible corruption problems of ext2 after several images were recorded (up to 25 images, $\simeq$ 160 MB) so vfat was selected (much more basic). Could be due to defective communication between Coldfire and MMC.

[2] S. Guinot, J.-M Friedt
Stockage de masse non-volatile : un block device pour MultiMediaCard
GNU/Linux Magazine France, Hors Série 25 (Avril 2006) [in French].
Archive available at `http://jmfriedt.free.fr`

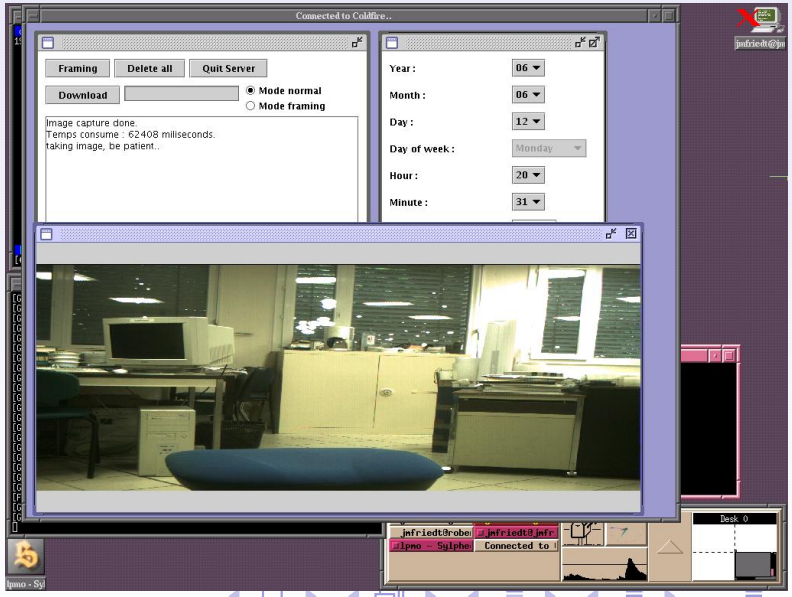
Initialization of the camera

- Issue concerning the proprietary register mapping of the camera, lack or erroneous documentation.
- I^2C communication between the CPU and the camera: set the resolution, basic setup and some undocumented functionalities.
- While the default values for the 100 kpixel camera gave satisfactory results, the high resolution camera will *not* work unless properly initialized $\Rightarrow$ reverse engineering of demo board.

Java mistakes for C programmers

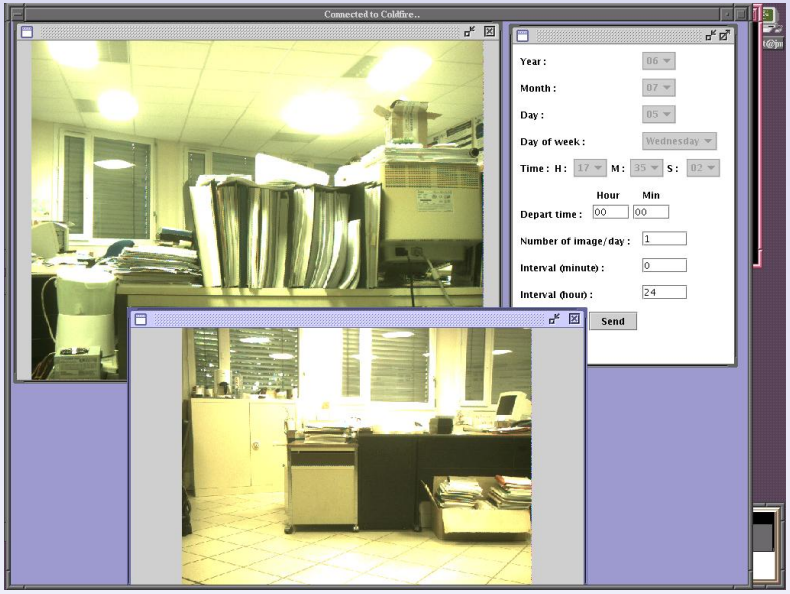
Java was selected as the portable language requiring the least effort from the programmer side (GTK, Qt, Perl/Tk).

- Avoid mistakes such as using a char as a byte (a char is *not* necessarily a byte depending on the LOCALE.
- Avoid strings for handling byte arrays.
- Generally, take care in handling data structures as defined in different languages
- Heap size (large image display and manipulation)



The Java client

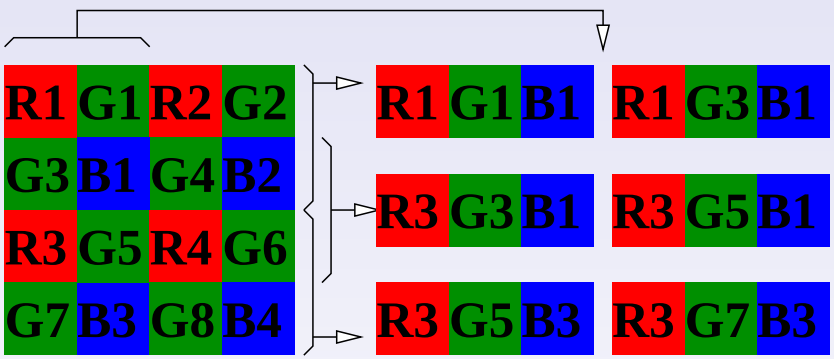
Functionalities: transfer low and normal resolution images in real time + program real time clock (RTC)



- Once the setup is complete, switch off most boards and only keep the RTC running.
- After a preset delay, the RTC triggers a signal that wakes up the Coldfire for automated data acquisition.

Bayer to RGB image conversion

- The raw data obtained from the CMOS sensor provides an oversampled green component and 1.5 Mpixels of R and B (3 Mpixels is the *total* number of pixels whether red, green *or* blue).
- Problem: how to combine these pixels in a meaningful RGB image ? How to avoid distortions associated with the combination of 4 pixels in Bayer mode ($R, G_1, G_2, B \rightarrow \{RG_1B\}, \{RG_2B\}$) ?

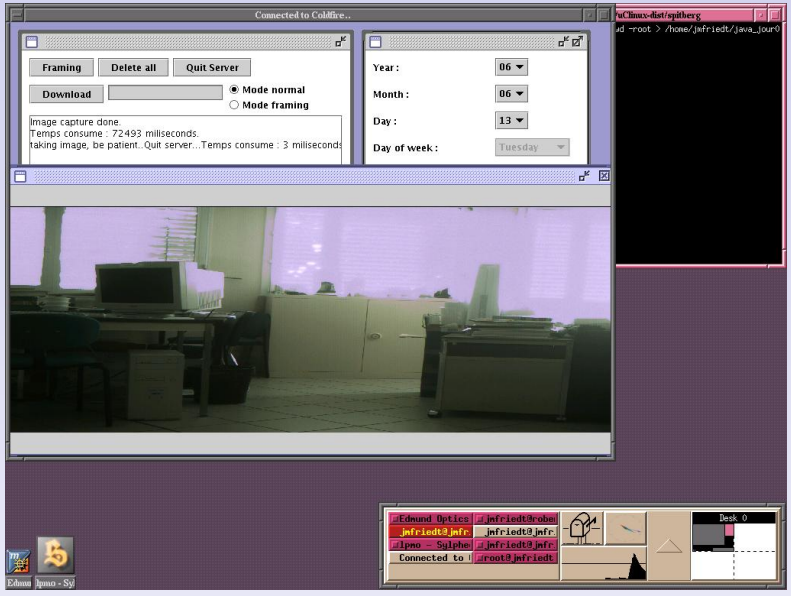


- More complex algorithms are available but beyond the scope of this presentations.
- Raw bayer data useful for quantitative data analysis.

Example: Bayer $\longrightarrow$ RGB

$N \times N$ lines in Bayer data $\longrightarrow N-1 \times N-1$

Effect of infrared wavelengths on CCD & CMOS sensors



- Well known effect: no longer an electronics but an optics problem, get the right filters (cf webcams)
- Strong improvement by replacing a single lens with a lens including an aperture
- A *system* includes hardware, software *and related components* (here tight enclosure and optics)

Conclusion

- Consumer electronics is a set of amazingly complex instruments, ranging from high bandwidth wireless transmission (GPRS, Wifi), high quality imaging sensor, mass storage and fast data transfer: we here realize how difficult it is to reach such a quality.
- Development of this project seen as a basic platform for future research on multisensor array (10 sensors have been built in this project). Additional sensors are to be added (temperature, gas sensing, dust density etc ...). Possibly mobile applications.
- We have included components with huge computing power (but high electrical power consumption) which provide opportunity to implement complex tasks.
- Initially client was supposed to retrieve data, but MMC access is so slow that downloading 1 GB would take too long $\Rightarrow$ replace MMC when visiting the location of the sensor platform.

Perspectives

- MMC bandwidth is the bottleneck during data storage/retrieval: either speed up access (parallel instead of synchronous serial protocol) or compress stored data (lossless)
- The image quality is fine in an indoor environment but usually saturated in direct sunlight: optics need to be improved.
- Add communication between multiple systems.
- Add sensors for environment monitoring.

Acknowledgement:
funding has been partly provided by M. Griselin (Laboratoire THEMA, CNRS & Université de Franche-Comté)