

**Description
of the second
Allais-Esclançon light deviations
measurement system.**

Vincent MORIN

With the kind support of

Thomas GOODEY

Copyright August 2004

Introduction

In a previous memoir in french, a first setup has been described which was made of off the shelf components. The purpose was to take a first look at the phenomenon of periodic light deviations pointed at by Maurice Allais, and earlier by Ernest Esclangon. Some indications that something unusual was happening let me push further the exploration with another setup. The main defects of the first setup were its thermal sensitivity and the asymmetrical shape of the light spot.

With a 3 K euros financial support from Mr. Thomas Goodey, it has been possible to build a much better setup. Thermal sensitivity has been reduced by a factor of 10 and more, and the laser light spots are well shaped. Also some improvements in the software monitoring system now allow a fully automated operation with automatic mains failure recovery. The spot image processing has also been improved to suppress the effects of limited sensor area (apodization of spot tails).

The experimental setup

The whole setup is build on an extruded polystyrene foam structure. This material brings both its rigidity as a supporting frame and its thermal isolation properties to help stabilize the laser beams attachments. The following photo shows it installed in a cellar where daily thermal variations are about 0.5 °C :



The setup is monitored by two PC computers (below the foam structure), each of which monitors 3 cameras, 3 thermal sensors and supplies the laser diodes. PC are networked and supplied by a UPS (on the foam structure pi shaped footing).

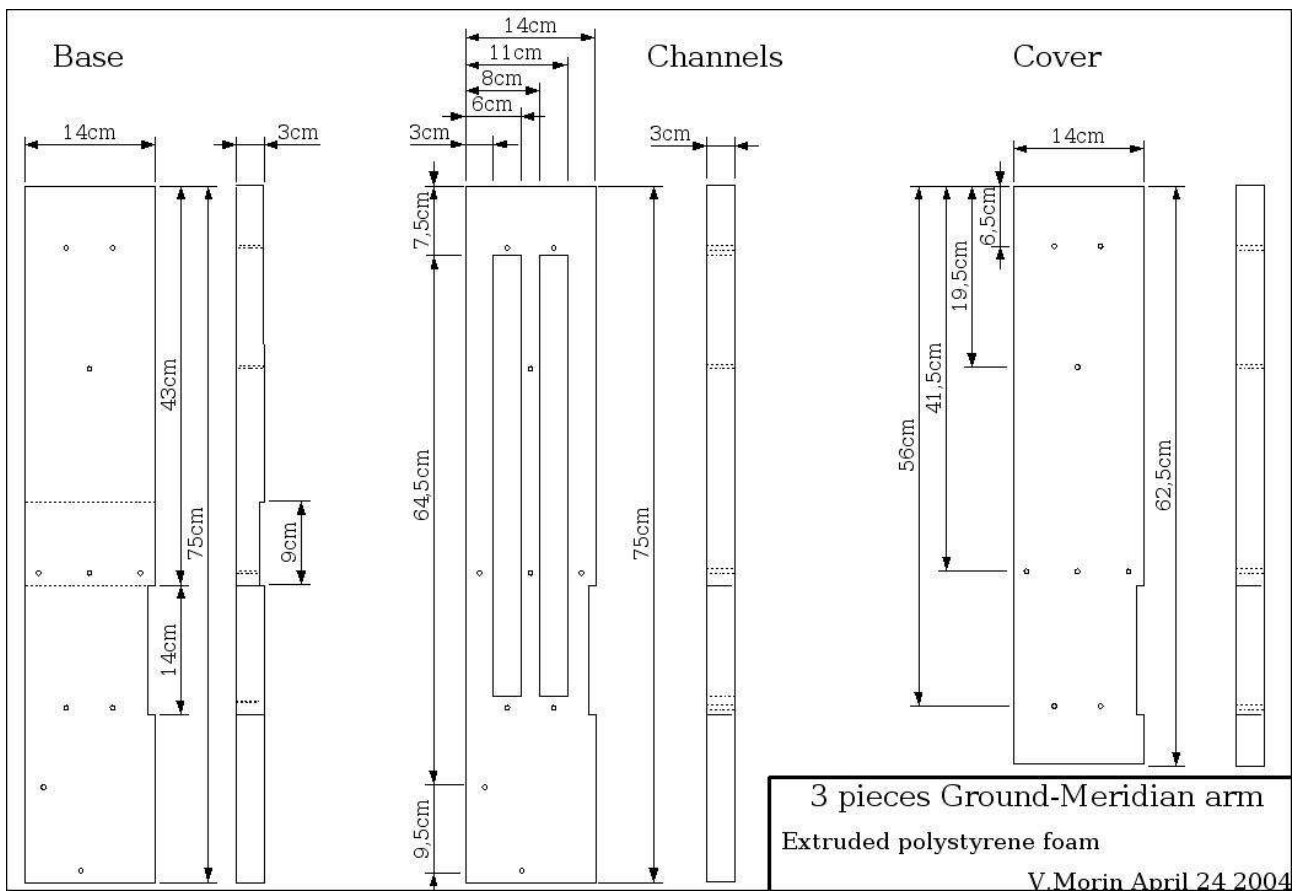
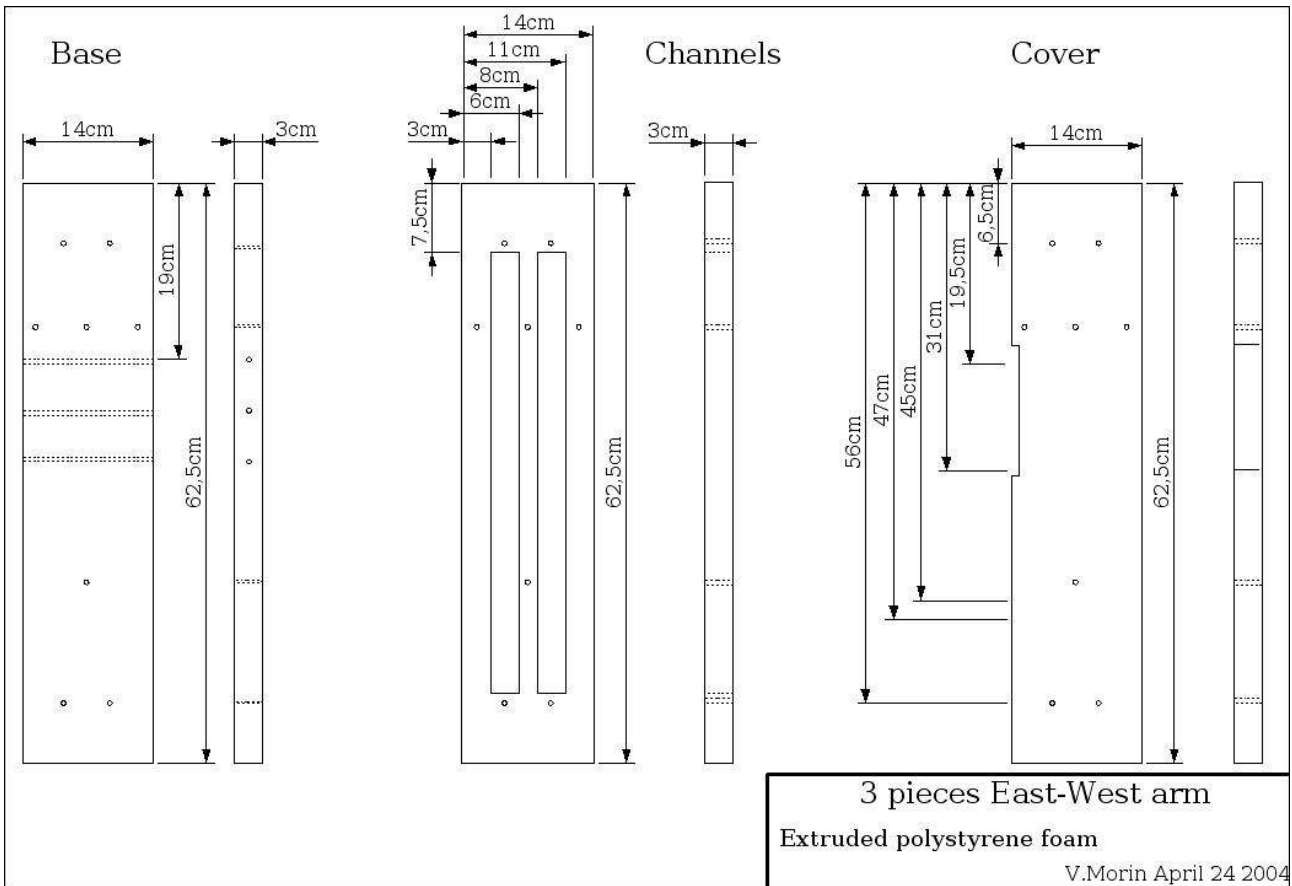
Foam structure

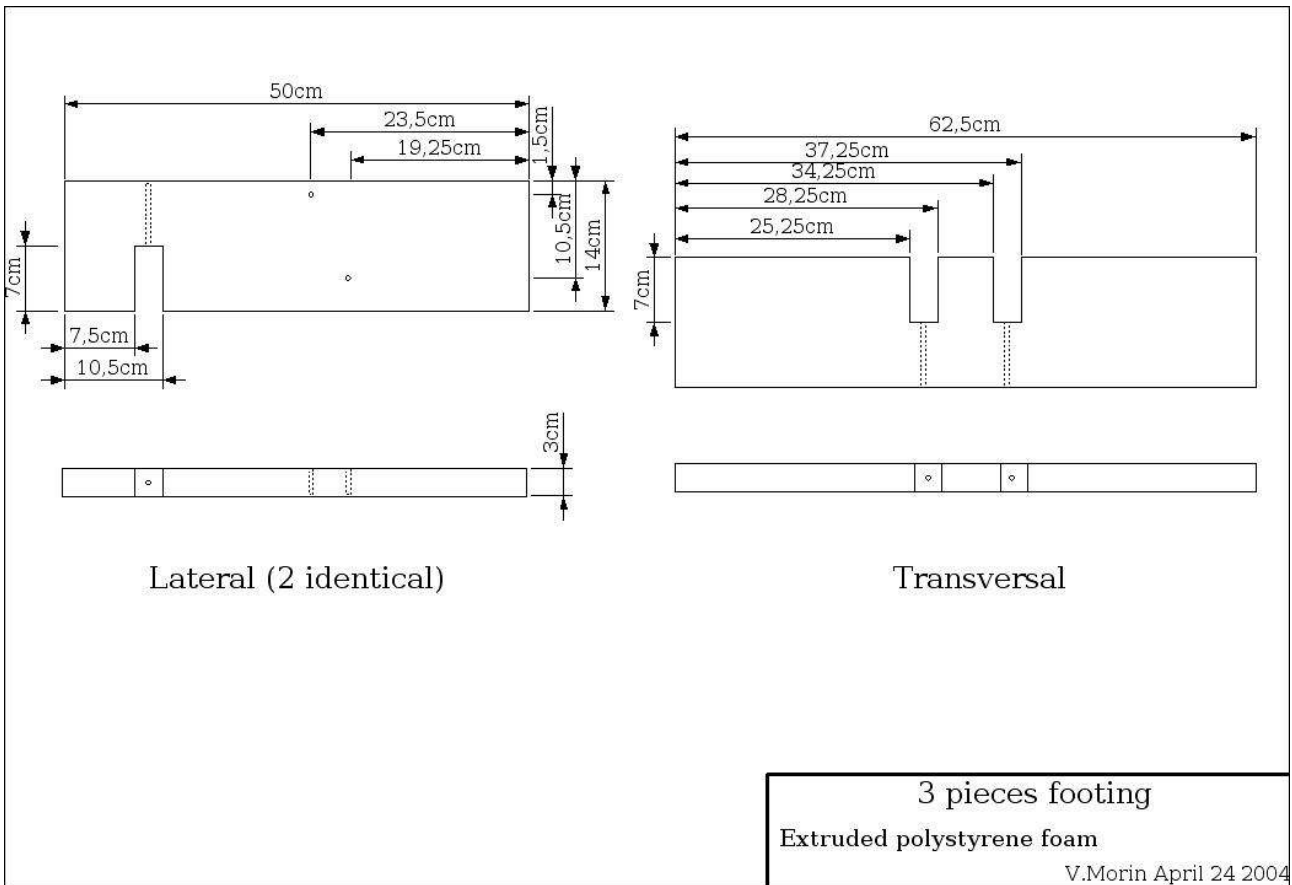
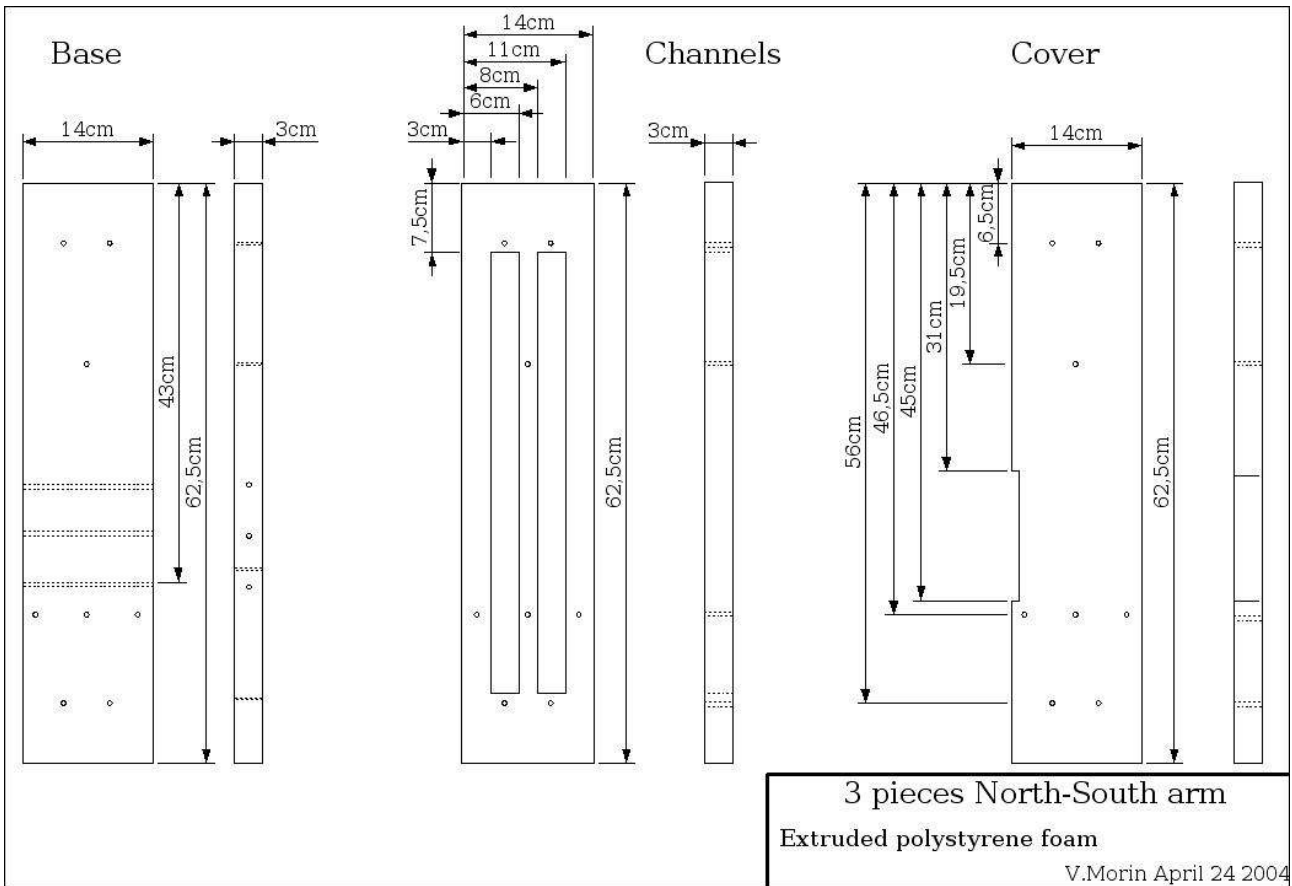
The foam structure is made with 60 mm thick extruded polystyrene cut in 2500 by 600mm raw panels. This is the minimal thickness and thicker would be better (80 being reasonable). The reason why 60 has been used is a consequence of the raw material poor surface quality which required surfacing, and available tools at the workshop for machining the panels. Such foam pieces are not extremely rigid but it is sufficient provided the laser beam mounts are very rigid and insensitive to flexion (as is the case for us and will be seen later).

Plans of the foam pieces are on the following pages. A photo of the stacked pieces with threaded rods and nuts follows :

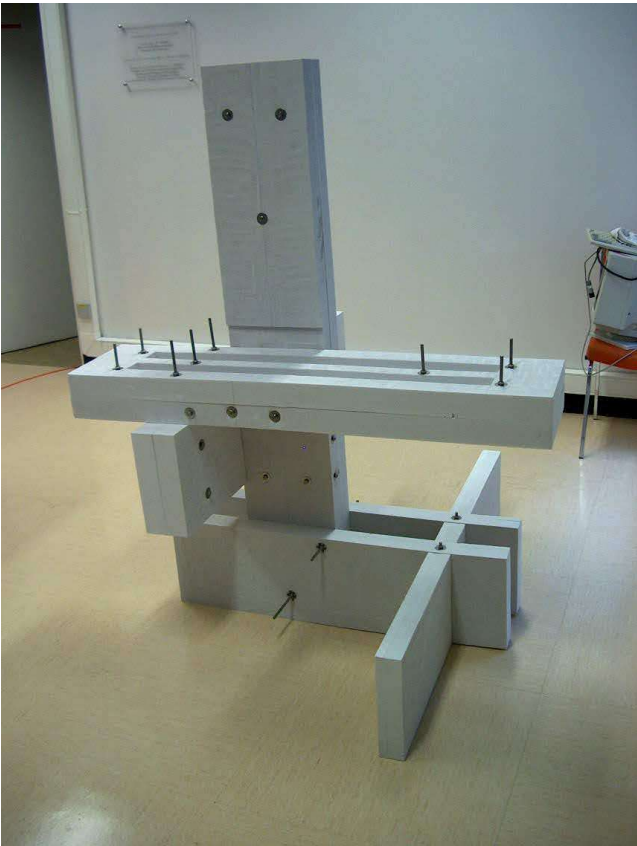
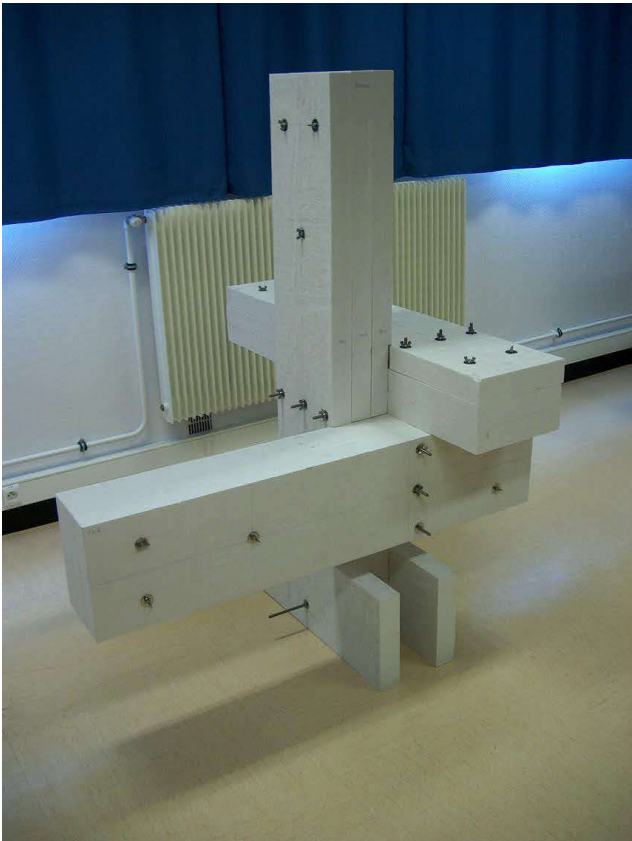
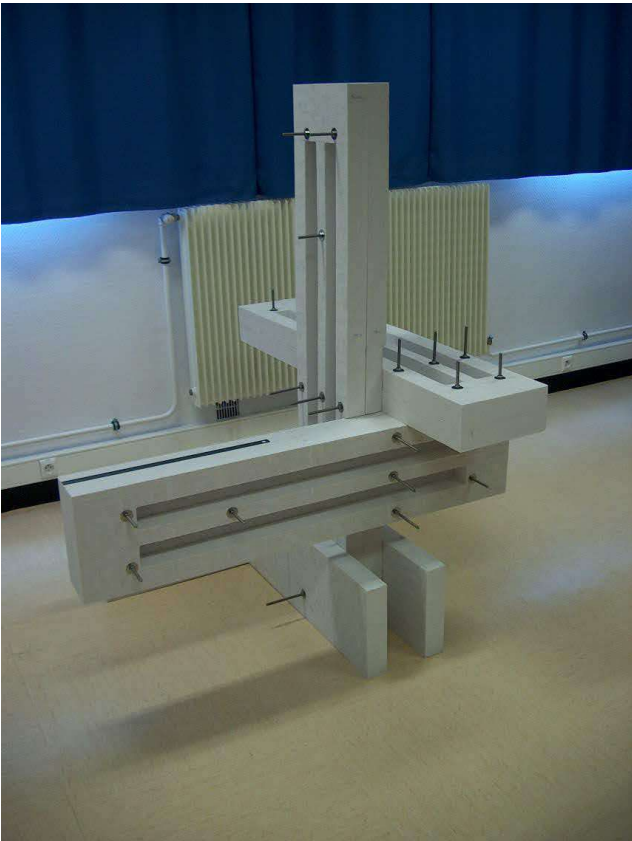


Foam parts are assembled with 9m of 8mm stainless steel threaded rods, washers, nuts and wingnuts. A plastic tubing 12mm OD 10mm ID is forced in the foam through assembly holes to avoid damaging foam lids when moving them and to facilitate mounting.





The assembly seen on under various angles with or without lids looks like :



Laser light beams

The laser light beams are mounted to be very rigid and immune to thermal variations. Rigidity must insure that when pushing beam mounts in the foam channels they so not get disaligned, also when lids are moved, light beams must not be affected.

I chose to mount the laser beam inside a borosilicate tube 50mm OD 2.8 mm thick 900mm long. Two robax glas ceramic strips (30mm 900mm 4mm thick) are put inside the borosilicate tube to form a V channel. The V channel is put on a 10mm long 2mm dia. borosilicate glass roller so that only the ends of the strips are in contact with the tube (and a third point on the 50mm outer tube. Five of those assemblies are shown below. Hose collars are used to secure electrical connections (USB cable for the camera, laser supply connector), to help handling the beam tubes and they penetrate in the foam channels walls so that a vertical positioning is possible.



And the detail of the arrangement at one end is as shown here :



This arrangement proved very rigid. The glass roller length must not exceed 10mm otherwise

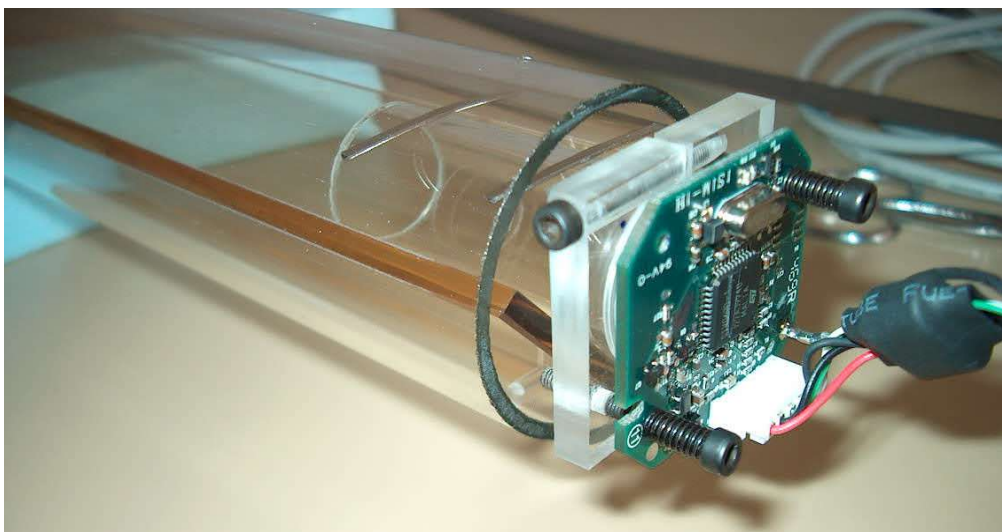
breakage frequently occurs when spring pressure is applied at camera or laser mounting.

Camera mounting.

The HDCS 1000 camera chip is mounted on a PCB extracted from a Logitech Quickcam Express webcam sphere. The optics is removed so that the CMOS chip window is visible.

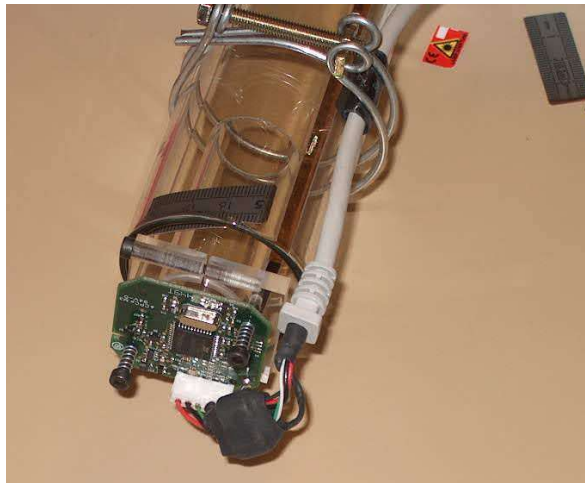
To reduce dilatation effects, the CMOS chip package is pressed on the flamed end of a 20mm ID 22mm OD 50mm long borosilicate tube. Pressure is applied by compression springs around screws driven in a plexiglass adaptator. The CMOS sensor cannot slip sideways, which is the main requirement. The borosilicate glass tube is pressed by an inox ribbon springs on the Robax V channel.

The arrangement is shown below :

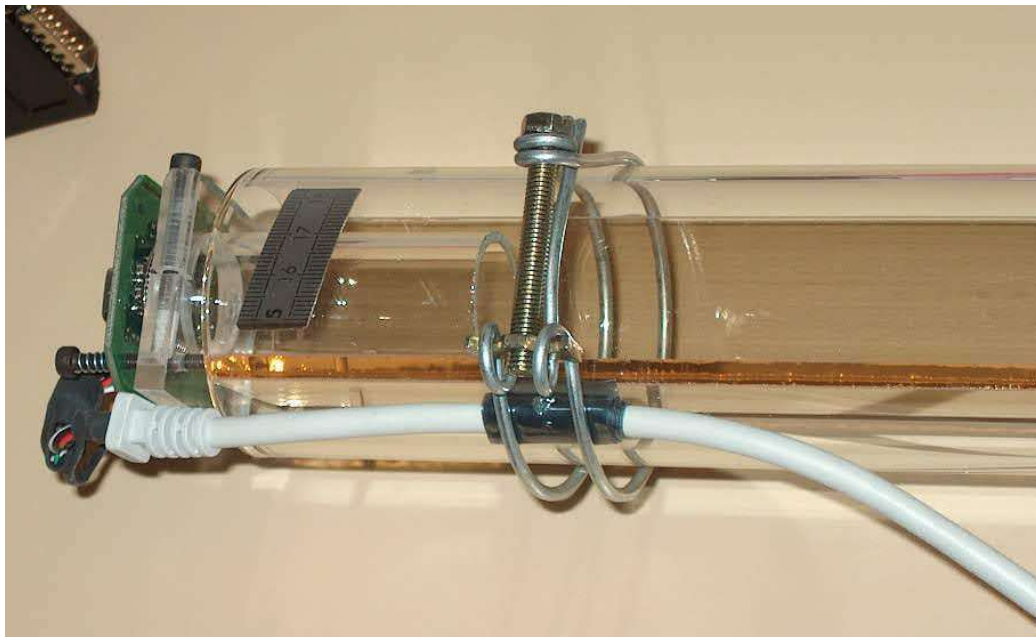


Springs used above for testing are replaced by an inox 10mm wide rule section for definitive assembly.

Test springs are replaced by an inox 10mm wide rule section for definitive assembly.
The USB cam cable is caught under the hose collar.
These are shown below :



The inox spring presses with some force. Using a ruler is practical for cutting the right length.



Laser mounting.

On the other end of the 50mm tube, a conical invar (Phynicx Dilaton 36) piece is placed which maintains the laser and allows adjustment by two 12 by 2.5 mm screws.

Photos of the invar piece follow :



The piece is lathed in a 25mm dia. invar bar. The cone angle is 2° . Front and rear cone have 45° angles.

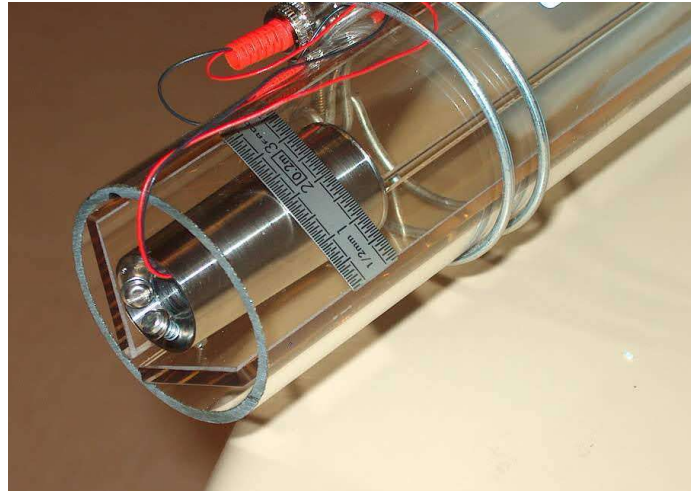
The laser diodes used are Imatronic LDM-115P/673/1 (1mW output power, circular beam from plastic lens, adjustable focus so that a parallel beam can be obtained. Supplied in 5V, 25mA.



Diodes are inserted at -20°C with a light forcing, so that they are blocked at room temperature. After diode insertion, angle adjust screws are driven in the rear hole at 45° angle on the Robax.

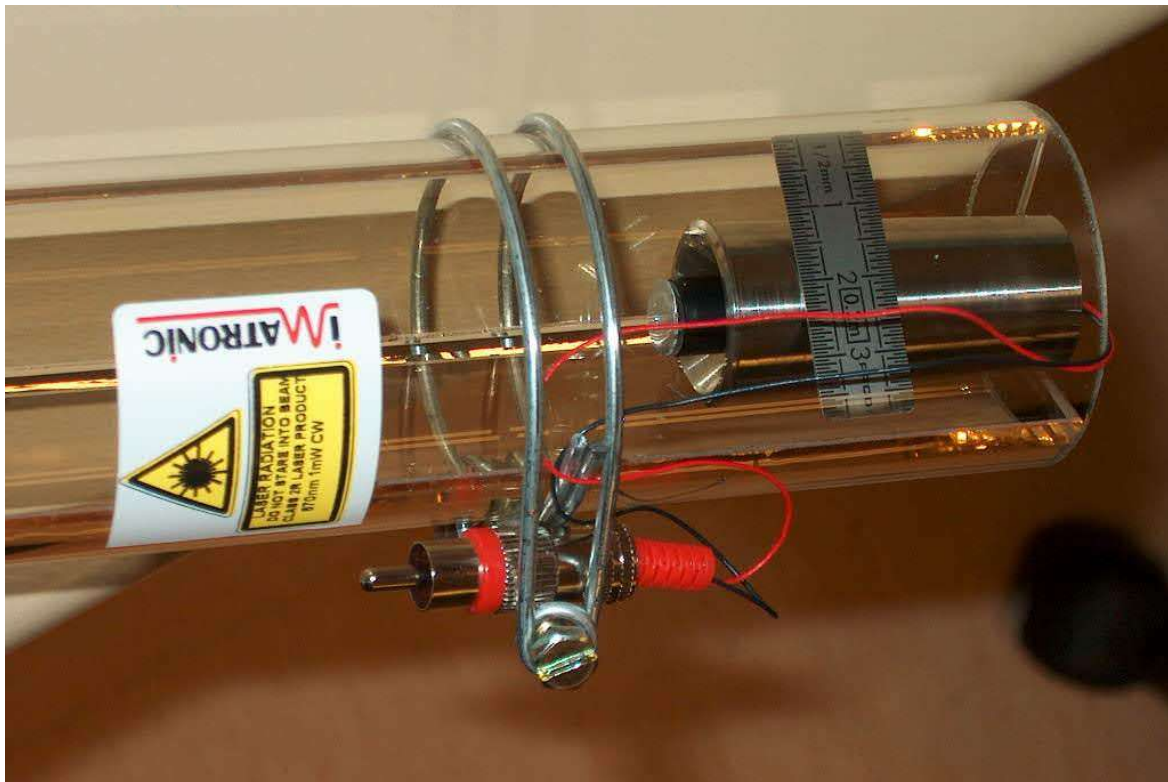
Several view of the laser mounting follow :

The inox spring presses the cone and works against the adjustment screws.



The laser is inserted, adjusted to provide a parallel beam and a 1mm pinhole drilled in an aluminum disc is stuck on the front. As pieces are small; one expects dilatation is negligible. This part is probably not entirely satisfactory.

The hose collar maintains an RCA plug for the laser supply. One is insured cables cannot pull the cone and adjustments are kept intact when transporting the whole beam.



The inox spring pressure is sufficiently strong to allow a vertical positioning of the beam without movement of the cone.

Beam insertion.

The glass tubes of the beams are inserted in the foam channels, collars are pushing the foam walls so that the tube does not move, even in vertical position.

For example the east to west beam inserted is shown below :



Supply, thermosensor and USB cables are pushed in knife made grooves in the foam, so that air or light does not pass through.

Temperature sensors.

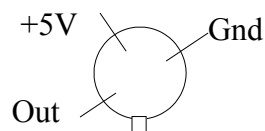
In each foam channel is inserted a Smartec SMT 160-30 TO18 thermo sensor. The sensor is pinched in a brass tube flattened on a 12mm OD copper tube water reserve which serves as thermal equipotential in the channel.

The SMT 160-30 is a 3 pins device which outputs a rectangular waveform with period 1 KHz and variable cycle ratio. It suffices to measure precisely in arbitrary time units the consecutive high time and low time to compute the temperature. 3 SMTs are thus connected on button inputs of the motherboard game port, so that a real time measurement by the number of processor cycles in each part of the SMT output cycle is possible by monitoring the 0x201 port at full speed with interrupts disabled.

On the game port which opens on a DB15 female connector, used pins are :

pin 1 : 5V supply	pin 7 : button 1 input
pin 2 : button 0 input	pin 10 : button 2 input
pin 4 : buttons 0 and 2 ground	pin 14 : button 3 input
pin 5 : buttons 1 and 3 ground	

The SMT pinout (bottom view) is :



I denote SMT(0) the first sensor, SMT(2) and SMT(3) the others. Button port 1 is reserved for detection (connected to supply to detect the plug).

All SMTs have their supply on pin 1. SMT(0) and SMT(2) have their ground on pin 4. SMT(3) has its ground on pin 5. SMT(0) outputs on pin 2, SMT(2) on pin 10, SMT(3) on pin 14. Software handling is shown in Ada listing.

Automated monitoring.

The setup uses 2 identical PC computers : K7VM2 Asrock motherboard with integrated video and gameport. XP2200+ processors are used, memory is 128Mo PC2700 DDR. A two USB1 extension card is added to provide a third USB controller (only one quickcam by controller is allowed for the driver to operate).

Operating system is Linux Mandrake 9.1. System clock is network synchronized with ntpd. The BIOS is configured on restart after power failure. A Linux autologin in KDE 3.1 is used to launch the camera config and acquisition software. On a short failure, only one data point is lost.

The quickcam driver is recompiled from CVS version of the qc-usb from Tuukaa Toivonen, with qc_set executable for camera parameters positioning. Qc_set is called in the starting shell script.

The thermal sensor deliver a square signal with cyclic ratio depending on the temperature. Three of them have their output on 3 button inputs of the gameport whose register is at 0x200. The gameport also supplies the sensors. A real time scanning of the gameport allows to determine the number of processor cycles in low and high states (interrupts must be disabled for the 1 msec measurement time).

The gnat 3.15 Ada 95 compiler is used with the GPS IDE to produce the “devlum” acquisition software.

The experiment operates on the basis of a 10 minutes cycle. During 9 minutes, more than 4000 frames are collected on the 3 cameras, the same number of temperature measurements is done on the 3 corresponding sensors. On the last minute a mean picture of each spot and the mean of the temperature measurements on each sensor is computed.

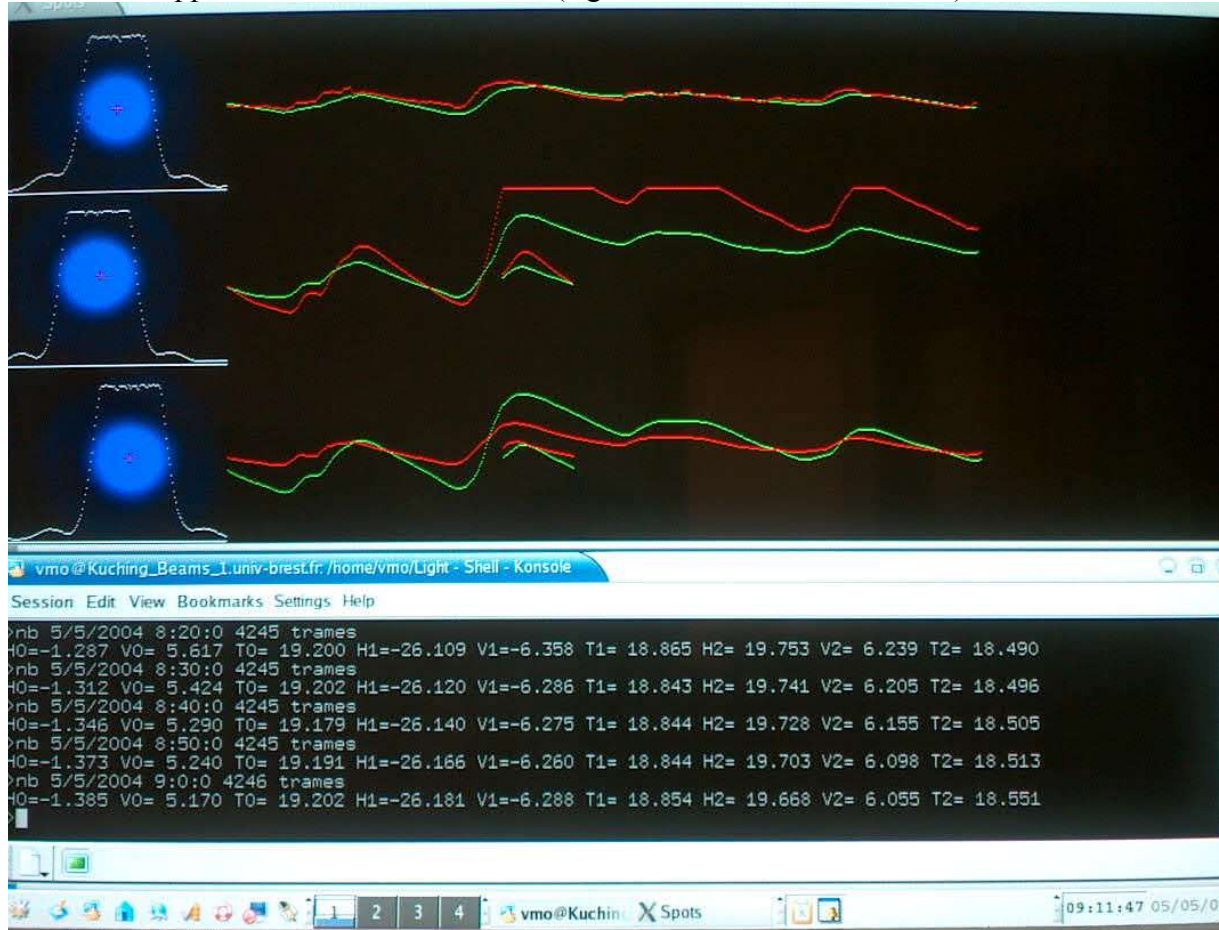
Then a line with current date and time, barycenter positions and temperatures is written to a file (10 items per line).

The whole setup produces 19 values : 1 for the time, 12 spot barycenter coordinates on 6 cameras, and 6 temperatures one on each laser beam.

Ada acquisition software.

The acquisition software uses the SDL library for spot viewing, a reduced interface is in `SDL.ads` for routine and types imports. The interface to the driver is a short piece of C code contained in `acq.c`. The main code is Ada 95 in `devlum.adb`.

The screen appearance is as shown below (signal curves result from trials) :



Curves on spot pictures are spot intensity profiles at barycenter level. Spot tails can be apodized to avoid edge effects.

Starting shell under KDE 3.0 :

```
konsole -e /home/oper/Light/devlum_start.rc
```

Launch script (`devlum_start.rc`) :

```
cd /home/oper/Light
/home/oper/Light/cnf.sh
sudo /home/oper/Light/devlum
```

Camera config script (`cnf.sh`) :

```
../qc-usb/qcset /dev/video0 compatible=3 adaptive=0 keepsettings=1 -c 125 -b 1400
../qc-usb/qcset /dev/video1 compatible=3 adaptive=0 keepsettings=1 -c 175 -b 1400
../qc-usb/qcset /dev/video2 compatible=3 adaptive=0 keepsettings=1 -c 175 -b 1400
```

Driver interface (`acq.c`) :

```

#include <fcntl.h>
#include <unistd.h>
/*-----*/

static int Camera_1;
unsigned char * Image_1;

static int Camera_2;
unsigned char * Image_2;

static int Camera_3;
unsigned char * Image_3;

const ImSize = 352 * 288 * 3;

/*-----*/
/* gport_perm */
void gport_perm()
{
    ioperm( 0x201, 1, 1 );
}
/*-----*/
/* open_cameras */
void open_cameras()
{
    Camera_1 = open("/dev/video0", O_RDWR);
    if (Camera_1 <= 0) printf( "erreur ouverture Cam 1 : %i\n", Camera_1 );
    Image_1 =(char*)( malloc( ImSize ));

    Camera_2 = open("/dev/video1", O_RDWR);
    if (Camera_2 <= 0) printf( "erreur ouverture Cam 2 : %i\n", Camera_2 );
    Image_2 =(char*)( malloc( ImSize ));

    Camera_3 = open("/dev/video2", O_RDWR);
    if (Camera_3 <= 0) printf( "erreur ouverture Cam 3 : %i\n", Camera_3 );
    Image_3 =(char*)( malloc( ImSize ));
}
/*-----*/
/* read_cameras */
void read_cameras() {
    int len1;
    int len2;
    int len3;

    len1 = read( Camera_1, Image_1, ImSize );
    if (len1 != ImSize) printf( "erreur lecture Cam 1 : %i\n", len1 );

    len2 = read( Camera_2, Image_2, ImSize );
    if (len2 != ImSize) printf( "erreur lecture Cam 2 : %i\n", len2 );

    len3 = read( Camera_3, Image_3, ImSize );
    if (len3 != ImSize) printf( "erreur lecture Cam 3 : %i\n", len3 );
}
/*-----*/
/* close_cameras */
void close_cameras() {
    close( Camera_1 ); close( Camera_2 ); close( Camera_3 );}
/*-----*/

```

SDL interface (sdl.ads) :

With System, Interfaces;

Use System, Interfaces;

Package sdl Is

```

Type SDL_Rect      Is Record
  x, y      : Integer_16;
  w, h      : Unsigned_16;
End Record;

```

```

Type SDL_Color     Is Record
  bleu      : Unsigned_8;
  vert      : Unsigned_8;

```

```

    rouge : Unsigned_8;
    unused : Unsigned_8;
End Record;
Type a_SDL_Color Is Access SDL_Color;

Type SDL_Palette Is Record
  ncolors      : Integer;
  colors : a_SDL_Color;
End Record;
Type a_SDL_Palette Is Access SDL_Palette;

Type SDL_PixelFormat Is Record
  palette      : a_SDL_Palette;
  BitsPerPixel : Unsigned_8;
  BytesPerPixel : Unsigned_8;
  Rloss       : Unsigned_8;
  Gloss       : Unsigned_8;
  Bloss       : Unsigned_8;
  Aloss       : Unsigned_8;
  URshift     : Unsigned_8;
  Gshift      : Unsigned_8;
  Bshift      : Unsigned_8;
  Ashift      : Unsigned_8;
  Rmask       : Unsigned_32;
  Gmask       : Unsigned_32;
  Bmask       : Unsigned_32;
  Amask       : Unsigned_32;
  colorkey    : Unsigned_32;
  alpha       : Unsigned_8;
End Record;
Type a_SDL_PixelFormat Is Access SDL_PixelFormat;

Type SDL_Surface Is Record
  Flags : Unsigned_32;
  Format : a_SDL_PixelFormat;
  w, h  : Integer;
  pitch : Unsigned_16;
  pixels : Address;
  offset : Integer;
  hwdata : Address;
  clip_rect : SDL_Rect;
  Unused1 : Unsigned_32;
  locked : Unsigned_32;
  map : Address;
  format_version : Unsigned_32;
  refcount : Integer;
End Record;

Type a_SDL_Surface Is Access SDL_Surface;

Procedure SDL_Init      ( Systeme : Integer );          Pragma Import(C, SDL_Init,
"SDL_Init");
Procedure SDL_Quit;          Pragma Import(C, SDL_Quit, "SDL_Quit");
Function  SDL_SetVideoMode (width, height, bpp: Integer; flags: Unsigned_32)
  Return a_SDL_Surface;          Pragma Import(C, SDL_SetVideoMode,
"SDL_SetVideoMode");
Procedure SDL_LockSurface ( a_S: a_SDL_Surface );          Pragma Import(C,
SDL_LockSurface, "SDL_LockSurface");
Procedure SDL_UnlockSurface ( a_S: a_SDL_Surface );          Pragma Import(C,
SDL_UnlockSurface, "SDL_UnlockSurface");
Procedure SDL_UpdateRect ( a_S : a_SDL_Surface; x, y, width, height: Integer);
  Pragma Import(C, SDL_UpdateRect, "SDL_UpdateRect");
Procedure SDL_WM_SetCaption ( aNomFen, aNomIcône : Address);          Pragma Import(C,
SDL_WM_SetCaption, "SDL_WM_SetCaption");

End SDL;

```

Devlum main code (devlum.adb) :

```

With Unchecked_Conversion;
with Machine_Code, System, SDL, Text_Io, Interfaces, Ada.Calendar, Ada.Command_Line;
use Machine_Code, System, SDL, Text_Io, Interfaces, Ada.Calendar;

```

```

procedure devlum is

    pragma Linker_Options("./acq.o");

--|
--|
package Temp is

    Function Temperature( NoCapteur: Unsigned_8) Return Float;
    Pragma Inline ( Temperature );

end Temp;
--|
--|

--|
--|
package acq is

    Mode_Rapide          : Boolean:= False;

    Procedure Options;
    procedure BoucleAcquisition;
    procedure Close_Cameras;      pragma Import(C, Close_Cameras, "close_cameras");

end acq;
--|
--|

--|
--|
package Body Temp is

    Function Temperature( NoCapteur: Unsigned_8) Return Float Is
        RC: Float; It0, It1: Integer;
    procedure iopl(level: Integer);      pragma Import(C, iopl, "iop1");
    Begin

        iopl(3);
        asm( "cli" );

        asm( "push %%ebx" );
        asm( "push %%ecx" );
        asm( "push %%edx" );
        asm( "push %%esi" );
        asm( "push %%edi" );

--| dans cl le numero de capteur
        asm( "movb %0, %%cl", No_Output_Operands, Unsigned_8'Asm_Input("r", NoCapteur) );
        asm( "movb $0x10, %%bh" );          --| Dans bh le bit 0 à 1
        asm( "shl %%cl, %%bh" );          --| Dans bh le masque de bit pour le port jeu
--| Lire le signal de sortie du capteur
        asm( "    movw $0x0201, %%dx" );
        asm( "    inb %%dx, %%al" );
        asm( "    andb %%bh, %%al" );
        asm( "    movb %%al, %%bl" );      --| Signal de depart dans bl
--| Attendre le changement du signal
        asm( "W_1: movw $0x0201, %%dx" );
        asm( "    inb %%dx, %%al" );
        asm( "    andb %%bh, %%al" );
        asm( "    cmp %%al, %%bl" );
        asm( "    je W_1" );
--| Prendre le temps au basculement
        asm( "    movb %%al, %%bl" );      --| Changer le signal de reference
        asm( "    rdtsc" );
        asm( "    movl %%eax, %%esi" );    --| Temps T1 dans esi
--| Attendre le basculement suivant
        asm( "W_2: movw $0x0201, %%dx" );
        asm( "    inb %%dx, %%al" );

```

```

asm( "    andb %%bh, %%al" );
asm( "    cmp %%al, %%bl" );
asm( "    je W_2" );
--| Prendre le temps au basculement
asm( "    movb %%al, %%bl" );    --| Changer le signal de reference
asm( "    rdtsc" );
asm( "    movl %%eax, %%edi" ); --| Temps T2 dans edi
asm( "    movl %%eax, %%ecx" ); --| Et ecx (calcul seconde portion de periode)
--| Calculer la premiere portion de periode
asm( "    cmp %%esi, %%edi" );  --| T2 - T1 ?
asm( "    jl Wrap1" );         --| T1 > T2 (retour à zero du compteur)
asm( "    subl %%esi, %%edi" ); --| edi contient une premiere portion de periode
asm( "    jmp ST1" );
asm( "Wrap1: movl $0xFFFFFFFF, %%eax" );
asm( "    subl %%esi, %%eax" );
asm( "    addl %%eax, %%edi" );
--| Stocker l'intervalle de temps
asm( "ST1:  cmpb $0, %%bl" );    --| le signal est passe a ?
asm( "    je SP1" );           --| A zero, il etait a 1
--| Stocker l'intervalle 0
asm( "    movl %%edi, %0", Integer'Asm_Output("=a", It0) );
asm( "    jmp W_3" );
--| Stocker l'intervalle 1
asm( "SP1:  movl %%edi, %0", Integer'Asm_Output("=a", It1) );
--| Attendre le basculement suivant
asm( "W_3:  movw $0x0201, %%dx" );
asm( "    inb %%dx, %%al" );
asm( "    andb %%bh, %%al" );
asm( "    cmp %%al, %%bl" );
asm( "    je W_3" );
--| Prendre le temps au basculement
asm( "    rdtsc" );
asm( "    movl %%eax, %%esi" ); --| Temps T3 dans esi (T2 est dans ecx)
--| Calculer la seconde portion de periode
asm( "    cmp %%ecx, %%esi" );  --| T3 - T2
asm( "    jl Wrap2" );
asm( "    subl %%ecx, %%esi" ); --| esi contient une seconde portion de periode
asm( "    jmp ST2" );
asm( "Wrap2: movl $0xFFFFFFFF, %%eax" );
asm( "    subl %%esi, %%eax" );
asm( "    addl %%eax, %%esi" );
--| Stocker l'intervalle de temps
asm( "ST2:  cmpb $0, %%bl" );    --| le signal etait a ?
asm( "    je SP2" );           --| A zero
--| Stocker l'intervalle 1
asm( "    movl %%esi, %0", Integer'Asm_Output("=a", It1) );
asm( "    jmp CRC" );
--| Stocker l'intervalle 0
asm( "SP2:  movl %%esi, %0", Integer'Asm_Output("=a", It0) );
--| Calcul du rapport cyclique
asm( "CRC:  fildl %0", No_Output_Operands, Integer'Asm_Input("m", It0) );
--| fpuST = L0 + L1
asm( "    fiaddl %0", No_Output_Operands, Integer'Asm_Input("m", It1) );
asm( "    fildl %0", No_Output_Operands, Integer'Asm_Input("m", It1) );
--| fpuST = L1 / (L0+L1)
asm( "    fdivp" );
asm( "    fstps %0", Float'Asm_Output("=m", RC) );

```

```

asm( "pop %%edi" );
asm( "pop %%esi" );
asm( "pop %%edx" );
asm( "pop %%ecx" );
asm( "pop %%ebx" );

```

```

asm( "sti" );
iopl(0);

```

```

Return (RC - 0.32) / 0.00470;
Exception When Storage_Error => Put_Line( "storage error in Temp" ); Return 0.0;
End;

```

```

procedure ioperm(From, Len, On: Integer);    pragma Import(C, ioperm, "ioperm");

```

```

Begin

```

```

ioperm( 16#0201#, 1, 1 );

```

```

end Temp;
--|-----|
--|-----|
--|-----|
--|-----|
package body acq is
--|-----|

Terminer : Boolean      := False;
Type Table_Reglages Is Array(0..2) Of Long_Float;
Zooms      : Table_Reglages := (1.0, 1.0, 1.0);
Decals_H   : Table_Reglages := (0.0, 0.0, 0.0);
Decals_V   : Table_Reglages := (0.0, 0.0, 0.0);
Apod_Levels : Table_Reglages := (0.0, 0.0, 0.0);
Jour       : Integer      := 0;

Params_FileName : Constant String := ("./devlum_params.par");
Params          : File_Type;
Package LF_io Is New FFloat_io(Long_Float);

--|-----|
Procedure Write_Params Is
Begin
  Open( Params, Out_File, Params_FileName );
  LF_io.Put( Params, Zooms(0),      3, 2 ); New_Line( Params );
  LF_io.Put( Params, Decals_H(0),   3, 0 ); New_Line( Params );
  LF_io.Put( Params, Decals_V(0),   3, 0 ); New_Line( Params );
  LF_io.Put( Params, Apod_Levels(0), 1, 7 ); New_Line( Params );

  LF_io.Put( Params, Zooms(1),      3, 2 ); New_Line( Params );
  LF_io.Put( Params, Decals_H(1),   3, 0 ); New_Line( Params );
  LF_io.Put( Params, Decals_V(1),   3, 0 ); New_Line( Params );
  LF_io.Put( Params, Apod_Levels(1), 1, 7 ); New_Line( Params );

  LF_io.Put( Params, Zooms(2),      3, 2 ); New_Line( Params );
  LF_io.Put( Params, Decals_H(2),   3, 0 ); New_Line( Params );
  LF_io.Put( Params, Decals_V(2),   3, 0 ); New_Line( Params );
  LF_io.Put( Params, Apod_Levels(2), 1, 7 ); New_Line( Params );
  Close( Params );
End;
--|-----|
Procedure Read_Params Is
Begin
  Begin
    Open( Params, In_File, Params_FileName );
  Exception When Name_Error =>
    Create( Params, In_File, Params_FileName );
    Close( Params );
    Write_Params;
    Open( Params, In_File, Params_FileName );
  End;
  LF_io.Get( Params, Zooms(0)      ); Skip_Line( Params );
  LF_io.Get( Params, Decals_H(0)   ); Skip_Line( Params );
  LF_io.Get( Params, Decals_V(0)   ); Skip_Line( Params );
  LF_io.Get( Params, Apod_Levels(0) ); Skip_Line( Params );
  Apod_Levels(0) := Apod_Levels(0);
  LF_io.Get( Params, Zooms(1)      ); Skip_Line( Params );
  LF_io.Get( Params, Decals_H(1)   ); Skip_Line( Params );
  LF_io.Get( Params, Decals_V(1)   ); Skip_Line( Params );
  LF_io.Get( Params, Apod_Levels(1) ); Skip_Line( Params );
  Apod_Levels(1) := Apod_Levels(1);
  LF_io.Get( Params, Zooms(2)      ); Skip_Line( Params );
  LF_io.Get( Params, Decals_H(2)   ); Skip_Line( Params );
  LF_io.Get( Params, Decals_V(2)   ); Skip_Line( Params );
  LF_io.Get( Params, Apod_Levels(2) ); Skip_Line( Params );
  Apod_Levels(2) := Apod_Levels(2);
  Close( Params );
End;
--|-----|

```

```

Task Commandeur;
Task Body Commandeur Is
  Package Nat_io Is New Integer_io(Natural);
  C : Character;
Begin
  Loop
    Get( C );
    If C = 'Q' Then
      Terminer:= True;
    Elsf C = 'Z' Or C = 'H' Or C = 'V' Or C = 'A' Then
      Declare
        No :Natural;
        No_Incorrect : Exception;
      Begin
        Nat_io.Get( No );
        If No Not In 0..2 Then Raise No_Incorrect; End If;
        if C = 'Z' Then
          LF_io.Get( Zooms(No) );
          Put( "Zoom of Beam " & Natural'Image(No) & " set to " );
          LF_io.Put( Zooms(No), 3, 2 ); New_Line;
          Write_Params;
        Elsf C = 'H' Then
          LF_io.Get( Decals_H(No) );
          Put( "Offset H of Beam " & Natural'Image(No) & " set to " );
          LF_io.Put( Decals_H(No), 3, 3 ); New_Line;
          Write_Params;
        Elsf C = 'V' Then
          LF_io.Get( Decals_V(No) );
          Put( "Offset V of Beam " & Natural'Image(No) & " set to " );
          LF_io.Put( Decals_V(No), 3, 3 ); New_Line;
          Write_Params;
        Elsf C = 'A' Then
          LF_io.Get( Apod_Levels(No) );
          Put( "Apod level of Beam " & Natural'Image(No) & " set to " );
          LF_io.Put( Apod_Levels(No), 3, 3 ); New_Line;
          Apod_Levels(No):= Apod_Levels(No) / 255.0;
          Write_Params;
        Else Null;
        End If;
        Exception When No_Incorrect =>
          Put_Line( "Error : Beam number must be in 0..2" );
        End;
      Elsf C = 'P' Then
        Put_Line( "parameters : " );
        Put( "Beam 0 : zoom = " ); LF_io.Put( Zooms(0), 3, 2 );
        Put( " offset H = " ); LF_io.Put( Decals_H(0), 3, 0 );
        Put( " offset V = " ); LF_io.Put( Decals_V(0), 3, 0 );
        Put( " apod lev = " ); LF_io.Put( Apod_Levels(0) * 255.0, 3, 0 );
        New_Line;
        Put( "Beam 1 : zoom = " ); LF_io.Put( Zooms(1), 3, 2 );
        Put( " offset H = " ); LF_io.Put( Decals_H(1), 3, 0 );
        Put( " offset V = " ); LF_io.Put( Decals_V(1), 3, 0 );
        Put( " apod lev = " ); LF_io.Put( Apod_Levels(1) * 255.0, 3, 0 );
        New_Line;
        Put( "Beam 2 : zoom = " ); LF_io.Put( Zooms(2), 3, 2 );
        Put( " offset H = " ); LF_io.Put( Decals_H(2), 3, 0 );
        Put( " offset V = " ); LF_io.Put( Decals_V(2), 3, 0 );
        Put( " apod lev = " ); LF_io.Put( Apod_Levels(2) * 255.0, 3, 0 );
        New_Line;
      Elsf C = 'I' Then
        Put_Line( "red curve is V signal. green curve is H signal" );
      Else Null;
      End If;
      Exit When Terminer;
    End Loop;
  End Commandeur;
  -----
  NCol : constant Positive:= 352;
  NLin : constant Positive:= 288;

  NLin_R : Constant Long_Float := Long_float(NLin);
  NCol_R : Constant Long_Float := Long_float(NCol);

  type ImageReelle is array(1..NLin,1..NCol) of long_float;

```

```

--|-----|
--|           Package AfficheImages
Package AfficheImages Is
--|-----|

    Procedure Prepare_Affichage;
    Procedure Affiche_Image( No: Natural; Im: ImageReelle; BaryH, BaryV: Long_Float );
    Procedure Maj_Affichage;

End AfficheImages;
--|-----|

--|-----|
Package Body AfficheImages Is

    a_Surface          : a_SDL_Surface;

    Type MatriceImages Is Array(1..3*(NLin/2),1..NCol/2+7*144) Of SDL_Color;
    Type a_MatriceImages Is Access MatriceImages;
    aMI : a_MatriceImages;

    Raz_Done : Boolean := True;

--|-----|
--|           Procedure Prepare_Affichage
Procedure Prepare_Affichage Is
    Function En_a_Matrice Is New Unchecked_Conversion(Address, a_MatriceImages);
    Decal_Jour : Natural := (Natural( Day( Clock ) ) - 1 ) mod 7;
Begin
    SDL_LockSurface( a_Surface );
    aMI:= En_a_Matrice(a_Surface.All.Pixels);
    If Decal_Jour = 0 Then
        If Not Raz_Done Then
            aMI.All:= ( Others=> (Others=>(0,0,0,0) ) );
            Raz_Done:= True;
        End If;
    Else
        Raz_Done:= False;
    End If;
End;
--|-----|
--|           Procedure Affiche_Image
Procedure Affiche_Image( No: Natural; Im: ImageReelle; BaryH, BaryV: Long_Float ) Is
    Decal_L : Natural := No * NLin/2;          --| Decalage vertical de l'image
reduite
Begin
    --| Ecriture de l'image reduite
    For L In 1..NLin/2 Loop
        For C In 1..NCol/2 Loop
            Declare
                Val : Long_Float := 0.25 * ( Im(2*L-1,2*C-1) + Im(2*L-1, 2*C)
                                           +Im(2*L, 2*C-1) + Im(2*L, 2*C)
                                           );
                Pixel : Long_Float := Long_Float'Rounding(255.0 * Val);
            Begin
                if Val >= Apod_Levels(No) Then
                    aMI( L+Decal_L, C ):= (Unsigned_8(Pixel), 0, 0, 0);          --| En
bleu pour les pixels utilises
                Else
                    aMI( L+Decal_L, C ):= (0, 0, Unsigned_8(Pixel), 0);          --| En
rouge pour les pixels apodises
                End if;
            End;
        End Loop;
    End Loop;
    For C In 1..NCol/2 Loop
        aMI( NLin/2+Decal_L, C ):= (255, 255, 255, 0);
    End Loop;
    --| Ecriture des infos barycentriques

```

```

Declare
  Hm : Natural:= 1 + Natural( Long_Float'Rounding( 0.25 * NCol_R + 0.5 * BaryH ));
  Vm : Natural:= 1 + Natural( Long_Float'Rounding( 0.25 * NLin_R + 0.5 * BaryV ))
      + Decal_L;
Begin
  --| Ecriture de la courbe de niveau au barycentre
  For C In 1.. NCol/2 Loop
    aMI( (NLin - Natural(aMI( Vm, C ).bleu)) / 2 + Decal_L , C ):= (255,255,255,0);
  End Loop;

  --| Trace de la croix barycentrique
  Declare
    Hd : Natural:= Hm - 4;
    Hf : Natural:= Hm + 4;
    Vd : Natural:= Vm - 4;
    Vf : Natural:= Vm + 4;
  Begin
    If Hd < 1 Then Hd:= 1; End If;                                --/
  Limiter la branche verticale
    If Hf > NCol Then Hf:= NCol; End If;
    For nP In Hd..Hf Loop aMI(Vm,nP):= (0,0,255,0); End Loop;    --|
  Tracer
    If Vd < Decal_L + 1 Then Vd:= Decal_L + 1; End If;          --|
  Limiter la branche horizontale
    If Vf > Decal_L + NLin Then Vf:= Decal_L + NLin; End If;
    For nP In Vd..Vf Loop aMI(nP, Hm):= (0,0,255,0); End Loop;  --|
  Tracer
  End;

  Declare
    R_BaryH      : Long_Float      := Long_Float'Rounding(
image reduite de moitie                                     0.25 * NLin_R                                --| Demi-
                                                         + Zooms(No) * (BaryH + Decals_H(No) )
                                                         );
    R_BaryV      : Long_Float      := Long_Float'Rounding(
                                                         0.25 * NLin_R
                                                         + Zooms(No) * (BaryV + Decals_V(No) )
                                                         );
    L_BaryH, L_BaryV : Positive;
  Begin
    If R_BaryH < 0.0 Then R_BaryH:= 0.0; End If;
    If R_BaryH >= 0.5*NLin_R Then R_BaryH:= 0.5*NLin_R - 1.0; End If;
    L_BaryH:= Decal_L + Natural(R_BaryH) + 1;

    If R_BaryV < 0.0 Then R_BaryV:= 0.0; End If;
    If R_BaryV >= 0.5*NLin_R Then R_BaryV:= 0.5*NLin_R - 1.0; End If;
    L_BaryV:= Decal_L + Natural(R_BaryV) + 1;

    If Mode_Rapide Then
      Declare
        Tx: Natural:= NCol/2 + Natural(Seconds(Clock)) mod (7*144);
      Begin
        aMI(L_BaryH, Tx):= (0,255,0,0);
        aMI(L_BaryV, Tx):= (0,0,255,0);
      End;
    Else
      Declare
        T      : Time      := Clock;
        Secs   : Long_Float := Long_Float( Seconds( T ) );
        Decal_Jour : Natural := (Natural( Day( T ) ) - 1 ) mod 7;
        Tx     : Natural   := NCol/2
          + 144 * Decal_Jour
          + Natural( Long_Float'Rounding( 144.0 * Secs /
86_400.0 ));
      Begin
        aMI( L_BaryH, Tx ):= (0, 255,0, 0);
        aMI( L_BaryV, Tx ):= (0, 0, 255,0);
      End;
    End If;
  End;
End;
End;

```

```

--|-----
Procedure Maj_Affichage Is
Begin
  SDL_UpdateRect( a_Surface, 0,0, NCol/2+7*144, 3*(NLin/2) );
  SDL_UnlockSurface( a_Surface );
End;
--|-----

Chn_1      : aliased String      := "Spots" & Ascii.Nul;
ChnIcône  : aliased String      := " " & Ascii.Nul;

Begin
  SDL_Init( 16#20# );
  a_Surface:= SDL_SetVideoMode( NCol/2+7*144, 3*(NLin/2), 32, 16#1# );
  SDL_WM_SetCaption( Chn_1(1)'Address, ChnIcône(1)'Address );
End AfficheImages;
--|-----

TempsAcquisition : Duration      := 9 * 60.0;
cumul             : Duration      := 10 * 60.0;
minutes de cycle  : Natural       := 365;
JoursExperience   : Natural       := 365;

Type aString      Is Access String;
aData_Filename   : aString;
Data             : File_Type;

procedure Open_Cameras;      pragma Import(C, Open_Cameras, "open_cameras");
procedure Read_Cameras;     pragma Import(C, Read_Cameras, "read_cameras");

type TripletRVB      is record
  B,V,R              : Unsigned_8;
end record; pragma Pack(TripletRVB);
24;
type ImageRVB        is array(1..NLin,1..NCol) of TripletRVB;
type aImageRVB       is access ImageRVB;
aImageCamera_1      : aImageRVB;
aImageCamera_1, "Image_1";
aImageCamera_2      : aImageRVB;
aImageCamera_2, "Image_2";
aImageCamera_3      : aImageRVB;
aImageCamera_3, "Image_3";

type ImageCumul is array(1..NLin,1..NCol) of natural;
CumulB_1, CumulB_2, CumulB_3 : ImageCumul;

ImMoyB_1, ImMoyB_2, ImMoyB_3 : ImageReelle;

BarXB_1, BarYB_1,
BarXB_2, BarYB_2,
BarXB_3, BarYB_3           : Long_Float;

--|-----
--|
procedure Bary(Ir: ImageReelle; No_Image: Natural; BarX, BarY:out Long_float) is
T, Bx, By: Long_Float:= 0.0;
begin
  for L in 1..NLin loop
    declare
      Y: Long_Float:= Long_Float(L-NLin/2) - 0.5;
    begin
      for C in 1..NCol loop
        declare
          Val : Long_Float:= Ir(L,C);
        Begin
          If Val >= Apod_Levels(No_Image) Then
            Declare

```

```

        X : Long_Float:= Long_float(C-NCol/2) - 0.5;
    begin
        T:= T + Val;
        Bx:= Bx + Val * X; By:= By + Val * Y;
    end;
    End if;
end;
end loop;
end;
end loop;
If T /= 0.0 Then
    BarX:= Bx / T; BarY:= By / T;
Else BarX:= 0.0; BarY:= 0.0;
End If;
end;
-----
--|
--|
Function Chaine_DateHeure (T: Time ; CompatibleFichier: Boolean:= False) Return String Is
    J : Day_Number := Day( T ); sJ: constant String:= Day_Number'Image( J );
    Mois : Month_Number:= Month( T ); sMois: constant String := Month_Number'Image( Mois );
    An : Year_Number := Year( T ); sA: constant String:= Year_Number'Image( An );
    Sec : Natural:= Natural(Seconds( T ));
    H : Natural:= Sec/3600; sH: constant String:= Natural'Image( H );
    M : Natural:= (Sec-3600*H) / 60; sM: constant String:= Natural'Image( M );
    S : Natural:= (Sec-3600*H-60*M); sS: constant String:= Natural'Image( S );
    SepaBlanc : String(1..1) := " ";
    SepaDate : String(1..1) := "/";
Begin
    If CompatibleFichier Then SepaBlanc:= "-"; SepaDate:= "_"; End If;
    Return sJ(2..sJ'Last) & SepaDate & sMois(2..sMois'Last) & SepaDate & sA(2..sA'Last)
        & SepaBlanc & sH(2..sH'Last) & ":" & sM(2..sM'Last) & ":" & sS(2..sS'Last);
End;
-----
--|
--|
procedure Acquiert is
    Nb_Acq : Natural:= 0;
    Present : Time:= Clock;
    Arret : Time:= Present + TempsAcquisition;
    Sort : Time:= Present + Intervalle;
    Moy_T0, Moy_T2, Moy_T3: Long_Float;
begin
    CumulB_1:= ( 1..NLin=> (1..NCol=> 0) );
    CumulB_2:= ( 1..NLin=> (1..NCol=> 0) );
    CumulB_3:= ( 1..NLin=> (1..NCol=> 0) );

    Put( ">" );
    Declare
        Cumul_T0, Cumul_T2, Cumul_T3: Long_Float := 0.0;
    Begin
CUMUL:
        Loop
            Exit CUMUL When Terminer;
si demande par la tache de surveillance
            Declare
                T0, T2, T3 : Long_Float;
            Begin
                T0:= Long_Float(Temp.Temperature( 0 ));
                T2:= Long_Float(Temp.Temperature( 2 ));
                T3:= Long_Float(Temp.Temperature( 3 ));

                Cumul_T0:= Cumul_T0 + T0;
                Cumul_T2:= Cumul_T2 + T2;
                Cumul_T3:= Cumul_T3 + T3;
            End;

            Select
                Delay 5.0;
de lecture pour les capteurs
                Put_Line( "Erreur de blocage des lectures de camera" );
                Then Abort
                    Read_Cameras;
les capteurs
            End Select;

```

--| Arret

--| Delai

--| Lire

```

Nb_Acq:= Nb_Acq + 1;

for L in 1..NLin loop
  for C in 1..NCol loop
    CumulB_1(L,C) := CumulB_1(L,C) + Natural(aImageCamera_1(L,C).B);
    CumulB_2(L,C) := CumulB_2(L,C) + Natural(aImageCamera_2(L,C).B);
    CumulB_3(L,C) := CumulB_3(L,C) + Natural(aImageCamera_3(L,C).B);
  end loop;
end loop;
exit CUMUL when Clock > Arret;
If Nb_Acq > 5000 Then Put( "Erreur arret" ); Exit CUMUL; End If;

End loop CUMUL;

Declare
  InvNbAcq: Long_Float:= 1.0 / Long_Float(Nb_Acq);
Begin
  Moy_T0:= Cumul_T0 * InvNbAcq;
  Moy_T2:= Cumul_T2 * InvNbAcq;
  Moy_T3:= Cumul_T3 * InvNbAcq;
End;
End;

Put( "n" );
NORMALISE:
Declare
  InvNbAcqFact: constant Long_Float:= 1.0 / (255.0 * Long_Float(Nb_Acq));
Begin
  for L in 1..NLin loop
    for C in 1..NCol loop
      ImMoyB_1(L,C) := Long_Float(CumulB_1(L,C)) * InvNbAcqFact;
      ImMoyB_2(L,C) := Long_Float(CumulB_2(L,C)) * InvNbAcqFact;
      ImMoyB_3(L,C) := Long_Float(CumulB_3(L,C)) * InvNbAcqFact;
    end loop;
  end loop;
End NORMALISE;

Put( "b " );
Bary( ImMoyB_1, 0, BarXB_1, BarYB_1 );
Bary( ImMoyB_2, 1, BarXB_2, BarYB_2 );
Bary( ImMoyB_3, 2, BarXB_3, BarYB_3 );

STOCKAGE:
Declare
  MiTemps : Time := Present + Intervalle/2.0;
  J : Day_Number := Day( MiTemps ); sJ: constant String:= Day_Number'Image( J );
  Mois : Month_Number:= Month( MiTemps ); sMois: constant String :=
Month_Number'Image( Mois );
  An : Year_Number := Year( MiTemps ); sA: constant String:= Year_Number'Image
( An );
  Sec : Natural:= Natural(Seconds( MiTemps ));
  H : Natural:= Sec/3600; sH: constant String:= Natural'Image( H );
  M : Natural:= (Sec-3600*H) / 60; sM: constant String:= Natural'Image( M );
  S : Natural:= (Sec-3600*H-60*M); sS: constant String:= Natural'Image( S );
type Fixe3 is delta 0.001 range -1_000.0..1_000.0;
Begin
  Put_Line( Chaîne_DateHeure(MiTemps) & Natural'Image(Nb_Acq)&" trames" );
  Put( "H0=" & Fixe3'Image(Fixe3(BarXB_1)) );
  Put( " V0=" & Fixe3'Image(Fixe3(BarYB_1)) );
  If Not Mode_Rapide Then Put( " T0=" & Fixe3'Image(Fixe3(Moy_T0)) ); End If;
  Put( " H1=" & Fixe3'Image(Fixe3(BarXB_2)) );
  Put( " V1=" & Fixe3'Image(Fixe3(BarYB_2)) );
  If Not Mode_Rapide Then Put( " T1=" & Fixe3'Image(Fixe3(Moy_T2)) ); End If;
  Put( " H2=" & Fixe3'Image(Fixe3(BarXB_3)) );
  Put( " V2=" & Fixe3'Image(Fixe3(BarYB_3)) );
  If Not Mode_Rapide Then Put( " T2=" & Fixe3'Image(Fixe3(Moy_T3)) ); End If;
  New_Line;

  If Not Mode_Rapide Then
    Open( Data, Append_File, aData_FileName.All );
    Put_Line( Data, Chaîne_DateHeure(MiTemps) & Ascii.HT
& Fixe3'Image(Fixe3(BarXB_1)) & Ascii.HT & Fixe3'Image(Fixe3(BarYB_1)) &
Ascii.HT & Fixe3'Image(Fixe3(Moy_T0)) & Ascii.HT
& Fixe3'Image(Fixe3(BarXB_2)) & Ascii.HT & Fixe3'Image(Fixe3(BarYB_2)) &

```

```

Ascii.HT & Fixe3'Image(Fixe3(Moy_T2)) & Ascii.HT
      & Fixe3'Image(Fixe3(BarXB_3)) & Ascii.HT & Fixe3'Image(Fixe3(BarYB_3)) &
Ascii.HT & Fixe3'Image(Fixe3(Moy_T3))
      );
  Close( Data );
End If;
End STOCKAGE;

AfficheImages.Prepare_Affichage;
AfficheImages.Affiche_Image( 0, ImMoyB_1, BarXB_1, BarYB_1 );
AfficheImages.Affiche_Image( 1, ImMoyB_2, BarXB_2, BarYB_2 );
AfficheImages.Affiche_Image( 2, ImMoyB_3, BarXB_3, BarYB_3 );
AfficheImages.Maj_Affichage;

loop exit when Clock > Sort Or Terminer; end loop;
end;
-----
--|
--|           procedure BoucleAcquisition
procedure BoucleAcquisition is
  NbAcquisitions: Natural:= 1 + JoursExperience*86400 / Natural(Intervalle);
begin
  Put_Line( "Nb acquisitions : " & Natural'Image(NbAcquisitions) );
  Read_Params;
  If Not Mode_Rapide Then
    aData_FileName:= New String'( "./DATA-" & Chaine_DateHeure(Clock,
CompatibleFichier=>True) & ".txt");
    Put_Line( "Filing on : " & aData_FileName.All );
    Create( Data, Out_File, aData_FileName.All );
    Close( Data );
    Loop
      Exit When Terminer;
      Exit When Natural(Seconds( Clock )) mod 600 = 300;
    End Loop;
  End If;

  Put_Line( "Parti" );
  for Na in 1..NbAcquisitions loop
    exit When Terminer;
    Acquiert;
  end loop;

end;
-----
--|
--|           procedure Options
procedure Options Is
  Use Ada.Command_Line;
  Nargs: Natural;
  Na: Natural:= 1;
Begin
  Nargs:= Argument_Count;

  While nA <= NArgs Loop
    If Argument( nA ) = "-f" Then
      Mode_Rapide      := True;
      TempsAcquisition := 0.7;
      Intervalle       := 1.0;
      nA:= nA + 1;
    Elself Argument( nA ) = "-Z" Then
      nA:= nA + 1;
      Declare
        No :Natural := Natural'Value( Argument( nA ) );
      Begin
        nA:= nA + 1;
        Zooms(No):= Long_Float'Value( Argument( nA ) );
        nA:= nA + 1;
      End;
    Elself Argument( nA ) = "-A" Then
      nA:= nA + 1;
      Declare
        No :Natural := Natural'Value( Argument( nA ) );
      Begin
        nA:= nA + 1;
        Apod_Levels(No):= Long_Float'Value( Argument( nA ) );
        nA:= nA + 1;
      End;
    End;
  End;

```

```
Elsif Argument( nA ) = "-d" Then
  nA:= nA + 1;
  JoursExperience:= Natural'Value( Argument( nA ) );
  nA:= nA + 1;
Elsif Argument( nA ) = "-h" Then
  Put_Line( "DevLum Options : " );
  Put_Line( "-f                                fast mode" );
  Put_Line( "-Z <natural> <float>              zoom factor" );
  Put_Line( "-A <natural> <float>              apodization level" );
  Put_Line( "-d <natural>                      acquisition duration in days" );
End If;
End Loop;
End;
```

```
begin
  Open_Cameras;
end acq;
```

```
begin
  acq.Options;
  acq.BoucleAcquisition;
  SDL_Quit;
  acq.Close_Cameras;
exception
  When Others =>
    SDL_Quit;
    acq.Close_Cameras;
end;
```

Realization costs

Category	Description	Reference	Manuf.	Shop		Unit Price	Ship.	Price	Total/categ
Optics	Red laser module 1mW	LDM115P-670/1	Imatronic		6	133,00 €		972,97 €	972,97 €
Tools	Alésoir 11mm	F1231	Titex	Orefi	1	66,16 €			
	Taraud 2,5	B1149	Titex	Orefi	1	31,12 €			
	Freinfillet faible	222	Loctite	Orefi		26,95 €		116,35 €	
	Forets HSS-TIN 2,1mm			Bellion	3	1,26 €		32,23 €	
	Foret 11mm HSS court		Titex	Orefi	1	6,69 €		4,52 €	
								8,00 €	161,10 €
Electronic Hardware	RCA Male/female			RadioSell	6	1,12 €		8,04 €	
	Temperature sensors SubD plugs wires...	SMT160-30 TO18	Smartec	Smartec RadioSell	8	5,95 €		56,93 € 20,61 €	
									85,58 €
Mechanical Hardware	Springs x10 3,5mm diam.			Castorama	1	2,34 €			
	Inox rule width 1cm			Castorama	1	2,15 €			
	Colliers fils x2			Castorama	6	1,24 €		7,94 €	
	Robax strips 900x30x4	Robax	Schott	VIO	14	20,00 €	30,00 €	8,88 €	400,66 €
	1m Inox tige fileté diam. 8mm			Orefi	9	1,25 €			
	Ecrous à oreilles inox			Orefi	27	0,12 €			
	Ecrous inox			Orefi	55	0,03 €			
	Rondelles inox			Orefi	82	0,04 €			
	Tube pvc blanc diam.12 ep 1,2				8	3,50 €		23,23 €	
	Inox rule width 2cm				1	5,38 €			
	Panneau 250x60x7	poly.extr. Foam	Knauf	PointP	3	13,33 €		39,96 €	
	1m Invar bar 25mm diam.	Dilaton 36	Phynicx		1	220,00 €		71,75 € 263,12 €	
								815,54 €	
Computer	Computer N°1			TechMedia	1			275,00 €	
	Atx box								
	Motherboard	Asrock K7VM2 R3							
	Athlon XP2200+ box								
	DDR PC2700 256Mo								
	Keyboard + mouse								
	Dlink PCI/USB								
Motherboard	Asrock K7VM2 R3		Materiel.Net	1	11,45 €		13,00 €		
Atx box			TechMedia	1	40,42 €	7,93 €	48,35 €		
Athlon XP2200+ box		AMD	Materiel.Net	1	40,97 €		49,00 €		
DDR PC2700 256Mo			Materiel.Net	1					
							128,02 €		
								513,37 €	
								2 548,56 €	

Part of the costs is induced by the prototype nature of the setup (trials, unavailable tools...), a second realization should be slightly less expensive.