HARDWARE DEVELOPMENT FOR COMPLETE SYSTEM CONTROL

1. Objective

Present lab session's objective is the design of the hardware needed to control both sensor (digital camera) and actuators (stepping motors) within the intended application. This hardware will have to communicate with application software when the whole design is integrated into a unique application including web page, camera and motors.
More precisely, in this session we'll have to design the hardware subsystem that detects image changes and decides how to move when we are in tracking mode.

2. Introduction

The final aim of our application is to implement a two axes camera positioning system with two working modes:

- **Scan Mode:** Camera is continuously moving along both axes and sends images at given positions along its path.
- **Tracking Mode:** Camera moves along both axes and acquires two images per position with an interval of 1s. These two images are compared according to an algorithm to be defined later and from the comparison we have to decide whether to move the camera and where to point it, with the final objective of keeping track of any image changes within our field of view.

Figure 1 show the mechanical hardware used in the application.

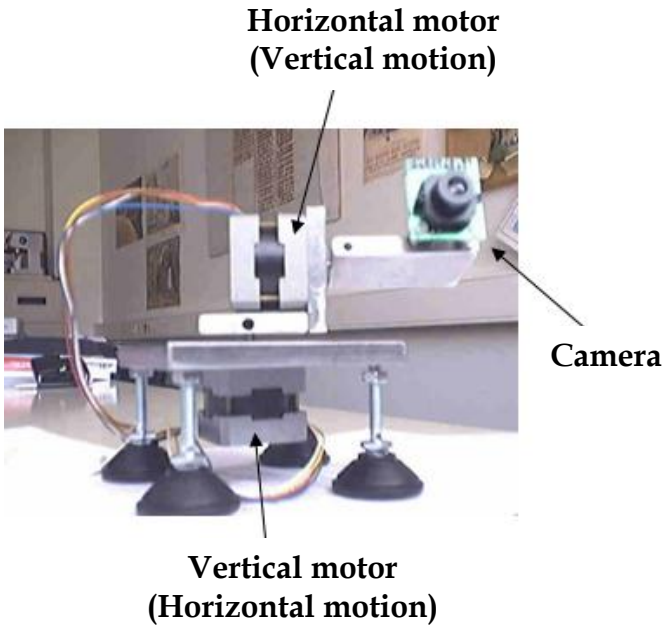

Figure 1. Camera fixture.

Both movements will be limited to a 90º sector from camera home position to avoid any damage to electrical connections.

In scan mode, stepping motors move in one step increments, i.e.,1.8º. This fact implies that the camera will go through 50 x 50 = 2500 different positions per scan. At the same time, the camera will capture images on the fly and send them to a software application that in its turn sends them to the remote site for visualisation.

In tracking mode, the camera will perform the same scan of 90º sectors in both axes but this time the increment will be of 10 steps, i.e.,18º between successive positions, to a total of 25. Camera movement sequence is shown in figure 2.

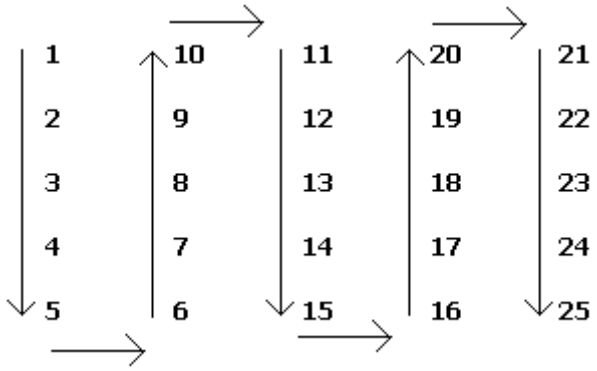

Figure 2. Camera sequence in tracking mode.

Position 1 is the camera home position. From that point, horizontal motor will move the camera downwards to position 5 in 18º increments. Then, vertical motor will move the camera to the right to position 6, again 18º. From this point, horizontal motor will move the camera upwards to position 10. Sequence will then continue as shown in figure 2 to position 25. After this, movement will proceed to follow the sequence in the opposite direction back to position 1 and so on.

The total number of 25 positions is a compromise between the time needed for a complete scan and the overlapping of successive images. Because the final objective is to detect any movement within the camera field of view, it is considered that 25 acquisitions will offer enough resolution for that purpose.

At each of the 25 positions, we'll store two images with 1s interval. However, because the camera is continuously capturing images, what we will do in fact is to store an image, discard a given number equivalent to 1s (i.e., 50 assuming camera is set at 50 images/s) and store a new one. These two images will be stored in an internal FIFO (First In First Out) subsystem. By comparing this two stored images, we have to decide where to go next. In case that the difference is below a given threshold (no object movement within the field of view) we'll move to the next point in the sequence. However, if the difference is above that threshold, indicating that something has changed within the field of view, we'll point the camera to the image section where the largest change occurred.

3. Development specification

As has been said, in tracking mode we'll acquire, at each of the 25 positions, two images separated by 1s interval. Control system has an internal signal named ***Move*** (valid high) indicating when we have to acquire a new image.

To rationalize the quantity of information to be stored and processed, only one out of every 16 pixels will be stored. In the same way, we'll keep only one out of every 16 lines. This sets the size of the two internal FIFO's at 24 x 18 = 432 bytes instead of the 110592 bytes that would be needed in case of storing the whole 384 x 288 píxel image. Control system will issue a signal named ***MNewdata*** (valid high) every time we have to store a new data into the corresponding FIFO. Also, two additional signals named ***Startframe*** and ***Endframe*** (both valid high) will be issued by the control system at start and end of a new image.

Once the two (compressed) images are in storage, we'll proceed to read and compare them, one pixel at a time. Control system has another internal signal called ***index*** that counts the pixel's index (up to 432) at every comparison step. If the comparison yields an absolute value greater than 12, pixels will be considered different.

Image is divided into nine zones of 8 x 6 = 48 pixel each, numbered as in figure 3. These zones will be used to identify where the camera has to point

0	3	6
1	4	7
2	5	8

Figure 3. Image control zones.

While we proceed with pixel comparison, we have to calculate, for each of the nine zones, the number of pixels that do not match, i.e., that are different. After that, we determine which zone has the maximum count of non-matching pixels. Considering then that the present camera position corresponds to zone 4, we can decide the next move. If, for example, the maximum count is for zone 8, then next move will be 18º downwards and 18º to the right. There is also a threshold, 15 in the present application, for the total number of different pixels within the dominant zone. Then, if the corresponding count is less than 15 and/or the maximum count is at zone 4, sequence will continue unaltered.

According to what previously stated, the state machine that controls all that process looks like that of figure 4.

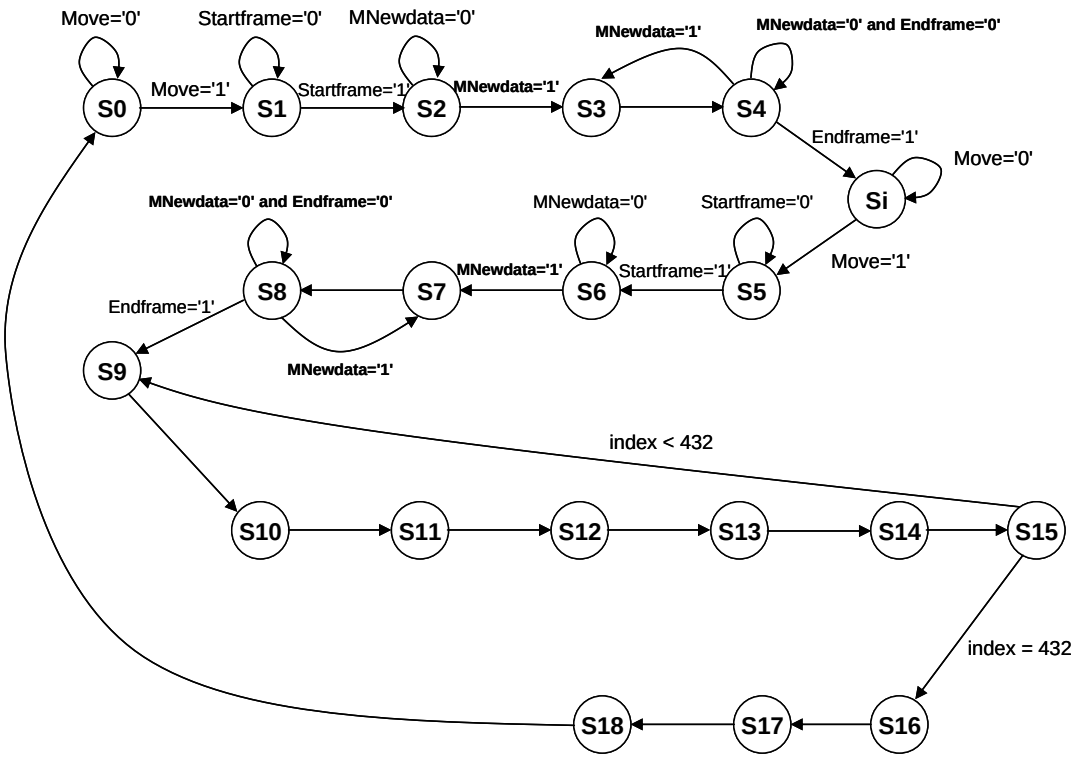

Figure 4. State diagram of the movement control subsystem

Here follows the description of the steps in figure 4:

- **S0:** Controller initial state. System will remain in this state until request for first image capture. System will go back to this state after a reset indicated by signal ***clearn*** (synchronous and active low).
- **S1:** Waiting state to the start of first image. System will remain in this state until the image start signal is active.
- **S2:** Waiting state to the start of first pixel. System will remain in this state until the received pixel signal is active.
- **S3:** First FIFO write state. During this state, signal that clocks a pixel into the FIFO is activated.
- **S4:** New pixel waiting state. System will remain in this state until a new pixel arrives or an end of frame signal is received.
- **Si:** Waiting state to the request for a second image. System will remain in this state until the image request signal is active again.
- **S5:** Waiting state to the start of second image. System will remain in this state until the image start signal is active.
- **S6:** Waiting state to the start of first pixel. System will remain in this state until the received pixel signal is active.
- **S7:** Second FIFO write state. During this state, signal that clocks a pixel into the FIFO is activated.
- **S8:** New pixel waiting state. System will remain in this state until a new pixel arrives or an end of frame signal is received.

- **S9:** FIFOs read state. During this state, signal enabling FIFOs read is activated.
- **S10:** FIFOs data read state. During this state, signal that reads a pixel out of the FIFOs is activated.
- **S11:** Pixel difference calculation state. During this state, the difference between the two corresponding pixels in the FIFOs is calculated.
- **S12:** Difference thresholding state. During this state, a signal called ***result*** becomes active if difference absolute value is greater than 12.
- **S13:** Increment count state. During this state, if signal ***result*** is active then the counter associated to the corresponding control zone will be incremented in one unit.
- **S14:** Increment index state. During this state, signal ***index*** that counts the number of pixels read will be incremented in one unit.
- **S15:** End of difference calculation state. In this state, ***index*** value is compared to the total number of pixels to be processed to detect the end of comparison stage.
- **S16:** Dominant zone decision state. During this state, the zone that has the maximum difference number is chosen.
- **S17:** Movement decision state. During this state, the dominant zone count is thresholded with 15 to decide the next move.
- **S18:** Movement generation state. During this state, motor controller is ordered next move.

To realize this design, a file named ***control_hardware.vhd*** is provided. This file already contains functional description of the controller. Using this file:

1. Add the state machine description. Signal that indicates machine present state has to be named ***ME2*** and is already defined within the file as:

```
Type   MEstat is( S0, S1, S2, S3, S4, S5, S6, S7, S8, S9,
S10, S11, S12, S13, S14, S15, S16, S17, S18, Si );
Signal ME2: MEstat;
```

2. Add the description for the calculation of the absolute value of the difference between two pixels in the FIFOs. FIFO output signals are named as ***dout_fifo(7 downto 0)*** and ***dout_fifo2(7 downto 0)***. Signal indicating that the calculated value is greater than 12 has to be called ***result***.

3. Add description corresponding to the calculation of difference count for each zone. Signal names for the required zones are: ***zone0(5 downto 0)***, ***zone1(5 downto 0)***, ***zone2(5 downto 0)***, ***zone3(5 downto 0)***, ***zone5(5 downto 0)***, ***zone6(5 downto 0)***, ***zone7(5 downto 0)*** i ***zone8(5 downto 0)***.

4. Add the description for the calculation of the critical zone, i.e., the zone with the greatest difference count (above 15). The signal holding the critical zone number, in binary, has to be called ***zone(3 downto 0)***. Apart from that, a signal called ***accum(5 downto 0)***, holding the maximum count number has to be generated. After a system reset or in state S0, ***zone*** will hold the value "0100" and ***accum*** "000000".

When realizing the descriptions specified above, note that signals ***Move***, ***Startframe***, ***Endframe*** i ***MNewdata*** are already defined. System clock will be named ***clk*** and will act at its rising edge. The system reset signal will be called ***clearn*** and will act synchronously and valid low.

Once finished the design, entity ***control_hardware*** will be simulated using provided files ***prova_control_hardware.vhd*** and ***simula.do***.