Sumari

ANNEXES

A. Característiques dels components

En aquest annex es dona el *datasheet* del LA12 i la documentació tècnica del mòdul MD03.

A1. Actuador LA12. Product Datasheet

PRODUCT DATA SHEET

ACTUATOR LA12

Features:

- *12/24 V DC permanent magnetic motor*
- *Max. thrust 750 N*
- *Duty cycle up to 100% at 20°C ambient temperature (see duty cycle graphs)*
- *Ambient temperature -20° to +40° C*
- *Reinforced glass fibre piston rod*
- *Compact design*
- *Protection class: IP 51*
- *Colour: black*
- *750 mm straight cable without plug or 2300 mm straight cable with jack.*
- *Back fixture available in 2 different variants: 01 or 02 (factory mounted)*
- *Built-in limit switches (not adjustable)*
- *High-strength plastic housing protects motor and gear*

Options:

- *Reed-switch*
- *Potentiometer (max. 100 mm stroke length)*
- *Protection class IP 65 and IP 66*
- *ATEX approved for dust explosive atmospheres*

Thanks to the small size and outstanding performance, the LA12 actuator provides a practical and cost-effective alternative to traditional pneumatic systems and gear motors.

The LA12 is ideal for automating industrial and agricultural machines, feeding, ventilation systems troughs and many other applications requiring short linear movement.

LINAK®

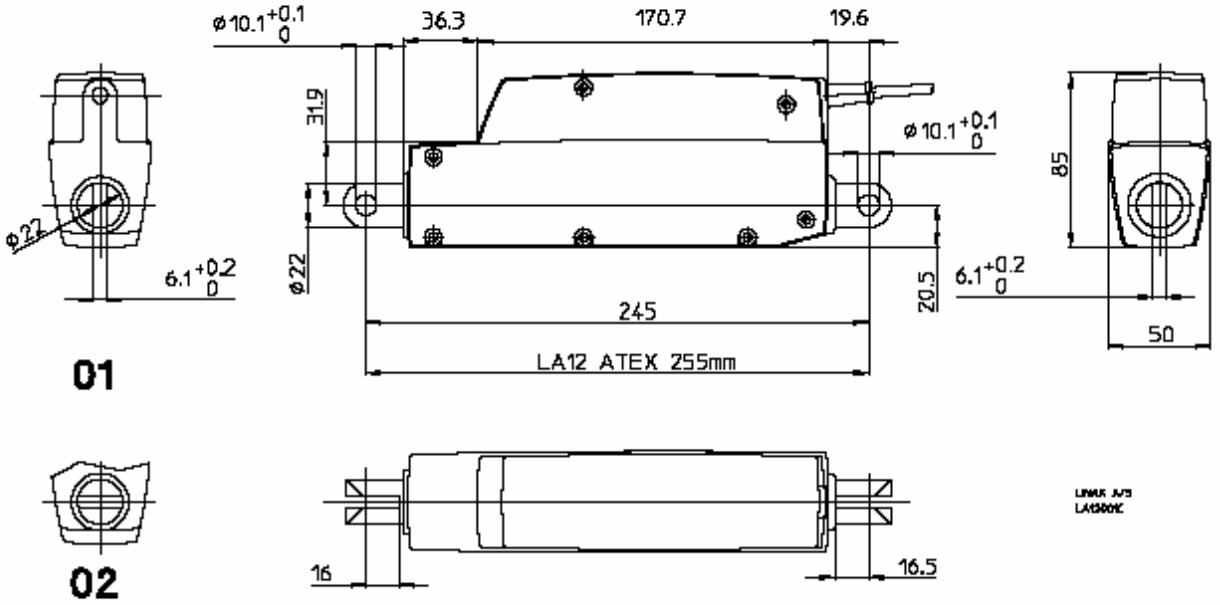
WE IMPROVE YOUR LIFE

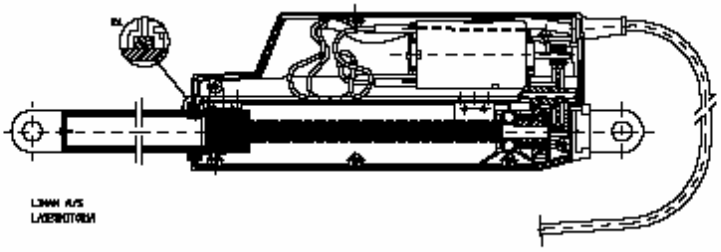
Technical specifications:

New Type		Old Type	Spindle pitch (mm)	Thrust max. Push/Pull (N)	Self-Lock Max. (Push) (N)	Self-Lock Max. (Pull) (N)	Typical speed 0/full load (mm/s)		Stroke length (in steps of 30 mm)			Typical Amp. at full load (A) 24V - 12V	
12XX00-1XXX12XX		12.1	2	750	750	375	14	5	40	-	130	-	4,6
12XX00-1XXX24XX		12.1	2	750	750	375	14	6	40	-	130	2,2	-
12XX00-2XXX12XX		12.2	4	300	300	150	27	16	40	-	130	-	2,5
12XX00-2XXX24XX		12.2	4	300	300	150	27	16	40	-	130	1,5	-
12XX00-2XXX24XX	#	12.2	4	300	300	150	50	27	40	-	130	2,0	-
12XX00-3XXX12XA		12.3	6	200	200	100	40	28	40	-	130	-	2,2
12XX00-3XXX24XA		12.3	6	200	200	100	40	28	40	-	130	1,0	-

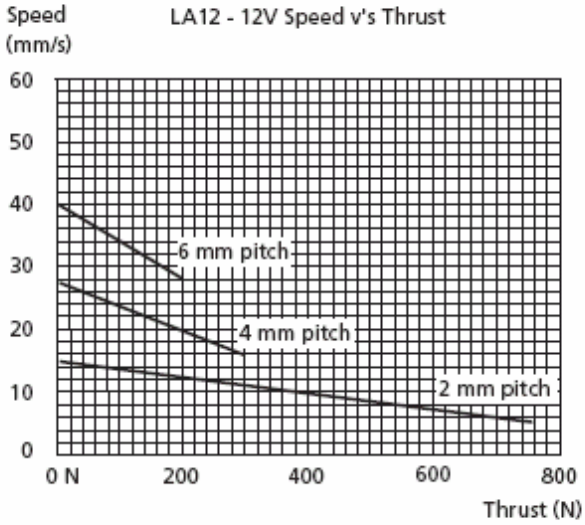
Estimation made in connection with a CBD1.
The above values are with an ambient temperature of 20°C

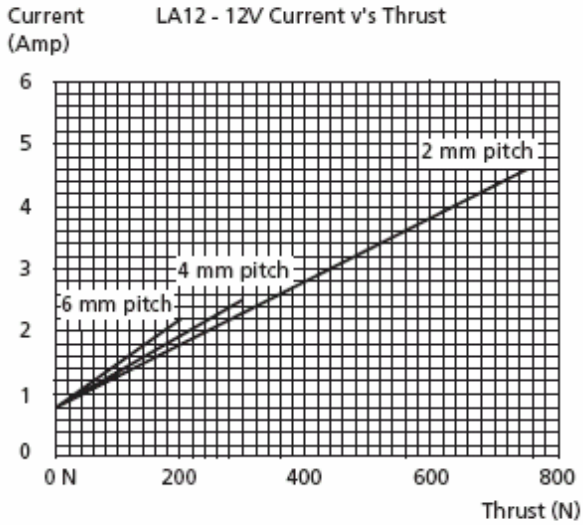
Dimensions:

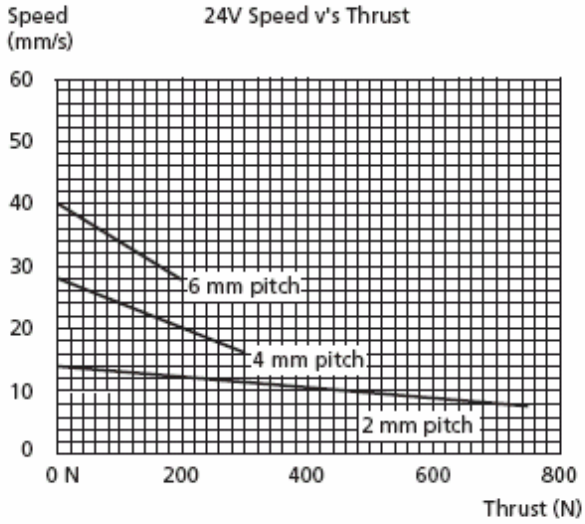


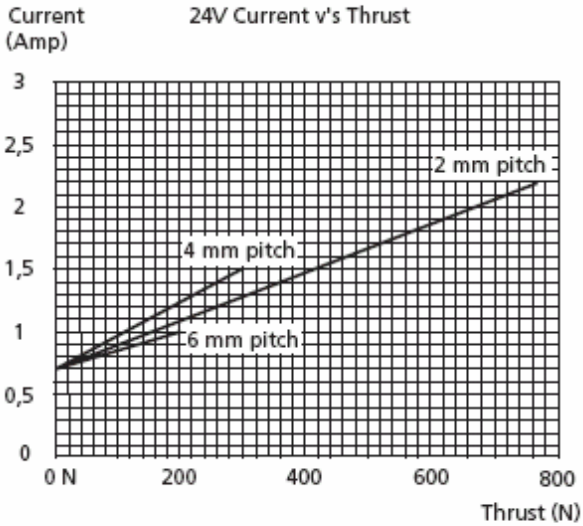


LA 12 Curves speed and current:





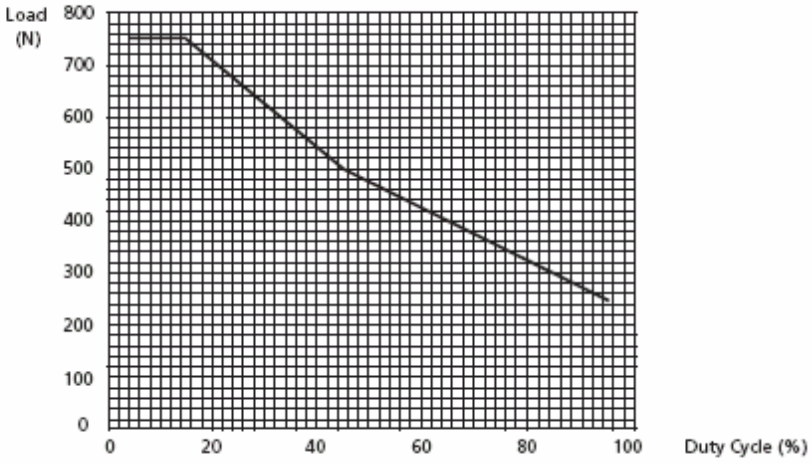




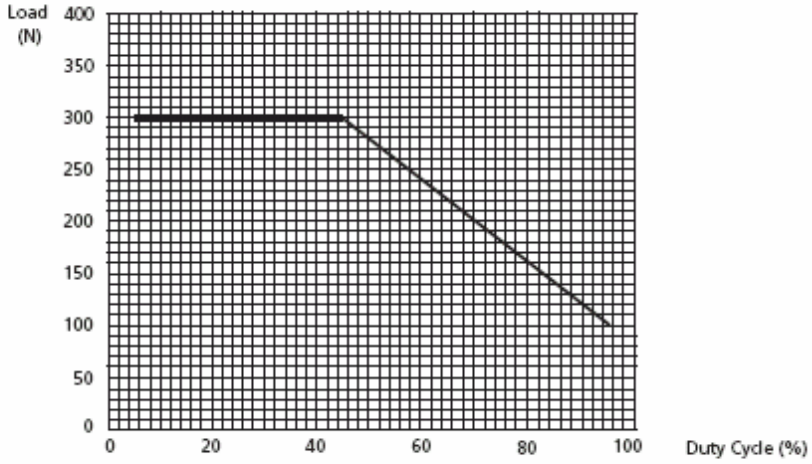
The above values are average values and made with a stable power supply and an ambient temperature of 20°C.

Graphs for Duty cycle

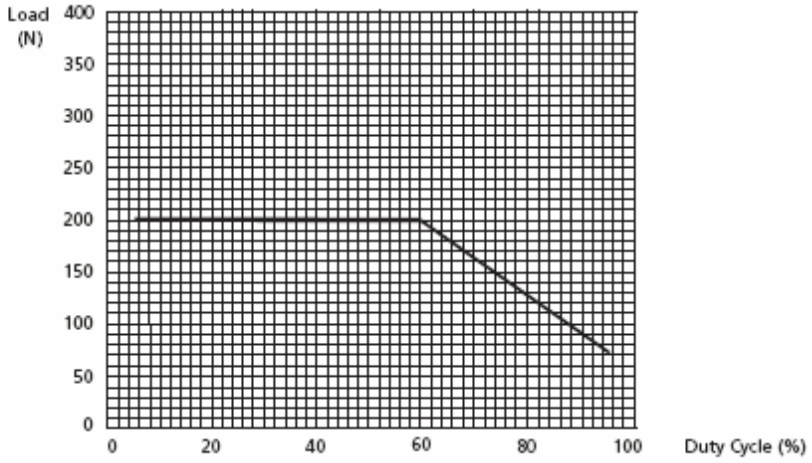
Duty Cycle versus Load - LA12 - 2 mm pitch (24 V)



Duty Cycle versus Load - LA12 - 4 mm pitch (24 V)



Duty Cycle versus Load - LA12 - 6 mm pitch (24 V)



Graphs for Duty Cycle

The graphs on the next page show which duty cycle can be expected compared to the load. The measurements for the graphs have been made with an ambient temperature of 20°C. If the actuator is used in a higher ambient temperature the duty cycle will decrease.

Test conditions for Duty Cycle.
The conditions of the tests carried out at LINAK A/S are as follows:

- Ambient temperature approx. 20°C.
- Actuator running at full stroke (between both end stop switches)
- Actuator running at max. recommended duty cycle
- Continuous operation 24 hours a day
- No vibrations
- Relatively clean enviroment (no extreme dust or dirt)

Please note that running the actuator in other conditions than the above mentioned may decrease or increase the duty cycle of the actuator. Therefore it is recommended that the customer always tests the actuator in the actual application, to ensure that the actuator fulfills the customers expectations.

Other information:

- It is possible to have other stroke lengths. From 20 mm - 130 mm in steps of 3,75 mm.
- Winding isolation class F (155 degrees).
- Typical noise level dB (A) 55-57 measuring method DS/EN ISO 3746, actuator not loaded.
- The maximum working voltage for the LA12 (24 V motor) is 28 V and 14 V for the LA12 (12 V motor)

LA12

Ordering example:

12 2 M 00 1 100 24 0 4

Cable:

Without reed:
0 = Black straight 0,75 m (2-core)
1 = Jack black straight 2,3 m (2-core)

With reed:
2 = 3-core stereo jack black straight 2,3 m
3 = 3-core black straight 0,75 m
4 = 4-core black straight 0,75 m
A = DESKLINE cable 2,06 m
C = DESKLINE cable 1,56 m
E = DESKLINE cable 0,66 m

With potentiometer:
0 = 5-core black straight 0,93 m

X = Special cable

IP version:
0 = IP51
1 = IP65
2 = IP66
8 = IP66 ATEX Zone 21 and 22

Motor:
12 = 12 V DC
24 = 24 V DC

Stroke length:
040, 070, 100, 130 = standard stroke length
XXX = special

Spindle type:
1 = 2 mm pitch
2 = 4 mm pitch
3 = 6 mm pitch

0 0
Not used

Positioning:

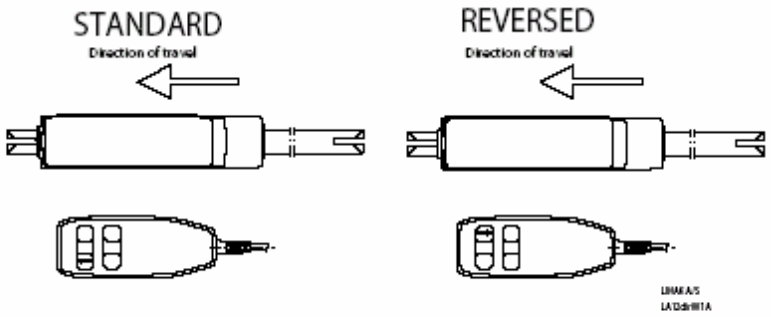
0 = none		
R = Reed switch	(3-core)	4 pulses per rev.
M = Reed switch	(4-core)	4 pulses per rev.
P = Potentiometer	(5-core)	
D = Reed switch for DESKLINE cable	-	2 pulses per rev.
T = Reed switch	(3-core)	10 pulses per rev.
E = Reed switch	(4-core)	10 pulses per rev.

Back fixture type:
1 = 01
2 = 02

Actuator type:
LA12

Note: Positioning / cable choice
When 0 choose cable 0 or 1
When R choose cable 2 or 3 (customer shall specify which cable)
When M choose cable 4
When Potentiometer choose cable 0 (with potentiometer MAX. Stroke length 100mm)
When D choose cable A, C or E
When T choose cable 2 eller 3
When E choose cable 4

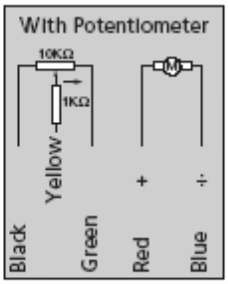
Beware of the direction of travel when ordering LA12 with Jack Plug, or DESKLINE® cable.



Connection diagrams:

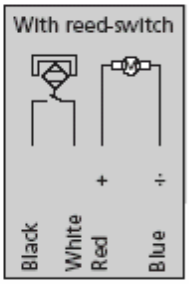
Actuator with potentiometer

- The built-in 10k. sliding potentiometer makes it possible to decide the position of the actuator. A 1k. dropping resistor has been mounted to protect the potentiometer against overload. Deflection is from 0k. + the dropping resistor's = 1k. (retracted position) to 10k. + 1k. (extended position). The absolute tolerance of the potentiometer is ± / 20%.
- The stroke length is limited to maximum 100 mm. With a stroke length of less than 100 mm the maximum potentiometer deflection will be correspondingly reduced: a stroke length of 70mm gives a deflection of 70/100 = 70 % deflection = 7k. + 1k. = 8k. (extended position).
- 930 mm straight 5-core cable: red, blue, yellow, green and black.
- The maximum permissible effect of the potentiometer is 0.1 W.
- The actuator, i.e. the motor and the potentiometer are connected as illustrated below:



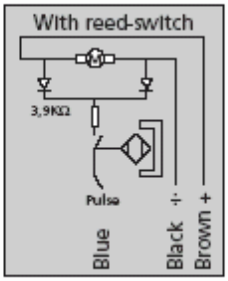
RED	BLUE	
+	-	Out going
-	+	In going

Actuator with potential free reed-switch



RED	BLUE	
+	-	Out going
-	+	In going

Actuator with reed



BROWN	BLACK	
+	-	Out going
-	+	In going

Please note that the voltage level of the feedback signal depends on the load of the actuator.

LA12 ATEX

Facts about LA12 ATEX version.

LINAK is able to supply LA12 ATEX for dust explosive atmospheres zones 21 and 22.

There are some differences between the "standard" LA12 and the "LA12 ATEX"

Standard LA12	LA12ATEX
Can be opened and modified by subsidiaries	Must be opened and modified by LINAK A/S only due to the fact that our quality systems have been approved according to ATEX.
Housing is made of plastic	Housing is made of plastic but coated with grey painting that is conductive. This is to prohibit static electricity.
Inner tube is made of plastic	Inner tube is made of stainless steel – the same as used on LA22. This to prohibit static electricity.
Installation dimension 245mm	Installation dimension 255mm due to the fact that the inner tube is made of stainless steel.

SUPPLEMENTARY FOR LA12 ATEX

Installation instructions.

The actuators model LA12 shall be used in a permanantly fixed installation.

Special conditions for safe use:

The permanently attached cable shall be terminated in a non-hazardous area or inside an enclosure that is certified under the type of explosion protection as described in EN 50014 clause 1.2.

The duty cycle of the actuator is limited to 6 minutes ON - 54 minutes OFF.

The actuator model LA12 is to be installed where it is protected from direct sunlight.

A2. MD03 Technical Documentation

MD03 - 50Volt 20Amp H Bridge Motor Drive

Overview

The MD03 is a medium power motor driver, designed to supply power beyond that of any of the low power single chip H-Bridges that exist. Main features are ease of use and flexibility. The motor's power is controlled by Pulse Width Modulation (PWM) of the H-Bridge at a frequency of 15khz (7.5khz before version 12).

The 15v MOSFET drive voltage is generated onboard with a charge pump, so the module requires only two supply voltages;

1. A standard 5V supply for the control logic, only 50mA maximum is required.
2. Motor voltage, anything from 5vdc to 50vdc to suit your motors

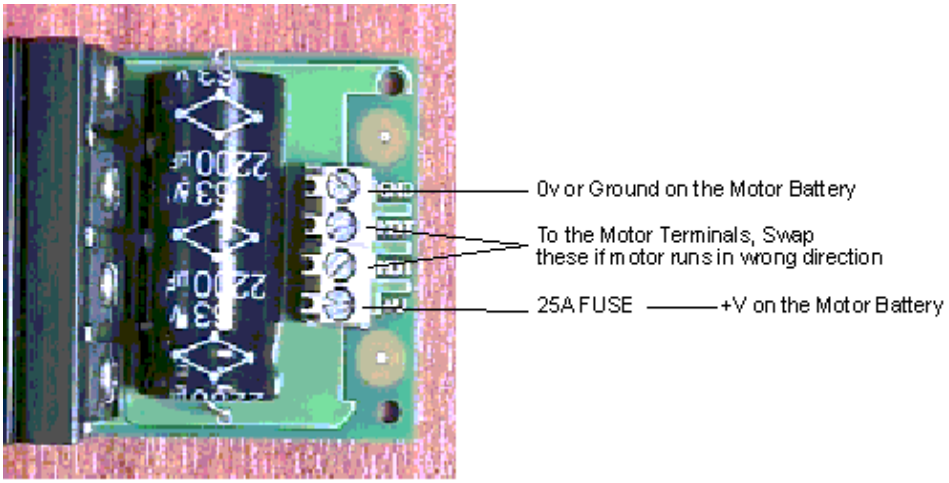
Control of the module can be any of;

a. I2C bus, up to 8 MD03 modules, switch selectable addresses.
b. 0v-2.5-5v analog input. 0v full reverse, 2.5v center stop, 5v full forward.
c. 0v-5v analog input with separate direction control
d. RC mode. Controlled directly from the RC receiver output.
e. PWM. A simple onboard filter means you can use a 0%-100% 20khz or greater instead of analog.

Motor connections

Note - There is no fuse on the PCB. You should provide a 25/30A fuse in line with the +v battery terminal.

Don't Ignore this, High currents can be dangerous!



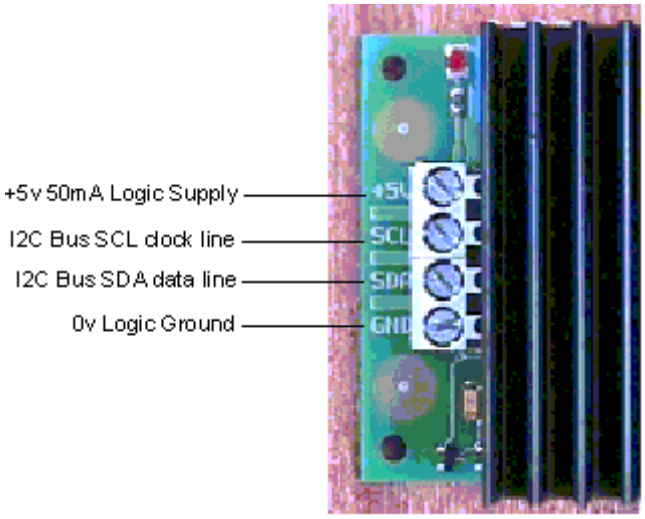
Be sure to use cable rated for at least 25/30A for the Battery, Fuse and Motor leads.

Motor Noise Suppression
Please note that using motors with the MD03 as with any other electronic device requires suppression of noise. This is easily achieved by the addition of a 10n snubbing capacitor across the motor. The capacitor should also be capable of handling a voltage of twice the drive voltage to the motor.

Control Connections

The connections for use with the I2C Bus are marked on the PCB.

See text below for details of the other modes.



Analog Mode - 0v-2.5v-5v
In this mode the motor is controlled by a 0v to 5v analog signal only on the SDA line. Pin SCL is unused and should be connected to either +5v or 0v.
0v is maximum reverse power
2.5v is the center stop position
5v is full forward power
The is a small (2.7%) dead band around 2.5v to provide a stable off position. Input impedance is 47k.

Analog Mode - 0v-5v
In this mode the motor is controlled by a 0v to 5v analog signal on the SDA line and direction on SCL.
0v is stop position
5v is full power
Pin SCL is the logic level (ttl) direction control. logic 0 for reverse direction and logic 1 for forward direction.

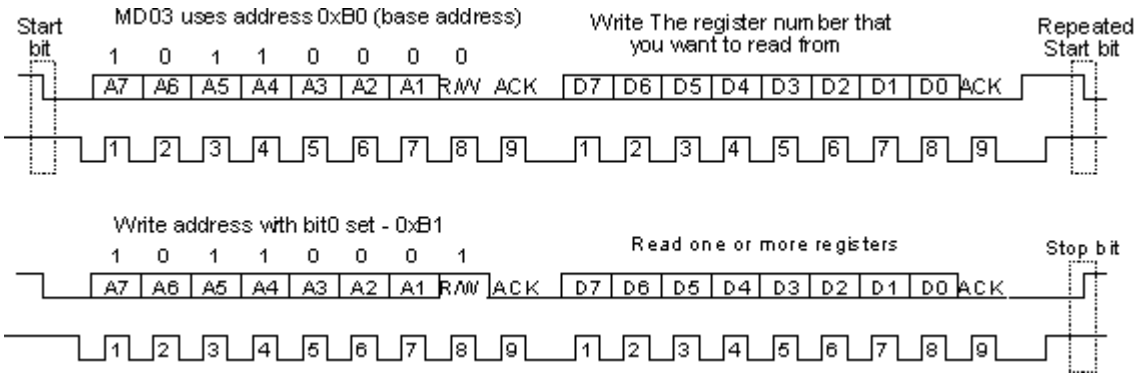
You may also use a PWM signal instead of an analog voltage on the SDA line. There is a simple resistor/capacitor filter on the module which will generate the analog voltage from the incoming PWM signal. your PWM signal should be 20khz or greater in frequency and ideally, come from a CMOS gate (0-5v) rather than ttl (0-3.5v ish). a 0% duty cycle will

represent 0v and a 100% duty cycle representing 5v. This applies to both the above analog modes.

RC Mode
This mode allows direct connection to standard model radio control receivers. Most receivers work from a 4.8v-6v battery pack and can be powered by 5v supply that powers the MD03 logic. The control pulse (Yellow) from the receiver should be connected to the SDA terminal. The SCL terminal is unused and should be connected to either +5v or 0v. Connect the receiver supply (Red) to +5v logic supply and the receiver 0v ground (Black) to the MD03 logic ground. The output from an RC receiver is a high pulse 1.5mS wide when the joystick is central. This varies down to 1.1mS and up to 1.9mS as the joystick is moved. This range can be shifted by the centering control by +/-100uS from 1mS-1.8mS to 1.2mS-2mS. The MD03 provides full control in the range 1.1mS to 1.9mS with 1.5mS being the center off position. There is a 7uS dead zone centered on 1.5mS for the off position. The Radio Transmitter centering control should be adjusted so that the motor is off when the joystick is released.

RC Mode With Timeout (from version 12)
An extra mode has been added and operates in much the same way as the normal RC control. The difference is the addition of a new timeout feature. If a RC pulse is not detected for a period in excess of 200ms, then the motor will be stop being driven until a valid RC signal is received

I2C Mode
I2C mode allows the MD03 to be connected to popular controllers such as the OOPic and BS2p, and a wide range of micro-controllers like PIC's, 8051's and H8's. There are instructions for connecting the MD03 to the OOPic here. If you're writing your own code in C, then we have some downloadable software to communicate with the CMPS03 compass module, SRF08 sonar and MD03 motor driver here.



I2C communication protocol with the MD03 module is the same as popular eeprom's such as the 24C04. To read one or more of the MD03 registers, first send a start bit, the module address (0XB0 is the base address) with the read/write bit low, then the register number you wish to read. This is followed by a repeated start and the module address again with the read/write bit high (0XB1). You now read one more registers. The MD03 has 8 registers numbered 0 to 7 as follows;

Register Address	Name	Read/Write	Description
0	Command	R/W	Write 01 Forwards - 02 Reverse (00 for instant stop - Rev9 firmware only)
1	Status	Read only	Acceleration, Temperature and Current Status
2	Speed	R/W	Motor Speed 0-255 (0x00 - 0xFF)
3	Acceleration	R/W	Motor Acceleration 0-255 (0x00 - 0xFF)
4	Temperature	Read only	Module temperature
5	Motor Current	Read only	Motor Current
6	Unused	Read only	Read as zero
7	Software Revision	Read only	Software Revision Number- Currently 9

Command Register
Controls the motor start, stop and direction. Write 1 to drive forwards, Write 2 to reverse, Write 0 to stop instantly (Rev 9 only). Be sure to have set the speed and acceleration before issuing these commands. Do not write 0 to the command register to stop on firmware revisions earlier than 9.
Note - On all revisions the way to stop the motor is the same as other speed changes, write the new speed to the speed register and re-issue the direction command. This will cause the motor to decelerate to a stop at the rate set by the acceleration register. On Rev 9 firmware, writing zero the command register will stop the motor instantly, bypassing the acceleration value.

Status Register
Shows the status of the MD03

Bit 7 (msb)	6	5	4	3	2	1	Bit 0 (lsb)
Busy	-	-	-	-	over-temperature limiter	over-current limiter	Acceleration in progress

Bit 0 is read as high when the drive is still accelerating the motor to the requested speed. It will be cleared when the requested speed is achieved or the over current or over temperature limiters are active.
Bit 1 set high indicates that the current through the motor has reached 20Amps and is being limited to that value. The Red LED will be illuminated when this happens.
Bit 2 set high indicates that the over temperature limiter is active. Above a preset

threshold, the motor current will be reduced in proportion to the MD03 temperature. The module can still be used but the motor power will be limited. The Red LED will be illuminated when this happens. It should be noted that a few minutes of running continuously at 20A will cause the heatsink to get hot - watch your fingers!
Bit 7 is the busy flag. It is set high when you issue a new command to the module. It is cleared very quickly and you are unlikely ever to see it set.

Speed Register
Sets the maximum speed that the motor will accelerate to. It is actually the 8-bit value sent to the modules PWM controller. Write a value of 0 to 243 (numbers from 243 to 255 are clamped to 243) . The larger the number, the more power is applied to the motor.

Acceleration Register
This sets the rate at which the motor accelerates or decelerates from where it is towards the new speed set by the speed register. Write a value of 0 to 255, the larger the number the longer the module will take to reach the new speed. Writing zero to the acceleration register will allow maximum acceleration of 0 to full in 0.187 seconds. This value actually controls a timer which steps the current motor speed towards the requested motor speed. It does this every (((acceleration register)*125)+768)uS. A value of zero gives 768uS/step or 243*768=186624uS (0.187s) to accelerate from 0 to full. A value of 255 is ((255*125)+768)*243=7932249uS or just under 8 seconds.

Temperature Register
This is the value used internally to limit the motor current if the module gets too hot. You do not need to read or do anything with it. It does not read degrees, in fact the number goes down as the temperature goes up! It is actually reading the forward voltage drop of a diode located under the 0.003R current sense resistor. The number drops by 1 count approx. every 1.42 degrees C.

Motor Current
This is the value used internally to limit the motor current to 20A. You do not need to read or do anything with it. The value is proportional to motor current, with a value of 186 representing the 20A limit.

Software Revision number
The revision number of the software in the modules PIC16F872 controller - currently 13 as of 1st August 2005.

Mode Switches
The 4 mode switches set the operating mode of the MD03. They are read once only when the module is powered up. You cannot switch modes while the unit is on.

Mode	Switch 1	Switch 2	Switch 3	Switch 4
I2C Bus - address 0xB0	On	On	On	On
I2C Bus - address	Off	On	On	On

0xB2				
I2C Bus - address 0Xb4	On	Off	On	On
I2C Bus - address 0xB6	Off	Off	On	On
I2C Bus - address 0xB8	On	On	Off	On
I2C Bus - address 0xBA	Off	On	Off	On
I2C Bus - address 0xBC	On	Off	Off	On
I2C Bus - address 0xBE	Off	Off	Off	On
0v - 2.5v – 5v Analog	On	On	On	Off
0v - 5v Analog + Direction	Off	On	On	Off
Radio Control	On	Off	On	Off

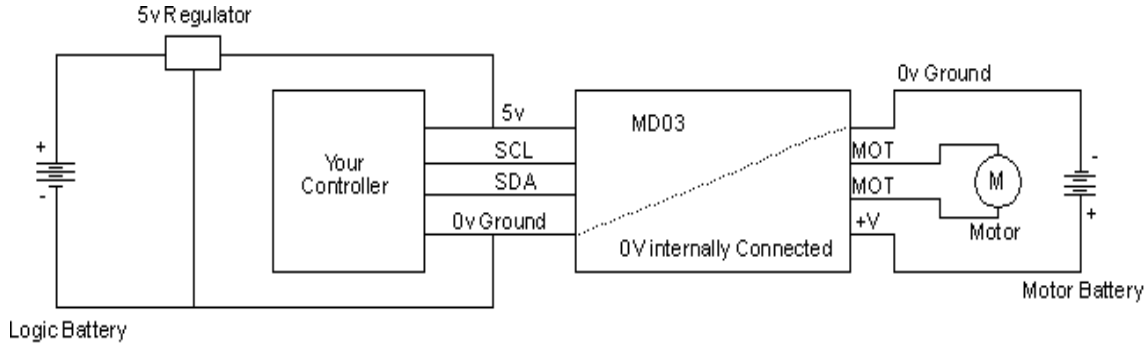
New Mode (from version 12)

Radio Control, timeout on	Off	Off	On	Off

All other combinations are invalid, the LED will blink and nothing else will happen. Note that I2C addresses are the upper 7 bits. Bit 0 the the read/write bit, so addresses 0xB0/0xB1 are write/read respectively to the same address.

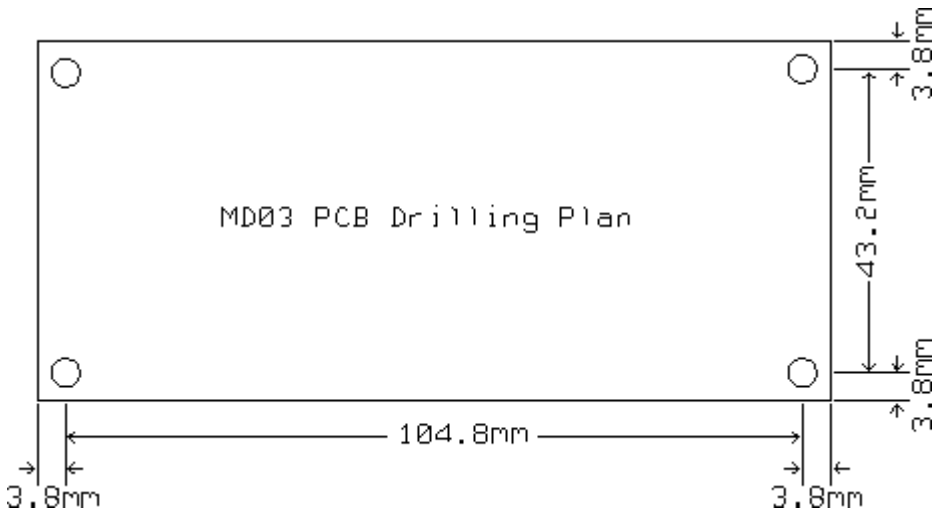
General Usage

The MD03 can handle high currents, and you will need to take a few precautions with wiring. It is very important that you do not allow motor current to flow in the logic ground path. Don't assume that just because the Battery, MD03 and Controller grounds are all together, that all is well. If at all possible, use two batteries, one for the logic and and the other for the motor power. Don't connect the battery grounds together, that is already done on the MD03. If you do, then you will only create a ground loop - and problems. The diagram below shows the general idea on keeping the logic and power sides electrically and physically separate from each other.



PCB Drill Plan

The following drawing shows the MD03 mounting hole positions.



Schematics

The MD03 schematics in GIF format md03sch1 md03sch2

CPU Code

For those who wish to, and are able to re-flash the PIC16F872, here is the Hex (Revision 13)

B. Desenvolupament de les equacions del sistema

B1. Càlcul de les posicions dels actuadors

A la Figura B.1 es pot observar la posició dels actuadors sobre el Mirall 1. Cada actuador es troba a una distància de 75mm respecte el punt central P.

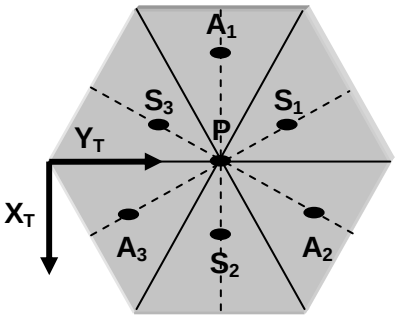


Fig. B.1. Posició dels actuadors sobre el Mirall 1.

Per tant, les coordenades de cadascun dels actuadors seran:

$A_1 = [-75, 150, Za_1]_T$ $A_2 = [37.5, 215, Za_2]_T$ $A_3 = [37.5, 85, Za_3]_T$

<u>Càlcul de la coordenada Za_1:</u>

Es defineix $\overrightarrow{V_{1'}}$ com el vector que va des del punt S_2 al punt central del segment, P:

$$\overrightarrow{V_{1'}} = (P - S_2) = \left[-64{,}0, \frac{Z_{S1} + Z_{S2} + Z_{S3}}{3} - Z_{S2} \right]_T$$

Cal notar que aquest vector té el mateix sentit i direcció que el vector que aniria del punt central P al punt A_1 on es troba aplicat l'actuador i, a més, aquesta propietat es compleix per a qualsevol possible orientació del mirall.

Si es modifica el mòdul del vector $\overrightarrow{V_{1'}}$ de forma que la seva component X_T sigui igual a 75 s'obté el vector $\overrightarrow{V_1}$:

$$\overrightarrow{V_1} = \frac{75}{64} \cdot \overrightarrow{V_{1'}} = \left[-75{,}0, \frac{75}{64} \cdot \left(\frac{Z_{S1} - 2 \cdot Z_{S2} + Z_{S3}}{3} \right) \right]_T$$

Finalment, si es suma aquest vector al punt P, s'obté la posició de l'actuador A_1 en funció de les coordenades (conegudes) dels sensors:

$$A_1 = \left[-75{,}150, \frac{75}{3 \cdot 64} \cdot \left(Z_{S1} - 2Z_{S2} + Z_{S3} \right) + \frac{Z_{S1} + Z_{S2} + Z_{S3}}{3} \right]_T$$

<u>Càlcul de la coordenada Za_2:</u>

Es defineix $\overrightarrow{V_{2'}}$ com el vector que va des del punt S_3 al punt central del segment, P:

$$\overrightarrow{V_{2'}} = \left(P - S_3 \right) = \left[32{,}55.4, \frac{Z_{S1} + Z_{S2} + Z_{S3}}{3} - Z_{S3} \right]_T$$

Cal notar que aquest vector té el mateix sentit i direcció que el vector que aniria del punt central P al punt A_2 on es troba aplicat l'actuador i, a més, aquesta propietat es compleix per a qualsevol possible orientació del mirall.

Si es modifica el mòdul del vector $\overrightarrow{V_{2'}}$ de forma que la seva component X_T sigui igual a 37.5 s'obté el vector $\overrightarrow{V_2}$:

$$\overrightarrow{V_2} = \frac{37.5}{32} \cdot \overrightarrow{V_{2'}} = \left[37.5{,}65, \frac{37.5}{32} \cdot \left(\frac{Z_{S1} + Z_{S2} - 2 \cdot Z_{S3}}{3} \right) \right]_T$$

Finalment, si es suma aquest vector al punt P, s'obté la posició de l'actuador A_2 en funció de les coordenades (conegudes) dels sensors:

$$A_2 = \left[37.5{,}215, \frac{75}{3 \cdot 64} \cdot \left(Z_{S1} + Z_{S2} - 2Z_{S3} \right) + \frac{Z_{S1} + Z_{S2} + Z_{S3}}{3} \right]_T$$

<u>Càlcul de la coordenada Za_3:</u>

Es defineix $\overrightarrow{V_{3'}}$ com el vector que va des del punt S_1 al punt central del segment, P:

$$\overrightarrow{V_{3'}} = \left(P - S_1 \right) = \left[32{,}-55.4, \frac{Z_{S1} + Z_{S2} + Z_{S3}}{3} - Z_{S1} \right]_T$$

Cal notar que aquest vector té el mateix sentit i direcció que el vector que aniria del punt central P al punt A_3 on es troba aplicat l'actuador i, a més, aquesta propietat es compleix per a qualsevol possible orientació del mirall.

Si es modifica el mòdul del vector $\overrightarrow{V_{3'}}$ de forma que la seva component X_T sigui igual a 37.5 s'obté el vector $\overrightarrow{V_3}$:

$$\overrightarrow{V_3} = \frac{37.5}{32} \cdot \overrightarrow{V_{3'}} = \left[37.5, -65, \frac{37.5}{32} \cdot \left(\frac{-2 \cdot Z_{S1} + Z_{S2} + Z_{S3}}{3} \right) \right]_T$$

Finalment, si es suma aquest vector al punt P, s'obté la posició de l'actuador A_3 en funció de les coordenades (conegudes) dels sensors:

$$A_3 = \left[37.5, 85, \frac{75}{3 \cdot 64} \cdot \left(-2Z_{S1} + Z_{S2} + Z_{S3} \right) + \frac{Z_{S1} + Z_{S2} + Z_{S3}}{3} \right]_T$$

B2. Càlcul de les coordenades Zs_{id}

Sigui $\overrightarrow{Nd}$ el vector normal al pla desitjat i Pd el punt central del segment en el pla desitjat:

$$\overrightarrow{Nd} = (xd, yd, zd)_T - Pd = \left[xd, yd - 150, (L - TL) - \frac{Z_{S1d} + Z_{S2d} + Z_{S3d}}{3} \right]_T$$

$$Pd = (X_{Pd}, Y_{Pd}, Z_{Pd}) = \left[0, 150, \frac{Z_{S1d} + Z_{S2d} + Z_{S3d}}{3} \right]_T$$

Cal notar que xd, yd i zd són valors coneguts, ja que arriben directament del PC.

Llavors es defineix l'equació del pla desitjat de la següent manera:

$$K = xd \cdot x + (yd - 150) \cdot y + \left(L - TL - \frac{Z_{S1d} + Z_{S2d} + Z_{S3d}}{3} \right) \cdot z$$

On (x,y,z) són les coordenades en la Base T de qualsevol punt pertanyent al pla desitjat. Així doncs, K es troba introduint a l'equació el punt Pd com un punt d'aquest pla desitjat.

$$K = xd \cdot 0 + (yd - 150) \cdot 150 + \left(L - TL - \frac{Z_{S1d} + Z_{S2d} + Z_{S3d}}{3} \right) \cdot \left(\frac{Z_{S1d} + Z_{S2d} + Z_{S3d}}{3} \right)$$

Simplificant,

$$K = 150 \cdot yd - 22500 + (L - TL) \cdot Z_{Pd} - (Z_{Pd})^2$$

Per tant, l'equació del pla desitjat queda de la següent forma:

$$150 \cdot yd - 22500 + (L - TL) \cdot Z_{Pd} - (Z_{Pd})^2 = xd \cdot x + (yd - 150) \cdot y + (L - TL - Z_{Pd}) \cdot z$$

<u>Càlcul de Zs_{1d}:</u>

Si es possa a l'equació la posició desitjada del sensor 1 (S1) es té:

$$150 \cdot yd - 22500 + (L - TL) \cdot Z_{Pd} - (Z_{Pd})^2 = -32 \cdot xd + (yd - 150) \cdot 205.4 + (L - TL - Z_{Pd}) \cdot Z_{S1d}$$

Simplificant,

$$Z_{S1d} = \frac{8310 - 55.4 \cdot yd + 32 \cdot xd}{(L - TL) - Z_{Pd}} + Z_{Pd}$$

<u>Càlcul de Zs_{2d}:</u>

Si es possa a l'equació la posició desitjada del sensor 2 (S2) es té:

$$150 \cdot yd - 22500 + (L - TL) \cdot Z_{Pd} - (Z_{Pd})^2 = 64 \cdot xd + (yd - 150) \cdot 150 + (L - TL - Z_{Pd}) \cdot Z_{S2d}$$

Simplificant,

$$Z_{S2d} = \frac{-64 \cdot xd}{(L - TL) - Z_{Pd}} + Z_{Pd}$$

<u>Càlcul de Zs_{3d}:</u>

Zs_{3d} es pot trobar a partir de Zs_{1d}, Zs_{2d} i Z_{Pd} de la següent manera:

$Z_{Pd} = \frac{Z_{S1d} + Z_{S2d} + Z_{S3d}}{3}$. Simplificant,

$$Z_{S3d} = 3 \cdot Z_{Pd} - Z_{S1d} - Z_{S2d}$$

C. Implementació del tractament d'imatge al PC

Per a dur a terme la calibració i el filtrat de les dades mostrejades per la càmera s'ha creat un Form de Visual Basic. Aquest Form s'utilitza també com a panell de control per a tot el procés del tractament d'imatge.

A la Figura C.1 es pot veure aquest panell de control.

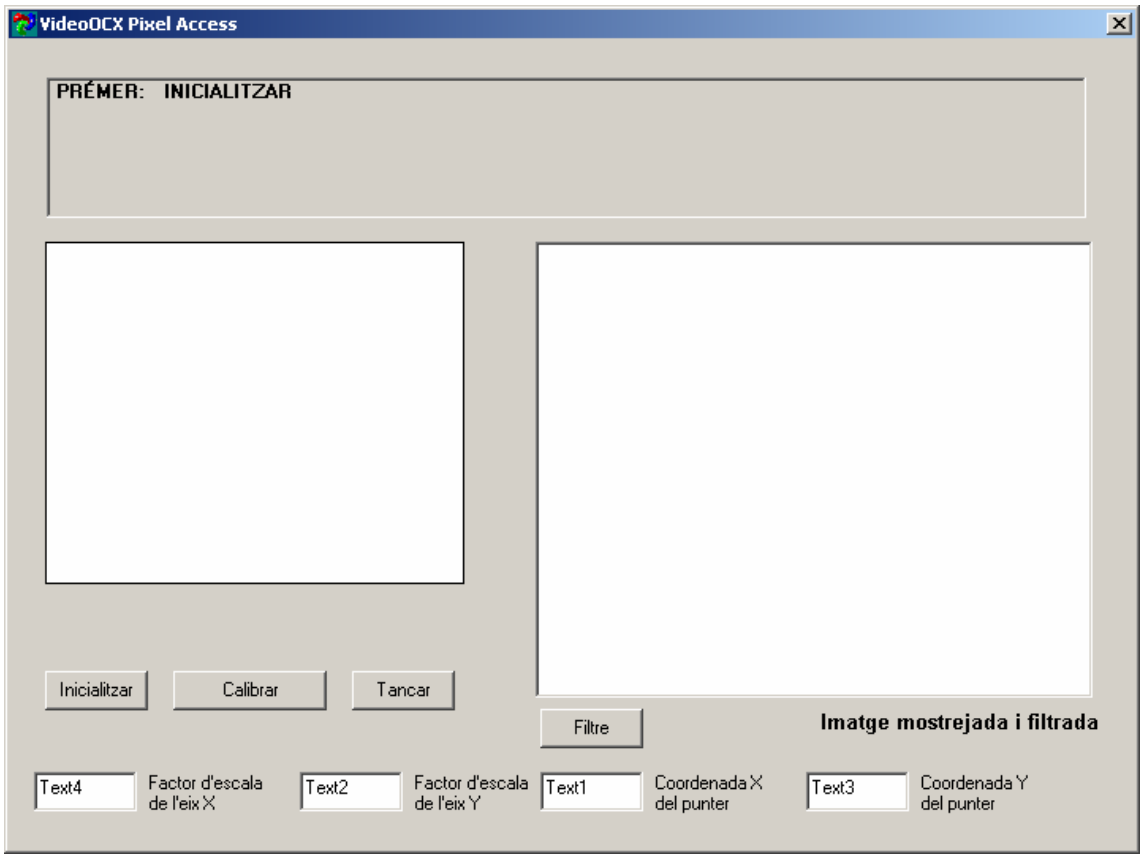


Fig. C.1. Panell de control del tractament d'imatge (no inicialitzat).

La implementació del programa associat a aquest panell de control es dóna a continuació, amb els comentaris necessàris per entendre la funcionalitat de cada part del codi.

```
Private Sub Calibrar_Click()
   Dim eixX(1) As Double
   Dim eixY(1) As Double
```

```
Dim alfa As Double
Dim theta As Double
Dim matrix As Variant
Dim x As Integer
Dim y As Integer
Dim i As Integer
Dim SXCI, SYCI As Single

If k = 0 Then
    Label4.Caption = " PAS 1:   APUNTAR EL CENTRE DEL TELESCOPI (CENTRE DE LA
BASE T)"
ElseIf k = 1 Then
    Label4.Caption = " PAS 2:   APUNTAR L'EIX X DE LA BASE T"
ElseIf k = 2 Then
    Label4.Caption = " PAS 3:   APUNTAR L'EIX Y DE LA BASE T"
Else
End If
Do
    ' Captura la imatge (Mostreig)
    VideoOCX1.Capture (m_Image)

    ' Obté la matriu de píxels
    matrix = VideoOCX1.GetMatrix(m_Image)

    ' Filtra la imatge
    SXCI = 0
    SYCI = 0
    i = 0
    For y = 0 To VideoOCX1.GetHeight - 1
        For x = 0 To VideoOCX1.GetWidth - 1
            If matrix(x, y, 1) > 254 Then
                SXCI = SXCI + x
                SYCI = SYCI + y
                i = i + 1
            Else
            End If
        Next x
    Next y

    ' Busca les coordenades del punter i calcula les matrius de canvi de base
```

```
' Matrius de rotació R, translació B i d'escalat E
If i >= 1 Then
   SXCI = SXCI / i     ' Coordenada x del centre del punter en la base CI
   SYCI = SYCI / i     ' Coordenada y del centre del punter en la base CI
   x_alternatiu = SXCI
   y_alternatiu = SYCI
   If k = 0 Then
      B(0) = SXCI
      B(1) = SYCI
   ElseIf k = 1 Then
      eixX(0) = SXCI - B(0)
      eixX(1) = SYCI - B(1)
      alfa = Atn(eixX(1) / eixX(0))
      If eixX(0) < 0 Then
         theta = PI + alfa
      Else
         theta = alfa
      End If
      R(0) = CSng(Cos(theta))
      R(1) = CSng(Sin(theta))
      E(0) = 200 / (eixX(0) * R(0) + eixX(1) * R(1))
      Text4.Text = E(0)
   ElseIf k = 2 Then
      eixY(0) = SXCI - B(0)
      eixY(1) = SYCI - B(1)
      E(1) = 200 / (eixY(0) * (-R(1)) + eixY(1) * R(0))
      Text2.Text = E(1)
   Else
      Label4.Caption = " ERROR DE CALIBRACIÓ"
      k = 0
   End If
   Exit Do
Else
End If

' Allibera la matriu de píxels i actualitza la imatge després del filtre
VideoOCX1.ReleaseMatrixToImageHandle m_Image

' Treu per pantalla la imatge
Picture1.Picture = VideoOCX1.ToPicture(m_Image)
```

```
        ' Retorna el control al OS per a que executi altres events
        DoEvents
    Loop
    k = k + 1
    If k <= 2 Then
        Label4.Caption = " PRÉMER:   CALIBRAR"
    Else
        Label4.Caption = " PRÉMER:   FILTRE"
        k = 0
    End If
End Sub
```

```
Private Sub Filtre_Click()
    Dim matrix As Variant
    Dim x As Integer
    Dim y As Integer
    Dim i, j As Integer
    Dim value1, value2 As Byte
    Dim SXCI, SYCI As Single
    Dim l, TL As Integer
    Dim xxd, yyd As Single
    Dim XD, YD, ZD As Integer
    Dim iniseg, finseg, espera As Single

    l = 1820        ' Paràmetre: Altura de la càmera al sostre en mm
    TL = -80        ' Paràmetre: Altura de la càmera al telescopi (base T) en mm
    ZD = l - TL     ' Coordenada Z del puter làser en la base T (només cal
                    ' enviar-la una vegada)
    j = 1           ' Ordena els missatges seqüencialment: ZD-XD-YD

    Do
        ' Captura la imatge (Mostreig)
        VideoOCX1.Capture (m_Image)

        ' Obté la matriu de píxels
        matrix = VideoOCX1.GetMatrix(m_Image)

        ' Filtra la imatge
        SXCI = 0
```

```
SYCI = 0
i = 0
For y = 0 To VideoOCX1.GetHeight - 1
   For x = 0 To VideoOCX1.GetWidth - 1
      matrix(x, y, 0) = 0
      If matrix(x, y, 1) > 254 Then
         SXCI = SXCI + x
         SYCI = SYCI + y
         i = i + 1
      Else
         matrix(x, y, 1) = 0
      End If
      If matrix(x, y, 2) > 254 Then
      Else
         matrix(x, y, 2) = 0
      End If
   Next x
Next y

' Marca el centre de la imatge
For y = 123 To 163
   For x = 155 To 195
      If (x = 175 Or y = 143) Then
         matrix(x, y, 0) = 255
      End If
   Next x
Next y

' Busca les coordenades del punter làser (punt verd)
If i >= 1 Then
   SXCI = SXCI / i     ' Coordenada x del centre del punter en la base CI
   SYCI = SYCI / i     ' Coordenada y del centre del punter en la base CI
Else
   Text1.Text = Empty
   Text3.Text = Empty
End If

' Transforma les coordenades del punter de la base CI (2D) --> base T (3D)
xxd = E(0) * (R(0) * (SXCI - B(0)) + R(1) * (SYCI - B(1)))
yyd = E(1) * ((-R(1)) * (SXCI - B(0)) + R(0) * (SYCI - B(1)))
```

```
XD = CInt(xxd)
YD = CInt(yyd)

' Envia les coordenades reals del punter pel RS232 (port serie) al bus CAN
' Cada missatge envia 3 char (3 bytes)
If i >= 1 Then
   Text1.Text = XD
   Text3.Text = YD
   If j = 1 Then
      If ZD >= 0 Then
         value1 = ZD \ 256
         value2 = ZD Mod 256
         ' ! Identificador de la coord. Z
         MSComm1.Output = "!" & (Chr$(value1)) & (Chr$(value2))
      Else
         MsgBox ("ERROR: valor de ZD incorrecte")
      End If
      j = 2
   ElseIf j = 2 Then
      If XD >= 0 Then
         value1 = XD \ 256
         value2 = XD Mod 256
         ' % Identificador de la coord. X positiva
         MSComm1.Output = "%" & (Chr$(value1)) & (Chr$(value2))
      Else
         value1 = Abs(XD) \ 256
         value2 = Abs(XD) Mod 256
         ' & Identificador de la coord. X negativa
         MSComm1.Output = "&" & (Chr$(value1)) & (Chr$(value2))
      End If
      j = 3
   ElseIf j = 3 Then
      If YD >= 0 Then
         value1 = YD \ 256
         value2 = YD Mod 256
         ' $ Identificador de la coord. Y
         MSComm1.Output = "$" & (Chr$(value1)) & (Chr$(value2))
      Else
         value1 = Abs(YD) \ 256
         value2 = Abs(YD) Mod 256
```

```
            ' ( Identificador de la coord. Y
            MSComm1.Output = "(" & (Chr$(value1)) & (Chr$(value2))
          End If
          j = 1
        Else
          j = 1
        End If
      Else
      End If

      ' Allibera la matriu de píxels i actualitza la imatge després del filtre
      VideoOCX1.ReleaseMatrixToImageHandle m_Image

      ' Treu per pantalla la imatge
      Picture1.Picture = VideoOCX1.ToPicture(m_Image)

      ' Introdueix una espera de 300 ms. per a no saturar el bus CAN
      espera = 0.3
      iniseg = Timer
      finseg = iniseg + espera
      While finseg > Timer
        DoEvents   ' Retorna el control al OS per a que executi altres events
      Wend
    Loop
End Sub
```

```
Private Sub Form_Load()
  k = 0
  x_alternatiu = 0
  y_alternatiu = 0
  If MSComm1.PortOpen = True Then
    MSComm1.PortOpen = False
    MSComm1.PortOpen = True
  Else
    MSComm1.PortOpen = True
  End If
End Sub
```

```
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
  End
```

```
End Sub
_____________________________________________________________________
Private Sub Inicialitzar_Click()
   VideoOCX1.SetErrorMessages (False)
   If Not VideoOCX1.Init Then
      MsgBox VideoOCX1.GetLastErrorString
   Else
      m_Image = VideoOCX1.GetColorImageHandle
      VideoOCX1.SetPreview (True)
      VideoOCX1.Start
   End If
   Label4.Caption = " PRÉMER:   CALIBRAR"
End Sub
_____________________________________________________________________
Private Sub Tancar_Click()
   VideoOCX1.Stop
   VideoOCX1.Close
End Sub
```

Per tant, quan el programa està treballant, el panell de control té l'aspecte mostrat per la Figura C.2.

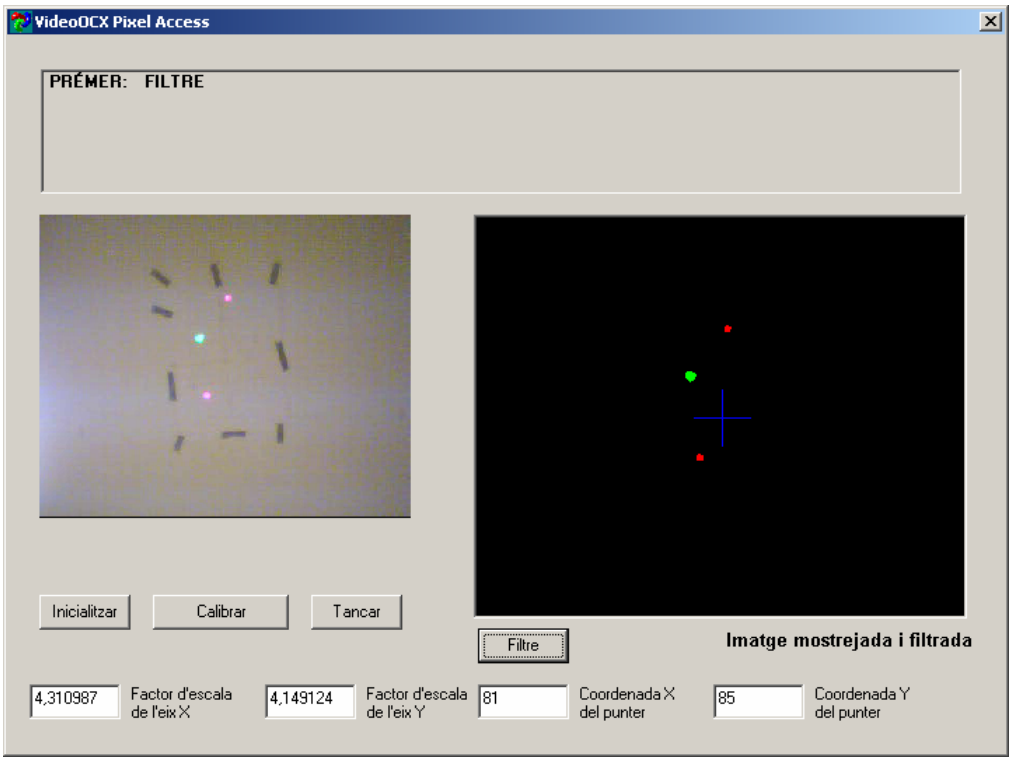


Fig. C.2. Panell de control del tractament d'imatge (inicialitzat).

D. Implementació del codi dels PICs

Tant els nou microcontroladors que calculen la llei de control com la *gateway* han estat programats en llenguatge C, utilitzant el PCWH Compiler facilitat per CCS Inc. Aquest compilador ja incorpora llibreries específiques per al PIC18F458, així com per al control del bus CAN.

En aquest annex es donen els programes en llenguatge C per a cadascun dels PICs. Els comentaris introduits al codi es suposen prou explícits per a no haver d'explicar amb més detall les particularitats del programa.

D1. Codi de la *gateway*

```
/* GATEWAY.
  Pasa les dades de la posicó del punter verd (Xd, Yd, Zd) del RS232 (port serie) al bus CAN
*/
#include <ccscana.c>
#include <STDLIB.H>

/* Identificadors dels nodes, per ordre de prioritat (de + a -) */
#define GATEWAY_Xd 0x001            //Identificador per Xd positiva
#define GATEWAY_mX 0x002            //Identificador per Xd negativa
#define GATEWAY_Yd 0x003            //Identificador per Yd positiva
#define GATEWAY_mY 0x004            //Identificador per Yd negativa
#define GATEWAY_Zd 0x005            //Identificador per Zd (sempre positiva)
#define PLACA_1_S1 0x006
#define PLACA_1_S2 0x007
#define PLACA_1_S3 0x008            /*************************************************          */
#define PLACA_2_S1 0x009            /* Identificadors dels sensors S1,S2 i  S3  */
#define PLACA_2_S2 0x00A            /* per a cadascun dels tres miralls.        */
#define PLACA_2_S3 0x00B            /* PLACA_i_Sj ---> Correspon al sensor      */
#define PLACA_3_S1 0x00C            /*                    j del mirall i        */
#define PLACA_3_S2 0x00D            /****************************************
        ***********/
#define PLACA_3_S3 0x00E

void main() {
  int value_id;          //variable que identifica els missatges del RS232
  int dades[2];          //variable (Xd, Yd o Zd) que s'envia pel bus CAN
```

```
int i;                          //variable auxiliar

/* Inicialització del bus CAN */
can_init();                                 //inicialització del bus CAN de la placa
can_putd(0x100,0,0,1,TRUE,FALSE);   //activació de tots els micros de la placa
delay_ms(1000);

/* Inicialització de variables */
i = 1;
value_id = 10;

/* Inici del bucle infinit */
while (TRUE) {

  /* Identificació i obtenció del missatge del PC */
  if (kbhit()) {
    value_id = getc();              //getc() → obtenció del caràcter del RS-232
    dades[0] = getc();
    dades[1] = getc();

    /* Escriptura del missatge al bus CAN */
    switch (value_id) {
      case '!':
        can_putd(GATEWAY_Zd, dades, 2, 1, TRUE, FALSE);     //can_putd() → escriptura
        value_id = 10;                                      //al bus CAN
        break;
      case '%':
        can_putd(GATEWAY_Xd, dades, 2, 1, TRUE, FALSE);
        value_id = 10;
        break;
      case '&':
        can_putd(GATEWAY_mX, dades, 2, 1, TRUE, FALSE);
        value_id = 10;
        break;
      case '$':
        can_putd(GATEWAY_Yd, dades, 2, 1, TRUE, FALSE);
        value_id = 10;
        break;
      case '(':
        can_putd(GATEWAY_mY, dades, 2, 1, TRUE, FALSE);
```

```
            value_id = 10;
            break;
          default:
            break;
        }
      }
    }
  }
```

D2. Codi del Node 1 del Mirall 1

```
/* PLACA (mirall_1).
   Calcula les posicions desitjades de TOTS els sensors del mirall_1.
   Calcula la posició desitjada de l'actuador A1.
   Mou l'actuador A1, llegeix directament del sensor S1.
*/
#include <ccscana.c>
#include <STDLIB.H>

/* Identificadors dels nodes, per ordre de prioritat (de + a -) */
#define GATEWAY_Xd 0x001            //Identificador per Xd positiva
#define GATEWAY_mX 0x002            //Identificador per Xd negativa
#define GATEWAY_Yd 0x003            //Identificador per Yd positiva
#define GATEWAY_mY 0x004            //Identificador per Yd negativa
#define GATEWAY_Zd 0x005            //Identificador per Zd (sempre positiva)
#define PLACA_1_S1 0x006
#define PLACA_1_S2 0x007
#define PLACA_1_S3 0x008            /*************************************************      */
#define PLACA_2_S1 0x009            /* Identificadors dels sensors S1,S2 i  S3  */
#define PLACA_2_S2 0x00A            /* per a cadascun dels tres miralls.        */
#define PLACA_2_S3 0x00B            /* PLACA_i_Sj ---> Correspon al sensor      */
#define PLACA_3_S1 0x00C            /*                    j del mirall i        */
#define PLACA_3_S2 0x00D            /*************************************
          ***********/
#define PLACA_3_S3 0x00E

void main() {
  signed int16 Xd, Yd, Zd;        //posició del punter verd en la base T
  float Zs1df, Zs2df, Zs3df;      //variables auxiliars
  float Za1df, Za2df, Za3df;      //variables auxiliars
```

```
float opt;                      //variable auxiliar
int S1, S2, S3;         //lectura dels sensors S1, S2 i S3
float Zpd;                      //altura desitjada del punt P (centre del mirall)
float Zs1d, Zs2d, Zs3d;         //altura desitjada dels sensors S1, S2 i S3
float Zs1, Zs2, Zs3;            //altura instantània dels sensors S1, S2 i S3
float Za1d, Za2d, Za3d;         //altura desitjada dels actuadors A1, A2 i A3
float Za1, Za2, Za3;            //altura instantània dels actuadors A1, A2 i A3
int i;                          //variable auxiliar
int j;                          //variable pel control del flux de dades dels sensors
float er;                       //error
int x;                          //entrada a la planta (etapa de potència + motor A1)
float SERR;             //variables auxiliars pel càlcul de la llei de control
float KP, KI;                   //constant proporcional i constant integral
int dades[2];                   //variable que emmagatzema els missatges del bus CAN
int sensor[1];                  //variable que emmagatzema els missatges del bus CAN
int rx_len,rxstat;              //variables auxiliars
int32 rx_id;                    //variables auxiliars

/* Inicialització del convertidor A/D */
setup_adc(ADC_CLOCK_INTERNAL);
setup_adc_ports(ALL_ANALOG);
set_adc_channel(1);                             //sel·lecció del pin RA1/AN1 "analog input"

/* Inicialització del PWM */
setup_timer_2(T2_DIV_BY_1, 255, 1);             /* Sel·lecció de:
                                                Prescaler = 1
                                                Period = 255
                                                Postscale = 1 */
setup_ccp1(CCP_PWM);            //inicialització del "PWM module"

/* Inicialització del bus CAN */
can_init();
can_putd(0x100,0,0,1,TRUE,FALSE);
delay_ms(1000);

/* Inicialització de variables auxiliars */
SERR = 0.0;             //inicialització de l'error acumulat
KP = 1;                 //constant proporcional del PI
KI = 0.02;              //constant integral del PI
```

```
i = 0;
j = 0;

Zpd = 18.0;                //valor inicial de la iteració per a trobar el pla desitjat

/* Inicialització de les variables Xd, Yd, Zd */
Xd = 0;
Yd = 0;
Zd = 1899;

/* Inicialització de les posicions dels sensors */
S1 = 255;
S2 = 255;
S3 = 255;

/* Espera a que s'inicialitzi tot el sistema i arribi el primer missatge amb
les coordenades del punter làser */
while (i == 0) {
  if (can_kbhit()) {
    if (can_getd(rx_id,&dades[0],rx_len,rxstat)) {
      if (rx_id == GATEWAY_Xd) {
        i = 1;
      }
      else if (rx_id == GATEWAY_Yd) {
        i = 1;
      }
      else if (rx_id == GATEWAY_Zd) {
        i = 1;
      }
      else if (rx_id == GATEWAY_mX) {
        i = 1;
      }
      else if (rx_id == GATEWAY_mY) {
        i = 1;
      }
      else {
      }
    }
  }
}
```

```
/* Inici del bucle infinit */
while (TRUE) {

  /* Obtenció dels missatges CAN (si existeixen)
  Coordenades desitjades (Xd,Yd,Zd) i sensors (S2 i S3) */
  if (can_kbhit()) {
    if (can_getd(rx_id,&dades[0],rx_len,rxstat)) {
      if (rx_id == GATEWAY_Xd) {
        Xd = dades[0] * 256;            //adquisició del byte més significatiu
        Xd = Xd + dades[1];             //adquisició del byte menys significatiu
      }
      else if (rx_id == GATEWAY_Yd) {
        Yd = dades[0] * 256;
        Yd = Yd + dades[1];
      }
      else if (rx_id == GATEWAY_Zd) {
        Zd = dades[0] * 256;
        Zd = Zd + dades[1];
      }
      else if (rx_id == GATEWAY_mX) {
        Xd = dades[0] * (-256);
        Xd = Xd - dades[1];
      }
      else if (rx_id == GATEWAY_mY) {
        Yd = dades[0] * (-256);
        Yd = Yd - dades[1];
      }
      else if (rx_id == PLACA_1_S2) {
        S2 = dades[0];
      }
      else if (rx_id == PLACA_1_S3) {
        S3 = dades[0];
      }
      else {
      }
    }
  }
  else {
```

```
}

/* Lectura del sensor S1 */
S1 = read_adc();
sensor[0] = S1;

/* Escriptura de la posició del sensor S1 al bus CAN (cada 10 voltes del bucle general per
 a no col·lapsar el bus */
if (j >= 10) {
  can_putd(PLACA_1_S1, sensor, 1, 1, TRUE, FALSE);
  j = 0;
}
else {
  j = j+1;
}

/* Càlcul de les posicions actuals */
Zs1 = 0.04706*S1+10;
Zs2 = 0.04706*S2+10;
Zs3 = 0.04706*S3+10;

Za1 = (((75*(Zs1-2*Zs2+Zs3)/3)/64)+(Zs1+Zs2+Zs3)/3);
Za2 = (((37.5*(Zs1+Zs2-2*Zs3)/3)/32)+(Zs1+Zs2+Zs3)/3);
Za3 = (((37.5*(Zs2-2*Zs1+Zs3)/3)/32)+(Zs1+Zs2+Zs3)/3);

/* Càlcul d'un possible pla desitjat */
Zs1df = ((8310.0-(55.4*(float)Yd)+(32.0*(float)Xd))/((float)Zd-Zpd))+Zpd;
Zs2df = ((-64.0*(float)Xd)/((float)Zd-Zpd))+Zpd;
Zs3df = 3.0*Zpd-Zs1df-Zs2df;

Za1df = (((75*(Zs1df-2*Zs2df+Zs3df)/3)/64)+(Zs1df+Zs2df+Zs3df)/3);
Za2df = (((37.5*(Zs1df+Zs2df-2*Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);
Za3df = (((37.5*(Zs2df-2*Zs1df+Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);

/* Cerca del pla desitjat per aproximacions succesives. Es buscarà
el pla mes baix possible */
if ((Za1df >= Za2df)&&(Za2df >= Za3df)) {
  opt = 14.0 - Za3df;
}
else if ((Za1df >= Za3df)&&(Za3df >= Za2df)) {
```

```
  opt = 14.0 - Za2df;
}
else if ((Za2df >= Za1df)&&(Za1df >= Za3df)) {
  opt = 14.0 - Za3df;
}
else if ((Za2df >= Za3df)&&(Za3df >= Za1df)) {
  opt = 14.0 - Za1df;
}
else if ((Za3df >= Za1df)&&(Za1df >= Za2df)) {
  opt = 14.0 - Za2df;
}
else {
  opt = 14.0 - Za1df;
}

Zpd = Zpd + opt;
Zs1df = ((8310.0-(55.4*(float)Yd)+(32.0*(float)Xd))/((float)Zd-Zpd))+Zpd;
Zs2df = ((-64.0*(float)Xd)/((float)Zd-Zpd))+Zpd;
Zs3df = 3.0*Zpd-Zs1df-Zs2df;

Za1df = (((75*(Zs1df-2*Zs2df+Zs3df)/3)/64)+(Zs1df+Zs2df+Zs3df)/3);
//Za2df = (((37.5*(Zs1df+Zs2df-2*Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);
//Za3df = (((37.5*(Zs2df-2*Zs1df+Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);

/* Comprovació de que les altures desitjades dels sensors estan dins del
rang permés: Zsidf<=21.96 */
i = 0;
if (Zs1df > 21.96) {
  i = 1;
}
else if (Zs2df > 21.96) {
  i = 2;
}
else if (Zs3df > 21.96) {
  i = 3;
}
else {
}

Za1d = Za1df;
```

```
    //Za2d = Za2df;
    //Za3d = Za3df;

    /* Error er sobre l'actuador A1 */
    er = Za1d - Za1;            /* Error de posició de l'actuador A1
                                Si er > 0 el motor ha de pujar (x -> 255)
                                Si er < 0 el motor ha de baixar (x -> 0) */
    /* Controlador PI */
    SERR = SERR + er;
    er = KP * er + KI * SERR;

    /* Saturació de l'error er */
    if (er < -22.74) {
      er = -22.74;
    }
    if (er > 22.74) {
      er = 22.74;
    }

    /* Càlcul de l'entrada x a la planta */
    if (er >= 0.0) {
      er = 5.1012 * er + 139.0;     //escalat de er del rang 0..22.74 al 139..255
    }
    else {
      er = 5.0131 * er + 114.0;     //escalat de er del rang -22.74..0 al 0..114
    }

    if (i==0) {
      x = (int8)er;
    }
    else {
      x = 126;                      //per a x pertanyents al rang 115..138 el motor
      SERR = 0;                     //no es mou (zona morta de les etapes de potència)
    }
    set_pwm1_duty(x);               //escriptura del "Duty cicle" (temps en alta)
                                    //(actuador A1 del Mirall_1)
  }
}
```

D3. Codi del Node 2 del Mirall 1

```
/* PLACA (mirall_1).
   Calcula les posicions desitjades de TOTS els sensors del mirall_1.
   Calcula la posició desitjada de l'actuador A2.
   Mou l'actuador A2, llegeix directament del sensor S2.
*/
#include <ccscana.c>
#include <STDLIB.H>

/* Identificadors dels nodes, per ordre de prioritat (de + a -) */
#define GATEWAY_Xd 0x001            //Identificador per Xd positiva
#define GATEWAY_mX 0x002            //Identificador per Xd negativa
#define GATEWAY_Yd 0x003            //Identificador per Yd positiva
#define GATEWAY_mY 0x004            //Identificador per Yd negativa
#define GATEWAY_Zd 0x005            //Identificador per Zd (sempre positiva)
#define PLACA_1_S1 0x006
#define PLACA_1_S2 0x007
#define PLACA_1_S3 0x008            /*************************************************          */
#define PLACA_2_S1 0x009            /* Identificadors dels sensors S1,S2 i  S3  */
#define PLACA_2_S2 0x00A            /* per a cadascun dels tres miralls.        */
#define PLACA_2_S3 0x00B            /* PLACA_i_Sj ---> Correspon al sensor      */
#define PLACA_3_S1 0x00C            /*                    j del mirall i        */
#define PLACA_3_S2 0x00D            /****************************************
          ***********/
#define PLACA_3_S3 0x00E

void main() {
   signed int16 Xd, Yd, Zd;        //posició del punter verd en la base T
   float Zs1df, Zs2df, Zs3df;      //variables auxiliars
   float Za1df, Za2df, Za3df;      //variables auxiliars
   float opt;                      //variable auxiliar
   int S1, S2, S3;          //lectura dels sensors S1, S2 i S3
   float Zpd;                      //altura desitjada del punt P (centre del mirall)
   float Zs1d, Zs2d, Zs3d;         //altura desitjada dels sensors S1, S2 i S3
   float Zs1, Zs2, Zs3;            //altura instantània dels sensors S1, S2 i S3
   float Za1d, Za2d, Za3d;         //altura desitjada dels actuadors A1, A2 i A3
```

```
float Za1, Za2, Za3;            //altura instantània dels actuadors A1, A2 i A3
int i;                          //variable auxiliar
int j;                          //variable pel control del flux de dades dels sensors
float er;                       //error
int x;                  //entrada a la planta (etapa de potència + motor A2)
float SERR;             //variables auxiliars pel càlcul de la llei de control
float KP, KI;           //constant proporcional i constant integral
int dades[2];           //variable que emmagatzema els missatges del bus CAN
int sensor[1];          //variable que emmagatzema els missatges del bus CAN
int rx_len,rxstat;      //variables auxiliars
int32 rx_id;            //variables auxiliars

/* Inicialització del convertidor A/D */
setup_adc(ADC_CLOCK_INTERNAL);
setup_adc_ports(ALL_ANALOG);
set_adc_channel(1);                         //sel·lecció del pin RA1/AN1 "analog input"

/* Inicialització del PWM */
setup_timer_2(T2_DIV_BY_1, 255, 1);   /* Sel·lecció de:
                                         Prescaler = 1
                                         Period = 255
                                         Postscale = 1 */
setup_ccp1(CCP_PWM);        //inicialització del "PWM module"

/* Inicialització del bus CAN */
can_init();
can_putd(0x100,0,0,1,TRUE,FALSE);
delay_ms(1000);

/* Inicialització de variables auxiliars */
SERR = 0.0;             //inicialització de l'error acumulat
KP = 1;                 //constant proporcional del PI
KI = 0.02;              //constant integral del PI

i = 0;
j = 0;

Zpd = 18.0;

/* Inicialització de les variables Xd, Yd, Zd */
```

```
Xd = 0;
Yd = 0;
Zd = 1899;

/* Inicialització de les posicions dels sensors */
S1 = 255;
S2 = 255;
S3 = 255;

/* Espera a que s'inicialitzi tot el sistema i arribi el primer missatge amb
les coordenades del punter làser */
while (i == 0) {
  if (can_kbhit()) {
    if (can_getd(rx_id,&dades[0],rx_len,rxstat)) {
      if (rx_id == GATEWAY_Xd) {
        i = 1;
      }
      else if (rx_id == GATEWAY_Yd) {
        i = 1;
      }
      else if (rx_id == GATEWAY_Zd) {
        i = 1;
      }
      else if (rx_id == GATEWAY_mX) {
        i = 1;
      }
      else if (rx_id == GATEWAY_mY) {
        i = 1;
      }
      else {
      }
    }
  }
}

/* Inici del bucle infinit */
while (TRUE) {

  /* Obtenció dels missatges CAN (si existeixen)
  Coordenades desitjades (Xd,Yd,Zd) i sensors (S1 i S3) */
```

```
if (can_kbhit()) {
  if (can_getd(rx_id,&dades[0],rx_len,rxstat)) {
    if (rx_id == GATEWAY_Xd) {
      Xd = dades[0] * 256;
      Xd = Xd + dades[1];
    }
    else if (rx_id == GATEWAY_Yd) {
      Yd = dades[0] * 256;
      Yd = Yd + dades[1];
    }
    else if (rx_id == GATEWAY_Zd) {
      Zd = dades[0] * 256;
      Zd = Zd + dades[1];
    }
    else if (rx_id == GATEWAY_mX) {
      Xd = dades[0] * (-256);
      Xd = Xd - dades[1];
    }
    else if (rx_id == GATEWAY_mY) {
      Yd = dades[0] * (-256);
      Yd = Yd - dades[1];
    }
    else if (rx_id == PLACA_1_S1) {
      S1 = dades[0];
    }
    else if (rx_id == PLACA_1_S3) {
      S3 = dades[0];
    }
    else {
    }
  }
}
else {
}

/* Lectura del sensor S2 */
S2 = read_adc();
sensor[0] = S2;

/* Escriptura de la posició del sensor S2 al bus CAN */
```

```
if (j >= 10) {
  can_putd(PLACA_1_S2, sensor, 1, 1, TRUE, FALSE);
  j = 0;
}
else {
  j = j+1;
}

/* Càlcul de les posicions actuals */
Zs1 = 0.04706*S1+10;
Zs2 = 0.04706*S2+10;
Zs3 = 0.04706*S3+10;

Za1 = (((75*(Zs1-2*Zs2+Zs3)/3)/64)+(Zs1+Zs2+Zs3)/3);
Za2 = (((37.5*(Zs1+Zs2-2*Zs3)/3)/32)+(Zs1+Zs2+Zs3)/3);
Za3 = (((37.5*(Zs2-2*Zs1+Zs3)/3)/32)+(Zs1+Zs2+Zs3)/3);

/* Càlcul d'un possible pla desitjat */
Zs1df = ((8310.0-(55.4*(float)Yd)+(32.0*(float)Xd))/((float)Zd-Zpd))+Zpd;
Zs2df = ((-64.0*(float)Xd)/((float)Zd-Zpd))+Zpd;
Zs3df = 3.0*Zpd-Zs1df-Zs2df;

Za1df = (((75*(Zs1df-2*Zs2df+Zs3df)/3)/64)+(Zs1df+Zs2df+Zs3df)/3);
Za2df = (((37.5*(Zs1df+Zs2df-2*Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);
Za3df = (((37.5*(Zs2df-2*Zs1df+Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);

/* Cerca del pla desitjat per aproximacions succesives. Es buscarà
el pla mes baix possible */
if ((Za1df >= Za2df)&&(Za2df >= Za3df)) {
  opt = 14.0 - Za3df;
}
else if ((Za1df >= Za3df)&&(Za3df >= Za2df)) {
  opt = 14.0 - Za2df;
}
else if ((Za2df >= Za1df)&&(Za1df >= Za3df)) {
  opt = 14.0 - Za3df;
}
else if ((Za2df >= Za3df)&&(Za3df >= Za1df)) {
  opt = 14.0 - Za1df;
}
```

```
else if ((Za3df >= Za1df)&&(Za1df >= Za2df)) {
  opt = 14.0 - Za2df;
}
else {
  opt = 14.0 - Za1df;
}

Zpd = Zpd + opt;
Zs1df = ((8310.0-(55.4*(float)Yd)+(32.0*(float)Xd))/((float)Zd-Zpd))+Zpd;
Zs2df = ((-64.0*(float)Xd)/((float)Zd-Zpd))+Zpd;
Zs3df = 3.0*Zpd-Zs1df-Zs2df;

//Za1df = (((75*(Zs1df-2*Zs2df+Zs3df)/3)/64)+(Zs1df+Zs2df+Zs3df)/3);
Za2df = (((37.5*(Zs1df+Zs2df-2*Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);
//Za3df = (((37.5*(Zs2df-2*Zs1df+Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);

/* Comprovació de que les altures desitjades dels sensors estan dins del
rang permés: Zsidf<=21.96 */
i = 0;
if (Zs1df > 21.96) {
  i = 1;
}
else if (Zs2df > 21.96) {
  i = 2;
}
else if (Zs3df > 21.96) {
  i = 3;
}
else {
}

//Za1d = Za1df;
Za2d = Za2df;
//Za3d = Za3df;

/* Error er sobre l'actuador A2 */
er = Za2d - Za2;            /* Error de posició de l'actuador A2
                               Si er > 0 el motor ha de pujar (x -> 255)
                               Si er < 0 el motor ha de baixar (x -> 0) */
/* Controlador PI */
```

```
      SERR = SERR + er;
      er = KP * er + KI * SERR;


      /* Saturació de l'error er */
      if (er < -22.74) {
        er = -22.74;
      }
      if (er > 22.74) {
        er = 22.74;
      }

      /* Càlcul de l'entrada x a la planta */
      if (er >= 0.0) {
        er = 5.1012 * er + 139.0;      //escalat de er del rang 0..22.74 al 139..255
      }
      else {
        er = 5.0131 * er + 114.0;      //escalat de er del rang -22.74..0 al 0..114
      }

      if (i==0) {
        x = (int8)er;
      }
      else {
        x = 126;                       //per a x pertanyents al rang 115..138 el motor
        SERR = 0;                      //no es mou (zona morta de les etapes de potència)
      }
      set_pwm1_duty(x);                //escriptura del "Duty cicle" (temps en alta)
                                       //(actuador A2 del Mirall_1)
    }
  }
```

D4. Codi del Node 3 del Mirall 1

```
/* PLACA (mirall_1).
   Calcula les posicions desitjades de TOTS els sensors del mirall_1.
   Calcula la posició desitjada de l'actuador A3.
   Mou l'actuador A3, llegeix directament del sensor S3.
*/
#include <ccscana.c>
```

```
#include <STDLIB.H>



/* Identificadors dels nodes, per ordre de prioritat (de + a -) */
#define GATEWAY_Xd 0x001            //Identificador per Xd positiva
#define GATEWAY_mX 0x002            //Identificador per Xd negativa
#define GATEWAY_Yd 0x003            //Identificador per Yd positiva
#define GATEWAY_mY 0x004            //Identificador per Yd negativa
#define GATEWAY_Zd 0x005            //Identificador per Zd (sempre positiva)
#define PLACA_1_S1 0x006
#define PLACA_1_S2 0x007
#define PLACA_1_S3 0x008            /*************************************************          */
#define PLACA_2_S1 0x009            /* Identificadors dels sensors S1,S2 i  S3  */
#define PLACA_2_S2 0x00A            /* per a cadascun dels tres miralls.        */
#define PLACA_2_S3 0x00B            /* PLACA_i_Sj ---> Correspon al sensor      */
#define PLACA_3_S1 0x00C            /*                    j del mirall i        */
#define PLACA_3_S2 0x00D            /****************************************
        ***********/
#define PLACA_3_S3 0x00E

void main() {
  signed int16 Xd, Yd, Zd;        //posició del punter verd en la base T
  float Zs1df, Zs2df, Zs3df;      //variables auxiliars
  float Za1df, Za2df, Za3df;      //variables auxiliars
  float opt;                      //variable auxiliar
  int S1, S2, S3;           //lectura dels sensors S1, S2 i S3
  float Zpd;                      //altura desitjada del punt P (centre del mirall)
  float Zs1d, Zs2d, Zs3d;         //altura desitjada dels sensors S1, S2 i S3
  float Zs1, Zs2, Zs3;            //altura instantània dels sensors S1, S2 i S3
  float Za1d, Za2d, Za3d;         //altura desitjada dels actuadors A1, A2 i A3
  float Za1, Za2, Za3;            //altura instantània dels actuadors A1, A2 i A3
  int i;                          //variable auxiliar
  int j;                          //variable pel control del flux de dades dels sensors
  float er;                       //error
  int x;                          //entrada a la planta (etapa de potència + motor A3)
  float SERR;               //variables auxiliars pel càlcul de la llei de control
  float KP, KI;                   //constant proporcional i constant integral
  int dades[2];                   //variable que emmagatzema els missatges del bus CAN
  int sensor[1];                  //variable que emmagatzema els missatges del bus CAN
```

```
int rx_len,rxstat;                  //variables auxiliars
int32 rx_id;                        //variables auxiliars



/* Inicialització del convertidor A/D */
setup_adc(ADC_CLOCK_INTERNAL);
setup_adc_ports(ALL_ANALOG);
set_adc_channel(1);                 //sel·lecció del pin RA1/AN1 "analog input"

/* Inicialització del PWM */
setup_timer_2(T2_DIV_BY_1, 255, 1);  /* Sel·lecció de:
                                     Prescaler = 1
                                     Period = 255
                                     Postscale = 1 */
setup_ccp1(CCP_PWM);        //inicialització del "PWM module"

/* Inicialització del bus CAN */
can_init();
can_putd(0x100,0,0,1,TRUE,FALSE);
delay_ms(1000);

/* Inicialització de variables auxiliars */
SERR = 0.0;             //inicialització de l'error acumulat
KP = 1;                 //constant proporcional del PI
KI = 0.02;              //constant integral del PI

i = 0;
j = 0;

Zpd = 18.0;

/* Inicialització de les variables Xd, Yd, Zd */
Xd = 0;
Yd = 0;
Zd = 1899;

/* Inicialització de les posicions dels sensors */
S1 = 255;
S2 = 255;
```

```
S3 = 255;

/* Espera a que s'inicialitzi tot el sistema i arribi el primer missatge amb
les coordenades del punter làser */
while (i == 0) {
  if (can_kbhit()) {
    if (can_getd(rx_id,&dades[0],rx_len,rxstat)) {
      if (rx_id == GATEWAY_Xd) {
        i = 1;
      }
      else if (rx_id == GATEWAY_Yd) {
        i = 1;
      }
      else if (rx_id == GATEWAY_Zd) {
        i = 1;
      }
      else if (rx_id == GATEWAY_mX) {
        i = 1;
      }
      else if (rx_id == GATEWAY_mY) {
        i = 1;
      }
      else {
      }
    }
  }
}

/* Inici del bucle infinit */
while (TRUE) {

  /* Obtenció dels missatges CAN (si existeixen)
  Coordenades desitjades (Xd,Yd,Zd) i sensors (S1 i S2) */
  if (can_kbhit()) {
    if (can_getd(rx_id,&dades[0],rx_len,rxstat)) {
      if (rx_id == GATEWAY_Xd) {
        Xd = dades[0] * 256;
        Xd = Xd + dades[1];
      }
      else if (rx_id == GATEWAY_Yd) {
```

```
        Yd = dades[0] * 256;
        Yd = Yd + dades[1];
      }
      else if (rx_id == GATEWAY_Zd) {
        Zd = dades[0] * 256;
        Zd = Zd + dades[1];
      }
      else if (rx_id == GATEWAY_mX) {
        Xd = dades[0] * (-256);
        Xd = Xd - dades[1];
      }
      else if (rx_id == GATEWAY_mY) {
        Yd = dades[0] * (-256);
        Yd = Yd - dades[1];
      }
      else if (rx_id == PLACA_1_S1) {
        S1 = dades[0];
      }
      else if (rx_id == PLACA_1_S2) {
        S2 = dades[0];
      }
      else {
      }
    }
  }
  else {
  }

  /* Lectura del sensor S3 */
  S3 = read_adc();
  sensor[0] = S3;

  /* Escriptura de la posició del sensor S3 al bus CAN */
  if (j >= 10) {
    can_putd(PLACA_1_S3, sensor, 1, 1, TRUE, FALSE);
    j = 0;
  }
  else {
    j = j+1;
  }
```

```
/* Càlcul de les posicions actuals */
Zs1 = 0.04706*S1+10;
Zs2 = 0.04706*S2+10;
Zs3 = 0.04706*S3+10;

Za1 = (((75*(Zs1-2*Zs2+Zs3)/3)/64)+(Zs1+Zs2+Zs3)/3);
Za2 = (((37.5*(Zs1+Zs2-2*Zs3)/3)/32)+(Zs1+Zs2+Zs3)/3);
Za3 = (((37.5*(Zs2-2*Zs1+Zs3)/3)/32)+(Zs1+Zs2+Zs3)/3);

/* Càlcul d'un possible pla desitjat */
Zs1df = ((8310.0-(55.4*(float)Yd)+(32.0*(float)Xd))/((float)Zd-Zpd))+Zpd;
Zs2df = ((-64.0*(float)Xd)/((float)Zd-Zpd))+Zpd;
Zs3df = 3.0*Zpd-Zs1df-Zs2df;

Za1df = (((75*(Zs1df-2*Zs2df+Zs3df)/3)/64)+(Zs1df+Zs2df+Zs3df)/3);
Za2df = (((37.5*(Zs1df+Zs2df-2*Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);
Za3df = (((37.5*(Zs2df-2*Zs1df+Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);

/* Cerca del pla desitjat per aproximacions succesives. Es buscarà
el pla mes baix possible */
if ((Za1df >= Za2df)&&(Za2df >= Za3df)) {
  opt = 14.0 - Za3df;
}
else if ((Za1df >= Za3df)&&(Za3df >= Za2df)) {
  opt = 14.0 - Za2df;
}
else if ((Za2df >= Za1df)&&(Za1df >= Za3df)) {
  opt = 14.0 - Za3df;
}
else if ((Za2df >= Za3df)&&(Za3df >= Za1df)) {
  opt = 14.0 - Za1df;
}
else if ((Za3df >= Za1df)&&(Za1df >= Za2df)) {
  opt = 14.0 - Za2df;
}
else {
  opt = 14.0 - Za1df;
}
```

```
Zpd = Zpd + opt;
Zs1df = ((8310.0-(55.4*(float)Yd)+(32.0*(float)Xd))/((float)Zd-Zpd))+Zpd;
Zs2df = ((-64.0*(float)Xd)/((float)Zd-Zpd))+Zpd;
Zs3df = 3.0*Zpd-Zs1df-Zs2df;

//Za1df = (((75*(Zs1df-2*Zs2df+Zs3df)/3)/64)+(Zs1df+Zs2df+Zs3df)/3);
//Za2df = (((37.5*(Zs1df+Zs2df-2*Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);
Za3df = (((37.5*(Zs2df-2*Zs1df+Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);

/* Comprovació de que les altures desitjades dels sensors estan dins del
rang permés: Zsidf<=21.96 */
i = 0;
if (Zs1df > 21.96) {
  i = 1;
}
else if (Zs2df > 21.96) {
  i = 2;
}
else if (Zs3df > 21.96) {
  i = 3;
}
else {
}

//Za1d = Za1df;
//Za2d = Za2df;
Za3d = Za3df;

/* Error er sobre l'actuador A3 */
er = Za3d - Za3;        /* Error de posició de l'actuador A3
                           Si er > 0 el motor ha de pujar (x -> 255)
                           Si er < 0 el motor ha de baixar (x -> 0) */
/* Controlador PI */
SERR = SERR + er;
er = KP * er + KI * SERR;

/* Saturació de l'error er */
if (er < -22.74) {
  er = -22.74;
}
```

```
    if (er > 22.74) {
      er = 22.74;
    }


    /* Càlcul de l'entrada x a la planta */
    if (er >= 0.0) {
      er = 5.1012 * er + 139.0;      //escalat de er del rang 0..22.74 al 139..255
    }
    else {
      er = 5.0131 * er + 114.0;      //escalat de er del rang -22.74..0 al 0..114
    }

    if (i==0) {
      x = (int8)er;
    }
    else {
      x = 126;                       //per a x pertanyents al rang 115..138 el motor
      SERR = 0;                      //no es mou (zona morta de les etapes de potència)
    }
    set_pwm1_duty(x);                //escriptura del "Duty cicle" (temps en alta)
                                     //(actuador A3 del Mirall_1)
  }
}
```

D5. Codi del Node 1 del Mirall 2

```
/* PLACA (mirall_2).
  Calcula les posicions desitjades de TOTS els sensors del mirall_2.
  Calcula la posició desitjada de l'actuador A1.
  Mou l'actuador A1, llegeix directament del sensor S1.
*/
#include <ccscana.c>
#include <STDLIB.H>

/* Identificadors dels nodes, per ordre de prioritat (de + a -) */
#define GATEWAY_Xd 0x001          //Identificador per Xd positiva
#define GATEWAY_mX 0x002          //Identificador per Xd negativa
#define GATEWAY_Yd 0x003          //Identificador per Yd positiva
#define GATEWAY_mY 0x004          //Identificador per Yd negativa
```

```
#define GATEWAY_Zd 0x005            //Identificador per Zd (sempre positiva)
#define PLACA_1_S1 0x006
#define PLACA_1_S2 0x007
#define PLACA_1_S3 0x008            /*************************************************          */
#define PLACA_2_S1 0x009            /* Identificadors dels sensors S1,S2 i  S3  */
#define PLACA_2_S2 0x00A            /* per a cadascun dels tres miralls.        */
#define PLACA_2_S3 0x00B            /* PLACA_i_Sj ---> Correspon al sensor      */
#define PLACA_3_S1 0x00C            /*                    j del mirall i        */
#define PLACA_3_S2 0x00D            /****************************************
        ***********/
#define PLACA_3_S3 0x00E

void main() {
  signed int16 Xd, Yd, Zd;          //posició del punter verd en la base T
  signed int16 Xdd, Ydd;            //posició del punter verd en la base T'
  float Xdf, Ydf;                   //variables auxiliars
  int p, q;                         //variables pel control del càlcul de Xd i Yd
  float Zs1df, Zs2df, Zs3df;        //variables auxiliars
  float Za1df, Za2df, Za3df;        //variables auxiliars
  float opt;                        //variable auxiliar
  int S1, S2, S3;           //lectura dels sensors S1, S2 i S3
  float Zpd;                        //altura desitjada del punt P (centre del mirall)
  float Zs1d, Zs2d, Zs3d;           //altura desitjada dels sensors S1, S2 i S3
  float Zs1, Zs2, Zs3;              //altura instantània dels sensors S1, S2 i S3
  float Za1d, Za2d, Za3d;           //altura desitjada dels actuadors A1, A2 i A3
  float Za1, Za2, Za3;              //altura instantània dels actuadors A1, A2 i A3
  int i;                            //variable auxiliar
  int j;                            //variable pel control del flux de dades dels sensors
  float er;                         //error
  int x;                            //entrada a la planta (etapa de potència + motor A1)
  float SERR;               //variables auxiliars pel càlcul de la llei de control
  float KP, KI;                     //constant proporcional i constant integral
  int dades[2];                     //variable que emmagatzema els missatges del bus CAN
  int sensor[1];                    //variable que emmagatzema els missatges del bus CAN
  int rx_len,rxstat;                //variables auxiliars
  int32 rx_id;                      //variables auxiliars

  /* Inicialització del convertidor A/D */
  setup_adc(ADC_CLOCK_INTERNAL);
  setup_adc_ports(ALL_ANALOG);
```

```
set_adc_channel(1);     //sel·lecció del pin RA1/AN1 "analog input"

/* Inicialització del PWM */
setup_timer_2(T2_DIV_BY_1, 255, 1);   /* Sel·lecció de:
                                         Prescaler = 1
                                         Period = 255
                                         Postscale = 1 */
setup_ccp1(CCP_PWM);          //inicialització del "PWM module"

/* Inicialització del bus CAN */
can_init();
can_putd(0x100,0,0,1,TRUE,FALSE);
delay_ms(1000);

/* Inicialització de variables auxiliars */
SERR = 0.0;             //inicialització de l'error acumulat
KP = 1;                 //constant proporcional del PI
KI = 0.02;              //constant integral del PI

p = 0;
q = 0;
i = 0;
j = 0;

Zpd = 18.0;

/* Inicialització de les variables Xd, Yd, Zd, Xdd, Ydd */
Xd = 0;
Yd = 0;
Zd = 1899;

Xdd = 0;
Ydd = 0;

/* Inicialització de les posicions dels sensors */
S1 = 255;
S2 = 255;
S3 = 255;

/* Espera a que s'inicialitzi tot el sistema i arribi el primer missatge amb
```

```
les coordenades del punter làser */
while (i == 0) {
  if (can_kbhit()) {
    if (can_getd(rx_id,&dades[0],rx_len,rxstat)) {
      if (rx_id == GATEWAY_Xd) {
        i = 1;
      }
      else if (rx_id == GATEWAY_Yd) {
        i = 1;
      }
      else if (rx_id == GATEWAY_Zd) {
        i = 1;
      }
      else if (rx_id == GATEWAY_mX) {
        i = 1;
      }
      else if (rx_id == GATEWAY_mY) {
        i = 1;
      }
      else {
      }
    }
  }
}

/* Inici del bucle infinit */
while (TRUE) {

  /* Obtenció dels missatges CAN (si existeixen)
  Coordenades desitjades (Xd,Yd,Zd) i sensors (S2 i S3) */
  if (can_kbhit()) {
    if (can_getd(rx_id,&dades[0],rx_len,rxstat)) {
      if (rx_id == GATEWAY_Xd) {
        Xd = dades[0] * 256;
        Xd = Xd + dades[1];
        p = 1;
      }
      else if (rx_id == GATEWAY_Yd) {
        Yd = dades[0] * 256;
        Yd = Yd + dades[1];
```

```
        q = 1;
      }
      else if (rx_id == GATEWAY_Zd) {
        Zd = dades[0] * 256;
        Zd = Zd + dades[1];
      }
      else if (rx_id == GATEWAY_mX) {
        Xd = dades[0] * (-256);
        Xd = Xd - dades[1];
        p = 1;
      }
      else if (rx_id == GATEWAY_mY) {
        Yd = dades[0] * (-256);
        Yd = Yd - dades[1];
        q = 1;
      }
      else if (rx_id == PLACA_2_S2) {
        S2 = dades[0];
      }
      else if (rx_id == PLACA_2_S3) {
        S3 = dades[0];
      }
      else {
      }
    }
  }
  else {
  }

  /* Canvi de base. Rotació de -120 graus. Pas de la Base T → Base T' */
  if ((p == 1)&&(q == 1)) {
    Xdf = (-0.5*(float)Xd)-(0.866*(float)Yd);
    Ydf = (0.866*(float)Xd)-(0.5*(float)Yd);
    Xdd = (signed int16)Xdf;
    Ydd = (signed int16)Ydf;
    p = 0;
    q = 0;
  }
  else {
  }
```

```
/* Lectura del sensor S1 */
S1 = read_adc();
sensor[0] = S1;

/* Escriptura de la posició del sensor S1 al bus CAN */
if (j >= 10) {
  can_putd(PLACA_2_S1, sensor, 1, 1, TRUE, FALSE);
  j = 0;
}
else {
  j = j+1;
}

/* Càlcul de les posicions actuals */
Zs1 = 0.04706*S1+10;
Zs2 = 0.04706*S2+10;
Zs3 = 0.04706*S3+10;

Za1 = (((75*(Zs1-2*Zs2+Zs3)/3)/64)+(Zs1+Zs2+Zs3)/3);
Za2 = (((37.5*(Zs1+Zs2-2*Zs3)/3)/32)+(Zs1+Zs2+Zs3)/3);
Za3 = (((37.5*(Zs2-2*Zs1+Zs3)/3)/32)+(Zs1+Zs2+Zs3)/3);

/* Càlcul d'un possible pla desitjat */
Zs1df = ((8310.0-(55.4*(float)Ydd)+(32.0*(float)Xdd))/((float)Zd-Zpd))+Zpd;
Zs2df = ((-64.0*(float)Xdd)/((float)Zd-Zpd))+Zpd;
Zs3df = 3.0*Zpd-Zs1df-Zs2df;

Za1df = (((75*(Zs1df-2*Zs2df+Zs3df)/3)/64)+(Zs1df+Zs2df+Zs3df)/3);
Za2df = (((37.5*(Zs1df+Zs2df-2*Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);
Za3df = (((37.5*(Zs2df-2*Zs1df+Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);

/* Cerca del pla desitjat per aproximacions succesives. Es buscarà
el pla mes baix possible */
if ((Za1df >= Za2df)&&(Za2df >= Za3df)) {
  opt = 14.0 - Za3df;
}
else if ((Za1df >= Za3df)&&(Za3df >= Za2df)) {
  opt = 14.0 - Za2df;
}
```

```
else if ((Za2df >= Za1df)&&(Za1df >= Za3df)) {
  opt = 14.0 - Za3df;
}
else if ((Za2df >= Za3df)&&(Za3df >= Za1df)) {
  opt = 14.0 - Za1df;
}
else if ((Za3df >= Za1df)&&(Za1df >= Za2df)) {
  opt = 14.0 - Za2df;
}
else {
  opt = 14.0 - Za1df;
}

Zpd = Zpd + opt;
Zs1df = ((8310.0-(55.4*(float)Ydd)+(32.0*(float)Xdd))/((float)Zd-Zpd))+Zpd;
Zs2df = ((-64.0*(float)Xdd)/((float)Zd-Zpd))+Zpd;
Zs3df = 3.0*Zpd-Zs1df-Zs2df;

Za1df = (((75*(Zs1df-2*Zs2df+Zs3df)/3)/64)+(Zs1df+Zs2df+Zs3df)/3);
//Za2df = (((37.5*(Zs1df+Zs2df-2*Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);
//Za3df = (((37.5*(Zs2df-2*Zs1df+Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);

/* Comprovació de que les altures desitjades dels sensors estan dins del
rang permés: Zsidf<=21.96 */
i = 0;
if (Zs1df > 21.96) {
  i = 1;
}
else if (Zs2df > 21.96) {
  i = 2;
}
else if (Zs3df > 21.96) {
  i = 3;
}
else {
}

Za1d = Za1df;
//Za2d = Za2df;
//Za3d = Za3df;
```

```
    /* Error er sobre l'actuador A1 */
    er = Za1d - Za1;        /* Error de posició de l'actuador A1
                               Si er > 0 el motor ha de pujar (x -> 255)
                               Si er < 0 el motor ha de baixar (x -> 0) */

/* Controlador PI */
    SERR = SERR + er;
    er = KP * er + KI * SERR;

    /* Saturació de l'error er */
    if (er < -22.74) {
      er = -22.74;
    }
    if (er > 22.74) {
      er = 22.74;
    }

    /* Càlcul de l'entrada x a la planta */
    if (er >= 0.0) {
      er = 5.1012 * er + 139.0;     //escalat de er del rang 0..22.74 al 139..255
    }
    else {
      er = 5.0131 * er + 114.0;     //escalat de er del rang -22.74..0 al 0..114
    }

    if (i==0) {
      x = (int8)er;
    }
    else {
      x = 126;                      //per a x pertanyents al rang 115..138 el motor
      SERR = 0;                     //no es mou (zona morta de les etapes de potència)
    }
    set_pwm1_duty(x);               //escriptura del "Duty cicle" (temps en alta)
                                    //(actuador A1 del Mirall_2)
  }
}
```

D6. Codi del Node 2 del Mirall 2

```
/* PLACA (mirall_2).
   Calcula les posicions desitjades de TOTS els sensors del mirall_2.
   Calcula la posició desitjada de l'actuador A2.
   Mou l'actuador A2, llegeix directament del sensor S2.
*/

#include <ccscana.c>
#include <STDLIB.H>

/* Identificadors dels nodes, per ordre de prioritat (de + a -) */
#define GATEWAY_Xd 0x001            //Identificador per Xd positiva
#define GATEWAY_mX 0x002            //Identificador per Xd negativa
#define GATEWAY_Yd 0x003            //Identificador per Yd positiva
#define GATEWAY_mY 0x004            //Identificador per Yd negativa
#define GATEWAY_Zd 0x005            //Identificador per Zd (sempre positiva)
#define PLACA_1_S1 0x006
#define PLACA_1_S2 0x007
#define PLACA_1_S3 0x008            /*************************************************      */
#define PLACA_2_S1 0x009            /* Identificadors dels sensors S1,S2 i  S3  */
#define PLACA_2_S2 0x00A            /* per a cadascun dels tres miralls.        */
#define PLACA_2_S3 0x00B            /* PLACA_i_Sj ---> Correspon al sensor      */
#define PLACA_3_S1 0x00C            /*                    j del mirall i        */
#define PLACA_3_S2 0x00D            /*************************************
          ***********/
#define PLACA_3_S3 0x00E

void main() {
  signed int16 Xd, Yd, Zd;        //posició del punter verd en la base T
  signed int16 Xdd, Ydd;          //posició del punter verd en la base T'
  float Xdf, Ydf;                 //variables auxiliars
  int p, q;                       //variables pel control del càlcul de Xd i Yd
  float Zs1df, Zs2df, Zs3df;      //variables auxiliars
  float Za1df, Za2df, Za3df;      //variables auxiliars
  float opt;                      //variable auxiliar
  int S1, S2, S3;           //lectura dels sensors S1, S2 i S3
  float Zpd;                      //altura desitjada del punt P (centre del mirall)
  float Zs1d, Zs2d, Zs3d;         //altura desitjada dels sensors S1, S2 i S3
  float Zs1, Zs2, Zs3;            //altura instantània dels sensors S1, S2 i S3
```

```
float Za1d, Za2d, Za3d;          //altura desitjada dels actuadors A1, A2 i A3
float Za1, Za2, Za3;             //altura instantània dels actuadors A1, A2 i A3
int i;                           //variable auxiliar
int j;                           //variable pel control del flux de dades dels sensors
float er;                        //error
int x;                           //entrada a la planta (etapa de potència + motor A2)
float SERR;                      //variables auxiliars pel càlcul de la llei de control
float KP, KI;                    //constant proporcional i constant integral
int dades[2];                    //variable que emmagatzema els missatges del bus CAN
int sensor[1];           //variable que emmagatzema els missatges del bus CAN
int rx_len,rxstat;       //variables auxiliars
int32 rx_id;             //variables auxiliars

/* Inicialització del convertidor A/D */
setup_adc(ADC_CLOCK_INTERNAL);
setup_adc_ports(ALL_ANALOG);
set_adc_channel(1);                      //sel·lecció del pin RA1/AN1 "analog input"

/* Inicialització del PWM */
setup_timer_2(T2_DIV_BY_1, 255, 1);   /* Sel·lecció de:
                                         Prescaler = 1
                                         Period = 255
                                         Postscale = 1 */
setup_ccp1(CCP_PWM);             //inicialització del "PWM module"

/* Inicialització del bus CAN */
can_init();
can_putd(0x100,0,0,1,TRUE,FALSE);
delay_ms(1000);

/* Inicialització de variables auxiliars */
SERR = 0.0;              //inicialització de l'error acumulat
KP = 1;                  //constant proporcional del PI
KI = 0.02;               //constant integral del PI

p = 0;
q = 0;
i = 0;
j = 0;
```

```
Zpd = 18.0;

/* Inicialització de les variables Xd, Yd, Zd, Xdd, Ydd */
Xd = 0;
Yd = 0;
Zd = 1899;

Xdd = 0;
Ydd = 0;

/* Inicialització de les posicions dels sensors */
S1 = 255;
S2 = 255;
S3 = 255;

/* Espera a que s'inicialitzi tot el sistema i arribi el primer missatge amb
les coordenades del punter làser */
while (i == 0) {
  if (can_kbhit()) {
    if (can_getd(rx_id,&dades[0],rx_len,rxstat)) {
      if (rx_id == GATEWAY_Xd) {
        i = 1;
      }
      else if (rx_id == GATEWAY_Yd) {
        i = 1;
      }
      else if (rx_id == GATEWAY_Zd) {
        i = 1;
      }
      else if (rx_id == GATEWAY_mX) {
        i = 1;
      }
      else if (rx_id == GATEWAY_mY) {
        i = 1;
      }
      else {
      }
    }
  }
}
```

```
/* Inici del bucle infinit */
while (TRUE) {

  /* Obtenció dels missatges CAN (si existeixen)
  Coordenades desitjades (Xd,Yd,Zd) i sensors (S1 i S3) */
  if (can_kbhit()) {
    if (can_getd(rx_id,&dades[0],rx_len,rxstat)) {
      if (rx_id == GATEWAY_Xd) {
        Xd = dades[0] * 256;
        Xd = Xd + dades[1];
        p = 1;
      }
      else if (rx_id == GATEWAY_Yd) {
        Yd = dades[0] * 256;
        Yd = Yd + dades[1];
        q = 1;
      }
      else if (rx_id == GATEWAY_Zd) {
        Zd = dades[0] * 256;
        Zd = Zd + dades[1];
      }
      else if (rx_id == GATEWAY_mX) {
        Xd = dades[0] * (-256);
        Xd = Xd - dades[1];
        p = 1;
      }
      else if (rx_id == GATEWAY_mY) {
        Yd = dades[0] * (-256);
        Yd = Yd - dades[1];
        q = 1;
      }
      else if (rx_id == PLACA_2_S1) {
        S1 = dades[0];
      }
      else if (rx_id == PLACA_2_S3) {
        S3 = dades[0];
      }
      else {
      }
```

```
  }
}
else {
}

/* Canvi de base. Rotació de -120 graus. Pas de la Base T → Base T' */
if ((p == 1)&&(q == 1)) {
  Xdf = (-0.5*(float)Xd)-(0.866*(float)Yd);
  Ydf = (0.866*(float)Xd)-(0.5*(float)Yd);
  Xdd = (signed int16)Xdf;
  Ydd = (signed int16)Ydf;
  p = 0;
  q = 0;
}
else {
}

/* Lectura del sensor S2 */
S2 = read_adc();
sensor[0] = S2;

/* Escriptura de la posició del sensor S2 al bus CAN */
if (j >= 10) {
  can_putd(PLACA_2_S2, sensor, 1, 1, TRUE, FALSE);
  j = 0;
}
else {
  j = j+1;
}

/* Càlcul de les posicions actuals */
Zs1 = 0.04706*S1+10;
Zs2 = 0.04706*S2+10;
Zs3 = 0.04706*S3+10;

Za1 = (((75*(Zs1-2*Zs2+Zs3)/3)/64)+(Zs1+Zs2+Zs3)/3);
Za2 = (((37.5*(Zs1+Zs2-2*Zs3)/3)/32)+(Zs1+Zs2+Zs3)/3);
Za3 = (((37.5*(Zs2-2*Zs1+Zs3)/3)/32)+(Zs1+Zs2+Zs3)/3);

/* Càlcul d'un possible pla desitjat */
```

```
Zs1df = ((8310.0-(55.4*(float)Ydd)+(32.0*(float)Xdd))/((float)Zd-Zpd))+Zpd;
Zs2df = ((-64.0*(float)Xdd)/((float)Zd-Zpd))+Zpd;
Zs3df = 3.0*Zpd-Zs1df-Zs2df;

Za1df = (((75*(Zs1df-2*Zs2df+Zs3df)/3)/64)+(Zs1df+Zs2df+Zs3df)/3);
Za2df = (((37.5*(Zs1df+Zs2df-2*Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);
Za3df = (((37.5*(Zs2df-2*Zs1df+Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);


/* Cerca del pla desitjat per aproximacions succesives. Es buscarà
el pla mes baix possible */
if ((Za1df >= Za2df)&&(Za2df >= Za3df)) {
  opt = 14.0 - Za3df;
}
else if ((Za1df >= Za3df)&&(Za3df >= Za2df)) {
  opt = 14.0 - Za2df;
}
else if ((Za2df >= Za1df)&&(Za1df >= Za3df)) {
  opt = 14.0 - Za3df;
}
else if ((Za2df >= Za3df)&&(Za3df >= Za1df)) {
  opt = 14.0 - Za1df;
}
else if ((Za3df >= Za1df)&&(Za1df >= Za2df)) {
  opt = 14.0 - Za2df;
}
else {
  opt = 14.0 - Za1df;
}

Zpd = Zpd + opt;
Zs1df = ((8310.0-(55.4*(float)Ydd)+(32.0*(float)Xdd))/((float)Zd-Zpd))+Zpd;
Zs2df = ((-64.0*(float)Xdd)/((float)Zd-Zpd))+Zpd;
Zs3df = 3.0*Zpd-Zs1df-Zs2df;

//Za1df = (((75*(Zs1df-2*Zs2df+Zs3df)/3)/64)+(Zs1df+Zs2df+Zs3df)/3);
Za2df = (((37.5*(Zs1df+Zs2df-2*Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);
//Za3df = (((37.5*(Zs2df-2*Zs1df+Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);

/* Comprovació de que les altures desitjades dels sensors estan dins del
```

```
rang permés: Zsidf<=21.96 */
i = 0;
if (Zs1df > 21.96) {
  i = 1;
}
else if (Zs2df > 21.96) {
  i = 2;
}
else if (Zs3df > 21.96) {
  i = 3;
}
else {
}

//Za1d = Za1df;
Za2d = Za2df;
//Za3d = Za3df;

/* Error er sobre l'actuador A2 */
er = Za2d - Za2;        /* Error de posició de l'actuador A2
                         Si er > 0 el motor ha de pujar (x -> 255)
                         Si er < 0 el motor ha de baixar (x -> 0) */
/* Controlador PI */
SERR = SERR + er;
er = KP * er + KI * SERR;

/* Saturació de l'error er */
if (er < -22.74) {
  er = -22.74;
}
if (er > 22.74) {
  er = 22.74;
}

/* Càlcul de l'entrada x a la planta */
if (er >= 0.0) {
  er = 5.1012 * er + 139.0;      //escalat de er del rang 0..22.74 al 139..255
}
else {
  er = 5.0131 * er + 114.0;      //escalat de er del rang -22.74..0 al 0..114
```

```
    }

    if (i==0) {
      x = (int8)er;
    }
    else {
      x = 126;                    //per a x pertanyents al rang 115..138 el motor
      SERR = 0;                   //no es mou (zona morta de les etapes de potència)

    }
    set_pwm1_duty(x);             //escriptura del "Duty cicle" (temps en alta)
                                  //(actuador A2 del Mirall_2)
  }
}
```

D7. Codi del Node 3 del Mirall 2

```
/* PLACA (mirall_2).
   Calcula les posicions desitjades de TOTS els sensors del mirall_2.
   Calcula la posició desitjada de l'actuador A3.
   Mou l'actuador A3, llegeix directament del sensor S3.
*/
#include <ccscana.c>
#include <STDLIB.H>

/* Identificadors dels nodes, per ordre de prioritat (de + a -) */
#define GATEWAY_Xd 0x001          //Identificador per Xd positiva
#define GATEWAY_mX 0x002          //Identificador per Xd negativa
#define GATEWAY_Yd 0x003          //Identificador per Yd positiva
#define GATEWAY_mY 0x004          //Identificador per Yd negativa
#define GATEWAY_Zd 0x005          //Identificador per Zd (sempre positiva)
#define PLACA_1_S1 0x006
#define PLACA_1_S2 0x007
#define PLACA_1_S3 0x008          /*************************************************      */
#define PLACA_2_S1 0x009          /* Identificadors dels sensors S1,S2 i  S3  */
#define PLACA_2_S2 0x00A          /* per a cadascun dels tres miralls.        */
#define PLACA_2_S3 0x00B          /* PLACA_i_Sj ---> Correspon al sensor      */
#define PLACA_3_S1 0x00C          /*                    j del mirall i        */
#define PLACA_3_S2 0x00D          /****************************************
          ***********/
```

```
#define PLACA_3_S3 0x00E

void main() {
  signed int16 Xd, Yd, Zd;          //posició del punter verd en la base T
  signed int16 Xdd, Ydd;            //posició del punter verd en la base T'
  float Xdf, Ydf;                   //variables auxiliars
  int p, q;                         //variables pel control del càlcul de Xd i Yd
  float Zs1df, Zs2df, Zs3df;        //variables auxiliars
  float Za1df, Za2df, Za3df;        //variables auxiliars
  float opt;                        //variable auxiliar
  int S1, S2, S3;           //lectura dels sensors S1, S2 i S3
  float Zpd;                        //altura desitjada del punt P (centre del mirall)
  float Zs1d, Zs2d, Zs3d;           //altura desitjada dels sensors S1, S2 i S3
  float Zs1, Zs2, Zs3;              //altura instantània dels sensors S1, S2 i S3
  float Za1d, Za2d, Za3d;           //altura desitjada dels actuadors A1, A2 i A3
  float Za1, Za2, Za3;              //altura instantània dels actuadors A1, A2 i A3
  int i;                            //variable auxiliar
  int j;                            //variable pel control del flux de dades dels sensors
  float er;                         //error
  int x;                            //entrada a la planta (etapa de potència + motor A3)
  float SERR;                       //variables auxiliars pel càlcul de la llei de control
  float KP, KI;                     //constant proporcional i constant integral
  int dades[2];                     //variable que emmagatzema els missatges del bus CAN
  int sensor[1];                    //variable que emmagatzema els missatges del bus CAN
  int rx_len,rxstat;                //variables auxiliars
  int32 rx_id;                      //variables auxiliars

  /* Inicialització del convertidor A/D */
  setup_adc(ADC_CLOCK_INTERNAL);
  setup_adc_ports(ALL_ANALOG);
  set_adc_channel(1);                       //sel·lecció del pin RA1/AN1 "analog input"

  /* Inicialització del PWM */
  setup_timer_2(T2_DIV_BY_1, 255, 1);   /* Sel·lecció de:
                                           Prescaler = 1
                                           Period = 255
                                           Postscale = 1 */
  setup_ccp1(CCP_PWM);          //inicialització del "PWM module"

  /* Inicialització del bus CAN */
```

```
can_init();
can_putd(0x100,0,0,1,TRUE,FALSE);
delay_ms(1000);

/* Inicialització de variables auxiliars */
SERR = 0.0;             //inicialització de l'error acumulat
KP = 1;                 //constant proporcional del PI
KI = 0.02;              //constant integral del PI

p = 0;
q = 0;
i = 0;
j = 0;

Zpd = 18.0;

/* Inicialització de les variables Xd, Yd, Zd, Xdd, Ydd */
Xd = 0;
Yd = 0;
Zd = 1899;

Xdd = 0;
Ydd = 0;

/* Inicialització de les posicions dels sensors */
S1 = 255;
S2 = 255;
S3 = 255;

/* Espera a que s'inicialitzi tot el sistema i arribi el primer missatge amb
les coordenades del punter làser */
while (i == 0) {
  if (can_kbhit()) {
    if (can_getd(rx_id,&dades[0],rx_len,rxstat)) {
      if (rx_id == GATEWAY_Xd) {
        i = 1;
      }
      else if (rx_id == GATEWAY_Yd) {
        i = 1;
      }
```

```
        else if (rx_id == GATEWAY_Zd) {
          i = 1;
        }
        else if (rx_id == GATEWAY_mX) {
          i = 1;
        }
        else if (rx_id == GATEWAY_mY) {
          i = 1;
        }
        else {
        }
      }
    }
  }

  /* Inici del bucle infinit */
  while (TRUE) {

    /* Obtenció dels missatges CAN (si existeixen)
    Coordenades desitjades (Xd,Yd,Zd) i sensors (S1 i S2) */
    if (can_kbhit()) {
      if (can_getd(rx_id,&dades[0],rx_len,rxstat)) {
        if (rx_id == GATEWAY_Xd) {
          Xd = dades[0] * 256;
          Xd = Xd + dades[1];
          p = 1;
        }
        else if (rx_id == GATEWAY_Yd) {
          Yd = dades[0] * 256;
          Yd = Yd + dades[1];
          q = 1;
        }
        else if (rx_id == GATEWAY_Zd) {
          Zd = dades[0] * 256;
          Zd = Zd + dades[1];
        }
        else if (rx_id == GATEWAY_mX) {
          Xd = dades[0] * (-256);
          Xd = Xd - dades[1];
          p = 1;
```

```
      }
      else if (rx_id == GATEWAY_mY) {
        Yd = dades[0] * (-256);
        Yd = Yd - dades[1];
        q = 1;
      }
      else if (rx_id == PLACA_2_S1) {
        S1 = dades[0];
      }
      else if (rx_id == PLACA_2_S2) {
        S2 = dades[0];
     }
      else {
      }
   }
}
else {
}

/* Canvi de base. Rotació de -120 graus. Passa de la Base T → Base T' */
if ((p == 1)&&(q == 1)) {
   Xdf = (-0.5*(float)Xd)-(0.866*(float)Yd);
   Ydf = (0.866*(float)Xd)-(0.5*(float)Yd);
   Xdd = (signed int16)Xdf;
   Ydd = (signed int16)Ydf;
   p = 0;
   q = 0;
}
else {
}

/* Lectura del sensor S3 */
S3 = read_adc();
sensor[0] = S3;

/* Escriptura de la posició del sensor S3 al bus CAN */
if (j >= 10) {
   can_putd(PLACA_2_S3, sensor, 1, 1, TRUE, FALSE);
   j = 0;
}
```

```
else {
  j = j+1;
}

/* Càlcul de les posicions actuals */
Zs1 = 0.04706*S1+10;
Zs2 = 0.04706*S2+10;
Zs3 = 0.04706*S3+10;

Za1 = (((75*(Zs1-2*Zs2+Zs3)/3)/64)+(Zs1+Zs2+Zs3)/3);
Za2 = (((37.5*(Zs1+Zs2-2*Zs3)/3)/32)+(Zs1+Zs2+Zs3)/3);
Za3 = (((37.5*(Zs2-2*Zs1+Zs3)/3)/32)+(Zs1+Zs2+Zs3)/3);

/* Càlcul d'un possible pla desitjat */
Zs1df = ((8310.0-(55.4*(float)Ydd)+(32.0*(float)Xdd))/((float)Zd-Zpd))+Zpd;
Zs2df = ((-64.0*(float)Xdd)/((float)Zd-Zpd))+Zpd;
Zs3df = 3.0*Zpd-Zs1df-Zs2df;

Za1df = (((75*(Zs1df-2*Zs2df+Zs3df)/3)/64)+(Zs1df+Zs2df+Zs3df)/3);
Za2df = (((37.5*(Zs1df+Zs2df-2*Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);
Za3df = (((37.5*(Zs2df-2*Zs1df+Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);

/* Cerca del pla desitjat per aproximacions succesives. Es buscarà
el pla mes baix possible */
if ((Za1df >= Za2df)&&(Za2df >= Za3df)) {
  opt = 14.0 - Za3df;
}
else if ((Za1df >= Za3df)&&(Za3df >= Za2df)) {
  opt = 14.0 - Za2df;
}
else if ((Za2df >= Za1df)&&(Za1df >= Za3df)) {
  opt = 14.0 - Za3df;
}
else if ((Za2df >= Za3df)&&(Za3df >= Za1df)) {
  opt = 14.0 - Za1df;
}
else if ((Za3df >= Za1df)&&(Za1df >= Za2df)) {
  opt = 14.0 - Za2df;
}
else {
```

```
  opt = 14.0 - Za1df;
}

Zpd = Zpd + opt;
Zs1df = ((8310.0-(55.4*(float)Ydd)+(32.0*(float)Xdd))/((float)Zd-Zpd))+Zpd;
Zs2df = ((-64.0*(float)Xdd)/((float)Zd-Zpd))+Zpd;
Zs3df = 3.0*Zpd-Zs1df-Zs2df;

//Za1df = (((75*(Zs1df-2*Zs2df+Zs3df)/3)/64)+(Zs1df+Zs2df+Zs3df)/3);
//Za2df = (((37.5*(Zs1df+Zs2df-2*Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);
Za3df = (((37.5*(Zs2df-2*Zs1df+Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);

/* Comprovació de que les altures desitjades dels sensors estan dins del
rang permés: Zsidf<=21.96 */
i = 0;
if (Zs1df > 21.96) {
  i = 1;
}
else if (Zs2df > 21.96) {
  i = 2;
}
else if (Zs3df > 21.96) {
  i = 3;
}
else {
}

//Za1d = Za1df;
//Za2d = Za2df;
Za3d = Za3df;

/* Error er sobre l'actuador A3 */
er = Za3d - Za3;        /* Error de posició de l'actuador A3
                           Si er > 0 el motor ha de pujar (x -> 255)
                           Si er < 0 el motor ha de baixar (x -> 0) */
/* Controlador PI */
SERR = SERR + er;
er = KP * er + KI * SERR;

/* Saturació de l'error er */
```

```
    if (er < -22.74) {
      er = -22.74;
    }
    if (er > 22.74) {
      er = 22.74;
    }

    /* Càlcul de l'entrada x a la planta */
    if (er >= 0.0) {
      er = 5.1012 * er + 139.0;      //escalat de er del rang 0..22.74 al 139..255
    }
    else {
      er = 5.0131 * er + 114.0;      //escalat de er del rang -22.74..0 al 0..114
    }

    if (i==0) {
      x = (int8)er;
    }
    else {
      x = 126;                       //per a x pertanyents al rang 115..138 el motor
      SERR = 0;                      //no es mou (zona morta de les etapes de potència)
    }
    set_pwm1_duty(x);                //escriptura del "Duty cicle" (temps en alta)
                                     //(actuador A3 del Mirall_2)
  }
}
```

D8. Codi del Node 1 del Mirall 3

```
/* PLACA (mirall_3).
   Calcula les posicions desitjades de TOTS els sensors del mirall_3.
   Calcula la posició desitjada de l'actuador A1.
   Mou l'actuador A1, llegeix directament del sensor S1.
*/
#include <ccscana.c>
#include <STDLIB.H>

/* Identificadors dels nodes, per ordre de prioritat (de + a -) */
#define GATEWAY_Xd 0x001            //Identificador per Xd positiva
#define GATEWAY_mX 0x002            //Identificador per Xd negativa
```

```
#define GATEWAY_Yd 0x003            //Identificador per Yd positiva
#define GATEWAY_mY 0x004            //Identificador per Yd negativa
#define GATEWAY_Zd 0x005            //Identificador per Zd (sempre positiva)
#define PLACA_1_S1 0x006
#define PLACA_1_S2 0x007
#define PLACA_1_S3 0x008            /*************************************************          */
#define PLACA_2_S1 0x009            /* Identificadors dels sensors S1,S2 i  S3  */
#define PLACA_2_S2 0x00A            /* per a cadascun dels tres miralls.        */
#define PLACA_2_S3 0x00B            /* PLACA_i_Sj ---> Correspon al sensor      */
#define PLACA_3_S1 0x00C            /*                    j del mirall i        */
#define PLACA_3_S2 0x00D            /****************************************
          ***********/
#define PLACA_3_S3 0x00E

void main() {
  signed int16 Xd, Yd, Zd;          //posició del punter verd en la base T
  signed int16 Xdd, Ydd;            //posició del punter verd en la base T''
  float Xdf, Ydf;                   //variables auxiliars
  int p, q;                         //variables pel control del càlcul de Xd i Yd
  float Zs1df, Zs2df, Zs3df;        //variables auxiliars
  float Za1df, Za2df, Za3df;        //variables auxiliars
  float opt;                        //variable auxiliar
  int S1, S2, S3;           //lectura dels sensors S1, S2 i S3
  float Zpd;                        //altura desitjada del punt P (centre del mirall)
  float Zs1d, Zs2d, Zs3d;           //altura desitjada dels sensors S1, S2 i S3
  float Zs1, Zs2, Zs3;              //altura instantània dels sensors S1, S2 i S3
  float Za1d, Za2d, Za3d;           //altura desitjada dels actuadors A1, A2 i A3
  float Za1, Za2, Za3;              //altura instantània dels actuadors A1, A2 i A3
  int i;                            //variable auxiliar
  int j;                            //variable pel control del flux de dades dels sensors
  float er;                         //error
  int x;                            //entrada a la planta (etapa de potència + motor A1)
  float SERR;               //variables auxiliars pel càlcul de la llei de control
  float KP, KI;                     //constant proporcional i constant integral
  int dades[2];                     //variable que emmagatzema els missatges del bus CAN
  int sensor[1];                    //variable que emmagatzema els missatges del bus CAN
  int rx_len,rxstat;                //variables auxiliars
  int32 rx_id;                      //variables auxiliars

  /* Inicialització del convertidor A/D */
```

```
setup_adc(ADC_CLOCK_INTERNAL);
setup_adc_ports(ALL_ANALOG);
set_adc_channel(1);                    //sel·lecció del pin RA1/AN1 "analog input"

/* Inicialització del PWM */
setup_timer_2(T2_DIV_BY_1, 255, 1);   /* Sel·lecció de:
                                       Prescaler = 1
                                       Period = 255
                                       Postscale = 1 */
setup_ccp1(CCP_PWM);          //inicialització del "PWM module"

/* Inicialització del bus CAN */
can_init();
can_putd(0x100,0,0,1,TRUE,FALSE);
delay_ms(1000);

/* Inicialització de variables auxiliars */
SERR = 0.0;             //inicialització de l'error acumulat
KP = 1;                 //constant proporcional del PI
KI = 0.02;              //constant integral del PI

p = 0;
q = 0;
i = 0;
j = 0;

Zpd = 18.0;

/* Inicialització de les variables Xd, Yd, Zd, Xdd, Ydd */
Xd = 0;
Yd = 0;
Zd = 1899;

Xdd = 0;
Ydd = 0;

/* Inicialització de les posicions dels sensors */
S1 = 255;
S2 = 255;
S3 = 255;
```

```
/* Espera a que s'inicialitzi tot el sistema i arribi el primer missatge amb
les coordenades del punter làser */
while (i == 0) {
  if (can_kbhit()) {
    if (can_getd(rx_id,&dades[0],rx_len,rxstat)) {
      if (rx_id == GATEWAY_Xd) {
        i = 1;
      }
      else if (rx_id == GATEWAY_Yd) {
        i = 1;
      }
      else if (rx_id == GATEWAY_Zd) {
        i = 1;
      }
      else if (rx_id == GATEWAY_mX) {
        i = 1;
      }
      else if (rx_id == GATEWAY_mY) {
        i = 1;
      }
      else {
      }
    }
  }
}

/* Inici del bucle infinit */
while (TRUE) {

  /* Obtenció dels missatges CAN (si existeixen)
  Coordenades desitjades (Xd,Yd,Zd) i sensors (S2 i S3) */
  if (can_kbhit()) {
    if (can_getd(rx_id,&dades[0],rx_len,rxstat)) {
      if (rx_id == GATEWAY_Xd) {
        Xd = dades[0] * 256;
        Xd = Xd + dades[1];
        p = 1;
      }
      else if (rx_id == GATEWAY_Yd) {
```

```
        Yd = dades[0] * 256;
        Yd = Yd + dades[1];
        q = 1;
      }
      else if (rx_id == GATEWAY_Zd) {
        Zd = dades[0] * 256;
        Zd = Zd + dades[1];
      }
      else if (rx_id == GATEWAY_mX) {
        Xd = dades[0] * (-256);
        Xd = Xd - dades[1];
        p = 1;
      }
      else if (rx_id == GATEWAY_mY) {
        Yd = dades[0] * (-256);
        Yd = Yd - dades[1];
        q = 1;
      }
      else if (rx_id == PLACA_3_S2) {
        S2 = dades[0];
      }
      else if (rx_id == PLACA_3_S3) {
        S3 = dades[0];
      }
      else {
      }
    }
  }
  else {
  }

  /* Canvi de base. Rotació de 120 graus. Pas de la Base T → Base T'' */
  if ((p == 1)&&(q == 1)) {
    Xdf = (-0.5*(float)Xd)+(0.866*(float)Yd);
    Ydf = (-0.866*(float)Xd)-(0.5*(float)Yd);
    Xdd = (signed int16)Xdf;
    Ydd = (signed int16)Ydf;
    p = 0;
    q = 0;
  }
```

```
else {
}

/* Lectura del sensor S1 */
S1 = read_adc();
sensor[0] = S1;

/* Escriptura de la posició del sensor S1 al bus CAN */
if (j >= 10) {
  can_putd(PLACA_3_S1, sensor, 1, 1, TRUE, FALSE);
  j = 0;
}
else {
  j = j+1;
}

/* Càlcul de les posicions actuals */
Zs1 = 0.04706*S1+10;
Zs2 = 0.04706*S2+10;
Zs3 = 0.04706*S3+10;

Za1 = (((75*(Zs1-2*Zs2+Zs3)/3)/64)+(Zs1+Zs2+Zs3)/3);
Za2 = (((37.5*(Zs1+Zs2-2*Zs3)/3)/32)+(Zs1+Zs2+Zs3)/3);
Za3 = (((37.5*(Zs2-2*Zs1+Zs3)/3)/32)+(Zs1+Zs2+Zs3)/3);

/* Càlcul d'un possible pla desitjat */
Zs1df = ((8310.0-(55.4*(float)Ydd)+(32.0*(float)Xdd))/((float)Zd-Zpd))+Zpd;
Zs2df = ((-64.0*(float)Xdd)/((float)Zd-Zpd))+Zpd;
Zs3df = 3.0*Zpd-Zs1df-Zs2df;

Za1df = (((75*(Zs1df-2*Zs2df+Zs3df)/3)/64)+(Zs1df+Zs2df+Zs3df)/3);
Za2df = (((37.5*(Zs1df+Zs2df-2*Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);
Za3df = (((37.5*(Zs2df-2*Zs1df+Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);

/* Cerca del pla desitjat per aproximacions succesives. Es buscarà
el pla mes baix possible */
if ((Za1df >= Za2df)&&(Za2df >= Za3df)) {
  opt = 14.0 - Za3df;
}
else if ((Za1df >= Za3df)&&(Za3df >= Za2df)) {
```

```
  opt = 14.0 - Za2df;
}
else if ((Za2df >= Za1df)&&(Za1df >= Za3df)) {
  opt = 14.0 - Za3df;
}
else if ((Za2df >= Za3df)&&(Za3df >= Za1df)) {
  opt = 14.0 - Za1df;
}
else if ((Za3df >= Za1df)&&(Za1df >= Za2df)) {
  opt = 14.0 - Za2df;
}
else {
  opt = 14.0 - Za1df;
}

Zpd = Zpd + opt;
Zs1df = ((8310.0-(55.4*(float)Ydd)+(32.0*(float)Xdd))/((float)Zd-Zpd))+Zpd;
Zs2df = ((-64.0*(float)Xdd)/((float)Zd-Zpd))+Zpd;
Zs3df = 3.0*Zpd-Zs1df-Zs2df;

Za1df = (((75*(Zs1df-2*Zs2df+Zs3df)/3)/64)+(Zs1df+Zs2df+Zs3df)/3);
//Za2df = (((37.5*(Zs1df+Zs2df-2*Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);
//Za3df = (((37.5*(Zs2df-2*Zs1df+Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);

/* Comprovació de que les altures desitjades dels sensors estan dins del
rang permés: Zsidf<=21.96 */
i = 0;
if (Zs1df > 21.96) {
  i = 1;
}
else if (Zs2df > 21.96) {
  i = 2;
}
else if (Zs3df > 21.96) {
  i = 3;
}
else {
}

Za1d = Za1df;
```

```
    //Za2d = Za2df;
    //Za3d = Za3df;

    /* Error er sobre l'actuador A1 */
    er = Za1d - Za1;        /* Error de posició de l'actuador A1
                               Si er > 0 el motor ha de pujar (x -> 255)
                               Si er < 0 el motor ha de baixar (x -> 0) */

/* Controlador PI */
    SERR = SERR + er;
    er = KP * er + KI * SERR;

    /* Saturació de l'error er */
    if (er < -22.74) {
      er = -22.74;
    }
    if (er > 22.74) {
      er = 22.74;
    }

    /* Càlcul de l'entrada x a la planta */
    if (er >= 0.0) {
      er = 5.1012 * er + 139.0;     //escalat de er del rang 0..22.74 al 139..255
    }
    else {
      er = 5.0131 * er + 114.0;     //escalat de er del rang -22.74..0 al 0..114
    }

    if (i==0) {
      x = (int8)er;
    }
    else {
      x = 126;                      //per a x pertanyents al rang 115..138 el motor
      SERR = 0;                     //no es mou (zona morta de les etapes de potència)
    }
    set_pwm1_duty(x);               //escriptura del "Duty cicle" (temps en alta)
                                    //(actuador A1 del Mirall_3)
  }
}
```

D9. Codi del Node 2 del Mirall 3

```
/* PLACA (mirall_3).
   Calcula les posicions desitjades de TOTS els sensors del mirall_3.
   Calcula la posició desitjada de l'actuador A2.
   Mou l'actuador A2, llegeix directament del sensor S2.
*/

#include <ccscana.c>
#include <STDLIB.H>

/* Identificadors dels nodes, per ordre de prioritat (de + a -) */
#define GATEWAY_Xd 0x001            //Identificador per Xd positiva
#define GATEWAY_mX 0x002            //Identificador per Xd negativa
#define GATEWAY_Yd 0x003            //Identificador per Yd positiva
#define GATEWAY_mY 0x004            //Identificador per Yd negativa
#define GATEWAY_Zd 0x005            //Identificador per Zd (sempre positiva)
#define PLACA_1_S1 0x006
#define PLACA_1_S2 0x007
#define PLACA_1_S3 0x008            /*************************************************      */
#define PLACA_2_S1 0x009            /* Identificadors dels sensors S1,S2 i  S3  */
#define PLACA_2_S2 0x00A            /* per a cadascun dels tres miralls.        */
#define PLACA_2_S3 0x00B            /* PLACA_i_Sj ---> Correspon al sensor      */
#define PLACA_3_S1 0x00C            /*                    j del mirall i        */
#define PLACA_3_S2 0x00D            /*************************************
          ***********/
#define PLACA_3_S3 0x00E

void main() {
  signed int16 Xd, Yd, Zd;        //posició del punter verd en la base T
  signed int16 Xdd, Ydd;          //posició del punter verd en la base T'
  float Xdf, Ydf;                 //variables auxiliars
  int p, q;                       //variables pel control del càlcul de Xd i Yd
  float Zs1df, Zs2df, Zs3df;      //variables auxiliars
  float Za1df, Za2df, Za3df;      //variables auxiliars
  float opt;                      //variable auxiliar
  int S1, S2, S3;          //lectura dels sensors S1, S2 i S3
  float Zpd;                      //altura desitjada del punt P (centre del mirall)
  float Zs1d, Zs2d, Zs3d;         //altura desitjada dels sensors S1, S2 i S3
  float Zs1, Zs2, Zs3;            //altura instantània dels sensors S1, S2 i S3
```

```
float Za1d, Za2d, Za3d;          //altura desitjada dels actuadors A1, A2 i A3
float Za1, Za2, Za3;             //altura instantània dels actuadors A1, A2 i A3
int i;                           //variable auxiliar
int j;                           //variable pel control del flux de dades dels sensors
float er;                        //error
int x;                           //entrada a la planta (etapa de potència + motor A2)
float SERR;                      //variables auxiliars pel càlcul de la llei de control
float KP, KI;                    //constant proporcional i constant integral
int dades[2];                    //variable que emmagatzema els missatges del bus CAN
int sensor[1];           //variable que emmagatzema els missatges del bus CAN
int rx_len,rxstat;       //variables auxiliars
int32 rx_id;             //variables auxiliars

/* Inicialització del convertidor A/D */
setup_adc(ADC_CLOCK_INTERNAL);
setup_adc_ports(ALL_ANALOG);
set_adc_channel(1);                      //sel·lecció del pin RA1/AN1 "analog input"

/* Inicialització del PWM */
setup_timer_2(T2_DIV_BY_1, 255, 1);   /* Sel·lecció de:
                                         Prescaler = 1
                                         Period = 255
                                         Postscale = 1 */
setup_ccp1(CCP_PWM);         //inicialització del "PWM module"

/* Inicialització del bus CAN */
can_init();
can_putd(0x100,0,0,1,TRUE,FALSE);
delay_ms(1000);

/* Inicialització de variables auxiliars */
SERR = 0.0;              //inicialització de l'error acumulat
KP = 1;                  //constant proporcional del PI
KI = 0.02;               //constant integral del PI

p = 0;
q = 0;
i = 0;
j = 0;
```

```
Zpd = 18.0;

/* Inicialització de les variables Xd, Yd, Zd, Xdd, Ydd */
Xd = 0;
Yd = 0;
Zd = 1899;

Xdd = 0;
Ydd = 0;

/* Inicialització de les posicions dels sensors */
S1 = 255;
S2 = 255;
S3 = 255;

/* Espera a que s'inicialitzi tot el sistema i arribi el primer missatge amb
les coordenades del punter làser */
while (i == 0) {
  if (can_kbhit()) {
    if (can_getd(rx_id,&dades[0],rx_len,rxstat)) {
      if (rx_id == GATEWAY_Xd) {
        i = 1;
      }
      else if (rx_id == GATEWAY_Yd) {
        i = 1;
      }
      else if (rx_id == GATEWAY_Zd) {
        i = 1;
      }
      else if (rx_id == GATEWAY_mX) {
        i = 1;
      }
      else if (rx_id == GATEWAY_mY) {
        i = 1;
      }
      else {
      }
    }
  }
}
```

```
/* Inici del bucle infinit */
while (TRUE) {

  /* Obtenció dels missatges CAN (si existeixen)
  Coordenades desitjades (Xd,Yd,Zd) i sensors (S1 i S3) */
  if (can_kbhit()) {
    if (can_getd(rx_id,&dades[0],rx_len,rxstat)) {
      if (rx_id == GATEWAY_Xd) {
        Xd = dades[0] * 256;
        Xd = Xd + dades[1];
        p = 1;
      }
      else if (rx_id == GATEWAY_Yd) {
        Yd = dades[0] * 256;
        Yd = Yd + dades[1];
        q = 1;
      }
      else if (rx_id == GATEWAY_Zd) {
        Zd = dades[0] * 256;
        Zd = Zd + dades[1];
      }
      else if (rx_id == GATEWAY_mX) {
        Xd = dades[0] * (-256);
        Xd = Xd - dades[1];
        p = 1;
      }
      else if (rx_id == GATEWAY_mY) {
        Yd = dades[0] * (-256);
        Yd = Yd - dades[1];
        q = 1;
      }
      else if (rx_id == PLACA_3_S1) {
        S1 = dades[0];
      }
      else if (rx_id == PLACA_3_S3) {
        S3 = dades[0];
      }
      else {
      }
```

```
  }
}
else {
}

/* Canvi de base. Rotació de 120 graus. Pas de la Base T → Base T'' */
if ((p == 1)&&(q == 1)) {
  Xdf = (-0.5*(float)Xd)+(0.866*(float)Yd);
  Ydf = (-0.866*(float)Xd)-(0.5*(float)Yd);
  Xdd = (signed int16)Xdf;
  Ydd = (signed int16)Ydf;
  p = 0;
  q = 0;
}
else {
}

/* Lectura del sensor S2 */
S2 = read_adc();
sensor[0] = S2;

/* Escriptura de la posició del sensor S2 al bus CAN */
if (j >= 10) {
  can_putd(PLACA_3_S2, sensor, 1, 1, TRUE, FALSE);
  j = 0;
}
else {
  j = j+1;
}

/* Càlcul de les posicions actuals */
Zs1 = 0.04706*S1+10;
Zs2 = 0.04706*S2+10;
Zs3 = 0.04706*S3+10;

Za1 = (((75*(Zs1-2*Zs2+Zs3)/3)/64)+(Zs1+Zs2+Zs3)/3);
Za2 = (((37.5*(Zs1+Zs2-2*Zs3)/3)/32)+(Zs1+Zs2+Zs3)/3);
Za3 = (((37.5*(Zs2-2*Zs1+Zs3)/3)/32)+(Zs1+Zs2+Zs3)/3);

/* Càlcul d'un possible pla desitjat */
```

```
Zs1df = ((8310.0-(55.4*(float)Ydd)+(32.0*(float)Xdd))/((float)Zd-Zpd))+Zpd;
Zs2df = ((-64.0*(float)Xdd)/((float)Zd-Zpd))+Zpd;
Zs3df = 3.0*Zpd-Zs1df-Zs2df;

Za1df = (((75*(Zs1df-2*Zs2df+Zs3df)/3)/64)+(Zs1df+Zs2df+Zs3df)/3);
Za2df = (((37.5*(Zs1df+Zs2df-2*Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);
Za3df = (((37.5*(Zs2df-2*Zs1df+Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);

/* Cerca del pla desitjat per aproximacions succesives. Es buscarà
el pla mes baix possible */
if ((Za1df >= Za2df)&&(Za2df >= Za3df)) {
  opt = 14.0 - Za3df;
}
else if ((Za1df >= Za3df)&&(Za3df >= Za2df)) {
  opt = 14.0 - Za2df;
}
else if ((Za2df >= Za1df)&&(Za1df >= Za3df)) {
  opt = 14.0 - Za3df;
}
else if ((Za2df >= Za3df)&&(Za3df >= Za1df)) {
  opt = 14.0 - Za1df;
}
else if ((Za3df >= Za1df)&&(Za1df >= Za2df)) {
  opt = 14.0 - Za2df;
}
else {
  opt = 14.0 - Za1df;
}

Zpd = Zpd + opt;
Zs1df = ((8310.0-(55.4*(float)Ydd)+(32.0*(float)Xdd))/((float)Zd-Zpd))+Zpd;
Zs2df = ((-64.0*(float)Xdd)/((float)Zd-Zpd))+Zpd;
Zs3df = 3.0*Zpd-Zs1df-Zs2df;

//Za1df = (((75*(Zs1df-2*Zs2df+Zs3df)/3)/64)+(Zs1df+Zs2df+Zs3df)/3);
Za2df = (((37.5*(Zs1df+Zs2df-2*Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);
//Za3df = (((37.5*(Zs2df-2*Zs1df+Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);

/* Comprovació de que les altures desitjades dels sensors estan dins del
rang permés: Zsidf<=21.96 */
```

```
i = 0;
if (Zs1df > 21.96) {
  i = 1;
}
else if (Zs2df > 21.96) {
  i = 2;
}
else if (Zs3df > 21.96) {
  i = 3;
}
else {
}
//Za1d = Za1df;
Za2d = Za2df;
//Za3d = Za3df;

/* Error er sobre l'actuador A2 */
er = Za2d - Za2;        /* Error de posició de l'actuador A2
                           Si er > 0 el motor ha de pujar (x -> 255)
                           Si er < 0 el motor ha de baixar (x -> 0) */
/* Controlador PI */
SERR = SERR + er;
er = KP * er + KI * SERR;

/* Saturació de l'error er */
if (er < -22.74) {
  er = -22.74;
}
if (er > 22.74) {
  er = 22.74;
}

/* Càlcul de l'entrada x a la planta */
if (er >= 0.0) {
  er = 5.1012 * er + 139.0;      //escalat de er del rang 0..22.74 al 139..255
}
else {
  er = 5.0131 * er + 114.0;      //escalat de er del rang -22.74..0 al 0..114
}
```

```
    if (i==0) {
      x = (int8)er;
    }
    else {
      x = 126;                    //per a x pertanyents al rang 115..138 el motor
      SERR = 0;                   //no es mou (zona morta de les etapes de potència)

    }
    set_pwm1_duty(x);             //escriptura del "Duty cicle" (temps en alta)
                                  //(actuador A2 del Mirall_3)
  }
}
```

D10. Codi del Node 3 del Mirall 3

```
/* PLACA (mirall_3).
   Calcula les posicions desitjades de TOTS els sensors del mirall_3.
   Calcula la posició desitjada de l'actuador A3.
   Mou l'actuador A3, llegeix directament del sensor S3.
*/
#include <ccscana.c>
#include <STDLIB.H>

/* Identificadors dels nodes, per ordre de prioritat (de + a -) */
#define GATEWAY_Xd 0x001          //Identificador per Xd positiva
#define GATEWAY_mX 0x002          //Identificador per Xd negativa
#define GATEWAY_Yd 0x003          //Identificador per Yd positiva
#define GATEWAY_mY 0x004          //Identificador per Yd negativa
#define GATEWAY_Zd 0x005          //Identificador per Zd (sempre positiva)
#define PLACA_1_S1 0x006
#define PLACA_1_S2 0x007
#define PLACA_1_S3 0x008          /*************************************************      */
#define PLACA_2_S1 0x009          /* Identificadors dels sensors S1,S2 i  S3  */
#define PLACA_2_S2 0x00A          /* per a cadascun dels tres miralls.        */
#define PLACA_2_S3 0x00B          /* PLACA_i_Sj ---> Correspon al sensor      */
#define PLACA_3_S1 0x00C          /*                    j del mirall i        */
#define PLACA_3_S2 0x00D          /*************************************
          ***********/
#define PLACA_3_S3 0x00E
```

```
void main() {
  signed int16 Xd, Yd, Zd;          //posició del punter verd en la base T
  signed int16 Xdd, Ydd;            //posició del punter verd en la base T'
  float Xdf, Ydf;                   //variables auxiliars
  int p, q;                         //variables pel control del càlcul de Xd i Yd
  float Zs1df, Zs2df, Zs3df;        //variables auxiliars
  float Za1df, Za2df, Za3df;        //variables auxiliars
  float opt;                        //variable auxiliar
  int S1, S2, S3;           //lectura dels sensors S1, S2 i S3
  float Zpd;                        //altura desitjada del punt P (centre del mirall)
  float Zs1d, Zs2d, Zs3d;           //altura desitjada dels sensors S1, S2 i S3
  float Zs1, Zs2, Zs3;              //altura instantània dels sensors S1, S2 i S3
  float Za1d, Za2d, Za3d;           //altura desitjada dels actuadors A1, A2 i A3
  float Za1, Za2, Za3;              //altura instantània dels actuadors A1, A2 i A3
  int i;                            //variable auxiliar
  int j;                            //variable pel control del flux de dades dels sensors
  float er;                         //error
  int x;                            //entrada a la planta (etapa de potència + motor A3)
  float SERR;                       //variables auxiliars pel càlcul de la llei de control
  float KP, KI;                     //constant proporcional i constant integral
  int dades[2];                     //variable que emmagatzema els missatges del bus CAN
  int sensor[1];                    //variable que emmagatzema els missatges del bus CAN
  int rx_len,rxstat;                //variables auxiliars
  int32 rx_id;                      //variables auxiliars

  /* Inicialització del convertidor A/D */
  setup_adc(ADC_CLOCK_INTERNAL);
  setup_adc_ports(ALL_ANALOG);
  set_adc_channel(1);                       //sel·lecció del pin RA1/AN1 "analog input"

  /* Inicialització del PWM */
  setup_timer_2(T2_DIV_BY_1, 255, 1);   /* Sel·lecció de:
                                           Prescaler = 1
                                           Period = 255
                                           Postscale = 1 */
  setup_ccp1(CCP_PWM);          //inicialització del "PWM module"

  /* Inicialització del bus CAN */
  can_init();
  can_putd(0x100,0,0,1,TRUE,FALSE);
```

```
delay_ms(1000);

/* Inicialització de variables auxiliars */
SERR = 0.0;             //inicialització de l'error acumulat
KP = 1;                 //constant proporcional del PI
KI = 0.02;              //constant integral del PI

p = 0;
q = 0;
i = 0;
j = 0;

Zpd = 18.0;

/* Inicialització de les variables Xd, Yd, Zd, Xdd, Ydd */
Xd = 0;
Yd = 0;
Zd = 1899;

Xdd = 0;
Ydd = 0;

/* Inicialització de les posicions dels sensors */
S1 = 255;
S2 = 255;
S3 = 255;

/* Espera a que s'inicialitzi tot el sistema i arribi el primer missatge amb
les coordenades del punter làser */
while (i == 0) {
  if (can_kbhit()) {
    if (can_getd(rx_id,&dades[0],rx_len,rxstat)) {
      if (rx_id == GATEWAY_Xd) {
        i = 1;
      }
      else if (rx_id == GATEWAY_Yd) {
        i = 1;
      }
      else if (rx_id == GATEWAY_Zd) {
        i = 1;
```

```
        }
        else if (rx_id == GATEWAY_mX) {
          i = 1;
        }
        else if (rx_id == GATEWAY_mY) {
          i = 1;
        }
        else {
        }
      }
    }
  }


  /* Inici del bucle infinit */
  while (TRUE) {

    /* Obtenció dels missatges CAN (si existeixen)
    Coordenades desitjades (Xd,Yd,Zd) i sensors (S1 i S2) */
    if (can_kbhit()) {
      if (can_getd(rx_id,&dades[0],rx_len,rxstat)) {
        if (rx_id == GATEWAY_Xd) {
          Xd = dades[0] * 256;
          Xd = Xd + dades[1];
          p = 1;
        }
        else if (rx_id == GATEWAY_Yd) {
          Yd = dades[0] * 256;
          Yd = Yd + dades[1];
          q = 1;
        }
        else if (rx_id == GATEWAY_Zd) {
          Zd = dades[0] * 256;
          Zd = Zd + dades[1];
        }
        else if (rx_id == GATEWAY_mX) {
          Xd = dades[0] * (-256);
          Xd = Xd - dades[1];
          p = 1;
        }
```

```
      else if (rx_id == GATEWAY_mY) {
        Yd = dades[0] * (-256);
        Yd = Yd - dades[1];
        q = 1;
      }
      else if (rx_id == PLACA_3_S1) {
        S1 = dades[0];
      }
      else if (rx_id == PLACA_3_S2) {
        S2 = dades[0];
     }
      else {
      }
   }
}
else {
}

/* Canvi de base. Rotació de 120 graus. Passa de la Base T → Base T'' */
if ((p == 1)&&(q == 1)) {
   Xdf = (-0.5*(float)Xd)+(0.866*(float)Yd);
   Ydf = (-0.866*(float)Xd)-(0.5*(float)Yd);
   Xdd = (signed int16)Xdf;
   Ydd = (signed int16)Ydf;
   p = 0;
   q = 0;
}
else {
}

/* Lectura del sensor S3 */
S3 = read_adc();
sensor[0] = S3;

/* Escriptura de la posició del sensor S3 al bus CAN */
if (j >= 10) {
   can_putd(PLACA_3_S3, sensor, 1, 1, TRUE, FALSE);
   j = 0;
}
else {
```

```
  j = j+1;
}

/* Càlcul de les posicions actuals */
Zs1 = 0.04706*S1+10;
Zs2 = 0.04706*S2+10;
Zs3 = 0.04706*S3+10;

Za1 = (((75*(Zs1-2*Zs2+Zs3)/3)/64)+(Zs1+Zs2+Zs3)/3);
Za2 = (((37.5*(Zs1+Zs2-2*Zs3)/3)/32)+(Zs1+Zs2+Zs3)/3);
Za3 = (((37.5*(Zs2-2*Zs1+Zs3)/3)/32)+(Zs1+Zs2+Zs3)/3);

/* Càlcul d'un possible pla desitjat */
Zs1df = ((8310.0-(55.4*(float)Ydd)+(32.0*(float)Xdd))/((float)Zd-Zpd))+Zpd;
Zs2df = ((-64.0*(float)Xdd)/((float)Zd-Zpd))+Zpd;
Zs3df = 3.0*Zpd-Zs1df-Zs2df;

Za1df = (((75*(Zs1df-2*Zs2df+Zs3df)/3)/64)+(Zs1df+Zs2df+Zs3df)/3);
Za2df = (((37.5*(Zs1df+Zs2df-2*Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);
Za3df = (((37.5*(Zs2df-2*Zs1df+Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);

/* Cerca del pla desitjat per aproximacions succesives. Es buscarà
el pla mes baix possible */
if ((Za1df >= Za2df)&&(Za2df >= Za3df)) {
  opt = 14.0 - Za3df;
}
else if ((Za1df >= Za3df)&&(Za3df >= Za2df)) {
  opt = 14.0 - Za2df;
}
else if ((Za2df >= Za1df)&&(Za1df >= Za3df)) {
  opt = 14.0 - Za3df;
}
else if ((Za2df >= Za3df)&&(Za3df >= Za1df)) {
  opt = 14.0 - Za1df;
}
else if ((Za3df >= Za1df)&&(Za1df >= Za2df)) {
  opt = 14.0 - Za2df;
}
else {
  opt = 14.0 - Za1df;
```

```
}

Zpd = Zpd + opt;
Zs1df = ((8310.0-(55.4*(float)Ydd)+(32.0*(float)Xdd))/((float)Zd-Zpd))+Zpd;
Zs2df = ((-64.0*(float)Xdd)/((float)Zd-Zpd))+Zpd;
Zs3df = 3.0*Zpd-Zs1df-Zs2df;

//Za1df = (((75*(Zs1df-2*Zs2df+Zs3df)/3)/64)+(Zs1df+Zs2df+Zs3df)/3);
//Za2df = (((37.5*(Zs1df+Zs2df-2*Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);
Za3df = (((37.5*(Zs2df-2*Zs1df+Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);

/* Comprovació de que les altures desitjades dels sensors estan dins del
rang permés: Zsidf<=21.96 */
i = 0;
if (Zs1df > 21.96) {
  i = 1;
}
else if (Zs2df > 21.96) {
  i = 2;
}
else if (Zs3df > 21.96) {
  i = 3;
}
else {
}

//Za1d = Za1df;
//Za2d = Za2df;
Za3d = Za3df;

/* Error er sobre l'actuador A3 */
er = Za3d - Za3;        /* Error de posició de l'actuador A3
                           Si er > 0 el motor ha de pujar (x -> 255)
                           Si er < 0 el motor ha de baixar (x -> 0) */
/* Controlador PI */
SERR = SERR + er;
er = KP * er + KI * SERR;

/* Saturació de l'error er */
if (er < -22.74) {
```

```
      er = -22.74;
    }
    if (er > 22.74) {
      er = 22.74;
    }

    /* Càlcul de l'entrada x a la planta */
    if (er >= 0.0) {
      er = 5.1012 * er + 139.0;      //escalat de er del rang 0..22.74 al 139..255
    }
    else {
      er = 5.0131 * er + 114.0;      //escalat de er del rang -22.74..0 al 0..114
    }

    if (i==0) {
      x = (int8)er;
    }
    else {
      x = 126;                       //per a x pertanyents al rang 115..138 el motor
      SERR = 0;                      //no es mou (zona morta de les etapes de potència)
    }
    set_pwm1_duty(x);                //escriptura del "Duty cicle" (temps en alta)
                                     //(actuador A3 del Mirall_3)
  }
}
```

D11. Codi de ccscana.c

L'arxiu ccscana.c és un archiu de configuració cridat pels codis de tots els microcontroladors.

```
/* ccscana.c */
#include <18F458.h>
#fuses HS,NOPROTECT,NOLVP,NOWDT
#use delay(clock=20000000)
#include <can-18xxx8.c>
#use rs232(baud=9600,xmit=PIN_C6,rcv=PIN_C7)
```

E. Exemple de codi utilitzat a la simulació

A continuació es dóna el codi associat al bloc "Codi micro 1" del Mirall 1. Aquest codi serveix d'exemple per a tots els altres utilitzats a la simulació, ja que aquests programes estan directament extrets dels codis en llenguatge C implementats per als microcontroladors.

```
function [sys,x0,str,ts] = microprocesador1(t,x,u,flag)
%TIMESTWO S-function whose output is two times its input.
%   This M-file illustrates how to construct an M-file S-function that
%   computes an output value based upon its input.  The output of this
%   S-function is two times the input value:
%
%     y = 2 * u;
%
%   See sfuntmpl.m for a general S-function template.
%
%   See also SFUNTMPL.
%   Copyright 1990-2002 The MathWorks, Inc.
%   $Revision: 1.7 $
%
% Dispatch the flag. The switch function controls the calls to
% S-function routines at each simulation stage of the S-function.
%
switch flag,
  %%%%%%%%%%%%%%%%%%%%
  % Initialization %
  %%%%%%%%%%%%%%%%%%%%
  % Initialize the states, sample times, and state ordering strings.
  case 0
    [sys,x0,str,ts]=mdlInitializeSizes;
    global SERR1;
    global Zpd1;
    SERR1=0;
    Zpd1=18;
  %%%%%%%%%%%%
  % Outputs %
  %%%%%%%%%%%%
  % Return the outputs of the S-function block.
  case 3
```

```
  sys=mdlOutputs(t,x,u);
 %%%%%%%%%%%%%%%%%%%%%
 % Unhandled flags %
 %%%%%%%%%%%%%%%%%%%%%
 % There are no termination tasks (flag=9) to be handled.
 % Also, there are no continuous or discrete states,
 % so flags 1,2, and 4 are not used, so return an emptyu
 % matrix
 case { 1, 2, 4, 9 }
  sys=[];
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 % Unexpected flags (error handling)%
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 % Return an error message for unhandled flag values.
 otherwise
  error(['Unhandled flag = ',num2str(flag)]);
end
% end timestwo
%
%=================================================================
========
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the S-function.
%=================================================================
========
%
function [sys,x0,str,ts] = mdlInitializeSizes()
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1;  % dynamically sized
sizes.NumInputs      = -1;  % dynamically sized
sizes.DirFeedthrough = -1;   % has direct feedthrough
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
str = [];
x0  = [];
ts  = [-1 0];   % inherited sample time
% end mdlInitializeSizes
%
```

```
%=================================================================
========
% mdlOutputs
% Return the output vector for the S-function
%=================================================================
========
%
function sys = mdlOutputs(t,x,u)
sys = codigomicro1(u);
 % end mdlOutputs
```

I la rutina asociada al "codigomicro1()" és

```
function sys = codigomicro1(u)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%esta es la función que representa el código
%del micro 1. Hay que inicializarla
global SERR1;
global Zpd1;
KP=5;
KI=0.005;
Zd=u(1);
Xd=u(2);
Yd=u(3);
Zs1=u(6);
Zs2=u(9);
Zs3=u(12);
%Zs1 = 0.04706*S1+10;
%     Zs2 = 0.04706*S2+10;
%     Zs3 = 0.04706*S3+10;

    Za1 = (((75*(Zs1-2*Zs2+Zs3)/3)/64)+(Zs1+Zs2+Zs3)/3);
    Za2 = (((37.5*(Zs1+Zs2-2*Zs3)/3)/32)+(Zs1+Zs2+Zs3)/3);
    Za3 = (((37.5*(Zs2-2*Zs1+Zs3)/3)/32)+(Zs1+Zs2+Zs3)/3);

    %/* Càlcul d'un possible pla desitjat */
    Zs1df = ((8310.0-(55.4*Yd)+(32.0*Xd))/(Zd-Zpd1))+Zpd1;
    Zs2df = ((-64.0*Xd)/(Zd-Zpd1))+Zpd1;
    Zs3df = 3.0*Zpd1-Zs1df-Zs2df;
```

```
Za1df = (((75*(Zs1df-2*Zs2df+Zs3df)/3)/64)+(Zs1df+Zs2df+Zs3df)/3);
Za2df = (((37.5*(Zs1df+Zs2df-2*Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);
Za3df = (((37.5*(Zs2df-2*Zs1df+Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);

%/* Cerca del pla desitjat per aproximacions succesives. Es buscarà
%el pla mes baix possible */
if ((Za1df >= Za2df)&&(Za2df >= Za3df))
  opt = 14.0 - Za3df;
elseif ((Za1df >= Za3df)&&(Za3df >= Za2df))
  opt = 14.0 - Za2df;
elseif ((Za2df >= Za1df)&&(Za1df >= Za3df))
  opt = 14.0 - Za3df;
elseif ((Za2df >= Za3df)&&(Za3df >= Za1df))
  opt = 14.0 - Za1df;
elseif ((Za3df >= Za1df)&&(Za1df >= Za2df))
  opt = 14.0 - Za2df;
else
  opt = 14.0 - Za1df;
end

Zpd1 = Zpd1 + opt;
Zs1df = ((8310.0-(55.4*Yd)+(32.0*Xd))/(Zd-Zpd1))+Zpd1;
Zs2df = ((-64.0*Xd)/(Zd-Zpd1))+Zpd1;
Zs3df = 3.0*Zpd1-Zs1df-Zs2df;

Za1df = (((75*(Zs1df-2*Zs2df+Zs3df)/3)/64)+(Zs1df+Zs2df+Zs3df)/3);
% //Za2df = (((37.5*(Zs1df+Zs2df-2*Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);
%//Za3df = (((37.5*(Zs2df-2*Zs1df+Zs3df)/3)/32)+(Zs1df+Zs2df+Zs3df)/3);

%/* Comprovació de que les altures desitjades dels sensors estan dins del
%rang permés: Zsidf<=21.96 */
i = 0;
if (Zs1df > 21.96)
  i = 1;
elseif (Zs2df > 21.96)
  i = 2;
elseif (Zs3df > 21.96)
  i = 3;
end
```

```
    if (i==0)
      Za1d = Za1df;
      SERR1=0;
    %   //Za2d = Za2df;
    %   //Za3d = Za3df;
   else
      %output_high(PIN_B1);
      Za1d = Za1;
    %   //Za2d = Za2;
    %   //Za3d = Za3;
    end

    %/* Error er sobre l'actuador A1 */
    er = Za1d - Za1;       %/* Error de posició de l'actuador A1
                      %     Si er > 0 el motor ha de pujar (x -> 255)
                      %     Si er < 0 el motor ha de baixar (x -> 0) */
    %/* Controlador PI */
    SERR1 = SERR1 + er;
    er = KP * er + KI * SERR1;

    %/* Saturació de l'error er */
    if (er < -22.74)
      er = -22.74;
    end
    if (er > 22.74)
      er = 22.74;
    end

sys=[er];
end
```