Platform controller

01/07/2016

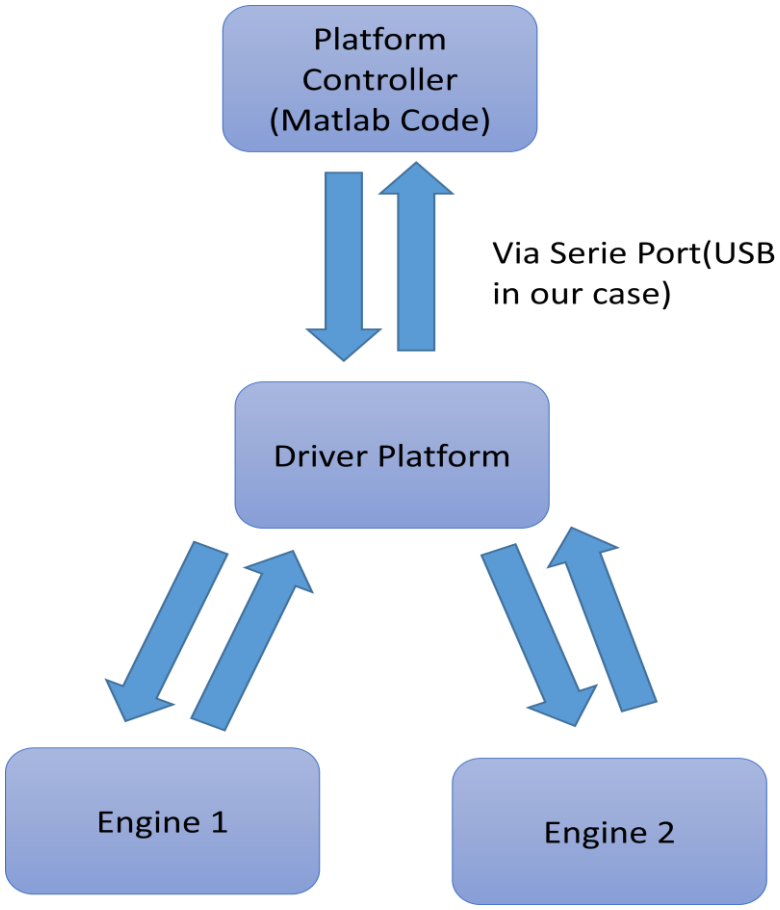
INDEX

Revision History

Data	**Versió**	**Descripció**	**Autor/s**
1/07/2016	1.0	First version of the documentation. At this point, the software to control the platform is 100% functional, with some additional features. This doc pretends to be a very simple guide of basic use of the software.	Victor Moyano

1. Preamble

This matlab software has been written in order to control the two engines that move the rotatory table. The engines are controlled by a Driver, which is connected to your PC via an USB cable. The connection is achieved via an asynchronous port serie RS-485.



As we can see, the matlab class communicates with the driver in the platform, which is connected to both engines. More engines could be connected to the driver (maximum 30), but at the time of writing this documentation, only 2 engines are used.

The code is written uniquely as a matlab class, and no extra librarys or files are needed for its execution. Only a matlab version, probably higher than 2014b, is needed.

2. Functionality of Matlab Class

2.1. Connecting and disconnecting the platform

```
>> p= platform

p =

  platform with properties:

                maximumCurrent: 60
                    deviceMode: 'Serie'
    positiveDirectionOfRotation: 'Clockwise'
              decelerationSteps: 999
               acelerationSteps: 500
                  rotationSpeed: 20
       stepsForCompleteRotation: 200
                      stepType: '16 microSteps'
                   stopCurrent: '15%'
                searchForLimit: 'False'
              continousMovement: 'False'
            waitForMoveToFinish: 'True'
                      lastMove: 'Clockwise'
                searchForLimit: 'False'
              continousMovement: 'False'
            waitForMoveToFinish: 'True'
                      lastMove: 'Clockwise'
                originPosition: [0 0]

>>
>> p.delete
```

The connecting and disconnecting procedures are trivial. An example here:

2.2. Move the platform:

Once the platform is connected, we can move the engines as we please. The unit for moving the platform are steps. 1 steps is equal to 0,045 degrees, so for moving 90 degrees, you should move 2000 steps. Then, for a complete revolution (360 degrees) you should move 8000 steps, and so on. To move the engine1 200 steps and -200 steps, you should do:

```
>> p.move([200 0])
>> p.move([-200 0])
```

If you want to move the engine1 400 steps, and the engine2 200 steps, do:

```
>> p.move([400 200])
```

A function in the class has been written so the user doesn't have to worry about the conversion from angles to steps. This function is the function deg2steps. A classic use is the following:

```
>> p.move(p.deg2steps([90 -27]))
```

If, for some reason, you need to convert steps to degrees, you can use the step2deg function, which does this for you as well:

```
>> p.steps2deg([200 0])

ans =

     9     0
```

2.3. Go to world's origin:

A sensor has been added to the horizontal engine, in order to be able to set a global 0. We can go to this position using the function home in the matlab class. A basic use is the following:

```
>> p.move([100,0])
p.home
Position 0 reached.
```

First we move the platform out of the origen, and then we move it to the origin using the home function. Probably, you could increase the speed at which the platform goes to the origin, because at 20 rpm (which is 0.5 in real world, due the effect of the redactors) is very slow.

2.4. Set custom origin:

If, for some reason, we want to set a new custom origin in a given position we can do this moving the platform to the position wanted, and then calling the function newOrigin, from the matlab class.

2.5. Set platform parameters:

To set the parameters of platform, you can do it as you do with any matlab object. Just do set ('nameOfParameter',valueToBeSet). And struct with all the platform parameters can be passed to the set function as well.

2.6. Get platform parameters:

To get the parameters of the platform, you can do it as you do with any matlab object. Just do get('nameOftheParameter') to get an struct with the parameter value. If you enter no input, an struct wih all the platform parameters will be returned.

2.7. Change default parameters :

At this time, if you want to change the default parameters, just edit the matlab class and put the values you want to be the default. You could also save the matlab parameters obtained with the get function in to a .mat, and then load this

parameters in to the matlab workspace and use the set function to set this parameters.

3. Platform Parameters

The current platform parameters supported in this code are the shown in the following table:

Name of Parameter	**Short Description**	**Range**	**Recommended Value**
maximumCurrent	Maximum current that will be given to the different engines.	From 1 to 200	105
deviceMode	Mode at which the camera will operate.	‘Serie’,’Enable’, ’Track’,’Special’	‘Serie’
positiveDirectionOfRotation	Direction of rotation which will be considered positive.	‘Clockwise, ‘AntiClockwise’	‘Clockwise’
decelerationSteps	Steps used for deceleration.	From 0 to 999	Maximum possible(999)
acelerationSteps	Steps used for aceleration	From 0 to 999	About 400
rotationSpeed	Speed at which the platform rotates	From 1 to 100	15
stepsForCompleteRotation	Steps needed for complete rotation. This is a variable that should be constant (depends of the type of engine).	200 or 300	200
stepType	This is another variable that depends from the engine. In this case, our engines work with step type 16 microSteps.	'Full Steps','Half Steps','16 microSteps','32 microSteps'	16 microSteps
stopCurrent	This is another variable that depends from the engine. We will work at 15%.	'15%' ,'40%' ,'55%' ,'75%'	‘15%’
searchForLimit	This is the variable that should be used if we want to search for the limit of movement. Used for going to worlds origin. User should not touch this.	‘False’,’True’	‘False’
continousMovement	Variable that we should use if we are doing a continous movement. User shouldn’t need this variable at all.	‘False’, ‘True’	‘False’
waitForMoveToFinish	Variable that indicates the software to wait for the	‘False’,’True’	‘True’

	platform to finish the move.		
lastMove	Variable that indicates in which direction the software did move.	‘Clockwise’,’AntiClockwise’	-
originPosition	Origin position at the current time.	[PositionEngine1 Position Engine2]	-

4. About the platform driver

Note: this section is a work in progress.

The messages sent to the driver are in the following format:

Nº Byte	Descripción	Valores posibles		Error asociado
0	Identificador del dispositivo	0x30, 0x31, 0x32, 0x33,... 0x4F		
1	Identificador de comando de movimiento	77D = 0x4D = M		m=109D=0x6D
2	Nº pasos movimiento	Unidades (0x30,...,0x39)		x=120D=0x78
3		Decenas (0x30,...,0x39)		
4		Centenas (0x30,...,0x39)		
5		Millares (0x30,...,0x39)		
6		Decenas de millar (0x30,...,0x39)		
7		Centenas de millar (0x30,...,0x39)		
8	Sentido de giro / Reset consigna acumulada	• Derecha = R (0x52) • Izquierda = L (0x4C)		
9	Nº pasos de la pendiente de aceleración	Unidades (0x30,...,0x39)		
10		Decenas (0x30,...,0x39)		
11		Centenas (0x30,...,0x39)		
12	Nº pasos de la pendiente de frenado	Unidades (0x30,...,0x39)		
13		Decenas (0x30,...,0x39)		
14		Centenas (0x30,...,0x39)		
15	Consigna de velocidad en RPM (0 a 999). *(Corresponde a velocidad 1 a 1000 rpm)*	Unidades (0x30,...,0x39)		
16		Decenas (0x30,...,0x39)		
17		Centenas (0x30,...,0x39)		
18	Selector de micropasos y tipo de motor	Motor 200 pasos • 0x31 = ½ paso • 0x32 = 8 • 0x33 = 16 • 0x34 = 32 • 0x35 = 64	Motor 400 pasos • 0x41 = ½ paso • 0x42 = 8 • 0x43 = 16 • 0x44 = 32 • 0x45 = 64	

19	Corriente de STOP (mantenimiento de par)	• 0x30 = 15 % • 0x31 = 40 % • 0x32 = 55 % • 0x33 = 75 %	
20	Búsqueda final de carrera (entrada tipo PNP)	• 0x30 = No testea el FC • 0x31 = Testea el FC	
21	Movimiento continuo	• 0x30 = Desactivado • 0x31 = Activado	
22	Respuesta final movimiento	• 0x30 = Sin respuesta • 0x31 = Con respuesta	
23	Carry Return	0x0D	

Respuesta desde el driver:

Nº Byte	Descripción	Valores posibles
0	Identificador del dispositivo	0x30, 0x31, 0x32, 0x33,... 0x4F
1	Identificador de comando de movimiento	77 = 0x4D = M
2	Error o resultado	• ACK=0x06 (Comunicación correcta) • NACK=0x15 (Error comunicación) • 121= 0x79=y (FC alcanzado) • 122=0x7A=z (Movimiento finalizado)
3	Carry Return	0x0D