UNIVERSIDAD DE OVIEDO

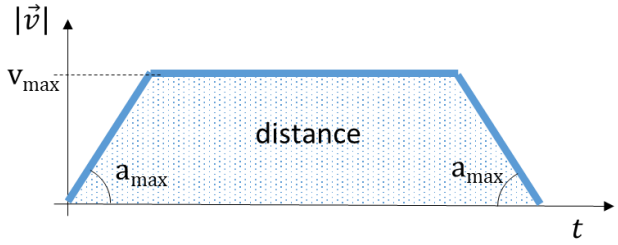
Departamento de Ingeniería Eléctrica,
Electrónica, de Computadores y Sistemas

Guía de Prácticas

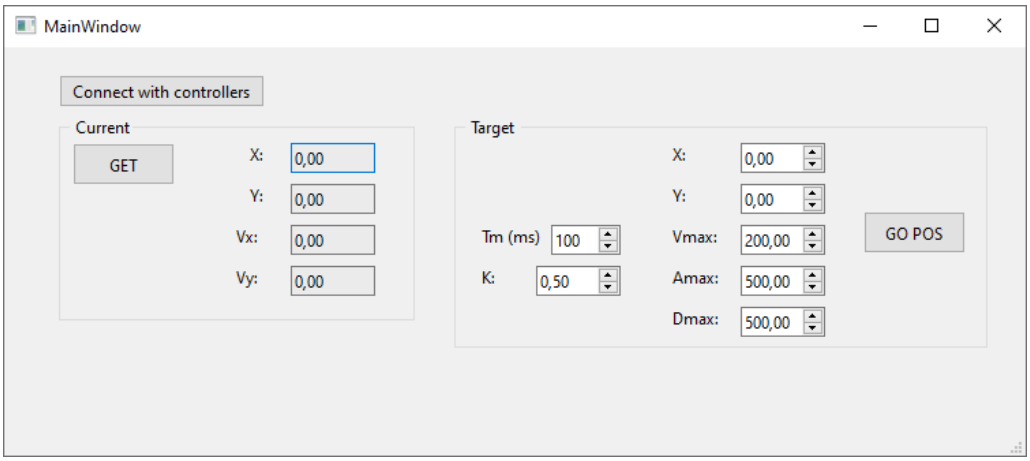
ASIGNATURA:	Implementación de Sistemas de Control		
CENTRO:	Centro Internacional de Postgrado		
ESTUDIOS:	Master en Ingeniería Mecatrónica		
CURSO:	2º	CUATRIMESTRE:	1
CARÁCTER:	Obligatoria	CRÉDITOS ECTS:	6
PROFESORADO:	Ignacio Alvarez, Fernando Briz		

PRACTICA 05: Programación y ejecución de trayectorias con Qt/C++ en modo Widgets

- Para el sistema VerticalPlotter simulado, añadir la opción de seguir una trayectoria rectilínea, a partir de datos obtenidos de una aplicación GUI (Qt Widgets), generando un perfil de velocidad trapezoidal:



1) Preparar aplicación Qt Widgets con los siguientes elementos (todos los datos en mm y mm/s):



(En letra cursiva: solamente si se dispone de la clase MyVerticalPlotterController funcional)

2) *Añadir variable miembro MyVerticalPlotterController vp; en la clase MainWindow.*

3) *En el slot para el botón "Connect with controllers", ejecutar vp.Connect( ) para iniciar la comunicación.*

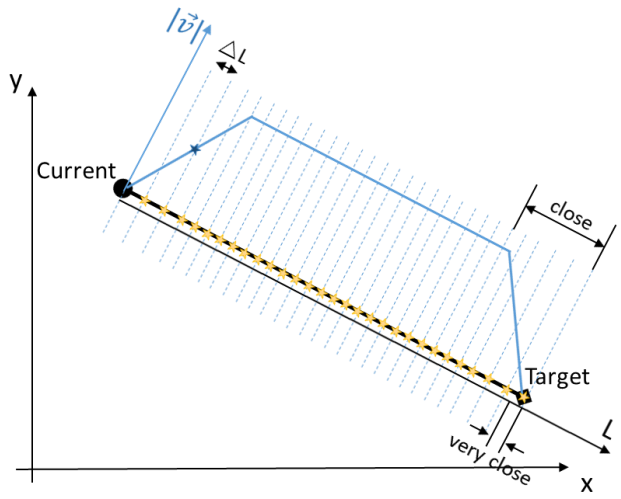
4) *En el slot para el botón "Connect with controllers", ejecutar vp.Get( ) para obtener la posición y velocidad actual, y presentarla en los controles del cuadro "Current" (serán necesarias señales/slots, ya que vp enviará mensajes GET a las controladoras, que no contestarán de forma inmediata).*

5) Crear una clase MyPointWithSpeed con los campos siguientes (public): x, y, vx, vy

6) Añadir un QVector<MyPointWithSpeed> points; a la clase MyWindow, y un entero iCurrent.

7) En el slot para el botón "Go Pos":

- Crear valores para points, de forma que se obtengan puntos intermedios en la trayectoria cada Tm:



$\vec{v}_{unit}$= vector unitario desde Current hasta Target → vux, vuy

pos_current_x = start point.x; pos_current_y=start_point.y;
speed_current = 0;
L_current=0; L_remaining=Distance(start_point,target_point);
state=ACCELERATING;
points.clear();

Para cada Tm:

state==ACCELERATING → speed_new = speed_cur + acc_max*Tm
if (speed_new >= speed_max) → state=MAX_SPEED
state== MAX_SPEED → speed_new = speed_cur
if (**close to target**) → state=DECELERATING
state== DECELERATING → speed_new = speed_cur - dec_max*Tm
if (**very close to target**) → state=STOPPED
state== STOPPED → speed_new = 0
△L= Tm*(speed_cur + speed_new)/2;
L_current += △L; L_remaining -= △L;

pos_new = MyPointWithSpeed con :
pos_new.vx=speed_new * vux
pos_new.vy=speed_new * vuy
pos_new.x = pos_current_x + vux*△L
pos_new.y = pos_current_y + vuy*△L

Añadir pos_new a la lista points

Actualizar para siguiente iteración: speed_cur = speed_new
pos_current_x = pos_new.x , pos_current_y= pos_new.y

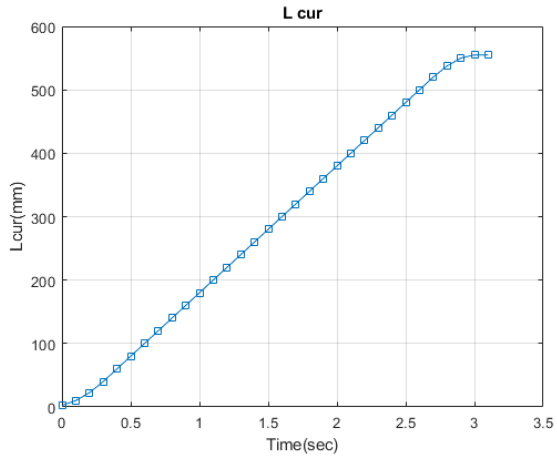
Escribir en archivo de texto la lista de puntos, cargar en Matlab, comprobar gráficamente.

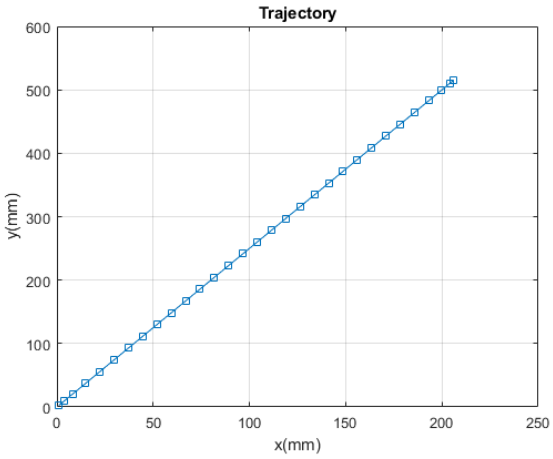
```
state  L_cur    L_rem     speed       x        y        vx       vy
0 ,    2.50 ,  536.02 ,   50.00,     0.93 ,    2.32 ,   18.57 ,   46.42
0 ,   10.00 ,  528.52 ,  100.00,     3.71 ,    9.28 ,   37.14 ,   92.85
0 ,   22.50 ,  516.02 ,  150.00,     8.36 ,   20.89 ,   55.71 ,  139.27
1 ,   40.00 ,  498.52 ,  200.00,    14.86 ,   37.14 ,   74.28 ,  185.70
1 ,   60.00 ,  478.52 ,  200.00,    22.28 ,   55.71 ,   74.28 ,  185.70
1 ,   80.00 ,  458.52 ,  200.00,    29.71 ,   74.28 ,   74.28 ,  185.70
1 ,  100.00 ,  438.52 ,  200.00,    37.14 ,   92.85 ,   74.28 ,  185.70
...
1 ,  480.00 ,   58.52 ,  200.00,   178.27 ,  445.67 ,   74.28 ,  185.70
1 ,  500.00 ,   38.52 ,  200.00,   185.70 ,  464.24 ,   74.28 ,  185.70
2 ,  520.00 ,   18.52 ,  200.00,   193.12 ,  482.81 ,   74.28 ,  185.70
2 ,  537.50 ,    1.02 ,  150.00,   199.62 ,  499.06 ,   55.71 ,  139.27
2 ,  550.00 ,  -11.48 ,  100.00,   204.26 ,  510.66 ,   37.14 ,   92.85
3 ,  555.00 ,  -16.48 ,    0.00,   206.12 ,  515.30 ,    0.00 ,    0.00
```

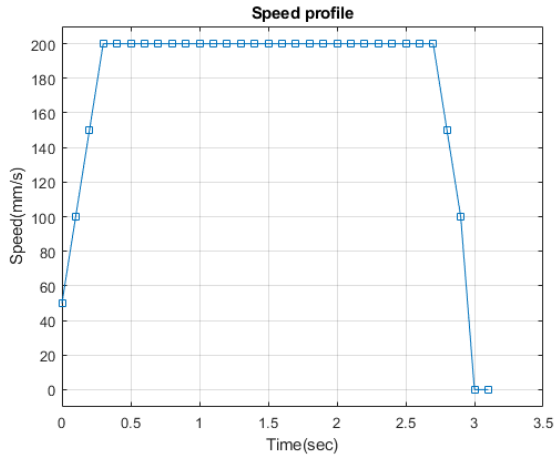
Copiar los datos al portapapeles (sin la línea de cabecera).

En Matlab:

```
>> d=importdata('-pastespecial');
>> state=d(:,1);  L_cur=d(:,2); …
>> t=(0:length(state)-1)*Tm_sec;
>> figure; plot(t,L_cur,'-s'); xlabel('Time(sec)'); ylabel('Lcur(mm)'); title('L cur');
>> figure; plot(x,y,'-s') ; xlabel('x(mm)'); ylabel('y(mm)'); title('Trajectory');
```







Tras la comprobación, una vez generada la lista de puntos:

- *Lanzar temporizador cada Tm, comenzar con iCurrent=0.*

8) En el slot para el temporizador:

- *Establecer en vp la velocidad points[iCurrent].vx, points[iCurrent].vy*
- *Solicitar posición y velocidad actuales a vp , para actualizar controles "current"*
- *Incrementar iCurrent (mientras iCurrent < points.size( ))*