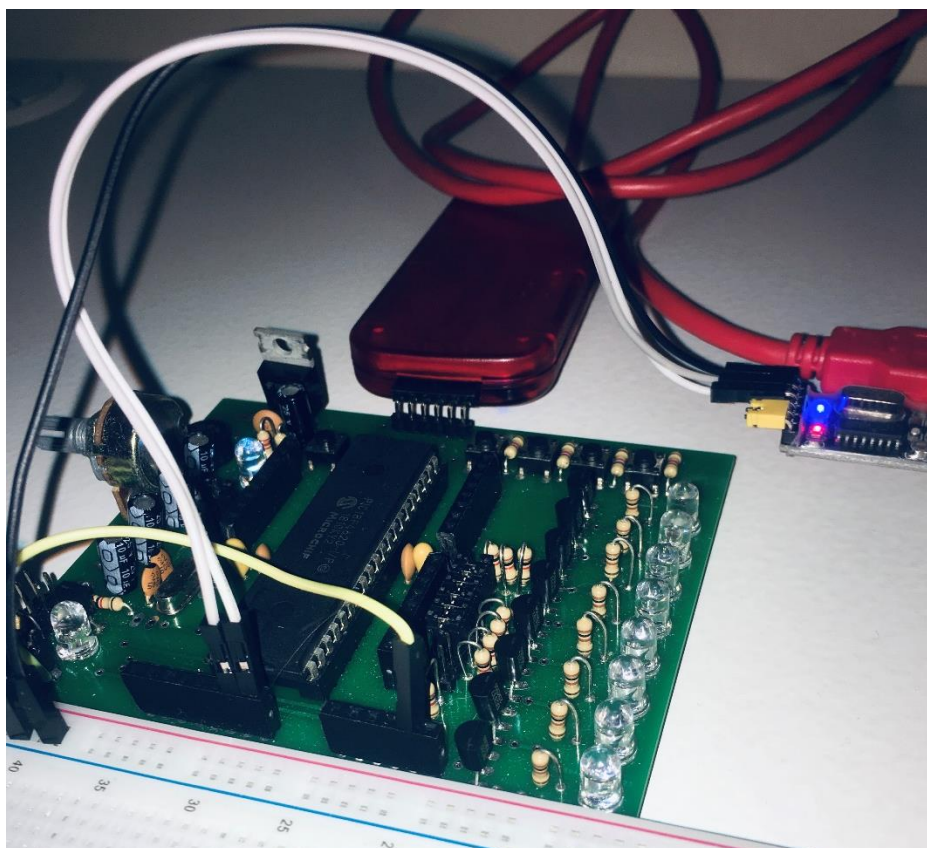




**Universidad Católica Argentina**

*Facultad de Ingeniería y Ciencias Agrarias - Carrera de Ing. Electrónica*

## **CONTROL DISCRETO**



<b>Registro</b>	<b>Apellido y Nombre</b>
15-162138-5	Burs, Trinidad

### **CORRECCIÓN**

<b>Docente</b>	<b>Fecha</b>	<b>Observaciones</b>
Vecchio, Ricardo		

### **APROBACIÓN:**

## DESARROLLO

El desarrollo del examen se basa en el ejemplo propuesto de péndulo invertido que se encuentra en: <https://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=ControlStateSpace>

El espacio de estados de la planta y el observador y el lazo de realimentacion se implementan de manera virtual (dentro del PIC18F4620).

El sistema no tendra errores de modelado ya que se define al observador con las mismas matrices A y B con las cuales se representa a la planta. Esto no sucede si la planta es real. Tampoco hay perturbaciones ni ruidos de observacion debido a que no se utilizan sensores. Teniendo en cuenta estas observaciones concluimos que se puede aplicar el principio de separacion que establece que los autovalores del observador y del sistema a lazo cerrado pueden diseñarse independientemente.

A pesar de esto, se decide situar los polos del observador para que estos sean 4 veces mas rapidos que los polos de la planta (mas adelante se concluye que esto no es posible cuando se trara de una planta digital, pero por ahora se partira de esta primicia).

La modelizacion de la planta se obtiene directamente de los datos del ejemplo. Se realiza una modificacion en la matriz C: se asigna la matriz identidad a C. En una aplicación real esto significa que se poseen 4 sensores, uno por variable de salida. Esto no se suele realizar ya que economicamente no es una buena opcion y muchas veces no se pueden medir todas las variables. No obstante, se le asigna a C ese valor para poder comparar la salida de la planta con la salida del bloque observador de manera mas precisa. Tambien se adaptara la matriz D (para no tener problemas de calculo numerico por su tamaño), aunque esto no tendra mucha relevancia en la implementacion en el microcontrolador ya que esta compuesta unicamente por ceros.

Las cuatro variables de salida del espacio de estados de la planta son:

$$\begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix}$$

*x: desplazamiento del carrito en metros (posición)*

*φ: ángulo de desplazamiento del péndulo en radianes*

Se comienza planteando el sistema en MatLab y posteriormente simulandolo en Simulink.

Se realizan las verificaciones de controlabilidad y observabilidad (el sistema será controlable y observable).

Para lograr un rise time menor a 0.5 segundos los polos de la planta se ubicaron según el criterio ITAE. Se fueron probando distintos valores de  $\omega_o$  hasta que se encontró el ideal.

También se encuentra que el sistema es inestable a lazo abierto (pero es estable a lazo cerrado). Esto se puede concluir observando los autovalores de A (uno de los polos de A esta en el semieje positivo):

```
>> eig(A)

ans =

         0
    -0.1428
   -5.6041
    5.5651
```

Imagen 1. Autovalores de A

Además se ve que una de las variables es inherentemente integradora. *Observación:* el orden en el que se despliegan los autovalores de A no corresponde al orden en el cual se presentan las magnitudes físicas. A su vez, los autovalores de A son los polos del sistema a lazo abierto

Para escalar la entrada y hacer que la posición en su estado estacionario tienda al valor de referencia, se debe redefinir C y D momentáneamente para poder utilizar “rscale”. Esto se debe a que solo se puede escalar en función de una variable, el resto se adapta a la constante que se elija para la otra (todas las variables están conectados por lo que no se puede definir el valor final de todas simultáneamente).

```
>> C

C =

    1    0    0    0
    0    1    0    0
    0    0    1    0
    0    0    0    1

>> C_modificada

C_modificada =

    1    0    0    0
```

Imagen 2. Matriz C y matriz C modificada

Si no se modificase C, estaría indicando que todas las variables deben tender al valor de referencia. Esto no es posible.

Luego se convierte el sistema analógico a discreto (esto es necesario para poder implementarlo dentro del microcontrolador). Para mapear los polos del plano continuo al discreto:

```
pd= exp(p*Ts)
```

Imagen 3. Mapeo de polos en el plano continuos al plano discreto

Para escalar la salida se hace uso de la función “dcgain”. Esta función de Matlab provee el valor al que estabilizan todas las variables cuando se las aplica una entrada continua de valor 1.

Una vez que se tiene todo esto planteado se procedo a construir el observador. El observador debe ser al menos 4 veces más rápido que el sistema original, luego la parte real de sus polos debe ser al menos 4 veces mayor (en modulo) a los polos del sistema.

```
po = p*4;
```

Imagen 4. Ubicación de los polos del observador

También se puede hacer: `po = real(p)*4` pero de la otra manera también se los ubica según ITAE. Ahora se procede a discretizar el observador:

```
Tso=Ts/4;
[Aod,Bod,Cod,Dod] = c2dm(Ao,Bo,Co,Do,Tso,'zoh');
```

Imagen 5. Discretización del observador

Es importante remarcar que Bo y Do no son propiamente las matrices del observador. Se recurre a un artificio matemático para poder utilizar el bloque de espacio de estado que se encuentra en Simulink.

```

po = p*4;
L=(place(A',C',po))';
Ao=A-L*C;
Bo=[B,L];
Co=eye(size(A));
Do=zeros(size(Bo));

```

Imagen 6. Construcción del Observador

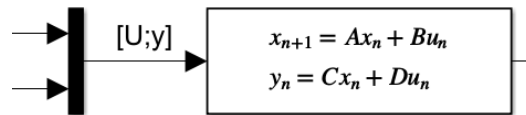


Imagen 7. Representación del observador en Simulink

Se procede a simular el sistema. Se observa que la salida oscilara.

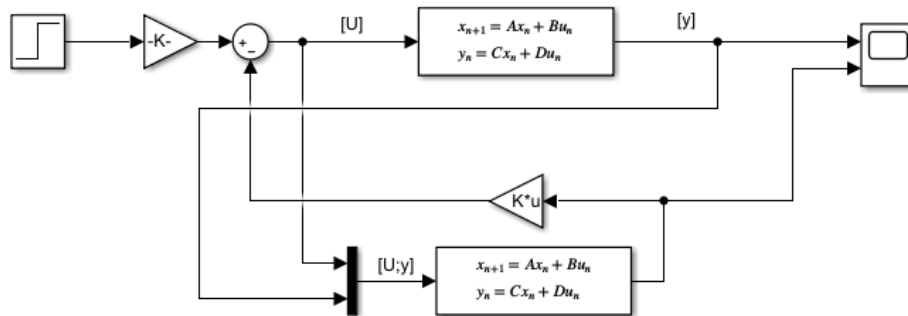


Imagen 8. Implementación del sistema en Simulink

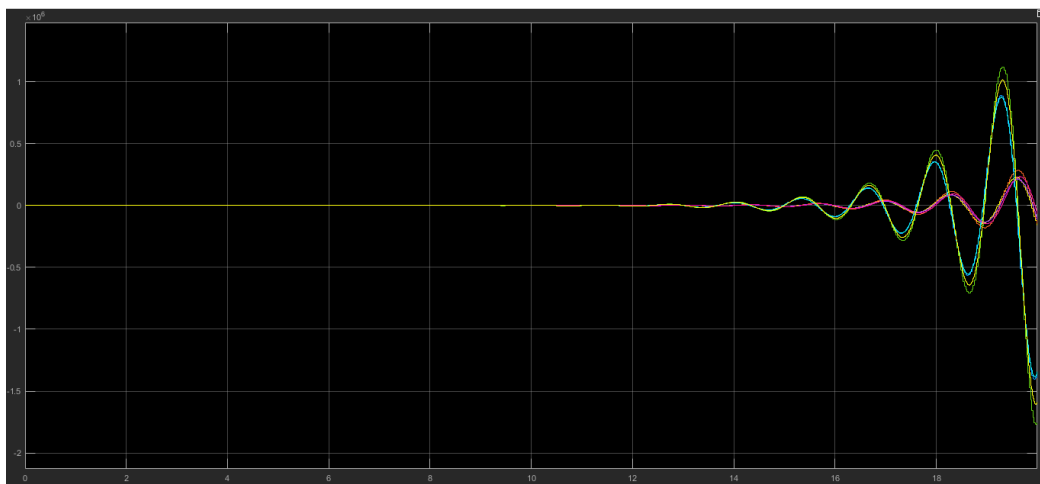


Imagen 9. Oscilación producto de utilizar la planta discretizada con  $T_s$

Esto se debe a que se plantea el lazo con el sistema discretizado con  $T_s$  y el observador muestreando cada  $T_{so}$ .  $T_s$  debe ser menor o igual a  $T_{so}$ . Si esto no sucede el observador registra que aunque cambia la entrada a la planta ( $U$ ), la salida ( $Y$ ) de este no cambia por periodos “largos” luego hace una mala estimación. Esto sucede porque la planta no es analógica.

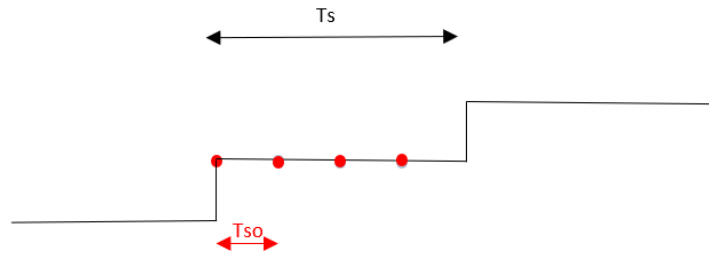


Imagen 10.  $T_s$  Vs  $T_{so}$

Lo que se hace para solucionar esto es discretizar el espacio de estados de la planta con  $T_{so}$  (tiempo de muestreo del observador). En este punto se crea el espacio de estados discreto “fast”. Este es el que se utiliza para la implementación en hardware. Luego se concluye que el observador es igual de rápido que la planta (los polos del observador no serán 4 veces más rápidos que los de la planta).

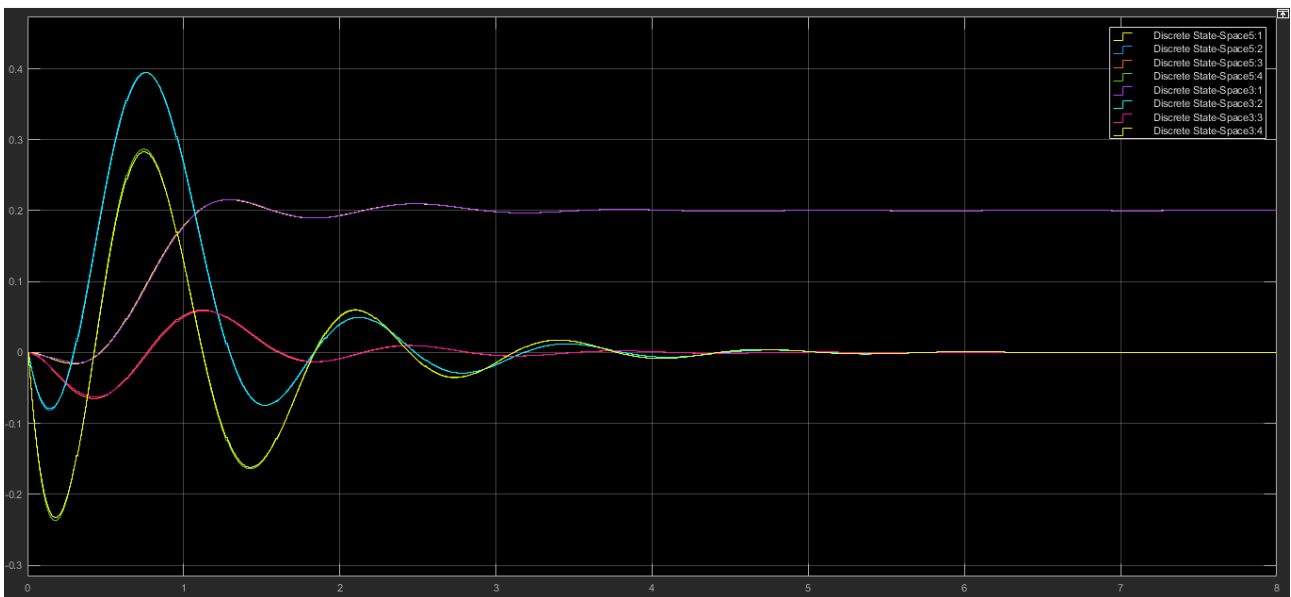


Imagen 11. Salida de la planta y del observador usando la versión “Fast” de la planta

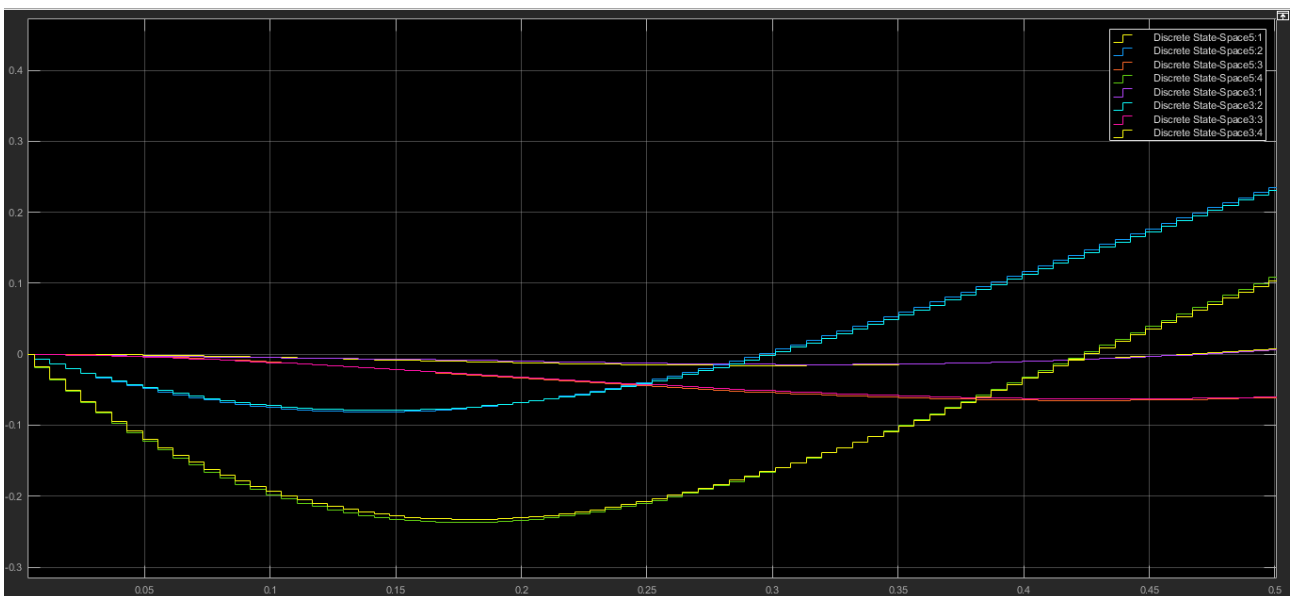


Imagen 12. Variación entre la salida de la planta y la salida del observador

## IMPLEMENTACION EN HARDWARE

Se comienza señalando que en el compilador XC8 el “double” tiene una precisión de 32 bits.

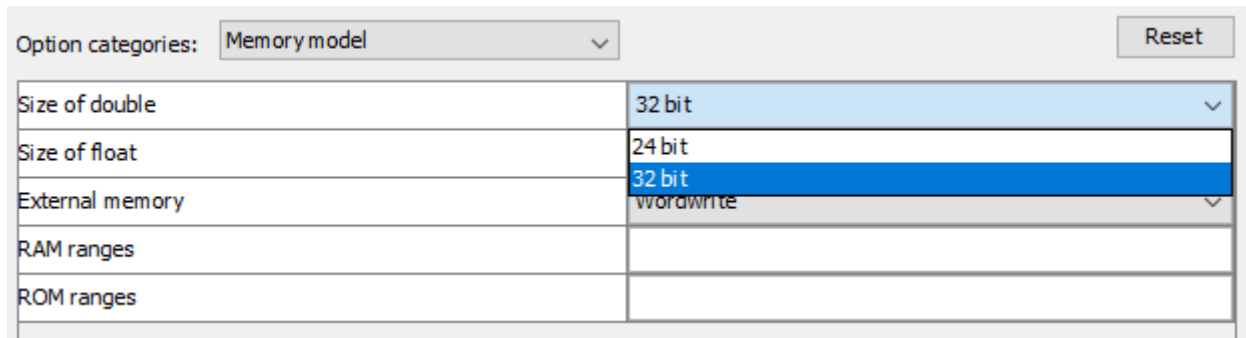


Imagen 13. Precisión de un “double” en MPLAB utilizando el compilador XC8

Las ecuaciones que se utilizan para representar el espacio de estado son las siguientes:

$$X[n + 1] = A \cdot X[n] + B \cdot U[n] \quad (1)$$

$$Y[n] = C \cdot X[n] \quad (2)$$

$$X[n] = X[n + 1] \quad (3)$$

Es muy importante respetar el orden de ejecución de estas ecuaciones cuando se las adapta dentro del microcontrolador.

Como C es la matriz identidad la ecuación (2) se la puede describir como:

$$Y[n] = X[n] \quad (2)$$

Esto permite disminuir el tiempo de ejecución.

Se parte con las siguientes condiciones iniciales:

$$Y[n] = 0$$

$$U[n] = ref * Nbard_{fast}$$

$$X[n] = 0$$

Las matrices que se utilizan para representar la planta, el observador y la ganancia de lazo son aquellas que fueron discretizadas. Se usan las matrices FAST para la planta.

Para realizar la suma y multiplicación de matrices se construyen funciones propias.

```
// Declaracion funciones complementarias
void matrix_multiply_c1_c1(volatile double M1[][1],volatile double M2[][1],volatile double res[][1]);
void matrix_multiply_c4_c1(volatile double M1[][4],volatile double M2[][1],volatile double res[][1]);
void matrix_multiply_c5_c1(volatile double M1[][5],volatile double M2[][1],volatile double res[][1]);
void matrix_sum_c1(volatile double M1[][1],volatile double M2[][1],volatile double res[][1]);
```

Imagen 14. Declaración de funciones de multiplicación y suma de matrices

```

void matrix_multiply_c1_c1(volatile double M1[][1],volatile double M2[][1],volatile double res[][1]){
    //El numero de columnas de M1 debe coincidir con el numero de filas de M2
    int i=0,j=0,k=0;
    for(i=0;i<fM1;i++){
        for(j=0;j<1;j++){
            res[i][j]=0;
            for(k=0;k<1;k++){
                res[i][j]+=M1[i][k]*M2[k][j];
            }
        }
    }
}

```

Imagen 15. Ejemplo de función de multiplicación de matrices

Se crean numerosas funciones ya que es necesario definir al menos una de las dimensiones (filas o columnas) de las matrices utilizadas. Una vez declarado este valor no se puede modificar. En todos los casos se define el numero de columnas y se deja variable el numero de filas de la matriz M1 (fM1 es una variable global). Se puede realizar una funcion generica trabajando con punteros pero visualizar su funcionamiento y accionar es mas engorroso. Recordar: es muy importante el orden en la multiplicación de matrices. El número de columnas de M1 debe coincidir con el número de filas de M2. La matriz resultante de la multiplicación tiene el número de filas de M1 y el número de columnas de M2.

Antes de utilizar cualquier funcion de operación de matrices se recomienda asegurarse que fM1 tiene asignado el valor correcto:

```

//X[n+1]=A*X[n]+B*U[n]
fM1=4;matrix_multiply_c4_c1(A,X_0,aux_1); //A*X[n]
fM1=4;matrix_multiply_c1_c1(B,U,aux_2); //B*U[n]
fM1=4;matrix_sum_c1(aux_1,aux_2,X_1);

```

Imagen 16. Implementación de la ecuación ' $X[n+1]=A \cdot X[n]+B \cdot U[n]$ ' utilizando las funciones de multiplicación de matrices

Se crean variables auxiliares para almacenar productos intermedios.

Se decide que el cálculo del sistema no se realice en tiempo real. Esto se debe a que se deben realizar numerosas operaciones y el tiempo de ejecución de la rutina supera el tiempo cada el cual se debe realizar una interrupcion ( $T_{so} < T_{ejecucion\_rutina}$ ). Esto no sera una simulación pero una emulación.

En el simulink se observa que el sistema estabiliza cerca de los 8,5 segundos. Si se hace  $8,5seg/T_{so}$  se determina que se necesitan tomar cerca de 1400 muestras.

```
while(count<muestras)
```

Imagen 17. Numero de muestras que se toman

Para visualizar los datos de utilizara el "Data Visualizer". Se utiliza un protocolo que tiene la siguiente forma:

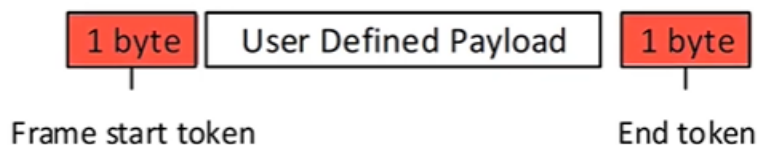


Imagen 18. Protocolo usado para la comunicación entre la UART y el "Data Visualizer"

Mi "Frame Start Token" es: 0x03. Mi "End Token" es: 0xFC (el complemento de 0x03). En el medio se colocan todos los datos que se quieren enviar. Todos los datos que se envian deben estar contenidos en 8 bits y el "Data Visualizer" se encarga de agruparlos según el tipo dato que se le indica que va a recibir.

```

ptrPlanta=(char *) & Y[0][0];
UART_Write(0x03);
//SALIDA PLANTA
for (int x=0;x<sizeof(Y[0][0]);x++){
    UART_Write(*(ptrPlanta++));
};
ptrPlanta=(char *) & Y[1][0];
for (int x=0;x<sizeof(Y[1][0]);x++){
    UART_Write(*(ptrPlanta++));
};
ptrPlanta=(char *) & Y[2][0];
for (int x=0;x<sizeof(Y[2][0]);x++){
    UART_Write(*(ptrPlanta++));
};
ptrPlanta=(char *) & Y[3][0];
for (int x=0;x<sizeof(Y[3][0]);x++){
    UART_Write(*(ptrPlanta++));
};
UART_Write(0xFC);

```

*Imagen 19. Transmisión de datos al “Data Visualizer”*

El corazon del programa esta en la ejecución de la siguiente funcion.



```

void plantaPLUSobservador() {
    //PLANTA
    //X[n+1]=A*X[n]+B*U[n]
    fM1=4;matrix_multiply_c4_c1(A,X_0,aux_1); //A*X[n]
    fM1=4;matrix_multiply_c1_c1(B,U,aux_2); //B*U[n]
    fM1=4;matrix_sum_c1(aux_1,aux_2,X_1);

    //Como C es la matriz identidad se puede dar una asignacion directa (reduzco los tiempos de ejecutado)
    //Si no fuese la matriz identidad, debo reemplazar por: fM1=4;matrix_multiply_c4_c1(C,X_0,Y);
    Y[0][0]=X_0[0][0];
    Y[1][0]=X_0[1][0];
    Y[2][0]=X_0[2][0];
    Y[3][0]=X_0[3][0];

    //X[n]=X[n+1] (Para la proxima vez que se invoco a la planta)
    X_0[0][0]=X_1[0][0];
    X_0[1][0]=X_1[1][0];
    X_0[2][0]=X_1[2][0];
    X_0[3][0]=X_1[3][0];

    //OBSERVADOR
    //"Multiplexo" la entrada al observador para poder usar la representacion del espacio de estados usada en simulink
    inputObs[0][0]=U[0][0];
    inputObs[1][0]=Y[0][0];
    inputObs[2][0]=Y[1][0];
    inputObs[3][0]=Y[2][0];
    inputObs[4][0]=Y[3][0];

    //Xo[n+1]=A*Xo[n]+B*inputObs[n]
    fM1=4;matrix_multiply_c4_c1(Ao,Xo_0,aux_4); //Ao*Xo[n]
    fM1=4;matrix_multiply_c5_c1(Bo,inputObs,aux_5); //Bo*inputObs[n]
    fM1=4;matrix_sum_c1(aux_4,aux_5,Xo_1);

    //Como Co es la matriz identidad se puede dar una asignacion directa (reduzco los tiempos de ejecutado)
    //Si no fuese la matriz identidad, debo reemplazar por: fM1=4;matrix_multiply_c4_c1(Co,Xo_0,Yo);
    Yo[0][0]=Xo_0[0][0];
    Yo[1][0]=Xo_0[1][0];
    Yo[2][0]=Xo_0[2][0];
    Yo[3][0]=Xo_0[3][0];

    //Xo[n]=Xo[n+1] (Para la proxima vez que se invoco al observador)
    Xo_0[0][0]=Xo_1[0][0];
    Xo_0[1][0]=Xo_1[1][0];
    Xo_0[2][0]=Xo_1[2][0];
    Xo_0[3][0]=Xo_1[3][0];

    //REALIMENTACION
    //U[n]=r[n]*Nbard-Kd*Yo[n]=r[n]*Nbard+(-Kd)*Yo[n]
    fM1=1;matrix_multiply_c4_c1(nKd,Yo,aux_3); //-Kd*X[n]
    fM1=1;matrix_sum_c1(ref_by_Nbard,aux_3,U);
}

```

*Imagen 20. Rutina de simulación del sistema realimentado con observador*

Se debe tener mucho cuidado cuando se establecen las dimensiones de todas las matrices.

Se supone que el valor de referencia es un escalon de 0.2 de amplitud.

Antes de implementar esto en MPLABX se “simuló” el sistema realimentado en excel para verificar que el orden de ejecucion y las funciones a utilizar se ejecutaran de manera correcta.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1						X <sub>n</sub>	0			U	-0.65883											
2		1	0.006142	5.04E-05	1.03E-07		0															
3		0	0.938883	0.016418	5.04E-05		0															
4		0	-8.58E-06	1.000589	0.006146		0															
5		0	-0.00273	0.191626	1.000589																	
6																						
7		B																				
8		3.43E-05				X <sub>n+1</sub>	-2.26E-05	-9.04E-05	-2.02E-04	-3.56E-04	-5.49E-04	-7.79E-04	-1.04E-03	-1.34E-03	-1.67E-03	-2.02E-03	-2.41E-03	-2.81E-03	-3.24E-03	-3.68E-03	-4.15E-03	-4.62E-03
9		0.01167					-0.00736	-0.014708	-0.02169	-0.02829	-0.03452	-0.04036	-0.04584	-0.05094	-0.05568	-0.06006	-0.06407	-0.06773	-0.07103	-0.07398	-0.07659	-0.07885
10		8.58E-05					-5.7E-05	-0.000226	-0.00051	-0.00089	-0.00137	-0.00195	-0.00262	-0.00337	-0.0042	-0.0051	-0.00607	-0.00711	-0.00821	-0.00936	-0.01056	-0.01182
11		0.027922					-0.0184	-0.036792	-0.0543	-0.07091	-0.08664	-0.10151	-0.11554	-0.12875	-0.14117	-0.1528	-0.16368	-0.17382	-0.18323	-0.19193	-0.19994	-0.20727
12							0	-2.26E-05	-9E-05	-0.0002	-0.00036	-0.00055	-0.00078	-0.00104	-0.00134	-0.00167	-0.00202	-0.00241	-0.00281	-0.00324	-0.00368	-0.00415
13							0	-0.007357	-0.01471	-0.02169	-0.02829	-0.03452	-0.04036	-0.04584	-0.05094	-0.05568	-0.06006	-0.06407	-0.06773	-0.07103	-0.07398	-0.07659
14							0	-5.65E-05	-0.00023	-0.00051	-0.00089	-0.00137	-0.00195	-0.00262	-0.00337	-0.0042	-0.0051	-0.00607	-0.00711	-0.00821	-0.00936	-0.01056
15		1	0	0	0		0	-0.018396	-0.03679	-0.0543	-0.07091	-0.08664	-0.10151	-0.11554	-0.12875	-0.14117	-0.1528	-0.16368	-0.17382	-0.18323	-0.19193	-0.19994
16		0	1	0	0		0	-0.018396	-0.03679	-0.0543	-0.07091	-0.08664	-0.10151	-0.11554	-0.12875	-0.14117	-0.1528	-0.16368	-0.17382	-0.18323	-0.19193	-0.19994
17		0	0	1	0		0	-0.018396	-0.03679	-0.0543	-0.07091	-0.08664	-0.10151	-0.11554	-0.12875	-0.14117	-0.1528	-0.16368	-0.17382	-0.18323	-0.19193	-0.19994
18		0	0	0	1		0	-0.018396	-0.03679	-0.0543	-0.07091	-0.08664	-0.10151	-0.11554	-0.12875	-0.14117	-0.1528	-0.16368	-0.17382	-0.18323	-0.19193	-0.19994
19							0	-0.018396	-0.03679	-0.0543	-0.07091	-0.08664	-0.10151	-0.11554	-0.12875	-0.14117	-0.1528	-0.16368	-0.17382	-0.18323	-0.19193	-0.19994
20							0	-0.018396	-0.03679	-0.0543	-0.07091	-0.08664	-0.10151	-0.11554	-0.12875	-0.14117	-0.1528	-0.16368	-0.17382	-0.18323	-0.19193	-0.19994
21							0	-0.018396	-0.03679	-0.0543	-0.07091	-0.08664	-0.10151	-0.11554	-0.12875	-0.14117	-0.1528	-0.16368	-0.17382	-0.18323	-0.19193	-0.19994
22							0	-0.018396	-0.03679	-0.0543	-0.07091	-0.08664	-0.10151	-0.11554	-0.12875	-0.14117	-0.1528	-0.16368	-0.17382	-0.18323	-0.19193	-0.19994
23							0	-0.018396	-0.03679	-0.0543	-0.07091	-0.08664	-0.10151	-0.11554	-0.12875	-0.14117	-0.1528	-0.16368	-0.17382	-0.18323	-0.19193	-0.19994
24							0	-0.018396	-0.03679	-0.0543	-0.07091	-0.08664	-0.10151	-0.11554	-0.12875	-0.14117	-0.1528	-0.16368	-0.17382	-0.18323	-0.19193	-0.19994
25							0	-0.018396	-0.03679	-0.0543	-0.07091	-0.08664	-0.10151	-0.11554	-0.12875	-0.14117	-0.1528	-0.16368	-0.17382	-0.18323	-0.19193	-0.19994
26							0	-0.018396	-0.03679	-0.0543	-0.07091	-0.08664	-0.10151	-0.11554	-0.12875	-0.14117	-0.1528	-0.16368	-0.17382	-0.18323	-0.19193	-0.19994
27							0	-0.018396	-0.03679	-0.0543	-0.07091	-0.08664	-0.10151	-0.11554	-0.12875	-0.14117	-0.1528	-0.16368	-0.17382	-0.18323	-0.19193	-0.19994
28							0	-0.018396	-0.03679	-0.0543	-0.07091	-0.08664	-0.10151	-0.11554	-0.12875	-0.14117	-0.1528	-0.16368	-0.17382	-0.18323	-0.19193	-0.19994
29							0	-0.018396	-0.03679	-0.0543	-0.07091	-0.08664	-0.10151	-0.11554	-0.12875	-0.14117	-0.1528	-0.16368	-0.17382	-0.18323	-0.19193	-0.19994
30							0	-0.018396	-0.03679	-0.0543	-0.07091	-0.08664	-0.10151	-0.11554	-0.12875	-0.14117	-0.1528	-0.16368	-0.17382	-0.18323	-0.19193	-0.19994
31							0	-0.018396	-0.03679	-0.0543	-0.07091	-0.08664	-0.10151	-0.11554	-0.12875	-0.14117	-0.1528	-0.16368	-0.17382	-0.18323	-0.19193	-0.19994
32							0	-0.018396	-0.03679	-0.0543	-0.07091	-0.08664	-0.10151	-0.11554	-0.12875	-0.14117	-0.1528	-0.16368	-0.17382	-0.18323	-0.19193	-0.19994
33							0	-0.018396	-0.03679	-0.0543	-0.07091	-0.08664	-0.10151	-0.11554	-0.12875	-0.14117	-0.1528	-0.16368	-0.17382	-0.18323	-0.19193	-0.19994
34							0	-0.018396	-0.03679	-0.0543	-0.07091	-0.08664	-0.10151	-0.11554	-0.12875	-0.14117	-0.1528	-0.16368	-0.17382	-0.18323	-0.19193	-0.19994
35							0	-0.018396	-0.03679	-0.0543	-0.07091	-0.08664	-0.10151	-0.11554	-0.12875	-0.14117	-0.1528	-0.16368	-0.17382	-0.18323	-0.19193	-0.19994
36							0	-0.018396	-0.03679	-0.0543	-0.07091	-0.08664	-0.10151	-0.11554	-0.12875	-0.14117	-0.1528	-0.16368	-0.17382	-0.18323	-0.19193	-0.19994

Imagen 21. Representación en Excel

## Salida Planta y Salida Observador

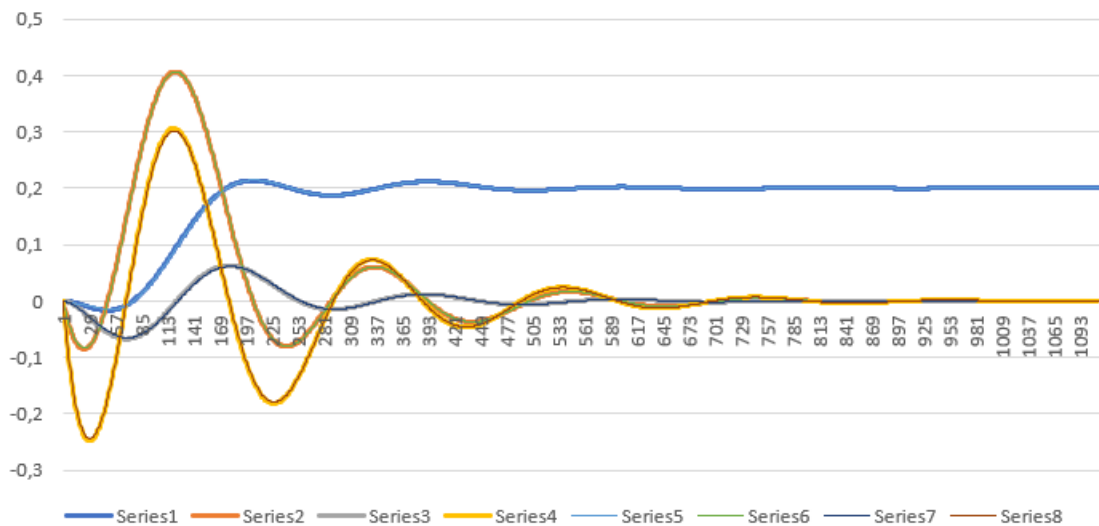


Imagen 22. Grafico obtenido a partir de la representación del sistema en Excel

Los datos se visualizan de la siguiente manera en el “Data Visualizer”:

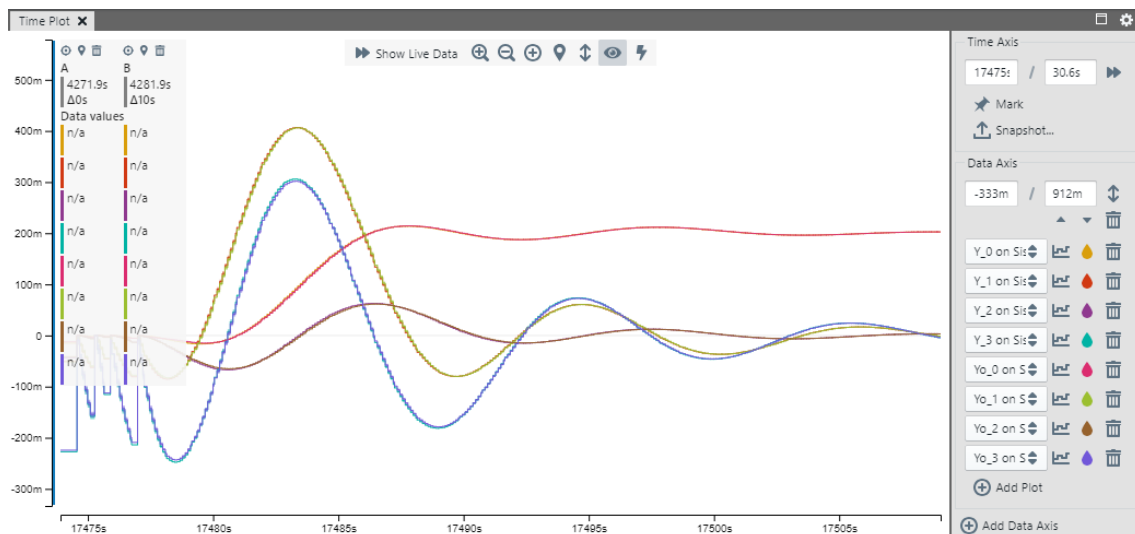


Imagen 23. Visualización de datos en el “Data Visualizer”

Se decide exportar los datos a un “.csv” para poder trabajarlos y visualizarlos de una manera más clara. Los datos de este archivo posteriormente se procesan con MatLab.

	A	B	C	D	E	F	G	H	I
1	timestam	Y_0	Y_1	Y_2	Y_3	Yo_0	Yo_1	Yo_2	Yo_3
2	27743,21	0	0	0	0	0	0	0	0
3	27743,26	-2,3E-05	-0,00736	-5,7E-05	-0,0184	-0,00039	-0,00721	-0,00032	-0,0179
4	27743,32	-9E-05	-0,0147	-0,00023	-0,0368	-0,00079	-0,0144	-0,00071	-0,0358
5	27743,37	-0,0002	-0,0217	-0,00051	-0,0543	-0,00119	-0,0212	-0,00117	-0,0529
6	27743,42	-0,00036	-0,0283	-0,00089	-0,0709	-0,00157	-0,0275	-0,0017	-0,0691
7	27743,47	-0,00055	-0,0345	-0,00137	-0,0866	-0,00195	-0,0335	-0,00229	-0,0844
8	27743,53	-0,00078	-0,0404	-0,00195	-0,102	-0,0023	-0,0392	-0,00294	-0,0989
9	27743,58	-0,00104	-0,0458	-0,00262	-0,116	-0,00265	-0,0444	-0,00365	-0,113
10	27743,63	-0,00134	-0,0509	-0,00337	-0,129	-0,00298	-0,0493	-0,00441	-0,126
11	27743,69	-0,00167	-0,0557	-0,0042	-0,141	-0,00329	-0,0538	-0,00523	-0,138
12	27743,74	-0,00202	-0,0601	-0,0051	-0,153	-0,00359	-0,058	-0,0061	-0,149
13	27743,79	-0,00241	-0,0641	-0,00607	-0,164	-0,00388	-0,0618	-0,00702	-0,16
14	27743,84	-0,00281	-0,0677	-0,00711	-0,174	-0,00416	-0,0653	-0,00799	-0,17
15	27743,9	-0,00324	-0,071	-0,00821	-0,183	-0,00444	-0,0685	-0,009	-0,179
16	27743,95	-0,00368	-0,074	-0,00936	-0,192	-0,0047	-0,0713	-0,0101	-0,187
17	27744	-0,00415	-0,0766	-0,0106	-0,2	-0,00496	-0,0738	-0,0111	-0,195
18	27744,06	-0,00462	-0,0788	-0,0118	-0,207	-0,00522	-0,076	-0,0123	-0,203
19	27744,11	-0,00511	-0,0808	-0,0131	-0,214	-0,00548	-0,0779	-0,0134	-0,209
20	27744,16	-0,00561	-0,0823	-0,0144	-0,22	-0,00574	-0,0795	-0,0146	-0,215

Imagen 24. Datos exportados del “Data Visualizer”

A continuación se presentan los gráficos obtenidos:

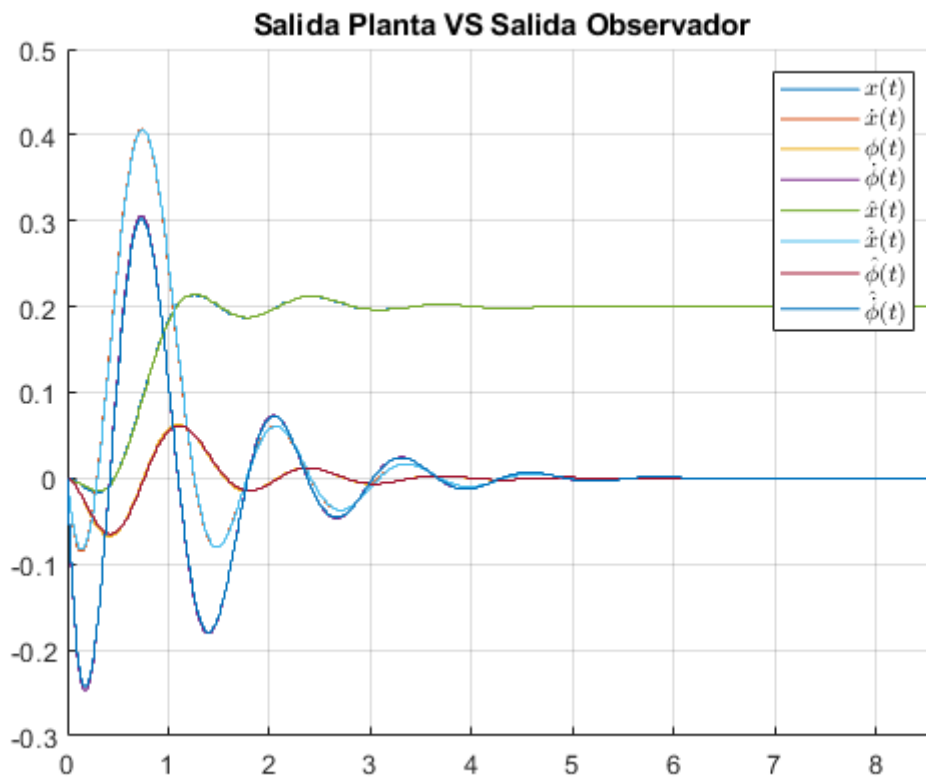


Imagen 25. PIC: Señal a la salida de la planta VS Señal a la salida del observador

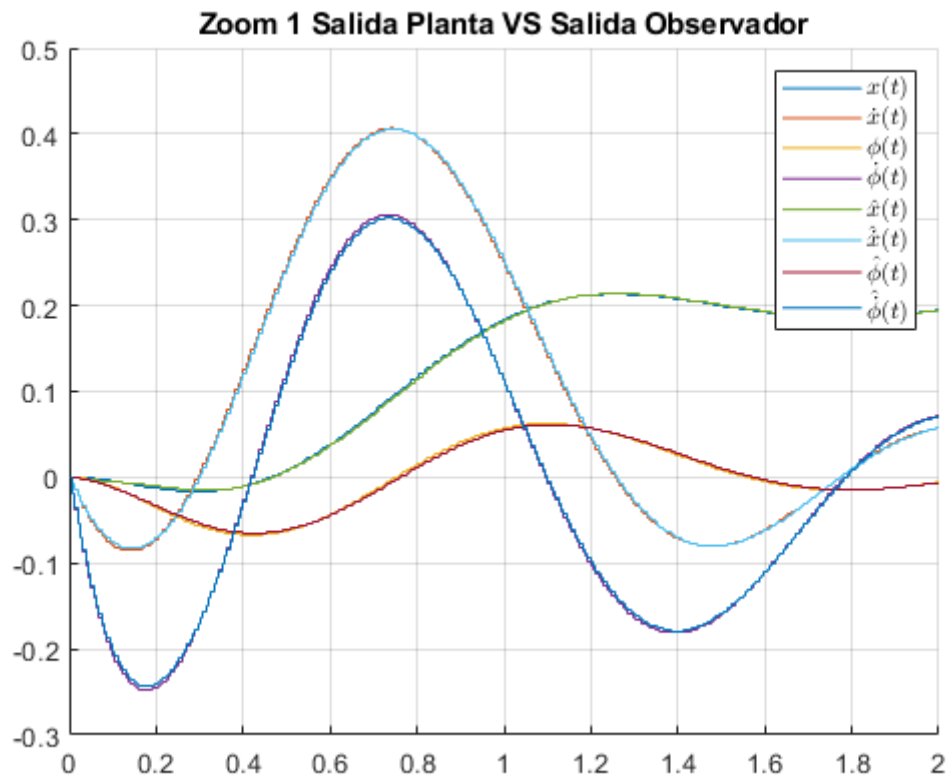


Imagen 26. PIC: Zoom sobre la señal a la salida de la planta VS señal a la salida del observador

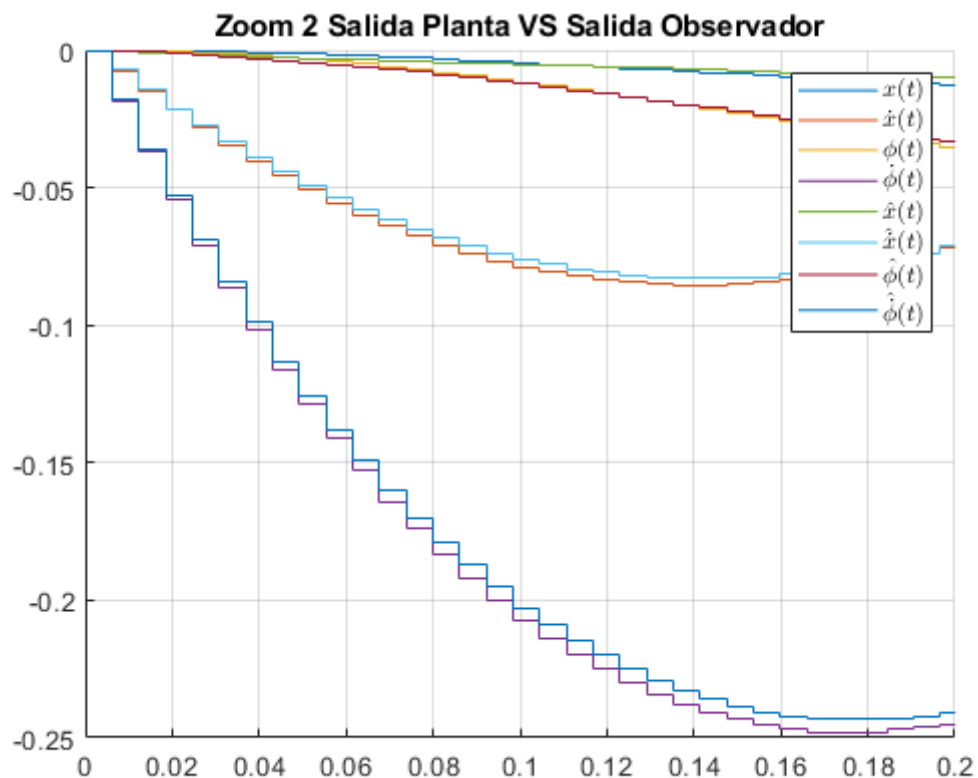


Imagen 27. PIC: Zoom 2 sobre la señal a la salida de la planta VS señal a la salida del observador

Se desea comparar las curvas obtenidas con el PIC con las simuladas utilizando Simulink. Para esto se exportan los datos de Simulink utilizando el bloque "To File" que genera un archivo ".mat" en la dirección indicada dentro del bloque.

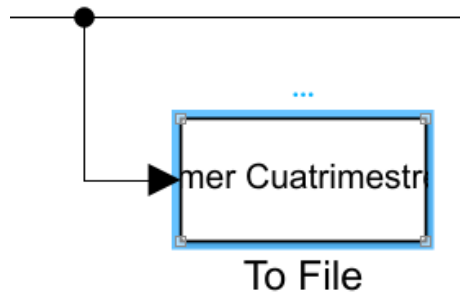


Imagen 28. Bloque generador de archivo '.mat' a partir de los datos de la señal

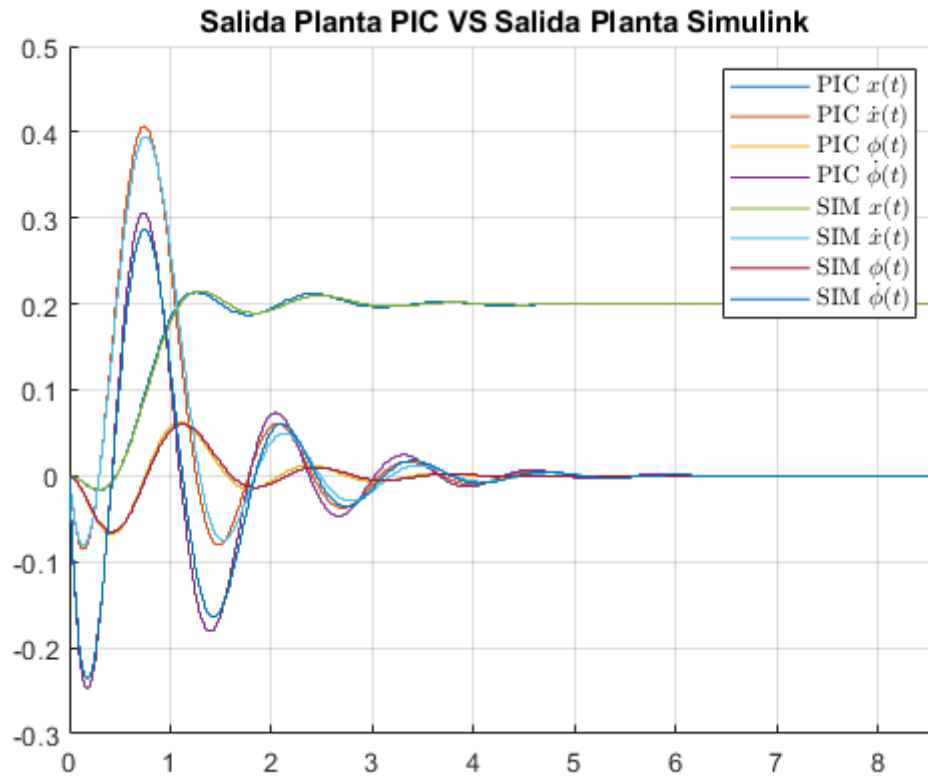


Imagen 29. Señal a la salida de la planta: PIC VS Simulink

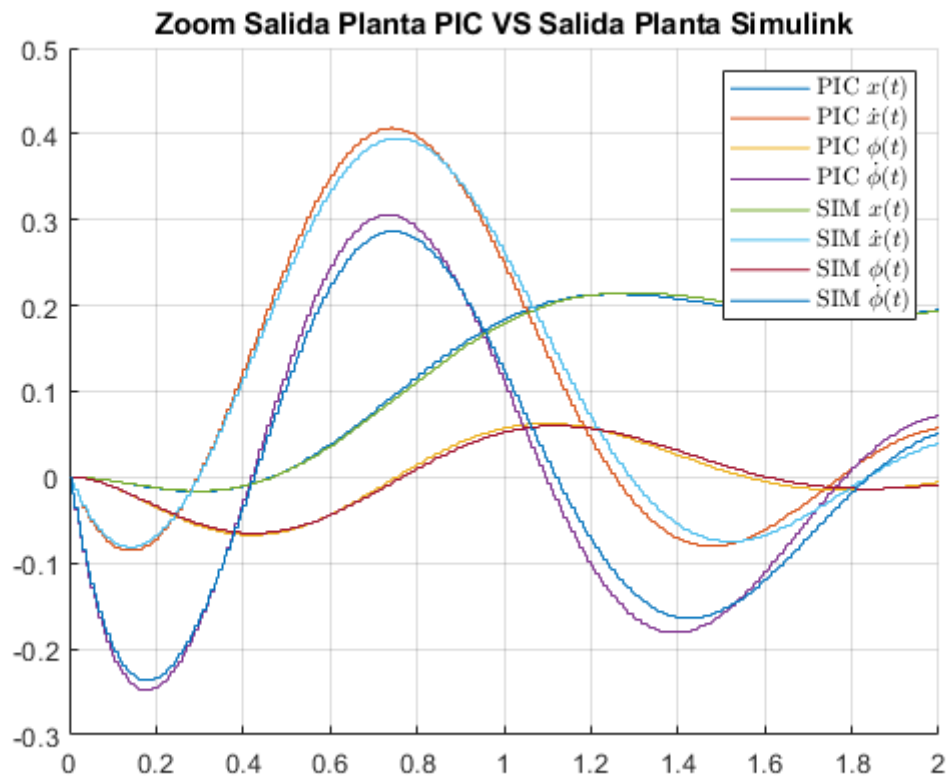


Imagen 30. Zoom señal a la salida de la planta: PIC VS Simulink

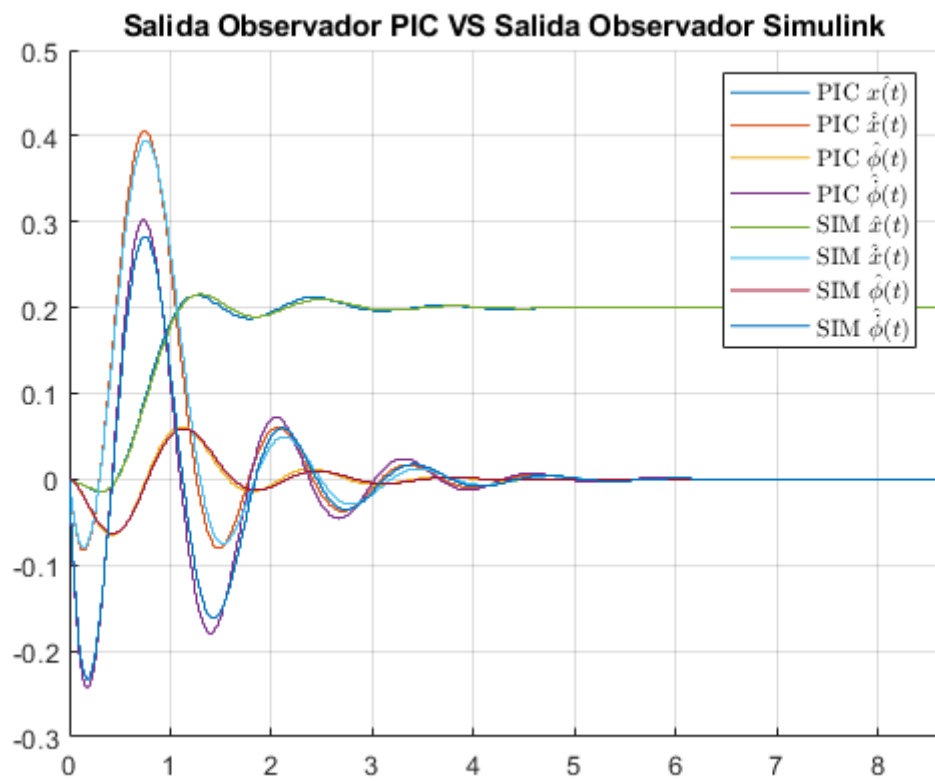


Imagen 31. Señal a la salida del observador: PIC VS Simulink

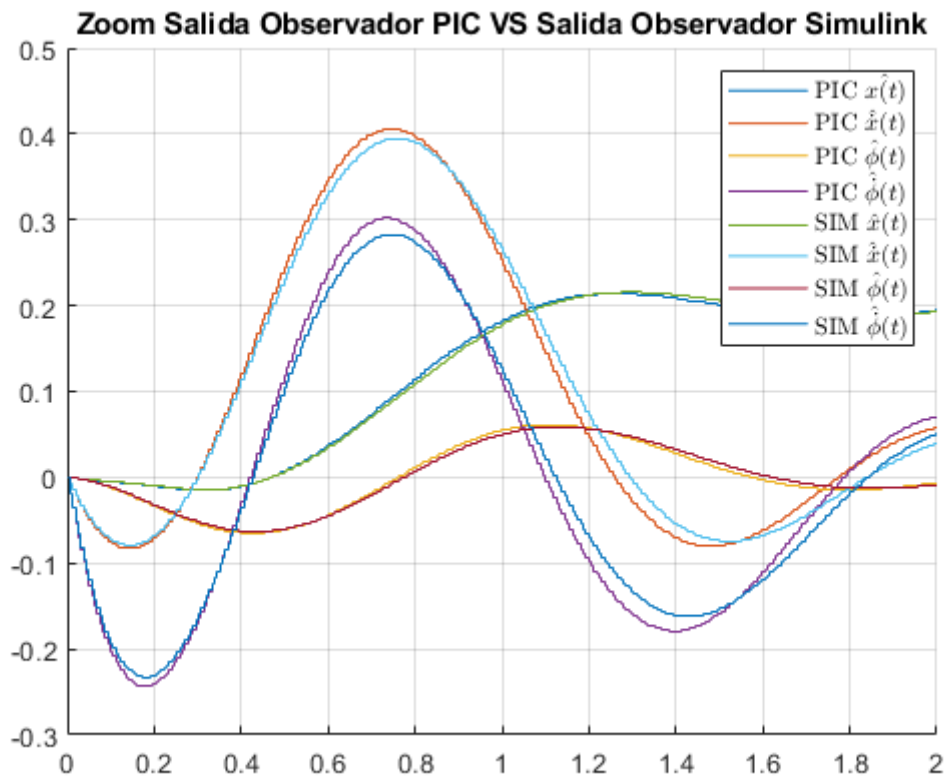


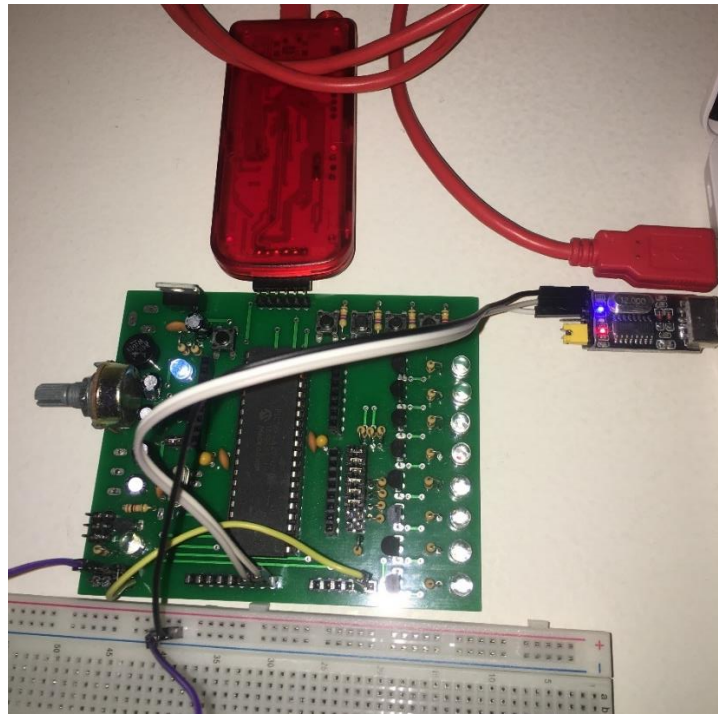
Imagen 32. Zoom señal a la salida del observador: PIC VS Simulink

Se observa que la salida estabiliza perfectamente en el valor esperado (0.2) y que las curvas son muy semejantes. Por otro lado, al ser una implementación virtual el ruido del sistema físico será muy bajo por lo que se concluye que las pequeñas discrepancias entre las señales de salida del PIC y de Simulink se deben a la reducción en la precisión numérica que se da cuando se pasa de un entorno a otro (MATLAB → MPLAB-XC8).

A continuación se incluyen imágenes de cómo fue implementado el sistema.



Imagen 33. Hardware Utilizado



*Imagen 34. Hardware Utilizado*



## PERTURBACIONES

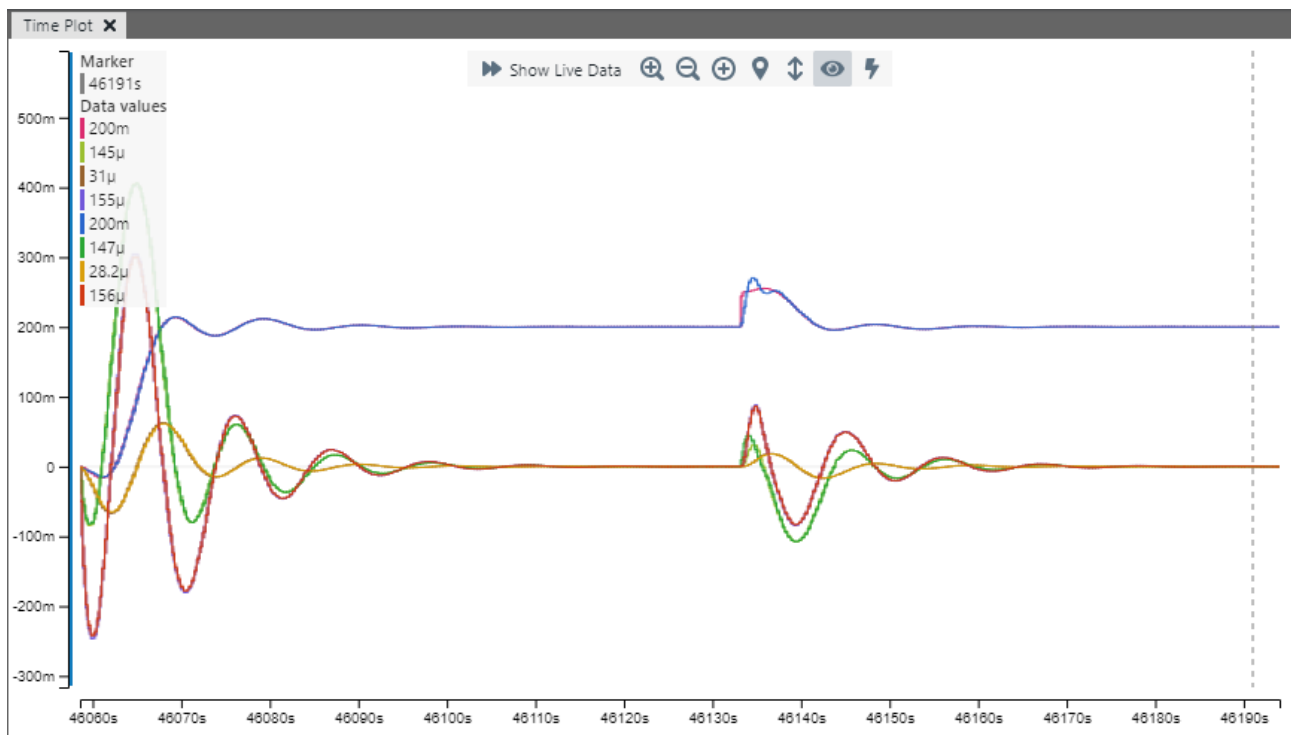
Se construye una nueva función llamada “plantaPLUSobservadorPLUSperturbacion();” que aplica una perturbacion constante a Y\_0 (posición). Esta es la variable inherentemente integradora del sistema.

Para aplicar la perturbacion se existe el numero de muestras. Se recomienda colocar 3000 (while(count<3000){...}). La perturbacion se coloca una vez que el sistema estabilizó.

```
Y[0][0]=X_0[0][0];
Y[1][0]=X_0[1][0];
Y[2][0]=X_0[2][0];
Y[3][0]=X_0[3][0];

if (count>1400){
    Y[0][0]+=0.05;
};
```

*Imagen 35. Perturbación aplicada a Y\_0*



*Imagen 36. Respuesta del sistema a la perturbación aplicada a Y\_0*

Aunque la perturbación sea sostenida, al ser una variable inherentemente integradora el sistema vuelve a sus valores de referencia.

Si aplicamos esta misma perturbacion a otra variable (el resto son no integradoras), el sistema no tiende a sus valores de referencia.

```
if (count>1400){
    Y[1][0]+=0.05;
};
```

*Imagen 37. Perturbación aplicada a Y\_1*

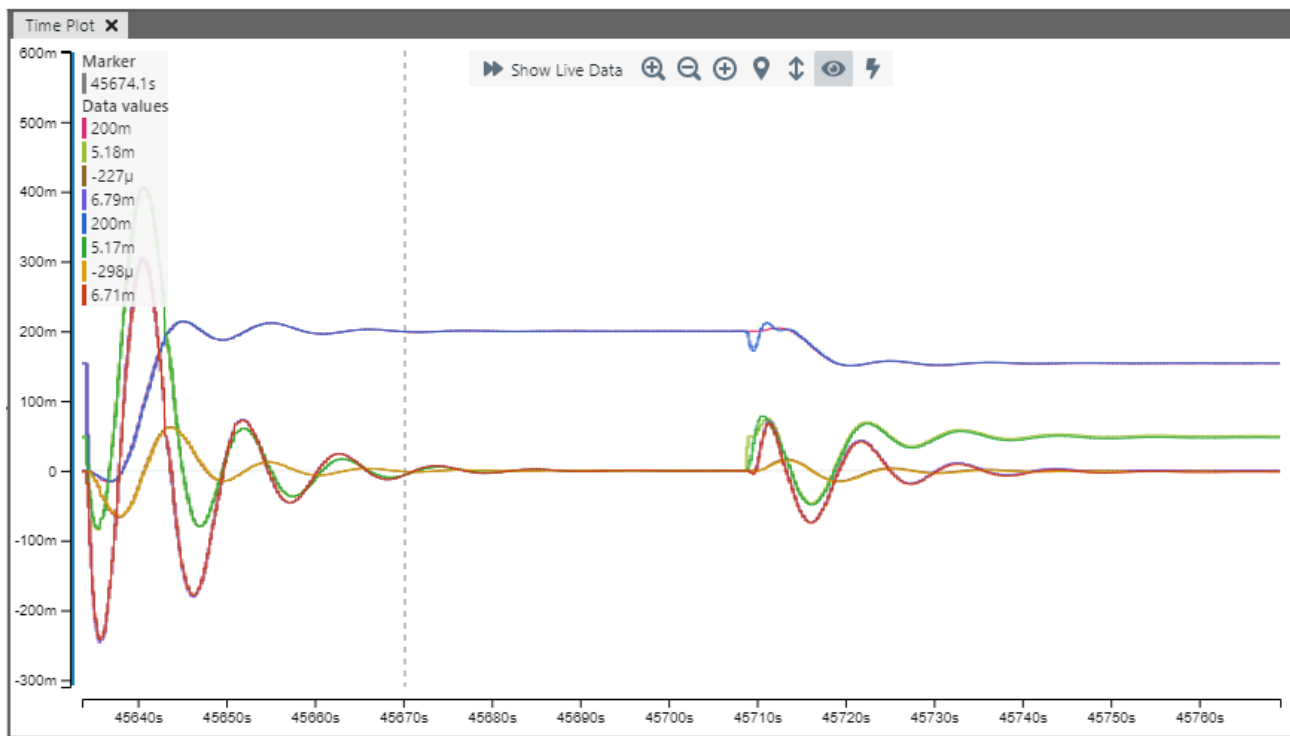
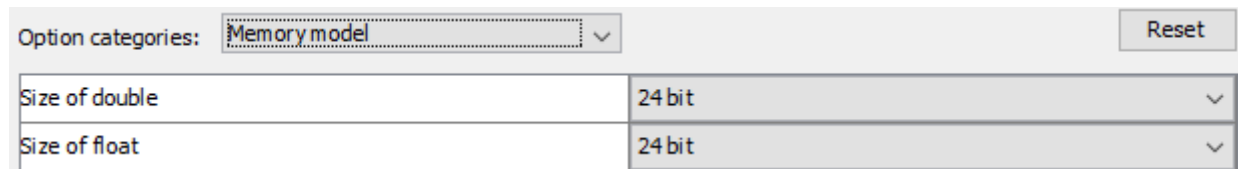


Imagen 38. Respuesta del sistema a la perturbación aplicada a  $Y_1$

## DOUBLE CON PRECISION DE 24 bits

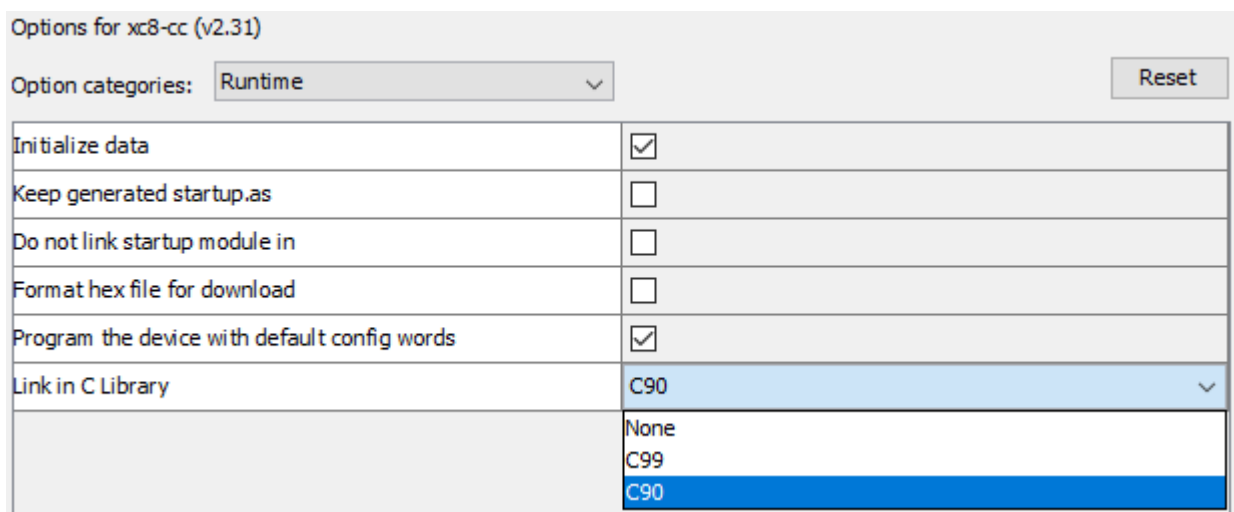
Se implementa el mismo programa (plantaPLUSobservador()) reconfigurando la precision del tipo de dato double a 24 bits (3 bytes).



Option categories: Memory model		Reset
Size of double	24 bit	▼
Size of float	24 bit	▼

*Imagen 39. Cambio de precisión de un “double” en MPLAB utilizando el compilador XC8*

Tambien se debe tener en cuenta que se debe seleccionar C90 en “Link in C Library”.



Options for xc8-cc (v2.31)		Reset
Option categories: Runtime		
Initialize data	<input checked="" type="checkbox"/>	
Keep generated startup.as	<input type="checkbox"/>	
Do not link startup module in	<input type="checkbox"/>	
Format hex file for download	<input type="checkbox"/>	
Program the device with default config words	<input checked="" type="checkbox"/>	
Link in C Library	C90	▼
	None	
	C99	
	C90	

*Imagen 40. Cambio necesario para implementar un “double” de 24 bits en el compilador XC8*

Posteriormente se exportaran los datos generados a MatLab y se realizan los graficos pertinentes. Se observa que practicamente no hay variacion entre ambas curvas.

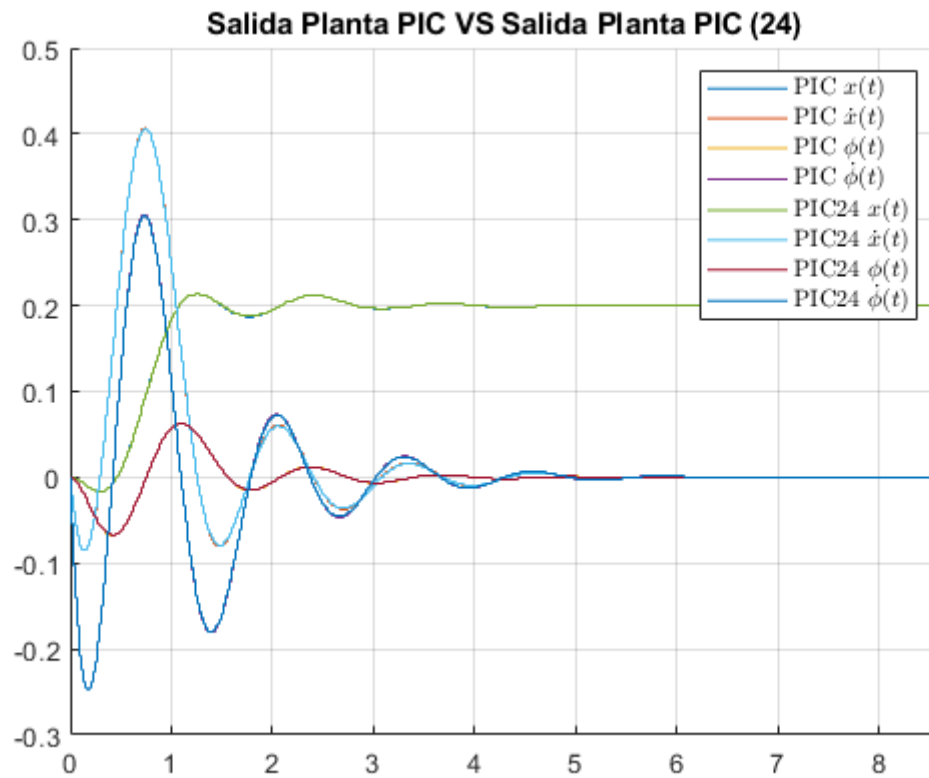


Imagen 41. Señal a la salida de la planta: PIC con double de 32 bits VS PIC con double de 24 bits

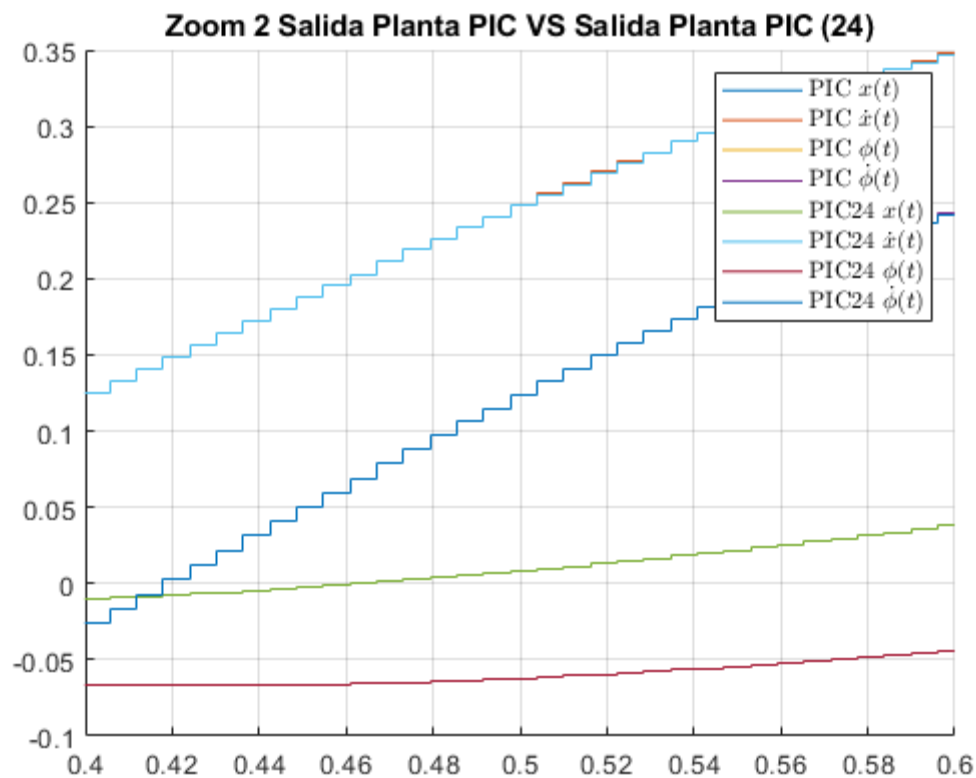


Imagen 42. Zoom señal a la salida de la planta: PIC con double de 32 bits VS PIC con double de 24 bits

	Salida Planta	Salida Observadro
PIC (double 32) VS SIMULINK	$4,50 \times 10^{-5}$	$4,41 \times 10^{-5}$
PIC (double 24) VS SIMULINK	$4,18 \times 10^{-5}$	$4,10 \times 10^{-5}$
PIC (double 32) VS PIC (double 24)	$1,21 \times 10^{-7}$	$1,15 \times 10^{-7}$

*Tabla 1. Error medio cuadrático*

Sorprendentemente el error es menor cuando se trabaja con double de 24 bits. Se esperaba lo contrario. Para sacar mayores conclusiones respecto a este tema se recomienda implementar mas ejemplos. A su vez observamos que el error a la salida del observador siempre sera menor al error de la salida de la planta

## Problemáticas encontradas (y superadas) a lo largo del TP

Problema	Solución
Función para la multiplicación de matrices	<p>Una vez declarada la función de multiplicación (donde se debe aclarar al menos una dimensión de cada una de la matriz) no se puede modificar el tamaño declarado. Al tener matrices de diversas dimensiones se decide crear una función específica para cada caso. Se deja el numero de columnas fijo (por función) permitiendo variaciones en el numero de filas de la primera matriz.</p> <p>A su vez se uso siempre el mismo tipo de dato ("volatile double") para evitar incompatibilidades. Como resultado algunos datos fueron declarados como "volatile" cuando en realidad no era necesario.</p> <p>Las variables que representaban un único valor se debían declarar con notación de matriz. Ej.: <i>volatile double val[1][1] = {{...}};</i></p>
Implementación de las ecuaciones del espacio de estados	<p>Se recurrió a Excel para "simular" el funcionamiento del sistema realimentado.</p> <p>Se solicitó una clase de consulta con el profesor a cargo para aclarar conceptos. Se comprendió que existirá un estado adelantado (<math>X[n+1]</math>) en el sistema como consecuencia de fijar las condiciones iniciales.</p>
Sistema oscilante	<p>Se hizo una revisión del código y se comprendió que era un problema de como se estaba determinando la señal de entrada del observador. Fue un error de atención.</p> <p>Forma incorrecta:</p> <pre>inputObs[0][0]=U[0][0]; inputObs[1][0]=Y[1][0]; inputObs[2][0]=Y[2][0]; inputObs[3][0]=Y[3][0]; inputObs[4][0]=Y[4][0];</pre> <p>Forma correcta:</p> <pre>inputObs[0][0]=U[0][0]; inputObs[1][0]=Y[0][0]; inputObs[2][0]=Y[1][0]; inputObs[3][0]=Y[2][0]; inputObs[4][0]=Y[3][0];</pre>
El sistema no alcanzaba los valores máximos y estabilizaba por debajo del valor de referencia	<p>En un principio se pensó que se debía a la precisión de los valores (en Matlab la precisión es mucho mayor que en el compilador XC8), pero se verificó que este no era el caso. Si el sistema no estabiliza en el valor de referencia en el permanente NO es debido al uso de "float". El error se debía a un error de escalamiento de la señal de referencia. Por un error de atención se utilizó Nbard y no Nbard_fast.</p>
Ts y Tso	<p>Es necesario que toda la simulación corra según Tso. No es posible hacer correr la planta más rápido que el observador ni el observador más rápido que la planta. Este segundo caso ya fue explicado. La planta no puede correr más rápido que el observador ya que si ningún valor es realimentado, la entrada U a la planta será constante. Como la planta es digital no evoluciona y mantiene su salida constante (esto sería distinto si la planta fuese analógica).</p>