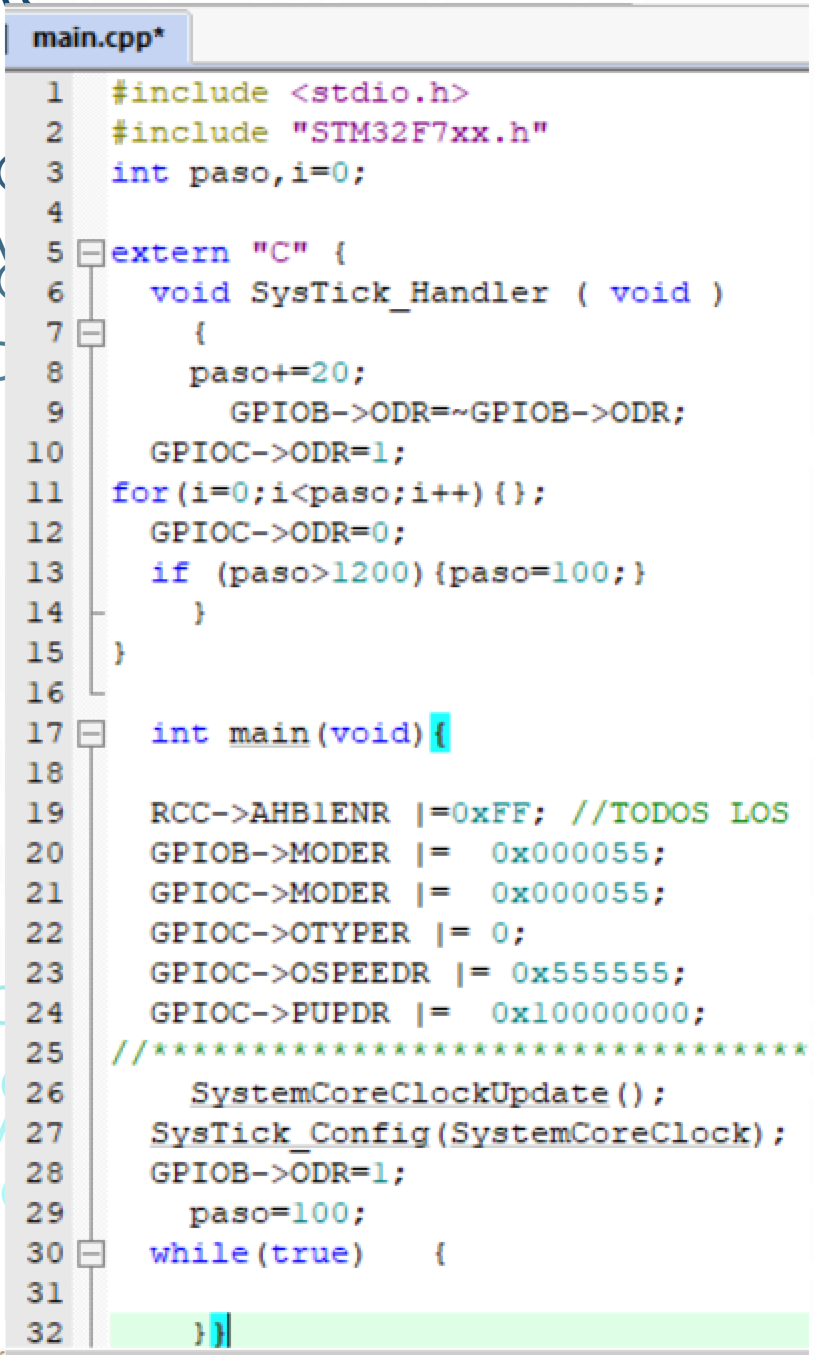
USART-ADC CON SYSTICK

ALGORITMO DE CONTROL PID

SCIENTIÆ · PATRIÆ · FAMILIÆ

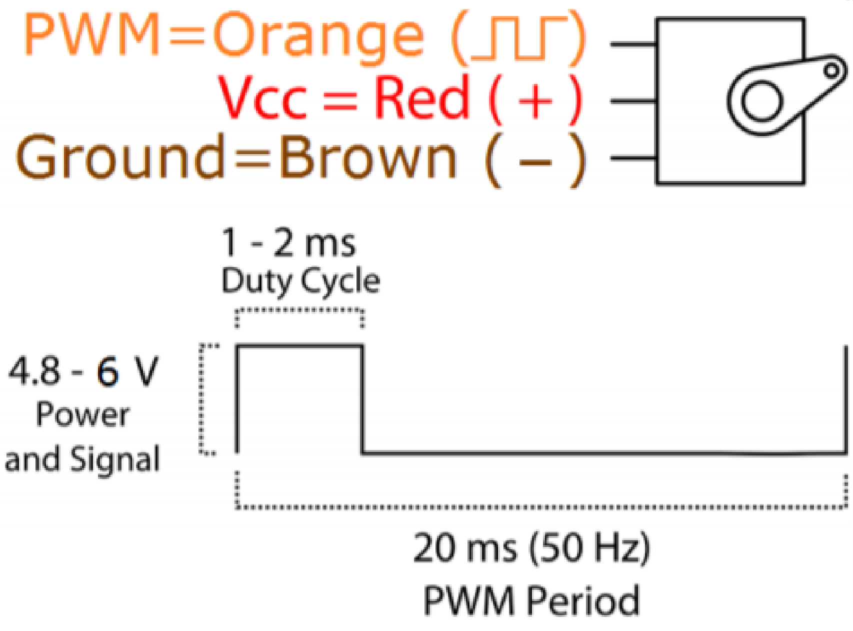
UNIVERSIDAD MILITAR
NUEVA GRANADA

INGENIERÍA
MECATRÓNICA
UMNG



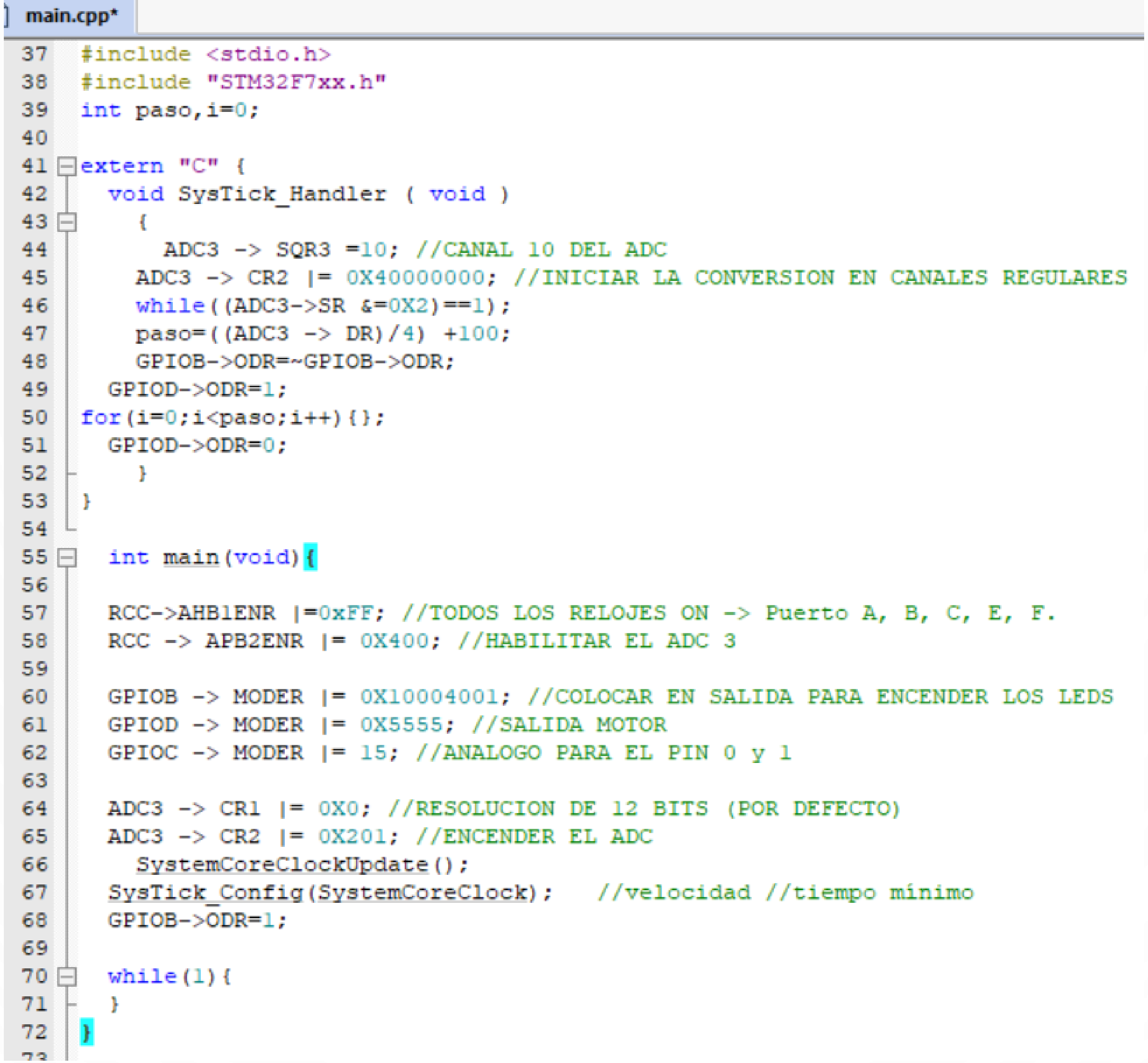
```
#include <stdio.h>
#include "STM32F7xx.h"
int paso,i=0;

extern "C" {
  void SysTick_Handler ( void )
    {
    paso+=20;
      GPIOB->ODR=~GPIOB->ODR;
  GPIOC->ODR=1;
for(i=0;i<paso;i++){};
  GPIOC->ODR=0;
  if (paso>1200){paso=100;}
    }
}

  int main(void){

  RCC->AHB1ENR |=0xFF; //TODOS LOS
  GPIOB->MODER |=  0x000055;
  GPIOC->MODER |=  0x000055;
  GPIOC->OTYPER |= 0;
  GPIOC->OSPEEDR |= 0x555555;
  GPIOC->PUPDR |=  0x10000000;
//*********************************
    SystemCoreClockUpdate();
  SysTick_Config(SystemCoreClock);
  GPIOB->ODR=1;
    paso=100;
  while(true)    {

    }}
```

MG90S
Metal Gear Servo



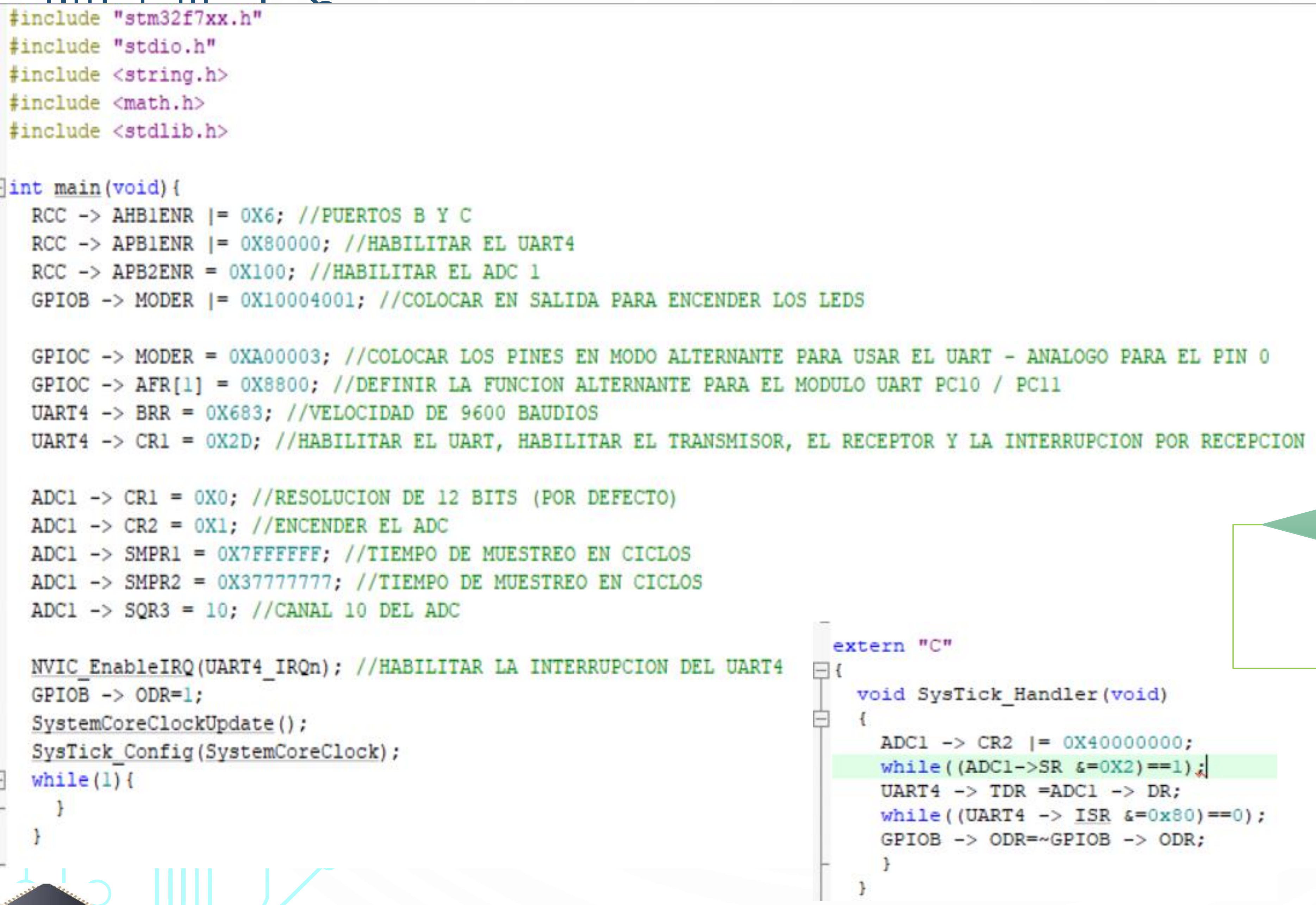
Specifications

- Weight: 13.4 g
- Dimension: 22.5 x 12 x 35.5 mm approx.
- Stall torque: 1.8 kgf·cm (4.8V), 2.2 kgf·cm (6 V)
- Operating speed: 0.1 s/60 degree (4.8 V), 0.08 s/60 degree (6 V)
- Operating voltage: 4.8 V - 6.0 V
- Dead band width: 5 µs



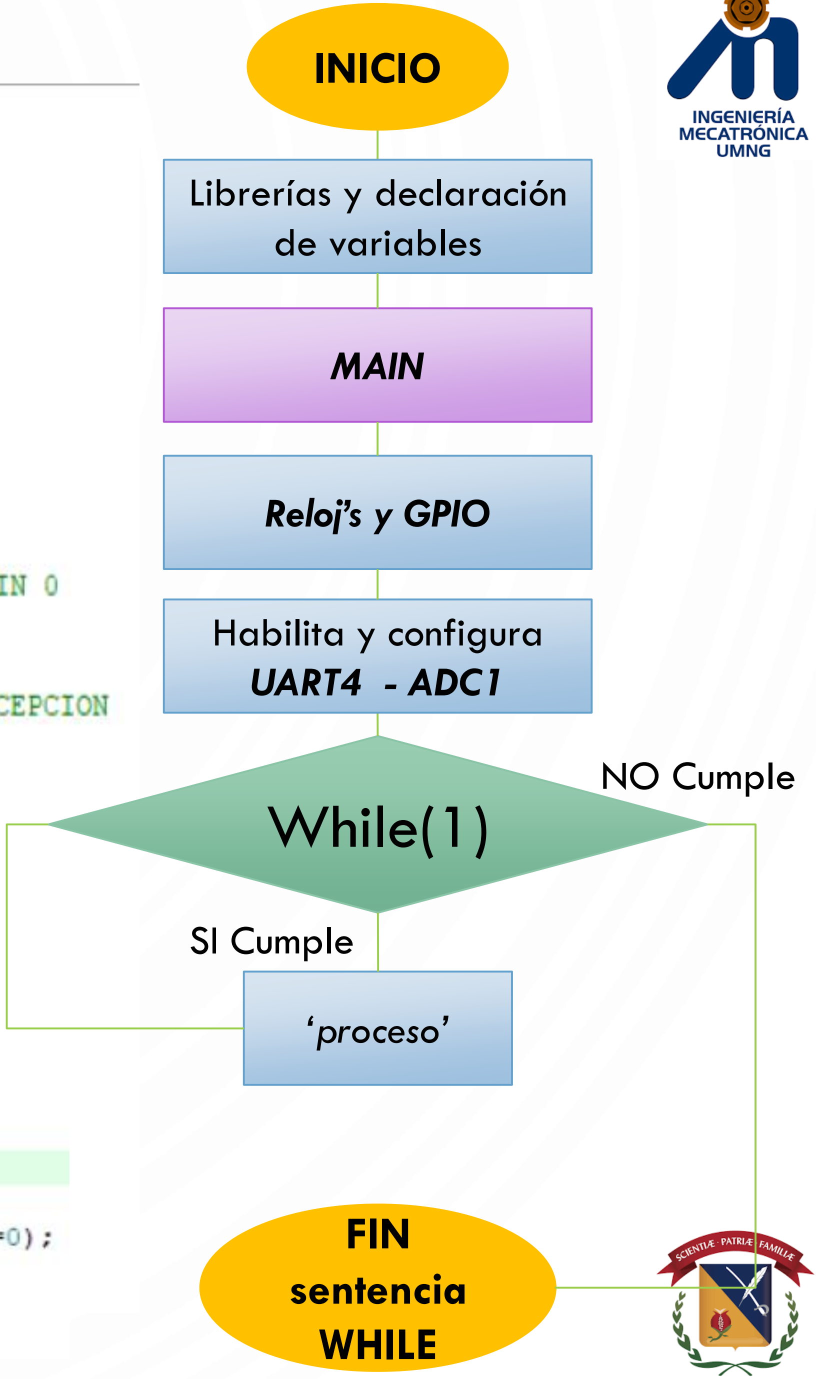
main.cpp*

```
#include <stdio.h>
#include "STM32F7xx.h"
int paso,i=0;

extern "C" {
 void SysTick_Handler ( void )
  {
   ADC3 -> SQR3 =10; //CANAL 10 DEL ADC
  ADC3 -> CR2 |= 0X40000000; //INICIAR LA CONVERSION EN CANALES REGULARES
  while((ADC3->SR &=0X2)==1);
  paso=((ADC3 -> DR)/4) +100;
  GPIOB->ODR=~GPIOB->ODR;
 GPIOD->ODR=1;
for(i=0;i<paso;i++){};
 GPIOD->ODR=0;
  }
}

 int main(void){

 RCC->AHB1ENR |=0xFF; //TODOS LOS RELOJES ON -> Puerto A, B, C, E, F.
 RCC -> APB2ENR |= 0X400; //HABILITAR EL ADC 3

 GPIOB -> MODER |= 0X10004001; //COLOCAR EN SALIDA PARA ENCENDER LOS LEDS
 GPIOD -> MODER |= 0X5555; //SALIDA MOTOR
 GPIOC -> MODER |= 15; //ANALOGO PARA EL PIN 0 y 1

 ADC3 -> CR1 |= 0X0; //RESOLUCION DE 12 BITS (POR DEFECTO)
 ADC3 -> CR2 |= 0X201; //ENCENDER EL ADC
  SystemCoreClockUpdate();
 SysTick_Config(SystemCoreClock);   //velocidad //tiempo mínimo
 GPIOB->ODR=1;

 while(1){
 }
}
```



```
#include "stm32f7xx.h"
#include "stdio.h"
#include <string.h>
#include <math.h>
#include <stdlib.h>

int main(void){
  RCC -> AHB1ENR |= 0X6; //PUERTOS B Y C
  RCC -> APB1ENR |= 0X80000; //HABILITAR EL UART4
  RCC -> APB2ENR = 0X100; //HABILITAR EL ADC 1
  GPIOB -> MODER |= 0X10004001; //COLOCAR EN SALIDA PARA ENCENDER LOS LEDS

  GPIOC -> MODER = 0XA00003; //COLOCAR LOS PINES EN MODO ALTERNANTE PARA USAR EL UART - ANALOGO PARA EL PIN 0
  GPIOC -> AFR[1] = 0X8800; //DEFINIR LA FUNCION ALTERNANTE PARA EL MODULO UART PC10 / PC11
  UART4 -> BRR = 0X683; //VELOCIDAD DE 9600 BAUDIOS
  UART4 -> CR1 = 0X2D; //HABILITAR EL UART, HABILITAR EL TRANSMISOR, EL RECEPTOR Y LA INTERRUPCION POR RECEPCION

  ADC1 -> CR1 = 0X0; //RESOLUCION DE 12 BITS (POR DEFECTO)
  ADC1 -> CR2 = 0X1; //ENCENDER EL ADC
  ADC1 -> SMPR1 = 0X7FFFFFF; //TIEMPO DE MUESTREO EN CICLOS
  ADC1 -> SMPR2 = 0X37777777; //TIEMPO DE MUESTREO EN CICLOS
  ADC1 -> SQR3 = 10; //CANAL 10 DEL ADC

  NVIC_EnableIRQ(UART4_IRQn); //HABILITAR LA INTERRUPCION DEL UART4
  GPIOB -> ODR=1;
  SystemCoreClockUpdate();
  SysTick_Config(SystemCoreClock);
  while(1){
    }
  }
```

```
extern "C"
{
  void SysTick_Handler(void)
  {
    ADC1 -> CR2 |= 0X40000000;
    while((ADC1->SR &=0X2)==1);
    UART4 -> TDR =ADC1 -> DR;
    while((UART4 -> ISR &=0x80)==0);
    GPIOB -> ODR=~GPIOB -> ODR;
    }
  }
```



INICIO

Librerías y declaración de variables

MAIN

Reloj's y GPIO

Habilita y configura *UART4 - ADC1*

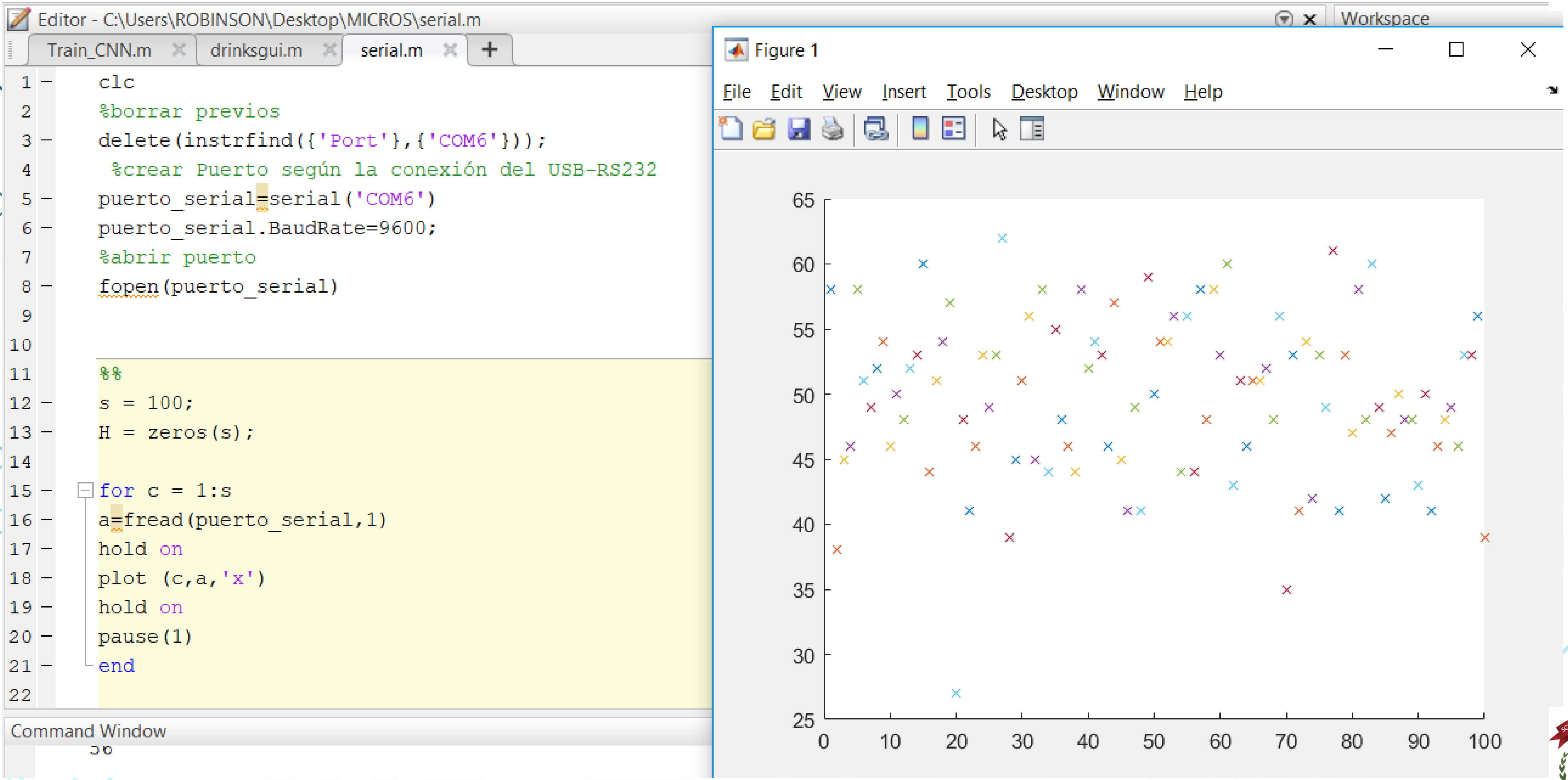
While(1)

NO Cumple

SI Cumple

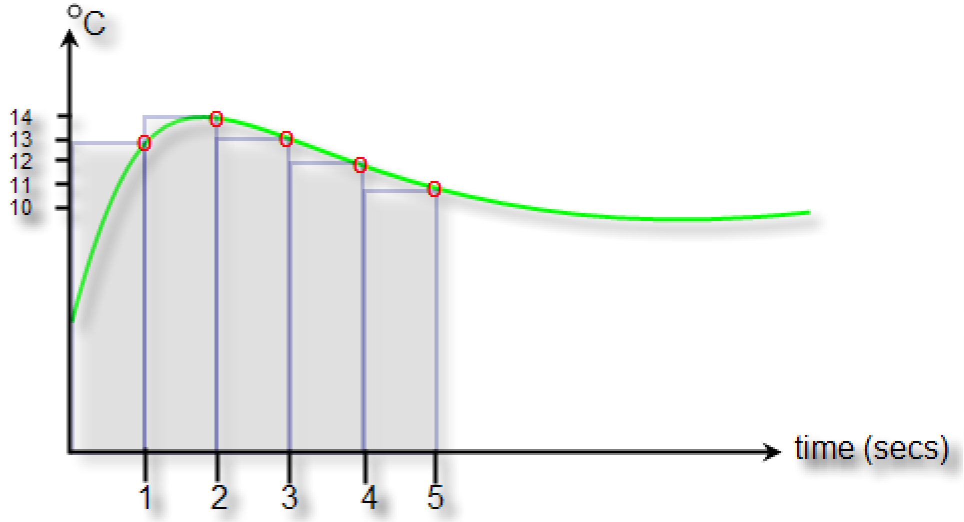
'proceso'

FIN sentencia WHILE



```
clc
%borrar previos
delete(instrfind({'Port'},{'COM6'}));
 %crear Puerto según la conexión del USB-RS232
puerto_serial=serial('COM6')
puerto_serial.BaudRate=9600;
%abrir puerto
fopen(puerto_serial)


%%
s = 100;
H = zeros(s);

for c = 1:s
a=fread(puerto_serial,1)
hold on
plot (c,a,'x')
hold on
pause(1)
end

```

ALGORITMO PID



$$\mathrm{Gr}(t) = \mathrm{Kp}*e(t) + \mathrm{Ki}\int_0^t e(t)\,dt + \mathrm{Kp}\frac{d}{dt}e(t)$$

$$G_c(z) = \frac{U(z)}{E(z)} = K_p + K_i\frac{zT}{z-1} + K_d\frac{z-1}{zT}$$

$$\frac{U(z)}{e(z)} = K_p + \frac{K_i}{1-z^{-1}} + K_d(1-z^{-1})$$

$$\frac{U(z)}{e(z)} = \frac{K_p(1-z^{-1}) + K_i + K_d(1-z^{-1})(1-z^{-1})}{1-z^{-1}}$$

$$\frac{U(z)}{e(z)} = \frac{K_p(1-z^{-1}) + K_i + K_d(1-z^{-1})^2}{1-z^{-1}}$$

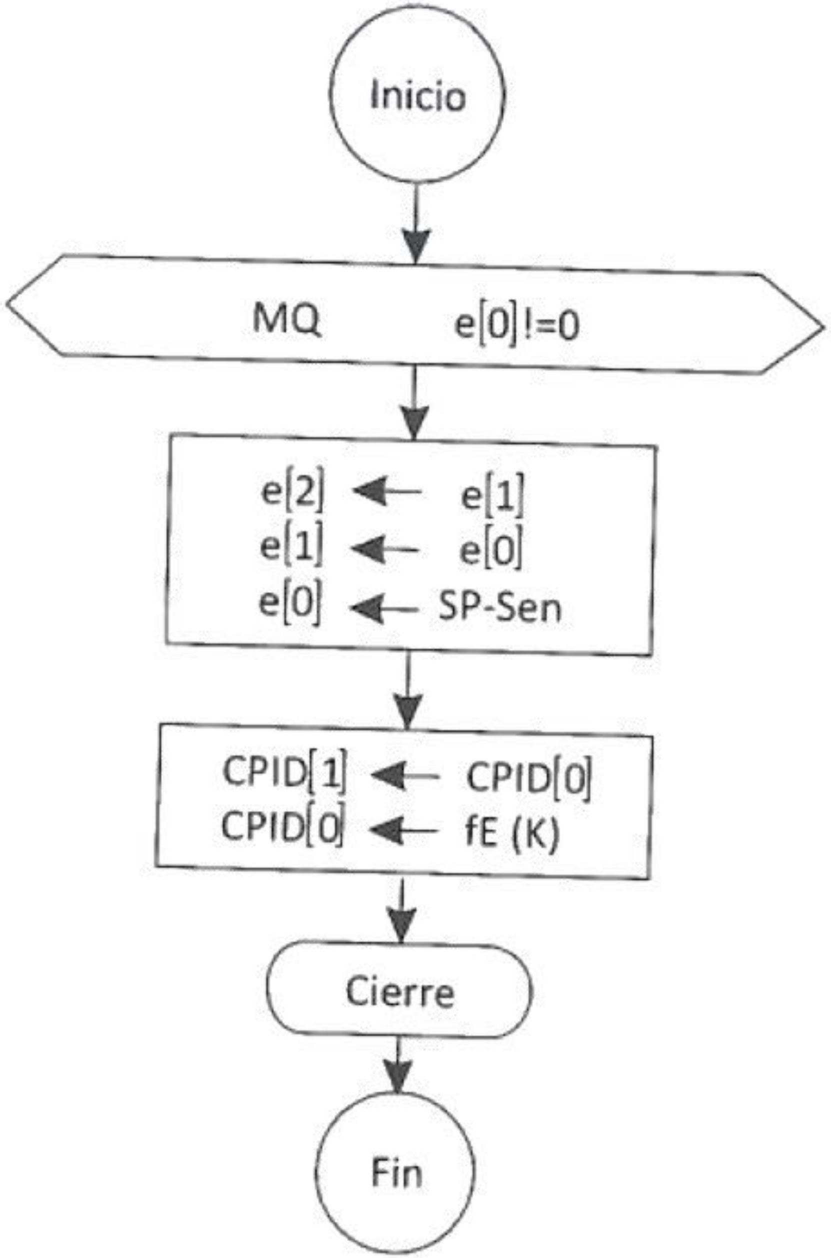
$$\frac{U(z)}{e(z)} = \frac{K_p(1-z^{-1}) + K_i + K_d(1-2z^{-1}+z^{-2})}{1-z^{-1}}$$

$$\frac{U(z)}{e(z)} = \frac{K_p - K_pz^{-1} + K_i + K_d - 2K_dz^{-1} + K_dz^{-2}}{1-z^{-1}}$$

$$\frac{U(z)}{e(z)} = \frac{K_p + K_i + K_d - 2K_dz^{-1} - K_pz^{-1} + K_dz^{-2}}{1-z^{-1}}$$

$$\frac{U(z)}{e(z)} = \frac{K_p + K_i + K_d + z^{-1}(-2K_d - K_p) + K_dz^{-2}}{1-z^{-1}}$$

$$u(k) = \left(K_p + K_i T + \frac{K_d}{T}\right) e(k) - \left(2\frac{K_d}{T} + K_p\right) e(k-1) + \frac{K_d}{T} e(k-2) + u(k-1)$$



Inicio

MQ e[0]!=0

e[2] ← e[1]
e[1] ← e[0]
e[0] ← SP-Sen

CPID[1] ← CPID[0]
CPID[0] ← fE (K)

Cierre

Fin