ADC CON LCD
USART-ADC CON SYSTICK

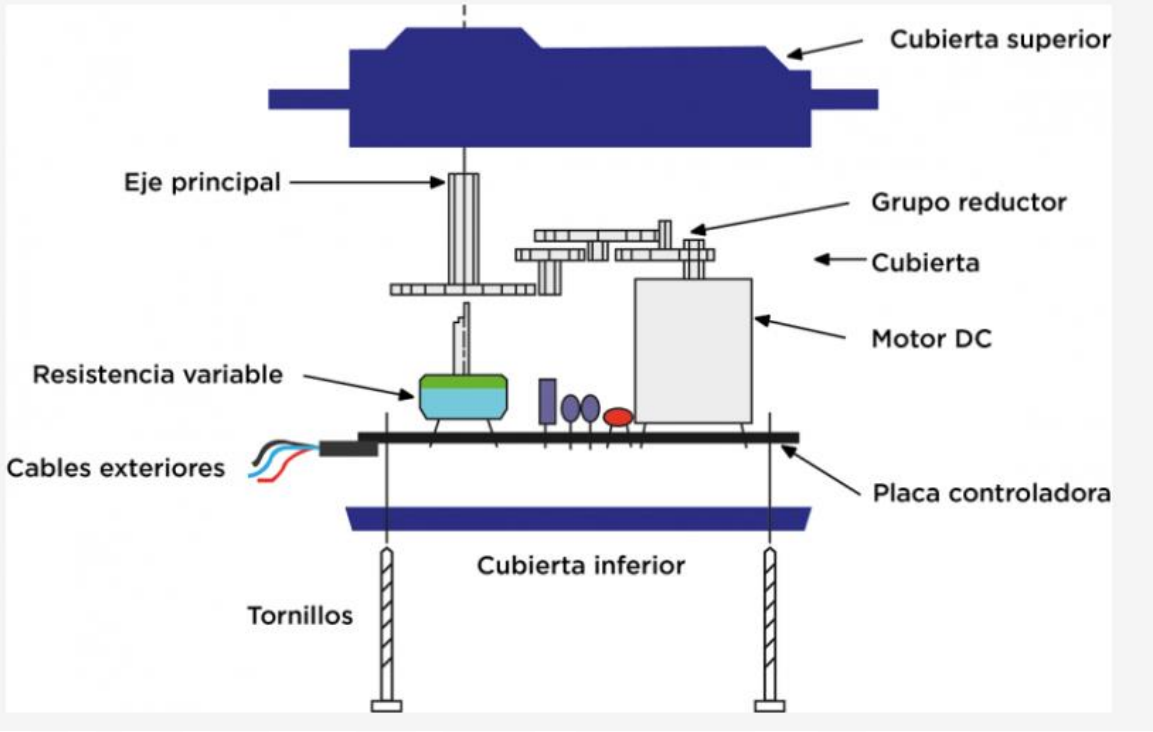
UNIVERSIDAD MILITAR NUEVA GRANADA

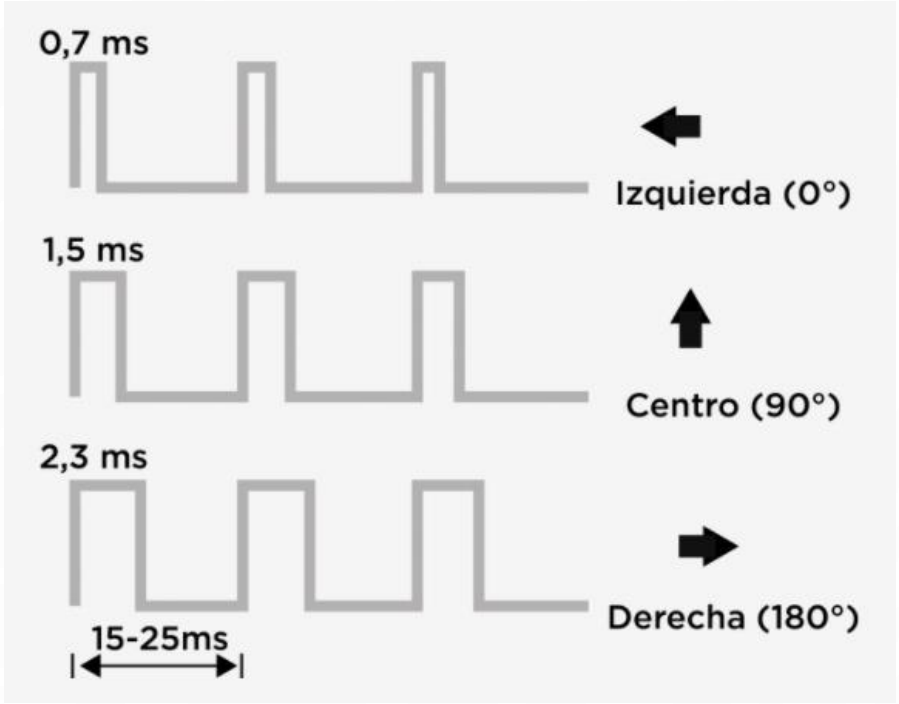
INGENIERÍA MECATRÓNICA UMNG

SERVOMOTOR

Un servomotor es un motor de corriente continua pero en vez de conseguir un giro continuo, está diseñado para conseguir que gire un determinado ángulo en respuesta a una señal de control, y que se mantenga fijo en esa posición.

https://www.ebotics.com/es/actividad/proyecto-no-3-controlar-un-servomotor/





CONEXIONES:

1.El cable **marrón** corresponde al GND del servomotor.

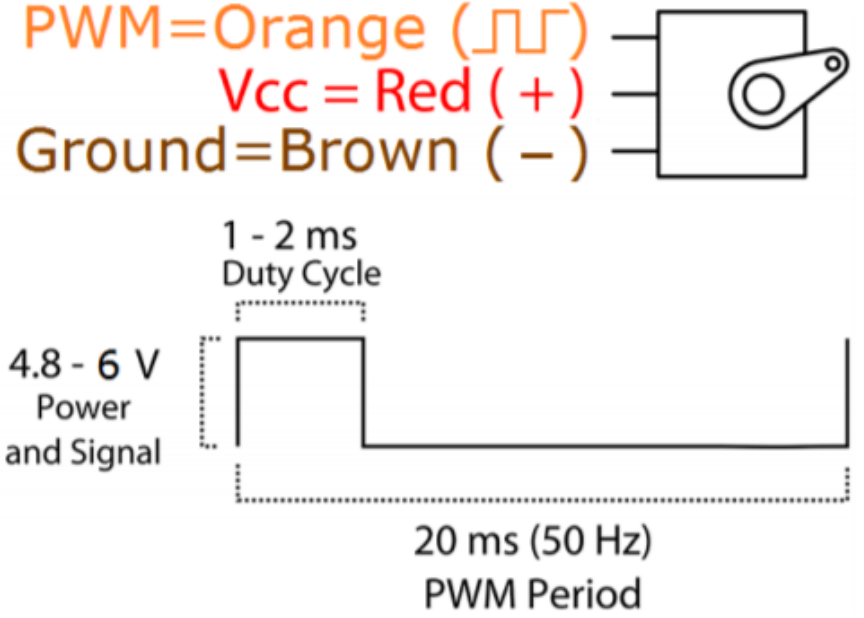
2.El cable **rojo** corresponde a los 5V.

3.El color **naranja** corresponde a la señal.

Esta señal de control es dada por los pines digitales PWM. Para controlar el servomotor se le envía pulsos cada 20 ms es decir a 50Hz. La anchura del pulso es lo que determinará el ángulo de giro

```
#include "STM32F4xx.h"
#define servo GPIOC->ODR
int paso;
extern "C"
{
  void SysTick_Handler(void){
    paso+=20;
    servo=1;
    for(int i=0; i<paso; i++){};
    servo=0;
  if (paso>1200){paso=1200;}
  }}
int main (void){
  RCC->AHB1ENR=0x7;
  GPIOC->MODER|=0X5555;
  SystemCoreClockUpdate();
  SysTick_Config(SystemCoreClock);
  paso=100;
  while(true){}
  }
```

MG90S
Metal Gear Servo



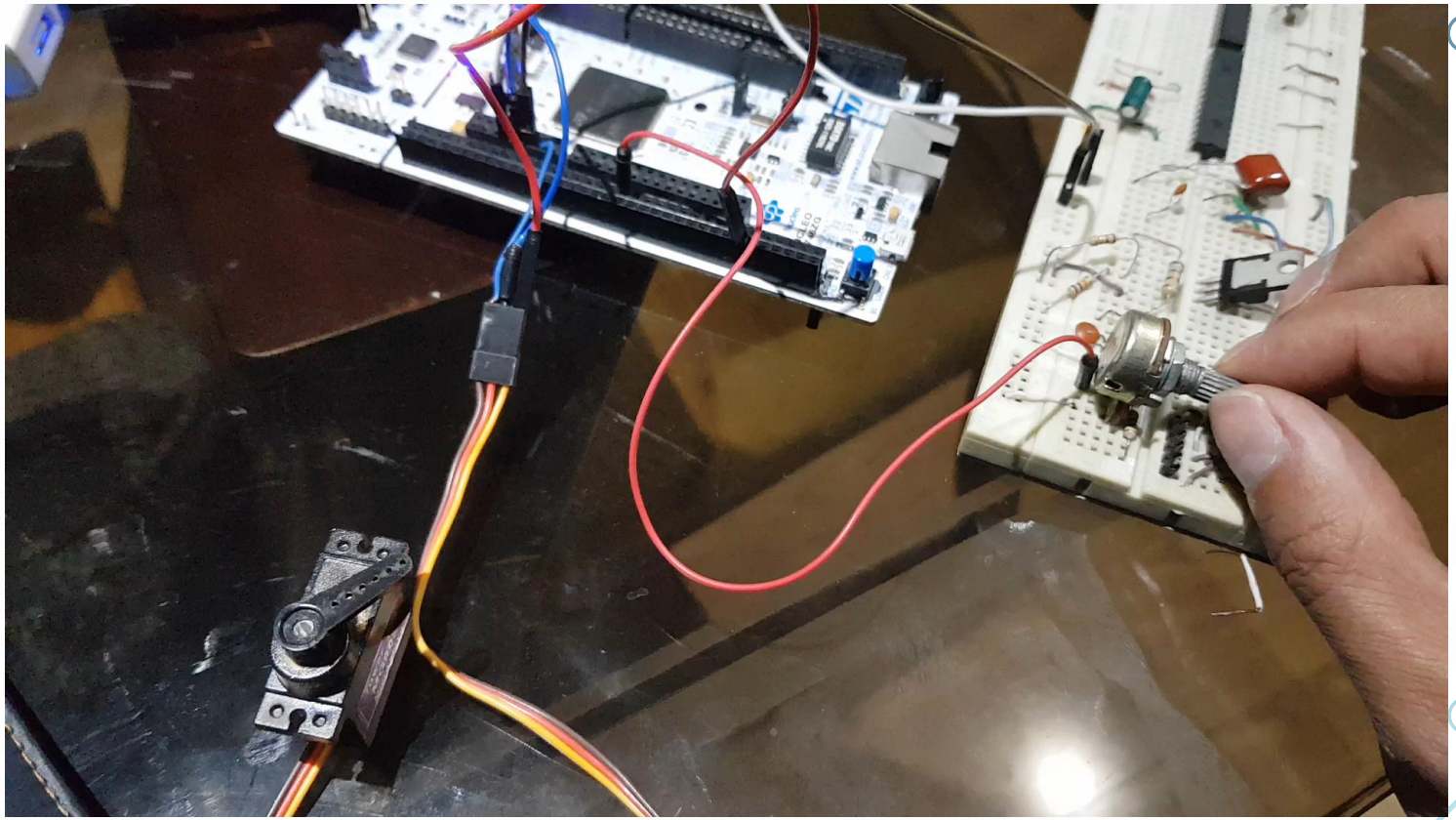
Specifications

- 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

```c
#include "STM32F4xx.h"
#define servo GPIOC->ODR
int paso;
extern "C"
{
  void SysTick_Handler(void){
    paso+=20;
    servo=1;
    for(int i=0; i<paso; i++){};
    servo=0;
  if (paso>1200){paso=1200;}
  }}
int main (void){
  RCC->AHB1ENR=0x7;
  GPIOC->MODER|=0X5555;
  SystemCoreClockUpdate();
  SysTick_Config(SystemCoreClock);
  paso=100;
  while(true){}
  }
```

main.cpp*

```
37  #include <stdio.h>
38  #include "STM32F7xx.h"
39  int paso,i=0;
40
41  extern "C" {
42    void SysTick_Handler ( void )
43      {
44        ADC3 -> SQR3 =10; //CANAL 10
45       ADC3 -> CR2 |= 0X40000000; //I
46       while((ADC3->SR &=0X2)==1);
47       paso=((ADC3 -> DR)/4) +100;
48       GPIOB->ODR=~GPIOB->ODR;
49    GPIOD->ODR=1;
50  for(i=0;i<paso;i++){};
51    GPIOD->ODR=0;
52      }
53  }
54
55    int main(void){
56
57    RCC->AHB1ENR |=0xFF; //TODOS LOS
58    RCC -> APB2ENR |= 0X400; //HABIL
59
60    GPIOB -> MODER |= 0X10004001; //
61    GPIOD -> MODER |= 0X5555; //SALI
62    GPIOC -> MODER |= 15; //ANALOGO
63
64    ADC3 -> CR1 |= 0X0; //RESOLUCION
65    ADC3 -> CR2 |= 0X201; //ENCENDER
66      SystemCoreClockUpdate();
67    SysTick_Config(SystemCoreClock);
68    GPIOB->ODR=1;
69
70    while(1){
71    }
72  }
```

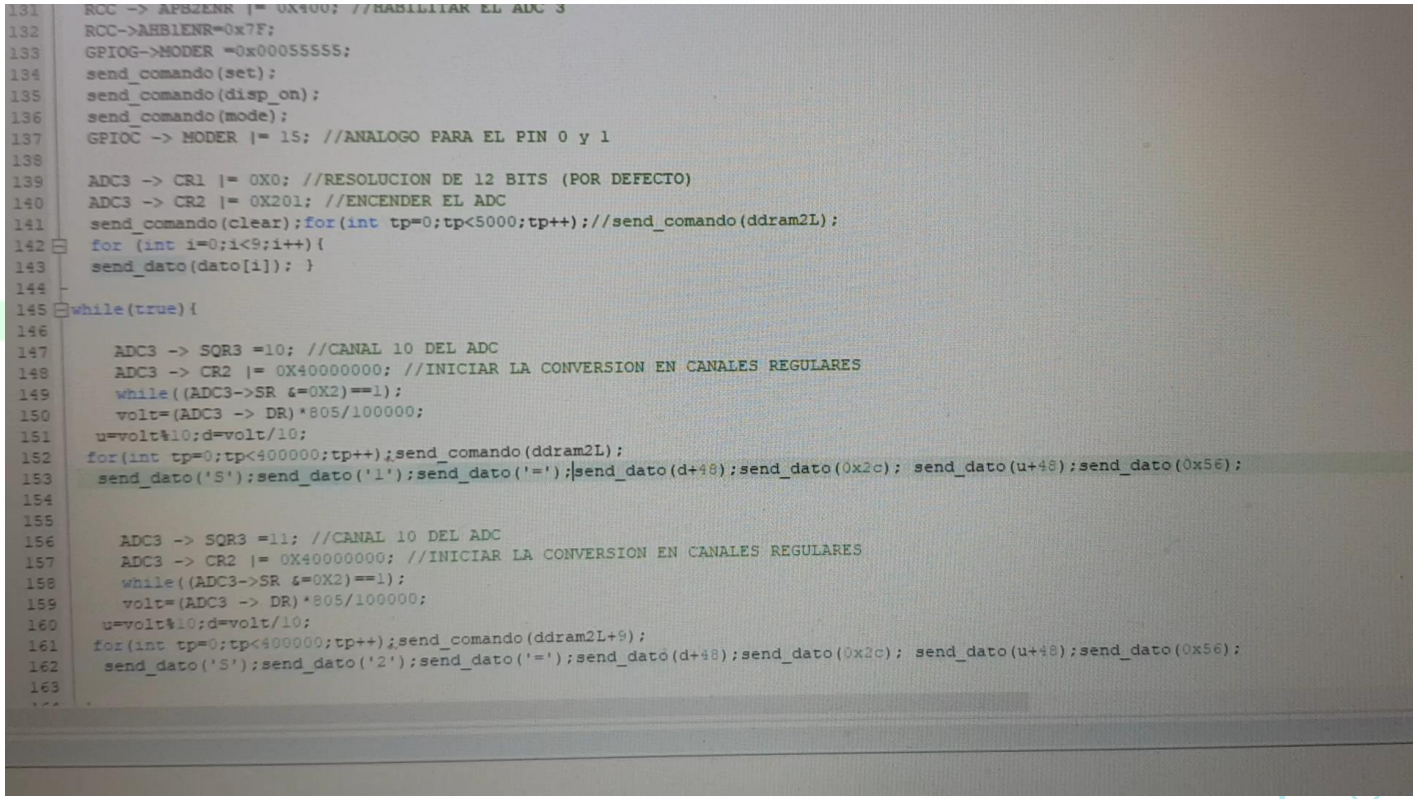


INGENIERÍA MECATRÓNICA UMNG

UNIVERSIDAD MILITAR NUEVA GRANADA

INGENIERÍA MECATRÓNICA UMNG

```
//////////////////////////LCD mas 2  adc/////////

#include "stm32f7xx.h"
char dato[9]={'V','o','l','t','a','j','e','s','='};
int d,u=0; int volt;
int dat[2]={0,0};
char clear=0x01;
char set = 0x38;
char disp_on= 0x0E;
char mode=0x06;
char ddram1L=0x80;
char ddram2L=0xC0;

void send_comando(char a){
  GPIOG ->ODR &=0XFF00;
  GPIOG ->ODR |= a;
  GPIOG->ODR &=~(1UL<<8);//RS=0
  GPIOG->ODR |=(1UL<<9); // Enable 1
  for(int Cont=0;Cont<10000;Cont++);
  GPIOG->ODR &=~(1UL<<9); // Enable 0
}

void send_dato(char b){
  GPIOG ->ODR &=0XFF00;
  GPIOG ->ODR |= b;
  GPIOG->ODR |=(1UL<<8);//RS=1
  GPIOG->ODR |=(1UL<<9); // Enable 1
  for(int Cont=0;Cont<10000;Cont++);
  GPIOG->ODR &=~(1UL<<9); // Enable 0
}
```



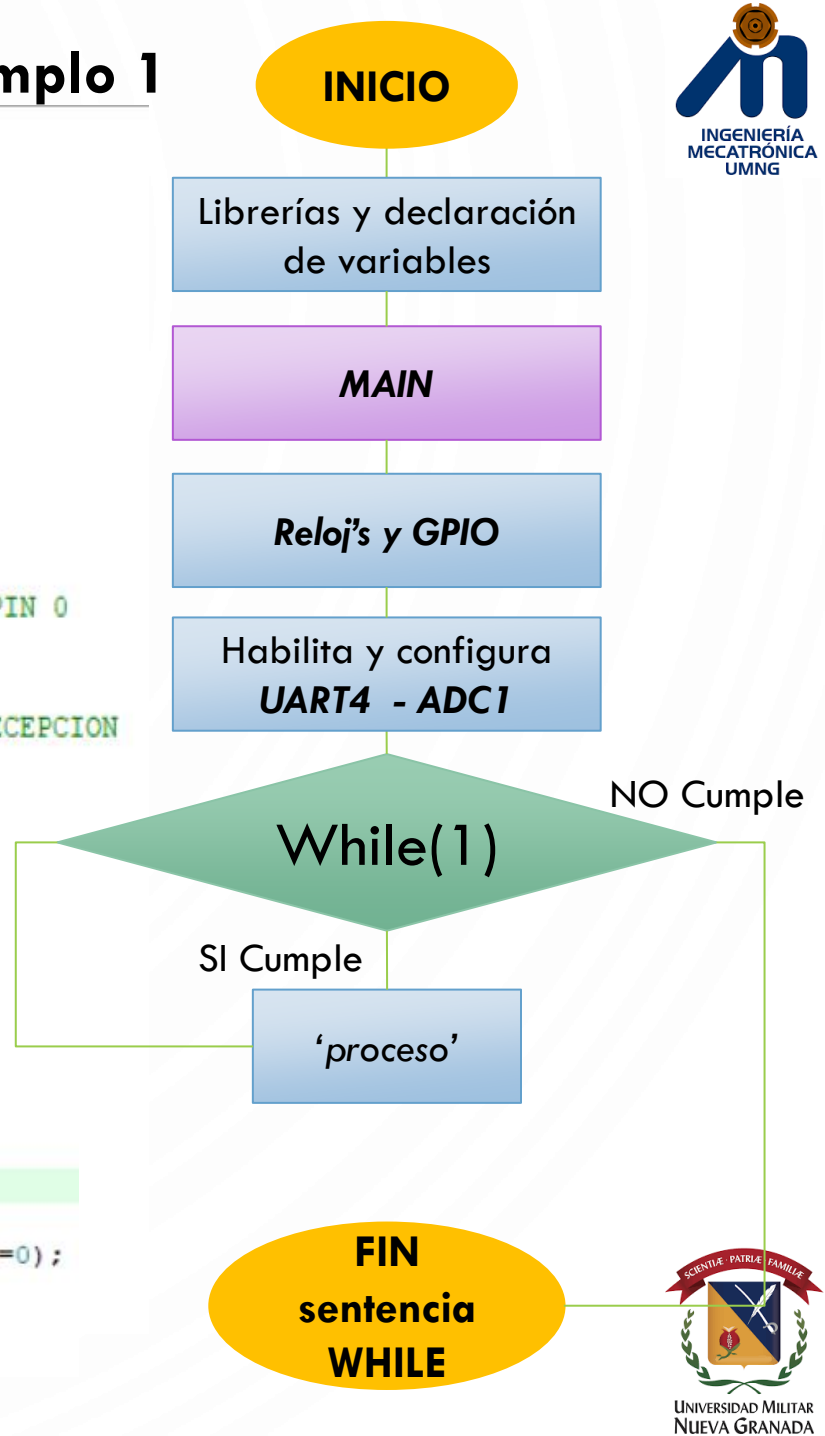
UNIVERSIDAD MILITAR NUEVA GRANADA

```
int main(void){
  RCC -> APB2ENR |= 0X400; //HABILITAR EL ADC 3
  RCC->AHB1ENR=0x7F;
  GPIOG->MODER =0x00055555;
  send_comando(set);
  send_comando(disp_on);
  send_comando(mode);
  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
  send_comando(clear);for(int tp=0;tp<5000;tp++);//send_comando(ddram2L);
  for (int i=0;i<9;i++){
  send_dato(dato[i]); }

while(true){
    ADC3 -> SQR3 =10; //CANAL 10 DEL ADC
    ADC3 -> CR2 |= 0X40000000; //INICIAR LA CONVERSION EN CANALES REGULARES
    while((ADC3->SR &=0X2)==1);
    volt=(ADC3 -> DR)*805/100000;
  u=volt%10;d=volt/10;
 for(int tp=0;tp<400000;tp++);send_comando(ddram2L);
  send_dato('S');send_dato('1');send_dato('=');send_dato(d+48);send_dato(0x2c); send_dato(u+48);send_dato(0x56);


    ADC3 -> SQR3 =11; //CANAL 10 DEL ADC
    ADC3 -> CR2 |= 0X40000000; //INICIAR LA CONVERSION EN CANALES REGULARES
    while((ADC3->SR &=0X2)==1);
    volt=(ADC3 -> DR)*805/100000;
  u=volt%10;d=volt/10;
 for(int tp=0;tp<400000;tp++);send_comando(ddram2L+9);
  send_dato('S');send_dato('2');send_dato('=');send_dato(d+48);send_dato(0x2c); send_dato(u+48);send_dato(0x56);
}
  }
```

ADC + comunicación serial, ejemplo 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;
    }
  }
```



Editor - C:\Users\ROBINSON\Desktop\MICROS\serial.m

Train_CNN.m | drinksgui.m | serial.m

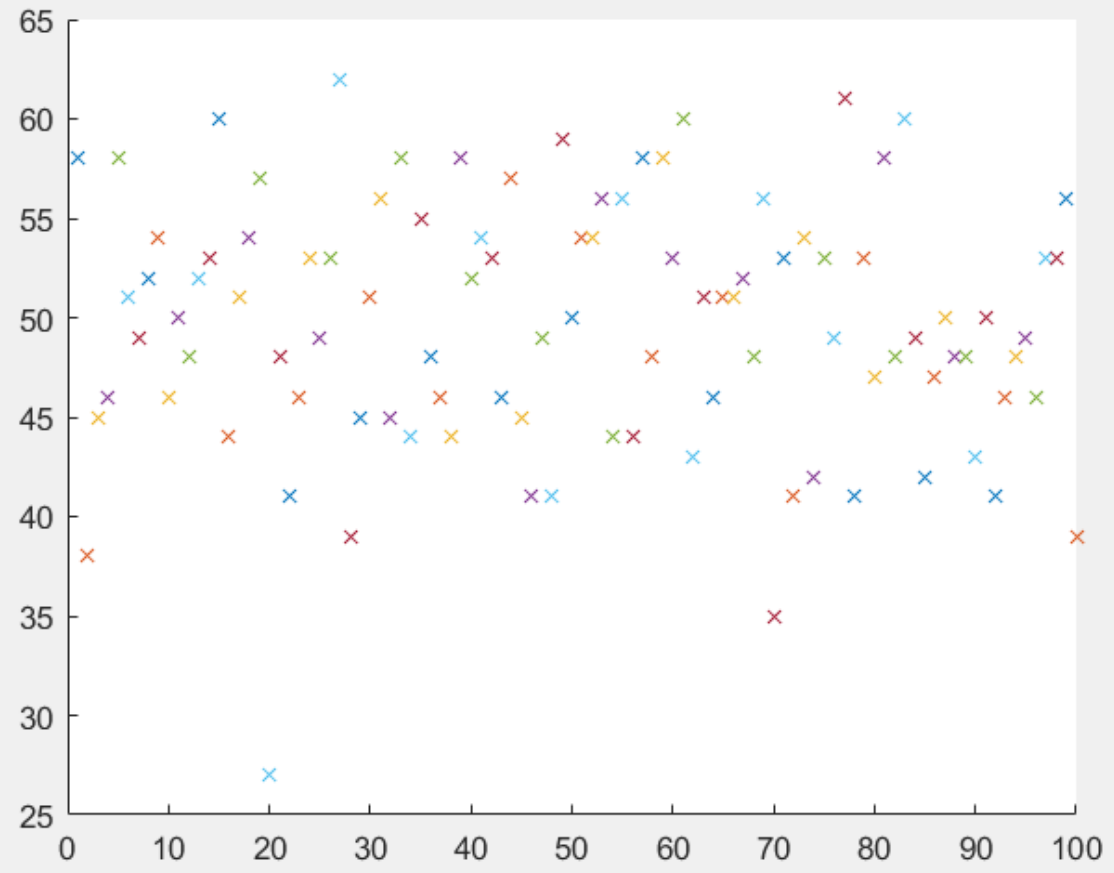
```
1  clc
2  %borrar previos
3  delete(instrfind({'Port'},{'COM6'}));
4   %crear Puerto según la conexión del USB-RS232
5  puerto_serial=serial('COM6')
6  puerto_serial.BaudRate=9600;
7  %abrir puerto
8  fopen(puerto_serial)
9
10
11 %%
12 s = 100;
13 H = zeros(s);
14
15 for c = 1:s
16 a=fread(puerto_serial,1)
17 hold on
18 plot (c,a,'x')
19 hold on
20 pause(1)
21 end
22
```

Command Window

Workspace

Figure 1

File Edit View Insert Tools Desktop Window Help

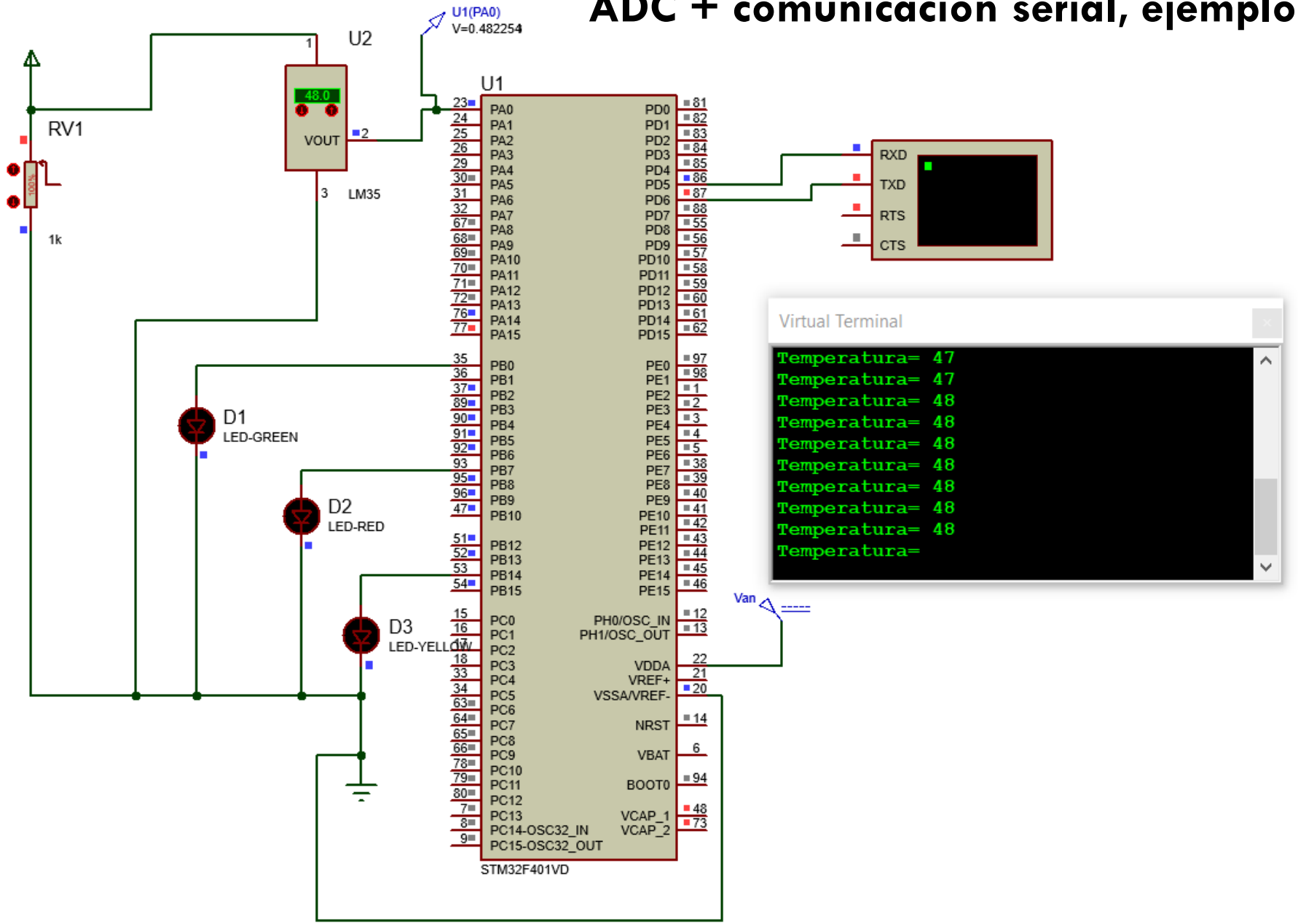


ADC + comunicación serial, ejemplo 2

```
//////////////ADC - SERIAL /////////
#include "STM32F4xx.h"

int temp,d,u;
char envio1[] = {"Temperatura= "};

extern "C" {

void send(){
 d=temp/10;
 u=temp%10;
    for ( char i=0;i !='=';i++){
      USART2->DR = envio1[i]; while ((USART2->SR &= 0x80)==0); }
      USART2->DR = d+0x30;  while ((USART2->SR &= 0x80)==0);
      USART2->DR = u+0x30;  while ((USART2->SR &= 0x80)==0);
      USART2->DR = 0x0d;     while ((USART2->SR &= 0x80)==0);
      for(int x=0;x<1000000;x++);

  }
}
```

```
int main(void){
  RCC->AHB1ENR =0xF; //Puertos A,B,C,D,E,F,G,H
  RCC -> APB2ENR = 0X100; //HABILITAR EL ADC 1
  GPIOB -> MODER = 0X55555555; //COLOCAR EN SAL
  GPIOA -> MODER |= 0X3; //ANALOGO PARA EL PIN

  ADC1 -> CR2 = 0X1; //ENCENDER EL ADC
  ADC1 -> SQR3 = 0; //CANAL 0 DEL ADC PIN PA0
//************************************************
  //CONFIGURACION UART
  RCC->APB1ENR |= (1UL<<17); // Enable clock fo
  USART2->BRR = 0x683;      // 9600 Baudios, fc
  USART2->CR1 = 0x012C;      // Tx habilitado,
  USART2->CR1 |= 0x2000;

  NVIC_EnableIRQ(USART2_IRQn);      // Llama in

  GPIOD->MODER |=  0x2800;  //pines PD5  PD6 en
  GPIOD->AFR[0] |= 0x007700000;            // P
                                           // PD

  while(true){

    ADC1 -> CR2 |= (1UL<<30); //0X40000000; //I
    while((ADC1->SR & 0X2)==1);

    temp=((ADC1->DR*805)/10000);send();
}}
```

ADC + comunicación serial, ejemplo 2

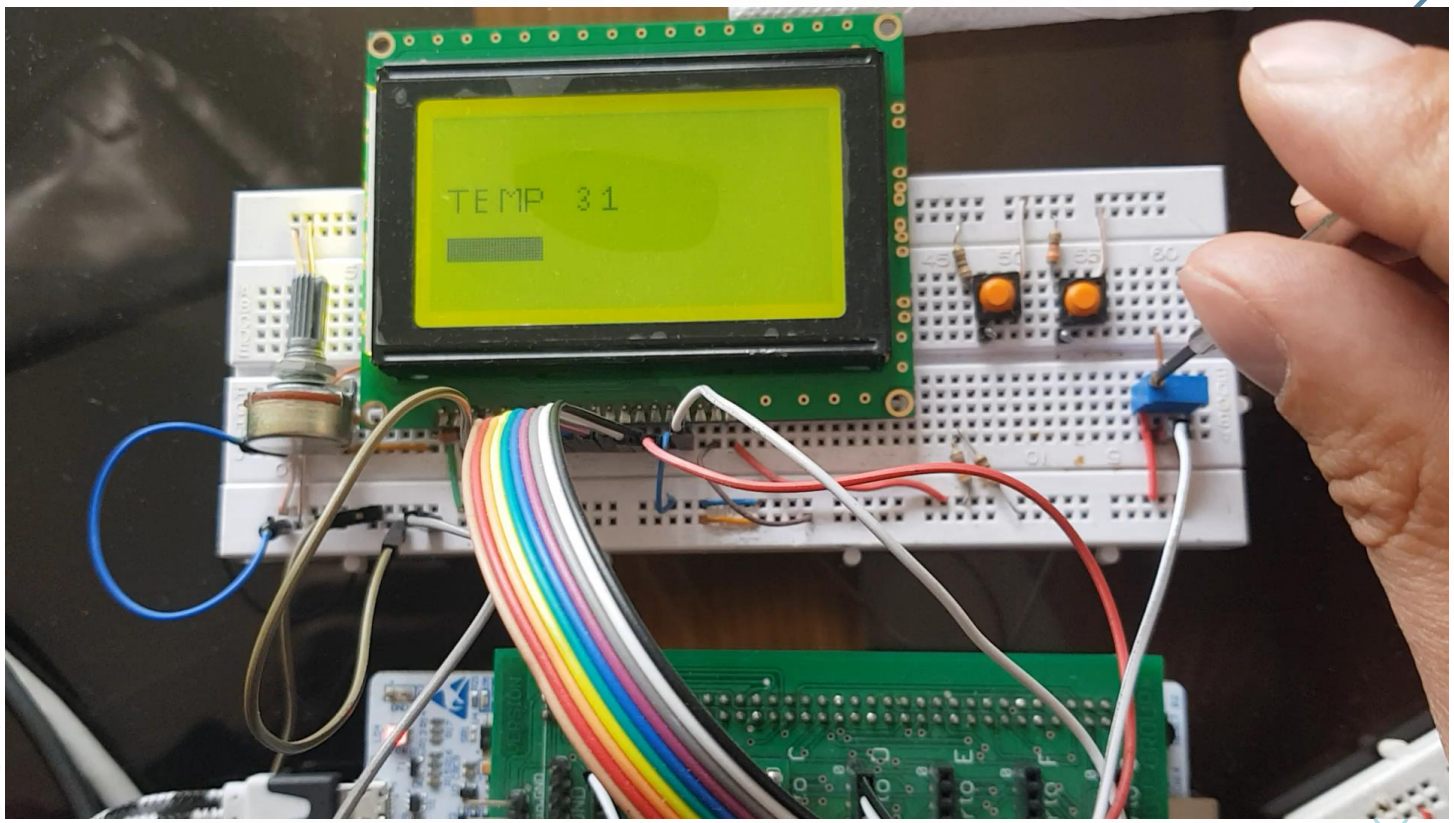


ADC EN GLCD

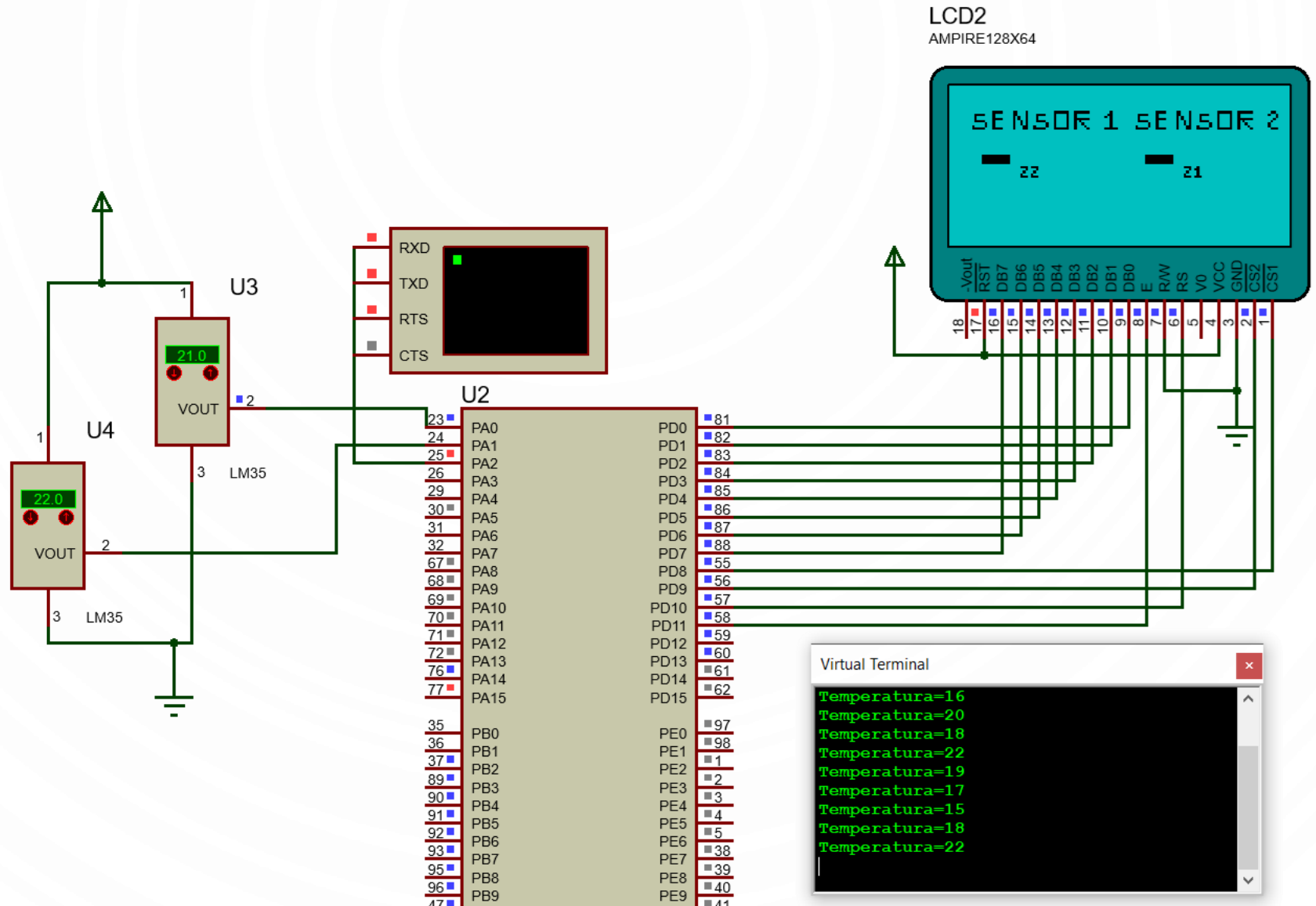
```
//////////////////// GLCD ( ADC )///////////////

#include "STM32F7xx.h"
#include "math.h"
void pixel(int , int );
void borrado(int a);
char setX= 0xB8;
char disp_on= 0x3F;
char disp_off= 0x3e;
char starL=0xC0;
char setY=0x40;
int csx,i, j;
int u,d,volt;

 char LT[8] = {0x01,0x01,0x01,0xff,0x01,0x01,0x01,0x00};
 char LE[8] = {0X00,0XFF,0X89,0X89,0X89,0X00,0X00,0X00};
 char LM[8] = {0x00,0xff,0x01,0x02,0x04,0x02,0x01,0xff};
 char LP[8] = {0x00,0xff,0x11,0x11,0x11,0x11,0x0e,0x00};

 char NUM[10][8] = { 0x00,0x7e,0x81,0x81,0x81,0x81,0x7e,0x00,
                     0x00,0x00,0x84,0x82,0xff,0x80,0x80,0x00,
                     0x00,0x00,0x62,0x91,0x89,0x86,0x00,0x00,
                     0x00,0x00,0x42,0x91,0x99,0x66,0x00,0x00,
                     0x00,0x0c,0x0a,0x09,0xff,0x08,0x00,0x00,
                     0x00,0x06,0x89,0x89,0x89,0x89,0x70,0x00,
                     0x00,0x00,0x30,0x4c,0x8a,0x91,0x60,0x00,
                     0x00,0x00,0x01,0x11,0x11,0xff,0x00,0x00,
                     0x00,0x20,0x56,0x89,0x89,0x56,0x20,0x00,
                     0x00,0x00,0x06,0x89,0x49,0x36,0x00,0x00};
```

ADC EN GLCD

```
int main(void){
RCC->AHB1ENR = 0xf;
RCC -> APB2ENR |= 0X400;
GPIOA -> MODER |= 3;
ADC3 -> CR1 |= 0X0;
ADC3 -> CR2 |= 0X401;
GPIOD->MODER =0x05555555;
send_comando(1,starL);     send_comando(2,starL);
send_comando(1,setY);     send_comando(2,setY);
send_comando(1,setX);     send_comando(2,setX);
send_comando(1,disp_on); send_comando(2,disp_on);
borrado(0);
while(true){
      ADC3 -> SQR3 =0;
      ADC3 -> CR2 |= 0X40000000;
      while((ADC3->SR &=0X2)==1);
      volt=(ADC3 -> DR)*805/100000;
      u=volt%10;d=volt/10;
      send_comando(1,setY+5);
      send_comando(1, 3| setX);
      for(j = 0; j < 8; j++) {send_dato(1,LT[j]);}
      for(j = 0; j < 8; j++) {send_dato(1,LE[j]);}
      for(j = 0; j < 8; j++) {send_dato(1,LM[j]);}
      for(j = 0; j < 8; j++) {send_dato(1,LP[j]);}
      send_comando(1,setY+45);
      for(j = 0; j < 8; j++) {send_dato(1,NUM[d][j]);}
      for(j = 0; j < 8; j++) {send_dato(1,NUM[u][j]);}
      send_comando(1, 5| setX);  send_comando(1,setY+5);
      for(j = 0; j < volt; j++) {send_dato(1,0xff);}
      for(int Cont=0;Cont<500000;Cont++);
      borrado(0);
}}
```



ADC + comunicación serial, ejemplo 3



ADC + comunicación serial, ejemplo 3

```
/////////////// VISUALIZA SERIAL ADC GLCD//////////////

#include "STM32F4xx.h"
#include "math.h"
void textol(void);
void borrado(int a);
char setX= 0xB8;
char disp_on= 0x3F;
char disp_off= 0x3e;
char starL=0xC0;
char setY=0x40;
int temp,tempA,csx,i, j,u,d,ul,dl;
float v=2;
char enviol[] = {"Temperatura= "};
void send(){
    for ( char i=0;i < 12;i++){
      USART2->DR = enviol[i]; while ((USART2->SR &= 0x80)==0); }
      USART2->DR = d+0x30;  while ((USART2->SR &= 0x80)==0);
      USART2->DR = u+0x30;  while ((USART2->SR &= 0x80)==0);
      USART2->DR = 0x0d;     while ((USART2->SR &= 0x80)==0);

  }
```

ADC + comunicación serial, ejemplo 3

```
char LA[8] = {0x00,0xfc,0x12,0x11,0x11,0x12,0xfc,0x00};
char LB[8] = {0x00,0xff,0x89,0x89,0x89,0x89,0x76,0x00};
char LC[8] = {0X00,0X3C,0X42,0X81,0X81,0X81,0X81,0X00};
char LD[8] = {0x00,0xff,0X81,0X81,0X81,0X42,0X3C,0x00};
char LE[8] = {0X00,0XFF,0X89,0X89,0X89,0X00,0X00,0X00};
char LF[8] = {0X00,0XFF,0X09,0X09,0X09,0X09,0X01,0X00};
char LG[8] = {0x00,0X7E,0X89,0X89,0X89,0X89,0X70,0x00};
char LH[8] = {0X00,0XFF,0X08,0X08,0X08,0X08,0XFF,0X00};
char LI[8] = {0X00,0X00,0X81,0X81,0XFF,0X81,0X81,0X00};
char LJ[8] = {0x00,0X81,0X81,0X81,0X7F,0X01,0X01,0x00};
char LK[8] = {0x00,0xff,0x18,0x14,0x22,0x41,0x80,0x00};
char LL[8] = {0x00,0xff,0x80,0x80,0x80,0x80,0x00,0x00};
char LM[8] = {0x00,0xff,0x01,0x02,0x04,0x02,0x01,0xff};
char LN[8] = {0x00,0xff,0x03,0x0c,0x30,0xc0,0xff,0x00};
char LO[8] = {0x00,0xff,0x81,0x81,0x81,0x81,0xff,0x00};
char LP[8] = {0x00,0xff,0x11,0x11,0x11,0x11,0x0e,0x00};
char LQ[8] = {0x00,0x7e,0x81,0x81,0xa1,0x7e,0x80,0x00};
char LR[8] = {0x00,0xff,0x09,0x19,0x29,0x49,0x89,0x00};
char LS[8] = {0x00,0xc0,0x9e,0x92,0x92,0x92,0x60,0x00};
char LT[8] = {0x01,0x01,0x01,0xff,0x01,0x01,0x01,0x00};
char LU[8] = {0x00,0x7f,0x80,0x80,0x80,0x7f,0x80,0x00};
char LV[8] = {0x03,0x0c,0x30,0xc0,0xc0,0x30,0x0c,0x03};
char LW[8] = {0x07,0x38,0xc0,0x30,0xc0,0x38,0x07,0x00};
char LX[8] = {0x81,0x42,0x24,0x18,0x18,0x24,0x42,0x81};
char NO[8] = {0x00,0xff,0x81,0x81,0x81,0x81,0xff,0x00};
char N1[8] = {0x00,0x00,0x84,0x82,0xff,0x80,0x80,0x00};
char N2[8] = {0x00,0x00,0x62,0x91,0x89,0x86,0x00,0x00};
char N3[8] = {0x00,0x00,0x42,0x91,0x99,0x66,0x00,0x00};
char N4[8] = {0x00,0x0c,0x0a,0x09,0xff,0x08,0x00,0x00};
char N5[8] = {0x00,0x06,0x89,0x89,0x89,0x89,0x70,0x00};
char N6[8] = {0x00,0x00,0x30,0x4c,0x8a,0x91,0x60,0x00};
char N7[8] = {0x00,0x00,0x01,0x11,0x11,0xff,0x00,0x00};
char N8[8] = {0x00,0x20,0x56,0x89,0x89,0x56,0x20,0x00};
char N9[8] = {0x00,0x00,0x06,0x89,0x49,0x36,0x00,0x00};
```

```
char num[10][4]={ 0x1f,0x11,0x1f,0x00,//0
                  0x12,0x1f,0x10,0x00,//1
                  0x19,0x15,0x13,0x00,//2
                  0x11,0x15,0x1f,0x00,//3
                  0x07,0x04,0x1f,0x00,//4
                  0x13,0x15,0x19,0x00,//5
                  0x1f,0x15,0x1c,0x00,//6
                  0x01,0x05,0x1f,0x00,//7
                  0x1f,0x15,0x1f,0x00,//8
                  0x07,0x05,0x1f,0x00};//9
```

ADC + comunicación serial, ejemplo 3

```
void send_comando(int cs, char a){
GPIOD->ODR =a;//RS=0
if(cs==1){GPIOD ->ODR |=(1UL<<8);} // Enable CS1
else if(cs==2){GPIOD ->ODR |=(1UL<<9);}//Enable CS2
GPIOD->ODR |=(1UL<<11); // Enable 1
for(int Cont=0;Cont<200;Cont++);
GPIOD->ODR &=~(1UL<<11); // Enable 0
GPIOD->ODR =0; // cs 0
}

void send_dato(int cs, char b){
GPIOD->ODR |=(1UL<<10);//RS=1
if(cs==1){GPIOD ->ODR |=(1UL<<8);} // Enable CS1
else if(cs==2){GPIOD ->ODR |=(1UL<<9);}//Enable CS2
GPIOD ->ODR |= b;
GPIOD->ODR |=(1UL<<11); // Enable 1
for(int Cont=0;Cont<200;Cont++);
GPIOD->ODR &=~(1UL<<11); // Enable 0
GPIOD->ODR =0; // cs 0
}
```

ADC + comunicación serial, ejemplo 3

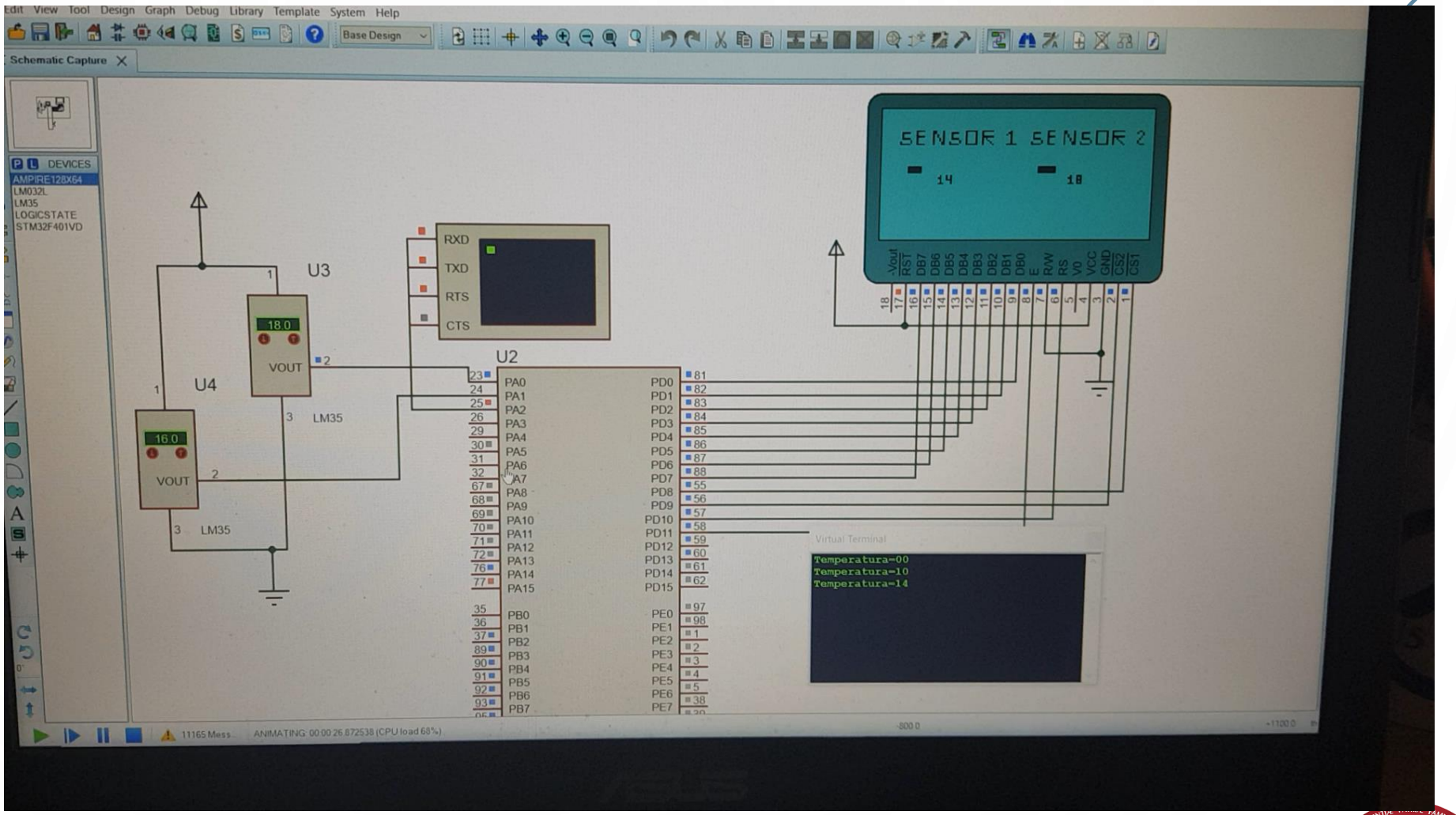
```
int main(void){
RCC->APB1ENR |= (1UL<<17); // Enable clock for USART2 (set el pin 17)
RCC->AHB1ENR = 15;
RCC -> APB2ENR = 0X100; //HABILITAR EL ADC 1
GPIOB -> MODER = 0X55555555; //COLOCAR EN SALIDA PARA ENCENDER LOS LEDS
GPIOA -> MODER |= 0x2F; //ANALOGO PARA EL PIN 0/1 -- PIN 2 alterno
GPIOA->AFR[0] |=0x700;
ADC1 -> CR1 = 0X0; //RESOLUCION DE 12 BITS (POR DEFECTO)
ADC1 -> CR2 = 0X1; //ENCENDER EL ADC
  //CONFIGURACION UART

  USART2->BRR = 0x683;   // 9600 Baudios, fclk=16Mhz, por defecto el HSI
  USART2->CR1 = 0x012C;  // Tx habilitado, Rx habilitado, Interrup de Rx h
  USART2->CR1 |= 0x2000; // CR2 se deja por defecto pues 1 bit de stop es
  NVIC_EnableIRQ(USART2_IRQn);      // Llama interrupción del USART2

GPIOD->MODER =0x05555555;
send_comando(1,starL);
send_comando(2,starL);
send_comando(1,setY);
send_comando(2,setY);
send_comando(1,setX);
send_comando(2,setX);
send_comando(1,disp_on);
send_comando(2,disp_on);
borrado(0);
tempA=0;
```

ADC + comunicación serial, ejemplo 3

```
while(true){

    ADC1 -> SQR3 = 0; //CANAL 0 DEL ADC PIN PA0
    ADC1 -> CR2 |= (1UL<<30); //0X40000000; //INICIAR LA CONVERSION EN CANALES REGULARES
    while((ADC1->SR & 0X2)==1);
    temp=((ADC1->DR*805)/10000);
    if((temp>=(tempA*1.2)) || (temp<=(tempA*0.9))){tempA=temp; //    temp +  temp*0.1=1.1*temp
      u=temp%10;d=temp/10;
      send();
      send_comando(2, 3| setX);send_comando(2,setY);
      for(j=0;j< 100;j++) {send_dato(2,0);}
      send_comando(2, 3| setX);send_comando(2,setY+10);
      for(j=0;j< temp;j++) {send_dato(2,0x78);}
      textol();
    }
    ADC1 -> SQR3 = 1; //CANAL 1 DEL ADC PIN PA1
    ADC1 -> CR2 |= (1UL<<30); //0X40000000; //INICIAR LA CONVERSION EN CANALES REGULARES
    while((ADC1->SR & 0X2)==1);
    temp=((ADC1->DR*805)/10000);
    ul=temp%10;dl=temp/10;
      send_comando(1, 3| setX);send_comando(1,setY);
      for(j=0;j< 100;j++) {send_dato(1,0);}
      send_comando(1, 3| setX);send_comando(1,setY+10);
      for(j=0;j< temp;j++) {send_dato(1,0x78);}
      for(j=0;j<5000;j++);
      send_comando(1,setY+25);
      send_comando(2,setY+25);
      send_comando(1, 4| setX);
      send_comando(2, 4| setX);
        for(j = 0; j < 4; ++j)
      { send_dato(2,num[d][j] );
        send_dato(1,num[dl][j] );
        }
        for(j = 0; j < 4; ++j)
      { send_dato(2,num[u][j] );
        send_dato(1,num[ul][j] );
        }
      for(j=0;j<50000;j++);
}}
```

ADC + comunicación serial, ejemplo 3

```
void textol( void)
{         send_comando(1,setY+5);
    send_comando(2,setY+5);
    send_comando(1, 1| setX);
    send_comando(2, 1| setX);
    for(j = 0; j < 8; ++j)
    {  send_dato(1,LS[j] );
       send_dato(2,LS[j]  );}
    for(j = 0; j < 8; ++j)
    {  send_dato(1,LE[j] );
       send_dato(2,LE[j]  );}
    for(j = 0; j < 8; ++j)
    {  send_dato(1,LN[j] );
       send_dato(2,LN[j]  );}
    for(j = 0; j < 8; ++j)
    {  send_dato(1,LS[j] );
       send_dato(2,LS[j]  );
      }
    for(j = 0; j < 8; ++j)
    { send_dato(1,LO[j] );
      send_dato(2,LO[j]  );
      }
    for(j = 0; j < 8; ++j)
    { send_dato(1,LR[j] );
      send_dato(2,LR[j]  );
      }
      for(j = 0; j < 3; ++j)
    { send_dato(1,0 );
      send_dato(2,0 );
      }
    for(j = 0; j < 8; ++j)
    { send_dato(1,N2[j] );
      send_dato(2,N1[j]  );
      }
}
```



Uso de librerias

En informática, una **biblioteca** o, llamada por vicio del lenguaje **librería** (del inglés *library*) es un conjunto de implementaciones funcionales, codificadas en un lenguaje de programación, que ofrece una interfaz bien definida para la funcionalidad que se invoca.

https://es.wikipedia.org/wiki/Biblioteca_(inform%C3%A1tica)#:~:text=En%20inform%C3%A1tica%2C%20una%20biblioteca%20o,la%20funcionalidad%20que%20se%20invoca.

http://diwo.bq.com/librerias-arduino/

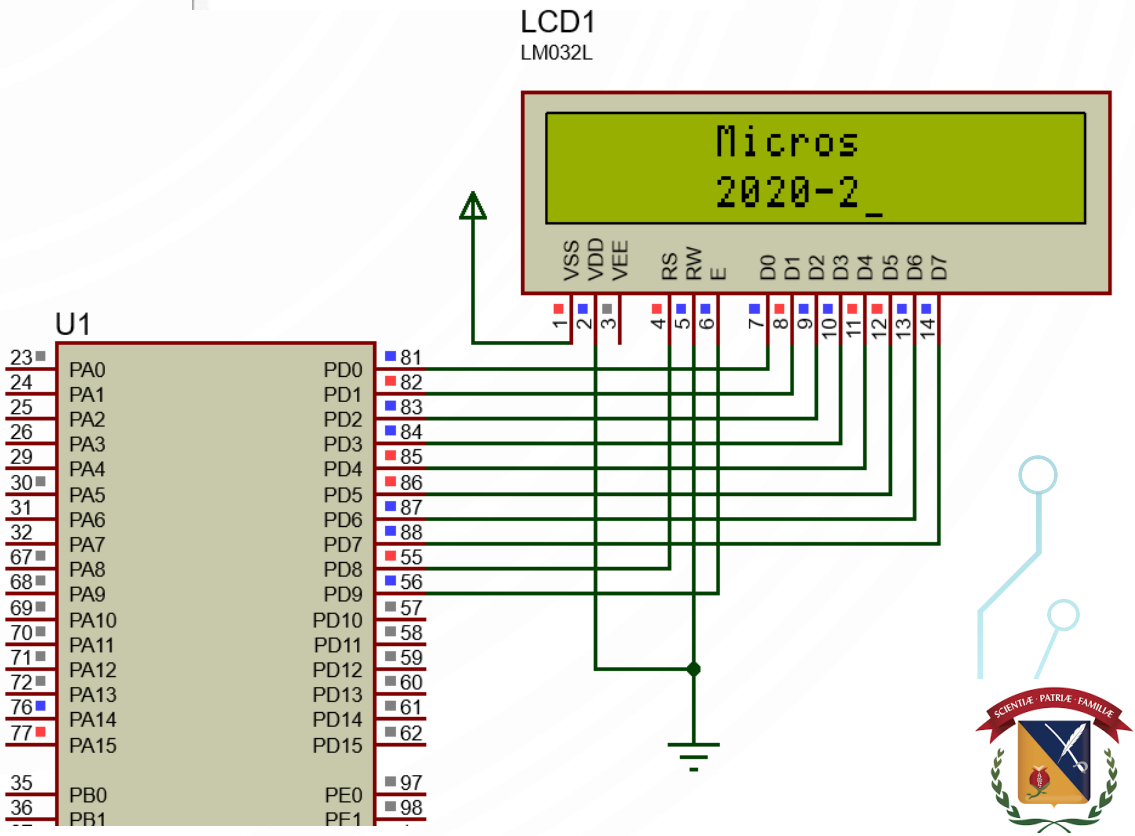
```c
#include "STM32F4xx.h"
char dato[6]={'M','i','c','r','o','s'};
char dato2[6]={'2','0','2','0','-','2'};
char clear=0x01;
char LCD1L=0x80;
char LCD2L=0xC0;

void send_comando(char b){
   GPIOD ->ODR &=0XFF00;
   GPIOD ->ODR |= b;
   GPIOD->ODR &=~(1UL<<8);//RS=0
   GPIOD->ODR |=(1UL<<9); // Enable 1
   for(int dl=0;dl<5000;dl++);
   GPIOD->ODR &=~(1UL<<9); // Enable 0
}
void send_dato(char b){
   GPIOD ->ODR &=0XFF00;
   GPIOD ->ODR |= b;
   GPIOD->ODR |=(1UL<<8);//RS=1
   GPIOD->ODR |=(1UL<<9); // Enable 1
   for(int dl=0;dl<5000;dl++);
   GPIOD->ODR &=~(1UL<<9); // Enable 0
}
void cfglcd(){
   RCC->AHB1ENR = 8;
   GPIOD->MODER =0x55555;
   //send_comando(clear);
   send_comando(0x38);
   send_comando(0x0E);
   send_comando(0x06);
}
```

Función 1

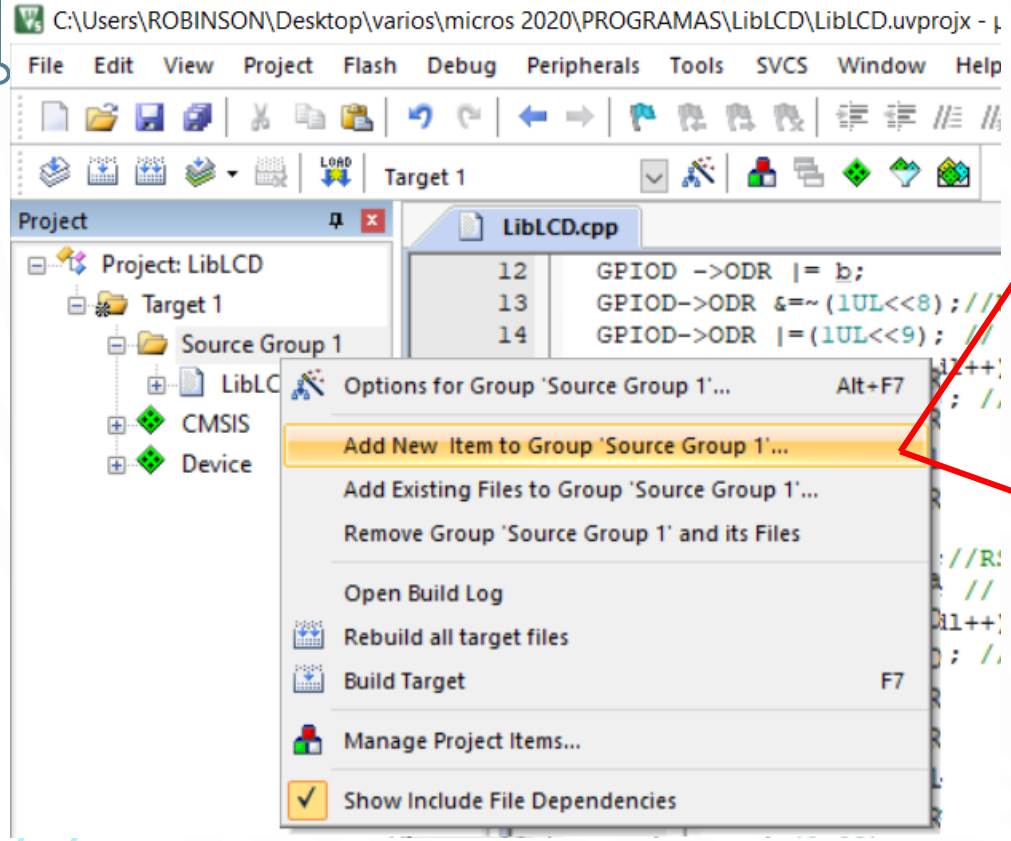
Función 2

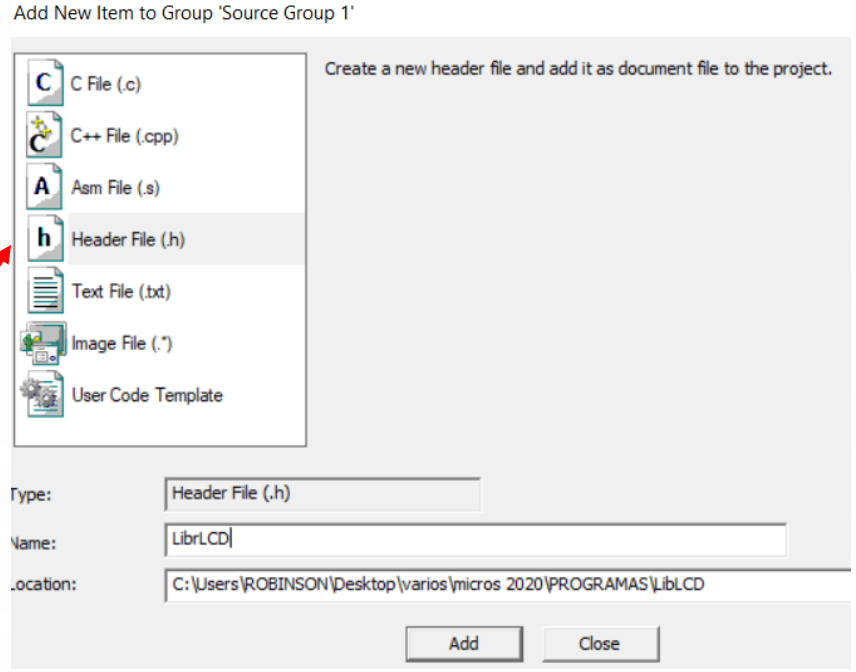
Función 3

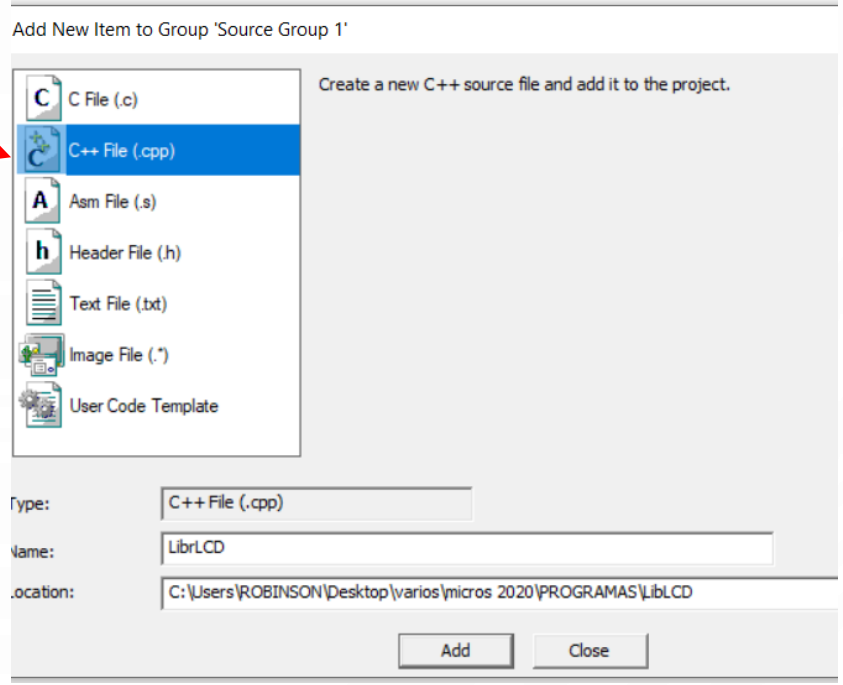
```c
int main(void){
cfglcd();
while(true){
send_comando(LCD1L+6);
for (int i=0;i<6;i++){
  send_dato(dato[i]); }
 send_comando(LCD2L+6);
for (int i=0;i<6;i++){
  send_dato(dato2[i]);}
for (int i=0;i<600000;i++);
}
}
```



Creación de librerias







Creación de librerias

LibLCD.cpp | LibrLCD.cpp | LibrLCD.h

```
1  ////////////// dato 2 lineas fijo////////
2
3  #include "STM32F4xx.h"
4  #include "LibrLCD.h"
5  LibrLCD lib;
6  char dato[6]={'M','i','c','r','o','s'};
7  char dato2[6]={'2','0','2','0','-','2'};
8  char clear=0x01;
9  char LCD1L=0x80;
10 char LCD2L=0xC0;
11
12
13 int main(void){
14 lib.cfglcd();
15 while(true){
16 lib.send_comando(LCD1L+6);
17 for (int i=0;i<6;i++){
18   lib.send_dato(dato[i]); }
19  lib.send_comando(LCD2L+6);
20 for (int i=0;i<6;i++){
21   lib.send_dato(dato2[i]);}
22 for (int i=0;i<600000;i++);
23 }
24 }
25
```

bLCD.cpp | LibrLCD.cpp | LibrLCD.h

```
#include "STM32F4xx.h"
class LibrLCD{
public:
void cfglcd();
void send_comando(char);
void send_dato(char);
};
```

bLCD.cpp | LibrLCD.cpp | LibrLCD.h

```
#include "LibrLCD.h"
void LibrLCD::send_comando(char b){
  GPIOD ->ODR &=0XFF00;
  GPIOD ->ODR |= b;
  GPIOD->ODR &=~(1UL<<8);//RS=0
  GPIOD->ODR |=(1UL<<9); // Enable 1
  for(int dl=0;dl<5000;dl++);
  GPIOD->ODR &=~(1UL<<9); // Enable 0
}
void LibrLCD:: send_dato(char b){
  GPIOD ->ODR &=0XFF00;
  GPIOD ->ODR |= b;
  GPIOD->ODR |=(1UL<<8);//RS=1
  GPIOD->ODR |=(1UL<<9); // Enable 1
  for(int dl=0;dl<5000;dl++);
  GPIOD->ODR &=~(1UL<<9); // Enable 0
}
void LibrLCD:: cfglcd(){
  RCC->AHB1ENR = 8;
  GPIOD->MODER =0x55555;
  //send_comando(clear);
  send_comando(0x38);
  send_comando(0x0E);
  send_comando(0x06);
}
```

Creación de librerias

```
#include "STM32F4xx.h"
char dato[6]={'M','i','c','r','o','s'};
char dato2[6]={'2','0','2','0','-','2'};
char clear=0x01;
char LCD1L=0x80;
char LCD2L=0xC0;

void send_comando(char b){
   GPIOD ->ODR &=0XFF00;
   GPIOD ->ODR |= b;
   GPIOD->ODR &=~(1UL<<8);//RS=0
   GPIOD->ODR |=(1UL<<9); // Enable 1
   for(int dl=0;dl<5000;dl++);
   GPIOD->ODR &=~(1UL<<9); // Enable 0
}
void send_dato(char b){
   GPIOD ->ODR &=0XFF00;
   GPIOD ->ODR |= b;
   GPIOD->ODR |=(1UL<<8);//RS=1
   GPIOD->ODR |=(1UL<<9); // Enable 1
   for(int dl=0;dl<5000;dl++);
   GPIOD->ODR &=~(1UL<<9); // Enable 0
}
void cfglcd(){
   RCC->AHB1ENR = 8;
   GPIOD->MODER =0x55555;
   //send_comando(clear);
   send_comando(0x38);
   send_comando(0x0E);
   send_comando(0x06);
}

int main(void){
cfglcd();
while(true){
send_comando(LCD1L+6);
for (int i=0;i<6;i++){
  send_dato(dato[i]); }
 send_comando(LCD2L+6);
for (int i=0;i<6;i++){
  send_dato(dato2[i]);}
for (int i=0;i<600000;i++);
}
}
```

```
#include "STM32F4xx.h"
#include "LibrLCD.h"
LibrLCD lib;
char dato[6]={'M','i','c','r','o','s'};
char dato2[6]={'2','0','2','0','-','2'};
char clear=0x01;
char LCD1L=0x80;
char LCD2L=0xC0;


int main(void){
lib.cfglcd();
while(true){
lib.send_comando(LCD1L+6);
for (int i=0;i<6;i++){
  lib.send_dato(dato[i]); }
 lib.send_comando(LCD2L+6);
for (int i=0;i<6;i++){
  lib.send_dato(dato2[i]);}
for (int i=0;i<600000;i++);
}
}
```

INGENIERÍA MECATRÓNICA UMNG

UNIVERSIDAD MILITAR NUEVA GRANADA

SERVO-TECLADO-GLCD

Ejercicio: diseñar un sistema mecatronico que lea por teclado matricial el angulo al que debe ubicarse un servomotor y visualizer dicho Angulo por matriz LCD grafica

```
#include "STM32F7xx.h"
#include "GLCD.h"
GLCD lib;

void cfgTeclado();
int teclado();
void inicio();
void servo();

bool alerta=false;
int paso,rx,i,j,dt,ut,tc,u,d,dist,alarm,volt,volta,Temp,delta;
int cont=0,a=0,tp=0,um=0,dm=0,uh=0,dh=0,ud=0,dd=0,uM=0,dM=0,kp=200;
int hora=0,fecha=0;
int dat[2]={0,0};
char setY=0x40,setX=0xB8;
char num[10][4]={ 0x1f,0x11,0x1f,0x00,//0
char numR[10][8]={0x00,0xff,0x81,0x81,0x81,0x81,0xff,0x00,
char micros[20]= {0x77,0x14,0x27,0x10,0x77,0x01,0x72,0x01,0x77,0x50,
char umng2 [] = {
char meca [] = {
char menuL [] = {
char menuR [] = {
char servoL [] = {
////////////   PROGRAMA PRINCIPAL  ////////////////
int main(void){
  lib.cfgGLCD();
  cfgTeclado();
  inicio();
while(true){
  servo();
    }
}
```