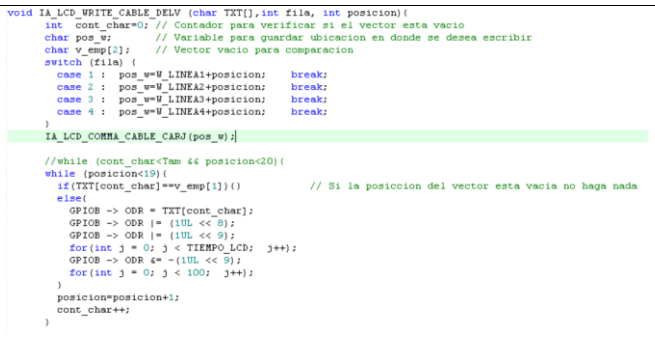
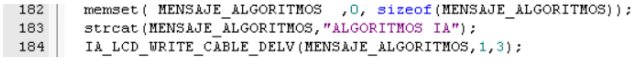
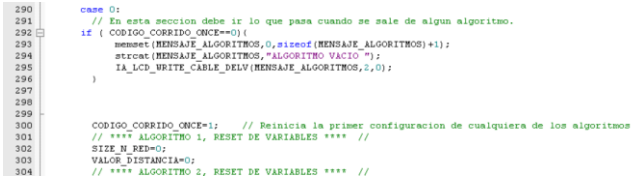
Unit Test Procedure and Report (Peripherals configuration)

Hardware Configuration	Microcontroller test (Flat configuration)
Software Configuration	Structured Project (Algorithms Functions)
Test Input	Configuration for peripherals functions.
Requirements	Keil uVision 5 and the STM32F07VG Board. Peripherals: Bluetooth module and an LCD
Instructions	- Connect the STM32F7 to a computer using the micro USB port. - For testing make sure that all variables had been initialized as you want. - Clean and check the variables you want to follow using the Debugger. - Build and program.
Special Notes	The test result for the peripherals configuration unit test is to create different classes to call in the main code in order to setup the maximum speed for the microcontroller and GPIO (General Purpose Input/Output) configuration to connect peripherals as a Bluetooth or an LCD.
Expected test results	Connect the microcontroller with an external GUIDE using Bluetooth, and visualize the changes in the GUIDE using the LCD connected to the microcontroller.
Test Engineer	David Enrique Lizarazo Vesga
QA	Robinson Jiménez Moreno

Instructions	Notes	Log	Test	QA
Power on spacecraft			X	
Run unit test	Clases_Leds		X	
Previous Instructions		Build time is: Jul 23 2019 **Open the project and check code:** Once you have opened the project and as explained in previous Unit test, you could find the classes codes in the project tree in the left side of the program. To run this Unit test, make sure that the header codes shown below are linked or imported correctly in the project.	X	

		`` #include <stdio.h> #include <stdlib.h> #include "stm32f7xx.h" #include "ALGORITMOS/CONFIGURACION/CONF.h" #include "ALGORITMOS/ALGORITMO_1/ALG1.h" #include "ALGORITMOS/ALGORITMO_2/ALG2.h" #include "ALGORITMOS/ALGORITMO_3/ALG3.h" #include "ALGORITMOS/ALGORITMO_4/ALG4.h" #include "ALGORITMOS/OPERACIONES/OPE.h" #include "ALGORITMOS/LCD/LCDFUN.h" ``` As you can see there are the algorithms codes as well as a class that helps to configurate the microcontroller, do matrix simple operations and setup the necessary GPIO for the LCD to work. To be cleared with the electrical connections in the flat configuration the next part of the code is the enumeration of the GPIO used to reproduce the code. This also helps to create the layout of the PCB when all the peripherals that are going to be used were set. ``` /* *** PINOUT PF3 -> ADC3 PC10 -> ES UART4 TX (Esquina superior tarjeta) -> Va a RX BLUETOOTH PC11 -> ES UART4 RX (Esquina superior tarjeta) -> Va a TX Bluetooth PA4 -> DAC (Chanel 1) PB0-PB9 -> LCD PD8 -> DHT11 */ ```		
Previous instructions		Build time is: **Set Peripherals:** All the peripherals that are going to be used have been set in different classes. This allows us to decelerate different functions to all the peripherals. This can be seen in the configuration functions for every peripheral, where different processes are done. Another strategy that has been used is the internal call of functions, this helps to minimize the code that must be call in the main code. ``` memset(MENSAJE_ALGORITMOS ,0, sizeof(MENSAJE_ALGORITMOS)); strcat(MENSAJE_ALGORITMOS,"ALGORITMOS IA"); IA_LCD_WRITE_CABLE_DELV(MENSAJE_ALGORITMOS,1,3); ``` In this example we had to do some previous operations to call the function “IA_LCD_WRITE_CABLE_DELV”, this must be done to avoid warnings given by the compiler. But if we check inside the function, you can find other function of the same class that is used.		

		 The use of constants is other useful practice that result in a more efficient and fast code. This is because when you create a constant variable (const int NAME=VALUE;) the compiler processes it with the respective steps as it was a common variable. Otherwise the macros and define values, just tell the compiler that where it finds the word defined it must replace it with the value given in the definition (#define NAME value).		
Previous instructions		Build time is: **Bluetooth:** To use the serial communication, some points have to be understood before working with this. In the interruptions (extern C) , you will find the UART4_IRQHandler functions, which have the process that are going to be do when a serial data is received. In this function have been clarified the order and data that have to be received. The reception data structure: X1 SIGN V1 VF X1 -> The variable to define what is the data type that are going to be received (A selection of an algorithm, parameters …) SIGN -> “+” or “- “ V1 -> the value, it must be 6 characters long. For example, if you want to send a single 1, it will be sent as 2 integer part, and other 4 decimal. So, a number 1, would have to be send as 010000 VF -> is the last data send. To specify that the communication has ended. It had been set as ‘F’		

Previous instructions		Build time is: **LCD:** This class have been created to work with an LCD that is connected to the B port, or I2C module , however the I2C module configuration hasn't been tried yet, so the test have to be done connecting the LCD pins to the STM. The main functions that must call to use the LCD are: IA_LCD_CONF_CABLEA_CARJ (void) -> Function used to set the configuration of the LCD. IA_LCD_WRITE_CABLE_DELV (char TXT [],int fila, int posicion) -> Function used to write in an specified place of the LCD (this will only write in a single row, so the long of the char send to the function must be 20 characters long if the initial position is the 0 column). You can also use the function IA_LCD_COMMA_CABLE_CARJ (char com), and use the define constants declared in LCD.h to specify an action with the LCD.		
Record test results		Build time is: The test consists in running the code using different peripherals. In this case are going to be used a LCD, and a Bluetooth. The hardware and electrical configuration set in this test is a flat cabled connection. To test the code would be first be run the code without a serial connection, to validate the LCD performance. Once you check the code, you would find 2 moments where the LCD function is called. After the configuration (To write a tittle)  And a second one, given when the code does not find and algorithm specified to work in, where some variables are set and LCD write that there is none algorithm running. 		

```
182   memset( MENSAJE_ALGORITMOS  ,0, sizeof(MENSAJE_ALGORITMOS));
183   strcat(MENSAJE_ALGORITMOS,"ALGORITMOS IA");
184   IA_LCD_WRITE_CABLE_DELV(MENSAJE_ALGORITMOS,1,3);
```

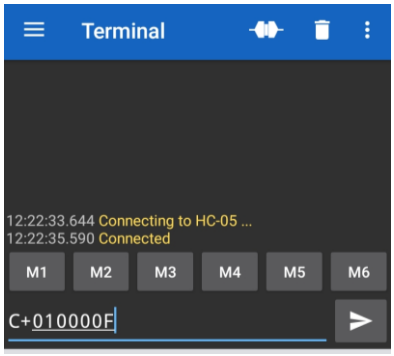
```
290       case 0:
291         // En esta seccion debe ir lo que pasa cuando se sale de algun algoritmo.
292       if ( CODIGO_CORRIDO_ONCE==0){
293             memset(MENSAJE_ALGORITMOS,0,sizeof(MENSAJE_ALGORITMOS)+1);
294             strcat(MENSAJE_ALGORITMOS,"ALGORITMO VACIO ");
295             IA_LCD_WRITE_CABLE_DELV(MENSAJE_ALGORITMOS,2,0);
296          }
297
298
299
300         CODIGO_CORRIDO_ONCE=1;    // Reinicia la primer configuracion de cualquiera de los algoritmos
301         // **** ALGORITMO 1, RESET DE VARIABLES ****  //
302         SIZE_N_RED=0;
303         VALOR_DISTANCIA=0;
304         // **** ALGORITMO 2, RESET DE VARIABLES ****  //
```

So when this code is running the correct response is the shown below.

After compiling and getting the same result, we will test the communication using the LCD.
So, to understand the result you would have to see that after sending the data for stablish connection in the STM, the LCD would have to write the state. (Remember the structure of the serial information, and how it has to be send)

```
case 'C':              /*  *** OPCION DE CONECCION *** */
    memset(MENSAJE_ALGORITMOS,0,sizeof(MENSAJE_ALGORITMOS)+1);
    strcat(MENSAJE_ALGORITMOS,"CONECTADO");
    IA_LCD_WRITE_CABLE_DELV(MENSAJE_ALGORITMOS,4,10);
    FLAG_RX=0;
break;
```

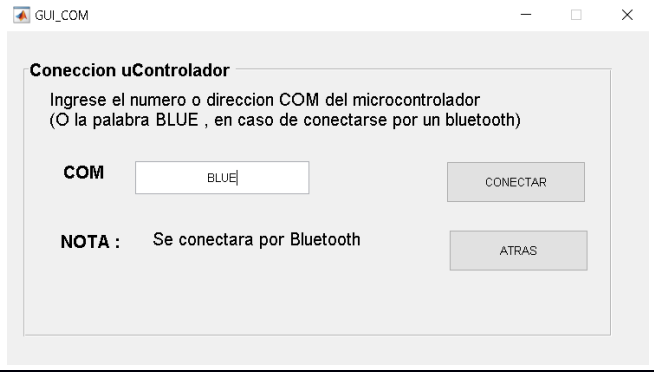
So, to test the serial communication, would be use an application that can be download in any android device (with Bluetooth) , and a created GUI using Matlab.
The first test will be made with the “Serial Bluetooth Terminal” application. Once you download it and specify the device you want to connect with, you would see something like the image below. (connecting with the Device).

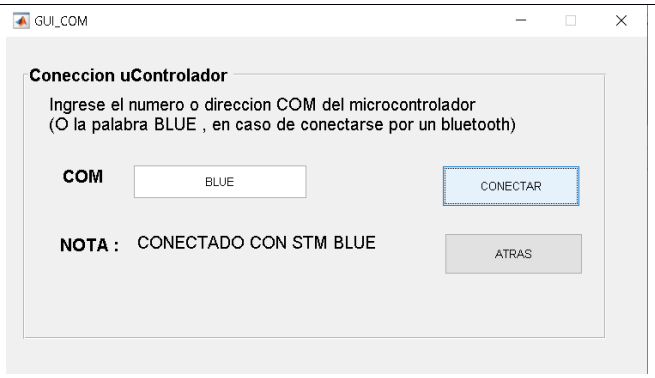


Before sending the data with serial, you will see again the same result as LCD.

After sending the data you will see in the fourth row that the LCD display the word “CONECTADA” , which is the way to know that the connection has been completed succesfully.

As the last test, you can obtain the same result with the LCD by using the GUI created in Matlab.
Using the GUI will be shown a POP-UP where you can connect with the STM using the Bluetooth. This GUI do already have the serial configuration to communicate with the STM if you follow the instructions showed in the POP-UP.




GUI_COM
Coneccion uControlador
Ingrese el numero o direccion COM del microcontrolador
(O la palabra BLUE , en caso de conectarse por un bluetooth)
COM
BLUE
CONECTAR
NOTA :
CONECTADO CON STM BLUE
ATRAS