```c
/*
                   _____ ____  _   _____    ____     ______  __
                  / ___// __ \/ | / /   |  / __ \   / __ ) \/ /
                  \__ \/ / / /  |/ / /| | / /_/ /  / __  |\  / 
                 ___/ / /_/ / /|  / ___ |/ _, _/  / /_/ / / /  
                /____/\____/_/ |_/_/  |_/_/ |_|  /_____/ /_/   

                _______    _____       _________    __ _    ______ 
               / ____/ |  / /   |     / ____/   |  / /| |  / / __ \
              / __/  | | / / /| |    / /   / /| | / / | | / / / / /
             / /___  | |/ / ___ |   / /___/ ___ |/ /__| |/ / /_/ / 
            /_____/  |___/_/  |_|   \____/_/  |_/_____/___/\____/  

                                 ___    _   ______ 
                                /   |  / | / / __ \
                               / /| | /  |/ / / / /
                              / ___ |/ /|  / /_/ / 
                             /_/  |_/_/ |_/_____/  

              _____ _    ________________     ____  ____  ______________
             / /   | |  / /  _/ ____/ __ \   / __ \/ __ \/_  __/  _/__  /
        __  / / /| | | / // // __/ / /_/ /  / / / / /_/ / / /  / /   / / 
       / /_/ / ___ | |/ // // /___/ _, _/  / /_/ / _, _/ / / _/ /   / /__
       \____/_/  |_|___/___/_____/_/ |_|   \____/_/ |_| /_/ /___/  /____/

*/


// Library of the LPC17.xx
#include <LPC17xx.H>

// Own libraries:
#include "modulos/timer05.h"
#include "modulos/keys.h"
#include "modulos/dac.h"
#include "modulos/screen.h"


// Global variables:
struct sonar_status sonar;                            // Struct that contais the state of the sonar.
uint16_t samples[N_SAMPLES];                          // Array that contains the samples of the DAC signal.

void config_priorities(void)
{
  /*
    config_priorities :: void -> void

    Set the priorities of all the
    interruptions that are used in
    the project, except the priority
    of the UART that is configured
    in its own configuration function.

  */

  NVIC_SetPriorityGrouping(3);                        // Only one bit is needed for the subpriority
  NVIC_SetPriority(TIMER3_IRQn,1);                    // UTS          -> (0,1).
  NVIC_SetPriority(TIMER0_IRQn,2);                    // 0.5 Timer    -> (1,0).
  NVIC_SetPriority(EINT0_IRQn, 4);                    // KEY ISP      -> (2,0).
  NVIC_SetPriority(EINT1_IRQn, 6);                    // KEY 1        -> (3,0).
  NVIC_SetPriority(EINT2_IRQn, 7);                    // KEY 2        -> (3,1).
  NVIC_SetPriority(TIMER1_IRQn,8);                    // DAC          -> (4,0).
}

int main(void)
{

  // Initialize the struct:
  sonar.state                = ST_SETUP;              // Sonar starts in Setup mode.
  sonar.distance             = 0;                     // Sonar distancie is initialize with a zero.
```

```c
    sonar.servo_pose          = 0;                        // The servo starts at zero degrees.
    sonar.servo_period        = 1;                        // The servo period is initialize with a period of a 0.5 seconds.
    sonar.servo_resolution    = 10;                       // The servo resolution is initialize with a resolution of 10 degrees.
    sonar.f_block_keys        = 0;                        // The flag f_block_keys is initialize with a zero.
    sonar.f_block_move        = 0;                        // The flag f_block_move is initialize with a zero because at beggining
                                                          // of the automatic mode the servo can move.
    sonar.f_block_measure     = 1;                        // The flag f_block_measure is initialize with one because at beggining
                                                          // of the manual mode the UTS can not move.
    sonar.f_block_transmision = 0;                        // The flag f_block_measure is initialize with zero because at beggining
                                                          // the transmision from the board via uart is enable in automatic mode.

    // Configure the hardware:
    config_timer05();
    config_keys();
    config_servo();
    config_UTS();
    config_DAC();
    config_timer_dac();
    config_priorities();
    LCD_Init();

    // Initialize the output
    // and the DAC Signal:
    generate_samples();                                   // Generate the samples of the sinusoidal signal of the DAC
    LCD_Clear(Blue);                                      // Fill the screen with blue
    set_servo(0);                                         // Initialize the servo pose

    while(1)                                              // Main loop:
    {
      sonar.f_block_keys = 0;                             // Clear the flag that blocks keys funcionalities.
      update_screen(&sonar);                              // Update the screen with the new status of the sonar.
      if(sonar.state == ST_AUTOMATIC)                     // If we are in automatic mode
        update_uart();                                    // We update the info via UART
    }

  }
```