# OSES ASSIGMENT #2

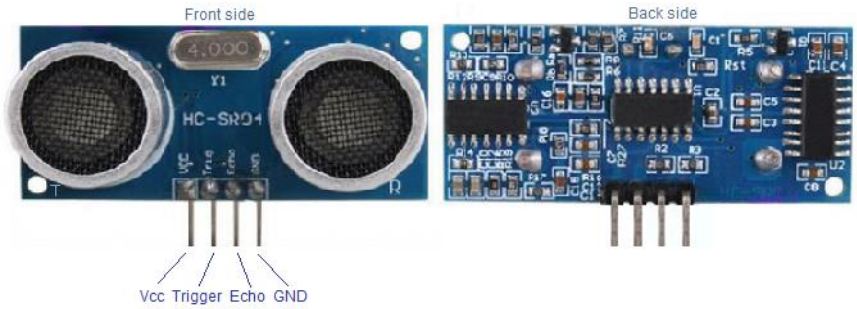
**Description**: The purpose of this assignment is to implement a proxity allert system using:

- The HC-SR04 Ultrasonic Ranging Module;
- The STM32F746-DISCOVERY board;
- The Linux OS.

Changing the led's lights and frequency accordig to documentation of assignment.

# Solution

## *1) HC-SR04 Ultrasonic Ranging Module*




- **Vcc**: 5V assigned to pin ***5V*** **(**Arduino connectors)

- **Trigger:** It's necessary to start the sensor generating a square wave at least whit a 10µs period, assigned to pin ***D7 (PI3)*** (in the code defined as *TRIG*). The pin assignment is:

  ***16*8(I)+3=131*** and this for other pins.

- **Echo:** After the trigger signal, the sensor generate a signal proportional to the distance to the nearest object. Formula:

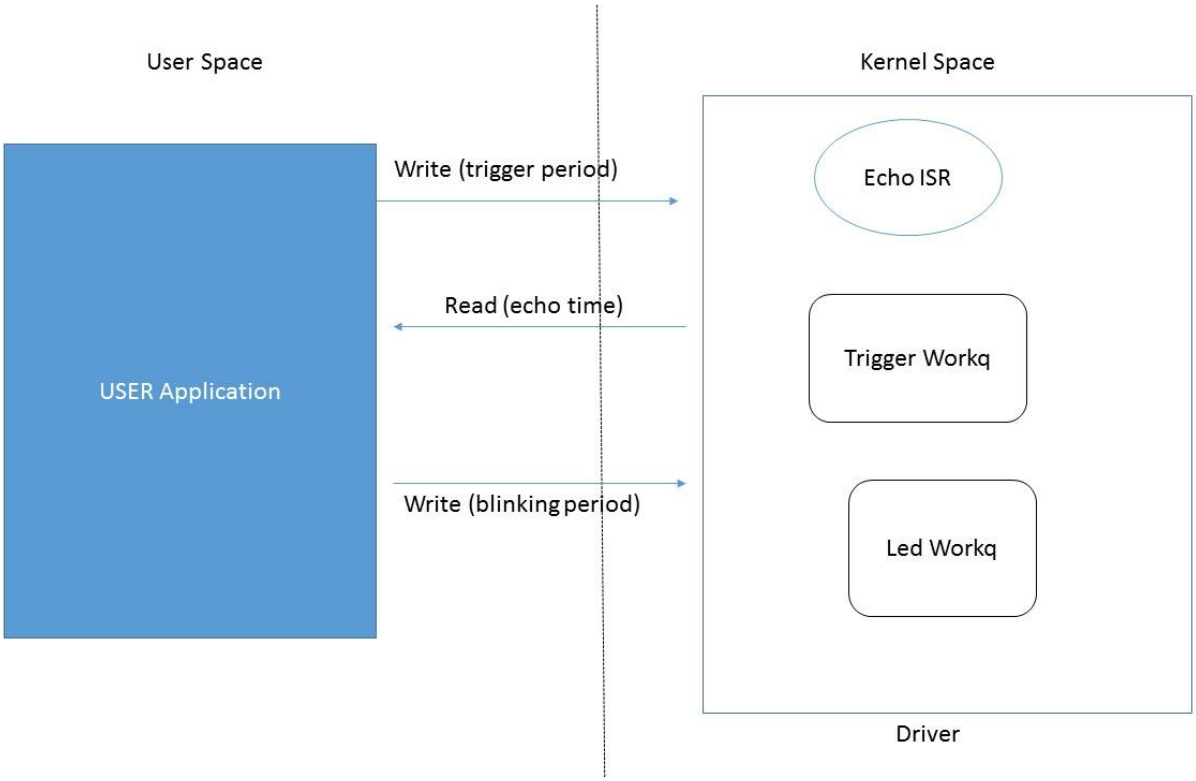
  ***Distance*** **=** Signal duration (in µs) / 58 (the result is in *cm* )

  Assigned to pin ***D8 (PI2)*** (in the code defined as *ECHO*)

- **GND:** assigned to one of the ground pins on the board (Arduino connectors)

## *2) Solution's schematic*

In this assignement we have to use Linux OS embedded. For our purpose we have to implement a device driver thath manage out board and sensor, and an application that use the primitives (read,open, close, write etc. etc.) in order to reach our purpose.




- **Kernel Space**

  In kernel space we have to develop the device driver that could manage our board implementing the specific primitives. In linux each device is seen as a file identified by a major and minor numer, so we've to override open and close function first of all. In initi_module we have to define the gpio, the semaphores, and workqueque that are a kind of kernel threads. In in the cleanup we free all the stuff that it was used. To reach our purpose is necessary to have two workq, one that generates a square wave in order to start the sensor measurement and another one that blink the led according to distance measured. Another thing that we need is the an ISR to evaluate the square wave of echo sensor that is proportional to distance measured. It's used some variables to control the correct reading synchonization.

  Device Driver ("*sample.c"*):

  - ***my_wq_function()*** : is the function that toggle the trigger pin in order to start the sensor***;***

  - ***led_wq_function()*** : is the function that blink the led with a certain blinking period

  - ***irq_handler_t echo_handler()*** : is the interrupt handler necessary to manage and measure the sensor Echo pin. I catch the rising value and save the timestamp, after that catch the time value on falling edge and save this value, in order to measure the elapsed time.

  - ***sample_write()*** : is the function that perform the writing on the device , according the the parameters (TRIGGER_GO,LED_GO,WRITE_PERIOD) , i can start trigger, led and set trigger period.

- ***sample_read()*** : is the function that perform the reading of the echo elapsed time in order to use it in user application

- ***sample_ioctl()*** : with this function i can choose which write operation i'm going to perform

- **User Space**

  In the user space we've to develop an application that can open the device driver, first of all, write the trigger period in order to start the measurement. After that we read the square wave duration throw the primitive read and manage this value in order to have a distance measured. In addition we have to start led's blinking and change the blinking period according the distance measured. At the end of this operations we could close the file.

## *3) Brief user manual*

In this part is described all the istructions to use this system.

- First thing to do is connect the target to the host device with serial and Ethernet cables. The first one is necessary to use the application and screen the results, the second one in needed in order to load the kernel image on the board.

- For this example i use a FTP server to load the kernel on the board, so it's necessary to configure correctly a FTP server on the host device and configure correctly the boot loader (in our case U-Boot) in order to have this comunication. After that we have to compile the application and link it with Linux kernel.

- After that the last thing to do is load the kernel on the board and launch the application using a terminal emulator and FTP server comunication. For our purpose we've implemented a kernel driver and an application. It's necessary to insert kernel module first and than the application.

  **N.B** *In order to use the shell linux command on the board it's necessary to customize Busy Box .*