# lab 3
# Ultrasonic sensor

M. Briday

November 18, 2019

## 1    Principle

This lab focuses on the interrupts, but still uses GPIOs and timers. The objective is to write the driver for an ultrasonic sensor, without any polling implementation.

The sensor embedds an integrated circuit for an easy management by the MCU. The interface consists in only 2 logic signals (*Trigger* and *echo*), as in figure 1.
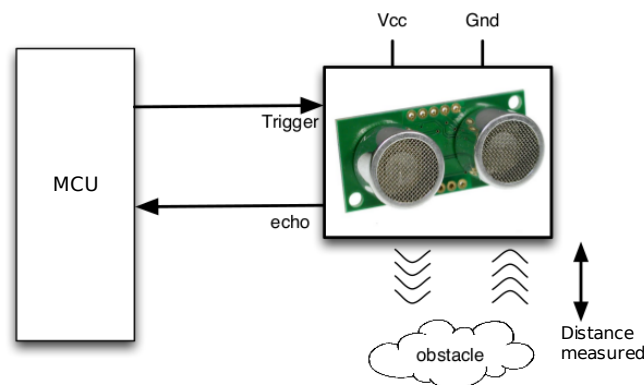


Figure 1: Connection between the ultrasonic sensor and the MCU.

The sensor works as follow (figure 2):

- a pulse to the *Trigger* signal is a request to perform a measure.

- the integrated circuit generates a sequence of 40KHz ultrasonic burst.

- an analog circuit detects the echo from an obstacle;

- The distance to the obstacle is deduced from the time elapsed between the trigger ultrasonic burst and the incoming echo, knowing the sound speed. A pulse is transmitted back on the *echo* signal, on which the pulse width depends on the distance to the obstacle: 58μs/cm.
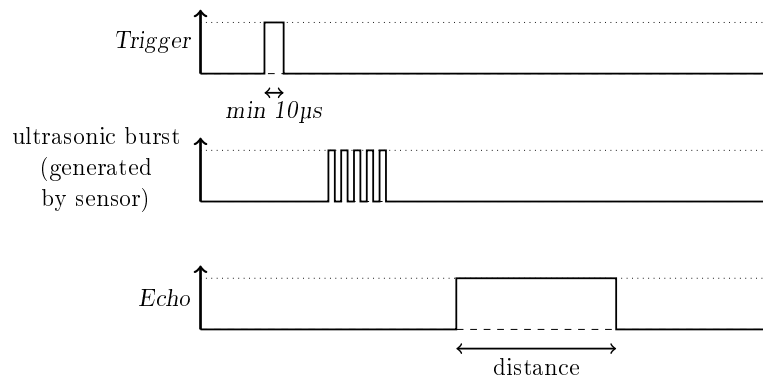
Figure 2: Time sequence of the ultrasonic sensor. The echo pulse width depends on the distance to the obstacle (58µs/cm)

In our configuration, signals *Trigger* and *Echo* share the same pin (`PA10`), we will have to change its configuration dynamically (output/input).

# 2   *Trigger* signal

The *Trigger* signal is just a pulse that should not be too fast, so that the integrated electronic in the sensor can detect it. The signal should be in high state for at least 10µs. We will insert a simple waiting loop to respect this constraint:

```
//set    PA10
for(volatile int i=0;i<20;i++);
//reset PA10
```

We will send *Trigger* request at 10Hz, using an interrupt and timer `TIM6`. To be sure that the *Trigger* signal is detected by the sensor, we just wait a little:

For an easy debug, we will toggle the green led `PB0` in the interrupt handler.

**Question 1** *Write the trigger part of the application:*

- *configuration of the sensor pin `PA10` as an output (do not forget the `GPIOA` clock!).*

- *interrupt @10Hz using `TIM6`.*

- *generation of the Trigger signal.*

Permanent configuration (for instance `GPIOx` clocks) should be done once in the `setup()`.

**NOTE**: pin `PA10` shares both *Trigger* (output) and *Echo* (input) signals. As a consequence, *the pin should be in the output mode as little time as possible.*

# 3 *Echo* signal

The distance to the obstacle is given by the duration of the `Echo` pulse. We use the `EXTI` peripheral to detect rising/falling edges, and a timer (`TIM7`) as a chronometer to get the pulse width. The measure is stored in a global variable (figure 3).
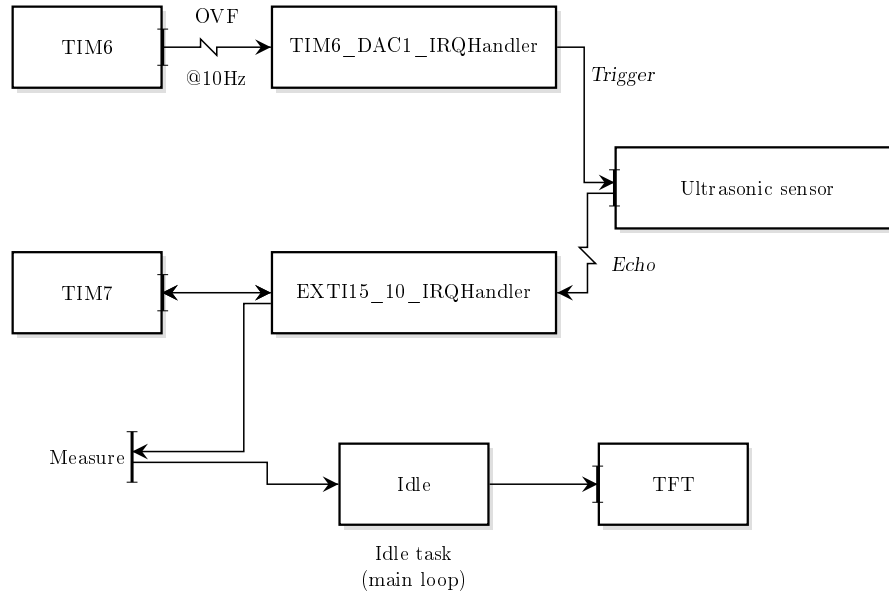


The principle uses `TIM7` as a chronometer (resolution 1µs) and an interrupt in *Echo*:

- if there is a rising edge on *Echo*, `TIM7` count value is reset

- if there is a falling edge on *Echo*, the value of `TIM7` is stored to the measure value.

To determine if an interrupt is a consequence of a rising or falling edge, you just have to read the logical level of the pin.

**Important note**: The *Trigger* signal should be updated, because it will generate an interrupt (modifying the output state generates an interrupt). So the `EXTI` interrupt should be deconfigured during the trigger generation, with `IMR` (Interrupt Mask Register):

```
EXTI ->IMR   &= ~EXTI_IMR_MR10 ;
//generate trigger signal ..
EXTI ->IMR   |= EXTI_IMR_MR10 ;
```

**Question 2** *Program the whole application, that refresh the value of `measure` periodically. check the correct value using the debugger.*

**Question 3** *Add a continuous display of the sensor value on the TFT (in the main loop), in mm. There is a quick documentation with an exemple in the `tft.md` file.*

# 4 Extension

The *echo* signal should occurs less that 50ms after the trigger.

**Question 4** *Add a* timeout *function that informs the user that the sensor is not available if there is no response after 50ms (using an interrupt of course - TIM7 for instance).*