LAB: Input Capture - Ultrasonic

Date: 2023-11-06

Author/Partner: NohYunKi

Demo Video:

Introduction

In this lab, Ultrasonic-Capture Sensor is used to calculate distance from obstacle. To operate this sensor, PWM timer will be used to send signals and another timer will be used to receive signals. By using time intervals calculated from these two timers, distance value can be obtained.

Requirement

Hardware

- MCU
  - NUCLEO-F411RE
- Actuator/Sensor/Others:
  - HC-SR04
  - breadboard

Software

- Keil uVision, CMSIS, EC_HAL library

Problem 1: Ultrasonic Distance Sensor (HC-SR04)

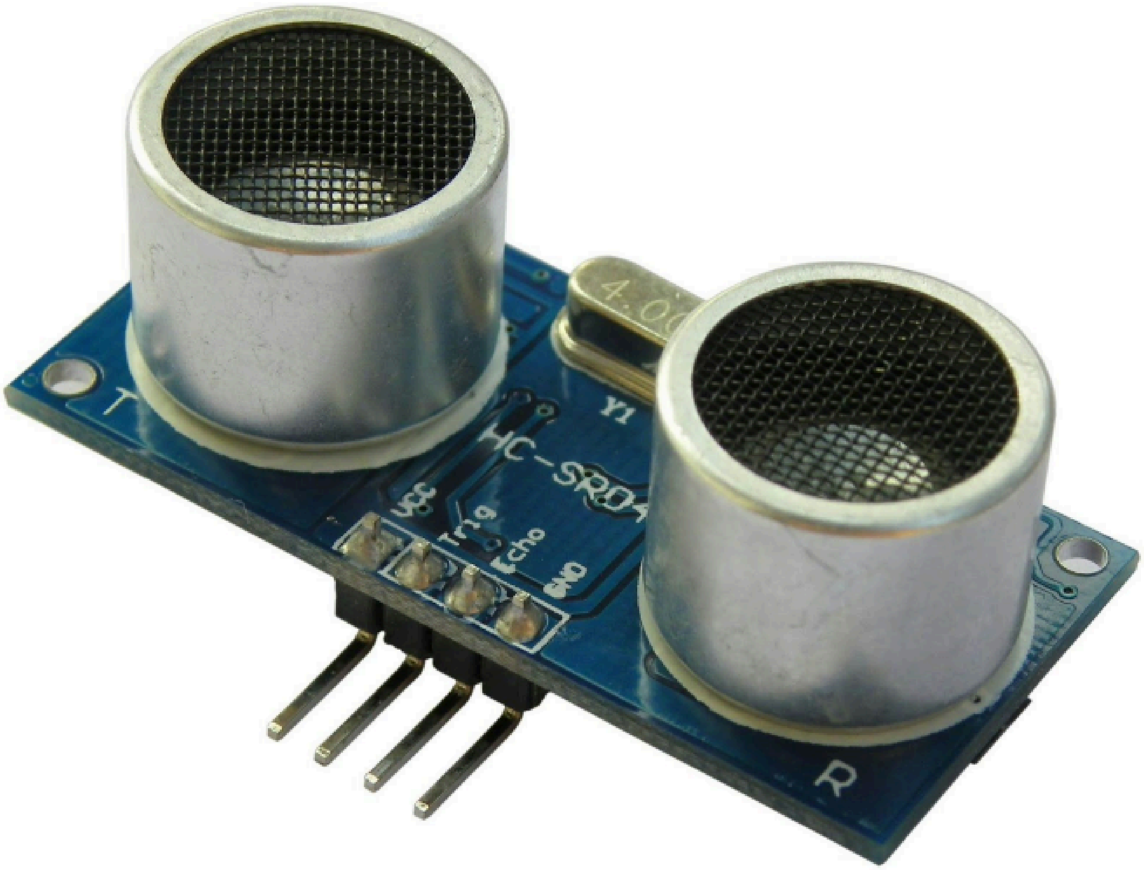


Figure 1. Ultra-Sonic Sensor(HC-SR04)

- Input Voltage: 5V
- Current Draw: 20mA (Max)
- Digital Output: 5V
- Digital Output: 0V (Low)
- Sensing Angle: 30° Cone
- Angle of Effect: 15° Cone
- Ultrasonic Frequency: 40kHz
- Range: 2cm - 400cm

Measurement of Distance

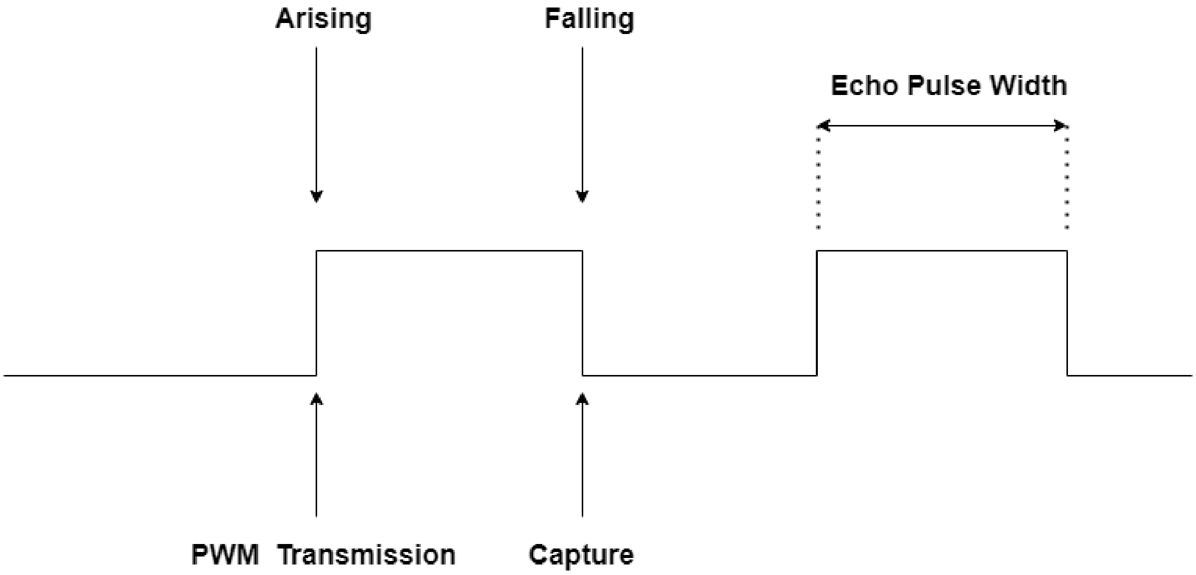


Figure 2. Echo Pulse Diagram

Ultrasonic Sensor generates rising signal when PWM signal is transmitted. After this, Ultrasonic Sensor will generate falling signal when this sensor receives PWM signal which are reflected from obstacle.

Using this Echo Pulse Width, we can obtain distance from obstacle.

$$Distance = PulseWidth(us)/58$$

Configuration

System Clock	PWM	Input Capture
PLL (84MHz)	PA6 (TIM3_CH1)	PB6 (TIM4_CH1)
	AF, Push-Pull, No Pull-up Pull-down, Fast	AF, No Pull-up Pull-down
	PWM period: 50msec pulse width: 10usec	Counter Clock : 0.1MHz (10us) TI4 -> IC1 (rising edge) TI4 -> IC2 (falling edge)

Circuit Diagram

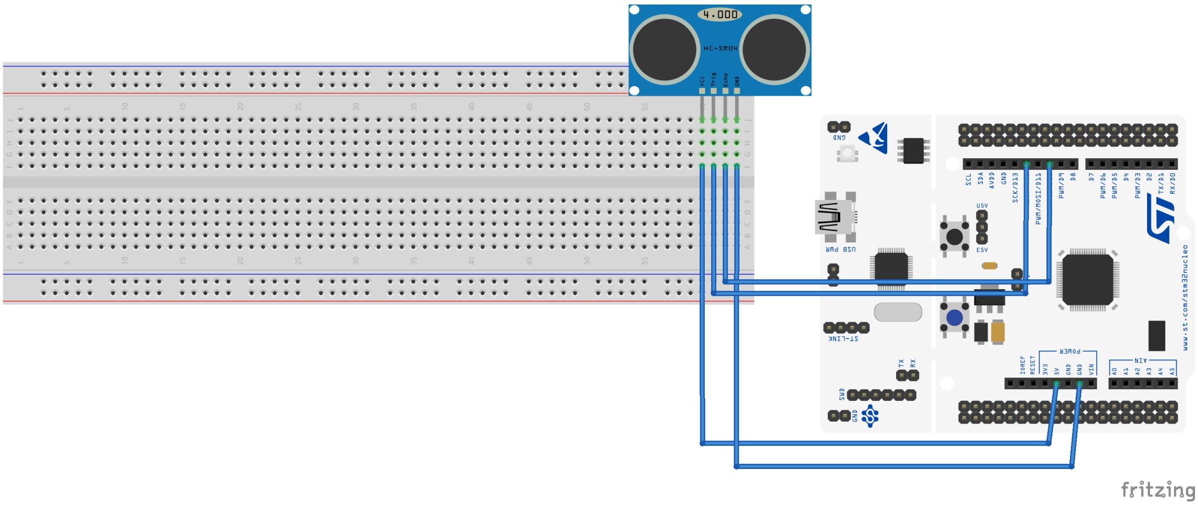
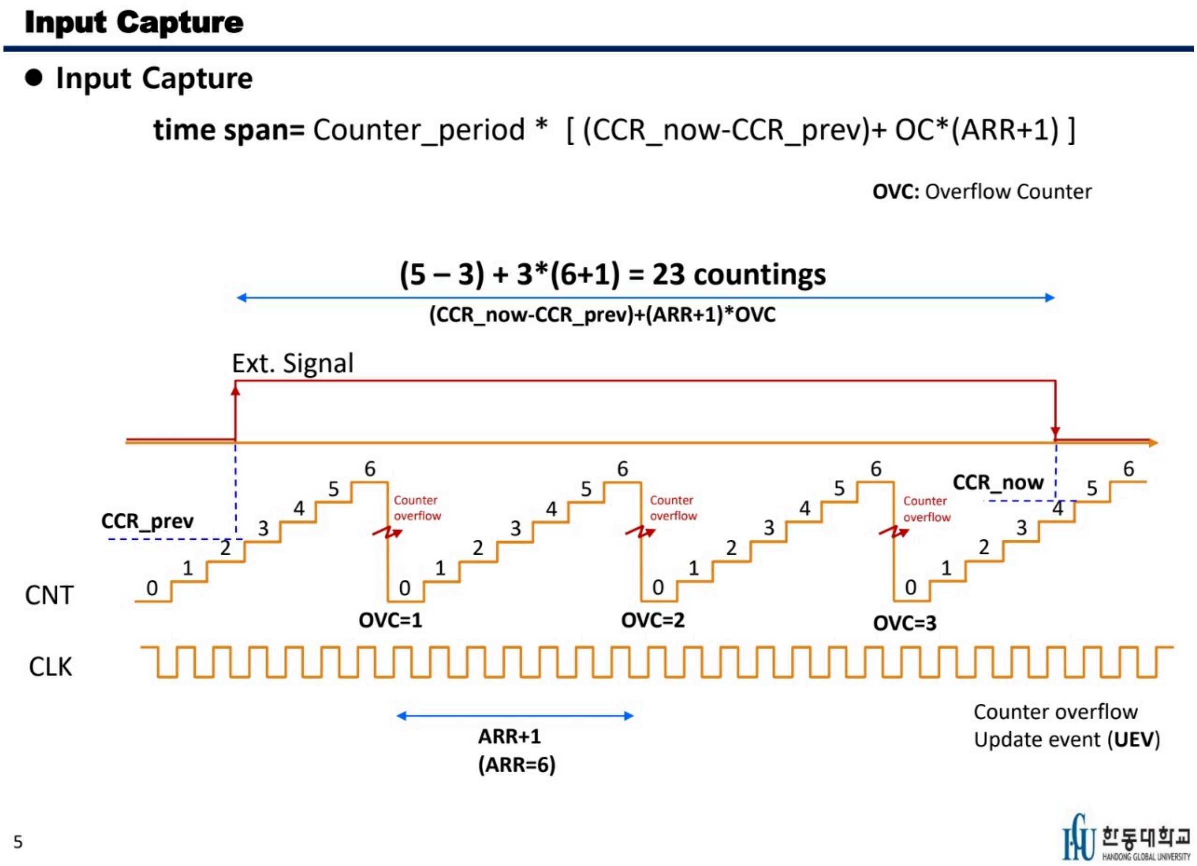


Figure 3. Ultra-Sonic Sensor Circuit Diagram

Discussion

1. There can be an over-capture case, when a new capture interrupt occurs before reading the CCR value. When does it occur and how can you calculate the time span accurately between two captures?



Input Capture

- **Input Capture**

time span= Counter_period * [(CCR_now-CCR_prev)+ OC*(ARR+1)]

OVC: Overflow Counter

Figure 4. Calculating Time Span Value

Several times of overflows will happen when the interval between first checking of CCR value(Arising moment) and second checking of CCR value(Falling moment) is long over one cycle.

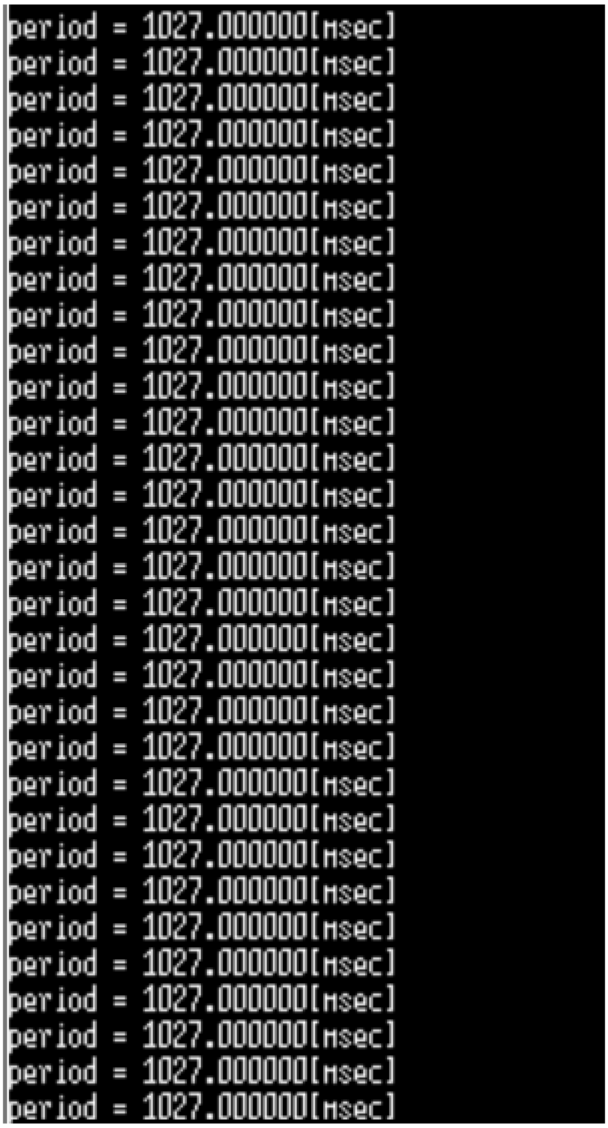
In this reason, the total number of counting can be obtained from this formula.

$$TotalCountingNumber = (CCR2 - CCR1) + (ARR + 1) * OVF$$

After this, time span can be obtained by using Total Counting Number.

$$TimeSpan = TotalCountingNumber * CountingPeriod$$

2. In the tutorial, what is the accuracy when measuring the period of 1Hz square wave? Show your result.



```
period = 1027.000000[msec]
period = 1027.000000[msec]
period = 1027.000000[msec]
period = 1027.000000[msec]
period = 1027.000000[msec]
period = 1027.000000[msec]
period = 1027.000000[msec]
period = 1027.000000[msec]
period = 1027.000000[msec]
period = 1027.000000[msec]
period = 1027.000000[msec]
period = 1027.000000[msec]
period = 1027.000000[msec]
period = 1027.000000[msec]
period = 1027.000000[msec]
period = 1027.000000[msec]
period = 1027.000000[msec]
period = 1027.000000[msec]
period = 1027.000000[msec]
period = 1027.000000[msec]
period = 1027.000000[msec]
period = 1027.000000[msec]
period = 1027.000000[msec]
period = 1027.000000[msec]
period = 1027.000000[msec]
period = 1027.000000[msec]
period = 1027.000000[msec]
period = 1027.000000[msec]
period = 1027.000000[msec]
period = 1027.000000[msec]
period = 1027.000000[msec]
```

Figure 5. Measuring 1Hz signal

Although we input 1.0Hz(1000.0ms) signal into PA0, period result which is calculated from CCR vlaue is slightly diffrent.[1027.0ms]. This difference will be occurred because counting CCR value is not completely match with actual counting moment.

Code

- **main**

```
/**
******************************************************************************
* @Content: Embeded Controller_Timer_InputCapture_Ultrasonic LAB
* @Date: 23/11/07
* @Name: NohYunKi
*
******************************************************************************
*/
```

```
#include "stm32f411xe.h"
#include "math.h"
#include "ecGPIO.h"
#include "ecRCC.h"
#include "ecTIM.h"
#include "ecPWM.h"
#include "ecUART.h"
#include "ecSysTIck.h"


uint32_t ovf_cnt = 0;
float distance = 0;
float timeInterval = 0;
float time1 = 0;
float time2 = 0;

#define TRIG PA_6 // PWM signal
#define ECHO PB_6 // ECHO signal

void setup(void);
void TIM4_IRQHandler(void);

int main(void){

   setup();

   while(1){
         if(distance > 30 || 0 > distance) printf("Bouncing Value \r\n"); // to recognize bouncing value
         else if(30>distance>0) printf("%f cm\r\n", distance);
      delay_ms(500);
   }
}

void TIM4_IRQHandler(void){
   if(is_UIF(TIM4)){                    // Update interrupt
      ovf_cnt++;                                       // overflow count
      clear_UIF(TIM4);                           // clear update interrupt flag
   }
   if(is_CCIF(TIM4, 1)){                         // TIM4_Ch1 (IC1) Capture Flag. Rising Edge Detect
          time1 = ICAP_capture(TIM4,IC_1);            // Capture TimeStart
      clear_CCIF(TIM4, 1);                // clear capture/compare interrupt flag
   }
   else if(is_CCIF(TIM4, 2)){                   // TIM4_Ch2 (IC2) Capture Flag. Falling Edge Detect
          time2 = ICAP_capture(TIM4,IC_2);            // Capture TimeEnd
      timeInterval = 10*((time2 - time1)+(TIM3->ARR+1)*ovf_cnt);
// (10us * counter pulse -> [msec] unit) Total time of echo pulse
      distance = (float) timeInterval/58.0;    // [mm] -> [cm]
      ovf_cnt = 0;                        // overflow reset
      clear_CCIF(TIM4,2);                          // clear capture/compare interrupt flag
   }
}
```

```
void setup(){

   RCC_PLL_init();
   SysTick_init();
   UART2_init();

// PWM configuration ------------------------------------------------------------
---------
   PWM_init(TRIG);                         // PA_6: Ultrasonic trig pulse
   GPIO_setting(GPIOA,6,AF,EC_PUSH_PULL,EC_NONE,EC_FAST);
   PWM_period_us(TRIG, 50000);    // PWM of 50ms period. Use period_us()
   PWM_pulsewidth_us(TRIG, 10);   // PWM pulse width of 10us


// Input Capture configuration --------------------------------------------------
---------------------
   ICAP_init(PB_6);                      // PB_6 as input caputre
   GPIO_pupd(GPIOB,6,EC_NONE);

   ICAP_counter_us(ECHO, 10);       // ICAP counter step time as 10us
   ICAP_setup(ECHO, 1, IC_RISE);  // TIM4_CH1 as IC1 , rising edge detect
   ICAP_setup(ECHO, 2, IC_FALL);  // TIM4_CH2 as IC2 , falling edge detect

}
```

In this code, I used Timer 4 of PB_6(1 channel & 2 channel) to detect rising & falling signal and Timer 3 of PA_6 to generate PWM signal. To prepare Timer's environment, I set up timer's setting in the 'void setup()' function. After this process, three values(Rising CCR value, Falling CCR vlaue, Overflow value) are obtained in the 'void TIM4_IRQHandler(void)' function. Finally these three values are used to calculate distance value and distance value is printed in the 'main' function.

Results

Link: https://youtube.com/shorts/k83tYI9u49s?feature=share

Reference

1. LAB: Input Capture - Ultrasonic - EC (gitbook.io)