LAB: Input Capture - Ultrasonic

Date: 2022-11-08

Author/Partner: 21800447 Jeahyun Oh / 21800222 Huynwoo Nam

Github: https://github.com/Ohjeahyun1/EC-jeahyun-447.git

Demo Video: https://youtu.be/_8FQ-9RFxlI

Introduction

In this lab, you are required to create a simple program that uses input capture mode to measure the distance using an ultrasonic distance sensor. The sensor also needs trigger pulses that can be generated by using the timer output.

You must submit

- LAB Report (*.md & *.pdf)
- Zip source files(main*.c, ecRCC.h, ecGPIO.h, ecSysTick.c etc...).
  - Only the source files. Do not submit project files

Requirement

Hardware

- MCU
  - NUCLEO-F411RE
- Sensor:
  - HC-SR04

Software

- Keil uVision, CMSIS, EC_HAL library

Problem 1: Create HAL library

Create HAL library

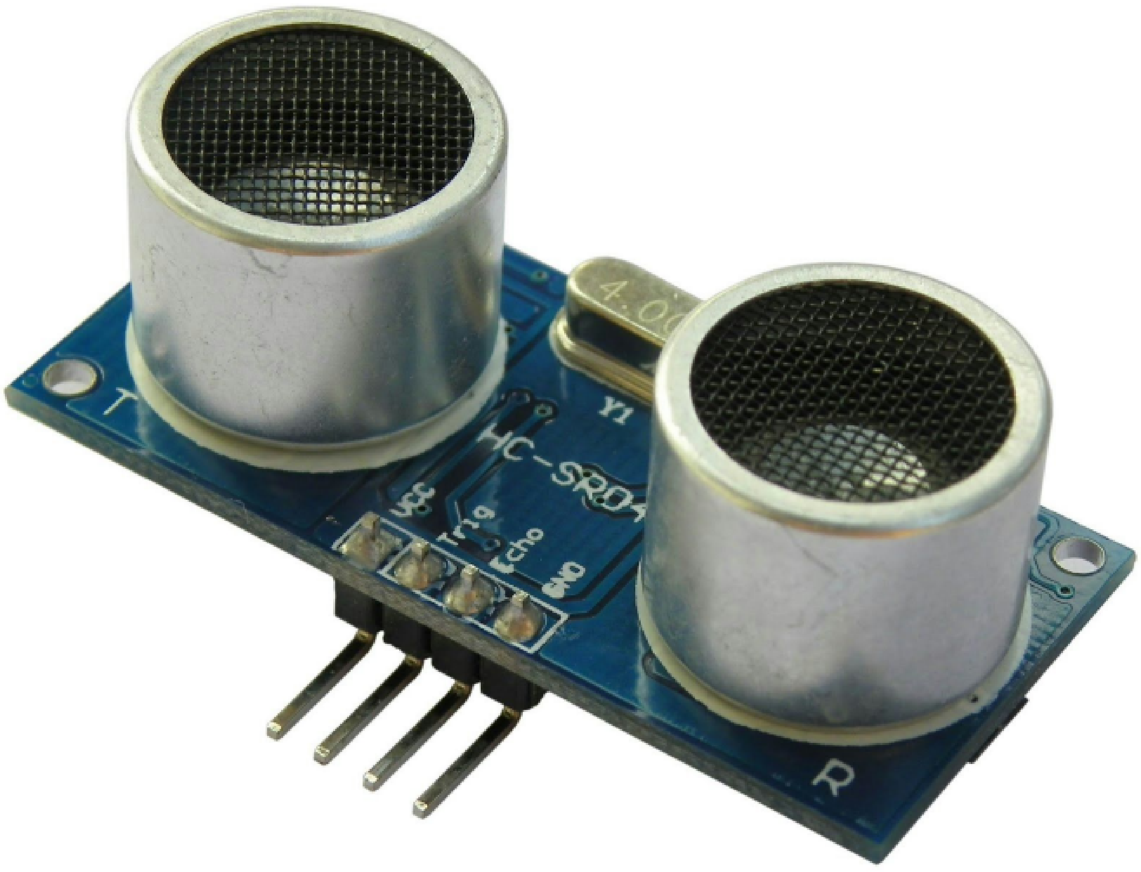
Declare and Define the following functions in your library. You must update your header files located in the directory `EC \lib\`.

ecTIM.h

```
// IC structure

typedef struct{

GPIO_TypeDef *port;

int pin;
TIM_TypeDef *timer;

int ch;         //int Timer Channel

int ICnum;  //int IC number

} IC_t;



// ICAP setup

void ICAP_init(IC_t *ICx, GPIO_TypeDef *port, int pin);     // Initialize input
capture mode (default setting)

void ICAP_setup(IC_t *ICx, int IC_number, int edge_type);   // Setup ICn and
Edge type

void ICAP_counter_us(IC_t *ICx, int usec);

// ICAP counter step time as us

// ICAP flag

uint32_t is_CCIF(TIM_TypeDef *TIMx, uint32_t ccNum);        // flag check Timer
ch

void clear_CCIF(TIM_TypeDef *TIMx, uint32_t ccNum);         // flag clear Timer
ch


void ICAP_pinmap(IC_t *timer_pin);                               //DO NOT
MODIFY THIS
```

Problem 2: Ultrasonic Distance Sensor (HC-SR04)

The HC-SR04 ultrasonic distance sensor. This economical sensor provides 2cm to 400cm of non-contact measurement functionality with a ranging accuracy that can reach up to 3mm. Each HC-SR04 module includes an ultrasonic transmitter, a receiver and a control circuit.



The HC-SR04 Ultrasonic Range Sensor Features:

- 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

Procedure

1. Create a new project under the directory `\repos\EC\LAB\LAB_Timer_InputCaputre_Ultrasonic`

- The project name is “**LAB_Timer_InputCaputre_Ultrasonic”.**
- Create a new source file named as “**LAB_Timer_InputCaputre_Ultrasonic.c”**

> You MUST write your name on the source file inside the comment section.

2. Include your updated library in `\repos\EC\lib\` to your project.

- **ecGPIO.h, ecGPIO.c**
- **ecRCC.h, ecRCC.c**
- **ecTIM.h, ecTIM.c**
- **ecPWM.h, ecPWM.c**
- **ecUART.h, ecUART.c**
- **ecSysTick.h, ecSysTick.c**

3. Connect the HC-SR04 ultrasonic distance sensor to MCU pins(PA6 - trigger, PB10 - echo), VCC and GND

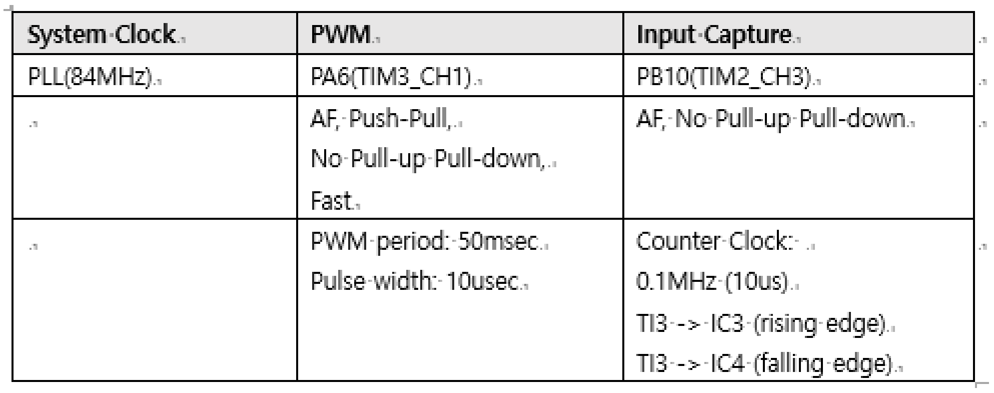
Measurement of Distance

The program needs to

- Generate a trigger pulse as PWM to the sensor.
- Receive echo pulses from the ultrasonic sensor
- Measure the distance by calculating pulse-width of the echo pulse.
- Display measured distance in [cm] on serial monitor of Tera-Term for

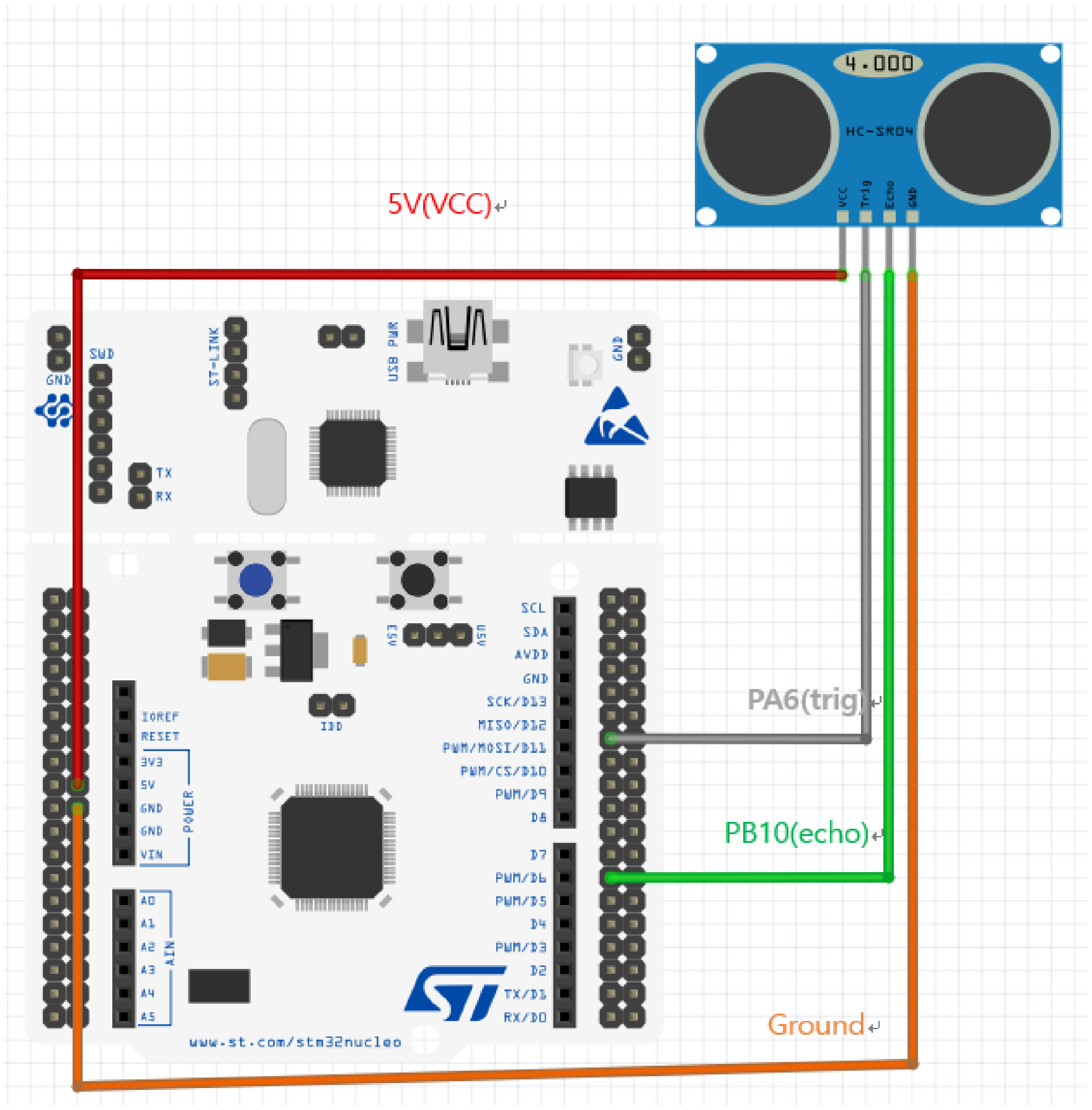
  (a) 10mm (b) 50mm (c) 100mm

Configuration



System Clock	PWM	Input Capture
PLL(84MHz)	PA6(TIM3_CH1)	PB10(TIM2_CH3)
	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) TI3 -> IC3 (rising edge) TI3 -> IC4 (falling edge)

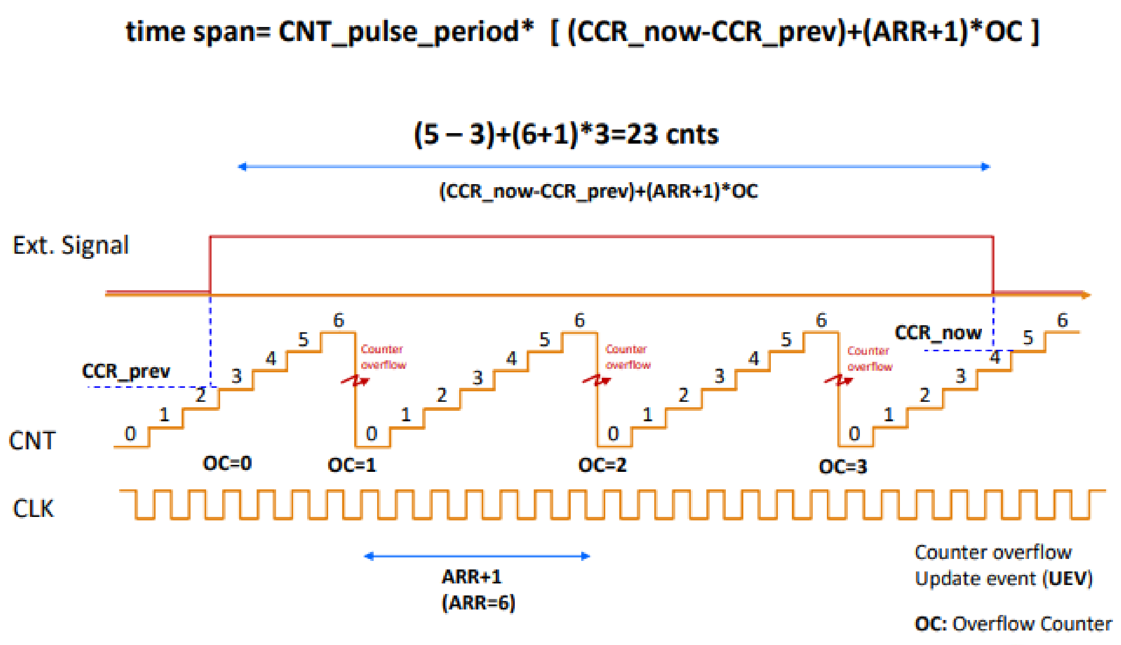
Circuit Diagram



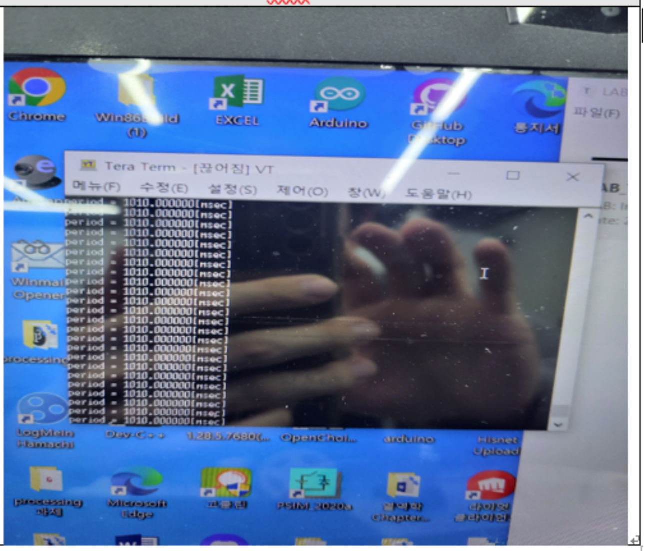
Discussion

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?

   Measure CCR_prev(rising) when the external input is first received.
   At this time, there is a period in the timer counter (CNT) that I set. However, when the external signal becomes longer than this period, overcapture occurs. Count the number of times these overcapture occurrences. Measure CCR_now when the external input goes back to zero (Falling).
   At this time, the time span between the two captures is shown in the figure below.



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

Function Generator	Tera Term
	

```
period = 1010.000000[msec]
period = 1010.000000[msec]
period = 1010.000000[msec]
period = 1010.000000[msec]
period = 1010.000000[msec]
period = 1010.000000[msec]
period = 1010.000000[msec]
period = 1010.000000[msec]
period = 1010.000000[msec]
period = 1010.000000[msec]
period = 1010.000000[msec]
period = 1010.000000[msec]
period = 1010.000000[msec]
period = 1010.000000[msec]
period = 1010.000000[msec]
period = 1010.000000[msec]
period = 1010.000000[msec]
period = 1010.000000[msec]
period = 1010.000000[msec]
period = 1010.000000[msec]
period = 1010.000000[msec]
period = 1010.000000[msec]
period = 1010.000000[msec]
period = 1010.000000[msec]
```

Calculate the percent relative error

P.R.E = (Measure-True)/True * 100[%]

= (1010-1000)/1000*100

= 1%

It shows an error accuracy of 1%.

Code

Your code goes here:

LAB_C:https://github.com/Ohjeahyun1/EC-jeahyun-447/blob/21ed5609ca760e99d6b9b5652ef85005704e5fe6/lab/LAB_Timer_InputCapture_Ultrasonic.c

include_C: https://github.com/Ohjeahyun1/EC-jeahyun-447/blob/23da2510360582129a02ccbb4b0df79eaf96ff53/include/ecTIM.c include_H:https://github.com/Ohjeahyun1/EC-jeahyun-447/blob/23da2510360582129a02ccbb4b0df79eaf96ff53/include/ecTIM.h

Explain your source code with necessary comments

Description with Code

- Main code

printf distance using Tera Term

change mm -> cm

```
//printf distance using Tera Term
//change mm -> cm
int main(void){

    setup();

    while(1){
        distance = (float) timeInterval * 340.0 / 2.0 / 10.0;           // [mm]
-> [cm]
        printf("%f [cm]\r\n", distance);                                // print
distance
        delay_ms(500);
    }
}
```

- TIMER interrupt

check the over count number and Capture rising, falling time

calculate TimeInterval

```
void TIM2_IRQHandler(void){
    if(is_UIF(TIM2)){                                                   //
Update interrupt
        ovf_cnt++;                                                      //
overflow count
        clear_UIF(TIM2);                                                // clear
update interrupt flag
    }
    if(is_CCIF(TIM2, 3)){                               // TIM2_Ch3 (IC3)
Capture Flag. Rising Edge Detect
        time1 = TIM2->CCR3;                                             //
Capture TimeStart
        clear_CCIF(TIM2, 3);                                        // clear
capture/compare interrupt flag
    }
    else if(is_CCIF(TIM2, 4)){           // TIM2_Ch3 (IC4) Capture Flag. Falling
Edge Detect
        time2 = TIM2->CCR4 ;                                             //
Capture TimeEnd
        if((time2-time1)<(TIM2->ARR+1)&(ovf_cnt==1)) ovf_cnt=0; // if (time2-
time1)< ARR+1 make over count 0
        timeInterval = ((time2-time1)+(TIM2->ARR+1)*ovf_cnt)/100;
        // Total time of echo pulse (10us * counter pulses -> [msec] unit)
        ovf_cnt = 0;                                                     //
overflow reset
```

```
        clear_CCIF(TIM2, 4);                             // clear
capture/compare interrupt flag
    }
}
```

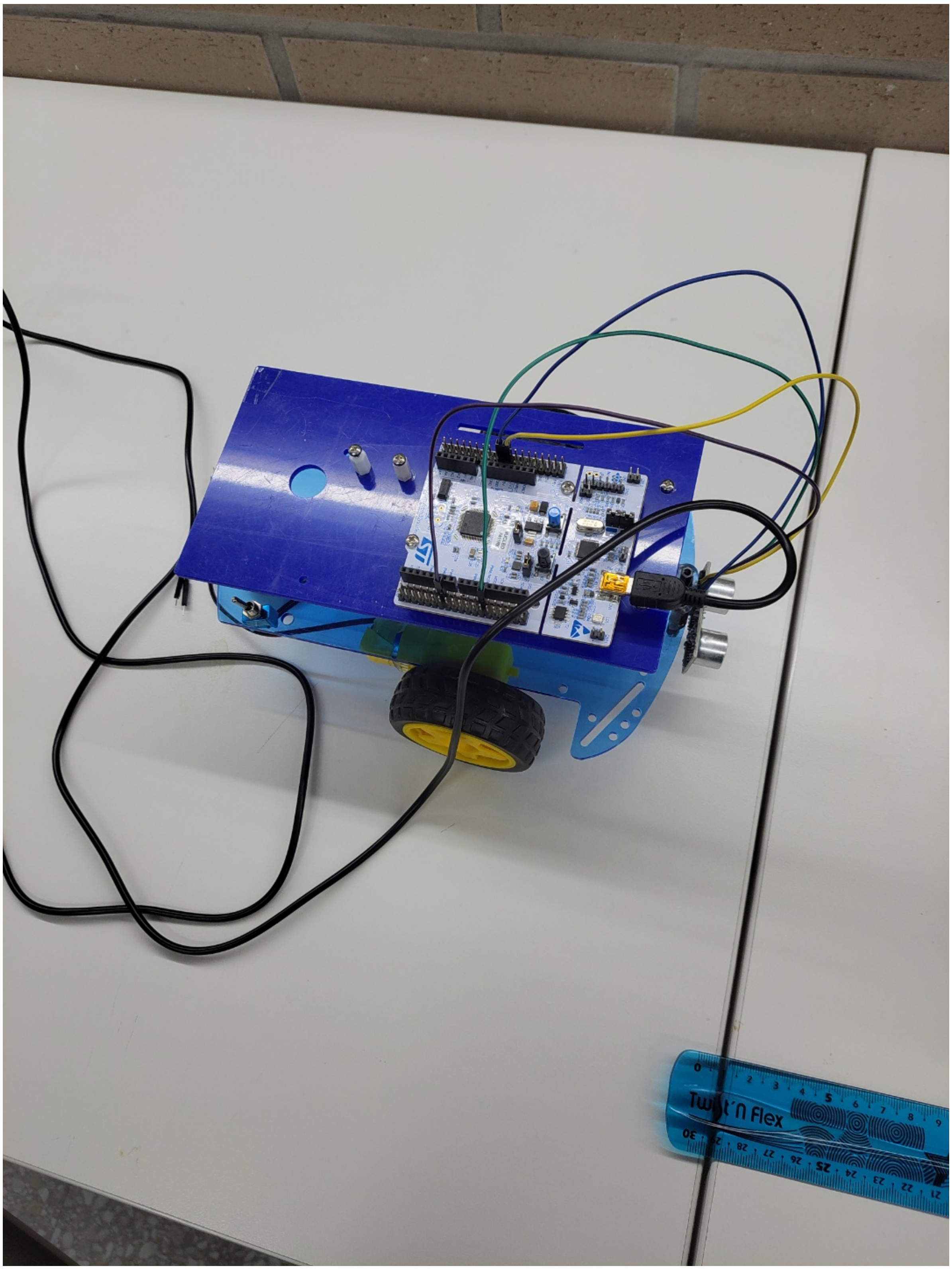
- setup

setting(PLL,PWM,Inputcapture)

```
void setup(){

    RCC_PLL_init();                                                     // 84Mhz
system clock
    SysTick_init();                                                     // using
sysTick
    UART2_init();                                                       // for
printf

// PWM configuration ------------------------------------------------------------
---------
    PWM_t trig;                                                         // PWM1
for trig
    PWM_init(&trig,GPIOA,6,UP,SFAST,PP,EC_NOPUPD,1);                    // PA_6:
Ultrasonic trig pulse
    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 --------------------------------------------------
---------------------
    IC_t echo;                                                           //
Input Capture for echo
    ICAP_init(&echo,GPIOB,10,EC_NOPUPD);                                 // PB10
as input caputre
    ICAP_counter_us(&echo, 10);                                         // ICAP
counter step time as 10us
    ICAP_setup(&echo, 3, IC_RISE);                                // TIM2_CH3 as
IC3 , rising edge detect
    ICAP_setup(&echo,4,IC_FALL);                                  // TIM2_CH3 as
IC4 , falling edge detect

}
```

Results

Experiment images and results



It sends a signal from the trigger(PWM), reads the signal that it returns after hitting an object, and outputs the distance using the Tera Term.

Since the range is 2cm - 400cm, it was confirmed that the correct value could not be read at 10mm. However, it was confirmed that the desired value came out at 50mm and 100mm.

Add demo video link: https://youtu.be/_8FQ-9RFxlI

Reference

Complete list of all references used (github, blog, paper, etc)

Troubleshooting

(Option) You can write Troubleshooting section

It was difficult to make the ultrasonic distance sensor accurately perpendicular. As a result, it was not easy to print out accurate values from low distances.

Fixed an issue where the streets are sometimes very large in Tera Term.

The reason why the distance is so big was the over count problem. In fact, even though the pwm was less than one cycle of Timer, the count went up in coding. This is a coding problem. Overcount increases according to TIM2 interrupt. However, if PWM enters at this time, it is calculated by recognizing that the over count is 1 even though pwm is not actually as large as one cycle. Therefore, to fix this problem, if the value minus time_pre and time_current is less than ARR+1 of the actual TIM2, I solved the problem by initializing overcount.