LAB: ADC - IR sensor

Date: 2022-11-30

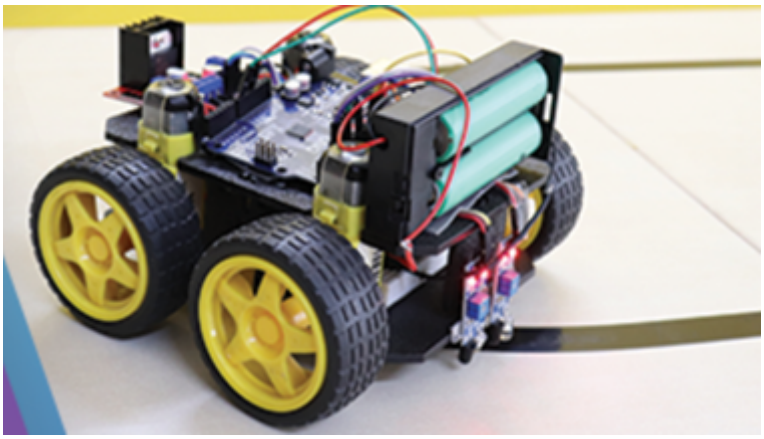
Author/Partner: 21800447 Jeahyun Oh / 21800222 Huynwoo Nam

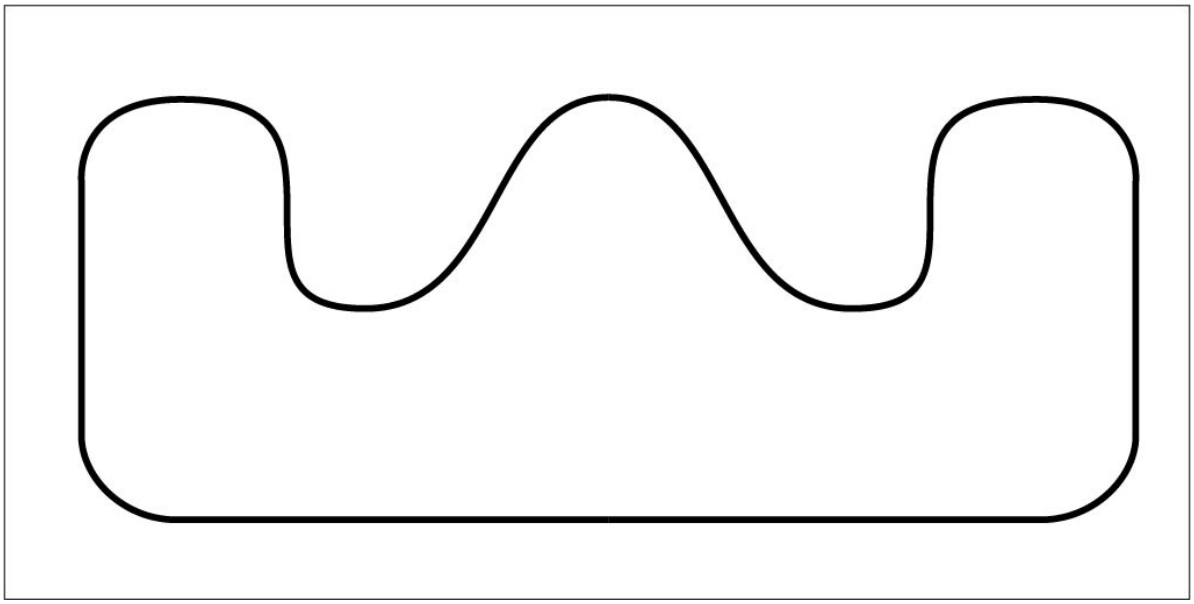
Github: https://github.com/Ohjeahyun1/EC-jeahyun-447.git

Demo Video: https://youtu.be/cscwfmkLoLQ

Introduction

In this lab, you are required to create a simple application that uses ADCs to implement the line tracing mission for an RC car. The analog measurement of reflection values from two IR reflective sensors are used . The ADCs are triggered by a timer of given sampling rate.





Track

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
- Actuator:
  - DC motor x2
- Sensor
  - IR Reflective Sensor (TCRT 5000) x2
  - HC-SR04
- Others
  - DC motor driver(L9110s)
  - breadboard
  - RC car

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\`.

ecADC.h

```
// ADC setting

void ADC_init(GPIO_TypeDef *port, int pin, int trigmode);   // trigmode : SW , TRGO

void ADC_continue(int contmode);                            // contmode : CONT, SINGLE / Operate both ADC,JADC

void ADC_TRGO(TIM_TypeDef* TIMx, int msec, int edge);       // set the TIMx TRGO (TIM2, TIM3 only)

void ADC_sequence(int length, int *seq);                    // configure the order of ADC channels



void ADC_start(void);

uint32_t ADC_read(void);



uint32_t is_ADC_EOC(void);

uint32_t is_ADC_OVR(void);
```

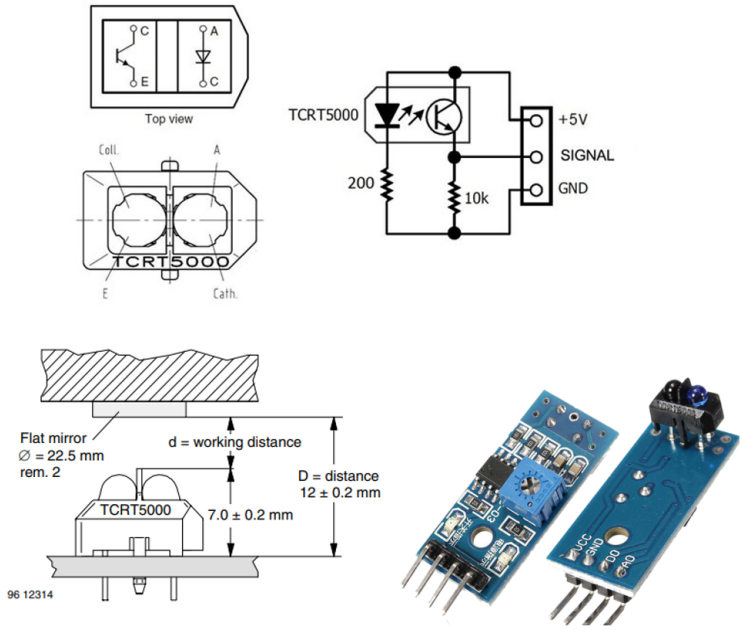
```
void clear_ADC_OVR(void);


uint32_t ADC_pinmap(GPIO_TypeDef *port, int pin);
```

Problem 2: IR Reflective Sensor (TCRT 5000)

IR Reflective Sensor(TCRT 5000): Spec Sheet

The TCRT5000 and TCRT5000L are reflective sensors which include an infrared emitter and phototransistor in a leaded package which blocks visible light.



image

The HC-SR04 Ultrasonic Range Sensor Features:

- Input Voltage : 5V
- Detector type: phototransistor
- Operating range within > 20 % relative collector current: 0.2 mm to 15 mm
- Emitter wavelength: 950 nm

APPLICATIONS

- Position sensor for shaft encoder
- Detection of reflective material such as paper, IBM cards, magnetic tapes etc.
- Limit switch for mechanical motions in VCR
- General purpose - wherever the space is limited

Procedure

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

- The project name is "**LAB_ADC_IR"**
- Create a new source file named as "**LAB_ADC_IR.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**
- **ecSysTick.h**, **ecSysTick.c**
- **ecUART.h, ecUART.c**
- **ecADC.h, ecADC.c**

Configuration

TIMER	**ADC**	**GPIO**
TIM2	ADC1_CH8 (1st channel) ADC1_CH9 (2nd channel)	PB_0 PB_1
Up-Counter, Counter CLK 1kHz OC1M(Output Compare 1 Mode) : PWM mode 1 Master Mode Selection: (TRGO) OC1REF	ADC Clock Prescaler/8 12-bit resolution, right alignment Single Conversion mode Scan mode: Two channels in regular group External Trigger (Timer3 TRGO) @1kHz Trigger Detection on Rising Edge	Analog Mode No Pull-up Pull-down

Ultrasonic Distance Sensor(HC-SR04)

System Clock	**PWM**	**Input Capture**
PLL(84MHz)	PA6(TIM3_CH1)	PB7(TIM4_CH2)
	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) TI2 -> IC2 (rising edge) TI2 -> IC1 (falling edge)

DC motor

DC Motor	TIM	Configuration
PWM (Motor A)	PC8(TIM3_CH3)	PWM period(1ms)
PWM (Motor B)	PC9(TIM3_CH4)	PWM period(1ms)

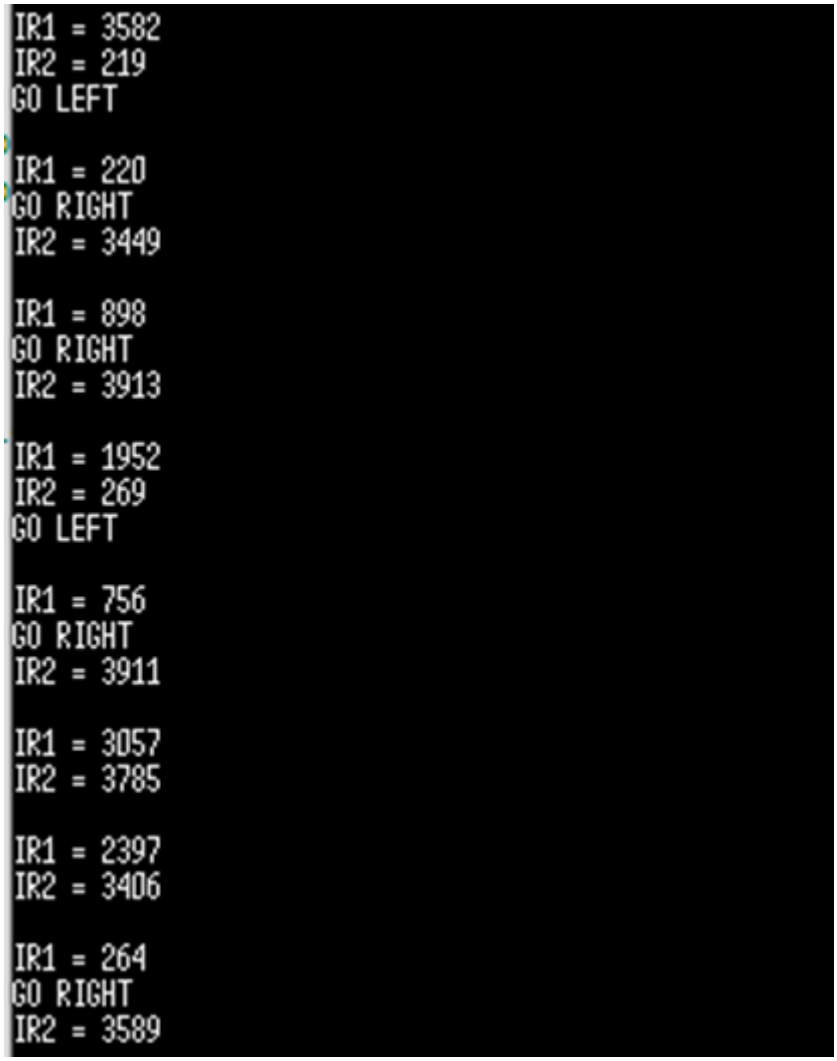
Line Tracing

- Create a logic to trace a dark line on white background surface for your RC car.
- Use 2 IR reflective sensors to detect if the black line is in between the sensors. It should display whether the system needs to move **Left or Right** to keep the line between sensors.
- Set the ADC sampling rate trigger to be 1KHz, to decrease burden to your CPU.
- Determine the threshold value to differentiate dark and white surface of the object.
- Display (1) and (2) on serial monitor of Tera-Term. Print the values every second.

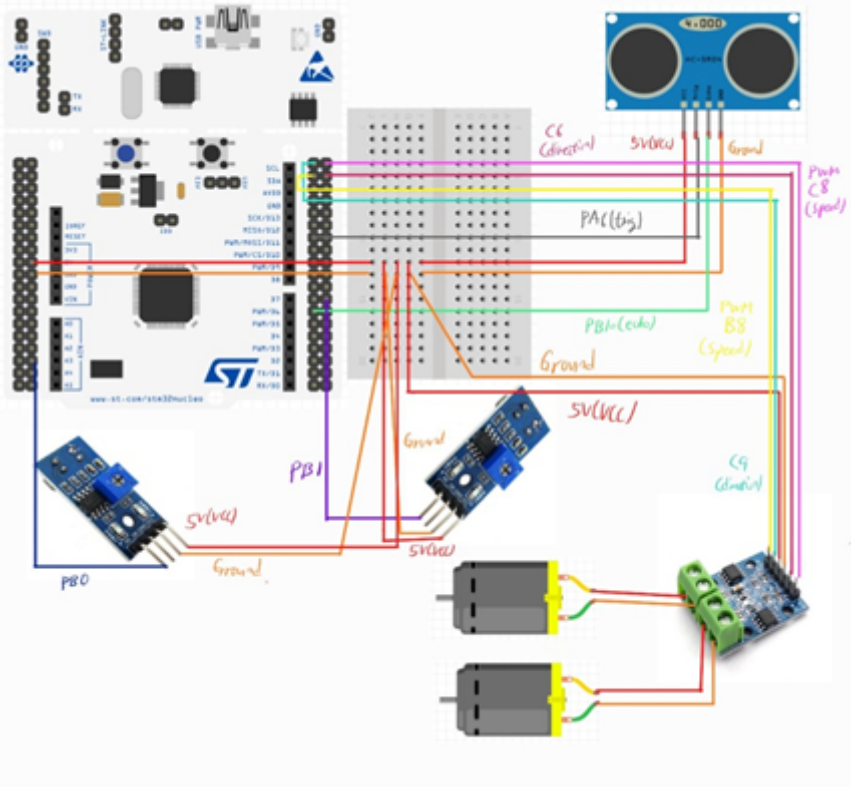
  (1) reflection value of IR1 and IR2

  (2) print 'GO LEFT' or 'GO 'RIGHT'

Display Example



Circuit Diagram



Discussion

1. How would you change the code if you need to use 3 Analog sensors?

Assuming that an IR sensor is added to the center, if the center sensor detects black, it goes straight. If black is not detected in the middle sensor, recognize the value of the sensors on both sides and rotate to the right or left accordingly.

1. Which registers should be modified if you need to use Injection Groups instead of regular groups for 2 analog sensors?

   Regular groups share a data register, but injected groups should be applied because each of the four channels has a data register.

Code

Your code goes here:

LAB_ADC_IR.c:https://github.com/Ohjeahyun1/EC-jeahyun-447/blob/f684fc1a98f74d2e1fda50583281b15f33c695ab/lab/LAB_ADC_IR.c

include_ADC.c: https://github.com/Ohjeahyun1/EC-jeahyun-447/blob/29dfa893d89f0d78de230b3ce56f7af78d1d4d3b/include/ecADC.c

include_ADC.h:https://github.com/Ohjeahyun1/EC-jeahyun-447/blob/29dfa893d89f0d78de230b3ce56f7af78d1d4d3b/include/ecADC.h

Description with Code

- Main code

Use IR sensors to detect any deviation from the line and follow the line

If Ultrasonic sensor recognize an obstacle, stop it, and when it disappears, follow the line again

```
int main(void) {
    // Initialiization --------------------------------------------------------
    setup();
    printf("Hello Nucleo\r\n");

    // Inifinite Loop ---------------------------------------------------------
    while (1){
            if (IR1 > 1000) DIR_flag = RIGHT;                    //when IR1
sensor detected the line RIGHT FLAG
        else if (IR2 > 1000) DIR_flag = LEFT;                   //when IR2 sensor
detected the line LEFT FLAG
        else if (IR1 < 999 && IR2 <999) DIR_flag = FORWARD;     //FORWARD FLAG

            if (DIR_flag == FORWARD ){                           //FORWARD input
                Right =0.56;                                       //0.62
                Left =0.50;                                        //0.6

            }else if(DIR_flag == LEFT ){                         //LEFT input
                        Right = 0.4;                                    // 0.4
                        Left = 0.8;                                     // 1.0
            }else if(DIR_flag == RIGHT ){                        //RIGHT input
                    Right = 0.8;                                    // 1.0
                    Left = 0.38;                                    // 0.5
            }

    //  printf("IR1: %f \r\n", IR1);
    //  printf("IR2: %f \r\n", IR2);
    //      printf("dirflag : %d \r\n\n",DIR_flag);
        distance = (float) timeInterval * 340.0 / 2.0 / 10.0;  // [mm] -> [cm]
//      printf("distance: %f \r\n", distance);
    //      printf("t1: %f \r\n", time1);
    //      printf("t2: %f \r\n", time2);
//    printf("K: %f \r\n", k);
        if(distance  < 8.0) {                                     //if obstacle
detected
            k = 1;                                               // obstacle
flag

        }else if(distance > 10.0 && k == 1){                    //when obstacle
run out flag clear
            k = 0;
        }
        if(k == 1){                                             // obstacle flag
on
            Right = 1.0;                                         // stop the RC
motor
            Left  = 1.0;                                         // stop the RC
motor
        }
        // change the PWM with the IR sensor flag
    PWM_duty(&dcPwm[A], Right);
      PWM_duty(&dcPwm[B], Left);

    }


}
```

- TIMER interrupt

```
// Detect the distance using Ultrasonic distance sensor
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, 2)){
   // TIM4_Ch2 (IC2) Capture Flag. Rising Edge Detect
        time1 = TIM4->CCR2;
       // Capture TimeStart
        clear_CCIF(TIM4, 2);                                              //
clear capture/compare interrupt flag
    }
    else if(is_CCIF(TIM4, 1)){
   // TIM4_Ch1 (IC1) Capture Flag. Falling Edge Detect
        time2 = TIM4->CCR1;
         // Capture TimeEnd
        if((time2-time1)<(TIM4->ARR+1)&(ovf_cnt==1)) ovf_cnt=0;         // if
(time2-time1)< ARR+1 make over count 0
        timeInterval = ((time2-time1)+(TIM4->ARR+1)*ovf_cnt)/100;           //
Total time of echo pulse (10us * counter pulses -> [msec] unit)
        ovf_cnt = 0;                                                      //
overflow reset
        clear_CCIF(TIM4, 1);
       // clear capture/compare interrupt flag
    }
}
```

- ADC interrupt

read IR sensors

```
// ADC interrupt
// read IR sensors
void ADC_IRQHandler(void){
    if((is_ADC_OVR())){                                                 // ADC
over
        clear_ADC_OVR();                                                  //clear
adc sr
    }

    if(is_ADC_EOC()){                                                   //after
finishing sequence
            if (flag==0){
                IR1 = ADC_read();                                             //
read ADC IR1
            }
            else if (flag==1){
                IR2 = ADC_read();                                             //
read ADC IR2
            }
        flag =! flag;
```

```
    }
}
```

- setting

```
void setup(void)
{   RCC_PLL_init();
    // USART congfiguration
    USART_init(USART2, 9600);
    SysTick_init();                          //SysTick init

    // ADC setting
  ADC_init(GPIOB, 0, TRGO);               //ch8
    ADC_init(GPIOB, 1, TRGO);               //ch9

    // ADC channel sequence setting
    ADC_sequence(2, seqCHn);

    // ADON, SW Trigger enable
    ADC_start();
    // 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

    // DC motor init
    PWM_init(&dcPwm[A], dcPwmPin[A].port, dcPwmPin[A].pin,UP,SFAST,PP,EC_PU,1);
    PWM_init(&dcPwm[B], dcPwmPin[B].port, dcPwmPin[B].pin,UP,SFAST,PP,EC_PU,1);
    // DC motor period(1ms)
    PWM_period_ms(&dcPwm[A], 1);
    PWM_period_ms(&dcPwm[B], 1);
  // DC motor output mode, no pull up pulll down, push-pull,High speed
    for (int i = 0; i < 2; i++){
        GPIO_init(dcDirPin[i].port, dcDirPin[i].pin, OUTPUT);
        GPIO_pupdr(dcDirPin[i].port, dcDirPin[i].pin, EC_PD);
        GPIO_otype(dcDirPin[i].port, dcDirPin[i].pin, PP);
        GPIO_ospeed(dcDirPin[i].port, dcDirPin[i].pin, SHIGH);
    }
    // DC motor
    GPIO_write(dcDirPin[A].port, dcDirPin[A].pin, mode);
    GPIO_write(dcDirPin[B].port, dcDirPin[B].pin, mode);


    // Input Capture configuration ------------------------------------------------
-------------------------

    IC_t echo;
           // Input Capture for echo
    ICAP_init(&echo,GPIOB,7 ,EC_NOPUPD);                                    // PB7
as input caputre
```

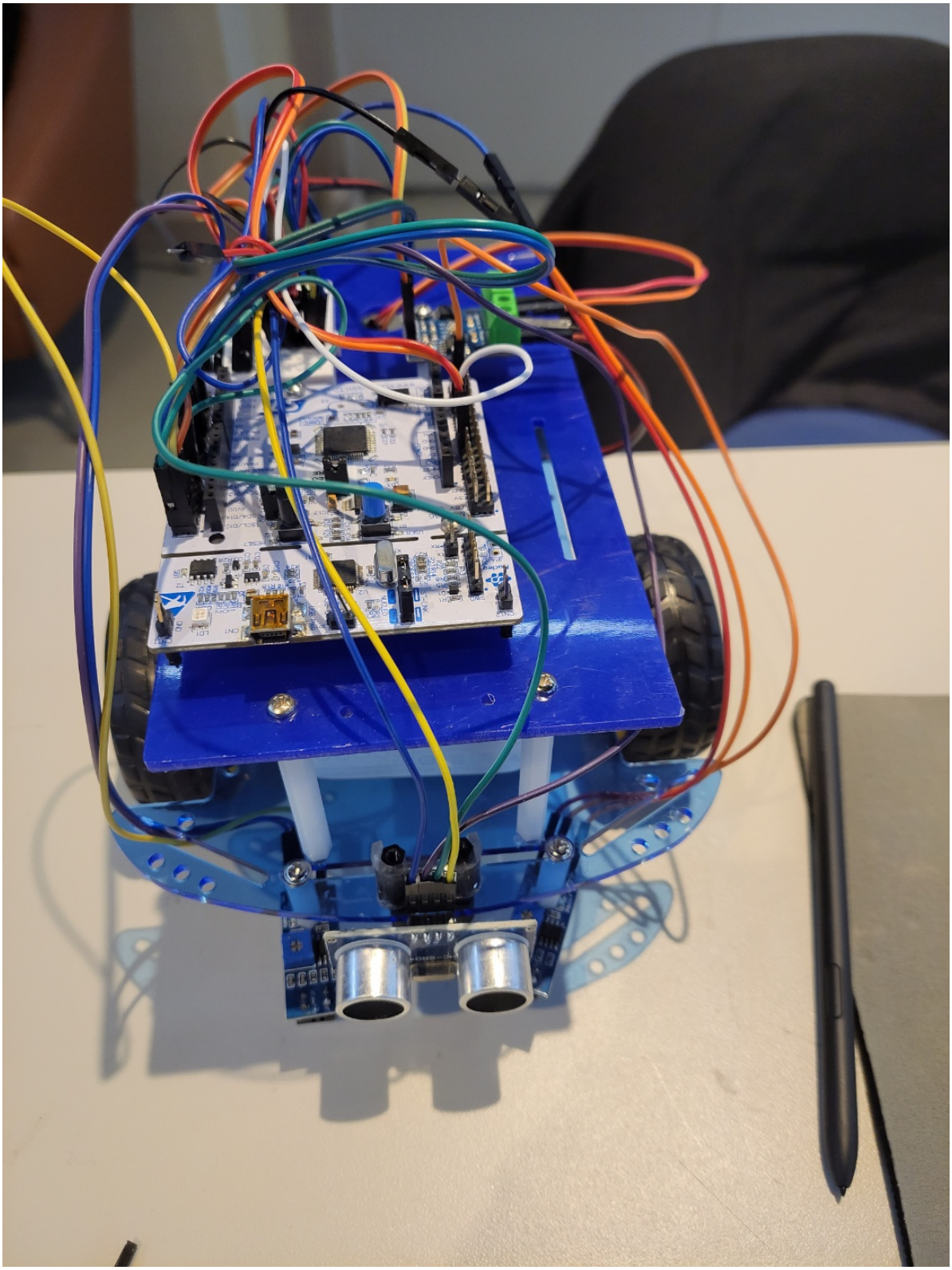
```
    ICAP_counter_us(&echo, 10);                                          // ICAP
counter step time as 10us
    ICAP_setup(&echo, 2, IC_RISE);                                       //
TIM4_CH2 as IC2 , rising edge detec
    ICAP_setup(&echo,1,IC_FALL);                                        //
TIM4_CH1 as IC1 , faling edge detec



}
```

Results

Experiment images and results



Use IR sensors to detect any deviation from the line and follow the line

If Ultrasonic sensor recognize an obstacle, stop it, and when it disappears, follow the line again

Add demo video link: https://youtu.be/cscwfmkLoLQ

Reference

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

Troubleshooting

(Option) You can write Troubleshooting section