Final project : Smart delivery system

Date: 2023-12-18

Author/Partner: 21900213 Hanmin Kim/21900253 Seongbin Mun

Github: https://github.com/ansterdaz/Embedded-Controller

Demo Video: https://youtu.be/y8p7ZOsekoc

I. Introduction

Overview

The importance of automation is increasing as demand for delivery and logistics warehouses increases. Therefore, 'Logistics classification according to weight' was selected as the topic. The goal is to move objects using an RC car and control the movement of the RC car through a pressure sensor at the destination.

Requirement

Hardware

- MCU
  - NUCLEO-F401RE x2

- Actuator/Sensor/Others:
  - Actuator list :

Actuator list	Qty
DC motor	2
RC servo motor (SG90)	2

  - Sensor list :

Sensor list	Qty
IR reflective sensor (TCRT 5000)	2
Ultrasonic distance sensor (HC-SR04)	2
Pressure sensor (FSR406)	2

- Others :

Others	Qty
Motor driver (L9100)	1
Bluetooth (HC06)	1
Registor	3
Breadboard	2

Software

- Keil uVision, CMSIS, EC_HAL library

II. Problem

Problem Description

Autonomous transportation system (MCU1)

- When loading the product into the vehicle and entering the A button on PC1 that controls the vehicle, the vehicle enters automatic mode.
- After that, the car performs line tracing and drives a predetermined path.
- If the object in front of the car is detected, the car will stop.
- After detecting the object, the vehicle returns to its original position.

Vehicle control system at arrival point (MCU2)

- Before the load is loaded, the vehicle is in a state of brightening the pressure sensor, and the sensor is detected with the blocking rod lowered at the back and is stopped.
- When the load exceeds a certain weight, the blocking rod goes up and the vehicle starts. After that, the blocking rod goes down again.
- When the vehicle reaches its destination and steps on the pressure sensor, a second blocking rod comes down and stops the vehicle.
- When the load on the vehicle is unloaded, the block rod returns to its original state and the vehicle returns to its reverse state.
- When the starting line is reached, the blocking rod is lowered to stop the vehicle, completing the overall cycle.

MCU Configuration

MCU1(RCcar)

Configuration list	Ultrasonic sensor1(Back)	
System clock	PLL (84MHz)	
PWM	GPIOA, PIN6(Timer3, Channel 1)	
	PWM period: 50[msec]	
	PWM pulsewidth:10[usec]	
Input capture	GPIOB, pin6(Timer 4, Channel 1)	
	Counter clock: 0.1[MHz]	
	Rising edge: IC1	Falling edge: IC2

Configuration list	Ultrasonic sensor2(Front)	
System clock	PLL (84MHz)	
PWM	GPIOA, PIN6(Timer3, Channel 1)	
	PWM period: 50[msec]	
	PWM pulsewidth:10[usec]	
Input capture	GPIOB, pin6(Timer 4, Channel 3)	
	Counter clock: 0.1[MHz]	
	Rising edge: IC3	Falling edge: IC4

Configuration list	DC motor(x2)	
Timer (for PWM)	GPIOA, pin0 (Timer2, Channel 1)	Period: 0.5[ms]
	GPIOA, pin1 (Timer2, Channel 2)	
DC direction pin	GPIOC, Pin2	
	GPIOC, Pin3	
Pin configuration	Analog mode, No pupd	

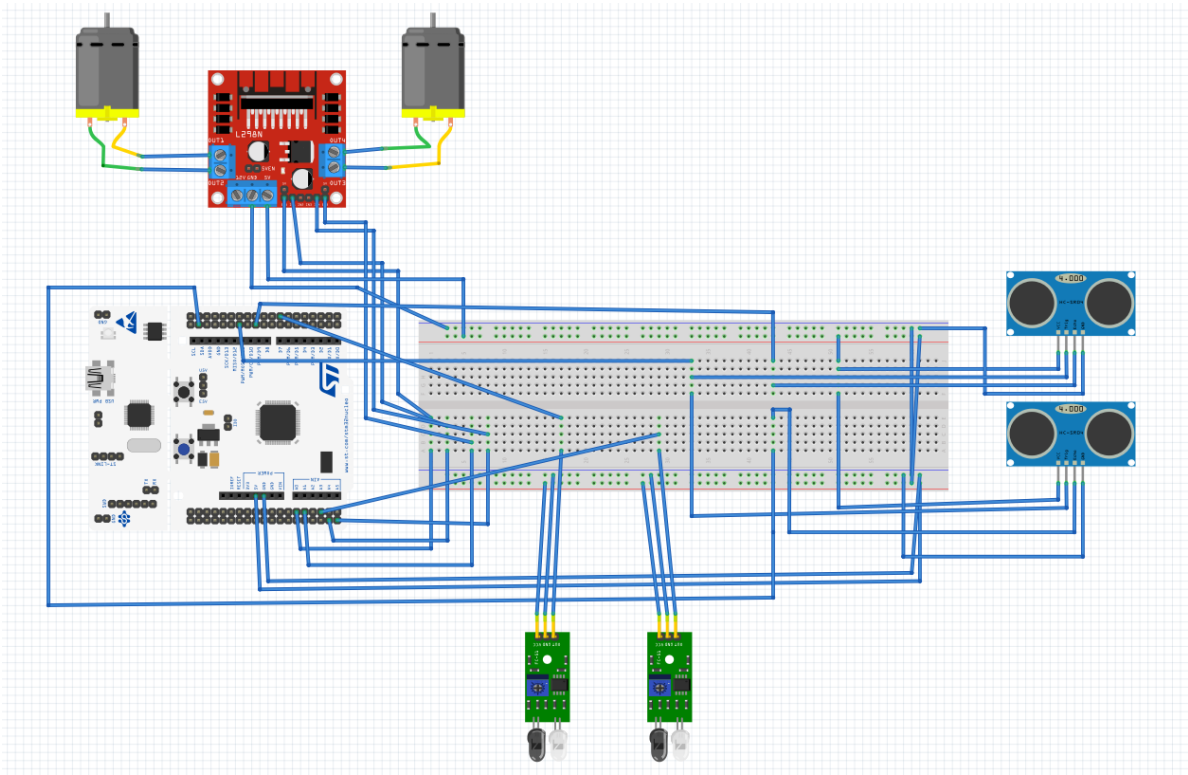
Configuration list	IR sensor
Timer	Timer3
	Up-counter, Counter CLK 1[kHz], OC1M, OC1REF
Pin	GPIOB, pin0
	GPIOB, pin1
Pin configuration	Analog mode, No pupd
ADC	ADC prescaler, Single conversion mode, Scan mode

MCU2

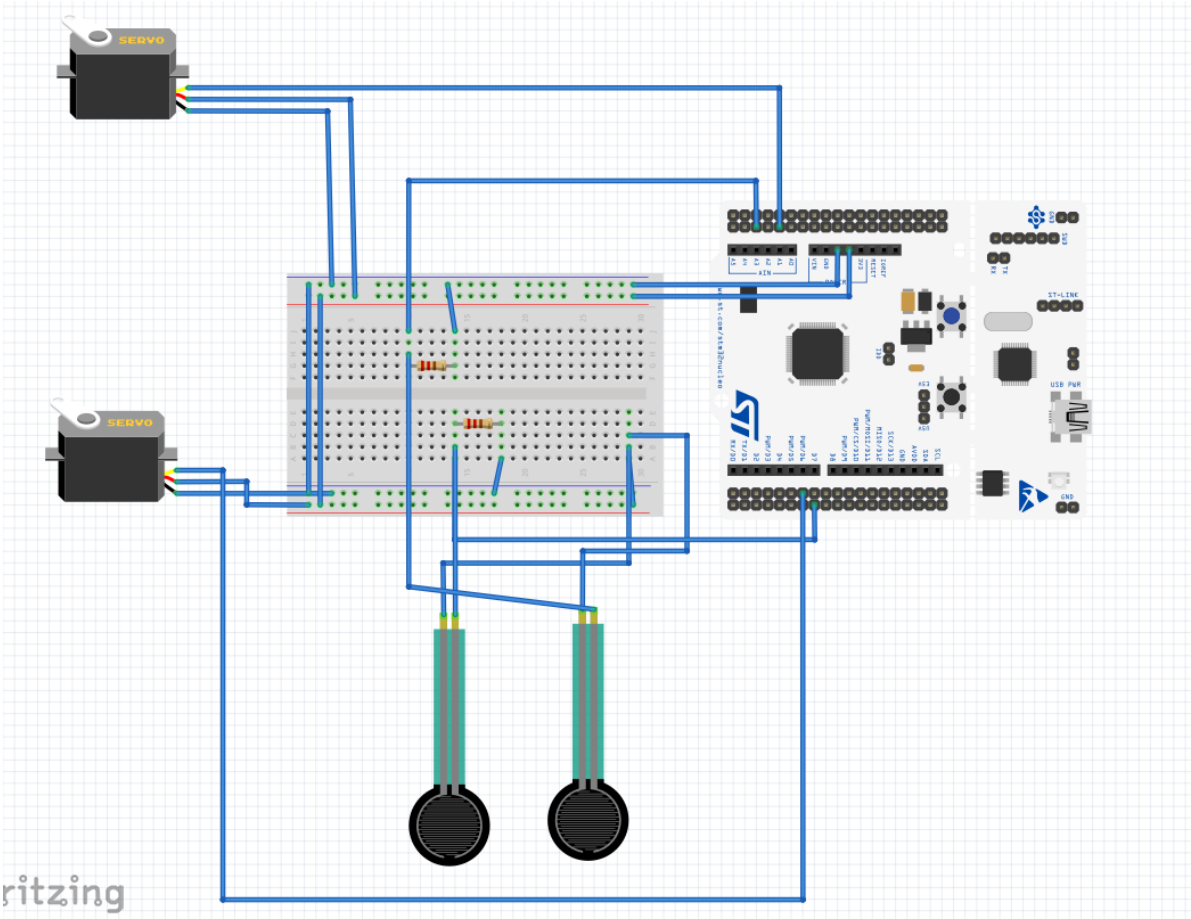
Configuration list	RC Servo motor(x2)
Pin	GPIOA, Pin1 (Timer2, channel 2)
	GPIOB, Pin10 (Timer2, channel 3)
Pin configuration	AF mode, push pull, fast speed
Driving method	If the pressure sensor is detected, Rotate 90 degree

Configuration list	Pressure Sensor(x2)
Pin	GPIOB, Pin0
	GPIOB, Pin1
Common configuration	Analog mode, no pull up pull down
ADC Hardware Trigger Configuration	Timer3, 1msec, Rising edge

Circuit Diagram



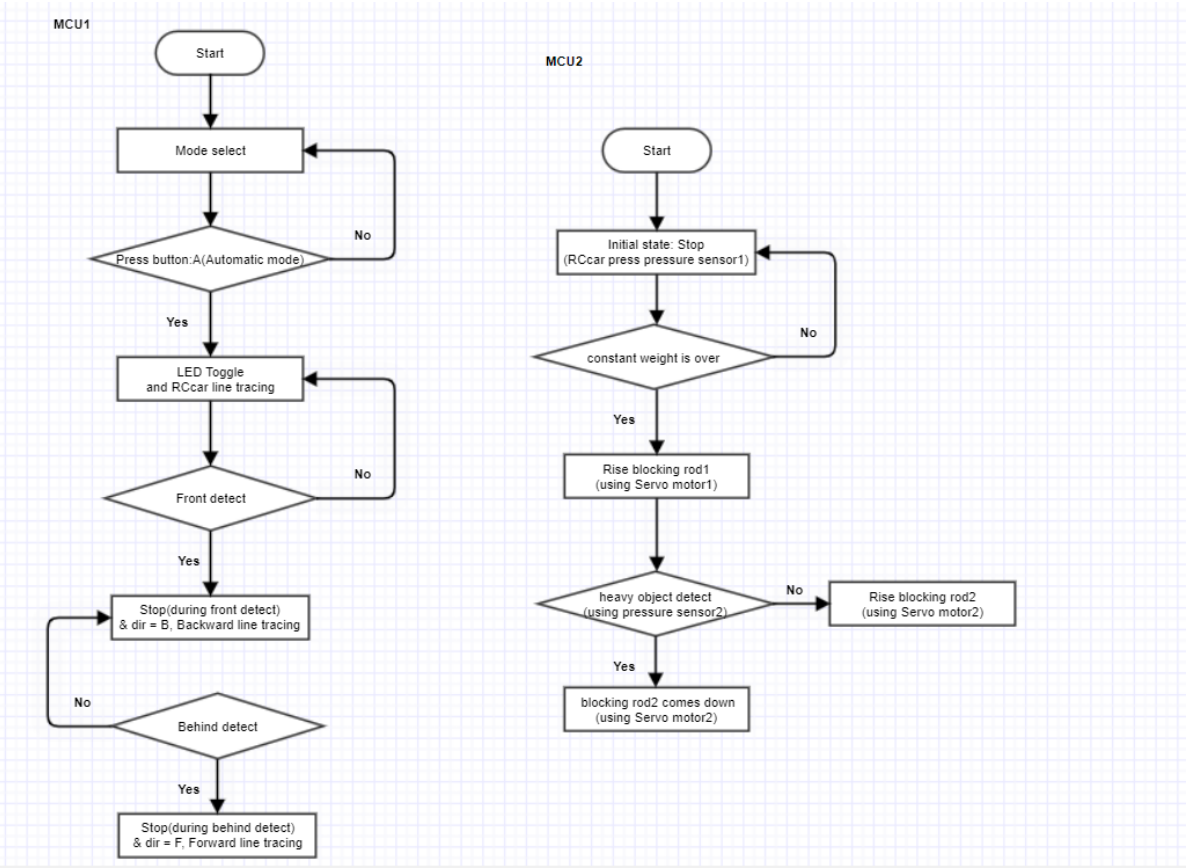
-> this is MCU1 circuit diagram



-> this is MCU2 circuit diagram

III. Algorithm

Logic Design



-> this is logic about MCU1 & MCU2

Code

MCU1

```
// Pin define
#define DIR_PIN1 2
#define DIR_PIN2 3
PinName_t PWM_PIN1 = PA_0; //right motor
PinName_t PWM_PIN2 = PA_1; //left motor

// 2 Ultra sensor
#define TRIG PA_6
#define ECHO_1 PB_6
#define ECHO_2 PB_8

// Velocity, Direction define
#define EX 1
#define v0 0.7
#define v1 0.5
#define v2 0.25
#define v3 0
#define F 1
#define B 0

// PWM period define
float period = 500;

// TIM4 count define
uint32_t _count = 0;

// UltraSonic parameter define
uint32_t ovf_cnt = 0;
float distance_1 = 0;  //back UltraSonic
float distance_2 = 0;  //front UltraSonic
float timeInterval_1 = 0;
float timeInterval_2 = 0;

float time1 = 0;
float time2 = 0;
float time3 = 0;
float time4 = 0;

// IR parameter define
uint32_t value1, value2;
int flag = 0;
PinName_t seqCHn[2] = {PB_0, PB_1};

// USART1(Bluetooth) parameter define
static volatile uint8_t PC_Data = 0;
static volatile uint8_t BT_Data = 0;

// Other parameter define
int i=0;      // speed level
char mode;      // mode = 'Manual' or 'Auto'
double vel[4] = {v0, v1, v2, v3};   // velocity levels
```

```
int str_level = 0;   // Steer level
double vel1 = 1;   // 1st DC motor duty ratio
double vel2 = 1;   // 2nd DC motor duty ratio
uint8_t dir = 1;   // Direction

//Flag
int flag_1 =0;

//TIM5
static volatile uint32_t count = 0;


// char for printState
char DIR;
char VEL[2];
char STR[2];


// Function Defines
void setup(void);
double str_angle(int str_level);
void printState(void);
void speedUP();
void speedDOWN();
void M_right();
void M_left();
void M_straight();
void B_stop();
void F_stop();
void M_back();
void LED_toggle();
void Automatic_mode(void);
void TIM5_IRQHandler(void);
```

Necessary variables and functions were defined.

```
void USART1_IRQHandler(){
    if(is_USART1_RXNE()){
        BT_Data = USART1_read();      // Send Data PC to Bluetooth'
            USART_write(USART1, &BT_Data,1);

        if(BT_Data == 'A'){      // Auto mode
                  USART1_write("Auto Mode\r\n",11);
            mode = 'A';
        }

    }
}
```

This code sets the RC car's mode to A by pressing 'A' using the USART1 IRQhandler.

```
void TIM4_IRQHandler(void){
```

```
    if(is_UIF(TIM4)){                       // Update interrupt
        ovf_cnt++;                          // overflow count
        _count++;
        clear_UIF(TIM4);                    // clear update interrupt flag
    }


      //1
    if(is_CCIF(TIM4, 1)){                          // TIM4_Ch1 (IC1) Capture
Flag. Rising Edge Detect
        time1 = TIM4->CCR1;                             // 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 = TIM4->CCR2;                               // Capture TimeEnd
        timeInterval_1 = ((time2 - time1) + (TIM4->ARR+1) * ovf_cnt) * 0.01;
 // (10us * counter pulse -> [msec] unit) Total time of echo pulse
        ovf_cnt = 0;                           // overflow reset
        clear_CCIF(TIM4,2);                             // clear capture/compare
interrupt flag

    }

      if(is_CCIF(TIM4, 3)){                           // TIM4_Ch3 (IC3) Capture
Flag. Rising Edge Detect
            time3 =  TIM4->CCR3;                       // Capture TimeStart  ,ARR
            clear_CCIF(TIM4, 3);                 // clear capture/compare
interrupt flag
      }

      else if(is_CCIF(TIM4, 4)){                               // TIM4_Ch4 (IC4)
Capture Flag. Falling Edge Detect
            time4 = TIM4->CCR4;                             // Capture TimeEnd
,ARR
            timeInterval_2 = (time4-time3 + ovf_cnt*((TIM4->ARR)+1))/100;    //
(10us * counter pulse -> [msec] unit) Total time of echo pulse
            ovf_cnt= 0;                           // overflow reset
            clear_CCIF(TIM4,4);                             // clear
capture/compare interrupt flag

     }

      distance_1 = (float) timeInterval_1 * 340.0 / 2.0 / 10.0;    // [mm] ->
[cm]
      distance_2= (float) timeInterval_2 * 340.0 / 2.0 / 10.0;    // [mm] ->
[cm]
}
```

This code uses TIM4_IRQhandler to operate two ultrasonic sensors.

```
void Automatic_mode(void){
```

```
            if(mode == 'A'){   // Auto mode

                    {
                       if(flag_1 == 0) //not front detected
                       {


                                dir = F;

                                if(value1 < 1000 && value2 < 1000){   // Move
Straight
                                        vel1 = 0.4;
                                        vel2 = 0.4;
                                  }
                                  else if(value1 > 1000 && value2 < 1000){   // Turn
right
                                        vel1 = 0.8;
                                        vel2 = 0.3;
                                  }
                                  else if(value1 < 1000 && value2 > 1000){   // Turn
left
                                        vel1 = 0.3;
                                        vel2 = 0.8;
                                  }
                                  else if(value1 > 1000 && value2 > 1000){   // STOP
                                        vel1 = 1;
                                        vel2 = 1;
                                  }
                                        // DC motor operate
                                  GPIO_write(GPIOC, DIR_PIN1, dir);
                                  GPIO_write(GPIOC, DIR_PIN2, dir);
                                  PWM_duty(PWM_PIN1, vel1);
                                  PWM_duty(PWM_PIN2, vel2);
                    }

                      else if(flag_1 == 1) //front detected
                       {
                                 dir = B;

                                if(value1 < 1000 && value2 < 1000){   // Move
Straight
                                        vel1 = 0.7;
                                        vel2 = 0.7;
                                  }
                                  else if(value1 < 1000 && value2 > 1000){   // Turn
right
                                        vel1 = 0.5;
                                        vel2 = 0.3;
                                  }
                                  else if(value1 > 1000 && value2 < 1000){   // Turn
left
                                        vel1 = 0.3;
                                        vel2 = 0.5;
                                  }
                                  else if(value1 > 1000 && value2 > 1000){   // STOP
                                        vel1 = 0;
                                        vel2 = 0;
```

```
                                }
                                      // DC motor operate
                                GPIO_write(GPIOC, DIR_PIN1, dir);
                                GPIO_write(GPIOC, DIR_PIN2, dir);
                                PWM_duty(PWM_PIN1, vel1);
                                PWM_duty(PWM_PIN2, vel2);



                            }
                    }

                    // car back sensor

                    if(distance_2 <7 && distance_1>7){   // front stop
                      B_stop();
                       flag_1 = 1;

                    }

                    if(distance_1 < 7 && distance_2 >7){     // back stop

                       B_stop();
                       flag_1 = 0;
                    }

                    if(_count >= 1){   // printing state, toggling every 1 second
                       LED_toggle();
                       printState();
                       _count = 0;
                    }

            }

}
```

This is a function for motor operation, and the code was written so that when the RC car detects an ultrasonic sensor while operating, the flag is toggled and the direction of the motor changes.

```
void setup(void){
    RCC_PLL_init();
    SysTick_init();                     // SysTick Init
    UART2_init();
    // LED
     GPIO_init(GPIOA, LED_PIN,OUTPUT);

    // BT serial init
    UART1_init();
    UART1_baud(BAUD_9600);

    // DIR1 SETUP
    GPIO_init(GPIOC, DIR_PIN1, OUTPUT);
    GPIO_otype(GPIOC, DIR_PIN1, EC_PUSH_PULL);

    // DIR2 SETUP
```

```
    GPIO_init(GPIOC, DIR_PIN2, OUTPUT);
    GPIO_otype(GPIOC, DIR_PIN2, EC_PUSH_PULL);

    // ADC Init
    ADC_init(PB_0);
    ADC_init(PB_1);

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

    // PWM1
    PWM_init(PWM_PIN1);
    PWM_period_us(PWM_PIN1, period);

    // PWM2
    PWM_init(PWM_PIN2);
    PWM_period_us(PWM_PIN2, period);

    // PWM configuration ---------------------------------------------------
-------------
    PWM_init(TRIG);         // 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 -----------------------------------------
-------------------------
    ICAP_init(ECHO_1);       // PB_6 as input caputre
    ICAP_counter_us(ECHO_1, 10);      // ICAP counter step time as 10us
    ICAP_setup(ECHO_1, 1, IC_RISE);  // TIM4_CH1 as IC1 , rising edge detect
    ICAP_setup(ECHO_1, 2, IC_FALL);  // TIM4_CH2 as IC2 , falling edge detect

      ICAP_init(ECHO_2);       // PB_8 as input caputre
    ICAP_counter_us(ECHO_2, 10);      // ICAP counter step time as 10us
    ICAP_setup(ECHO_2, 3, IC_RISE);  // TIM4_CH1 as IC1 , rising edge detect
    ICAP_setup(ECHO_2, 4, IC_FALL);  // TIM4_CH2 as IC2 , falling edge detect

}
```

This is a code that sets up functions related to RC car operation.

MCU2

```
#include "ecSTM32F411.h"

//--Servo Motor1--//
#define PWM_PIN PA_1


//--Servo Motor2--//
#define PWM_PIN3 PB_10


//Piezo parameter//
uint32_t PI1; //Heavy detect
uint32_t PI3; //Start line detect
```

```
int flag = 0;
PinName_t seqCHn[2] = {PB_0, PB_1};
```

This code defines two RC Motor pins and defines the values of the two pressure sensors as PI1 and PI3.

```
int main(void) {

   // Initialiization --------------------------------------------------------
   setup();

   // Inifinite Loop ----------------------------------------------------------
   while(1){
      printf("PI1 = %d \r\n",PI1);
      printf("PI3 = %d \r\n",PI3);
      printf("\r\n");


      delay_ms(1000);
   }
}
```

This code prints the pressure sensor value every second in the main function.

```
// Initialiization
void setup(void)
{
   RCC_PLL_init();                         // System Clock = 84MHz
   UART2_init();
   SysTick_init();

   // ADC setting
  ADC_init(PB_0); //Start line
  ADC_init(PB_1); //Heavy object detect

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

   // PWM of 20 msec:  TIM2_CH2 (PA_1 AFmode)
   PWM_init(PWM_PIN);
   PWM_period(PWM_PIN,20);   // 20 msec PWM period


    // PWM of 20 msec:  TIM2_CH3 (PB_10 AFmode)
   PWM_init(PWM_PIN3);
   PWM_period(PWM_PIN3,20);   // 20 msec PWM period

   TIM_UI_init(TIM3, 1);         // TIM3 Update-Event Interrupt every 1 msec
   TIM_UI_enable(TIM3);

}
```

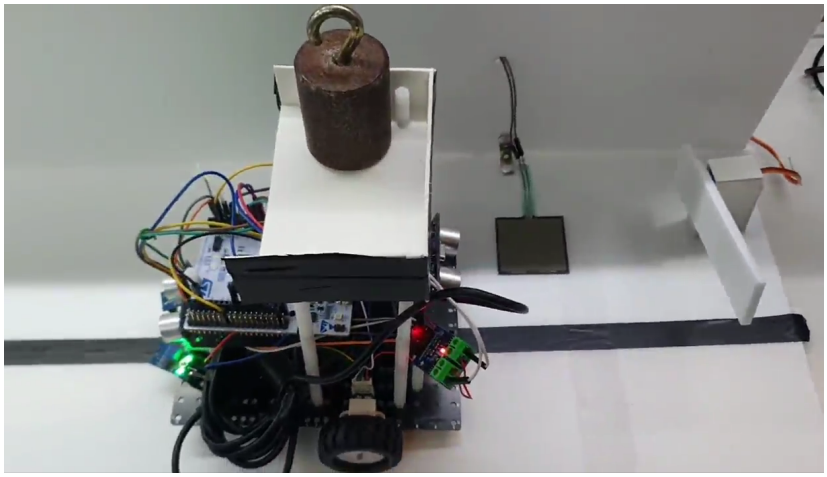
ADC was used to use the pressure sensor, and this is the corresponding code.

```
void TIM3_IRQHandler(void){

   if(is_UIF(TIM3)){         // Check UIF(update interrupt flag)

       if (PI1 >1000.0) { //Pressure sensor detected on
           PWM_duty(PWM_PIN,(1.5/20));  //servo motor rotate 90 degree
          }
       else if (PI1 <1000.0){
           PWM_duty(PWM_PIN,(0.5/20));  //servo motor comes back 0 degree
       }


       if (PI3 >1000.0) {  //Pressure sensor detected on
           PWM_duty(PWM_PIN3,(1.5/20)); //servo motor rotate 90 degree
          }
       else if (PI3 <1000.0){
           PWM_duty(PWM_PIN3,(0.5/20)); //servo motor comes back 0 degree
       }


      clear_UIF(TIM3);       // Clear UI flag by writing 0
   }
}
```

This code uses TIM3_IRQHandler to control the RC motor according to the value of the pressure sensor.

IV. Results and Demo

Press the 'A' key to turn the RC car on. When cargo is loaded, the blocking bar at the starting point rises and the RC car operates. When the arrival point is reached, the blocking bar comes down and the RC car stops. When the cargo is unloaded, the blocking bar at the destination rises and the RC car returns to the departure point.



Video Link: https://youtu.be/y8p7ZOsekoc

V. Reference

https://ykkim.gitbook.io/ec/ec-course/lab/lab-line-tracing-rc-car

https://ykkim.gitbook.io/ec/ec-course/lab/lab-usart-led-bluetooth

https://ykkim.gitbook.io/ec/ec-course/tutorial/tutorial-bluetooth

https://ykkim.gitbook.io/ec/ec-course/project/past-projects

https://ykkim.gitbook.io/ec/ec-course/project

VI. Troubleshooting

1. There were difficulties in using two ultrasonic sensors in one MCU.

   -> Trigger used the same timer, and echo used a different channel.

2. Problem with line tracing not working well when reversing depending on the position of the RC car's IR sensor and motor.

   -> This was resolved by setting the reverse speed to be different from the forward speed and also adjusting the left and right turn speeds.

VII. Appendix

MCU1 Code

```
#include "ecSTM32F411.h"
#include "math.h"
#include "stdio.h"
// Pin define
#define DIR_PIN1 2
#define DIR_PIN2 3
PinName_t PWM_PIN1 = PA_0; //right motor
PinName_t PWM_PIN2 = PA_1; //left motor

// 2 Ultra sensor
#define TRIG PA_6
#define ECHO_1 PB_6
#define ECHO_2 PB_8

// Velocity, Direction define
#define EX 1
#define v0 0.7
#define v1 0.5
#define v2 0.25
#define v3 0
#define F 1
#define B 0

// PWM period define
float period = 500;
```

```
// TIM4 count define
uint32_t _count = 0;

// UltraSonic parameter define
uint32_t ovf_cnt = 0;
float distance_1 = 0;  //back UltraSonic
float distance_2 = 0;  //front UltraSonic
float timeInterval_1 = 0;
float timeInterval_2 = 0;

float time1 = 0;
float time2 = 0;
float time3 = 0;
float time4 = 0;

// IR parameter define
uint32_t value1, value2;
int flag = 0;
PinName_t seqCHn[2] = {PB_0, PB_1};

// USART1(Bluetooth) parameter define
static volatile uint8_t PC_Data = 0;
static volatile uint8_t BT_Data = 0;

// Other parameter define
int i=0;      // speed level
char mode;      // mode = 'Manual' or 'Auto'
double vel[4] = {v0, v1, v2, v3};   // velocity levels
int str_level = 0;   // Steer level
double vel1 = 1;   // 1st DC motor duty ratio
double vel2 = 1;   // 2nd DC motor duty ratio
uint8_t dir = 1;   // Direction

//Flag
int flag_1 =0;

//TIM5
static volatile uint32_t count = 0;


// char for printState
char DIR;
char VEL[2];
char STR[2];


// Function Defines
void setup(void);
double str_angle(int str_level);
void printState(void);
void speedUP();
void speedDOWN();
void M_right();
void M_left();
void M_straight();
void B_stop();
void F_stop();
```

```
void M_back();
void LED_toggle();
void Automatic_mode(void);
void TIM5_IRQHandler(void);

void main(){
    setup();
      // Initial State (STOP)
    GPIO_write(GPIOC, DIR_PIN1, dir);
    GPIO_write(GPIOC, DIR_PIN2, dir);
    PWM_duty(PWM_PIN1, vel1);
    PWM_duty(PWM_PIN2, vel2);

      //TIM 5
      TIM_UI_init(TIM5,1);

    while(1){


      printf("value_1=%d \r\n", value1);
         printf("value_2 = %d \r\n", value2);
         printf("flag = %d \r\n", flag_1);
         printf("distance 2 = %d \r\n", distance_2);
         printf("distance1  = %d \r\n", distance_1);
         delay_ms(1000);
    }
}

void USART1_IRQHandler(){
    if(is_USART1_RXNE()){
        BT_Data = USART1_read();      // Send Data PC to Bluetooth'
            USART_write(USART1, &BT_Data,1);

        if(BT_Data == 'A'){      // Auto mode
                  USART1_write("Auto Mode\r\n",11);
            mode = 'A';
        }

    }
}
// Print the state (Manual or Auto mode)
void printState(void){

   if(mode == 'A'){   // Automation mode
      if(distance_2 < 8){
         USART1_write("Obstacle Infront\r\n", 18);
      }
      else{
         if(value1 < 1000 && value2 < 1000){
            USART1_write("Straight\r\n",10);
         }
         else if(value1 > 1000 && value2 < 1000){
            USART1_write("Turn right\r\n", 13);
         }
         else if(value1 < 1000 && value2 > 1000){
            USART1_write("Turn left\r\n", 12);
         }
      }
```

```
   }
}
// IR sensor Handler
void ADC_IRQHandler(void){
    if(is_ADC_OVR())
        clear_ADC_OVR();

    if(is_ADC_EOC()){      // after finishing sequence
        if (flag==0)
            value1 = ADC_read();
        else if (flag==1)
            value2 = ADC_read();
        flag =! flag;      // flag toggle
    }
}
// TIM4 Handler (Ultra Sonic)
void TIM4_IRQHandler(void){
    if(is_UIF(TIM4)){                     // Update interrupt
        ovf_cnt++;                                    // overflow count
            _count++;                                       // count for 1sec
        clear_UIF(TIM4);                           // clear update interrupt
flag
    }


      //1
    if(is_CCIF(TIM4, 1)){                         // TIM4_Ch1 (IC1) Capture Flag.
Rising Edge Detect
        time1 = TIM4->CCR1;                            // 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 = TIM4->CCR2;                             // Capture TimeEnd
        timeInterval_1 = ((time2 - time1) + (TIM4->ARR+1) * ovf_cnt) * 0.01;
// (10us * counter pulse -> [msec] unit) Total time of echo pulse
        ovf_cnt = 0;                          // overflow reset
        clear_CCIF(TIM4,2);                             // clear capture/compare
interrupt flag

    }

      if(is_CCIF(TIM4, 3)){                         // TIM4_Ch3 (IC3) Capture
Flag. Rising Edge Detect
            time3 =  TIM4->CCR3;                     // Capture TimeStart  ,ARR
            clear_CCIF(TIM4, 3);                // clear capture/compare
interrupt flag
      }

      else if(is_CCIF(TIM4, 4)){                             // TIM4_Ch4 (IC4)
Capture Flag. Falling Edge Detect
            time4 = TIM4->CCR4;                             // Capture TimeEnd
,ARR
            timeInterval_2 = (time4-time3 + ovf_cnt*((TIM4->ARR)+1))/100;    //
(10us * counter pulse -> [msec] unit) Total time of echo pulse
            ovf_cnt= 0;                          // overflow reset
```

```
            clear_CCIF(TIM4,4);                             // clear capture/compare
interrupt flag

     }

      distance_1 = (float) timeInterval_1 * 340.0 / 2.0 / 10.0;    // [mm] ->
[cm]
      distance_2= (float) timeInterval_2 * 340.0 / 2.0 / 10.0;    // [mm] ->
[cm]
}


void TIM5_IRQHandler(void){
   if(is_UIF(TIM5)){

     count++;
      if(count>200){
      Automatic_mode();
      count = 0;
      }
   }
   clear_UIF(TIM5);
}


void Automatic_mode(void){

           if(mode == 'A'){   // Auto mode

                 {
                    if(flag_1 == 0) //not front detected
                    {


                           dir = F;

                           if(value1 < 1000 && value2 < 1000){   // Move
Straight
                                  vel1 = 0.4;
                                  vel2 = 0.4;
                             }
                             else if(value1 > 1000 && value2 < 1000){   // Turn
right
                                  vel1 = 0.8;
                                  vel2 = 0.3;
                             }
                             else if(value1 < 1000 && value2 > 1000){   // Turn
left
                                  vel1 = 0.3;
                                  vel2 = 0.8;
                             }
                             else if(value1 > 1000 && value2 > 1000){   // STOP
                                  vel1 = 1;
                                  vel2 = 1;
                             }
                                  // DC motor operate
                             GPIO_write(GPIOC, DIR_PIN1, dir);
                             GPIO_write(GPIOC, DIR_PIN2, dir);
```

```
                        PWM_duty(PWM_PIN1, vel1);
                        PWM_duty(PWM_PIN2, vel2);
                }

                  else if(flag_1 == 1) //front detected
                   {
                            dir = B;

                           if(value1 < 1000 && value2 < 1000){   // Move
Straight
                                  vel1 = 0.7;
                                  vel2 = 0.7;
                            }
                            else if(value1 < 1000 && value2 > 1000){   // Turn
right
                                  vel1 = 0.5;
                                  vel2 = 0.3;
                            }
                            else if(value1 > 1000 && value2 < 1000){   // Turn
left
                                  vel1 = 0.3;
                                  vel2 = 0.5;
                            }
                            else if(value1 > 1000 && value2 > 1000){   // STOP
                                  vel1 = 0;
                                  vel2 = 0;
                            }
                                  // DC motor operate
                            GPIO_write(GPIOC, DIR_PIN1, dir);
                            GPIO_write(GPIOC, DIR_PIN2, dir);
                            PWM_duty(PWM_PIN1, vel1);
                            PWM_duty(PWM_PIN2, vel2);



                   }
              }

                 // car back sensor

                 if(distance_2 <7 && distance_1>7){   // front stop
                   B_stop();
                    flag_1 = 1;

                 }

                 if(distance_1 < 7 && distance_2 >7){     // back stop

                    B_stop();
                    flag_1 = 0;
                 }

                 if(_count >= 1){   // printing state, toggling every 1 second
                    LED_toggle();
                    printState();
                    _count = 0;
                 }
```

```
        }

}

void B_stop(){
    dir = F;
    vel1 = EX;
    vel2 = EX;
      GPIO_write(GPIOC, DIR_PIN1, dir);
      GPIO_write(GPIOC, DIR_PIN2, dir);
      PWM_duty(PWM_PIN1, vel1);
      PWM_duty(PWM_PIN2, vel2);
}


void F_stop(){
    dir = B;
    vel1 = 0;
    vel2 = 0;
      GPIO_write(GPIOC, DIR_PIN1, dir);
      GPIO_write(GPIOC, DIR_PIN2, dir);
      PWM_duty(PWM_PIN1, vel1);
      PWM_duty(PWM_PIN2, vel2);


}


void LED_toggle(void){
    static unsigned int out = 0;
    if(out == 0) out = 1;
    else if(out == 1) out = 0;
    GPIO_write(GPIOA, LED_PIN, out);
}
void setup(void){
    RCC_PLL_init();
    SysTick_init();                     // SysTick Init
    UART2_init();
    // LED
     GPIO_init(GPIOA, LED_PIN,OUTPUT);

    // BT serial init
    UART1_init();
    UART1_baud(BAUD_9600);

    // DIR1 SETUP
    GPIO_init(GPIOC, DIR_PIN1, OUTPUT);
    GPIO_otype(GPIOC, DIR_PIN1, EC_PUSH_PULL);

    // DIR2 SETUP
    GPIO_init(GPIOC, DIR_PIN2, OUTPUT);
    GPIO_otype(GPIOC, DIR_PIN2, EC_PUSH_PULL);

    // ADC Init
    ADC_init(PB_0);
    ADC_init(PB_1);

    // ADC channel sequence setting
```

```
    ADC_sequence(seqCHn, 2);

    // PWM1
    PWM_init(PWM_PIN1);
    PWM_period_us(PWM_PIN1, period);

    // PWM2
    PWM_init(PWM_PIN2);
    PWM_period_us(PWM_PIN2, period);

    // PWM configuration ---------------------------------------------------------
-------------
    PWM_init(TRIG);         // 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 -----------------------------------------------
------------------------
    ICAP_init(ECHO_1);       // PB_6 as input caputre
    ICAP_counter_us(ECHO_1, 10);      // ICAP counter step time as 10us
    ICAP_setup(ECHO_1, 1, IC_RISE);  // TIM4_CH1 as IC1 , rising edge detect
    ICAP_setup(ECHO_1, 2, IC_FALL);  // TIM4_CH2 as IC2 , falling edge detect

      ICAP_init(ECHO_2);       // PB_8 as input caputre
    ICAP_counter_us(ECHO_2, 10);      // ICAP counter step time as 10us
    ICAP_setup(ECHO_2, 3, IC_RISE);  // TIM4_CH1 as IC1 , rising edge detect
    ICAP_setup(ECHO_2, 4, IC_FALL);  // TIM4_CH2 as IC2 , falling edge detect

}
```

MCU2 Code

```
#include "ecSTM32F411.h"

//--Servo Motor1--//
#define PWM_PIN PA_1


//--Servo Motor2--//
#define PWM_PIN3 PB_10


//Piezo parameter//
uint32_t PI1; //Heavy detect
uint32_t PI3; //Start line detect


int flag = 0;
PinName_t seqCHn[2] = {PB_0, PB_1};




void setup(void);
```

```
int main(void) {

   // Initialiization --------------------------------------------------------
   setup();

   // Inifinite Loop ---------------------------------------------------------
   while(1){
      printf("PI1 = %d \r\n",PI1);
      printf("PI3 = %d \r\n",PI3);
      printf("\r\n");


      delay_ms(1000);
   }
}

// Initialiization
void setup(void)
{
   RCC_PLL_init();                          // System Clock = 84MHz
   UART2_init();
   SysTick_init();

   // ADC setting
  ADC_init(PB_0); //Start line
  ADC_init(PB_1); //Heavy object detect

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

   // PWM of 20 msec:  TIM2_CH2 (PA_1 AFmode)
   PWM_init(PWM_PIN);
   PWM_period(PWM_PIN,20);   // 20 msec PWM period


    // PWM of 20 msec:  TIM2_CH3 (PB_10 AFmode)
   PWM_init(PWM_PIN3);
   PWM_period(PWM_PIN3,20);   // 20 msec PWM period

   TIM_UI_init(TIM3, 1);         // TIM3 Update-Event Interrupt every 1 msec
   TIM_UI_enable(TIM3);

}

void ADC_IRQHandler(void){
   if((is_ADC_OVR())){
       clear_ADC_OVR();
   }

   if(is_ADC_EOC()){       //after finishing sequence

         if(flag==0){
             PI1 = ADC_read();

          }
         else if(flag==1){
             PI3 = ADC_read();
```

```
          }
          flag =!flag;
   }
}


void TIM3_IRQHandler(void){

   if(is_UIF(TIM3)){         // Check UIF(update interrupt flag)

       if (PI1 >1000.0) { //Pressure sensor detected on
           PWM_duty(PWM_PIN,(1.5/20));  //servo motor rotate 90 degree
          }
       else if (PI1 <1000.0){
           PWM_duty(PWM_PIN,(0.5/20));  //servo motor comes back 0 degree
       }


       if (PI3 >1000.0) {  //Pressure sensor detected on
           PWM_duty(PWM_PIN3,(1.5/20)); //servo motor rotate 90 degree
          }
       else if (PI3 <1000.0){
           PWM_duty(PWM_PIN3,(0.5/20)); //servo motor comes back 0 degree
       }


      clear_UIF(TIM3);       // Clear UI flag by writing 0
   }
}
```