LAB: Line Tracing RC Car

Date: 2023.11.23

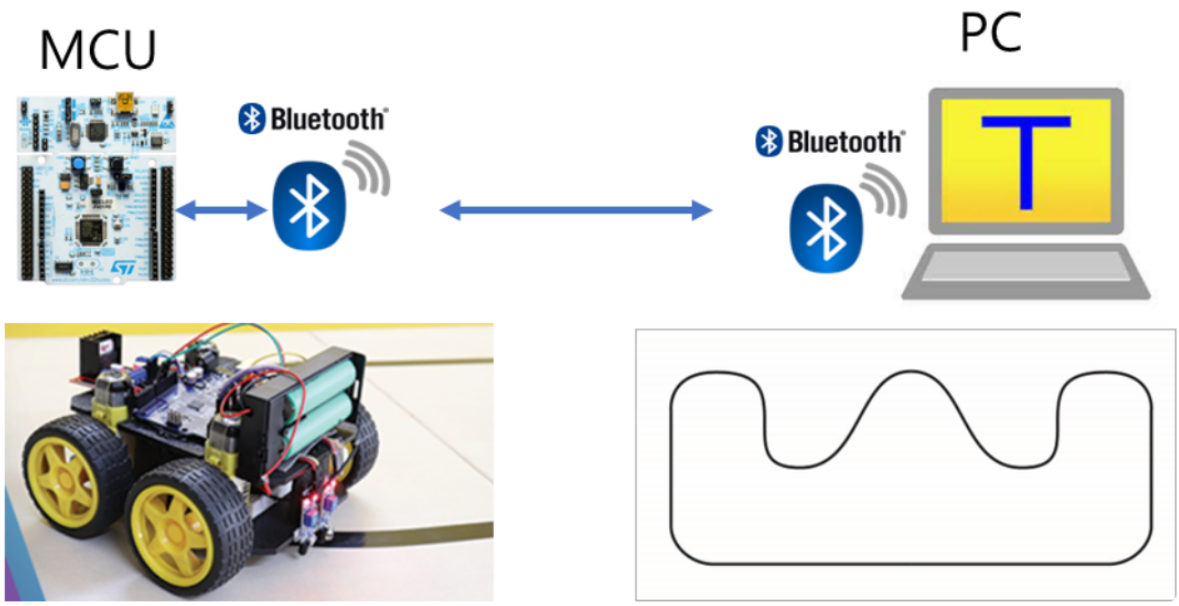
Author: Hanmin Kim

Github: https://github.com/ansterdaz/HKim

Demo video: https://youtu.be/qzlFspFOxAk

Introduction

Design an embedded system to control an RC car to drive on the racing track. The car is controlled either manually with wireless communication or automatically to drive around the track. When it sees an obstacle on the driving path, it should temporarily stop until the obstacle is out of the path.



Requirement

Hardware

MCU

- NUCLEO-F411RE

Actuator/Sensor/Others: Minimum

- Bluetooth Module(HC-06)
- DC motor x2, DC motor driver(L9110s)
- IR Reflective Sensor (TCRT 5000) x2
- HC-SR04
- additional sensor/actuators are acceptable

Software

- Keil uVision, CMSIS, EC_HAL library

Problem Definition

Design your RC car that has the following functions:

1. Line tracing on the given racing track
2. has 2 control modes: **Manual Mode** to **AUTO Mode**
3. stops temporally when it detects an object nearby on the driving path

On the PC, connected to MCU via bluetooth

- Print the car status every 1 sec such as " (" MOD: A DIR: F STR: 00 VEL: 00 ")

Manual Mode

- Mode Change(MOD):
  - When 'M' or 'm' is pressed, it should enter **Manual Mode**
  - LD2 should be ON in Manual Mode
- Speed (VEL):
  - Increase or decrease speed each time you push the arrow key "UP" or "DOWN", respectively.
  - You can choose the speed keys
  - Choose the speed level: V0 ~ V3
- Steer (STR):
  - Steering control with keyboard keys
  - Increase or decrease the steering angles each time you press the arrow key "RIGHT" or "LEFT", respectively.
  - Steer angles with 3 levels for both sides: e.g: -3, -2, -1, 0, 1, 2, 3 // '-' angle is turning to left
- Driving Direction (DIR)
  - Driving direction is forward or backward by pressing the key "F" or "B", respectively.
  - You can choose the control keys
- Emergency Stop
  - RC car must stop running when key "S" is pressed.

Automatic Mode

- Mode Change:
  - When 'A' or 'a' is pressed, it should enter **AUTO Mode**
- LD2 should blink at 1 second rate in AUTO Mode
- It should drive on the racing track continuously
- Stops temporally when it detects an object nearby on the driving path
- If the obstacle is removed, it should drive continuously

Procedure

1. Discuss with the teammate how to design an algorithm for this problem
2. In the report, you need to explain concisely how your system works with state tables/diagram or flow-chart.
   - Listing all necessary states (states, input, output etc) to implement this design problem.
   - Listing all necessary conditional FLAGS for programming.
   - Showing the logic flow from the initialization

   and more

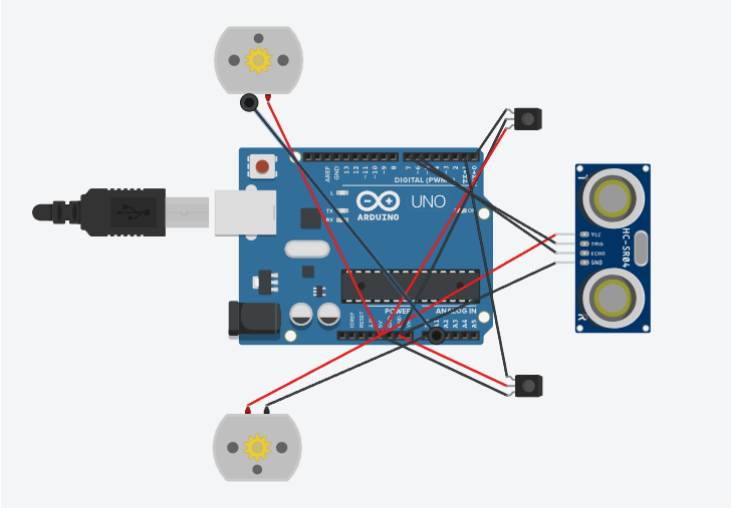
3. Select appropriate configurations for the design problem. Fill in the table.

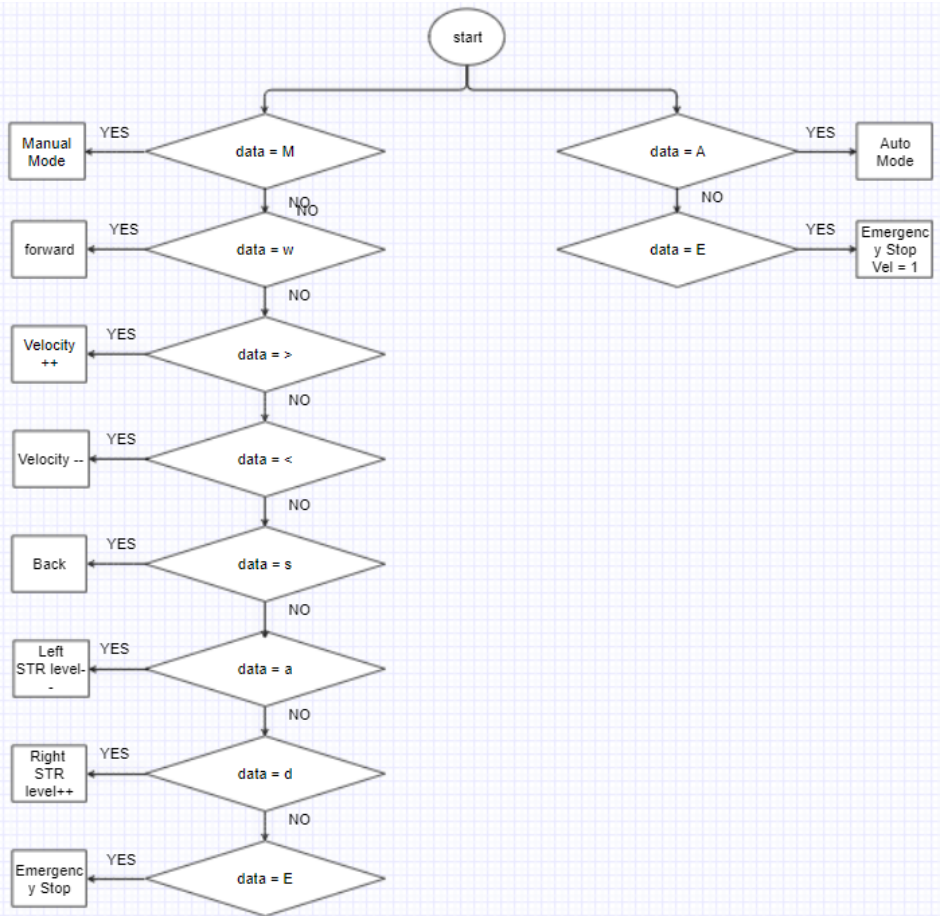
Functions	**Register**	**PORT_PIN**	**Configuration**
System Clock	RCC		PLL 84MHz
delay_ms	SysTick		
Motor DIR	Digital Out	R:PC3,L:PC2	
TIMER	TIMER1		
	TIMER2		
Timer Interrupt	…		10msec
ADC	ADC	PB0,PB1	
DC Motor Speed	PWM2	PA0,PA1	TIM2(PWM) perioud: 6msec, Duty ratio: 0~1
RS-232 USB cable(ST-LINK)	USART2		No Parity, 8-bit Data, 1-bit Stop bit 38400 baud-rate
Bluetooth	USART1	TXD: PA9 RXD: PA10	No Parity, 8-bit Data, 1-bit Stop bit 9600 baud-rate

4. Creat a new project under the directory \repos\EC\LAB\LAB_RCcar

- The project name is “**LAB_RCcar”**
- You can share the same code with your teammate. But need to write the report individually

Circuit Diagram





Code

```
// DIR pin
#define DIR_PIN1 2
#define DIR_PIN2 3
//PWM pin
PinName_t PWM_PIN1 = PA_0;
PinName_t PWM_PIN2 = PA_1;
//UltraSonic pin
#define TRIG PA_6
#define ECHO PB_6
//Emergency v
#define EX 1
// velocity
#define v0 0.7
#define v1 0.5
#define v2 0.25
#define v3 0
#define F 1
#define B 0

uint32_t _count = 0;
float period = 6;
// UltraSonic
uint32_t ovf_cnt = 0;
float distance = 0;
float timeInterval = 0;
float time1 = 0;
float time2 = 0;
//IR parameter//
uint32_t value1, value2;
int flag = 0;
PinName_t seqCHn[2] = {PB_0, PB_1};

static volatile uint8_t PC_Data = 0;
static volatile uint8_t BT_Data = 0;

int i=0;
char mode;
double vel[4] = {v0, v1, v2, v3};
int str_level = 0;
double vel1 = 1;
double vel2 = 1;
char DIR;
char VEL[2];
char STR[2];
uint8_t dir = 1;
```

Variables to be used in the code below, such as the motor's DIR pin and pwm pin, are defined.

Manual Mode

```
void USART1_IRQHandler(){
    if(is_USART1_RXNE()){
        BT_Data = USART1_read();
      // Manual Mode
        if(BT_Data == 'M') {
            USART1_write("Manual Mode\r\n",13);
            mode = 'M';
        }
      // Auto Mode
        else if(BT_Data == 'A'){
            USART1_write("Auto Mode\r\n",11);
            mode = 'A';
        }
      // Velocity increase
        if(mode == 'M') {
            if (BT_Data == '>'){
                speedUP();
            }
      // Velocity decrease
            else if (BT_Data == '<'){
                speedDOWN();
            }
      // right
            else if (BT_Data == 'd') {
                M_right();
      // left
            } else if (BT_Data == 'a') {
                M_left();
            } else if (BT_Data == 'w') {
                M_straight();
            }
      // back
            else if (BT_Data == 's') {
                M_back();
            }
      // Emergency Stop
            else if (BT_Data == 'E'){
                E_stop();
                USART1_write("Emergency\r\n", 11);
            }
        }
    }
}
```

Enter 'M' in USART1_IRQHandler to enter Manual Mode, then enter 'w', 's','>','<','a','d','E' to move forward, backward, increase/decrease speed, etc. Left turn, right turn, and emergency stop were implemented.

Auto Mode

```
  // Auto Mode
if(mode == 'A'){
    distance = (float) timeInterval * 340.0 / 2.0 / 10.0;    // [mm] -> [cm]

  // Forward
    if(value1 < 1000 && value2 < 1000){
        vel1 = 0;
        vel2 = 0;
    }
  // right
    else if(value1 > 1000 && value2 < 1000){ //
        vel1 = 0;
        vel2 = 0.6;
    }
  // left
    else if(value1 < 1000 && value2 > 1000){
        vel1 = 0.6;
        vel2 = 0;
    }
  // stop
    else if(value1 > 1000 && value2 > 1000){
        vel1 = 1;
        vel2 = 1;
    }
    // Obstacle Detection
    if(distance < 7){
      E_stop();
    }
    else if(distance > 3000){
      continue;
    }
```

Enter 'A' to set it to Auto Mode and write a code according to the value received by the IR sensor so that the RC car can do line tracing.

Rc car state print function

```
// Print the RC car state
void printState(void){

  if(mode == 'M'){
    sprintf(VEL, "%d", i);
    sprintf(STR, "%d", str_level);

    USART1_write("MOD: ", 5);
    USART1_write(&mode, 1);
    USART1_write(" DIR: ", 6);
    USART1_write(&DIR, 1);
    USART1_write(" STR: ", 6);
    USART1_write(&STR, 2);
    USART1_write(" VEL: ", 6);
    if(str_level == 0){
    USART1_write("V", 1);
    USART1_write(&VEL, 2);
    }
    USART1_write("\r\n", 2);
  }

  else if(mode == 'A'){
    if(distance < 7){
      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);
      }
    }
  }
}
```

In Manual Mode, the current Mode, DIR, STR, and VEL of the RC car are output.

In Auto Mode, the current operating status of the RC car is output as straight, right, left, and obstacle infront.

RC car operation function

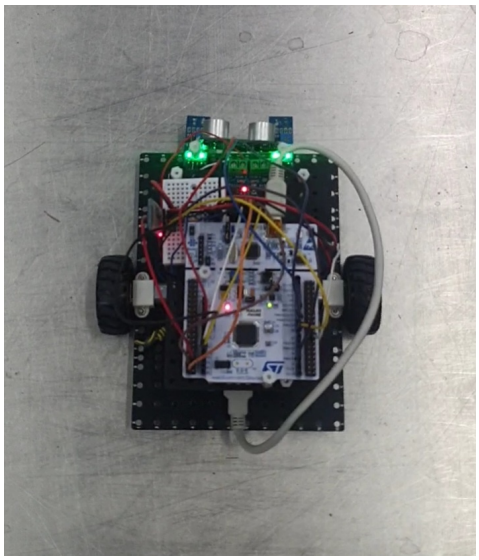
```
void speedUP(){
    i++;
    if(i>=3) i=3;
    vel1 = vel[i];
    vel2 = vel[i];
}

void speedDOWN(){
    i--;
    if(i<=0) i=0;
    vel1 = vel[i];
    vel2 = vel[i];
}

void M_right(){
    str_level--;
    if(str_level<-3) str_level=-3;
    str_angle(str_level);
}

void M_left(){
    str_level++;
    if(str_level>3) str_level=3;
    str_angle(str_level);
}

void M_straight(){
    str_level = 0;
    dir = F;
    vel1 = vel[i];
    vel2 = vel[i];
    DIR = 'F';
}

void M_back(){
    str_level = 0;
    dir = B;
    vel1 = vel[1];
    vel2 = vel[1];
    DIR = 'B';
}

void E_stop(){
    dir = F;
    vel1 = EX;
    vel2 = EX;
}
```

Created functions for motor speed increase/decrease, left/right turn angle increase/decrease, forward, reverse, and emergency stop.

```
double str_angle(int str_level){
    if(str_level == -1){
        vel1 = v2;
        vel2 = v1;
    }
    else if(str_level == -2){
        vel1 = v2;
        vel2 = v0;
    }
    else if(str_level == -3){
        vel1 = v3;
        vel2 = v0;
    }
    else if(str_level == 1){
        vel1 = v1;
        vel2 = v2;
    }
    else if(str_level == 2){
        vel1 = v0;
        vel2 = v2;
    }else if(str_level == 3){
        vel1 = v0;
        vel2 = v3;
    }
    else if(str_level == 0){
        vel1 = vel[i];
        vel2 = vel[i];
    }
}
```

The speed for each STR level was defined when turning left and right.

Results



Auto Mode

Enter A to operate the RC car in Auto Mode. At this time, the LED blinks and line tracing begins. If it detects an obstacle, it stops, and if there is no obstacle, it moves forward again.

Manual Mode

Enter M to operate the RC car in Manual Mode. At this time, the LED remains on. Enter 'a' and 'd' to move left and right, and input '>' and '< ' to increase or decrease the speed of the RC car. Press 's' to go backwards and 'E' to make an emergency stop.

video link: https://youtu.be/qzlFspFOxAk

Reference

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

https://github.com/ykkimhgu/Tutorial-C-Program/tree/main/structure

Trouble Shooting

1. Problem with Auto Mode not working

   Previously, Manual Mode was first designed within the USART1 handler and operation was checked.

   When designing Auto Mode, a problem occurred where Auto Mode did not work.

   -> The problem was solved by writing Auto Mode separately in a while statement.