Embedded Controller

LAB: RC Car Control with Bluetooth

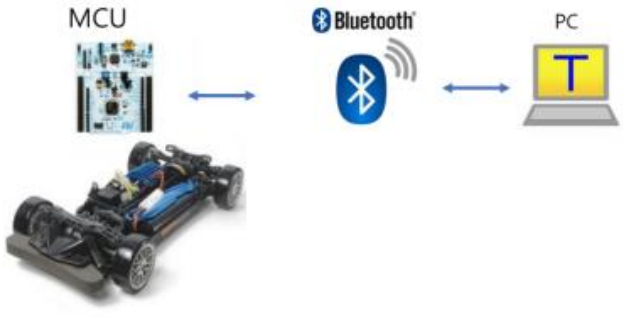
Date: 2021.12.02

Name(ID): Park Jeong Woo

Partner Name: Kim Ji Sung

I. Introduction

Design a simple program to control an RC car steering and speed by sending the command message from PC via bluetooth. Motor 1: RC servo – for Steering (LEFT or RIGHT) Motor 2: DC motor– for Speed and Heading Direction (FWD or BWD)



Hardware

NUCLEO -F411RE, Bluetooth Module(HC-06), breadboard, DC motor, DC motor driver(L9110s), LEDs x 3, Resistor 330 ohm x 3, breadboard, RC Servo Motor (SG90)

Software

Keil uVision IDE, CMSIS, EC_HAL

II. Problem

Create a program to control two DC motors by giving a command from PC using Bluetooth module. The program should perform the following tasks by the user keyboard input. You must use appropriate interrupts

Print the status every 1 sec such as " ("RC car: DIR: 00[deg] VEL: 00[%] FWD")

Steering: RC Servo (Motor 1)

- Divide 180° into 4 intervals.
- Intially start the angle of RC servor motor at 90°.
- Steering cotrol with keyboard key. Increase or decrease the angle of RC servor motor by 45° each time you push the arrow key "RIGHT" or "LEFT", respecively.

Speed: DC motor (Motor 2)

- Intially configure the duty ratio of DC motor at "0%"
- Increase or decrease the speed by 25% duty each time you push the arrow key "UP" or "DOWN", respecively.
- The RC car driving direction should be forward or backward by pressing the key "F" or "B", respecively.
- The RC car must stop running whe the key "S" is pressed.
- Apply PWM to (A-IA) of motor driver
- Apply Direction (H or L) to (A-IB) of motor driver

Key	Task	Comment
U	Motor Speed Increases	Increase PWM duty ratio by 25% for each press.Initial value : duty 0% The maxim duty value should be 100%
D	Motor Speed Decreases	Decrease PWM duty ratio by 25% for each press The minimum duty value should be 0%.
L	Left turn	Turn Left by 45degree
R	Right turn	Turn Right by 45degree
F	Foward direction	Go forward. Default setting DIR=1
B	Backward direction	Reverse driving. DIR=0
S	Stop	Make duty=0% for both motors

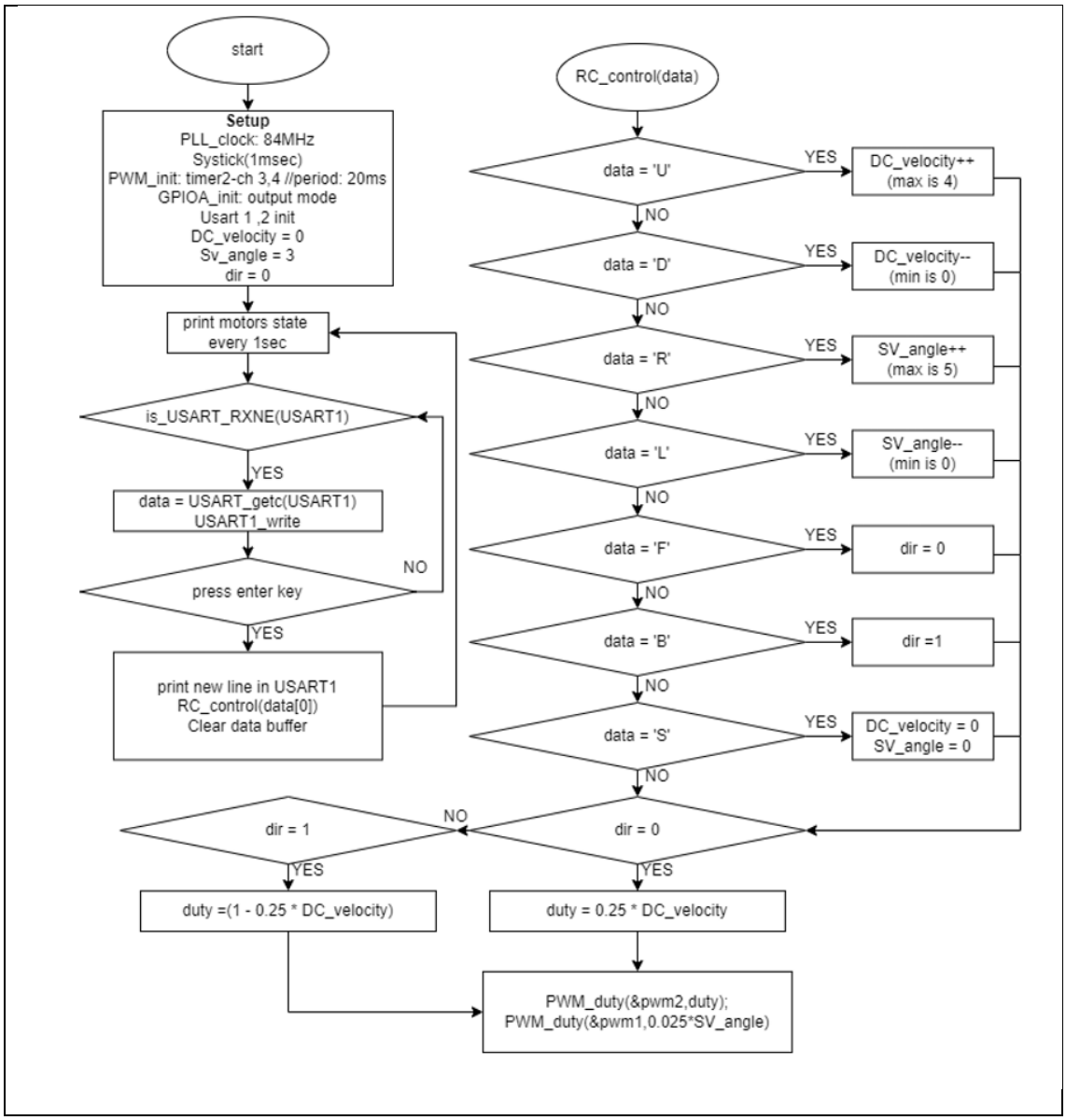
A. Configuration

You are free to select appropriate configurations for the design problem. Fill in the table.

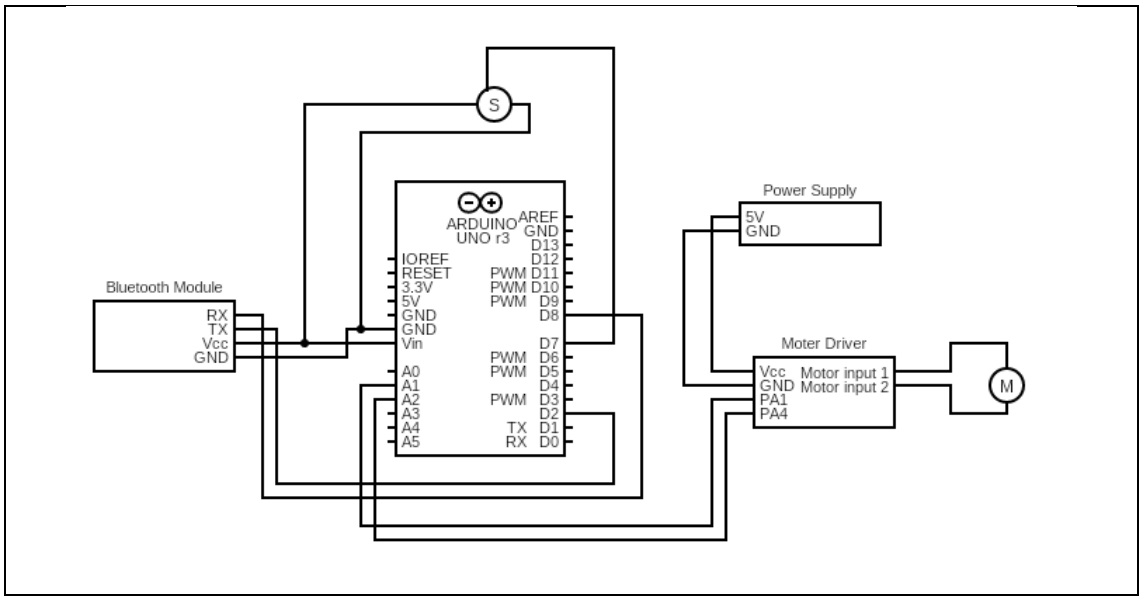
Functions	Register	PORT_PIN	Configuration
System Clock	RCC		PLL 84MHz
delay_ms	SysTick		
Motor DIR	Dıgital Out	PA4	
TIMER	TIMER2		For PWM1 – ch2
	TIMER2		For PWM2 – ch2
RC servo angle	PWM1	PA1	
DC Motor Speed	PWM2	PB10	
RS-232 USB cable(ST-LINK)	USART2	USB_cable	No Parity, 8-bit Data, 1-bit Stop bit 38400 baud-rate
Bluetooth	USART1	TXD: PA9 RXD: PA10	No Parity, 8-bit Data, 1-bit Stop bit 9600 baud-rate

Embedded Controller

- Circuit Diagram



B. Connetion wiring



C. Software

```
/**
*****************************************************************************************
* @author  SSSLAB
* @Mod       2021-11-27 by Park Jeong Woo
* @brief   Embedded Controller:  RC car
*
*****************************************************************************************
*/

// USART2 : MCU to PC via usb
// USART1 : MCU to MCU2

#include "ecInclude.h"

PWM_t pwm2;
PWM_t pwm1;

uint8_t mcu2Data = 0;
uint8_t pcData = 0;
int indx =0;
int maxBuf=10;
uint8_t buffer[100]={0,};
uint8_t buffer2 = '\r\n';
int endChar = 13;

int dir =0;
int DC_velocity =0;
int SV_angle = 3;
float duty = 0.f;

```

Embedded Controller

```c
void RC_control(char cmd);
void setup(void);

int main(void) {
    // Initialiization --------------------------------------------------------
    setup();
    printf("Hello Nucleo\r\n");
    delay_ms(500);


    // Inifinite Loop ---------------------------------------------------------
    while (1){

      if(dir ==0)
      printf("RC car: DIR:%d[deg] VEL:%d[%%] FWD\r\n\r\n",(SV_angle-1)*45,DC_velocity*25);
      else if(dir ==1)
      printf("RC car: DIR:%d[deg] VEL:%d[%%] BWD\r\n\r\n",(SV_angle-1)*45,DC_velocity*25);
      delay_ms(1000);
  }
}

// Initialiization
void setup(void)
{
   RCC_PLL_init();
   Systick_Init(1);

   // USART congfiguration
   USART_init(USART2, 38400);
   USART_begin(USART1, GPIOA,9,GPIOA,10, 9600);    // PA9 (D8) - RXD , PA10 (D2) - TXD

   //PWM setting
   PWM_init(&pwm1, GPIOA, 1);  //RC servo angle //TIM2 - ch2
   PWM_init(&pwm2, GPIOB, 10); //DC Motor Speed //TIM2 - ch3

   PWM_period_ms(&pwm1, 20);  //TIMER 2 period
   PWM_duty(&pwm2,0);          //DC default
   PWM_duty(&pwm1,0.075);      //servo default


   //GPIO output setting
   GPIO_init(GPIOA,4,OUTPUT);
   //GPIO_init(GPIOB,4,OUTPUT);


}

void RC_control(char cmd)
{
  switch(cmd){
    case 'U' : DC_velocity++; break;
    case 'D' : DC_velocity--; break;
    case 'R' : SV_angle++;    break;
    case 'L' : SV_angle--;    break;
    case 'F' : dir =0;        break;
    case 'B' : dir =1;        break;
    case 'S' : DC_velocity = 0; SV_angle = 3; break;
    }

  if     (DC_velocity < 0) DC_velocity = 0;
  else if(DC_velocity > 4) DC_velocity = 4;

  if     (SV_angle < 1) SV_angle = 1;
  else if(SV_angle > 5) SV_angle = 5;

  GPIO_write(GPIOA,4,dir);
  //GPIO_write(GPIOB,4,dir);

  if     (dir == 0) duty = 0.25 * (float)DC_velocity;
  else if(dir == 1) duty = 1.0 - 0.25 * (float)DC_velocity;

  PWM_duty(&pwm2,duty);
  PWM_duty(&pwm1,0.025*SV_angle);

}
```

```
void USART1_IRQHandler(){      //USART1 INT
   if(is_USART_RXNE(USART1)){
      mcu2Data = USART_getc(USART1);
      USART_write(USART1,&mcu2Data,1);
      if(mcu2Data==endChar) {
         RC_control(buffer[0]);
         USART_write(USART1,&buffer2,1);
         buffer[0] = NULL;
         indx = 0;

      }
      else{
         if(indx>maxBuf){
            indx =0;
            memset(buffer, 0, sizeof(char) * maxBuf);
            printf("ERROR : Too long string\r\n");
         }
         buffer[indx] = mcu2Data;
         indx++;

      }
   }
}
```

D. Conclusion

Using the libraries learned in the embedded controller class in this experiment, DC motor speed control, servo motor angle control, and Bluetooth communication were used to implement what RC-car needs. And I understood the driving principle of the motor driver for controlling the direction of the DC motor.

E. TrobleShooting

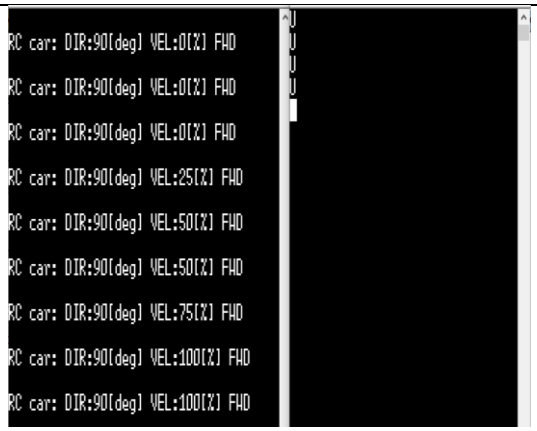
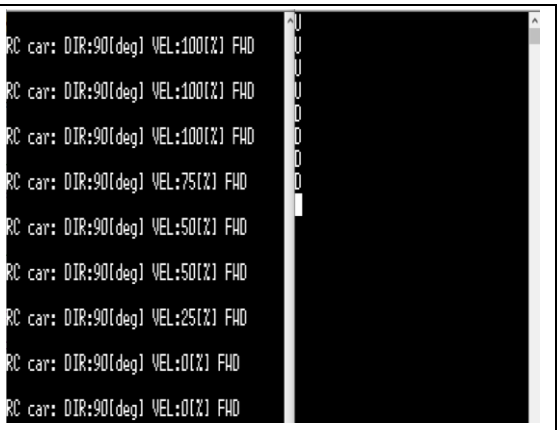
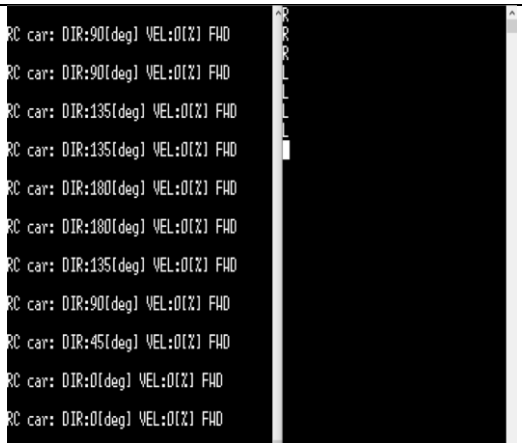
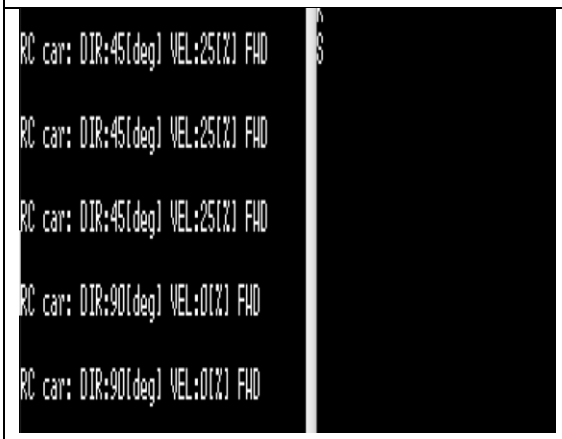
Q. Too many if-else phrases make the code very long.

A. Use the function and switch function

Q. Method of immediately responding to a user's input.

A. It executes the function in the USART handler

F. Result

<table>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>Press ‘U’</td><td>Press ‘D’</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>Press ‘L’</td><td>Press ‘B’, ‘F’</td></tr>
<tr><td> 
</td><td></td></tr>
<tr><td>Press ‘S’</td><td></td></tr>
</table>

- DEMO LINK (Partner name: Kim Ji Sung

https://youtu.be/pOYd79gK6P8

F. Appendix

- doucmentation

UART2_init()

Initialize only UART2. when we communicate with Tera-Term, this function is necessary.

```
void UART2_init();
```

Example code

```
UART2_init(); // Start UART2
```

USART_write()

this function for writing in tera-term.

```
void USART_write(USART_TypeDef * USARTx, uint8_t *buffer, uint32_t nBytes);
```

Parameters

- **USART_TypeDef * USARTx**: USART1, USART2 ...
- **buffer**: saved data
- **nBytes**: the number of Character

Example code

```
USART_write(USART1, &muc2Data, 1); //write down the muc2Data's value in tera-term
```

USART_delay()

We need waiting time so Use this function. This function works similar to delay_ms()

```
void USART_delay(uint32_t us);
```

Parameters

- **us**: delay time (Unit: usec)

Example code

```
USART_delay(300); delay 300 usec
```

USART_init()

in this funcition, Specify the pin according to Uart's number.

```
void USART_init(USART_TypeDef* USARTx, uint32_t baud);
```

Parameters

- *USART_TypeDef USARTx**: USART1, USART2 ...
- **baud**: communication rate //9600 ..

Example code

```
  USART_init(USART2, 38400); //ON USART2 and baud-rate 38400
```

USART_begin()

Turn on the USART communication. this function usually use in the USART_init()

```
void USART_begin(USART_TypeDef* USARTx, GPIO_TypeDef* GPIO_TX, int pinTX,int GPIO_TypeDef* GPIO_RX, int pinRX, int baud);
```

Parameters

- *USART_TypeDef USARTx**: USART1, USART2 ...
- *GPIO_TypeDef GPIO_TX**: GPIOA~GPIOH
- **pinTX**: 0~15
- *pinTX,GPIO_TypeDef GPIO_RX**: GPIOA~GPIOH
- **pinRX**: 0~15

  *Check the UASRAT Pin map

USART_getc()

Get the Character value by user

```
uint8_t USART_getc(USART_TypeDef *USARTx);
```

Parameters

- *USART_TypeDef USARTx**: USART1, USART2 ...

Example code

```
USART_getc(USART1); //get Usart1's value
```

is_USART_RXNE()

Make sure you are ready to read the value.

```
uint32_t is_USART_RXNE(USART_TypeDef * USARTx);
```

Parameters

- *USART_TypeDef USARTx**: USART1, USART2 ...

  Bit 5 **RXNE**: Read data register not empty

  This bit is set by hardware when the content of the RDR shift register has been transferred to the USART_DR register. An interrupt is generated if RXNEIE=1 in the USART_CR1 register. It is cleared by a read to the USART_DR register. The RXNE flag can also be cleared by writing a zero to it. This clearing sequence is recommended only for multibuffer communication.
  0: Data is not received
  1: Received data is ready to be read.

Example code

```
is_USART_RXNE(USART1);
```

- Pinmap

Type	Pin Mode	PinName	Arduino	Comments
USART2	AF(USART)	TX: PA_2 RX: PA_3	D1 D0	RCC_APB1
USART1		TX: PB_6, PA_9 RX: PB_3, PA_10	D10, D8 D3 ,D2	RCC_APB2
USART6		TX: PA_11, PC_6 RX: PA_12, PC_7	N/A	
SPI2	AF(USART)	SCLK: PC_7, PB_10, PB_13 MOSI: PB15 MISO: PB_14 SSEL: PB_12		RCC_APB1
SPI3		SCLK: PB_3 MOSI: PB_5 MISO: PB_4 SSEL: N/A		RCC_APB1 No SSEL
SPI1		SCLK: PA_5 MOSI: PA_7 MISO: PA_6 SSEL: N/A		LED1(PA_5) No SSEL RCC_APB2
SPI4		SCLK: N/A MOSI: N/A MISO: PA_11 SSEL: N/A		No SCLK No MOSI No SSEL RCC_APB2
SPI5		SCLK: N/A MOSI: PA_5, PB_8, PA_10 MISO: PA_12 SSEL: N/A		No SCLK No SSEL RCC_APB2