LAB: USART - LED,Bluetooth

Date: 2023-11-14

Author/Partner: GyeonhealAn (21900416)/ HanminKim(21900213)

[Github]

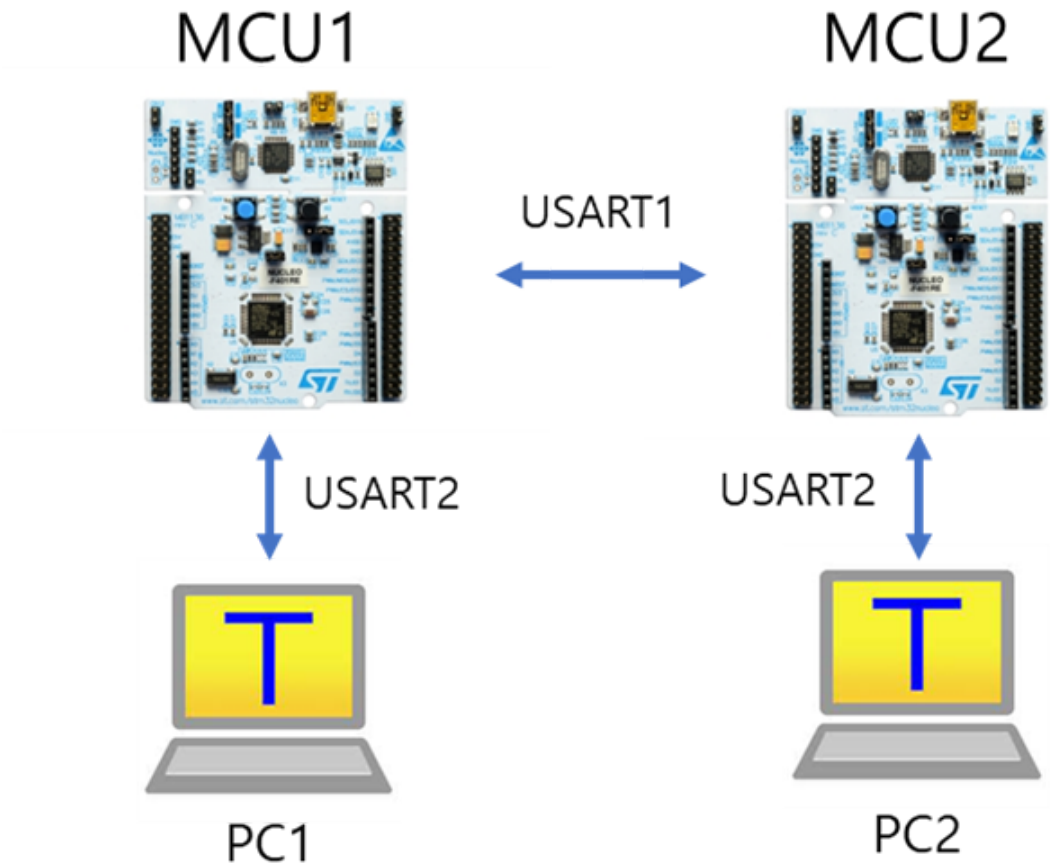
[Demo Video1]

[Demo Video2]

I. Introduction

In this lab, we will learn how to configure and use 'USART(Universal synchronous asynchronous receiver transmitter)' of MCU. Then, we will learn how to communicate between your PC and MCU and MCU to another MCU with wired serial communication.

- **Mission 1**: Control LED(LD2) of each other MCU.
- **Mission 2**: Run DC motors with Bluetooth



i. Requirement

Hardware

- MCU
  - NUCLEO-F411RE
- Actuator/Sensor/Others:

- DC motor, DC motor driver(L9110s),
- Bluetooth Module(HC-06),

Software

- Keil uVision, CLion(IDE), CMSIS, EC_HAL library

II. Problem 1: EC HAL library

i. Using HAL library

ecUSART.h

```
// Configuration UART 1, 2 using default pins
void UART1_init(void);
void UART2_init(void);
void UART1_baud(uint32_t baud);
void UART2_baud(uint32_t baud);
// USART write & read
void USART1_write(uint8_t* buffer, uint32_t nBytes);
void USART2_write(uint8_t* buffer, uint32_t nBytes);
uint8_t USART1_read(void);
uint8_t USART2_read(void);
// RX Interrupt Flag USART1,2
uint32_t is_USART1_RXNE(void);
uint32_t is_USART2_RXNE(void);
```

General USART Setup

```
// General Setting
void setup(){
  RCC_PLL_init();
  // BT serial : specific RX/TX pins
  USART_setting(USART1, GPIOA,9,GPIOA,10, BAUD_9600);   // PA9 - RXD , PA10 -
TXD
}
```

III. Problem 2: Communicate MCU1-MCU2 using RS-232

i. Procedure

1. Send a message from PC_1 by typing keys on Teraterm. It should send that message from MCU_1 to MCU_2.
2. The received message by MCU_2 should be displayed on PC_2.
3. Turn other MCU's LED(LD2) On/OFF by sending text:

- "**L**" for Turn OFF
- "**H**" for Turn ON

ii. Configuration

Type	Port - Pin	Configuration
System Clock		PLL 84MHz
USART2 : USB cable (ST-Link)		No Parity, 8-bit Data, 1-bit Stop bit, 38400 baud-rate
USART1 : MCU1 - MCU2	TXD: PA9 RXD: PA10	No Parity, 8-bit Data, 1-bit Stop bit, 38400 baud-rate
Digital Out: LD2	PA5	

iii. Code

```
/**
******************************************************************************
* @author   2023-10-31 by GH An
* @brief   Embedded Controller:  LAB - USART
*
******************************************************************************
*/

#include "stm32f4xx.h"
#include "ecGPIO.h"
#include "ecRCC.h"
#include "ecUART.h"
#include "ecSysTick.h"

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

void setup(void){
    RCC_PLL_init();

    // USB serial init
    UART2_init();
    UART2_baud(BAUD_38400);

    // BT serial init
    UART1_init();
    UART1_baud(BAUD_38400);

    //LED setup
    GPIO_init(GPIOA, LED_PIN, OUTPUT);    // calls RCC_GPIOA_enable()
}
void main(){
    setup();
    while(1){
    }
}
```

setup() function initializes system clock, UART1,2 LED Pin.

The clock set to 38400 which is declared in configuration.

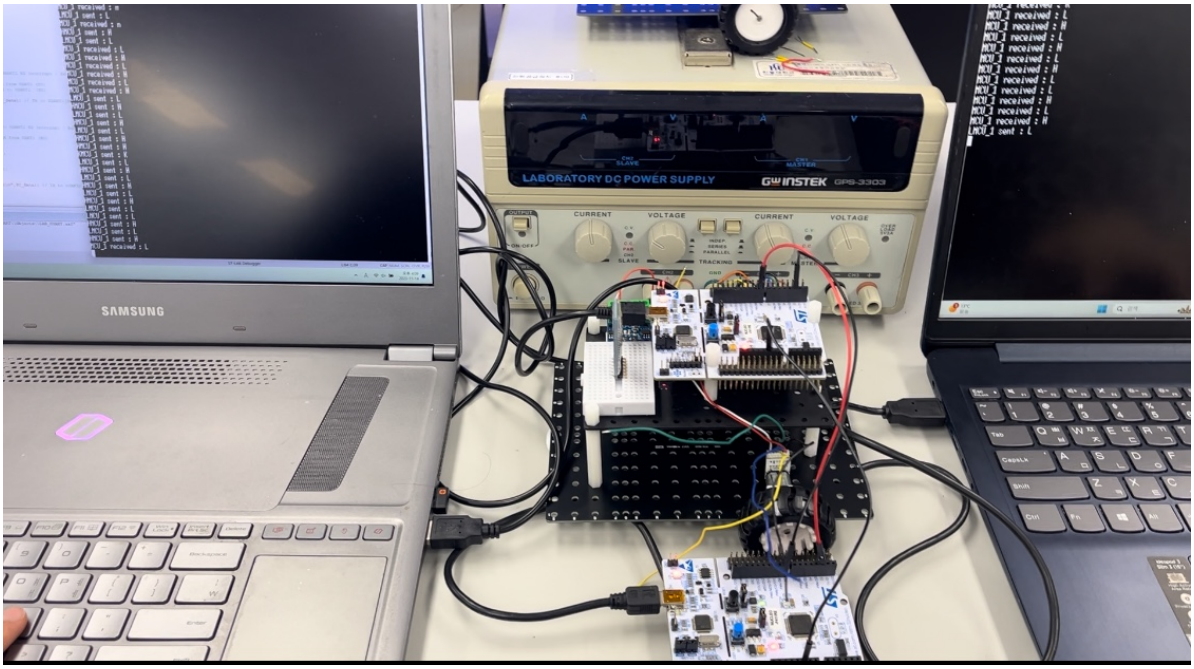
```
void USART2_IRQHandler(){                // USART2 RX Interrupt : Recommended
    if(is_USART2_RXNE()){
        PC_Data = USART2_read();         // RX from UART2 (PC)
        USART2_write(&PC_Data,1);        // TX to USART1  (BT)
        USART1_write(&PC_Data,1);
        printf("MCU_1 sent : %c \r\n",PC_Data); // TX to USART2(PC)
    }
}
```

This function reads PC data which user want to send and it sends to USART1 and itself.

```
void USART1_IRQHandler(){                // USART2 RX Interrupt : Recommended
    if(is_USART1_RXNE()){
            PC_Data = USART1_read();         // RX from UART1 (BT)
            if(PC_Data == 'L'){
                GPIO_write(GPIOA, LED_PIN, 0);
            }
            else if(PC_Data == 'H'){
                GPIO_write(GPIOA, LED_PIN, 1);
            }
      printf("MCU_1 received : %c \r\n",PC_Data); // TX to USART2(PC)
    }
}
```

If USART1 reads the data (L or H) from connected person's message, it turns on or off the LED and displays message on tera term

iv. Result

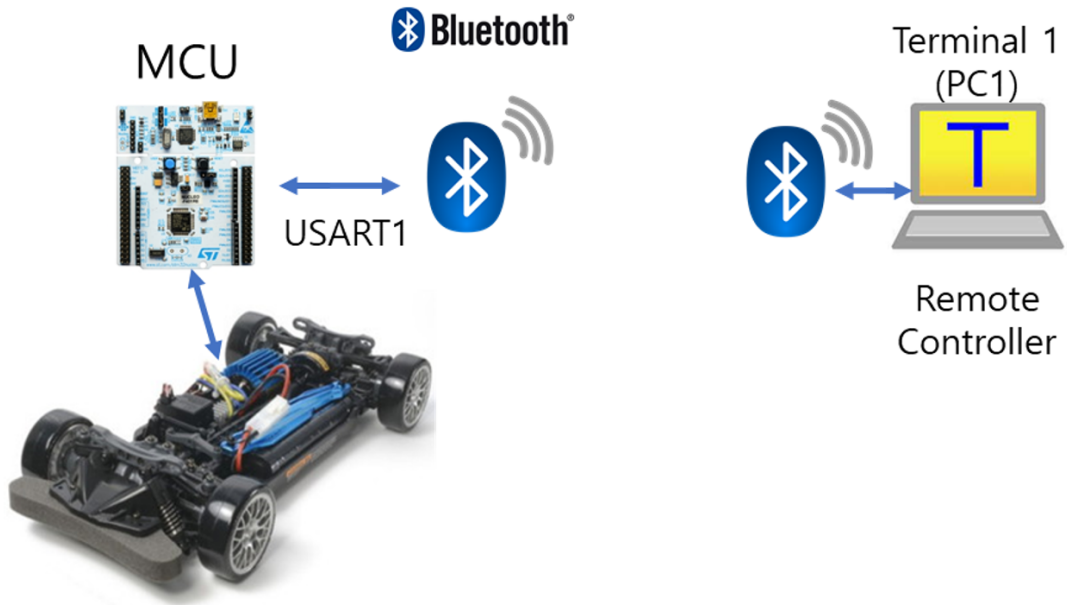


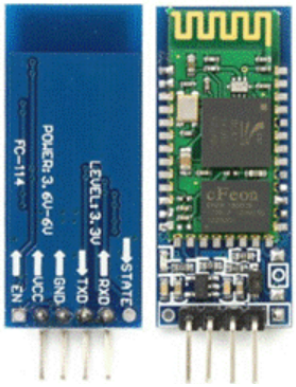
demo video link

IV. Problem 3: Control DC Motor via Bluetooth

i. Hardware Connection

Bluetooth





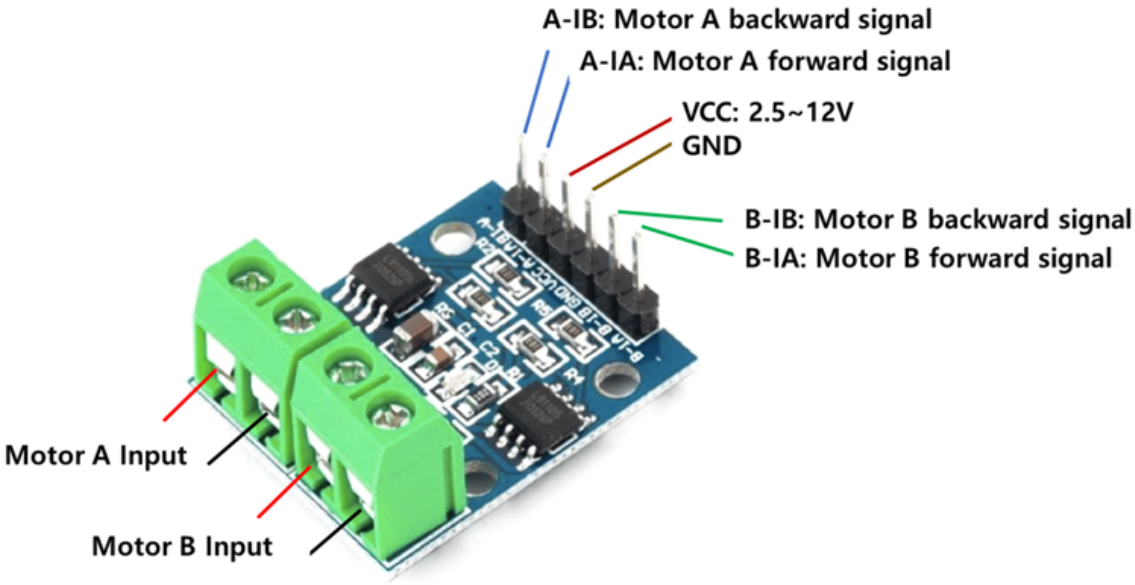
Bluetooth Module (HC-06)	STM32F411RE
RxD	PA_9(UART1_TX)
TxD	PA_10(UART1_RX)
GND	GND
VCC	5V

DC Motor Driver

Connect DC motor driver(L9110s) module pins to MCU as shown below.

DO NOT use MCU's VCC to motor driver. You should use external voltage source.

- A- IA: PWM pin (0~100% duty) for Motor A
- A- IB: Direction Pin (Digital Out H or L) for Motor B



ii. Procedure

1. Check the Bluetooth connection by turning MCU's LED(LD2) On/OFF by sending text of "**L0**" or "**L1**" from PC.
2. Run 2 DC motors(Left-wheel, Right-wheel) to steer.

- Turn Left: MotorA / MotorB = (50 / 80%) duty
- Turn Right: MotorA / MotorB = (80 / 50%) duty
- Go straight: MotorA / MotorB = (80 / 80 %) duty
- STOP: MotorA / MotorB = (0 / 0 %) duty

You may use the key inputs as your preference for Left, Right, Straight.

- I used the computer game command keys

- w or W: Go straight
- s or S: Stop
- a or A: Turn Left
- d or D: Turn Right

iii. Configuration

Type	Port - Pin	Configuration
System Clock		PLL 84MHz
USART1 : MCU - Bluetooth	TXD: PA9 RXD: PA10	No Parity, 8-bit Data, 1-bit Stop bit, 9600 baud-rate
Digital Out: LD2	PA5	
PWM (Motor A)	TIM2-Ch1	PWM period (2kHz~10kHz)
PWM (Motor B)	TIM2-Ch2	

iv. Code

```
/**
********************************************************************************
* @author   2023-10-31 by GH An
* @brief   Embedded Controller:  LAB - USART-Bluetooth
*
********************************************************************************
*/

#include "stm32f4xx.h"
#include "ecGPIO.h"
#include "ecRCC.h"
#include "ecUART.h"
#include "ecPWM.h"

#define DIR_PIN1 2
#define DIR_PIN2 3
PinName_t PWM_PIN1 = PA_0;
PinName_t PWM_PIN2 = PA_1;
float period = 500;

static volatile uint8_t PC_Data = 0;
static volatile uint8_t BT_Data = 0;
void setup(void);

void main(){
    setup();
    GPIO_write(GPIOC, DIR_PIN1, 1);
    GPIO_write(GPIOC, DIR_PIN2, 1);
    PWM_duty(PWM_PIN1, 1);
    PWM_duty(PWM_PIN2, 1);
    while(1){
    }
}
```

We set 500 period because we use micro sec PWM function.

If the period is 500, frequency is 1/(500x10^-6)=2kHz. It's inside the boundary we set in configuration.

We initialized the direction and duty to stop at initial time.

```
void USART1_IRQHandler(){                       // USART2 RX Interrupt :
Recommended
    if(is_USART1_RXNE()){
        BT_Data = USART1_read();
        USART1_write(&BT_Data,1);
        USART1_write("\r\n", 2);

        if(BT_Data == 's'){
            PWM_duty(PWM_PIN1, 1);
            PWM_duty(PWM_PIN2, 1);
        }
        else if(BT_Data == 'd'){
            PWM_duty(PWM_PIN1, 0.2);
            PWM_duty(PWM_PIN2, 0.5);
```

```
        }
        else if(BT_Data == 'a'){
            PWM_duty(PWM_PIN1, 0.5);
            PWM_duty(PWM_PIN2, 0.2);
        }
        else if(BT_Data == 'w'){
            PWM_duty(PWM_PIN1, 0);
            PWM_duty(PWM_PIN2, 0);
        }
        else if(BT_Data == 'L'){
            GPIO_write(GPIOA, LED_PIN, 0);
        }
        else if(BT_Data == 'H'){
            GPIO_write(GPIOA, LED_PIN, 1);
        }
    }
}
```

s=stop

d=go right (50%, 80%)

a=go left (80%, 50%)

w=go straight (80%, 80%)

L, H = LED on, off

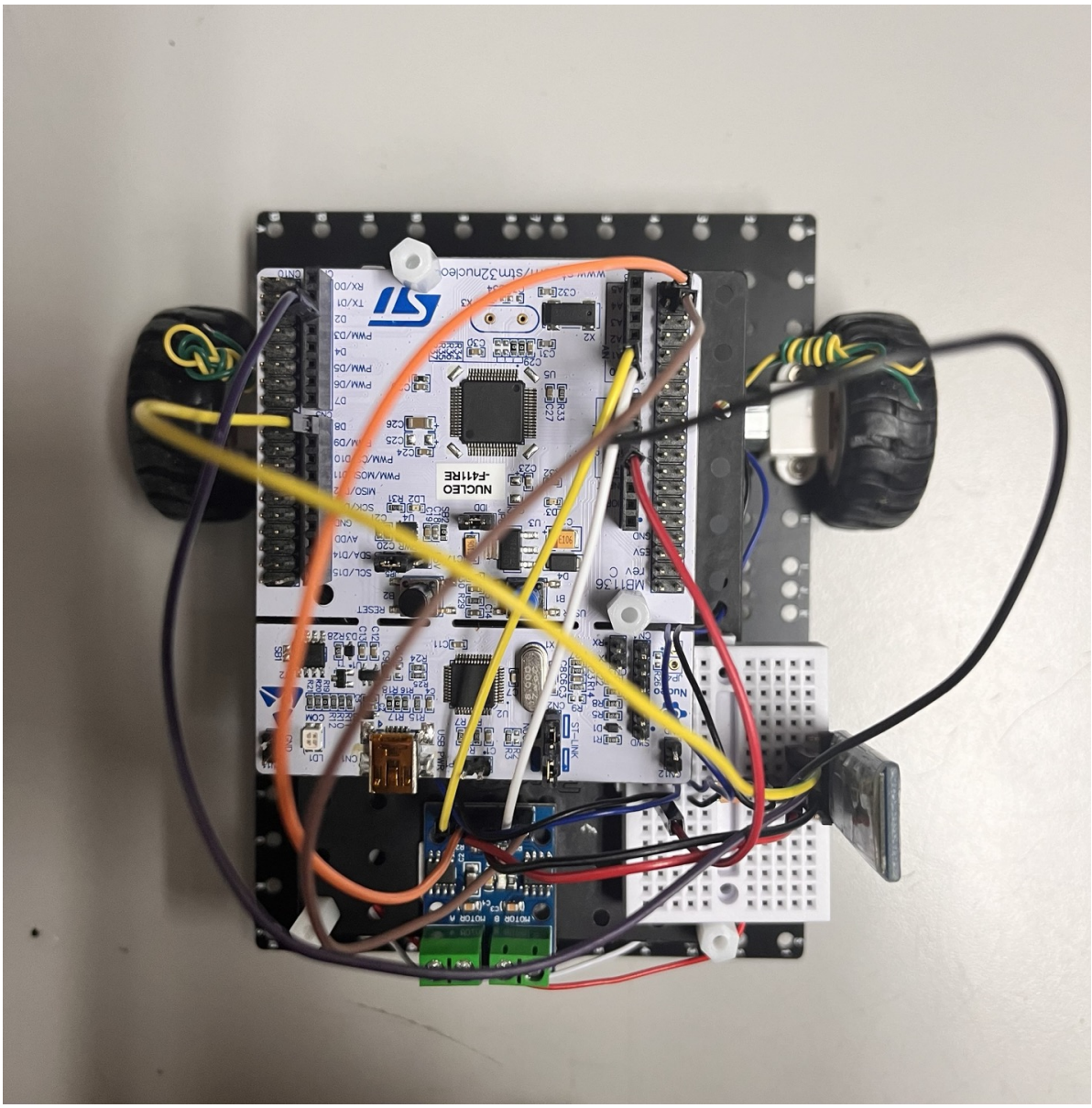
It sends BT_Data to bluetooth module and it displays the same value on tera term.

```
void setup(void){
    RCC_PLL_init();
    // LED
    GPIO(GPIOA, LED_PIN, OUTPUT, EC_MEDIUM, EC_PUSH_PULL, EC_NONE);
    // 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);
    // PWM1
    PWM_init(PWM_PIN1);
    PWM_period_us(PWM_PIN1, period);
    // PWM2
    PWM_init(PWM_PIN2);
    PWM_period_us(PWM_PIN2, period);
}
```

This is setup function which initializes pins.

We set 9600 clock because the bluetooth module works only in 9600 clock.

v. Result

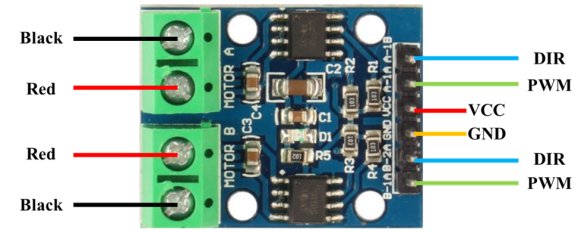


Demo Video

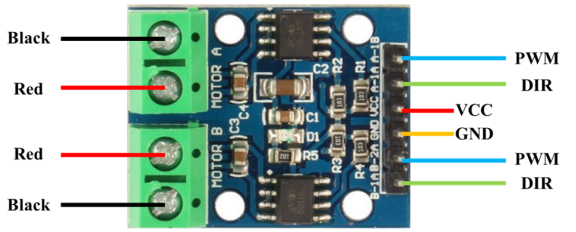
V. Reference

Prof. YoungKeun Kim Gitbook

VI. Appendix

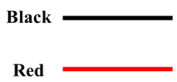
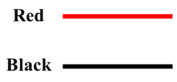


Case 1



Case 2

DC_motor_drive_case

Case	Motor	MCU	DIR	Duty ratio: 0⟶1	Direction
Case 1			LOW	Velocity ↑	CCR
			HIGH	Velocity ↓	CR
Case 2			LOW	Velocity ↑	CR
			HIGH	Velocity ↓	CCR

VII. Troubleshooting