**LAB: Wireless Communication Using Bluetooth**

**I. Overview**

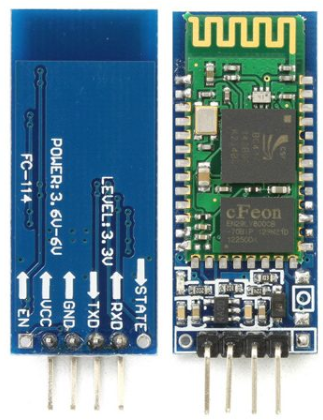
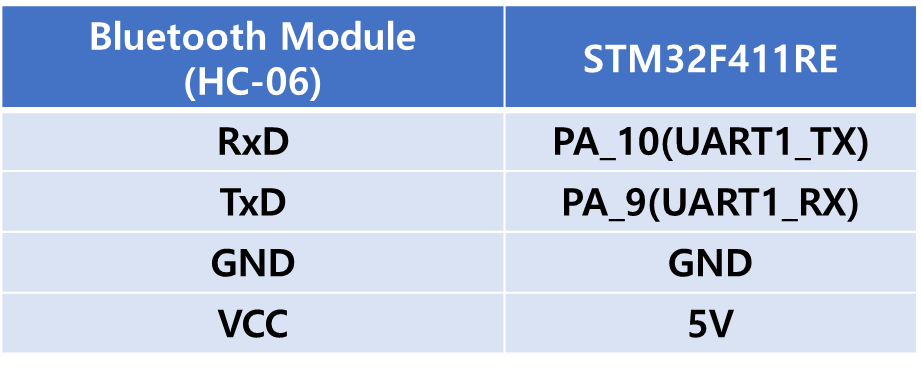
In this lab, we will learn how to configure and use ‘USART(Universal synchronous asynchronous receiver transmitter)’ of MCU. Then, we will learn how to communicate between your PC and MCU through serial monitor (Tera Term / Bluetooth module ).

The objectives of this lab are learning how to

* Configure registers of USART
* Create your own functions for configuring and controlling USART
* Use Bluetooth module to communicate wirelessly.

**Preparation**:

* Bluetooth Module, breadboard, Tera Term (Connect your PC with Serial and BT Module with Serial)
* You need to study the following registers: USART register in ‘STM Reference Manual pg. 505 - 557’’

 ****  
**Figure 1. Connection of a potentiometer**

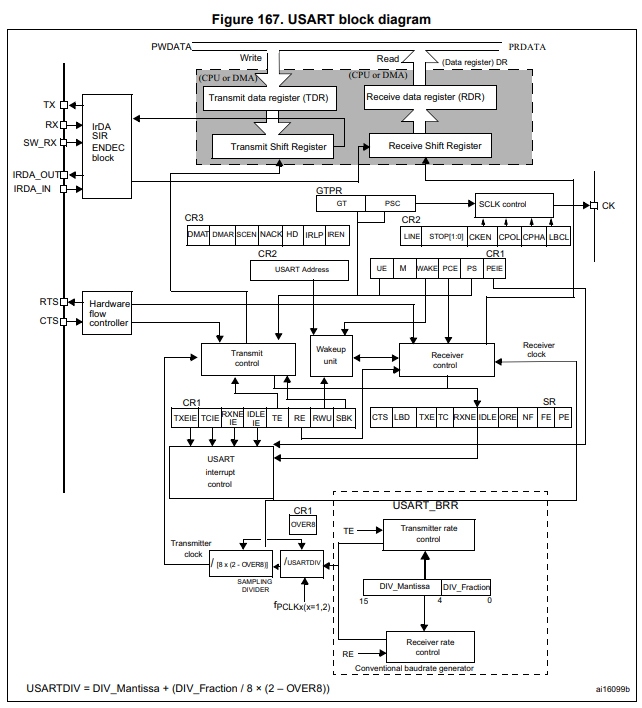
**II. Pre-Lab**

**A. USART Register**

* List of USARTx registers for this LAB

|  |  |  |
| --- | --- | --- |
| Type | Register Name | Description |
| RCC | RCC\_AHB1ENR | Enable the clock of GPIO Port |
| RCC\_APB2ENR | Enable the clock of USART |
| GPIO | GPIOx\_MODER | Alternative Function |
| GPIOx\_AFRL | Alternative Function Register |
| GPIOx\_PUPDR | No Pull-Up, No Pull-Down: |
| USART | USARTx\_CR1 | Select the parrity mode |
| Select the word length to 8 bit |
| Select the oversampling mode |
| USARTx\_CR2 | Choose the stop bit |
| USARTx\_CR3 | Set DMA enable transmitter and Receiver |
| USARTx\_BBR | Calculate baudrate and set the value. |
| USARTx\_SR | Read the status of USART pin |
| USARTx\_DR | Read the data from USART pin |

* USART block diagram

  
 **Figure 2. Connection of a potentiometer**

* Process of USART register initiation

|  |
| --- |
| **GPIO Pin setting for TXD and RXD pin.**  1. Enable GPIO peripheral clock (**AHB1ENR**)  2. Alternative Function mode selection for Pin\_y in GPIOx (**GPIO\_MODER**)  3. Set Alternative Function Register for USARTx. (**GPIO\_AFR[n]**)  4. Set the GPIO pin configuration as No pull up, No pull down (**GPIO\_PUPDR**)  **USARTx setting**   1. Enable USART peripheral clock (**RCC\_APBxENR)** 2. Disable USARTx. **(USARTx\_CR1)** 3. Configure as No Parity / 8-bit word length / Oversampling by 16 **(USARTx\_CR1)** 4. Set the stop bit **(USARTx\_CR2)** 5. Calculate USARTDIV and configure the register depending on the baud-rate **(USARTx\_BRR)** 6. Enable Transmitter, Receiver, and USARTx **(USARTx\_CR1)**   **Read USARTx Data (Interrupt)**   1. Set the priority of USARTx\_IQRn and enable interrupt.  (**NVIC\_Setpriority, NVIC\_EnableIRQ)** 2. Create user code in ‘void USART2\_IRQHandler ()’ to read the converted data from USART pin. |

**B. Register Setting**

**1. GPIO Pin Initialization**

* USART1 TX pin: Port A pin 9 / Alternative Function / No pull-up & No pull-down.
* USART1 RX pin: Port A pin 10 / Alternative Function / No pull-up & No pull-down.
* Configure alternative function register for USARTx. (Refer to the reference manual)
* AF7: USART1, USART2
* AF8: USART6
* Use your own functions created in previous lab for GPIO configuration.

**2. USART Initialization**

* USART1: No Parity / 8-bit word length / Oversampling by 16
* **RCC\_APB2ENR:** Enable USART1 peripheral clock

|  |
| --- |
| *Register map goes here* |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Register** |  |  |  |  |  |  |  |  |  |  |  | **20** | **19** | **18** | **17** | **16** |  | **14** | **13** | **12** | **11** |  |  | **8** |  |  | **5** | **4** |  |  |  | **0** |
| **Mask** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Value** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

* **USART\_ CR1:** Disable USART / No Parity / word length to 8 bit / Oversampling mode

|  |
| --- |
| *Register map goes here* |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Register** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Mask** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Value** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

* **USART\_ CR2:** 1 stop bit

|  |
| --- |
| *Register map goes here* |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Register** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Mask** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Value** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

* **USART\_BRR:** 9600 [bps]

|  |
| --- |
| *Register map goes here* |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Register** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Mask** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Value** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

* **USART\_ CR1:** Enable Transmitter, Receiver, and USART

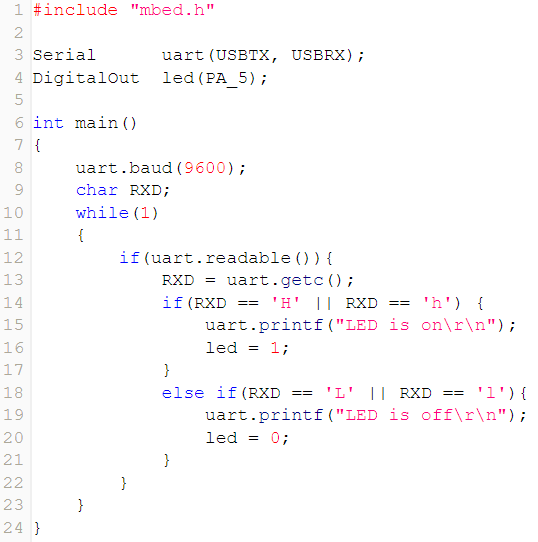
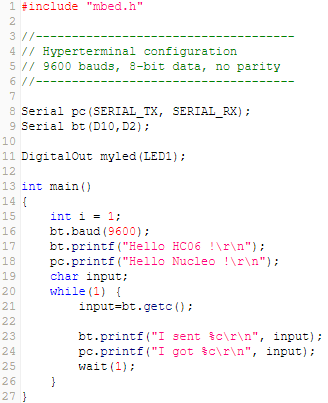
|  |
| --- |
| *Register map goes here* |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Register** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Mask** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Value** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**III. Tutorial**

**A. Mbed online compiler**

* Open <https://www.mbed.com/en/> and create new program ‘Tutorial\_USART’
* Open ‘main.cpp’ and delete the example codes.
* Write the following code on ‘main.cpp’.

* Click on ‘**Compile’** button. Then, the binary file will be created and downloaded on your computer.
* Connect the MCU board to your PC. Copy and paste the downloaded binary file to the drive “NODE\_F411RE (E:)”.
* Open ‘Tera Term’ and verify the performance. Check the output voltage of photo sensor.

**B. Pairing Bluetooth MCU-PC**

[MCU]

* Connect Bluetooth module pins to MCU. This will supply power to the BT module.

[PC]

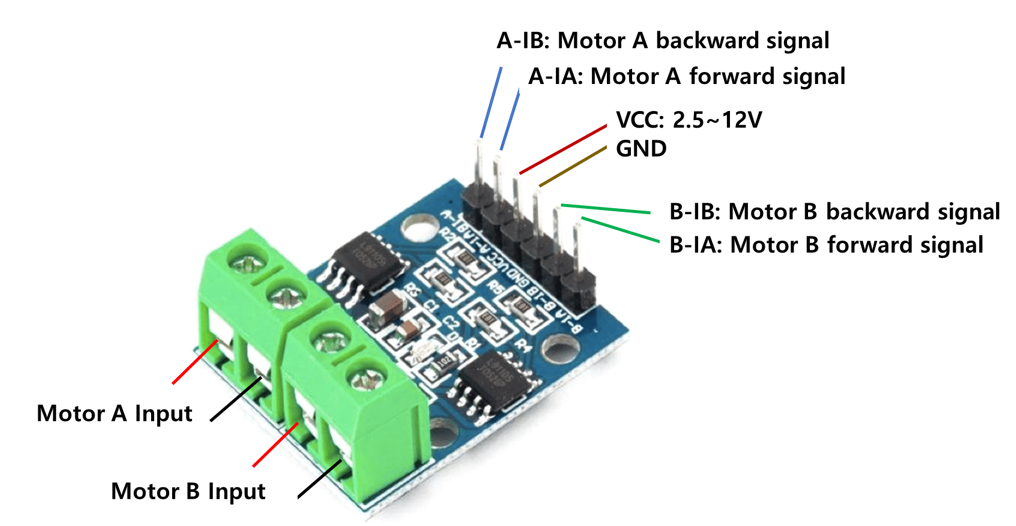
* Open your laptop setting 🡪 [Bluetooth 및 기타 디바이스] and add your Bluetooth module (HC-06)
* PIN number : 1234

|  |  |
| --- | --- |
|  |  |

**C. Connecting DC motor drive with MCU**

[MCU-DC motor Drive]

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



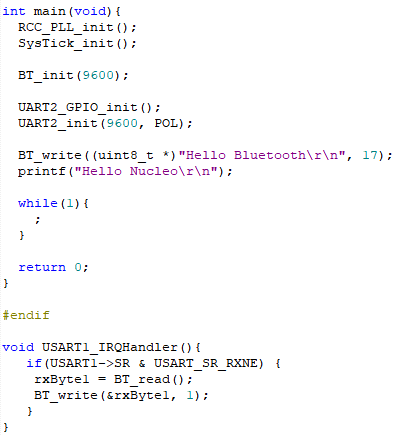
**IV. Exercise**

1. **Create your own functions for USART**

* Create your own function to initialize USART
* Below are some of the examples of necessary functions. Attach your codes in Appendix.
* If necessary, you can also create other functions to configure timer input capture mode.

|  |  |  |
| --- | --- | --- |
| **File** | **Example Code** | **Description** |
| myUART.h, myUART.c | void USART\_init(USART\_TypeDef \*USARTx, uint32\_t baud\_rate, uint32\_t mode) | // Default Setting No parity  M: 8 data bits / 1 stop bit oversampling by 16 Mode: interrupt |
| *Other configuration functions* | *Other configuration functions* |
| uint8\_t USART\_read (USART\_TypeDef \* USARTx) | Read the data register of USARTx |
| void USART\_write(USART\_TypeDef \* USARTx, uint8\_t \*buffer, uint32\_t nBytes) | Write the data register of USARTx |
| void BT\_init(uint32\_t baud\_rate) | // Default Setting for BT module  USART1, No parity  Data 8 data bits / 1 stop bit oversampling by 16 Mode: interrupt |
| uint8\_t BT\_read ()  void BT\_write(uint8\_t \*buffer, uint32\_t nBytes) | Read/write USART1 for BT |

Sample code:



1. **Create a simple program to communicate between STM32F411RE board and PC via Bluetooth**
2. Configure PC-MCU serial monitor communication using USART2

* Configure USART2 receive mode as interrupt mode
* Configure: Odd Parity, 8-bit Data, 1-bit Stop bit, 38400 baud

1. Bluetooth pairing between PC and MCU. See tutorial.

* Configure MCU USART1 as the Bluetooth TXD/RXD: PA9 and PA10, respectively
* Configure USART1 receive mode as interrupt mode
* Configure: No Parity, 8-bit Data, 1-bit Stop bit, 9600 baud

1. Transmit a character at a time from the PC to MCU

* MCU should receive the character data by interrupt method
* Display the received character by writing it on Serial Monitor(USART2)

|  |  |
| --- | --- |
| (PC- MCU BT UART1) | (PC- MCU UART2) |

1. **Create a program to run 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 interrupt

|  |  |  |
| --- | --- | --- |
| Key board | Task | Comment |
| UP | Motor Speed Increase | Increase PWM duty ratio(Right/Left motor) by 10% for each press.  Initial value : duty 0% |
| DOWN | Motor Speed Decrease | Decrease PWM duty ratio(Right/Left motor) by 10% for each press |
| LEFT | Left Motor ON | Turn on left motor only |
| RIGHT | Right Motor ON | Turn on right motor only |
| F | Forward motor direction | Both motor runs to go forward. Default setting |
| B | Backward motor direction | Reverse driving |
| S | Stop | Make duty=0% for both motors |

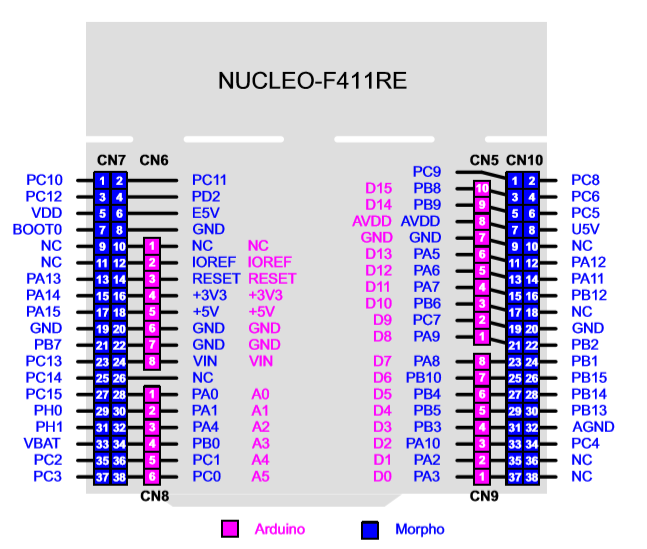
* Motor PWM Setting:
  + Initial: TIM1. PWM Frequency= 1kHz, Duty Ratio= 10%
  + Use Channel 1, Channel 2 for Left motor and Right motor, respectively

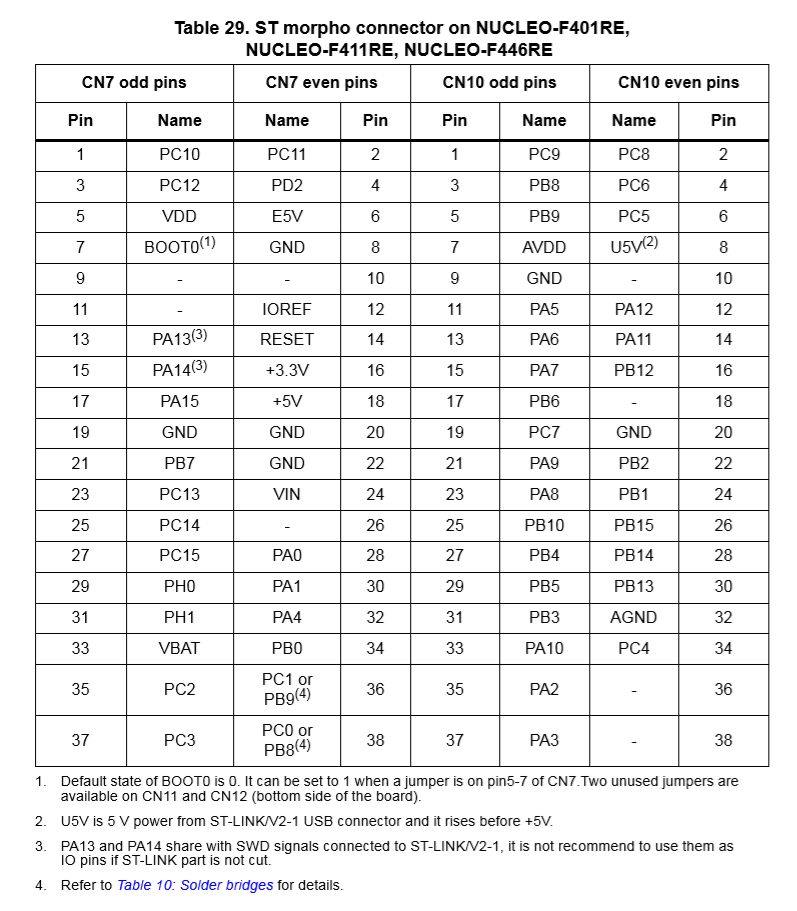
**V. Questions**

1. Using an oscilloscope or digital analyzer to capture the signal on the USART\_TX pin of STM32F4. What is the bit rate? Is it the same the baud rate?
2. Explain your reason. Why DMA or interrupts are highly recommended for USART?
3. If the parity bit is used in the data frame, how to detect transmission errors?

**Appendix**

1. **Pin Configuration of NUCLE-F411RE**

****

****

1. **Arduino Code Example**

|  |  |
| --- | --- |
| **#define MOTOR\_A\_a 3 //define + output pin of motor A as 3**  **#define MOTOR\_A\_b 11 // define - output pin of motor A as 11**  **#define MOTOR\_B\_a 5 // define + output pin of motor B as 5**  **#define MOTOR\_B\_b 6 // define - output pin of motor B as 6**  **#define MOTOR\_SPEED 150 // motor speed (0~255)**  **unsigned char m\_a\_spd = 0, m\_b\_spd = 0;**  **boolean m\_a\_dir = 0, m\_b\_dir = 0;**  **void setup()**  **{**  **pinMode(MOTOR\_A\_a, OUTPUT);**  **pinMode(MOTOR\_A\_b, OUTPUT);**  **pinMode(MOTOR\_B\_a, OUTPUT);**  **pinMode(MOTOR\_B\_b, OUTPUT);**  **Serial.begin(9600);**  **Serial.println("Hello!");**  **}**  **void loop()**  **{**  **unsigned char bt\_cmd = 0;**  **if (Serial.available())**  **{**  **bt\_cmd = Serial.read();**  **rc\_ctrl\_val(bt\_cmd);**  **}**  **motor\_drive();**  **}**  **void rc\_ctrl\_val(unsigned char cmd)**  **{**  **if(cmd == 'w') // Forward direction**  **{**  **m\_a\_dir = 0; // motor A right direction**  **m\_b\_dir = 0; // motor B right direction**  **m\_a\_spd = MOTOR\_SPEED; // motor A speed control**  **m\_b\_spd = MOTOR\_SPEED; // motor B speed control**  **}**  **else if(cmd == 'a') //Backward direction**  **{**  **m\_a\_dir = 1; // motor A inverse direction**  **m\_b\_dir = 0; // motor B right direction**  **m\_a\_spd = MOTOR\_SPEED; // motor A speed control**  **m\_b\_spd = MOTOR\_SPEED; // motor B speed control** | **}**  **else if(cmd == 'd') // Turn right**  **{**  **m\_a\_dir = 0; // motor A right direction**  **m\_b\_dir = 1; // motor B inverse direction**  **m\_a\_spd = MOTOR\_SPEED; // motor A speed control**  **m\_b\_spd = MOTOR\_SPEED; // motor B speed control**  **}**  **else if(cmd == 's') // Turn left**  **{**  **m\_a\_dir = 1; // motor A inverse direction**  **m\_b\_dir = 1; // motor B inverse direction**  **m\_a\_spd = MOTOR\_SPEED; // motor A speed control**  **m\_b\_spd = MOTOR\_SPEED; // motor B speed control**  **}**  **else if(cmd == 'x')**  **{**  **m\_a\_dir = 0; // motor A right direction**  **m\_b\_dir = 0; // motor B right direction**  **m\_a\_spd = 0; // motor A stop**  **m\_b\_spd = 0; // motor B stop**  **}**  **}**  **void motor\_drive()**  **{**  **if(m\_a\_dir == 0)**  **{**  **digitalWrite(MOTOR\_A\_a, LOW); //motor A+ LOW**  **analogWrite(MOTOR\_A\_b, m\_a\_spd); //motor A-speed PWM output**  **}**  **else**  **{**  **analogWrite(MOTOR\_A\_a, m\_a\_spd); // motor A+ speed PWM output**  **digitalWrite(MOTOR\_A\_b, LOW); // motor A- LOW**  **}**  **if(m\_b\_dir == 1)**  **{**  **digitalWrite(MOTOR\_B\_a, LOW); // motor B+ LOW**  **analogWrite(MOTOR\_B\_b, m\_b\_spd); // motor B- speed PWM output**  **}**  **else**  **{**  **analogWrite(MOTOR\_B\_a, m\_b\_spd); // motor B+ speed PWM output**  **digitalWrite(MOTOR\_B\_b, LOW); // motor B- LOW**  **}**  **}** |