<u>การทดลองที่ 7</u> การใช้งาน PWM และการควบคุมมอเตอร์

วัตถุประสงค์

1) เข้าใจการทำงานของ Pulse-Width Modulation
2) สามารถเขียนโปรแกรมควบคุมการทำงานของ Timer เพื่อสร้างสัญญาณ Pulse-Width Modulation

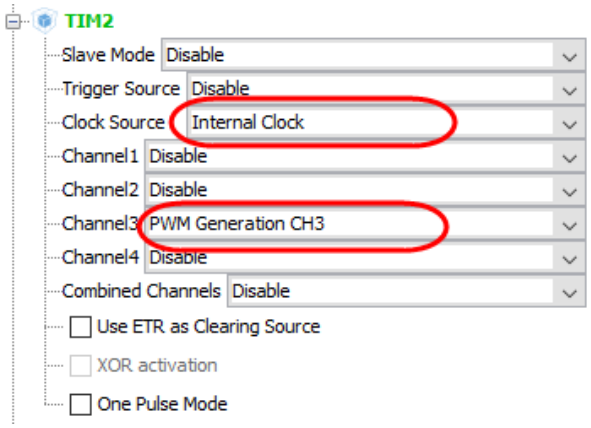
1. Pulse-Width Modulation Generation

วงจร Timer ภายในไมโครคอนโทรลเลอร์ STM32F107 จะมีโหมดการทำงานเพื่อสร้างสัญญาณ PWM สำหรับส่งออกไปควบคุมอุปกรณ์ภายนอก เช่น LED หรือมอเตอร์ได้ โดยความถี่ของสัญญาณถูกควบคุมโดยรีจิสเตอร์ TIMx_ARR ส่วน Duty Cycle ถูกควบคุมโดยรีจิสเตอร์ TIMx_CCRx วงจร Timer 1 โมดูลประกอบไปด้วยช่องสัญญาณจำนวน 4 ช่องสัญญาณ ดังนั้นจึงสามารถสร้างสัญญาณ PWM จำนวน 4 สัญญาณได้จาก Timer 1 โมดูล โดยจะใช้การเปรียบเทียบระหว่าง counter ของ Timer กับรีจิสเตอร์ TIMx_CCRx เพื่อสร้างสัญญาณ PWM ด้วยกัน 2 โหมด ดังนี้

1. **โหมด1** ในการนับขึ้นจะสร้างสัญญาณ PWM สถานะ SET เมื่อ counter ของโมดูล Timer มีค่าน้อยกว่าค่าในรีจิสเตอร์ TIMx_CCRx และเมื่อ counter มีค่ามากกว่ารีจิสเตอร์ดังกล่าวสัญญาณ PWM จะมีสถานะ RESET
2. **โหมด2** ในการนับลงจะสร้างสัญญาณ PWM สถานะ SET เมื่อ counter ของโมดูล Timer มีค่ามากกว่าค่าในรีจิสเตอร์ TIMx_CCRx และเมื่อ counter มีค่าน้อยกว่ารีจิสเตอร์ดังกล่าวสัญญาณ PWM จะมีสถานะ RESET

2. การตั้งค่าในโปรแกรม STM32CubeMX

หากต้องการใช้งานช่องสัญญาณ 3 ของ TIM2 เพื่อสร้างสัญญาณ PWM สามารถตั้งค่า TIM2 ได้ดังรูปที่ 2.1 จากนั้นโปรแกรมจะกำหนดให้ค่า PA2 ทำหน้าที่จ่ายสัญญาณ PWM ดังรูปที่ 2.2 (a) โดยเปลี่ยนให้ขา PA2 ทำหน้าที่ Alternate function แบบ Default ทั้งนี้สามารถเลือกให้ขา PB10 ทำหน้าที่จ่ายสัญญาณ PWM ของ TIM2_CH3 แทนได้ ซึ่งต้องเปลี่ยนให้ PB10 ทำหน้าที่ Alternate function แบบ Remap ดังรูปที่ 2.2 (b)

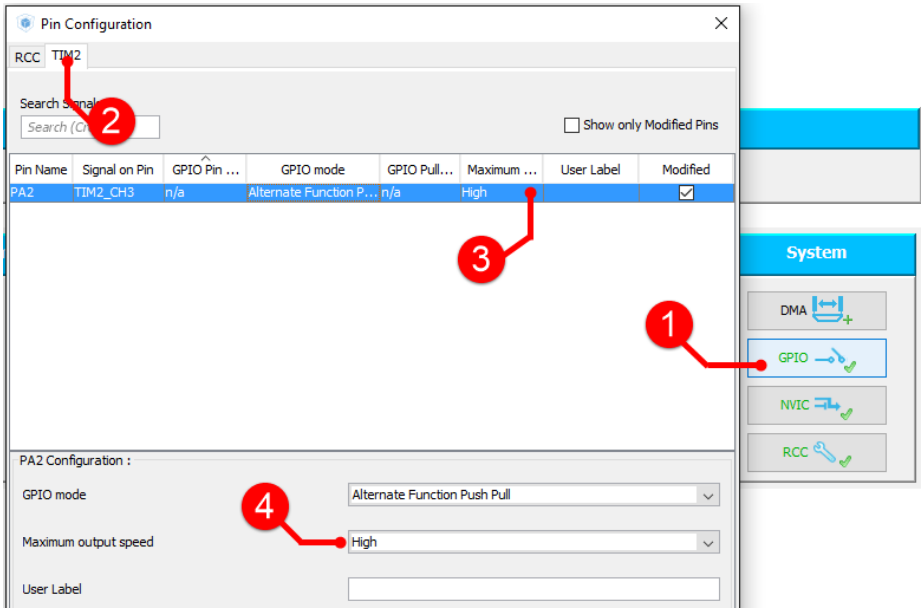


รูปที่ 2.1 แสดงการตั้งค่าให้โมดูล TIM2 Channel 3 จ่ายสัญญาณ PWM

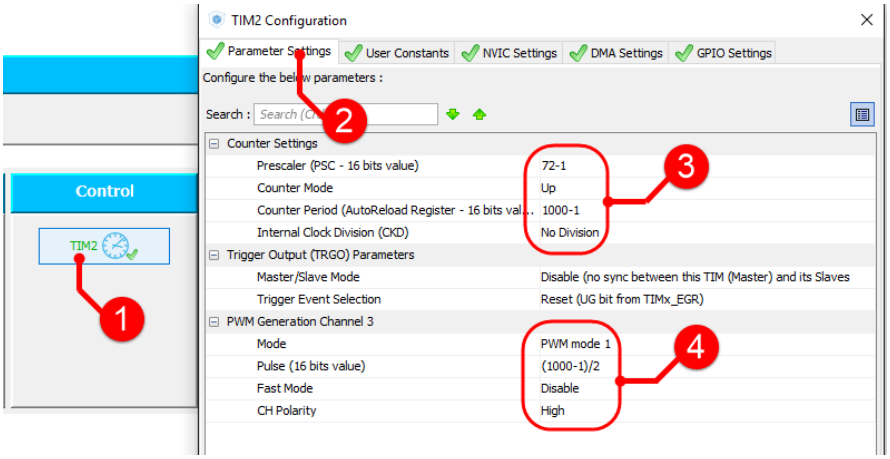
(a) (b)

รูปที่ 2.2 แสดงขาเอาต์พุตที่สามารถนำสัญญาณ PWM จาก TIM2_CH3 ไปใช้งานได้

จากนั้นตั้งค่าให้ขา PA2 ทำงานที่ความเร็วสูงสุดดังรูปที่ 2.3 และตั้งค่าโมดูล TIM2 ได้ดังรูปที่ 2.4 ซึ่งตั้งค่าให้สัญญาณ PWM มีคาบเวลา 1 ms และมี Duty Cycle 50%



รูปที่ 2.3 แสดงการตั้งค่าขา PA2



รูปที่ 2.4 แสดงการตั้งค่าคาบเวลาและ Duty Cycle ของ TIM2_CH3

3. อธิบายการตั้งค่า

โปรแกรม STM32CubeMX ตั้งค่า TIM2 เพื่อสร้างสัญญาณ PWM ด้วยฟังก์ชัน `MX_TIM2_Init` และฟังก์ชัน `MX_GPIO_Init` ในไฟล์ `main.c` และฟังก์ชัน `HAL_TIM_MspPostInit` ในไฟล์ `stm32f1xx_hal_msp.c` ดังรูปที่ 3.1 และรูปที่ 3.2

ฟังก์ชัน **`MX_TIM2_Init`**

- เป็นฟังก์ชันที่โปรแกรม STM32CubeMX สร้างขึ้นมา เพื่อตั้งค่าคาบเวลาและ Duty Cycle ของสัญญาณ PWM บนไมโครคอนโทรลเลอร์ให้สอดคล้องกับที่กำหนดไว้ในโปรแกรม
- เริ่มต้นด้วยการประกาศตัวแปร Global htim2 สำหรับตั้งค่า TIM2 ที่ส่วนต้นของไฟล์ `main.c`

```
TIM_HandleTypeDef htim2;
```

- ส่วนฟังก์ชันเริ่มต้นด้วยการประกาศตัวแปร สำหรับตั้งค่าภายในโมดูล Timer

```
TIM_ClockConfigTypeDef sClockSourceConfig;
TIM_MasterConfigTypeDef sMasterConfig;
```

- จากนั้นตั้งค่าคาบเวลาของสัญญาณ PWM โดยใช้ Clock Division, Prescaler และ Period เหมือนกับการทดลองที่ 6

```
htim1.Init.ClockDivision  = TIM_CLOCKDIVISION_DIV1;
htim1.Init.Prescaler      = 72-1;
htim1.Init.Period         = 1000-1;
```

- แล้วจึงตั้งค่าให้ TIM2 นับขึ้น

```
htim1.Init.CounterMode    = TIM_COUNTERMODE_UP;
```

- จากนั้นตั้งค่าให้สัญญาณ PWM แบบ Active High มี Duty Cycle 50% ให้กับช่องสัญญาณที่ 3

```
sConfigOC.OCMode          = TIM_OCMODE_PWM1;
sConfigOC.Pulse           = (1000-1)/2;
HAL_TIM_PWM_ConfigChannel(&htim2, &sConfigOC, TIM_CHANNEL_3);
```

```
/* TIM2 init function */
void MX_TIM2_Init(void)
{

  TIM_ClockConfigTypeDef sClockSourceConfig;
  TIM_MasterConfigTypeDef sMasterConfig;
  TIM_OC_InitTypeDef sConfigOC;

  htim2.Instance = TIM2;
  htim2.Init.Prescaler = 72-1;
  htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim2.Init.Period = 1000-1;
  htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  HAL_TIM_Base_Init(&htim2);

  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig);

  HAL_TIM_PWM_Init(&htim2);

  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig);

  sConfigOC.OCMode = TIM_OCMODE_PWM1;
  sConfigOC.Pulse = (1000-1)/2;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  HAL_TIM_PWM_ConfigChannel(&htim2, &sConfigOC, TIM_CHANNEL_3);

  HAL_TIM_MspPostInit(&htim2);

}
```

รูปที่ 3.1 แสดงการตั้งค่า TIM2 ในฟังก์ชัน `MX_TIM2_Init()`

ฟังก์ชัน `MX_GPIO_Init`

- เป็นฟังก์ชันที่โปรแกรม STM32CubeMX สร้างขึ้นมา เพื่อตั้งค่า GPIO โดยจ่ายสัญญาณนาฬิกาให้กับ GPIO พอร์ต A

ฟังก์ชัน `HAL_TIM_MspPostInit`

- เป็นฟังก์ชันที่โปรแกรม STM32CubeMX สร้างขึ้นมาในไฟล์ `stm32f1xx_hal_msp.c` เพื่อเปลี่ยนการทำงานของขา PA2 จาก GPIO เป็น Alternate function แบบ Default

```c
void MX_GPIO_Init(void)
{

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOA_CLK_ENABLE();

}
```

(a)

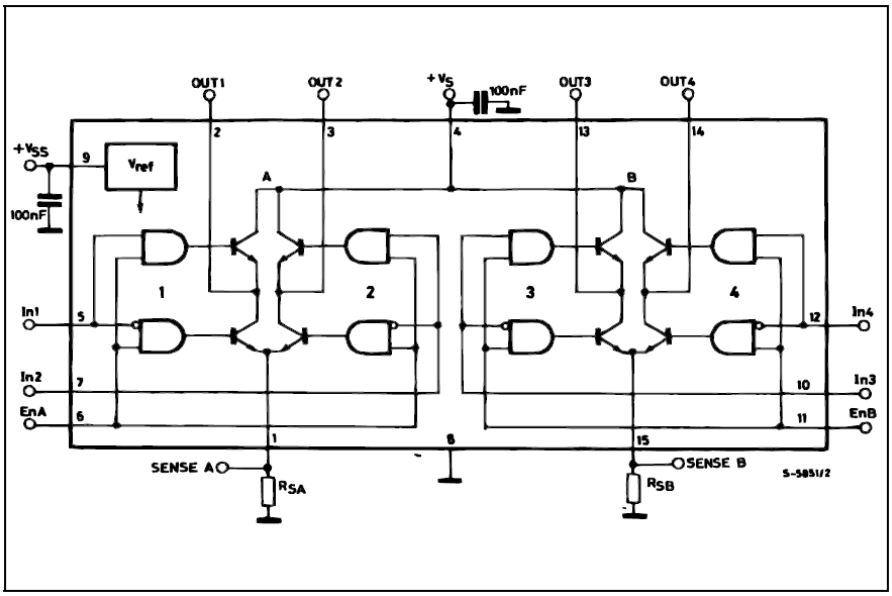
```c
void HAL_TIM_MspPostInit(TIM_HandleTypeDef* htim)
{

  GPIO_InitTypeDef GPIO_InitStruct;
  if(htim->Instance==TIM2)
  {
  /* USER CODE BEGIN TIM2_MspPostInit 0 */

  /* USER CODE END TIM2_MspPostInit 0 */

    /**TIM2 GPIO Configuration
    PA2     ------> TIM2_CH3
    */
    GPIO_InitStruct.Pin = GPIO_PIN_2;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Speed = GPIO_SPEED_HIGH;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /* USER CODE BEGIN TIM2_MspPostInit 1 */

  /* USER CODE END TIM2_MspPostInit 1 */
  }

}
```

(b)

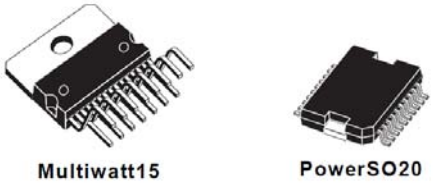
รูปที่ 3.2 แสดงการตั้งค่า PA2 ให้จ่ายสัญญาณ PWM จาก TIM2_CH3

4. การควบคุมมอเตอร์ด้วยสัญญาณ PWM

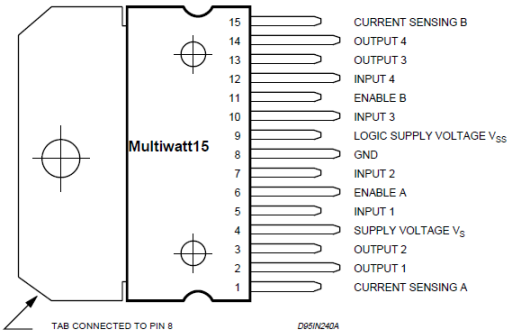
เนื่องจากมอเตอร์เป็นอุปกรณ์ที่ใช้กระแสไฟฟ้าเกินกว่าที่พิกัดที่ไมโครคอนโทรลเลอร์จ่ายให้ได้ หากเชื่อมต่อขาเอาต์พุตไปยังมอเตอร์โดยตรงจะส่งผลให้วงจรที่ขับเอาต์พุต (output driver) เสียหาย ดังนั้นหากต้องการควบคุมมอเตอร์โดยใช้ไมโครคอนโทรลเลอร์จำเป็นต้องสร้างวงจรควบคุมมอเตอร์ขึ้นมาเพื่อทำหน้าที่จ่ายกระแสไฟฟ้าให้กับมอเตอร์ เช่น วงจร H-Bridge ที่สร้างจากทรานซิสเตอร์ 4 ตัว เป็นต้น แต่เพื่อความสะดวกสามารถใช้ไอซีสำหรับควบคุมมอเตอร์แทนการต่อวงจรได้ เช่น ไอซีหมายเลข L298 ซึ่งมีวงจร full bridge จำนวน 2 วงจร รองรับแหล่งจ่ายไฟได้สูงสุด 46 V และจ่ายกระแสได้สูงสุดรวม 4 A โดย block diagram ของ L298 แสดงได้ดังรูปที่ 4.1 ไอซี L298 มี package 2 แบบ ได้แก่ แบบ Multiwatt15 และ PowerSO20 ดังรูปที่ 4.2 ตำแหน่งและการทำงานของแต่ละขาของ L298 แสดงได้ดังรูปที่ 4.3 และตารางที่ 4.1 ตามลำดับ



รูปที่ 4.1 แสดง Block Diagram ของ L298



รูปที่ 4.2 แสดง IC Package ของ L298



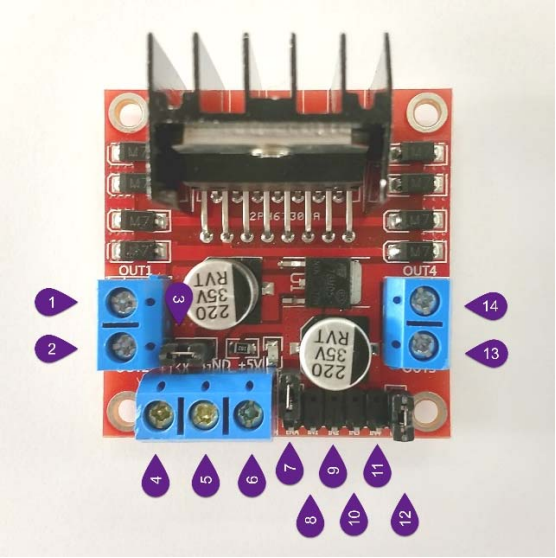
รูปที่ 4.3 แสดงตำแหน่งขาของ L298 แบบ Multiwatt15

ซึ่งภายในมีทรานซิสเตอร์จำนวน 4 ตัว สามารถเลือกใช้เพื่อควบคุมมอเตอร์ให้หมุนทิศทางเดียวหรือสองทิศทางได้ โดยไอซี L293B ไม่มีไดโอดสำหรับป้องกัน (flyback diode) มาให้จึงต้องต่อไดโอดป้องกันเพิ่มเติม หากเลือกใช้ L293D ที่ภายในมีไดโอดอยู่แล้วก็ไม่จำเป็นต้องต่อเพิ่มเติม ดัง**Error! Reference source not found.** และแสดงรายละเอียดของขาสัญญาณต่างๆ ของไอซี L293D ได้ดังตารางที่ 4.1

ตารางที่ 4.1 รายละเอียดของขาสัญญาณของ L293D

สัญญาณ	หมายเลขขา	หน้าที่
Sense A; Sense B	2; 19	ต่อตัวต้านทานระหว่างขานี้กับกราวนด์เพื่อควบคุมกระแส
Out 1; Out 2	4; 5	เอาต์พุตของ bridge A โดยกระแสจะไหลระหว่าง 2 ขานี้
V_S	6	แหล่งจ่ายไฟภายนอก
Input 1; Input 2	7; 9	สัญญาณอินพุตแบบ TTL เพื่อควบคุมทิศทาง สำหรับ bridge A
Enable 1; Enable 2	8; 14	สัญญาณอินพุตแบบ TTL เพื่อควบคุมการทำงานของ bridge A และ B
GND	1, 10, 11, 20	กราวนด์
V_{SS}	12	แหล่งจ่ายไฟสำหรับอ้างอิงสัญญาณลอจิก
Input 3; Input 4	13; 15	สัญญาณอินพุตแบบ TTL เพื่อควบคุมทิศทาง สำหรับ bridge B
Out 3; Out 4	16; 17	เอาต์พุตของ bridge B โดยกระแสจะไหลระหว่าง 2 ขานี้
N.C.	3; 18	ไม่ถูกเชื่อมต่อ (Not Connected)

ทั้งนี้เพื่อความสะดวกในการใช้งานจึงได้มีการผลิตโมดูลไดรฟ์มอเตอร์ที่สร้างจากไอซี L298 ดังรูปที่ 4.4 โดยมีรายละเอียดของคอนเนคเตอร์แต่ละจุดดังตารางที่ 4.2



รูปที่ 4.4 แสดงรายละเอียดไอซี L293D

ตารางที่ 4.2 รายละเอียดของคอนเนคเตอร์โมดูล L298

หมายเลขคอนเนคเตอร์	หน้าที่	สำหรับการทดลองข้อ 4
1; 2	มาจากเอาต์พุตของ bridge A เพื่อต่อเข้ามอเตอร์	เชื่อมต่อมอเตอร์
3	จัมเปอร์สำหรับเลือกใช้งานไอซี 7805 บนโมดูล ซึ่งทำหน้าที่จ่ายแรงดันไฟฟ้า 5 V เพื่ออ้างอิงสัญญาณลอจิก **โดยขนาดแหล่งจ่ายไฟภายนอกห้ามเกิน 12 V** เพื่อป้องกันไม่ให้ไอซี 7805 เกิดความร้อนสูงจนเกินไป	คงจัมเปอร์ไว้
4	เชื่อมต่อแหล่งจ่ายไฟภายนอก	เชื่อมต่อ power supply
5	กราวนด์	เชื่อมต่อ ground ของ power supply และ MCU
6	เอาต์พุตไฟ 5 V จากไอซี 7805 เมื่อคงจัมเปอร์ไว้	ไม่ใช้งาน
7	เชื่อมต่อขา Enable A ของไอซี L298 โดยเชื่อมต่อกับสัญญาณ PWM จาก MCU เพื่อควบคุมความเร็วของมอเตอร์ที่คอนเนคเตอร์นี้	เชื่อมต่อสัญญาณ PWM จาก MCU

หมายเลขคอนเนคเตอร์	หน้าที่	สำหรับการทดลองข้อ 4
8; 9	เชื่อมต่อสัญญาณอินพุตแบบ TTL สำหรับควบคุมทิศทางของมอเตอร์ที่เชื่อมต่อ bridge A มอเตอร์จะหมุนเมื่อสัญญาณลอจิกทั้งสองมีค่าตรงข้ามกัน หากสัญญาณลอจิกเหมือนกันมอเตอร์จะไม่หมุน	เชื่อมต่อสัญญาณ GPIO จาก MCU
10; 11	เชื่อมต่อสัญญาณอินพุตแบบ TTL สำหรับควบคุมทิศทางของมอเตอร์ที่เชื่อมต่อ bridge B มอเตอร์จะหมุนเมื่อสัญญาณลอจิกทั้งสองมีค่าตรงข้ามกัน หากสัญญาณลอจิกเหมือนกันมอเตอร์จะไม่หมุน	ไม่ใช้งาน
12	เชื่อมต่อขา Enable B ของไอซี L298 โดยเชื่อมต่อกับสัญญาณ PWM จาก MCU เพื่อควบคุมความเร็วของมอเตอร์ที่คอนเนคเตอร์นี้	ไม่ใช้งาน
13; 14	มาจากเอาต์พุตของ bridge B เพื่อต่อเข้ามอเตอร์	ไม่ใช้งาน

5. การทดลอง

1. การสร้างสัญญาณ PWM จาก TIM2_CH3

จงเขียนโปรแกรมควบคุม TIM2 ช่องสัญญาณ 3 เพื่อสร้างสัญญาณ PWM ที่มีคาบเวลา 1 ms และมี Duty Cycle 50%

- โดยตั้งค่า TIM2 ดังรูปที่ 2.1 รูปที่ 2.3 และรูปที่ 2.4
- เพิ่มคำสั่งใน while loop เพื่อให้ไมโครคอนโทรลเลอร์สร้างสัญญาณ PWM ที่มีคาบเวลา 1 ms มี Duty Cycle 50% ตามที่ตั้งค่าไว้ เป็นระยะเวลานาน 1 วินาที แล้วจึงหยุดสร้างสัญญาณ PWM ได้ดังนี้
  - `HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_3) ;`
  - `HAL_Delay(1000);`
  - `HAL_TIM_PWM_Stop(&htim2, TIM_CHANNEL_3) ;`
- ใช้ Logic Analyzer เชื่อมต่อขา PA2 เพื่อตรวจสอบคาบเวลาและ Duty Cycle

2. การกำหนด Duty Cycle ให้กับสัญญาณ PWM

จงเขียนโปรแกรมควบคุม TIM2 ช่องสัญญาณ 3 เพื่อสร้างสัญญาณ PWM ที่มี Duty Cycle แตกต่างกันโดยการแก้ไขค่าในรีจิสเตอร์ TIM2_CCR3

- สร้างตัวแปร `float dutyCycle = 0.25` ในฟังก์ชัน `main` เพื่อใช้ปรับเปลี่ยนค่า Duty Cycle
- เปลี่ยนคำสั่งใน while loop ให้เป็นคำสั่งดังต่อไปนี้แทน
  - `htim2.Instance -> CCR3 = (1000-1)*dutyCycle;`
  - `HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_3);`
  - `HAL_Delay(1000);`
  - `HAL_TIM_PWM_Stop(&htim2, TIM_CHANNEL_3);`
- ใช้ Logic Analyzer เพื่อตรวจสอบผลลัพธ์แล้วบันทึกรูปสัญญาณที่ได้ลงในตารางที่ 5.1

ตารางที่ 5.1 สำหรับบันทึกผลการทดลองที่ 2

ค่าตัวแปร dutyCycle	รูปสัญญาณ PWM ที่ได้	Duty cycle ของสัญญาณ PWM
0.25		
0.5		
0.75		

ค่าตัวแปร dutyCycle	รูปสัญญาณ PWM ที่ได้	Duty cycle ของสัญญาณ PWM
1.0		
2.0		

3. การสร้างสัญญาณ PWM ที่สามารถปรับเปลี่ยน Duty Cycle ขณะประมวลผลได้

จงสร้างสัญญาณ PWM ทีมีคาบเวลา 100 ms โดยใช้ **TIM3 Channel 4 ที่ขา PB1** โดยเริ่มต้นกำหนดให้มี duty cycle 50% เมื่อกดสวิตช์ wakeup ให้เพิ่ม duty cycle ครั้งละ 10% โดยไม่ให้เกิน 100% เมื่อกดสวิตช์ tamper ให้ลด duty cycle ครั้งละ 10% โดยไม่ให้ต่ำกว่า 0% ตรวจสอบผลการโดยใช้ logic analyzer (ควรใช้ external interrupt ของสวิตช์ทั้งสอง)

4. การทดลองใช้โมดูล L298 ควบคุมมอเตอร์

จงศึกษาข้อมูลจากตารางที่ 4.2 เพื่อต่อวงจรควบคุมมอเตอร์โดยใช้โมดูล L298 โดยใช้ power supply เพื่อจ่ายไฟให้กับโมดูลขนาด 5-12 V สัญญาณอินพุต TTL ให้ใช้สัญญาณ GPIO ขาใดก็ได้ และให้นำสัญญาณ PWM ที่สร้างจากบอร์ดทดลองในการทดลองที่ 3 เชื่อมต่อไปที่คอนเนคเตอร์หมายเลข 9 ทดลองกดสวิตช์ Wakeup และ Temper แล้วสังเกตการทำงานและบันทึกผล โดยให้ต่อสัญญาณ PWM เข้ากับ logic analyzer ตอนตรวจด้วย

**** แหล่งจ่ายไฟของ logic trainer ไม่สามารถจ่ายกระแสให้มอเตอร์ได้เพียงพอ จะทำให้เกิดการ overload ได้ ****

**** ห้ามปรับ Power Supply เกิน 12 V ****

**** ตอนสั่งมอเตอร์ให้ทำงาน ควรจับมอเตอร์เอาไว้ ไม่ควรวางไว้กับพื้นโต๊ะ เพราะมอเตอร์จะสะบัด ****

6. การทดลองพิเศษ (เลือก 1 ข้อ)

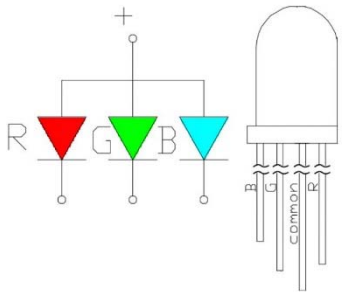
1. จงต่อวงจรควบคุมมอเตอร์แบบสองทิศทาง โดยดูตัวอย่างการต่อได้จาก**Error! Reference source not found.** โดยใช้ UART เพื่อป้อนข้อมูลจากแป้นพิมพ์ซึ่งสามารถกำหนดทิศทางการหมุนและ duty cycle ได้ ดังนี้

- ปุ่ม A เพื่อให้มอเตอร์หมุนทวนเข็ม
- ปุ่ม D เพื่อให้มอเตอร์หมุนตามเข็ม
- ปุ่ม W สำหรับเร่งความเร็วขึ้น 10%
- ปุ่ม D สำหรับลดความเร็วลง 10%

โดยให้ต่อสัญญาณ PWM เข้ากับ logic analyzer ตอนส่งด้วย โดยให้ TA กำหนดขาสัญญาณ PWM ที่ใช้

Time ..... Channel ..... ขาไอซี .....

2. ให้สร้างวงจรผสมสีจากหลอดไฟ LED RGB โดยให้สร้างสัญญาณ PWM จำนวน 3 สัญญาณเพื่อควบคุมความเข้มของสีแดง สีเขียว และสีน้ำเงิน โดยกดปุ่ม R เพื่อปรับความเข้มสีแดง ปุ่ม G เพื่อปรับความเข้มสีเขียว และปุ่ม B เพื่อปรับความเข้มสีน้ำเงิน ในการกดแต่ละครั้งจะเพิ่มความเข้มขึ้น 10% เริ่มจาก 0% – 100% แล้ววนมาที่ 0% อีกครั้ง โดยให้ TA กำหนดขาสัญญาณ PWM ที่ใช้



LED RGB คือ LED ที่ภายในประกอบไปด้วย LED ที่เป็นแม่สีจำนวน 3 สี ได้แก่ สีแดง สีเขียว และสีน้ำเงิน ทำให้สามารถนำมาสร้างเป็นแสงสีต่างๆ ได้ตามต้องการ โดยการกำหนดความเข้มแสงให้ LED แต่ละสีด้วยสัญญาณ PWM โดย LED ทั้งสามมีขา Common ร่วมกัน 1 ขา ซึ่งมีทั้งแบบ Common Anode และ Common Cathode และการใช้งานจะต้องต่อตัวต้านเพื่อจำกัดกระแสจำนวน 3 ตัวด้วย

สีแดง Time ..... Channel ..... ขาไอซี .....

สีเขียว Time ..... Channel ..... ขาไอซี .....

สีน้ำเงิน Time ..... Channel ..... ขาไอซี .....

ใบตรวจการทดลองที่ 7

วัน/เดือน/ปี _____ ☐ **Sec 1** ☐ **Sec 2 กลุ่มที่** _____

1. รหัสนักศึกษา _____ ชื่อ-นามสกุล _____

2. รหัสนักศึกษา _____ ชื่อ-นามสกุล _____

3. รหัสนักศึกษา _____ ชื่อ-นามสกุล _____

ลายเซ็นผู้ตรวจ

การทดลองข้อ 2 ผู้ตรวจ _____ สัปดาห์ที่ตรวจ ☐ W ☐ W+1

การทดลองข้อ 3&4 ผู้ตรวจ _____ สัปดาห์ที่ตรวจ ☐ W ☐ W+1

คำถามท้ายการทดลอง

1. ในฟังก์ชัน `MX_TIM2_Init` ของการทดลองที่ 2 หากเปลี่ยนคำสั่ง

```
sConfigOC.OCMode  = TIM_OCMODE_PWM1;
```

เป็น

```
sConfigOC.OCMode = TIM_OCMODE_PWM2;
```

ให้ผลเหมือนเดิมหรือต่างจากเดิมอย่างไร
