Table of Contents

1. Fan and Servo controlling 2

1. Fan and Servo controlling

Both the fan and servo are controlled with a PWM signal. In the STM32WB55 microcontroller, there are two timers capable of generating PWM signals. One of them is connected to the RGB LED, second is connected to the fan and servo.

Tab. 1: Assignment of pins for the fan and servo

GPIO no	element
PA0	Servo
PA1	Fan

In the stm32duino examples [1)], there is a simple program which allows us to control a single LED with PWM. It is named “All-in-one_setPWM” [2)]. It uses hardware timer output connected directly to an LED. We will modify this example to control the fan and servo. At the beginning of the code, we need to define pins.

```
// PWM timer 2 for Servo and Fan control
#define SERVO_PWM_PIN PA0
#define FAN_PWM_PIN PA1
```



Pin names use STM naming, not Arduino naming.

Both outputs are controlled with the same timer, so only one instance of the timer is needed. It will control two channels one for fan, one for servo.

```
// PWM variables definitions
TIM_TypeDef *SRVInstance;         //General hardware instance
uint32_t channelSRV, channelFAN; //2 channels
HardwareTimer *MyTimServo;        //Hardware Timer class for Fan and Servo
PWM
```

At the beginning of the code timer object must be created and two channels activated. Notice that because there is one timer, the base frequency for the fan and servo is the same. Certainly, the duty cycle can be different.



The frequency used must fulfil the requirements of the servo. The one used in the laboratory stand requires a PWM signal with a pulse sent every 20ms (50Hz). The duty cycle can vary between 500us (microseconds) and 1500us, where 1000us gives 90 degrees of the angle.

```
  //Create Timer instance type based on the servo pin (the same timer is
used for fan).
  TIM_TypeDef *SRVInstance = (TIM_TypeDef
*)pinmap_peripheral(digitalPinToPinName(SERVO_PWM_PIN), PinMap_PWM);
```

```
  //Define two separate channels for servo and fan
  channelSRV =
STM_PIN_CHANNEL(pinmap_function(digitalPinToPinName(SERVO_PWM_PIN),
PinMap_PWM));
  channelFAN =
STM_PIN_CHANNEL(pinmap_function(digitalPinToPinName(FAN_PWM_PIN),
PinMap_PWM));
  // Instantiate HardwareTimer object. Thanks to 'new' instantiation,
HardwareTimer is not destructed when setup() function is finished.
  MyTimServo = new HardwareTimer(SRVInstance);
  // Configure and start PWM
  MyTimServo->setPWM(channelSRV,SERVO_PWM_PIN,50,5); // 50 Hertz, 5%
dutycycle.
  MyTimServo->setPWM(channelFAN,FAN_PWM_PIN,50,50);   // 50 Hertz, 50%
dutycycle.
```

To modify the duty cycle it is possible to use the function setCaptureCompare() which has the possibility of passing the argument as the value of microseconds or percents. The following lines show examples of setting that fan speed in percents (provided by FanSpeed variable), and servo range in duty cycle time (provided by SrvTime variable).

```
  MyTimServo->setCaptureCompare(channelFAN, FanSpeed,
PERCENT_COMPARE_FORMAT);
  MyTimServo->setCaptureCompare(channelSRV, SrvTime,
MICROSEC_COMPARE_FORMAT);
```

1)
https://github.com/stm32duino
2)
https://github.com/stm32duino/STM32Examples/tree/main/examples/Peripherals/HardwareTimer/All-in-one_setPWM

From:
https://dokuwiki.zmitac.aei.polsl.pl/ - **RAU6 Dokuwiki**

Permanent link:
https://dokuwiki.zmitac.aei.polsl.pl/en:edu:zmitac:s2:aih:lab:stm32fan

Last update: **2024/01/16 09:37**