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. The code is composed of two files FanSrvTest.h and FanSrvTest.cpp. The *.h file defines pins for the fan and servo, and declares variables for the timer, channels and timer class.

FanSrvTest.h

```
#include "Arduino.h"
#include <HardwareTimer.h>

#ifndef _SRV_TestLib
#define _SRV_TestLib

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

// 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

//Prototypes of funcions
void FanSrvSetup();
void fFanSet(int pPercent);      //Change the fan speed (0-100)
void fSrvSet(int pAngle);        //Change the servo angle (0-180)

#endif
```

The *.cpp file implements three functions. The FanSrvSetup() function creates a timer instance, assigns channels to timer outputs and pins, and creates a timer class object. After the successful creation, it configures the frequency and initial duty cycle for both channels. Fan frequency is not essential, so the base frequency for the timer is set based on the requirements of the servo and equals 50Hz. The initial duty cycle is set to 5%. The fFanSet() function sets the speed of the fan. The parameter provided is expressed in percentages and can vary between 0 and 100. The fSrvSet() function sets the angle of the servo. The parameter is expressed in degrees and can vary between 0

and 180.

FanSrvTest.cpp

```
#include "FanSrvTest.h"

void FanSrvSetup()
{
  //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,50,5);   // 50 Hertz, 5%
dutycycle.
};

void fFanSet(int pSpeed)
{
  MyTimServo->setCaptureCompare(channelFAN, pSpeed, PERCENT_COMPARE_FORMAT);
//modify duty cycle
};

void fSrvSet(int pAngle)
{
  if (pAngle<0) pAngle=0;        //check boundaries of angle parameter
  if (pAngle>180) pAngle=180;
  int i_srv=500+(100*pAngle)/18; //minimal duty cycle is 0,5ms, maximal
1,5ms
  MyTimServo->setCaptureCompare(channelSRV, i_srv, MICROSEC_COMPARE_FORMAT);
//modify duty cycle
};
```

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?rev=1705174378

Last update: **2024/01/13 20:32**