## 杭州电子科技大学创新实践实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 学院 | 计算机学院 | 班级 |  | 学号 |  |
| 姓名 |  | 日期 | 2024.04.16 | 成绩 |  |
| 实验题目 | 电机驱动的实现 | | | | |
|  | | | | | |
| 实验目的 | 1、学习网站上模块12的视频部分，<https://edu.21ic.com/lesson/1975>  2、学习“讲解”“实验”pdf，重点学习“讲解”的第19页，实验部分12.4  3、完成maze代码中的Lab12\_motor（Program12-1函数）  4、补全函数Motor\_InitSimple，Motor\_ForwardSimple，Motor\_BackwardSimple，Motor\_LeftSimple和Motor\_RightSimple函数，不需要用PWM  上传实验报告，实验结果拍摄功能演示视频 | | | | |
| 硬件原理 | 电机旋转应用到电磁感应原理， 电机驱动及配电板上PH端口控制小车轮子转动方向，高电平为反向，低电平为正向，PH分别和MSP432的P1.7和P1.6相接控制左右轮转动方向；EN端口控制小车轮子是否转动，高电平转动，低电平不转动，分别和MSP432的P2.7和P2.6连接控制左右轮是否转动；nSLEEP控制小车是否休眠，低电平休眠，高电平不休眠，分别和MSP432的P3.7和P3.6相连接控制小车是否休眠。 | | | | |
| 关键代码及注释 | // MotorSimple.c  // Runs on MSP432  // Provide mid-level functions that initialize ports and  // set motor speeds to move the robot.  // Starter code for Lab 12, uses Systick software delay to create PWM  // Daniel Valvano  // July 7, 2017  /\* This example accompanies the books  "Embedded Systems: Introduction to the MSP432 Microcontroller",  ISBN: 978-1512185676, Jonathan Valvano, copyright (c) 2017  "Embedded Systems: Real-Time Interfacing to the MSP432 Microcontroller",  ISBN: 978-1514676585, Jonathan Valvano, copyright (c) 2017  "Embedded Systems: Real-Time Operating Systems for ARM Cortex-M Microcontrollers",  ISBN: 978-1466468863, , Jonathan Valvano, copyright (c) 2017  For more information about my classes, my research, and my books, see  http://users.ece.utexas.edu/~valvano/  Simplified BSD License (FreeBSD License)  Copyright (c) 2017, Jonathan Valvano, All rights reserved.  Redistribution and use in source and binary forms, with or without modification,  are permitted provided that the following conditions are met:  1. Redistributions of source code must retain the above copyright notice,  this list of conditions and the following disclaimer.  2. Redistributions in binary form must reproduce the above copyright notice,  this list of conditions and the following disclaimer in the documentation  and/or other materials provided with the distribution.  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"  AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE  IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE  ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE  LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL  DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;  LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED  AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,  OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE  USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.  The views and conclusions contained in the software and documentation are  those of the authors and should not be interpreted as representing official  policies, either expressed or implied, of the FreeBSD Project.  \*/  // Sever VCCMD=VREG jumper on Motor Driver and Power Distribution Board and connect VCCMD to 3.3V.  // This makes P3.7 and P3.6 low power disables for motor drivers. 0 to sleep/stop.  // Sever nSLPL=nSLPR jumper.  // This separates P3.7 and P3.6 allowing for independent control  // Left motor direction connected to P1.7 (J2.14)  // Left motor PWM connected to P2.7/TA0CCP4 (J4.40)  // Left motor enable connected to P3.7 (J4.31)  // Right motor direction connected to P1.6 (J2.15)  // Right motor PWM connected to P2.6/TA0CCP3 (J4.39)  // Right motor enable connected to P3.6 (J2.11)  **#include** <stdint.h>  **#include** "msp.h"  **#include** "../inc/SysTick.h"  **#include** "../inc/Bump.h"  **#include** "../inc/Clock.h"  // \*\*\*\*\*\*\*Lab 12 solution\*\*\*\*\*\*\*  **void** **Motor\_InitSimple**(**void**){  // Initializes the 6 GPIO lines and puts driver to sleep  // Returns right away  // initialize P1.6 and P1.7 and make them outputs  // write this as part of Lab 12  P1->SEL0 &= ~0xC0;  P1->SEL1 &= ~0xC0;//DIR  P2->SEL0 &= ~0xC0;  P2->SEL1 &= ~0xC0;//PWM  P3->SEL0 &= ~0xC0;  P3->SEL1 &= ~0xC0;//nSLEEP  P1->DIR |= 0xC0;  P2->DIR |= 0xC0;  P3->DIR |= 0xC0;  P1->OUT &= ~0xC0;  P2->OUT &= ~0xC0;  P3->OUT &= ~0xC0;  }  **void** **Motor\_StopSimple**(**void**){  // Stops both motors, puts driver to sleep  // Returns right away  P1->OUT &= ~0xC0;  P2->OUT &= ~0xC0; // off  P3->OUT &= ~0xC0; // low current sleep mode  }  **void** **Motor\_ForwardSimple**(uint16\_t duty, uint32\_t time){  // Drives both motors forward at duty (100 to 9900)  // Runs for time duration (units=10ms), and then stops  // Stop the motors and return if any bumper switch is active  // Returns after time\*10ms or if a bumper switch is hit  // write this as part of Lab 12  P3->OUT |= 0xC0;  P1->OUT &= ~0xC0;  P2->OUT |= 0xC0; //activate motor  uint32\_t circle=0;  **for**(circle=0;circle<time;circle++)// 10ms/circle  {  Clock\_Delay1us(duty);  P2->OUT &= ~0xC0;  Clock\_Delay1us(10000-duty);  P2->OUT |= 0xC0;  **if**(Bump\_Read())  {  Motor\_InitSimple();  **break**;  }  }  P2->OUT &= ~0xC0;  }  **void** **Motor\_BackwardSimple**(uint16\_t duty, uint32\_t time){  // Drives both motors backward at duty (100 to 9900)  // Runs for time duration (units=10ms), and then stops  // Runs even if any bumper switch is active  // Returns after time\*10ms  // write this as part of Lab 12  P3->OUT |= 0xC0;  P1->OUT |= 0xC0;  P2->OUT |= 0xC0; //activate motor  uint32\_t circle=0;  **for**(circle=0;circle<time;circle++)// 10ms/circle  {  Clock\_Delay1us(duty);  P2->OUT &= ~0xC0;  Clock\_Delay1us(10000-duty);  P2->OUT |= 0xC0;  **if**(Bump\_Read())  {  Motor\_InitSimple();  **break**;  }  }  P2->OUT &= ~0xC0;  }  **void** **Motor\_LeftSimple**(uint16\_t duty, uint32\_t time){  // Drives just the left motor forward at duty (100 to 9900)  // Right motor is stopped (sleeping)  // Runs for time duration (units=10ms), and then stops  // Stop the motor and return if any bumper switch is active  // Returns after time\*10ms or if a bumper switch is hit  P3->OUT |= 0x80;  P1->OUT &= ~0x80;  P2->OUT |= 0x80; //activate motor  uint32\_t circle=0;  **for**(circle=0;circle<time;circle++)// 10ms/circle  {  Clock\_Delay1us(duty);  P2->OUT &= ~0x80; //stop  Clock\_Delay1us(10000-duty);  P2->OUT |= 0x80; //continue  **if**(Bump\_Read())  {  Motor\_InitSimple();  **break**;  }  }  P2->OUT &= ~0x80; //stop  // write this as part of Lab 12  }  **void** **Motor\_RightSimple**(uint16\_t duty, uint32\_t time){  // Drives just the right motor forward at duty (100 to 9900)  // Left motor is stopped (sleeping)  // Runs for time duration (units=10ms), and then stops  // Stop the motor and return if any bumper switch is active  // Returns after time\*10ms or if a bumper switch is hit  P3->OUT |= 0x40;  P1->OUT &= ~0x40;  P2->OUT |= 0x40; //activate motor  uint32\_t circle=0;  **for**(circle=0;circle<time;circle++)// 10ms/circle  {  Clock\_Delay1us(duty);  P2->OUT &= ~0x40; //stop  Clock\_Delay1us(10000-duty);  P2->OUT |= 0x40; //continue  **if**(Bump\_Read())  {  Motor\_InitSimple();  **break**;  }  }  P2->OUT &= ~0x40; //stop  // write this as part of Lab 12  } | | | | |
| 实验步骤 | 补全函数Motor\_InitSimple，Motor\_ForwardSimple，Motor\_BackwardSimple，Motor\_LeftSimple和Motor\_RightSimple函数，将代码烧录到小车上，运行小车。 | | | | |
| 实验结果 | 小车前进，后退，只让左轮转动和只让右轮转动都能实现。 | | | | |
| 思考与反馈 | 根据12\_Motor\_Lecture\_cn.pdf中的电机驱动和配电板原理图可以快速写出控制小车轮子转动的代码 | | | | |