课程作业1：用arduino在指定引脚设定可调周期与占空比信号

18机械工程4班 罗宏利 2018G0505025

设计思路：

1. 、指定引脚信号利用LED灯闪烁模拟波形
2. 、利用多任务调度框架定时执行不同任务
3. 、利用引脚的高低电平实现指示灯闪烁
4. 、串口数据用字节的方法读取

实验材料：Arduino-UNO3、ArduinoIDE编译环境

实验过程：

一、首先下载调度器文件TaskScheduler.h，利用调度器对处理13引脚的两个任务：拉高引脚电平和拉低引脚电平。程序开头先定义变量及设定引脚的初始参数，设定矩形波参数为T=1000ms，dc=500ms，即设定占空比为0.5。

二、向调度器添加任务ida和idb。

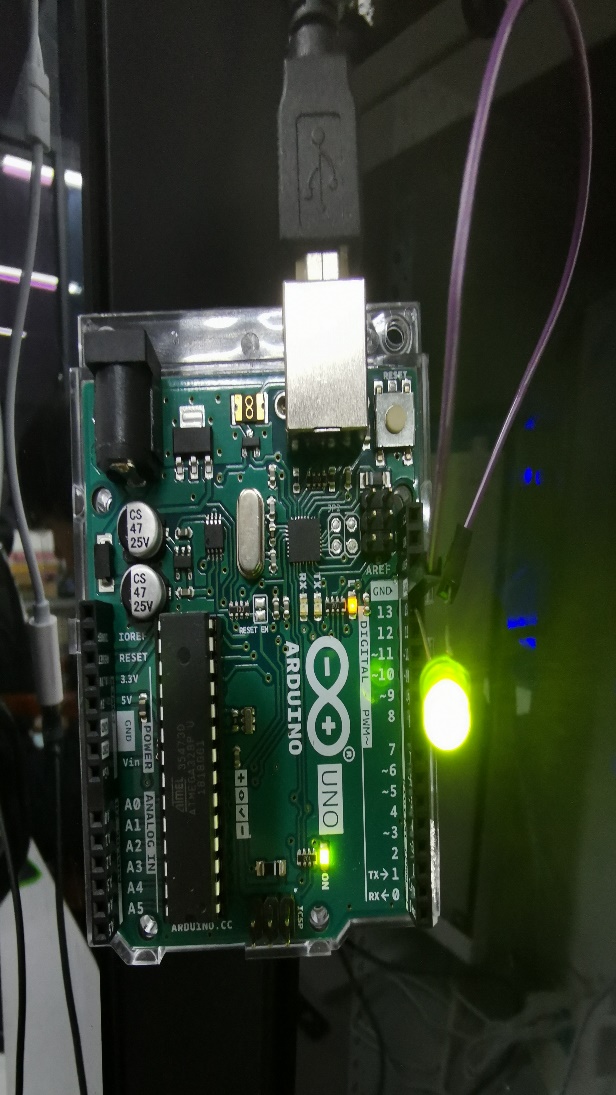
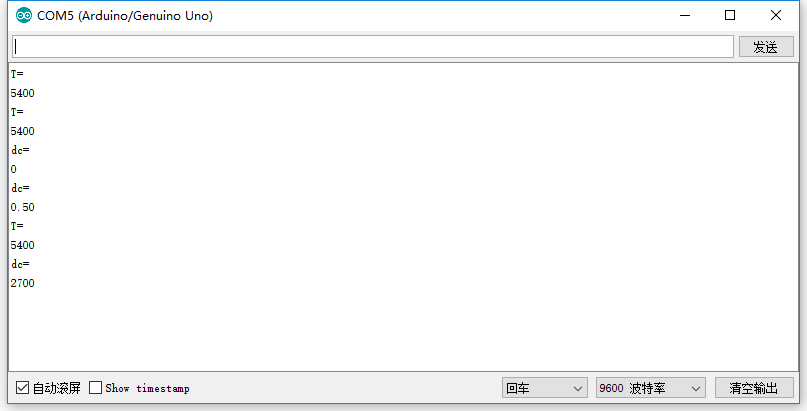
ida= Sch.addTask(led1Update,0,t,1);

idb= Sch.addTask(led2Update,dc,t,1);

任务ida为从第0ms开始闪烁LED，每隔1000ms改变一次LED灯的状态，同理idb为从第20ms开始闪烁，每隔500ms改变一次LED状态。循环执行调度器任务。

三、利用串口调试助手来输入数据，以回车\r作为指令接收符号，当检测到输入回车键后，统一处理用户输入。因为采用的是通过字节的形式来采取输入数据，而输入格式一般为T=4000（输入周期为4000ms）或dc=0.25（占空比为0.25），由此用第一个字节做一个if判断，来检测输入内容。当第一个字节为T时，即dtaUart[0]=='T'，执行以下命令：从第三个字节开始，读取数据，因为读取的字节是字符串形式，故用一个switch函数来将字符串转化为相应的值，得到输入周期或占空比的具体的值，并输出显示得到的值。同理输入的占空比dc从第六个字节开始读取，通过相同操作得到占空比的大小，并与周期相乘得到准确的时间值并输出。

四、将得到的周期及占空比大小拿到任务ida和idb中执行，便可看到在Arduino板子上输出对应周期和占空比的指示灯闪烁情况。

输出如图：

图表 2：Arduino引脚LED灯闪烁图

图表 2：串口输入输出显示图

反思与总结：

本次设计比较简单，只是初步实现既定目标。应往更多功能的方面设计。例如：应该设计一个功能界面，提供周期及占空比输入，通过输入的值多少可以直接取用，通过串口的字节读取一方面效率不高，一方面难以做更为复杂的计算。同时也可设计一些报警提醒，如当用户输入参数格式不对或者超出范围时，弹出警告提示，更加人性化。而本设计中没有相关功能。

程序代码：

主程序：

#include "TaskScheduler.h" // include this file to use this library

#include <Wire.h> //serial port?

// the state of LEDs

boolean g\_led1State=1;

boolean g\_led2State=0;

//串口缓冲

char dtaUart[15];

char dtaLen = 0;

char DealFlag=0;

//矩形波参数

int t=1000;

int dc=500;

//任务ID

int ida,idb,idc;

void setup()

{

pinMode(13,OUTPUT); //led的闪烁模拟波形

Serial.begin(9600);//串口初始化

digitalWrite(13,LOW);//预设

//pinMode(12,OUTPUT);

Sch.init(); // Initialize task scheduler

ida= Sch.addTask(led1Update,0,t,1); //从第 0 毫秒开始闪烁 LED，每隔 1s, LED 状态改变一次

idb= Sch.addTask(led2Update,dc,t,1); //从第 20 毫秒开始闪烁 LED，每隔 0.5s, LED 状态改变一次

Sch.start(); // Start the task scheduler

}

void loop()

{

Sch.dispatchTasks();

}

void led1Update()

{

digitalWrite(13,HIGH);

}

void led2Update()

{

digitalWrite(13,LOW);

}

void serialEvent()

{

char temp;

unsigned int a,c;

float m;

char b;

while(Serial.available())

{

temp=Serial.read();

dtaUart[dtaLen++] =temp ;

if(temp=='\r') //回车'\r'换行'\n'

{

DealFlag=1;

digitalWrite(13,HIGH);

//判断

if(dtaUart[0]=='T')

{

dtaLen=dtaLen-2;

for(c=0,b=2;dtaLen>1;dtaLen--)

{

switch(dtaUart[b++])

{

case '0':a=0;break;

case '1':a=1;break;

case '2':a=2;break;

case '3':a=3;break;

case '4':a=4;break;

case '5':a=5;break;

case '6':a=6;break;

case '7':a=7;break;

case '8':a=8;break;

case '9':a=9;break;

}

c=c\*10+a;

}

Serial.println("T=");

Serial.println(c);

t=c;

// dtaLen=0;

}

else if(dtaUart[0]=='d')

{

dtaLen=dtaLen-5;

for(m=0,b=5;dtaLen>1;dtaLen--)

{

switch(dtaUart[b++])

{

case '0':a=0;break;

case '1':a=1;break;

case '2':a=2;break;

case '3':a=3;break;

case '4':a=4;break;

case '5':a=5;break;

case '6':a=6;break;

case '7':a=7;break;

case '8':a=8;break;

case '9':a=9;break;

}

m=m\*10+a;

}

m=m/100;

Serial.println("dc=");

Serial.println(m);

//dc=m\*t;

//dtaLen=0;

}

// t=c;

dc=m\*t;

dtaLen=0;

Serial.println("T=");

Serial.println(t);

Serial.println("dc=");

Serial.println(dc);

Sch.deleteTask(ida);

Sch.deleteTask(idb);

ida= Sch.addTask(led1Update,0,t,0); //从第 0 毫秒开始闪烁 LED，每隔 1s, LED 状态改变一次

idb= Sch.addTask(led2Update,dc,t,0); //从第 20 毫秒开始闪烁 LED，每隔 0.5s, LED 状态改变一次

}

}

}

调度器调用TaskScheduler.h文件：

// TaskScheduler.h

// A task scheduler library for arduino - header file

// http://blanboom.org

#ifndef TASK\_SCHEDULER\_H

#define TASK\_SCHEDULER\_H

#if ARDUINO >= 100

#include "Arduino.h"

#else

#include "WProgram.h"

#endif

#define RETURN\_ERROR 0

#define RETURN\_NORMAL 1

// Error codes

//#define REPORT\_ERRORS // Remove "//" to enable error report

#define ERROR\_TOO\_MANY\_TASKS (1)

#define ERROR\_CANNOT\_DELETE\_TASK (2)

// The max number of tasks

#define MAX\_TASKS (10)

// The data structure of per task

typedef struct

{

void (\*pTask)(void);

uint16\_t delay;

uint16\_t period;

uint8\_t runMe;

uint8\_t co\_op;

} Task;

class Schedule

{

//Scheduler functions

private:

void \_reportStatus(void);

void \_goToSleep(void);

public:

void init(void);

void start(void);

void dispatchTasks(void);

uint8\_t addTask(void ( \*)(void), uint16\_t, uint16\_t, boolean);

boolean deleteTask(uint8\_t);

};

static Schedule Sch;

#endif