**Arduino课堂笔记**

1.复习中断

外部中断是由外部设备发起请求的中断，在Arduino UNO板上仅有2和3号引脚可以用于捕获外部中断信号，其中2号引脚的中断编号为0,3号引脚的中断编号为1。

中断模式即中断触发的方式，有LOW低电平触发、CHANGE电平变化触发、RISING上升沿触发、FALLING下降沿触发四种。

(1)自定义中断服务函数：中断被触发后，Arduino要执行的中断函数，该函数不能带任何参数，而且返回值类型为空(void)。

(2)attachInterrupt(interrupt,function,mode);//在setup()中使用此函数对中断引脚进行初始化配置

参数：interrupt，中断编号，UNO板的2号引脚的中断编号为0，3号引脚的中断编号为1。

function,中断函数名，当中断被触发后要运行的函数(程序)。

mode，中断模式，即中断触发的方式。

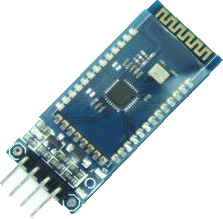
示例：attchInterrupt(0,inteHello,FALLING);

2.认识蓝牙

蓝牙（Bluetooth）是由东芝、爱立信、IBM、Intel和诺基亚于1998年5月共同提出的近距离无线数据通讯技术标准。它能够在10 米的半径范围内实现单点对多点的无线数据和声音传输，其数据传输带宽可达1Mbps。通讯介质为频率在2.402GHz到2.480GHz之间的电磁波。一句话概括：蓝牙是当前最主流的、2.4G、近距离、低功耗无线通信方式之一。

3.蓝牙模块

蓝牙模块（Bluetooth module）是指集成蓝牙功能的芯片基本电路集合，用于短距离2.4G的无线通讯模块。

VCC 电源正极，接（3.3v～5v）

GND 电源负极，接GND

TXD 模块串口发送脚，接Arduino的RX引脚。

RXD 模块串口接收脚，接Arduino的TX引脚。

蓝牙模块按照应用和支持协议划分大致分为两种：经典蓝牙模块和低功耗蓝牙模块。

经典蓝牙模块（BT）：泛指支持蓝牙协议在4.0以下的模块，一般用于数据量比较大的传输。经典蓝牙模块可再细分为：传统蓝牙模块（BR）和高速蓝牙模块（EDR）。传统蓝牙模块支持蓝牙2.1协议，在智能手机爆发的时期得到广泛支持。高速蓝牙模块速率提高到约24Mbps，是传统蓝牙模块的八倍。

低功耗蓝牙模块（BLE）：是指支持蓝牙协议4.0或更高的模块，也称为BLE模块，其最重要的特性是支持省电，以不需占用太多带宽的设备连接为主。

蓝牙模块与Arduino UNO之间通过串口进行通信，所有串口通信的语法对蓝牙模块同样适用。

Arduino UNO的0和1引脚为串口通信专用引脚，也是程序下载的专用引脚，如果蓝牙模块占用此引脚，那么下载程序前可能需要先拔掉蓝牙模块的接线，下载完程序后再接上。

4.复习串口程序

Serial.begin(speed); //初始化串口，设置串口波特率，Arduino支持的波特率包括：300,1200,2400,4800,9600,14400,19200,28800,38400,57600,和115200。

Serial.available(); //判断串口缓冲器是否有数据装入

Serial.read();    //读取串口数据

Serial.print();    //串口输出数据，不包括换行

Serial.println();   //串口输出数据，包括换行

5.手机APP与蓝牙模块的通信

我们使用的手机上也有蓝牙模块，只要在手机端安装一个蓝牙App就可以使用手机蓝牙连接单片机的蓝牙模块，从而实现蓝牙通信。课程中我们使用的是基于安卓系统的蓝牙APP（蓝牙调试助手），安装以后直接打开蓝牙App，根据提示打开手机蓝牙，然后搜索附近的蓝牙设备，找到名为BT05的蓝牙模块就是我们课程中使用的蓝牙模块。

(1)编程任务：利用手机App通过蓝牙控制LED的亮灭。

手机App端一个对话模式，当通过App发送指令给蓝牙模块时，可以控制Arduino端的LED灯亮灭状态

注：

1)蓝牙模块默认名称为BT05

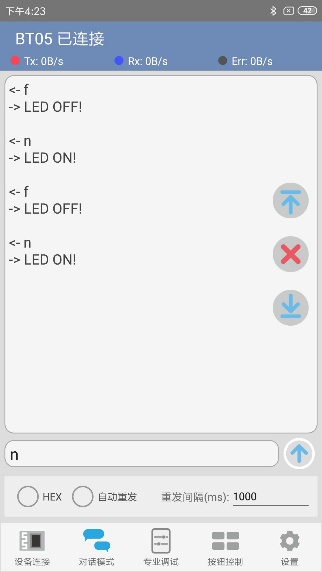
2)由于此蓝牙模块为BLE低功耗蓝牙，无需在手机端进行配对，直接打开蓝牙调试器app，搜索到BT05后直接连接。

3)连接成功后，在对话模式窗口发送指令。

4)BT05蓝牙模块默认波特率为115200，所以在程序中设置串口波特率为115200。

char val;

void setup()

{

Serial.begin(115200);

pinMode(13, OUTPUT);

}

void loop()

{

val = Serial.read();

if (val == 'n')

{

digitalWrite(13, HIGH);

Serial.println("LED ON!");

}

else if (val == 'f') {

digitalWrite(13, LOW);

Serial.println("LED OFF!");

}

}

(2)蓝牙遥控小车，使用蓝牙App的按钮控制模块，遥控小车前后左右移动。

注：

1)蓝牙App的按钮控制模块可以修改按钮的值，也就是发送给蓝牙模块的数值。

2)Arduino端根据蓝牙模块接收到的数值控制小车移动

3)在蓝牙App的按钮控制模式下，打开编辑模式，然后再点击按钮，就可以修改按钮的数值了，修改完以后再把编辑模式关闭。

4)程序示例中，蓝牙App按钮、按钮的值以及小车移动方向对应关系如下：



程序示例：

#define IN1 2

#define IN2 4

#define ENA 3

#define IN3 7

#define IN4 6

#define ENB 5

char val;

void setup() {

pinMode(IN3, OUTPUT);

pinMode(IN4, OUTPUT);

pinMode(ENB, OUTPUT);

pinMode(IN1, OUTPUT);

pinMode(IN2, OUTPUT);

pinMode(ENA, OUTPUT);

Serial.begin(115200);

delay(1000);

}

int a;

void loop() {

val = Serial.read();

if (val == 'G')

{

mForward(120, 120);

Serial.println("forward!");

} else if (val == 'K') {

mBackward(120, 120);

Serial.println("backward!");

} else if (val == 'H') {

mLeft(120, 120);

Serial.println("Left!");

} else if (val == 'J') {

mRight(120, 120);

Serial.println("Right!");

} else if (val == 'I') {

mStop(0, 0);

Serial.println("stop!");

}

delay(10);

}

void mForward(int spl, int spr) {

//右上角电机正转

digitalWrite(IN1, HIGH);

digitalWrite(IN2, LOW);

analogWrite(ENA, spr);

//左上角电机正转

digitalWrite(IN3, HIGH);

digitalWrite(IN4, LOW);

analogWrite(ENB, spl);

}

void mBackward(int spl, int spr) {

//右上角电机反转

digitalWrite(IN1, LOW);

digitalWrite(IN2, HIGH);

analogWrite(ENA, spr);

//左上角电机反转

digitalWrite(IN3, LOW);

digitalWrite(IN4, HIGH);

analogWrite(ENB, spl);

}

void mLeft(int spl, int spr) {

//右上角电机正转

digitalWrite(IN1, HIGH);

digitalWrite(IN2, LOW);

analogWrite(ENA, spr);

//左上角电机反转

digitalWrite(IN3, LOW);

digitalWrite(IN4, HIGH);

analogWrite(ENB, spl);

}

void mRight(int spl, int spr) {

//右上角电机反转

digitalWrite(IN1, LOW);

digitalWrite(IN2, HIGH);

analogWrite(ENA, spr);

//左上角电机正转

digitalWrite(IN3, HIGH);

digitalWrite(IN4, LOW);

analogWrite(ENB, spl);

}

void mStop(int spl, int spr) {

//右上角电机停止

digitalWrite(IN1, LOW);

digitalWrite(IN2, LOW);

analogWrite(ENA, spr);

//左上角电机停止

digitalWrite(IN3, LOW);

digitalWrite(IN4, LOW);

analogWrite(ENB, spl);

}