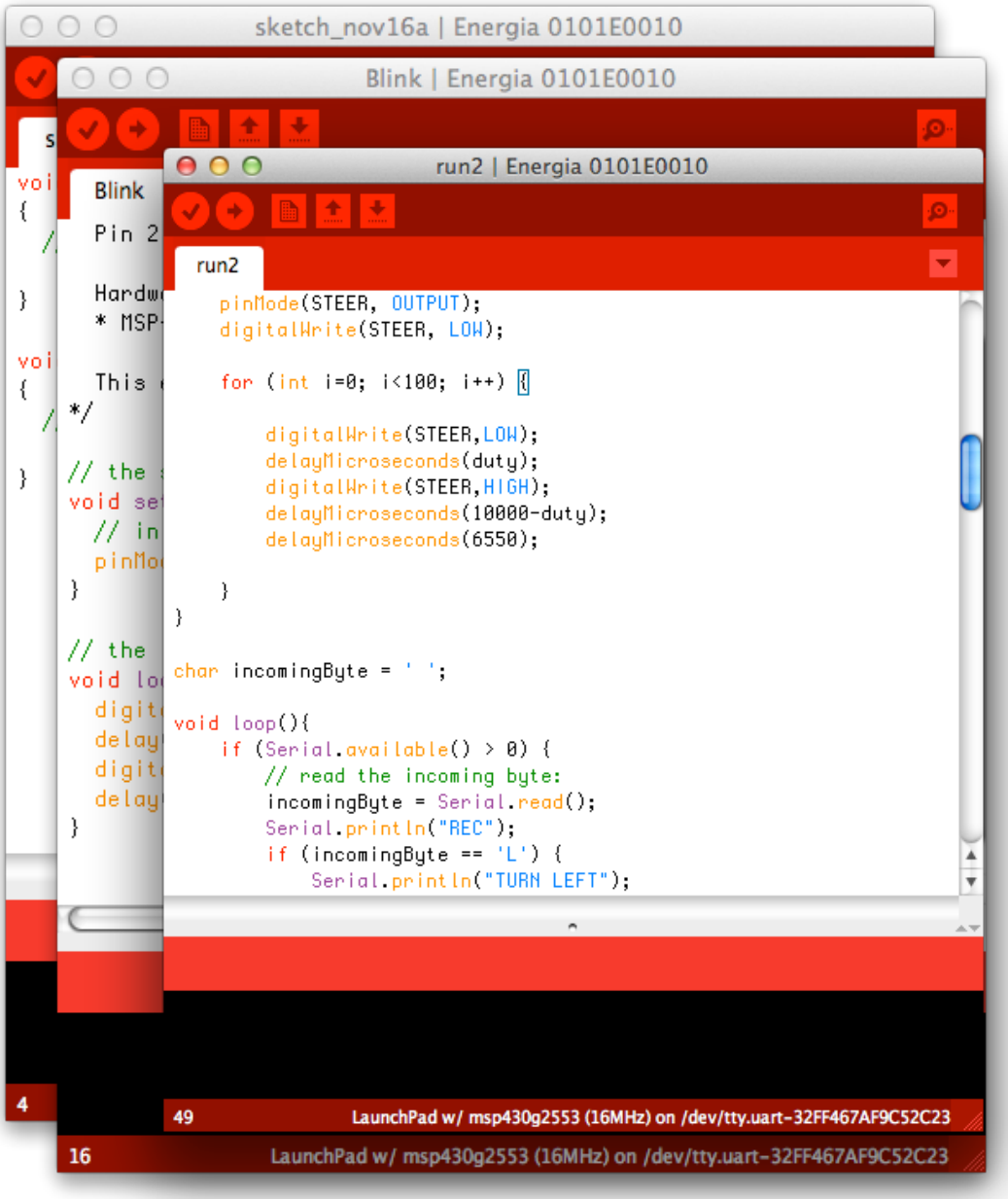
基于Energia的单片机编程

从零基础开始学习MSP-430单片机开发



目录

摘要

在学习工科创IIB的过程中，单片机编程是一个很重要的环节。Energia作为一个简单易用的开发工具，易于上手，适合各个层次的开发者学习、使用。本文从一个从零基础开始学习的MSP-430单片机学习者角度讲述如何在此工具下针对本次工程实践与科技创新 IIB 课题编程及应用。本方法适合有一定C或C++语言基础的单片机初学者学习使用。

关键词：初学者、开源软件

Energia 集成开发环境简介

Energia是一个专为德州仪器公司MSP-430单片机开发的集成开发环境。

该软件在不同操作系统中有良好的兼容性，在Windows，Mac OS，Linux系统中均可使用。

该软件为免费软件，非常适合学习者使用。

该软件自带端口通讯模块，可与MSP-430单片机进行端口通讯。

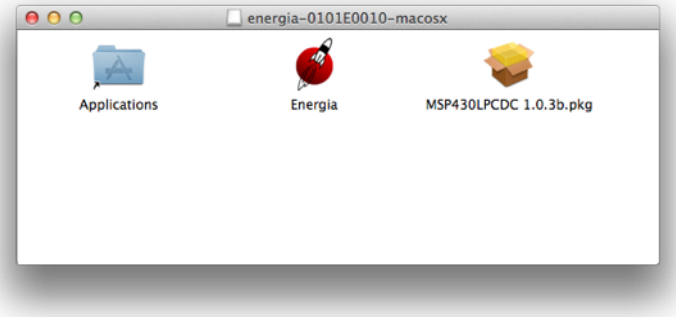
软件下载

该软件为免费软件,可从官方网站 http://www.energia.nu 下载所需版 本,支持 Max OS X, windows 及 Linux 操作系统。

本教程中以 Mac OS X系统为例,故选择版本

energia- 0101E0010-macosx.dmg 下载安装。

软件安装



打开下载的dmg文件,其中包含文件如下:

重启启动后,Energia程序已经安装在Application目录下,按下control键 同时双击该程序图标可以打开该程序。[1]

将其中Energia文件拖拽入 Application文件夹 中。

按下control建,双击MSP430LPCD文件选择安装。 默认安装后,操作系 统会重新启动。

至此,msp430单片机所需的开发环境已经搭建完成,若安装后仍无法打 开,可能出于系统设置或系统版本原因,请参考:
http://energia.nu/Guide MacOSX.html 以下是软件截图:

[1] 由于Energia是一个未在Apple公司注册的应用程序,故第一次打开时需按住control键 强制运行

创建程序

首先学习创建第一个程序用来控制MSP430单片机上的LED灯 闪烁。

打开Energia软件,首先通过 Tools - Serial Port 选择使用的传输端口(通常只有一个选项,故为默认),在选择版面(board)类型。针对我们本次

所使用的MSP-EXP430G2单片机,我们选择

LaunchPad w/ msp430g2553 进 行操作。

所使用程序代码如下:

```
// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(RED_LED, OUTPUT);
}
// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(RED_LED, HIGH);
   // turn the LED on (HIGH is the voltage level)
  delay(1000);
   // wait for a second
  digitalWrite(RED_LED, LOW);
   // turn the LED off by making the voltage LOW
  delay(1000);
   // wait for a second
}
```

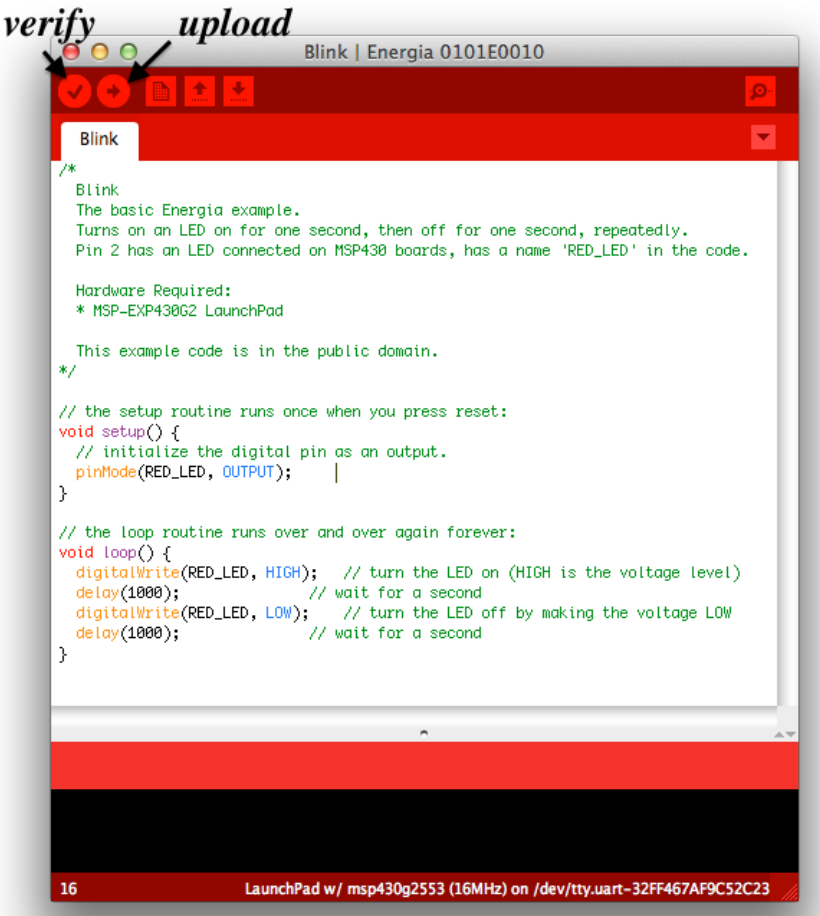
程序说明:

1.该程序包含两端函数,分别用来做初始化设定(setup)和循环操作(loop)。

2.初始化时,选择红色LED灯用作信号输出。

3.循环过程中,使用 digitalWrite(RED_LED,HIGH) 向红色LED灯提供高 电位,使其发光。使用delay函数做1000毫秒延时,使红色LED灯持续 发光1秒。使用 digitalWrite(RED LED,LOW) 命令将红色LED电位调 至低电位,从而LED灯熄灭。再做1000毫秒延时后完成循环。

点击Verify 可以对该程序在计算机上编译、检查错误。编译通过后,点 击upload 可讲程序上传至单片机。参见下图:



运行程序后,可看到红色LED灯开始闪烁。实验成功。

类似的,通过对红色LED灯和绿色LED灯的间断控制,我们可以实现两小灯泡组合闪烁的效果。

串口通讯

Energia自带了串口通讯工具，在与小车建立连接后，通过选择Tools - Serial Monitor 可建立与小车通讯。

在打开该串口工具前，需将含串口通讯的文件传入单片机中，并建立连接。本文以通过给单片机发送字符信号控制LED灯变换，过程如下：

代码：

```
char incomingByte = 'a';   // for incoming serial data

void setup() {
  Serial.begin(9600);     // opens serial port, sets data rate to 9600 bps
  pinMode(RED_LED,OUTPUT);
  pinMode(GREEN_LED,OUTPUT);
  establishContact(); // send a byte to establish contact until receiver responds
}

void loop() {

  // send data only when you receive data:
  if (Serial.available() > 0) {
    // read the incoming byte:
    incomingByte = Serial.read();
    if (incomingByte == 'a') {
      digitalWrite(RED_LED, HIGH);
    } else if (incomingByte == 'b'){
      digitalWrite(RED_LED,LOW);
    } else if (incomingByte == 'c'){
      digitalWrite(GREEN_LED,HIGH);
    } else if (incomingByte == 'd'){
      digitalWrite(GREEN_LED,LOW);
    }
  }
}
```

程序说明：

在setup中，通过 Serial.begin(9600) 函数打开串口，并将数据传输速率设置为9600bps。将LED_RED和LED_GREEN设置为输出端。Serial.available()将判断是否收到信息，当收到有信息接收时，返回值为1，否则为0。

Serial.read()能够从穿孔中读取信息，并将信息做为返回值返回。

运行：

上传并运行该程序后， 打开Tool - Serial Monitor 输入‘a’可打开红的，输入‘b’可关闭红灯，输入‘c’可打开绿灯，输入‘d’可关闭绿灯。

另外，通过Serial.println(string str)；可输出括号中的字符串，注意若要直接输出，需将内容写在双引号，如Serial.println(“Hello World”);

关于更多串口通讯的函数可参见附录A。

数据读取

在与单片机通讯时，难免会遇到需要收发数据的情况，默认传输为char（8位）若要传输int或float等数据类型需要对数据进行处理后才能收发【2】。此处介绍将数据转换为字符串后通过字符串传输的方式收发数据。参考Energia中例程 File - Examples - 6.Strings - StringToInt。

代码：

```
String inString = "";    // string to hold input

void setup() {
  // Initialize serial communications:
  Serial.begin(9600);
}
```

```
void loop() {
  // Read serial input:
  while (Serial.available() > 0) {
    int inChar = Serial.read();
    if (isDigit(inChar)) {
      // convert the incoming byte to a char
      // and add it to the string:
      inString += (char)inChar;
    }
    // if you get a newline, print the string,
    // then the string's value:
    if (inChar == '\n') {
      Serial.print("Value:");
      Serial.println(inString.toInt());
      Serial.print("String: ");
      Serial.println(inString);
      // clear the string for new input:
      inString = "";
    }
  }
}
```

程序说明：

此程序可以将读入的String类型数据转换为整型数。

从串行端口读入字符串，直至读到换行符时停止。然后通过String类中的toInt函数将该字符串转换为int变量。

程序中，isDigit（char a）函数可以用来判断字符a是否指代数字（0~9）。

将该程序上传至单片机，通过Energia所带的串口通信模块，可以与向其发送数字，之后会返回用数字和字符串表示的所发送数据。

应用Energia为走迷宫小车编程

1. 舵机的控制

普通舵机需要使用PWM信号进行控制，简单来说，控制该舵机需要向舵机发送一定周期个固定频率（50Hz左右）的信号，并且，该信号需要有一个较为固定的占空比（duty cycle）（7.5%左右）【3】。详细原理可参见参考文献中中连接，在此不以赘述。

在Energia中，可以使用 delayMicroseconds(int a) 和 delay（int a）函数来记录时间延迟，前者单位为微秒，即为1/1000 000秒，可以在a≤16383时精确延迟a个单位。注意，此处a的值一定不能大于16383，若要精确延迟大于16383个单位时，可以多次使用delayMicroseconds(int a)来实现。

delay(int a)函数延迟单位为毫秒，即为1/1000秒。

若要想舵机发送50Hz信号，则信号周期为20ms，高电位信号约为1.5ms，为了提高精确度，此处统一使用delayMicroseconds函数。

代码：

```
void steer(){
  int tmp = duty;
  for (int i=0; i<CYC; i++) {
    digitalWrite(STEER,LOW);
    delayMicroseconds(tmp);
    digitalWrite(STEER,HIGH);
    delayMicroseconds(10000-tmp);
    delayMicroseconds(6550);
  }
}
```

程序说明：

由于电路连接原因，从STEER端口发送低信号时，舵机可接受到高位信号。具体运行时，需要对全局标量duty值进行反复调试，通过改变

duty及延迟值，得到正方向时全局变量duty对应值。我们的实验中，duty约为1230时可以调节至正方向。

另外，由于额外的预算也会导致时间的延迟，在循环过程中应尽量避免数学运算。

除使用计时函数delay来控制时间延迟外，Energia中还提供了时间读取函数，可以从单片机中读取时间来提高计时的精准度，并可避免其他运算带来的时间延误。所需函数及代码见附录A。

2. 电机控制

电机控制原理较为简单，当给电机一端以高电位信号时，电机开始转动，两端电位相同时，电机静止。

为了调节电机转速，可以使用向电机发送周期信号的方式调节。在Energia中提供了analogWrite函数，可以发出固定占空比的频率约为490Hz的模拟信号，具体使用方法见附录A。使用该函数，通过改变发出信号的占空比，可以改变电机转速。

3. 收发信号

在已有小车上已安装蓝牙模块，可通过蓝牙与计算机进行串口通讯。通讯办法与USB串口通讯相同，若要改变收发信号端口或是增加收发端口，可以通过引入新库来实现。选择Sketch - import Library选择所需新的库，如若要增加通信端口，可以选择SoftwareSerial增加。

4. 整体实现

代码：

```
#define LEFT_AHEAD 10
#define LEFT_BACK 9
#define RIGHT_AHEAD 13
#define RIGHT_BACK 12
#define STEER 14
int duty = 1230;
String inString="";
float RunInit = 3.0;
float RUN_LEFT = 3.0;
```

```
float RUN_RIGHT = 3.0;

int TURN = 60;
int RUN1 = 255;
int CYC = 5;
float calibrateNum = 0;
void stopBack();
void turnLeft();
void turnRight();
void goAhead();
void park();
void goBack();

void setup(){
   Serial.begin(9600);
   pinMode(LEFT_AHEAD,OUTPUT);
   pinMode(LEFT_BACK, OUTPUT);
   pinMode(RIGHT_AHEAD, OUTPUT);
   pinMode(RIGHT_BACK, OUTPUT);
   digitalWrite(LEFT_AHEAD, LOW);
   digitalWrite(LEFT_BACK, LOW);
   digitalWrite(RIGHT_AHEAD, LOW);
   digitalWrite(RIGHT_BACK, LOW);;

   pinMode(STEER, OUTPUT);
   digitalWrite(STEER, LOW);

   for (int i=0; i<100; i++) {

      digitalWrite(STEER,LOW);
      delayMicroseconds(duty);
      digitalWrite(STEER,HIGH);
      delayMicroseconds(10000-duty);
      delayMicroseconds(6550);

   }

}

char incomingByte = ' ';
```

```
void loop(){
  if (Serial.available() > 0) {
    // read the incoming byte:
    incomingByte = Serial.read();
    if (incomingByte == 'L') {
      Serial.println("TURN LEFT");
      turnLeft();
      Serial.read();
    } else if (incomingByte == 'R'){
      Serial.println("TURN RIGHT");
      turnRight();
      Serial.read();
    } else if (incomingByte == 'A'){
      Serial.println("GO AHEAD");
      goAhead();
      Serial.read();
    } else if (incomingByte == 'B'){
      Serial.println("GO BACK");
      goBack();
      Serial.read();
    } else if (incomingByte == 'P'){
      Serial.println("PARK");
      park();
      Serial.read();
    } else if (incomingByte == 'C'){
      calib();
    } else if (incomingByte == 'K'){
      changeP();
    } else if (incomingByte == 'S'){
      changeS();
    } else if (incomingByte == 'D'){
      changeD();
    }

  }
}

void steer(float x){
  int tmp = x*duty;
```

```
    for (int i=0; i<CYC; i++) {

        digitalWrite(STEER,LOW);
        delayMicroseconds(tmp);
        digitalWrite(STEER,HIGH);
        delayMicroseconds(10000-tmp);
        delayMicroseconds(6550);

    }
    //    digitalWrite(STEER,LOW);

}

void goAhead(){
    stopBack();
    analogWrite(LEFT_AHEAD,RUN1);
    analogWrite(RIGHT_AHEAD,RUN1);
    steer(1.0);
}

void turnLeft(){
    stopBack();
    analogWrite(RIGHT_AHEAD,RUN1);
    analogWrite(LEFT_AHEAD,TURN);
    steer(1.1);
}

void turnRight(){
    stopBack();
    analogWrite(LEFT_AHEAD,RUN1);
    analogWrite(RIGHT_AHEAD,TURN);
    steer(0.88);

}

void park(){
    stopBack();
    digitalWrite(LEFT_AHEAD, LOW);
    digitalWrite(RIGHT_AHEAD, LOW);
    steer(1);
```

```
}

void goBack(){
   //    stopBack();
   digitalWrite(LEFT_AHEAD,LOW);
   digitalWrite(RIGHT_AHEAD,LOW);
   analogWrite(LEFT_BACK,RUN1);
   analogWrite(RIGHT_BACK,RUN1);
   steer(1);
}

void stopBack(){
   digitalWrite(LEFT_BACK, LOW);
   digitalWrite(RIGHT_BACK, LOW);
}
int readInt(){
 int x;
 int flag=1;
 while(flag){
  while (Serial.available() > 0) {
    int inChar = Serial.read();
    if (isDigit(inChar)) {
      // convert the incoming byte to a char
      // and add it to the string:
      inString += (char)inChar;
    }
    // if you get a newline, print the string,
    // then the string's value:
    if (inChar == '\n') {
      x=inString.toInt();
      flag=0;
      // clear the string for new input:
      inString = "";
    }
  }
 }
 return x;
}
void calib(){
  TURN = readInt();
```

```
    Serial.println("calibrate:");
    Serial.println(TURN,DEC);
}
void changeP(){
    CYC = readInt();
    Serial.println("DELAY(ms):");
    Serial.println(CYC*2,DEC);
}

void changeS(){
    RUN1 = readInt();
    Serial.println("new spead:");
    Serial.println(RUN1,DEC);
}

void changeD(){
    duty = readInt();
    Serial.println("new duty:");
    Serial.println(duty,DEC);
}
```

参考文献

1. http://energia.nu/Reference_Index.html Energia Reference 2013.11
2. http://blog.csdn.net/liangwei88624/article/details/6885803 串口通信中的int float型数据的处理和发送 2013.11
3. http://wenku.baidu.com/view/01762668a98271fe910ef978.html 详细的舵机控制原理资料 2013.11

附页A *Energia* 语法参考

Language Reference

Energia programs can be divided in three main parts: *structure*, *values* (variables and constants), and *functions*.

Structure

- setup()
- loop()

Control Structures

- if
- if...else
- for
- switch case
- while
- do... while
- break
- continue
- return
- goto

Further Syntax

- ; (semicolon)
- {} (curly braces)
- // (single line comment)
- /* */ (multi-line comment)
- #define
- #include

Variables

Constants

- HIGH | LOW
- INPUT | OUTPUT
- INPUT_PULLUP | INPUT_PULLDOWN
- true | false
- integer constants
- floating point constants

Data Types

- void
- boolean
- char
- unsigned char
- byte
- int
- unsigned int
- word
- long
- unsigned long
- float
- double
- string - char array

Functions

Digital I/O

- pinMode()
- digitalWrite()
- digitalRead()

Analog I/O

- analogReference()
- analogRead()
- analogWrite() - *PWM*

Advanced I/O

- tone()
- noTone()
- shiftOut()
- shiftIn()
- pulseIn()

Time

- millis()
- micros()

Arithmetic Operators

- = (assignment operator)
- + (addition)
- - (subtraction)
- * (multiplication)
- / (division)
- % (modulo)

Comparison Operators

- == (equal to)
- != (not equal to)
- < (less than)
- > (greater than)
- <= (less than or equal to)
- >= (greater than or equal to)

Boolean Operators

- && (and)
- || (or)
- ! (not)

Pointer Access Operators

- * dereference operator
- & reference operator

Bitwise Operators

- & (bitwise and)
- | (bitwise or)
- ^ (bitwise xor)
- ~ (bitwise not)
- << (bitshift left)
- >> (bitshift right)

Compound Operators

- ++ (increment)
- -- (decrement)
- += (compound addition)
- -= (compound subtraction)
- *= (compound multiplication)
- /= (compound division)

- &= (compound bitwise and)
- |= (compound bitwise or)

- String - object
- array

Conversion

- char()
- byte()
- int()
- word()
- long()
- float()

Variable Scope & Qualifiers

- variable scope
- static
- volatile
- const

Utilities

- sizeof()

- delay()
- delayMicroseconds()

Math

- min()
- max()
- abs()
- constrain()
- map()
- pow()
- sqrt()

Trigonometry

- sin()
- cos()
- tan()

Random Numbers

- randomSeed()
- random()

Bits and Bytes

- lowByte()
- highByte()
- bitRead()
- bitWrite()
- bitSet()
- bitClear()
- bit()

External Interrupts

- attachInterrupt()
- detachInterrupt()

Interrupts

- interrupts()
- noInterrupts()

Communication

- Serial
- Stream

Looking for something else?

See the libraries page for interfacing with particular types of hardware. Try the list of community-contributed code on the Forum. The Energia language is based on C/C++ and uses the MSPGCC compiler.

Reference Home

Corrections, suggestions, and new documentation should be posted to the Forum.