

Arduino & Galileo Dev Board Programming

郑灵翔

lxzheng@xmu.edu.cn

厦门大学

Arduino介绍

Arduino是致力于电子产品原型设计的一种开源工具，並且具有使用类似java,C语言的开发环境，让开发者可以快速使用Arduino语言与Flash或Processing等软件，作出互动作品。

Arduino包括一块包含微控制器(MCU)的简单电路板，以及用于为电路板进行软件编程的一套集成开发环境(Arduino IDE)。

Arduino可以用于与现实世界进行交互，通过传感器和开关设备感知物理环境和接受用户操作，并通过控制电灯、扬声器、电机来改变光线、声音和运动。

Arduino也可以独立运作（作为一台微型计算机）；也可以与通用计算机通信，成为一个可以跟其他软件沟通的平台，比如说：LabView，Processing或其他软件。



Arduino与单片机

1、什么是单片机？它与个人计算机有什么不同？

一台能够工作的计算机要有这样几个部份构成：

中央处理单元CPU（进行运算、控制）

随机存储器RAM（数据存储）

存储器ROM（程序存储）

输入/输出设备I/O（串行口、并行输出口等）

在个人计算机（PC）上这些部份被分成若干块芯片，安装在一个被称之为主板的印刷线路板上。而在单片机中，这些部份全部被做到一块集成电路芯片中了，所以就称为单片（单芯片）机，而且有一些单片机中除了上述部份外，还集成了其它部份如模拟量/数字量转换（A/D）和数字量/模拟量转换（D/A）等。

2、单片机有什么用？

实际工作中并不是任何需要计算机的场合都要求计算机有很高的性能，一个控制电冰箱温度的计算机难道要用酷睿处理器吗？应用的关键是看是否够用，是否有很好的性能价格比。如果一台冰箱都需要用酷睿处理器来控制，那价格就是天价了。

单片机通常用于工业生产的控制、生活中与程序和控制有关（如：电子琴、冰箱、智能空调等）的场合。

Why Arduino ?

- 1、便宜——你可以自己搭建电路板，也可以买成品，成本不会超过200
- 2、跨平台——可以在windows下也可以在Linux下进行开发
- 3、简易的编程环境——相比于其他平台或者单片机的开发环境来说
- 4、软件开源并且可扩展

Arduino软件是开源的，并且Arduino编程语言可以通过C++库进行扩展，如果你想了解Arduino到底是怎么工作的？你可以自由查看它的核心代码。

- 5、硬件开源并且可扩展

Arduino基于Creative Commons许可协议，所以有经验的电路设计师能够根据需求设计自己的模块，可以对其进行扩展或改进。甚至是对于一些相对没有什么经验的用户，也可以通过制作试验板来理解Arduino硬件是怎么工作的，省钱又省事。

Arduino历史

Arduino源于意大利伊夫雷亚交互设计学院 (Interaction Design Institute Ivrea) 在2005年开展的一项学生计划，当时的学生使用一款售价100美元的昂贵设备BASIC Stamp进行编程开发。后来这所学校的教师Massimo Banzi连同另外两人一起开始着手设计一款廉价并且使用简单的开发板，数天后，代码和电路板相继完工，这块板子就被命名为Arduino。

此后设计人员不断地对Arduino进行改进，虽然最终学校倒闭了，但是Arduino这一产品却延续了下来。因为Arduino是开源的（遵循一种CC开源协议），你可以在其基础上进行修改，并出售自己设计的产品，但如果要使用Arduino这一标识，就需要支付一定的版权费用。



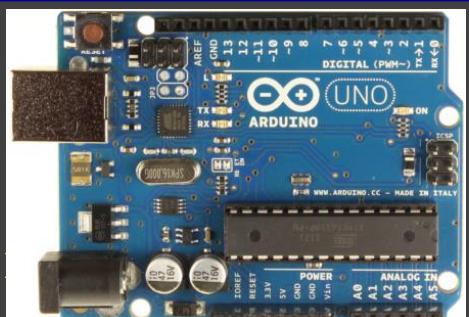
Arduino标识



Massimo Banzi

Arduino历史

Arduino源于意大目前Arduino的硬件已经演进出多种版本，包括简易的Arduino Uno、基于ARM芯片的Arduino Due、集成以太网和WiFi的Arduino Yún以及使用TI AM333x系列Cortex-A8芯片，功能强大的Arduino Tre等。



Arduino UNO



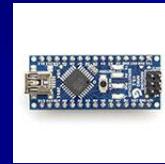
Arduino MEGA



Arduino Due



Yún
Tre



Arduino生态圈

除了核心的电路板以外，Arduino还有种类繁多的扩展功能模块。

Arduino具有标准的接口，许多主板都互相兼容，因此一个兼容Arduino的模块可以供不同型号的Arduino板使用，而且多个模块也可以叠加在同一块Arduino板上。

在某种意义上这种标准化接口和模块复用让Arduino成为电子积木，使用者像搭积木一样将实现不同功能的模块和电路板拼接在一起，这为Arduino建立了良好的生态圈，也吸引了众多爱好者投入到Arduino的开发行列中。



Arduino

Arduino生态圈

Arduino扩展功能模块囊括了各种应用领域，如：

- 光线、声音、温度、湿度等传感器模块
- 用于数据通信的GSM、GPRS、Bluetooth模块
- 用于测量距离与障碍物的红外、超声波模块
- 用于控制运动和姿态的电机、舵机控制模块

除了官方推出的Shield以外，很多第三方机构或个人也制作了个性鲜明的系列模块和扩展板。

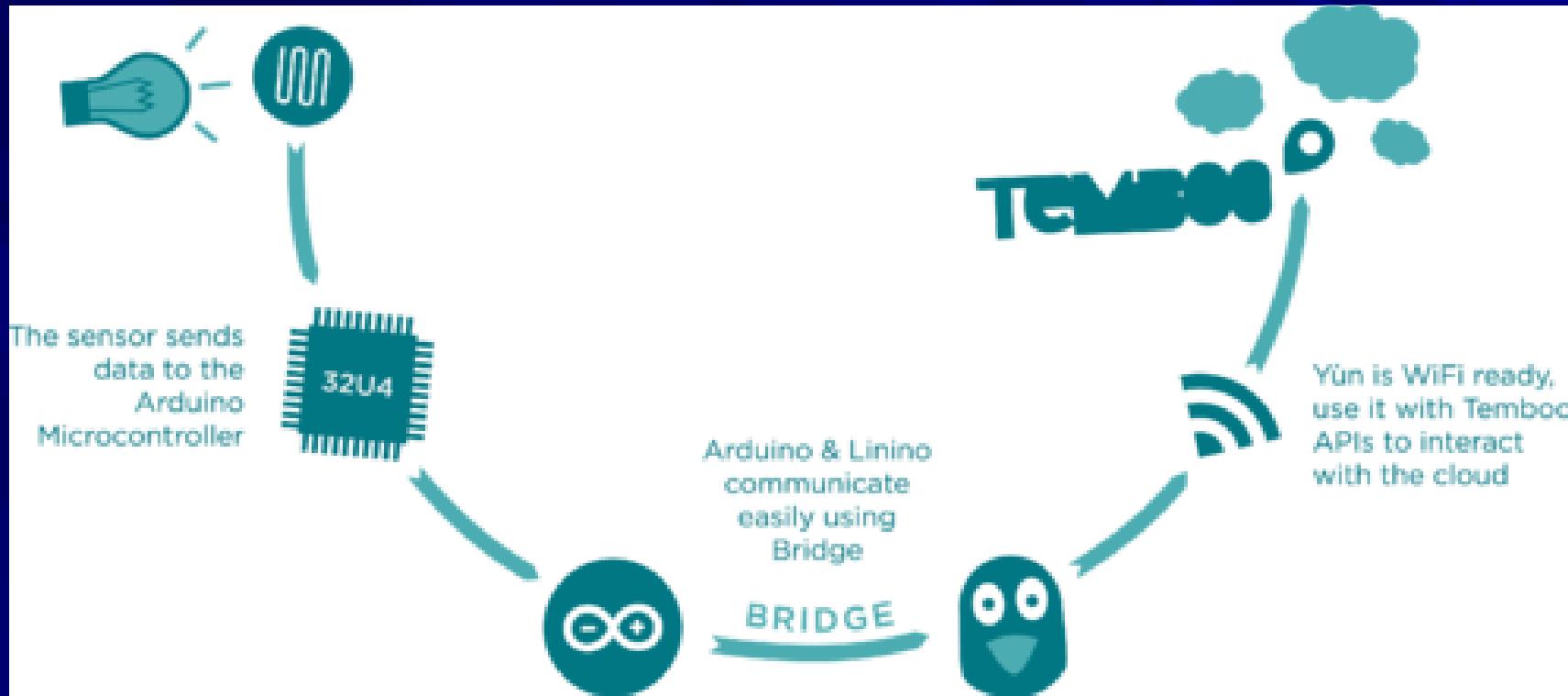
有如此丰富的模块来实现各种功能的组合，就不用发愁如何将灵感变成实物了。

例如，某位Maker想要为卧室的窗户做一个每天早上自动拉开的电动窗帘，而他的手里正好有一块Arduino板，那么他只需要一个用来连接电机的电机驱动模块，外加一个可以知道当前时间的时钟模块即可。但是另外一个Maker也许对电动窗帘不感兴趣，他似乎更关心当他外出后谁来给他的宠物狗喂食，那么好吧，只要将电机驱动模块拿掉，换上一个控制投食口的继电器模块，即可定时定量地向宠物狗的餐盆中投放食物。

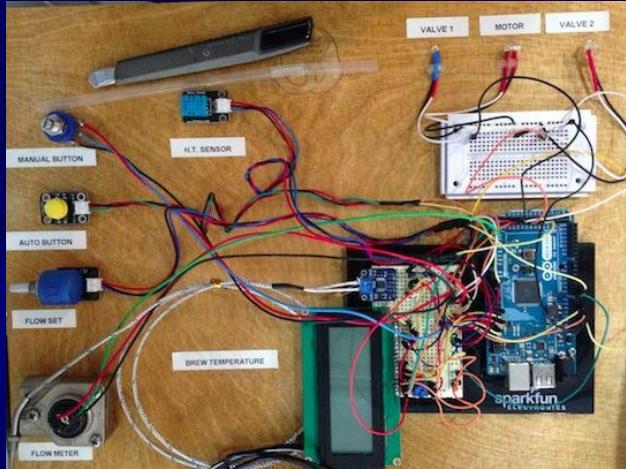


Arduino Shield

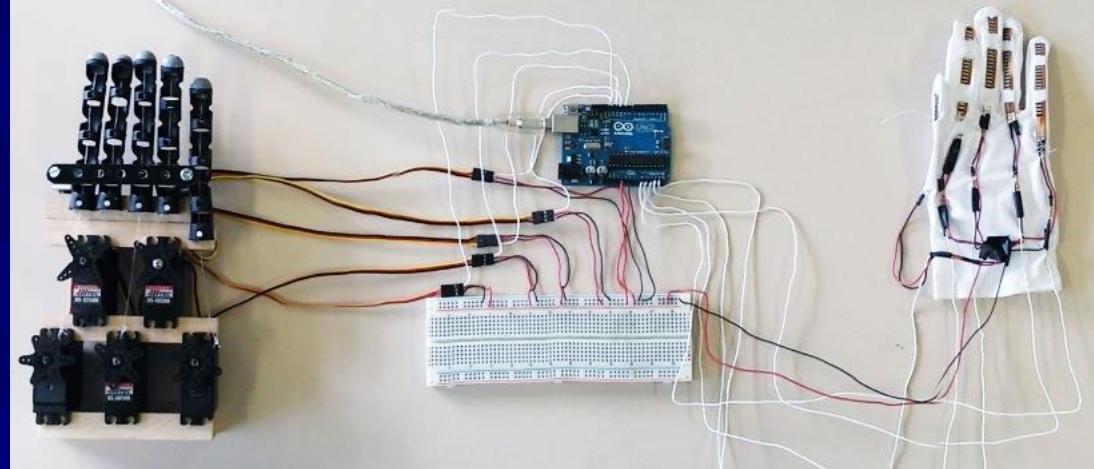
Arduino从创意到实现



Arduino创意



Arduino与模块

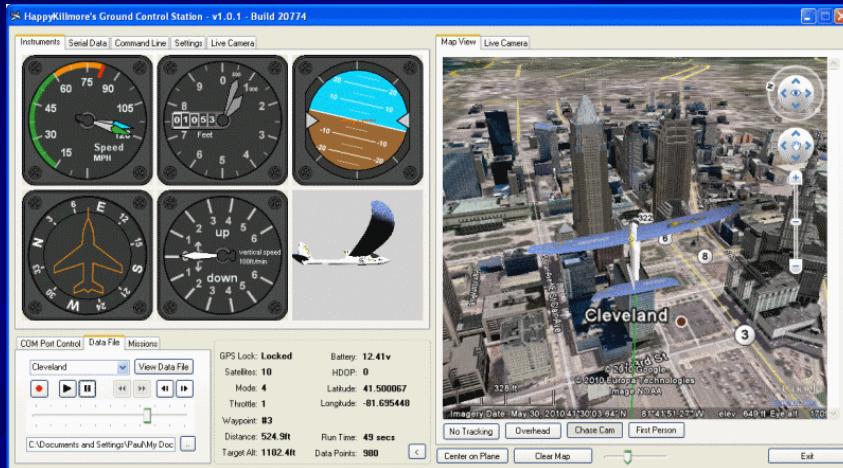


Arduino传感手臂

Arduino创意

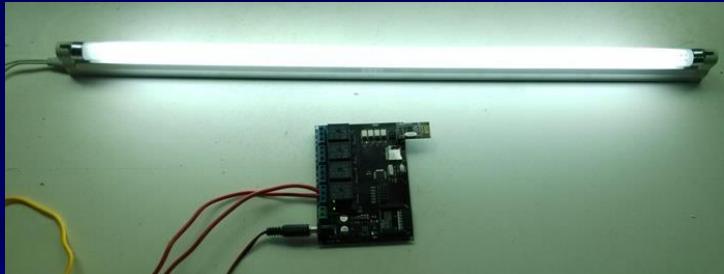


Arduino飞控系统

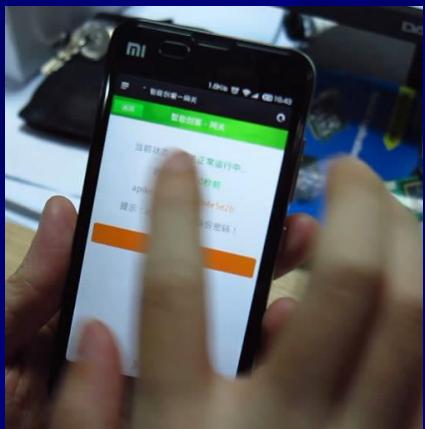


基于Arduino和LabView的
多路数据采集系统

Arduino创意

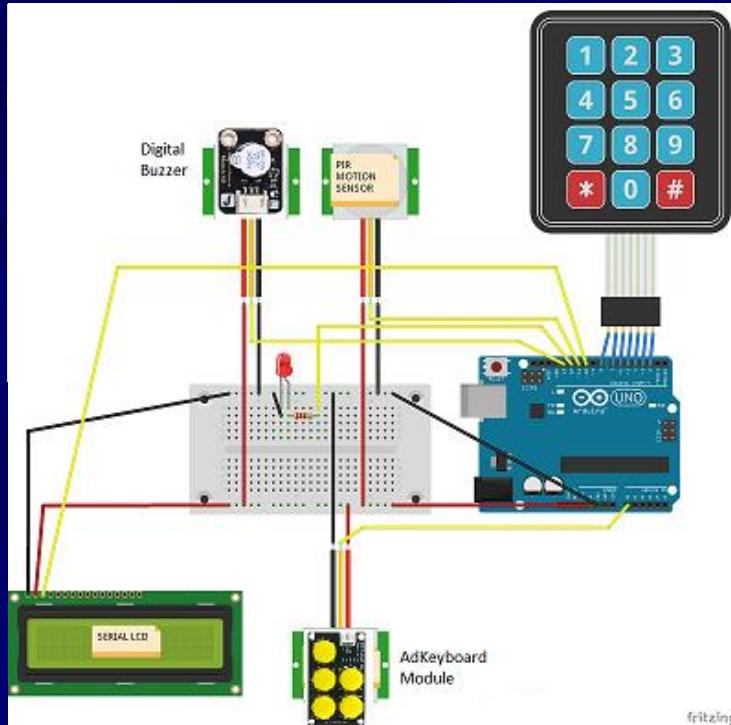


Arduino 智能家居



Arduino 智能家居手机终端

Arduino创意



利用Arduino制作的一个儿童教学计算器原型

需求

- 系统可以显示一个菜单，提供一些基本的操作：加、减、乘、除。
- 用户（我的女儿）可以用键盘从菜单上选择一种算数运算来学习。
- 会有一些难度级别：在选择运算后，难度级别会显示出来。
- 根据选择的难度级别，会随机显示出一些问题，用户可以用键盘回答这些问题。
- 用户可以在确认前修改自己的答案。
- 在确认答案后，根据正确与否会显示出一条信息。
- 如果三次答错，将会显示出正确答案。
- 用户可以浏览菜单（点开菜单并选择菜单项）。
- 系统应具有音频和视频警告API，错误信息可以通过该API发送。
- 每种算术运算有一个限时小测环节。
- 限时小测随机从简到难给出问题。
- 测试后会显示出统计数据（回答了多少题目，答对了多少题目）。
- 在用户接近时系统可以引起她的注意。
- 可以有一些数学以外的娱乐环节，比如让她唱首歌、亲自己的爸爸等等。
- 警告API可以用来在娱乐环节做一些有趣的事情。

Arduino创意

Arduino蒸汽朋克意大利浓咖啡机



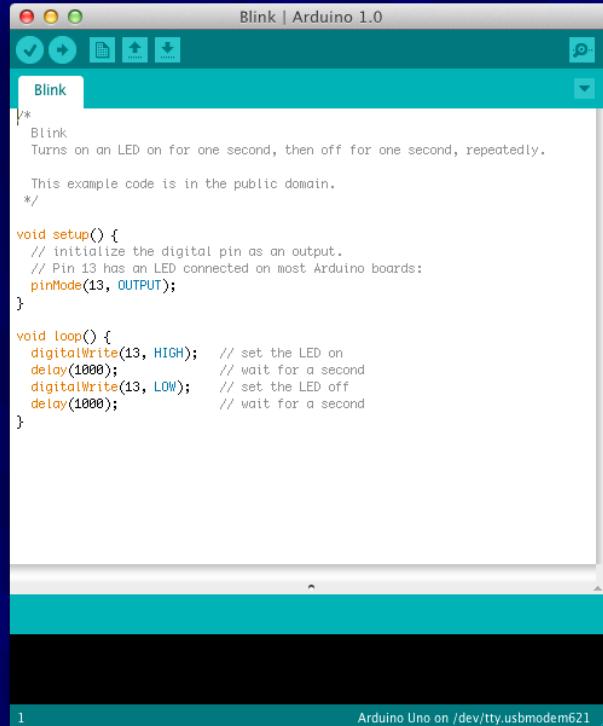
Arduino编程环境

Arduino的编程语言和开发环境基于Wiring和Processing架构。Processing包含了发展自Java的编程语言和开发环境，而Wiring是用于微控制器的一种开源编程框架，可以让用户编写跨平台的程序，这些代码无需修改就可以在不同的微控制器和电路板上运行。

Arduino的开发环境中集成了很多代码库，通过使用库文件Maker（创客）可以绕开“如何学习使用不熟悉的外围模块”这一令人头疼的问题。

比如当你需要为自己的工程添加一个点阵式的LCD显示器，用来显示一些数据和菜单，只要购买或自制一块LCD Shield，然后在IDE中添加LiquidCrystal库，就可以通过几行简单的代码在LCD上显示文字。

Arduino可以让没有太多编程基础的人做出很酷的东西，这或许这是它的魅力所在。



The screenshot shows the Arduino IDE interface with the title bar "Blink | Arduino 1.0". The main window displays the "Blink" example sketch. The code is as follows:

```
/*
 * Blink
 * Turns on an LED on for one second, then off for one second, repeatedly.
 *
 * This example code is in the public domain.
 */
void setup() {
    // initialize the digital pin as an output.
    // Pin 13 has an LED connected on most Arduino boards:
    pinMode(13, OUTPUT);
}

void loop() {
    digitalWrite(13, HIGH); // set the LED on
    delay(1000);           // wait for a second
    digitalWrite(13, LOW);  // set the LED off
    delay(1000);           // wait for a second
}
```

The status bar at the bottom indicates "1" and "Arduino Uno on /dev/tty.usbmodem621".

Arduino IDE

Arduino编程环境

•Arduino IDE的安装

1.可以在官方网站 <http://arduino.cc/> 上下载Arduino IDE，并且针对Windows操作系统Arduino官方还提供了免安装版本。选择“Windows (ZIP file)”下载Arduino IDE压缩包。

2.解压下载好的zip文件，无需进行安装，直接双击arduino.exe运行Arduino IDE。

The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** arduino_st7920 | Arduino 1.0
- File Menu:** File Edit Sketch Tools Help
- Sketch Editor:** The code editor contains C++ code for an ST7920 display. It includes includes for `<avr/pgmspace.h>`, defines for PROGMEM, and functions for setup and loop. It also includes calls to `ST7920_api_init()` and `ST7920_api_display_strings()`.
- Output Window:** Shows the message "Done compiling."
- Status Bar:** Binary sketch size: 1920 bytes (of a 32256 byte maximum)
- Bottom Bar:** Arduino Uno on COM1

Arduino IDE

Arduino编程环境

•Arduino IDE的使用

IDE界面主要分为三部分：

- 上方的菜单栏和工具栏
- 中间的代码编辑器
- 以及下方的调试信息栏和状态栏



菜单栏包含了Arduino IDE的全部功能选项。

File菜单用于文件操作，如新建Sketch或打开实例；

Edit菜单包含文本编辑和查找等功能；

Sketch菜单包括编译和导入函数库选项；

Tools菜单用于对目标板类型、编程器和串口进行选择和操作。

菜单栏和工具栏



The screenshot shows the Arduino IDE interface. The title bar reads "arduino_st7920 | Arduino 1.0". The menu bar includes File, Edit, Sketch, Tools, and Help. The main area is a code editor with the following pseudocode:

```
#include <avr/pgmspace.h>
PROGMEM prog_uchar str1[] = "您好！我是是";
PROGMEM prog_uchar str2[] = "Arduino";
PROGMEM prog_uchar str3[] = "爱好者";
PROGMEM prog_uchar str4[] = "希望能帮到您。";
//PROGMEM prog_uchar pic1[1024] = {0};
//PROGMEM prog_uchar pic2[1024];
//.....pic3[1024];//
```

The code includes sections for setup and loop. Below the code editor, a status bar displays "Done compiling." and "Binary sketch size: 1920 bytes (of a 32256 byte maximum)". The bottom right corner shows "Arduino Uno on COM1".

代码编辑器

调试信息栏和状态栏

Arduino IDE

通常Arduino程序不用Program表示，习惯上称为Sketch，中文意为画或草图。这是因为Arduino的编程语言是基于Processing的，而Processing的设计针对的就是视觉艺术家而不是传统意义上的程序员。

Arduino编程环境

• Arduino IDE的使用



相比菜单栏来说，工具栏显得十分简洁，总共有6个按钮。
从左至右依次为“校验”、“上传”、“新建”、“打开”、“保存”和“串口监视器”按钮。

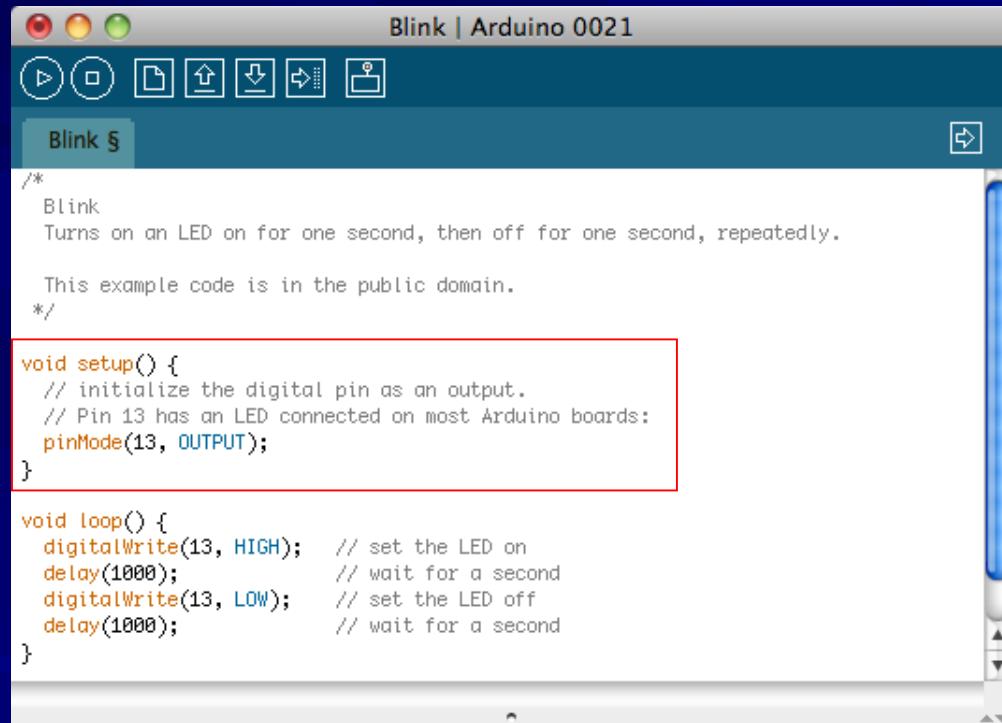
校验 (Verify)：通过使用编译器编译程序来检查代码的语法错误；

上传 (Upload)：即通常所说的下载或烧写功能，用于更新Arduino板中的程序；

串口监视器 (Serial Monitor)：打开一个串口终端窗口，通过这个窗口可以使用串口与Arduino通信。

Arduino程序结构

· 初始化：setup()函数



```
/*
Blink
Turns on an LED on for one second, then off for one second, repeatedly.

This example code is in the public domain.
*/

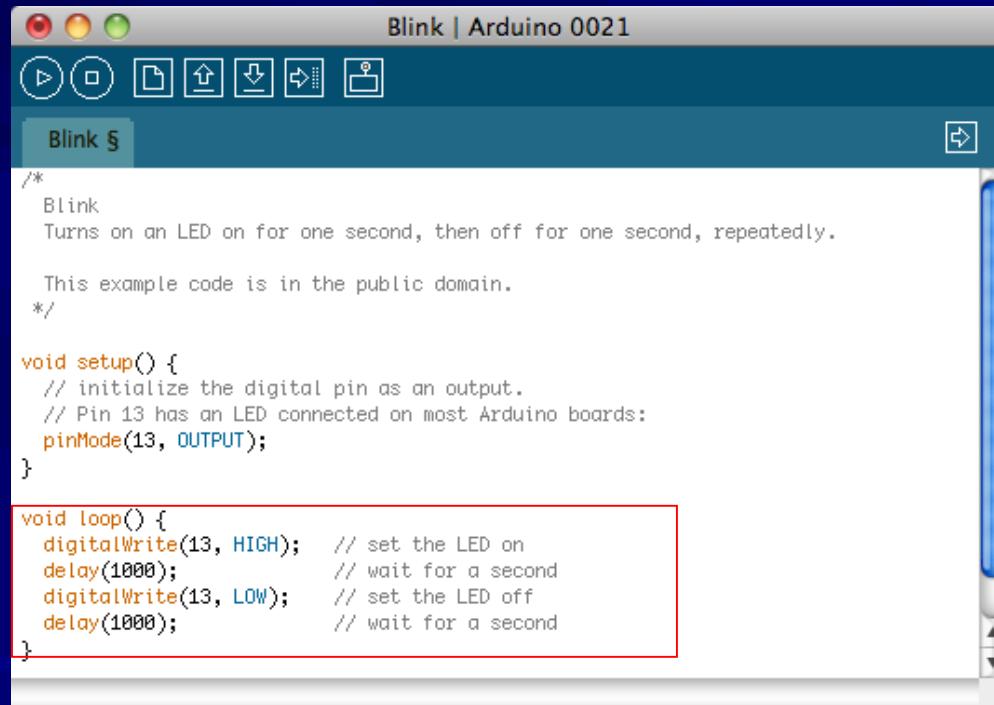
void setup() {
    // initialize the digital pin as an output.
    // Pin 13 has an LED connected on most Arduino boards:
    pinMode(13, OUTPUT);
}

void loop() {
    digitalWrite(13, HIGH);    // set the LED on
    delay(1000);              // wait for a second
    digitalWrite(13, LOW);     // set the LED off
    delay(1000);              // wait for a second
}
```

上电或复位后setup()
函数执行一次

Arduino程序结构

· 主体 : **loop()** 函数



```
Blink | Arduino 0021

Blink §

/*
Blink
Turns on an LED on for one second, then off for one second, repeatedly.

This example code is in the public domain.
*/

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);    // set the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);     // set the LED off
  delay(1000);              // wait for a second
}
```

loop()函数一直循环运行

Hello World , Arduino

开始你的第一个Arduino Sketch :

初始化串口Serial.begin(9600);

Serial.println() 通过串口输出字符串"Hello World,Arduino."

delay(500) 延时500ms

The screenshot shows the Arduino IDE interface with the following code:

```
Uart§
// Uart.ino

void setup() {
    // 初始化串口
    Serial.begin(9600);
}

void loop() {
    Serial.println("Hello World,Arduino.");
    delay(500);
}
```

Hello World , Arduino

开始你的第一个Arduino Sketch :

➤ Serial.begin()

描述：设置串行通信波特率(bit/s)。

语法：Serial.begin(speed)

参数：speed: 位/秒 (波特) - long

➤ Serial.println()

描述：打印数据到串行端口，输出可识别的ASCII码文本并回车。

语法：Serial.println(val)

Serial.println(val, format)

参数：val: 打印的内容

format: 指定基数 (整数数据类型)

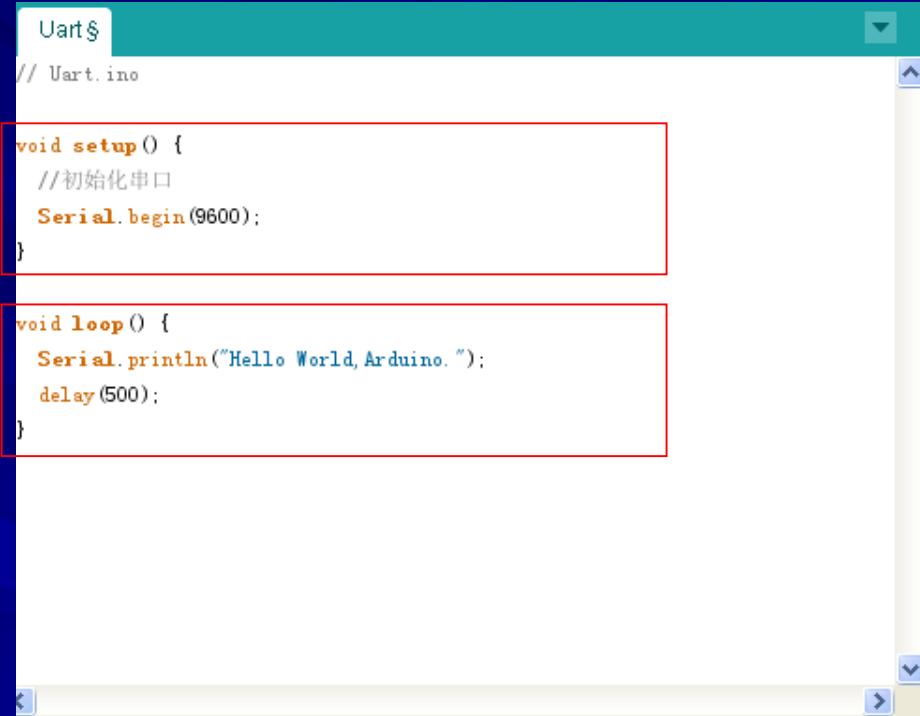
或小数位数 (浮点类型)

➤ delay()

描述：延时函数，使程序等待一段时间，单位为毫秒。

语法：delay(ms)

参数：ms :暂停的毫秒数



```
Uart§
// Uart.ino

void setup() {
    //初始化串口
    Serial.begin(9600);
}

void loop() {
    Serial.println("Hello World, Arduino.");
    delay(500);
}
```

更多Arduino语法请参考：
<http://arduino.cc/en/Reference/>

Hello World , Arduino

开始你的第一个Arduino Sketch :

➤ 选择开发板类型

菜单栏Tools(工具)→Board(板)

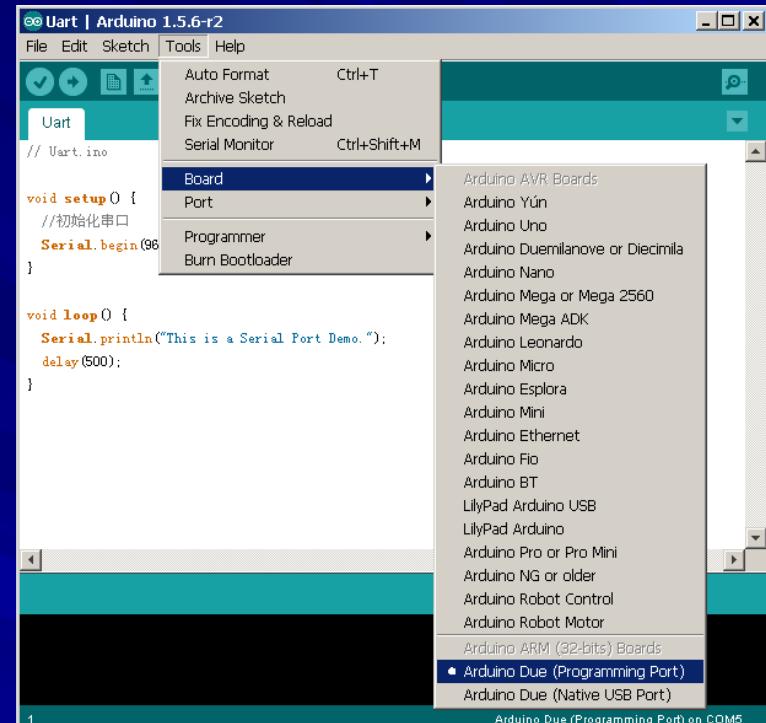
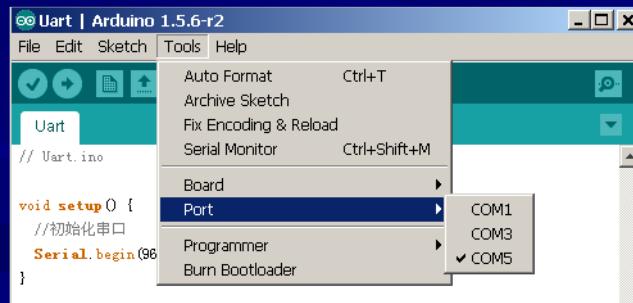
选择相应的Arduino板类型

Arduino Due (Programming Port)

➤ 选择端口

将Mini USB线的一端与Arduino模块板的DEBUG接口相连，另一端与PC计算机USB接口相连。

点击Arduino IDE的Tools菜单，将 Serial Port设为设备管理器中的Arduino Due Programming Port (COMx)对应的端口。



Hello World , Arduino

开始你的第一个Arduino Sketch :

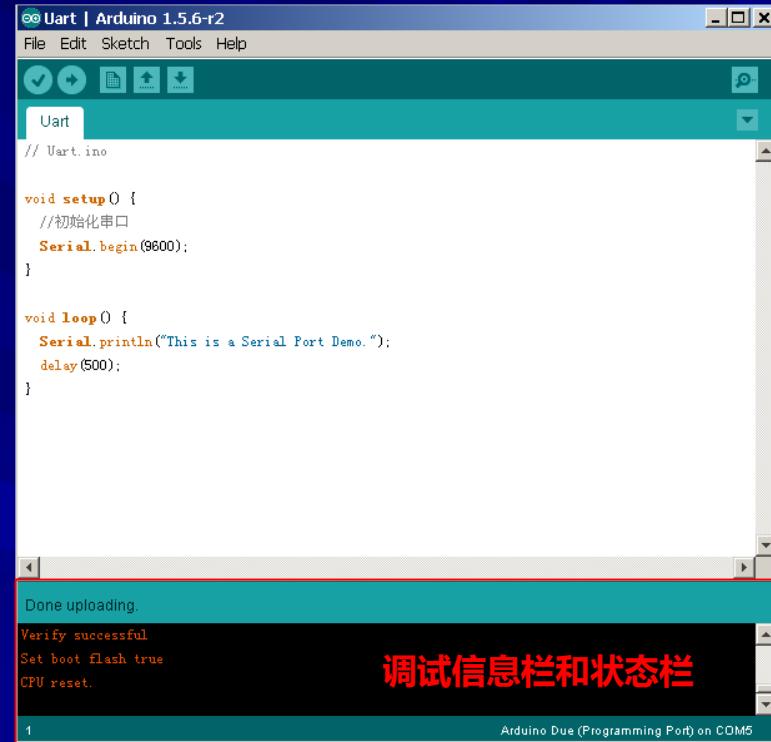
➤ 编译上传

点击Arduino IDE的File(菜单)→Upload(上传)或者工具栏上的按钮 ，即可完成程序的编译和上传

➤ 调试

可以在调试信息栏和状态栏查看编译和上传的状态

你可以使用你熟悉的串口调试工具来查看程序运行的结果，也可以使用IDE自带的串口监视器 



The screenshot shows the Arduino IDE interface with the title bar "Uart | Arduino 1.5.6-r2". The code editor contains the following sketch:

```
// Uart.ino

void setup() {
  //初始化串口
  Serial.begin(9600);
}

void loop() {
  Serial.println("This is a Serial Port Demo.");
  delay(500);
}
```

The status bar at the bottom displays the message "Done uploading." followed by "Verify successful", "Set boot flash true", and "CPU reset.". A red box highlights this status information with the text "调试信息栏和状态栏" (Debugging Information Bar and Status Bar).

Arduino程序语言

Arduino语言是建立在C/C++基础上的，其实也就是基础的C语言，Arduino语言只不过把单片机（微控制器）相关的一些参数设置都函数化，不用我们去了解他的底层，让不了解单片机（微控制器）编程的创客也能轻松上手。

关键字：

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

语法符号：

- ;
- {}
- //
- /* */

运算符：

- =
- +
- -
- *
- /
- %
- ==
- !=
- <
- >
- <=
- >=
- &&
- ||
- !
- ++
- --
- +=
- -=
- *=
- /=

常量：

- HIGH | LOW
表示数字IO口的电平，HIGH 表示高电平，LOW 表示低电平。
- INPUT | OUTPUT
表示数字IO口的方向，INPUT 表示输入（高阻态），OUTPUT 表示输出。
- true | false
true 表示真（1），false 表示假（0）

Arduino基本函数

- 数字I/O
 - pinMode(pin,mode)
 - digitalWrite(pin,value)
 - digitalRead(pin)
- 模拟I/O
 - analogRead
 - analogWrite
- 高级I/O
- 时间函数
- 数学库
- 随机数

- **pinMode**
 - 用以配置引脚为输出或输出模式，它是一个无返回值函数，函数有两个参数**pin**和**mode**，**pin**参数表示所要配置的引脚，**mode**参数表示设置的模式—**INPUT**（输入）或**OUTPUT**（输出）。

pinMode(pin, mode)函数原型：

```
• void pinMode(uint8_t pin, uint8_t mode)
{
    uint8_t bit = digitalPinToBitMask(pin);
    uint8_t port = digitalPinToPort(pin);
    volatile uint8_t *reg;
    if (port == NOT_A_PIN)
        return;
    reg = portModeRegister(port);
    if (mode == INPUT)
    {
        uint8_t oldSREG = SREG;
        cli();
        *reg &= ~bit;
        SREG = oldSREG;
    }
    else
    {
        uint8_t oldSREG = SREG;
        cli();
        *reg |= bit;
        SREG = oldSREG;
    }
}
```

digitalWrite

- `digitalWrite`函数也是在Blink程序中见到过的，它的作用是设置引脚的输出的电压为高电平或低电平。该函数也是一个无返回值的函数，函数有两个参数`pin`和`value`，`pin`参数表示所要设置的引脚，`value`参数表示输出的电压—`HIGH`（高电平）或`LOW`（低电平）。

digitalWrite (pin,value) 函数模型

```
• void digitalWrite(uint8_t pin, uint8_t val)
{
    uint8_t timer = digitalPinToTimer(pin);
    uint8_t bit = digitalPinToBitMask(pin);
    uint8_t port = digitalPinToPort(pin);
    volatile uint8_t *out;
    if (port == NOT_A_PIN) return;
    // If the pin that support PWM output, we need to turn it off
    // before doing a digital write.
    if (timer != NOT_ON_TIMER) turnOffPWM(timer);
    out = portOutputRegister(port);
    if (val == LOW)
    {
        uint8_t oldSREG = SREG;
        cli();
        *out &= ~bit;
        SREG = oldSREG;
    }
    else
    {
        uint8_t oldSREG = SREG;
        cli();
        *out |= bit;
        SREG = oldSREG;
    }
}
```

digitalRead

- `digitalRead`函数用在引脚为输入的情况下，可以获取引脚的电压情况—**HIGH**（高电平）或**LOW**（低电平），参数`pin`表示所要获取电压值的引脚，该函数返回值为**int**型，表示引脚的电压情况。

digitalRead (pin)的函数原型

- int digitalRead(uint8_t pin)
 - {
 - uint8_t timer = digitalPinToTimer(pin);
 - uint8_t bit = digitalPinToBitMask(pin);
 - uint8_t port = digitalPinToPort(pin);
 - if (port == NOT_A_PIN) return LOW;
 - // If the pin that support PWM output, we
need to turn it off
 - // before getting a digital reading.
 - if (timer != NOT_ON_TIMER)
turnOffPWM(timer);
 - if (*portInputRegister(port) & bit) return HIGH;
return LOW;
 - }

analogWrite(pin,value)

- `analogWrite`函数为无返回值函数，有两个参数`pin`和`value`,参数`pin`表示所要设置的引脚，只能选择函数支持的引脚；参数`value`表示PWM输出的占空比，范围在0~255的区间，对应的占空比为0%~100%。

analogWrite(pin.value)函数原型

- int ledPin = 9; // LED connected to digital pin 9
int analogPin = 3; // potentiometer connected to analog pin 3
int val = 0; // variable to store the read value

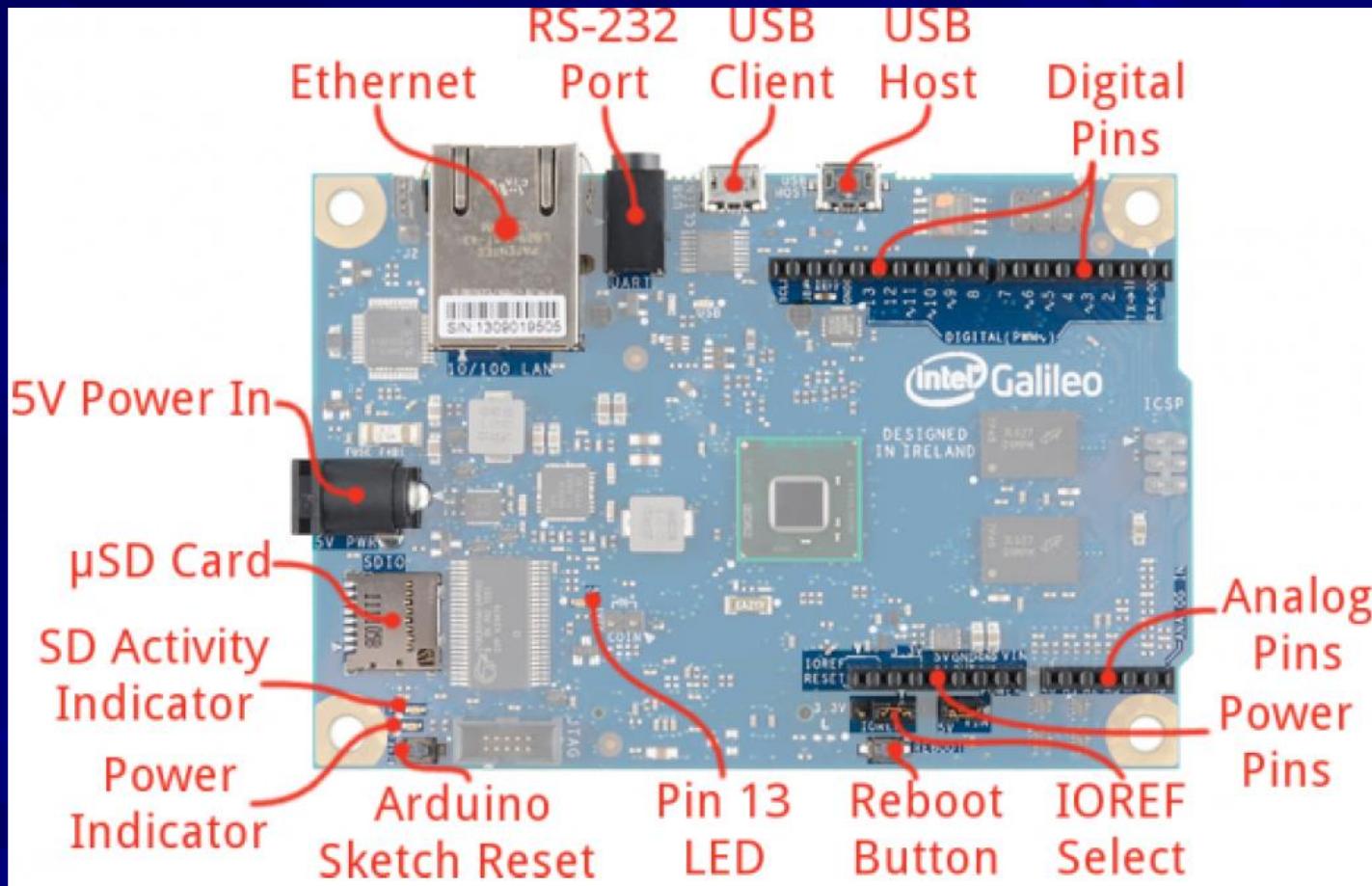
```
void setup()  
{  
    pinMode(ledPin, OUTPUT); // sets the pin as output  
}
```

```
void loop()  
{  
    val = analogRead(analogPin); // read the input pin  
    analogWrite(ledPin, val / 4); // analogRead values go from 0 to  
    // 1023, analogWrite values from 0 to 255  
}
```

Introduction to Galileo

- The Intel® Galileo board is based on the Intel® Quark SoC X1000, a 32-bit Intel Pentium®-class system on a chip (SoC). It is the first board based on Intel® architecture designed to be hardware and software pin-compatible with shields designed for the Arduino Uno R3
- The Galileo board is also software-compatible with the Arduino Software Development Environment, which makes getting started a snap.

Board overview



Features

- Shield Compatibility
- Familiar IDE
- Ethernet Library Compatibility
- Real Time Clock (Battery not included)
- Works with PCI Express Mini Cards
- USB Host Port
- MicroSD Support
- TWI/I2C, SPI Support
- Serial Connectivity
- Linux on Board

Intel Galileo Board Characteristics

■ Physical Characteristics

- 10 cm long and 7 cm wide with the USB connectors
- UART jack
- Ethernet connector
- Four screw holes
- Reset button for sketch

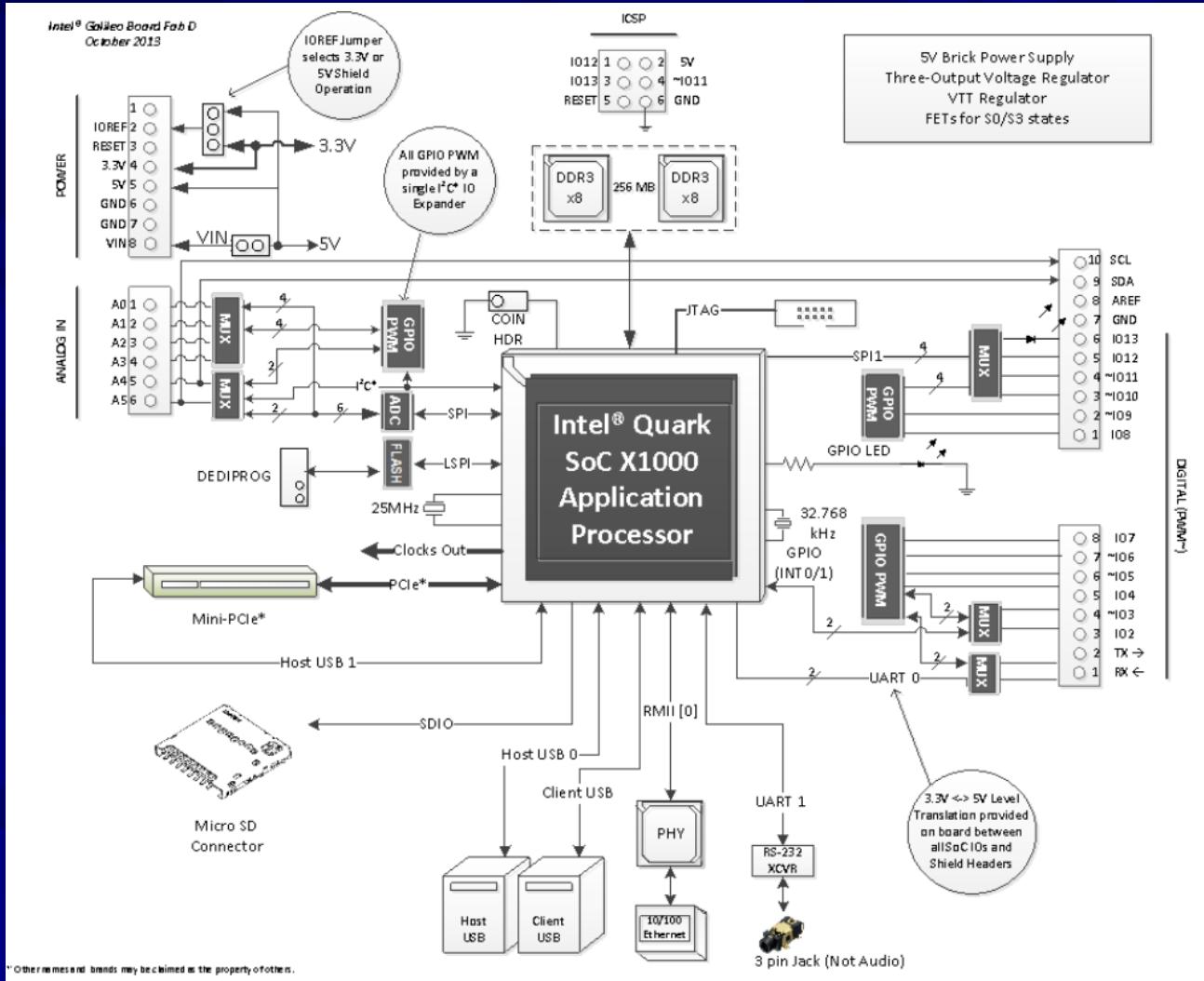
■ Processor Features

- 400 MHz 32-bit Intel® with 512K on-die SRAM, 800 MTs Memory Speed
- TDP 1.9W – 2.2W (depending on VR), 15mm x 15mm
- Pentium® instruction set architecture (ISA) compatible processor
- Supports Arduino shields
- Integrated Real Time Clock (RTC), with optional 3V “coin cell” battery
- Operating temps from 0 to 70 degrees C (commercial) and more variants soon

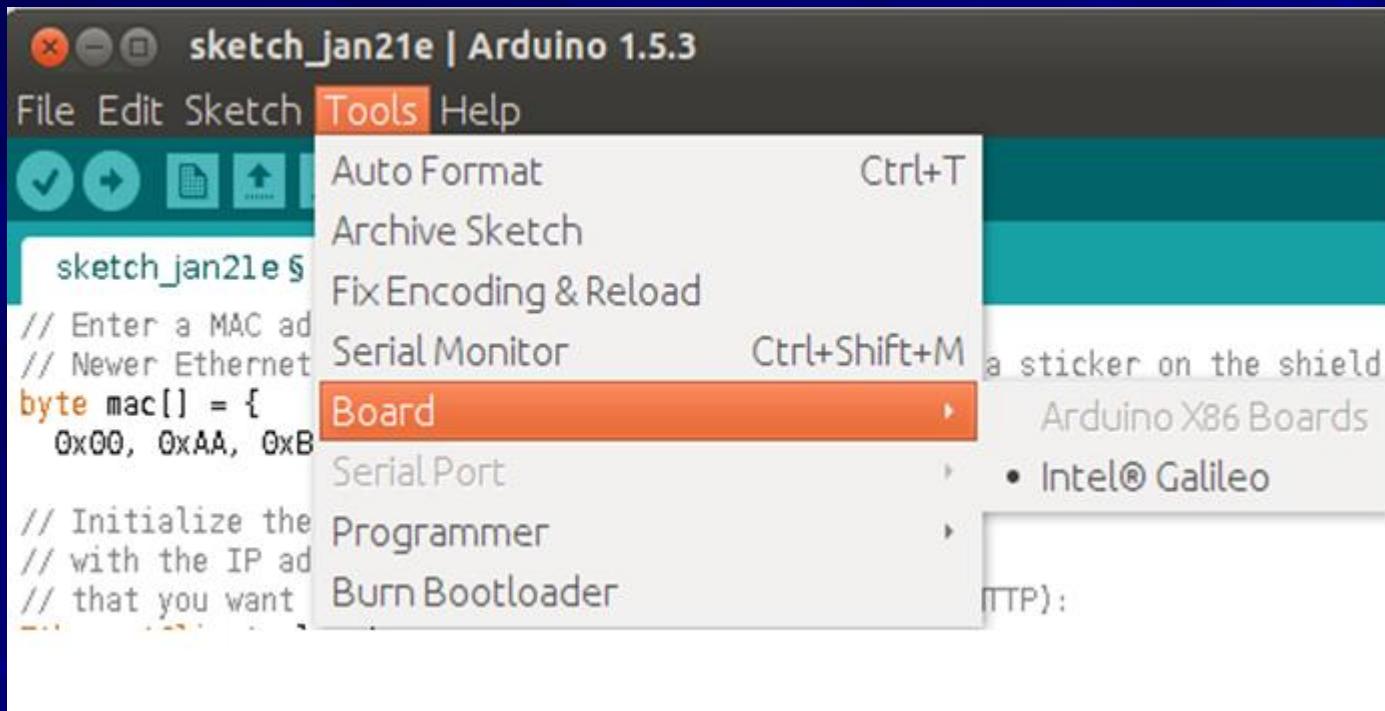
■ Storage Options

- 8 MByte Legacy SPI Flash to store firmware (bootloader) and the latest sketch
 - Between 256 KByte and 512 KByte dedicated for sketch storage
 - 256 MByte DRAM
 - Optional micro SD card offers up to 32GByte of storage
 - USB storage works with any USB 2.0 compatible drive
 - 11 KByte EEPROM programmed via the EEPROM library

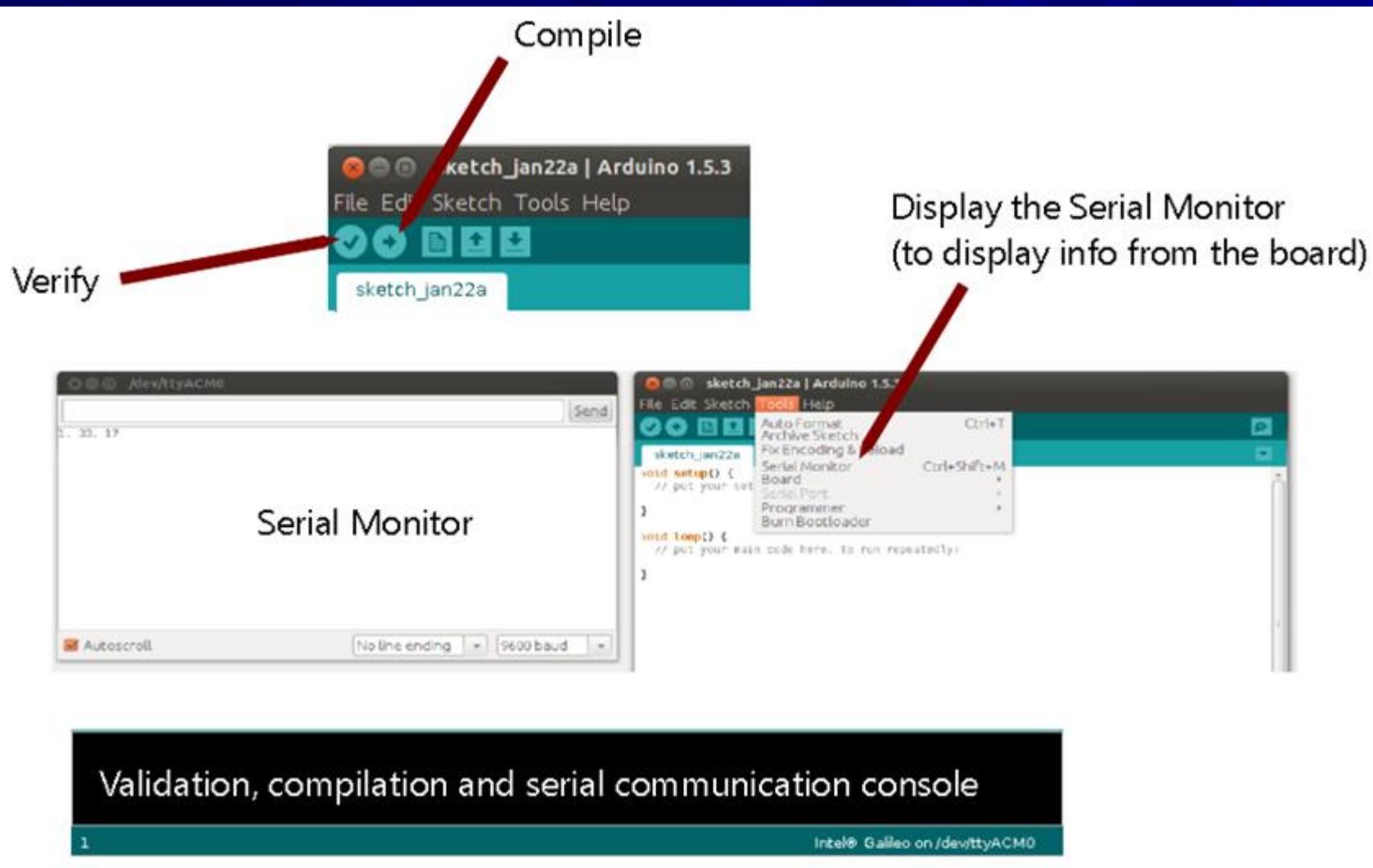
Intel Galileo Block Diagram



Development Environment



Development Environment



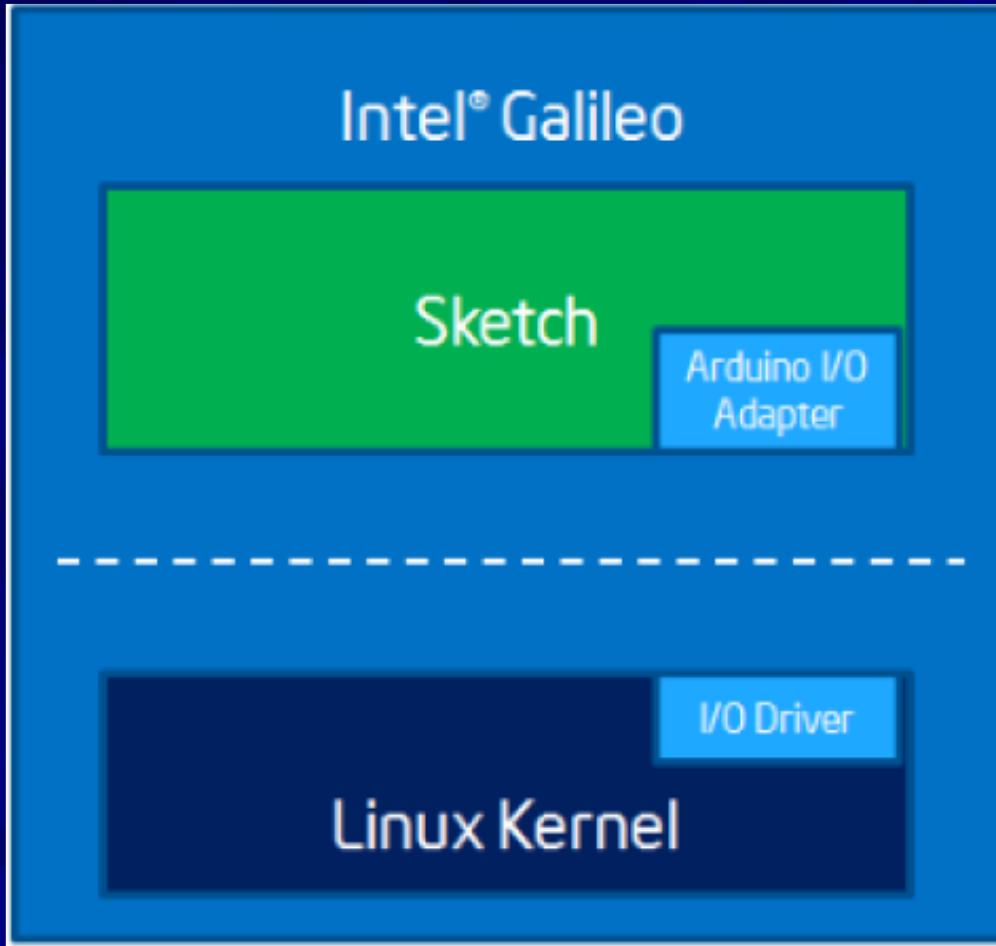
HelloWorld with Arduino IDE

```
/ put your setup code here, to run once
void setup() {
    // initialize serial communication at 9600 bits per second
    Serial.begin(9600);
    // sending characters on serial port (visible if you display the serial
    // monitor)
    Serial.println("Setup says: Hello World!");
}

// put your main code here, to run repeatedly
void loop() {
    //sending characters the same way as setup function
    Serial.println("Loop says: Hello World!");
    //waiting a second
    delay(1000);
}
```

Galileo Dev Board I/O Control Based on Linux

Arduino Sketches and the Linux



Galileo I/O Mapping

Galileo I/O Mappings

Arduino IDE ID	GPIO			PWM Linux	Int	Dir	Muxed with	Initial Setup
	Source	Pin	Linux					
I00	Cypr	GPORT4_BIT6_PWM2	50	N/A	-	BI	UART0_RXD	I w/ pullup off
I01	Cypr	GPORT4_BIT7_PWM0	51	N/A	-	BI	UART0_TXD	I w/ pullup off
I02	SoC (Cypr)	GPIO<6> (GPORT0_BIT4_PWM7)	14 (32*)	-	0	BI	-	I w/ pullup off
I03	SoC (Cypr)	GPIO<7> (GPORT0_BIT2_PWM3)	15 (18*)	3	1	BI	(PWM)	I w/ pullup off
I04	Cypr	GPORT1_BIT4_PWM6	28		-	BI	-	I w/ pullup off
I05	Cypr	GPORT0_BIT1_PWM5	17	5	-	BI	(PWM)	I w/ pullup off
I06	Cypr	GPORT1_BIT0_PWM6	24	6	-	BI	(PWM)	I w/ pullup off
I07	Cypr	GPORT1_BIT3_PWM0	27		-	BI	-	I w/ pullup off
I08	Cypr	GPORT1_BIT2_PWM2	26		-	BI	-	I w/ pullup off
I09	Cypr	GPORT0_BIT3_PWM1	19	1	-	BI	(PWM)	I w/ pullup off
I010	Cypr	GPORT0_BIT0_PWM7	16	7	-	BI	(PWM) SPI1_SS_B	I w/ pullup off
I011	Cypr	GPORT1_BIT1_PWM4	25	4	-	BI	(PWM) SPI1_MOSI	I w/ pullup off
I012	Cypr	GPORT3_BIT2_PWM3	38		-	BI	SPI1_MISO	I w/ pullup off
I013	Cypr	GPORT3_BIT3_PWM1	39		-	BI	SPI1_SCK	I w/ pullup off
I014	Cypr	GPORT4_BIT0_PWM6	44		-	BI	AD7298:VINO	I w/ pullup off
I015	Cypr	GPORT4_BIT1_PWM4	45		-	BI	AD7298:VIN1	I w/ pullup off
I016	Cypr	GPORT4_BIT2_PWM2	46		-	BI	AD7298:VIN2	I w/ pullup off
I017	Cypr	GPORT4_BIT3_PWM0	47		-	BI	AD7298:VIN3	I w/ pullup off
I018	Cypr	GPORT4_BIT4_PWM6	48		-	BI	AD7298:VIN4	I w/ pullup off
I019	Cypr	GPORT4_BIT5_PWM4	49		-	BI	AD7298:VIN5	I w/ pullup off

* See Galileo I/O Function Muxing table below.

Galileo I/O Function Muxing

Galileo I/O Function Muxing

Mux Selector		Cypress GPIO pin	Linux GPIO ID	Dir	Initial Setup
0	1				
UART0_RXD	IO0	GPORT3_BIT4_PWM7	40	0	unknown
UART0_TXD	IO1	GPORT3_BIT5_PWM5	41	0	unknown
SPI1_SS_B	IO10	GPORT3_BIT6_PWM3	42	0	unknown
SPI1_MOSI	IO11	GPORT3_BIT7_PWM1	43	0	unknown
SPI1_MISO	IO12	GPORT5_BIT2_PWM3	54	0	unknown
SPI1_SCK	IO13	GPORT5_BIT3_PWM1	55	0	unknown
AD7298:VIN0	IO14	GPORT3_BIT1_PWM5	37	0	0
AD7298:VIN1	IO15	GPORT3_BIT2_PWM3	36	0	0
AD7298:VIN2	IO16	GPORT0_BIT7_PWM1	23	0	0
AD7298:VIN3	IO17	GPORT0_BIT6_PWM3	22	0	0
AD7298:VIN4	IO18	GPORT0_BIT5_PWM5	21	0	0
AD7298:VIN5	IO19	GPORT0_BIT4_PWM7	20	0	0
IO2 via SoC GPIO<6>	IO2 via Cypress GPORT0_BIT4_PWM7	GPORT1_BIT7_PWM0	31	0	unknown
IO3 via SoC GPIO<7>	IO3 via Cypress GPORT0_BIT2_PWM3	GPORT1_BIT6_PWM2	30	0	unknown
I2C	(AD7298:VIN4 or IO18) and (AD7298:VIN5 or IO19)	GPORT1_BIT5_PWM4	29	0	1

Steps

1. 配置电脑的IP地址（注意是网口，不是无线网卡的）。
2. 配置Galileo开发板的IP地址
3. 用putty通过telnet连接电脑与Galileo
4. 用Linux语言对于Galileo板子进行编程(shell,GCC etc)

配置Galileo开发板的IP地址

1、用Arduino开发环境编写图示代码；

(Arduino驱动配置在/arduino-1.5.3/hardware/arduino/x86/tools;

Arduino出现闪退现象，可以通过把系统语言改为英语解决。)

2、在代码upload到Galileo板子上，便完成了IP的配置；



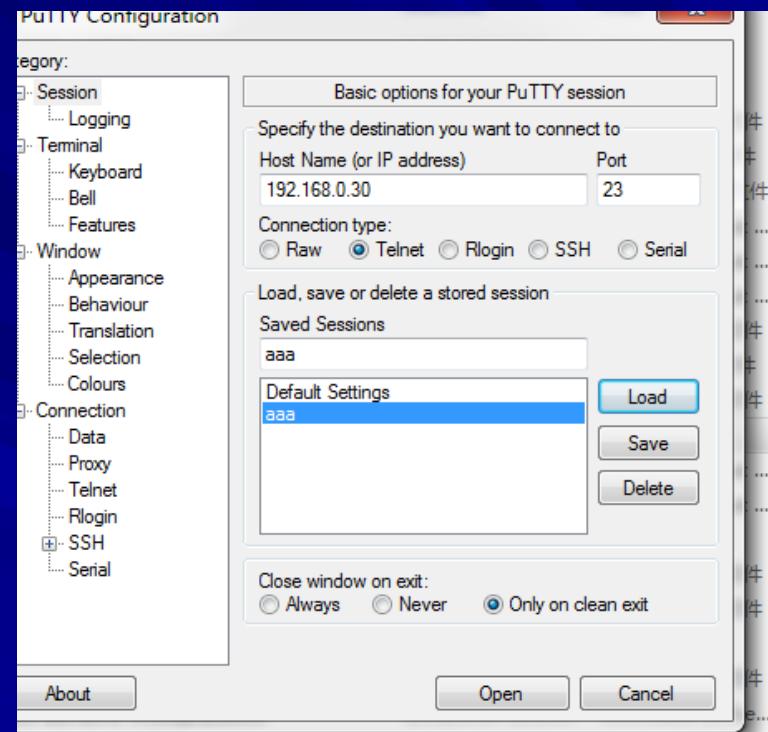
```
sketch_may27b

void setup() {
  // put your setup code here, to run once:
  system("telnetd -l /bin/sh");
  system("ifconfig eth0 192.168.0.30 netmask 255.255.255.0 up");
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

用putty通过telnet连接电脑与Galileo

- 1、下载putty;
- 2、打开putty，如图所示；
- 3、配置Host Name， Host Name为电脑的IP地址； Connection type 选择 Telnet ；
- 4、如果要直接使用，跳过这个步骤。可以选择在**save section** 栏里输入名称，然后保存，方便下次使用；下次使用时点击该名称，然后按load;
- 5、按open按键,就进入Galileo板子的Linux系统。



1、建立I/O口可执行文件：

```
echo -n "24" > /sys/class/gpio/export; //24对应的是Arduino的I/O口6
```

2、设置I/O口方向（输入in 或输出out）：

```
echo -n "out" > /sys/class/gpio/gpio24/direction;
```

3、设置I/O口的电平（输入“1”或“0”）：

```
echo -n "1" > /sys/class/gpio/gpio24/value ;
```

Linux PWM控制

1、PWM配置

```
echo -n "6" > /sys/class/pwm/pwmchip0/export;
```

2、开启PWM功能(开启写“1”，关闭写“0”)

```
echo -n "1" > /sys/class/pwm/pwmchip0/pwm6/enable;
```

3、设置周期(以ns为单位)

```
echo -n "1000000" > /sys/class/pwm/pwmchip0/pwm6/period;
```

4、设置占空比

```
echo -n "500000" > /sys/class/pwm/pwmchip0/pwm6/duty_cycle;
```

参考文献

<http://www.malinov.com/Home/sergey-s-blog/>

<https://communities.intel.com/content>