寻迹感光智能车使用手册

寻迹/感光智能车

ASURO-Car 使用说明书

V1.0 – 2008.3

北京亿学通电子
北京海淀上地七街国际创业园2号院C座812
TEL: 86-10-62669059 *E-mail:fae_61mcu@163.com* *http://www.61mcu.com*

介绍

ASURO 是一个完全使用 C 语言编程的可运动微型机器人，它是德国太空署属下一个负责设计机器人与机械装置的机构 DLR 为了教育用途而开发的。有经验的电子技术人员甚至是初学者均能完成 ASURO 的组装。除了印刷电路板外，该装置完全使用标准元件构成。

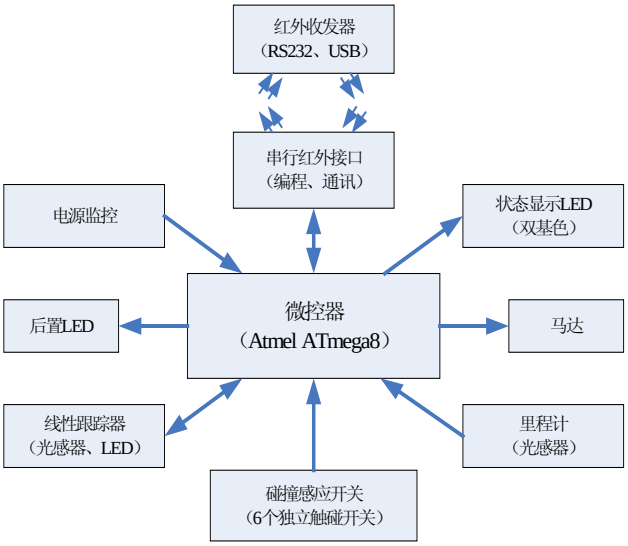
ASURO 仅使用免费软件完成全部编程，因此 ASURO 非常适合用于微控器装置入门学习，可用作大学或中学的课程设计与研究或是专业技术培训中心的教学。ASURO 在整个硬件的设计过程与软件的编制过程都使用了免费软件，由此也展示了如何不使用昂贵的工具与设备而完成机器人设计的方法。

ASURO 配备了一个 RISC 处理器（精简指令集处理器）和两个独立控制的马达，一个线性跟踪器，六个碰撞检测开关，两个行程感应器，三个指示灯和一个用于编程及由 PC 遥控用的红外通讯口（见图 01）。

危险标志  标示的段落都应该仔细阅读，以避免意外损坏器件及造成人身伤害。

请注意，ASURO 不是玩具，切忌把它交予三岁以下的儿童，因为这样极易发生细小元件被小童吞食的危险。
用户需自备电池以便组装置与测试 ASURO。

图 0.1 ASURO 结构框图
（见附录 E）



警告：

- 散件打开后，不能退货。
- 开始组装前请仔细阅读本手册。
- 请小心使用工具。
- 请勿让小童触及本产品，组装时请勿让小童靠近，因为某些工具及元件可能会对儿童造成危险。
- 检查电池的极性。
- 保持电池干燥。当 ASURO 被弄湿时应该取出电池，并把 ASURO 放置一段时间。
- 当长时间不使用 ASURO 时应取出电池。

目录

Part I. 机械部分

1. 必备工具

除本套件提供的元件之外，装配 ASURO 需要以下工具：

- 一个辅佐器
- 界纸刀
- 小铁锤
- 砂纸
- 长鼻勾钳
- 电线剪
- 电烙铁
- 焊锡条
- 熔焊片 (2~3 毫米宽)，用于清除电路板上的焊锡
- 胶水 (一盒干胶，一个热胶手枪)
- 计算机: 一台手提电脑或个人计算机 (运行 Windows 或 Linux 系统)
- 可选件: 一个万用表

*如使用 USB 红外收发器则需要准备 USB 的 A、B 连接线

图 1.1: 必需的工具

2. 机械准备

首先请检查配件是否齐全。附件 A 已经列出了配件清单。在进行电子元件的装配之前，我们需要在机械部件方面作好准备。

2.1. 马达

马达动力必须通过一个被紧套到马达轴承上的小齿轮（上有直径 1.9mm 的钻孔）传送到其他齿轮。把这个小齿轮的平的一面放在平面上，然后把马达的轴承小心地按进小齿轮的钻孔里。你可以在一个稍软的平面（如厚纸板）使用一个小铁锤把马达轴承轻轻敲进小齿轮里(见图 2.1)。

或者也可以用台砧把小齿轮按到马达的轴承上。请注意不要压坏外壳或轴承。

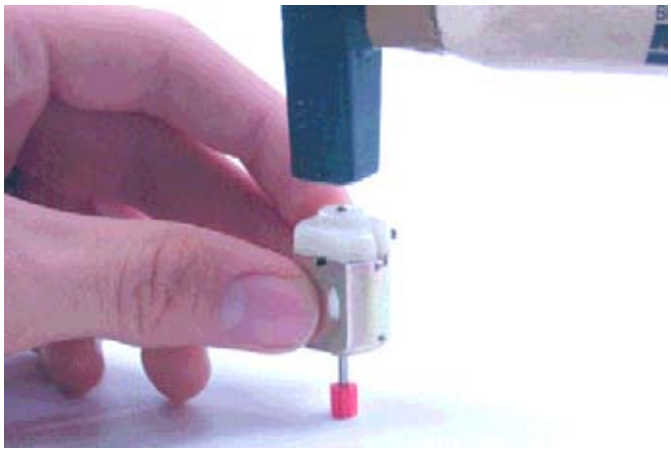


图 2.1.: 马达装配

2.2. 乒乓球

ASURO 设计成可借助半边乒乓球进行滑动。这半边乒乓球必须另外准备好。我们可以用锯或者界纸小刀把一个完整的乒乓球切开两半，切口要用锉子或砂纸处理平整。

图 2.2. 乒乓球

由于乒乓球的材质非常容易着火，故处理乒乓球时切忌使用电锯或电刀。

2.3. 轴承

轴承是由黄铜棒制作而成的。在此需要两对长度各为 24.5mm、42.0mm 的轴承。本套件已包含这两对所需长度的轴承。

2.4. 轮子转速感应器

ASURO 的行程计由朝向带有黑白相间标记的 50/10 型齿轮的一个发光二极管和一个光感器构成。

本套装包含整套行程计标记。按图 2.3 把行程计标记粘贴在 50/10 型齿轮上。建议使用分别有 6 个黑色和白色分块的标记，以便兼容配套的 ASURO 示例程序。

图 2.3.: 建议使用的齿轮标记

标记划分的黑白分块越多，就能越好地分辨和控制齿轮转速。然而过多的分块会对光暗间的辨识变差。(如图 2.4)

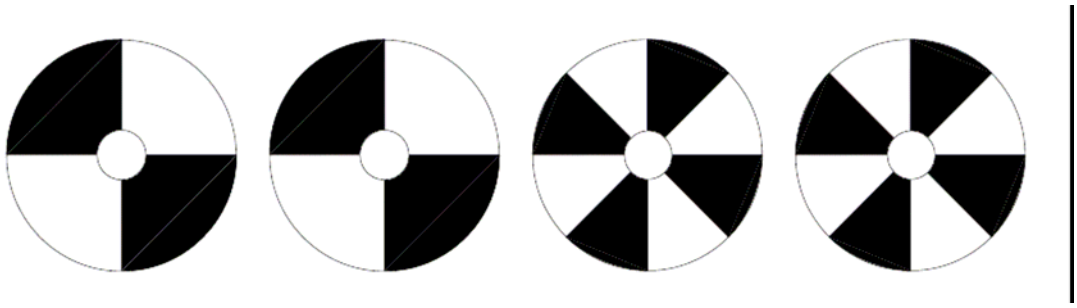


图 2.4.:齿轮标记的例子

至此已经完成了所有机械部分的准备，下面我们将会进行继续电子元件的装配。

Part II. 电子部分

3. 焊接指引

ASURO 的设计均为使用插脚类型的元件（相比表面贴装元件体积较大）。 图 3.1 展示了 ASURO 采用的处理器型号的最小表面贴装封装与实际采用的插脚封装。这些不同封装的内部逻辑是完全一样的! 尽管焊接插脚类型的元件十分容易，但没有相关经验的人也必须注意。

焊接的时候印刷电路板必须是不带电的。
仅断开电池开关并不足够!! 必须拔掉电池和电源!!!

3.1. 烙铁，焊锡丝和温度

图 3.2 展示了焊接的基本方法。烙铁的发热尖端应该达到 360℃以熔解含铅焊锡或达到 390℃以熔解不含铅的焊锡，焊接轴承可能需要更高的温度(420℃)。一般电子元件使用一个较小的烙铁尖端进行焊接，轴承则需要用一个较大的烙铁尖端进行焊接。

可以使用稍湿润的海绵包裹焊锡丝置于烙铁尖端上。在初次或隔一段时间再使用烙铁之前，烙铁尖端上的焊锡必须清除掉。

焊接电子元件时通常使用直径为 0.8mm 或 1mm 的焊锡丝。

图 3.1.: 比较 ATmega8L 芯片的最大和最小封装的情况。

危险端！

图 3.2.: 焊接的基本方法

焊接过程产生的烟可能对你的健康造成伤害。
请避免吸入这些有害气体与在空气不流通的环境工作。焊接时必须使用适合的焊锡丝以避免损坏元件。

3.2. 准备配件

焊接小器件时，你可能需要注意一些小技巧以便于焊接。使用一些辅助的工具例如镊子等非常有效。

某些部元件如三极管，LED，IC，开关座，电容和跳线可以直接焊接在 PCB 上（印刷电路板），但是二极管和电阻则需要先作一些处理。

在 ASURO 的设计中，电阻采用垂直放置的方式焊接。因此电阻的一端保持不变，另一个端弯曲 180 度。弯曲部分形成 2.5 毫米直径的孤型，为避免损坏电阻，应使此弧型与电阻距离若干毫米。

二极管则采用水平放置的方式焊接，可以使两脚同时弯折以插入电路板上的相应插孔。

处理器 IC1 ATmega，IC3 CD4081 和红外接收器 IC2 SFH5110-36 容易被静电损坏。当你身上带电时轻微的接触都可能损坏它们。因此接触这些器件前，需使用接地的放电手镯，至少接触一些接地体或者金属外壳。

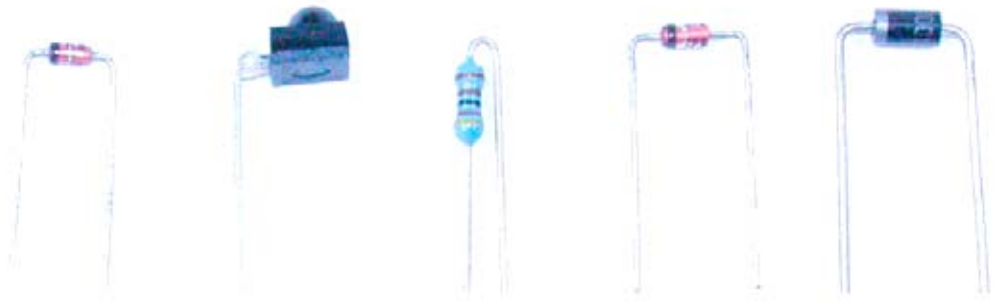


图 3.3: 接脚被弯折的部件

A. 元件
B. 引脚，焊盘和弯折位需同时加热。
C. 焊锡需熔进钻孔。
D. 焊锡
E. 圆滑的圆形弯位
F. 烙铁尖端
G. 锥形的光洁焊点
H. 弯折接脚以防元件跌落。
I. 弯折位需与元件有一定距离

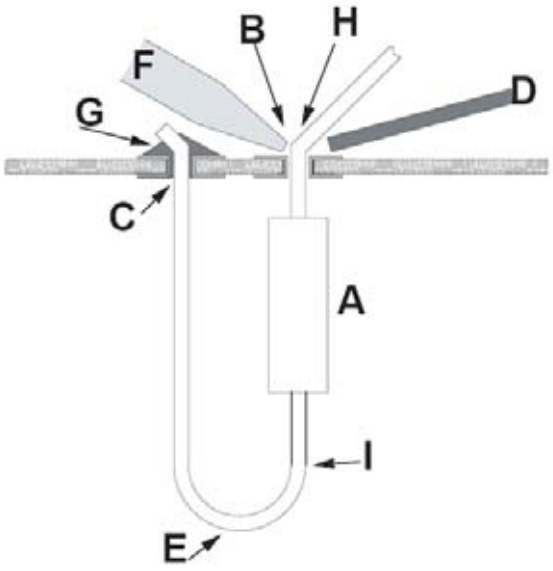


图 3.4 效果良好的焊接

3.3. 焊接零配件

引脚弯折好后，元件可放置到电路板相应的钻孔中，一般可先放置两脚或是三脚的元件。在电路板背面把引脚弯曲可以防止元件跌落。对于有多引脚的元件，例如 IC 的插座，可以弯折处于对角线的两个接脚就可以固定元件了。

固定元件后，使用烙铁的尖端对元件的引脚以及焊盘同时加热，同时添加小量的焊锡。熔解后焊锡会流进钻孔中，加足够的焊锡使钻孔完全充满。此时可相继移开焊锡与烙铁，在焊接位冷却并固定之前不能移动元件或电路板，否则很可能会令焊接位松脱。

焊位的球状焊锡滴常常意味着不良的焊接，必须被重新焊接。

把插座或其他需要水平放置的元件焊接在电路板上，可能需要使用以下的方法：首先只焊接元件的其中一个引脚，然后轻轻按下此元件并再次加热之前的引脚 (注意：此时元件可能非常热)。此时元件就已经可靠地固定在电路板上，紧接着就可以焊接其他引脚了。最后如有需要，可添加一点焊锡加固第一个引脚的焊接。

在焊接完一个元件后，对突出的多余部分引脚需用手钳切断。尽量在接近电路板的位置剪断引脚，并注意不要牵拉引脚。

在切断引脚时注意不要让锋利的引脚线散落四处。
垂直放置的元件不能粘连邻近的元件。
如果元件间太靠近，必须弯曲相应的元件使相互隔开。

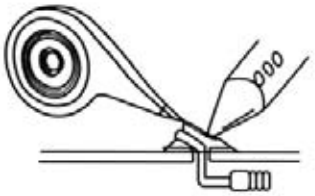
3.4 熔焊

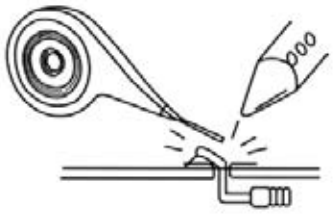
有时你需要移走焊接位置错误的元件，由于 ASURO 的电路板是双面而且通孔的，因此移走元件有一定难度。

可以参考以下方法：
在需要熔焊移走的部件上放置一些金属线（一种简单的方法就是添加一些焊锡），当该元件的所有焊点都被加热的时候可使用一个镊子移走它，之后就能够轻易地除去余下的焊锡。

若元件仍然在电路板上未能除去，则可把熔焊金属片放在焊点上（参考下面的第一和第二步骤）。把金属片和焊点一起加热。到达一定温度时，焊点会熔解焊点并将其吸附在金属片上。此时，应迅速松开烙铁和金属片。

若该钻孔的另一面仍然有焊锡时，如法除去即可。





第一步骤：
把金属片放在需取走的元件的焊点上。
加热金属片和焊点，熔解的焊锡将会吸附在金属片上。

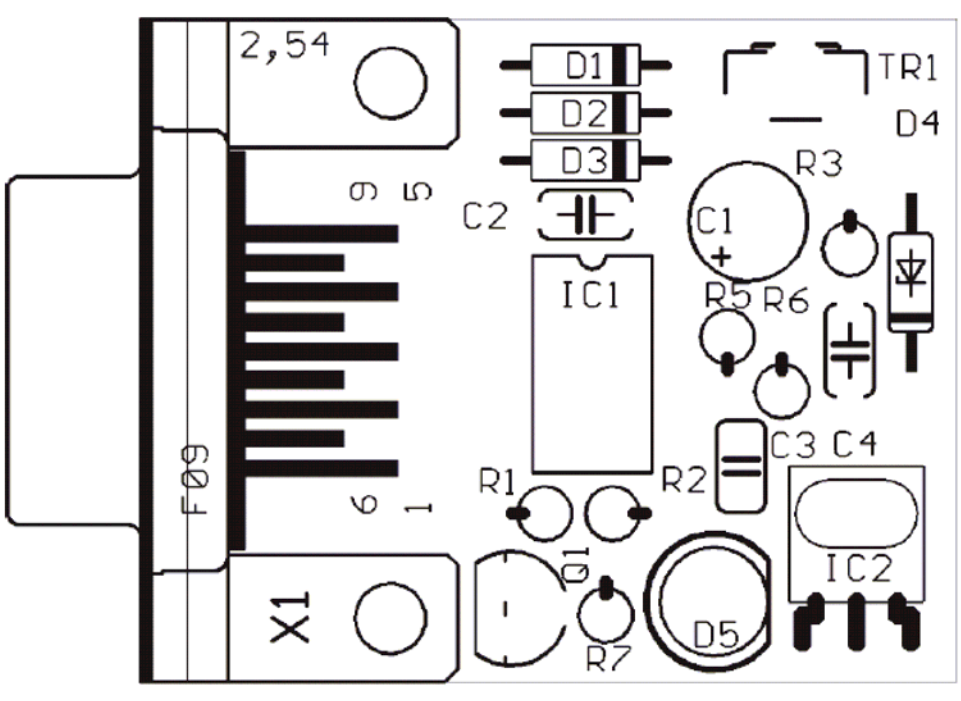
第二步骤
同时松开金属片和烙铁。

4. 电子装配

如果已经阅读了之前讲述焊接工艺的内容，接着就可以进行下面的元件焊接工作。

4.1 焊接 RS232 红外收发器

- IC1：首先插入 8 脚的插座。插座的极性标记一定要对应电路板上的极性标记。
- D1，D2，D3：1N4148，注意极性！注意元件上的标识，不要错用 ZPD5.1 或 BZX55-C5V1。
- D4：ZPD5.1 或 BZX55-C5V1，注意极性！注意元件上的标识，不要错用 1N4148
- D5：SFH-415-U IR LED（黑色的 LED），注意极性！需向下压平贴在电路板。
- C1：100μF 最低耐压 16 伏，注意极性！
- C2，C4：100nF 陶瓷电容器，标识：104
- C3：680 pF 陶瓷电容器，标识：681
- Q1：BC547（A，B 或 C）或 BC548（A，B 或 C）
- R1，R5：20 kΩ（红色，黑色，橙色，金色）
- R2：4.7 kΩ（黄色，紫色，红色，金色）
- R3：470Ω（黄色，紫色，黑色，金色）
- R6：10 kΩ（褐色，黑色，橙色，金色）
- R7：220Ω（红色，红色，褐色，金色）
- TR1：10 KΩ可变电阻器
- IC2：SFH5110-36 红外接收器，需用夹钳把插脚弯曲！注意极性！（有突面的一面需被往外放置）注意：静电放电（ESD）和长时间焊接导致的过热极易损坏此元件！
- X1：9 针 SUB-D 接头，外壳需按紧贴在电路板，固定端需焊接牢固。
- IC1：插入 NE555P，注意极性！



最后检查电路板确认没有出现短路或极性错误。检查焊点质量和重新焊接不牢固的焊点。

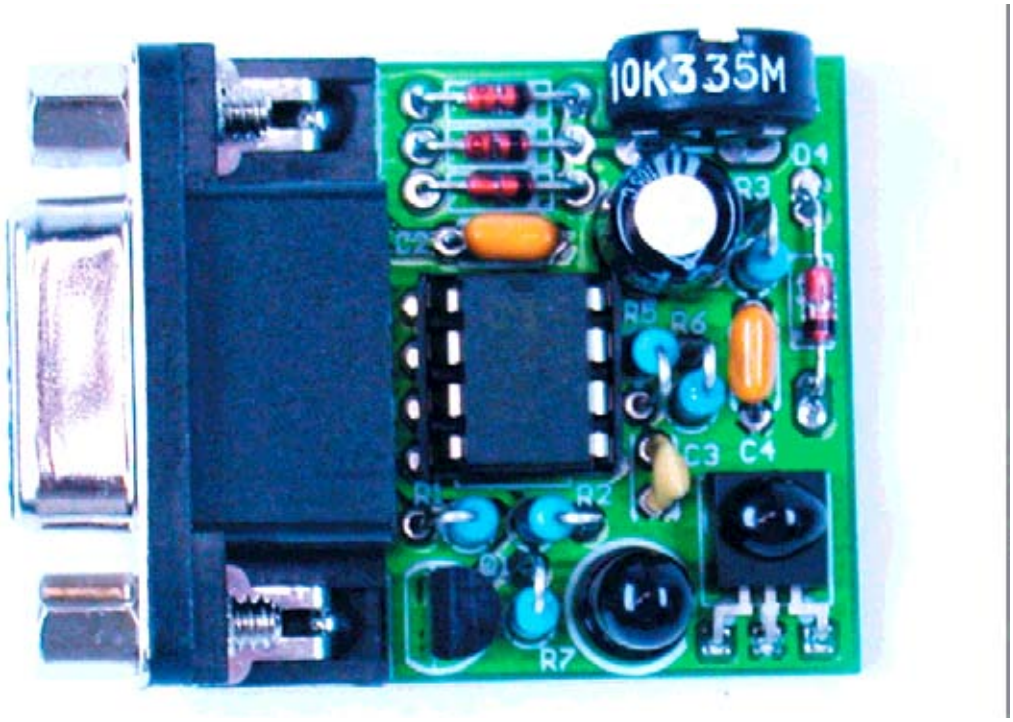


图 4.1.: 焊接好的红外收发器

上面的照片就是已经完成元件焊接的 RS-232 红外收发器。

4.2. 焊接 USB-红外收发器

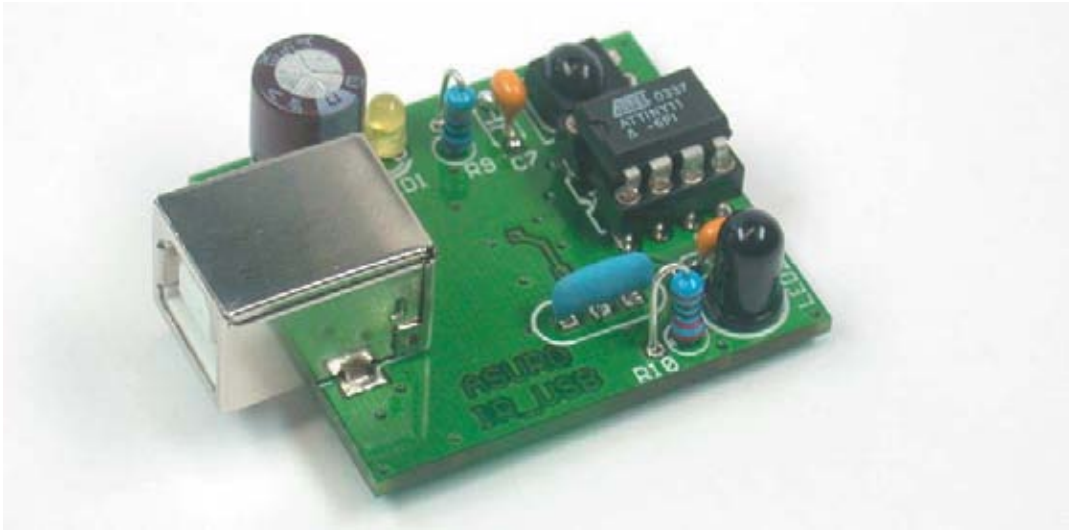


图 4.2：USB 红外收发器

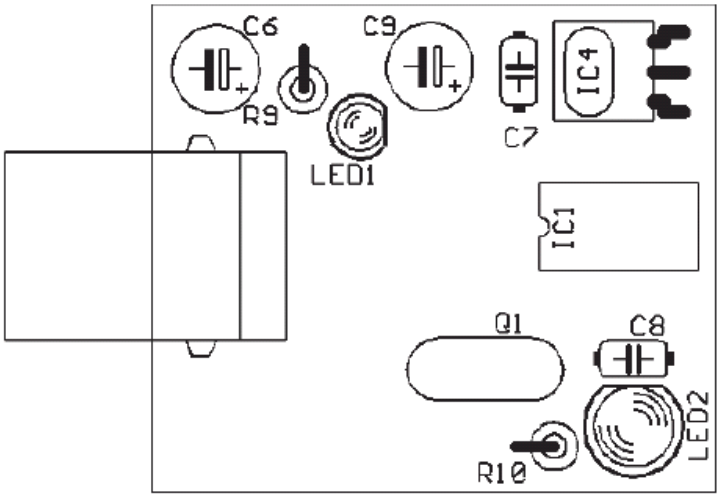


图 4.3：USB 红外收发器元件面

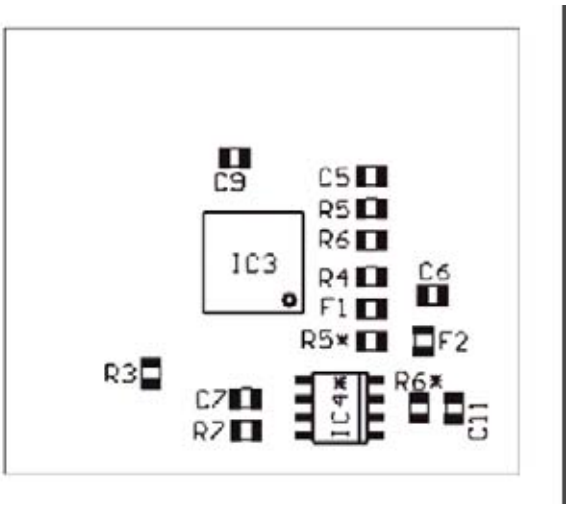


图 4.4：USB 红外收发器底面

4.3. 焊接 ASURO 的其他零件

安装第二级齿轮的两个较长的轴承需焊接或粘合在电路板的底面。现对而言使用焊接会更快更牢固，需要时能很容易通过熔焊来修改。

两个较短的轴承需安装在电路板上面靠近板的中央位置。装配前一定要用较细密的砂纸把准备焊接的部分打磨光洁（可先不打磨安装轮子的一端）。在焊接轴承时，可参照下列步骤进行：

首先安装长轴承。把电路板平放在桌子，底面向上，把长轴承完全放进槽位里。这时轴承应该是水平放置在槽位里的。在烙铁的尖端粘上的焊锡，把它熔在轴承上。轴承加热到达一定温度后，在轴承下的周围开始添加焊锡焊接。焊接完成后，移开烙铁并马上用螺丝刀把轴承向下按紧直到焊锡完全冷却粘牢。

按照以上的步骤继续焊接第二条长轴承。然后把电路板翻转，重复以上的方法焊接短轴承。图 4.3 为已经安装好四条轴承的电路板。

稍冷却一段时间后就可以在轴承上安装轮子。齿轮的安装位置需恰到好处并使轮子能顺畅的滚动。如果轮子滚动不顺畅，很可能因为轴承焊接不良，这时可考虑重新焊接。如果因为在安装轮子或齿轮的轴承端粘上焊锡而影响它们的转动，则可用砂纸或锉小心清除。

如果轮子能顺畅的滚动，可以把轮子暂时取开以便继续焊接电路板上的电子元件。

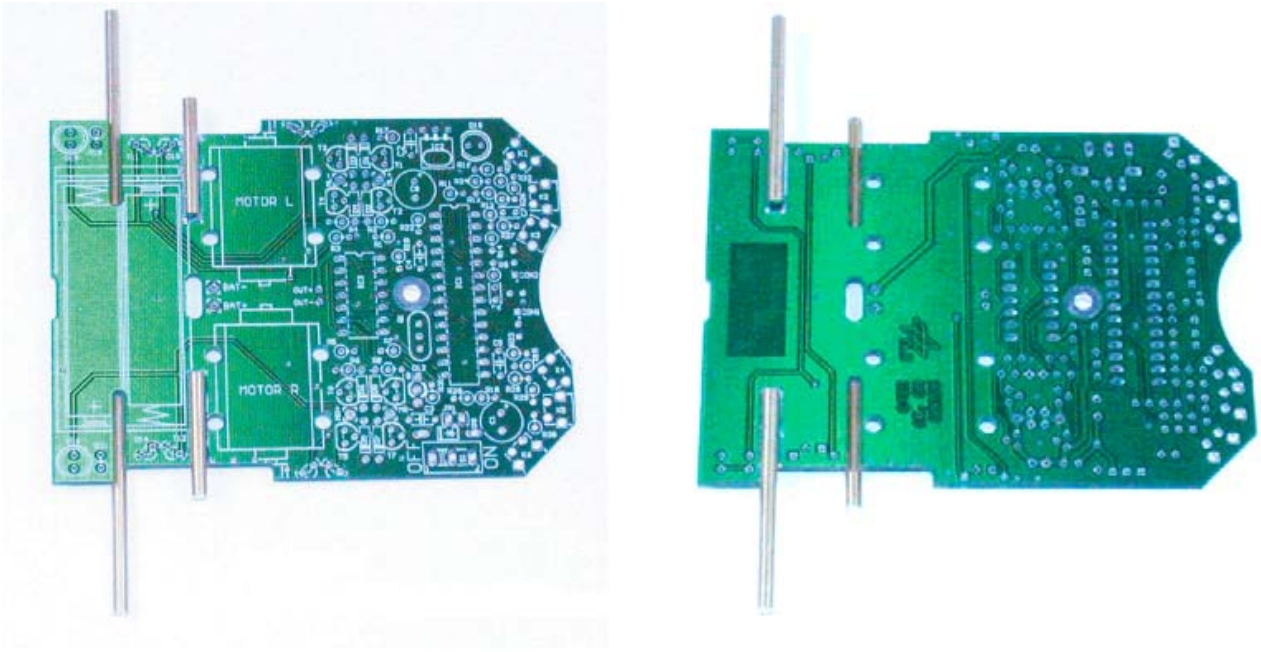


图 4.3.:
安装好轴承的 ASURO 电路板

010-62669059 fae@61mcu.com www.61mcu.com

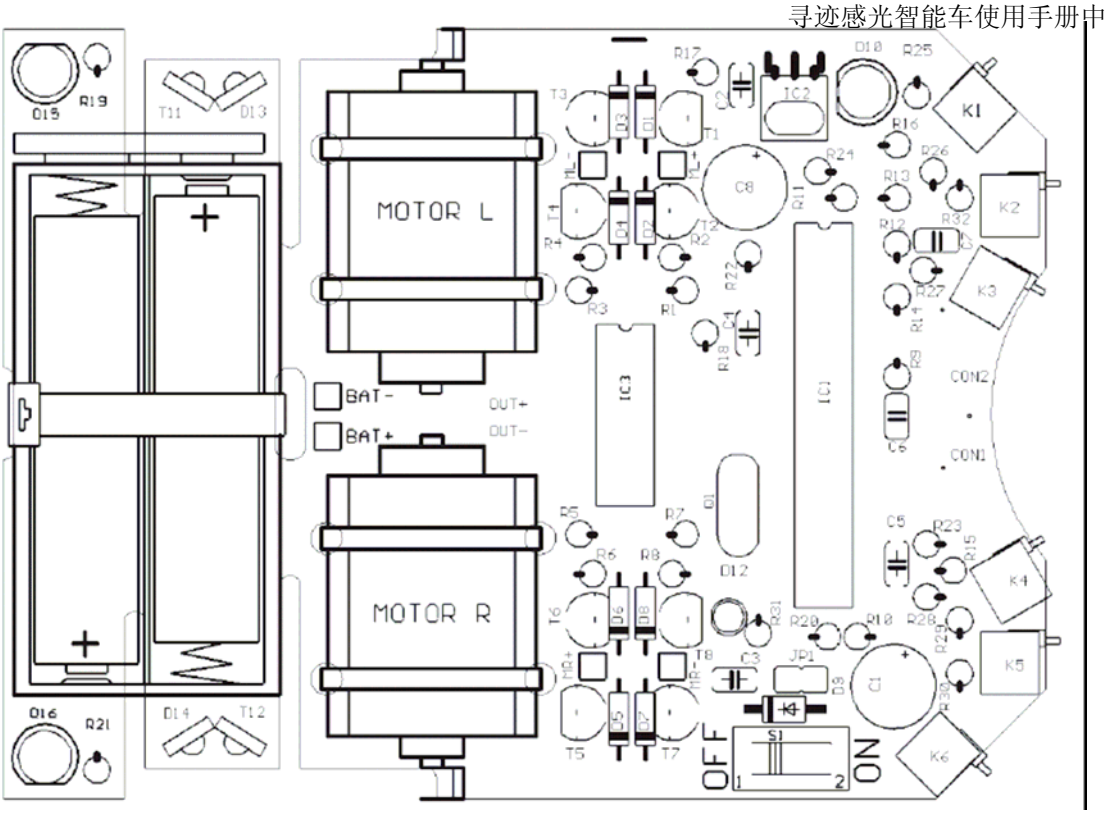


图 4.2：安插好电子元件的 ASURO 电路板俯视图

可以参照以下先后顺序安装元件：

- IC1：首先插入 IC 插座（双列共 28 脚插座或者两列 14 脚插座）。注意极性！插座的极性标记应与电路板上的极性标记相对应。

- IC3：也是先插入 IC 插座（双列共 14 脚插座）注意极性！插座的极性标记应与电路板上的极性标记相对应。

- K1，K2，K3，K4，K5，K6：触碰开关，需水平安装在电路板上。

- Q1：8 MHz 晶振

- D1，D2，D3，D4，D5，D6，D7，D8：1N4148 二极管；注意极性！

- D9：1N4001；注意极性！

- JP1：跳线；短脚的一端用于焊接，且先不要接上跳线帽！

- D12：双基色 LED，3 毫米直径，3 个引脚，注意极性！（此器件是区分极性的，最短的脚需要对应插在方形焊盘的插孔）！

- C2，C3，C4，C5：100 nF 陶瓷电容；标识：104

- C6，C7：4,7 nF 陶瓷电容；标识：472

- T1，T3，T5，T7：BC327-40 或 BC328-40
- T2，T4，T6，T8：BC337-40 或 BC338-40
- R1，R2，R3，R4，R5，R6，R7，R8，R19，R21：1KΩ 5%（褐色，黑色，红色，金色）
- R9，R16：220Ω 5%（红色，红色，褐色，金色）
- R10，R17，R22，R31：5% 470Ω（黄色，紫色，褐色，金色）
- R11：100Ω 5%（褐色，黑色，褐色，金色）
- R12：12 KΩ 1%（褐色，红色，橙色，金色）
- R13：10 KΩ 1%（褐色，黑色，橙色，金色）
- R14，R15：20 KΩ 5%（红色，黑色，橙色，金色）
- R18，R20：4,7 KΩ 5%（黄色，紫色，红色，金色）,金)
- R23：1 MΩ 5%（褐色，黑色，绿色，金色）
- R24：1 KΩ 1%(褐色，黑色的，和黑色的，褐色，褐色）
- R25，R26，R32：2 KΩ 1%（红色，黑色，黑色，褐色，褐色）
- R27： 8,2 K Ω 1%(灰的，红色的，和黑色的，褐色,褐色)
- R28：16 K Ω 1%（褐色，蓝的，黑色，红色，褐色）
- R29：33 K Ω 1%（橘色，橙色，黑色，红色，褐色）
- R30：68 K Ω 1%（蓝色，灰色，黑色，红色，褐色）
- C1，C8：Elco 220μF 耐压 10 V 或更高；注意极性！
- IC2：SFH5110-36 红外接收器，需用夹钳把插脚弯曲！注意极性！（有突面的一面需被往外放置）注意：静电放电（ESD）和长时间焊接导致的过热极易损坏此元件！

- D10：SFH 415-U 红外 LED；5mm，黑色外壳；需贴近电路板放置；注意极性。
- T11，T12：LPT80A，IR-LED；无色外壳；需贴近电路板放置；注意极性。
- D13，D14：IRL80A，IR-LED；粉红色外壳；需贴近电路板放置；注意极性。
- D15，D16：LED 5 毫米 红色，红色的外壳。注意极性（短脚需插在有标示的位置）。
- S1：通/断型开关

还需要安装另外三个元件（将用于沿直线行走），但要将他们放置在电路板的底面，并且要从电路板正面焊接：（见到图 4.5）

。 T9，T10：SFH300，光感器 5 毫米；注意极性！这两个元件需要离开电路板一些距离放置。
。 D11：发光二极体 5 毫米 红色，红色外壳；注意极性！（短脚需插在有标示的位置）!

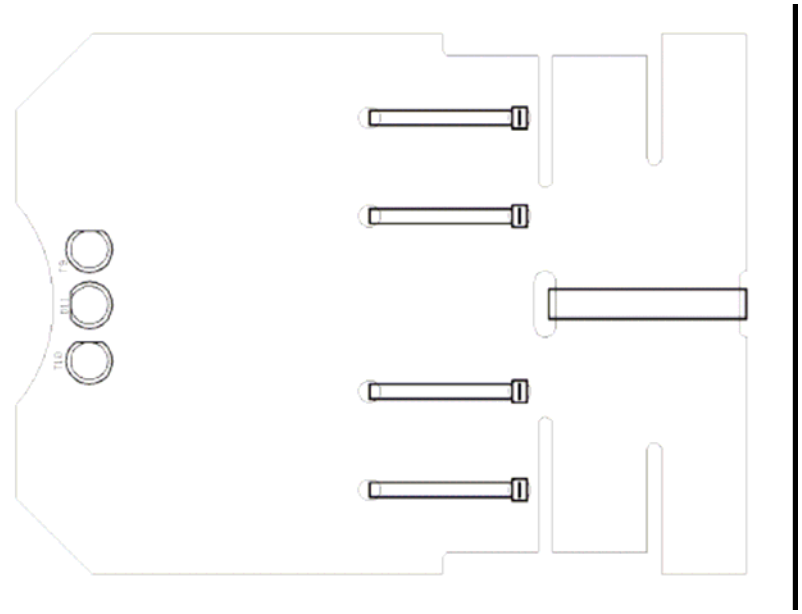


图 4.5： 展示了焊接好所有电子元件的电路板的正面和底面。

至此，我们不再需要焊接任何电子元件了。
在下一阶段中我们将要安装电动机械元件和机械的零配件。

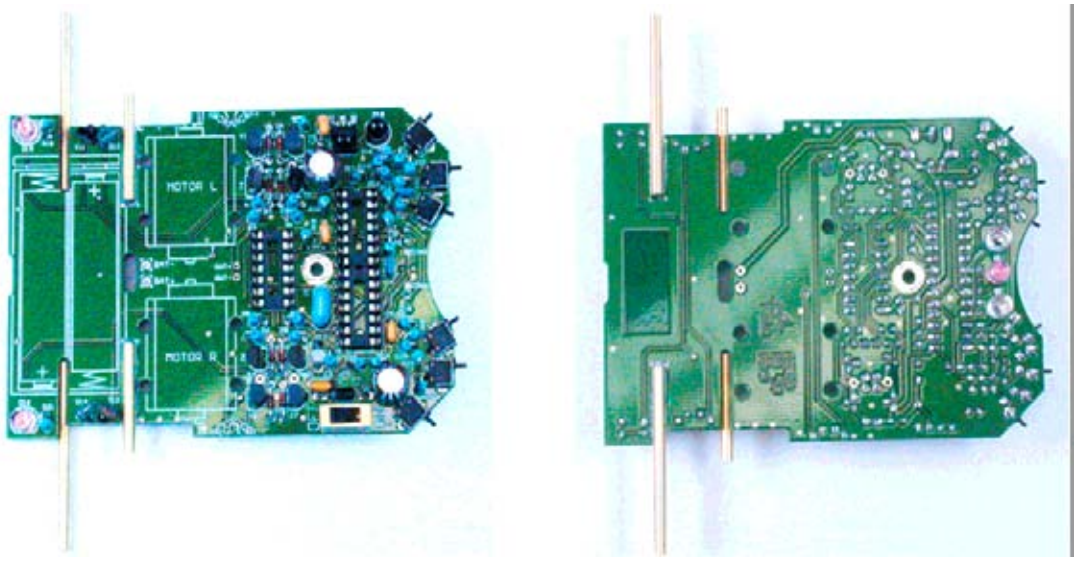


图 4.6：插入零配件的 ASURO 电路板底面图

4.4. 马达安装

在完成 ASURO 电路板上的零配件焊接安装之后，我们需要给马达绑线并暂时固定下来以便进行测试。

为了给马达供电，我们需要 70 毫米长的红色和黑色的电线各两根。这四根线均需在两头脱去线皮，把裸露的导线拧成一股并熔一些锡在上面。附在两端的细小开叉金属线或是过量的锡都需用工具除去。把红色的电线焊接在马达上有一个点或一个正号的一端，黑色的电线则接在余下的一端。

马达的两根电源线应相互绕紧在一起（这样做可以减少电磁的可能的干扰及有更好的观感）。

左边马达的红色电线需要焊接在“ML+”焊孔，黑色电线则焊接在“ML-”焊孔，右边马达的红色电线需要焊接在“MR+”焊孔，黑色电线则焊接在“MR-”焊孔。

然后用塑料扎带把马达暂时固定在电路板上。初始测试只需要各使用一根扎带作固定即可。

4.5. 电源

如果 ASURO 使用普通电池供电，跳线 JP1 需要断开! 如果使用可充电电池，跳线 JP1 需要接通。如果在跳线 JP1 接通的情况下放反电池极性会很容易损坏电子元件!

把电池盒的红色电线焊接在电路板的"BAT+"焊孔，黑色电线则焊接在" BAT -"焊孔。此时应检查确认通/断型开关是处在断路位置。在电池盒中正确放入四个电池，并可用一条较大的塑料扎带把它固定在 PCB 板上。

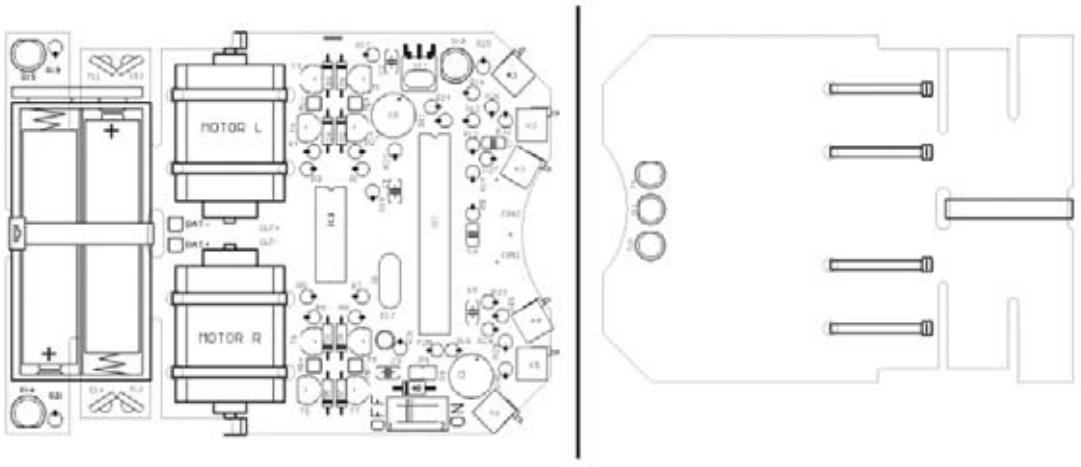


图 4.7. PCB 板的正面与背面视图

5. 硬件测试

在焊接安装完毕之后我们将开始操作 ASURO。但首先需要查找并排除之前的组装过程中可能出现的错误，以最大限度地避免损坏器件。

5.1. RS232-红外收发器

下面的测试内容仅仅是针对 RS232-红外收发器(图 4.8)。
我们必须首先测试红外收发器，因为紧接着的 ASURO 系统自测必须使用到它。
我们通过一个带九针串行接口延长电缆把这个红外收发器与 PC 的一个可用串行端口相连，启动 Windows 系统的超级终端程序（Linux 系统使用 Minicom 程序）。一般可以通过 Windows 系统的[附件]→[通讯]→[超级终端]打开这个终端程序。如果系统没有安装此程序，可以通过 Windows 系统光盘进行安装。

在启动超级终端程序后，将需要为本次连接命名，我们可以把它命名为 ASURO，或其他随意的名称。在下一步系统弹出的“连接到”窗口中选择之前所连接的串行端口。按下“确认”后选择以下的参数设置：

- 每秒位数：2400
- 数据位：8
- 奇偶校验：无
- 停止位：1
- 数据流控制：无

按下“确认”完成参数设置

手持红外收发器，使它的元件面朝向一张白纸并相距约 10cm 的距离。

按下键盘输入一些字符
终端程序应该会显示所输入的字符。当输入字符时，数据通过红外收发器的红外发送器(D5)发送，发送出来的信号经由白纸的表面反射回红外接收器(IC2)接收，继而送回计算机显示出来。如果没有字符返回或返回不正确的字符，可以尝试调整收发器上面的可调电阻阻值的大小。用一个较小的螺丝刀一边调整可调电阻一边在键盘输入，直到终端程序上显示的字符正确为止。

如果经过上面的调整仍然未能成功，则需要在硬件电路上查找问题所在(见 6.1 节)。

测试完成后应该拔开红外收发器，若拔开之后再按键盘，在终端程序上不应该再有任何字符显示。

5.2. USB-红外收发器

下面的说明仅仅是针对 USB-红外收发器。
注意：由于 USB 红外收发器没有外壳包装，因而极易受静电而损坏。在使用它之前，切记先要通过触摸计算机的金属外壳或其他接地体释放身体的静电。当然也可以使用一个透明的外壳包装加强保护。

5.2.1 Windows

下面的测试内容仅仅是针对 USB-红外收发器。
我们必须首先测试红外收发器，因为紧接着的 ASURO 系统自测必须使用到它。
我们通过一个 USB 接口延长电缆把这个红外收发器与 PC 的一个可用 USB 端口相连，
现在计算机应该会提示“发现新硬件”
AREXX ASURO USB-IR-TRANSCEIVER

此时可从 ASURO 提供的光盘安装 USB 驱动程序。 若系统不能自动检测到新硬件，我们可以通过手动选择光盘里的驱动程序\windows\USB Driver 进行安装(安装驱动程序需要有系统管理员权限)。成功安装完驱动程序后，使用时 USB 红外收发器就如一个普通串行口。

在启动超级终端程序后，将需要为本次连接命名，我们可以把它命名为 ASURO，或其他随意的名称。在下一步系统弹出的“连接到”窗口中选择之前所连接的串行端口。按下“确认”后选择以下的参数设置：

- 每秒位数：2400
- 数据位：8
- 奇偶校验：无
- 停止位：1
- 数据流控制：无

按下“确认”完成参数设置

手持红外收发器，使它的元件面朝向一张白纸并相距约 10cm 的距离。

按下键盘输入一些字符
终端程序应该会显示所输入的字符。当输入字符时，数据通过红外收发器的红外发送器(D5)发送，发送出来的信号经由白纸的表面反射回红外接收器(IC2)接收，继而送回计算机显示出来。如果没有字符返回或返回不正确的字符，可以尝试调整收发器上面的可调电阻阻值的大小。用一个较小的螺丝刀一边调整可调电阻一边在键盘输入，直到终端程序上显示的字符正确为止。

如果经过上面的调整仍然未能成功，则需要在硬件电路上查找问题所在(见 6.1 节)。

5.2.2 Linux

下面的测试内容仅仅是针对 USB-红外收发器在 Linux 系统下使用的情况。
我们必须首先测试红外收发器，因为紧接着的 ASURO 系统自测必须使用到它。
我们通过一个 USB 接口延长电缆把这个红外收发器与 PC 的一个可用 USB 端口相连，
如果听到“beep”的一声则表明 Linux 系统已经检测到相连的红外收发器。为进一步确认，
更可以通过检查系统是否有如以下的信息显示：
foo@bar:/>cat /proc/tty/driver/usb-serial

也可检查是否有类似如下的信息(其中下面的‘0’可以是‘1’或‘2’等数字)
Usbserinfo：1.0 driver: v1.4
0：modulef：ftdi_sio name："FTDI 8 U232AM Compatible" vendor：0403 product：6001
num_ports：1 port：1path：usb-00：11.2-1

在进行测试前请先设置好 Minicom 在端口/dev/ttyUSB0 (0 可以为 1 或 2 等)的参数

- 每秒位数：2400
- 数据位：8
- 奇偶校验：无
- 停止位：1
- 数据流控制：无

按下“确认”完成参数设置
以上的设置很可能需要 root 权限。

Linux 系统可能需要用户为这个新设备/dev/ttyUSB 设置读写权限。具体设置可以通过使用命令 chmod u+rw /dev/ttyUSB 或 chmod g+rw /dev/ttyUSB0(0 可以为 1 或 2 等)实现。

手持红外收发器，使它的元件面朝向一张白纸并相距约 10cm 的距离。

按下键盘输入一些字符
终端程序应该会显示所输入的字符。当输入字符时，数据通过红外收发器的红外发送器(D5)发送，发送出来的信号经由白纸的表面反射回红外接收器(IC2)接收，继而送回计算机显示出来。如果没有字符返回或返回不正确的字符，可以尝试调整收发器上面的可调电阻阻值的大小。用一个较小的螺丝刀一边调整可调电阻一边在键盘输入，直到终端程序上显示的字符正确为止。

5.3. 测试 ASURO

注意：处理器 (IC1) 仍未插在电路板上

经过以上的各项检查之后，把总开关打开。两个后置 LED(D15,D16)应该会微微发亮。如果没有点亮，请立即关上电源并阅读 6.3 节内容。如果是微微发亮，则可关上电源并插上 IC1(处理器)和 IC3(“与”门) (见到图 5.1)。

有时 IC 的插脚需要小心地作一定程度的折弯以刚好能插入 IC 插座。折弯时可以把 IC 芯片的放在一个平的桌面上，然后把 IC 芯片侧放，轻轻地向下挤压整行的 IC 引脚到一定位置即可。

处理器 IC1 ATmega8 和“与”门 IC3 CD4081 均为易受静电损坏的器件。如果身体积聚了一定量的静电时，触碰这些器件会很容易使它们损坏。在触碰这些器件时应戴上一个防静电手镯或至少应先触摸一个与大地相连的金属体。

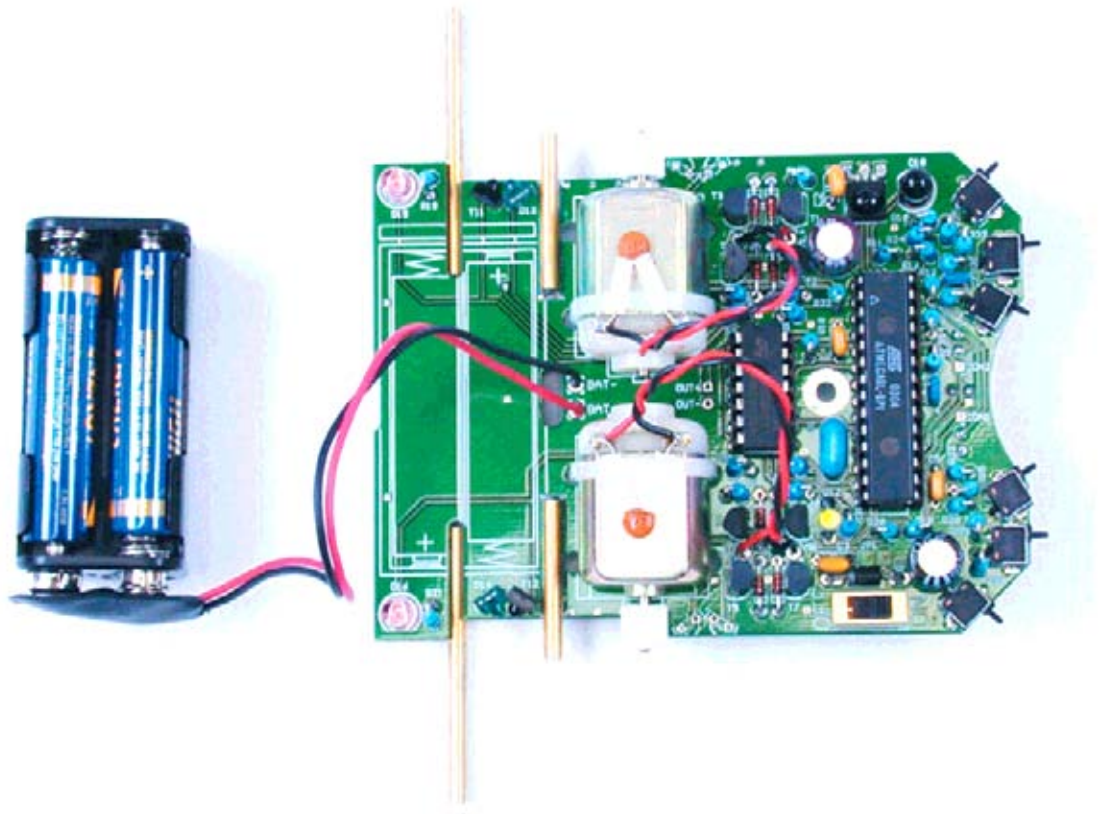


fig.5.1.: 已经安装了 IC 芯片的 ASURO

如果使用了可充电电池则需要插上 J1 的跳线帽子。各个 IC 芯片的极性标记应与插座的极性标记和 PCB 上的极性标记相一致。运行于处理器的检测程序将会在系统上电之时检测所有器件的状态。为了尽可能避免错误，最好能在仔细阅读完下一章之后再返回本节内容。打开电源后，自检程序便开始运行，此时应仔细观察 ASURO 的现象。

当超级终端程序(Windows)或 minicom(Linux)已经打开，并且此时在红外收发器与 ASURO 之间有直视距离时，在计算机上便可看到系统自检的整个过程。

5.3.1. 状态显示元件

状态 LED(D12)会短暂的显示橙色且两个后置 LED(D15,D16)也会点亮，但这两个仅仅是微微发亮。如果现象并非如上所述，请马上关断电源并开始排查错误(见第 6.3.3 段)。之前的为 ASURO 的启动初期的特征，下一步将会顺序地检查全部的状态显示元件，每步检查会间隔 3 秒，顺序如下：

- 状态 LED(D12) 绿色
- 状态 LED(D12) 红色
- 前置 LED(D11) 位于 ASURO 的底部
- 后置 LED(D15) 左边
- 后置 LED(D16) 右边
- 全部状态显示元件同时点亮

如果在上述过程中出现异常，请马上关断电源并开始排查错误(见第 6.4 段)。因为每一个状态显示元件都在以下的测试中起重要的作用，故这里需确保正常。

5.3.2. 光感器 (T9,T10)

在完成所有状态显示元件的检验之后，状态 LED (D12) 会亮为绿色，此时位于 ASURO 底部用于线性跟踪的光感器(T9,T10)开始了为期 10 秒的测试。

当光感器(T9,T10)从感光到不感光时，与之相对应的两个后置 LED(D15,D16)会被点亮并处于微亮状态(右边的光感器 (T10) → 右边后置 LED(D16)；右边的光感器 (T10) → 右边后置 LED(D16))。如果两个后置 LED 在光感器不感光时仍继续保持微亮，则说明检测结果正确。如果这步检测出现异常，自检程序仍会继续进行。故障排查将会在之后进行。

5.3.3. 触碰开关

ASURO 的全部指示灯都开始熄灭，从现在开始大约 15 秒时间内可以对触碰开关进行测试。轻触各触碰开关，正常状态下将有相应的现象产生。
正确的结果应该是：

K1 → 状态 LED (D12) 点亮为绿色
K2 → 状态 LED (D12) 点亮为红色
K3 → 在底部的前置 LED 点亮
K4 → 左边的后置 LED (D15)
K5 → 右边的后置 LED (D16)
K6 → 马达的左边开始运转(如果马达不运转，可能是由于下一步的自检已开始。稍后两个马达都会分别进行检测。如这步检测异常则可根据第 6.8 段进行排查错误)。

如果同时按下多个触碰开关则会出现相应现象的组合。即使这一步出现异常，自检程序仍会继续进行。故障排查将会在之后进行。

5.3.4. 行程计

现在线性跟踪器(D11)开始点亮。这表明下一阶段的检验已开始(15 秒)，在这阶段可以对两个行程计(每个行程计均由一个发光 LED 和一个光感器构成)进行检测。手持白纸放于在行程计之前会使状态 LED(D12)发亮。可先放一张白纸在左边的行程计前(T11)，这样会使状态 LED(D12)点亮绿色。然后在放一张白纸在右边的行程计前(T12)，这样会使状态 LED(D12)点亮红色。移开白纸将会使状态 LED 熄灭。如果两个行程计已能辨别是否有白纸遮挡，则说明两个行程计工作正常。如果这步检测出现异常，自检程序仍会继续进行。故障排查将会在之后进行。

5.3.5. 马达

两个后置 LED(D15,D16)开始点亮。这表明已到了自检程序的下一步，这一步骤将会持续 15 秒。两个马达在这段时间内会得到全面的检测。左边的马达首先会启动正转，速度由零加速到最大，之后又降低到零。继而马达改变转动方向，速度也是由零加速到最大，之后又降低到零。左边的马达检测完毕，右边的马达会进行同样的检测。经过先后单独的检测后，两个马达随后会同时进行检测。同样如果某个检测环节出现异常，自检程序仍会继续进行。故障排查将会在之后进行。

5.3.6. 红外接口

如果状态 LED 闪动黄色，表明已经开始最后一步检测(约 15 秒)，在这段时间内在持续的收发数据。如果需要接收这些数据，先前焊接好的红外收发器需与 PC 相连并打开终端程序，如 Windows 的超级终端，参数的设置可参照之前调试红外收发器时的做法。
这时如果收到字符信息，ASURO 会回答该接收字符的下一个字符。如果在设定的时间内没有收到字符信息，ASURO 会发送一个字符 ‘ T ’，发送字符时会使状态 LED 显示红色，这样与一直点亮的绿色叠加就显示为闪动的橙色。

如果已经在 ASURO 与红外收发器之间建立器红外连接(两者距离不大于 50cm)，终端程序会不断地显示字符 ‘ T ’。如果按下 PC 键盘，终端程序会显示该键下的字符及紧接着的下一个字符，例如：

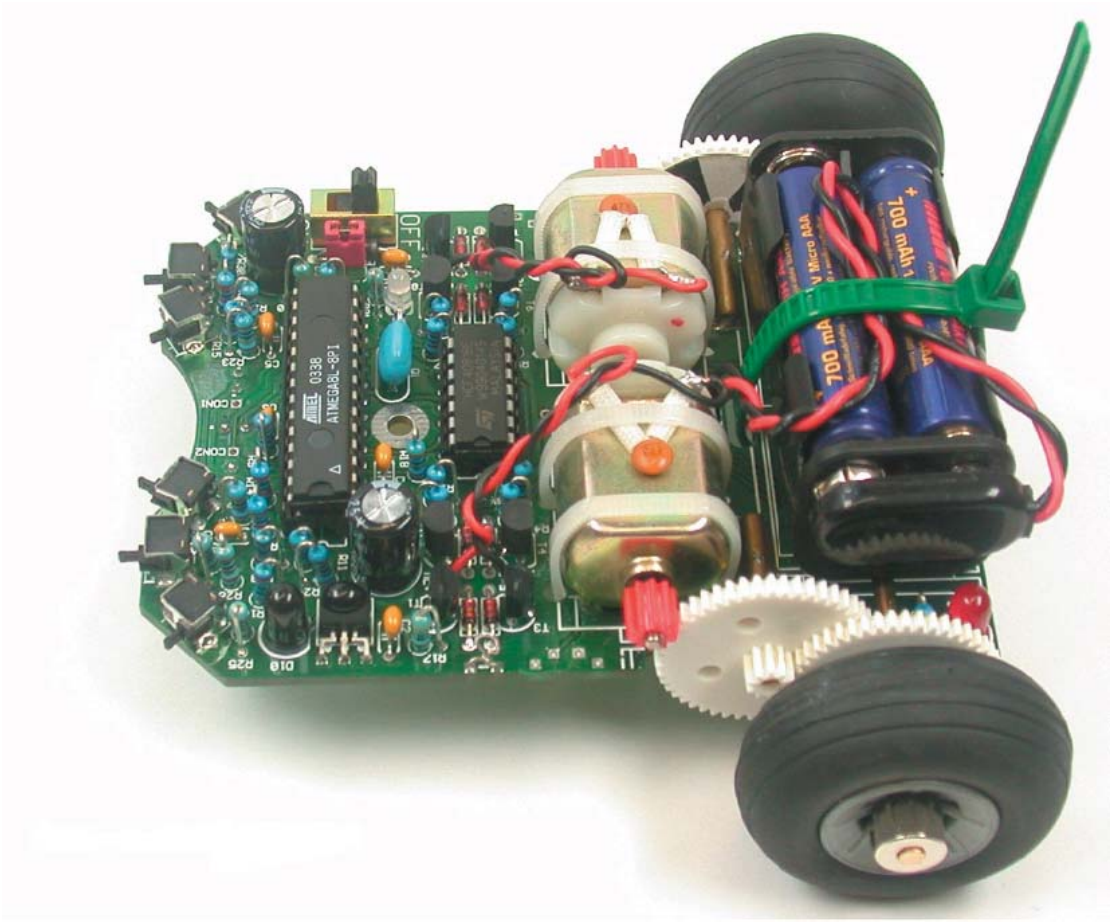
按下键盘 “e” 键 → 终端程序显示 “ef”
按下键盘 “j” 键 → 终端程序显示 “jk”
按下键盘 “3” 键 → 终端程序显示 “34”

如有错误请参阅 6.9 章解决办法。

5.3.7. 硬件是否完全正确

一旦有故障发生，请关断电源并取去电池并参照第 6 章的解决方法排查错误。最后再次运行自检程序以确认每一个部件都工作正常。只有在整个自检程序运行过程均无出现任何错误才能说明 ASURO 的硬件是正常的，之后出现的其他错误应该在软件上找原因。

如果已经在 ASURO 中安装了其他应用程序后再需要检测系统硬件，就需要重新安装由光盘提供的“SelfTest.hex”程序。



6. 故障排查

6.1. RS232 红外收发器故障

6.1.1. 按键与显示字符不对应

调整可调电阻 TR1 直到按下的按键与显示的字符相同。

6.1.2. 终端程序没有任何显示

检查是否插入时序 IC (IC1)或插入的极性是否正确(极性的标记指向 3 个二极管)。
使用一般家用电器(如电视、影碟机)的遥控器靠近红外收发器按下按键，如果在终端程序能看到一些数据，则说明收发器(IC2, R3, C4, D4, T1)已经工作，此时应该检查其他部件的原因。

6.1.3. 仍然有故障

检查所有元件的极性及参数(见图 4.1)是否正确。检查元件焊接是否有短路现象或是焊接不稳固，又或是焊点有开裂。如果这些检查仍然未能解决问题，则可借助测量仪器(如万用表或示波器)并参照收发器的硬件原理图(见附件 B)逐一检查各元件的工作状态。(问题会较多出现在以下器件上：IC1, IC2, Q1, D4)。

6.2. USB 红外收发器故障

6.2.1 Windows

检查驱动程序是否安装正确，COM 端口是否选择正确。

6.2.2 Linux

先断开 USB 收发器，稍等几分钟再尝试连接，这样可能会解决问题。此外也可以尝试安装一个新的 Linux 内核。

6.3. 后置 LED(D15, D16) 在系统启动时不亮

6.3.1. 两个背部后置 (D15,D16) 都不亮

请在一个稍暗的地方仔细观察，如果仍看不到有点亮，可按以下步骤检查：
是否已经放进 4 块电池，并确认电池有充足的电量。
电池盒的连线是否正确(红线接 Bat+，黑线接 Bat-)
二极管 D9(1 N4001)极性是否正确
R22 使用的阻值是否 470Ω(黄色，紫色，褐色，金色)
检查 R18, R19 , R20 , R21：
4.7KΩ(黄色，紫色，红色，金色)
1 KΩ (褐色，黑色，红色，金色)

6.3.2. 仅其中一个后置 LED 发亮

检查是否已经插入发光二极管(玫瑰色或红色) D13(左边)，D14(右边)，和光感器(透明或黑色外壳) T11(左边)，T12(右边)，确认它们焊接的位置及极性正确。

检查电阻的阻值 R18，R19(左边)和 R20，R21 (右边)。

4.7KΩ(黄色, 紫色, 红色, 金色)
1 KΩ (褐色, 黑色, 红色, 金色)

并检查这些电阻的焊接位置是否正确(与电路板上的标识相对应)。

6.3.3. 状态 LED(D12) 在启动之初没有点亮红色及绿色

状态 LED 没有点亮 → 见 6.4 节

状态 LED 光亮跳动，可能由于电池电压过低，需更换电池。

如果不是电池电压的问题，则需要检查电阻 R12 和 R13。

12 KΩ (褐色, 红色, 黑色, 红色, 褐色)
10 KΩ (褐色, 黑色, 黑色, 红色, 褐色)

6.4. 某一显示元件不工作

处理器是否已经正确装上(注意极性)。

6.4.1. 状态 LED D12 不工作

检查 D12 的安装极性。
检查电阻 R10，R31 的阻值

470Ω (黄色, 紫色, 褐色, 橙色)

也可用以下的方法作判断：取开处理器(IC1)并把它的 IC 插座的 Pin7(VCC)脚分别与 Pin14 脚(状态 LED 应会点亮为绿色)和 Pin4 脚(状态 LED 应会点亮为红色)用一根电线相连。如果结果如上面描述，则要么是处理器已损坏，要么是电路板的某根连线已经断开。

6.4.2. 前置 LED D11 不工作

检查 D11 的安装极性。
检查电阻 R9 的阻值

220Ω (红色, 红色, 褐色, 金色)

也可用以下的方法作判断：取开处理器(IC1)并把它的 IC 插座的 Pin7(VCC)脚与 Pin12 脚(前置 LED 应会点亮为红色)用一根电线相连。如果结果如上面描述，则要么是处理器已损坏，要么是电路板的某根连线已经断开。

6.4.3. 左边的后置 LED D15 不工作

检查 D15 的安装极性。
检查电阻 R19，R18 的阻值
1KΩ (褐色, 黑色, 红色, 金色)
4,7 KΩ (黄色, 紫色, 红色, 金色)
也可用以下的方法作判断：取开处理器(IC1)并把它的 IC 插座的 Pin7(VCC)脚与 Pin24 脚(左边的后置 LED 应会点亮为红色)用一根电线相连。如果结果如上面描述，则要么是处理器已损坏，要么是电路板的某根连线已经断开。

6.4.4. 右边的后置 LED D16 不工作

检查 D16 的安装极性。
检查电阻 R21，R20 的阻值
1KΩ (褐色, 黑色, 红色, 金色)
4,7 KΩ (黄色, 紫色, 红色, 金色)
也可用以下的方法作判断：取开处理器(IC1)并把它的 IC 插座的 Pin7(VCC)脚与 Pin23 脚(右边的后置 LED 应会点亮为红色)用一根电线相连。如果结果如上面描述，则要么是处理器已损坏，要么是电路板的某根连线已经断开。

6.5. 线性跟踪器(T9,T10)不工作

检查 T9，T10 的安装极性。
检查电阻 R14，R15 的阻值
20KΩ (红色, 黑色, 橙色, 金色)
检查 R15，R23 与 R28 的焊接位置是否恰当。
取开处理器(IC1)，使用万用表分别检查 IC 插座的 Pin25 脚与 Pin26 脚判别 T9,T10 返回的信号是否正确(黑暗时约为 0V，光亮时约等于 VCC)

6.6. 某个触碰开关不工作

6.6.1. 未按下触碰开关，但现象就如已经按下了某些触碰开关

先检查 R12 和 R13
12 KΩ (褐色, 红色, 黑色, 红色, 褐色)
10 KΩ (褐色, 黑色, 黑色, 红色, 褐色)
再检查 R24, R25, R26, R27, R28, R29, R30, R32
1 KΩ (褐色, 黑色, 黑色, 褐色, 褐色)
2 KΩ (红色, 黑色, 黑色, 褐色, 褐色)
8,2 KΩ (灰色, 红色, 黑色, 褐色, 褐色)
16 KΩ (褐色, 蓝色, 黑色, 红色, 褐色)
33 KΩ (橘色, 橘色, 黑色, 红色, 褐色)
68 KΩ (蓝色, 灰色, 黑色, 红色, 褐色)
2 KΩ (红色, 黑色, 黑色, 褐色, 褐色)

6.6.2. 按下触碰开关时现象显示错位

问题可能是与触碰开关相连的电阻错误调换
检查 R24, R25, R26, R27, R28, R29, R30, R32

1 KΩ (褐色, 黑色, 黑色, 褐色, 褐色)
2 KΩ (红色, 黑色, 黑色, 褐色, 褐色)
8,2 KΩ (灰色, 红色, 黑色, 褐色, 褐色)
16 KΩ (褐色, 蓝色, 黑色, 红色, 褐色)
33 KΩ (橘色, 橘色, 黑色, 红色, 褐色)
68 KΩ (蓝色, 灰色, 黑色, 红色, 褐色)

6.6.3. 仍然不能良好运作

先检查 R23，R24，再检查 R12，R13 和 C7

1MΩ (褐色, 黑色, 绿色, 金色)
1KΩ (褐色, 黑色, 黑色, 褐色, 褐色)
12KΩ (褐色, 红色, 黑色, 红色, 褐色)
10KΩ (褐色, 黑色, 黑色, 红色, 褐色)
220 μF/10V

6.7. 光感器 (行程计) 不工作

6.7.1. 所有光感器(行程计)不工作

检查电阻 R22

470Ω (黄色, 紫色, 褐色, 橙色)

检查 D13 和 D14 的朝向。D13 和 D14 是玫瑰色或灰色的并在一边的中间有一个小突点的双脚元件。有突点的一面必须指向电路板的外面。

6.7.2. 左边光感器(行程计)不工作

检查电阻 R18

4,7 KΩ (黄色, 紫色, 红色, 金色)

检查 T11 的朝向。T11 是透明或黑色的并在一边的中间有一个小突点的双脚元件。有突点的一面必须指向电路板的外面。

6.7.3. 右边光感器(行程计)不工作

检查电阻 R20

4,7 KΩ (黄色, 紫色, 红色, 金色)

检查 T12 的朝向。T12 是透明或黑色的并在一边的中间有一个小突点的双脚元件。有突点的一面必须指向电路板的外面。

取开处理器(IC1),分别检查 IC 插座的 Pin24 脚与 Pin23 脚判别 T11,T12 返回的信号是否正确。(黑暗时约为 0V，光亮时约等于 VCC)

6.8. 某个马达不运转

6.8.1. 两个马达都不运转

检查 IC3 的位置及极性是否正确。

6.8.2. 左边的马达不运转或只能往一个方向运转

驱动左边马达的整流桥由三极管 T1，T2，T3，T4(注意放置的位置是否恰当)及二极管 D1，D2，D3，D4(注意极性)电阻 R1，R2，R3，R4 构成。

BC327-4 或 BC328-40，BC337-40 或 BC338-40，BC327-4 或 BC328-40，BC337-40 或 BC338-40

1KΩ (褐色，黑色，黑色，褐色，褐色)

6.8.3. 右边的马达不运转或只能往一个方向运转

驱动右边马达的整流桥由三极管 T1，T2，T3，T4(注意放置的位置是否恰当)及二极管 D1，D2，D3，D4(注意极性)电阻 R1，R2，R3，R4 构成。

BC327-4 或 BC328-40，BC337-40 或 BC338-40，BC327-4 或 BC328-40，BC337-40 或 BC338-40

1KΩ (褐色，黑色，黑色，褐色，褐色)

6.8.4. 马达运转方向相反

检查马达与电路板相连的电线，电线的极性极有可能接反。

6.9. 红外接口

6.9.1. ASURO 不能发送数据

检查红外二极管 D10 的极性。
检查电阻 R16 是否为 220Ω (红色，红色，褐色，金色)

6.9.2. ASURO 不能接收数据

首先需要确保 ASURO 与红外收发器之间有可视距离(最大距离不应超过 50 cm)
而且红外收发器需要经过测试验证。(见第 6.1 章)
检查 C2 的焊接位置与极性。
检查电阻 R17 和电容 C2。

470Ω (黄色，紫色，褐色，金色)

100nF (标识 104)

如果经过上面的检查仍然未发现问题，则可检查 IC2 的焊接，由于这个元件不能受热，故在焊接过程很容易损坏，这时可以尝试更换这个 IC(SFH 5110-36)。当 PC 与 ASURO 之间的数据传输经常出错，则可以通过调整收发器的可调电阻 TR1 改善。

6.9.3. 如果此红外接口工作仍未正常

检查电容 C8 的极性。

220μF/至少耐压 10V

当 PC 与 ASURO 之间的数据传输经常出错，则可以通过调整收发器的可调电阻 TR1 改善。

快干胶有可能引起皮肤的过敏反应，所以必须避免与这些物料接触，塑胶手套可以保护你的双手，如果保护失效的话，立即用肥皂和水彻底洗净接触过的身体。快干胶一般是用作表面粘合，如果你的手指被粘结在一起的时候，你可以用暖水、肥皂来软化粘结，但需要有耐性，绝对小心不要让快干胶接触到你的嘴唇和眼帘。当你使用快干胶的时候，要控制所有下意识的用手擦面和眼的动作！

7. 最后的调整

稍微润滑一下轴承，把贴有黑白分块标记的齿轮安装在短轴上，粘有齿轮的车轮则安装在长轴上并由一个金属环锁紧。轻轻移动被临时固定的马达，使之与相应的齿轮配合良好，并使两组齿轮都能顺滑地运转起来。

(运行自检程序并观察由马达带动的两组齿轮是否能顺滑地运转起来)。
如果马达的位置确定下来，则涂适量快干胶在马达与 PCB 板上并压紧。等待快干胶完全粘合牢固之后，可再使用扎带把马达绑紧在 PCB 板上。

把加工好的半球状乒乓球用快干胶粘合在 PCB 板底部的中间位置，并使它刚好位于线性跟踪器之后(见图 7.1)。注意快干胶需要一点时间才能完全粘合。

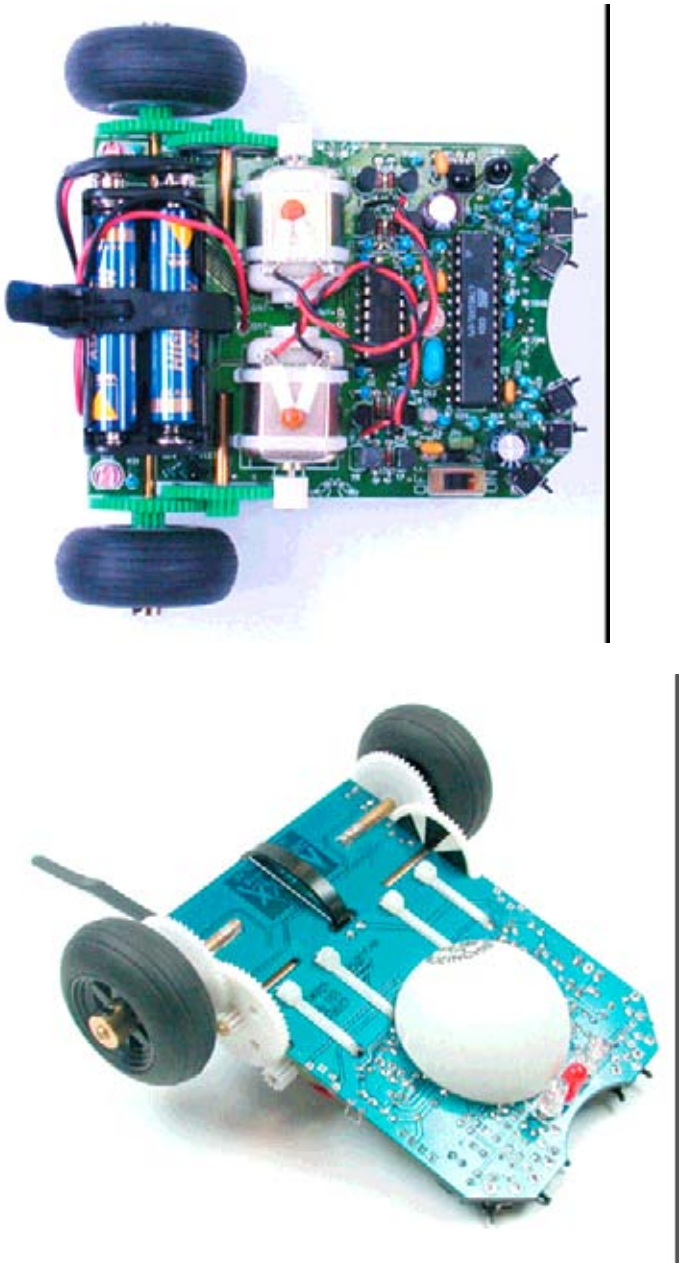


图 7.1.: 组装完成的 ASURO

Part III 软件

8. 软件安装和初始阶段

插入 ASURO 光盘，安装程序会自动进行，否则可以通过浏览器打开光盘。选择语言后，在软件安装菜单中会出现所有与 ASURO 相关的程序。在使用这些程序之前，必须首先在电脑上安装完毕。安装这些程序需要管理员权限，如未用管理员身份登陆只需重启后以管理员身份重新登陆即可。

在整个安装过程中会具体执行以下内容：

1. 安装用于传送程序到 ASURO 的 FLASH 软件。
2. 安装一个程序编辑器（Programers Notepads 2，PN2）和一个编译器（WinAVR）。
3. 从光盘拷贝示例程序会到电脑硬盘上。
4. 用程序编辑器（PN2）生成用于 Make 与 Clean 文件的输入菜单 。

8.1 Windows

8.1.1 FLASH 程序

拷贝一份 Flash 程序到硬盘上，例如放在 C:\ program\flash 目录。同样也可以直接在光盘上运行 Flash 程序，为方便一般可在桌面建立一个到 Flash 程序的链接。

8.1.2 安装程序编辑器和编译器

安装编译器时必须以管理员身份登陆师 (因为在安装过程需要改动注册表)。如未用管理员身份登陆只需重启后以管理员身份重新登陆即可。

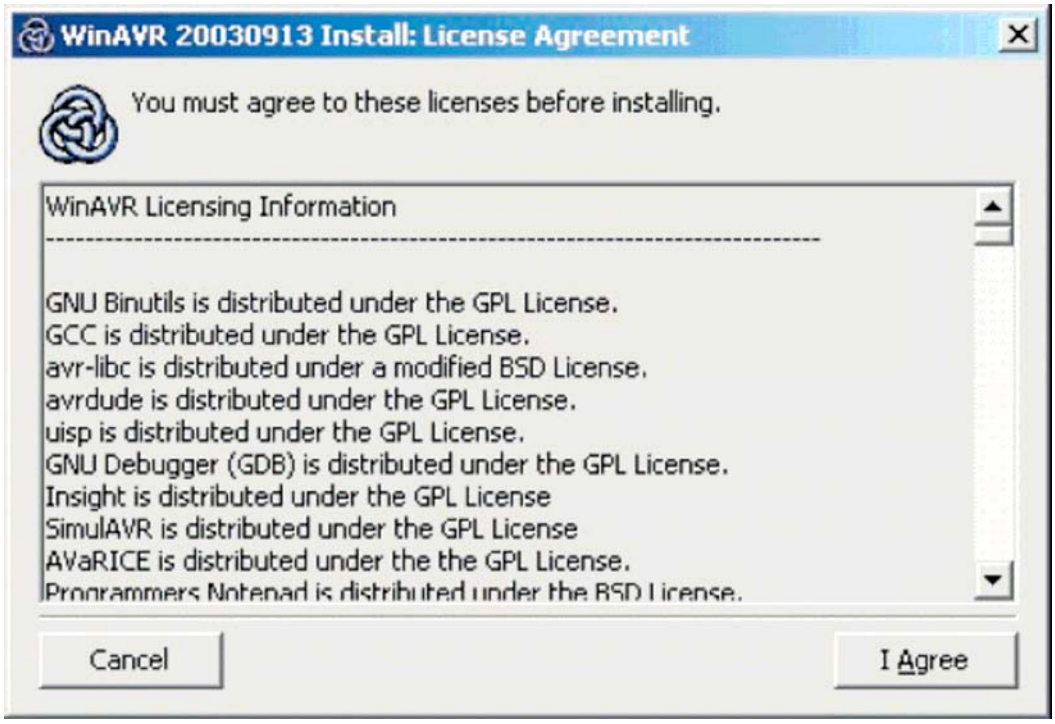
点击安装程序的图标。

WinAVR(20030913)

如果点击后没有启动安装可使用浏览器打开 CD 光盘。

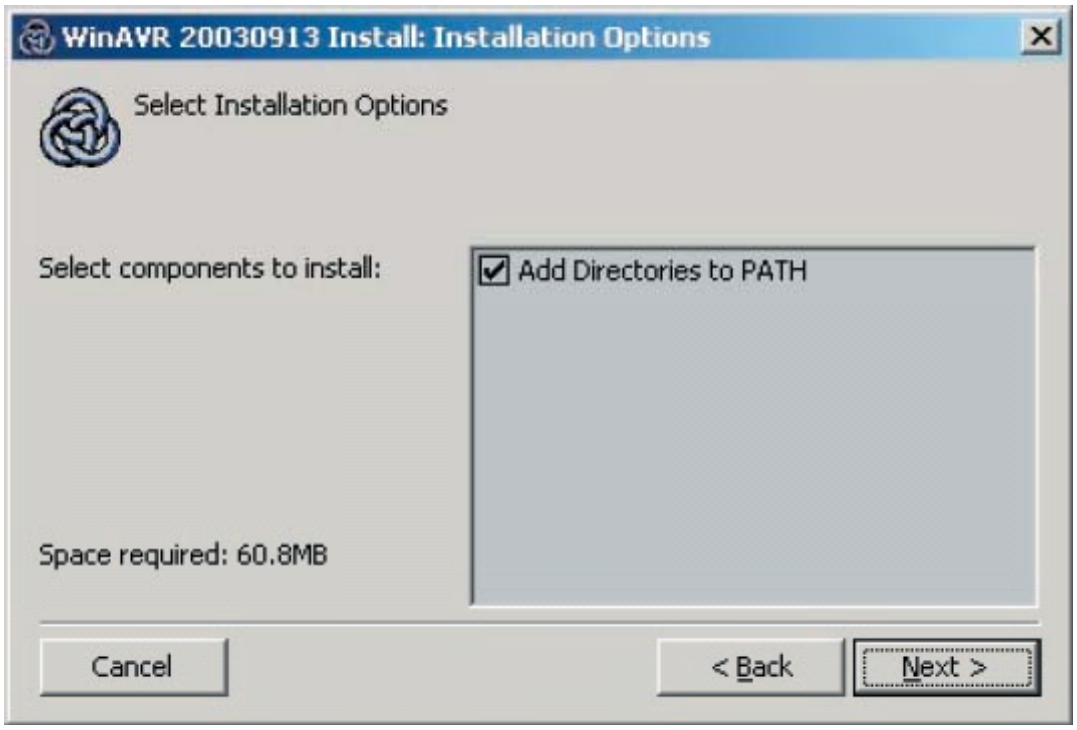
Windows 是微软公司的注册商标

这时屏幕会出现如下画面：



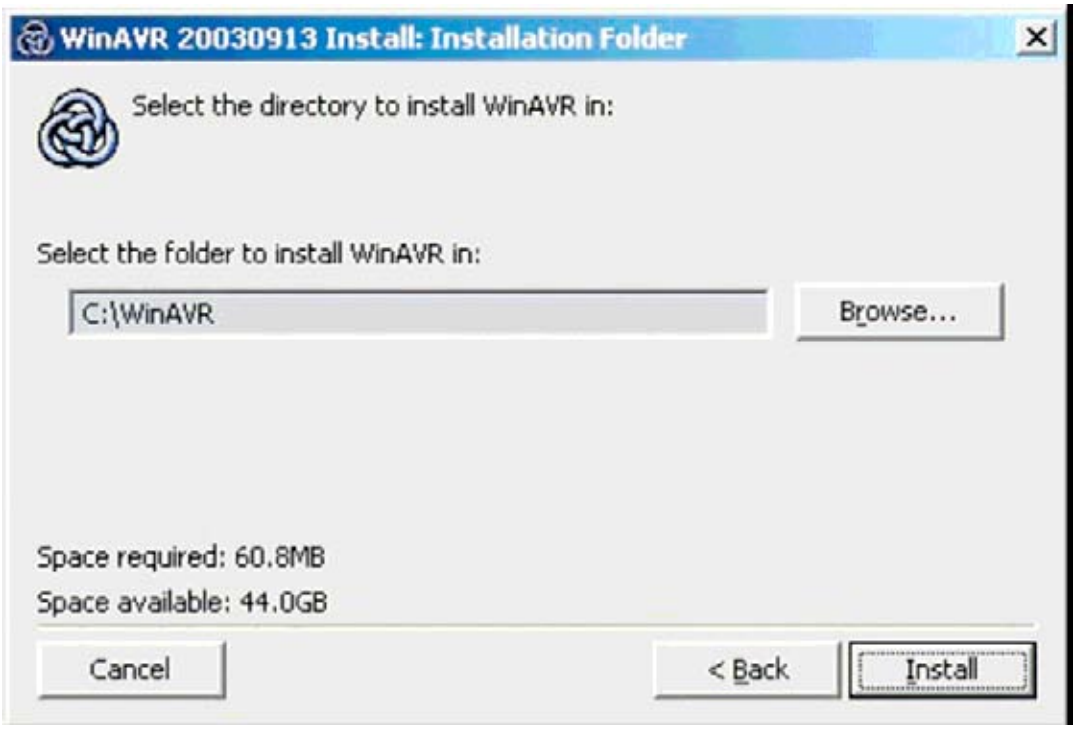
点击[I Agree]

屏幕显示：



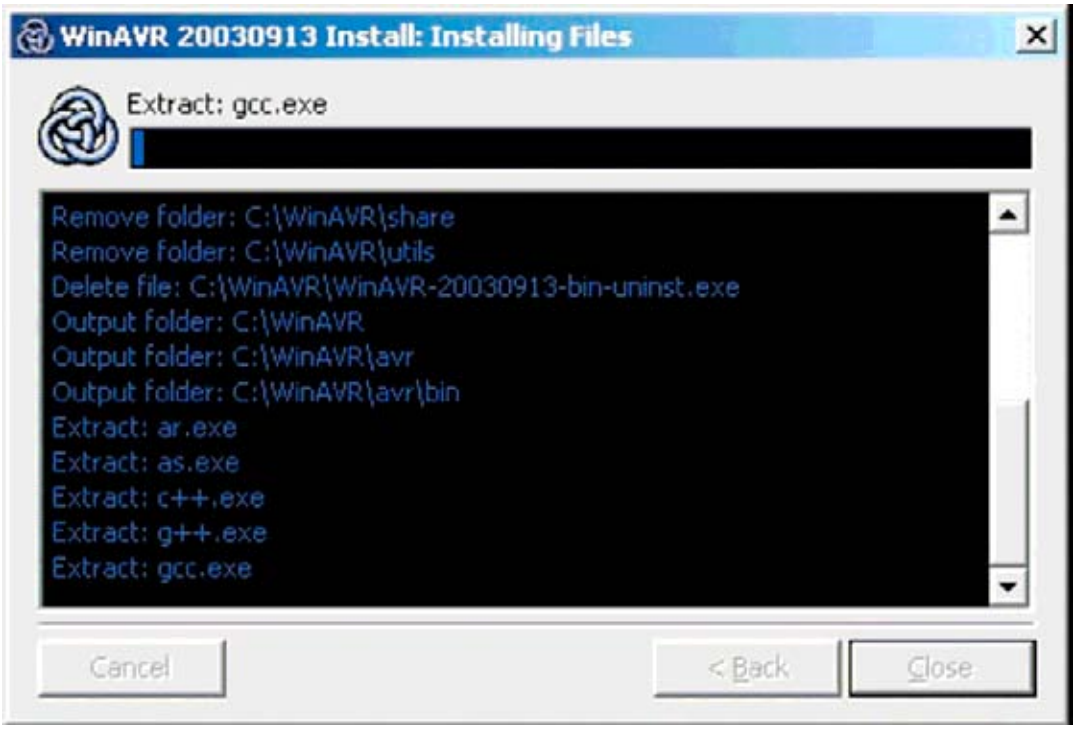
点击[Next]

下一个窗口显示：



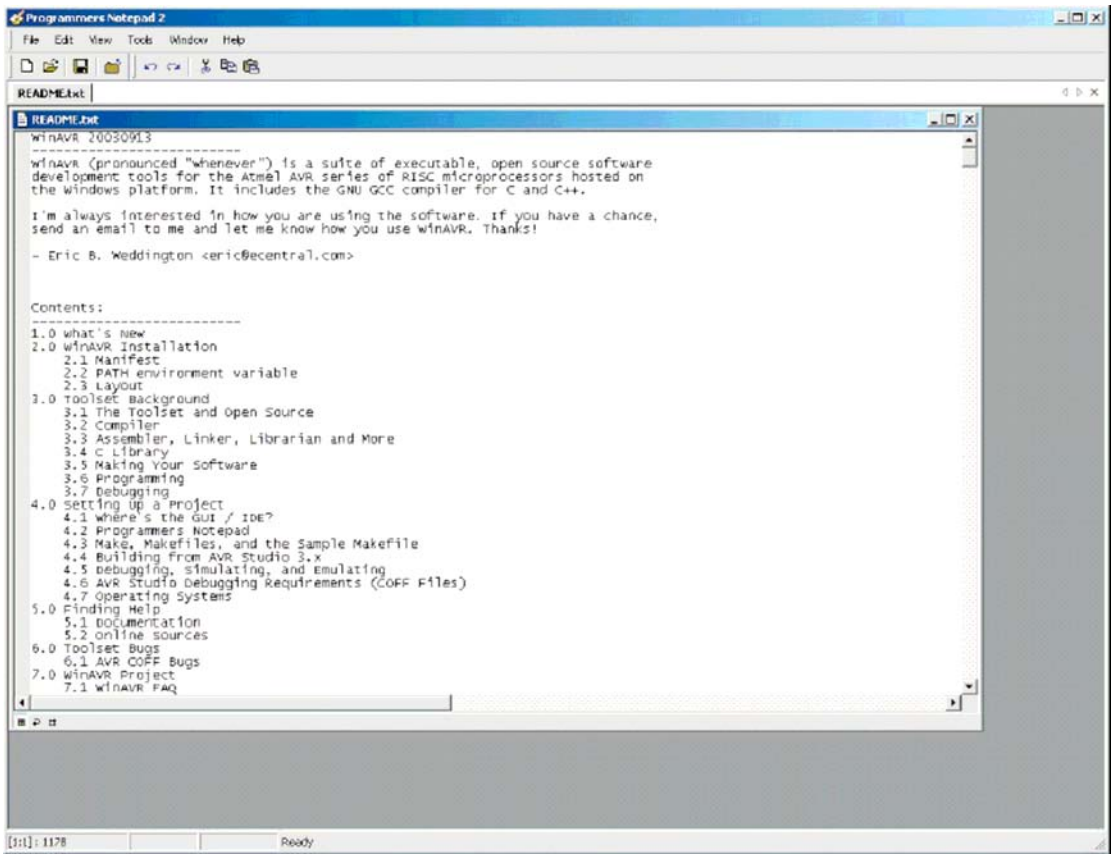
点击[Install]

下一个窗口显示：



稍等待一阵…

将见到 Programers Notepads 2（PN2）编辑器和 readme.txt。



关闭 Programers Notepads 2（PN2）

在电脑桌面上出现“Programers Notepads 2”图标：

程序编辑器和编译器已经安装成功。

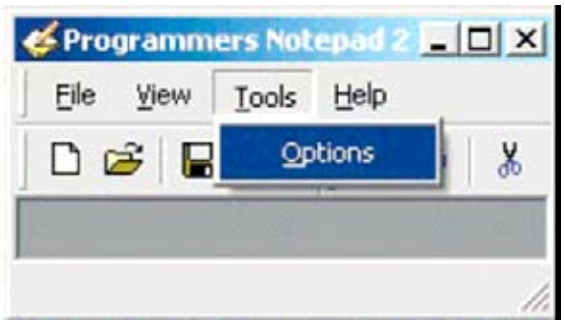
8.1.3. 从光盘拷贝示例程序会到电脑硬盘上

从 ASURO 光盘中拷贝文件夹‘ASURO_src’到硬盘(把它放在一个文件夹中，如‘C:\ ASURO_src’)。

如果文件已经设置了保护属性，按下鼠标右键选择属性页取消保护设置。

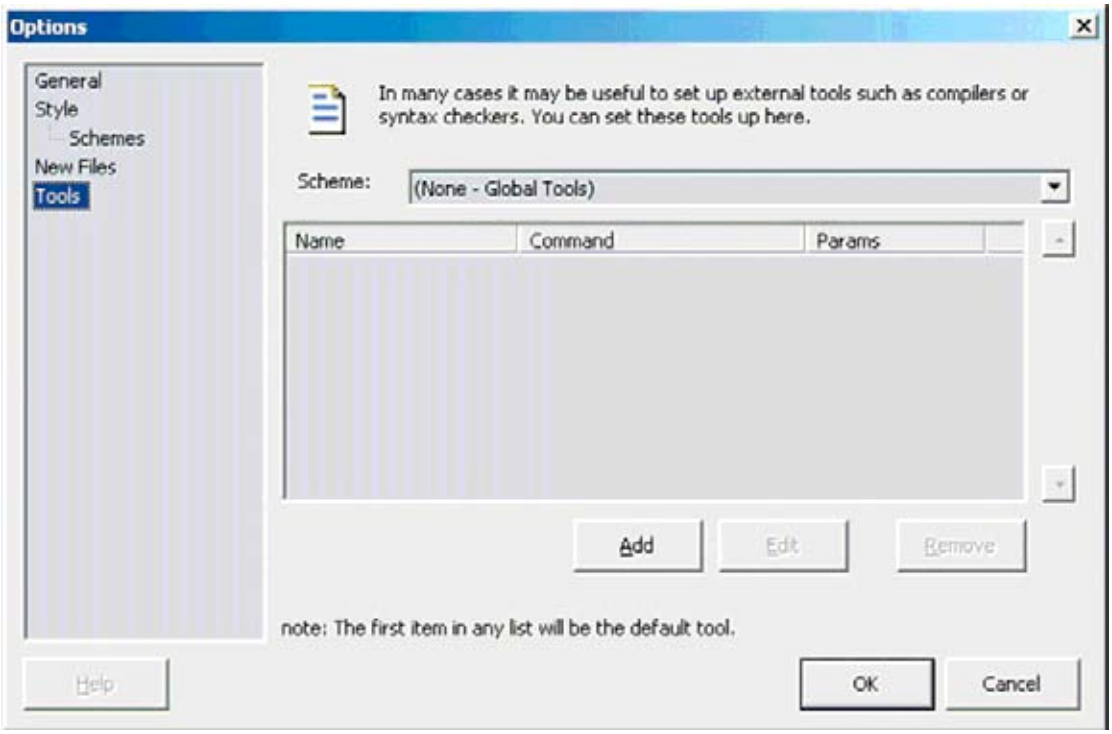
为程序编辑器的编译器的建立一个执行编译的菜单命令。

打开“Programers Notepads 2”程序(双击在电脑桌面上的图标)



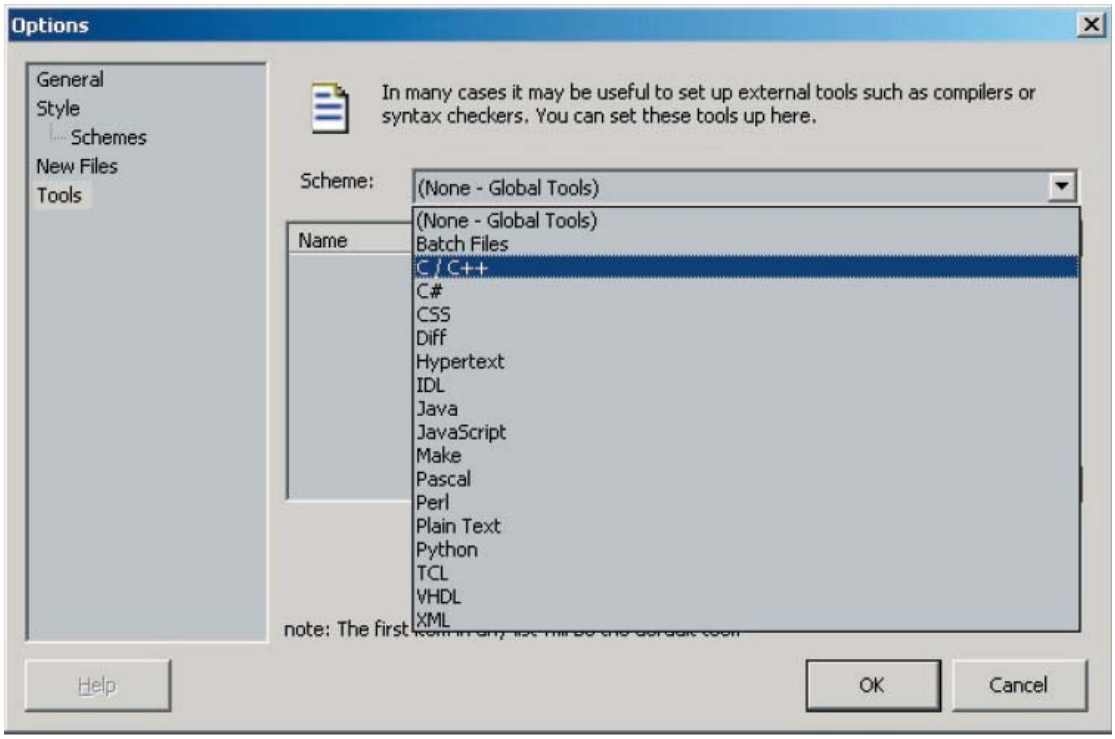
在菜单 Tools 中选择 Options 选项。

Options 选项页如下。

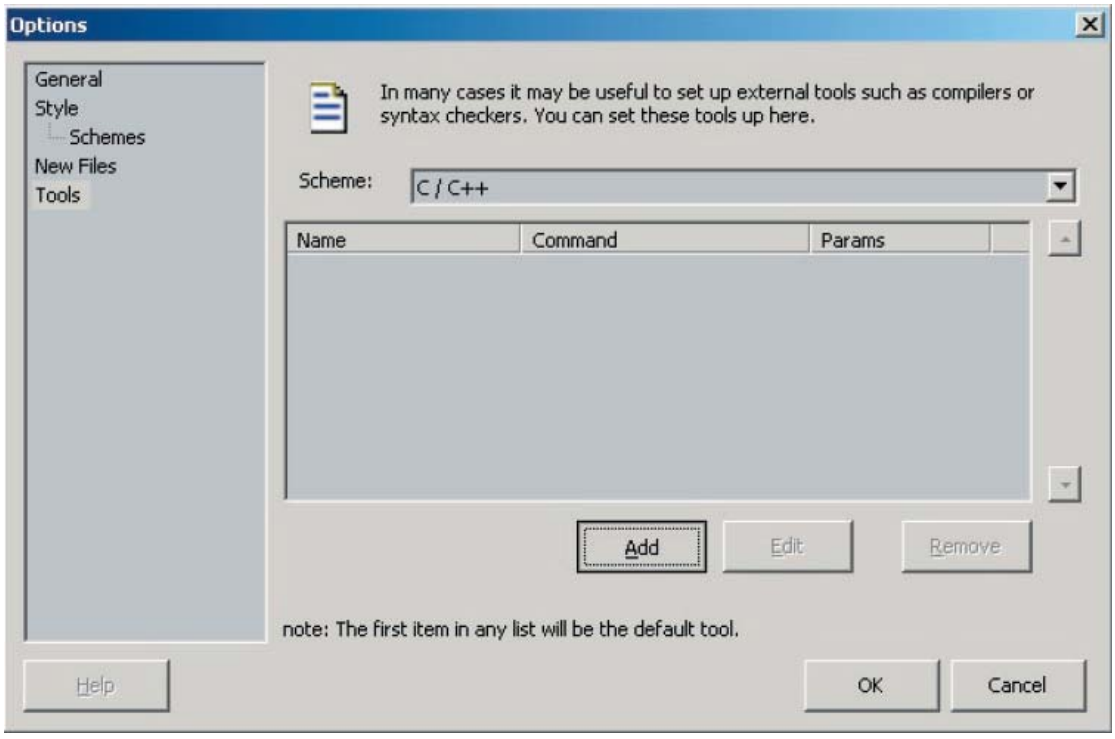


选择‘Tools’项。

在右边选择‘C/C++’项。

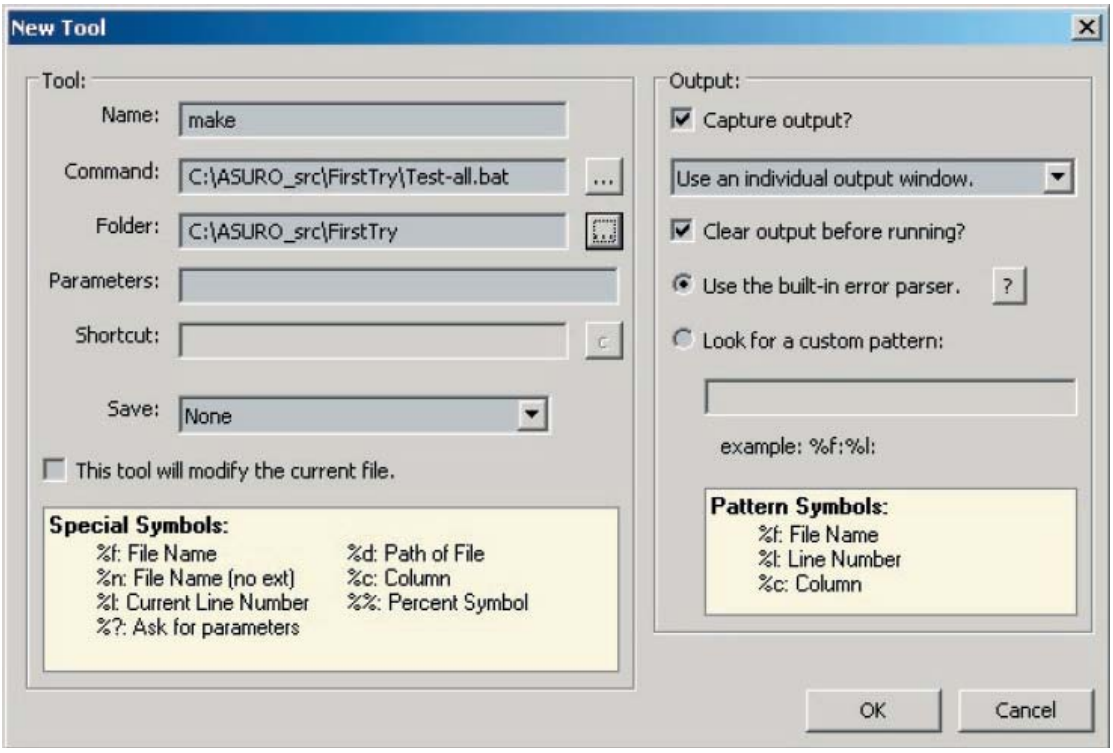


‘C/C++’项被选中



点击 [Add] (用于插入一个新的 Tool)

‘New Tool’窗口如下。



键入下列内容或通过浏览按钮进行选择 ...：

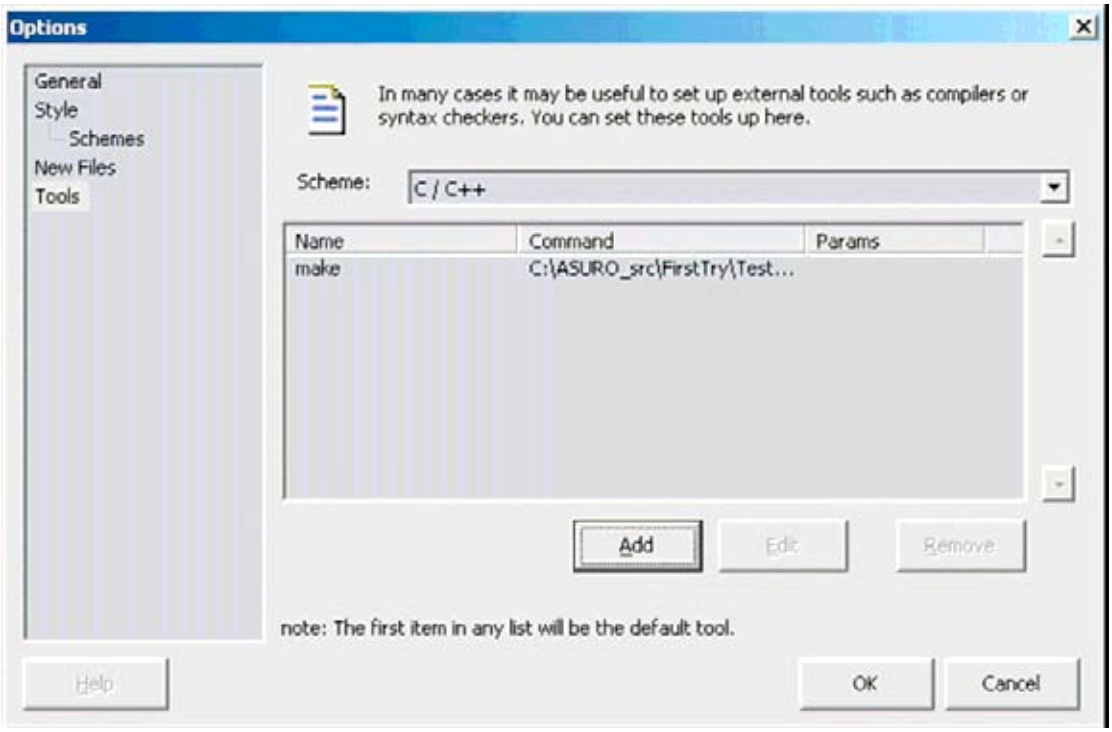
Name： make
Command： C:\ ASURO_src\FirstTry\Test-all.bat
Folder： C:\ ASURO_src\FirstTry

点击 [OK]

一个命名为 make 的 PN2-tool 出现在 Tools 窗口页
(当执行刚建立的名为 make 的 tool 时，批处理文件 Test-all.bat 就会运行，由 Test-all.bat 文件控制下编译的结果便会放在文件夹 C:\ ASURO_src\FirstTry 之内)。

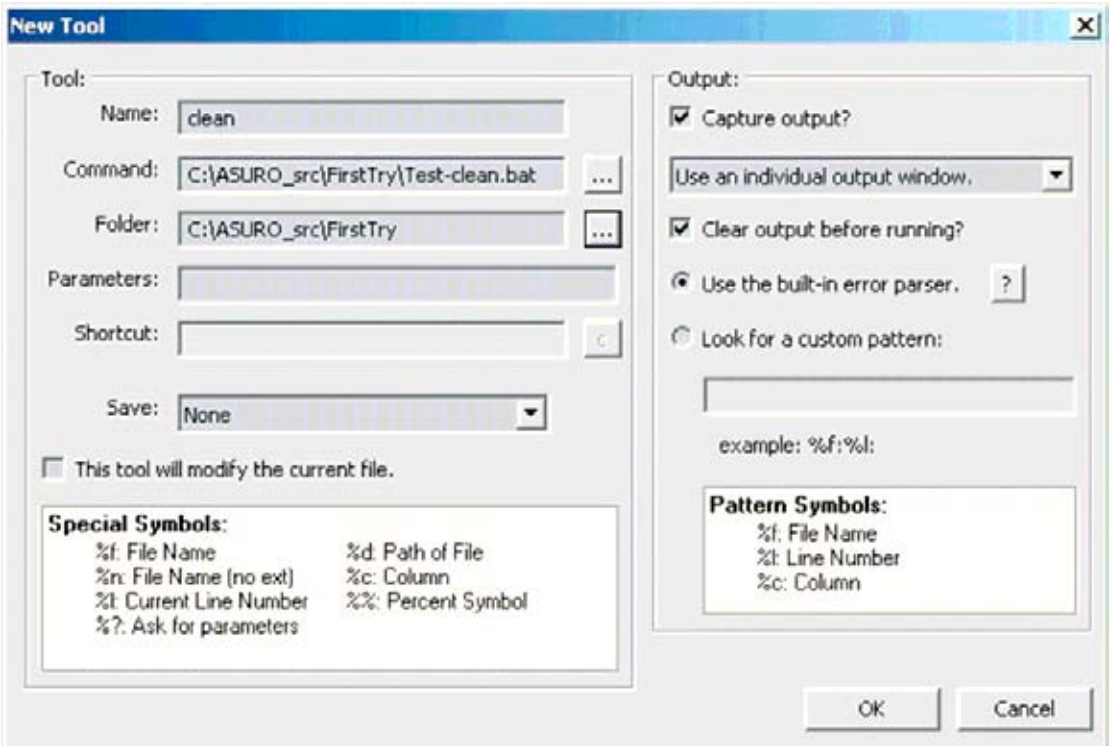
程序编辑器的菜单中建立一个名为‘clean’的 tool 相。

在主菜单的" Tools " 再次选择"Options" 然后在右边选择 "C/C++"项：



点击 [Add] 添加一个新的 tool：

‘New Tool’窗口如下。



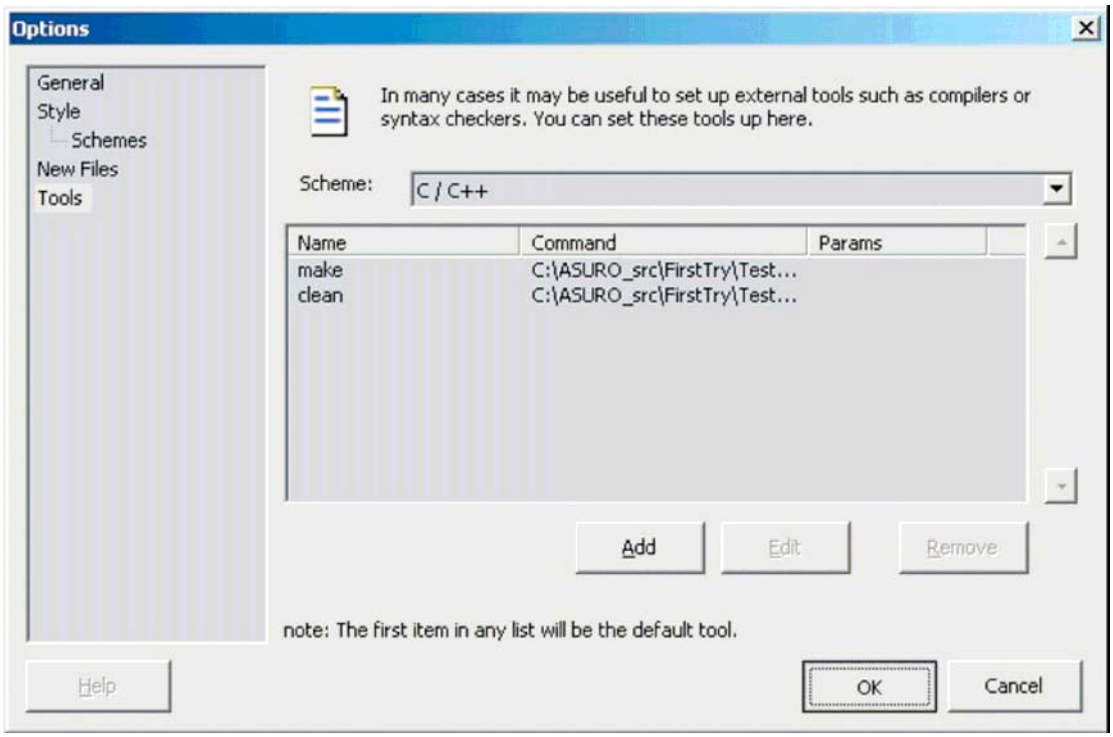
键入下列内容或通过浏览按钮进行选择  ：

Name： clean
Command： C:\ ASURO_src\FirstTry\Test-clean.bat
Folder： C:\ ASURO_src\FirstTry

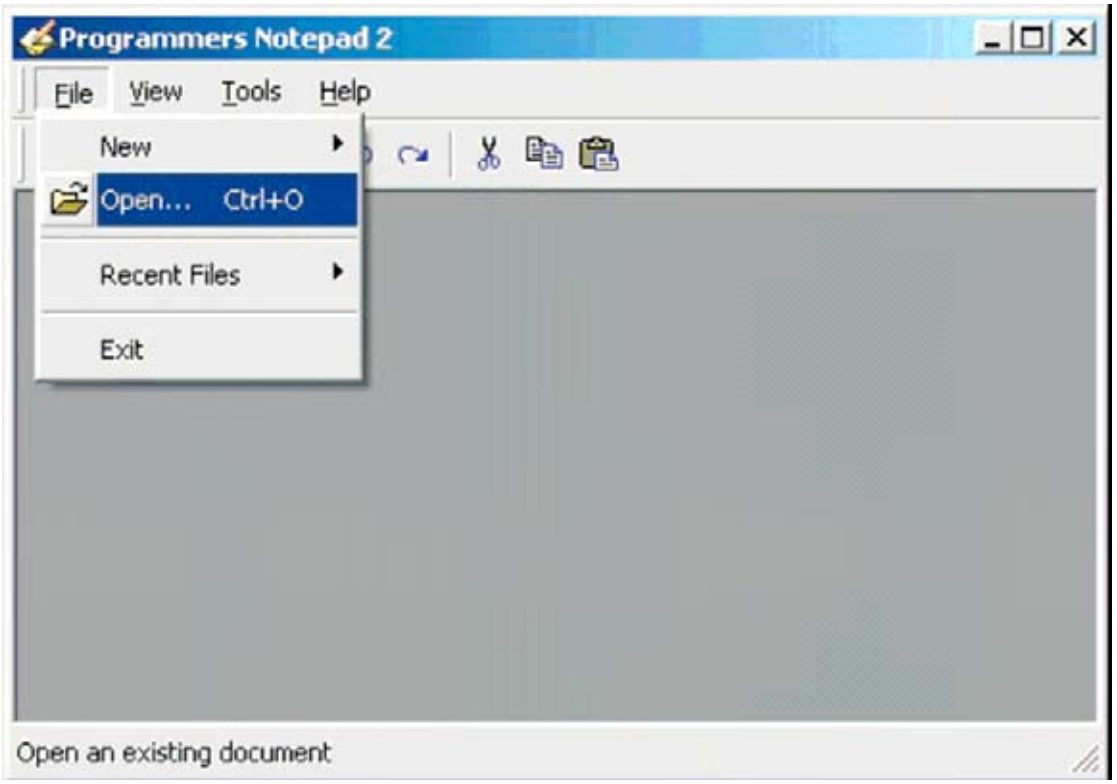
点击 [OK]

一个命名为 clean 的 PN2-tool 出现在 Tools 窗口页
(当执行刚建立的名为 clean 的 tool 时，批处理文件 Test-clean.bat 就会运行，它会自动清除位于文件夹 C:\ ASURO_src\FirstTry 内的编译结果)。

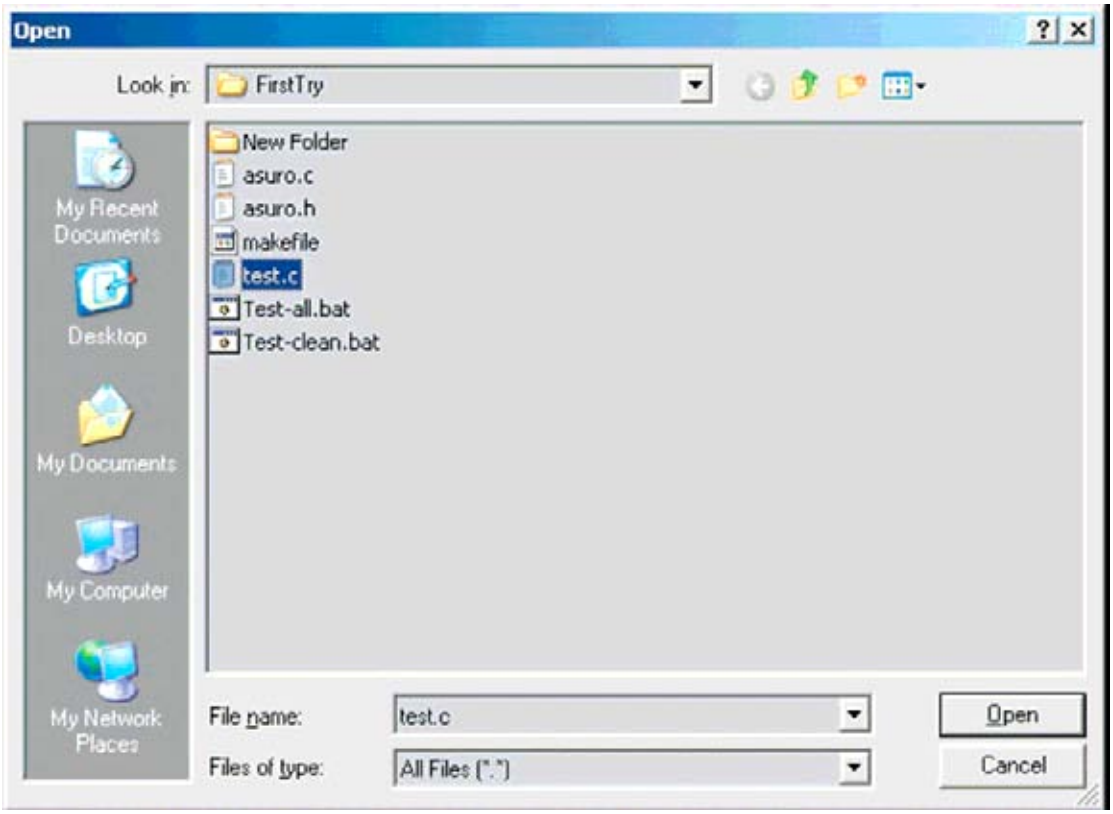
在 Options 页面的 Tools 选项下可以见到之前建立的两个 tool 文件‘make’和‘clean’。



点击 [OK]

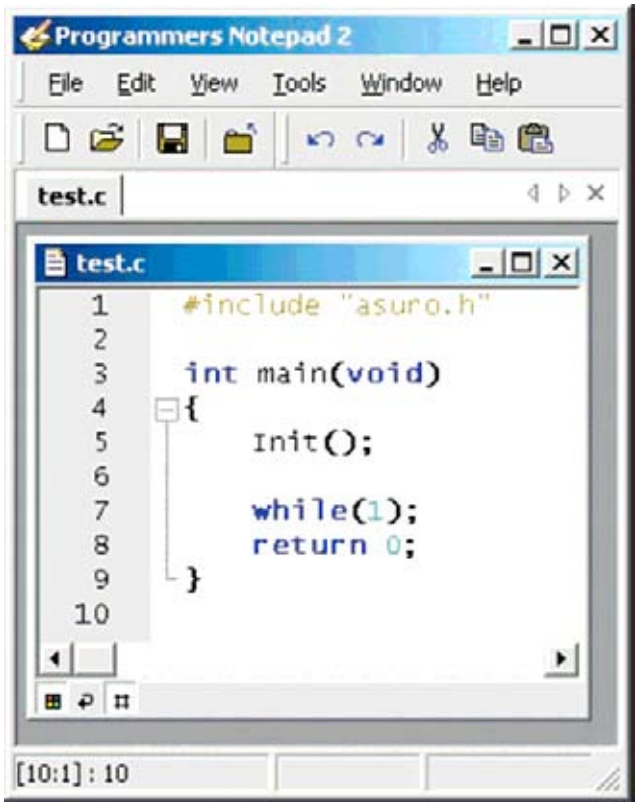


为了测试之前建立的两个文件，我们打开文件'C:\ ASURO_src\FirstTry\test.c'：

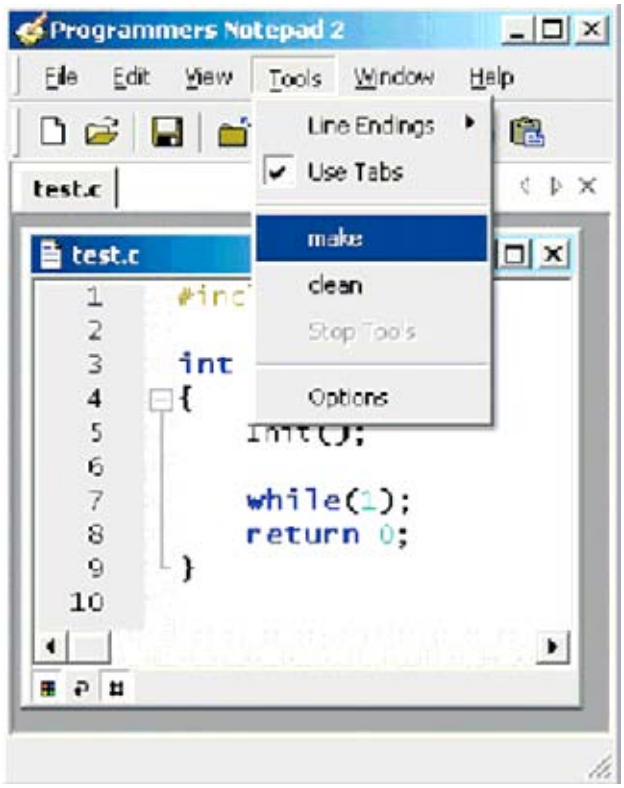


点击 [Open]。

文件 test.c 打开如下。



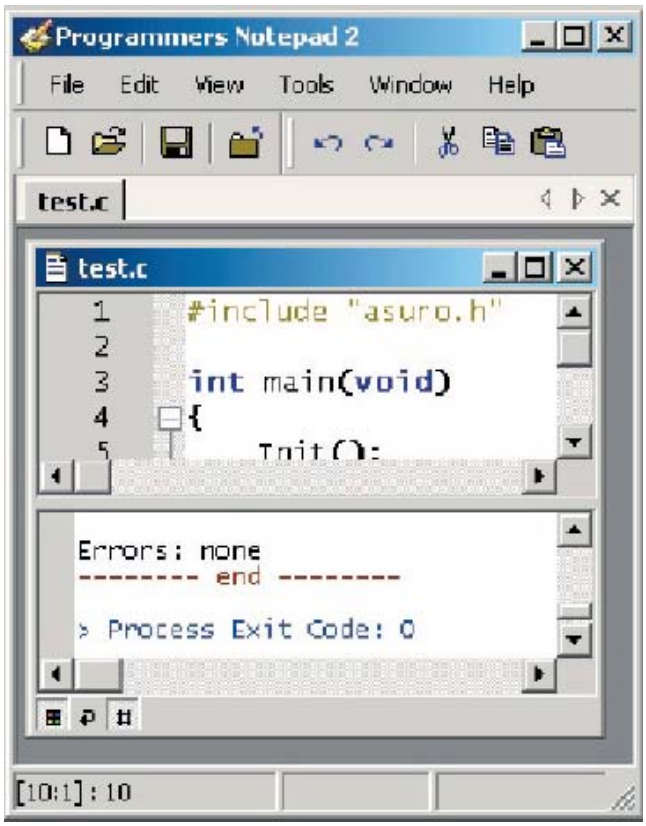
选择菜单 Tools



可见之前建立的两个 tool 即‘make’和‘clean’已经在菜单里。

点击‘make’项

文件 test.c(连同文件 asuro.c 一起)将会开始编译…



…如果当程序没有任何错误(这个示例程序已经过测试是完全正确的)，在窗口的底部将会出现信息：Errors：none

编译的结果
由文件 test.c(包含 asuro.c)生成了新文件 test.hex，这个新文件由从高级语言程序转换而来的机器码构成，机器码程序可以拷入 ASURO 的存储器中。这个程序并没有执行什么操作，下一步，我们会演示如何使用 Flash 程序把 test.hex 拷贝进 ASURO 中。

Make 命令的工作原理
菜单上的‘make’命令执行批处理文件 test-all.bat (批处理文件由多行的命令行构成，这些命令行会顺序地执行)。
在 Test.bat 文件中的命令‘make all’将会得到执行。‘make’命令会生成一个 make 文件，它位于 Test.bat 同一文件夹内。

Make 文件是文本类型的文件，它定义了程序编译的方法。在编译程序时，如果是单一文件时会比较操作，但对于较多文件构成的大程序，文件之间有很多的相互关联与链接关系时，使用 make 文件控制程序编译是非常必要的。

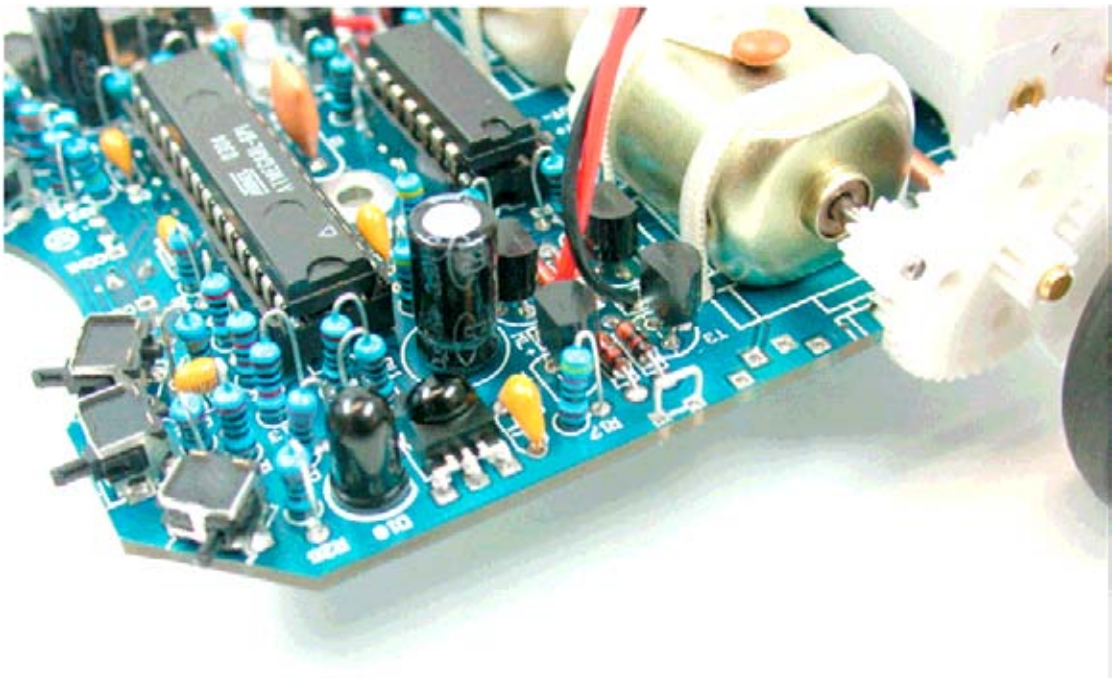
本示例程序的 make 文件控制 test.c 与 asuro.c 文件一起编译并生成一个新的.hex 后缀文件，这个新生成的文件可以拷贝进 ASURO 中运行。

注意! 这意味着如果 make 文件只是复制过来的且未经修改内容时，必须把程序名称命名为 test.c 。

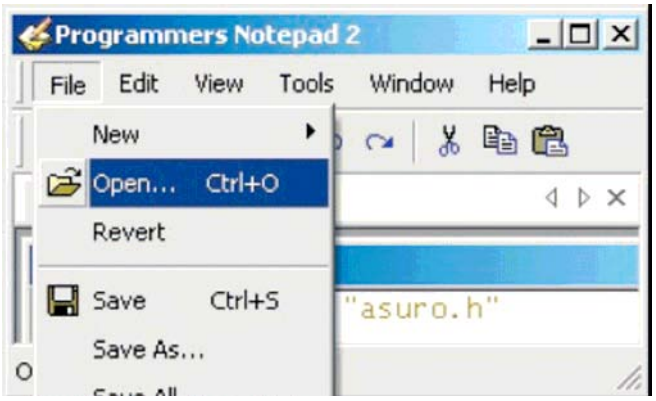
如果你想了解更多关于make文件的信息(如仅对ASURO编程可不需这样做) ，可以通过浏览如下网址http:// www.gnu.org/directory/make.html

关于 ASURO 编程的基本方法将会在第 9 章讲述。

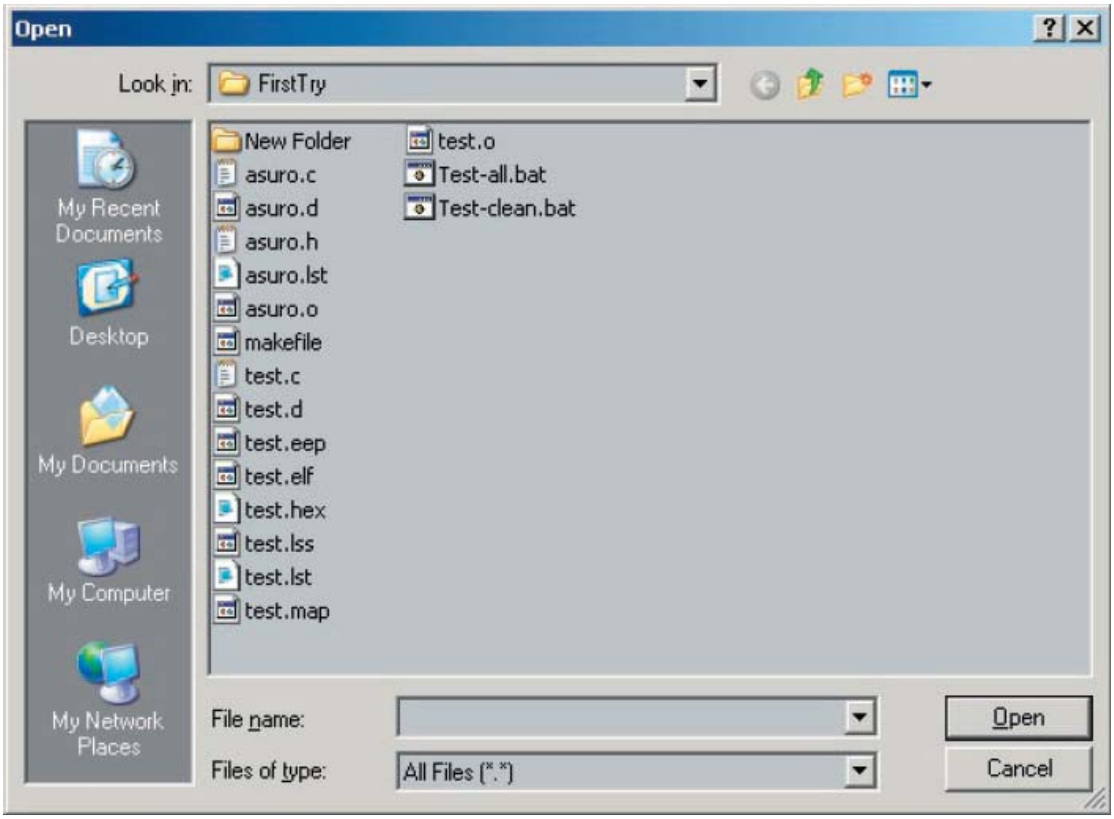
当编译完成一个程序的时候，会产生一些附加的文件。这些附加的文件仅在编译程序之时有作用，编译完成后他们是没有作用的。这些文件可用之前建立的 clean 命令清除。



选择如下菜单命令时

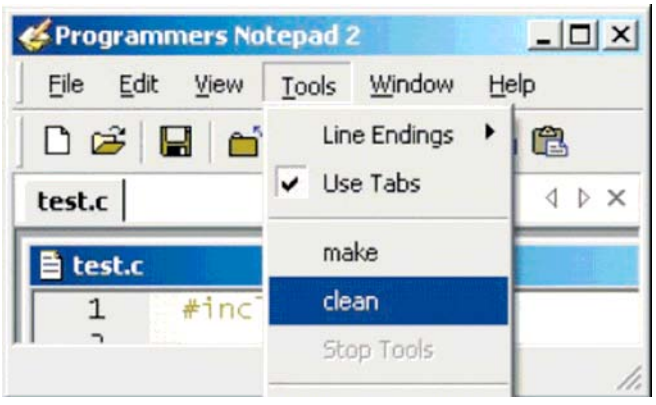


…你可以见到编译过程中所产生的数据文件…

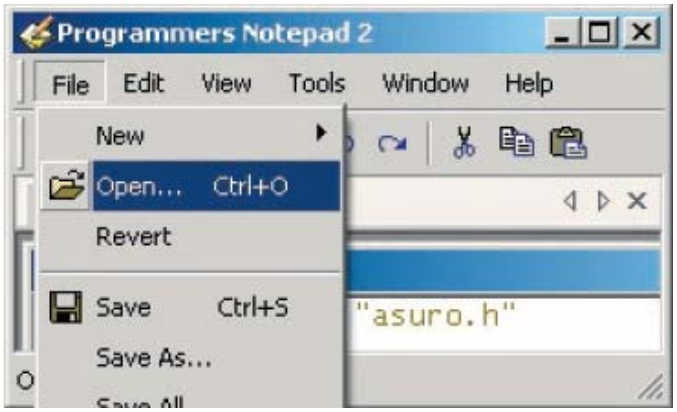


(点击[Cancel])

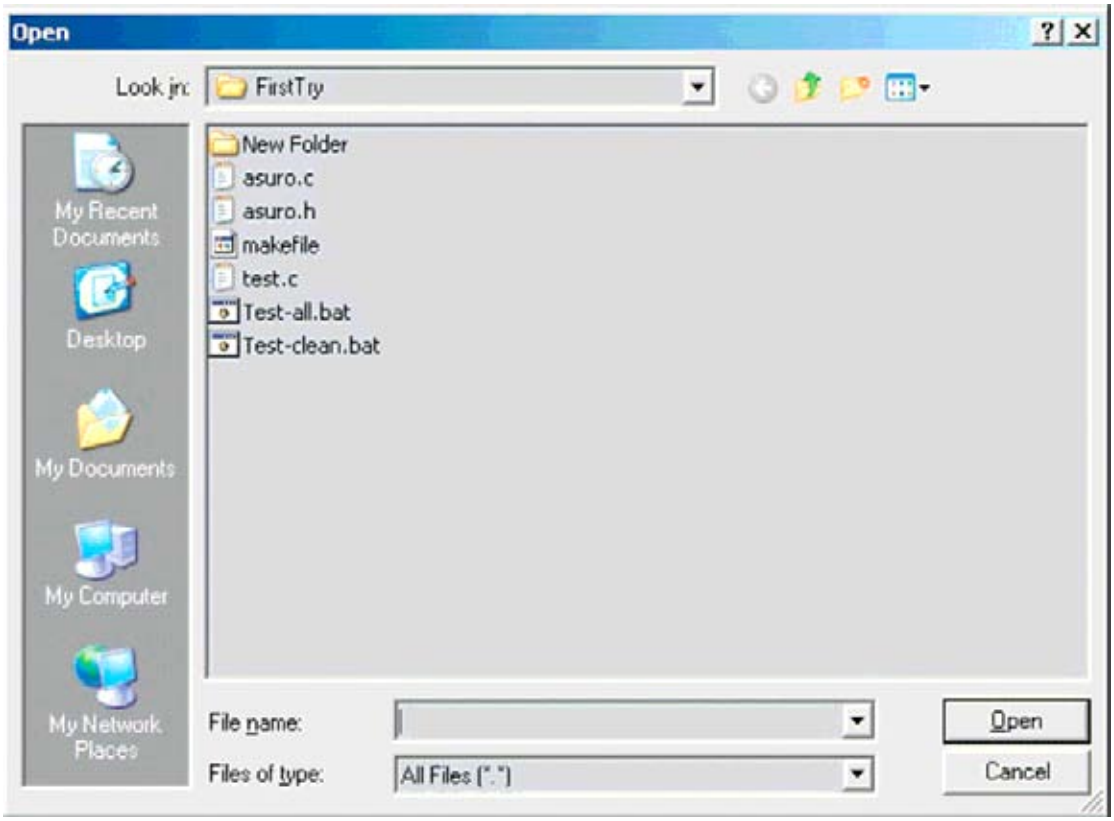
…在执行完"clean"命令之后…



…再次选择‘打开’命令…



…许多编译过程所产生的附加文件都除去。



为什么会如此？

菜单的“clean”命令调用批处理文件 Test-clean.bat。Make 开始用参数“clean”作处理，之后所以不再需要的数据文件都会被清除掉。

8.2. Linux

在 Linux 系统下安装本软件需要 root 权限。如果未是 root 权限则可退出登陆并再次以 root 身份登陆或是打开一个 shell 使用‘su’命令更改权限。

8.2.1 Flash – ASURO的编程软件

从光盘的软件目录里启动 Flash 程序并从"/linux/tools"目录拷贝两个 flash 工具" asuroflash"与"asurocon" 到 目 录 "usr/local/bin" 。 拷 贝 完 成 后 再 使 用 命 令 设 置 它 们 的 属 性 "chmod a+x/usr/local/bin/asurocon(和 asuroflash)"

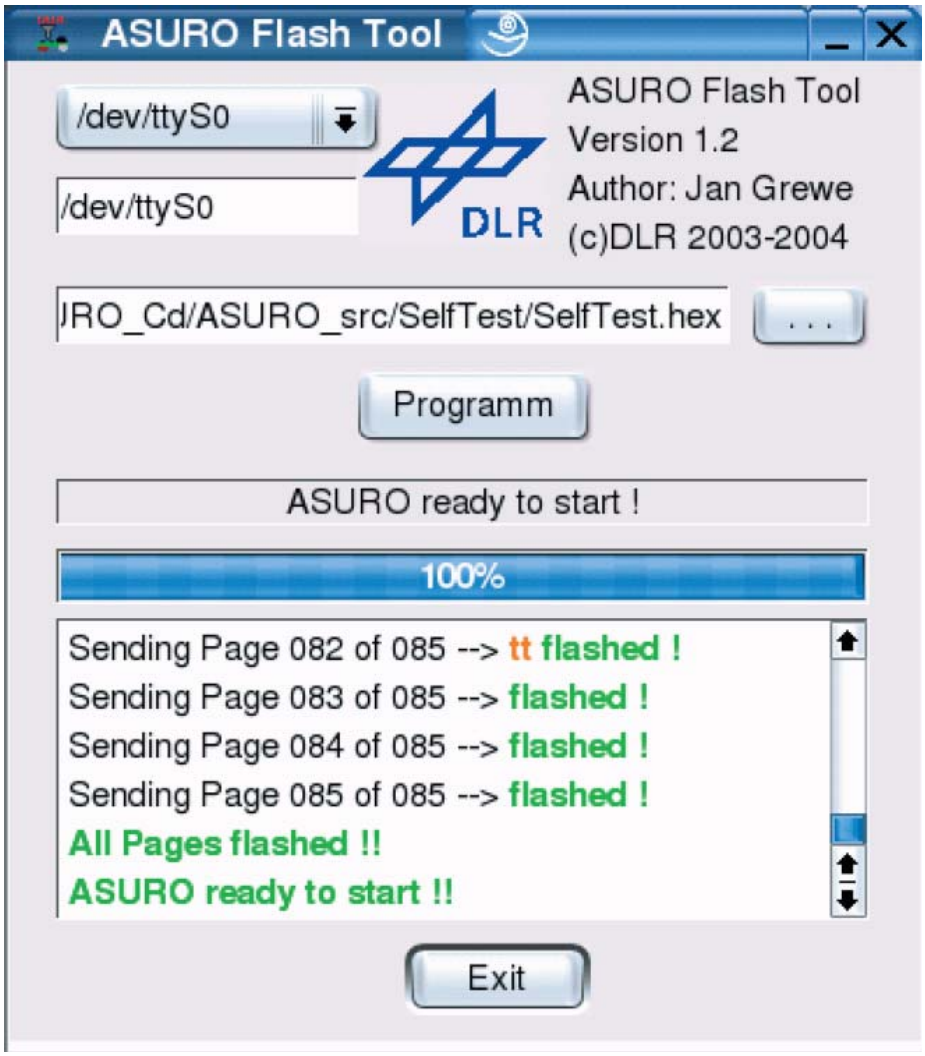


图 8.1.: Flash 程序

8.2.2 编译器

插入 ASURO 光盘，使用目录"/Linux/Compiler/"下的文件安装 AVR 处理器的 GNU 编译器：

1. avr-binutils-... .rpm
2. avr-gcc-... .rpm
3. avr-libc-... .rpm

这里的安装过程相当简单，仅需在根目录下使用命令：rpm –i <paket>i.rpm

对于程序编辑器，可使用 Exmacs，Kate 或 Kedit。可以从光盘的以下目录"/ASURO_src/FirstTry/"中拷贝演示程序用作实验，只需打开 shell 程序，改变文件目录并输入"make"命令。如果正确无误，则可见到如下窗口所示内容(见图 8.2)。

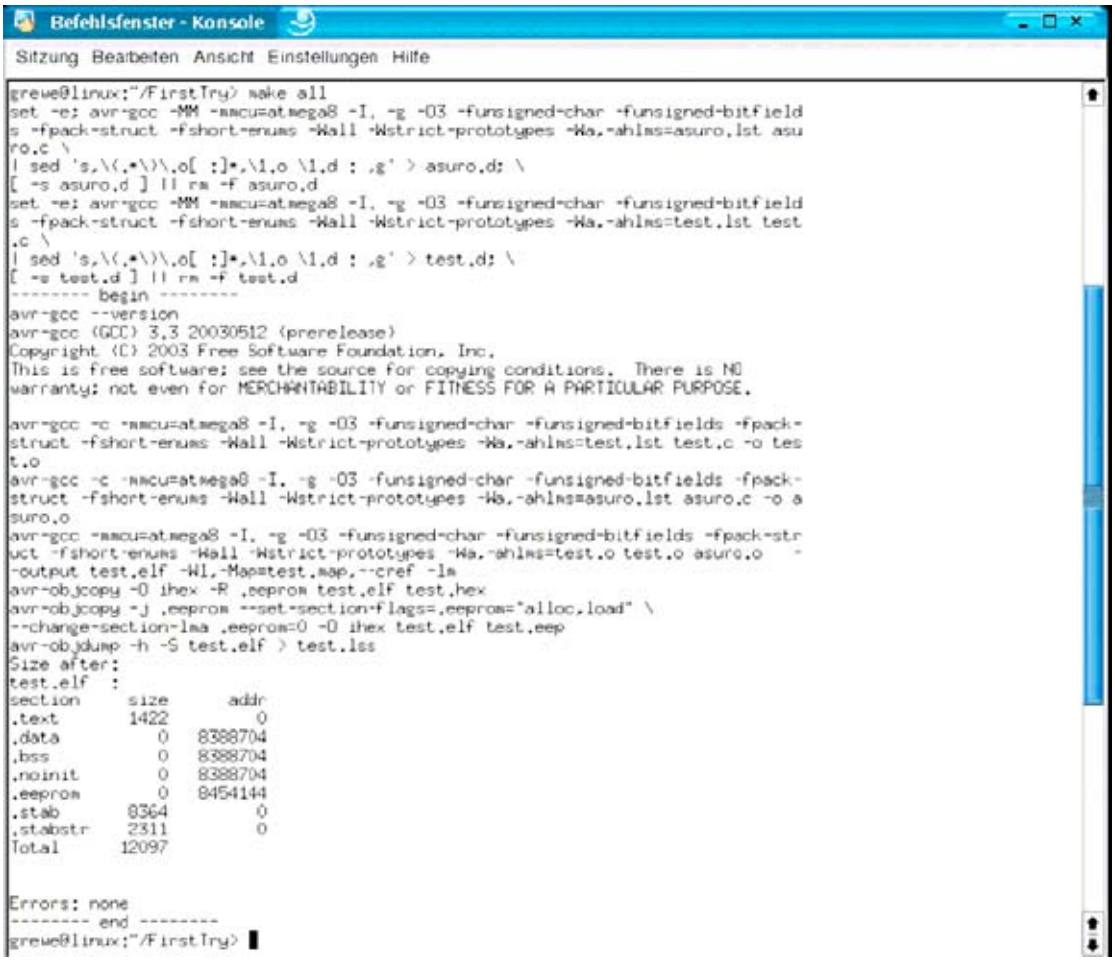


图 8.2.: **Make all**

8.3. Flash – ASURO的编程软件

在这一步中我们需要用到 Flash 程序.(见图 8.3)

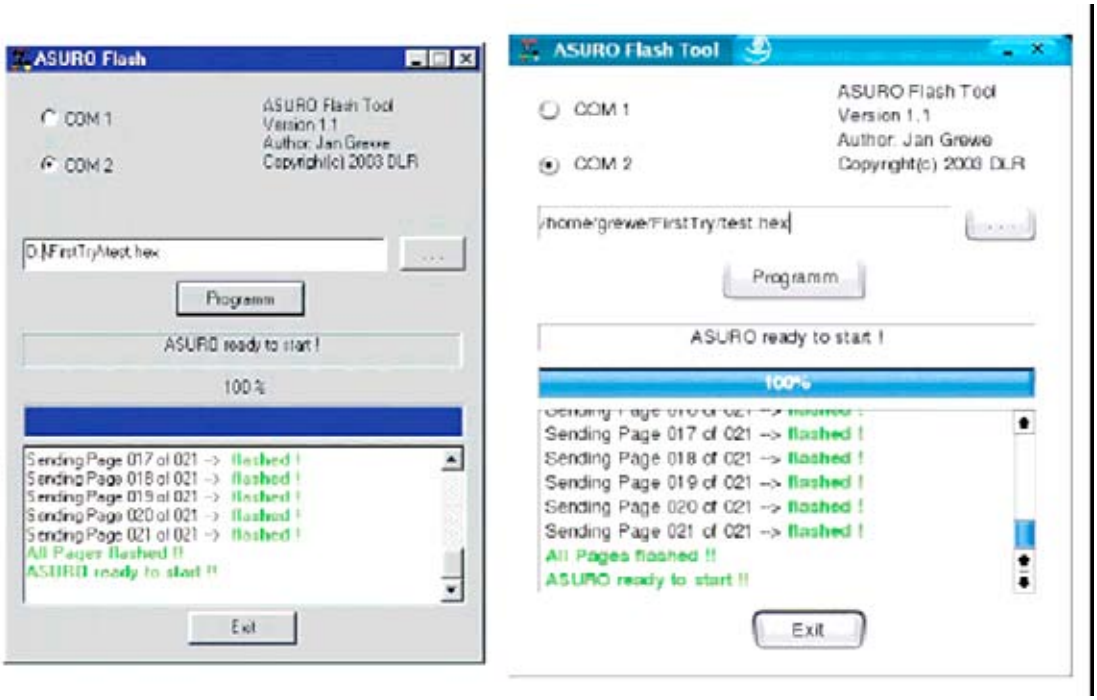


图 8.3.: 用于 Windows 和 Linux 系统的 Flash 软件

启动程序并选择与红外收发器相连的端口号。
从目录 C:\own file\ ASURO_src\FirstTry 选择 Test.hex 文件。

把装配完成并经过测试的 ASURO 与红外收发器放在一起，使两者的距离不超过 50 cm，而且两者的红外收发元件必须有直视距离。先按下 Flash 程序的[Programm]按钮，接着马上在 Flash 程序上的进度条到达最右端前打开 ASURO 的总开关 S1。如果操作失败可以再次按上面所述的顺序尝试连接。

如果连接成功便可通过 Flash 程序的进度条观察到程序 Test.hex 传送到 ASURO 的整个过程。程序会存储在处理器的闪存空间中，系统掉电以后也能长时间保存。

在载入程序之后，ASURO 需要先关断电源后再重新打开才能启动这个程序。重新打开电源后，这个程序会被执行并使 ASURO 的绿色 LED 点亮。

8.3.1. 程序的传送过程

当执行 Flash 程序的[Programm]命令后，PC 开始尝试与 ASURO 通讯。打开 ASURO 后，状态 LED 点亮双色大约一秒时间，表明系统开始启动。ASURO 会检查是否有新程序需要传送，如有则开始传送。程序传送完毕，需关断电源再重新开启才能使程序得到执行。

8.4. 程序传送出错

在程序传送的过程中可能会出现以下错误：

" c " 数据校验错误。ASURO 已经接收到一些数据，但数据在传送过程中受到其它光源的干扰，例如荧光，又或者光通路被短暂打断。

" t " 超时。在 ASURO 和红外收发器之间的可视距离被完全中断。

" v " 数据写入错误。这是由于在 ASURO 的闪存空间中写入数据无效。这种错误较少发生，因为这意味着闪存空间已不能再使用了，而通常这种闪存可以反复编程 10000 次。

误差校正被设定为最多重试十次，最后仍未正确则会取消本次程序传送。

如果在程序传送过程中频繁出现数据校验错误，应该关掉或调暗室内的灯光，尤其是荧光灯。

务必在打开 ASURO 总开关之前按下 Flash 程序的[Programm]按钮，否则不会启动程序传送。

8.5 编写第一个程序

我们通过修改已经 Programers Notepads 2 打开的示例程序来编写第一个程序。请注意：完全安装下面例子的书写，包括字母的大小写。

```
#include"asuro.h"
int main(void){
Init ();
StatusLED(RED);
while (1);
return 0;
}
```

输入完成后需要保存文件，再选择菜单 Tools→make 命令，注意观察消息输出窗口的消息。几秒之后完成编译，查看消息窗口的最后一行 Process Exit Code：0，表明此源文件编译正确完成。

如果编译器显示了一些其他的信息，你必须通过分析源程序来查找错误。通常可以先按编译器指出的第一个出错行来检查错误。其中通过编辑器左下角的一个按钮可以切换是否在程序中显示行号。

经过分析，更改程序后再次选择 make 命令进行编译，注意检查之前提示错误的程序行是否已经编译通过。

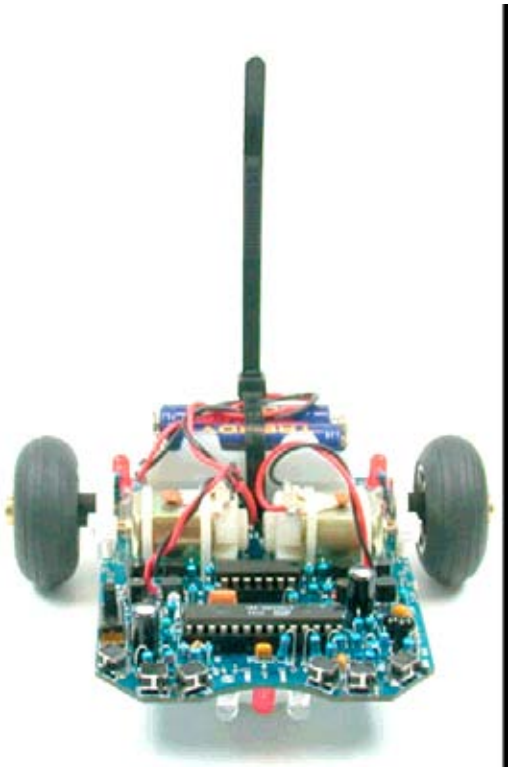
程序无误，编译通过后可把新程序传送到 ASURO 中。连接好红外收发器并启动 Flash 程序，选中新编译好的程序及正确的 COM 端口。把 ASURO 与红外收发器放在一起，执行[Programm]命令并等待新程序传送完毕。

在 Flash 软件提示文件传送完毕后，关断 ASURO 电源并再次打开，等待一秒后检查状态 LED 转为红色。在正式编写程序之前，了解编程相关知识是很必要的，建议首先阅读下一章的内容。

注意!

上述测试程序可以通过修改在 ASURO_src \FirstTry 的 test.c 程序编写；
这个程序包含以下的程序行；

```
#include"asuro.h"
int main(void){
Init ();
while (1);
return 0;
}
```



9. ASURO 使用的 C

这一章主要讲述如何编写 C 语言程序，但述及的内容仅限于对 ASURO 控制的编程所需。所以本章并不是一本 C 语言的完整手册，如需要这样的手册可到图书市场选购。

我们选择使用 C 语言，是因为它使用广泛并已成为标准，而且 C 编译器几乎可以应用在所有种类的处理器上。我们选择 Gnu-C 作为 ASURO 的 C 编译器，这是一个免费的程序软件，该编译器可以为 ASURO 的 ATmega8 处理器生成最优化的代码。

熟悉 C 语言的读者不需要阅读本小节的内容，可以直接跳到第 9.2 节继续学习。我们会介绍对 ASURO 编程控制所需的重要函数。

实际上，C 语言非常容易掌握，使用时只要小心使用分号和括弧就可以了。

9.1. C 语言编程初步

9.1.1. 简介

处理器处理 C 语言程序一般都是从程序的开始到结束一步一步顺序执行的。ASURO 的处理器不支持并行处理程序，所以我们必须按顺序执行的方式设计系统程序。

在下文例子各例里，程序行开始的空格仅为了便于程序的阅读，所以完全可以省略。插入空格键便于分清程序的结构，特别对于长的程序来说，这样的对程序的清晰及可阅读性都是非常有用的。

C 语言中，每一条命令都必须以 “;” 结束，使编译器能正确地对程序进行编译。如果在函数、循环或条件语句加入一组程序，这一组程序要放在“{ ”, “} ”括弧内。

例：

```
#include “asuro.h”
int main(void) {
/*  在括弧中的所有程序行属于同一组的命令*/
}
```


010-62669059 fae@61mcu.com www.61mcu.com

如果在程序行中加入注释，注释语句应该以”/*”开始，并以”*/”结束。单行的注释也可用“//”开始表示[1]。被注释的程序不会被执行，编译器会自动忽略注释行的命令。只要正确插入注释行，就不会引起程序执行的问题。

9.1.2. 变量和数据类型

在程序设计中，变量是用于程序执行过程中储存数据，在程序运行时变量可以被定义、写入、读出和改变数值。在使用变量之前，一定要先作声明。变量声明用于说明变量的类型，同时也可以为变量设置初值。定义的变量类型规定了储存在变量中的数据类型，例如整数、正整数、实数等。

变量用一个名字定义，变量名由字母与数字组成，但其第一个字符必须为字母（下划线__也被看做是字母）。变量名字不能用特殊符号表示。大写字母与小写字母是有区别的， 例如 x 与 X 是两个不同的名字，一般把由大写字母组成的名字用做符号常量（在程序运行期间不能改变其数值, 例如常量 PI = 3.1415)，小写字母用作变量。

下面列出的关键词是保留字，不能把它们用做变量名：

auto default float long sizeof union
break do for register static unsigned
case double goto return struct void
char else if short switch volatile
const enum int signed typedef while
continue extern

下列的数据类型可以用于ASURO 编程：

类型	数据范围	注释
char	-128 ... +127	单字节,可以存放字符集中一个字。
unsigned char	0 ... 255	没有符号的字符,只能储存正整数。
int	-32768 .. +32767	取值在-32768 ~ + 32767 之间 16 位整数。
unsigned int	0 ... 65535	0 ... 65535 之间的正整数。
float		单精度浮点数。

[1] 在 C++语言标准中，“//”是一个注释符号。实际上 ASURO 编译器是个 C++语言编译器，会把“//”看作一个注释符号。“//”作为一个注释符号可能会在其他编译器出现问题。

在变量声明中，在 main()函数外声明的变量是全局变量，在 main()里声明的是局部变量。全局变量在整个程序范围内都是有效的，局部变量仅在 main()函数范围内或其他函数内有效，变量在这些函数内的使用是有效的，在函数外面则是无效的。

如果我们不对变量赋值，变量并不用处。给变量赋值很简单：

```
a=17;                //把 17 给变量 a 赋值
```

或者用计算式表达：

```
a=17+23;            // 为变量 a 赋值 40
Speed=a+3;          // 为变量 Speed 赋值 43
Speed=Speed*2;      // 为变量 Speed 赋值 86
```

在定义变量时应使用达意清晰的变量名。如上例中的变量名“speed”就说明了该变量的大体含义，而使用过于简单的变量名容易造成程序难以被理解。

下面的例子示范了一个简单、完整的程序：

```
#include “asuro.h”
int main(void) {
int i;                    // 变量 i 可以代表的值在-32768 与 32767 之间
char any_token;           // 变量 any_token 可以代表 ASCII 符合或-128 与
                          // 127 之间的数值
i=3;
any_token =17+i;          // any_token 现在的值为 20
i=i/2;                    // 除 2，数值的小数部分会被舍弃，因而 i 的值为 1。
return 0;
}
```

C 语言中有一些有趣的简写，例如：

```
i=i+1;
```

也可写做：

```
i++;
```

等于：

```
i=i-1;
```

也可以写作：

```
i--;
```

9.1.3. 编译器命令

在上面程序中的第一行的#include “asuro.h”是编译器命令。这个#include 命令声明了在本程序中包含了一个外部源编码，并在编译时把它包括进来。这个包含进来的文件含有对 ASURO 编写控制程序时所需的一些函数。

另外一个重要命令（其他更多的编译器命令超出了这里的讨论范围）被称为文本替换。这个命令可以如下形式使用：

#define NAME replacement_text

这在 C 的程序中用作常量定义。

无论 NAME 这个符号在程序的任何地方出现，编译时它将会自动以 replacement_text 代替。C 程序设计者已习惯在#define 中采用大写字母（例如 NAME）。

例：

```
#include “asuro.h”
#define STARTINGVALUE 33
int main(void) {
int i;
i= STARTINGVALUE;  //  i 的数值现在是 33
return 0;
}
```

注意：编译器指令不使用分号作结尾！

9.1.4. 条件表达式

有时我们想要在特定的条件式之下运行指令。这些具有条件判断功能的语句叫做控制结构。这些控制结构中最简单的是“if-else”语句。

在 C 语言中正确的条件语句语法是：

```
if (条件)
    指令组合 1
else
    指令组合 2
```

程序会首先检查括号中的条件表达式的值。如果条件是真（其值不等于零）的，程序运行指令组合 1，否则运行指令组合 2。

如果程序要在数个选择项选择其中一个，则需使用多个“else if”语句。

```
if (条件 1)
        指令组合 1
else if (条件 2)
        指令组合 2
else if (条件 3)
        指令组合 3
else if (条件 4)
        指令组合 4
else
        指令组合 5
```

条件语句可能会使用以下的运算符：

运算符	解释
= =	等于
! =	不等于
<	少于
<	大于
<=	不大于
>=	不少于

例：

```
#include “asuro.h”
int main(void) {
while (1) {
if (PollSwitch()>0) {StatusLED(RED);}
else {StatusLED(GREEN);}
}
}
```

如果其中一个碰撞检测开关被接通，状态 LED 将变为红色，否则为绿色。其他程序将会在稍后解释。

在 C 语言中，“1”表示为真，“0”表示为非真。

有如下的条件语句：

```
if (0) {StatusLED(RED);}
```

则表明语句 StatusLED(RED)绝不会被执行。

9.1.5. 循环语句

循环语句是用作重复执行命令的。
在“while”循环中，每次执行循环完成都要检查一次条件。如果条件是真的，则执行语句，并再次重复检查条件。这一周期性过程一直进行下去，直到条件表达式变为非真，此时程序从该语句的下一个语句接着执行。

```
While (条件)
        指令组合
例:
        #include “asuro.h”
        int main(void) {
        MotorDir(FWD,FWD);              // 两个马达向前运转
        MotorSpeed(120,120);            // 两个马达以半速前进
        StatusLED(GREEN);               // 打开状态 LED 为绿色
        while (PollSwitch()= =0) {      // 只在没有碰撞的时候
        SerWrite(“All OK!\n”,10);       // 发送字符
        }
        MotorSpeed(0,0);                // 有碰撞产生，立刻停止
        StatusLED(RED);                 // 打开状态 LED 为红色
        while (1) {
        SerWrite(“Ouch!\n”,5);          // 发送字符
        }
        }
```

“for (expr1, epr2, expr3)”- 语句相当于：

```
        expr1;
        while (expr2) {
        Command block
        expr3;
        }
```

“for”语句经常被用作顺序计数：.

```
    for (i = 0; i < n; i++)
    . . .
```

例：

```
  #include “asuro.h”
  int main (void) {
        int counter;                        // 定义用于计数的变量
        for (counter =0;counter <10;counter ++) { // 重复十次:
        SerWrite (“Go ahead!\n”,10);        // “Go ahead” 信息
        }
        MotorDir(FWD,FWD);      // 两个马达向前运转
        MotorSpeed(120,120);    // 两个马达以半速前进
       while (1) {              // 无其他动作
       }
    }
```

“while(1)”语句等于“for(;;)”，两者都是一个无限循环语句，因为中止循环的条件永不会符合。

另外一个循环语句是“do”循环：

```
do
command block
while (条件);
```

与“while-”循环语句相反，该循环语句在命令执行完毕才检查条件。这个循环语句使命令最少执行一次。

9.1.6. 函数

函数的定义语句是这样的：

```
函数类型 函数名字（参数类型 1 参数名称 1，
参数类型 2 参数名称 2，…）
```

函数的定义虽然比较复杂，但它们十分有用。对函数的解释相当复杂，将在下面稍加说明。

某些时候，我们需要在我们的程序不同的位置调用相同的程序语句。当然我们可以通过重复输入或者拷贝贴上的方法达到这个目的（这样做会使我们的程序变得混乱），但一般我们采用函数的方式。

有时我们需要传递一些变数给一个函数，举例来说告诉我们的 GoAhead()函数一个给定的速度，持续的时间或行进的距离。 我们将会使用参数说明这些。

有时函数会返回一个数值。
例如 HowManySwitchesHaveBeenActivated() 函数会返回一个数值，用于告知函数执行期间的具体情况。数值由函数末尾的一个返回指令返回。这就是函数为什么以 return；结束或 return 一个数值。

Main()函数是特别的函数，它用于定义程序的主体功能。在 ASURO 程序中的 main()函数将会在系统上电之时执行。当然每个程序都需要一个 main()函数。

在学习了 C 语言的数据类型及函数方面的内容以后，我们编写一个简单的函数，使二个 8 位数相乘并返回结果。

```
int Mult (char a, char b)
/*  函数返回一个整型值，函数名为 Mult，使用两个字符型变量用于输入  */
{                                   //  函数开始
        int c;                      //  定义一个整形变量 c
        c = a * b;                  //  计算 c
        return c;                   //  返回整型变量  c
}                                   //  函数结束
```

下面是一个例子程序，它将会调用我们刚定义的函数 Mult：

```
int main (void)                     //  函数 main 通常返回  int 型值
{                                   // main 函数开始
      char mult1,mult2;             //  定义两个 char 类型变量
      int result;                   //  定义 int 型变量用于存放相乘结果
      mult1 = 2;                    //  赋值
      mult2 = 10;                   //  赋值
      result = Mult(mult1,mult2);   //  调用已定义的 Mult 函数
      return 0;
}                                   //  函数 main 结束
```

9.1.7. 指针和数组

指针和向量部分的知识将会根据控制 ASURO 的需求的而作介绍。

如果我们需要获取线性跟踪器或里程计的数据我们将会需要数组。
数组的声明相当简单：

```
int lineData[2];
int odometrieData[2];
```

我们将会为读取线性跟踪器与里程计的数值而定义二个数组，通过调用 ASURO 的函数(LineData(), OdometrieData())，数组名[0]存放左边感应器的数值而数组名[1]存放右边感应器的数值。

我们在下面的例子验证这一个方法：
如果右边的感光器接收的光比左边的感光器多强，command1 会得到执行，否则就会执行 command2。

```
int lData[2];                    // 为测量结果提供存储器空间
LineData(lData);                 // 读取测量结果
If (lData[1]> lData[0])
      command1;
else
      command2;
```

使用串行口函数(SerWrite()，SerRead())时，我们需要字符串定义。可如下定义字符串：

```
char message[]="This is a text string"
```

为了要送 ASURO 的一个文字字串，

通过调用函数 SerWrite()并设置正确的参数便可由 ASURO 发送字符串。第一个参数表示需要发送的字符串或存放字符串的变量，第二个参数描述需要发送的字符数目，例如：

```
SerWrite(message,20);
```

或

```
SerWrite(“This is a text”，14);
```

将会由红外收发器发送字符串" This is a text "。

010-62669059 fae@61mcu.com www.61mcu.com

ASURO 使用函数 SerRead ()来接收字符，这个函数的第一个参数使指向存储空间的指针，这个存储空间用于存放接收到的数据，第二个参数描述将会收到多少个字符，而第三个参数则是定义一个等待超时值。这个数值可以避免接收的数据未够时出现无限等待。如果在设定的超时值内仍未接收完数据，这个函数会中止执行且之前收到的字符都会被置为‘T’.(=Timeout)。如果定义了超时值为'0'，函数不会中止执行，而会一直等待到所有的字符接收完毕。下面的例子将会说明这个函数的用法。

ASURO 将会通过红外接口接收字符串“Hi, here I am”。通过定义一个字符串：

```
char message []="01234567890123456789"
```

我们定义了一定的储存空间存放将要接收到的字符串。当然这个储存空间一定要足够大存放将要接收的数据。

```
SerRead (message,13,0);
```

接收 13 个字符并且等候直到 13 个字符已经被接收完成。我们现在考虑字符串“Hi, here I am”已经接收完。函数现在将会在已经定义的字符空间中顺序写入收到的 13 个字符信息“Hi, here I am”，最后结果如下面所示：

```
Hi, here I am 3456789
```

9.2. ASURO 系统函数概览

为了方便对 ASURO 编程，系统已经提供了一些可供使用的函数。这些系统函数仅为实现各种实际功能的编程提供方便，具体实现仍需针对实际进行程序编写。所有系统函数均如函数声明的功能进行编写，可以通过阅读以下的例子掌握应用。

为避免系统函数的误用：请记住调用一些如对马达或状态显示实现控制的函数后，被改变后对象的状态会一直保持，直到再次调用函数改变为止。如状态 LED 显示绿色以后会一直保持直到调用函数改变其颜色或保持到断掉电源。

9.2.1. void Init(void)

这个函数会使处理器复位到设定的初始状态，必须在程序的开头处调用。如果不调用这个函数，处理器甚至不会与 PC 终端进行通讯。简单的 ASURO 程序一般有如下的结构：

```
#include “asuro.h”
int main(void) {                    // 在这里定义变量
        Init();                     // 这里添加自己的函数
        while(1);                   // 无限循环
        return 0;                   // 这一句不会被执行
}
```

在 main()函数最后加入无限循环一行语句有特别的作用。通常 main ()函数会在执行完标志程序结尾的 return 0；这一行之后结束。但在处理器的程序空间中可能会有一些之前遗留的程序代码，这些代码仍然可以执行并会得到一些意外的执行结果。故为了避免执行遗留程序而出现非期望的结果，需要在添加一个无限循环语句限制程序的执行空间，这样我们能确定程序结束在一个确定的的状态。

9.2.2. void StatusLED（unsigned char color）
状态 LED (D12) 可通过此函数实现开或关。可用的参数数值为 OFF，GREEN，RED，YELLOW

例：
状态 LED 将会打开为红色：

```
StatusLED(red);
```

下面我们将会给出一个完整的示例程序：

```
#include “asuro.h”
int main(void) {
        Init();
        StatusLED (YELLOW);
        while(1);                   // 无限循环
        return 0;
}
```

9.2.3. void FrontLED (unsigned char status)

前置 LED (D11) 可通过此函数实现开或关。可用的参数数值为 ON 与 OFF

例：
前置 LED 可以通过以下调用点亮：

```
FrontLED(ON);
```

9.2.4. void BackLED (unsigned char left, unsigned char right)

后置 LED (D15 与 D16)可通过此函数实现开或关。前一个参数用于控制左边后置 LED(D15)而后一个参数用于控制右边后置 LED(D16)。可用的参数数值为 ON 与 OFF

例：
以下调用实现控制左边后置 LED(D15)关闭而右边后置 LED(D16)点亮：

```
BackLED(OFF,ON);
```

9.2.5. void Sleep (unsigned char time72kHz)

这个函数会令处理器等待一段时间。等待的时间长度由参数定义(unsigned char)，这个参数的值最大为 255，计算频率为 72kHz。

例：

处理器等待 3 ms ==> 0.003s / (1/72kHz) = 216

函数 sleep (216)将会令处理器等候 3 ms 。

```
sleep (216);
```

9.2.6. void MotorDir(unsigned char left_dir，unsigned char right_dir)

这个函数用于控制两个马达的方向，需要在对马达进行速度控制之前调用它。可用的参数数值为 FWD(前进)，RWD(后退)，BREAK(减速或突然停止)和 FREE(放开控制)。

例：
左边的马达向前方运转而右边的马达停止运转。

```
MotorDir(FWD,BREAK);
```

9.2.7. voidMotorSpeed (unsigned char left_speed, unsigned char right_speed)

这个函数可以控制两个马达的速度。最大的速度值为 255（unsigned char）。马达会在约 60 的速度数值开始运转，这取决于机械方面的情况。参数值本质上是控制供给马达的电能，实际的转速受摩擦力或斜坡等外在因素的影响。

执行这个程序后，ASURO 将会启动运转。有时程序的执行结果会出现意外，故要小心注意确保不要因此而造成他人的损伤。

例：
左边的马达是以最高速度运转，右边的马达则停止不动。马达运转的方向由之前调用的函数 MotorDir( )确定。

MotorSpeed (255,0)；

9.2.8. void SerWrite (unsigned char *data, unsigned char length)

调用这个函数会使 ASURO 的红外接口以 2400B/s，无校验，1 个停止位，无流控方式发送数据。第一个参数指向需要发送的数据，第二个参数表示需发送的字符数。

例：
通过红外接口发送一组字符串：“Hello how are you?”：

SerWrite (“Hello how are you?”，18)；

9.2.9. void SerRead(unsigned char*data, unsigned char length, unsigned int timeout)

这个函数用于从红外接口接收数据。函数的第一个参数使指向存储空间的指针，这个存储空间用于存放接收到的数据，第二个参数描述将会收到多少个字符，而第三个参数则是定义一个等待超时值。这个数值可以避免接收的数据未够时出现无限等待。如果在设定的超时值内仍未接收完数据，这个函数会中止执行且之前收到的字符都会被置为‘T’.(=Timeout)。如果定义了超时值为'0'，函数不会中止执行，而会一直等待到所有的字符接收完毕。

例：
我们希望 ASURO 在接收完字符串“go ahead”之后再开始运转。

```
#include “asuro.h”
int main(void) {
        char data[8];                   // 配置储存区
        Init();
        SerRead (data,8,0);             // 接收数据
        MotorDir(FWD,FWD);
        MotorSpeed(120,120);
        while(1);                       // 无限循环
        return 0;
}
```

9.2.10. void LineData(unsigned int *data)

这个函数用于获取 ASURO 电路板底部两个光感器的感光强度。使用这个函数需要先定义一个指向双整型的存储空间的指针。函数会读取通过 AD 转换器转换的两个光感器的感光度。第一个整型值代表左边光感器(T9)，第二个整型值代表右边光感器(T10)。最强的感光度会以'1023'表示，而完全不感光则用'0'表示。一般情况下这两个极端值都不会出现，较多的是介于两者之间的值。

例：
读取两个光感器的感光度 (T9,T10)

```
        unsigned int data[2];       // 配置储存区
        .
        .
        LineData(data);
```

data[0]存放左边光感器测量的数值 (T9)

data[1]存放右边光感器测量的数值 (T10)

下面给出一个完整的程序作例子：

```
#include “asuro.h”                    //线迹最容易的方法
int main(void) {
        unsigned int data[2];          // 配置储存区
        Init();
        FrontLED(ON);                  // 打开用于线性跟踪用的光源
        MotorDir(FWD,FWD);             // 两个马达均往前运转
        while(1){                      // 无限循环，ASURO 始终沿线运行
            LineData(data);            // 读取光感器的感光度
            if (data[0]>data[1])       // 左边比右边感光度高
            {MotorSpeed(200,150);}     // 提高左边马达速度
        else
            {MotorSpeed(150,200);}     // 提高右边马达速度
    }
    return 0;
}
```

9.2.11. void OdometrieData(unsigned int *data)

这个函数用于扫描反射光感器：两个 LED(D13,D14)被点亮，此函数读取由 AD 转换器转换而得的光感器(T11,T12)的光感度。这个函数需要定义两个整型变量以存放函数的运算结果。。第一个整型值代表左边光感器(T11)的 AD 转换值，第二个整型值代表右边光感器(T12)的 AD 转换值。最强的感光度会以'1023'表示，而完全不感光则用'0'表示。一般情况下这两个极端值都不会出现，较多的是介于两者之间的值。

例:
扫描反射光感器。

```
    unsigned int data[2]; //Allocate memory
    .
    .
    OdometrieData(data);
```

data[0]存放左边光感器测量的数值 (T11)

data[1]存放右边光感器测量的数值 (T12)

下面的原理应该注意：OdometrieData()函数并不提供轮子的转数，只是给出光感器的感光度。因此根据感光度便可知道光与暗的改变，从而通过编程计算得轮子的转速。

9.2.12. unsigned char PollSwitch (void)

这个函数用于扫描触碰开关的状态(K1-K6)并返回一个字节，这个字节就表示了各个开关的状态。开关 1 对应该字节的位 5，开关 1 对应该字节的位 5，**…**开关 6 对应该字节的位 0。

Bit0(1)-> K6
Bit1(2)-> K5
Bit2(4)-> K4
Bit3(8)-> K3
Bit4(16)-> K2
Bit5(32)-> K1

启动开关 1,3 和 5 将会使函数返回 42(32+8+2=42)。

需要强调：往往需要多次调用这个函数才能得到各开关真实的状态。而又因为电容器 C7 的充放电需要一定的时间，如果扫描 AD 转换器过快，会导致读取的数值不真实。

例：

```
unsigned char taste；
.
.
switch = PollSwitch()；
if (switch>0) { MotorSpeed(0,0)；  }
```

关于 ASURO 的硬件及软件的介绍就到此为止，其他的应用与创造将完全由你发挥！

PartⅣ. 附件

A. 元件清单

除一个乒乓球外，组装 ASURO 需要下列的零配件：
1 x 印刷电路板 ASURO
2 x 电动机 Igarashi 2025-02
1 x 二极管 1 N4001
8 x 二极管 1 N4148
4 x 三极管 327/40 或三极管 328/40
4 x 三极管 337/40 或三极管 338/40
1 x 集成电路 CD 4081 BE
1 x 处理器 ATmega 8L-8PC (已有程序)
1 x 红外接收器 SFH 5110-36
1 x 红外发光二极管 SFH415-U
2 x 光感器 SFH300
3 x LED 5 毫米 红色
1 x 双基色发光二极管 3 毫米
2 x 扁平光感器 LPT80A
2 x 扁平 LED IRL80A
1 x 晶体振荡器 8 MHz
2 x 电解电容 容量 220 µ F 至少 10V RM 3,5/10
4 x 陶瓷电容 100 nF RM 5,08
2 x 陶瓷电容 4.7 nF RM 2,54
1 x 100 Ω 1/4 W 5%
2 x 220 Ω 1/4 W 5%
4 x 470 Ω 1/4 W 5%
10 x 1 K Ω 1/4 W 5%
1 x 1 K Ω 1/4 W 1%
3 x 2 K Ω 1/4 W 1%
2 x 4,7 K Ω 1/4 W 5%
1 x 8,2 K Ω 1/4 W 1%
1 x 10 K Ω 1/4 W 1%
1 x 12 kK Ω 1/4 W 1%
1 x 16 K Ω 1/4 W 1%
1 x 20 K Ω 1/4 W 5%
1 x 33 kK Ω 1/4 W 1%
1 x 68 K Ω 1/4 W 1%
1 x 1 M Ω 1/4 W 5%
3 x 芯片座 14 脚
6 x 碰撞检测开关
1 x 电池盒
1 x 电池夹子
1 x 跳线
1 x 跳线帽子
2 x 齿轮 10/50 齿轮比; 3.1 毫米钻孔
2 x 齿轮 12/50 齿轮比; 3.1 毫米钻孔
2 x 驱动小齿轮 10 齿; 1.9 毫米钻孔
2 x 3 毫米轴承

A. 附件清单

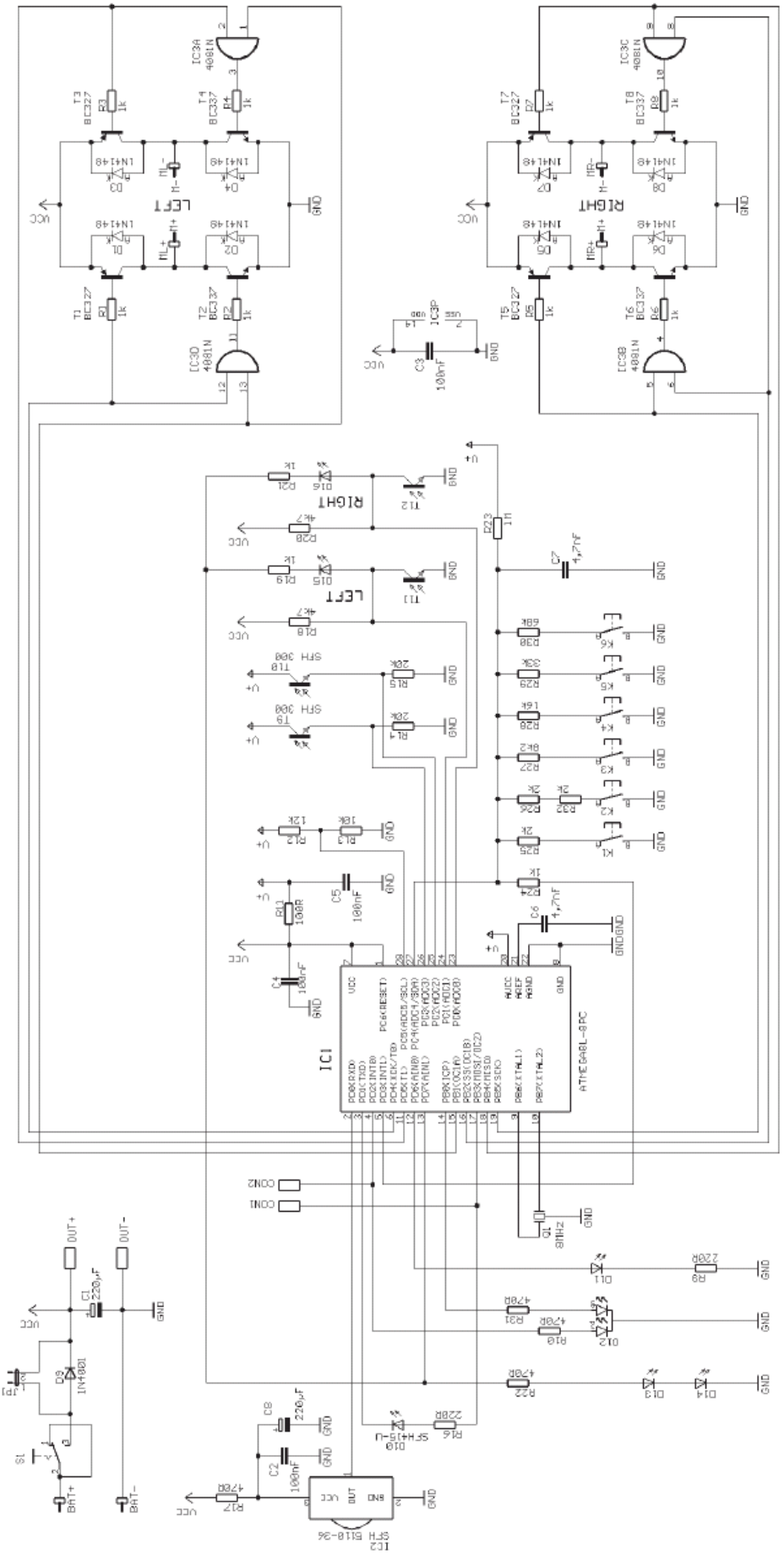
4x 绑带
1x 可松脱绑带
2x 橡胶轮胎 38mm
2x 轴承 长 42mm 直径 3mm
2x 轴承 长 24.5mm 直径 3mm
红色电线 约 15 cm 0.14mm
黑色电线 约 15 cm 0.14mm
2x 编码纸片(见 2.4)

RS-232 红外收发器由以下零配件构成：

1x RS232 红外收发器电路板
3x 二极管 1N4148
1x 二极管 ZPD5.1
1x 三极管 BC547 A,B 或 C 或 BC548 A,B 或 C
1x 芯片 NE555N
1x 红外接收器 SFH 5110-36
1x 红外 LED SFH415-U
1x 电解电容 100µF 至少 16V RM 2,5/6
2x 陶瓷电容 100nF RM 5,08
1x 陶瓷电容 680pF RM 2,54
1x 220Ω 1/4 W 5%
1x 470Ω 1/4 W 5%
1x 4,7KΩ 1/4 W 5%
1x 10KΩ 1/4 W 5%
2x 20KΩ 1/4 W 5% or better
1x 可调电阻 10k 直立式 RM 2,5/5
1x 芯片插座
1x 9 针插头 SUB-D

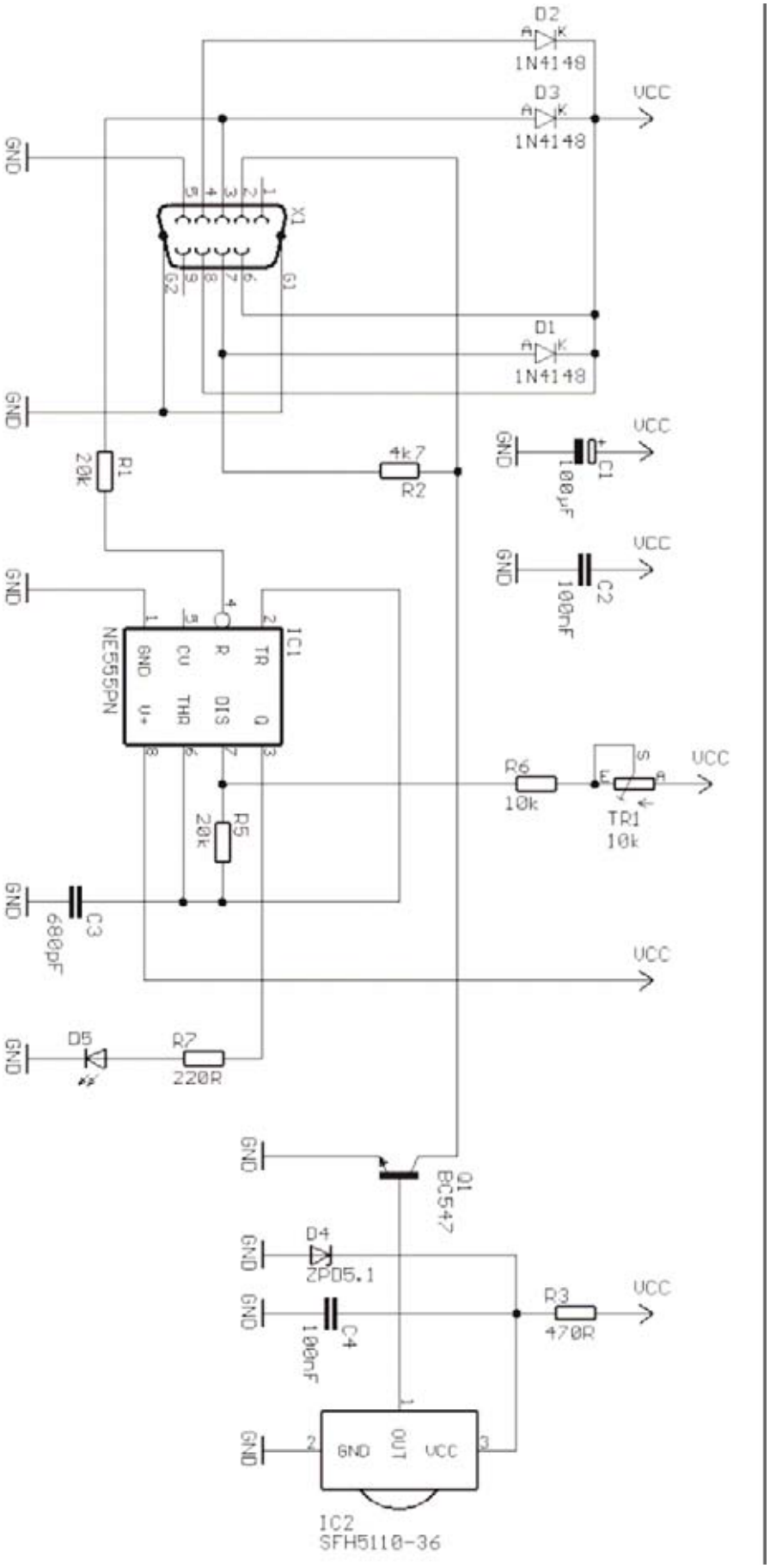
USB红外收发器是一个可选件：

B. ASURO 原理图

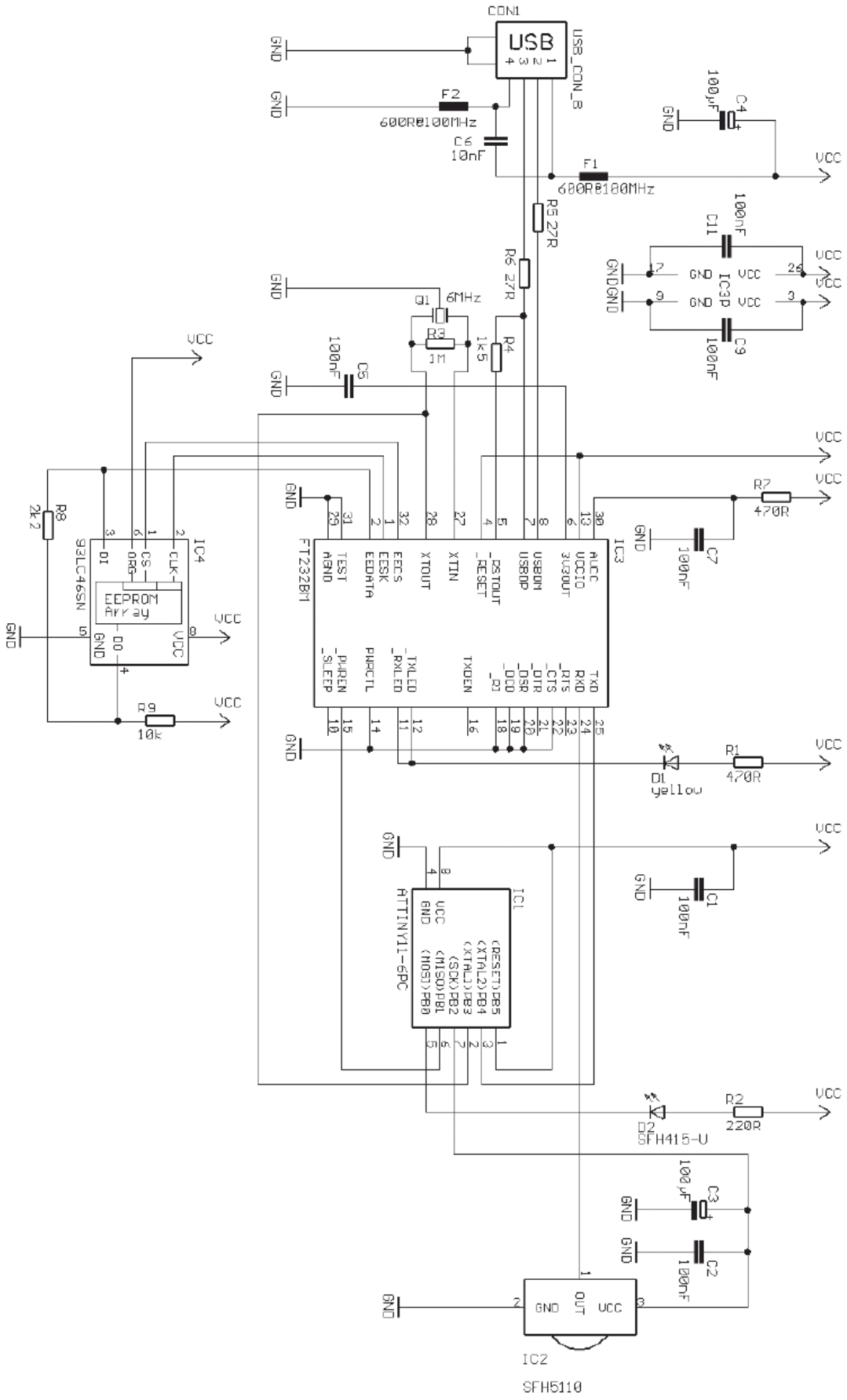


010-62669059 fae@61mcu.com www.61mcu.com

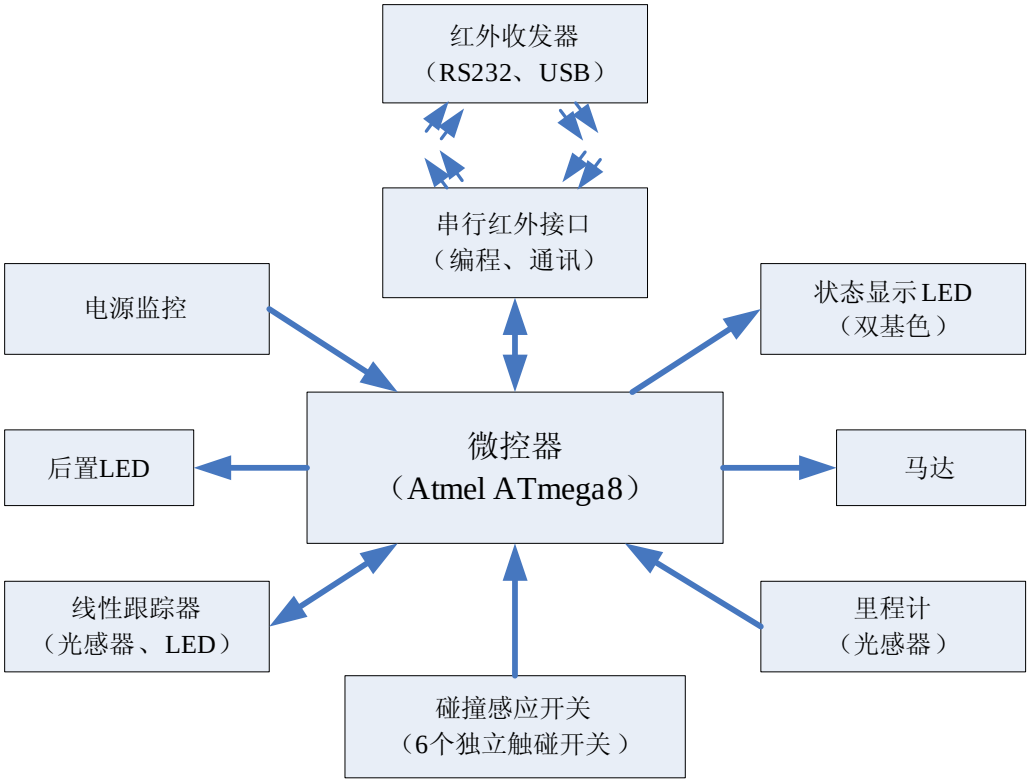
C. RS-232 红外收发器原理图



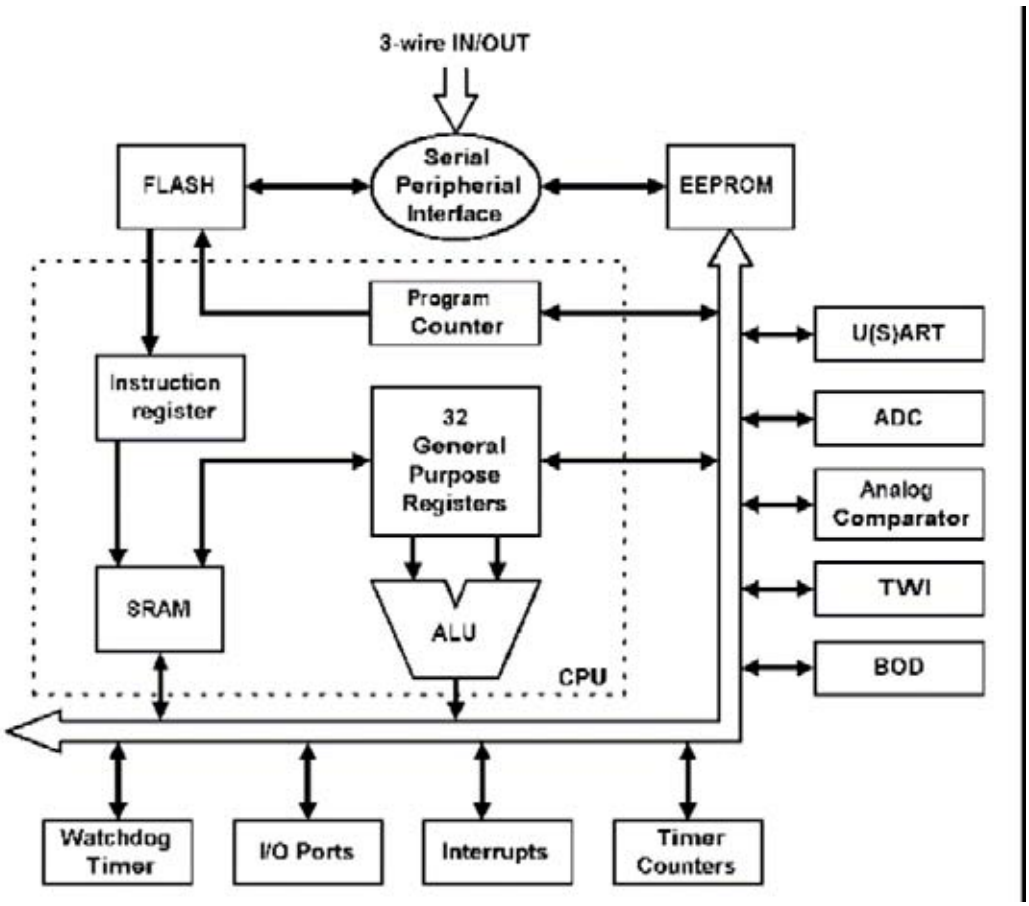
D. USB 红外收发器原理图



E. ASURO 结构框图



F. PIC 处理器结构框图



G. ASURO 零配件实物图