工科：将书边上的所有疑点都用代码实现才算是真的懂！工科知识的转化唯一的方法是代码验证理论。方法：看图说话，图表是最易懂的文字。

程序调试：依照别人的51程序，或者是给的相应的程序，明白每一个程序的功能，和要修改的地方，做好记录。程序出错时，先用单步调试，看自己定义的每一个变量的值，明白程序的数据流动变化过程，并结合测试工具（示波器，万用表，频谱分析仪等）对程序进行分析检查,

程序心情的调整：

a.调程序，明白最小单元，结合分析最小单元的实现

b.写好程序的流程图和调试时按流程调试

c.看状态寄存器的值，明白程序的状态

d.如何选择合适的时机与别人交流互动：团队精神重要，在程序分析不出问题时——1.说明文档 2.帮助手册 3.网页和论坛 4.中国知网 5.技术支持6.书籍7.同事 8.FQA中

程序良好的习惯

1. 变量命名，及数量类型的选取，多局部变量少全局变量；
2. 程序计算是的注解及精度（四舍五入，先乘后除）；
3. 程序注解与代码相对应，结构清晰，各个不同功能语句分割；
4. 软件测试和硬件测试，符合规定指标，并考虑多种应用情况；
5. 严格按照协议要求，防患未然；
6. 每个功能函数都要测试是否能实现，整体流程是否正确；
7. 程序流程设计及功能语句多请教别人，虚心请教多多交流；
8. 程序调试时，按照思想流程设计无误后，若无法实现，清楚明白问题所在，团队解决；
9. 技术文档参数多问如何实现；
10. 写程序做出成果，多想想衡量好坏的标准，凡事都有一个好坏标准。
11. 代码可扩展性
12. 变量名尽可能短，过长可能会导致编译器出错，类型转换和有符号变无符号，如keil4

学习方法心得：

1.通读理论：a.对知识点进行全面扫盲，知道有这么一回事，对以后定位问题(背目录)大有帮助，工科的最可悲在于无知；b.记录哪些是常用的技术，哪些是过时的技术，在代码验证时掌握最常用的就行了，其所替代的知识就不用深究，反正也不用；c.思考问题的大致方式 d.查资料、求助的途径

2.代码验证：i.先运行代码，对通读的疑问点进行解惑（代码验证书本的内容）；ii.明白基本功能代码(可移植的单元代码)，类似I/O，中断，串口等方便重组；iii.熟悉项目编写思维方式、Debug方式、原则。iV.明白该功能到底是个怎么个效果 V.记录哪些是可以继承前人的成果。

3.实践项目：1）站在巨人的肩膀上（向业内人士进行咨询，向书本寻找相关的demo）2）增量建模技巧，先破0（平台搭建，功能开发），再精细到业务需求。

ios第三方库使用心得：

快速了解第三方库：1.运行库里的demo并结合github的README.md或是官网资料介绍理解；2.通过中断和中断的输出信息，大致定位程序流程，弄明白库里的对象、对象方法（初始化用）、数据源协议（数据显示用）、协议（触发的动作，对外的输出）3.有问题在issue里面先找，然后在百度，解决不了就发布issue 4.实践之前多看看别人是怎么做的，有没有利用开源第三方库。

C语言TCP/IP协议移植心得：协议的数据封装和解封装、断点跟踪程序进入的函数和全局变量（C程序的接口）。

USB协议代码实现心得：协议的分层是因为上一层数据解析的方式至少有2种以上，（物理层和数据链路层用来得到package数据，必须要有）根据协议的数据流图（对数据位的不同解析）来实现。

Ucos-II移植ftp心得

1.ucos-II提供的是一个库，最小内核包括时间管理、多线程管理、内存管理、线程通信构成内核，内核的扩展包括一种/多种网络管理、一种/多种文件系统管理再集中抽象 对外提供统一接口，硬件驱动对上统一抽象接口为文件，构成复杂的linux内核