目录

一 产品与服务

本节将对核心产品——基于 MicroPython 的开发板进行详细的原理说明.

§1.1 研究开发

1.1.1 产品描述

(1) Python 的优越性

Python 是典型的解释型编程语言，省去了编译的过程可以减少编辑、测试和排除错误的时间，大幅提高工作效率；在当下时代已经是数据分析和 AI 研究的第一语言，很早就成为 Web 开发、游戏脚本、计算机视觉、物联网管理和机器人开发的主流语言之一。随着用户可以预期的增长，python 有机会在多个领域里登顶，继续扩大其潜在市场，成为未来机器学习的主流语言。

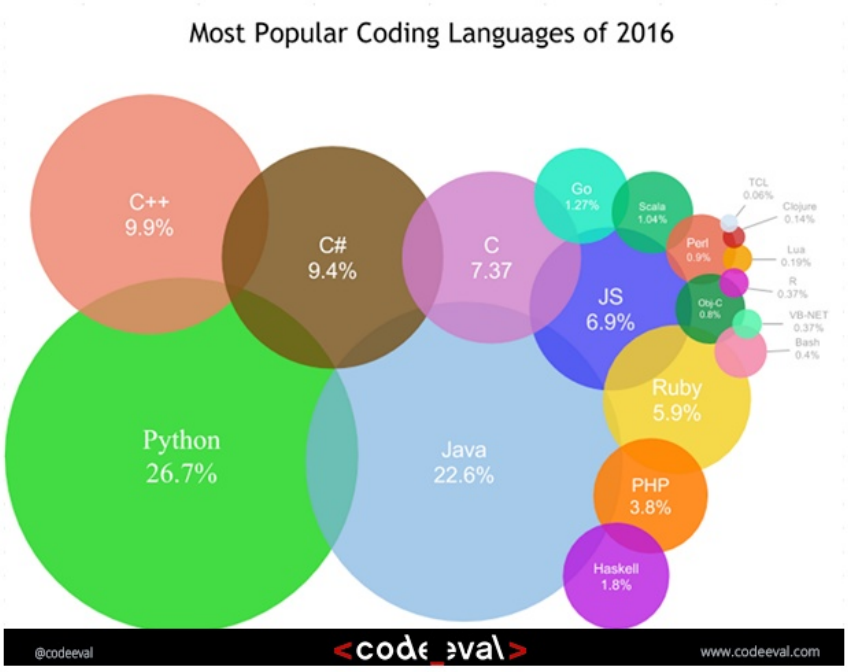


图 1.1 2016 主流语言市场统计

Python 也是众多主流语言中唯一一个战略定位明确，而且始终坚持原有战略定位不动摇的语言。其语言设计简洁优雅，对程序员友好，开

发效率高；且拥有坚实的数值算法、图标和数据处理基础设施，开发生态成熟，建立了非常良好的生态环境。

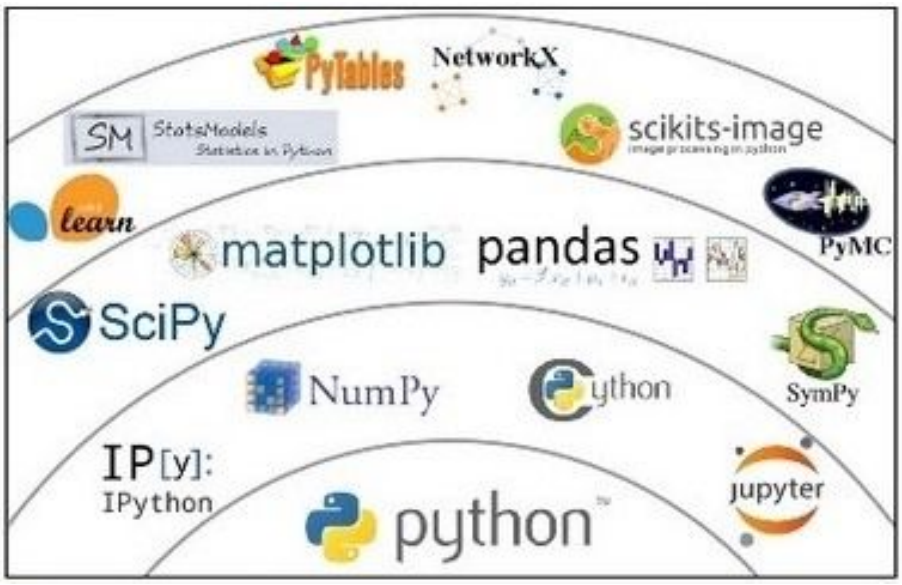


图 1.2 Python 算法库

(2) 机器人编程语言

首先，至少会一门编程语言。机器人的开发语言一般为 C、C++、C++ Builder、VB、VC 等语言。其次至少要对一种控制器使用熟练。STM32、PLC、PMAC 等是市面上常用的语言。但是机器人语言品种繁多，而且新的语言层出不穷。这是因为机器人的功能不断拓展，需要新的语言来配合其工作。另一方面，机器人语言多是针对某种类型的具体机器人而开发的，所以机器人语言的通用性很差，几乎一种新的机器人问世，就有一种新的机器人语言与之配套。而且各家公司的机器人编程语言都不相同，自成一套系统，彼此之间交流困难重重。Python 作为

(3) Micropython 简介

MicroPython 是包括的 Python 标准库，并可在微控制器和约束环境下实现优化运行的精干高效的 Python 的编程语言. MicroPython 兼容性优良，可以让用户将代码从电脑易于移植进单片机或嵌入系统.

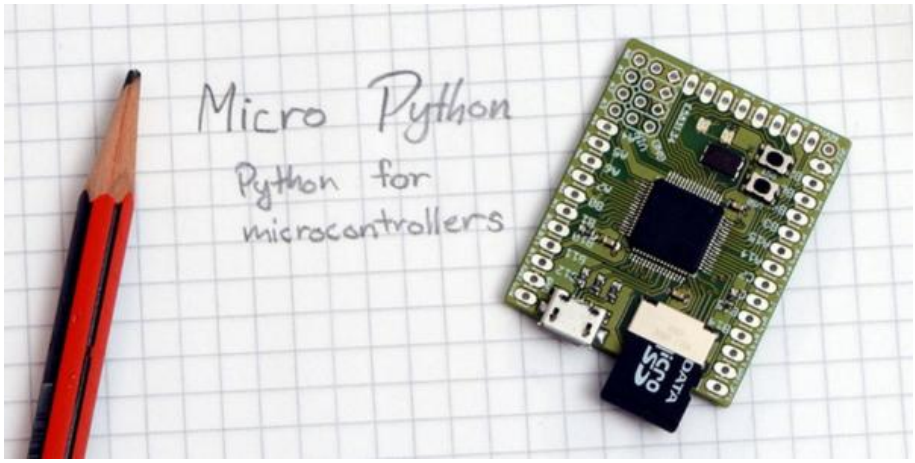
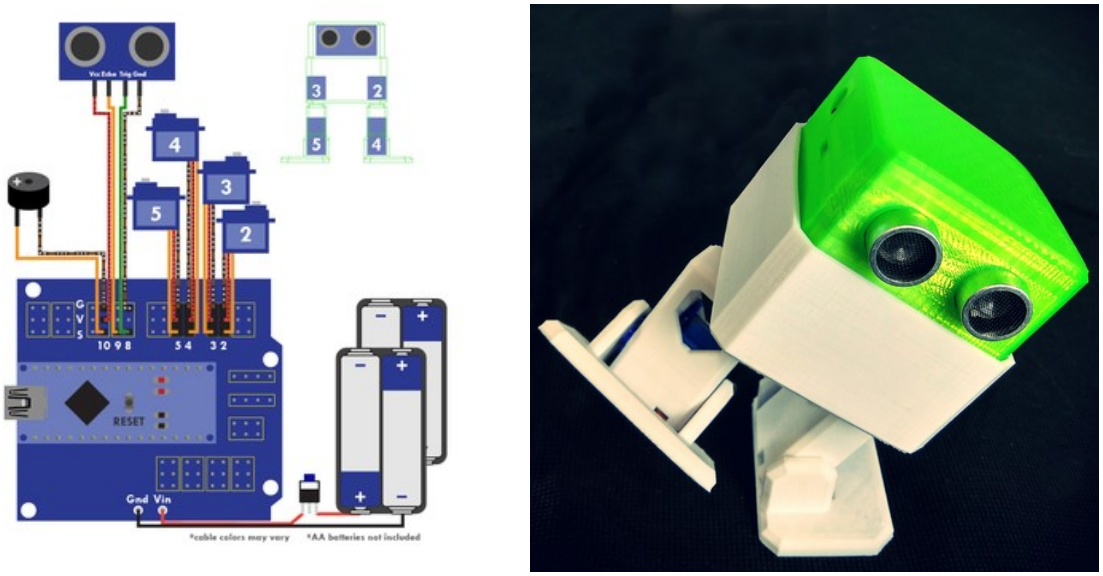


图 1.3 MicroPython

用户通过 MicroPython 可以轻松实现对微控制器的控制，不需要通过复杂的编程，可直接通过 MicroPython 脚本语言进行操作. 不同于传统的微控制器控制需要重新修改、编译、上传程序，MicroPython 脚本语言可实现实时的操作，去掉繁琐的流程，操作更加简单化. 用户完全可以通过 MicroPython 语言实现硬件底层的访问和控制，比如说控制 LED 灯泡、LCD 显示器、读取电压、控制电机、访问 SD 卡等. 由于 MicroPython 的简约特性，用户完全可以在学习脚本语言的同时，进行个人兴趣 DIY 制作，减少了学习时间和成本，增强了编程的体验感，在最短时间内看到学习成果的体现.

以 Otto 机器人为例，借助 MicroPython 平台，用户可以自主设计小型玩具机器人，简单且易于操作，互动性和趣味性较强。



(a) (b)

图 1.4 基于 MicroPython 的 Otto 机器人

1.1.2 产品原理

(1) MicroPython 编译原理

单片机的编程环境默认为 C 语言。目前在单片机业内有两种编译机制，主流是以 Arduino 为代表的 AVR16 单片机编译机制：

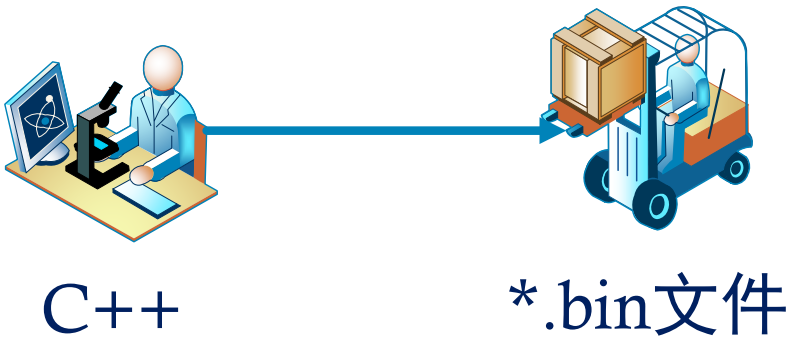


图 1.5 基于 C++ 的单片机编译机制

而借助 python 语言编译过程中虚拟机 (VM-Visual Machine) 的特性，可以使编译速度大大加快。python 更偏向解释型语言，虚拟机即为

Python 的解释器，模拟可执行程序 X86 机器上的运行，像 CPU 一样将字节码一条一条的执行。

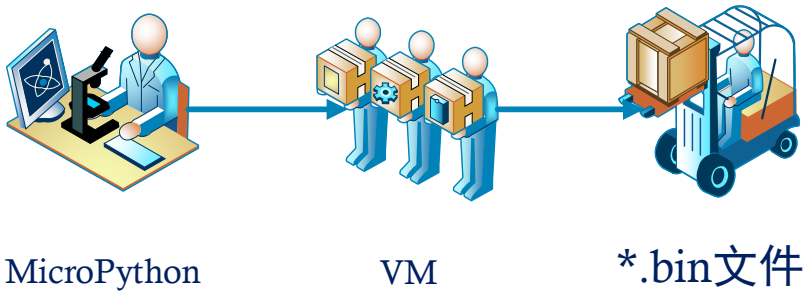


图 1.6 基于 MicroPython 的单片机编译机制

C 语言对于普通用户来说，学习成本是较大的。而 Python 语言需要在 PC 或者其他嵌入式 linux 平台才可以运行，在单片机，比如说 STM32，是不可运行的。所以 MicroPython 的出现，综合了单片机使用 C 语言的局限和 Python 的简单性，帮助用户解决了这个问题。MicroPython 为面向对象的语言，将用户界面输入的 python 语句进行封装，利用 VM 虚拟机实时转换为单片机可识别的 *.bin 文件，使得 python 语言可在例如 STM32 之类的单片机上运行。

(2) PYboard 运行原理

PYboard 是 MicroPython 官方单片机支持电路板，编程环境为 MicroPython。PYboard 基于 STM32F405RG 微控制器，通过板载的 microUSB 接口供电以及进行数据传输，板载外设包括了一个 MicroSD 卡座接口、4 个 LED 灯、两个机械按键、一个加速传感器、时钟模块，其它外设信号都通过板子上的金属通孔引出，具体的信号定义可以参考下图。

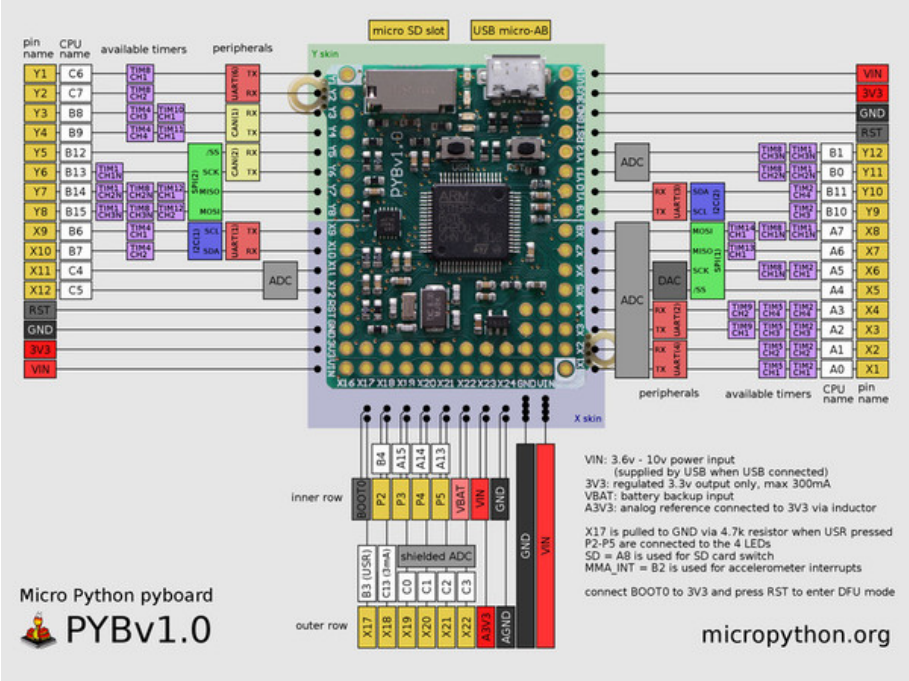


图 1.7 PYboard 信号输出

以 STM32 单片机为例，通过 MicroUSB 线连接上电脑后，设备管理器出现了一个需要安装驱动的虚拟串口，从电脑中看到 pyboard 的 U 盘。U 盘中的有几个重要的文件 boot.py、main.py、pybcdc.inf。若板子正常启动，上电先会运行 boot.py，然后再配置 USB，最后运行 main.py。其中 pybcdc 是需要安装的驱动。

图 1.8 U 盘中的文件

此时，将 **MicroSD 卡**插入上电，pyboard 会默认**从 SD 卡启动**来代替原本的微控制器中的 ROM 中启动。

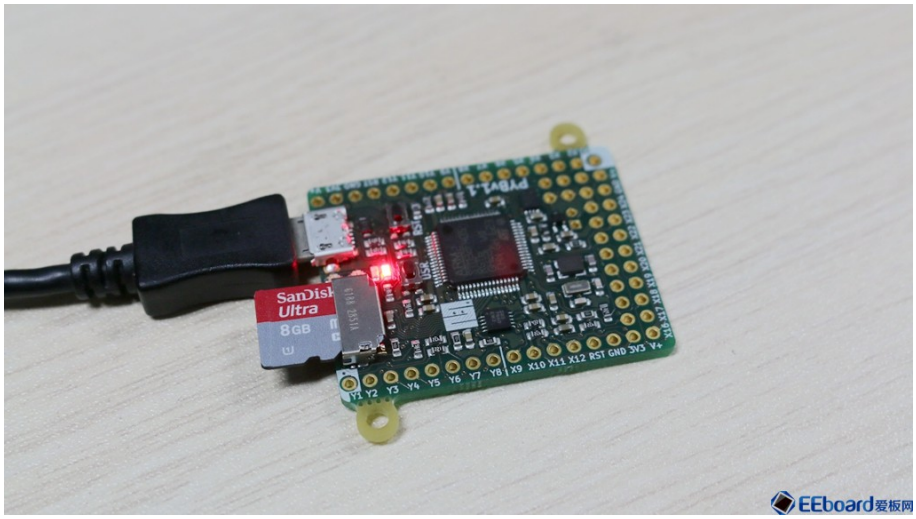


图 1.9 最终连接状态

在 SD 卡中，通过使用 python 语言编辑 main.py 文件，添加相应脚本代码，就可以实现上电即运行程序的操作。

1.1.3 未来产品与规划

(1) 初期产品

初期团队规模较小、技术水平不够高、资金资源有限，在满足用户核心需求的基础上，将进行主要的功能实现. 同时，为践行“创新”与“颠覆”的产品理念，初期产品的主要任务在于带来全新的用户使用体验，从而建立品牌效应，为中后期产品的推出打好基础. 在单片机市场逐渐被 Arduino 平台占领，且同期无其他替代产品出现时，MicroPython 的出现是极具颠覆性的.

基于以上考虑，初期产品主要偏向于采用 MicroPython 平台的开发，建立自身的 python 库，为用户建立强大的代码支持，提升自身信誉度.

(2) 中期产品

在得到稳定市场份额、确定目标人群、积累品牌效应的基础上，中期产品将更多地体现智能化设计，在中端产品中争取更大的话语权. 中期产品将对初期产品进行优化升级，在硬件方面针对用户反馈与售后信息进行修正改动，在软件方面进行功能拓展与性能提升. 在保证系列产品较小的价位浮动的前提下，尽量提升用户体验，扩大目标人群数量.

基于以上考虑，中期产品可以考虑加入单片机生态链以及创客教育，与国内几大创客品牌教育达成合作，一方面获取更多的用户需求的资料后，可以对不同需求，例如信息安全方向的 wifi 欺骗器，及时扩充 python 库；一方面利用自身优势扩大市场，与 arduino 平台达成良性竞争。

(3) 终期产品

在经过初期和中期产品的多层迭代后，MicroPython 平台已日趋完善。此时可以考虑利用周边效应，建立自身创客教育品牌——极客教育. 将前期所有研发成果转变成线上线下课堂的形式，加入新媒体模式，与 MicroPython 平台相辅相成，融入国内 STEM 生态圈。

§1.2 资金需求

§1.3 服务支持

(1) PYboard 安装

随产品会配发相应说明书，网站上放置链接以供用户参考。

(2) Python 脚本语言编写

MicroPython 平台提供基本案例和教程。用户还可以在 GitHub 平台上与其他用户交流，查看其他用户编写的代码。https://github.com/micropython

(3) 官方售后中心

设立专门的组织结构，选派业务素质高、责任心强的人员负责售后管理，实施定人、定责、定任务、定范围的岗位责任制. 讨论建立定期巡回制度、质量跟踪制度、定期座谈会制度、来访接待制度、零配件供应制度、“包退、包换、包修”制度、销售档案制度等，明确服务职责，规范服务行为，完善服务流程.