



个 人 设 计

基于树莓派与 OpenCV 的 图像回传智能小车

姓 名：_____张天羽_____

地 点：_____云南昆明_____

二〇一二年 八 月

摘 要

树莓派 (Raspberry Pi) 是一款基于 ARM 的微型电脑主板, 以 SD 卡为内存硬盘, 卡片主板周围有两个 USB 接口和一个网口, 可连接键盘、鼠标和网线, 同时拥有视频模拟信号的电视输出接口和 HDMI 高清视频输出接口, 并且引出了 GPIO 与 SPI 等低速总线。以上部件全部整合在一张仅比信用卡稍大的主板上。

OpenCV 的全称是: Open Source Computer Vision Library。OpenCV 是一个开源发行的跨平台计算机视觉库, 可以运行在 Linux、Windows 和 Mac OS 操作系统上。它轻量级而且高效——由一系列 C 函数和少量 C++ 类构成, 同时提供了 Python、Ruby、MATLAB 等语言的接口, 实现了图像处理 and 计算机视觉方面的很多通用算法。应用于人机互动、物体识别、图象分割、人脸识别、动作识别、运动跟踪、机器人、运动分析、机器视觉、结构分析等各种领域。

本设计选用了树莓派板卡为核心控制器, 常见的 USB 摄像头作为图像传感器, 以及三轮小车底盘。简单学习了 OpenCV 库的使用以及树莓派上底层端口的编程, 对图像回传的智能小车进行了简单的系统设计。最后通过一系列的工程制作与计算机编程, 实现了此系统的全部功能。

最终完成后, 对装置进行了测试, 成功获取了大概 2fps 的图像回传。以及对回传图像的人脸识别处理。

关键词: 树莓派; USB 摄像头; 智能小车; OpenCV; 人脸识别

目 录

引 言	3
1 硬件模块介绍.....	4
1.1 树莓派板卡（Raspberry Pi）	4
1.2 USB 摄像头	4
1.3 智能小车底盘.....	4
1.4 L298N 电机驱动板.....	5
1.5 双 USB 输出移动电源	5
2 软件支撑描述.....	6
2.1 OpenCV ^[1]	6
2.2 树莓派的 GPIO 库 ^[2]	6
2.3 交叉编译环境的设置.....	7
3 系统设计与软件实现.....	8
3.1 图像的采集与回传显示.....	8
3.2 控制指令的发送与执行	8
3.3 全部代码（不在此文档中）	9
4 测试及结果.....	10
结 论	11

引 言

智能小车是当前嵌入式计算机技术的一个重要领域，作为现代的新发明，是以后的发展方向，智能小车可以按照预先设定的模式在一个环境里自动的运作，不需要人为的管理，可应用于科学勘探等等的用途。

由于产品级智能小车系统结构复杂，制作周期长，并且难度较大。在本设计中没有考虑单个智能小车的成本以及智能小车的使用体验，仅仅是以制作时间短，结构简单，并且拥有视频回传与远程控制的功能为目标，对整个系统进行结构设计与工程制作。

为学习 OpenCV 的简单使用，设计中图像回传使用了 OpenCV 计算机视觉库。并且在接收端做了人脸识别处理。并且使用了树莓派的 GPIO 库进行端口输出控制小车行走。

最后在测试环节中，小车的图像回传与行走控制均正常完成。当回传的图像中有清晰的人脸时，能够自动标注出人脸与眼睛在图像中的位置。

1 硬件模块介绍

1.1 树莓派板卡 (Raspberry Pi)



图为 树莓派板卡外形图

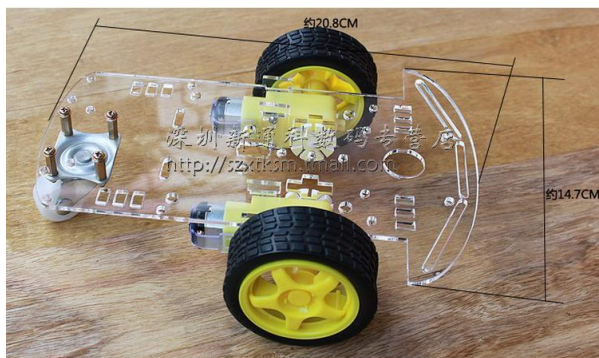
在核心控制器上，我选用了树莓派板卡。B 树莓派配备一枚 700MHz 博通出产的 ARM 架构 BCM2835 处理器，256MB 内存（B 型已升级到 512MB 内存），使用 SD 卡当作储存媒体，且拥有一个 Ethernet，两个 USB 接口，以及 HDMI(支持声音输出)和 RCA 端子输出支援，并且支持 GPIO 与 SPI 等多种总线。具有体积小，功耗低，功能强的特点。

1.2 USB 摄像头

本设计使用了普通的免驱 USB 摄像头做图像采集。USB 摄像头使用方便，并且能够免去编写驱动的时间与工作。

1.3 智能小车底盘

在小车底盘上，采用了淘宝购买的两轮小车底盘。这种小车底盘机械结构简单，非常方便安装，使用了 2 个减速直流电机转弯灵活，方向性好。并且底盘大而稳非常容易扩展。



图为 智能小车底盘

1.4 L298N 电机驱动板

为了驱动小车底盘，有两个方案，一是采用继电器，这种方案成本低但是控制效果差，二是采用电机驱动板，这种方案成本略高但是控制效果较好。

经考虑决定使用 L298N 电机驱动板。L298N 是 ST 公司生产的一种高电压、大电流电机驱动芯片。该芯片采用 15 脚封装。主要特点是：工作电压高，最高工作电压可达 46V；输出电流大，瞬间峰值电流可达 3A，持续工作电流为 2A；额定功率 25W。



图为 L298N 电机驱动板

1.5 双 USB 输出移动电源

对于智能小车的供电系统，我使用了一个双输出的移动电源。使用移动电源的优点在于电量充足达到了 1W mah，并且具有很强的输出能力。

将额定输出 2.1A 电流的输出口引出给 L298N 电机驱动板用于驱动小车底盘上安装的直流电机。而另一个 1A 电流输出口则引出给树莓派为树莓派供电。

2 软件支撑描述

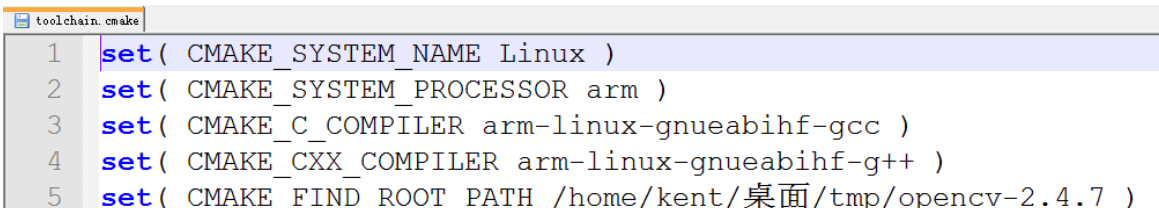
2.1 OpenCV^[1]

在本设计中，使用了 OpenCV 代码库的图像采集，图像压缩解压，以及人脸识别的功能。

在使用之前，由于目标平台为 ARM，需要对应 ARM 平台进行 OpenCV 的交叉编译。并且也需要使用本地编译器编译对应 x86 平台的 OpenCV 库。

OpenCV 的项目构建工具为 CMake，不同于传统的 Makefile，CMake 对于交叉编译工具链有非常好的支持。

交叉编译时，须注意创建指定了交叉编译工具链的 CMake 文件，我的 toolchain.cmake 文件内容如下：



```
1 set(CMAKE_SYSTEM_NAME Linux )
2 set(CMAKE_SYSTEM_PROCESSOR arm )
3 set(CMAKE_C_COMPILER arm-linux-gnueabihf-gcc )
4 set(CMAKE_CXX_COMPILER arm-linux-gnueabihf-g++ )
5 set(CMAKE_FIND_ROOT_PATH /home/kent/桌面/tmp/opencv-2.4.7 )
```

这个文件描述了当前主机的系统，目标平台的硬件架构，C 编译器，C++编译器，以及 opencv 代码根目录。

在交叉编译时，还需要修改编译目标列表，libjpeg 等图像处理库需要全部编译，libgnome 等与图形用户界面相关的模块须禁止编译，否则在执行 make 的时候会报错。

编译完成之后，执行 make install，make 工具就会将所有目标复制到 CMake 指定的安装目录中。

2.2 树莓派的 GPIO 库^[2]

与 OpenCV 的使用准备相同，树莓派的 GPIO 库也需要在使用前对代码库进行交叉编译。

在进行编译之前，需要对编译进行配置，其配置工具使用了一个 GNU 的配置脚本，常用于各种小工程的编译。使用这种编译配置脚本时，推荐使用 bash 作为 shell 环境。

配置时需要指定目标平台与工具链前缀，命令如下：

```
./configure.sh -host=arm -target=arm-linux-gnueabihf
```

配置指定了编译工具链前缀为 arm-linux-gnueabihf

配置完成之后，就可以对代码库进行 make 构建。

2.3 交叉编译环境的设置

在交叉编译之前，需要对编译环境进行设置。由于我使用的编译主机为Ubuntu64 位 Linux 发行版，因而设置环境时只需要设置环境变量。

我编写了一个简单脚本，用于实现环境变量的设置。

脚本 arm-linux.sh 内容如下：

```
arm-linux.sh
1  if [ $CXX != "g++" ];then
2      export CXX=g++
3      export CC=gcc
4      echo "当前为标准C编译器"
5  else
6      export PATH=$PATH:/home/kent/arm-bcm2708/
7      gcc-linaro-arm-linux-gnueabihf-raspbian/bin
8      export TOOLCHAIN=/home/kent/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian
9      export TB_CC_PREFIX=arm-linux-
10     export PKG_CONFIG_PREFIX=$TOOLCHAIN/arm-linux
11     alias arm-linux-gcc=arm-linux-gnueabihf-gcc
12     alias arm-linux-g++=arm-linux-gnueabihf-g++
13     export CXX=arm-linux-gnueabihf-g++
14     export CC=arm-linux-gnueabihf-gcc
15     echo "当前为ARM-C编译器"
16 fi
```

脚本中/home/kent/arm-bcm2708/是我的交叉编译工具链目录，如果使用了其他目录，则需要修改脚本中相应部分。

脚本实现了使用普通编译器与使用交叉编译器之间的转换，使用方式如下：

source ./arm-linux.sh

3 系统设计与软件实现

树莓派通常以 Linux 的 ARM 平台发行版为操作系统。由于 Linux 拥有优异的多任务性能，在本设计中，树莓派上会运行两个守护进程作为服务器。

与此相对的，在笔记本运行的 Ubuntu 操作系统上，也会使用两个进程作为客户端与树莓派相连。

而网络连接都使用无线局域网。

四个进程分为两组，每组都是一个 C/S 架构。Ubuntu 上运行客户端，树莓派上运行服务器。

一组 C/S 架构负责图像采集与回传后的显示。另外一组 C/S 架构则负责控制指令的发送与执行。

两套 C/S 架构都循环进行客户端发送请求，然后服务器发送应答。

3.1 图像的采集与回传显示

本设计中，图像的采集与回传显示都使用了 OpenCV 代码库。

在 Ubuntu 笔记本上对应 x86 平台，使用普通编译器编译了 OpenCV。

在树莓派上则运行的是 Arch 系统，使用交叉编译器编译了 OpenCV。

代码库编译完成无误之后。编写对应树莓派上的服务器与 Ubuntu 笔记本上的客户端。

客户端与服务器之间采用 1:1 的 TCP 连接，不考虑多客户端接入的情况。当传输开始时，从客户端输入服务器的 ip 地址，客户端根据 ip 地址主动连接服务器完成连接。

连接完成之后，客户端开始向服务器循环请求数据，服务器每次应答都采集一帧图片，压缩打包后传回客户端，客户端使用 OpenCV 将图片解压，并且根据 OpenCV 提供的 xml 文件，对图片进行人脸识别处理，识别完成之后使用 OpenCV 内置的图像绘制函数对图像中的人脸进行标注，最后将标注后的图像显示在客户端上。人脸识别不会对人像进行辨别。仅仅是标注图像中的全部可识别人脸的位置。

由于无线局域网的 802.11b/g/n 标准中，无线网的数据包 MTU 大概为 1488 个字节，并且每一帧图片的数据会达到 1500 字节及以上，因而客户端读取一次可能无法完整读取一帧数据，因而服务器应该先发送数据长度，客户端根据收到的数据长度进行多次读取。

3.2 控制指令的发送与执行

本设计中，控制指令为命令字与参数。比如前进 1 秒为：

go 1000

而左转 2 秒则为：

left 2000

命令字为 `go,left,right`。而参数为操作持续的时间，单位为毫秒。

控制指令客户端启动时，会主动建立与服务器之间的 **TCP** 连接。建立完成之后会等待用户输入控制指令。

用户输入的控制指令会由客户端以明文直接发送到树莓派上的服务器。服务器对指令进行解析，如果指令解析失败或者参数无效，则会关闭连接，并且打印错误信息。

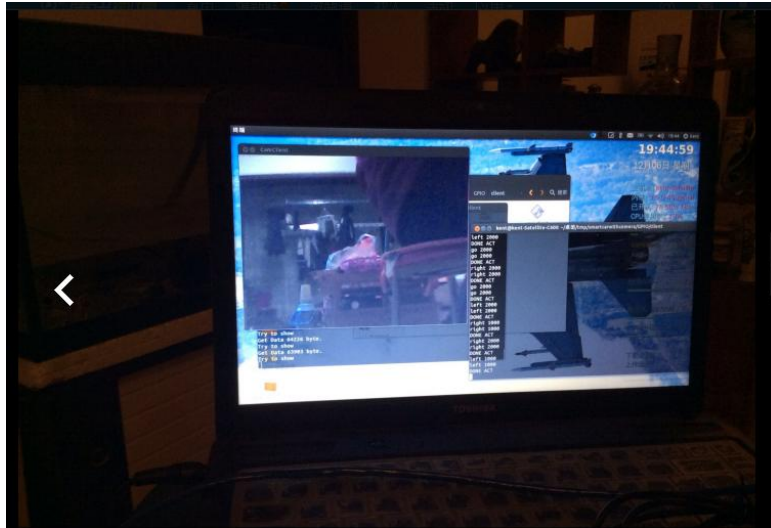
指令解析成功后，树莓派上的控制指令服务器会根据指令控制 **GPIO** 口的输出电平，将相应的输出口的电平拉高一段时间，之后拉低电平并且从 **TCP** 连接返回“操作完成”信息。

3.3 全部代码（不在此文档中）

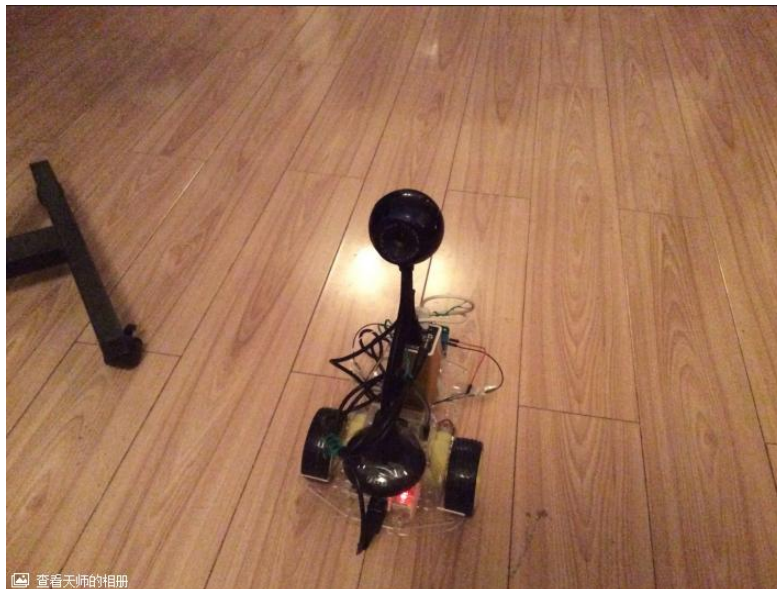
4 测试及结果

在完成设计之后，我对智能小车进行了简单的基本功能测试。

系统启动之后，笔记本成功接收到了树莓派通过 USB 摄像头采集的图像数据，客户端下发指令之后，树莓派服务器成功驱动小车进行了相应操作。



图为 客户端成功显示由智能小车回传的图像数据



图为 小车制作完成后的外观

结 论

本设计在现有开源软件与开源硬件的基础上,改进简单的系统设计与软件编程,用最简单最快速的方案构造了智能小车的图像回传与智能控制。

相信本设计对当前理工学生会有一定启示。

本设计存在很多问题,代码上存在缺陷,性能上也有不足。

比如图像回传系统中客户端会多次读取服务器传输的数据,但是在 GPIO 控制系统中,就没有进行多次读取。虽然 WLAN 的 MTU 有 1488 字节,而 GPIO 的控制指令肯定不会超过 100 字节,但是如果网络速率不佳,控制指令传输不全的情况有可能会出现。

再比如当前即便是在树莓派服务器端,对于采集到的数据进行了有损的图片压缩,使得数据由 70000 字节以上的原始数据压缩到了 2000 字节左右,仍然无法使得客户端上的图像显示形成连贯的视频。并且没有对音频进行采集,没有音频与视频的同步等等。

但本设计的目的就在于一个简单的实验,以及学习 OpenCV 的简单使用。通过不择手段的快速设计,以锻炼我自己的创造力与实现能力。整个设计在 4 天之内顺利完成。

参考文献（电子发行版）

- [1] Open Source Computer Vision Library (<http://opencv.org/>)
- [2] C and C++ support for Broadcom BCM 2835 as used in Raspberry Pi
(<http://www.airspayce.com/mikem/bcm2835>)