


售货机安全卫士技术文档

User Guide

内部文件：B07-1710024EM01

颁布时间：2017/10/2

目 录

	文件版本说明.....	2
	参考资料.....	2
	手册目的.....	2
	声明.....	2
	名词定义和缩略语说明.....	2
1	项目名称.....	3
2	设计要求及性能指标.....	3
3.	项目总体设计方案.....	3
3.1	设计方案.....	3
3.2	功能概述.....	3
3.3	技术难点.....	4
4	模块详细.....	4
4.1	主控中心介绍.....	4
4.1.1	模块软件设计.....	5
4.1.2	模块硬件设计.....	6
4.2	继电器和报警灯.....	7
4.2.1	继电器和报警灯的硬件设计.....	7
4.2.2	软件设计.....	8
4.3	磁力开关模块.....	8
4.3.1	磁力开关硬件设计.....	8
4.3.2	磁力开关软件设计.....	9
4.4	振动传感器阵列.....	10
4.4.1	振动传感器硬件设计.....	11
4.4.2	振动传感器软件设计.....	11
4.5	摄像头模块.....	14
4.5.1	摄像头硬件设计.....	14
4.5.2	摄像头的软件设计.....	15
1.	安装摄像头工具包.....	16
2.	使用 opencv.....	16
4.6	GPRS 模块.....	16
4.6.1	GPRS 硬件设计.....	17
4.6.2	GPRS 软件设计.....	17
4.7	声音播放设备.....	18
5	软件与算法设计.....	19
5.1	软件流程设计.....	19
5.2	通信协议设计.....	19
5.2.1	通信协议概述.....	19
5.2.2	通信协议数据格式.....	20
5.2.3	通信协议数据解析.....	20
5.2.4	图像数据的通信.....	21
5.3	技术开锁认定.....	21

文件版本说明

表 1 版本说明

版本	发布时间	修订章节	作者
1.0	2017/10/2	创建文档	Carlos Wei
1.0	2017/10/3	编辑文档	Chris Zhu

参考资料

1. 无

手册目的

该手册用于用户参考、传达设计思路、兼具培训讲义功能。

声明

手册内部培训使用，不要上传到任何公共区域。

名词定义和缩略语说明

表 2 名词定义及缩略语说明

序号	缩写	说明
1	报警行为	闪光灯闪烁 拨打电话 播放警报

1 项目名称

《售货机安全卫士》

2 设计要求及性能指标

根据需求分析书，附录 B07 文献中，设计一个售货机防盗检测装置，该装置具备防盗检测的功能，即可以检测暴力及技术开锁盗窃行为、给出报警指示（包含电话呼叫、现场警报播放、现场防盗灯闪烁）、并可以由用户远程获取现场照片、并播放语音警告盗窃者。

3. 项目总体设计方案

3.1 设计方案

根据需求分析书，系统设计方案如下：以树莓派 3 代 B+作为 CPU 的主控中心，外围有报警闪光灯及继电器模块、GPRS 模块、能够发出声音的音响设备、振动传感器阵列、CSI 接口的摄像头、检测门被打开的磁力开关、还有电源装置为整个模块提供电源。如图 1-1 所示为整个系统的结构，数据方向如图所示。

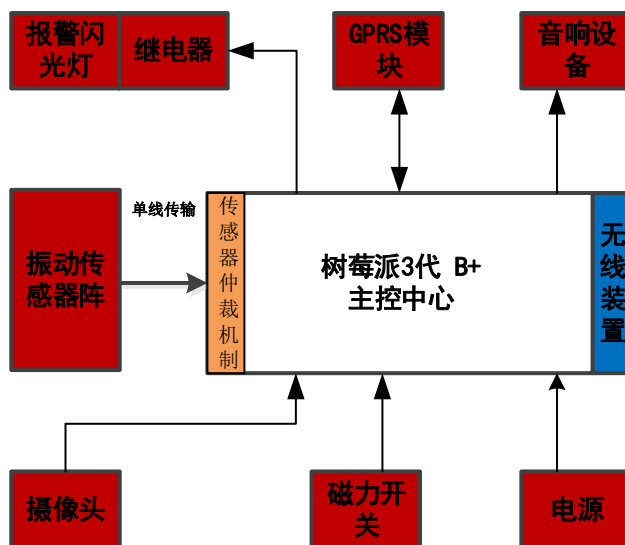


图 1-1 系统结构图

3.2 功能概述

树莓派 B+作为主控中心，能够安装 Linux 系统，在系统上整体运行统筹控制外设和实现控制算法，在系统以系统服务机制的方式随系统启动在后台静默运行，（后文将树莓派为核心的主控中心简称为**主控中心**）；GPRS 模块能够实时的进行电话拨打，当被盗检测信号发生时，主控中心将向 GPRS 模块将发送 AT 指令，控制其拨打拨打电话；磁力开关模块能够检测售货机的门是否被打开，类似于冰箱内灯一样，当我们开启冰箱门时，冰箱内部的灯就会打开，为我们照明，当冰箱的门闭合的时候，内部的灯会被熄灭，节约电能，

磁力开关的作用就是如此；摄像头模块负责现场照片的采集，当需求拍照的指令发送给主控中心时，由主控中心驱动摄像头提取图像数据；音响设备选择在市面上最常见的迷你小音箱，由于主控中心采用树莓派 B+，上面正常运行 Linux 系统，有集成的声卡，所以可以采用常见的播放设备；振动传感器阵列，负责检测暴力盗窃行为。在这一章中，只对功能进行解释，在后续章节将分模块对各个功能进行剖析。

3.3 技术难点

在本设计中，有几项技术难点需要。

- 1) 振动传感器的仲裁机制。
- 2) 图像数据的传输，此处使用 SCI 协议对图像进行传输，需要将图像包装成字节流形式。
- 3) 拍照驱动。

我们将在后续的章节，给出难点的解决方案，**会使用蓝色字体标记。**

4 模块详细

为了简化和方便用户阅读，在本节，本文将以模块的方式分别进行详细的介绍，每一个章节中将包含选型、模块特征、模块软硬件设计的方式给出。在本章的最后一节，将给出所有的模块的硬件连线 and 软件流程设计。**请在阅读前看好标题的层次，方便理解。**

4.1 主控中心介绍

主控中心选用第三代树莓派 B+，

它是一款基于 ARM 的微型电脑主板，以 SD/MicroSD 卡为内存硬盘，卡片主板周围有 1/2/4 个 USB 接口和一个 10/100 以太网接口（A 型没有网口），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，以上部件全部整合在一张仅比信用卡稍大的主板上，具备所有 PC 的基本功能只需接通电视机和键盘，就能执行如电子表格、文字处理、玩游戏、播放高清视频等诸多功能。 Raspberry Pi B 款只提供电脑板，无内存、电源、键盘、机箱或连线。
(引用于/百度百科/树莓派磁条)

通过上面的几段文字，我们可以总结出来树莓派就相当于一台迷你的小电脑，而这个小电脑的价格仅仅只有近 300 元。相比于单片机或者 DSP 的片上编程，树莓派可以在 Linux 或者上面运行 windows 系统运行，性能十分强大，接口十分友好，**无论是成本还是功能上**，非常适合于我们的这个项目。另外，树莓派的多任务性，允许在此项目中也预留出了其他的空间，就允许我们类似于积木一样，不断增加其他功能和进行升级。

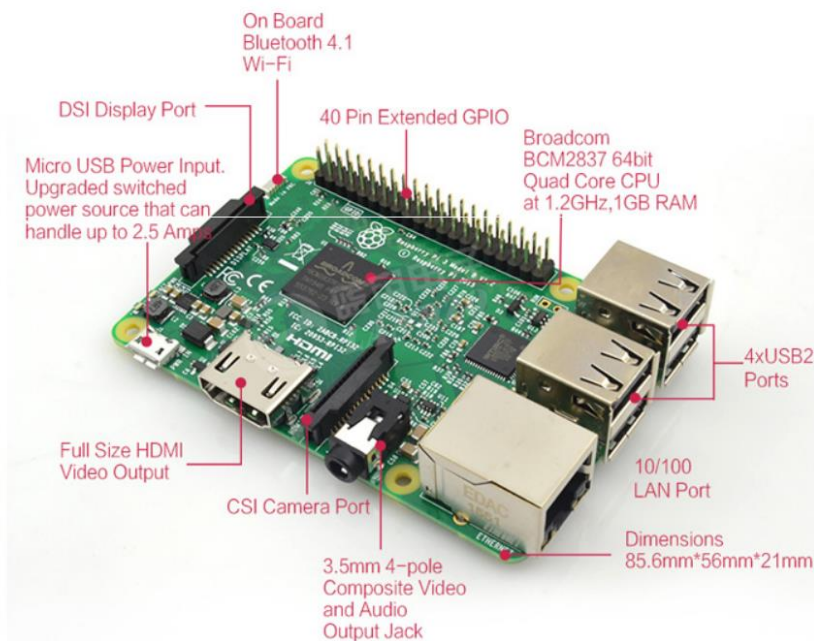


图 2 树莓派外观图和外设资源配置

我们利用 CSI Camera Port 接我们的摄像头模块，利用 3.5mm 4-pole Audio 接口接入我们的音响设备，在 USB 接口上插入鼠标键盘、在 HDMI 接口插入我们的显示器，在 40 Pin Extended GPIO 上面接入我们各种传感器设备，按照传感器的文档进行相关的驱动开发。

4.1.1 模块软件设计

主控中心的软件设计，则是我们整个项目的重中之重，在整个系统上需要对：

- 1) 各个模块的驱动和检测
- 2) 防盗算法设计
- 3) 工作流程设计

这三大部分是整个项目核心中的核心。在模块上面我们选择使用 Python 语言进行开发。**下面对于软件这一部分我们采用 FAQ 的方式进行，请仔细阅读问题和答案，可能是在比赛中评委会问到的。!!! 请一定要配合附录文档观看，如果我把附录文档引入这里，很可能打乱我们的文档结构，所以我们这里给出指引，什么时候看什么文档，看哪部分。**

Q1：为什么要选择 Python 进行开发，树莓派当然不是也支持 C++\C\Java 等等语言？

A1：Python 作为树莓派最开始使用的语言，经历了漫长的验证和广大的用户群和前辈开发，则有着丰富的库函数，极大的缩短了我们的开发周期。

Q2：拿到树莓派我们第一步应该做什么？

A2：刚拿到的树莓派，上面是没有任何操作系统的，就像是我们新买的计算机一样，硬盘里面没有任何的操作系统，我们需要做好树莓派的连接（例如电脑连上键盘鼠标显示器准备好系统安装盘），相应地，在树莓派上，我们也需要连接上鼠标和键盘还有连接好显示器，相对于电脑的安装盘，我们只需要准备一张手机或者照相机使用的 microSD 卡即可。但是 microSD 卡则需要在电脑上运行一个软件把在树莓派上下载好的操作系统准备好。**详见附录文档(X1)。**

Q3：安装完系统该如何使用，让其正常工作。

A2: 当我们安装完操作系统的时候, 就会进入到树莓派的系统里面, 你会发觉可能和我们使用的 windows 系统不一样, 但是你可以通过鼠标点击上面的应用可以通过键盘打字。这个就是一个 Linux 系统, 这个系统会稳定的运行在树莓派这个硬件上, 此时我们可以在系统的桌面上点击右键 新建一个空白文档, 将文档的名字改为 main.py 此时我们就建立了一个 python 的脚本文件, 我们按照 python 的规则在上面进行写 python 语言, 通过控制台来运行这个脚本文件就可以让树莓派按照脚本文件执行任务了, 我们的整个项目的思路就是这样。**如何在树莓派上使用 python, 请详见附录文档 X2 的第二章开始 (19 页-34 页) 跟着书上面做, 完成在控制台输出 “hello world.” 这几个字。**

上面, 则是对主控中心的一个最基本的环境配置, 完成这里的时候我们已经给做项目完成了一个基础, 可能到此你只是按照你的要求输出了 Hello World 这几个字, 但还是对如何控制硬件如何拍照这些产生了疑问, 我该怎么去做呢? 我该写什么样的字母和数字的组合让树莓派按照我的想法完成它。当你输入 “printf” 的时候意思是让树莓派向屏幕打印, 当然还有其他的命令, 让树莓派去做更复杂的, 更多元, 更有顺序的工作。我们会在下面模块介绍中介绍, 如何去让他工作。

4.1.2 模块硬件设计

在家里新买了电脑的时候, 我们需要组装好电脑才能使用, 你要知道键盘要接到 USB 口上, 鼠标要接到 USB 口上, 我们要给电脑主机插上电源, 还要连接显示器, 这些工作都是为了让电脑开机而做的准备, 而做的这些工作我们可以统称为硬件设计亦或是硬件连接。那么对于树莓派, 这个微型的专用的电脑也需要这样的工作, 我们要让家里新装的电脑的显示器好使的, 就要链接正确的显示器接口, 那么我们如果让树莓派能够检测盗贼暴力破机的行为, 则就要正确的连接传感器, 还有我们想让树莓派能够给我们打电话, 那么我们就要正确的连接 GPRS 模块, 那么这一切, 都是可以说是在树莓派上面的硬件设计, 我们的目的就明确了, 接对线!

那么如何接呢? 显示器有那样特殊明显特征的接口, 鼠标和键盘都是 USB 的, 电脑主机上只有 USB, 我们毫无难度的就可以把硬件连接起来, 但是树莓派不一样, 除了极个别的, 他们的样子都是一样的, 如图 3 所示:

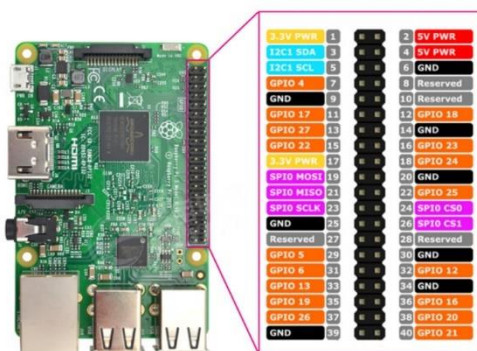


图 3 树莓派的接口

这些扩大化的粉色框内的就是连接我们各种传感器的接口的。那问题又来了, 如何接? 怎么接? 每个传感器都有自己的特征, 他们有的需要三根线通信有的需要四根线五根线, 甚至还有的需要几十根线, 这些传感器的自述自我介绍都在每个传感器的说明书中, 传感器的说明书会告诉我们如何去衔接。我们在下一个小节的

各个传感器中介绍, **介绍传感器的顺序是先易后难的。**

4.2 继电器和报警灯

我们这里选择了继电器和报警灯，继电器用来控制报警灯的开关，报警灯用来在自动售货机上闪烁，在发生盗窃时就会响，警示盗贼。

4.2.1 继电器和报警灯的硬件设计

先说报警灯，报警灯正常直接插上 220V 的市电之后就开始闪烁，可是我们不能让他一直在闪烁，我们需要等到盗窃行为发生时进行闪烁，这里让他按照我们的意愿进行闪烁，就引入继电器模块，继电器模块当我们的树莓派给出信号，则会让电通，报警灯通了电就开始闪烁，当我们的树莓派没有给信号，则电一直断开，报警灯没有电就不闪烁。



图 4 报警灯实物图



图 5 继电器实物图

我们把继电器和报警灯组合起来，然后从继电器中引出一根信号线接到树莓派上面，树莓派可以通过一根信号线来控制报警灯的通断了。



图 6 继电器和报警灯接线图

右侧信号线要和树莓派建立起连接，我们上面说给的信号，根据继电器的说明书，当信号线给的是低电平的时候，继电器会控制图 6 中橘黄色的一端是断开的，则报警器和 220V 的市电不构成回路，那么报警灯就不

会亮：当信号线给的是高电平的时候，继电器会控制图 6 中橘黄色的一端是闭合的，那么报警灯此时和市电形成回路，报警器亮。这就完成了报警灯的控制。我们会在下一小节中告诉你如何通过编程控制树莓派的信号线给高电平和低电平。该模块和树莓派的连接图如图 7 所示：

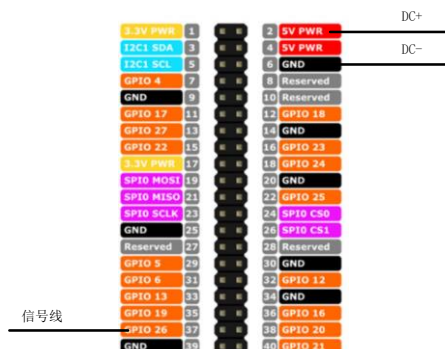


图 7 树莓派连接位置图

4.2.2 软件设计

上面的章节说到了，我们需要控制一根信号线，这个信号线发射出高电平，或者低电平来控制继电器的通断，那么我们很容易想到使用 GPIO 口，GPIO 口的配置步骤：

- 1) 选定一个 GPIO 口
- 2) 设定这个 GPIO 口为输出方向。
- 3) 制作一个函数，让 GPIO 口输出高电平
- 4) 制作一个函数，让 GPIO 口输出低电平

就这四个功能，那么对应的我们如何用 Python 语言来表达呢？

- Python 第一步需要引入控制 GPIO 的库，在程序的最前面加入这个语句

```
import RPi.GPIO as GPIO
```

- 选定一个 GPIO 口，这个 GPIO 口就应该是我们上一节连接信号线的口，是 GPIO26 口

```
vibSensorAlarmLight = 26
```

- 设定该 GPIO 口为输出方向

```
GPIO.setup( vibSensorAlarmLight, GPIO.OUT )
```

- 制作一个函数让 GPIO 口输出高电平的，这里为了更好地理解我们这样命名：

```
def startAlarmLight( ):
```

```
    GPIO.output( vibSensorAlarmLight, GPIO.HIGH )
```

- 制作一个函数让 GPIO 输出低电平，也是为了好理解我们这样命名：

```
def closeAlarmLight():
```

```
    GPIO.output( vibSensorAlarmLight, GPIO.HIGH )
```

这个时候我们就写完了报警灯的驱动程序，当我们检测到盗贼行为的时候就 startAlarmLight 函数就可以了。

4.3 磁力开关模块

4.3.1 磁力开关硬件设计

磁力开关模块能够检测售货机的门是否被打开，类似于冰箱内灯一样，当我们开启冰箱门时，冰箱内部的灯就会打开，为我们照明，当冰箱的门闭合的时候，内部的灯会被熄灭，节约电能，磁力开关的作用就是

如此。该传感器的原理非常简单，通过感应磁场的有无，当有在传感器的范围内有磁场，传感器则一直输出高电平的信号，当传感器离开磁场，则输出低电平的信号，所以我们通过检测传感器输出的信号，来间接检测门是否打开。

在架设传感器方面，我们选择软性磁铁，该磁铁能够粘在门的一侧，传感器在门的另一侧，当门闭合的时候，传感器和软性磁铁靠近，当门打开的时候，传感器和软性磁铁分离，则产生不同的信号，这样我们就能够检测门是否被打开。

表 1 磁力开关技术规格

技术规格/specifications	
尺寸	24*29mm
电压	3.3-5V
接口	GPIO 接口
用途	位置、转速检测与控制，安全报警装置，纺织控制系统
控制	无磁铁保持高电平，有磁力输出低电平，带有 LED 输出指示

实物图如图 7 所示，

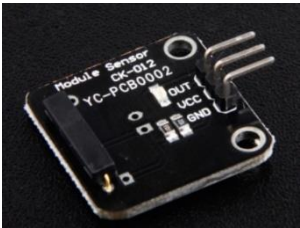


图 7 磁力开关实物图

磁力开关有三个接口，一个是 VCC，一个为 GND，另一个为 OUT，是输出信号。我们通过树莓派供电，从 OUT 口得到开关信号，实现检测即可。图 8 为接线图。



图 8 接线图

4.3.2 磁力开关软件设计

信号分析，如图 9 所示，为磁力开关开门和关门时信号输出的信号抓取，在中线处可以信号显示出一个上升沿，我们在软件中抓取这个上升沿，即可检测门是否被打开。

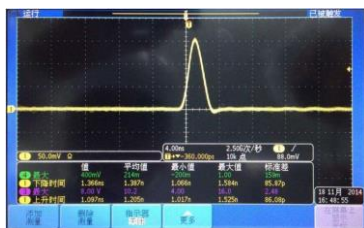


图9 磁力开关信号输出

同上一个继电器一样，我们一样用到 GPIO 口，上一个为输出信号，这个是输入信号，那么配置步骤：

- 1) 选择一个 GPIO 口作为磁力开关的信号线。
- 2) 设定 GPIO 口为输入模式。
- 3) 设定事件捕捉，这里用上升沿捕捉。
- 4) 写事件处理函数，也就是当捕捉到上升沿之后自动执行的函数。

那么在 Python 中如何表达着事件呢？

● 选择 GPIO21 作为信号线

vibSensorDoorCheck = 21

- 设定 GPIO 口为输入模式

```
GPIO.setMode( vibSensorDoorCheck, GPIO.IN )
```

- 增加捕捉事件和定义处理函数

```
def SensorA1CallBackEvent( channel ):
```

```
    #####处理任务内容
```

```
GPIO.add_event_detect( vibSensorDoorCheck,GPIO.RISING callback = SensorA1CallBackEvent )
```

这个就是处理函数增加的方法。

当这个引脚检测到有上升沿的时候，自动就会跳入那个我们指向的 CallBack 函数。

4.4 振动传感器阵列

振动传感器使用高灵敏震动传感器 SW-1801P 报警感应模块，该模块由 TELESKY 公司生产制造。该模块的特色：

采用高灵敏度震动开关，默认用 SW-18010P 震动传感器。

比较器输出，信号干净，波形好，驱动能力强，超过 15mA

3、工作电压 3.3V-5V

4、输出形式：数字开关量输出（0 和 1）

5、设有固定螺栓孔，方便安装

6、小板 PCB 尺寸：3.2cm x 1.4cm

7、使用宽电压 LM393 比较器

该模块的使用说明：

1、产品不震动时，震动开关呈断开状态，输出端输出高电平，绿色指示灯不亮；

2、产品震动时，震动开关瞬间导通，输出端输出低电平，绿色指示灯亮；

3、输出端可以与单片机直接相连，通过单片机来检测高低电平，由此来检测环境是否有震动，起到报警作用（来源于振动传感器使用说明书）

从说明书中我们可以看到，该传感器依然使用 GPIO 口进行状态检测，原理类似于上一节的磁力开关传感器，检测方式依然设定 GPIO 口为输入模式。如图 10 所示为实物图。



图 10 振动传感器

在本设计中由于要求对于振动传感的准确性，所以不能使用单个传感器，要使用多个传感器进行组网，组成一个传感器阵列，根据不同的传感器不同的位置有不同的加权，当这些传感器阵列加权后的值超过警戒值则判定为盗窃行为。

4.4.1 振动传感器硬件设计

这里共使用四个振动传感器组成网络，当然，如果想更进一步提高传感器的精度，则可以扩展更多的传感器组成传感器阵列，传感器的数量越多，则越精准，而对于树莓派的处理速度，可挂载几百个振动传感器同时工作。

振动传感器一共需要用 3 个引脚，根据说明书，需要给定 5V 的电源，另一端为信号输出，原理同磁力开关传感器，这里不再赘述。本文给定振动传感器的接线：

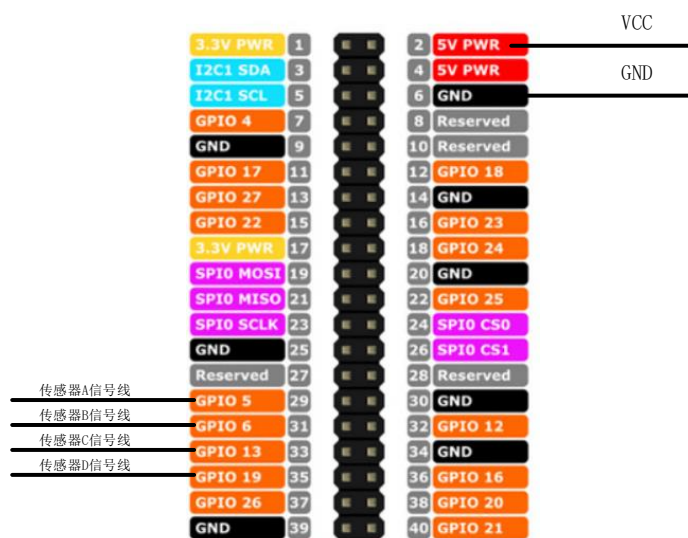


图 11 四个振动传感器接线方式

4.4.2 振动传感器软件设计

对于振动传感器的软件设计，在树莓派的 Python 设计方法上，与磁力开关在方法上没有任何差别。程序如下：

```
# Init Sensors
vibSensorA0Channel = 5
vibSensorA1Channel = 6
vibSensorA2Channel = 13
vibSensorA3Channel = 19
```

```
GPIO.setmode( GPIO.BCM )
GPIO.setup( vibSensorA0Channel, GPIO.IN, pull_up_down=GPIO.PUD_DOWN )
GPIO.setup( vibSensorA1Channel, GPIO.IN, pull_up_down=GPIO.PUD_DOWN )
GPIO.setup( vibSensorA2Channel, GPIO.IN, pull_up_down=GPIO.PUD_DOWN )
GPIO.setup( vibSensorA3Channel, GPIO.IN, pull_up_down=GPIO.PUD_DOWN )
```

到此完成 GPIO 的配置

```
# Add GPIO event processor
def SensorA0CallBackEvent( channel ):
    global bool_vibSensor0State

    bool_vibSensor0State = True
    print("Detected the A0 OK")
GPIO.add_event_detect( vibSensorA0Channel, GPIO.FALLING, callback = SensorA0CallBackEvent )

def SensorA1CallBackEvent( channel ):

    global bool_vibSensor1State

    bool_vibSensor1State = True
    print("Detected the A1 OK")
GPIO.add_event_detect( vibSensorA1Channel, GPIO.FALLING, callback = SensorA1CallBackEvent )

def SensorA2CallBackEvent( channel ):

    global bool_vibSensor2State

    bool_vibSensor2State = True
    print("Detected the A2 OK")
GPIO.add_event_detect( vibSensorA2Channel, GPIO.FALLING, callback = SensorA2CallBackEvent )

def SensorA3CallBackEvent( channel ):

    global bool_vibSensor3State

    bool_vibSensor3State = True
    print("Detected the A3 OK")
GPIO.add_event_detect( vibSensorA3Channel, GPIO.FALLING, callback = SensorA3CallBackEvent )
# _^_____ The Function end.
```

上面的程序用四个颜色标注，则代表四个检测函数，对信号进行下降沿判断，当检测到判断之后，进入到响应的 CallBack 函数。

上述仅仅完成了驱动的配置，你可能现在问，该如何融合算法？

原理如图 11 所示：

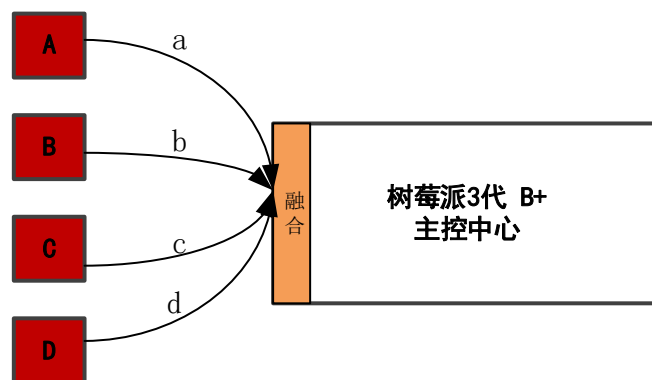


图 11 传感器仲裁机制算法

图 11 中，a,b,c,d 为四个传感器的权重系数，同一时刻振动信号产生，四个传感器在四个不同的位置，如果感受的强度激发了传感器则量化强度为单位 1，此时通过 a,b,c,d 不同的系数组合加权完成强度 E 的求解，那么 E 为：

$$E = a + b + c + d \quad (a, b, c, d < 1)$$

如果 $E > 1$ ，则判定为判定为盗窃行为，这个 abcd 权重的系数要根据实际情况（包括面板厚度，传感器位置，面板材料）调整到最好的位置。

上述的原理清晰了，那么如何用 Python 来实现它呢？我们分析一下上面所需要的条件，第一个条件需要在同一时刻进行传感，这个时刻四个传感器可能有一定的先后顺序，不可能完全达到并发，所以同一时刻不如表达成为同一个允许的时段，在这个极短的时段内，四个传感器传感的加权树莓派会进行相加，如果四个传感器分散传感超过了这个时段，则数据会判定为无效，所以这里需要一个计时机制，来判断传感器的数据是否有效。这样表述可能比较抽象，下面用图片的方法来说明。

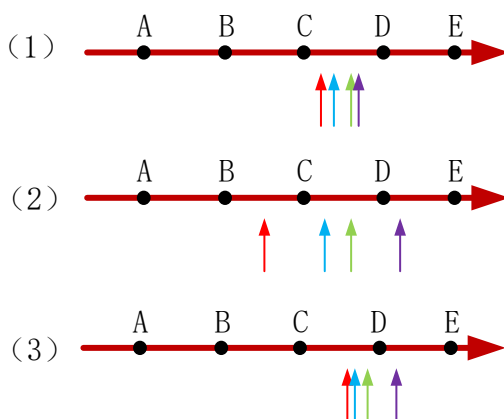


图 12 时间判定机制原理

我们来观察图 12 时间判定机制，深红色数轴表示时间，上面 ABCDE 五个点将数据分为五个区域，线段 AB, BC, CD, DE 为 4 个有效时间段，下面四个颜色的箭头为四个传感器传递四个权重系数不一样数值，先看图 (1)，在 (1) 中，四个箭头落入 CD 线段，在一个有效时间段，则该四个数据判定为有效数据，然后树莓派就会进行加权计算，如果结果超过 1 则认定为是盗窃行为；再看图 (2)，红色箭头落入 BC 线段，绿色和蓝色落入 CD 线段，紫色线段落入 DE，跨越了三个区域，则很显然，该组传递数据被抛弃；再看图 (3)，三个传感器的数据落入 CD 线段，但是第四个紫色箭头落入了 DE 线段，此时应该判定为无效数据，但这种情况，还是会被判定为有效数据，为什么？因为红色箭头和紫色箭头的距离长度没有超过 AB, BC, CD, DE 线段的长度。也就是说，这里判定的是相对长度。

时间机制的使用，则需要启动树莓派的定时器机制，定时器机制不断的计时，配合传感器阵列完成。那

么如何用 Python 实现。请查看项目源代码的 `main.py` 文件的, `CheckTheVibSensorsState` 函数。

4.5 摄像头模块

在树莓派的图片中找到 CSI 接口，那个通过 FPC 排线和摄像头衔接的即是我们使用的摄像头模块。在本设计中使用的摄像头模块特点如下：

500 万像素

感光芯片 OV5647

静态图片分辨率为 2592 × 1944

支持 1080p30, 720p60 以及 640×480p60/90 视频录像

尺寸: 25mm × 24mm × 9mm

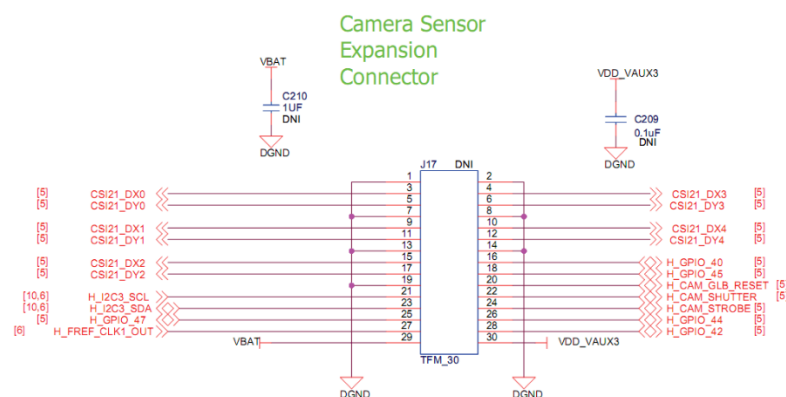
角度：65 度

如图 13 为摄像头的模块的实物图片:



图 13 摄像头实物图

4.5.1 摄像头硬件设计



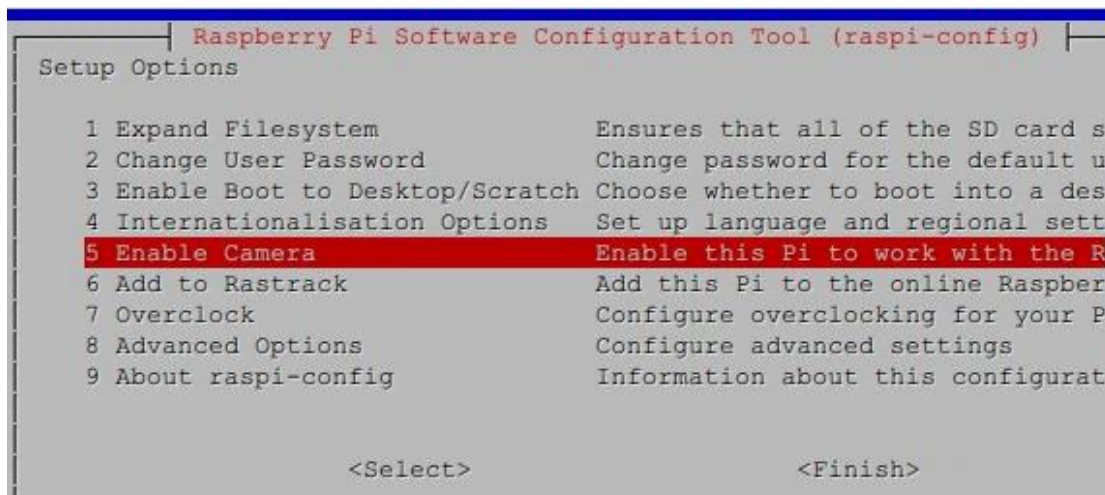
CSI 的引脚定义如图所示，CSI 是一个标准的接口，如同是 USB，作为应用场合不需要对其进行了解，只要知道 CSI 接口是存在的，和摄像头进行链接的即可。

4.5.2 摄像头的软件设计

（一）软件配置安装摄像头

首先确保树莓派的系统已经启动，并且开启摄像头功能。

打开 terminal 输入命令：`sudo raspi-config` 并勾选第 5 个的 `enable camera` 功能选择“YES”，然后点击“Finish”自动重启。如下图：



（二）尝试使用命令行拍摄照片

`raspistill` 命令是树莓派提供的捕获图像的工具，输入下面命令获取一张图像保存到 `image.jpg`

输入：`raspistill -o image.jpg -rot 180`

参数 `o` 是输出到文件的意思；参数 `rot` 是旋转图片 180 度，因为摄像头我是让他排线向上摆放的，所以需要旋转一下。命令输入后，会发现摄像头上的 led 灯亮，大概持续 7 秒钟左右熄灭，同时命令执行完毕。查看图片，发觉成像非常清晰。



调节照片等待时间，比如输入：`raspistill -t 20000 -o image.jpg`

参数 `-t` 调节速度，20000 则为速度。 5000 等同于 5 秒。

`raspistill` 常用的参数

`-v`: 调试信息查看。

`-w`: 图像宽度

`-h`: 图像高度

- rot: 图像旋转角度, 只支持 0、90、180、270 度
- o: 图像输出地址, 例如 image.jpg, 如果文件名为“-”, 将输出发送至标准输出设备
- t: 获取图像前等待时间, 默认为 5000, 即 5 秒
- tl: 多久执行一次图像抓取。

(三) 使用 Python 进行摄像头拍照

1. 安装摄像头工具包

安装 picamera 使用命令中断安装命令包: `apt-get install picamera`, 我们在开发的时候已经安装了这个包, 所以你已经不需要再装了, 但是要知道换个过程哦。

2. 使用 opencv

这里要注意一点, 树莓派官方摄像头插入后没有 `/dev/video0` 节点, 这就导致无法直接用 opencv 调用。

```
import numpy as np
import cv2 as cv

def take_photo():
    cap = cv.VideoCapture(0)
    ret, photo = cap.read()
    if ret:
        print "take photo successfully"
        cv.imwrite("./photo.png", photo)
    else:
        print "Error! Photo failed!"
```

输入这个代码, 就可以拍一个照片, 并且这个照片保存在我们的当前目录下, 名字叫做 photo.png。当手机端发送命的时候, 就会调用这个方法, 把现场的照片拍摄下来, 然后保存在树莓派的内存里, 待手机端点击读取照片的时候, 照片就会发送过去。

4.6 GPRS 模块

GPRS 模块就是我们打电话、发短信、用手机上网的模块, 这个模块每个手机里面都有, 如果没有这个模块那么我们将无法打电话, 发短信。而在这个应用里面, 我们对 GPRS 进行了改装, 让其面向物联网, 并做了物联网常用的接口, 串口, 本小节将介绍 GPRS 模块的使用和配置, 这个 GPRS 模块的名字为 Goouuu_A6 模组。

Goouuu_A6 模组简介

[A6 简介](#)

[GSM A6 模组:](#)

[尺寸 22.8×16.8×2.5mm;](#)

[工作温度-30°Cto+80°C;](#)

[工作电压 3.3V-4.2V;](#)

[开机电压>3.4V;](#)

[待机平均电流 3ma 以下;](#)

支持 GSM/GPRS 四个频段，包括 850,900,1800,1900MHZ;

灵敏度<-105;

支持语音通话;

支持 SMS 短信;

支持 GPRS 数据业务，最大数据速率，下载 85.6Kbps,上传 42.8Kbps;

支持标准 GSM07.07,07.05 AT 命令及 Ai Thinker 扩展命令;

支持 2 个串口，一个下载串口，一个 AT 命令口;

命令支持标准 AT 和 TCP/IP 命令接口;

支持数字音频和模拟音频，支持 HR, FR, EFR, AMR 语音编码;

支持 ROHS, FCC, CE, CTA 认证;

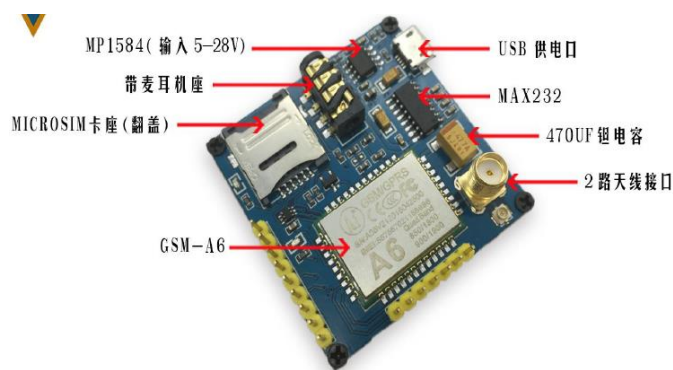


图 14 GPRS 模块整体结构图

4.6.1 GPRS 硬件设计

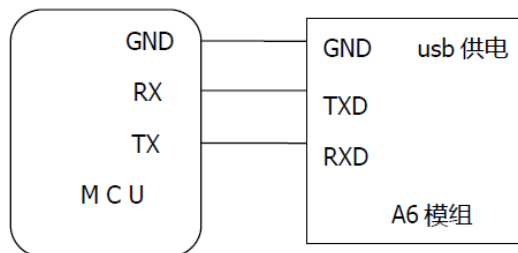


图 15 GPRS 接线方式

从图中可以看出，GPRS 通过串口进行通信，所以在树莓派上面还需要增加一个 USB 转串口的工具。然后树莓派通过 USB 和串口相连，我们通过串口协议发送数据给 AT 指令给 GPRS 模块，让 GPRS 模块，拨打电话，发短信，挂断电话等等。

4.6.2 GPRS 软件设计

那如何来控制 GPRS，则通过串口进行控制，那么既然是串口通信就涉及：正确的波特率，同样的停止位校验位等等。->那么串口如何控制呢？GPRS 遵循国际标准，则有一套国际通信协议，这个协议的名字叫做 AT 指令，AT 指令是一种字符串格式如同是“AT+MESS”这样的一串字符。我们至于按照协议标准用串口向 GPRS 发送 AT 指令的字符就可以控制 GPRS 模块了。

举个例子：

用串口发送“AT+CMGL”这几个字符，GPRS 模块会自动识别列出所有收到的短信信息。

用串口发送“AT+COPS”这几个字符，GPRS 模块会返回 SIM 卡运营商的信息。

类似的，AT 指令提供了很多功能，包含 2G 上网，发短信，拨打电话，只要按照 AT 指令的要求发送字符就可以完成对于 GPRS 模块的控制。**具体参考文档 X3。**

那么树莓派如何通过 Python 来控制 GPRS 模块呢？

- 首先要引入串口的 Python 包

```
import serial
```

- 定义串口句柄，设定好波特率信息

```
gprsPort = serial.Serial('/dev/ttyUSB0',115200)
```

- 定一个拨打电话的函数

```
# 首先是定义一个拨打电话 AT 指令字符串
```

```
# 红色标记就是你要拨打的电话号码
```

```
at_cmd_string_dialnumber = "ATD+8613812345678\r\n"
```

```
def CallTheHost():
```

```
    global gprsPort
```

```
    gprsPort.write( at_cmd_string_dialnumber )
```

```
# 我们只需要调用该函数就可以完成拨打电话的操作，当检测到盗窃行为的时候，即可完成。
```

4.7 声音播放设备

声音播放，就如同我们的音响，直接插在树莓派的音频接口即可，树莓派的硬件上集成了声卡，树莓派并在系统上自驱声卡驱动，这里仅仅需要 Python 进行对声音文件的播放即可。

- 首先要包含声音播放功能的 Python 包

```
import pygame
```

- 初始化声音播放功能

```
pygame.mixer.init()
```

- 构造声音播放子函数

```
def PlayTheToneAudio() :
```

```
    fileName = "/home/pi/script/a.mp3"          # 打开声音文件所在的路径
```

```
    pygame.mixer.music.load( fileName )          # 载入声音文件
```

```
    pygame.mixer.music.play( loops = 5, start = 0.0 ) # 播放声音，循环 5 次，立即开始
```

此时，声音就开始播放了。

5 软件与算法设计

5.1 软件流程设计

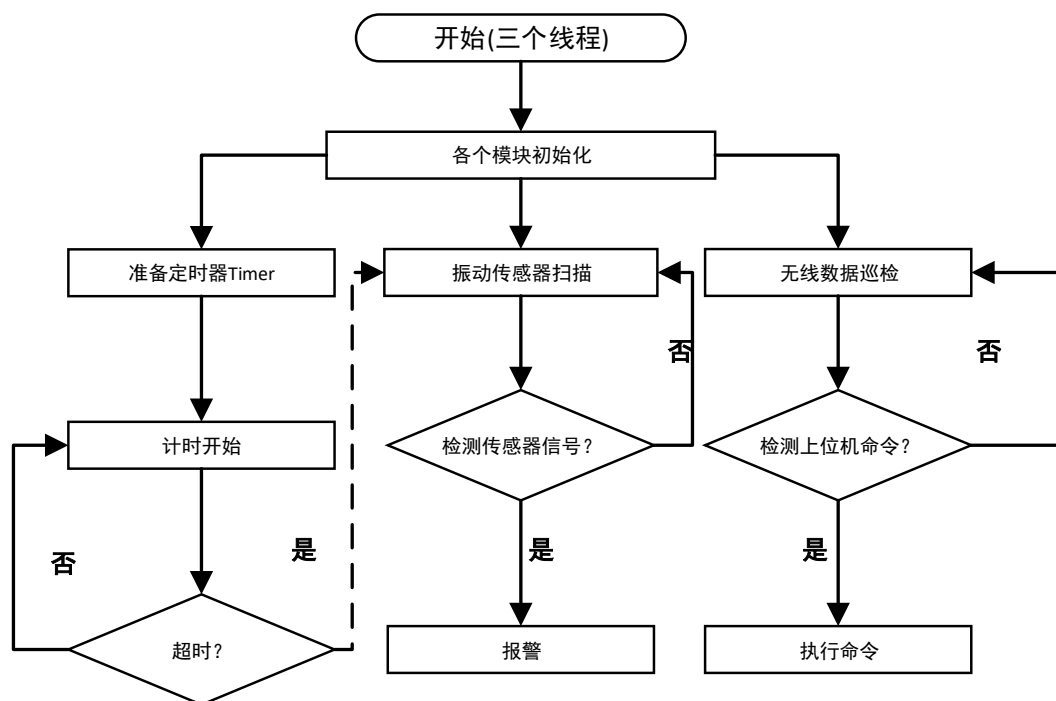


图 16 软件总流程图

为了更高效的利用树莓派，提升整体性能，我们启用了 Python 的多线程编程模式，一共启用了三个线程，一个线程用于定时器计时，为振动传感器阵提供时间信息，以更准确的判断暴力盗窃行为；第二个线程不断循环扫描振动传感器的状态，一旦检测到振动状态即报警；第三个线程负责通信，不断检测上位机（APP）是否传输过来命令，如果传输过来命令，立即去执行命令。

5.2 通信协议设计

5.2.1 通信协议概述

通信协议是指实体双方为完成通信连接或者命令动作所必须遵循的规定。要使得上位机和下位机完成通信并协同工作实现信息交换和资源共享，这就需要上下位机之间具有共同的语言作为二者的连接媒介，即通信协议。通信协议主要包含了做什么任务、怎样完成及何时完成，都得遵守实物双方都能接受理解的规定。通信协议包含了数据单元的格式，信息单元包含的信息，连接方式，信息发送和接收的时序，并确保网络中的数据可以准确地传送到指定实物。

通信协议主要由以下三个要素组成：

- 1、语法：即如何通信，包括数据的格式、编码和信号等级（电平的高低）等。
- 2、语义：即通信内容，包括数据内容、含义以及控制信息等。
- 3、定时规则（时序）：即何时通信，明确通信的顺序、速率匹配和排序。

首先要由上位机（电脑或者手机 APP）将树莓派所需的数据传送给树莓派，发送的命令为获取图像、播报语

音等等。

5.2.2 通信协议数据格式

数据头	命令内容	命令内容	数据长度	数据内容				补充数据
1	2	3	4	5	6	7	8	9

本文通信协议如下图 xx 所示，共有 9 个存储单元，每个存储单元都是一个 8 位的数据，以上每一个数据存储单元中存储的都是一个 8 位的数据，

1 号存储单元存放：数据头，为 0xAA

2 号存储单元存放：数据头，为 0xBB

3 号存储单元存放：命令内容

4 号存储单元存放：数据长度

5-8 号存储单元存放：数据内容

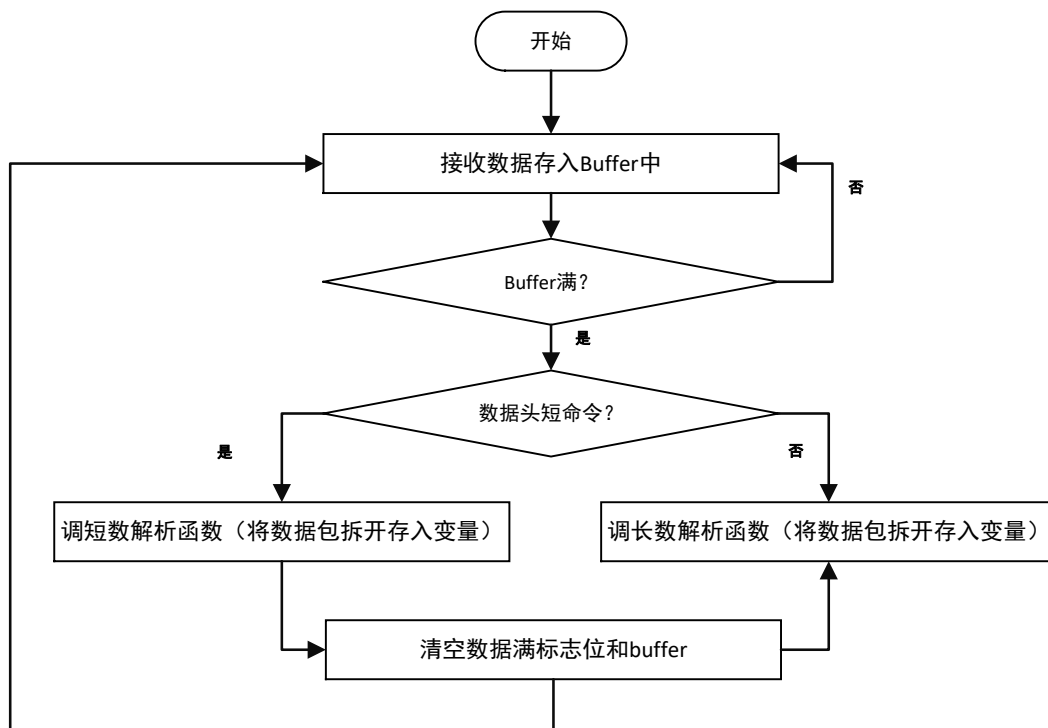
9 号存储单元存放：补充数据内容

为了树莓派对命令进行更好的识别以及对相应的通信数据进行处理，上位机需向树莓派发送数据头，目的就是告诉对方这是命令。命令代号识别给哪个命令，不涉及数据的比如温度请求之类的，为 0xAA。命令内容就是指该台电机需要完成的任务，数据长度就是告诉通信对方，下面多少是长度。数据内容这部分由于有浮点类数据，而串口不能传输浮点类的数据，所以这里需要对浮点数据进行一个简单的编码。

例如：1.889652 这个数据，将 1.889652×1000000 得到 1889652，整形数据，然后对数据进行分割，转换为 16 进制为 001CD574，那么数据位 5-8 就为 00 1C D5 74 拆分为 4 个，然后接到数据方将数据整合为一个 001CD574 最后进行除以 1000000 得到原始浮点数据。

5.2.3 通信协议数据解析

本设计的数据解析，在 while 主循环中进行，流程如下：



数据将会被存储到全局变量 Buffer 中，在利用到数据的时候自动就会去 Buffer 里面读取发送过来的数据。

5.2.4 图像数据的通信

在本文中，有个大数据量的数据需要单独进行判断，就是图像数据，由于将图像数据做成数据流的模式，无法确定图像的数据头和数据尾在哪里。所以需要人为的增加数据头和数据尾，在树莓派这一端，对图像数据进行编码，则在图像数据前增加“@@@”三个这样的字符，在数据尾增加“###”三个这样的字符以示图像数据结束。

具体如何解析，请见上位机软件技术文档。

5.3 技术开锁认定

在树莓派这一端，我们在 Python 上设定全局变量，身份认证标识位，在 main.py 中可以找到安全变量 bool_idCheck，这个变量只有在接受上位机正确的命令的时候才能被使能。如果售货机的门被打开，树莓派就会扫描该标志位，如果这个位被使能了，那么树莓派不会报警，如果这个位未被使能，树莓派就会发生报警行为

表 5-1 表位记录

(This is the last page)