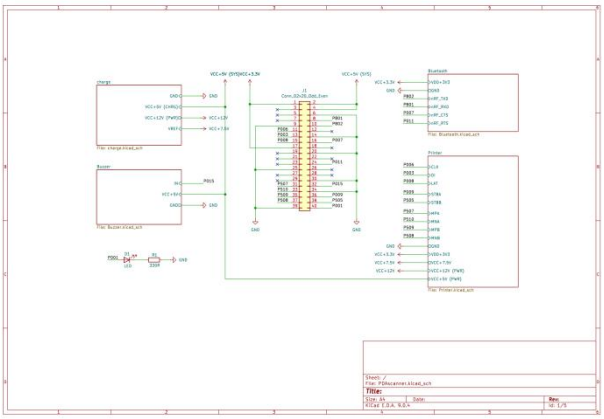
一、作品概要

1.1	作品名称	**基于 VisionBoard 的 PDA 扫描器设计**
1.2	项目概述 PDA 扫描器通常被称为工业 PDA、数据采集器或手持终端，广泛应用与物流、仓库管理和零售行业等，面对日益复杂的生产生活环境以及物联网、AI 技术的发展，PDA 扫描器也变的越来越智能化。本项目在 VisionBoard 基础上进行设计，可满足日常工作中复杂场景，基于瑞萨 RA8 系列的强大的算力，摄像头可实现快速准确识别条码，在电源管理上满足长时间充放电和续航能力，人性化操作设计，方面可以实现微打印功能，如仓库盘点或商品上货时方便打价签，提升工作效率。	
1.3	主要功能介绍 **1. 数据获取与微打印一体化设计** 作为 PDA 扫描器最具实用性的部分，扩展板设计有热敏打印机驱动电路（包括电机和加热控制部分），具有实时信息打印功能，摄像头识别到二维码或条形码后扫描得到的信息，如商品名称、价格、SN 号和批号等数据直接通过外接的热敏打印机很便捷地打印出来，无需等货架上的商品都扫一遍后再通过电脑打印，解决了后期裁剪价签和流程繁琐等问题，随扫随打可极大地避免因后期遗忘商品位置导致价签漏贴货架的情况发生。 **2. 业务数据实时共享——低功耗蓝牙无线传输** 该扩展板载有 nRF51822 蓝牙芯片，可实现端对端数据实时交互功能，也可无需建立连接，以 BLE 广播的形式单向传输，无论是其它扫描器终端设备还是手机，都可以做到安全共享业务数据，nRF51822 还侧重于蓝牙的低功耗性能，支持 1Mbps 的标准低功耗速率，其传输距离能达到 10-30m，完全满足门店和、中小型仓库的场景应用。 **3. 可满足长续航下的高可靠性电源管理** 对于 PDA 扫描器这种长时间工作的移动终端来说，充放电方面的功能不可忽视，该扩展板的电源管理部分采用国产厂商自主可控充电管理芯片（5V 充三串锂电池 12.6V）PW4053A 和 PW7126 保护芯片的经典组合方案的基础是增加了软硬件结合的充放电控制逻辑，采用可靠性强的逻辑互锁电路实现充放电过程切换的控制，从而形成一个功能完善的智能充电管理单元。	

二、硬件设计

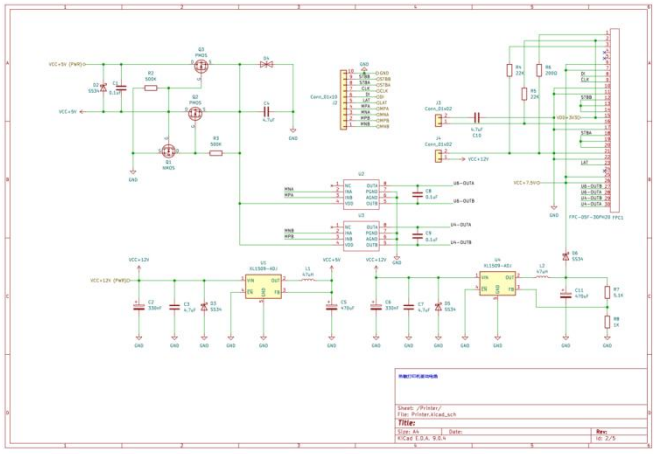
硬件部分已经集成了上述主要功能，分为打印机驱动电路、nrf51822 蓝牙模组最小系统、充放电电源管理电路、无源蜂鸣器驱动电路 4 个部分。

1. 原理图



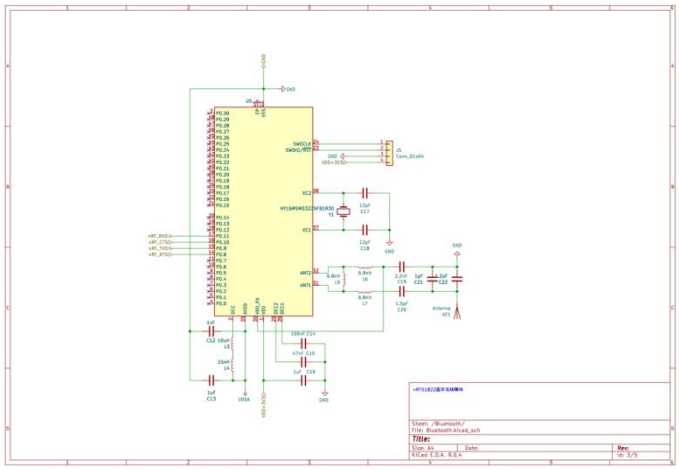
系统模块框图

热敏打印机驱动部分供电采用切换设计，VCC+5V（PWR）和 VCC+5V（VCC+12V 转+5V）同时给系统供电时，只会使用 VCC+5V，只要有 VCC+5V ，PMOS 管 Q1 导通使得 Q3 的 G 极接地，然后 Q3 也导通，Q2 的 G 极连接 VCC+5V，S 极基本也是 VCC+5V，所以 Q2 截止，U2 和 U3 电机驱动芯片的 VDD 引脚供电来自 VCC+5V，当 VCC+12V 拔出，VCC+5V 消失，则自动切换 VCC+5V（PWR）供电。这样双电源切换可以很好地使电机驱动在+5V 临时供电的情况下也能驱动，但不作为正常驱动，这是因为外部的热敏打印机还有一部分集成了加热模块，该部分也是设计了两个分别独立的 DC-DC 12V 转 5V 和 12V 转 7.5V Buck 电路，如果想稳定运行还是需要 VCC+12V 电源来驱动整改系统。采用 30P 的 FPC 柔性电路连接器将所有驱动信号和电源输出至外部打印机。



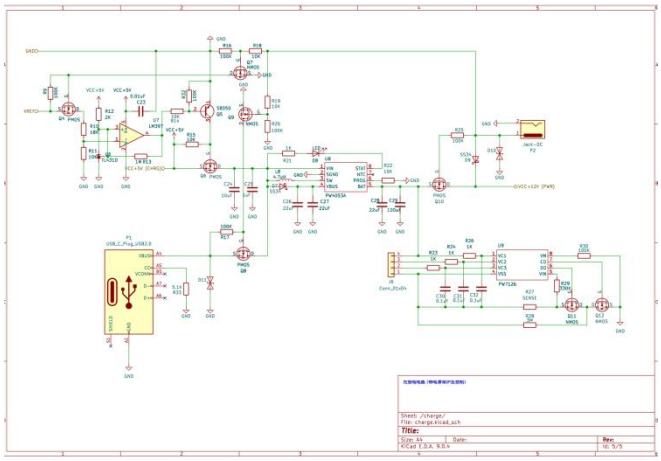
热敏打印机驱动部分

nRF51822 是一款集成了 2.4GHz 射频收发器、ARM Cortex-M0 内核、丰富外设和蓝牙低功耗协议栈的单芯片，其具有宽电压供电范围（1.8V-3.6V），这里直接使用了 VisionBoard 开发板提供的 3.3V 电源进行供电，并在 VDD 和 VSS 之间增加了一些去耦电容进行滤波，参考 Datasheet 采用 16MHz 作为芯片的时钟基准。最小系统天线部分设计重点参考了 Datasheet，采用 Datasheet 给出的 Pi 型 LC 平衡-非平衡转换电路，因为 nRF51822 的差分射频输出引脚（ANT1 和 ANT2）必须通过一个转换将芯片内部的差分信号转换为单端信号，以便与 50 欧姆的单端天线连接，采用一个电感和电容构成的 LC 网络实现，电感和电容的选取已通过理论计算选取，以确保在 2.4GHz 频段实现最佳的阻抗匹配。



nRF51822 无线蓝牙芯片最小系统部分

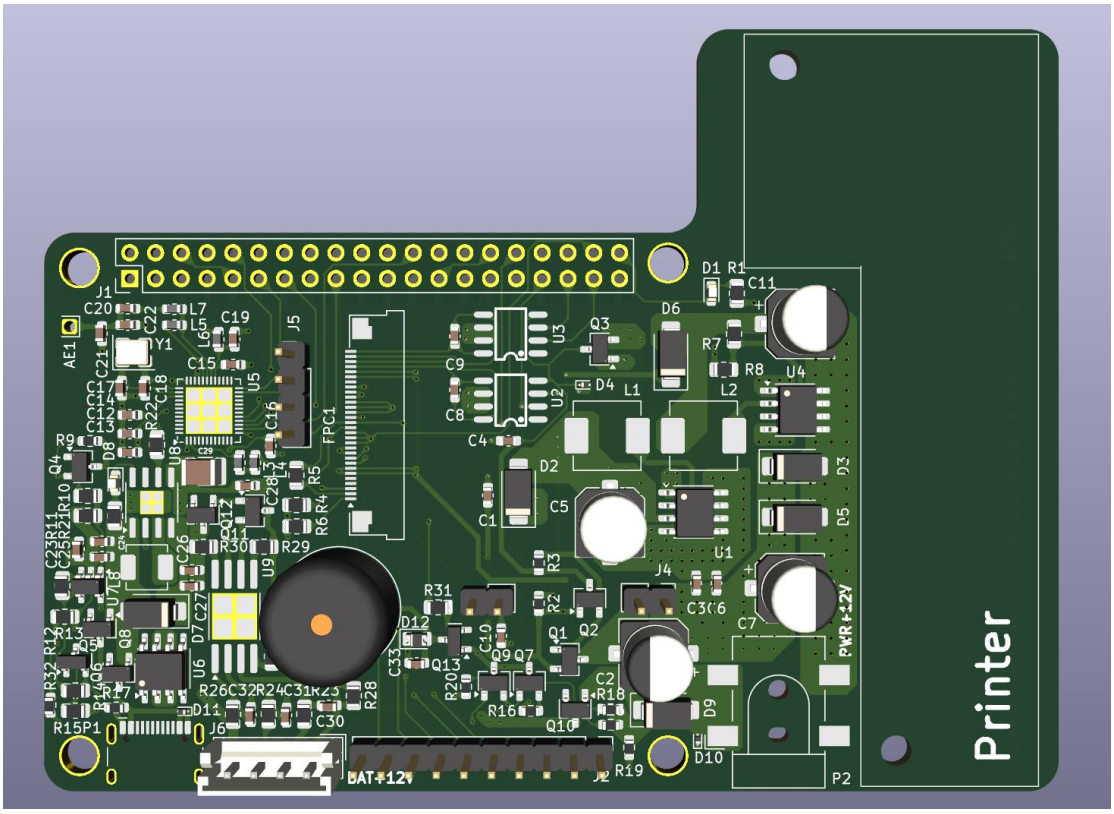
电源管理部分采用双电源充电设计，电池采用 12V 锂电池，可通过 5V 充三串 12V 锂电池，该充放电核心部分基于国产平芯微的 PW4053A 电源管理芯片和 PW7126 锂电池保护芯片标准设计电路，并增加 Type-C 充电端和 DC12V 充电端，切换实现采用 LM397 比较器、Q5 NPN 型三极管和 Q4、Q6、Q7、Q8、Q9、Q10 这 6 个 MOS 管实现，具体如原理图所示。

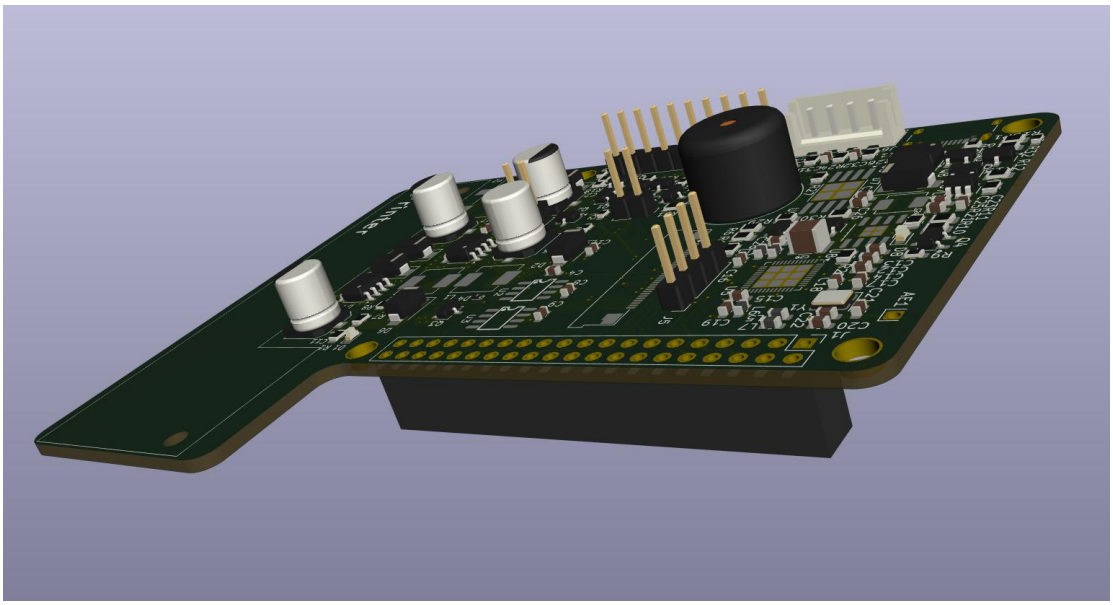


电源管理单元部分

2. PCBA 效果图部分

该 PCBA 为四层板，扩展板上预留有热敏打印机的放置位置，2.54mm2X20Pin 排母可以直插 VisionBoard 主板的扩展接口，采用+12V 电源供电，也可外接锂电池组输出+12V 进行独立供电，预留有 Type-C 充电接口。





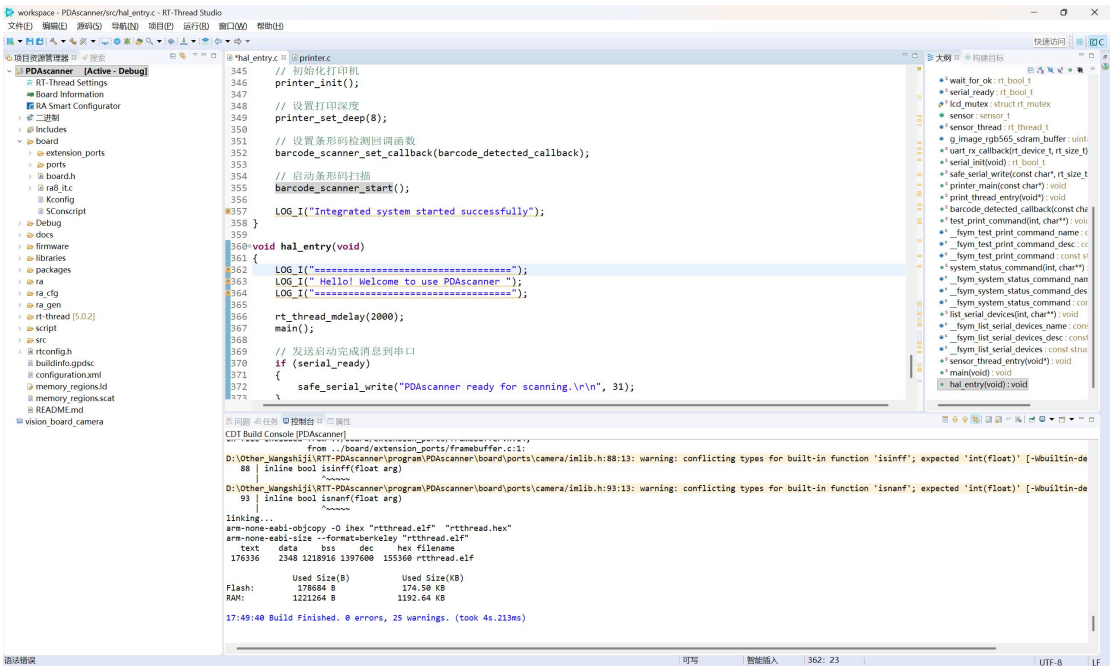
3. BOM 清单

日期	2025.11.3	优先级:	参考型号 > (值 + 描述 + 封装)					
版本	描述							
V10.2								
序号	名称	数量	位号	参考型号	值	封装	描述	厂商
1	FFC/FPC (柔性印刷电路) 连接器	1	FPC1	FPC-05F-30PM20		SMD,P=0.5mm,卧贴	翻盖式 下接 P数:30P 间距:0.5mm	XUNPU(讯普)
2	DC电源连接器	1	P2	DC-005-A200		插件	DC电源插座 内径:2mm 外径:6.4mm	XUNPU(讯普)
3	Type-C-16P连接器	1	P1	HC-TYPE-C-16P-01A		SMD	Type-C 母 卧贴	HCTL(华灿天禄)
4	2.54mm排针1x2P	2	J3, J4	PZ254V-11-02P		插件,P=2.54mm	1x2P 间距:2.54mm 方针 直插	XFCN(兴飞)
5	2.54mm排针1x10P	1	J2	PZ254V-11-10P		插件,P=2.54mm	1x10P 间距:2.54mm 方针 直插	XFCN(兴飞)
6	DC-DC电源芯片	2	U1, U4	XL1509-ADJ		SOIC-8	150 KHz 固定频率 PWM 降压 DC/DC 转换器	UMW(友台半导体)
7	有刷直流电机驱动芯片	2	U2, U3	TC118S		SOP-8	单通道直流马达驱动器	FM(富满)
8	肖特基二极管	6	D2, D3, D5, D6, D7, D9	SS34-E		SMA	电压:40V 电流:3A	MDD(辰达半导体)
9	静电和浪涌保护(TVS/ESD)二极管-5.5V截止	2	D4, D11	D5VDM1U2S9-7		SOD-923S	5.5V截止 峰值浪涌电流: 9A@8/20us	DIODES(美台)
10	静电和浪涌保护(TVS/ESD)二极管-12V截止	1	D10	D12VDM1U2S9-7		SOD-923S	12V截止 峰值浪涌电流: 4A@8/20us	DIODES(美台)
11	功率电感 47uH	2	L1, L2	SMMS0650-470M		SMD,7.1x6.6mm	47uH ±20% 2.8A	SXN(顺翔诺)
12	功率电感 4.7uH	1	L8	FXLD420-4R7-M		SMD,4.4x4.2mm	4.7uH ±20% 3A	cjiang(长江微电)
13	铝电解电容 330uF	2	C2, C6	JVJ16V330M6x8		SMD,D6.3xL7.7mm	330uF ±20% 16V	JIEKR(捷而瑞)
14	铝电解电容 470uF	1	C5	JVJ16V470M6x8		SMD,D6.3xL7.7mm	470uF ±20% 16V	JIEKR(捷而瑞)
15	贴片电容 1uF	2	C13, C25	CL10A105KB8NNNC		C_0603	1uF ±10% 50V	SAMSUNG(三星)
16	贴片电容 0.1uF	7	C1, C8, C9, C30, C31, C32, C33	CL10B104KB8NNNC		C_0603	100nF ±10% 50V	SAMSUNG(三星)
17	贴片电容 4.7uF	3	C3, C4, C10	CL10A475KO8NNNC		C_0603	4.7uF ±10% 16V	SAMSUNG(三星)
18	贴片电阻 5.1KΩ	1	R7	0805W8F5101T5E		R_0805	5.1kΩ ±1% 125mW	UNI-ROYAL(厚声)
19	贴片电阻 1KΩ	6	R8, R21, R23, R24, R26, R31	0805W8F1001T5E		R_0805	1kΩ ±1% 125mW	UNI-ROYAL(厚声)
20	贴片电阻 200Ω	1	R6	0805W8F2000T5E		R_0805	200Ω ±1% 125mW	UNI-ROYAL(厚声)
21	贴片电阻 22KΩ	2	R4, R5	0805W8F2202T5E		R_0805	22kΩ ±1% 125mW	UNI-ROYAL(厚声)
22	贴片电阻 10KΩ	5	R11, R14, R15, R18, R19	0805W8F1002T5E		R_0805	10kΩ ±1% 125mW	UNI-ROYAL(厚声)
23	贴片电阻 18KΩ	1	R10	0805W8F1802T5E		R_0805	18kΩ ±1% 125mW	UNI-ROYAL(厚声)
24	贴片电阻 500KΩ	2	R2, R3	AR03BTCX5003N		R_0603	500kΩ ±0.1% 100mW	Viking(光颉)
25	贴片电阻 330KΩ	1	R29	0805W8F3303T5E		R_0805	330kΩ ±1% 125mW	UNI-ROYAL(厚声)
26	贴片电阻 1MΩ	1	R13	0805W8F1004T5E		R_0805	1MΩ ±1% 125mW	UNI-ROYAL(厚声)
27	贴片LED	2	D1, D8	MHT19CGCT		D_0603	SMD，黄绿色，0603	MEIHUA(美华)
28	N沟道场效应管 (NMOS)	5	Q1, Q7, Q9, Q11, Q12	2N7002		SOT-23	N沟道,电流:115mA,耐压:60V	R+O(宏嘉诚)
29	P沟道场效应管 (PMOS)	6	Q2, Q3, Q4, Q6, Q8, Q10	SI2301		SOT-23	P沟道 耐压:20V 电流:2.8A	TWGMC(迪嘉)
30	NPN型三极管	2	Q5, Q13	S8050		SOT-23	hFE:200-350，丝印:J3Y S8050	R+O(宏嘉诚)
31	2.54mm排针1x4P	1	J5	PZ254V-11-04P		插件,P=2.54mm	1x6P 间距:2.54mm 方针 直插	XFCN(兴飞)
32	贴片电容 12pF	2	C17, C18	CL10C120JB8NNNC		C_0603	12pF ±5% 50V	SAMSUNG(三星)
33	贴片电容 2.2nF	1	C19	CL21C222JBFNNNE		C_0603	2.2nF ±5% 50V	SAMSUNG(三星)
34	贴片电容 1.5pF	1	C20	0603CG1R5C500NT		C_0603	±0.25pF 1.5pF 50V	FH(风华)
35	贴片电容 1pF	1	C21	0603CG1R0C500NT		C_0603	±0.25pF 1pF 50V	FH(风华)
36	贴片电容 1.2pF	1	C22	0603CG1R2C500NT		C_0603	±0.25pF 1.2pF 50V	FH(风华)
37	贴片电容 1nF	1	C12	0603B102K500NT		C_0603	1nF ±10% 50V	FH(风华)
38	贴片电容 100nF	1	C14	0603B104K500NT		C_0603	100nF ±10% 50V	FH(风华)
39	多层陶瓷电容器 100uF	1	C29	EMK325ABJ107MM-P		C_1210	100uF±20%,16V	TAIYO YUDEN(太诱)
40	贴片电容 22uF	3	C26, C27, C28	CGA0603X5R226M160JT		C_0603	22uF ±20% 16V	MRE(芯声)
41	贴片电容 47nF	1	C15	0603B473K500NT		C_0603	47nF ±10% 50V	FH(风华)
42	贴片电感 10uH	1	L3	CMI160808J100KT		L_0603	10uH ±20%	FH(风华)
43	贴片电感 15nH	1	L4	VHF160808H15NJT		L_0603	15nH ±5%	TAIYO YUDEN(太诱)
44	贴片电感 6.8nH	3	L5, L6, L7	FHW0603UC6N8JST		L_0603	6.8nH ±5%	FH(风华)
45	16MHz无源晶振	1	Y1	HY16MSMD3225FB1R30		SMD3225-4P	16MHz 12pF	Huiyuan crystal(惠源晶工)
46	nRF51822蓝牙传输芯片	1	U5	nRF51822-QFAA-R		QFN-48-EP(6x6)	多协议蓝牙4.0低功耗/2.4 GHz射频系统级芯片 (SoC)	NORDIC
47	贴片电阻 100KΩ	6	R9, R16, R17, R20, R25, R32	RS-03K1003FT		R_0603	100kΩ ±1% 100mW	FH(风华)
48	贴片电阻 100Ω	1	R30	RS-05K1000FT		R_0805	100Ω ±1% 125mW	FH(风华)
49	贴片电阻 10KΩ	5	R1, R11, R14, R15, R22	RS-05K1002FT		R_0805	10kΩ ±1% 125mW	FH(风华)
50	贴片电阻 10KΩ	3	R18, R19	RS-03K1002FT		R_0603	10kΩ ±1% 100mW	FH(风华)
51	贴片电阻 1mΩ	1	R27	PAD805FRL470R001L		R_0805	1mΩ ±1% 500mW	YAGEO(国巨)
52	开关二极管	2	D10, D11	CD4148WSF		D_0805	0805电阻式封装的4148，便于贴片机操作。	LIZ(丽智电子)
53	电压基准芯片	1	U6	TL431AIDR		SOIC-8	TL431 可调精密并联稳压器	TI(德州仪器)
54	单通道通用电压比较器	1	U7	LM397MFX/NOPB		SOT-23-5	LM397 单路通用电压比较器	TI(德州仪器)
55	充电管理芯片	1	U8	PW4053A		SOP-8-EP	5V充三串锂电池12.6V	PWCHIP(无锡平芯微)
56	锂电池保护芯片	1	U9	PW7126		SOP-8	三串4.2V锂电池保护板电路IC	PWCHIP(无锡平芯微)
57	2.5mm贴对板针座	1	J6	B4B-EH-A(LF)(SN)		插件,P=2.5mm	1x4P 间距:2.5mm 直插 系列:EH	JST
58	2.54mm排母2x20P	1	J1	2.54-2*20		插件,P=2.54mm	间距:2.54mm 2x20P 直插	ZHOURI(洲日)
59	无源蜂鸣器	1	BZ1	YS-MBZ12085C05R42		插件,D=12mm	电磁式无源分体蜂鸣器5V	Fengming(锋鸣电子)

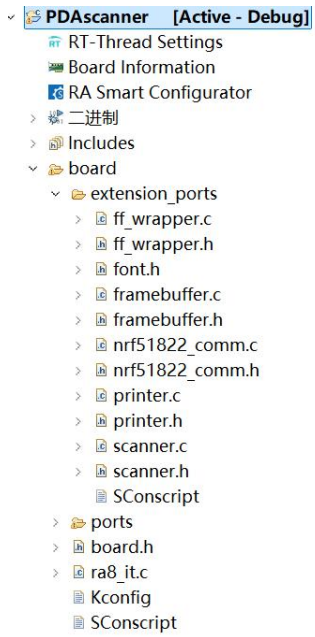
三、软件设计

软件设计采用 RT-Thread Studio 开发，实现了条码识别、图像显示、串口通信和热敏打印控制的功能，系统采用多线程架构，通过精密的资源管理和事件驱动机制实现各功能模块的协同工作。

具体程序详见文件“RTT-PDAscanner-program-V10.1.rar”。



1 扩展板驱动接口 BSP



1.1 热敏打印机驱动程序：printer.h、printer.c

该热敏打印机驱动控制采用通过四相步进电机精确控制进纸，每步进纸0.125mm，支持正反转和退纸操作，系统采用了模块化设计理念，通过精密的时序协同和智能参数调节，确保热敏打印的稳定性和可靠性。

```c
#define MNB_PIN          508    // P508 -> 电机B相负端
#define MB_PIN           509    // P509 -> 电机B相正端
#define MNA_PIN          510    // P510 -> 电机A相负端
#define MA_PIN           507    // P507 -> 电机A相正端
#define LAT_PIN          8      // P008 -> 数据锁存
#define DI_PIN           3      // P003 -> 串行数据输入
#define CLK_PIN          6      // P006 -> 时钟信号
#define STBA_PIN         9      // P009 -> 加热控制A
#define STBB_PIN         505    // P505 -> 加热控制B

// 引脚操作函数 - 使用RT-Thread PIN设备API
#define MNB(x)           rt_pin_write(MNB_PIN, x)
#define MB(x)            rt_pin_write(MB_PIN, x)
#define MNA(x)           rt_pin_write(MNA_PIN, x)
#define MA(x)            rt_pin_write(MA_PIN, x)
#define LAT(x)           rt_pin_write(LAT_PIN, x)
#define DI(x)            rt_pin_write(DI_PIN, x)
#define CLK(x)           rt_pin_write(CLK_PIN, x)
#define STBA(x)          rt_pin_write(STBA_PIN, x)
#define STBB(x)          rt_pin_write(STBB_PIN, x)

// 功能配置开关
#define IS_PRINT_PICT    1   // 是否打印黑白图片
#define IS_PRINT_ASCII   1   // 是否打印ASCII字符串
#define IS_ASCII12_EN    1   // 是否打印12号ASCII字符
#define IS_ASCII16_EN    1   // 是否打印16号ASCII字符
#define IS_ASCII24_EN    1   // 是否打印24号ASCII字符
```

```c
// 停止
#define MOTOR_STOP       do{MA(0);MNA(0);MB(0);MNB(0);}while(0)
// 停止加热
#define HEAT_STOP        do{STBA(0);STBB(0);}while(0)

// 全局变量声明
extern rt_uint16_t Ptdeepcolor;  // 打印颜色深度参数
extern rt_uint16_t Motorspeed;   // 步进电机转速参数
extern rt_uint16_t Leftdeep;     // 左半部分颜色深度参数

// 函数声明
void printer_init(void);
void printer_motor_run(rt_uint8_t cont, rt_uint8_t dir);
void printer_motor_back(rt_uint16_t dot);
void printer_set_deep(rt_uint8_t deepcolor);
void printer_set_lfdeep(rt_uint8_t leftdeepcolor);
void printer_stop(void);
void printer_one_line(rt_uint8_t *pixlindat);
void printer_lines(rt_uint8_t linesnum, rt_uint8_t dist);

#if IS_PRINT_PICT
void printer_picture(rt_uint8_t startx, rt_uint8_t *picdata, rt_uint16_t widthpix, rt_uint16_t heightpix);
void printer_img2lcd(rt_uint8_t x, rt_uint8_t *pic);
#endif

#if IS_PRINT_ASCII
void printer_string(rt_uint8_t *onestr, rt_uint8_t strnum, rt_uint8_t sizeh, rt_uint8_t sizew);
void printer_sstr(rt_uint8_t *mulstr, rt_uint8_t sizeh, rt_uint8_t sizew);
#endif
```

在硬件控制层面，驱动程序通过四相步进电机实现精确的进纸控制。电机驱动采用经典的四相八拍工作方式，每步进纸精度达到 0.125mm，支持正转进纸和反转退纸的操作。特别设计的退纸函数能够处理最大 65025 点的长距离回退，并通过“回退+前进”的补偿机制消除机械行程误差，确保定位的精度。

1.1.1 打印头区域独立控制方式

核心的打印机制建立在 384 点热敏打印头的基础上，打印头被划分为左右两个独立控制的 192 点区域。数据传输采用串行移位寄存器方式，通过 CLK 时钟线和 DI 数据线将 48 字节的点阵数据逐位送入打印头。每个字节传输过程中，驱动程序会实时统计需要加热的点数，为后续的智能温控提供依据。

```
    }
    MOTOR_STOP; // 停转
}

/*
步进电机退纸
输入参数：
    dot: 退纸点数,范围：[1,65025]
*/
void printer_motor_back(rt_uint16_t dot)
{
    rt_uint8_t mtim, ddot;

    if (dot > 255)
    {
        mtim = dot / 255;
        ddot = dot % 255;
        while (mtim--) printer_motor_run(255, 1);
    }
    else
    {
        ddot = dot;
    }

    printer_motor_run(ddot + 7, 1);
    printer_motor_run(7, 0); // 排除行程误差
}
```

```
entry.c  printer.c  printer.h
/*
设置打印颜色深度
输入参数：
    deepcolor: 打印深度，范围为0-100的整数
注：数值越大打印颜色越深，打印速度越慢，同时越容易因高温损坏打印模块！
*/
void printer_set_deep(rt_uint8_t deepcolor)
{
    if (deepcolor > 100) deepcolor = 100;
    Ptdeepcolor = deepcolor * 50;
    Motorspeed = (2 * (HT_minhttim + Ptdeepcolor) + Leftdeep);
    if (Motorspeed < 2000) Motorspeed = 2000;

    LOG_D("Set print depth: %d, Motor speed: %d", deepcolor, Motorspeed);
}

/*
设置左半部分打印颜色深度
*/
void printer_set_lfdeep(rt_uint8_t leftdeepcolor)
{
    Leftdeep = leftdeepcolor * 10;
    Motorspeed = (2 * (HT_minhttim + Ptdeepcolor) + Leftdeep);
    if (Motorspeed < 2000) Motorspeed = 2000;

    LOG_D("Set left depth: %d, Motor speed: %d", leftdeepcolor, Motorspeed);
}

/*
```

1.1.2 基于自适应加热控制算法

系统根据每行实际需要加热的点数动态计算加热时间并采用线性函数拟合（y=ax+b）。当点数少于 30 个时使用最小加热时间保证基础效果，超过 100 个点时启用最大加热时间防止过热，中间区间则按线性关系精确控制。左右分区独立加热的设计还允许对两侧进行不同的深度补偿，有效解决了因热分布不均导致的打印深浅不一的难题，参数调节机制提供了丰富的可配置选项。打印深度支持 0-100 级调节，深度值通过转换为时间补偿量影响实际加热时间，数值越大打印颜色越深但速度越慢。电机速度会根据加热时间自动调整，确保在加热完成后才执行进纸操作，这种自适应机制保证了系统的协同工作。

```
i
    rt_uint8_t i, k, cont;
    rt_uint16_t a[2];
    rt_uint8_t data[48];

    // 复制数据到本地数组
    rt_memcpy(data, pixlindat, 48);

    for (k = 0; k < 2; k++)
    {
        cont = 0;
        for (i = 0; i < 24; i++)
            cont += send_one_byte(data[k * 24 + i]); // 查待加热的点数

        if (cont < HT_mindots)
            a[k] = HT_minhttim; // 加热时间下限，保证每个点都能打印上
        else if (cont > HT_maxdots)
            a[k] = HT_maxhttim; // 加热时间上限
        else
            a[k] = cont * HT_kdottim + HT_bdottim; // 平均加热时间
    }

    LAT(1);
    pdelay_us(10);
    LAT(0); // 拉低锁存端
    pdelay_us(10);
    LAT(1);

    print_control(a[0], 1); //加热前半部分
```

```
打印虚线
输入参数：
    linesnum：打印几条虚线
    dist：线间距，单位：mm，最大值31
*/
void printer_lines(rt_uint8_t linesnum, rt_uint8_t dist)
{
    rt_uint8_t aa[48], i;

    // 创建虚线模式：0xFF, 0x00, 0xFF, 0x00, ...
    for (i = 0; i < 48;)
    {
        aa[i++] = 0xFF;
        aa[i++] = 0x00;
    }

    while (linesnum--)
    {
        i = 8 * dist;

        printer_one_line(aa);
        printer_motor_run(1, 0); // 进纸一步

        while (--i)
        {
            printer_motor_run(1, 0); // 继续进纸
        }
    }
    printer_stop();
```

1.1.3 打印过程中采取的安全保护措施

在安全保护方面，驱动程序设置了多重防护措施。包括加热时间上下限保护、电机转速下限保护、急停功能等。所有的加热操作结束后都会立即切断加热信号，电机停止时所有相线置零，有效防止了过热损坏和堵转现象。

1.2 扫描器驱动核心程序：scanner.h、scanner.c

采用模块化设计实现了完整的基于条形码识别功能框架。该系统通过多线程架构将图像采集、条码识别和结果回调解耦，提供了稳定可靠的扫描解决方案。程序创建了独立的图像处理线程“image_processing_thread”，该线程以 10fps 的速率持续处理摄像头数据。线程内部采用状态机设计，通过“barcode_detection_enabled”全局标志控制扫描启停，确保资源的高效利用。当检测功能启用时，系统会循环执行图像采集、条码识别和结果处理流程。

```c
#include <rtthread.h>
#include <rtdevice.h>
#include <stdio.h>
#include <string.h>
#include "scanner.h"

#define LOG_TAG "scanner"
#define LOG_LVL LOG_LVL_INFO
#include <rtdbg.h>

// 全局变量定义
static rt_bool_t barcode_detection_enabled = RT_FALSE;
static rt_thread_t barcode_thread = RT_NULL;
static barcode_callback_t barcode_callback = RT_NULL;

// 简化的条形码检测函数 - 使用模拟数据
static int detect_barcodes_simple(simple_image_t *img)
{
    RT_UNUSED(img); // 标记未使用参数

    int barcode_count = 0;

    /*****************************************/
    // 测试阶段将会使用模拟检测逻辑
    // 实际则需要调用图像处理算法API
```

```c
// 条形码识别主函数
void barcode_scanner_start(void)
{
    LOG_I("Starting Barcode Scanner Demo...");
    if (barcode_thread != RT_NULL && barcode_detection_enabled)
    {
        LOG_I("Barcode scanner is already running");
        return;
    }

    // 初始化摄像头
    if (camera_init_simple() != 0)
    {
        LOG_E("Camera initialization failed!");
        return;
    }

    // 创建图像处理线程
    barcode_thread = rt_thread_create("barcode_scanner", image_processing_thread, RT_NULL, 4096, RT_THREAD_PRIORITY_

    if (barcode_thread != RT_NULL)
    {
        barcode_detection_enabled = RT_TRUE;
        rt_thread_startup(barcode_thread);
    }
}
```

1.2.1 图像预处理流程

首先通过“get_camera_frame”函数获取 320x240 分辨率的 RGB565 格式图像数据，该函数目前使用模拟实现，为后续接入 VisionBoard 摄像头模块部分预留的 API 接口。接着调用“detect_barcodes_simple”函数进行条码识别，当前版本采用每 30 帧模拟检测一次的简化逻辑，实际应用中可替换为真实的图像识别算法。

1.2.2 系统回调机制

程序设有回调机制，通过“barcode_scanner_set_callback”函数注册用户自定义的回调函数，当识别到条码时会自动触发回调，将条码数据字符串传递给应用程序。这种异步通知模式确保了系统的实时响应能力，同时保持了架构的灵活性。除此之外，程序提供了完整的控制接口：

“barcode_scanner_start”用于初始化摄像头并启动扫描线程；

“barcode_scanner_stop”用于安全停止扫描并释放资源；

“barcode_scanner_trigger_test”支持手动触发测试，便于开发和调试。

```
    static int test_index = 0;
    const char *test_data = test_barcodes[test_index % 3];
    test_index++;

    LOG_I("Test barcode: %s", test_data);

    // 调用回调函数
    if (barcode_callback != RT_NULL)
    {
        barcode_callback(test_data);
    }
}

// 控制命令
void scanner_control(int argc, char **argv)
{
    if (argc == 2)
    {
        if (rt_strcmp(argv[1], "start") == 0)
        {
            if (!barcode_detection_enabled)
            {
                barcode_scanner_start();
            }
            else
            {
                LOG_I("Barcode scanner is already running");
            }
```

2 程序入口函数：hal_entry.c

主程序创建了三个核心线程：传感器线程负责以 25fps 的速率持续捕获 320x240 的 RGB565 图像数据并实时显示在 LCD 屏幕上；条码扫描线程在后台运行识别算法；打印线程作为工作线程按需创建，处理具体的打印任务。这种多线程设计通过互斥锁和信号量实现资源同步，确保图像采集、显示和处理的线程安全。

```
static struct rt_mutex lcd_mutex;
extern sensor_t sensor;
static rt_thread_t sensor_thread = RT_NULL;
uint8_t g_image_rgb565_sdram_buffer[CAM_WIDTH * CAM_HEIGHT * 2] BSP_PLACE_IN_SECTION(".sdram") BSP_ALIGN_VARIABLE(8)

// 串口接收回调函数
static rt_err_t uart_rx_callback(rt_device_t dev, rt_size_t size)

// 初始化串口
static rt_bool_t serial_init(void)

// 安全的串口写入函数
static void safe_serial_write(const char *data, rt_size_t size)

// 打印阶段主程序
static void printer_main(const char *barcode)

// 打印线程入口函数
static void print_thread_entry(void *parameter)

// 条形码检测回调函数
static void barcode_detected_callback(const char *barcode_data)

// 手动触发打印测试
static void test_print_command(int argc, char **argv)
MSH_CMD_EXPORT(test_print_command, test print function);

// 系统状态反馈
```

```
    LOG_I("Integrated system started successfully");
}

void hal_entry(void)
{
    LOG_I("====================================");
    LOG_I(" Hello! Welcome to use PDAscanner ");
    LOG_I("====================================");

    rt_thread_mdelay(2000);
    main();

    // 发送启动完成消息到串口
    if (serial_ready)
    {
        safe_serial_write("PDAscanner ready for scanning.\r\n", 31);
    }

    while (1)
    {
        rt_thread_mdelay(1000);
        static int heartbeat = 0;
        if (++heartbeat % 10 == 0)
        {
            LOG_D("System running...");
        }
    }
}
```

串口通信子系统是整个系统的控制枢纽。程序支持自动设备发现机制，依次尝试 uart2、uart6、uart1 等串口设备，uart7 则负责与 nrf51822 蓝牙芯

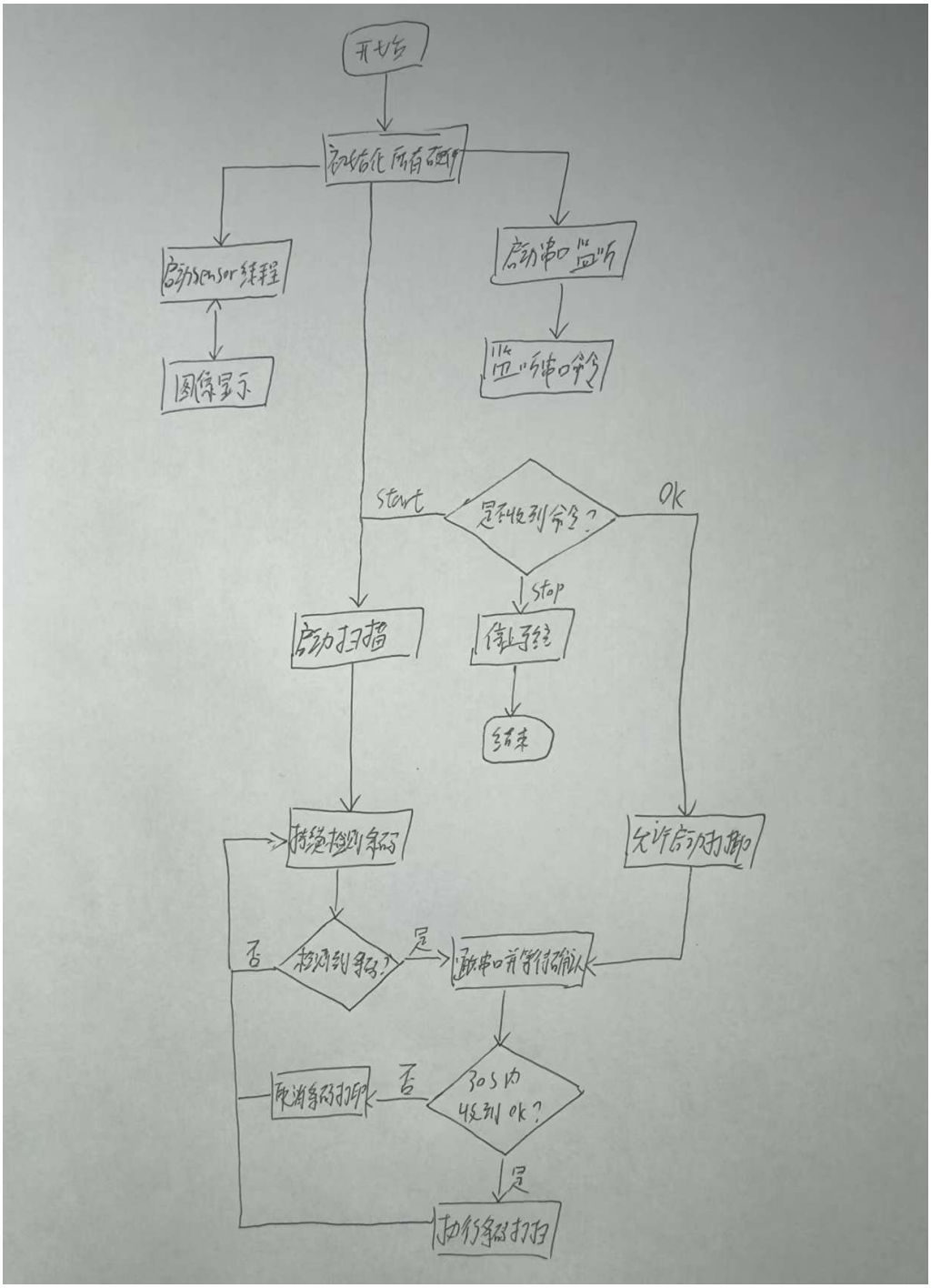
片进行通讯，实现数据交互传输，确保在不同硬件平台上的兼容性。串口配置为 115200 波特率，采用中断接收模式实时处理外部命令。系统能够识别"start"、"stop"、"ok"等控制指令，其中"ok"命令通过信号量机制触发打印操作，在调试过程中实现了安全的人机交互流程。

```
// 条形码检测回调函数
static void barcode_detected_callback(const char *barcode_data)
{
    if (barcode_data == RT_NULL) return;

    LOG_I("Barcode detected: %s", barcode_data);

    // 保存检测到的条形码
    rt_strncpy(detected_barcode, barcode_data, sizeof(detected_barcode) - 1);

    // 创建打印线程
    rt_thread_t print_thread = rt_thread_create("print_thread",
                                                print_thread_entry,
                                                (void *)detected_barcode,
                                                2048,
                                                RT_THREAD_PRIORITY_MAX / 3,
                                                20);

    if (print_thread != RT_NULL)
    {
        rt_thread_startup(print_thread);
    }
    else
    {
        LOG_E("Failed to create print thread");
    }
}
```

该程序业务逻辑所实现的功能是扫描器检测到有效条码时，会触发 barcode_detected_callback 回调函数，系统立即保存条码数据并创建独立的打印线程。打印流程设计为两步确认机制：首先通过串口通知用户检测到的条码内容，然后等待用户发送"ok"确认指令，30 秒超时保护防止系统阻塞。这种设计从而保证了操作的安全性。系统还集成了完善的调试和监控功能。通过 MSH 命令接口，用户可以手动测试打印功能、查看系统状态、枚举可用串口设备。实时心跳监控和状态日志为系统维护提供了便利。内存管理方面，图像缓冲区被精心放置在 SDRAM 区域，确保大数据量的图像处理不会导致内存瓶颈。

```
static void main(void)
{
    // 初始化互斥锁
    if (rt_mutex_init(&lcd_mutex, "lcd_mutex", RT_IPC_FLAG_FIFO) != RT_EOK)
    {
        LOG_E("Failed to create LCD mutex");
        return;
    }

    // 初始化传感器
    sensor_init();
    sensor_reset();
    sensor_set_pixformat(PIXFORMAT_RGB565);
    sensor_set_framesize(FRAMESIZE_QVGA);

    // 创建sensor线程-降低优先级
    sensor_thread = rt_thread_create("sensor_thread", sensor_thread_entry, RT_NULL, 4096, RT_THREAD_PRIORITY_MAX / 3
    if (sensor_thread != RT_NULL)
    {
        rt_thread_startup(sensor_thread);
    }

    LOG_I("Sensor ready, Starting integrated barcode scanner and printer system...");

    // 初始化串口
    if (!serial_init())
    {
        LOG_W("Serial initialization failed, continuing without serial...");
```

3 程序设计流程图


开始
初始化所有函数
启动Sensor线程
图像显示
启动串口监听
监听串口命令
Start
是否收到命令?
Ok
Stop
启动扫描
停止扫描
结束
持续检测条码
允许启动扫描
检测到条码?
否
是
跳出并等待确认
取消条码打印
否
3s内
收到ok?
是
执行条码打印

4 作品展示