# 工业物品检测平台源代码说明文档

目录

[工业物品检测平台源代码说明文档 1](#_Toc144737041)

[一、程序整体架构说明 1](#_Toc144737042)

[二、Application/User/Core分组文件说明 1](#_Toc144737043)

[1、main.c文件 1](#_Toc144737044)

[2、gpio.c文件 2](#_Toc144737045)

[3、dma.c文件 2](#_Toc144737046)

[4、spi.c文件 2](#_Toc144737047)

[5、stm32f4xx\_it.c文件 2](#_Toc144737048)

[6、stm32f4xx\_hal\_msp.c文件 3](#_Toc144737049)

[7、core.c文件 3](#_Toc144737050)

[8、usart.c文件 3](#_Toc144737051)

[9、tim.c文件 4](#_Toc144737052)

[10、adc.c文件 4](#_Toc144737053)

[二、HardWare分组文件说明 4](#_Toc144737054)

[1、led.c文件 4](#_Toc144737055)

[2、beep.c文件 5](#_Toc144737056)

[3、lcd.c文件 5](#_Toc144737057)

[4、touch\_key.c文件 6](#_Toc144737058)

[5、sys\_font.c文件 6](#_Toc144737059)

[6、vision.c文件 6](#_Toc144737060)

[7、voice.c文件 7](#_Toc144737061)

[8、diwen\_lcd.c文件 7](#_Toc144737062)

[9、infared.c文件 8](#_Toc144737063)

[10、matrix\_key.c文件 8](#_Toc144737064)

[11、tripod.c文件 8](#_Toc144737065)

[12、poewr.c文件 9](#_Toc144737066)

[13、relay.c文件 9](#_Toc144737067)

[三、StateMachine分组文件说明 9](#_Toc144737068)

[1、run\_fast.c文件 9](#_Toc144737069)

[2、run\_slow.c文件 10](#_Toc144737070)

[3、run\_manual.c文件 11](#_Toc144737071)

[4、run\_static.c文件 11](#_Toc144737072)

[5、run\_stop.c文件 11](#_Toc144737073)

[6、run\_mode.c文件 11](#_Toc144737074)

[7、logic.c文件 11](#_Toc144737075)

[8、osci.c文件 12](#_Toc144737076)

## 一、程序整体架构说明

程序使用keil MDKV5.31.0.0编写，编译器使用default compiler version 5。当使用keil打开该工程的时候，可以看到源码整体架构如下图1所示。

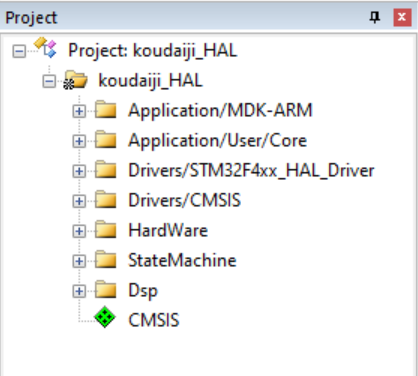


图1：源码整体架构

其中Application/MDK\_ARM分组里放的是startup\_stm32f407xx.s即stm32的启动文件；Application/User/Core里面放的是关于芯片片上资源和内核的一些初始化文件；Drivers/STM32F4xx\_HAL\_Driver里面放的是stm32的HAL库的库文件；Drivers/CMSIS里面放的是stm32的系统初始化文件；HardWare里面放的是关于整个平台的一些硬件部分如LCD、语音板等相关的代码；StateMachine里放的是关于系统状态机的代码；Dsp里面放的是stm32数字信号处理库的库文件。

关于stm32的启动文件、系统初始化文件、HAL库库文件、数字信号处理库库文件均是移植的意法半导体官方的库文件，这里不过多描述。下面重点讲解一下关于Application/User/Core、HardWare、StateMachine这三个文件里面的源码。

## 二、Application/User/Core分组文件说明

### 1、main.c文件

main.c是程序主函数入口所在的文件，程序运行到main()文件里之后，首先会初始化HAL库、系统时钟、必要的GPIO引脚、口袋机LED指示灯、口袋机蜂鸣器、口袋机触摸按键，然后会检测口袋机上按键C是否被按下，如果检测到按键C一直被按下就会执行ADC采样和波形分析显示功能，如果按键C没有被按下则继续初始化运行物品检测功能所需要的一些片上资源和硬件然后执行物品检测的相关逻辑。在运行物品检测逻辑之前需要先对云台进行校准，*while(!manual\_claib\_finish)*这是一个while循环，在此循环中会运行关于云台和镜头等手动校准的逻辑，知道按下口袋机的“右”键时才会退出循环进入主循环。

主循环即while(1)循环中所处理的逻辑如下，首先不断循环处理矩阵按键和口袋机触摸按键的按键执行逻辑，然会主循环里有一个*main\_1ms\_flag*标志位该标志位在*SysTick\_Handler()*中断里1ms翻转一次，然后主循环里有一部分在*if(main\_1ms\_flag == true)*条件的控制下运行即1ms运行一次，需要1ms时序运行的程序有前台LED指示灯闪烁的控制逻辑和物品指示的逻辑，因为这两部分程序需要比较准确的时序控制，所以控制1ms运行一次。（能够在*while(1)*里面实现1ms时序控制的原理是*while(1)*运行的非常快，运行一遍的时间远远小于1ms）。之后是控制1s一次向触摸屏发送当前运行时间计数，*run\_time*变量在*SysTick\_Handler()*中断里面1ms自加一次，所以*if(run\_time % 1000 == 0)*条件正好1s满足一次，当条件满足时向触摸屏发送一次时间，之后有一个*run\_time++;*是因为while(1)循环太快，避免*if(run\_time % 1000 == 0)*条件满足多次造成频繁发送。最后是运行触摸屏的翻页逻辑。

### 2、gpio.c文件

gpio.c文件里面主要是使能一些必要端口（如晶振接口、SWD接口等）的时钟，否则晶振接口等无法使用。

### 3、dma.c文件

dma.c里面主要是使能DMA2的时钟，然后设置DMA2必要通道的中断优先级并打开中断。其中，DMA2的通道0连接到ADC1；DMA2的通道2连接到UART1的RX；DMA2的通道3连接到SPI1的TX；DMA2的通道7连接到UART1的TX。

### 4、spi.c文件

spi1用来控制口袋机上的lcd屏幕，spi1初始化为8位数据，速率42M/S，数据高位在前，软件控制NSS，同时将spi1通过dma2发送数据。spi1使用的引脚位：SPI1\_SCK->PB3, SPI1\_MISO->PB4, SPI1\_MOSI->PB5。

### 5、stm32f4xx\_it.c文件

stm32f4xx\_it.c文件里包含了stm32所有中断的中断回调函数，其中使用到的中断有如下几个：

1、*void SysTick\_Handler(void)；*是系统systick中断，每1ms中断1次，在sysytick中断里面主要执行的程序有触摸按键扫描、云台电子限位信号检测、矩阵按键扫描、语音模块忙检测、后台指示灯闪烁控制。

2、*void TIM2\_IRQHandler(void)；*是定时器2的中断回调函数，定时器2每100us中断一次，定时器2中断里面主要是翻转云台步进电机驱动板X轴STEP引脚的电平以产生方波来驱动云台的X轴步进电机。

3、*void TIM4\_IRQHandler(void)；*是定时器4的中断回调函数，定时器4每100us中断一次，定时器4中断里面主要是翻转云台步进电机驱动板Y轴STEP引脚的电平以产生方波来驱动云台的Y轴步进电机。

4、*void MX\_TIM3\_Init(void)；*是定时器3的中断回调函数，定时器3每26us中断一次（38KHz），定时器3中断的频率和红外遥控的NEC协议载波频率一致，用来模拟电视机遥控器的按键信号来控制电视机的播放和暂停。

5、*void USART1\_IRQHandler(void)；*是uart1的中断入口函数，串口1用来和AI识别模组通信，AI模组发送过来的数据是不定长的数据包。串口1使用DMA接收，并且打开了串口空闲中断，当进入串口1中断函数之后，会先判断空闲中断的标志位，判断是否是空闲中断，如果是空闲中断则先清除中断标志位，由于是DMA接收所以当触发空闲中断的时候数据已经通过DMA自动传输到内存指定地址，然后标志数据帧接收完成可以取出并处理数据，然后重新打开DMA接收。

6、*void USART5\_IRQHandler(void)；*是串口5的中断入口函数，串口5用来和触摸屏通信，串口5打开了数据接收中断和空闲中断，同样进入之后先判断空闲中断的标志位，如果是空闲中断同样先清除中断标志位然后标记数据帧接收完成，然后再重新打开数据接收中断。串口5没有使用DMA接收，而是使用中断接收。

7、*void HAL\_UART\_RxCpltCallback(UART\_HandleTypeDef \*huart)；*是串口数据接收中断，串口5打开了数据接收中断，所以串口5每收到1位数据就会触发一次该中断，在该中断中，首先判断如果是串口5，即将数据接收地址在内存中后移一位。

8、*void HAL\_ADC\_ConvCpltCallback(ADC\_HandleTypeDef\* hadc)；*是ADC转换完成中断，ADC1是用来采集排故板输出的电压波形，为了保证采样速率和做傅里叶变换时的准确度，ADC1打开了DMA接收并且一次转换1024个电压值，当1024个采样采集并且DMA传输完成之后会触发该中断，在中断里置位ADC转换完成标志。

### 6、stm32f4xx\_hal\_msp.c文件

该文件里主要是打开stm32单片机内核的一些电源。

### 7、core.c文件

该文件里*void SystemClock\_Config(void)*函数的功能是初始化单片机的系统时钟，将系统总线时钟初始化位168MHz，APB1总线时钟为84MHz，APB2总线时钟为168MHz。*void Error\_Handler(void)*函数是硬件错误中断回调函数，不做处理。*void Soft\_Delay\_Us(uint16\_t time)*函数是自定义的us级软件延时函数。

### 8、usart.c文件

该文件里是stm32单片机上一些串口的初始化，串口4连接语音板，通过串口控制语音播放，串口4初始化为波特率9600，8位数据位，1位停止位，无校验位。串口5连接触摸屏，用来向触摸屏发送显示数据和接收触摸屏的按键数据，初始化为波特率115200，8位数据位，1位停止位，无校验位，并且打开了数据接收中断和空闲中断来接收触摸屏的按键数据。串口1连接AI识别模组，用来控制AI模组识别物体的类型和颜色和接收AI模组识别到的物体的坐标信息，初始化为波特率115200，8位数据位，1位停止位，无校验位，AI模组传过来的物体坐标信息是不定长的数据帧而且传输速率较高所以串口1打开了DMA接收和空闲中断。以上串口使用的引脚信息如下：UART4\_TX->PC10；UART4\_RX->PC11；UART5\_TX->PC12；UART5\_RX->PD2；UART1\_TX->PA9；UART1\_RX->PA10。

### 9、tim.c文件

time.c文件主要用来初始化用到的定时器，定时器2初始化为100us周期并打开中断，用来产生方波信号驱动步进电机，定时器4初始化为100us周期并打开中断，用来产生方波信号驱动步进电机，定时器3初始化为26us周期（38KHz）并打开中断，用来模拟遥控器红外协议的38KHz载波信号，定时器8更新频率为800KHz，未打开定时器中断，定时器溢出时会关联溢出事件触发ADC信号采集，用于故障排除板的电压采集和波形分析。

### 10、adc.c文件

adc.c文件用于初始化adc1采集故障排除板的电压波形，由于故障排除板的电压频率可能比较高，所以dac采集需要的采集速率也要够快，通过定时器触发adc采样加上采样数据通过dma传输，最终将adc1的采样速率设置为800KHz。ADC1的的通道1通过PA1引脚进行电压采集。

## 二、HardWare分组文件说明

### 1、led.c文件

Led.c文件是关于口袋机上8个led灯的初始化和操作相关的程序。其中，在gpio.h里面定义了一个关于端口的结构体，结构体成员里有端口组和端口号，后续对各个外设所需要的端口的定义都可以使用该结构体。

*typedef struct \_GPIO*

*{*

*GPIO\_TypeDef \*GPIOx;*

*uint16\_t GPIO\_Pin;*

*} GPIO\_Struct, \*GPIO\_Struct\_P;*

在led.c里面使用该结构体定义了8个led灯的实例，*GPIO\_Struct LED0, LED1, LED2, LED3, LED4, LED5, LED6, LED7;*

*void Led\_Init(void)* 函数是led灯所用GPIO口的初始化，初始化中先对这8个led灯的结构体实例赋值，使其对应上stm32的实际引脚，然后将这些引脚初始化为推挽输出、无上下拉模式。

在led.h文件里定义了一个枚举变量用来描述led灯的状态即开和关

*typedef enum*

*{*

*LED\_ON = 0,*

*LED\_OFF,*

*} LED\_State;*

*void Led\_Control(GPIO\_Struct\_P led, LED\_State state)* 函数是关于led灯操作的代码，其第一个参数是要控制的led灯的实例，第二个参数就是枚举的led灯的状态。

### 2、beep.c文件

beep.c文件是口袋机内蜂鸣器到的初始化和操作相关的代码，口袋机的蜂鸣器是有源蜂鸣器，蜂鸣器的操作和led灯相同，只是蜂鸣器的控制引脚需要初始化为推挽输出、下拉电阻使能模式。

### 3、lcd.c文件

lcd.c文件是口袋机上lcd屏幕初始化和操作相关的代码，lcd屏幕使用spi1接口和单片机连接，此外lcd还有两个额外的DC和CS引脚，这两个引脚的初始化方法同样参照led灯的初始化。

*void Lcd\_Pin\_Init(void)*是lcd的控制引脚即DC和CS引脚的初始化函数。其中整个文件中外部可能调用的函数及其功能如下。

*void Lcd\_Clear(uint16\_t color)*是lcd清屏函数，参数是清屏的颜色值。颜色值是RGB565格式。

*void Lcd\_Draw\_Point(uint16\_t x, uint16\_t y, uint16\_t color)*是在指定位置上画点，第一个参数是x坐标，第二个参数是y坐标，当口袋机整放时屏幕左上角为原点，第三个参数是颜色值。

*void Lcd\_Init(void)*是初始化lcd，设置lcd的显示方向，工作状态等。

*void Lcd\_ShowChar(uint16\_t x, uint16\_t y, uint8\_t num, Show\_Size\_Enum size, uint16\_t Color)*是在lcd上显示一个字符，前两个参数是显示的位置坐标，第三个参数是字体的大小，同样是一个枚举变量，有*SHOW\_SIZE\_16 、SHOW\_SIZE\_32*两种字体，分别是1608和3216字体，最后一个参数是显示的颜色。

*void Lcd\_ShowString(uint16\_t x, uint16\_t y, char \*p, Show\_Size\_Enum size, uint16\_t Color)*是在指定坐标上显示一个字符串，同样大小和颜色可以自定义。

void Lcd\_printf(char \*fmt, ...)函数是在lcd屏幕上打印字符串，用法和C库的printf相同，每调用一次，打印的字符串自动向下换一行，打印满14行之后清屏自动再从第一行开始。

*void Lcd\_Clear\_Osci(uint16\_t color)*函数是在绘制故障板的电压波形时的清屏函数，保留了顶端的一部分。

*void Lcd\_Osci\_printf(char \*fmt, ...)*函数是在绘制电压波形的时候使用的打印函数，没有自动增行功能，只在最顶端一行打印。

### 4、touch\_key.c文件

touch\_key.c是口袋机上触摸按键的按键扫描函数，同样通过*GPIO\_Struct*结构体定义了每个按键的实例，void Touch\_Key\_Init(void)是按键初始化函数，初始化按键接口为输入上拉模式。*Touch\_Key\_Scan()*函数是触摸按键扫描函数，该函数放在systick中断里面，每1ms扫描一次，*Touch\_Key\_Value*是触摸按键的键值，当触摸按键按下的时候，键值会被赋值为该按键的键值，键值是一个枚举变量，枚举了每个按键（口袋机上的触摸按键一共有“上”、“下”、“左”、“右”、“A”、“B”、“C”、“D” 8个按键）。

*typedef enum*

*{*

*TOUCH\_KEY\_NONE = 0,*

*TOUCH\_KEY\_UP,*

*TOUCH\_KEY\_DOWN,*

*TOUCH\_KEY\_LEFT,*

*TOUCH\_KEY\_RIGHT,*

*TOUCH\_KEY\_A,*

*TOUCH\_KEY\_B,*

*TOUCH\_KEY\_C,*

*TOUCH\_KEY\_D,*

*}TOUCH\_KEY\_Enum;*

### 5、sys\_font.c文件

sys\_font.c是ASCII码字库，里面包含了1608和3216两种字体。

### 6、vision.c文件

vision.c是关于和AI识别模组通信的代码，通过串口1和AI模组通信，串口1已经打开了DMA收发。*void Vision\_Send\_Cmd(uint8\_t color, uint8\_t shape)*函数是向AI识别模组发送命令告诉AI识别模组当前要检测的物品的颜色和形状，第一个参数是颜色标号，第二个参数是形状标号，其中标号和实际物体的对应关系需要根据模型训练时的标签而定。目前我们提供的训练模型中，颜色标号从1-4分别是红、绿、蓝、黄。形状标号从1-5分别是正方形、圆形、三角形、圆环、四边形。

*uint8\_t Vision\_Dat\_Explan(void)*函数是解析AI模组发过来的数据包，如果需要接收AI模组的识别信息就需要不断地循环调用该函数，该函数的返回值代表AI模组识别到的指定物体的数量，如果该函数返回0则证明没有识别到指定物体。在.h文件中定义了一个关于识别到的物体信息的结构体

*typedef struct vision\_coord\_str*

*{*

*Res\_Coord\_Struct Res\_Coord[10];*

*uint8\_t Res\_Num;*

*}Vision\_Coord\_Struct, \*Vision\_Coord\_Struct\_p;*

其中*Res\_Num*代表识别到的物品的数量，*Res\_Coord[10];里面存放的是物体的坐标，同样Res\_Coord[10]*是一个*Res\_Coord\_Struct*结构体的数组，该结构体只有x和y坐标两个成员变量。当需要获取AI模组的识别结果时，就需要不断的调用*Vision\_Dat\_Explan()*函数如果该函数的返回结果不是0就在*Vision\_Coord*变量里取出物体的数量，并根据物体的数量从*Vision\_Coord.Res\_Coord[10]*里面挨个取出物体的坐标。

*static void Res\_Sort(Vision\_Coord\_Struct\_p wait\_sort)*函数是对检测到物体进行从左到右从上到下的顺序排序。当调用*Vision\_Dat\_Explan()*函数发现物品数量大于1的时候就需要排序，该函数的参数是一个*Vision\_Coord\_Struct\_p*类型的结构体指针，使用时只需要将*Vision\_Coord*结构体变量取地址传入就会对该结构体里面的物体坐标进行排序。

*static void Res\_Sort2(Vision\_Coord\_Struct\_p wait\_sort)*同样是排序，用法和上述相同，只是该函数是对物体进行从下到上从左到右的排序。第一种排序方法适用于视频中物体从右向左流动的时候，第二种排序方法适用于视频中物体从上向下流动的时候。

### 7、voice.c文件

voice.c是和语音板通信的程序，单片机通过串口4向语音板发送命令，*void Voice\_Init(void)*函数是初始化语音板的忙检测接口，该引脚被初始化为输入下拉模式，语音板对外提供忙检测功能，当播放语音时端口为高电平，播放完毕时端口恢复低电平。*void Voice\_Play(uint8\_t voice)*是控制语音文件播放的函数，参数就是要播放的语音文件的序号。*void Voice\_Busy\_Scan(void)*是语音板忙检测信号的端口电平扫描函数，当检测到语音正在播放时*voice\_is\_busy*变量会被置true。

### 8、diwen\_lcd.c文件

diwen\_lcd.c文件是关于机箱上触摸屏的程序。单片机通过串口5和触摸屏通信，触摸屏的通信接口是485电平，所以中间需要485板将单片机的TTL电平转成485电平。*void MAX485\_Init(void)*函数是初始化MAX485芯片的控制引脚，TTL转485使用的是MAX485芯片，该芯片是半双工工作模式，需要由一个控制引脚来控制接收和发送。

*void DiWenLcd\_Printf(const char \*fmt, ...)*函数是向触摸屏发送显示信息，用法和C库的printf函数一样，发送的字符串信息显示在屏幕的调试输出窗口并自动向下排列显示，当一屏显示满之后会自动向上滚动显示，并且程序会记录下现实的信息（最多记录512条），可以通过触摸屏上面的上下翻页按键查看历史数据。

*void DiWenLcd\_Init(void)*函数是初始化触摸屏，主要是清空显示区域。

*void DiWenLcd\_Cmd\_Data(uint16\_t addr, uint16\_t data)*函数是向触摸屏固定地址发送显示的数据，主要是显示运行时间，检测到的物品的数量等，不同的显示区域有不同的地址。

*void DiWenLcd\_Show\_RunState(uint16\_t addr, const char \*fmt, ...)*函数是控制触摸屏显示当前的运行模式，如“停止检测”、“静态模式”等。

*void DiWenLcd\_Show\_Page(uint8\_t page)*函数是控制触摸屏切页，初始上电时触摸屏显示欢迎页，需要调用此函数切到第二页即数据显示界面，参数传2就是切第二页。

*uint8\_t DiWen\_Dat\_Explan(void)*函数是解析触摸屏传回的数据（触摸屏的上下翻页按钮点击时会传回数据）调用该函数，返回0是未解析到数据，返回1是触摸屏按下了“上一页”按钮，返回2是触摸屏按下了“下一页”按钮。

### 9、infared.c文件

Infared.c是关于红外协议发送的源码，虽然红外的NEC协议比较复杂，但是在主程序中需要调用的部分却比较简单，*void Infared\_Init(void)*该函数是初始化红外的引脚，该引脚通过一个三极管开关电路驱动红外发射管，将该引脚初始化为推挽输出下拉电阻模式。

*void Infared\_Send\_Data(uint8\_t user1, uint8\_t user2, uint8\_t data)*该函数是通过红外发送一帧指定的数据，一帧数据包含四位，*user1*和*user2*是两位用户码，*data*是要发送的数据，后面还要跟一个数据反码，该函数会自动加上。*void Infared\_Send\_OK(void)*该函数的功能是模拟遥控器的“OK”键键码，电视机在播放视频时调用该函数会暂停，暂停时调用该函数会再次播放。

### 10、matrix\_key.c文件

Matrix\_key.c文件是矩阵按键扫描函数，和touch\_key.c的逻辑基本相同。在进行矩阵按键扫描的时候，将按键的端口模式改为了开漏输出、上拉电阻模式，在开漏上拉模式下stm32的端口可以实现准双向口的功能。*Matrix\_Key\_Value*同样是按键的键值，这是一个枚举变量，枚举了矩阵按键的所有按键情况，*void Matrix\_Key\_Scan(void)*是矩阵按键扫描函数，该函数在systick中断里每1ms扫描一次，然后在主函数里不断判断*Matrix\_Key\_Value*的值就可以检测按键是否被按下，当检测到*Matrix\_Key\_Value*的值不是*MATRIX\_KEY\_NONE*的时候，证明有按键按下，根据*MATRIX\_KEY\_NONE*的值判断具体是哪个按键然后执行对应的操作，操作完成之后再将键值置为*MATRIX\_KEY\_NONE*。触摸按键键值的使用也是需要在用完之后清零。

### 11、tripod.c文件

Tripod.c文件是关于云台驱动的，主要就是驱动两个步进电机。步进电机驱动板上面有四个信号接口，DIR是控制电机转动的方向的，EN是使能引脚，可以理解为驱动板的开关，STEP引脚是脉冲输入引脚，每输入一个脉冲步进电机转动一步，除此之外还有一个限位信号读取引脚，云台的两轴上都有一个电子限位检测，是一个光电开关，当云台的一个轴转到一定的角度之后就会触发该限位信号，用于云台上电时的校准和机械归零。*void Tripod\_Init(void)*是初始化和驱动板相连接的单片机引脚。*void Tripod\_X\_Dir(uint8\_t dir)*是设置X轴的运动方向，参数传入0是向左转，传入1是向右转，同样还有一个设置Y轴的运动方向，0是向上转，1是向下转。*void Tripod\_X\_Step(uint16\_t num)*是设置X轴的步进步数，传入的参数num就是步进步数。*void Tripod\_Key\_Scan(void)*是电子限位信号检测，主要就是读取电子限位检测接口的电平然后加入一些消抖。*void Tripod\_X\_Go\_Mechzero(void)*是X轴机械轴归零，上电自检的时候就是使用该函数归零。*void Tripod\_X\_Angle(float x\_dat)*是使X轴转到目标角度，传入的参数就是目标的角度，单位为°。void Tripod\_X\_Coord(int x\_dat)是使X轴转到目标坐标，坐标的划分是以屏幕上的一个像素点为一个单位，电视左上角为原点，电视机的像素分辨率是1366\*768。

### 12、poewr.c文件

Power.c是控制电源开关的源码，现在使用的电源取消了开关，这部分不需要了解。

### 13、relay.c文件

Relay.c是控制继电器的，继电器通过通断控制激光笔。继电器的控制就是通过一个IO口驱动一个三极管开关电路，这部分的逻辑和控制蜂鸣器相同。

## 三、StateMachine分组文件说明

StateMachine分组里是上层逻辑部分的代码，其中物品识别指示部分的整体架构使用状态机驱动，不同的工作模式（如静态模式、慢速模式等）是不同的状态，切换模式按键的按下是事件，通过事件驱动不同的状态切换，整体构成一个事件驱动的状态机。

### 1、run\_fast.c文件

Run\_fast.c是快速模式的运行逻辑源码，在.h文件中定义了一个关于该模式的结构体：

*typedef struct Run\_Fast\_Str //快速模式运行相关的结构体*

*{*

*uint8\_t step; //运行的步数标记*

*Res\_Coord\_Struct res\_coord[10]; //AI模组识别到的物体坐标*

*uint8\_t res\_num; //AI模组识别到的物体的数量*

*bool first\_flag; //是否是首次检测标志*

*Res\_Coord\_Struct last\_res; //记录当前页面最后一个物体的坐标*

*uint16\_t res\_sum; //当前模式下识别物体总数*

*uint8\_t need\_point\_num;*

*uint16\_t user\_num1;*

*uint16\_t user\_num2;*

*uint8\_t i;*

*bool new\_flag;*

*} Run\_Fast\_Struct;*

在该模式的运行逻辑里同样使用了一个小型的状态机，该状态机不是事件驱动的，结构体成员变量step就是控制不同状态的，程序根据当前的状态和一些条件控制step变量的值，然后根据step的值切换运行不同状态下的代码。*void Run\_Mode\_Fast\_Init(void)*是快速模式的初始化函数，通过该函数可以使快速模式完成运行前的所有准备工作。快速模式和其他模式相同，都有四个比较重要的函数：初始化函数、模式进入函数、模式运行函数、模式退出函数。在快速模式初始化函数一开始有三行赋值代码：

*Run\_Mode\_Fast.mode\_enter = Run\_Mode\_Fast\_Enter;*

*Run\_Mode\_Fast.mode\_exit = Run\_Mode\_Fast\_Exit;*

*Run\_Mode\_Fast.mode\_run = Run\_Mode\_Fast\_Run;*

*Run\_Mode\_Fast*是一个*Mode\_Struct*类型的结构体变量，*Mode\_Struct*在run\_mode.h头文件中

typedef struct \_Mode\_Struct

{

void (\*mode\_enter)(void);

void (\*mode\_run)(void);

void (\*mode\_exit)(void);

}Mode\_Struct;

该结构体里面的三个成员变量是三个结构体指针，这三个指针需要分别指向不同模式的进入函数、运行函数、退出函数。然后通过该结构体类型实例化了四个运行模式的结构体变量

*Mode\_Struct Run\_Mode\_Stop, Run\_Mode\_Static, Run\_Mode\_Slow, Run\_Mode\_Fast, Run\_Mode\_Manual;* 分别是停止模式、静态模式、慢速模式、快速模式。

然后快速模式初始化函数一开始的三行赋值就是给快速模式结构体里面的三个函数指针赋值，让这三个函数指针指向快速模式下模式进入、模式运行、模式停止这三个函数的实现。

然后初始化函数后面是清除一些快速模式运行时需要用到的变量和标志位。

*void Run\_Mode\_Fast\_Enter(void)*函数是快速模式的进入函数，在从其他状态进入快速模式时需要先调用该函数，该函数里面同样是清空一些变量和标志位然后控制触摸屏的状态显示和语音播报。

*void Run\_Mode\_Fast\_Exit(void)*函数是快速模式的退出函数，该函数里面只有一个操作关闭激光笔，在退出快速模式的时候需要调用该函数一次。

*void Run\_Mode\_Fast\_Run(void)*函数是快速模式的运行函数，该函数第一行就是判断*if(Run\_Fast.first\_flag == true)*，是为了判断是否是切换到快速模式之后首次运行该函数，因为首次运行该函数时需要指示屏幕上识别到的所有物体，而不是首次运行时则只需要指示新出现的物体即可，所以需要两个运行逻辑。具体逻辑的实现可以参照代码讲解视频。

### 2、run\_slow.c文件

慢速模式的实现逻辑，和快速模式逻辑一样。

### 3、run\_manual.c文件

Run\_manual.c是手动模式运行逻辑的实现函数，其代码结构和快速模式一样，手动模式有两种，一种是坐标模式，就是根据AI识别软件上识别出的物体的坐标通过键盘输入，激光笔就会指向该坐标（坐标的输入根据语音提示分别输入X和Y然后按键D就可输入成功），另一种模式通过口袋机上的触摸按键“D”切换，口袋机的lcd屏会提示“dist mode”此时就是距离模式，输入的坐标是相对于屏幕左上角偏移的距离，单位是厘米。在手动模式下，矩阵按键上的“上”、“下”、“左”、“右”四个按键可以微调激光笔的位置。具体的代码实现逻辑可以参考视频讲解。

### 4、run\_static.c文件

Run\_static.c是静态模式的代码，整体的代码架构和快速模式一样，具体的实现逻辑参照视频讲解。

### 5、run\_stop.c文件

Run\_stop.c是停止状态下的运行代码，停止状态的代码架构和快速模式一样，由于停止状态下不需要进行任何操作，所以停止状态下的运行逻辑为一个空函数。

### 6、run\_mode.c文件

Run\_mode.c里主要就是一个*void Run\_Mode\_Init(void)*函数该函数可以理解为整个状态机的初始化，该函数里面也是调用了以上五种状态的初始化函数。

### 7、logic.c文件

Logic.c是实现所有的按键逻辑，*void Logic\_Matrix\_Key(void)*是实现矩阵按键的模式切换功能，例如“F1”按键按下从停止状态进入静态模式时通过函数指针调用了两个函数，首先*Run\_Mode\_Now->mode\_exit();* 调用了当前状态的退出函数即停止状态的退出函数，然后*Run\_Mode\_Now = &Run\_Mode\_Static;* 重新给*Run\_Mode\_Now*指针赋值，让*Run\_Mode\_Now*指针指向*Run\_Mode\_Static*的结构体变量，然后再调用*Run\_Mode\_Now->mode\_enter();* 此时运行的就是静态模式的进入函数了。

*void Logic\_Touch\_Key(void)*是触摸按键的运行逻辑，主要是触摸按键的“A”切换检测的颜色，“B”切换检测的形状，“D”切换手动模式下输入的是坐标还是厘米。

*void Logic\_Tripod\_Calib(void)*是通过触摸按键进行云台校准的相关程序，在云台自检完成之后需要手动校准云台和摄像头，此时按触摸按键的“左”会进行横向位置校准，观察激光笔是否能够照射到电视机屏幕的左右边缘上，然后观察激光笔照射到屏幕中间的坐标值，按“E”键开始输入坐标，按“D”键确认。之后按键“上”进行竖直方向的校准，同样观察激光笔有没有照射到电视机上下边框。然后在进行AI识别软件坐标边界的校准，具体的校准方法比较复杂，可以观看操作视频里面的讲解。

### 8、osci.c文件

osci.c是模拟示波器功能的代码，模拟示波器功能主要是显示波形图像，然后显示波形的频率、最大值、最小值。通过定时器8触发ADC采集，采集1024个数据之后对采集的数据进行一次快速傅里叶变换，通过对变换之后的数组检索求出信号的频率，具体的逻辑实现可以参考代码注释和源代码视频讲解。