

[前序](#)

[1 注册GizWits账号](#)

[2 定义"微信宠物屋"产品](#)

[3 生成微信宠物屋代码](#)

[4 移植微信宠物屋驱动代码](#)

[5 编译烧录固件并进行测试](#)

[6 相关说明](#)

前序

机智云提供了官方 微信宠物屋 测试固件以及相应平台的驱动库文件，开发者可以直接烧录并使用机智云APP进行测试，烧录方式请查看本文档的 编译烧录固件并进行测试 一节。

同时机智云也为开发者提供了SOC版与MCU版的代码自动生成功能，开发者可根据此文档进行自主开发，完成属于自己的 微信宠物屋 。

1 注册GizWits账号

首先登陆[机智云官网](#)，注册开发者账号。

2 定义"微信宠物屋"产品

1)选择个人项目，点击创建新产品



2)输入相应的产品信息，注意这里的技术方案选择“WiFi/移动网络方案”，通信方式选择“WiFi”，最后点击“保存”

产品列表 / 创建新产品

产品分类：

可穿戴产品及智能硬件

其他

产品名称：

微信宠物屋

技术方案：

Wi-Fi/移动网络方案

蓝牙方案

网关方案

云端

Wi-Fi/3G等移动网络

设备

手机

蓝牙

设备

云端

Wi-Fi/3G等移动网络

网关

子设备

子设备

子设备

选择通讯方式：

☒ Wi-Fi
 ☐ 移动网络

是否变长数据点：

☐

保存

在线咨询

3)点击“去添加数据点”

个人项目 / 微信宠物屋

选项

申请发布

产品信息

基本信息

数据点

虚拟设备

设备日志

开发向导

服务

应用配置

应用开发

MCU开发

产测工具

固件升级 (OTA)

+ 添加服务

统计

概览

新增上线

活跃设备

活跃周期

连接时长

开发向导

1 定义产品功能

2 MCU开发
App/微信开发

3 功能调试

4 发布产品

定义产品功能说明

产品开发的第一步是定义产品的功能，一个数据点可以定义为产品的某个功能，如开关等。产品的数据点如何定义，请查看教程 [《如何定义数据点》](#)。

去添加数据点

MCU 开发资源

机智云根据你定义的产品数据点，会自动生成MCU串口通信代码或整个MCU工程代码，同时也有SOC方案的工程代码。

如果想了解接入机智云的串口通信协议，可下载 [《微信宠物屋 - 机智云SOC方案接入通信协议文档》](#) 或 [《微信宠物屋 - 机智云独立MCU方案接入通信协议文档》](#)。此外，也提供功能参数的 [《微信宠物屋 - 机智云接入JSON文档》](#)，此文档是对协议的格式化说明，包含每个数据点的ID、描述、数据类型、位置信息等。

进入MCU开发

App 开发资源

4)在“管理”中点击“选择产品数据点模板”

产品信息

基本信息

数据点

虚拟设备

设备日志

开发向导

服务

应用配置

应用开发

MCU开发

产测工具

固件升级 (OTA)

+ 添加服务

统计

概览

新增上线

活跃设备

活跃周期

连接时长

数据点 ?

定义数据点教程



尚未创建产品的数据点，快去建立适合的数据点吧~

+ 新建数据点

管理

选择产品数据点模板

导入Excel

5)选择“Gokit Demo”，并“应用此模板”

产品信息

基本信息

数据点

虚拟设备

设备日志

开发向导

服务

应用配置

应用开发

MCU开发

产测工具

固件升级 (OTA)

+ 添加服务

统计

概览

新增上线

活跃设备

活跃周期

连接时长

数据点 / 方案模板

机智云智能电暖机

开源项目《智能电暖机》数据点。

通信方式：Wi-Fi

数据点：10

应用此模板

机智云智能热水器

开源项目《智能热水器》数据点。

通信方式：Wi-Fi

数据点：12

应用此模板

机智云空气净化器

开源项目《智能空气净化器》数据点。

通信方式：Wi-Fi

数据点：20

应用此模板

机智云插座

开源项目《机智云插座》设备数据点。

通信方式：Wi-Fi

数据点：8

应用此模板

智能云空调

开源项目《智能云空调》的数据点模板...

通信方式：Wi-Fi

数据点：11

应用此模板

Gokit Demo

基于Gokit板载元器件的智能宠物屋

通信方式：Wi-Fi

数据点：15

应用此模板

机智云窗帘

注：这里会导入基于Gokit板载元器件的智能宠物屋数据点模板

接下来点击“添加”



6)可以看到导入的“微信宠物屋”的相关数据点



3 生成微信宠物屋代码

1) 首先，在使用“代码自动生成工具”前要获取产品所对应的“**Product Secret**”（后文简称“PS”）

产品信息

- 基本信息
- 数据点
- 虚拟设备
- 设备日志
- 开发向导

服务

- 应用配置
- 应用开发
- MCU开发
- 产测工具
- 固件升级 (OTA)
- + 添加服务

统计

- 概览
- 新增上线
- 活跃设备
- 活跃周期
- 连接时长

基本信息

产品名称：	微信宠物屋
产品类型：	智能家居/生活小家电/咖啡机
通讯方式：	Wi-Fi
Product Key：	189655b1df9a473c8312f6792b917dc5
Product Secret：	9161*****e87f 显示完整密钥
是否变长数据点：	否
设备分享功能：	未开启
创建时间：	2017-06-01
更新时间：	2017-06-01
描述：	无

[修改](#)

2) 生成MCU版代码

在“服务”中选择“**MCU方案**”，平台选择为“**STM32F103C8x**”，填入之前在“基本信息”中获得的 **PS**，然后点击“生成代码包”，等待一段时间后便会生成相应的源码工程。

产品信息

基本信息

数据点

虚拟设备

设备日志

开发向导

服务

应用配置

应用开发

1. MCU开发

产测工具

固件升级 (OTA)

+ 添加服务

统计

概览

新增上线

活跃设备

活跃周期

连接时长

MCU开发

硬件方案:

2. 独立MCU方案

WiFi

MCU

传感器

传感器

传感器

SOC方案

WiFi

传感器

传感器

传感器

硬件平台: 3. STM32F103C8x

Product: 4. 91614823aa 7f626203e87f

Secret:

5. 生成代码包

3) 下载生成好的代码工程压缩包并解压

MCU开发

MCU代码生成结果

硬件方案: MCU

硬件平台: ArduinoUNOR3

下载

修改

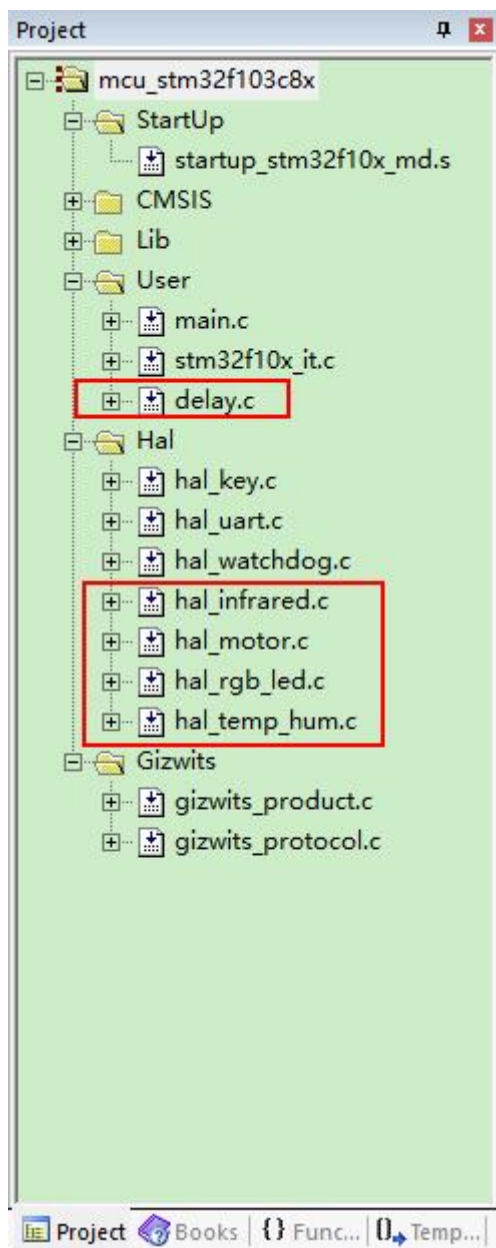
4 移植微信宠物屋驱动代码

1)将 微信宠物屋教程\STM32\驱动库代码 中的驱动库文件拷贝到自动生成代码工程中的 MCU_STM32F103C8x_source\Hal 文件夹中

MCU_STM32F103C8x_source > Hal

名称	修改日期	类型
Hal_infrared	2017/6/1 17:21	文件夹
Hal_key	2017/3/6 10:16	文件夹
hal_motor	2017/6/1 17:21	文件夹
Hal_rgb_led	2017/6/1 17:21	文件夹
Hal_Soil_moisture	2017/6/1 17:21	文件夹
hal_temp_hum	2017/6/1 17:25	文件夹
Hal_Uart	2017/3/6 10:16	文件夹
Hal_Watchdog	2017/3/6 10:16	文件夹

2)在工程项目中国添加驱动代码的“.c”文件



3)在代码中添加相应的函数调用

在MCU_STM32F103C8x_source\User\main.c 文件中添加各驱动库的头文件

```
#include "Hal_motor/Hal_motor.h"
#include "Hal_rgb_led/Hal_rgb_led.h"
#include "Hal_temp_hum/Hal_temp_hum.h"
#include "Hal_infrared/Hal_infrared.h"
#include "gizwits_product.h"
```

在**MCU_STM32F103C8x_source\User\main.c**文件的**userInit()**函数中添加各sensor的初始化

```
void userInit(void)
{
    uartxInit();
    watchdogInit(2);
    memset((uint8_t*)&currentDataPoint, 0, sizeof(dataPoint_t));

    delayInit(72);
    rgbLedInit();          ///< 新添加代码: RGB LED初始化
    rgbKeyGpioInit();      ///< 新添加代码: RGB LED初始化
    motorInit();           ///< 新添加代码: 电机初始化
    dht11Init();           ///< 新添加代码: 温湿度初始化
    irInit();              ///< 新添加代码: 红外初始化
    motorStatus(0);        ///< 新添加代码: 电机初始化
}
```

在**MCU_STM32F103C8x_source\User\main.c**文件的**userHandle()**函数中添加只读型传感器数据点相关的代码

```
void userHandle(void)
{
    uint8_t ret = 0;
    static uint32_t thLastTimer = 0;

    ///< 新添加代码: 红外传感器数据获取
    currentDataPoint.valueInfrared = irHandle();

    ///< 新添加代码: 温湿度传感器数据获取
    if((gizGetTimerCount()-thLastTimer) > SAMPLING_TIME_MAX)    //上报间隔2S
    {
        ret = dht11Read((uint8_t *)&currentDataPoint.valueTemperature, (uint8_t
*)&currentDataPoint.valueHumidity);
        if(ret != 0)
        {
            printf("Failed to read DHT11\r\n");
        }

        thLastTimer = gizGetTimerCount();
    }
}
```

在**MCU_STM32F103C8x_source\User\main.c**文件的**key2ShortPress()**函数与**key2LongPress()**函数中添加长/短按key2时的LED点亮代码


```
void key2ShortPress(void)
{
    printf("KEY2 PRESS ,Soft AP mode\n");

    ///< 新添加代码: Soft AP mode, RGB red
    ledRgbControl(250, 0, 0);

    gizwitsSetMode(WIFI_SOFTAP_MODE);
}
```

```
void key2LongPress(void)
{
    //AirLink mode
    printf("KEY2 PRESS LONG ,AirLink mode\n");

    ///< 新添加代码: AirLink mode, RGB Green
    ledRgbControl(0, 250, 0);

    gizwitsSetMode(WIFI_AIRLINK_MODE);
}
```

在**MCU_STM32F103C8x_source\Gizwits\gizwits_product.c**文件的**gizwitsEventProcess()**函数中添加可写型传感器数据点相关的代码

```

int8_t gizwitsEventProcess(eventInfo_t *info, uint8_t *data, uint32_t len)
{
    uint8_t i = 0;
    dataPoint_t *dataPointPtr = (dataPoint_t *)data;
    moduleStatusInfo_t *wifiData = (moduleStatusInfo_t *)data;

    protocolTime_t *ptime = (protocolTime_t *)data;

    if((NULL == info) || (NULL == data))
    {
        return -1;
    }

    for(i=0; i<info->num; i++)
    {
        switch(info->event[i])
        {
            case EVENT_LED_ONOFF:
                currentDataPoint.valueLED_OnOff = dataPointPtr->valueLED_OnOff;
                GIZWITS_LOG("Evt: EVENT_LED_ONOFF %d \n", currentDataPoint.valueLED_OnOff);
                if(0x01 == currentDataPoint.valueLED_OnOff)
                {
                    ledRgbControl(254,0,0);          ///< 新添加代码：对应开启红灯
                }
                else
                {
                    ledRgbControl(0,0,0);          ///< 新添加代码：对应关闭红灯
                }
                break;

            case EVENT_LED_COLOR:
                currentDataPoint.valueLED_Color = dataPointPtr->valueLED_Color;
                GIZWITS_LOG("Evt: EVENT_LED_COLOR %d\n", currentDataPoint.valueLED_Color);
                switch(currentDataPoint.valueLED_Color)
                {
                    case LED_COLOR_VALUE0:

ledRgbControl(currentDataPoint.valueLED_R,currentDataPoint.valueLED_G,currentDataPoint.valueLED_B)
;

                                break;

                    case LED_COLOR_VALUE1:
                        ledRgbControl(254, 254, 0);          ///< 新添加代码：对应LED组合颜色-黄色
                        break;
                    case LED_COLOR_VALUE2:
                        ledRgbControl(254, 0, 70);          ///< 新添加代码：对应LED组合颜色-紫色
                        break;
                    case LED_COLOR_VALUE3:
                        ledRgbControl(238 ,30 ,30);          ///< 新添加代码：对应LED组合颜色-粉色
                        break;
                    default:
                        break;
                }
                break;
        }
    }
}

```

```

    case EVENT_LED_R:
        currentDataPoint.valueLED_R = dataPointPtr->valueLED_R;
        GIZWITS_LOG("Evt:EVENT_LED_R %d\n",currentDataPoint.valueLED_R);
        ///< 新添加代码：对应设置LED组合色

ledRgbControl(currentDataPoint.valueLED_R,currentDataPoint.valueLED_G,currentDataPoint.valueLED_B)
;

        break;

    case EVENT_LED_G:
        currentDataPoint.valueLED_G = dataPointPtr->valueLED_G;
        GIZWITS_LOG("Evt:EVENT_LED_G %d\n",currentDataPoint.valueLED_G);
        ///< 新添加代码：对应设置LED组合色

ledRgbControl(currentDataPoint.valueLED_R,currentDataPoint.valueLED_G,currentDataPoint.valueLED_B)
;

        break;

    case EVENT_LED_B:
        currentDataPoint.valueLED_B = dataPointPtr->valueLED_B;
        GIZWITS_LOG("Evt:EVENT_LED_B %d\n",currentDataPoint.valueLED_B);
        ///< 新添加代码：对应设置LED组合色

ledRgbControl(currentDataPoint.valueLED_R,currentDataPoint.valueLED_G,currentDataPoint.valueLED_B)
;

        break;

    case EVENT_MOTOR_SPEED:
        currentDataPoint.valueMotor_Speed = dataPointPtr->valueMotor_Speed;
        GIZWITS_LOG("Evt:EVENT_MOTOR_SPEED %d\n",currentDataPoint.valueMotor_Speed);
        ///< 新添加代码：对应设定电机转速
        motorStatus(currentDataPoint.valueMotor_Speed);
        break;
    case WIFI_SOFTAP:
        break;
    case WIFI_AIRLINK:
        break;
    case WIFI_STATION:
        break;
    case WIFI_CON_ROUTER:
        ledRgbControl(0, 0, 0);    ///< 新添加代码：连接路由后关闭LED灯
        break;
    case WIFI_DISCON_ROUTER:
        break;
    case WIFI_CON_M2M:
        break;
    case WIFI_DISCON_M2M:
        break;
    case WIFI_RSSI:
        GIZWITS_LOG("RSSI %d\n", wifiData->rssi);
        break;
    case TRANSPARENT_DATA:
        GIZWITS_LOG("TRANSPARENT_DATA \n");
        //user handle , Fetch data from [data] , size is [len]
        break;

```

```
case WIFI_NTP:
    GIZWITS_LOG("WIFI_NTP : [%d-%d-%d %02d:%02d:%02d][%d] \n",ptime->year,ptime->month,ptime->day,ptime->hour,ptime->minute,ptime->second,ptime->ntp);
    break;
default:
    break;
}

return 0;
}
```

5 编译烧录固件并进行测试

可在 [下载中心](#) 中下载对应平台的微信宠物屋官方测试固件以及驱动库文件

The screenshot shows the GIZWITS website's download center. On the left is a navigation menu with categories like '硬件开发资源', '客户端开发资源', '开发与调试工具', and '开放源码'. The main content area is titled '微信宠物屋' (WeChat Pet House). It contains a description of the product and a list of downloadable firmware files. A red box highlights three specific files:

固件名称	发布时间	操作
微信宠物屋 for GoKit3(S) ESP8266 V03000003	2016.12.01 19:46 更新信息 旧版本下载	资源下载
微信宠物屋 for GoKit 2/3 STM32 V03010101	2016.12.01 19:10 更新信息 旧版本下载	资源下载
微信宠物屋 for GoKit 2 Arduino 2.3.1	2016.1.04 12:19 更新信息 旧版本下载	资源下载

Below this, another file is listed: '微信宠物屋 for GoKit 1.0.20141116' with a release time of 2015.4.22 17:38 and a '资源下载' link.

测试固件位于：微信宠物屋教程\Arduino\官方成品固件\mcu_stm32f103c8x.hex

MCU版的固件烧录方式可查看 [ST底板程序编译及下载教程](#) 一文。

微信宠物屋的操作方式可查看 [GoKit使用说明书](#) 一文

6 相关说明

说明：SOC版与MCU版的工程环境搭建与代码细节介绍请查看[文档中心](#)中设备开发的**Gokit**资料一节，本章节只介绍微信宠物屋驱动程序的移植方法。

目录

[机智云平台概述](#)
[机智云平台架构](#)
[机智云优势](#)
[机智云全球联网报告](#)
[展开全部](#)

平台概述

机智云平台概述

机智云平台是机智云物联网公司经过多年行业内的耕耘及对物联网行业个人、企业开发者的一站式智能硬件开发及云服务平台。平台提供了试、应用开发、产测、云端开发、运营管理、数据服务等覆盖智能硬件服务的能力。

机智云平台为开发者提供了自助式智能硬件开发工具与开放的云端服务完善的SDK与API服务能力最大限度降低了物联网硬件开发的技术门槛，提升开发者的产品投产速度，帮助开发者进行硬件智能化升级，更好的连接、服务最终消费者。

机智云平台架构

[GAgent详解](#)
[MCU开发框架使用说明](#)
[2. Gokit资料](#)
[调试与烧写](#)
[ECE Demo开发教程](#)
[ArduinoUno开发教程](#)
[Gokit3系列开发套件简介](#)
[Gokit3硬件手册](#)
[GoKit3\(S\)使用说明书](#)
[Gokit3\(S\)开发套件介绍](#)
[GoKit3\(S\)开发环境搭建](#)
[Gokit3\(S\)源码详解](#)
[GoKit3\(V\)使用说明书](#)
[Gokit3\(V\)开发指南](#)
[GoKit3\(V\)词条管理](#)
[Gokit3\(V\)源码详解](#)
[Gokit2 使用指南](#)