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** 。

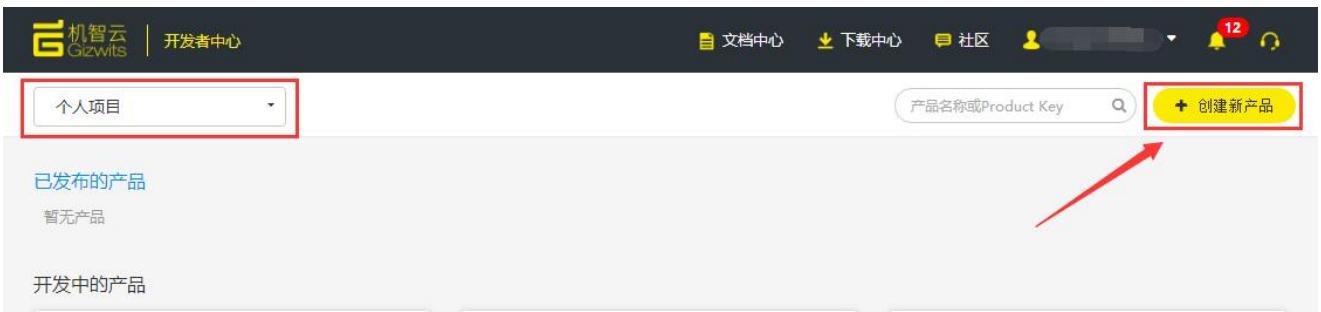
本文主要介绍 **微信宠物屋程序的移植方法** ，Arduino的SDK API说明请查看文档中心 **设备开发中ArduinoUno开发教程中的**ArduinoUnoWiFi SDK之API介绍 一节。

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

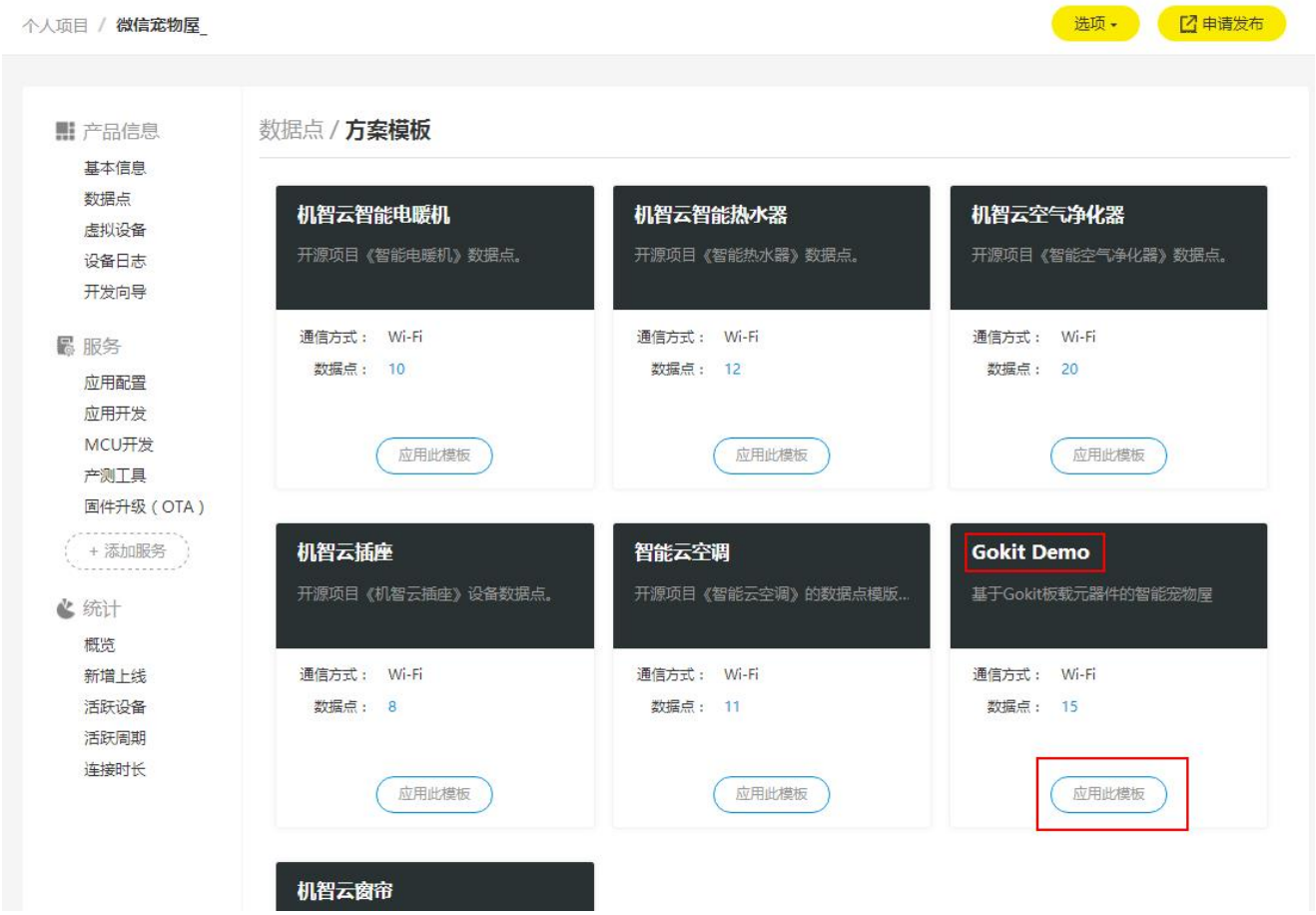


2)输入相应的产品信息，注意这里的技术方案选择“WiFi/移动网络方案”，通信方式选择“WiFi”，最后点击“保存”

3)点击“去添加数据点”

4)在“管理”中点击“选择产品数据点模板”

5)选择“Gokit Demo”，并“应用此模板”



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

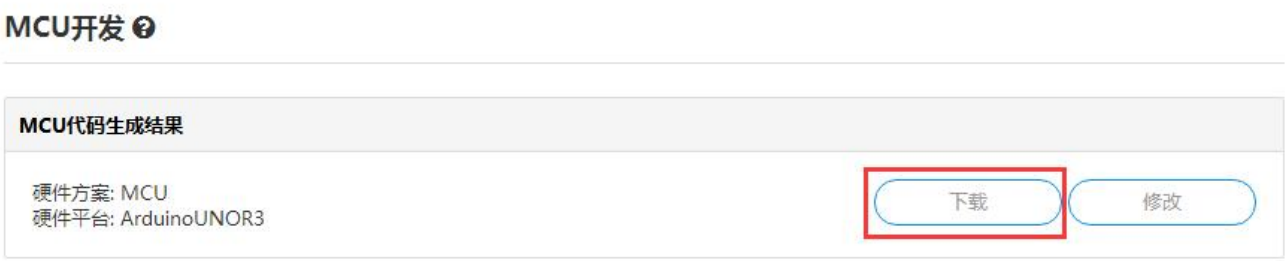
3 生成微信宠物屋代码

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”）

2) 生成Arduino版代码

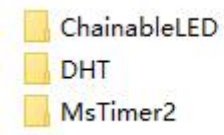
在“服务”中选择“MCU方案”，平台选择默认的"AruinoR3"，填入之前在“基本信息”中获得的 **PS**，然后点击“生成代码包”，等待一段时间后便会生成相应的源码工程。

3) 下载生成好的代码工程

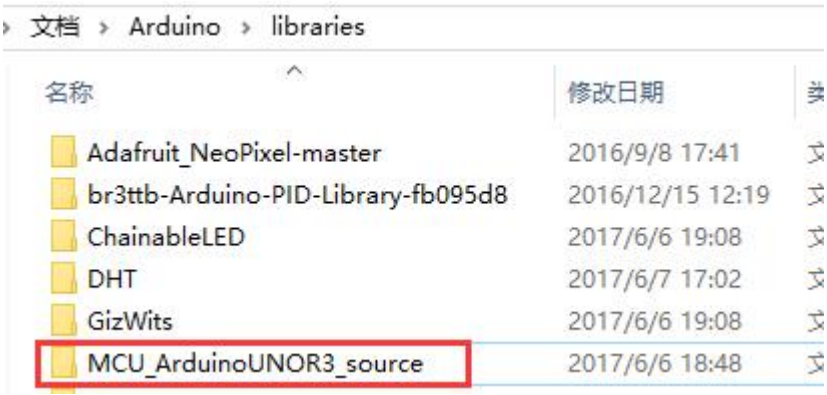


4 移植微信宠物屋驱动代码

1)将 微信宠物屋教程**\Arduino**驱动库代码 目录下的驱动文件拷贝到 **我的文档/Arduino/libraries** 目录下



2)根据 ArduinoUNOWiFi接入机智云介绍 一文将自动生成的Arduino代码包导入ArduinoIDE



3)在数据点示例工程文件 **MCU_ArduinoUNOR3_source\examples\simpleTry\simpleTry.ino** 中添加各驱动库的头文件，以及全局变量

```
#include <DHT.h>
#include <ChainableLED.h>
#include <MsTimer2.h>

/************************** HAL define ***************************/
#define   Infrared_PIN      2      ///< 红外IO管脚
#define   DHTPIN            3      ///< 温湿度IO管脚

#define   MOTOR_PINA        4      ///< 电机IO管脚
#define   MOTOR_PINB        5      ///< 电机IO管脚
#define   KEY1              6      ///< 按键IO管脚
#define   KEY2              7      ///< 按键IO管脚

//温湿度功能值定义
#define   DHTTYPE           DHT11

//电机功能值定义
#define   MOTOR_MAX         100
#define   MOTOR_MAX1        -100
#define   MOTOR_MIN         0
#define   MOTOR_16

//按键功能值定义
#define   KEY1_SHORT_PRESS  1
#define   KEY1_LONG_PRESS   2
#define   KEY2_SHORT_PRESS  4
#define   KEY2_LONG_PRESS   8
#define   NO_KEY            0
#define   KEY_LONG_TIMER    3
unsigned long Last_KeyTime = 0;
uint8_t NetConfigureFlag = 0;
uint8_t gaterSensorFlag = 0;

typedef enum
{
  LED_Costom      = 0x00,
  LED_Yellow      = 0x01,
  LED_Purple      = 0x02,
  LED_Pink        = 0x03,
} LED_ColorTypeDef;

DHT dht(DHTPIN, DHTTYPE);
ChainableLED leds(A5, A4, 1);
```

在**simpleTry.ino**文件中添加温湿度传感器数据获取的函数

```
void DHT11_Read_Data(unsigned char * temperature, unsigned char * humidity)
{
  *temperature = (unsigned char)dht.readTemperature();
  *humidity = (unsigned char)dht.readHumidity();
  return;
}
```

在**simpleTry.ino**文件中添加控制电机的驱动函数

```
void Motor_status(long motor_speed)
{

  unsigned char Temp_motor_speed = 0;
  if (motor_speed == 0) //停止□
  {
    digitalWrite(MOTOR_PINA, LOW);
    digitalWrite(MOTOR_PINB, LOW);
  }
  if (motor_speed > 0) //正转
  {
    Temp_motor_speed = (motor_speed - 0) * 51;
    if (Temp_motor_speed > 255) Temp_motor_speed = 255;
    digitalWrite(MOTOR_PINA, LOW);
    analogWrite( MOTOR_PINB, Temp_motor_speed);
  }
  if (motor_speed < 0) //反转
  {
   Temp_motor_speed = 255 - (0 - motor_speed) * 51; //Temp_motor_speed = (255 - (5 + motor_speed))
* 51;
    if (Temp_motor_speed > 255) Temp_motor_speed = 255;
    digitalWrite(MOTOR_PINA, HIGH);
    analogWrite( MOTOR_PINB, Temp_motor_speed );
  }
}
```

在**simpleTry.ino**文件中添加**RGB LED**控制函数

```
void LED_RGB_Control(byte red, byte green, byte blue)
{
  leds.setColorRGB(0, red, green, blue);
}
```

在**simpleTry.ino**文件中添加按键控制的驱动函数

```
unsigned long gokit_time_s(void)
{
  return millis() / 1000;
}

char gokit_key1down(void)
{
  unsigned long keep_time = 0;
  if (digitalRead(KEY1) == LOW)
  {
    delay(100);
    if (digitalRead(KEY1) == LOW)
    {
      keep_time = gokit_time_s();
      while (digitalRead(KEY1) == LOW)
      {
        if ((gokit_time_s() - keep_time) > KEY_LONG_TIMER)
        {
          Last_KeyTime = gokit_time_s();
          return KEY1_LONG_PRESS;
        }
      } //until open the key

      if ((gokit_time_s() - Last_KeyTime) > KEY_LONG_TIMER)
      {
        return KEY1_SHORT_PRESS;
      }
      return 0;
    }
    return 0;
  }
  return 0;
}

char gokit_key2down(void)
{
  int unsigned long keep_time = 0;
  if (digitalRead(KEY2) == LOW)
  {
    delay(100);
    if (digitalRead(KEY2) == LOW)
    {
      keep_time = gokit_time_s();
      while (digitalRead(KEY2) == LOW) //until open the key
      {

        if ((gokit_time_s() - keep_time) > KEY_LONG_TIMER)
        {
          Last_KeyTime = gokit_time_s();
          return KEY2_LONG_PRESS;
        }
      }
```

```
      if ((gokit_time_s() - Last_KeyTime) > KEY_LONG_TIMER)
      {
        return KEY2_SHORT_PRESS;
      }
      return 0;
    }
    return 0;
  }
  return 0;
}

char gokit_keydown(void)
{
  char ret = 0;
  ret |= gokit_key2down();
  ret |= gokit_key1down();
  return ret;

}

void KEY_Handle(void)
{
  /*  Press for over than 3 second is Long Press  */
  switch (gokit_keydown())
  {
    case KEY1_SHORT_PRESS:
      Serial.println(F("KEY1_SHORT_PRESS , Production Test Mode "));
      myGizwits.setBindMode(WIFI_PRODUCTION_TEST);
      break;
    case KEY1_LONG_PRESS:
      Serial.println(F("KEY1_LONG_PRESS ,Wifi Reset"));
      myGizwits.setBindMode(WIFI_RESET_MODE);
      break;
    case KEY2_SHORT_PRESS:
      Serial.println(F("KEY2_SHORT_PRESS Soft AP mode"));
      myGizwits.setBindMode(WIFI_SOFTAP_MODE);
      ///< 进入Soft AP mode RGB LED亮红灯
      NetConfigureFlag = 1;          ///< 等待连接路由的标志位
      LED_RGB_Control(100, 0, 0);    ///< 红绿灯
      break;
    case KEY2_LONG_PRESS:
      Serial.println(F("KEY2_LONG_PRESS ,AirLink mode"));
      myGizwits.setBindMode(WIFI_AIRLINK_MODE);
      ///< 进入AirLink mode RGB LED亮绿灯
      NetConfigureFlag = 1;          ///< 等待连接路由的标志位
      LED_RGB_Control(0, 100, 0);    ///< 亮绿灯
      break;
    default:
      break;
  }
}
```

在**simpleTry.ino**文件中添加**WiFi**状态事件触发函数

```
void wifiStatusHandle()
{
  if(myGizwits.wifiHasBeenSet(WIFI_SOFTAP))
    Serial.println(F("WIFI_SOFTAP!"));

  if(myGizwits.wifiHasBeenSet(WIFI_AIRLINK))
    Serial.println(F("WIFI_AIRLINK!"));

  if(myGizwits.wifiHasBeenSet(WIFI_STATION))
    Serial.println(F("WIFI_STATION!"));

  if(myGizwits.wifiHasBeenSet(WIFI_CON_ROUTER))
  {
    Serial.println(F("WIFI_CON_ROUTER!"));
    if(1 == NetConfigureFlag)
    {
       LED_RGB_Control(0, 0, 0);   ///< Airlink配置成功即模组连接到路由后关闭绿灯
       NetConfigureFlag = 0;
    }
  }

  if(myGizwits.wifiHasBeenSet(WIFI_DISCON_ROUTER))
    Serial.println(F("WIFI_DISCON_ROUTER!"));

  if(myGizwits.wifiHasBeenSet(WIFI_CON_M2M))
    Serial.println(F("WIFI_CON_M2M!"));

  if(myGizwits.wifiHasBeenSet(WIFI_DISCON_M2M))
    Serial.println(F("WIFI_DISCON_M2M!"));
}
```

> 说明：通过**myGizwits.hasBeenSet()**这个方法判断这个数据点事件有是否发生，例如发生了连接到路由的事件***WIFI_CON_ROUTER***，就去执行关闭LED灯的动作。

在**simpleTry.ino**文件中添加只读型数据点相关的处理代码

```
void GizWits_GatherSensorData(void)
{
  uint8_t curTem, curHum;
  bool Infrared;

  //获取红外传感器IO的高低电平状态
  if (digitalRead(Infrared_PIN))
  {
    Infrared = 0;
  }
  else
  {
    Infrared = 1;
  }

  //获取温湿度传感器的数值
  DHT11_Read_Data(&curTem, &curHum);

  myGizwits.write(VALUE_INFRARED, Infrared);               ///< 上报红外状态数据
  myGizwits.write(VALUE_TEMPERATURE, (long)curTem);        ///< 上报温度数据
  myGizwits.write(VALUE_HUMIDITY, (unsigned long)curHum);  ///< 上报湿度数据
}

void gokit_timer(void)
{
    static long gaterTime = 0;

    gaterTime++;

    if(gaterTime >= 1000)
    {
        gaterTime = 0;
        gaterSensorFlag = 1;
    }
}
```

> 说明：通过**myGizwits.write()**这个方法来上报对应数据点的数据，例如：获取到红外传感器的数值后，使用 **myGizwits.write** 第一个参数 ***VALUE_INFRARED*** 表示红外传感器的数据点事件宏，第二个参数表示数据点的状态数值。

在 **simpleTry.ino** 文件中的**loop**（）函数中添加 间隔一秒的状态机调用

```
  if (gaterSensorFlag != 0)
  {
    GizWits_GatherSensorData();
    gaterSensorFlag = 0;
  }
```

> 说明：***gokit_timer()*** 函数实现了周期性（这里间隔1秒）获取只读类型数据点的的数据，它是由初始化时的定时器实现的。每隔1秒设置一次***gaterSensorFlag*** 标志位，然后触发 **loop()** 中的 **GizWits_GatherSensorData** 标志进行数据采集。如果在 ***loop()*** 中直接调用例如温湿度这样的时序实现的驱动函数的话，由于loop执行过快会造成读传感器的异常，同时也避免了上报数据过频繁造成的问题 。

在**simpleTry.ino**文件中的**loop**（）函数中添加可写型传感器数据点相关的处理代码

```
 bool varR_LED_OnOff = 0;
 unsigned long varR_LED_Color = 0;
 static unsigned long varR_LED_R = 0;
 static unsigned long varR_LED_G = 0;
 static unsigned long varR_LED_B = 0;
 long varR_Motor_Speed = 0;

//执行"开启/关闭红色灯"数据点的操作
 if(myGizwits.hasBeenSet(EVENT_LED_ONOFF))
 {
   myGizwits.read(EVENT_LED_ONOFF,&varR_LED_OnOff);//Address for storing data
     if(varR_LED_OnOff == 1)
     {
         LED_RGB_Control(100, 0, 0);
     }
     else
    {
        LED_RGB_Control(0, 0, 0);
     }
 }

 //执行"设定LED组合颜色"数据点的操作
 if(myGizwits.hasBeenSet(EVENT_LED_COLOR))
 {
   myGizwits.read(EVENT_LED_COLOR,&varR_LED_Color);//Address for storing data

   if (varR_LED_Color == LED_Yellow)
   {
     LED_RGB_Control(254, 254, 0);
   }

   if (varR_LED_Color == LED_Purple)
   {
     LED_RGB_Control(254, 0, 70);
   }
   if (varR_LED_Color == LED_Pink)
   {
     LED_RGB_Control(238 , 30 , 30);
   }
 }

 //执行"设定LED红色值"数据点的操作
 if(myGizwits.hasBeenSet(EVENT_LED_R))
 {
   myGizwits.read(EVENT_LED_R,&varR_LED_R);//Address for storing data
   LED_RGB_Control(varR_LED_R , varR_LED_G , varR_LED_B);
 }

 //执行"设定LED绿色值"数据点的操作
 if(myGizwits.hasBeenSet(EVENT_LED_G))
 {
   myGizwits.read(EVENT_LED_G,&varR_LED_G);//Address for storing data
   LED_RGB_Control(varR_LED_R , varR_LED_G , varR_LED_B);
```

```
  }

  //执行"设定LED蓝色值"数据点的操作
  if(myGizwits.hasBeenSet(EVENT_LED_B))
  {
    myGizwits.read(EVENT_LED_B,&varR_LED_B);//Address for storing data
    LED_RGB_Control(varR_LED_R , varR_LED_G , varR_LED_B);
  }

  //执行"设定电机转速"数据点的操作
  if(myGizwits.hasBeenSet(EVENT_MOTOR_SPEED))
  {
    myGizwits.read(EVENT_MOTOR_SPEED,&varR_Motor_Speed);//Address for storing data
    Motor_status(varR_Motor_Speed);
  }
```

> 说明：通过**myGizwits.hasBeenSet()**这个方法判断这个数据点事件有是否发生，如果数据点事件发生了就通过**myGizwits.read()**这个方法把发生的事件所产生的数据读取出来，然后再做相应的判断。

在**simpleTry.ino**文件中的**loop**（）函数中添加**WiFi状态事件触发函数 、按键控制的驱动函数**的调用

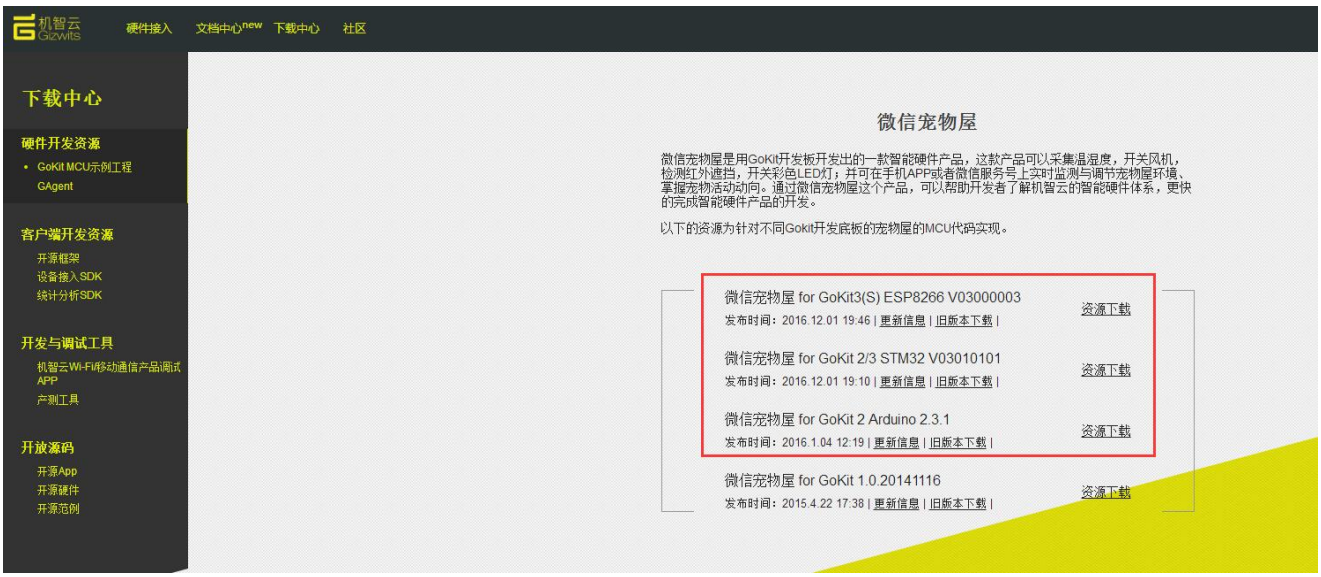
```
  KEY_Handle();         ///< key handle , network configure
  wifiStatusHandle();   ///< WIFI Status Handle
```

在**simpleTry.ino**文件的**setup( )**函数中添加各sensor的初始化

```
void setup() {
  // put your setup code here, to run once:

  Serial.begin(9600);
  myGizwits.begin();

  ///<  温度传感初始
  dht.begin();

  ///<  RGB LED初始
  leds.init();
  digitalWrite(A0, HIGH);//使能RGB LED

  ///<  新添加代码: 按键初始
  pinMode(KEY1, INPUT_PULLUP); //KEY1 上拉输入
  pinMode(KEY2, INPUT_PULLUP); //KEY2 上拉输入

  ///<  电机初始化
  pinMode(MOTOR_PINA, OUTPUT);
  pinMode(MOTOR_PINB, OUTPUT);
  digitalWrite(MOTOR_PINB, LOW);
  digitalWrite(MOTOR_PINA, LOW);
  Motor_status(0);

  //定时中断初始
  MsTimer2::set(1, gokit_timer); // 1ms period
  MsTimer2::start();
}
```

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



测试固件位于：**微信宠物屋教程\Arduino\官方成品固件\MCU_ArduinoUNOR3_source.cpp.hex**

Arduino的HEX烧录方式可查看 给Arduino上传HEX文件 一文

微信宠物屋的操作方式可查看 GoKit3使用说明书 一文

6 相关说明

> Arduino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**ArduinoUno**开发教程一节，本章节主要介绍微信宠物屋驱动程序的移植方法， 可查看API的使用介绍 和 智能小夜灯的实现教程 。