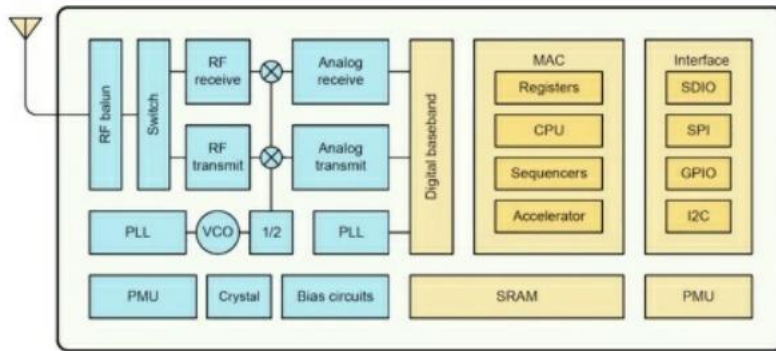


# 第一章----第三章

## 1、大体结构



第一章 认识BSP32

功能强大：双核CPU 工作频率达100MHz 100MHz 100MHz  
可连接摄像头 识别语音数据 内置3WIFI 蓝牙 4.0  
Arduino库 等...  
内置传感器(温度、湿度、气压、触摸)  
具有深度睡眠模式(低功耗 10~15 uA)

总结：相比于其前身BSP866 其有更高的可靠性、安全性与更高性能、更多功能。

第二章 新的BSP32的外观。

外观主要以图片形式介绍，详情看电子书。

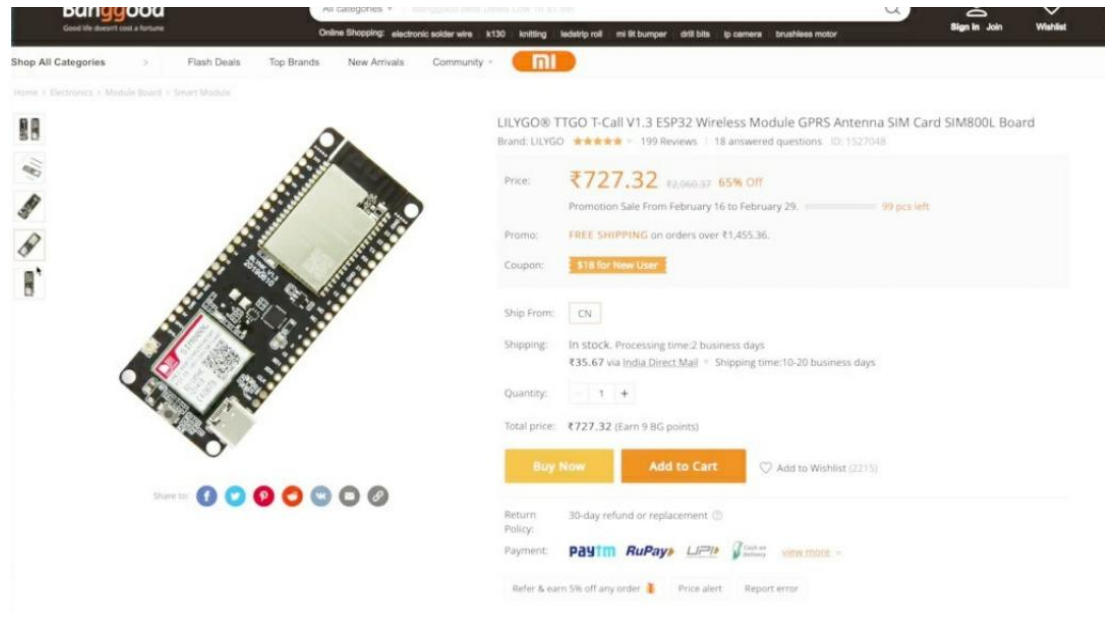
第三章 BSP32 - Arduino 替代品。

总体概述、特点介绍。

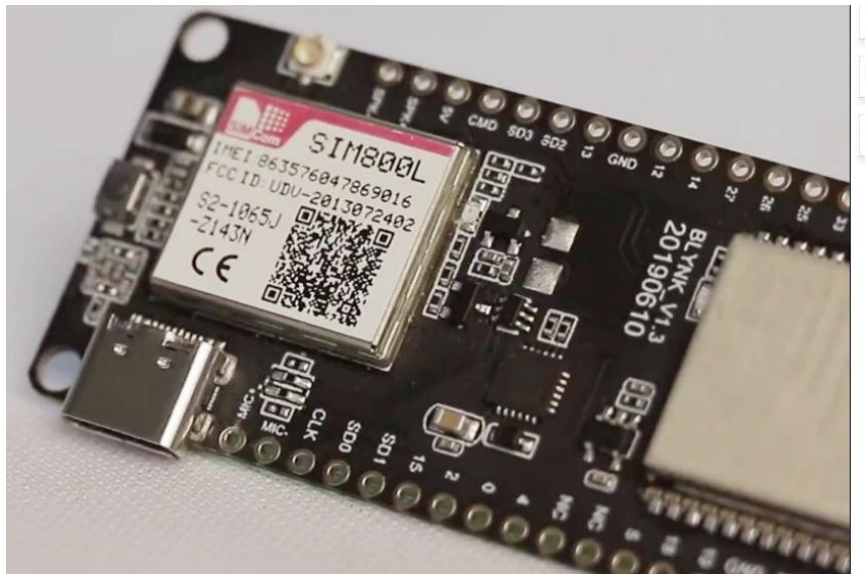
第四章 BSP32 SIM801。

## 第四章 *ESP32 SIM800L*

订购



模型样子及其型号



可以实现 wifi、蓝牙、连接或双线扩展 32 位处理器

更有 *DSM, GPRS, connectivity*

We can

communicate via SMS, via phone calls, and GPRS,  
internet connection.

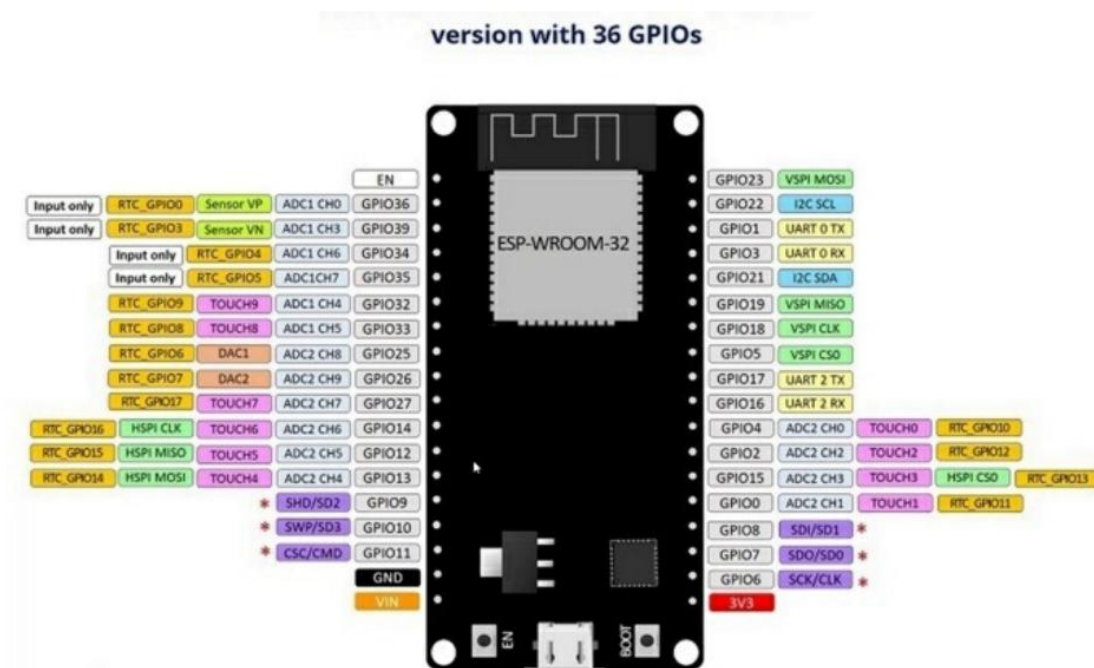
不再需要 WiFi 和蓝牙了，可以在任何地方实现联网项目

## 第五章 ESP32 PINOUT V1 DOIT

非常类似于 Arduino 纳米,include 18 analogs to digital  
converter or ADC (在那里你可以接收模拟信号，这些信号可  
以在内部转换为数字信号)

还有三个用于 Cehre 通信的 S.P.I 接口，三个用于 S.R.O.通信  
的艺术接口和两个用于西雅图通信的视线接口

板块模型



每个引脚接口都有不同的作用，如上图所示

可否作为输入输出如下显示

GPIO	Input	Output	Notes
0	Output only	OK	outputs PWM signal at boot
1	TX pin	OK	debug output at boot
2	OK	OK	connected to on-board LED
3	OK	RX pin	HIGH at boot
4	OK	OK	
5	OK	OK	outputs PWM signal at boot
6	X	X	connected to the integrated SPI flash
7	X	X	connected to the integrated SPI flash
8	X	X	connected to the integrated SPI flash
9	X	X	connected to the integrated SPI flash
10	X	X	connected to the integrated SPI flash
11	X	X	connected to the integrated SPI flash
12	OK	OK	boot fail if pulled high
13	OK	OK	
14	OK	OK	outputs PWM signal at boot
15	OK	OK	outputs PWM signal at boot
16	OK	OK	

绿色突出显示的针可以使用黄色突出显示的针是可以的工具

17	OK	OK	
18	OK	OK	
19	OK	OK	
21	OK	OK	
22	OK	OK	
23	OK	OK	
25	OK	OK	
26	OK	OK	
27	OK	OK	
32	OK	OK	
33	OK	OK	
34	OK		input only
35	OK		input only
36	OK		input only
39	OK		input only

而红色突出显示的针不建议用作输入或输出，17 号及之后

的指针如下图

连接时确保指针连接正确并且代码与实物对应

## 第六章 ESP32 VS ARDUINO SERVO MOTOR CONTROL



*It has the NRF 24 L zero one radio built-in, which is awesome for creating the wireless index.*

相比之下 esp32 拥有更多的 *GPIO* 引脚，并且可以使用几乎任何一个这些引脚。作为伺服控制的 *PWM* 引脚

## 第七章 ARDUINO CODING

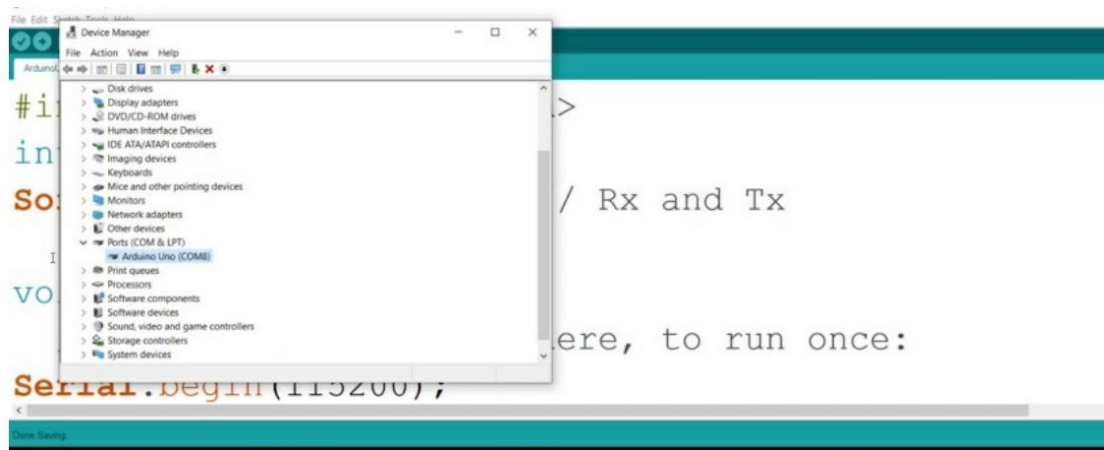
*The first thing that we need to do is include the soft Syrian communication protocol.*

代码示例



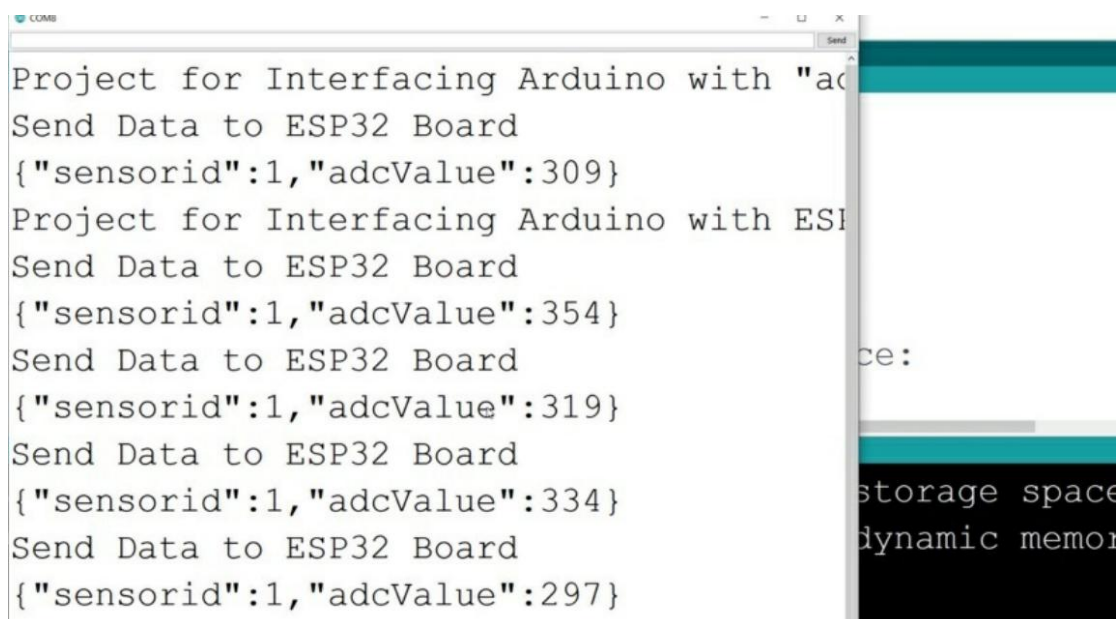
```
void loop() {
  // put your main code here, to run repeatedly:
  Serial.println("Send Data to ESP32 Board");
  int adcValue = analogRead(A0);
  Serial.print("{ \"sensorid\":");    // { \ sensor data
}
```

Now that we have  
the boards connected, go to the device manager



最后被发送的数据

结果如下图所示



## 第八章 ESP CODING PART1 DEFINE

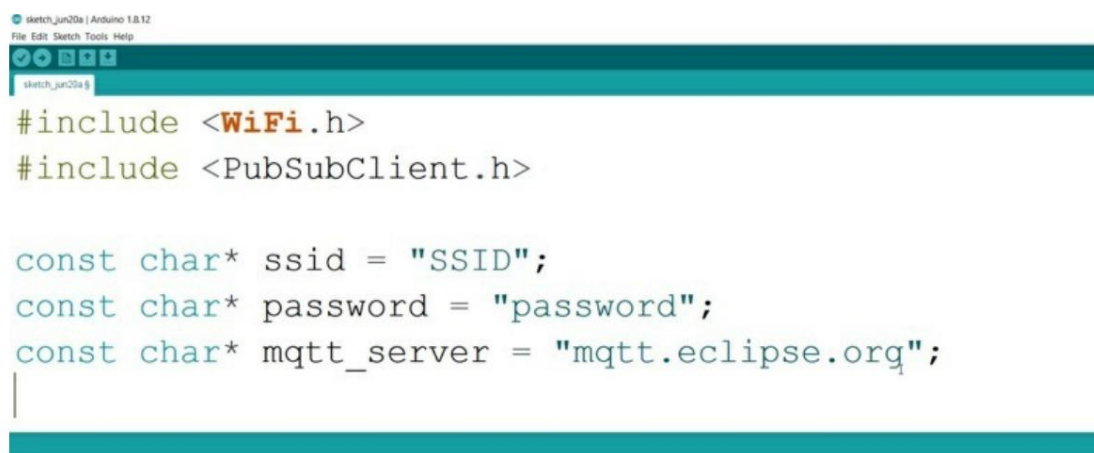
### VARIABLES

首先打开 ARDUINO 软件，给板子接入一个 WiFi 网络，Then we need to connect to that, um, Q Titi broadcast. And we also need to send every new line received on the Syria line to the um Q Titi blocker.最后我们的 esp 板子会接收到所有的数据。

create a new sketch.（首先安装一些库，WiFi 库）

And we also need to include a public subclan Tabari, which is, Atlant, liable for that E. S. P 32 that provides support for Amcu S. T. 然后初始化一些设置，设置以后用到的 WiFi 密码，After that, we need the MQ TTR setup link。

具体如图

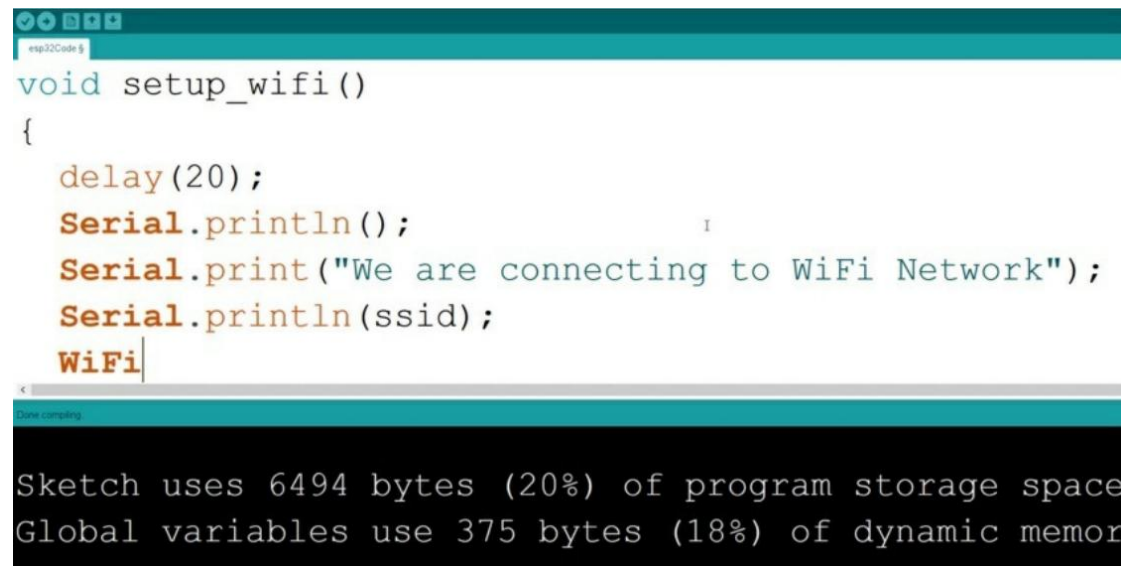


```
sketch_jun20a | Arduino 1.8.12
File Edit Sketch Tools Help
sketch_jun20a.g
#include <WiFi.h>
#include <PubSubClient.h>

const char* ssid = "SSID";
const char* password = "password";
const char* mqtt_server = "mqtt.eclipse.org";
|
```

## 第九章 ESP CODING PART2 WIFI AND MQTT

首先命名一种方法 *void*, 尝试把 *WiFi* 连接到我们的 *iPhone* 上,  
*inside this method to add a delay of 20 milliseconds and*  
*Syrian*



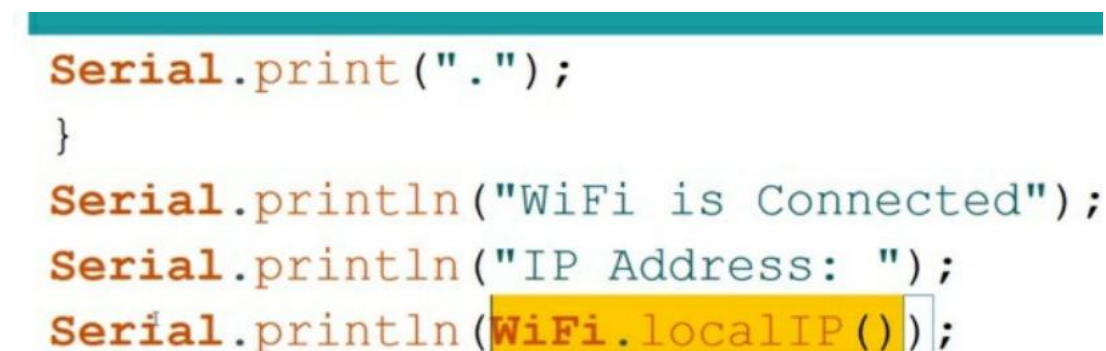
```
void setup_wifi()
{
    delay(20);
    Serial.println();
    Serial.print("We are connecting to WiFi Network");
    Serial.println(ssid);
    WiFi
```

Done compiling

Sketch uses 6494 bytes (20%) of program storage space  
Global variables use 375 bytes (18%) of dynamic memory

这需要 *ID* 和密码, 如果没有连接成功的话尝试延迟后重新再试一次

调用一个叫做 *WiFi* 的函数将他的 *IP* 地址打印出来, 如下图黄色部分



```
Serial.print(".");
}
Serial.println("WiFi is Connected");
Serial.println("IP Address: ");
Serial.println(WiFi.localIP());
```

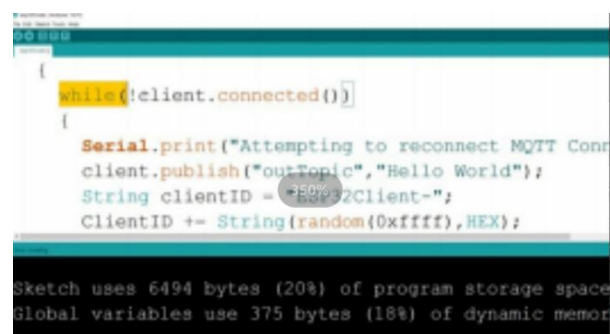


第二个函数是通过 **void** 来实现重新连接

*This will make sure that each time won't have a new client I. D.*

```
void reconnect()
{
  while(!client.connected())
  {
    Serial.print("Attempting to reconnect MQTT Conn
    client.publish
  }
}
```

*This is the reconnect function for the Kuttly server.*



```
{
  while(!client.connected())
  {
    Serial.print("Attempting to reconnect MQTT Conn
    client.publish("outTopic","Hello World");
    String clientID = "ESP8260Client-";
    ClientID += String(random(0xffff),HEX);
  }
}
```

Sketch uses 6494 bytes (20%) of program storage space.  
Global variables use 375 bytes (18%) of dynamic memory.

如果没有连接成功，可以再确认一下客户端没有连接

## 第十章 This is the reconnect function for the Kutty server.

*The first thing that we need to do is initialize Pacoli communication to a specific third-rate, a simple date, or in the board.*

为串行通信设置一个半秒的延迟，调用设置的 *WiFi* 函数，在此之后，我们需要调用客户端来设置服务器来传递服务器参数

```
// put your setup code here, to run once:
Serial.begin(115200);
Serial.setTimeout(500);
setup_wifi();
client.setServer(mqtt_server, mqtt_port);
reconnect();
}
```

*Now we need*

*to define a new function between the setup and Laub, let's call it to publish.*

```

esp32Code $
if(!client.connected())
{
    reconnect();
}
client.publish(MQTT_SERIAL_PUBLISH_CH,);
}

```

确保 BenKutty 服务器已使用 if 语句连接

*We have created defined some variables and we have used them, Kutta T Ledbury, the wildfire library set up wildfire to make sure that we are connecting or we are connected to our Wi-Fi network. And we are online.*

```

File Edit Sketch Tools Help
esp32Code $
#define mqtt_port 1883
#define MQTT_USER "mqtt username"
#define MQTT_PASSWORD "MQTT PASSWORD"
#define MQTT_SERIAL_PUBLISH_CH "/ic/esp32/serialdata/uno/"

WiFiClient wifiClient;

PubSubClient client(wifiClient);

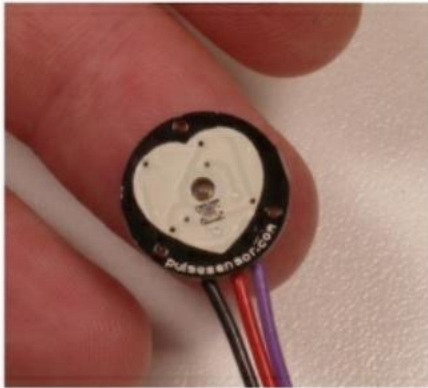
void setup_wifi()
{
    delay(20);
    Serial.println();
    Serial.print("We are connecting to WiFi Network");
}

```

## 第十一章 BLE SERVER AND CLIENT COMMUNICATION

脉冲传感器，如下所示

## Pulse Sensor Getting Started Guide



我们将学习以下如何建立一个 BLE 连接两个 ESP32 一套服务器

我们需要首先先导入这些 BLE 库

```
A connect handler associated with the se
every couple of seconds.
*/
#include <BLEDevice.h>
#include <BLEServer.h>
#include <BLEUtils.h>
#include <BLE2902.h>

BLEServer* pServer = NULL;
BLECharacteristic* pCharacteristic = NULL;
bool deviceConnected = false;
bool oldDeviceConnected = false;
```

*Next, we are assigning the pulse sensor signal pin as  
PIN34 on the Thing*

```
bool oldDeviceConnected = false;

int PulseSensorPurplePin = 34;
int LED13 = 5;    // The on-board
```

现在设置的代码使我们包括自定义特性的通知属性，通知属性使我们能够将消息从服务器发送到客户端设备

```
void setup() {  
    Serial.begin(115200);  
  
    // Create the BLE Device  
    BLEDevice::init("ESP32");  
  
    // Create the BLE Server  
    pServer = BLEDevice::createServer();  
    pServer->setCallbacks(new MyServerCallbacks());  
  
    // Create the BLE Service  
    BLEService *pService = pServer->createService(SERVICE_UUID);  
  
    // Create a BLE Characteristic  
    pCharacteristic = pService->createCharacteristic(  
        CHARACTERISTIC_UUID,  
        BLECharacteristic::PROPERTY_READ |  
        BLECharacteristic::PROPERTY_WRITE |  
        BLECharacteristic::PROPERTY_NOTIFY
```

```
class MyServerCallbacks: public BLEServerCallbacks {  
    void onConnect(BLEServer* pServer) {  
        deviceConnected = true;  
    };  
  
    void onDisconnect(BLEServer* pServer) {  
        deviceConnected = false;  
    }  
};
```

We will be using the variable device connected here

通知回调（每当有消息从服务器端传来）*This is used to print the UUID of the server characteristic*



```
static void notifyCallback(
    BLERemoteCharacteristic* pBLERemoteCharacteristic,
    uint8_t* pData,
    size_t length,
    bool isNotify) {
    Serial.print("Notify callback for characteristic ");
    Serial.print(pBLERemoteCharacteristic->getUUID().toString().c_str());
    Serial.print(" of data length ");
    Serial.println(length);
    Serial.print("data: ");
    Serial.println((char*)pData);
}
```

这里我们设置客户端回调的连接状态，现在用于连接到服务器设备，这里我们试图获得 BLE 服务器的服务 UUID

```
BLEClient* pClient = BLEDevice::createClient();
Serial.println(" - Created client");
```

```
pClient->setClientCallbacks(new MyClientCallback());
```

```
// Connect to the remote BLE Server.
```

如果没有可用的服务 UUID，它将断开与服务器的连接并返回，如果有可用的服务，它将打印此消息

```
// Obtain a reference to the service we are after in the
BLERemoteService* pRemoteService = pClient->getService(
if (pRemoteService == nullptr) {
    Serial.print("Failed to find our service UUID: ");
    Serial.println(serviceUUID.toString().c_str());
    pClient->disconnect();
    return false;
}
Serial.println(" - Found our service");
```

读取特征值并将其打印为字符串值

```
Serial.print("The characteristic v
Serial.println(value.c_str());
```

如果在服务器端启用了，通知回调将被执行

```
if(pRemoteCharacteristic->canNotify())  
    pRemoteCharacteristic->registerForNotify(notifyCallback);
```

## 第十二章 ESTABLISHING WI-FI CONNECTION WITH THE THING

连接到 WiFi 网络的代码

```
Wifi  
#include <WiFi.h>  
  
const char* ssid    = "makerdemy1";  
const char* password = "india123";  
  
void setup() {  
    Serial.begin(115200);    // set the LED pin mode  
  
    delay(10);  
  
    // We start by connecting to a WiFi network  
  
    Serial.println();  
    Serial.println();  
    Serial.print("Connecting to ");  
    Serial.println(ssid);  
  
    WiFi.begin(ssid, password);  
  
    while (WiFi.status() != WL_CONNECTED) {  
        delay(500);  
    }
```

首先，我们必须导入 wifi 库来使用与 WiFi 库相关的各种内置功能  
将串行波特率设置为 115200

*Here we are printing the SSID to the Serial monitor*

Next，我们有 WiFiBegin 功能，它用于用代码前面指定的 SSID 和密码连接到网络  
用于检测 WiFi 是否连接，一旦连接到网络，我们将打印出此消息和分配给 ESP32 的 IP 地址

```
Serial.println(ssid);  
  
WiFi.begin(ssid, password);  
  
while (WiFi.status() != WL_CONNECTED) {  
    delay(500);  
    Serial.print(".");  
}
```

通过浏览器使我们的 LED 闪烁  
我们将放置一个 HTTP 请求通过浏览器  
将 WiFi 库和 WiFi 凭证放在这里

```
#include <WiFi.h>

const char* ssid      = "makerdemy1";
const char* password = "india123";

WiFiServer server(80);

void setup()
{
    Serial.begin(115200);
    pinMode(5, OUTPUT);    // set the LED pin mode

    delay(10);

    // We start by connecting to a WiFi network

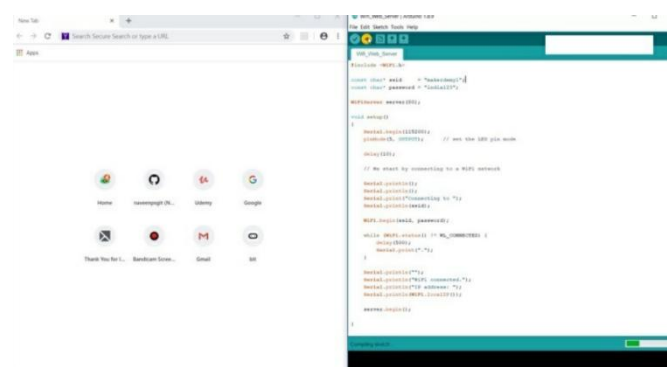
    Serial.println();
    Serial.println();
    Serial.print("Connecting to ");
```

这里我们设置了板载 LED，即 PIN5 作为输出，代码的其他部分和我们使用的 WiFi 连接的差不多

开启服务器侦听

```
Serial.println("IP address:");
Serial.println(WiFi.localIP(
server.begin());
```

现在要了解循环代码，让我们首先上传代码并打开串行监视器  
确保和电脑连接到同一个 WiFi



如果字符串长度为零则说明请求已经结束

```
// at the current time as reading, you get  
// that's the end of the client HTTP request  
if (currentLine.length() == 0) {  
    // HTTP headers always start with a res  
    // and a content-type so the client knows  
    client.println("HTTP/1.1 200 OK");  
    client.println("Content-type:text/html");  
    client.println();
```

断开客户端的连接以停止连接

```
// close the connection:  
client.stop();  
Serial.println("Client Disconnected.");
```

## 第十三章 UNDERSTANDING RSSI AND MEASURING SIGNAL STRENGTH

进行对 RSSI 的了解与学习



RSSI 通常用来比较信号强度，通常被转换为一个标准的单位 kn 作为 dBm

$$S_{\text{dBm}} = 10 \log_{10} \frac{P}{1 \text{ mW}}$$

$$0_{\text{dBm}} = 10 \log_{10} \frac{1 \text{ mW}}{1 \text{ mW}}$$

大于 1mW 的信号具有正的 dBm 值，而弱于 1mW 的信号具有负的 dBm 值。WiFi 网络中通常接收到的信号功率小于 1mW，因此 dBm 值通常从 0dBm 到-100dBm

$$1\text{mW} = 0.001\text{W} = 0\text{ dBm}$$

1mW ↑  
+ve dBm

1mW ↓  
-ve dBm

<1mW  
0 dBm to -100 dBm

测量 Wi-Fi 网络的 RSSI 值的代码

```
RSSI$
#include <WiFi.h>

const char* ssid      = "makerdemy1";
const char* password = "india123";

void setup()
{
    Serial.begin(115200);    // set the LED pin mode

    delay(10);

    // We start by connecting to a WiFi network

    Serial.println();
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);

    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
```

现在 Wifi 库包含了一个叫做 WiFiRSSI 的有用函数，这个函数获得到路由器的连接的 RSSI 值，每隔半秒打印一次

```
// print the received signal strength:
long rssi = WiFi.RSSI();
Serial.print("RSSI:");
Serial.println(rssi);
delay(500);
}
```

```
RSSI:-50
RSSI:-53
RSSI:-51
RSSI:-49
RSSI:-46
RSSI:-43
```

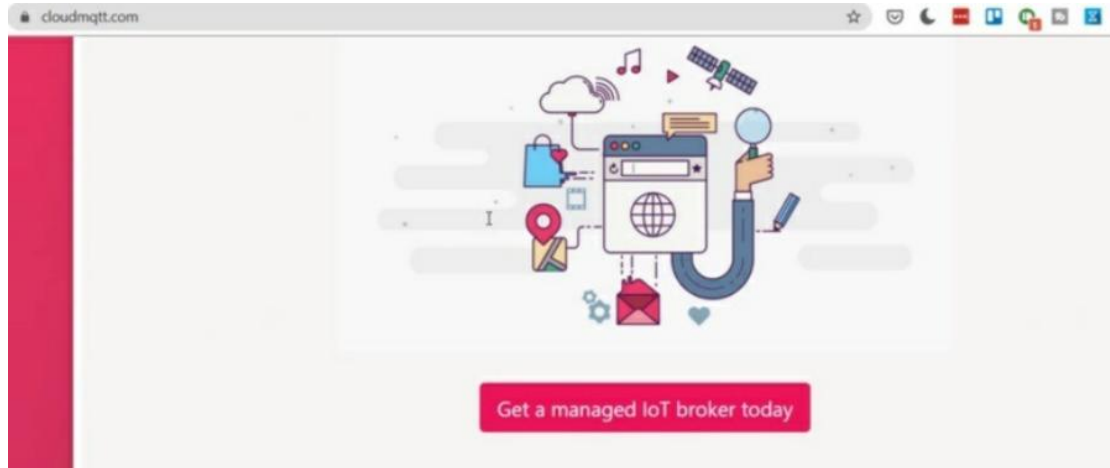
通常，RSSI 低于-90 被认为不可用

如果连接手机热点的话，则 RSSI 明显与手机距离有关系

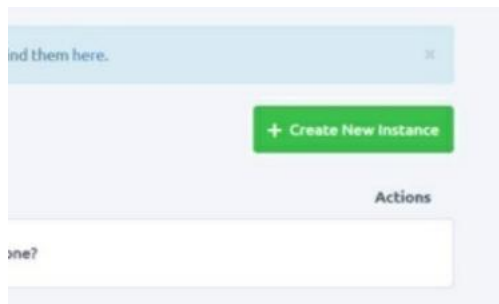


## 第十四章 CREATE MQTT SERVER ACCOUNT

*Now let's create a new account in the cloud.*



这里有很多计划可供选择，可以根据自身需要来作出选择  
点击绿色按钮创建



然后输入项目名称，选择项目类型计划

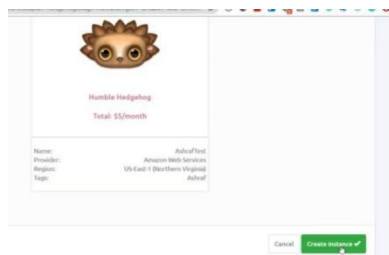
Select a plan and name - Step 1 of 4

Name	<input type="text" value="AshrafTest"/>	Plan
Plan	<input type="text" value="Humble Hedgehog (\$5/month)"/>	
Tags	<input type="text" value="Ashraf"/>	

Tags are used to separate your instances between projects. This is primarily used in the project listing view for easier navigation and access control.

Tags allow admins to manage team members access to different groups of instances.

点击绿色部分



创建成功

# 第十五章 UPLOAD CODE TO ARDUINO AND TEST IT


首先将代码上传到板子上

```
#include <SoftwareSerial.h>
int sensorid = 1;
SoftwareSerial sw(2,3); // Rx and Tx

void setup() {
    // put your setup code here, to run once:
    Serial.begin(115200);
    Serial.println("Project for Interfacing Arduino with ESP32");
    sw.begin(115200);
}

void loop() {
    // put your main code here, to run repeatedly:
    Serial.println("Send Data to ESP32 Board");
    int adcValue = analogRead(A0);
    Serial.print("{\"sensorid\":\"");    // (\ sensor data and sensor id)
    Serial.print(sensorid);
    Serial.print(",");
    Serial.print("\"adcValue\":\"");
    Serial.print(adcValue);
    Serial.print("\");
    Serial.println();
}
```

TitISpy 是一个开源实用程序，旨在帮助监控 *Kutty topics* 上的活动，可以从 GitHub 上加载它。

 **mqtt-spy**

an open source utility intended to help you with monitoring activity on MQTT topics

mqtt-spy is an open source utility intended to help you with monitoring activity on MQTT topics. It's been designed to deal with high volumes of messages, as well as occasional publications.

mqtt-spy is a JavaFX application, so in theory should work on any operating system with an appropriate version of Java 8 installed.

mqtt-spy-daemon is a Java-based command line tool that does not require a GUI environment. Basic functionality works with Java 7, whereas some of the advanced features like scripting require Java 8 to be installed.

For more information on the available functionality see the project's wiki at <https://github.com/eclipse/paho.mqtt-spy/wiki>.

If you find mqtt-spy helpful in your day to day work, please consider a donation at <http://www.justgiving.com/mqtt-spy>. All money goes to UNICEF.

For latest mqtt-spy and mqtt-spy-daemon downloads, go to <https://github.com/eclipse/paho.mqtt-spy/wiki/Downloads>.

他有很多版本，你可以选择其中任意一个

## Downloads

Kamil Baczkowicz edited this page on Jul 7, 2017 · 3 revisions

### mqtt-spy

Compatible with JRE/JDK 8u60 and above:

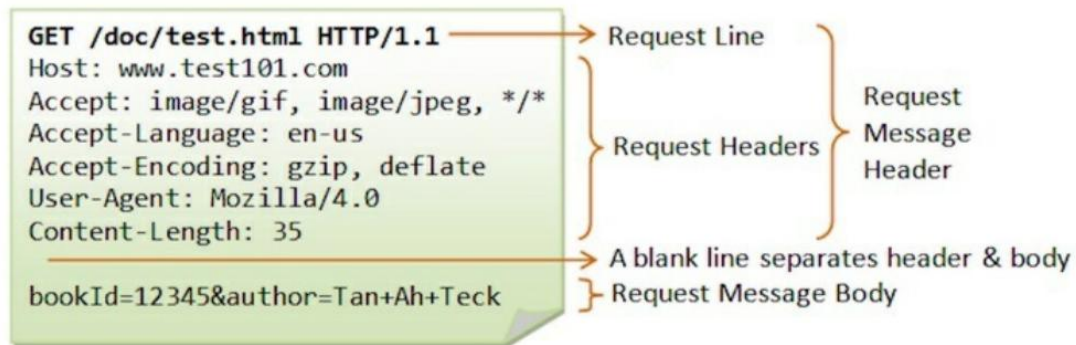
- 0.5.4 - [GitHub download](#) ([release note](#))
- 1.0.0 - [GitHub download](#) ([release note](#))

### mqtt-spy-daemon

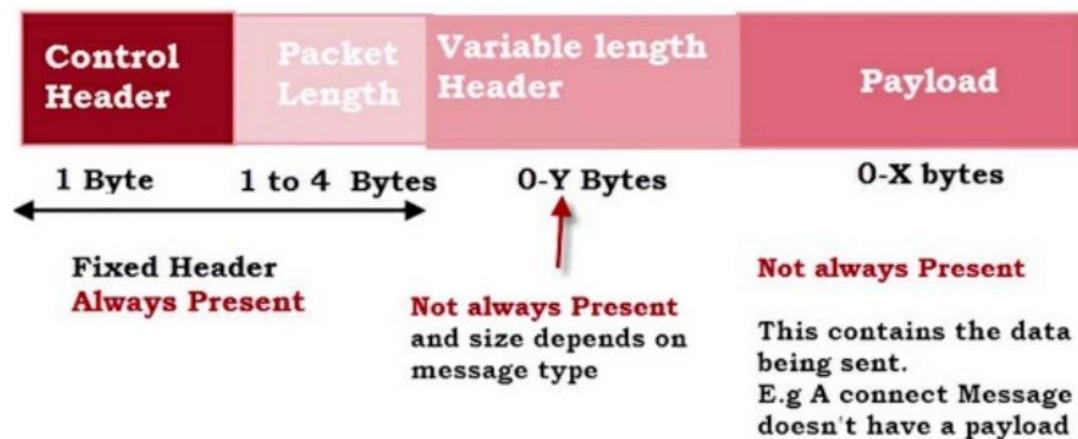
- 0.1.0 - [GitHub download](#) ([release note](#))
- 1.0.0 - [GitHub download](#) ([release note](#))

## 第十六章 INTRODUCTION TO MQTT

与 HTTP 相比, MQTT 是一个轻量级协议, 这意味着 MQTT 消息包大小比 HTTP 低得多。在最轻的时候, 最小的 MQTT 包大小只有 2 字节



不像 HTTP 在类似文档的结构中传输消息, MQTT 消息以字节数组的形式传输

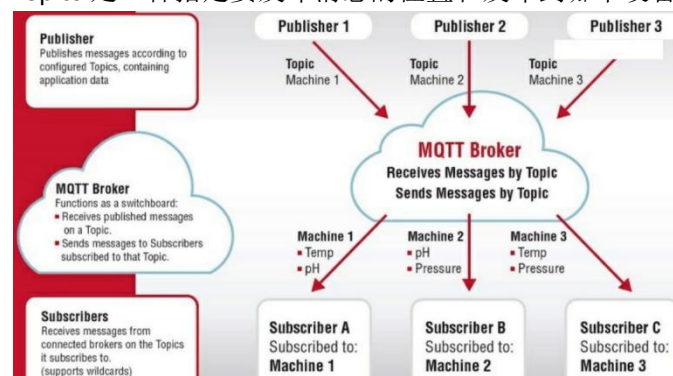


这使得 MQTT 非常适合于低带宽网络, 因为它降低了在受限网络上传输的数据量  
MQTT 适用于发布/订阅模型, 而 HTTP 则适用于请求/响应模型

发布/订阅模型工作原理

客户端设备可以将消息发布到一个 topic 里, 消息是包含实际数据的内容

Topics 是一种指定要发布消息的位置和发布到哪个设备的方法



同时要创建 **topic** 级别来使消息发送的目的地更加准确

如果另一个客户端设备想要接收已发布的消息，它将需要订阅已发布消息的特定主题  
为了管理正在传递给订阅的客户端的所有主题和消息，我们需要使用 **MQTT** 代理

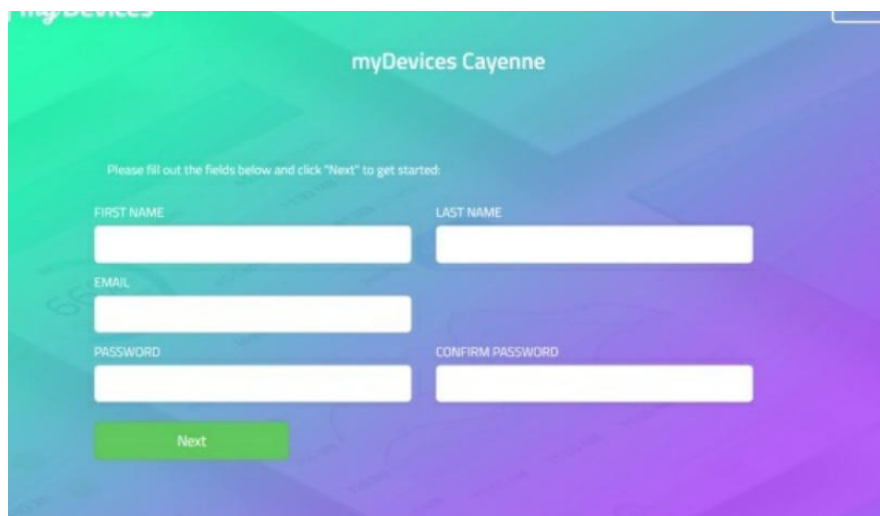
## 第十七章 INTERFACING THE THING TO CAYENNE

我们将在这节了解

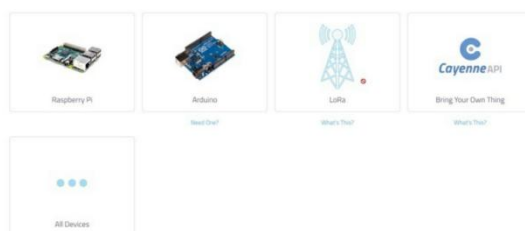
1、*How to add the Thing to cayenne using the Cayenne MQTT API*

2、*How to write data to a Channel on the Cayenne Dashboard*

首先要创建一个账户

The image shows the registration page for myDevices Cayenne. The page has a green and blue gradient background. At the top, it says "myDevices Cayenne". Below that, it says "Please fill out the fields below and click 'Next' to get started:". There are four input fields: "FIRST NAME", "LAST NAME", "EMAIL", and "PASSWORD". There is also a "CONFIRM PASSWORD" field. A green "Next" button is at the bottom.

填写以上信息进行注册，然后返回登录界面重新登陆即可进入  
在这里选择接口和设备



但是你在这里找不到 *esp32*，你需要手动添加

MQTT USERNAME: d53c20d0-2f9f-11e9-b96f-c12a468aadc

MQTT PASSWORD: e87696fb361d5ba8ca9eca39bb9984bcb7d0e5d

CLIENT ID: f73e88d0-5733-11e9-b1a2-d1dd4219210

MQTT SERVER: mqtt.mydevices.com

MQTT PORT: 1883

NAME YOUR DEVICE (optional): My\_ESP32

Waiting for board to connect...

如何安装卡宴 ESPMQTT 库？

首先进入工具-----管理库，搜索 *cayenne*，找到后安装最新版本然后连接就好了

```
#define CAYENNE_DEBUG
#define CAYENNE_PRINT Serial
#include <CayenneMQTTESP32.h>

// WiFi network info.
```

虚拟写函数的一般参数格式如下

Cayenne Community

HOME PROJECTS DOCS SUPPORTED HARDWARE SIGN IN GET STARTED

Data types for Cayenne MQTT API

The Library

virtualWrite(5, 1, "digital\_sensor", "d")

Data Type	Type Value	Unit	
Digital Actuator	digital_actuator	Digital (0/1)	
Analog Actuator	analog_actuator	Analog	
Digital Sensor	digital_sensor	Digital (0/1)	
Analog Sensor	analog_sensor	Analog	
Acceleration	accel	Acceleration	
Barometric pressure	bp	Pascal	
Barometric pressure	bp	Hectopascal	
Battery	batt	% (0 to 100)	
Battery	batt	Ratio	
Battery	batt	Volts	
Carbon Dioxide	co2	Parts per million	
Carbon Monoxide	co	Parts per million	
Counter	counter	Analog	
Current	current	Ampere	
Current	current	Milliampere	
Energy	energy	Kilowatt Hour	
External Waterleak	ext_leak	Analog	
Frequency	freq	Hertz	
Gyroscope	g	Rotation per minute	
Gyroscope	g	Degree per second	
Humidity	humidity	Humidity	

12



# 第十八章 SETTING THE MESSAGE RATE

## AND CREATING A CUSTOM WIDGET

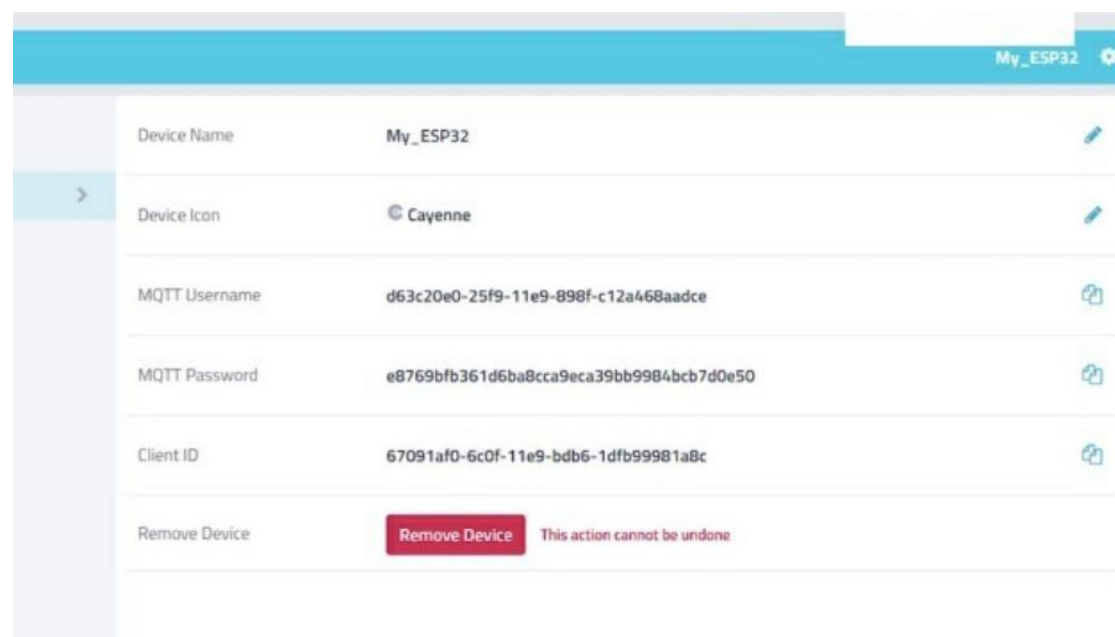
消息速率最高为每分钟 60，让我们来对其进行调整以适应我们的项目

```
#define CAYENNE_DEBUG
#define CAYENNE_PRINT Serial
#include <CayenneMQTTESP32.h>

// WiFi network info.
char ssid[] = "makerdemy1";
char wifiPassword[] = "india123";

// Cayenne authentication info. This should be obtained from the Cayenne Dashboard.
char username[] = "d63c20e0-25f9-11e9-898f-c12a468aadce";
char password[] = "e8769bfb361d6ba8cca9eca39bb9984bcb7d0e50";
char clientID[] = "12abfec0-6bc2-11e9-bdb6-1dfb99981a8c";
```

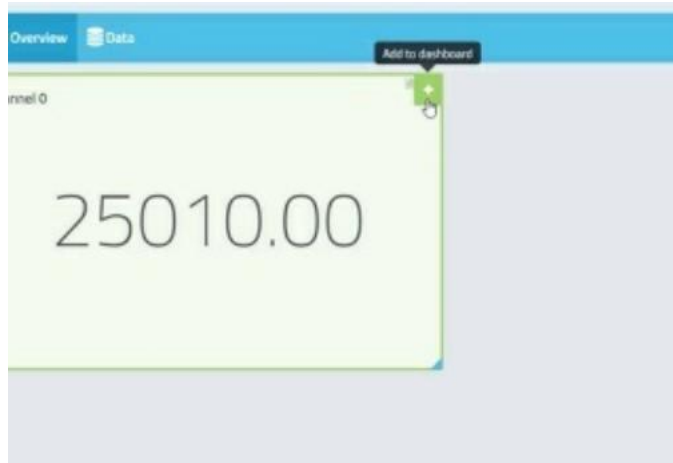
在上传和运行代码之前，进入设备栏点击我的设备并配置，出现以下信息



The screenshot shows the Cayenne dashboard interface for a device named 'My\_ESP32'. The configuration table is as follows:

Field	Value	Action
Device Name	My_ESP32	Edit
Device Icon	Cayenne	Edit
MQTT Username	d63c20e0-25f9-11e9-898f-c12a468aadce	Copy
MQTT Password	e8769bfb361d6ba8cca9eca39bb9984bcb7d0e50	Copy
Client ID	67091af0-6c0f-11e9-bdb6-1dfb99981a8c	Copy
Remove Device	<button>Remove Device</button> This action cannot be undone	

将 MQTT 的用户名、密码和客户端 ID 复制到代码中

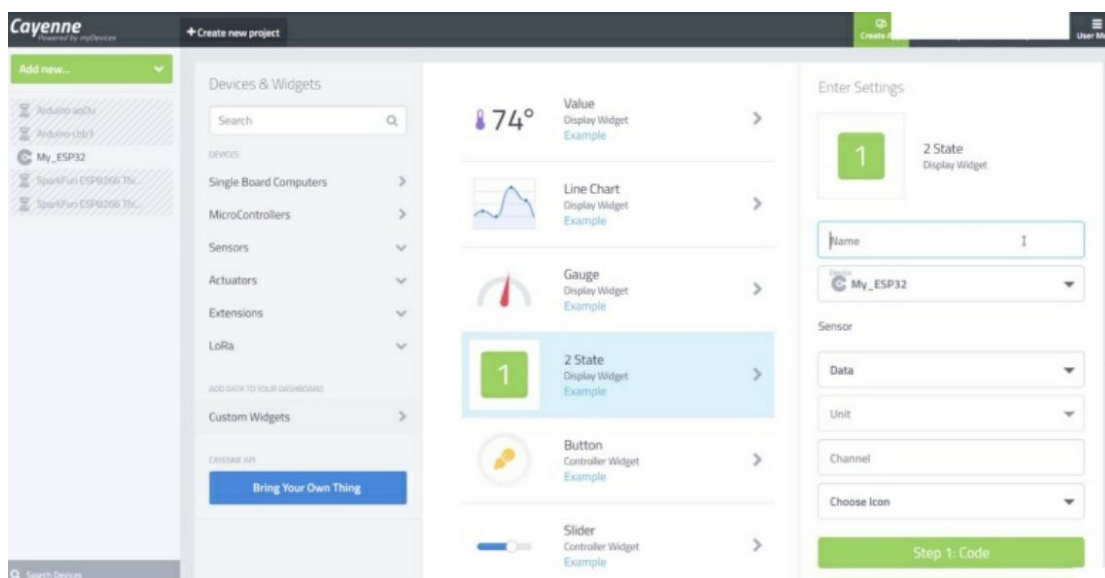


点击这里进入

在这里，您可以获得关于接收消息的时间、接收数据的设备名称、通道号、传感器名称、传感器 ID、数据类型、消息中包含的数据的单位和实际值等信息，如下

Timestamp	Device Name	Channel	Sensor Name	Sensor ID	Data Type	Unit	Values
2019-05-03 11:31:43	My_ESP32	0	Channel 0	e3d7fc40-6d68-11e9-80b6-4...			44043
2019-05-03 11:31:42	My_ESP32	0	Channel 0	e3d7fc40-6d68-11e9-80b6-4...			43041
2019-05-03 11:31:41	My_ESP32	0	Channel 0	e3d7fc40-6d68-11e9-80b6-4...			42039
2019-05-03 11:31:40	My_ESP32	0	Channel 0	e3d7fc40-6d68-11e9-80b6-4...			41037
2019-05-03 11:31:39	My_ESP32	0	Channel 0	e3d7fc40-6d68-11e9-80b6-4...			40036
2019-05-03 11:31:38	My_ESP32	0	Channel 0	e3d7fc40-6d68-11e9-80b6-4...			39034
2019-05-03 11:31:38	My_ESP32	0	Channel 0	e3d7fc40-6d68-11e9-80b6-4...			39035
2019-05-03 11:31:38	My_ESP32	0	Channel 0	e3d7fc40-6d68-11e9-80b6-4...			37032

自定义小部件



我们分配 PIN15 是触摸传感器

```
char password[] = "es/b9d1b361d6ba8cca9eca39bb9984dcd/d0e30";
char clientID[] = "c2f2a2a0-6bd8-11e9-af96-af70e905ba69";
unsigned long lastMillis = 0;
int touch = 15;
int sense;
int state;

void setup() {
  Serial.begin(9600);
  Cayenne.begin(username, password, clientID, ssid, wifiPassword);
}

void loop() {
  Cayenne.loop();
  sense = touchRead(touch);

  if(millis() - lastMillis > 1000) {
    lastMillis = millis();
    if(sense < 10)
    {
      state = 1;
      Serial.println(state);
      Cayenne.virtualWrite(0, state, "digital_sensor", "d");
    }
  }
}
```

现在在循环代码中，我们使用函数从引脚 15 读取模拟值，并将其分配给变量意义

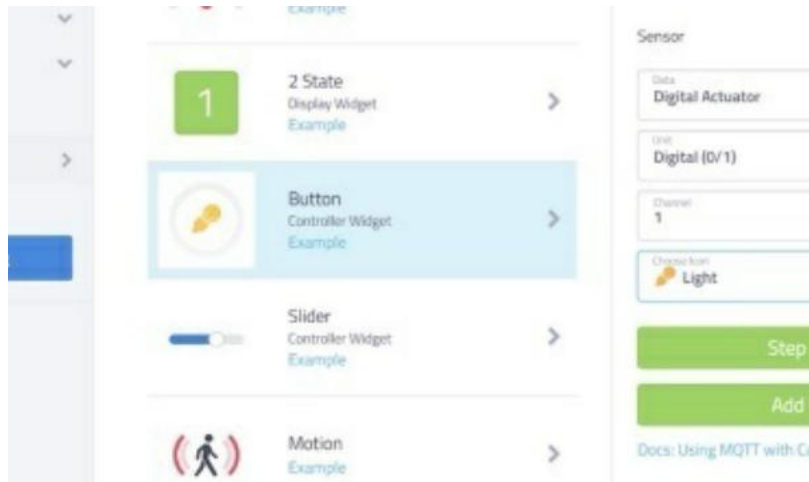
```
if(millis() - lastMillis > 1000) {
  lastMillis = millis();
  if(sense < 10)
  {
    state = 1;
    Serial.println(state);
    Cayenne.virtualWrite(0, state, "digital_sensor", "d");
  }
  else
  {
    state = 0;
    Serial.println(0);
    Cayenne.virtualWrite(0, state, "digital_sensor", "d");
  }
}
```

如果触摸读数低于 10，该状态将被分配为 1

如果触摸针读数超过 10，这意味着没有检测到触摸

## 第十九章 ACTUATING THE ONBOARD

### LED AND USING TRIGGERS



在此处添加小部件，点击 button  
引入 pin5 为 LED

```
#define CAYENNE_DEBUG
#define CAYENNE_PRINT Serial
#include <CayenneMQTTESP32.h>

// WiFi network info.
char ssid[] = "makerdemy1";
char wifiPassword[] = "india123";

// Cayenne authentication info. This should be obtained from the Cayenne Dashboard.
char username[] = "d63c20e0-25f9-11e9-898f-c12a468aadce";
char password[] = "e8769bfb361d6ba8cca9eca39bb9984bcb7d0e50";
char clientID[] = "67091af0-6c0f-11e9-bdb6-1dfb99981a8c";
unsigned long lastMillis = 0;
int touch = 15;
int led = 5;
int sense;
int state;
String button_state;
```

现在我们不在这里更改循环代码，因为我们仍然将使用我们之前创建的触摸小部件

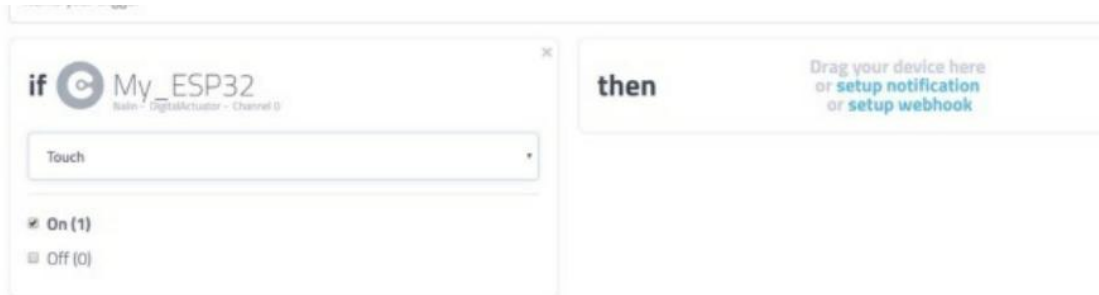
```
void loop() {
  Cayenne.loop();
  sense = touchRead(touch);

  if(millis() - lastMillis > 1000) {
    lastMillis = millis();
    if(sense < 10)
    {
      state = 1;
      Serial.println(state);
      Cayenne.virtualWrite(0, state, "digital_sensor", "d");
    }
  }
}
```

打开串行监视器，并再次单击该小部件，将看到调试消息，其中包括频道号、主题和已收到的值

```
i [4043332] Publish: topic 1, channel 0, value 0, subkey d, key digital_se
i 0
S [4044334] Publish: topic 1, channel 0, value 0, subkey d, key digital_se
0
v [4045338] Publish: topic 1, channel 0, value 0, subkey d, key digital_se
0
[4046342] Publish: topic 1, channel 0, value 0, subkey d, key digital_se
[4046924] Message received: topic 2, channel 1
[4046924] In: value 1, channel 1
[4046924] Channel 1, value 1
[4046924] publishState: topic 1 channel 1
v [4046953] Publish: topic 1, channel 1, value 1, subkey , key
[4047019] Send response: zLJEnEsKEwCEJyI
0
[4047634] Publish: topic 1, channel 0, value 0, subkey d, key digital s
```

If 代表条件，此处设置为触摸，选择 on 打开，然后引发 then



在这里 then 我们引入小灯，即摸到后开灯

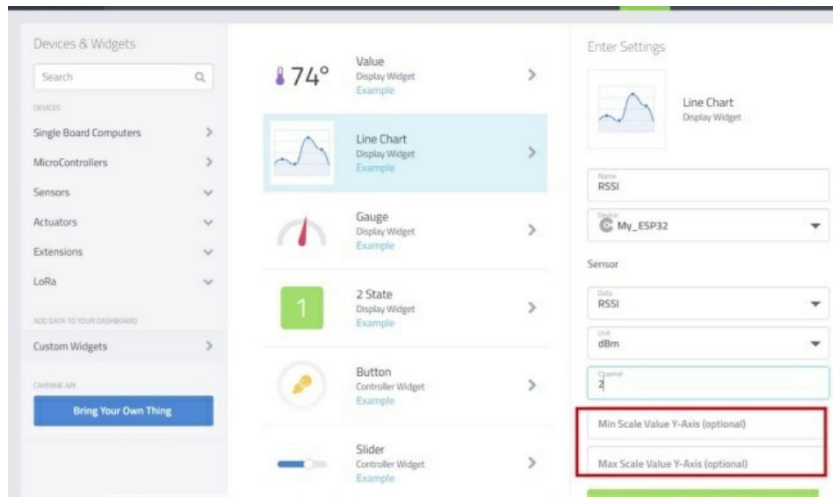


## 第二十章 USING TRIGGER NOTIFICATION AND SCHEDULING

以手机的热点作为接入点来监测信号的强度，如果它下降到某个阈值以下，将执行一封电子邮件通知

同样的，首先引入小部件“值”，做好相关配置





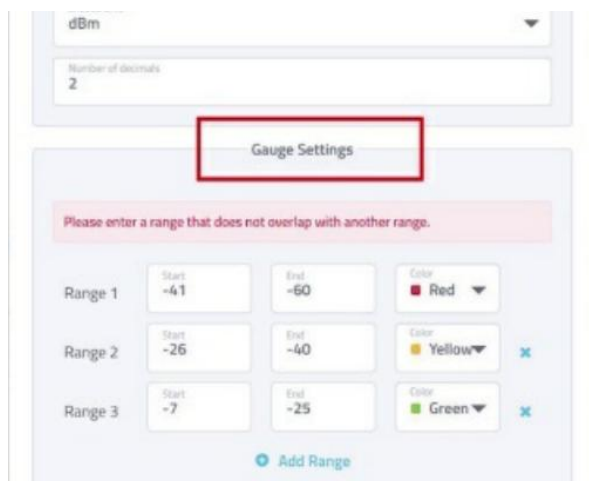
此处需要使用 RSSI 函数，并且要用到手机热点凭证

```
pinMode(led, OUTPUT);

void loop() {
  Cayenne.loop();
  long rssi = WiFi.RSSI();
  Serial.println("RSSI:");
  Serial.println(rssi);
  sense = touchRead(touch);

  if(millis() - lastMillis > 1500) {
```

代码部署后需要在小部件设置里设置三个级别（信号强度）



经过实践可知随着距离变远信号强度低了



现在，设置一个通知触发器如果值低于-45dBm，我们将发送电子邮件  
此处设置触发值

The screenshot shows the configuration for an IFTTT trigger. The sensor is set to 'RSSI'. A slider is positioned at 'null'. Below the slider, the 'Value' field is set to '-45', with 'Min' at '-50' and 'Max' at '500'. The unit is set to 'dBm'. The condition is set to 'Sensor below'.

同理，运用 if、then 完成部署，then 部分可以设置需要发送的消息通知

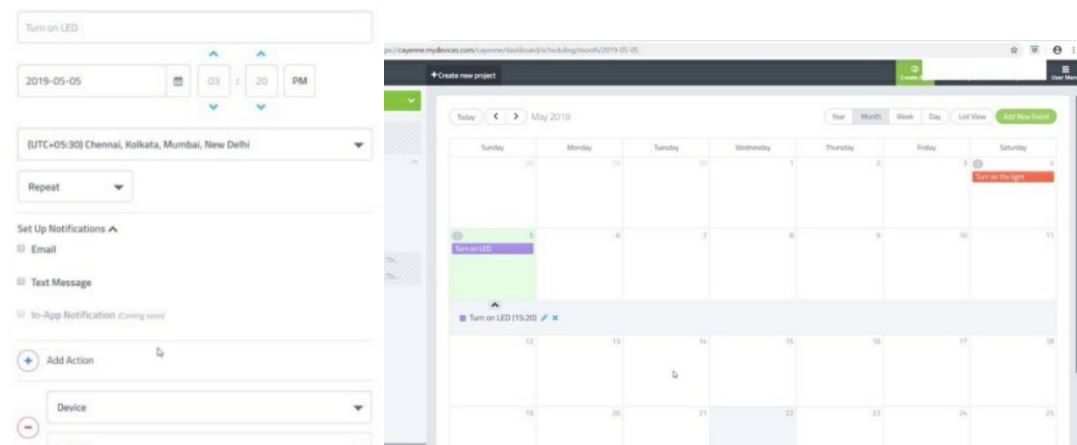
The screenshot shows the 'then' action configuration. The trigger is 'if My\_ESP32 RSSI' with a value of '-45'. The action is 'then notify...'. The notification is set to 'Send Email' to the recipient 'test@test.com'. The 'Add more recipients?' link is visible.

另一个功能----预定活动

输入事件标题，然后选择事件发生的日期，并设置时间

The screenshot shows the 'New Event' form. The event title is 'I'. The date is set to '2019-05-05' and the time is '03:20 PM'. The timezone is set to 'Default'. The 'Repeat' dropdown is set to 'Repeat'. The 'Set Up Notifications' section is expanded, showing 'Add Action'. A 'Save' button is at the bottom right.

例如设置一个时间点触发通知并使 LED 发光



## 第二十一章 INTERFACING PIR SENSORS

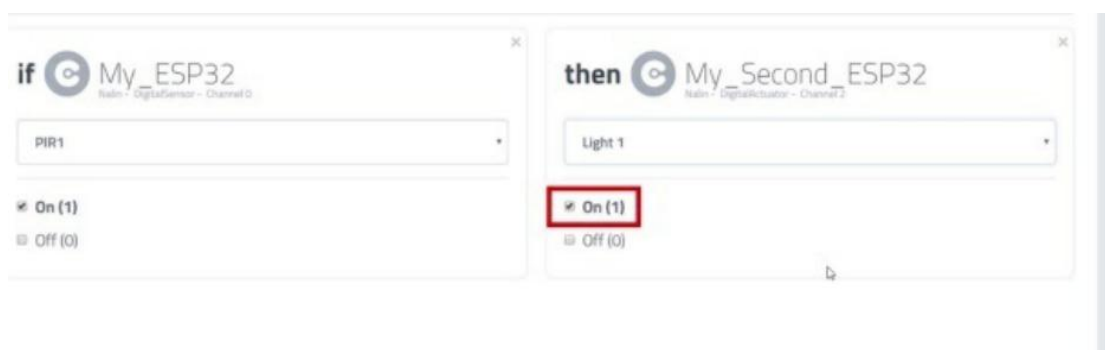
### WITH THE THING

下面进行 *Pir* 传感器的学习

试着做出检测运动的灯光（节电走廊）

传统走廊上所有的灯泡都开着，但对于智能走廊，只有在特定的 *Pir* 传感器附近检测到运动时，灯才会切换，所以当一个人走进走廊时，灯泡只会在项目中接近的人中打开，这意味着当这个人走进走廊时，第一个 *Pir* 传感器将检测运动并打开第一个灯泡当这个人穿过第一个 *Pir* 传感器并现在进入第二个 *Pir* 传感器范围时，第一个灯泡应该关闭，第二个灯泡应该打开，同样，当这个人穿过第二个 *Pir* 传感器的范围时，第二个灯泡应该关闭

引入小部件之后如图所示设置



同理为了达到这个目的，最终设置如下所示



我们要保证在 pir 传感器检测到运动的时候只发送一个信号 1 到服务器，否则会出现多个指令

我们在这里初始化 2 个布尔变量，最后运动 1 和最后运动 2 为 0

```
int pin1 = 25;
int pin2 = 22;
bool lastmotion1 = 0;
bool lastmotion2 = 0;
// Cayenne authentication
char username[] = "d63c20e
```

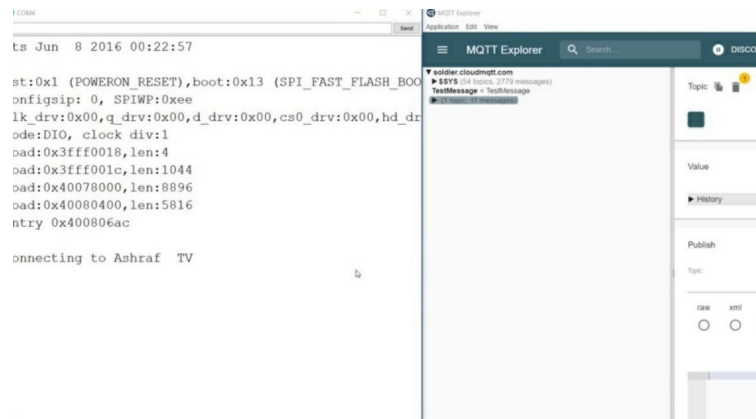
我们不要连续的 1，因此一个 1 前应该是 0

```
if((motion1 == 1) && (lastmotion1 == 0)){
    Serial.println(motion1);
```

## 第二十二章 FINAL ESP32 TESTING

*to make sure that our E. S. P has connected to our Cloud and Kuttly server. This has to dload Kuttly Explorer.*

进入后可以看到我们接收的信息



每当我们进入都会收到一条新的消息

*As you can see now, it's 12 messages. And if I clicked this, again, it will be 13 messages.*

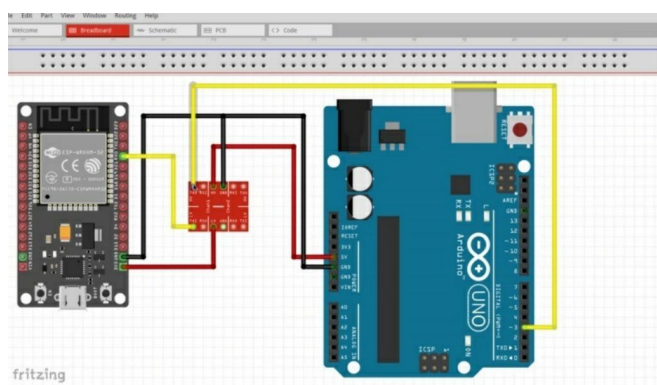


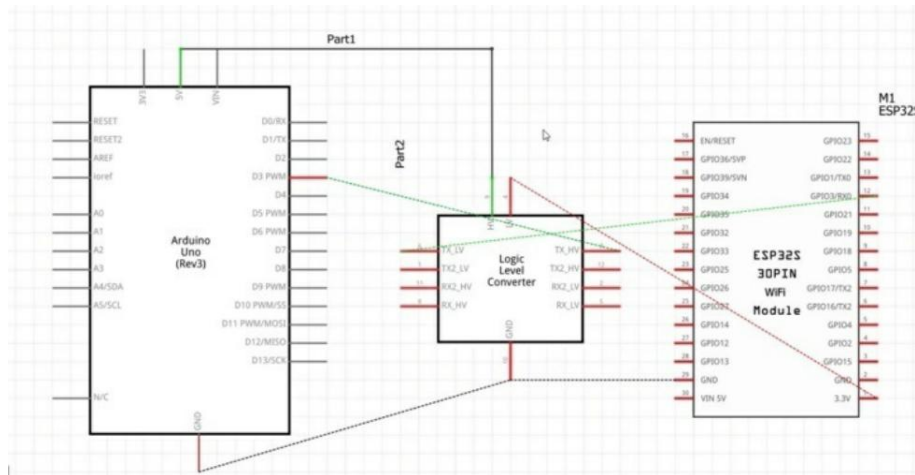
## 第二十三章 CIRCUIT CONNECTIO EXPLAINED

*flighting software*: 一款创建电路设计、为任何电子电路的布线布局的软件

实物图和原理图如下所示

与我们所学的 *pcb* 课程异曲同工，我们可以用 *PCB* 来进行代替





根据原理图进行连接

## 第二十四章 HOOKING UP THE ESP32

### THING TO THE ARDUINO IDE

在 ArduinoIDE 上安装 ESP32 核心有很多编程方法，但最简单和最直观的编程方法之一是通过 ArduinoIDE

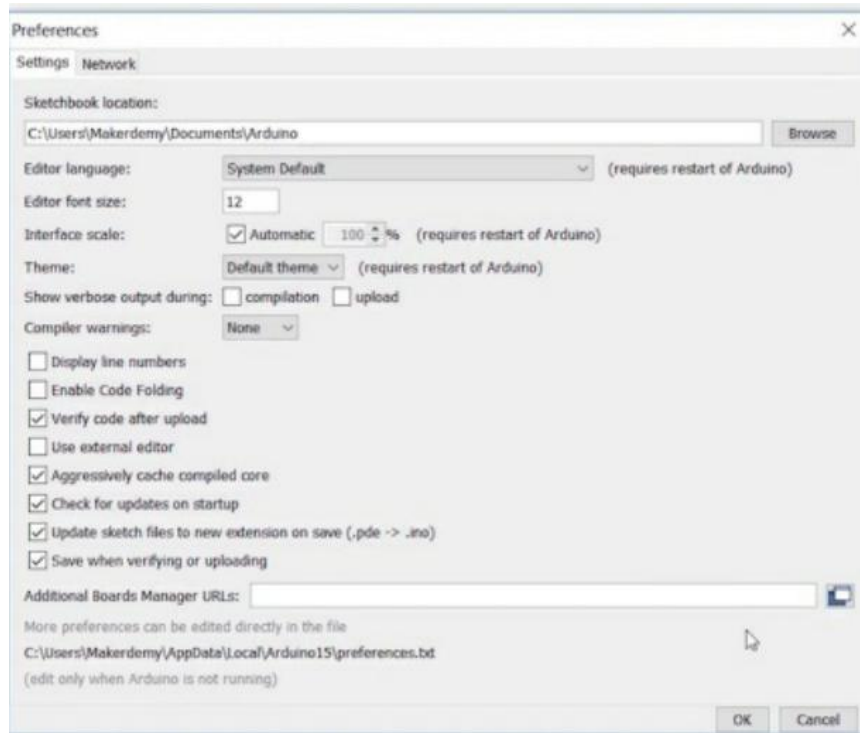
```
void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```



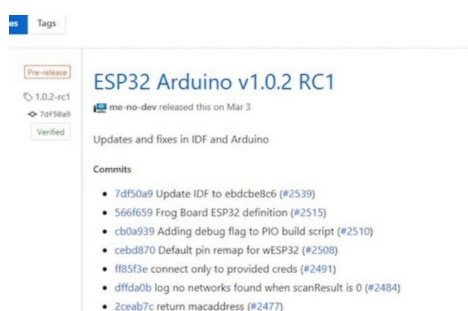
打开后是开发环境本身，然后去工具里找“板”部分  
选择 Arduino 板管理器，将会看到很多种板子  
但是我们找不到 ESP32，我们要另外为他安装进来

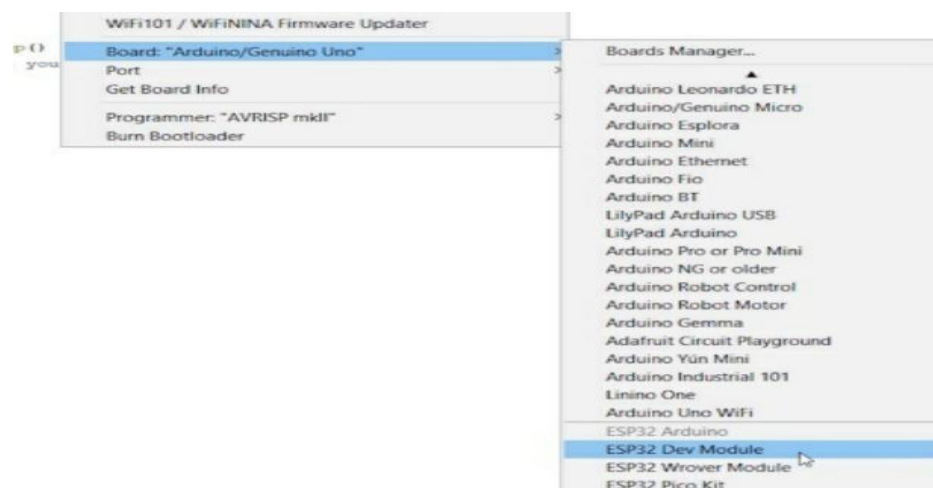
选择文件选项，找到 *preferences*



*You will find a field called the Additional Board Manager URLs. The field needs a specific type of file*  
这是一个以 *JSON* 格式编写的文件，需要在字段中输入，以便  
其他板子管理器获取第三方核心的列表

可以在资源部分找到链接，复制链接并粘贴到字段中，点击 *ok*  
然后我们就可以找到我们的 *esp32* 开发模块  
我们可以下载最新的版本





进入编程，首先介绍两个函数

第一个是设置函数，设置函数用于初始化变量、库、声明引脚模式等。

第二个是循环函数，这是编写实际代码的地方，这将运行多次。

举一个例子，使 LED 闪烁，如下

```
int led = 5;

void setup() {
  // initialize digital pin led as an output.
  pinMode(led, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);             // wait for a second
  digitalWrite(led, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);             // wait for a second
}
```

## 第二十五章 WORKING WITH THE ONBOARD SENSORS ON THE THING

Esp32 板载温度传感器

温度传感器的温度范围为-40 摄氏度到 125 摄氏度，但是绝对温度并不真确，这是因为和厂家生产有关系

这里的温度传感器仅仅用来显示板子自身温度，并不可以用于测量外界温度，这对于板子正常工作也是非常重要的，可以时刻监视温差是否正常，是否可以正常工作

打开串行监视器可以看到温度显示

```
const char *pass = "india123";

WiFiClient client;

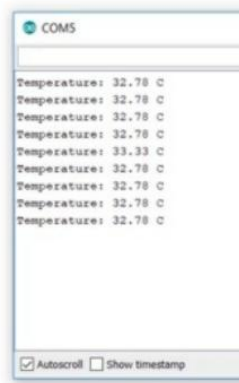
void setup() {
  Serial.begin(115200);
  delay(10);

  Serial.println("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, pass);

  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
}

void loop() {
```



## 霍尔传感器

霍尔效应传感器用于确定磁场的方向和强度，它可用于接近传感应用

这是测量霍尔效应传感器读数的代码，我们将波特率设置高一点，越高传输越快

```
void setup() {
  Serial.begin(115200);
}

void loop() {

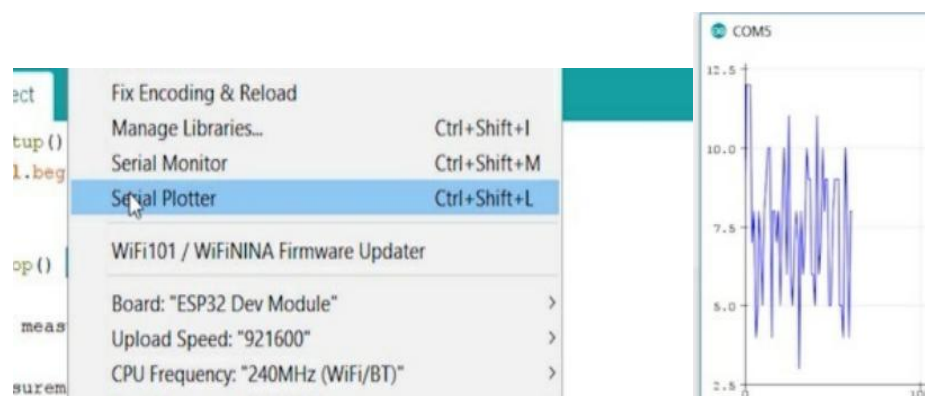
  int measurement = 0;

  measurement = hallRead();

  Serial.print("Hall sensor measurement: ");
  Serial.println(measurement);

  delay(100);
}
```

下面使用一个新工具来显示数据，串行绘图仪，效果如下

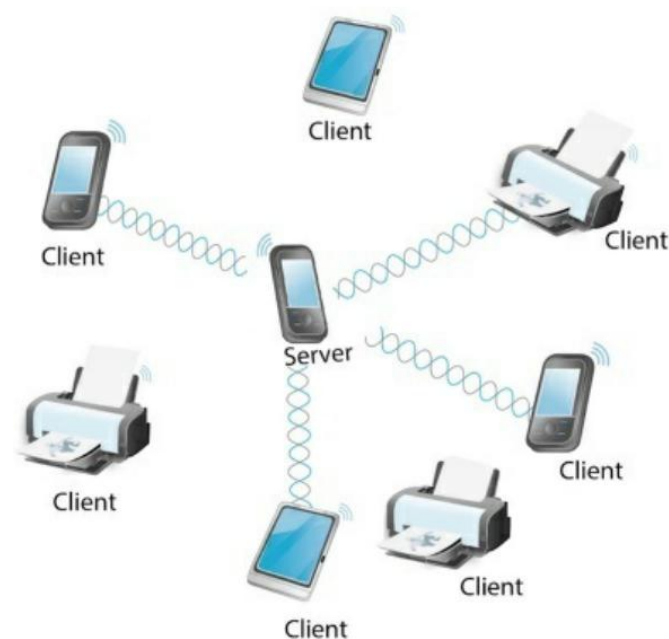


## 第二十六章 UNDERSTANDING BLUETOOTH

### LOW ENERGY AND WIFI

蓝牙是一种通信标准，它运行在 2.4 千兆赫的无线电频谱中，以及其他协议，如 ZigBee，Wi-Fi，BLE 等，这意味着蓝牙设备之间的数据传输在 2.40 到 2.48 千兆赫

蓝牙使用主从服务器架构，其中主服务器可以在一个网络中与多达 7 个副服务器进行通信



低能耗蓝牙设备应用很多，电子设备，智能家居等等

Wi-Fi 是一种无线局域网或无线局域网连接，而蓝牙是一种 WPAN 或无线个人局域网，因此 Wi-fi 比蓝牙具有更大的覆盖范围，因此当需要使用更大范围、更大带宽时选择 WiFi

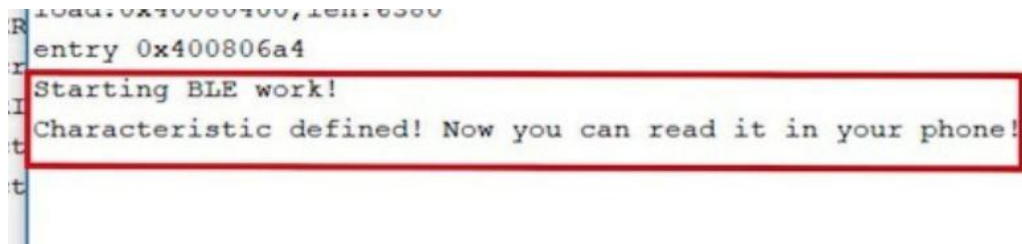
## 第二十七章 ESTABLISHING BLE

### CONNECTION WITH THE THING

首先下载这个 *APP*，打开后可以发现附近可连接的设备



在 esp32 资源部份找到打开蓝牙服务器的代码，然后打开串行监视器



然后到移动端选择连接即可



打开自定义服务，我们可以发现有一个读写属性，这使我们能够从 BLE 设备读取数据



代码部分  
首先导入库

```
#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEServer.h>
```

*Next, we define two UUIDs or Universally Unique Identifiers*

到下面的网站生成 UUID

See the following for generating UUIDs:  
<https://www.uuidgenerator.net/>

设置串行传输波特率 115200

```
void setup() {
  Serial.begin(115200);
  Serial.println("Starting
```

初始化设备名称可以改变它为任何你想要的，这个名字将显示在程序上  
创建两个属性，这将使我们能够与应用程序交互。

第一个是读取属性

1914B



作为智能手机的应用程序，以读取可用的数据

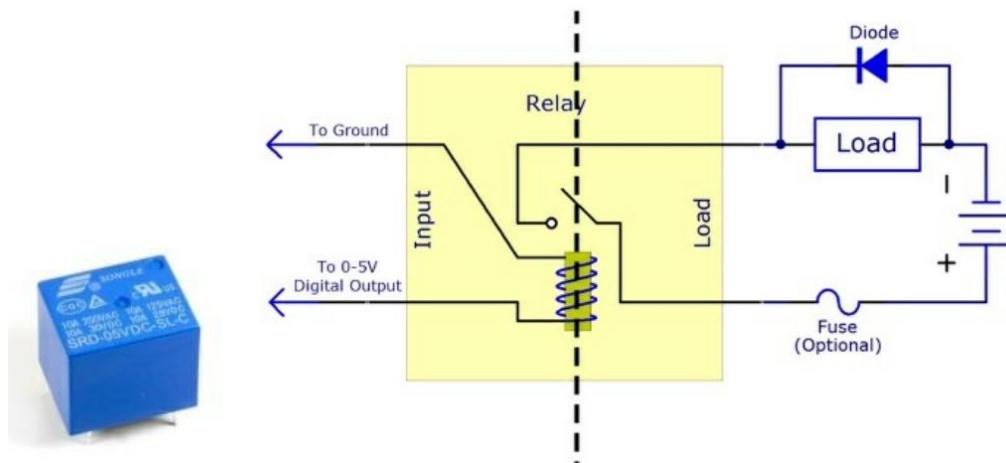
下一个属性是 **write** 属性，你可以在这里输入任何你想要的数据，当你读取它时，它将在应用程序上显示为一个值

```
pCharacteristic->setValue("I am right here!");
pService->start();
// BLEAdvertising *pAdvertising = pServer->getAdvertising();
BLEAdvertising *pAdvertising = BLEDevice::getAdvertising();
```



## 第二十八章 INTERFACING A RELAY WITH THE THING

继电器一般被用于利用低压电路来控制高压电路



额定功率在 250V 和 125V 交流 10 安培，在 30V 和 28V 直流 10 安培

我们需要将高功率电路与低功率电路隔离，因此，我们需要使用一个可选的耦合器电路存在一个模块满足条件



```
int relayPin1 = 23;  
int relayPin2 = 19;
```

引脚连接

初始化两个引脚作为输出，设置 PIN23 为高，因此第一个继电器将不会被激活，灯泡将被关闭，在延迟 4 秒后将 PIN19 设置为高值，因此第二个继电器将不会被激活，第二个灯泡也将

被关闭，在延迟 4 秒后，将 PIN23 设置为 LOW，激活第一个继电器并打开灯泡，对 PIN19 同理，在 4 秒的延迟后打开第二个灯泡

```
int relayPin1 = 23;
int relayPin2 = 19;

void setup() {
  pinMode(relayPin1, OUTPUT);
  pinMode(relayPin2, OUTPUT);
}

void loop() {
  digitalWrite(relayPin1, HIGH);
  delay(4000);
  digitalWrite(relayPin2, HIGH);
  delay(4000);
  digitalWrite(relayPin1, LOW);
  delay(4000);
  digitalWrite(relayPin2, LOW);
  delay(4000);
}
```

## 第二十九章 INTERFACING THE RELAY

### SETUP WITH CAYENNE AND CREATING A PROJECT

*We will learn the following How to control the Relay setup from the Cayenne dashboard.*

首先分配指针

```
char wifiPassword[] = "inc
int relayPin1 = 23;
int relayPin2 = 19;
String state1;
String state2;
```

继电器引脚 1 和 2 作为输出

```
Cayenne.begin(username, password, cl:  
  pinMode(relayPin1, OUTPUT);  
  pinMode(relayPin2, OUTPUT);  
}
```

分别使用两个 *Cayenne* 来控制两个继电器

我们将使用状态 1 来代表我们用于控制继电器 1 的按钮小部件的状态

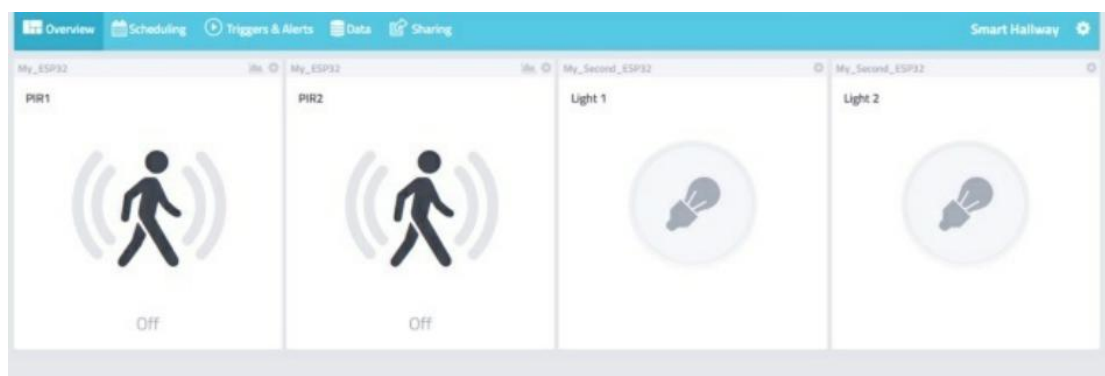
我们将继电器 pin1 设置为低值，以激活继电器 1，从而打开灯泡 1

同理设置其他引脚

```
CAYENNE_IN(2)  
{  
  CAYENNE_LOG("Channel %u, value %s", request.channel, getValue.asString());  
  //ss message here. If there is an error set an error message using getValue.setError(), e.g getValue.setError("Error message");  
  state1 = getValue.asString();  
  
  if(state1 == "1")  
  {  
    digitalWrite(relayPin1, LOW);  
  }  
  else  
  {  
    digitalWrite(relayPin1, HIGH);  
  }  
}  
  
CAYENNE_IN(3)  
{  
  CAYENNE_LOG("Channel %u, value %s", request.channel, getValue.asString());  
  //ss message here. If there is an error set an error message using getValue.setError(), e.g getValue.setError("Error message");  
  
  state2 = getValue.asString();  
  
  if(state2 == "1")  
  {  
    digitalWrite(relayPin2, LOW);  
  }  
  else  
  {  
    digitalWrite(relayPin2, HIGH);  
  }  
}
```

最后效果就是按下一次第一个灯亮，再按一次第一个灭第二个灯亮

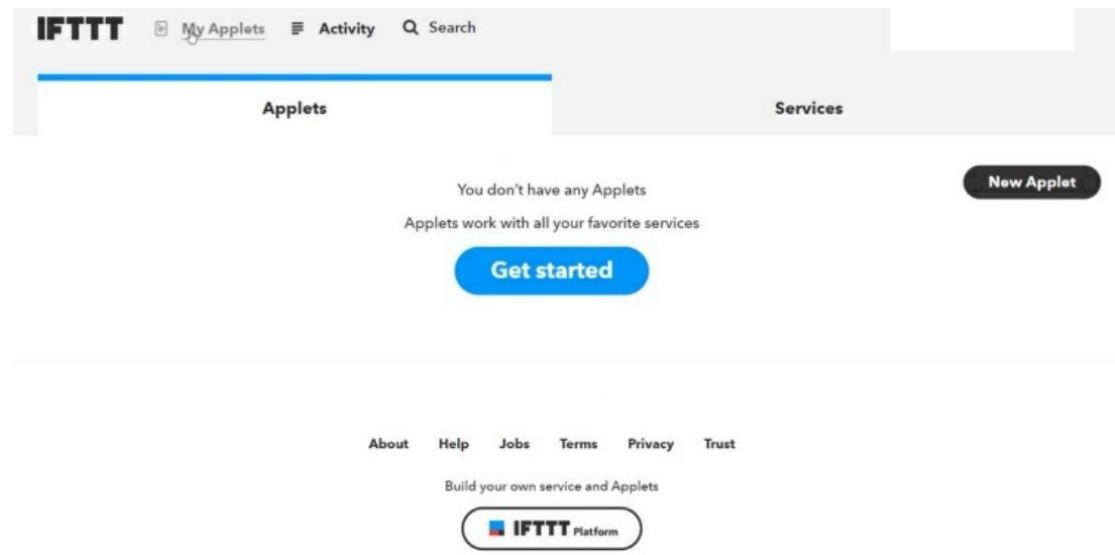
*Cayenne* 拥有一个“项目”功能，它允许将不同设备上的小部件组合到一起，这样统筹起来会很方便



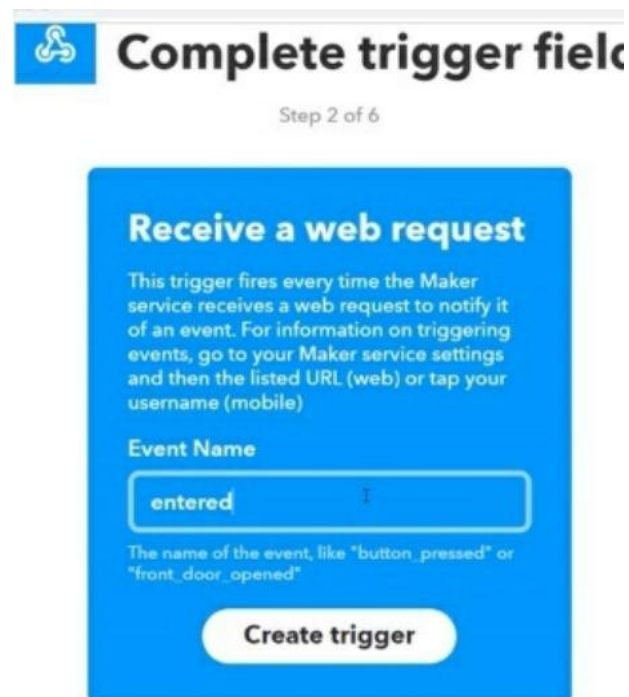
## 第三十章 SETTING UP NOTIFICATION

### SMS USING IFTTT

首先找到 IFTTT 网站注册登录



在这里有两个选项卡小程序和服务，这将找到我们创建的所有小程序



为自己创造一个名字后进入选择触发条

件，我们的项目是想要发送短信



## Connect Android SMS

Step 3 of 6

Android SMS is a native service that allows you to receive Short Message Service (SMS) messages on your device and send messages to other phone numbers. Standard carrier rates may apply. This service requires the IFTTT app for Android.

Connect

因此搜寻短信服务

然后输入电话号码和短信内容

### Send an SMS

This Action will send an SMS from your Android device to any phone number you specify.

**Phone number**

Include country code e.g. 12024561111 **Add ingredient**

**Message**

The event named "**EventName**" occurred on the Maker service

**Add ingredient**

**Create action**

可以在 [Webhook](#) 文档中使用的 Webhook 服务，您将找到发出 POST 请求所需的 URL

Your key is: **cFKjJQ7Mqj9\_beVecHpIBu**

[Back to service](#)

**To trigger an Event**

Make a POST or GET web request to:

继续设置

Name your trigger

if My\_ESP32

PIR1

On (1)  
Off (0)

then webhook

URL

GET

Emoji Win+Period

Undo Ctrl+Z

Redo Ctrl+Shift+Z

Cut Ctrl+R

Copy Ctrl+C

Paste Ctrl+V

Paste as plain text Ctrl+Shift+V

Select all Ctrl+A

Spellcheck

Writing Direction

Inspect Ctrl+Shift+I

选择 webhook 服务

让我们检查我们的触发器是否运行，当您输入第一个 Pir 传感器范围时，这里的触发器已经运行，这意味着已经向指定的 URL 发出了 HTTPPost 请求  
返回 IFTTT 我的小程序可以发现已经在运行状态。

