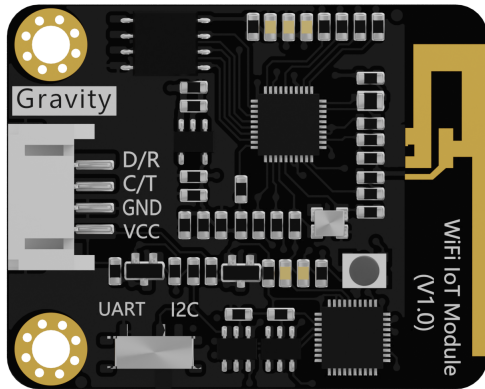


## SKU:TEL0126 (<https://www.dfrobot.com/product-2180.html>)

---



(<https://www.dfrobot.com/product-2180.html>)

## Introduction

---

This WiFi IoT module could be an excellent choice for IoT teaching and smart home projects. It supports for multiple programming platforms, such as MakeCode, Mind+ and Arduino IDE, and can also be used on various popular IoT platforms like Easy IoT, IFFTTT, ThingSpeak, ONENET, SloT, BeeBotte. Besides that, the module is designed with easy-to-use Gravity interface and employs UART and I2C communication protocols. You can use it to build IoT applications with other mainboards like micro:bit, Arduino, STM32, etc.

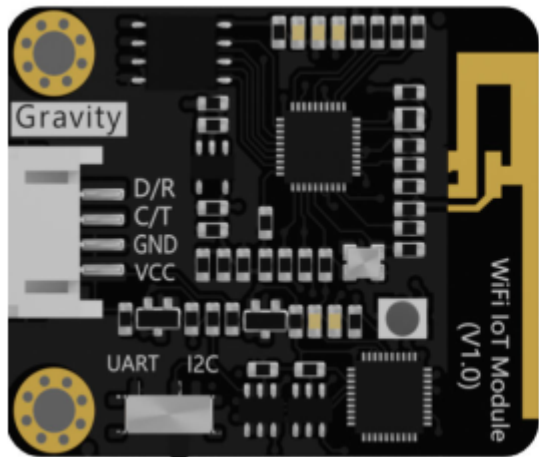
## Specification

---

- Power Supply: 3.3~5.0V
- Operating Current: <240mA
- Connector: Gravity-4P I2C/UART
- Data Transmission Rate: 9600
- Wireless Mode: IEEE802.11b/g/n
- Encryption Type: WPA WPA2/WPA2-PSK
- WiFi Frequency: 2.4GHz

- Dimension: 35mm×32mm /1.38inch×1.26inch
- Built-in Protocol: TCP/IP Protocol Stack
- Supported IoT Platform: Easy IoT, IFFTTT, ThingSpeak, ONENET, SIoT, BeeBotte
- Supported Programming Platforms: Arduino IDE, MakeCode, Mind+

## Board Overview



No.	Name	Function
1	D/R	Data Line(I2C)/Receiver(UART)
2	C/T	Clock Line(I2C)/Transmitter(UART)
3	GND	-
4	VCC	+

## Indicator Description

Indicator Color	Status
Red	Connection failed
Blue	Connecting
Green	Working Properly
Purple	MQTT Disconnected

## Programming on MakeCode


This part will be mainly introducing how to use WiFi IoT module and how to program it on MakeCode. About how to use MakeCode, click here (<https://wiki.dfrobot.com/Makecode%20Get-started%20Tutorial>).



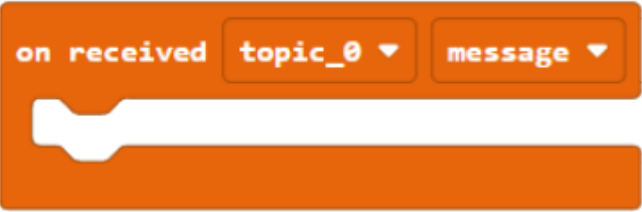

## MakeCode Platform and Program Library






- MakeCode Programming Platform: <https://makecode.microbit.org/>  
(<https://makecode.microbit.org/>)
- MakeCode UART Library Address: [https://github.com/DFRobot/pxt-DFRobot\\_WiFi\\_IoT\\_UART](https://github.com/DFRobot/pxt-DFRobot_WiFi_IoT_UART)  
([https://github.com/DFRobot/pxt-DFRobot\\_WiFi\\_IoT\\_UART](https://github.com/DFRobot/pxt-DFRobot_WiFi_IoT_UART))
- MakeCode I2C Library Address: [https://github.com/DFRobot/pxt-DFRobot\\_WiFi\\_IoT\\_I2C](https://github.com/DFRobot/pxt-DFRobot_WiFi_IoT_I2C)  
([https://github.com/DFRobot/pxt-DFRobot\\_WiFi\\_IoT\\_I2C](https://github.com/DFRobot/pxt-DFRobot_WiFi_IoT_I2C))

## Block Function Description



	<h3>WiFi Configuration-I2C</h3> <p><b>Function:</b> configure your WiFi, and only need to be set once in one program. Place it in "On Start" .</p> <p><b>Name:</b> fill in the WiFi name</p> <p><b>Password:</b> fill in WiFi password</p> <p><b>Note:</b> the block is used for I2C WiFi configuration.</p>
	<h3>WiFi Initialize Setup -UART</h3> <p><b>Function:</b> set pin and WiFi parameters. Only need to be set once in one program. Place it in "On start" .</p> <p>Pin setup: set receiving/transmitting pin</p> <p><b>Name:</b> fill in the WiFi name</p> <p><b>Password:</b> fill in the WiFi password</p> <p><b>Note:</b> the block is used for UART WiFi and Pin configuration.</p>
	<h3>MQTT Configuration</h3> <p><b>Function:</b> use this block to set MQTT if the IoT platform you used adopts MQTT protocol (like Easy IoT, SloT, etc).</p> <p><b>IOT_ID(User):</b> fill in the user ID of the IoT Platform.</p> <p><b>IOT_PWD(Password):</b> fill in the user password of the IoT platform.</p> <p><b>Topic (Default topic_0):</b> fill in the Code</p>

	<p>generated in Topic of the MQTT platform.</p> <p><b>Sever Option:</b> EasyIoT_CN, EasyIoT_EN, SIOT</p> <p><b>IP Address:</b> click the “+” to see it. EasyIoT-CN and EasyIoT-EN use the default address, no need to revise. SIOT IP address should be filled in correctly.</p>
	<p>MQTT Subscribe additional</p> <p><b>Function:</b> users can create and operate multiple Topics(up to 5) in IoT platform of MQTT protocol like Easy IoT.</p> <p>When using this block, it is necessary to create the corresponding Topic in IoT platform.</p> <p>Fill the text-box with the corresponding Code in Topic.</p> <p><b>Options:</b> topic_0, topic_1, topic_2, topic_3, topic_4</p>
	<p>Execute the program inside when received messages from MQTT. (Event Trigger)</p> <p><b>Function:</b> run the code inside when received messages from MQTT. This is an event trigger block. When the event is triggered, there will be a data of string-type named message.</p> <p><b>Option:</b> topic_0, topic_1, topic_2, topic_3, topic_4</p>
	<p>Configure IFTTT event name and password</p> <p><b>Function:</b> create event on IFTTT, then fill in the related event name and password here.</p>

	<p>Send message to IFTTT  <b>Function:</b> send string message to IFTTT, three values available.</p>
	<p>Configure ThingSpeak Password  <b>Function:</b> configure the password of the ThingSpeak platform.</p>
	<p>Send Message to ThingSpeak  <b>Function:</b> send string message to ThingSpeak, click" +" to send multiple messages.</p>
	<p>Configure BeeBotte Password  <b>Function:</b> configure the password of BeeBotte platform.</p>
	<p>Send message to BeeBotte  <b>Function:</b> send message to BeeBotte, fill in the related channel, resource, and message to be sent.</p>

## Tutorial

### Project 1-Upload Data-UART

#### 1-1. Introduction

Press down the button A on the micro:bit, the WiFi IoT module send a message "Hi, DFRobot" to Easy IoT via UART communication.

#### 1-2. Preparation

- Hardware

- micro:bit board x 1
- IO Extender for micro:bit x1
- WiFi IoT module x1

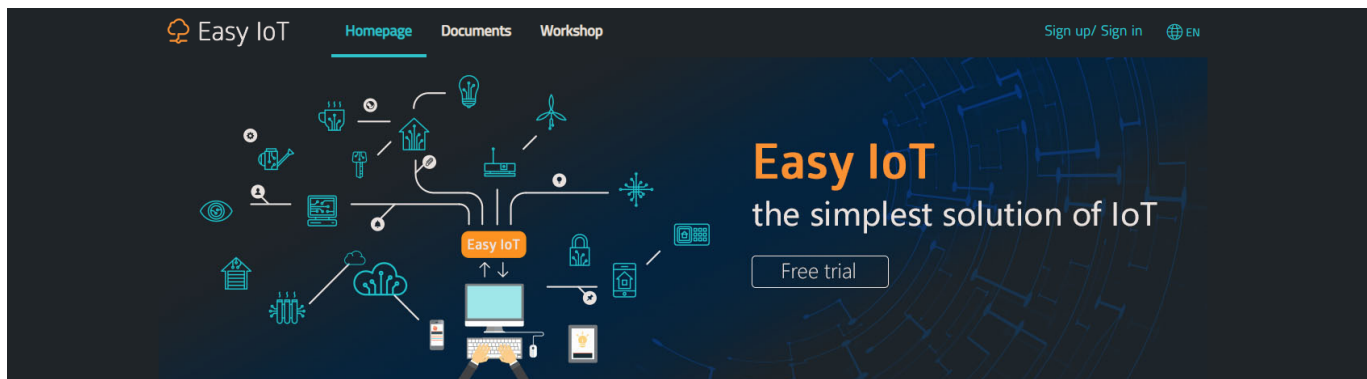
WiFi IoT Module Kit  
- Wires

- UART Library

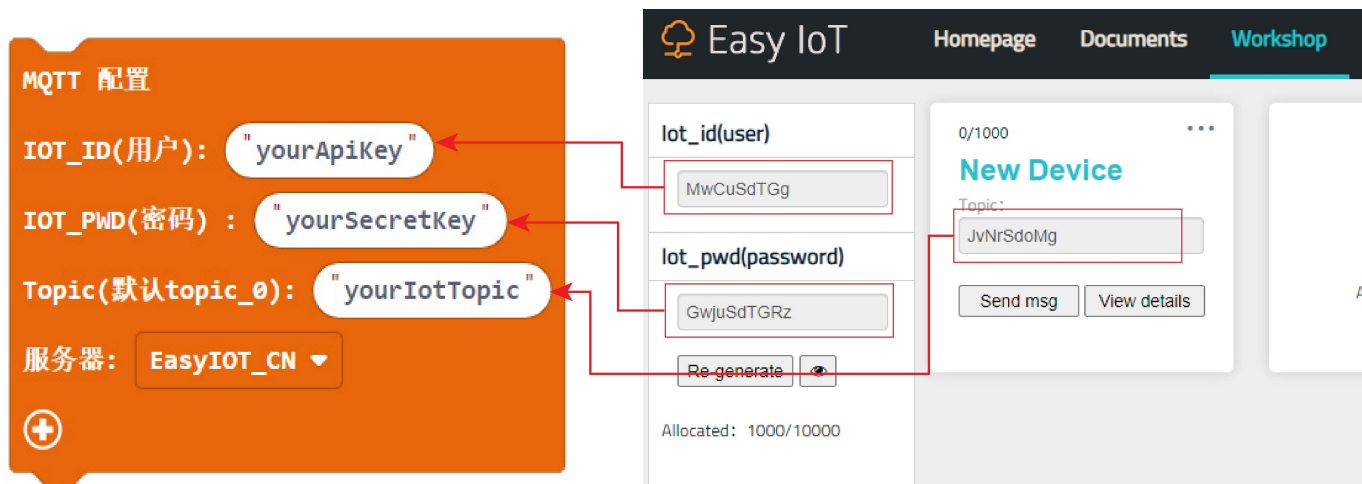
MakeCode UART Library: [https://github.com/DFRobot/pxt-DFRobot\\_WiFi\\_IoT\\_UART](https://github.com/DFRobot/pxt-DFRobot_WiFi_IoT_UART)  
([https://github.com/DFRobot/pxt-DFRobot\\_WiFi\\_IoT\\_UART](https://github.com/DFRobot/pxt-DFRobot_WiFi_IoT_UART))

- Configure IoT Platform

1. Open the Easy IoT website: <http://iot.dfrobot.com> (<http://iot.dfrobot.com>), register an account if you don't have one, and then sign in.



2. Click the icon to check the generated ID and Password, and fill them in the corresponding MQTT configuration blocks. Select server "Easy IoT\_EN".



3. Find your local WiFi and Password, fill them in the WiFi configuration block.



```

Wi-Fi configure

Pin set:

receiving data (green wire): P1 ▾

sending data (blue wire): P2 ▾

Wi-Fi:

name: "yourSSID"

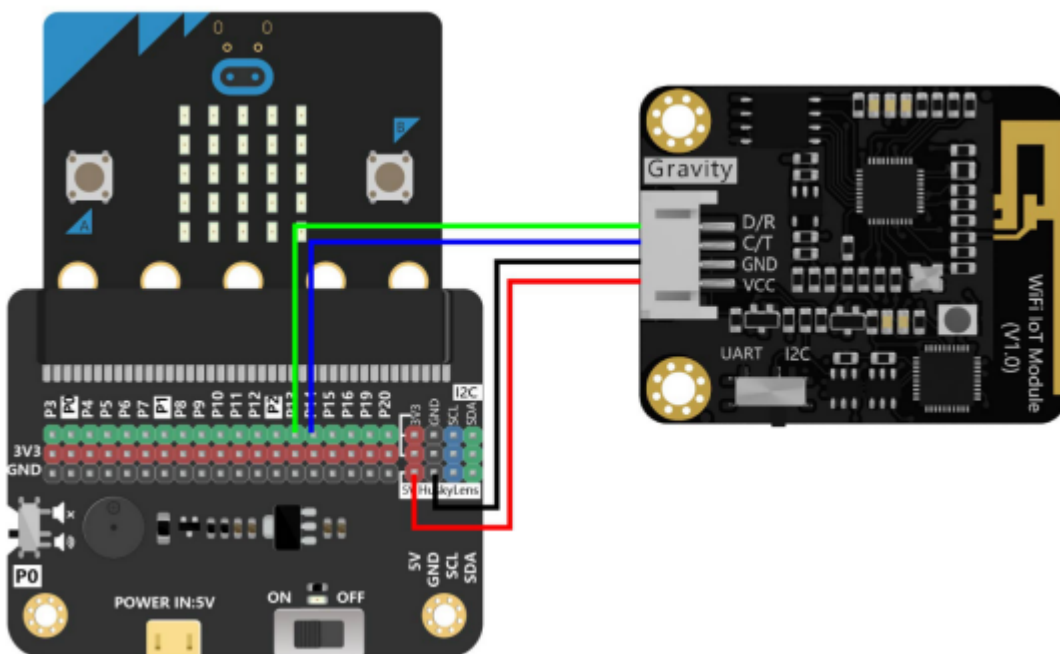
password: "yourPASSWORD"

start connection

```

- Hardware Connection

Connect the pin D/R, C/T, GND and VCC of the WiFi IoT module to P13, 14, GND and VCC of the IO expansion board respectively. You can also define them by programming. Connect them as follows.



### 1-3. Program Link

[https://makecode.microbit.org/\\_gTjH91cbTTtt](https://makecode.microbit.org/_gTjH91cbTTtt) ([https://makecode.microbit.org/\\_gTjH91cbTTtt](https://makecode.microbit.org/_gTjH91cbTTtt))

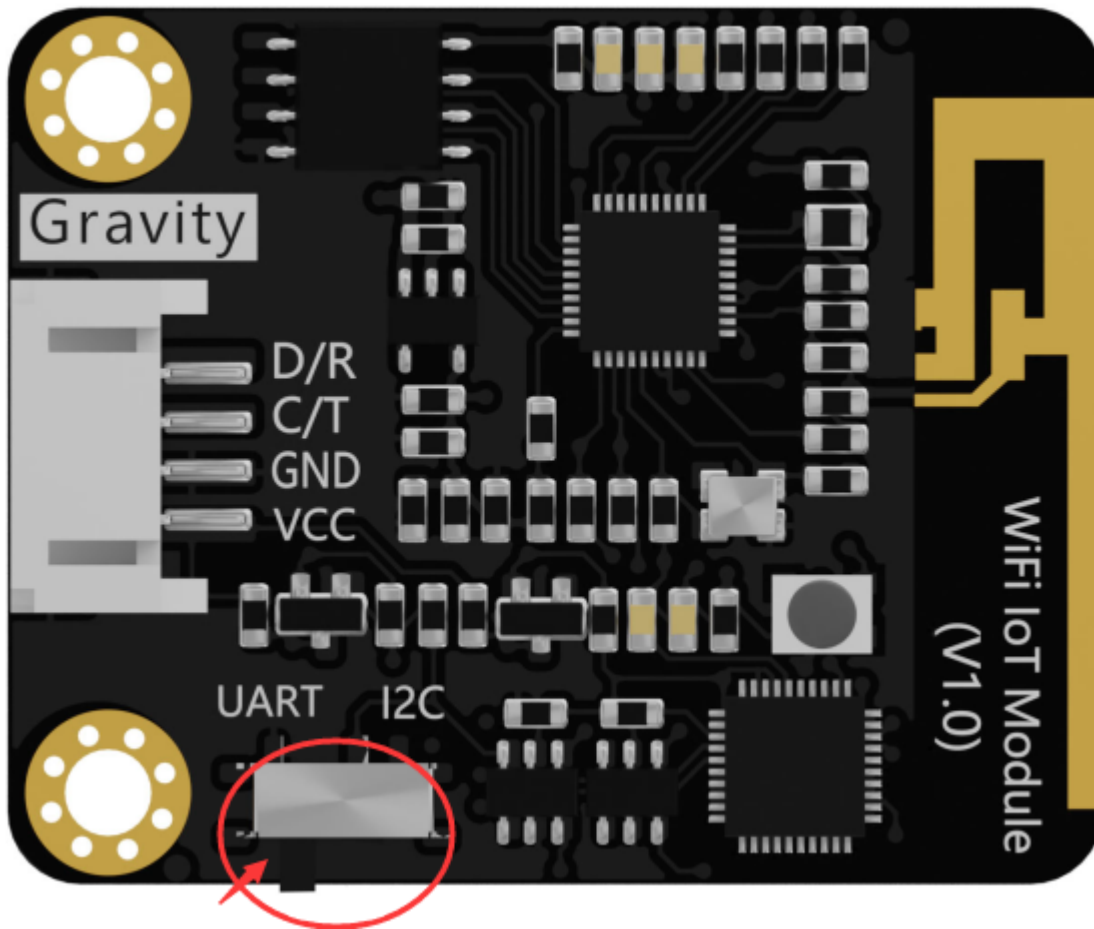
## 1-4. Program Blocks

The image shows two code blocks from a Microbit programming environment. The first block is a blue 'on start' block containing an orange 'Wi-Fi configure' block and another orange 'MQTT configure' block. The 'Wi-Fi configure' block has fields for 'Pin set:', 'receiving data (green wire):' (set to P13), 'sending data (blue wire):' (set to P14), 'Wi-Fi: name:' (set to 'hitest'), and 'password:' (set to '12345678'). The 'MQTT configure' block has fields for 'IoT\_ID(user):' (set to 'MwCuSdTGg'), 'IoT\_PWD(password):' (set to 'GwjuSdTGRz'), 'Topic(default topic\_0):' (set to 'JvNrSdoMg'), and 'server:' (set to 'EasyIoT\_EN'). The second block is a purple 'on button A pressed' block containing an orange 'send message' block with the message 'Hi DFRobot' and the topic 'topic\_0'.

Note: please fill in the correct Easy IoT ID and password.


## 1-5. Effect Display

Since we use UART communication mode here, the control switch on the WiFi IOT module needs to be turned to the UART side, otherwise the initialization of the module will fail and the data transmission will not be possible.



After the program is burned successfully, the WiFi IOT module indicator will show the connection process: red-blue-green. Press the button A on micro:bit, the WiFi IOT sends a "Hi DFRobot" message to the Easy IOT platform. Click "view details" in the position as shown in the figure below to enter the data page to see the received data.

The screenshot shows the Easy IoT web interface. At the top, there is a navigation bar with 'Homepage', 'Documents', and 'Workshop' tabs. The 'Workshop' tab is active. Below the navigation bar, there are two main panels. The left panel contains two sections: 'lot\_id(user)' with a text input field containing 'MwCuSdTGg', and 'lot\_pwd(password)' with a text input field containing 'GwjuSdTGRz', a 'Re-generate' button, and an eye icon. Below these is the text 'Allocated: 1000/10000'. The right panel shows a 'New Device' card with a '0/1000' counter and a three-dot menu. The card title is 'New Device' in blue. Below the title is a 'Topic:' label and a text input field containing 'JvNrSdoMg'. At the bottom of the card are two buttons: 'Send msg' and 'View details'. The 'View details' button is highlighted with a red box and a red arrow pointing to it.

 **Query result**

Time	Message	Operate
2020/11/16 11:54:52	Hi DFRobot	
2020/11/16 11:54:52	Hi DFRobot	
2020/11/16 11:54:52	Hi DFRobot	
2020/11/16 11:54:51	Hi DFRobot	

Note: power the IO shield via USB when running the program.

## Project 2-Upload Data-I2C

### 2-1. Introduction

Press down the button A on the micro:bit, the WiFi IoT module will send a message "Hi, DFRobot" to Easy IoT via I2C communication.

## 2-2. Preparation

- Hardware

- micro:bit board x 1
- IO Extender for micro:bit x1
- WiFi IoT module x1
- Wires

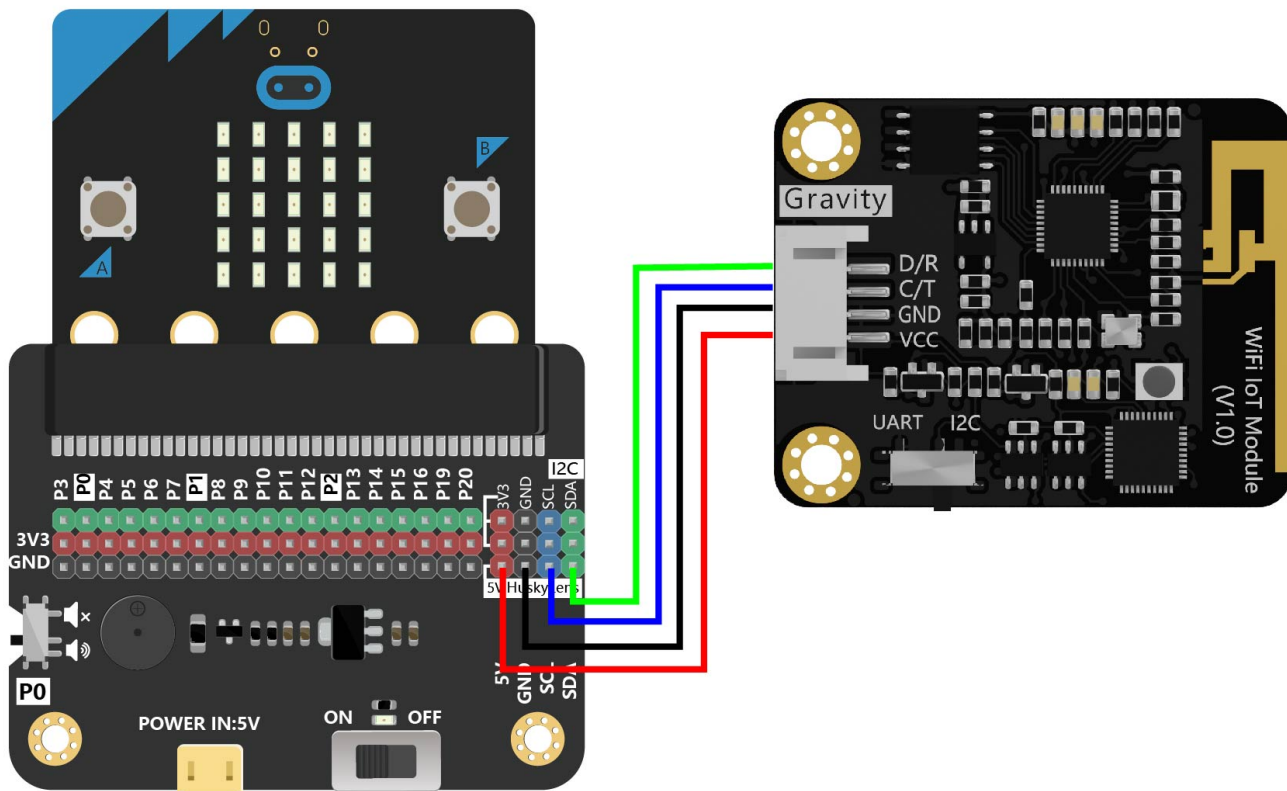
- I2C Library

[https://github.com/DFRobot/pxt-DFRobot\\_WiFi\\_IoT\\_I2C](https://github.com/DFRobot/pxt-DFRobot_WiFi_IoT_I2C) ([https://github.com/DFRobot/pxt-DFRobot\\_WiFi\\_IoT\\_I2C](https://github.com/DFRobot/pxt-DFRobot_WiFi_IoT_I2C))

- Configure IoT Platform

The platform configuration of project 1 can be directly used here.

- Hardware Connection



## 2-3. Program Link

[https://makecode.microbit.org/\\_Fxy6MLgmJ2M](https://makecode.microbit.org/_Fxy6MLgmJ2M)  
 ([https://makecode.microbit.org/\\_Fxy6MLgmJ2M](https://makecode.microbit.org/_Fxy6MLgmJ2M))

## 2-4. Program Blocks



## 2-5. Effect Display

Since we use I2C communication mode here, the control switch on the WiFi IOT module needs to be turned to the I2C side, otherwise the initialization of the module will fail and the data transmission will not be possible.



Press down the button A on the micro:bit, the WiFi IoT module send a message "Hi, DFRobot" to Easy IoT platform.



From the two projects above, it can be seen that when we use UART or I2C, the final function is basically the same. However, I2C communication will not occupy the serial port, and more convenient to use. So we will use I2C in the later projects.

# Project 3-Temperature and Brightness Remote Monitor(Based on Easy IoT)

## 3-1. Introduction

Use the micro:bit onboard temperature and light sensors to monitor environment. The WiFi IoT Module, micro:bit and Easy IoT server will be used for data exchange to realize remote monitoring of temperature and brightness based on IoT.

## 3-2. Preparation


- **Hardware**

```
- micro:bit board x 1
- IO Extender for micro:bit x1
- WiFi IoT module x1
- Wires
```

- I2C Library

MakeCode I2C library: [https://github.com/DFRobot/pxt-DFRobot\\_WiFi\\_IoT\\_I2C](https://github.com/DFRobot/pxt-DFRobot_WiFi_IoT_I2C)  
([https://github.com/DFRobot/pxt-DFRobot\\_WiFi\\_IoT\\_I2C](https://github.com/DFRobot/pxt-DFRobot_WiFi_IoT_I2C))


- Hardware Connection

 Connection 3

## 3-3. Program Link

[https://makecode.microbit.org/\\_Pow1b0MKRVsm](https://makecode.microbit.org/_Pow1b0MKRVsm)  
([https://makecode.microbit.org/\\_Pow1b0MKRVsm](https://makecode.microbit.org/_Pow1b0MKRVsm))

## 3-4. Program Blocks

 Project 3 blocks

## 3-5. Effect Display

Send data from IoT platform to micro:bit via WiFi module. When micro:bit received data "L", the micro:bit board displays the current brightness and upload it to the IoT platform; When the received data is "T", display the current temperature and upload it to the IoT platform. Refresh the page to view the uploaded messages.



## Project 4-Sending Email (Based on IFTTT)

## 4-1. Introduction

Connect to IFTTT platform with the WiFi IoT module. When the button A on the micro:bit is pressed, a message recording the current temperature and light will be sent to IFTTT. In this way, we can send the current temperature and light value to the people we care about in real-time by email.

## 4-2. Preparation

- **Hardware**

```
- micro:bit board x 1  
- IO Extender for micro:bit x1  
- WiFi IoT module x1  
- Wires
```

- I2C Library

MakeCode I2C Library: [https://github.com/DFRobot/pxt-DFRobot\\_WiFi\\_IoT\\_I2C](https://github.com/DFRobot/pxt-DFRobot_WiFi_IoT_I2C)  
([https://github.com/DFRobot/pxt-DFRobot\\_WiFi\\_IoT\\_I2C](https://github.com/DFRobot/pxt-DFRobot_WiFi_IoT_I2C))

- Configure IoT platform

1. Enter IFTTT website (<https://ifttt.com/>), register an account if you don't have one. Then Sign in.



2. The following interface will appear when you signed in.



3. Click "Create" to enter the interface below.



4. Click "if this" and input "webhooks" in the search bar.



5. The following interface when entering for the first time, click "Receive a web request".



6. Fill in the Event Name, then you have created this.(The event name can be defined by yourself).



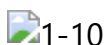
7. The webpage will return back automatically. Click "that" and select "Email".



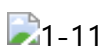
8. Click "Connect", fill in your email address, and click "send PIN" to send a PIN code to your email box.



9. Check your email to find the PIN code, and fill it in the webpage, then click "Connect".



10. Click "Send me an Email" to set the email.

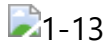




11. You can write the content to be sent to your email box in the interface below, here we select the default one. Click "Create action".



12. Enter the interface as follows.



13. Check Password: click your avatar, click "My services".



14. Click webhooks to enter the event created.



15. Enter webhooks, click "Documentation" then the password can be seen.



- Hard Connection



4-3. Program Link

[https://makecode.microbit.org/\\_6r9WkE7Lrfi9](https://makecode.microbit.org/_6r9WkE7Lrfi9) ([https://makecode.microbit.org/\\_6r9WkE7Lrfi9](https://makecode.microbit.org/_6r9WkE7Lrfi9))

4-4. Program Blocks



4-5. Effect Display

Press the button A on the micro:bit, the current temperature and light values will be detected and uploaded to IFTTT, and then forward to the mailbox you set.



## Project 5-Sending Twitter (Based on IFTTT)

5-1. Introduction

This project demonstrates how to send a text message to twitter by pressing the button on

micro:bit. You can extend this to other application scenarios according to this project, for instance, sending current water quality, air quality and other data to twitter.

## 5-2. Preparation

- **Hardware**

```
- micro:bit board x 1
- IO Extender for micro:bit x1
- WiFi IoT module x1
- Wires
```

- I2C Library

MakeCode I2C Library Address: [https://github.com/DFRobot/pxt-DFRobot\\_WiFi\\_IoT\\_I2C](https://github.com/DFRobot/pxt-DFRobot_WiFi_IoT_I2C)  
([https://github.com/DFRobot/pxt-DFRobot\\_WiFi\\_IoT\\_I2C](https://github.com/DFRobot/pxt-DFRobot_WiFi_IoT_I2C))

- Configure IoT platform

In the last project, we have logged in the account on the computer. Next, we use the IFTTT app on the iPad to create events.

1. Search IFTTT in your APP store and download it. Log in your IFTTT account.



2. Click "Create", then the following interface will appear, click "Start".



3. Click "This" to configure it.



4. Search "Webhooks" in the search bar and click the result.



5. Click "Receive a web request" to enter "create your own" setup, input your Event name, and click "Create trigger".



6. It will return to the following page after clicking "Create trigger". Click "That" to configure it.



7. Click that to enter the search page, search for twitter in the input box and click to enter the twitter settings. After entering, there are four options: post a tweet, post a tweet with image, update profile picture, and update bio. Here, we select post a tweet.



8. Click "Add" in the interface below, and then fill in your twitter account and password in the pop-up box to add twitter account.



9. Enter the interface below after the last step. Click "Create action", and then click "Finish".



- Hardware Connection



5-3. Program Link

[https://makecode.microbit.org/\\_8bpXa41DfTU7](https://makecode.microbit.org/_8bpXa41DfTU7)  
([https://makecode.microbit.org/\\_8bpXa41DfTU7](https://makecode.microbit.org/_8bpXa41DfTU7))

5-4. Program Block



5-5. Effect Display

Press the button A on the micro:bit, send a twitter "Hi DFRobot 2020".



## Project 6 Ambient Noise Analysis (based ThingSpeak)

6-1. Introduction

This project reads the data changes of the sound sensor, uploads them to thingspeak, and generates graphs for analyzing noise situation in the current environment, and then we can find

the period with the most serious noise in a day.

## 6-2. Preparation

- **Hardware**

```
- micro:bit board x 1
- IO Extender for micro:bit x1
- WiFi IoT module x1
- Analog Sound sensor
- Wires
```

- I2C Library

MakeCode I2C Library: [https://github.com/DFRobot/pxt-DFRobot\\_WiFi\\_IoT\\_I2C](https://github.com/DFRobot/pxt-DFRobot_WiFi_IoT_I2C)

([https://github.com/DFRobot/pxt-DFRobot\\_WiFi\\_IoT\\_I2C](https://github.com/DFRobot/pxt-DFRobot_WiFi_IoT_I2C))

- How to use ThingSpeak

1. Open ThingSpeak website, <https://thingspeak.com/> (<https://thingspeak.com/>), click the right-upper corner to enter the register and login interface.



2. When using for first time, register an account in the following interface. Fill in the correct email address and verify the mail box.



3. The following interface will appear after finishing the registration.



4. The above interface can also be found by clicking "Channels->My Channel". Now click new channel.



5. Enter the interface below then.



6. Here, we need to fill in the name, channel description, and tick the number of fields. In the

makecode library, the fields are the same as the fields in the channel. In the routine, only the noise analog value is transferred to the channel. So only one is ticked here, and the field name can be customized. We modify the field name here as noise. Save the channel now. Here is the channel I created.



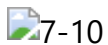
7. After saving, the channel is created. The created channel is shown in the figure below, the parameters uploaded can be seen here.



8. If more data needs to be uploaded, you need to click channel settings, click to enter the figure below, tick the "box" behind the field, and save it. You can also delete channel and clear channel data here.



9. Channel added successfully.



10. Click API Keys then we can see the related password of the channel. The password in the red box should be filled in the corresponding block in programming.



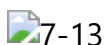
- Hardware Connection



6-3. Program Link

[https://makecode.microbit.org/\\_MA3LC6cYwRoH](https://makecode.microbit.org/_MA3LC6cYwRoH)  
([https://makecode.microbit.org/\\_MA3LC6cYwRoH](https://makecode.microbit.org/_MA3LC6cYwRoH))

6-4. Program Blocks



6-5. Effect Display

Check the uploaded data in private view of thingSpeak platform, the sound value read by analog sound sensor will be uploaded to IoT platform in real time, and displayed in the form of line graph.



## Project 7-Alcohol concentration detector(Based on BeeBotte)

### 7-1. Introduction

Let's imagine, in the alcohol breath testing, if there is an alcohol concentration detector in a car that can upload the detected data to beebotte platform in real time, then the police only need to use the data on the platform to roughly judge whether there are people drinking in the current vehicle, and then check the vehicles with alcohol concentration over 800 (assuming that the alcohol concentration value exceeds 800 belongs to drunken driving). This can not only save time, but also greatly reduce the workload of traffic police.

### 7-2. Preparation

- **Hardware**

```
- micro:bit board x 1
- IO Extender for micro:bit x1
- WiFi IoT module x1
- Analog Alcohol sensor
- Wires
```

- I2C Library

MakeCode I2C Library: [https://github.com/DFRobot/pxt-DFRobot\\_WiFi\\_IoT\\_I2C](https://github.com/DFRobot/pxt-DFRobot_WiFi_IoT_I2C)  
([https://github.com/DFRobot/pxt-DFRobot\\_WiFi\\_IoT\\_I2C](https://github.com/DFRobot/pxt-DFRobot_WiFi_IoT_I2C))

- How to use BeeBotte

1. Open BeeBotte website: <https://beebotte.com/login> (<https://beebotte.com/login>). Register an account and log in.

8-1

2. Enter the following interface when logged in. Click "Create New" to create a new channel.

8-2

3. Write in the corresponding Channel name and resource according to the tip.

8-3

4. Click "Create channel" to build up your own BeeBotte Channel.

8-4

5. Check the channel token in the created channel of BeeBotte.



- Hardware Conneciton



7-3. Program Link

[https://makecode.microbit.org/\\_8PbTJAFPL2CH](https://makecode.microbit.org/_8PbTJAFPL2CH) ([https://makecode.microbit.org/\\_8PbTJAFPL2CH](https://makecode.microbit.org/_8PbTJAFPL2CH))

7-4. Program Blocks



7-5. Effect Display

Check the alcohol concentration on the corredspoding channel in BeeBotte.



Turn the received data into line chart, table, etc



## Programming on Arduino IDE

---

In this part, we will program the WiFi IoT module on Arduino IDE.

- **Software**
  - Arduino IDE (<https://www.arduino.cc/en/Main/Software>)
  - Download and install the **WiFi IoT Module Library** ([https://github.com/DFRobot/DFRobot\\_WiFi\\_IoT\\_Module](https://github.com/DFRobot/DFRobot_WiFi_IoT_Module)) (About how to install the library? (<https://www.arduino.cc/en/Guide/Libraries#.UxU8mdzF9H0>))

## Project 1- Ambient Light Analysis -UART (Based on ThingSpeak)

Upload the detected light value to ThingSpeak via the UART communication of WiFi IoT module, and generate curve graph, then find the period with the most and least daylight in a day.

1-2. Preparation

- **Hardware**

```
- micro:bit board x 1
- IO Extender for micro:bit x1
- WiFi IoT module x1
- Analog Ambient light sensor
- Wires
```

- How to use ThingSpeak

Since we have registered the ThingSpeak account, here we only need to create a channel to receive the ambient light. The detailed way to do that, refer to project 7 above.

- Hardware Connection

Connect the pin D/R, C/T, GND, VCC of the WiFi IoT module to the Pin D10, D11, GND and VCC of the Arduino IDE. The analog ambient light sensor goes to A0. You can also define the related pins in programming.

 Connection 1

1-3. Sample Code



```

#include "DFRobot_WiFi_IoT_Module.h"
#include <SoftwareSerial.h>

//UART mode
//Use software serialport RX: 10, TX: 11
SoftwareSerial mySerial(10, 11);
DFRobot_WiFi_IoT_Module_UART IoT (&mySerial);

//Configure WiFi
const char *WIFI_SSID = "hitest"; //WiFi Name
const char *WIFI_PASSWORD = "12345678"; //WiFi Password

//Configure Thingspeak
const char *ThingSpeak_ADDRESS = "api.thingspeak.com"; //ThingSpeak address, not
const char *THINGSPEAK_KEY = "4I1G5SXC5PGU3HA4"; //Fill in the created event

//Send Message to ThingSpeak
const char *THINGSPEAK_VALUE_1 = "Value1";
const char *THINGSPEAK_VALUE_2 = "Value2";
const char *THINGSPEAK_VALUE_3 = "Value3";

/*****
Function: getAmbient
Description: Get light sensor value
Params:
Return: current light value
*****/
int getAmbient()
{
  int val;
  val = analogRead(0);
  return val;
}

void setup() {
  //Use software serialport 9600 as communication port
  mySerial.begin(9600);

  //Use serialport 115200 as print port
  Serial.begin(115200);

  //Init communication port
  while(IoT.begin() != 0){
    Serial.println("init ERROR!!!!");
    delay(100);
  }
}

```

```
}
Serial.println("init Success");

//Connect to WiFi
while(IoT.connectWifi(WIFI_SSID, WIFI_PASSWORD) != 0){

    Serial.print(".");
    delay(100);
}
Serial.println("Wifi Connect Success");

while(IoT.HTTPBegin(ThingSpeak_ADDRESS) != 0){
    Serial.println("Parameter is empty!");
    delay(100);
}
Serial.println("HTTP Configuration Success");

while(IoT.thingSpeakBegin(THINGSPEAK_KEY) != 0){
    Serial.println("Parameter is empty!");
    delay(100);
}
Serial.println("ThingSpeak Configuration Success");
}

void loop() {
    String data =(String)getAmbient();
    //Upload current ambient light data to ThingSpeak every 1s
    IoT.thingSpeakSendMessage(data.c_str());
    delay(1000);
}
```

1-4. Upload the data detected by the analog ambient light sensor to ThingSpeak, and display them in an line chart.



## Project 2-Ambient Noise Analysis (based ThingSpeak)

### 2-1. Introduction

Read the data changes of the sound sensor, upload them to ThingSpeak, and generates graphs for analyzing noise situation in the current environment, and then we can find the period with the most serious noise in a day.

### 2-2. Preparation

- **Hardware**

- Arduino UNO board x 1
- IO Expansion board x1
- WiFi IoT module x1
- Analog Sound sensor
- Wires

- Hardware Connection

 Connecion 2

2-3. Sample Code

```

#include "DFRobot_WiFi_IoT_Module.h"
#include <SoftwareSerial.h>

//I2C Mode
DFRobot_WiFi_IoT_Module_I2C IoT;

//configure WiFi account and password
const char *WIFI_SSID          = "hitest";
const char *WIFI_PASSWORD      = "12345678";

//Configure ThingSpeak
const char *ThingSpeak_ADDRESS = "api.thingspeak.com"; //ThingSpeak address, do
const char *THINGSPEAK_KEY     = "RXARKPLSDMR3G8MA"; //Fill in the event password

//Send message to ThingSpeak
const char *THINGSPEAK_VALUE_1 = "Value1";
const char *THINGSPEAK_VALUE_2 = "Value2";
const char *THINGSPEAK_VALUE_3 = "Value3";

/*****
Function:      getNoise
Description:   Get the value of sound sensor
Params:
Return:       Current sound value
*****/
int getNoise()
{
    int val;
    val = analogRead(0);
    return val;
}

void setup() {
    //use serialport 115200 as print port
    Serial.begin(115200);

    //init communication port
    while(IoT.begin() != 0){
        Serial.println("init ERROR!!!!");
        delay(100);
    }
    Serial.println("init Success");

    //Connect to WiFi
    while(IoT.connectWifi(WIFI_SSID, WIFI_PASSWORD) != 0){

```

```
    Serial.print(".");
    delay(100);
}
Serial.println("Wifi Connect Success");

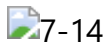
while(IoT.HTTPBegin(ThingSpeak_ADDRESS) != 0){
    Serial.println("Parameter is empty!");
    delay(100);
}
Serial.println("HTTP Configuration Success");

while(IoT.thingSpeakBegin(THINGSPEAK_KEY) != 0){
    Serial.print("Parameter is empty!");
    delay(100);
}
Serial.println("ThingSpeak Configuration Success");
}

void loop() {
    String data =(String)getNoise();
    //Upload current sound value to ThingSpeak every 1s
    IoT.thingSpeakSendMessage(data.c_str());
    delay(1000);
}
```

## 2-4. Effect Display

Check the uploaded data in private view of thingSpeak platform, the sound value read by analog sound sensor will be uploaded to IoT platform in real time, and displayed in the form of line graph.



## Project 3-Temperature Sending (Based on IFTTT)


### 3-1. Introduction

### 3-2. Preparation

- **Hardware**

- Arduino UNO board x 1
- IO Expansion board x1
- WiFi IoT module x1
- LM35 Temperature Sensor x1
- Digital Push Button
- Wires

- Hardware Connection

 Connection 3

3-3. Sample Code

```

#include "DFRobot_WiFi_IoT_Module.h"
#include <SoftwareSerial.h>

int button = 7;

//UART mode
//Use software serialport: D/R: 10, C/T: 11
SoftwareSerial mySerial(10, 11);
DFRobot_WiFi_IoT_Module_UART IoT(&mySerial);

//Configure WiFi name and password
const char *WIFI_SSID          = "hitest";
const char *WIFI_PASSWORD      = "12345678";
//Configure IFTTT
const char *IFTTT_ADDRESS      = "maker.ifttt.com";
const char *IFTTT_EVENT        = "Temperature";
const char *IFTTT_KEY          = "cdvP-sL1L9m3e1d_T0xR3r";
//IFTTT Send message
const char *IFTTT_VALUE_1      = "Value1";
const char *IFTTT_VALUE_2      = "Value2";
const char *IFTTT_VALUE_3      = "Value3";

/*****
Function      : getTemp
Description   : Get temperature detected by LM35
Params       :
Return       : Converted temperature value
*****/
float getTemp()
{
    uint16_t val;
    float dat;
    val = analogRead(A0);
    dat = (float)val * (5/10.24);
    return dat;
}

void setup() {
    //Use software serialport 9600 as communication port
    mySerial.begin(9600);
    //Use serialport 115200 as print port
    Serial.begin(115200);
    //Init communication port
    while(IoT.begin() != 0){

```

```
Serial.println("init ERROR!!!!");
delay(100);
}
Serial.println("init Success");
//Connect to WiFi


while(IoT.connectWifi(WIFI_SSID, WIFI_PASSWORD) != 0){
  Serial.print(".");
  delay(100);
}
Serial.println("Wifi Connect Success");
//Init HTTP
while(IoT.HTTPBegin(IFTTT_ADDRESS) != 0){
  Serial.println("Parameter is empty!");
  delay(100);
}
Serial.println("HTTP Configuration Succeeded");
//Init IFTTT
while(IoT.IFTTTBegin(IFTTT_ENVENT, IFTTT_KEY) != 0){
  Serial.println("Parameter is empty!");
  delay(100);
}
Serial.println("IFTTT Configuration Succeeded");

pinMode(button,INPUT);
}

void loop() {
  String data =(String)getTemp();
  //Press down button to send an Email recording the current temperature
  if(digitalRead(7) == 1)
  {
    IoT.IFTTTSendMessage("reminder","The current temperature is",data.c_str());
    Serial.println("send message success!");
  }
}
```

### 3-4. Effect Display

Press down the button, IFTTT sends an Email recording the current temperature to the corresponding mail box.

 Project 3 effect

## Project 4-Smart Watering System (Based on Easy IoT)

### 4-1. Introduction

It is said that 80% of the plant growth problems are caused by untimely watering or excessive



watering. We assume that the best soil moisture suitable for succulents growth is 100-300. When the soil moisture is less than or equal to 100, the plant will say, "Hi, I am thirsty, please give me water"; when the soil moisture is  $\geq 300$ , the plant will say: "Hi, I am full, and don't

need water now", so we can water the plants according to the needs of plants. If you can control watering remotely, you can even don't have to worry about the plants at home when going out for work for a long time.

## 4-2. Preparation

- **Hardware**

- Arduino UNO board x 1
- IO Expansion board x1
- WiFi IoT module x1
- Soil Moisture Sensor x1
- Digital Relay Module x1
- Water Pump x1
- Wires

- **Hardware Connection**

 Connection 4

## 4-3. Sample Code

```

#include "DFRobot_WiFi_IoT_Module.h"
#include <SoftwareSerial.h>
int Relay = 3;
//I2C mode
DFRobot_WiFi_IoT_Module_I2C IoT;

//I2C mode
DFRobot_WiFi_IoT_Module_I2C IoT;

//You can change them based on your needs
const char *WIFI_SSID          = "hitest";           //WiFi account
const char *WIFI_PASSWORD      = "12345678";        //WiFi password

//Easy IOT English configuration
//const char *EASY_IOT_SERVER   = "iot.dfrobot.com";

//EASY IoT Chinese configuration
const char *EASY_IOT_SERVER    = "iot.dfrobot.com.cn"; //IoT host address, no need
const char *EASY_IOT_PORT      = "1883";             //IoT connect port, no need
const char *EASY_IOT_ID        = "smofVJDMR";        //Iot_id, the account received
const char *EASY_IOT_PWD       = "ymoBVJvMgz";       //Iot_pwd, the password received
const char *SUBSCRIBE_TOPIC    = "9qnYVJDMR";        //Subscribe a device, device
const char *PUBLISH_TOPIC      = "9qnYVJDMR";        //Send data to the device, device
const char *EASY_IOT_SEND_MESSAGE = "Send_Message";

/*****
Function      : callback
Description   : print the subscribed IoT device number and message
Params       : topic: subscribed device number; message: subscribed device number message
Return       :
*****/
String data = "0";
void callback(const char*topic,const char*message)
{
  Serial.println("Receive [ " + (String)topic + " ]," + "Message : " + (String)message);
  data = (String)message;
}

/*****
Function      : getSoilHumidity
Description   : Get soil humidity value
Params       :
Return       : Current soil humidity
*****/
float getSoilHumidity()

```

```
{
  uint16_t val;
  val = analogRead(A0);
  return val;
}

void setup(void){
  Serial.begin(115200);
  //Init communication port
  while(IoT.begin() != 0){
    Serial.println("init ERROR!!!!");
    delay(100);
  }
  Serial.println("init Success");
  //Connect to WiFi
  while(IoT.connectWifi(WIFI_SSID, WIFI_PASSWORD) != 0){
    Serial.print(".");
    delay(100);
  }
  Serial.println("Wifi Connect Success");
  //Init MQTT and connect to IoT platform
  while(IoT.MQTTBegin(EASY_IOT_SERVER, EASY_IOT_PORT, EASY_IOT_ID, EASY_IOT_PWD) != 0){
    Serial.print(".");
    delay(100);
  }
  Serial.println("MQTT Connect Success");
  //call callback function
  IoT.setCallBack(callback);
  //subscribe device SUBSCRIBE_TOPIC
  while(IoT.subscribe(SUBSCRIBE_TOPIC) != 0){
    Serial.print(".");
    delay(100);
  }
  Serial.println("Subscribe Topic Success");

  pinMode(13, OUTPUT);      //Set Pin13 as output
  digitalWrite(13, HIGH);  //Set Pin13 High
  pinMode(Relay, OUTPUT);  //Set Pin3 as output
}

static long timestamp = millis();
void loop(void){

  IoT.loop();
  //Received command "1" from Easy IoT, set relay to high
  if(strcmp(data.c_str(),"1") == 0){
    digitalWrite(Relay, HIGH);
    data = "0";
  }

  IoT.loop();
}
```

```
//Received command "2" from Easy IoT, set relay to low
if(strcmp(data.c_str(),"2") == 0){
    digitalWrite(Relay, LOW);
    data = "0";
}

if(millis()-timestamp>5000){
    timestamp = millis();
    int Hum = getSoilHumidity();
    //When current soil humidity Hum<=100, send the current humidity to Easy IoT
    if(Hum <= 100){
        IoT.publish(PUBLISH_TOPIC, (String)Hum);
    }
    //When current soil humidity Hum>=300, send the current humidity to Easy IoT
    if(Hum >= 300){
        IoT.publish(PUBLISH_TOPIC, (String)Hum);
    }
}
}
```

#### 4-4. Effect Display

Upload the detected soil moisture. The received message from Easy IoT is shown below:



Send command "1" from Easy IoT to water the plant(control the water pump by the relay module); command "2" for stopping watering.

## Project 5-People Counting (Based on BeeBotte)

### 5-1. Introduction

Install a ultrasonic sensor at the entrance of a mall to count the number of people passing by, and send the statistics of people flow in a certian period to BeeBotte.

### 5-2. Preparation

- **Hardware**

```
- Arduino UNO board x 1
- IO Expansion board x1
- WiFi IoT module x1
- HC-SR04 Ultrasonic Sensor x1
- Wires
```

- **Hardware Connection**

 Connection 5

## 5-3. Sample Code

```

#include "DFRobot_WiFi_IoT_Module.h"
#include <SoftwareSerial.h>

//I2C mode
DFRobot_WiFi_IoT_Module_I2C IOT;

#define time(x) (x*1000)

//configure WiFi
const char *WIFI_SSID           = "hitest";
const char *WIFI_PASSWORD       = "12345678";
//configure Beebotte
const char *BEEBOTTE_ADDRESS    = "api.beebotte.com";
const char *BEEBOTTE_TOKEN      = "token_PjXx9xPsygyz5GMU";
const char *BEEBOTTE_CHANNEL    = "Data";
const char *BEEBOTTE_RESOURCE   = "Data";

const char *BEEBOTTE_SEND_MESSAGE = "Send_Message";

int TrgPin = A0;
int EcoPin = A1;
int dist;

static long timestamp = millis();
int count;

void setup(void){
  //Use 115200 as print port
  Serial.begin(115200);
  //Init port
  while(IOT.begin() != 0){
    Serial.println("init ERROR!!!!");
    delay(100);
  }
  Serial.println("init Success");
  //Connect to WiFi
  while(IOT.connectWifi(WIFI_SSID, WIFI_PASSWORD) != 0){
    Serial.print(".");
    delay(100);
  }
  Serial.println("Wifi Connect Success");
  //Init HTTP
  while(IOT.HTTPBegin(BEEBOTTE_ADDRESS) != 0){
    Serial.println("Parameter is empty!");
    delay(100);
  }
}

```

```
}
Serial.println("HTTP Configuration Succeeded");
//Init beebotte
while(IOT.beebotteBegin(BEEBOTTE_TOKEN) != 0){
    Serial.print("Parameter is empty!");

    delay(100);
}
Serial.println("Beebotte Configuration Succeeded");

//Set TrgPin to output status
pinMode(TrgPin, OUTPUT);
//Set EcoPin to input status
pinMode(EcoPin, INPUT);
}

//ultrasonic distance measurement
int getDist(void)
{
    digitalWrite(TrgPin, LOW);
    delayMicroseconds(8);
    digitalWrite(TrgPin, HIGH);
    //Maintian high for 10ms to generate a pulse
    delayMicroseconds(10);
    digitalWrite(TrgPin, LOW);
    //Read the pulse width and convert to distance
    dist = pulseIn(EcoPin, HIGH) / 58;
    return dist;
}

void loop(void){
    if(getDist()<10){
        while(1){
            Serial.print("dist:");
            Serial.println(getDist());
            if(getDist()>30){
                count++;
                Serial.println(count);
                delay(100);
                break;
            }
        }
    }
    //Send the counted number every 10s, time (10)
    if(millis()-timestamp > time(10)){
        String data = (String)count;
        IOT.beebotteSendMessage(BEEBOTTE_CHANNEL, BEEBOTTE_RESOURCE,data.c_str());
        Serial.println(data.c_str());
        timestamp = millis();
        count = 0;
    }
}
```

## 5-4. Effect Display

Check the number of people in a certain period of time on BeeBotte Platform.

 Result

## Use with STM32

The WiFi IOT module can also be used with STM32 development board, and the functions achieved are the same. Here the details will be omitted. You can try it if you are interested.

WiFi IoT Module Library Address for STM32:

[https://github.com/DFRobot/DFRobot\\_WIFI\\_IOT\\_Module\\_STM32](https://github.com/DFRobot/DFRobot_WIFI_IOT_Module_STM32)

([https://github.com/DFRobot/DFRobot\\_WIFI\\_IOT\\_Module\\_STM32](https://github.com/DFRobot/DFRobot_WIFI_IOT_Module_STM32))

- Sample Code

```
#include "system.h"
#include "SysTick.h"
#include "led.h"
#include "usart.h"
#include "DFRobot_wifi_iot.h"

//wifi
#define WIFISSID      "hitest" //WiFi name
#define WIFIPWS      "12345678" //wifi password

//MQTT
//#define MQTT
#ifdef MQTT
#define SERVER        "iot.dfrobot.com.cn" //server address
#define PORT          "1883" //Port number
#define DEVICENAME    "rHpr0RcWR" //User name
#define DEVICESECRET  "9NtrAg5ZRz" //User login password
#define TOPIC         "0SpwrHHMg" //Subscribe channel
#endif

//IFTTT
#define IFTTT
#ifdef IFTTT
#define IFTTTKEY      "dtpfTlU3Wqa8y0HRh77xXE"
#define IFTTTEVENT   "BBB"
#endif

//ThingSpeak
//#define THINGSPEAK
#ifdef THINGSPEAK
#define THINGSPEAKKEY "U01NPZTC2G9WTDNY"
#endif

int main()
{
    u8 i=0;
    SysTick_Init(72);
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);
    LED_Init();
    //USB print port, corresponding pin: RX to PA10, TX to PA9
    USART1_Init(9600);
    //IOT communication port, corresponding pin: RX to PB11, TX to PB10
    USART3_Init(9600);
    //Connect to wifi
    connectWifi(WIFISSID,WIFIPWS);
```



```
//Access EASYIOT
#ifdef MQTT
mqtt(SERVER,PORT,DEVICENAME,DEVICSECRET, TOPIC);
#endif

#ifdef IFTTT
configIFTTT(IFTTTEVENT,IFTTKEY);
#endif

#ifdef THINGSPEAK
configThingSpeak(THINGSPEAKKEY);
#endif
while(1){


    #ifdef MQTT
    //send data shen using MQTT
    publish(TOPIC,"HI TANG");
    #endif
    #ifdef IFTTT
    //Access IFTTT, send messages to IFTTT registered event
    IFTTTSendMasage("100","78","78");
    #endif
    #ifdef THINGSPEAK
    //Access thingSpeak, send messages to ThingSpeak
    thingSpeakSendMasage("5000", "100");
    #endif
    i+=10;
    if(i%20==0)
    {
        i=0;
        led1=!led1;
    }

    delay_ms(100);
    loop();
}
}
```

## FAQ

For any questions, advice or cool ideas to share, please visit the **DFRobot Forum** (<https://www.dfrobot.com/forum/>).

## More Documents

 **DFshopping\_car1.png** Get **WiFi IoT Module** (<https://www.dfrobot.com/product-2180.html>)

from DFRobot Store or **DFRobot Distributor**. (<https://www.dfrobot.com/index.php?route=information/distributorslogo>)

**Turn to the Top**