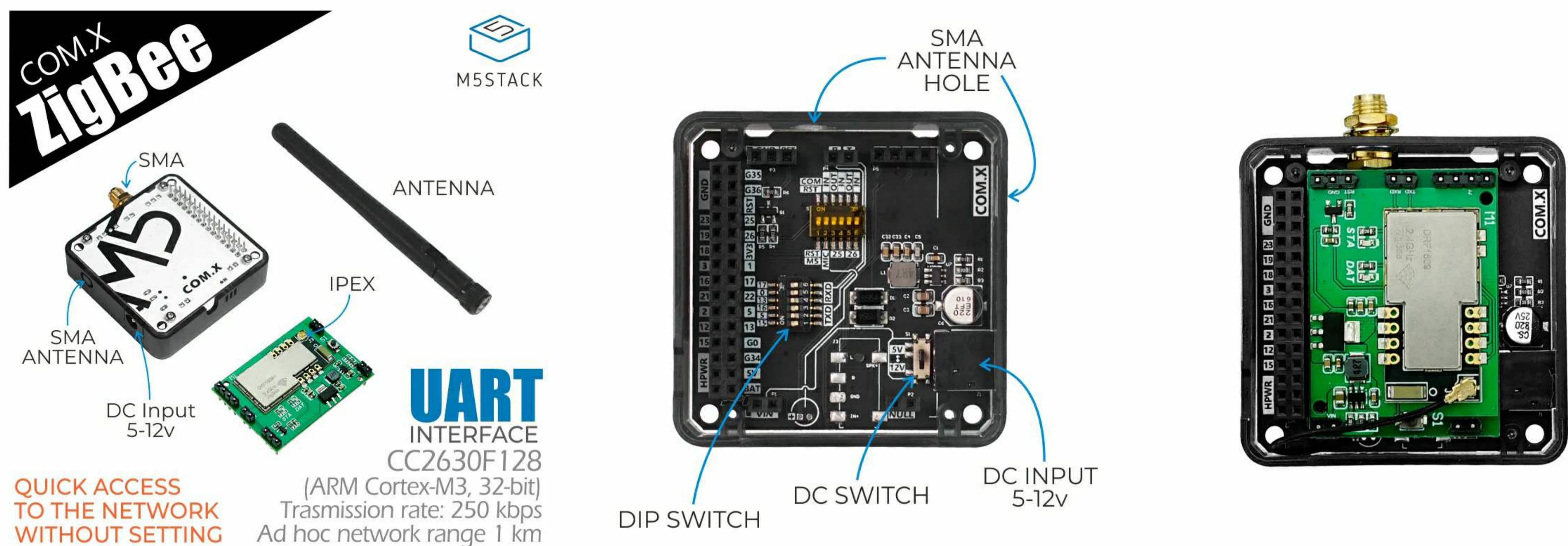


COM.Zigbee

SKU:M031-Z



Description

COM.Zigbee is a Zigbee self-organizing network communication module launched by M5Stack. The module adopts the CC2630F128 solution, internally integrates the Zigbee protocol stack, and opens the serial communication interface. Integrated external antenna, single node stable communication distance up to 1km, 200-level router depth, through the MESH networking mode, you can extend your IoT application in a wide range, with both ultra-low power consumption and high sensitivity. The Zigbee network can support hundreds of nodes and has enhanced security features. It can provide complete and interoperable IoT solutions for home and building automation.

Notice



When used with the **FIRE** main control, due to the PSRAM pin conflict, please switch the DIP switch pins of the module base to TX(0/13), RX(5/15), and the device can use USB/ External DC is used for power supply.



When used with **Core2** series main control, due to the different order of the base pin array, COM.X base pins TX(16), RX(17) correspond to the actual pins TX(14), RX of the Core2 main control (13) The equipment needs to use external DC for power supply.

Note: When using the CORE2 or CORE2-FOR-AWS main control to drive the module, please use the DC power input of the base and set the power mode to input mode (refer to the code below), otherwise it will not start normally

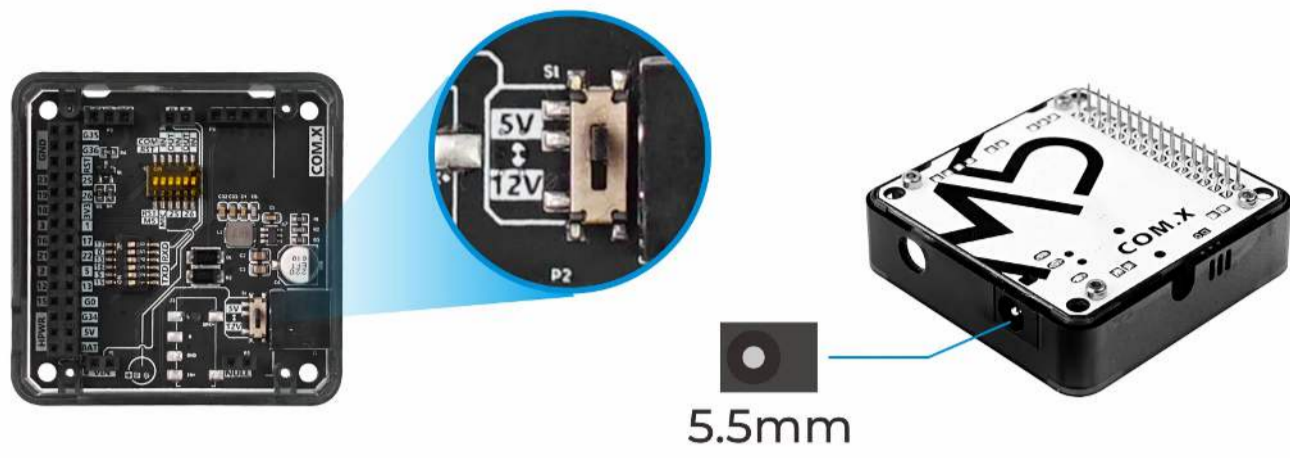
```
//mbus_mode_t:  
//kMBusModeOutput: Use USB or battery power  
//kMBusModeInput: Use external power supply 5V, or DC interface
```

```
M5.begin(true, true, true, false, kMBusModeInput);
```

```
//Initialize Serial according to the actual connected pins
```

```
Serial2.begin(115200, SERIAL_8N1, 13, 14);
```

The module base has a DC power input interface. Please strictly follow the input range (5-12V) to prevent damage to the module when using this interface to connect to the power supply. The internal power DIP switch can adjust the voltage level of the internal terminal VIN to adapt to different modules.



Product Features

- CC2630F128
- Quick access to the network without setting
 - initialize the coordinator and configure the router preset, which can realize the automatic access to the network by pressing the button three times
- Serial communication
- Low power consumption (module working current: 25mA, sleep 5uA)
- Dynamic routing maintenance, supporting 200-level routing depth
- Transmission speed 250Kbps
- Node communication distance 1km
- UART transparent transmission/broadcasting/P2P

Include

- 1x Zigbee Unit
- 1x SMA antenna

Applications

- Smart Home
- IoT collection node
- Building Automation

Specification

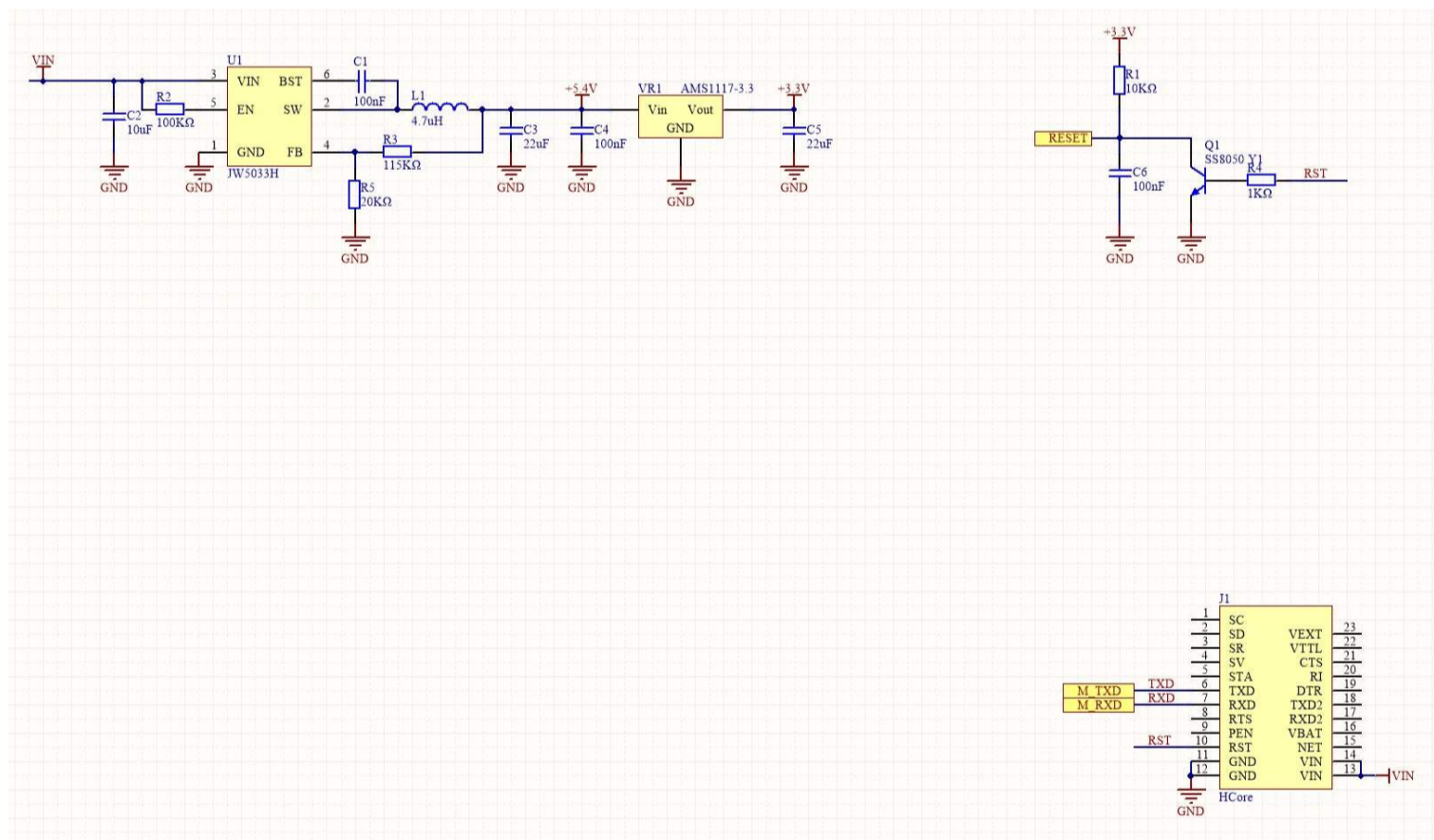
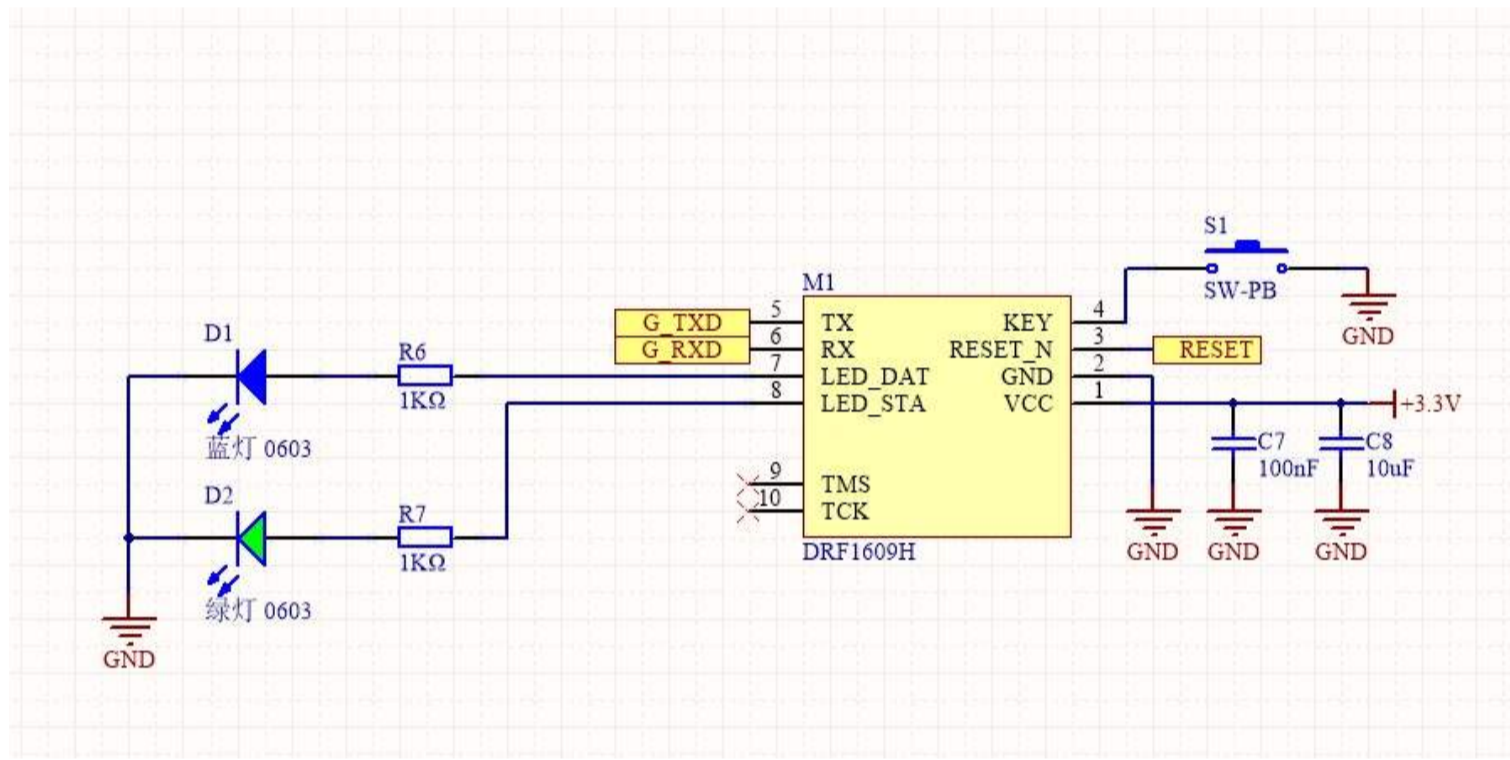
Resources	Parameter
CC2630F128	ARM Cortex-M3 32bit
communication	UART 38400bps 8N1(default)
Communication distance	1km (Open area)

working frequency	2.4GHZ (2405MHz-2480MHz, step:5MHz)
DC interface specifications	5.5mm
Net weight	37g
Gross weight	70g
Product Size	54*54*13.2mm
Package Size	165*60*36mm

PinMap

Core	TX(GPIO 0/13/17)	RX(GPIO 5/15/16)	5V	GND
COM.Zigbee	RX	TX	VIN	GND

Schematic



Example

- [Zigbee P2P CHAT ROOM](#)
- [Zigbee RSSI TEST](#)

EasyLoader

- [Zigbee P2P CHAT ROOM](#)
- [Zigbee RSSI TEST](#)
 - [Coordinator](#)
 - [End Device](#)

Related Link

- CC2630 Datasheet
- Module User Manual
- Instructions for use of the host computer
- PC debugging tool

Video

DRFZigbee.h - API

//Initialize the serial port of the module

```
void begin(HardwareSerial & uart){ _uartp = &uart;}
```

//Connect the module. After execution, the module will disconnect the wireless link and enter the configuration mode.

```
int linkMoudle();
```

//Read and write module configuration parameters

```
int readModuleparm(zigbee_arg_t *parm);
```

```
int setModuleparm(zigbee_arg_t &parm);
```

//Get network topology

```
int getNetworksTopology();
```

//After completing the configuration, you need to execute the program to restart the module and restore the wireless connection

```
int rebootModule();
```

//Get module signal quality

```
int8_t getModuleRSSI(nodeRSSI_t *nodeRSSIPtr = nullptr);
```

//Receive data

```
int reviceData(reviceData_t *revice,uint8_t type = kP2PCustomIDMode,size_t timeout = 1000);
```

//send data

```
void sendData(uint8_t cmd, const std::initializer_list<uint8_t> args);
```

```
int sendCMDAndWaitRevice(uint8_t cmd, byteArray &array, byteArray *reviceArray = nullptr, size_t timeout = 1000);
```

```
int sendCMDAndWaitRevice(uint8_t cmd, const std::initializer_list<uint8_t> args, byteArray *reviceArray = nullptr, size_t timeout = 1000);
```

```
int sendDataP2P(uint8_t mode,uint16_t addr,uint8_t *dataptr,size_t length);
```

```
int sendDataP2P(uint8_t mode,uint16_t addr,byteArray &array);
```

```
int sendDataP2P(uint8_t mode,uint16_t addr,const std::initializer_list<uint8_t> args);
```

//Module configuration parameter item

```
DRFZigbee::zigbee_arg_t *arg = new DRFZigbee::zigbee_arg_t;
```

```
uint8_t main_pointType;
```

```
uint16_t main_PANID;
```

```
uint8_t main channel;
```

```
uint8_t    main_transmissionMode;
uint16_t   main_customID;

uint16_t   main_res0;
uint8_t    main_uartBaud;
uint8_t    main_uartBit;
uint8_t    main_uatrtStop;
uint8_t    main_uartCheck;
uint16_t   main_res1;
uint8_t    main_ATN;
uint8_t    main_mac[8];
```

//Configure preset parameters-complete configuration preset parameters can be used to quickly access the network from the node without settings

```
uint8_t    preset_pointType;
uint16_t   preset_PANID;
uint8_t    preset_channel;
uint8_t    preset_transmissionMode;
uint16_t   preset_customID;
```

//reserved

```
uint16_t   preset_res0;
```

```
uint8_t    preset_uartBaud;
uint8_t    preset_uartBit;
uint8_t    preset_uatrtStop;
uint8_t    preset_uartCheck;
```

//reserved

```
uint16_t   preset_res1;
```

```
uint8_t    preset_ATN;
uint16_t   shortAddr;
uint8_t    res3;
uint8_t    encryption;
uint8_t    password[4];
```