

Gravity: UART OBLOQ IoT Module

SKU: TEL0118



The core of IoT is user experience. It integrates embedded system technology, mobile technology and network technology all together, making up a smart system. It unifies hardware, software and network services, makes them become a whole.

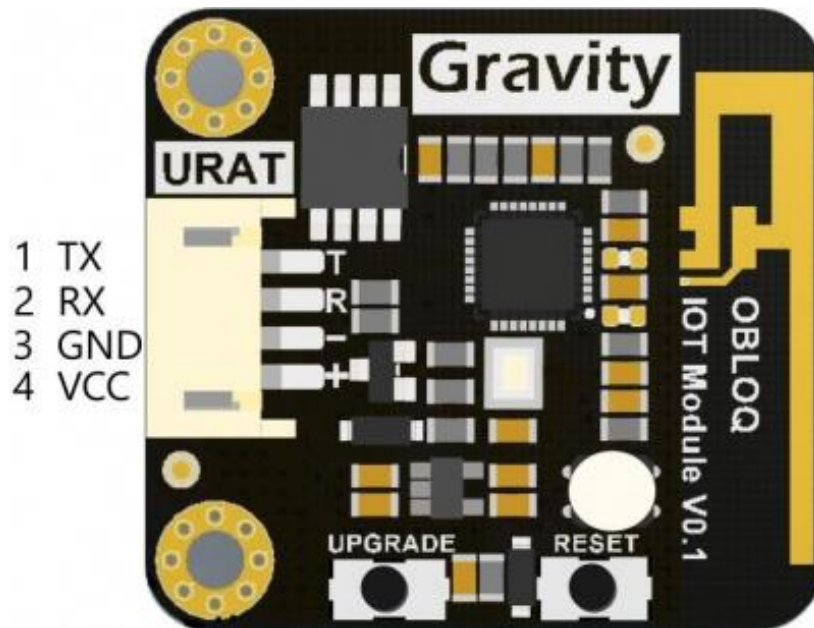
Gravity: UART OBLOQ module is a WiFi to Serial device. It faces to non-professional developers, supports standard MQTT protocol IOT service such as Microsoft Azure IOT. By OBLOQ module, you can quickly build the IoT applications without having a complex background knowledge of IoT. In this way, makers can really focus on the purpose of creating interesting things on the IoT.

OBLOQ is designed based on ESP8266 WiFi chip, it has compact size, low price, simple interface, plug and play feature, and it works stably under the 3.3V ~ 5V control system.

Specifications

- Supply Voltage: 3.3~5.0V
- Operating Current: <240mA
- Interface Type: Gravity UART 4PIN
- Baud rate: 9600 bps
- Wireless mode: IEEE802.11b/g/n
- Encryption type: WPA WPA2/WPA2-PSK
- Wireless frequency: 2.4GHz
- Product Size: 35mm * 32mm / 1.38inch * 1.26inch
- Built-in agreement: TCP/IP Protocol stack
- Weight:16g

Board Overview



OBLOQ pin Description

Pin definition		
Num	Label	Description
1	TX	Serial port sender
2	RX	Serial port receiver
3	GND	GND
4	VCC	Power + (3.3/5V)

Arduino OBLOQ Tutorial

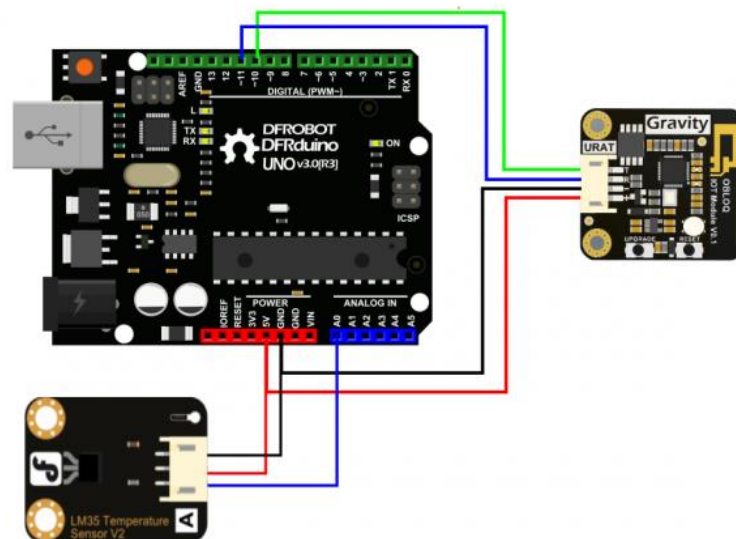
OBLOQ module has two basic functions: send data to IoT and receive IoT data. Here's an experimental demonstration of OBLOQ connecting Azure IOT device and sending data to Azure IOT.

- The Arduino reads data from temperature sensor LM35 and sends the data to the Azure IOT device via the OBLOQ module.

Requirements

- Hardware
DFRduino UNO R3 (or similar) x 1
Gravity: Analog LM35 Temperature Sensor For Arduino x1
1 x OBLOQ module
some connecting wires
- Software
Arduino IDE, Click to Download Arduino IDE from Arduino®
- Create Azure IOT device
Register a device in the IoT hub for your device
Make a note of the primary key of the connection string.
Device Explorer tool and how to use it

Connection Diagram



Arduino OBLOQ and LM35 Module Connection

- TX-Pin10,
- RX-Pin11
- A0-LM35
- GND-GND
- VCC-VCC

Sample code

- Functions realized: Arduino reads temperature data from LM35, and sends the data to Azure IOT device via OBLOQ module.

```
#include <Arduino.h>
#include <SoftwareSerial.h>

//wifi SSID and password
#define WIFISSID "DFRobot-guest"
#define WIFIPWD "dfrobot@2017"

//Azure IOT device connection string, this string need modification when connect to different devices
const String connectionString = "HostName=dfrobot.azure-devices.cn;DeviceId=temperature;SharedAccessKey=CR5gUclNaT7skX9WP+e6oIB/BnkZnTIEReKBX870SNY=";
const String separator = "|";
bool pingOn = true;
bool createIoTClientSuccess = false;
static unsigned long previoustGetTempTime = 0;
static unsigned long pingInterval = 2000;
static unsigned long sendMessageInterval = 10000;
unsigned long previousPingTime = 0;
String receiveStringIndex[10] = {};

//Disconnected wifi detection variable
bool wifiConnect = false;
bool wifiAbnormalDisconnect = false;

SoftwareSerial softSerial(10,11);

enum state{
```

```

    WAIT,
    PINGOK,
    WIFIOK,
    AZURECONNECT
}obloqState;

/*****
*****

Function      : sendMessage
Description   : send message via serial port
Params       : message : message content, it is a string
Return       : None
*****
*****/

void sendMessage(String message)
{
    softSerial.print(message+"\r");
}

/*****
*****

Function      : ping
Description   : Check if the serial communication is normal. When Obloq serial
port receives the pingcommand, it will return: |1|1|\r
Return       : None
*****
*****/

void ping()
{
    String pingMessage = F("|1|1|");
    sendMessage(pingMessage);
}

/*****
*****

Function      : connectWifi

```

Description : Connect wifi, once successfully connected Obloq returns: |2|3|i
p|\r

Params : ssid : wifi SSID

Params : pwd : wifi password

Return : None

```
*****  
*****/
```

```
void connectWifi(String ssid,String pwd)
```

```
{  
    String wifiMessage = "|2|1|" + ssid + "," + pwd + separator;  
    sendMessage(wifiMessage);  
}
```

```
/*****  
*****
```

Function : createIoTClient

Description : Create device client handle, once successfully created Obloq re
turns: |4|2|1|1|\r

Params : connecttionString device connection string

Return : None

```
*****  
*****/
```

```
void createIoTClient(String connecttionString)
```

```
{  
    String azureConnectMessage = "|4|2|1|" + connecttionString + separator;  
    sendMessage(azureConnectMessage);  
}
```

```
/*****  
*****
```

Function : subscribeDevice

Description : Subscribe to device. Obloq returns message content after the de
vice receives message.

Params : None

Return : None

```
*****  
*****/
```

```
void subscribeDevice()
```

```

{
    String subscribeDeviceMessage = "|4|2|2|";
    sendMessage(subscribeDeviceMessage);
}

/*****
*****

Function      : unsubscribeDevice
Description   : Unsubscribe to device. Once successfully unsubscribed, Obloq re
turns: |4|2|6|1|\r
Params       : None
Return       : None
*****/

void unsubscribeDevice()
{
    String unsubscribeDeviceMessage = "|4|2|6|";
    sendMessage(unsubscribeDeviceMessage);
}

/*****
*****

Function      : publish
Description   : Publish message. Before this command you must create device cli
ent handle first.
Params       : message: message content to publish, once successfully sent, OB
loq returns : |4|2|3|1|\r
Return       : None
*****/

void publish(String message)
{
    String publishMessage = "|4|2|3|" + message + separator;
    sendMessage(publishMessage);
}

```

```

/*****
*****

Function      : disconnect
Description   : destroy device client handle: |4|2|4|1|\r
Params       : None
Return       : None

*****/

void distoryIotClient()
{
    String distoryIotClientMessage = "|4|2|4|";
    sendMessage(distryIotClientMessage);
}

/*****
*****

Function      : recreateIoTClient
Description   : Recreate device client handle, once created successfully, Obloq
returns      : |4|2|1|1|\r
Params       : None
Return       : None

*****/

void recreateIoTClient()
{
    String recreateIoTClientMessage = "|4|2|5|";
    sendMessage(recreateIoTClientMessage);
}

/*****
*****

Function      : splitString
Description   : split a string
Params       : data: save the cutted char array
Params       : str: the string that will be cutted
Params       : delimiters: the delimiter that cut string
Return       : None

```



```

*****
*****/
int splitString(String data[],String str,const char* delimiters)
{
    char *s = (char *) (str.c_str());
    int count = 0;
    data[count] = strtok(s, delimiters);
    while(data[count]){
        data[++count] = strtok(NULL, delimiters);
    }
    return count;
}

/*****
*****

Function      : handleUart
Description   : handle data received by serial port
Params        : None
Return        : None

*****
*****/

void handleUart()
{
    while(softSerial.available() > 0)
    {
        String receivedata = softSerial.readStringUntil('\r');
        const char* obloqMessage = receivedata.c_str();
        if(strcmp(obloqMessage, "|1|1|") == 0)
            {
                Serial.println("Pong");
                pingOn = false;
                obloqState = PINGOK;
            }
        if(strcmp(obloqMessage, "|2|1|") == 0)
            {
                if(wifiConnect)

```

```

        {
            wifiConnect = false;
            wifiAbnormalDisconnect = true;
        }
    }

    else if(strstr(obloqMessage, "|2|3|") != 0 && strlen(obloqMessage) != 9)
    {
        Serial.println("Wifi ready");
        wifiConnect = true;
        if(wifiAbnormalDisconnect)
        {
            wifiAbnormalDisconnect = false;
            createIoTClientSuccess = true;
            return;
        }

        obloqState = WIFIOK;
    }
    else if(strcmp(obloqMessage, "|4|2|1|1|") == 0)
    {
        Serial.println("Azure ready");
        createIoTClientSuccess = true;
        obloqState = AZURECONNECT;
    }
}

}

/*****
*****

Function    : sendPing
Description : Check if serial port is communicating normally
Params      : None
Return      : None

*****/

void sendPing()

```

```

{
    if(pingOn && millis() - previousPingTime > pingInterval)
    {
        previousPingTime = millis();
        ping();
    }
}

/*****
*****

Function      : execute
Description   : send different command according to different status.
Params       : None
Return       : None
*****/

void execute()
{
    switch(obloqState)
    {
        case PINGOK: connectWifi(WIFISSID,WIFIPWD); obloqState = WAIT; break;
        case WIFIOK: createIoTClient(connectionString);obloqState = WAIT; bre
ak;
        case AZURECONNECT : obloqState = WAIT; break;
        default: break;
    }
}

/*****
*****

Function      : getTemp
Description   : Get temperature measured by LM35
Params       : None
Return       : None
*****/

```

```

float getTemp()
{
    uint16_t val;
    float dat;
    val=analogRead(A0);//Connect LM35 on Analog 0
    dat = (float) val * (5/10.24);
    return dat;
}

/*****
*****

Function      : checkWifiState
Description   : Callback function of receiving message
Params       : message: message content string that received
Return       : None
*****
*****/

void checkWifiState()
{
    static unsigned long previousTime = 0;
    if(wifiAbnormalDisconnect && millis() - previousTime > 60000) // reconne
ct once per minute after wifi abnormally disconnect
    {
        previousTime = millis();
        createIoTClientSuccess = false;
        connectWifi(WIFISSID,WIFIPWD);
    }
}

void setup()
{
    Serial.begin(9600);
    softSerial.begin(9600);
}

void loop()

```

```
{
  sendPing();
  execute();
  handleUart();
  checkWifiState();
  //Sending data every 5 seconds
  if(createIoTClientSuccess && millis() - previoustGetTempTime > 5000)
  {
    previoustGetTempTime = millis();

    //Get data from temperature sensor
    float temperature = getTemp();
    publish((String)temperature);
    Serial.println(temperature);
  }
}
```

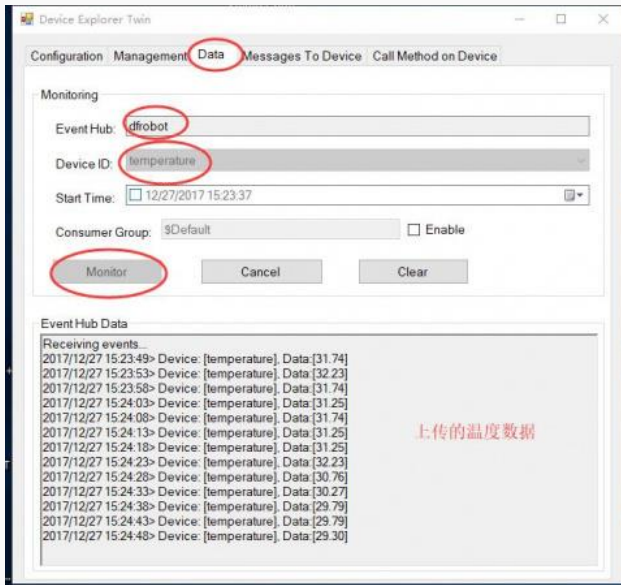
Result demonstration

After successfully uploading the program to Arduino, reset both OBLOQ module and Arduino. When not connected to the wifi, OBLOQ shows red light, but once connected, OBLOQ shows green light.

Use the tool Device explorer, in menu "Data", select the device that you are going to send message to, then monitor it. You will see the uploaded message in the EventHub Data window.

- Results show

```
COM17  
  
Wifi ready  
Azure ready  
31.74  
32.23  
31.74  
31.25  
31.74  
31.25  
31.25  
32.23  
30.76  
30.27  
29.79  
29.79  
29.30
```



More Documents

[Sample code](#)

https://www.dfrobot.com/wiki/index.php/Gravity:_UART_OBLOQ_IoT_Module_SKU:_TEL0118_2-6-18