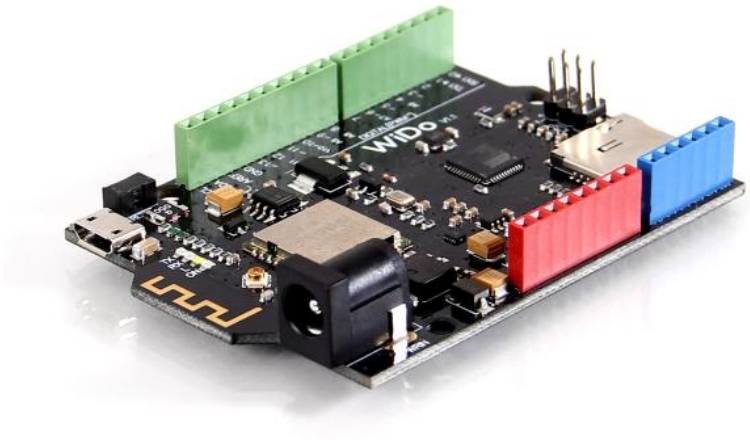
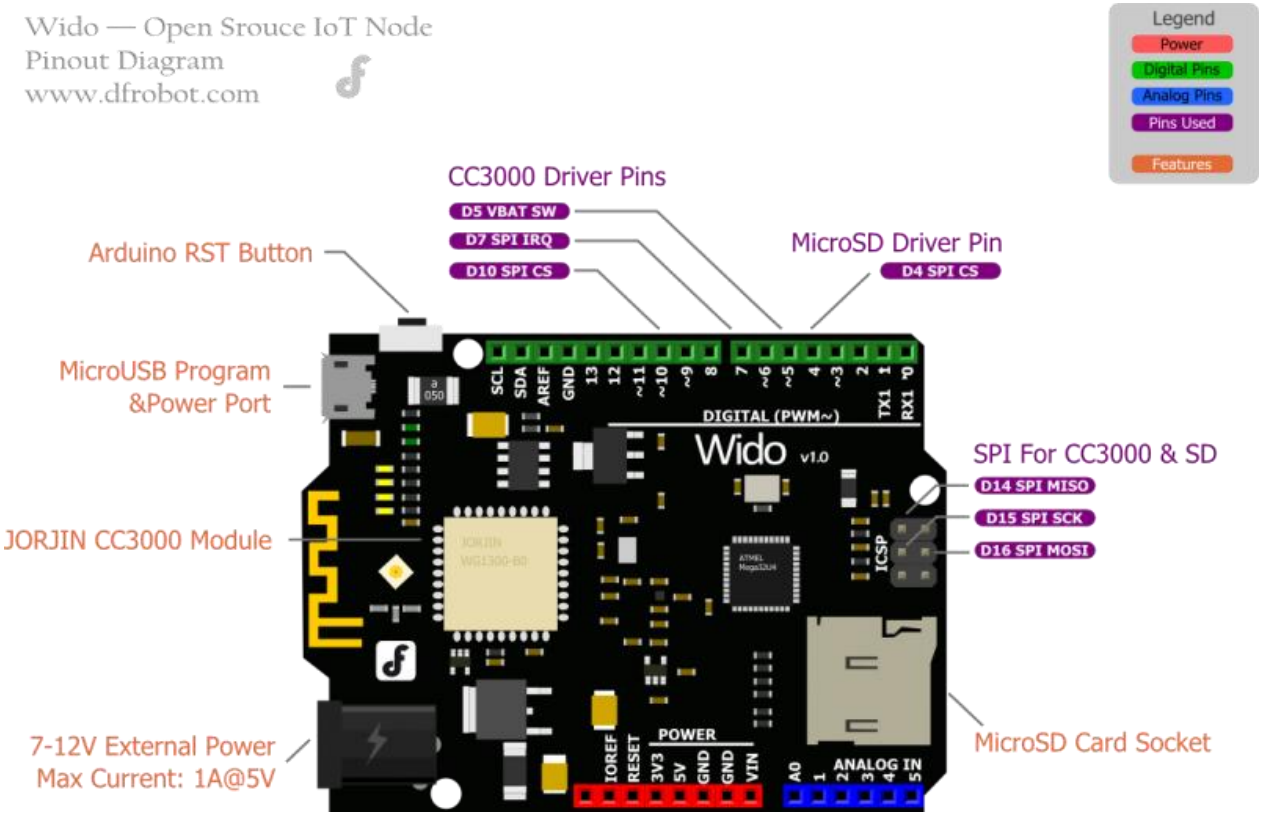
Wido-WIFI IoT Node
Model:DFR0321
User Manual



PinOut



Tutorial

Tutorial 1

First of all, we will bring you a step by step tutorial to lead finish the Wido router connection configuration and make it work as a TCP client connected to the local server.

Step 1

Wido 1unit
MicroUSB Cable 1unit

Step 2

1. Install the Arduino library to your Arduino IDE. This library for Wido is forked from Adafruit. They've finished an awesome project for this CC3000 development. Based on this library, we updated the pin configuration and extended some application sample codes.

2. Open the sample code, which is named build test.

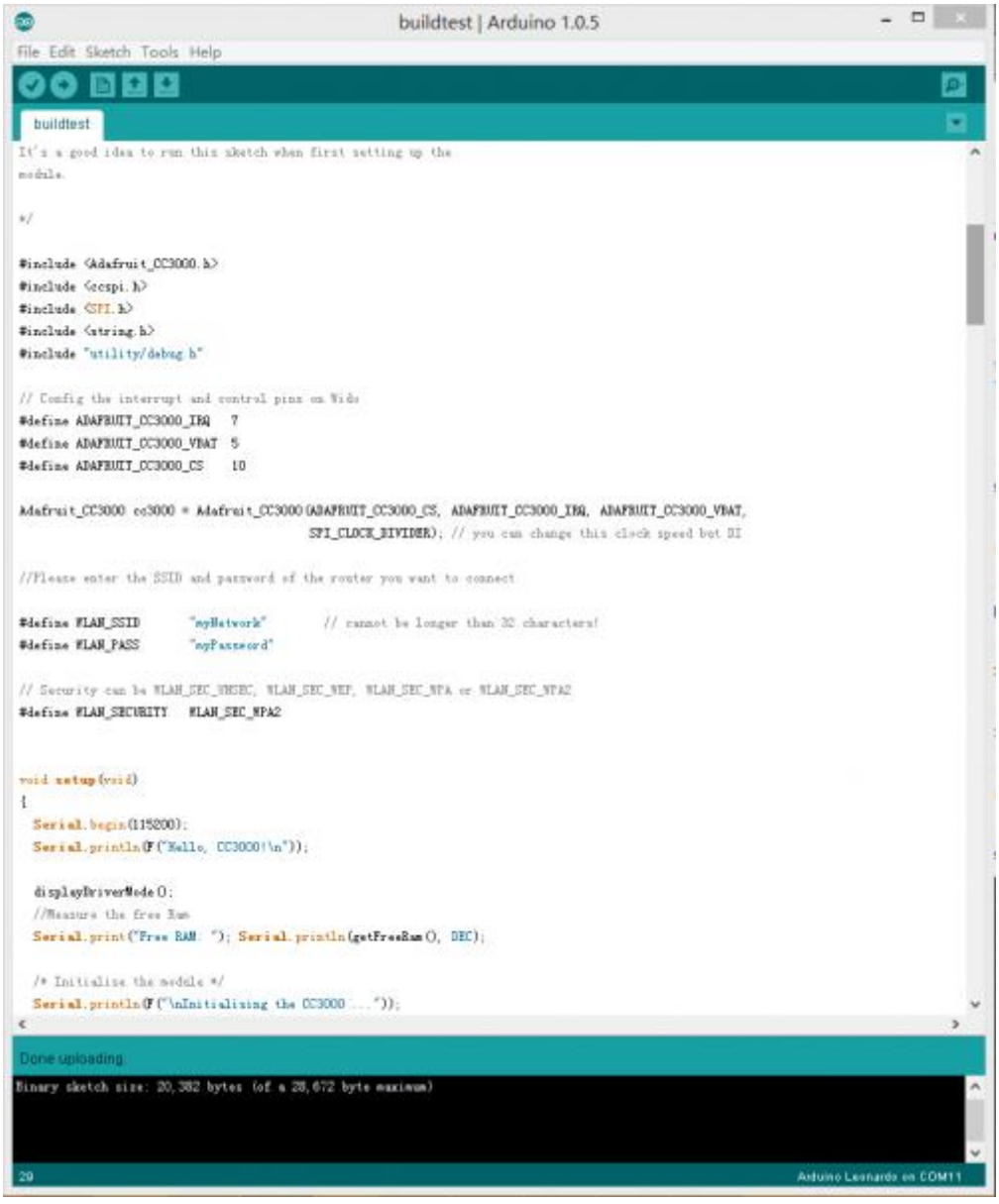


Fig2: build test

3. Upload the sample code to Wido and check the Serial monitor after programming.

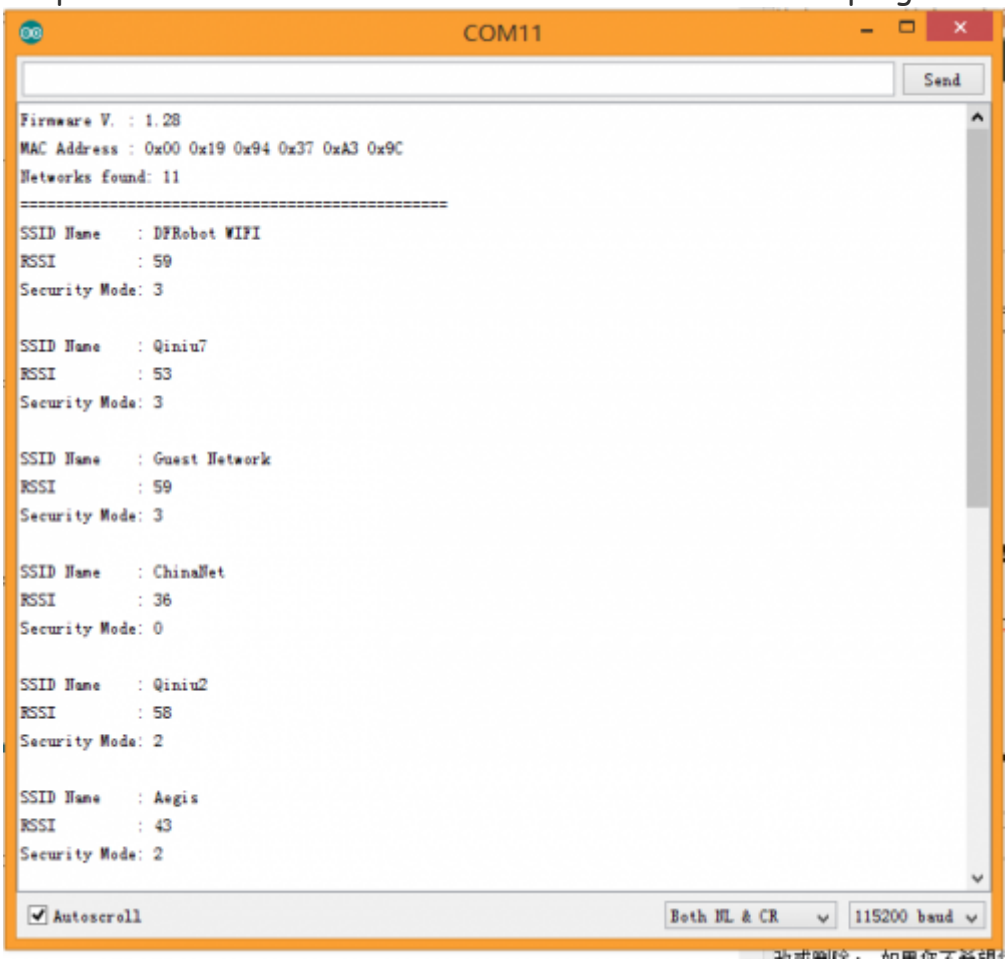


Fig3: Scan the Routers

You will see the information printed including the MAC address and local router information detected by Wido.

4. Update the SSID and password configuration in your code!

```
//Please enter the SSID and password of the router you want to connect
#define WLAN_SSID       "myNetwork"        // cannot be longer than 32 characters!
#define WLAN_PASS       "myPassword"
```

5. Then upload the sample sketch again. And after several seconds. You will see the effect, like the picture attached.

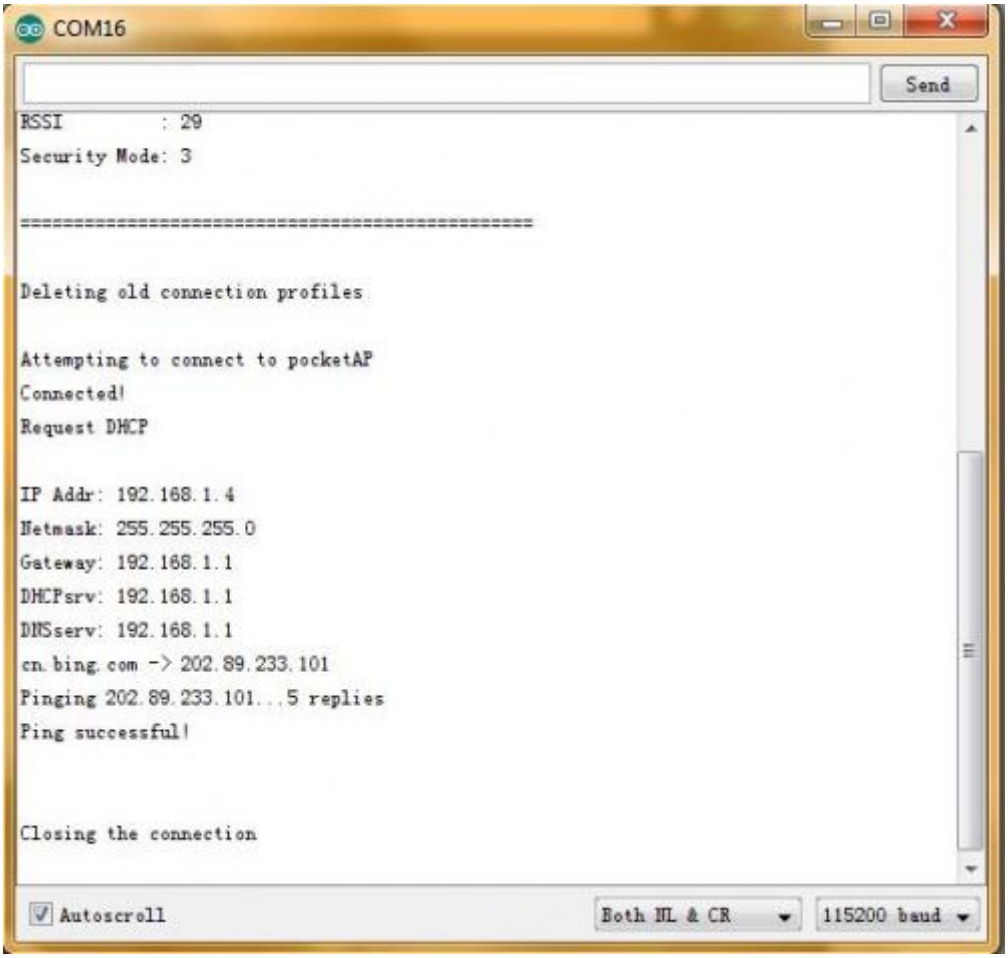


Fig4: Ping bing.com

Tutorial 2

Step 1

TCP Server Tool, used to create a local TCP server from your PC

Step 2

1. Open the Tool above. Config the port number, and click listening to wait for the client connection.

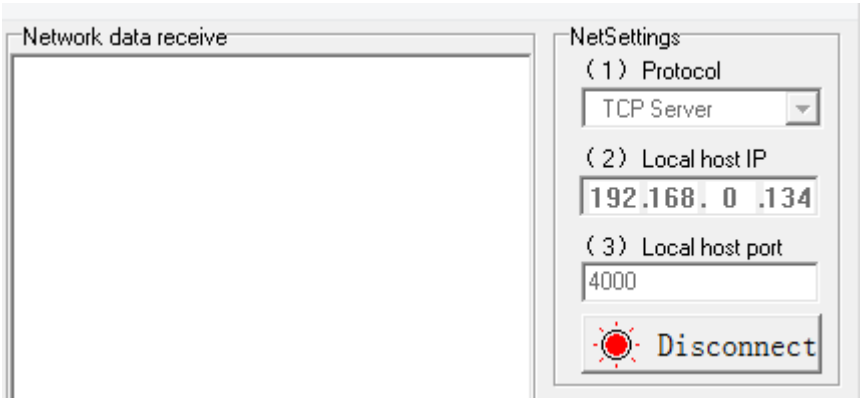


Fig1: Open TCP Server

2. Open the sketch named "Wido2LocalTcpServer", and config the TCP server address and port according to your tool setup.

```
/* Set the target ip address and connection port */
  uint32_t ip = WiDo.IP2U32(192,168,0,134);
  tcpClient = WiDo.connectTCP(ip, 9000);
```

Upload the full sample after TCP/Router configuration.

```
#include <Adafruit_CC3000.h>
#include <ccspi.h>
#include <SPI.h>
#include <string.h>
#include "utility/debug.h"

#define WiDo_IRQ   7
#define WiDo_VBAT  5
#define WiDo_CS    10
Adafruit_CC3000 WiDo = Adafruit_CC3000(WiDo_CS, WiDo_IRQ, WiDo_VBAT,
                                       SPI_CLOCK_DIVIDER); // you can change this clo
ck speed

#define WLAN_SSID       "myNetwork"           // cannot be longer than 32 characters!
#define WLAN_PASS       "myPassword"

// Security can be WLAN_SEC_UNSEC, WLAN_SEC_WEP, WLAN_SEC_WPA or WLAN_SEC_WPA2
#define WLAN_SECURITY   WLAN_SEC_WPA2
#define TIMEOUT_MS  1000

void setup(){

  Serial.begin(115200);
  /* Initialise the module */
  Serial.println(F("\nInitialising the CC3000 ..."));
  if (!WiDo.begin())
  {
    Serial.println(F("Unable to initialise the CC3000! Check your wiring?"));
    while(1);
  }

  /* NOTE: Secure connections are not available in 'Tiny' mode!
     By default connectToAP will retry indefinitely, however you can pass an
     optional maximum number of retries (greater than zero) as the fourth parameter.
  */
```

```
  Serial.println(F("Connecting Router/AP"));
  if (!WiDo.connectToAP(WLAN_SSID, WLAN_PASS, WLAN_SECURITY)) {
    Serial.println(F("Failed!"));
    while(1);
  }

  Serial.println(F("Router/AP Connected!"));

  /* Wait for DHCP to complete */
  Serial.println(F("Request DHCP"));
  while (!WiDo.checkDHCP())
  {
    delay(100); // ToDo: Insert a DHCP timeout!
  }
}

void loop(){

  static Adafruit_CC3000_Client tcpClient;
  static unsigned long heartRate = millis();

  if(!tcpClient.connected()){
    Serial.println("Try to connect the Local Server");
    tcpClient.close();

    /* Set the target ip address and connection port */
    uint32_t ip = WiDo.IP2U32(192,168,0,134);
    tcpClient = WiDo.connectTCP(ip, 4000);

    if(!tcpClient.connected()){
      Serial.println(F("Couldn't connect to server! Make sure TCP Test Tool is running
on the server."));
      while(1);
    }
  }
  else if(millis() - heartRate > 1000){
    heartRate = millis();  // Update time stamp of the microcontroller system

    char clientString[30];
    sprintf(clientString, "%s%d%s", "Wido heartRate: ",heartRate/1000," s\r\n");
```

```
    Serial.println(clientString);
    tcpClient.fastrprintln(clientString);
  }

  /* Read data until either the connection is closed, or the timeout is reached. */
  unsigned long lastRead = millis();
  while (tcpClient.connected() && (millis() - lastRead < TIMEOUT_MS)) {
    while (tcpClient.available()) {
      char c = tcpClient.read();
      Serial.print(c);
      lastRead = millis();

      // Disable sending message for a moment
      heartRate = millis();
    }
  }
}
```

3. Open the serial monitor. After connecting the router, your Wido will start to upload data to the TCP server tool!

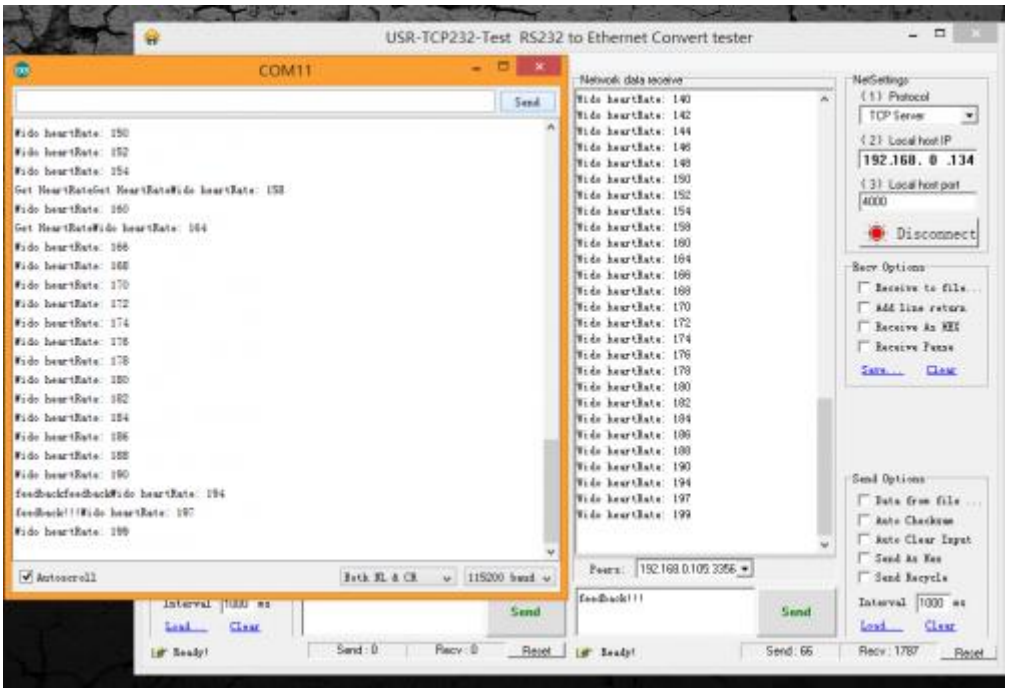


Fig1: Open TCP Server

Finish the local server connection and communication now. You get the access to build a WIFI controlled robot with Wido or some local cloud service project.But it's not enough for most of WIFI application!

Tutorial 3

Xively (formerly Cosm) is a Platform as a Service that provides everything you need to simpify and accelerate the creation of compelling connected products and solutions. In this section, we will bring your sensor to the **cloud**.☺

Step 1

1. Create your own Xively account and login the develop page. Create the a new device.

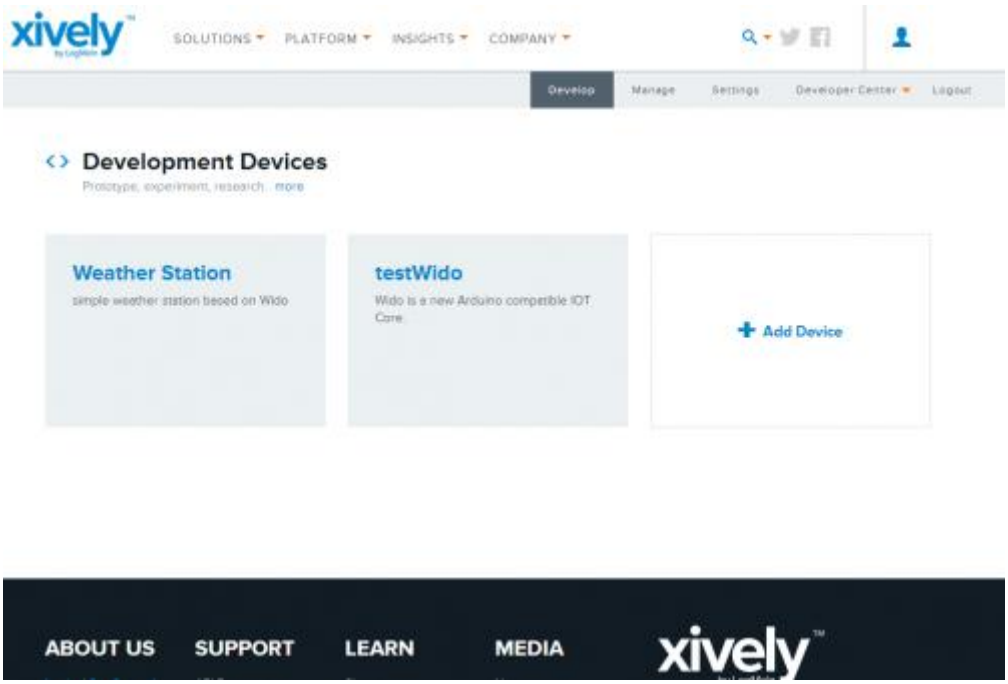


Fig1: Create Xively Account

2. Click and enter your device.

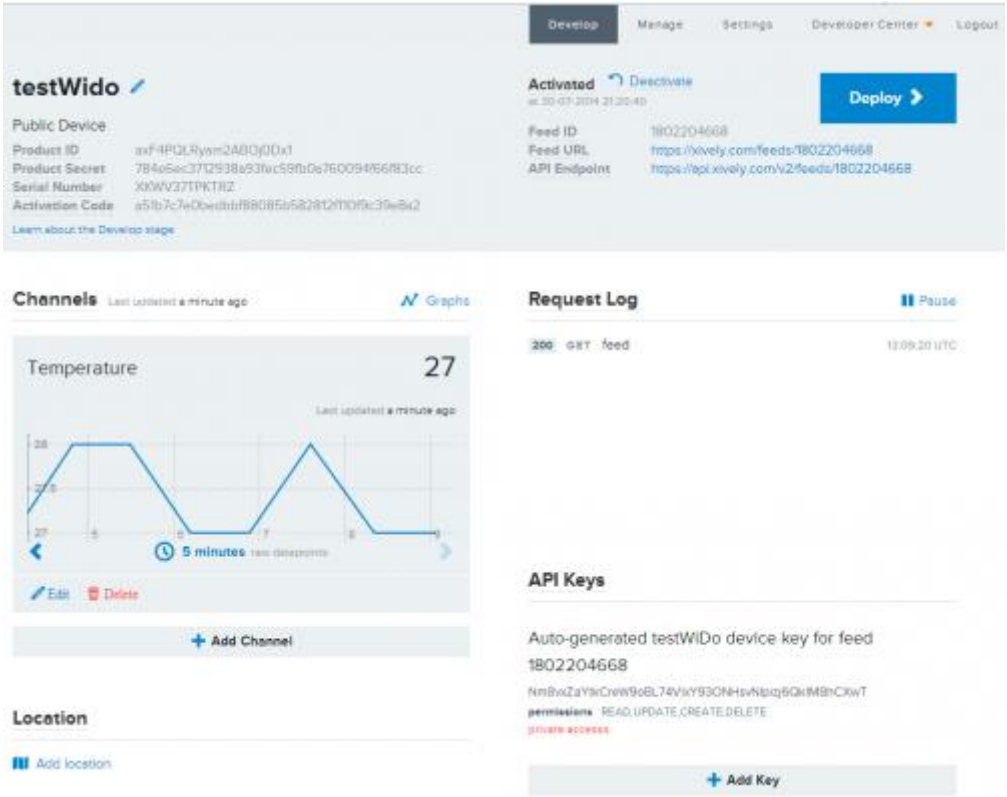


Fig2: Open the device page

3. Open the example code named "Wido2Xively" included in the Arduino library. Modify the info below in the sample code according to the device information from the step 2.

```
#define WEBSITE  "api.xively.com"
#define API_key  "Nm8vxZaYtkCreW9oBL74VIxY93ONHsvNlpizj6QkIM8hxxxx"  // Check your API Key from device page
#define feedID  "180220xxxx"                                        // Check your feed ID
```

The sample code:

```
/*************************************************
 * This is an example for the DFRobot Wido - Wifi Integrated IoT lite sensor and control node
 *
 * Designed specifically to work with the DFRobot Wido products:
 *
 *
 * The main library is forked from Adafruit
 *
 * Written by Lauren
 * BSD license, all text above must be included in any redistribution
 *
 *************************************************/
/*
This example code is used to connect the Xively cloud service.

The device required is just:

1. LM35 low cost temperature sensor or any device you used to upload data
2. And Wido

*/



#include <Adafruit_CC3000.h>
#include <ccspi.h>
#include <SPI.h>

#define Wido_IRQ   7
#define Wido_VBAT  5
```

```
#define Wido_CS    10
Adafruit_CC3000 Wido = Adafruit_CC3000(Wido_CS, Wido_IRQ, Wido_VBAT,
SPI_CLOCK_DIVIDER); // you can change this clock speed

#define WLAN_SSID       "myNetwork"           // cannot be longer than 32 characters!
#define WLAN_PASS       "myPassword"
// Security can be WLAN_SEC_UNSEC, WLAN_SEC_WEP, WLAN_SEC_WPA or WLAN_SEC_WPA2
#define WLAN_SECURITY   WLAN_SEC_WPA2

#define IDLE_TIMEOUT_MS  2000
#define TCP_TIMEOUT      3000

#define WEBSITE  "api.xively.com"
#define API_key  "Nm8vxZaYtkCreW9oBL74VIxY93ONHsvNlpizj6QkIM8hxxxx"  // Update Your API
Key
#define feedID  "180220xxxx"                                        // Update Your own
feedID

void setup(){

  Serial.begin(115200);
  Serial.println(F("Hello, CC3000!\n"));

  /* Initialise the module */
  Serial.println(F("\nInitialising the CC3000 ..."));
  if (!Wido.begin())
  {
    Serial.println(F("Unable to initialise the CC3000! Check your wiring?"));
    while(1);
  }

  /* Attempt to connect to an access point */
  char *ssid = WLAN_SSID;             /* Max 32 chars */
  Serial.print(F("\nAttempting to connect to "));
  Serial.println(ssid);

  /* NOTE: Secure connections are not available in 'Tiny' mode!
   By default connectToAP will retry indefinitely, however you can pass an
   optional maximum number of retries (greater than zero) as the fourth parameter.
   */
  if (!Wido.connectToAP(WLAN_SSID, WLAN_PASS, WLAN_SECURITY)) {
```

```
    Serial.println(F("Failed!"));
    while(1);
  }

  Serial.println(F("Connected!"));

  /* Wait for DHCP to complete */
  Serial.println(F("Request DHCP"));
  while (!Wido.checkDHCP())
  {
    delay(100); // ToDo: Insert a DHCP timeout!
  }

}

uint32_t ip = 0;    // Store Xively ip address
float temp = 0;     // Store temporary sensor data for post

void loop(){

  static Adafruit_CC3000_Client WidoClient;
  static unsigned long RetryMillis = 0;  // timer stamp for building the connection
  static unsigned long uploadtStamp = 0; // timer stamp for posting data to service
  static unsigned long sensortStamp = 0; // timer stamp for reading data to LM35

  // Apply for the connection with the cloud service

  if(!WidoClient.connected() && millis() - RetryMillis > TCP_TIMEOUT){
    // Update the time stamp
    RetryMillis = millis();

    Serial.println(F("Try to connect the cloud server"));

    //Get Xively IOT Server IP
    ip = Wido.IP2U32(216,52,233,120);
    WidoClient = Wido.connectTCP(ip, 80);
  }

  // After building the connection with the service
  // Post the sensor data to Xively
```

```
if(WidoClient.connected() && millis() - uploadtStamp > 2000){
  uploadtStamp = millis();
  // If the device is connected to the cloud server, upload the data every 2000ms.

  // Prepare JSON for Xively & get length
  int length = 0;
  // JSON beginning
  String data_start = "";
  data_start = data_start + "\n"
    + "{\"version\":\"1.0.0\",\"datastreams\" : [ ";
  // JSON for temperature & humidity
  String data_temperature = "{\"id\" : \"Temperature\",\"current_value\" : \""
    + String(int(temp)) + "\"}]}";
  // Get length
  length = data_start.length() + data_temperature.length();

  Serial.println(F("Connected to Xively server."));

  // Send headers
  Serial.print(F("Sending headers"));
  WidoClient.fastrprint(F("PUT /v2/feeds/"));
  WidoClient.fastrprint(feedID);
  WidoClient.fastrprintln(F(".json HTTP/1.0"));
  Serial.print(F("."));
  WidoClient.fastrprintln(F("Host: api.xively.com"));
  Serial.print(F("."));
  WidoClient.fastrprint(F("X-ApiKey: "));
  WidoClient.fastrprintln(API_key);
  Serial.print(F("."));
  WidoClient.fastrprint(F("Content-Length: "));
  WidoClient.println(length);
  Serial.print(F("."));
  WidoClient.fastrprint(F("Connection: close"));
  Serial.println(F(" done."));

  // Send data
  Serial.print(F("Sending data"));
  WidoClient.fastrprintln(F(""));
  WidoClient.print(data_start);
  Serial.print(F("."));
```

```
    WidoClient.print(data_temperature);
    Serial.print(F("."));
    WidoClient.fastrprintln(F(""));
    Serial.println(F(" done."));

    /* Get the http page info
    Serial.println(F("Reading answer..."));
    while (WidoClient.connected()) {
      while (WidoClient.available()) {
        char c = WidoClient.read();
        Serial.print(c);
      }
    }
    */
    delay(1000);              // Wait for 1s to finish posting the data stream
    WidoClient.close();       // Close the service connection
    RetryMillis = millis();  // Reset the timer stamp for applying the connection with
the service
  }


  //Realtime update the latest sensor data from LM35 once per 100ms and convert the uni
t (degree)
  if(millis() - sensortStamp > 100){
    sensortStamp = millis();
    // read the LM35 sensor value and convert to the degrees every 100ms.

    int reading = analogRead(0);
    temp = reading *0.0048828125*100;
    Serial.print(F("Real Time Temp: "));
    Serial.println(temp);
  }


}
```

4. Then Wido will upload the sensor data to the cloud once every 2s. You could check the Request Log and the Channel info from the device page now.

Step 2

The Adafruit library for CC3000 is really good and extending lots of feature for the WG1300. This library is also modified based on the TI smartlink solution.

Here're some simple introduction for the functions commonly used!

1. Trick for saving the programming space. The ATmega32U4 programming space is limited. So call the feature is really useful for your program.

```
#define CC3000_TINY_DRIVER
```

The code above will launch the tiny driver function.

2. Initialise the module.

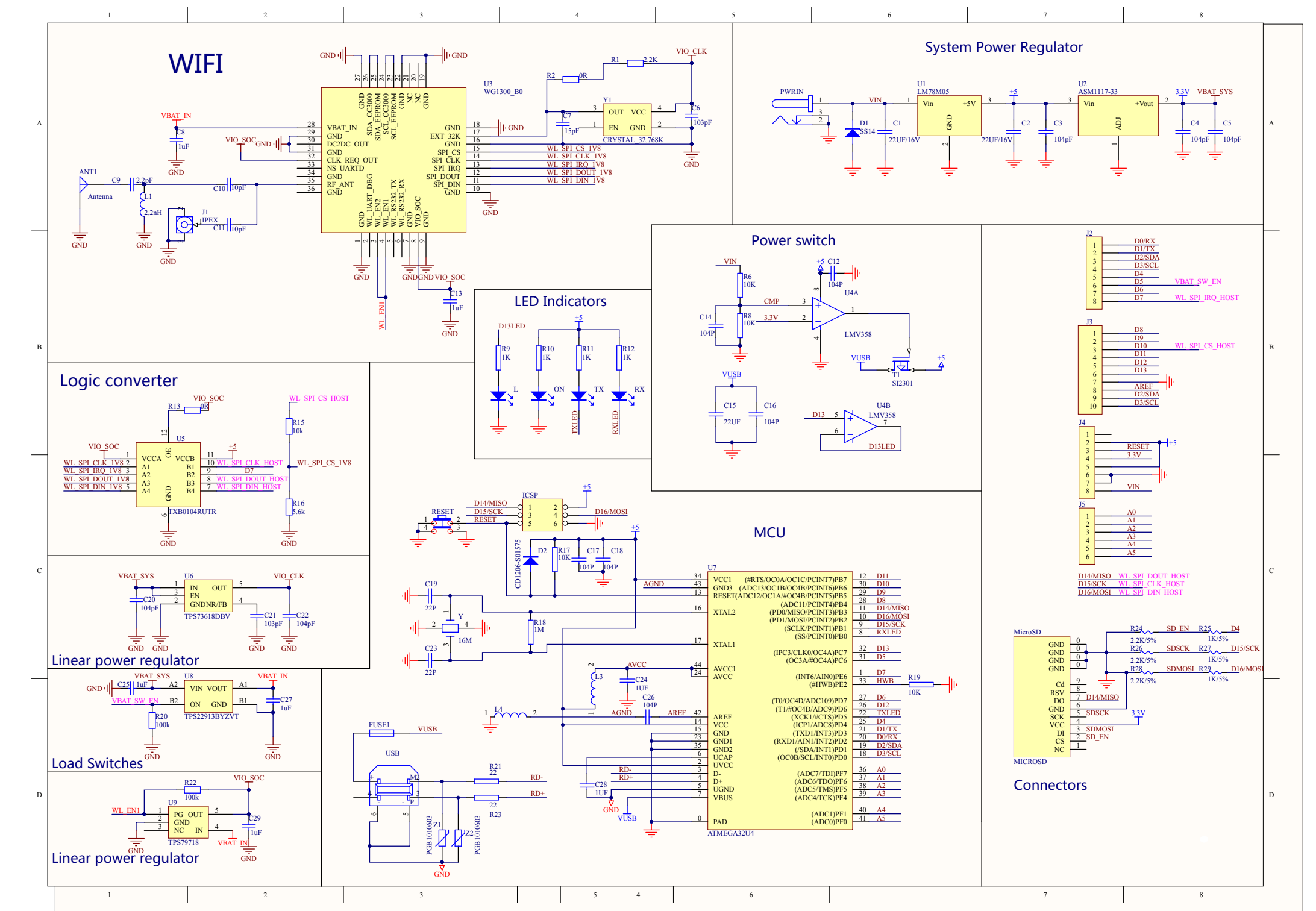
```
  if (!cc3000.begin())
  {
    Serial.println(F("Unable to initialise the CC3000! Check your wiring?"));
    while(1);
  }
```

3. Setup the router connection!

```
  if (!cc3000.connectToAP(WLAN_SSID, WLAN_PASS, WLAN_SECURITY)) {
    Serial.println(F("Failed!"));
    while(1);
  }
```

4. Finish and get the DHCP info from the router/AP

```
while (!cc3000.checkDHCP())
  {
    delay(100); // ToDo: Insert a DHCP timeout!
  }
```


WIFI
U3
WG1300_B0
ANT1
Antenna
J1
IPEX
Y1
CRYSTAL_32.768K
System Power Regulator
U1
LM78M05
U2
ASM1117-33
Power switch
LMV358
SI2301
LED Indicators
D13LED
L
ON
TX
RX
TXLED
RXLED
Logic converter
U5
TXB0104RUTR
Linear power regulator
U6
TPS73618DBV
Load Switches
U8
TPS22913BYZVT
U9
TPS79718
MCU
U7
ATMEGA32U4
ICSP
USB
FUSE1
Connectors
MicroSD
MICROSD