Twilio Narrowband Alfa Developer Kit MQTT Quickstart Guide

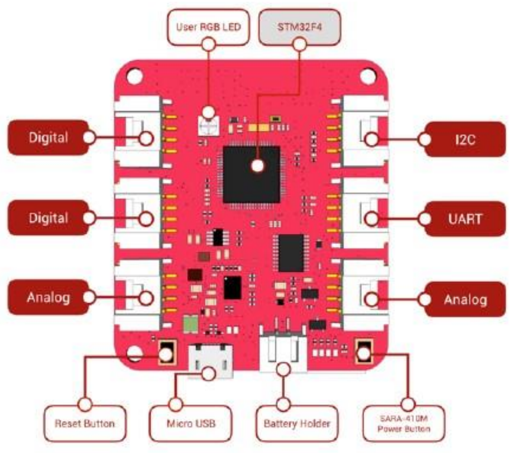
This guide will help you with registering your Narrowband SIM in the Programmable Wireless console, setting up the Alfa Developer Kit and then connecting to a public MQTT broker.

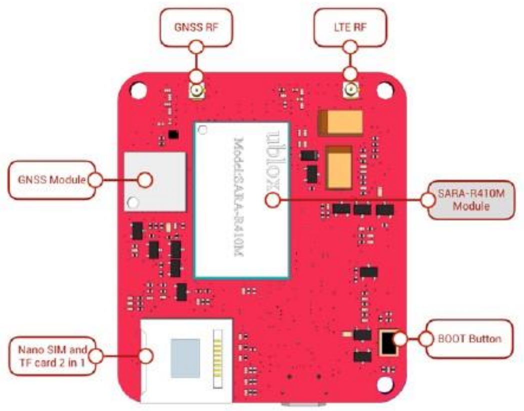
The following components are in your kit:

- Programmable Wireless Narrowband IoT SIM
- Alfa Development board
- LTE Antenna
- GPS Antenna
- Set of Sensors – Pushbutton, Ultrasonic, Temperature/Humidity
- Lithium battery
- Micro-USB cable

Registering the Narrowband SIM

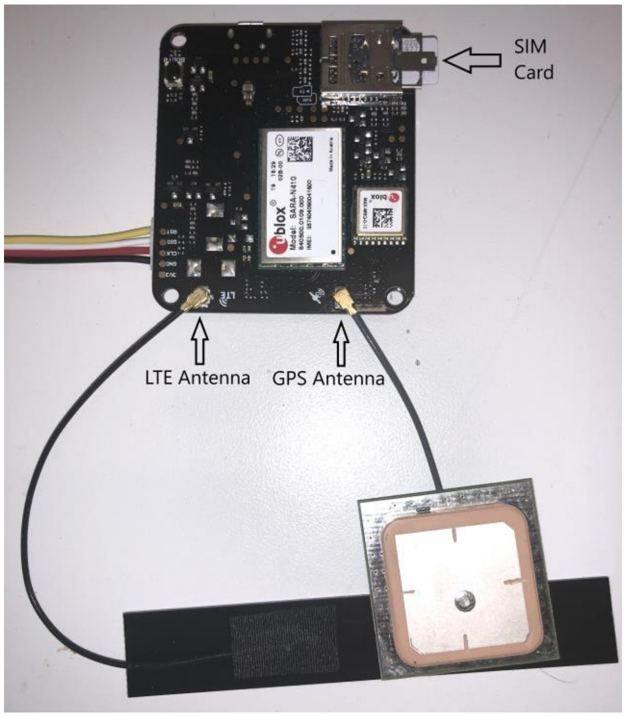
1. Open the Programmable Wireless Console (https://www.twilio.com/login?g=/console/wireless/sims/register)
2. Open the Register a Starter SIM page
3. Find the SIM registration number on the Narrowband IoT SIM
4. Enter the registration number in the **Registration Code** field and register the SIM card
5. Give your SIM a unique name and continue
6. Create a Narrowband rate plan by giving it a unique name and saving it
7. Click the **Activate Now** button to activate the SIM on the network

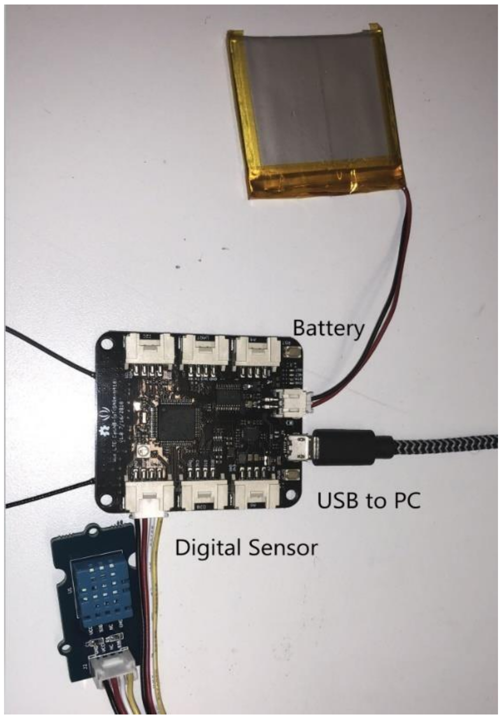




Hardware Connection

1. Insert the Narrowband SIM into the bottom of the two slots on the developer board
2. Connect the LTE Antenna to the pin labeled LTE on the developer board
3. Plug in the micro USB cable into the developer board and the other end into your computer to install the firmware
4. Insert the JST connector to the Lithium battery into the battery port to the right of the micro USB port





Connecting to the T-Mobile Network

The network LED to the left of the **Reset** button will initially turn orange. When it is eventually connected, it will turn blue. Connecting to the network may take a couple of minutes initially.

Connecting to the MQTT Broker

For the purpose of this Quickstart guide we will use the MQTT broker and MQTT web client provided by HiveMQ. However, you may use any broker and web client that you wish to.

1. Visit HiveMQ's MQTT web client page (http://www.hivemq.com/demos/websocket-client/)
2. Add the required details and click **Connect**
3. Create a random string for your topic and subscribe to it. For example – if your topic is named **randomstring**, you will need to subscribe to **randomstring/#**
4. To test it – under Publish, use the topic **randomstring/data** and send a test message

Getting Development Environment Setup

Installing USB programming support

Windows Users:

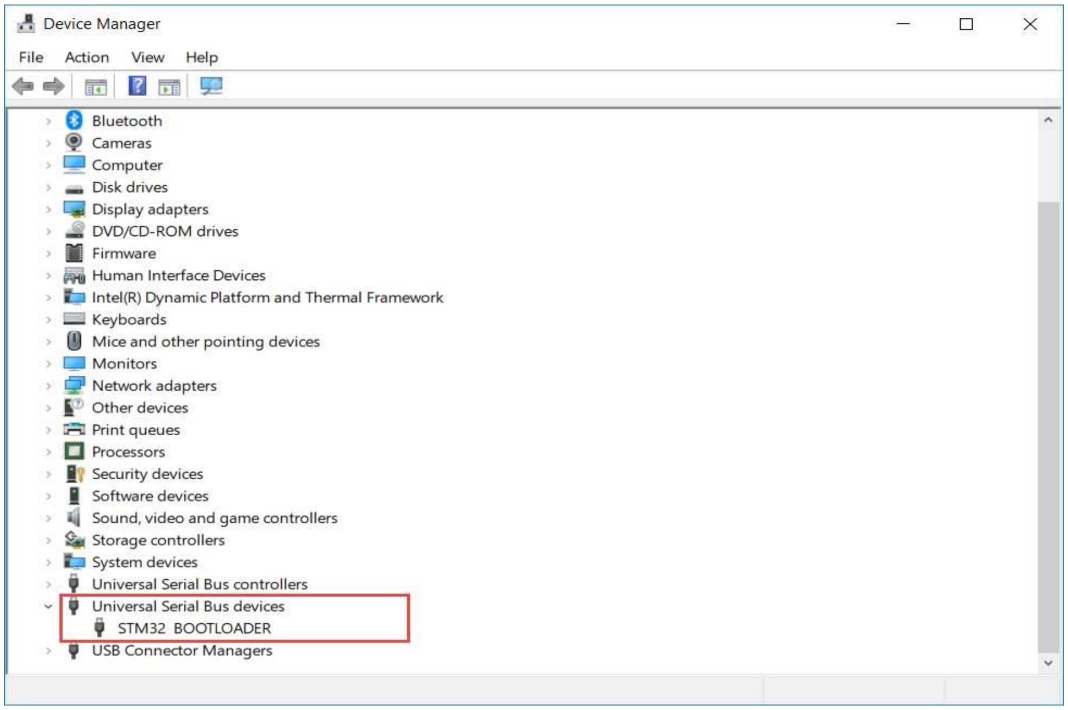
Most versions of Windows won't automatically load the built-in driver for USB com ports. You'll have to download ST's USB driver:

- Non-Windows XP Users download version 1.4.0 drivers. Unzip the file, run the executable, and then go to C:\Program Files (x86)\STMicroelectronics\Software\Virtual comport driver in Windows Explorer and double-click either dpinst_amd64.exe for 64bit systems, or dpinst_x86.exe for 32 bit. (For Windows 10, run as administrator)
- Windows XP Users download version 1.3.1 drivers. Unzip the file, run VCP_V1.3.1_Setup.exe, and then go to C:\ProgramFiles\STMicroelectronics\Software\Virtual comport driver in Windows Explorer and double-click the executable.

Windows 10:

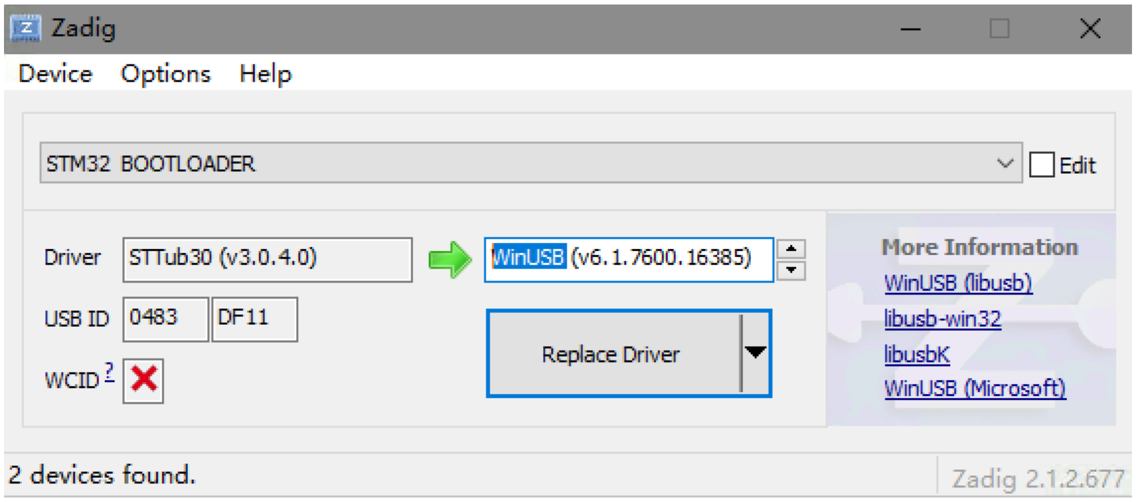
Windows 10 users will have the change the DFU driver

Connect the dev kit to computer, press and hold both BOOT0 and RST buttons, release the RST button than the BOOT0 button, you will see the device in **DFU Mode** in the device manager



Use Zadig xx.exe (run as administrator) to change DFU driver from **STTub30** to **WinUSB** as below.

- Select Options>List All Devices
- Choose STM32 BOOTLOADER from the list of devices and Replace Driver



STM32 BOOTLOADER

Driver STTub30 (v3.0.4.0) → WinUSB (v6.1.7600.16385)

USB ID 0483 DF11

WCID ? ✗

2 devices found. Zadig 2.1.2.677

MacOS

The quickest way to install the necessary driver on OSX is by using Homebrew.

1. Install Homebrew by running this in a Terminal: in a terminal: /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
2. Install `dfu-util` using Homebrew by typing the following in a Terminal: brew install dfu-util libusb

Note: Use dfu-util 0.9 or greater if available. If you already have it installed, check the dfu-util version using brew info dfu-util

Ubuntu

1. Install an updated dfu-util: sudo apt-get install dfu-util
2. Configure the permissions for the new USB serial port

```
sudo tee -a /etc/udev/rules.d/50-twilio-local.rules << _DONE_
# Twilio Alfa kit
# Bus 001 Device 035: ID 0483:5740 STMicroelectronics STM32F407
SUBSYSTEMS=="usb", ATTRS{idVendor}=="0483", ATTRS{idProduct}=="5740",
GROUP="dialout", MODE="0666"
_DONE_
```

3. Plug in the developer kit
4. Verify the permissions: ls -l /dev/ttyACM* the device should be readable and writable by all users.

Setup Arduino IDE

1. Download [Arduino IDE 1.8.9+](https://www.arduino.cc/en/Main/Software)

Install Board Support for the Alfa Development Board

1. Open the Arduino IDE
2. Open the menu Arduino > preferences
3. Copy this URL into the "Additional Boards Manager URLs" field: https://raw.githubusercontent.com/Seeed-Studio/Seeed_Platform/master/package_seeeduino_boards_index.json
4. Click "OK"
5. Click back into the menu *Tools > Boards > Boards Manager*
6. Search for "Seeed"
7. Select "Seeed STM32F4 Boards"
8. Click install
9. Navigate to the menu *Tools > Boards > Wio Tracker LTE*

Install the MQTT Client

1. Open the menu: *Tools > Library Manager*
2. Search for "Paho"
3. Select the library ArduinoMQTT by Oleg Kovalenko and install it
4. Close the Arduino IDE

Install Breakout SDK

1. Navigate to https://github.com/twilio/Breakout_Massive_SDK_Arduino/releases and download the latest release
2. Open the Arduino IDE and navigate to *Sketch > Include Library > Add .zip library* and select the zip folder you downloaded
3. Restart the Arduino IDE

Connect to the MQTT Broker and Publish Messages

1. Create a new example sketch with the sample code. In the 'File' menu: Examples > Breakout Arduino Library > AlfaKit > Sample
2. Navigate to the first tab of the resulting IDE window. Uncomment the code which will turn on SampleButton.h
3. Navigate to the config.h tab of the IDE window.
4. Change the MQTT_BROKER_HOST, MQTT_PUBLISH_TOPIC, MQTT_STATE_TOPIC, MQTT_LOGIN, and MQTT_PASSWORD variables to:
   MQTT_BROKER_HOST: Your broker, or broker.hivemq.com if you are using HiveMQ.
   MQTT_BROKER_PORT: Leave it as 1883 if you want it unencrypted

   MQTT_CLIENT_ID: Change the USERNAME part of the string to your topic, **randomstring**
   MQTT_PUBLISH_TOPIC: Use your topic string followed by /data, **randomstring/data**
   MQTT_STATE_TOPIC: Use your random string followed by /state, **randomstring/state**
   MQTT_LOGIN: you can use anything, it is ignored on a public broker
   MQTT_PASSWORD: you can use anything, it is ignored on a public broker

5. Uncomment the line #define USE_USERNAME_PASSWORD
6. Put your board in bootloader mode. (Hold BOOT0, hit RST, release RST, release BOOT0)
7. Click *Sketch* > *Upload* to upload the example to the Developer Board
8. Tap the RST button to reboot the board

Verify that you can send messages to the cloud:

1. Open the serial monitor again (Tools > Serial Monitor)
2. Watch for connection on the board. The STA status light on board will be yellow when seeking connection, and the serial monitor will show:

   ```
   waitForNetworkRegistration():259 .. waiting for network registration
   processURCEPSRegistration():260 Received URC for CEREG [2,2]. Set a
   handler with setHandlerEPSRegistrationURC() if you wish to receive this
   event in your application
   ```

   In a loop. - When connected, the light will turn blue and you will see this message:

3. Surf back to the browser tab where you have the MQTT client. **Hit the button** and watch for the message in the browser client. If you see a message congratulating you on the button, you did it! You now have one-way communication with the cloud. Your messages are traveling over the air and making it to the MQTT broker:

Send a message from the cloud client to the broker

1. Switch back over to the cloud MQTT client
2. Send a message to the **randomstring/state** topic

Programming the LED to change color in reaction to messages

1. Add code to import the LED library and set a constant pin (at the top of your sketch/main.ino file (the **FIRST** tab in the IDE), **ABOVE** #include "mqtt.h"):

```
#include <Seeed_ws2812.h>
WS2812 strip = WS2812(1, 17);
```

2. In the setup() function, add code to initialize the LED to white:

```
void setup() {
pinMode(RGB_LED_PWR_PIN, OUTPUT);
digitalWrite(RGB_LED_PWR_PIN, HIGH);
strip.begin();
strip.brightness = 5;
strip.WS2812SetRGB(0, 0x20, 0x20, 0x20);
strip.WS2812Send();

owl_log_set_level(L_INFO);
...
```

3. Edit the 'state' callback in the mqtt.h tab of the IDE

   Find the static void device_state_callback(MQTT::MessageData &message) {function - Find the LOG(L_WARN, "Unknown state: %.*s\r\n", message_str.len, message_str.s); line

   DELETE that line and in its place paste this handler code:

```
if (str_equal_char(message_str, "red")) {
strip.WS2812SetRGB(0, 0x30, 0x00, 0x00);
} else if (str_equal_char(message_str, "green")) {
strip.WS2812SetRGB(0, 0x00, 0x30, 0x00);
} else if (str_equal_char(message_str, "blue")) {
strip.WS2812SetRGB(0, 0x00, 0x00, 0x30);
} else if (str_equal_char(message_str, "yellow")) {
strip.WS2812SetRGB(0, 0x30, 0x30, 0x00);
} else if (str_equal_char(message_str, "teal")) {
strip.WS2812SetRGB(0, 0x00, 0x30, 0x30);
} else if (str_equal_char(message_str, "purple")) {
strip.WS2812SetRGB(0, 0x30, 0x00, 0x30);
} else if (str_equal_char(message_str, "white")) {
strip.WS2812SetRGB(0, 0x30, 0x30, 0x30);
} else if (str_equal_char(message_str, "black")) {
strip.WS2812SetRGB(0, 0x00, 0x00, 0x00);
} else {
LOG(L_WARN, "Unknown state: %.*s\r\n", message_str.len, message_str.s);
return;
}
strip.WS2812Send();
LOG(L_WARN, "Set color to: %.*s\r\n", message_str.len, message_str.s);
```

4. Reset the board and cycle the serial monitor
5. Switch to the HiveMQ client tab in your browser. - Send any of the following commands (case sensitive, no whitespace) - `red`, `green`, `blue`, `yellow`, `teal`, `purple`, `white`, `black`