What we will be building

We will build a very simple IoT device with a button, a LED and a potentiometer. When we click the button the value of the potentiometer is sent via the NB-IoT network to the backend server.

This is more or less the essence of an IoT device but rather than fancy air quality sensors we just read the value of the potentiometer and the button works as our firmware logic.

Hardware

You'll need the following:

- Arduino UNO
- EE-NBIOT-01 module
- LTE antenna
- Potentiometer
- A small LE
- 220 Ohm resistor
- Microswitch
- Jumper wires
- Breadboard

Set up software

If you haven't installed the Arduino IDE please do so now. You can either download the IDE as a binary at https://www.arduino.cc/en/Main/Software or use the online editor. The online editor requires registration. Once it is installed you are ready to program the Arduino.

You'll also need the Telenor NB-IoT library from https://github.com/ExploratoryEngineering/ArduinoNBIoT. Download a Zip file and use Library Manager in the Arduino IDE to install the library.

Verifying connectivity

Let's start with verifying connectivity and that we are able to communicate with the EE-NBIOT-01 module.

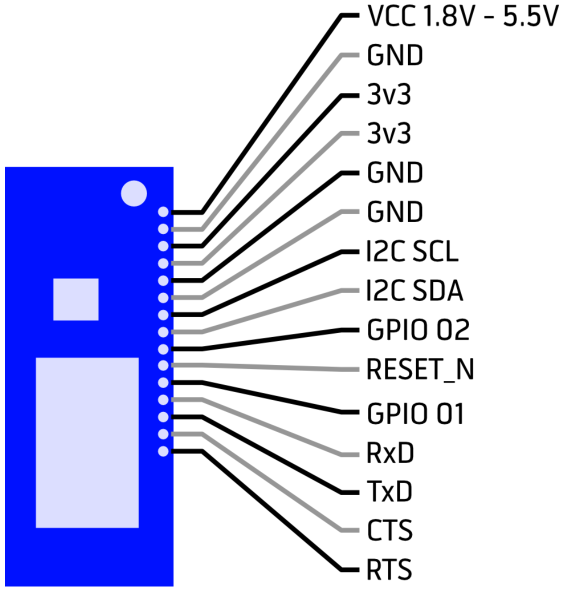
Connect the antenna

Start by connecting the antenna to the uFL connector. You'll feel a small click when it slots into place.

Place the module on the breadboard

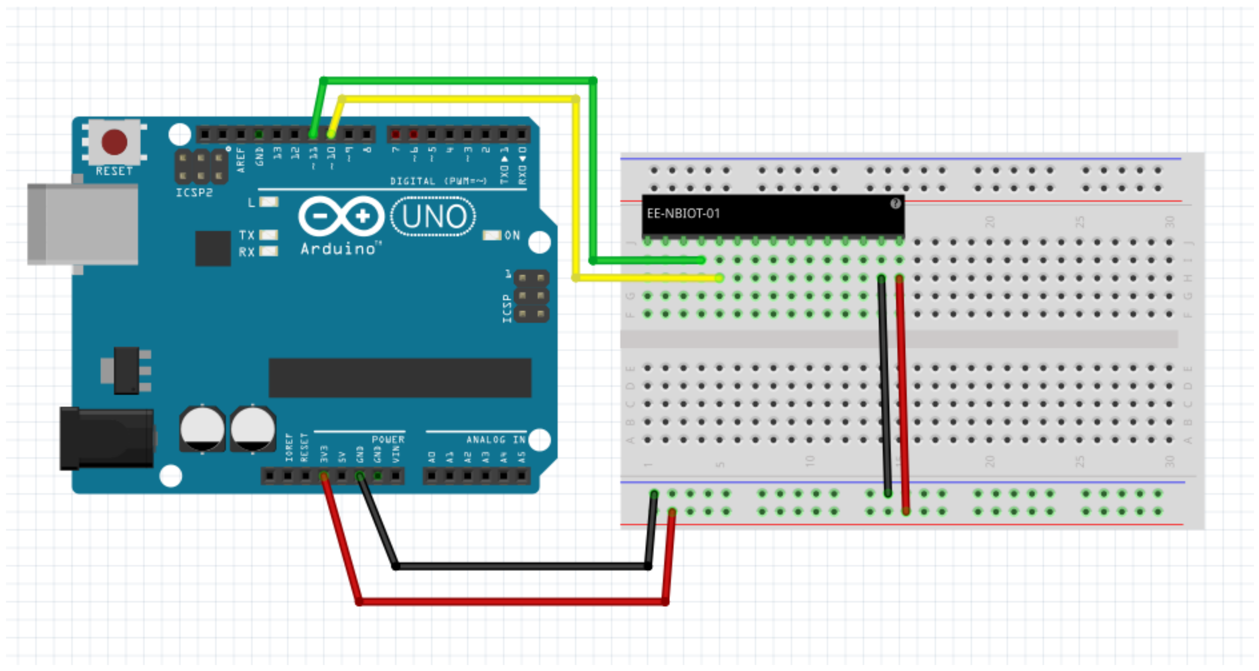
Place the module on the breadboard. It might be a tight fit. Make sure you don't bend the legs of the module when you insert it. Use the middle part of the breadboard and place it on the upper left hand corner so that the first pin on the left hand side sits in the row marked with a 1.

Wire up the module



The module have several pins but we're only interested in four of those -- the VCC and GND pins to power it and the RX and TX pins to talk to it. Connect a black wire from the 2nd pin (row 14 on the breadboard) on the right to the blue strip on the side of the breadboard and a red wire from the first pin (row 15 on the breadboard) to the red strip on the side of the breadboard.

Connect a yellow wire from the RxD pin on the module (4th from left, row 4 on the breadboard) and a green wire from the TxD pin on the module (3rd from the left, row 3 on the breadboard). Connect the yellow wire to pin 10 on the Arduino and the green wire to pin 11 on the Arduino. Find the 3v3 pin on the Arduino and connect that pin to the red strip on the breadboard. Use a red wire for this. Connect the GND pin on the arduino with the blue strip on the breadboard. Use a black wire for this. The black wires will now be the ground and the red wires will be the power. It's not *really neccessary to use the different colours but it makes it a lot easier later on.



Technically we should have added a few components to ensure we don't break the module (Arduino talks 5V while the EE-NBIOT-01 module prefers 3.3V but this works fine. We might shave a few months or years off the module's lifetime but this is for demonstration purposes so we'll use the Arduino directly.

We're now ready to test the module via the Arduino IDE.

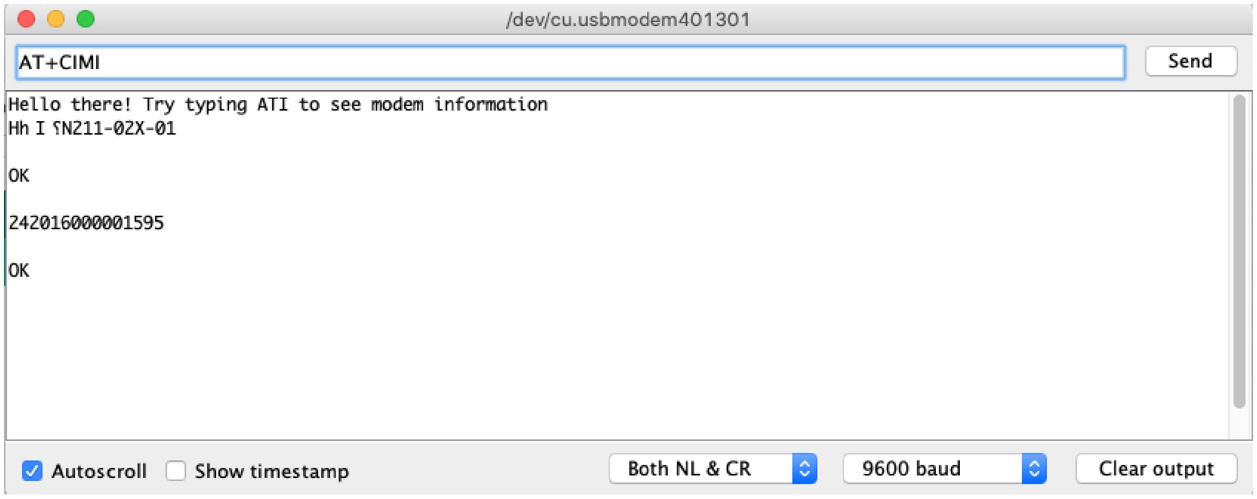
Open the sketch "SerialCheck" in the Arduino IDE -- go to it should look like the following:

```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(10, 11); // RX, TX

void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }



  // set the data rate for the SoftwareSerial port
  mySerial.begin(9600);
  while (!mySerial) {
    ;
  }
  Serial.println("Hello there! Try typing ATI to see modem information");
  mySerial.print("ATI\r\n");
}

void loop() { // run over and over
  if (mySerial.available()) {
    Serial.write(mySerial.read());
  }
  if (Serial.available()) {
    mySerial.write(Serial.read());
  }
}
```

It can be downloaded from GitHub (if for obvious reasons) you don't want to type it in): https://github.com/ExploratoryEngineering/arduino-demo

Upload the sketch to the Arduino through the IDE (Sketch | Upload in the menu) and open the Serial Monitor. (Tools | Serial Monitor in the menu)

Type AT+CIMI in the command window and press enter. Make sure that "Both NL & CR is selected and that the speed is set to 9600 baud):

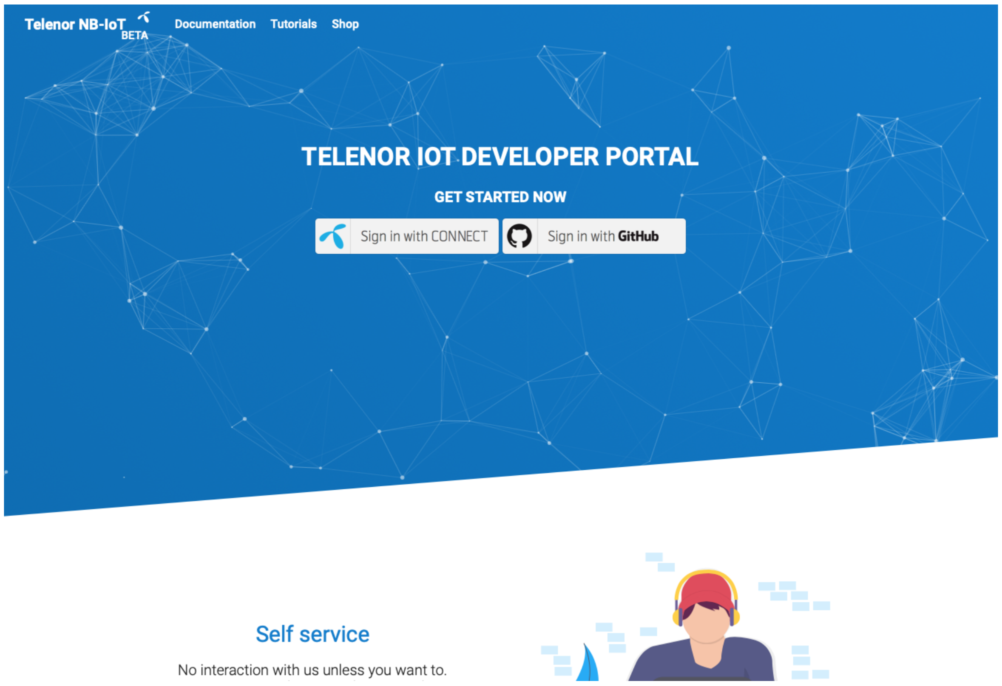


```
Hello there! Try typing ATI to see modem information
Hh I ʕN211-02X-01

OK

242016000001595

OK
```

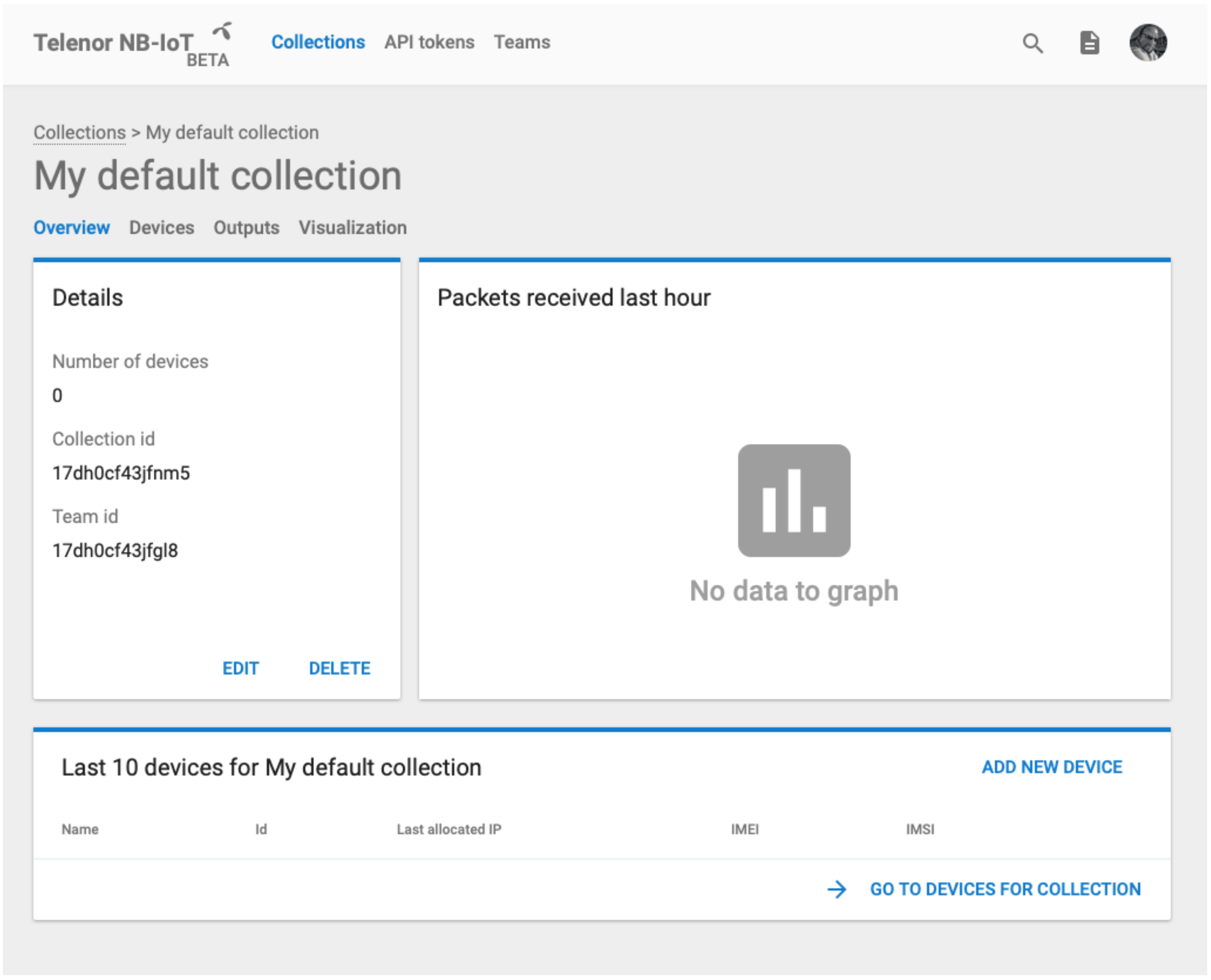
Type AT+CIMI in the command box and press enter. The module should respond with it's IMSI number. Write down this number -- you'll need it later!

Sending a message to the backend

Let's try to send a message to the backend manually. Go to https://nbiot.engineering/ and log in via CONNECT ID or GitHub.



Once you have logged in you'll see only the default collection. Click on the default collection, then the "Add device" button on the right hand side.



Collections > My default collection

My default collection

Details

Number of devices
0

Collection id
17dh0cf43jfnm5

Team id
17dh0cf43jfgl8

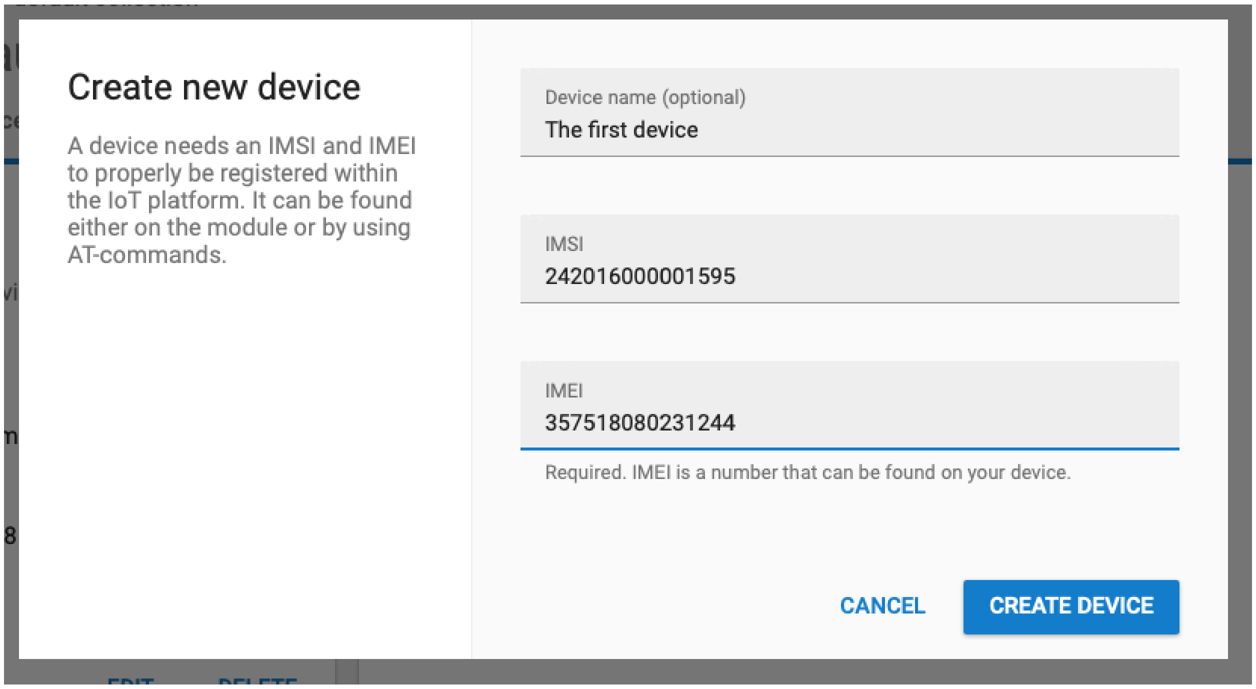
Packets received last hour

No data to graph

Last 10 devices for My default collection

Name	Id	Last allocated IP	IMEI	IMSI

Give the device a name and use the IMSI and IMEI from the leaflet that came with the module. In addition the IMEI is printed on top of the module and you can query the module via the Serial Monitor with AT+CIMI (for IMSI) and AT+CGSN=1 (for IMEI).



Create new device

A device needs an IMSI and IMEI to properly be registered within the IoT platform. It can be found either on the module or by using AT-commands.

Device name (optional): The first device

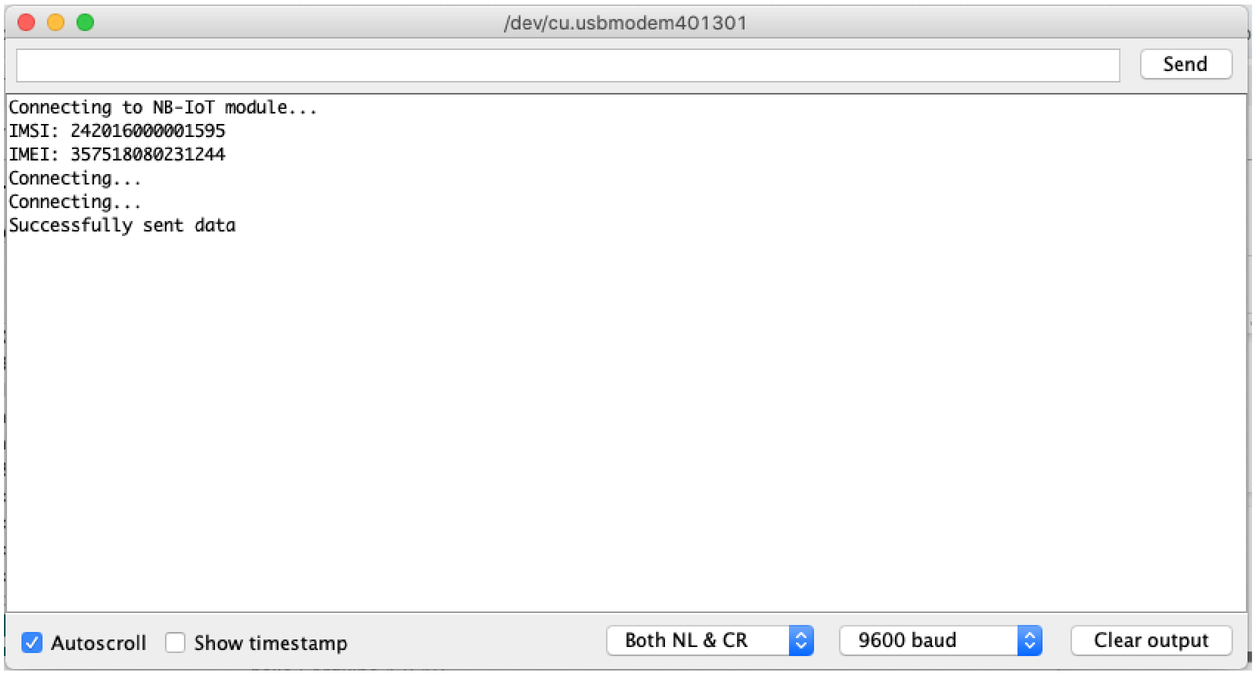
IMSI: 242016000001595

IMEI: 357518080231244

Required. IMEI is a number that can be found on your device.

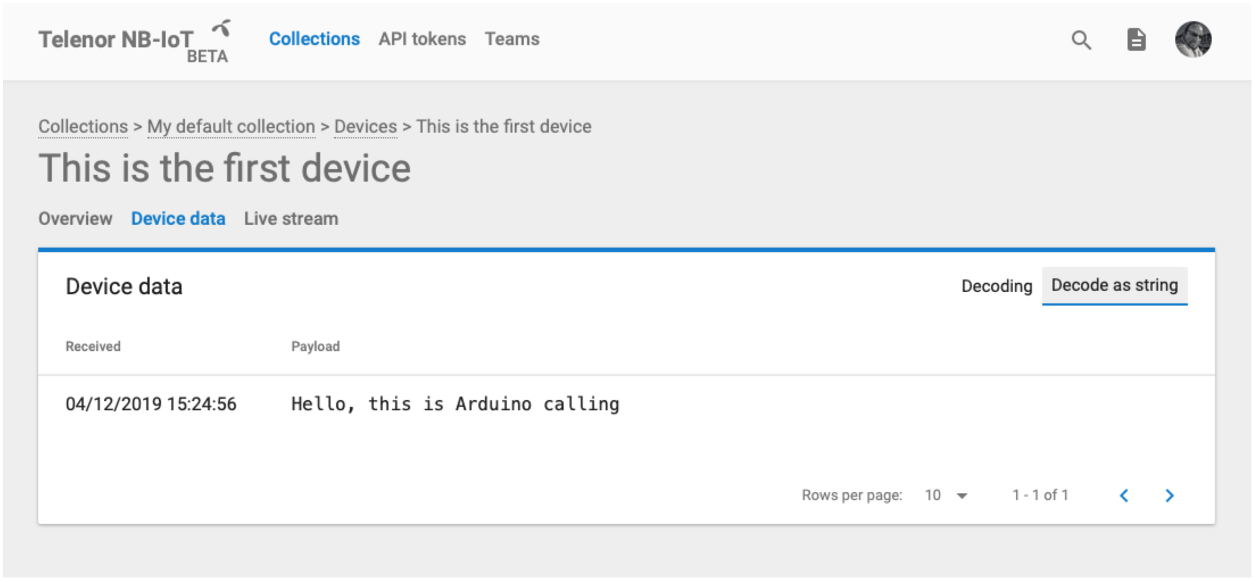
When the device is created it will say “Never seen” in the list. Let’s fix that by connecting to the backend and then send a test message. Setting up the device manually via AT commands can be a bit cumbersome so let’s use the NB-IoT library instead.

Go to “File | Examples | Telenor NB-IoT | hello” in the menu to open the default example. Everything is already wired up correctly so let’s start it. Click on the upload button in the toolbar, then open the Serial Monitor and check the output. It should say something similar to this:



```
Connecting to NB-IoT module...
IMSI: 242016000001595
IMEI: 357518080231244
Connecting...
Connecting...
Successfully sent data
```

If we click on the “Device Data” tab in the Horde console we’ll see the data. Select “Decode as string” from the upper right corner to see the string that was sent.



Received	Payload
04/12/2019 15:24:56	Hello, this is Arduino calling

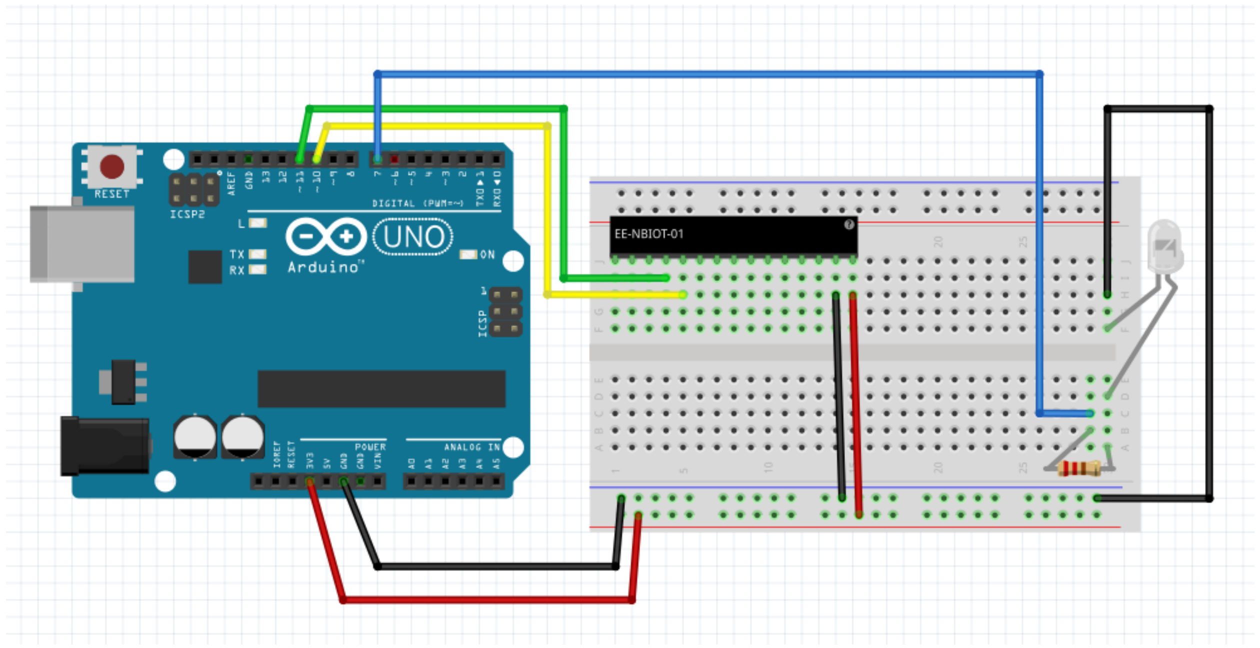
If you leave the Arduino running it will send a new message after roughly 15 minutes.

Now that we’ve got a working mode let’s try something a bit more involved -- lets’s make a sensor.

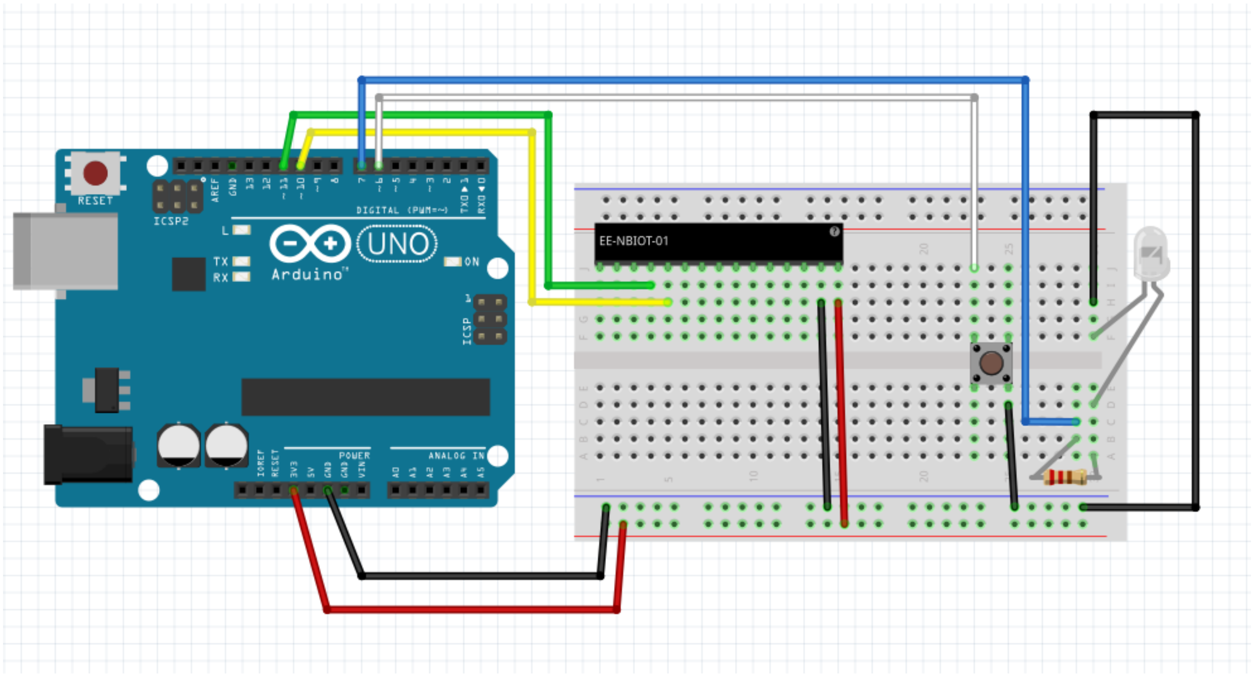
Making a device with a sensor

We'll use a very simple sensor -- the potentiometer. When the button is clicked we'll send the reading from the potentiometer. The Arduino board will read the value from the potentiometer and send a message to the Horde backend with the reading.

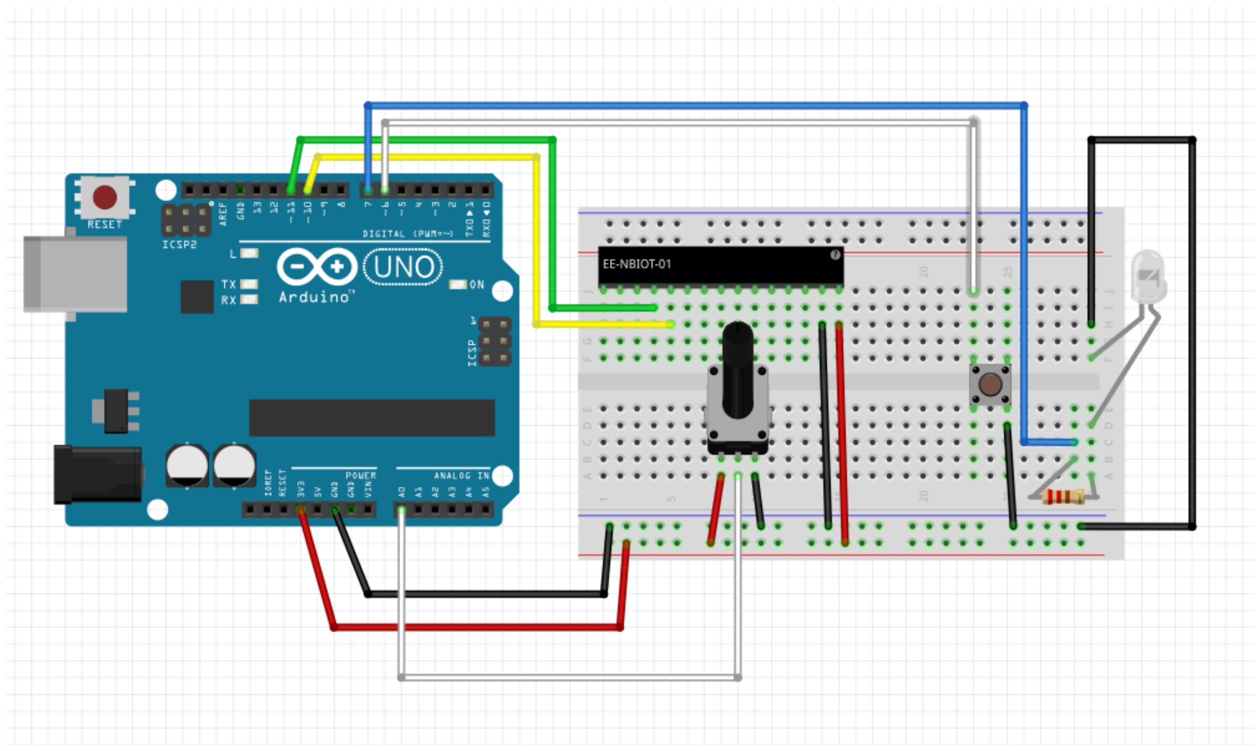
Start by connecting a LED to the breadboard. Place the long wire in row 30 on the bottom and the short wire in row 30 on the top. Connect the top (with the short wire) to ground (the blue stripe at the bottom of the breadboard). Use a black wire for this since this is connected to ground. Add a resistor from row 30 and connect it to row 29. Connect a wire from row 30 to pin 7 on the Arduino. I'm using a blue wire for the LED.



Insert the microswitch into the breadboard with the pins facing the ground and power strips. It can be a bit fiddly but it is easier if you connect it across the two sides of the breadboard.. Connect a wire from one of the sides to ground (a black wire) and the other (the white wire) to pin 6 on the Arduino:



Next we'll connect the potentiometer to the breadboard. Connect one of the sides (left or right does not matter) to ground (the blue stripe at the bottom of the breadboard, black wire) and the other side to the power (the red stripe at the bottom of the breadboard, red wire). Connect the middle pin with the A0 pin on the Arduino (a white wire in the diagram).



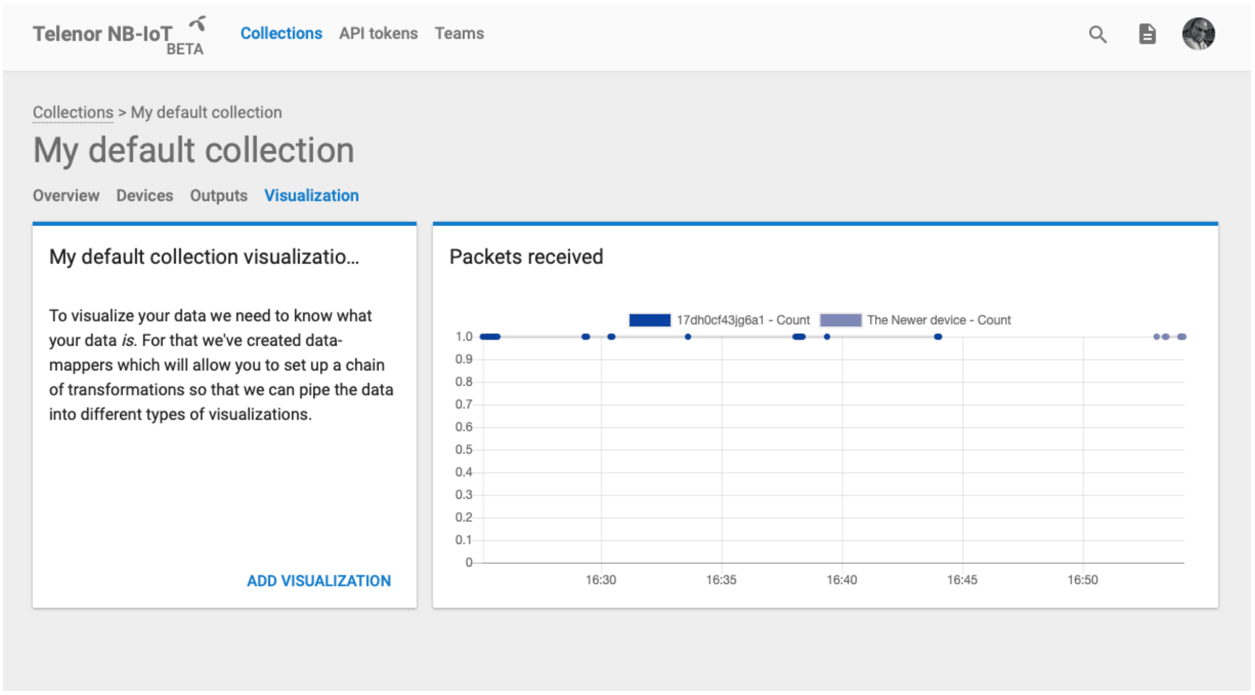
That’s the hardware. Your breadboard should look like a rat’s nest by now.

Copy and paste the sketch from https://github.com/ExploratoryEngineering/arduino-demo named PotmeterButton.ino, build it and upload it to the Arduino. When you run it you should see the LED flash twice when it is connected and if you press the switch afterwards it will send a message with the potmeter reading to the Horde backend. If you open the Serial Monitor you’ll see the value that is sent. Open the device’s data page at https://nbiot.engineering/ and you’ll see the messages as they come in.

Twist the potmeter then press the button again to send a new message. Make sure that the payload changes between each button press.

Visualizing the result

Let’s make a chart of the values. The charting functions in Horde is mostly for development work so it’s not a full featured charting (or dashboard) solution. Click on the “Overview” tab then on the “Visualization” tab.

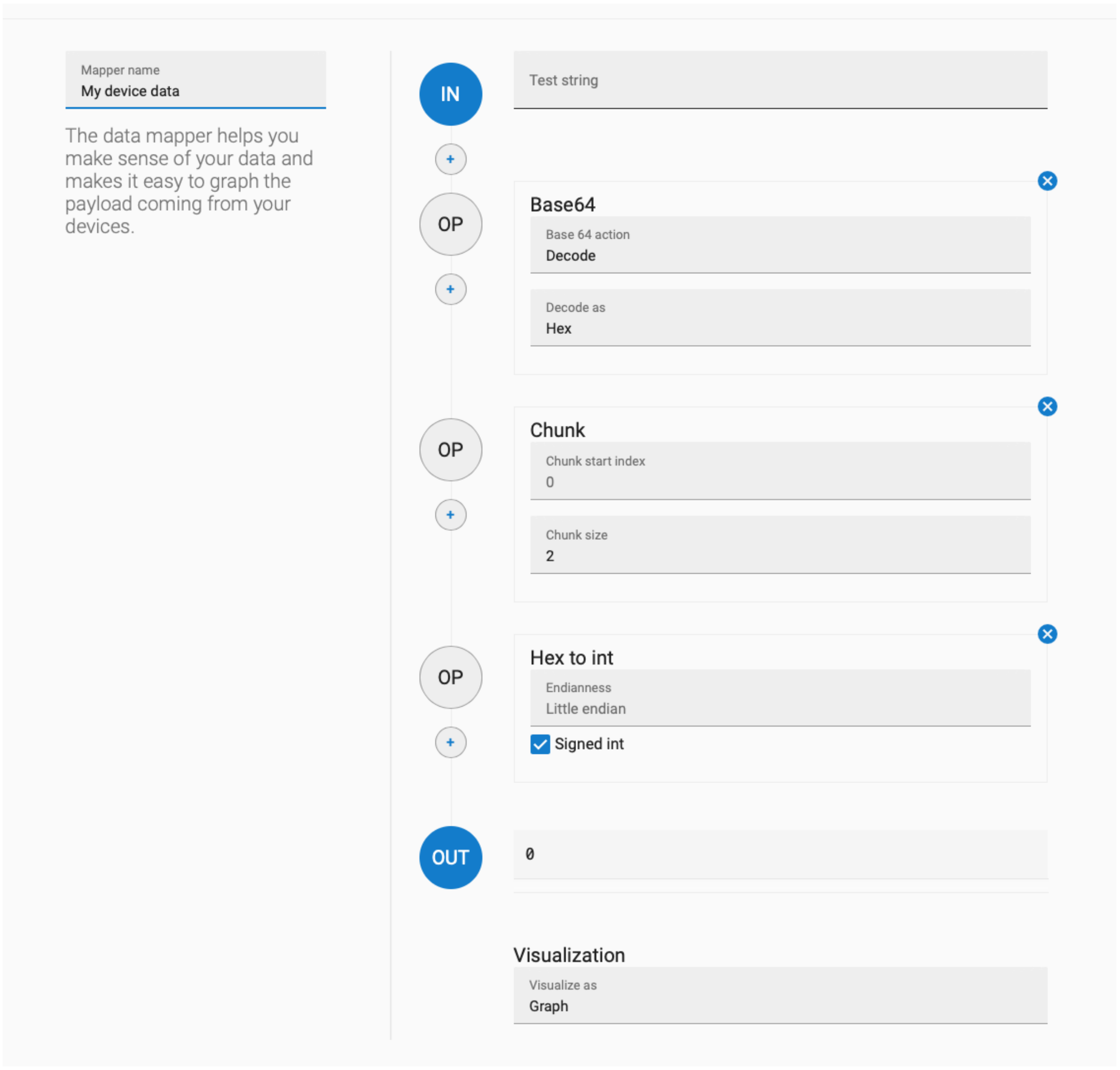


My default collection

My default collection visualizatio…

To visualize your data we need to know what your data *is*. For that we've created data-mappers which will allow you to set up a chain of transformations so that we can pipe the data into different types of visualizations.

Click on “add visualization” to add a new conversion function ,then set the decoding to “Hex”, add a “Chunk” with length 4 (this will grab the first four characters of the data, ie the first two bytes) and finally add the “Hex to int” conversion. Make sure it is set to “Little endian”.


Mapper name
My device data

The data mapper helps you make sense of your data and makes it easy to graph the payload coming from your devices.

Test string

Base64

Base 64 action
Decode

Decode as
Hex

Chunk

Chunk start index
0

Chunk size
2

Hex to int

Endianness
Little endian

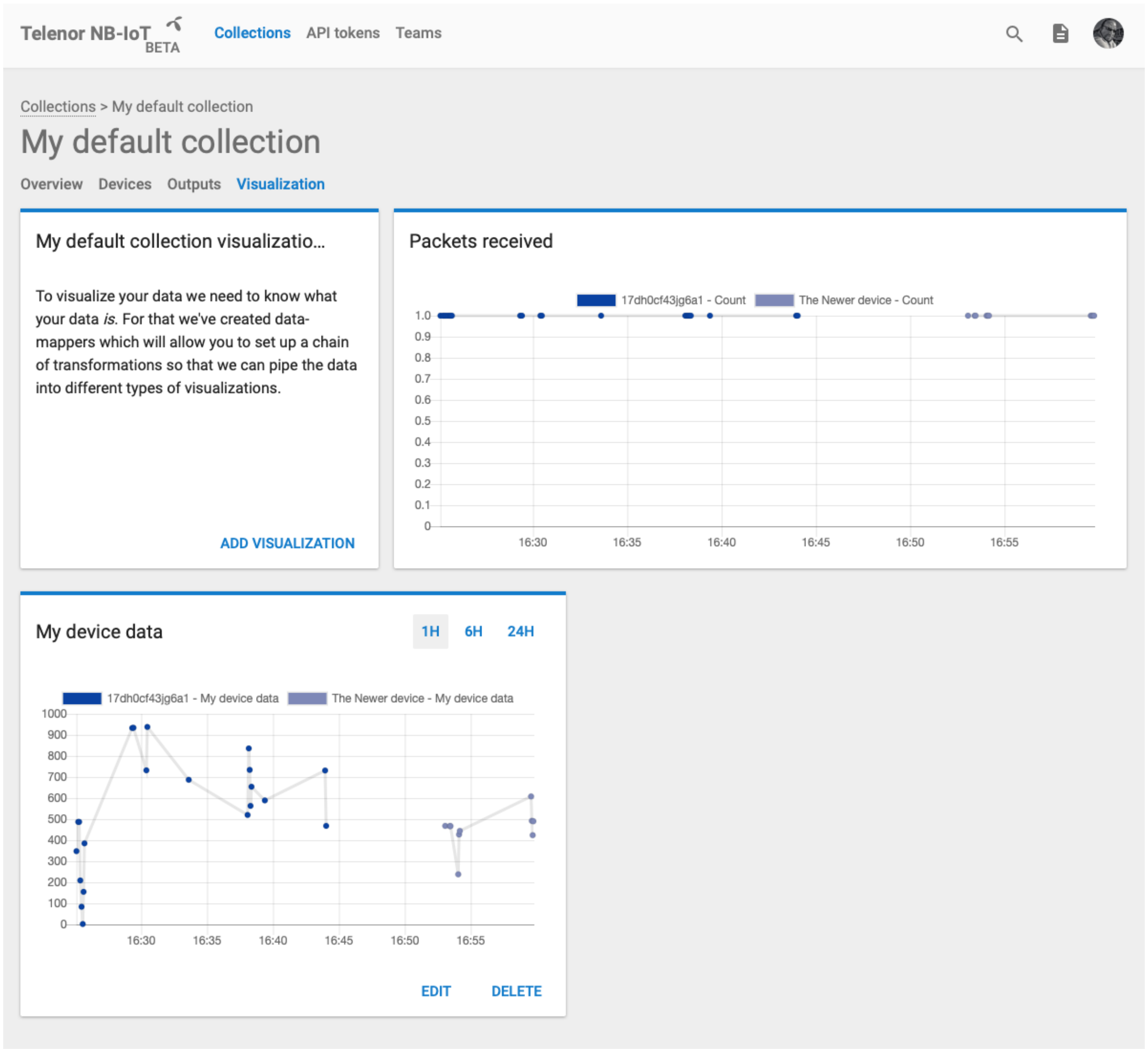
[✓ Signed int]

```
0
```

Visualization

Visualize as
Graph

The visualization page should now look more or less like this:



My default collection

My default collection visualizatio...

To visualize your data we need to know what your data *is*. For that we've created data-mappers which will allow you to set up a chain of transformations so that we can pipe the data into different types of visualizations.

If you see a lot of interference from the test data sent in the first steps you can delete the existing device and re-add it. You'll see two devices in the chart and you can filter out the data from the old device.