Connecting the ESP32 to the Internet with Arduino and Losant

Le document original est en anglais. Il est disponible sur internet en tapant dans Google : Connecting the ESP32 to the Internet with Arduino and Losant

Le plus intéressant dans ce tutoriel est la création du compte Losant puis l'étude du programme donné en exemple.

Je vous conseille d'étudier attentivement les chapitres suivants :

Setting up Losant

Flashing ESP32 Firmware

Now that it's been some time since the ESP32 has been released and the software stack has matured, it's safe to start building projects using this connected-ready hardware.

I'm a huge fan of Adafruit's Feather boards; they are very reliable with an awesome community behind them. The Adafruit HUZZAH32 is their ESP32-based board.

Because of the cost—it's just $19.95—and its support of Wi-Fi and Bluetooth, it's my recommended connected microcontroller. In this tutorial, I am going to walk you through setting up the device and connecting it to the Internet.

To illustrate this, I've gathered the following components:

Adafruit Feather HUZZAH32

1 Solderless Breadboard

8 Jump Wires

2 Resistors

1 LED

1 TMP36 temperature sensor

1 Button

1 Micro-USB cable

Let's get started.

Download USB Drivers

If this is your first time using the ESP32, you will have to install the USB drivers. If not, you won’t be able to flash your device. Here is where you can find the drivers:

https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers

Download and Install Arduino IDE

Download and install the latest version of the Arduino IDE by following the instructions at:

https://www.arduino.cc/en/Main/Software

We will be using the Arduino IDE to flash the ESP32. Before we can begin flashing, we need to configure the IDE to support an ESP32-based board.

Install Arduino Core for ESP32

Installing the ESP32 Arduino core is a little daunting, but it should get better over time as the APIs mature. For now, you’ll have to install the board manually. There are instructions for every operating system here:

https://github.com/espressif/arduino-esp32#installation-instructions

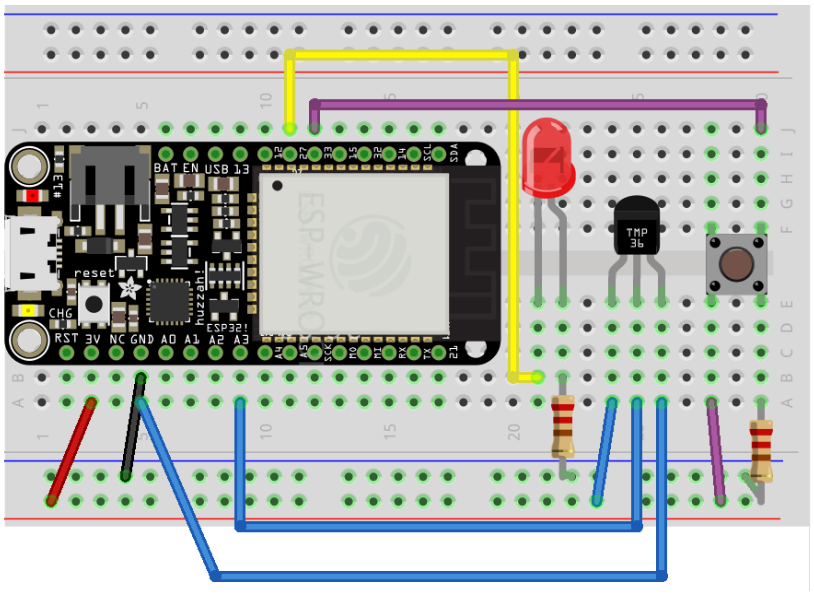
Once you have the core installed, you should be able to go to Tools > Board and select Adafruit ESP32 Feather.

Use the Tools -> Port menu to select the port of your ESP32. This will change depending on your operating system. On a Mac, it’s typically SLAB_USBtoUART. On Windows, it will be named COM and then a number, for example COM3.

Now, our environment is ready to flash the ESP32. But first, let’s get the components wired up.

How to wire the ESP32

Here is the wiring diagram with all of the materials:



This wiring will appropriately set up the LED, the TMP36 (temperature sensor), and the switch (button).

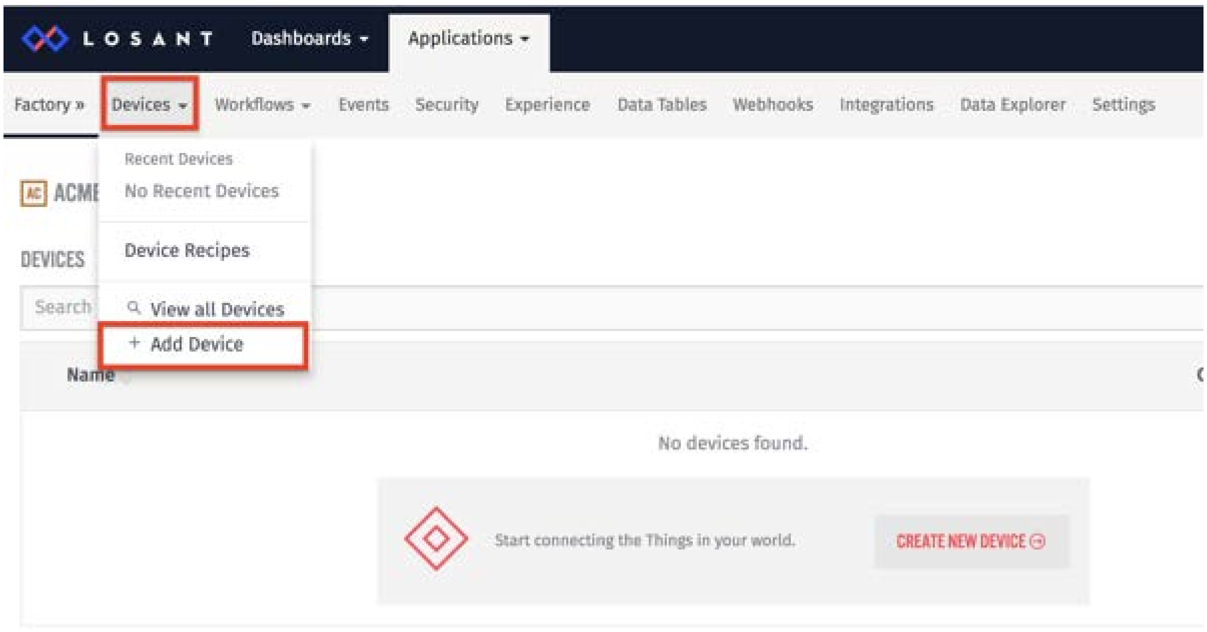
We are so close to flashing! We need to gather some credentials from the cloud.

Setting up Losant

To begin, we need to create a device within Losant. Before we can flash the firmware, we need to input credentials from Losant. If you don't already have a Losant account, sign up for a free developer sandbox.

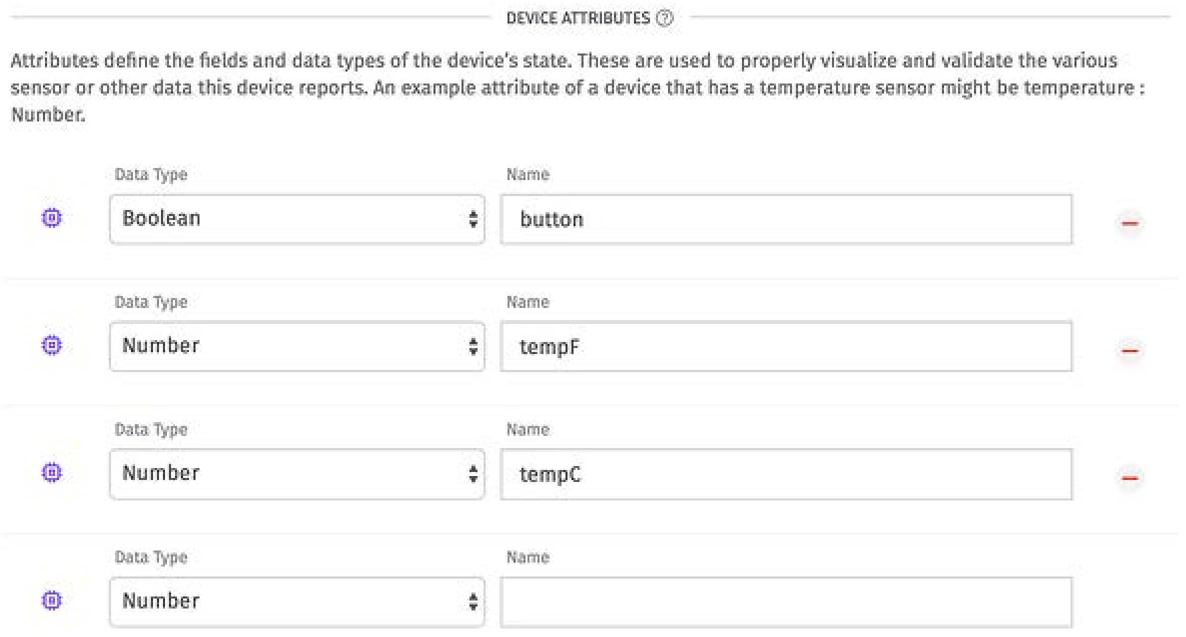
Create an Application & Device

Once you're in Losant, create an application, then create a device.



Device configuration

Our device will have the following three attributes:



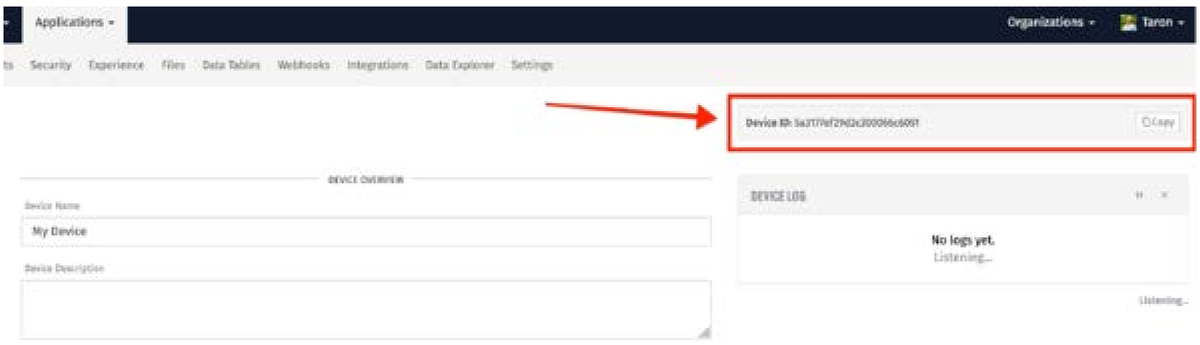
DEVICE ATTRIBUTES

Attributes define the fields and data types of the device's state. These are used to properly visualize and validate the various sensor or other data this device reports. An example attribute of a device that has a temperature sensor might be temperature : Number.

Data Type	Name
Boolean	button
Number	tempF
Number	tempC
Number	

The device attributes specify what state information the device reports. The firmware that you'll flash in the following sections will report whenever a button is pressed, the temperature in Celsius (tempC), and the temperature in Fahrenheit (tempF).

Once you create your device, you will be presented with a Device ID. Be sure to copy this ID, you will need it later.



Device ID: 5a2177ef29d2c200066c6091

DEVICE OVERVIEW

Device Name: My Device

Device Description

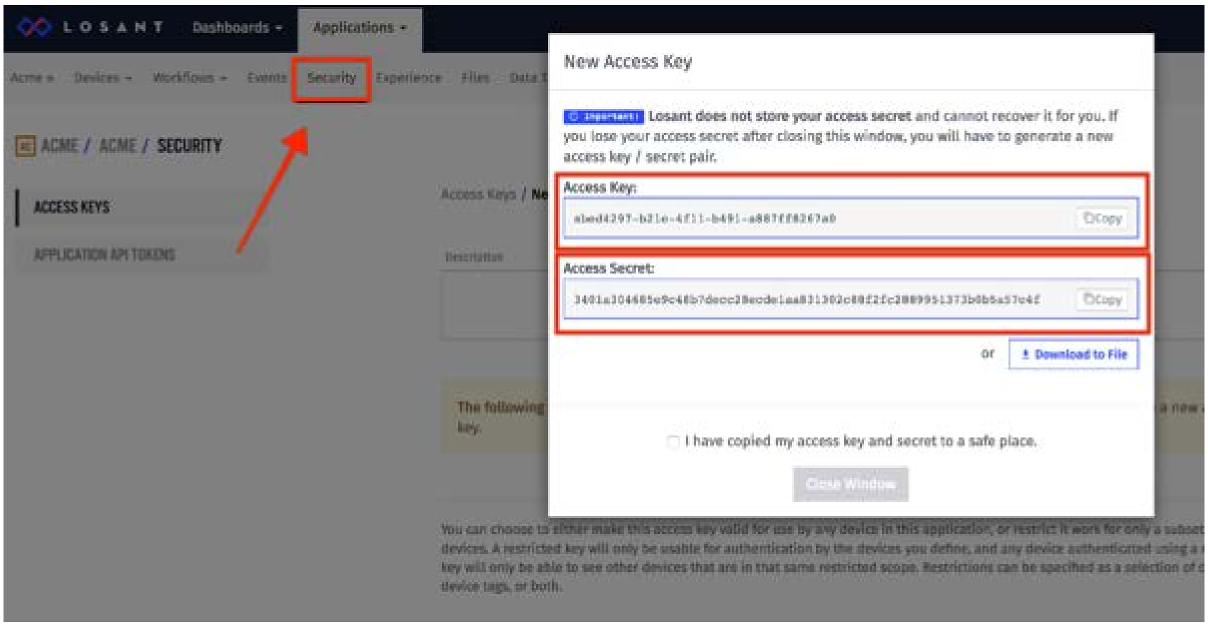
No logs yet.
Listening...

Listening...

Create Security Credentials

Now we need to generate an access key so your device can authenticate against the Losant Platform. Select Security from the application menu bar, and then Device Access Keys from the left navigation. Then click Add Access Key.

This will cause a popup to appear with your access tokens. Be sure to copy the key and secret, you will need them later.



New Access Key

Important! Losant does not store your access secret and cannot recover it for you. If you lose your access secret after closing this window, you will have to generate a new access key / secret pair.

Access Key:

```
a1ed4297-b21e-4f11-b491-a887ff8267a0
```

Access Secret:

```
3401a304685e9c48b7decc28ecde1aa831302c88f2fc2889951373b0b5a57e4f
```

I have copied my access key and secret to a safe place.

You should now have a Device ID, Access Key, and Access Secret. We will need all three to connect to Losant. It is now time to flash the firmware.

Flashing ESP32 Firmware

Open up your Arduino IDE and copy/paste the following:

```
/**

Copyright (c) 2016 Losant IoT. All rights reserved.

https://www.losant.com

*/

#include <WiFiClientSecure.h>

#include <Losant.h>

// WiFi credentials.

const char* WIFI_SSID = "WIFI_SSID";

const char* WIFI_PASS = "WIFI_PASS";

// Losant credentials.

const char* LOSANT_DEVICE_ID = "LOSANT_DEVICE_ID";

const char* LOSANT_ACCESS_KEY = "LOSANT_ACCESS_KEY";

const char* LOSANT_ACCESS_SECRET = "LOSANT_ACCESS_SECRET";
const int BUTTON_PIN = 33;
const int LED_PIN = 27;
const int TEMP_PIN = 39;
bool ledState = false;
WiFiClientSecure wifiClient;
LosantDevice device(LOSANT_DEVICE_ID);
void toggle() {
Serial.println("Toggling LED.");
ledState = !ledState;
digitalWrite(LED_PIN, ledState ? HIGH : LOW);
}
void handleCommand(LosantCommand *command) {
Serial.print("Command received: ");
Serial.println(command->name);
if (strcmp(command->name, "toggle") == 0) {
toggle();
}
}
void connect() {
// Connect to Wifi.
Serial.println();
Serial.println();
Serial.print("Connecting to ");
Serial.println(WIFI_SSID);
WiFi.begin(WIFI_SSID, WIFI_PASS);
while (WiFi.status() != WL_CONNECTED) {
```

```
delay(1000);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

Serial.println();

Serial.print("Connecting to Losant...");

device.connectSecure(wifiClient, LOSANT_ACCESS_KEY,
LOSANT_ACCESS_SECRET);

while (!device.connected()) {

delay(500);

Serial.print(".");

}

Serial.println("Connected!");

Serial.println("This device is now ready for use!");

}

void setup() {

Serial.begin(115200);

// Giving it a little time because the serial monitor doesn't

// immediately attach. Want the workshop that's running to

// appear on each upload.

delay(2000);

Serial.println();

Serial.println("Running Firmware.");
```

```
pinMode(BUTTON_PIN, INPUT);
pinMode(LED_PIN, OUTPUT);
device.onCommand(&handleCommand);
connect();
}
void buttonPressed() {
Serial.println("Button Pressed!");
StaticJsonBuffer<200> jsonBuffer;
JsonObject& root = jsonBuffer.createObject();
root["button"] = true;
device.sendState(root);
}
void reportTemp(double degreesC, double degreesF) {
StaticJsonBuffer<200> jsonBuffer;
JsonObject& root = jsonBuffer.createObject();
root["tempC"] = degreesC;
root["tempF"] = degreesF;
device.sendState(root);
}
int buttonState = 0;
int timeSinceLastRead = 0;
int tempSum = 0;
int tempCount = 0;
void loop() {
bool toReconnect = false;
if (WiFi.status() != WL_CONNECTED) {
```

```
Serial.println("Disconnected from WiFi");
toReconnect = true;
}
if (!device.connected()) {
Serial.println("Disconnected from MQTT");
Serial.println(device.mqttClient.state());
toReconnect = true;
}
if (toReconnect) {
connect();
}
device.loop();
int currentRead = digitalRead(BUTTON_PIN);
if (currentRead != buttonState) {
buttonState = currentRead;
if (buttonState) {
buttonPressed();
}
}
tempSum += analogRead(TEMP_PIN);
tempCount++;
// Report every 15 seconds.
if (timeSinceLastRead > 2000) {
// Take the average reading over the last 15 seconds.
double raw = (double)tempSum / (double)tempCount;
// The tmp36 documentation requires the -0.5 offset, but during
// testing while attached to the Feather, all tmp36 sensors

// required a -0.52 offset for better accuracy.

double degreesC = (((raw / 1023.0)) - 0.5) * 100.0;

double degreesF = degreesC * 1.8 + 32;

Serial.println();

Serial.print("Raw: ");

Serial.println(raw);

Serial.print("Temperature C: ");

Serial.println(degreesC);

Serial.print("Temperature F: ");

Serial.println(degreesF);

Serial.println();

reportTemp(degreesC, degreesF);

timeSinceLastRead = 0;

tempSum = 0;

tempCount = 0;

}

delay(100);

timeSinceLastRead += 100;

}
```

view raw esp32-arduino-losant.ino hosted with ❤ by GitHub

At the top, there are placeholders for credentials that you need to fill out. It looks like this:

```
// WiFi credentials.
const char* WIFI_SSID = "WIFI_SSID";
const char* WIFI_PASS = "WIFI_PASS";

// Losant credentials.
const char* LOSANT_DEVICE_ID = "LOSANT_DEVICE_ID";
const char* LOSANT_ACCESS_KEY = "LOSANT_ACCESS_KEY";
const char* LOSANT_ACCESS_SECRET = "LOSANT_ACCESS_SECRET";
```

You’ll need to provide:

WIFI_SSID: The name of your WiFi network.

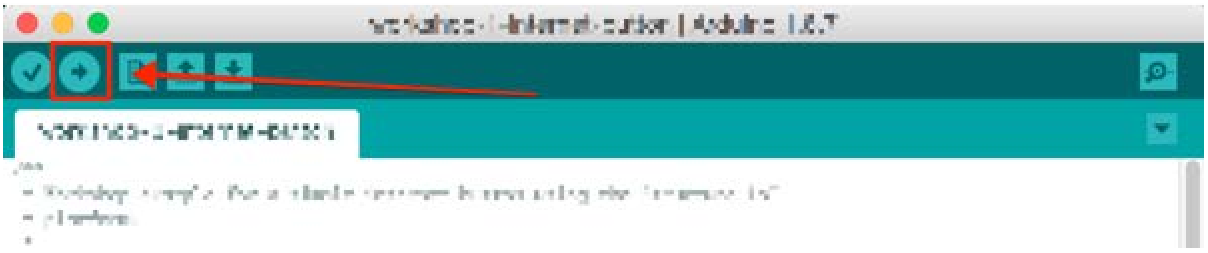
WIFI_PASS: Your WiFi password.

LOSANT_DEVICE_ID: After creating your device, the Device ID is printed on the page in a gray box. You can also find it next to the name of your device on your application's "View All Devices" page.

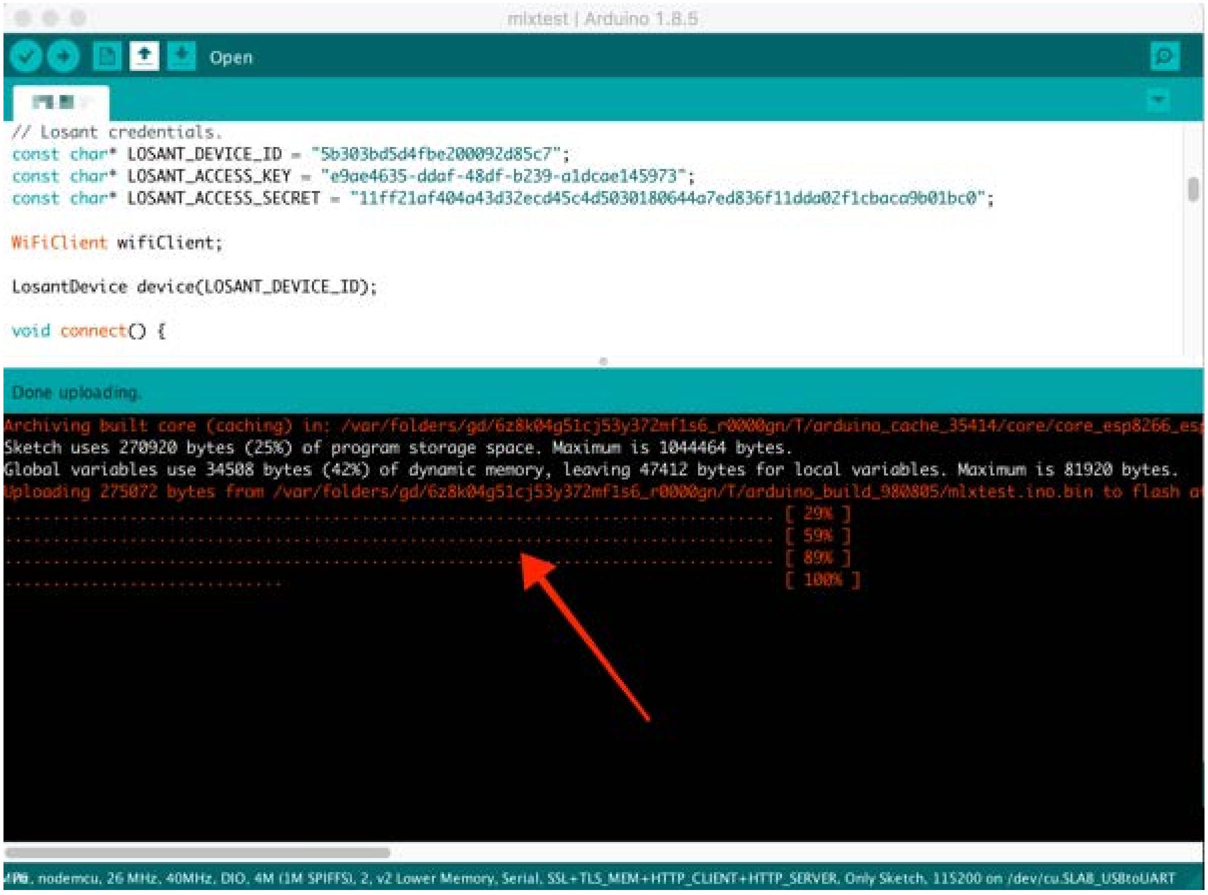
LOSANT_ACCESS_KEY: Set this to the access key you generated after creating the Losant application.

LOSANT_ACCESS_SECRET: Set this to the access secret you generated after creating the Losant application.

Next, you can begin the flashing process by pressing the upload button:



Then, you should see a successful upload message like so:



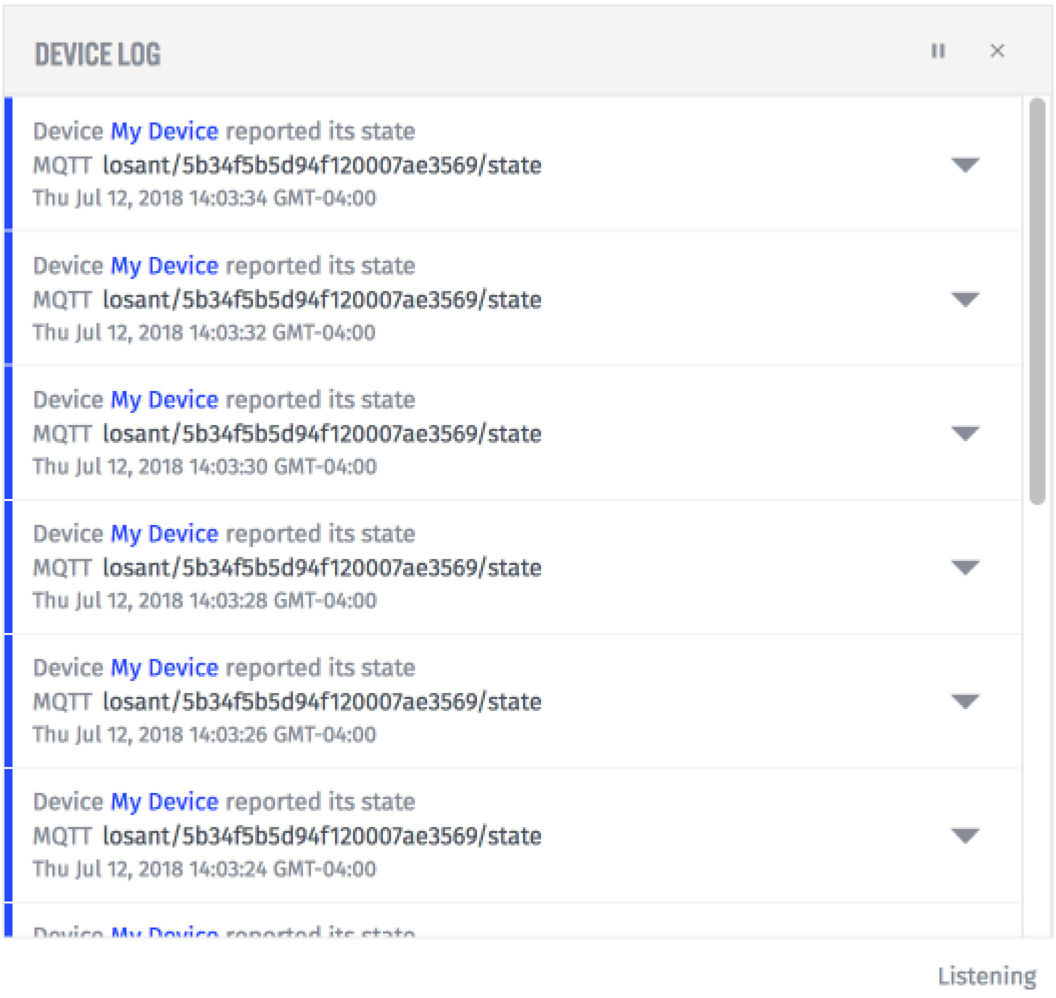
```
// Losant credentials.
const char* LOSANT_DEVICE_ID = "5b303bd5d4fbe200092d85c7";
const char* LOSANT_ACCESS_KEY = "e9ae4635-ddaf-48df-b239-a1dcae145973";
const char* LOSANT_ACCESS_SECRET = "11ff21af404a43d32ecd45c4d5030180644a7ed836f11dda02f1cbaca9b01bc0";

WiFiClient wifiClient;

LosantDevice device(LOSANT_DEVICE_ID);

void connect() {
```

```
Done uploading.
Archiving built core (caching) in: /var/folders/gd/6z8k04g51cj53y372mf1s6_r0000gn/T/arduino_cache_35414/core/core_esp8266_es
Sketch uses 270920 bytes (25%) of program storage space. Maximum is 1044464 bytes.
Global variables use 34508 bytes (42%) of dynamic memory, leaving 47412 bytes for local variables. Maximum is 81920 bytes.
Uploading 275072 bytes from /var/folders/gd/6z8k04g51cj53y372mf1s6_r0000gn/T/arduino_build_980805/mlxtest.ino.bin to flash a
................................................................................ [ 29% ]
................................................................................ [ 59% ]
................................................................................ [ 89% ]
.............................                                                    [ 100% ]
```

```
MS, nodemcu, 26 MHz, 40MHz, DIO, 4M (1M SPIFFS), 2, v2 Lower Memory, Serial, SSL+TLS_MEM+HTTP_CLIENT+HTTP_SERVER, Only Sketch, 115200 on /dev/cu.SLAB_USBtoUART
```

Verifying Connection

Once you have successfully flashed, we can check Losant to verify that it's reporting data. The best way to do so is by looking at the device logs. You can find this by looking to the right of your device's settings.

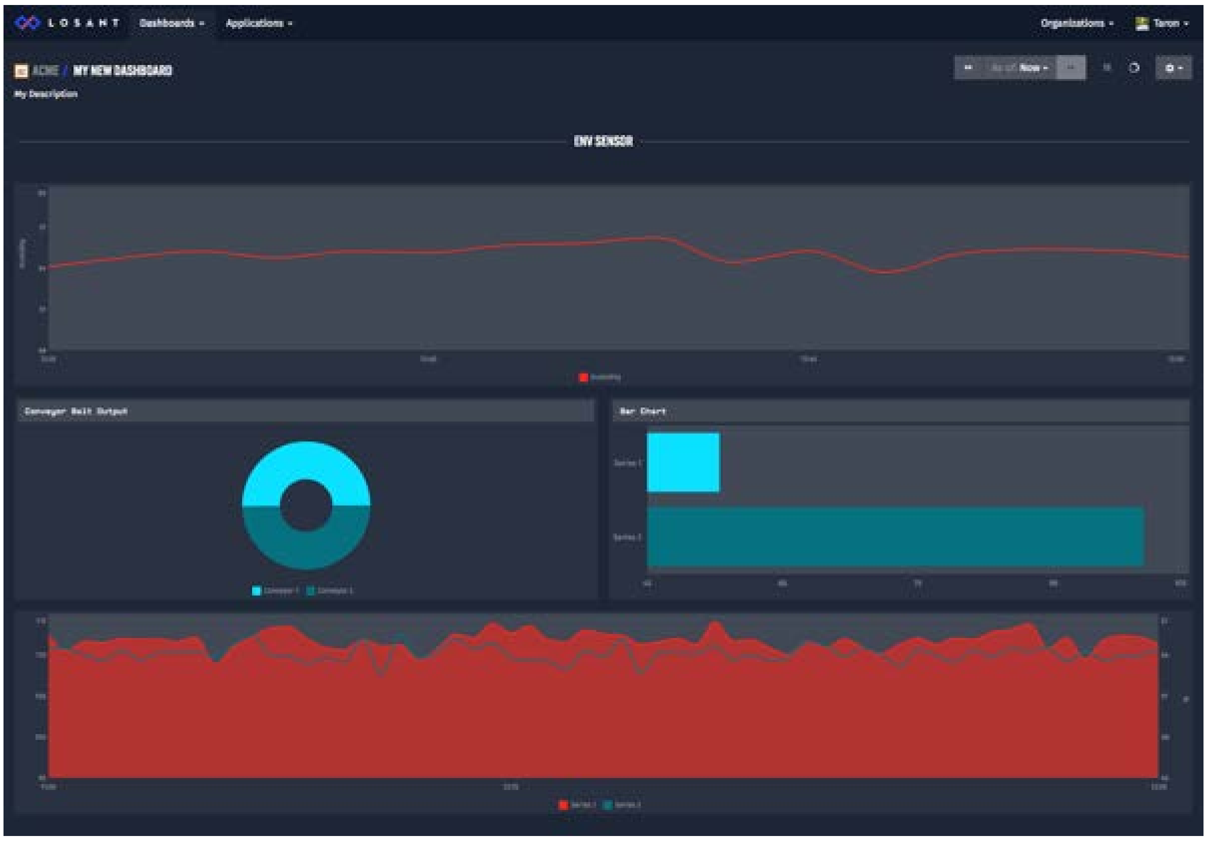


```
Device My Device reported its state
MQTT losant/5b34f5b5d94f120007ae3569/state
Thu Jul 12, 2018 14:03:34 GMT-04:00

Device My Device reported its state
MQTT losant/5b34f5b5d94f120007ae3569/state
Thu Jul 12, 2018 14:03:32 GMT-04:00

Device My Device reported its state
MQTT losant/5b34f5b5d94f120007ae3569/state
Thu Jul 12, 2018 14:03:30 GMT-04:00

Device My Device reported its state
MQTT losant/5b34f5b5d94f120007ae3569/state
Thu Jul 12, 2018 14:03:28 GMT-04:00

Device My Device reported its state
MQTT losant/5b34f5b5d94f120007ae3569/state
Thu Jul 12, 2018 14:03:26 GMT-04:00

Device My Device reported its state
MQTT losant/5b34f5b5d94f120007ae3569/state
Thu Jul 12, 2018 14:03:24 GMT-04:00

Device My Device reported its state
```

If you see logs, your device is connected and reporting properly.

That's a Wrap

Now that you have data being sent to the cloud, there is a lot you can do with it. You can react to it using the Losant Workflow Engine or even create a Dashboard. There is a lot to explore.



Resources:

[Losant Dashboards]()

[Losant Workflows]()

For more information and inspiration, check out these tutorials:

Until next time, stay connected.