ECOLE NATIONALE
SUPÉRIEURE
D'INFORMATIQUE

الجـــمهوريـــة الجزائريـــة الديمـــقراطيـــة الشـعبيـة
وزارة التعليـــم العـــالي والبـــحث العلمـــي
People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

Intelligent and Communicating Systems, ICS

2^{nd} Year Specialty SIL G1

Lab Report n°08

Title:

Iot System based Platform Wifi-Cloud and Standalone

Studied by:

First Name: Yasmine

Last Name: DINARI

E_mail: ly_dinari@esi.dz

Contents

Theory

1. IoT-Focused Cloud-Standalone Platforms

Many platforms serve the IoT field, facilitating user-friendly automation, device management, and monitoring. Aside from **Ewelink**, **Blynk**, **Arduino Cloud**, and **SmartThings**, which are already used for IoT, several other cloud-standalone solutions are tailored especially for IoT applications.

Blynk Cloud, for example, allows easy control and monitoring of Arduino, ESP, and Raspberry Pi devices via a mobile app, making it ideal for prototyping and home automation. **Arduino Cloud** is tightly integrated with the Arduino ecosystem and supports real-time device monitoring and simple dashboard creation for IoT projects. **SmartThings** by Samsung is more commercial, supporting a broad range of smart devices with advanced automation options.

array

Platform	**Features**	**Integration**	**Price**	**Device Compatibility**
Ewelink	Easy mobile control, automation, scenes	Integrates with Sonoff, Alexa, Google Home	Free with hardware, premium for more devices	Sonoff, some generic WiFi devices
Blynk	Custom dashboards, notifications, automation	Arduino, ESP32, Raspberry Pi support	Free tier, paid advanced features	Arduino, ESP, Raspberry Pi, NodeMCU
Arduino Cloud	Real-time monitoring, easy setup, OTA updates	Native integration with Arduino boards	Free basic tier, paid for extra devices	Arduino IoT devices
SmartThings	Advanced automation, routines, voice control	Many third-party smart devices	Free with Samsung, optional premium	Samsung, Z-Wave, Zigbee, WiFi devices

2. Other Well-Known IoT Cloud-Standalone Platforms

There are other cloud-standalone platforms, for instance:

- **ThingsBoard**: This is an open-source IoT platform for data collection, processing, visualization, and device management. It supports both cloud and on-premise deployment. It is suitable for industrial and smart city projects, offering dashboards, alarms, and integration with various protocols. A community edition is free, while advanced enterprise features are paid.
- **Particle Cloud**: A commercial IoT platform designed for rapid prototyping and deployment of connected products. It provides device management, firmware updates, and secure communications. Pricing varies, with a free tier for small projects and paid plans for scaling.
- **AWS IoT Core**: A scalable, fully managed cloud platform from Amazon, suitable for connecting billions of IoT devices. It provides robust security, analytics, and integration with other AWS services. It is commercial with a pay-as-you-go pricing model, targeting industrial and enterprise IoT.
- **Microsoft Azure IoT Hub**: Cloud service that enables secure communication, monitoring, and management of IoT devices. It is widely used in both industrial and consumer IoT. Pricing is usage-based, and the platform is closed-source but highly scalable.

These platforms differ in openness (open-source or proprietary), pricing models (free community versions or paid enterprise options), and target use cases, such as industrial IoT, smart home, or prototyping.The choice of platform depends on required features, scalability, budget, and support for specific hardware.

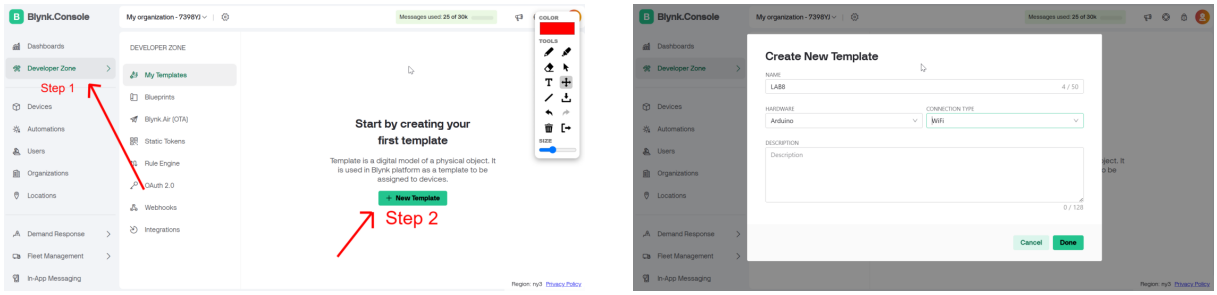
Activity

1. Cloud-based with Blynk

We connect a simple IoT system (Arduino and LED) to the Blynk cloud platform for remote control and monitoring:

1.1. Remote Control Steps:

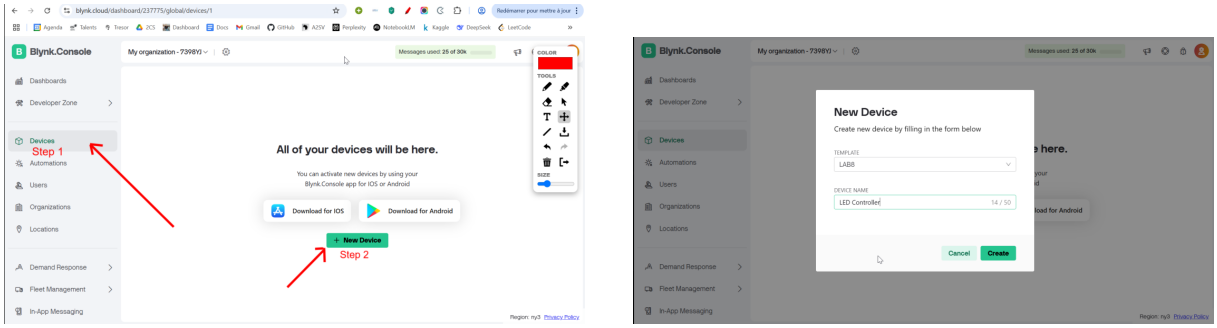
- Create a Blynk account: Go to the Blynk platform and sign up for an account.
- Create a new template: In the Developer Zone, go to “My Templates” and click “Add New Template”. Enter a name, set the hardware type to Arduino, and the connection type to Wi-Fi. Note the following details: **BLYNK_TEMPLATE_ID, BLYNK_TEMPLATE_NAME, BLYNK_AUTH_TOKEN**



(a) Adding Template (b) Template creation

Figure 2.1: Template creation steps.

- Add a new device: In the Devices section, add a new device “from template” and assign the template you created to your device.



(a) Add device (b) Device details

Figure 2.2: Device setup (1/2).

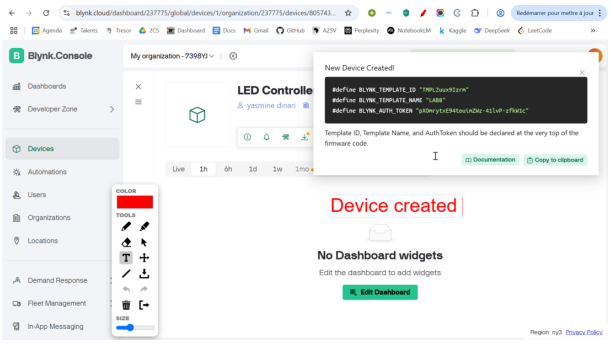
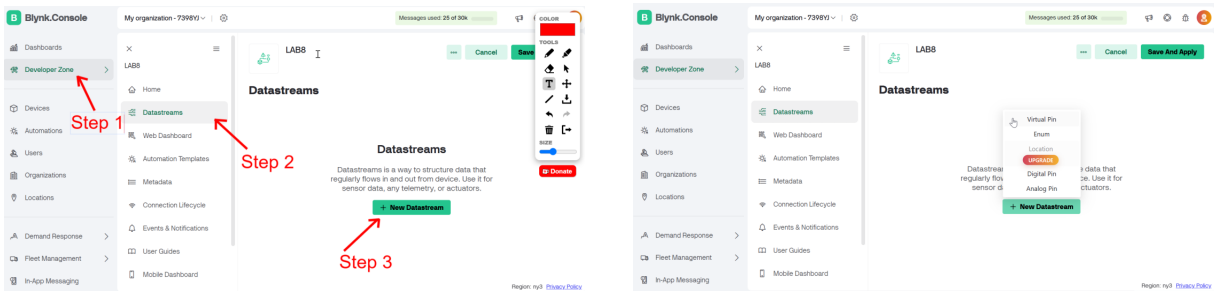


Figure 2.3: Device created

- Create a virtual pin: Select your device, go to the DataStreams tab, and add a DataStream of type “Virtual Pin” (e.g., V0) for LED control. Set the data type (e.g., Integer) and save.



(a) Add datastream (b) Add virtual pin

Figure 2.4: Virtual pin creation (1/2).

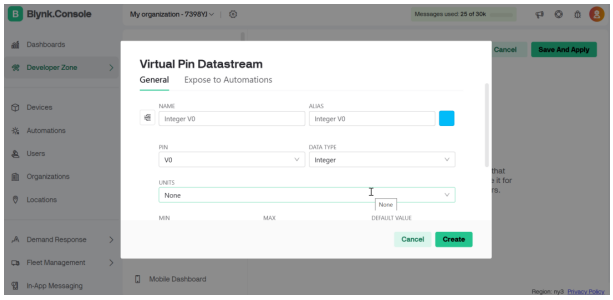


Figure 2.5: Virtual pin details

- Set up the mobile app: Open the Blynk app on your phone, select your project, and add a Switch widget. Link this widget to the virtual pin (V0) you created for the LED and connect your Arduino and run the code below:

```
#define BLYNK_TEMPLATE_ID "TMPL2uux9Izrm"
#define BLYNK_TEMPLATE_NAME "LAB8"
#define BLYNK_AUTH_TOKEN "pXOmrytxE94touimZWz-41lvP-zfkW1c"

// Include required libraries for Blynk and WiFi
#include <WiFiNINA.h>
```

```
#include <BlynkSimpleWiFiNINA.h>

// Wi-Fi credentials (change these to match your local network)
char ssid[] = "ALHN-A013";          // Wi-Fi name (SSID)
char pass[] = "9999999";            // Wi-Fi password

void setup() {
  Serial.begin(115200);
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
  pinMode(7, OUTPUT);             // Set digital pin 7 as an output (connected
    to LED)
}

BLYNK_WRITE(V0) {
  int ledState = param.asInt(); // Read value from Blynk (0 or 1)
  digitalWrite(7, ledState);     // Turn LED ON if 1, OFF if 0
}

void loop() {
  Blynk.run();
}
```

Listing 2.1: Blynk LED Control

Figure 2.6: Arduino circuit that controls a LED

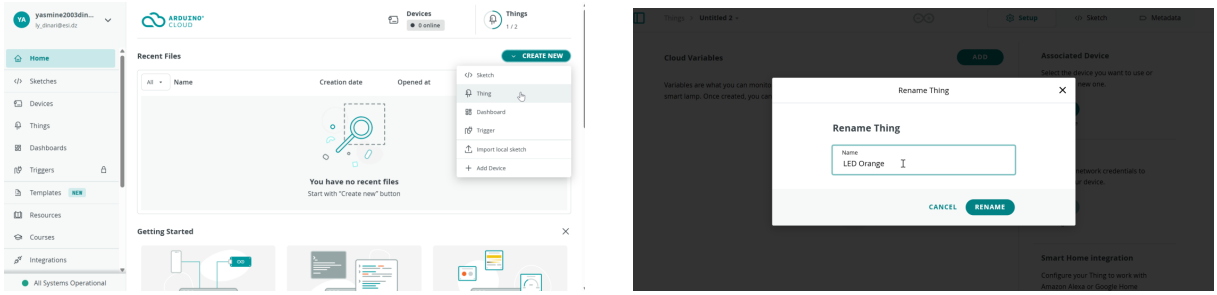
2. Cloud-based with Arduino IoT Cloud

This section demonstrates connecting a simple IoT system (Arduino and LED) to the Arduino IoT Cloud platform for remote control and monitoring, following the same

principle as with Blynk (see Lab 8 and course week 9).

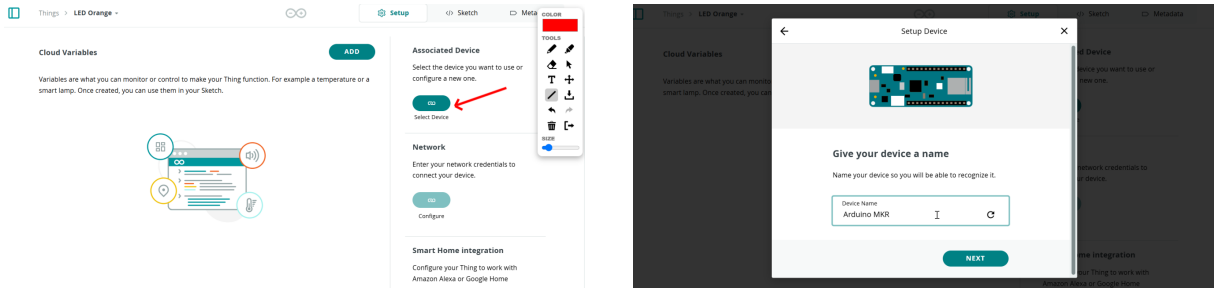
2.1. Remote Control Steps:

- **Create an Arduino IoT Cloud account:** Go to the Arduino IoT Cloud website and register for an account.
- **Add and configure your device:** In the IoT Cloud dashboard, add a new device and link your Arduino board (e.g., Arduino MKR, UNO WiFi, or compatible board) via USB.
- **Create a Thing and define variables:** Make a new "Thing" and create a cloud variable (e.g., `orange_led_light`) with type *Boolean* (read/write) to control the LED.



(a) Add Thing (b) Name Thing

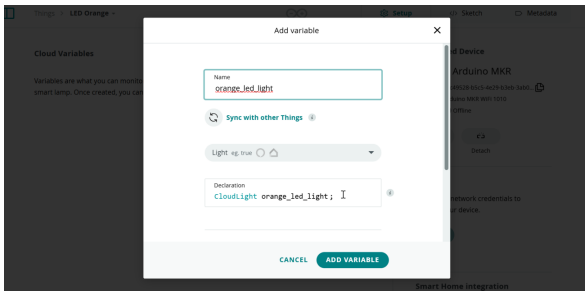
Figure 2.7: Create and name a Thing.



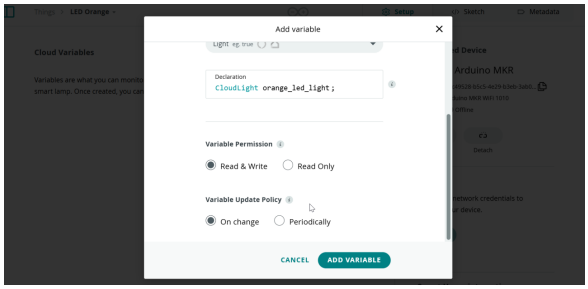
(a) Associate Device (b) Name Device

Figure 2.8: Associate and name your device.

- **Upload the code:** Copy code bellow or use the code provided below. Upload it to your Arduino board using the Arduino IDE.
- **Configure the dashboard:** Add a Switch widget to the dashboard and link it to the variable `orange_led_light`. This allows remote control of the LED from web/mobile.
- **Test your project:** Use the dashboard or the Arduino IoT Cloud mobile app to toggle the LED from your PC or phone.

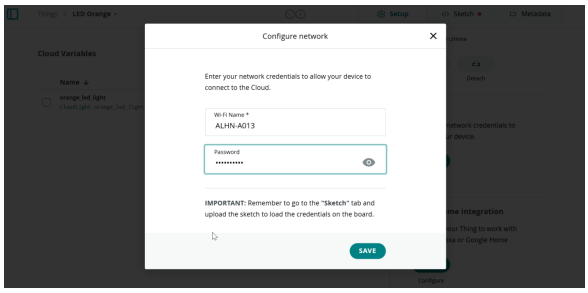


(a) Add Variable

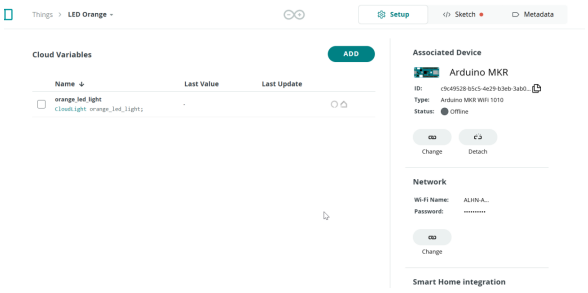


(b) Variable Details

Figure 2.9: Create and configure the variable.



(a) Network Config



(b) Final Config

Figure 2.10: Final network and board configuration.

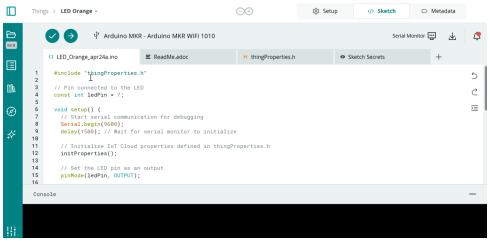
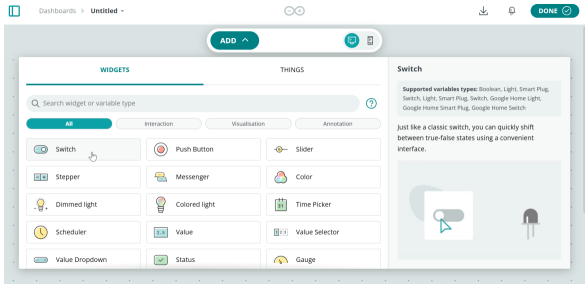
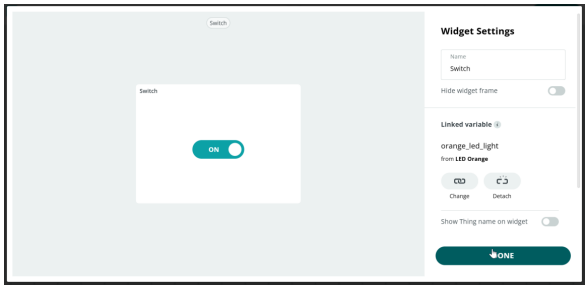


Figure 2.11: Upload code in Sketch



(a) Add Switch Widget



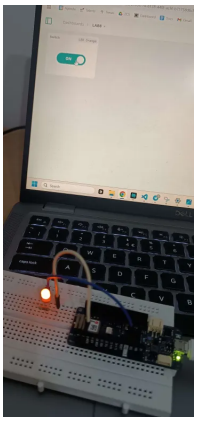
(b) Link Switch to Variable

Figure 2.12: Dashboard configuration for remote control.

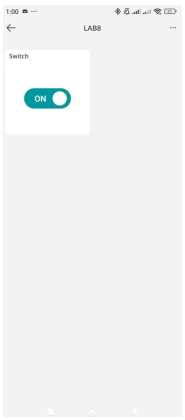
Arduino IoT Cloud Code:

```
#include "thingProperties.h"
const int ledPin = 7;

void setup() {
  Serial.begin(9600);
```



(a) Test from PC



(b) Test from Phone

Figure 2.13: Test the LED remotely.

```
  delay(1500);
  pinMode(ledPin, OUTPUT);
  initProperties();
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();
}

void loop() {
  ArduinoCloud.update();
}

// This function is called when 'orange_led_light' changes in the cloud
void onOrangeLedLightChange() {
  Serial.print("LED state changed to: ");
  Serial.println(orange_led_light);
  if (orange_led_light) {
    digitalWrite(ledPin, HIGH);   // LED ON
    Serial.println("LED is ON");
  } else {
    digitalWrite(ledPin, LOW);    // LED OFF
    Serial.println("LED is OFF");
  }
}
```

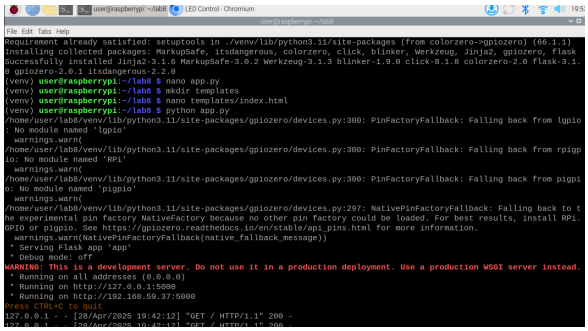
Listing 2.2: Arduino IoT Cloud LED Control

Note: Both Arduino and your phone/PC must have internet access for real-time cloud control, just as when using Blynk.

3. Standalone Mode: Local IoT Platform with Raspberry Pi

In standalone mode, we control an LED from a simple web interface running locally on a Raspberry Pi, without relying on a cloud platform. This is useful if you want your

IoT system to work even without internet access, or to keep all data and processing inside your own network.

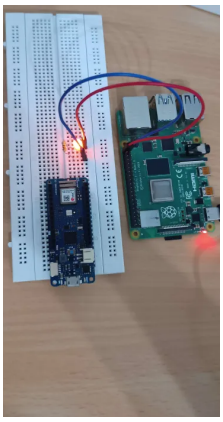


(a) Running the Flask app

(b) LED turned off (web interface)

Figure 2.14: Launching the server and user interface for LED control.

(a) LED turned on (web interface)



(b) LED physically on in the circuit

3.1. Steps to Deploy Standalone Mode

- **Update the Raspberry Pi:**

```
sudo apt update
sudo apt upgrade
```

- **Install Python virtual environment:**

```
sudo apt install python3-venv
```

- **Create a project folder:**

```
mkdir lab8
cd lab8
```

- **Create and activate a virtual environment:**

```
python3 -m venv venv
source venv/bin/activate
```

- **Install required Python libraries:**

```
pip install flask gpiozero
```

- **Create the Flask application:** Save the following code as `app.py` in your project folder.

```python
from flask import Flask, render_template
from gpiozero import LED

app = Flask(__name__)
led = LED(21)  # GPIO pin 21

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/on')
def led_on():
    led.on()
    return render_template('index.html', status="LED is ON")

@app.route('/off')
def led_off():
    led.off()
    return render_template('index.html', status="LED is OFF")

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

Listing 2.3: Flask app for standalone LED control

- **Create the HTML user interface:** In the `templates` directory, create a file `index.html`:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>LED Control</title>
    <style>
        body { text-align: center; margin-top: 50px; font-family: Arial; }
        button { padding: 15px 30px; font-size: 18px; margin: 10px; cursor: pointer; }
    </style>
</head>
<body>
    <h1>Control the LED</h1>
```

```
13
14      {% if status %}
15      <h2>{{ status }}</h2>
16      {% endif %}
17
18      <a href="/on"><button style="background-color: lime;">Turn ON</
        button></a>
19      <a href="/off"><button style="background-color: red;">Turn OFF</
        button></a>
20  </body>
21  </html>
```

Listing 2.4: HTML interface for controlling the LED

- **Start the Flask web server:**

  ```
  python app.py
  ```

- **Access the web interface:** From any device on the same local network, open a browser and go to:

  ```
  http://<raspberry-pi-ip>:5000
  ```

  For example: `http://192.168.1.50:5000`

Note: No remote or cloud service is necessary. The system works entirely in our local network, giving us complete control over our data and setup. This approach is ideal for privacy-focused or offline IoT projects, and can easily be extended to larger systems with more devices and controls.

Conclusion

In this lab, we explored three different approaches to IoT device control: cloud-based platforms like Blynk and Arduino IoT Cloud, and a standalone local web server using Flask on Raspberry Pi. Each method allowed remote control and monitoring of an LED, but differed in terms of required connectivity, setup, and data privacy. Cloud-based solutions are convenient and accessible from anywhere, while the local standalone mode offers control inside the local network without internet dependency. Overall, these experiments provided practical insight into designing flexible IoT systems suitable for various environments and requirements.