_Projet CleanVibes

Créer un bot Telegram

1. **Créer le Bot :**

    - Chercher l'utilisateur **@BotFather** dans Telegram.
    - Lui envoyer la commande `/newbot`.
    - Donner un nom (ex: `FabLabProtoBot`) et un nom d'utilisateur (doit finir par `bot`, ex: `MonProjet_bot`).
    - **IMPORTANT :** BotFather va donner un **API Token** (une longue chaîne de caractères). _Il faut le copier._

2. **Trouver son propre ID (Chat ID) :**

    - Le bot doit savoir à _qui_ envoyer le message.
    - Chercher l'utilisateur **@userinfobot** (ou `@myidbot`).
    - Lui envoyer n'importe quel message.
    - Il répond avec l'ID (une suite de chiffres, ex: `123456789`). _Il faut le copier._

3. **Initialiser la conversation :**

    - **Vital :** Chercher _son_ nouveau bot (celui créé à l'étape 1) et cliquer sur **DÉMARRER** (ou envoyer `/start`).
    - _Si on ne fait pas ça, le bot n'a pas le droit d'envoyer le premier message (anti-spam)._

```
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>
#include <ArduinoJson.h>

// --- Configuration ---
const char* ssid = "NOM_DU_WIFI";
const char* password = "MOT_DE_PASSE";

// Infos Telegram
#define BOTtoken "XXXXXXXXX:AAHdqTcvCH1vGWJxf..." // Token donné par BotFather
#define CHAT_ID "123456789" // ID donné par userinfobot

// --- Objets ---
WiFiClientSecure client;
```

```
UniversalTelegramBot bot(BOTtoken, client);

void setup() {
Serial.begin(115200);

// 1. Connexion WiFi
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
Serial.println("\nWiFi connecté !");

// 2. Configuration Sécurité (L'ASTUCE EST ICI)
client.setInsecure(); // <--- Ignore la vérification du certificat SSL
// Si on voulait faire propre, il faudrait setCACert() et avoir l'heure précise.

// 3. Test d'envoi au démarrage
bot.sendMessage(CHAT_ID, "Salut ! Le prototype est allumé.", "");
}

void loop() {
// Ici, tu mets ta condition (ex: si vibration détectée)
// if (vibration) {
// bot.sendMessage(CHAT_ID, "Alerte ! Mouvement détecté !", "");
// delay(5000); // Anti-spam
// }
}
```

https://espressif.github.io/arduino-esp32/package_esp32_index.json

https://randomnerdtutorials.com/esp32-useful-wi-fi-functions-arduino/

Prompt à utiliser pour avoir du code propre :

> Tu vas agir comme un expert en systèmes embarqués et un mentor pédagogique au sein d'un FabLab. Je suis étudiant et je réalise un prototype.

Voici ma configuration matérielle (Hardware) :

1. Carte de développement : Adafruit Feather ESP32-S2.
2. Capteurs : BME280 (Température/Pression/Humidité) et ADXL345 (Accéléromètre), branchés en I2C.
3. Actionneur : Une LED RGB Neopixel intégrée à la carte.

Voici tes contraintes de programmation (Software) :

1. Langage : C++ pour l'IDE Arduino.
2. Vitesse Série : Toujours utiliser Serial.begin(115200);
3. Bibliothèques : Privilégie les bibliothèques officielles "Adafruit" (Adafruit_NeoPixel, Adafruit_BME280, Adafruit_ADXL345).
4. Structure :
   - Découpe ton code en fonctions claires (ne mets pas tout dans le loop).
   - Ajoute des commentaires expliquant la logique pour que je puisse apprendre.
   - Au début du code, définis les broches (PINS) avec des #define.
5. Sécurité : Ajoute toujours dans le "setup" une vérification que le capteur est bien détecté (avec un while(1) ou un message d'erreur si la connexion échoue).

Mon objectif est de comprendre ce que je fais. Si je te demande un code, ne me donne pas juste la solution : explique brièvement les fonctions clés utilisées.

Si tu as compris, réponds juste par : "Prêt à coder avec le Feather ESP32 ! Qu'est-ce qu'on fait ?"

```
// NeoPixel Ring simple sketch (c) 2013 Shae Erisson
// Released under the GPLv3 license to match the rest of the
// Adafruit NeoPixel library

#include <Adafruit_NeoPixel.h>

// Which pin on the Arduino is connected to the NeoPixels?
#define PIN        6 // On Trinket or Gemma, suggest changing this to 1

// How many NeoPixels are attached to the Arduino?
#define NUMPIXELS 1 // Popular NeoPixel ring size

// When setting up the NeoPixel library, we tell it how many pixels,
// and which pin to use to send signals. Note that for older NeoPixel
// strips you might need to change the third parameter -- see the
// strandtest example for more information on possible values.
Adafruit_NeoPixel pixels(NUMPIXELS, PIN_NEOPIXEL, NEO_GRB + NEO_KHZ800);

#define DELAYVAL 500 // Time (in milliseconds) to pause between pixels
```

```
void setup() {

  pixels.begin(); // INITIALIZE NeoPixel strip object (REQUIRED)
}

void loop() {
    pixels.clear(); // Set all pixel colors to 'off'

    // pixels.Color() takes RGB values, from 0,0,0 up to 255,255,255
    // Here we're using a moderately bright green color:
    pixels.setPixelColor(0, pixels.Color(150, 0, 0));

    pixels.show();   // Send the updated pixel colors to the hardware.

    delay(DELAYVAL); // Pause before next pass through loop

    pixels.clear();
    pixels.show();
    delay(DELAYVAL);
  }



#include <Wire.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#include <Adafruit_NeoPixel.h>


#define BME_SCK 13
#define BME_MISO 12
#define BME_MOSI 11
#define BME_CS 10
#define NUMPIXELS 1

#define SEALEVELPRESSURE_HPA (1013.25)
Adafruit_NeoPixel pixels(NUMPIXELS, PIN_NEOPIXEL, NEO_GRB + NEO_KHZ800);
#define DELAYVAL 500

Adafruit_BME280 bme; // I2C
//Adafruit_BME280 bme(BME_CS); // hardware SPI
//Adafruit_BME280 bme(BME_CS, BME_MOSI, BME_MISO, BME_SCK); // software SPI

unsigned long delayTime;
```

```
void pixel_on (int r, int g, int b) {
pixels.clear();
pixels.setPixelColor(0,pixels.Color(r, g, b));
pixels.show();
}

void pixel_off(){
pixels.clear();
pixels.show();
}

void setup() {

Serial.begin(9600);
while(!Serial); // time to get serial running
Serial.println(F("BME280 test"));

unsigned status;

// default settings
status = bme.begin();
// You can also pass in a Wire library object like &Wire2
// status = bme.begin(0x76, &Wire2)
if (!status) {
Serial.println("Could not find a valid BME280 sensor, check wiring, address, sensor ID!");
Serial.print("SensorID was: 0x"); Serial.println(bme.sensorID(),16);
Serial.print(" ID of 0xFF probably means a bad address, a BMP 180 or BMP 085\n");
Serial.print(" ID of 0x56-0x58 represents a BMP 280,\n");
Serial.print(" ID of 0x60 represents a BME 280.\n");
Serial.print(" ID of 0x61 represents a BME 680.\n");
while (1) delay(10);
}

Serial.println("-- Default Test --");
delayTime = 1000;

Serial.println();
}


void loop() {
printValues();

if(bme.readTemperature()<30){
pixel_on(0, 150, 0);
}
else {
```

```
pixel_on(150, 0, 0);
}
delay(delayTime);
}


void printValues() {
Serial.print("Temperature = ");
Serial.print(bme.readTemperature());
Serial.println(" °C");

Serial.print("Pressure = ");

Serial.print(bme.readPressure() / 100.0F);
Serial.println(" hPa");

Serial.print("Approx. Altitude = ");
Serial.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
Serial.println(" m");

Serial.print("Humidity = ");
Serial.print(bme.readHumidity());
Serial.println(" %");

Serial.println();
}
```

Revision #9
Created 2 February 2026 16:09:31 by Stephane Muller
Updated 6 March 2026 10:01:29 by Stephane Muller