Fab Lab
Saint Gély du Fesc
Pic Saint Loup

Saint-Gély-du-Fesc
La vie côté bien-être !

Présentation du M5StickC PLUS et applications

- Mesure de la température ambiante et une température distante.
- Affichage de ces températures dans un smartphone par bluetooth.
- Transfert du programme en mode OTA (Over The AiR) et affichage des températures par client internet (M5 connecté à internet par WiFi).

François Fabre
FabLab de Saint Gély du Fesc – Pic Saint Loup
frafaster@gmail.com

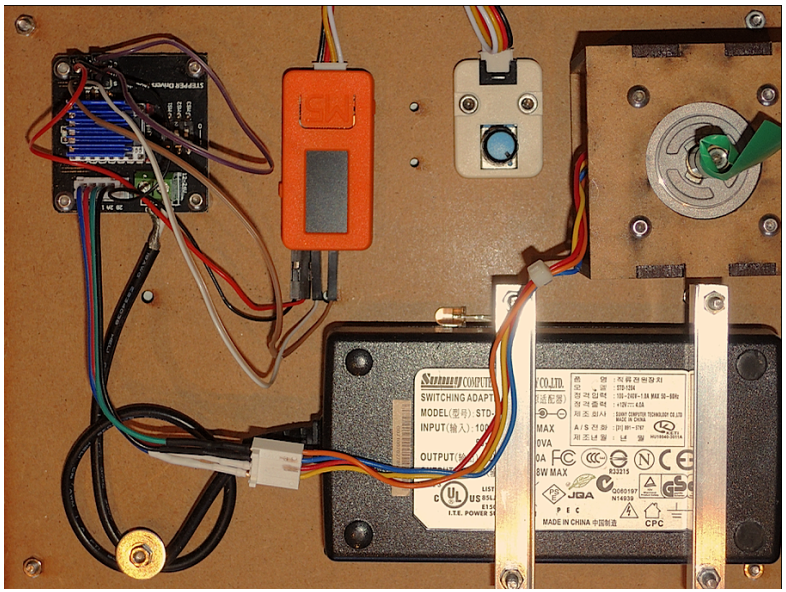
Jeudi 3 mars 2022 à 20h salle Fontgrande

Quelques utilisations du M5 StickC Plus

Montre

Avec batterie

Rover et pince



Commande de moteurs pas à pas

Joystick

Présentation du jour : trois versions d'une mesure de température.

1. **Affichage sur M5**
2. **Affichage sur un smartphone avec une liaison Bluetooth**
3. **Affichage sur un PC avec une Liaisons WiFi**

Capteur

1

Liaison I2C

Câble I2C

M5 StickC

2

Liaison Bluetooth

Liaison OTA (WiFi)

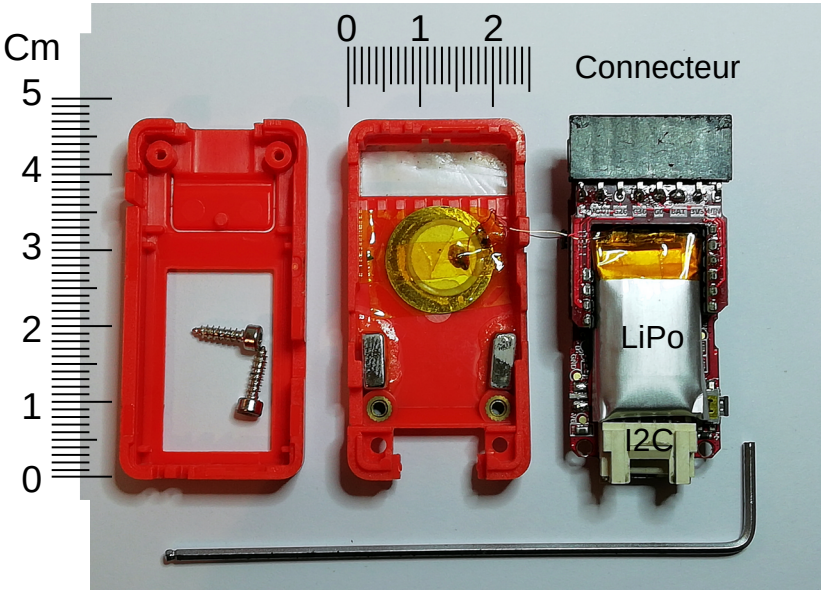
3

Propriétés du M5 StickC Plus

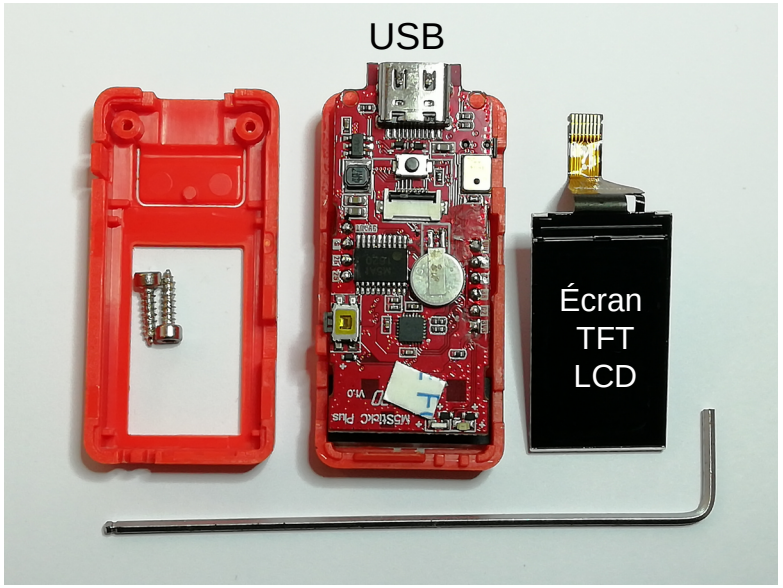
Présentation du M5 StickC Plus

Le M5+ sur le site de son fabricant M5stack :
https://shop.m5stack.com/products/m5stickc-plus-esp32-pico-mini-iot-development-kit?variant=35275856609444

Éclaté du M5 StickC Plus



Vue de dessous



Vue de dessus

Dimensions (mm) L=48, l=24, h=14

Spécifications du M5 StickC Plus

Caractéristiques du produit	Spécification
Bluetooth 4.2	ESP32 240MHz dual core, 600 DMIPS, 520KB SRAM, Wi-Fi, dual mode Bluetooth
WiFi 2.4G	Flash Memory 4MB
IMU 6 axes intégrée Gyroscope, Accéléromètre, Température	Power Input 5V @ 500mA
LED rouge	Port TypeC x 1, GROVE(I2C+I/0+UART) x 1
Emetteur infrarouge	LCD screen 1.14 inch, 135*240 Colorful TFT LCD, ST7789v2 ♥
Microphone	Button Custom button x 2
RTC	LED RED LED
3 Boutons, On:Off, A et B	MEMS MPU6886
LCD (1,14 pouces, 135 * 240)	Buzzer built-in buzzer
Batterie Lipo intégrée 120 mAh @ 3.7V	IR Infrared transmission
Connexions TypeC x 1, GROVE(I2C+I/0+UART) x 1	MIC SPM1423 Microphone numérique
Plateforme de développement UIFlow , MicroPython , Arduino	RTC BM8563 Real Time Clock
	PMU AXP192 Gestion de l'alimentation
	Battery 120 mAh @ 3.7V
	Antenna 2.4G 3D Antenna
	PIN port G0, G25/G36, G26, G32, G33
	Operating Temperature 0°C to 60°C
	Net weight 15g
	Gross weight 21g
	48.2*25.5*13.7mm
	65*25*15mm
	Case Material Plastic (PC)

♥ Écran couleur TFT (Thin-Film Transistor technology) LCD (Liquid-Cristal Display)

Sites web associés au M5 StickC Plus et son contrôleur ESP32 PICO MINI

- Le M5+ sur le site de son fabricant M5Stack :

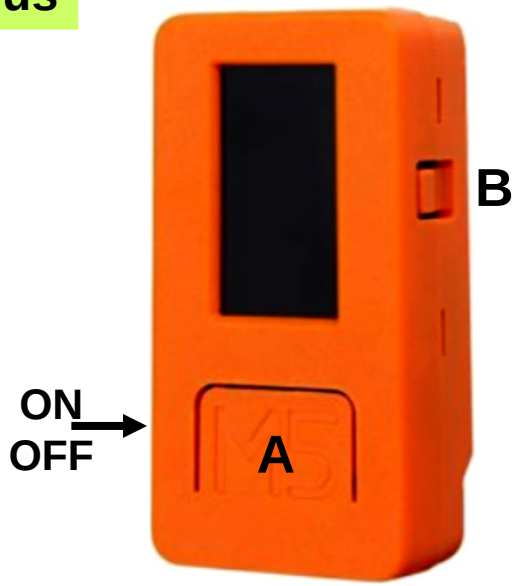
https://shop.m5stack.com/products/m5stickc-plus-esp32-pico-mini-iot-development-kit?variant=35275856609444

- ESP32 PICO MINI sur le site du fabricant Espressif :

https://www.espressif.com/sites/default/files/documentation/esp32-pico-mini-02_datasheet_en.pdf

- Dépôt Github : https://github.com/m5stack/M5StickC-Plus

Boutons et broches du M5 StickC Plus



Les 3 Boutons du M5 Stickc Plus

Le bouton d'**alimentation** se trouve sur le coté gauche
Mise sous tension du M5 : appuyer 2s
Mise hors tension du M5 : appuyer 6s

À programmer
Le bouton A se trouve sur le dessus (M5)
Le bouton B se trouve sur le coté droit

5 broches sont disponibles. Une configuration possible est par exemple :
Une broche ADC,
Un port I2c (SDA/SCL),
Une connecteur série (RX/TX).

Cela parait pauvre, mais je vous le promets si elles sont bien utilisées on arrive à s'en débrouiller. D'autant plus que, bien des composants internes sont déjà disponibles à l'intérieur.

On trouve aussi une masse, une sortie 5 volts, une sortie en 3,3 volts, un accès direct à la batterie et un port de charge en 5 volts.

Brochage du M5 StickC Plus (vue arrière)

PORT I2C

Broche « G35/G36 »

L'utilisation des ports "G36/G25" est un peu particulière. Les ports G36 et G35 partagent la même broche.

Par exemple, pour utiliser la broche G36 comme entrée ADC. Il faut configurer le port G25 comme FLOATING dans le « setup ».

Exemple

```
setup()
{
M5.begin();
pinMode(36, INPUT);
gpio_pulldown_dis(GPIO_NUM_25);
gpio_pullup_dis(GPIO_NUM_25);
}
```

Pour utiliser le port G25 on déclare FLOATING le port G36.

Structure de l'alimentation électrique du M5 StickC Plus

Le M5 dispose d'un circuit intégré AXP 192 qui contient une batterie Li-ion de capacité 120 mAh.

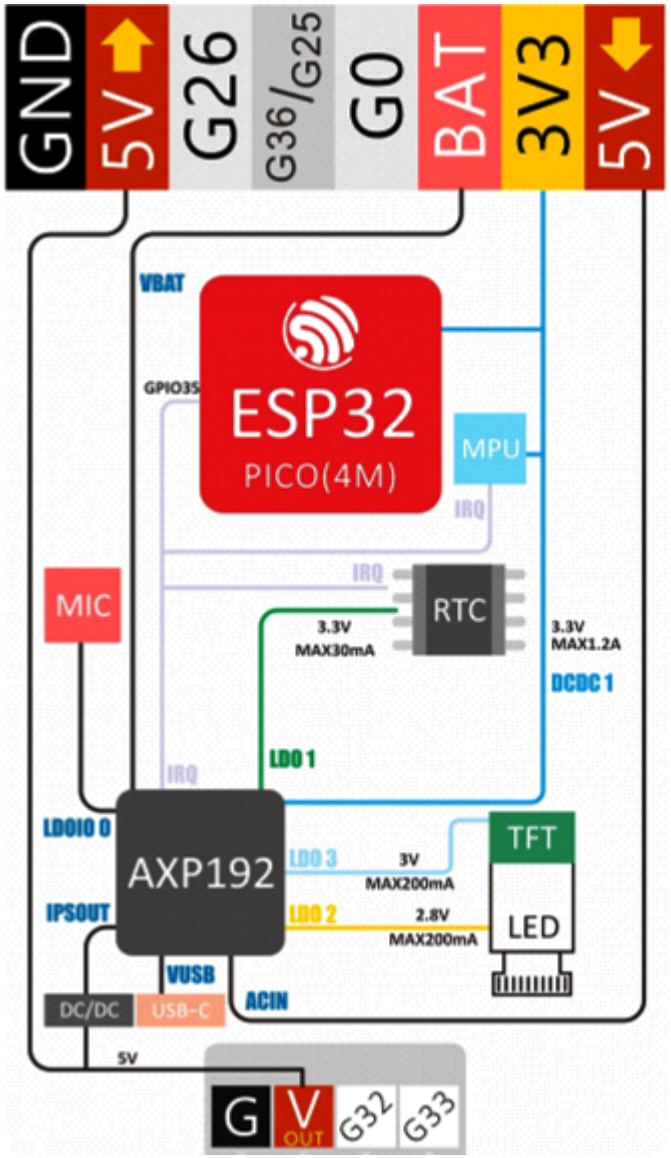
La batterie se charge par le port USB ou par l'entrée 5V.

L'AXP 192 fourni 3 circuits de régulation faible bruit LDO (Low DropOut regulator) qui alimentent divers circuits : LED (LDO2), l'écran (LDO3)

L'AXP 192, alimente en 3.3 V:

- L'ESP32,
- Le circuit RTC (LDO1)
- L'IMU

L'AXP 192, alimente en 5V la broche V_{out}



Programmation du **M5 StickC Plus** pour les mesures des températures

issues du capteur **MLX90614**

et affichées sur l’écran du M5StickC Plus

Installation du M5 StickC Plus dans l'IDE d'Arduino

On va installer le M5+ avec l'Environnement de Développement Informatique d'Arduino (IDE)

Dans un premier temps il va falloir configurer l'IDE d'Arduino.
Le site web https://docs.m5stack.com/en/quick_start/m5stickc_plus/arduino
explique en détails les opérations à effectuer.

L'API d'Arduino donne des exemples d'utilisations de la librairie #include <M5stickCPlus.h> :
- Système.
- AXP192 : Gestion de l'alimentation.
- LCD : Gestion de l'écran TFT LCD.
- IMU : Gestion des paramètres du gyroscope et de la température de l'IMU.
- PWM : Gestion des signaux analogiques.
- RTC : Gestion de l'heure et du temps

Quelques propriétés utiles de l'ESP32 :
- 2 cœurs à 240 MHz.
- 4 timers à 80 MHz.
- 600 MIPS (Millions of Instructions Per Second).
- 520k de Ram. (il y a de quoi faire !).
- 4Mo de mémoire Flash.

Propriétés du capteur de température MLX90614 version DCI

Ce capteur permet deux mesures de température :
La température ambiante (en fait la température du circuit).
La température d'un objet distant basé sur le rayonnement infrarouge émis par l'objet.

Ce capteur peut être utilisé entre 3 et 5 Volts, il intègre un régulateur 3.3 V.

Il communique avec le M5 par liaison I2C (**I**nter **I**ntegrated **C**ircuit)

Propriétés particuliaires de la version DCI
D Précision médicale (±0.03°C autour de 37°C).
C Compensation du gradient de température.
I Angle du champ de vision 3°(FOV - Field Of View).

Plage de température du capteur ambiant : -40 à +125 ° C. Précision de 0,5 °C entre 0 et 50 °C.

Plage de température de l'objet pour une mesure distante -70 à + 380 ° C
Précision pour les hautes température ±5°C

Résolution numérique des résultats : jusqu'à 0,02 ° C. (Rien à voir avec la précision de la mesure)

Recherche et installation de la librairie Adafruit associé au MLX90614

Nous allons utiliser le librairie Adafruit_MLX90614 , merci la communauté !
Site web https://github.com/adafruit/Adafruit-MLX90614-Library

On installe la librairie directement dans l'IDE Arduino. Méthode d'installation :
https://arduino.blaisepascal.fr/installer-une-bibliotheque/#Installation_automatique

Son utilisation est simple on inclut la librairie Adafruit_MLX90614.h dans le sketch :

```
#include <Adafruit_MLX90614.h>
```

On déclare la connexion du MLX90614 dans le setup :

```
void setup // init MLX90614
if (!mlx.begin()) {
Serial.println("Erreur de connexion au capteur MLX. Vérifier le câblage.");
while (1);
};
```

On dispose de 2 commandes, l'une pour relever la température Ambiante :

```
temp_Ambient = mlx.readAmbientTempC();
```

L'autre pour relever la température de l'objet distant :

```
temp_Object = mlx.readObjectTempC();
```

Sketch de mesure des températures et affichage sur M5+

Téléchargement du code Arduino :

https://e1.pcloud.link/publink/show?code=kZfcJpZEKIjcWF3QHYLHvmjcPzJH4DaDJNV

Actions disponibles via ce sketch basique :

- Un appui long sur le bouton M5 (2 secondes) va éteindre le M5+.
- Sans appui il prend des mesures continuellement.
- Un appui court bloque l'affichage de la mesure et allume la led rouge du M5+.
- Un nouvel appui court repasse le M5+ en mode de mesures continu et éteins la led rouge.

Sketch de mesure des températures et affichage sur M5+

Setup

M5StickC-MLX90614-simple §

```
/*M5StickC_MLX90614
  https://fr.aliexpress.com/item/4000089394595.html
  */
//#include <M5StickC.h>
#include "M5StickCPlus.h"
#include <Adafruit_MLX90614.h>

double temp_Ambient; double temp_Object;
double old_temp_Ambient; double old_temp_Object;

Adafruit_MLX90614 mlx = Adafruit_MLX90614();

//**********************************************setup**********************************
void setup() {
  Serial.begin(115200);
  Serial.println("Adafruit MLX90614 test");
  M5.begin();
  delay(100);

  // init sensor
  if (!mlx.begin()) {
    Serial.println("Erreur de connexion au capteur MLX. Vérifier le câblage.");
    while (1);
  };

  pinMode(M5_LED, OUTPUT);
  digitalWrite(M5_LED, HIGH); // Eteindre

  M5.Lcd.setRotation(1);
  M5.Lcd.setTextColor(WHITE, RED);//Couleur texte, Couleur fond texte
  M5.Lcd.fillScreen(RED); // Nettoyer ecran
  M5.Lcd.setCursor(30, 0, 4);// ligne1
  M5.Lcd.println("FabLab ST Gely");
  M5.Lcd.setCursor(30, 25, 4);// ligne2
  M5.Lcd.println("Mesure continue                ");

} // End setup
//**********************************************/setup*********************************
```

Mesures

```
//**********************************************loop**********************************
void loop() {
  M5.update(); //  indispensable pour boutons M5.BtnA/B/C ...

  // PowerOff via bouton M5
  if (M5.BtnA.pressedFor(2000)) {
    M5.Lcd.fillScreen(RED); // Nettoyer ecran
    M5.Lcd.setCursor(15, 60, 4); M5.Lcd.print("Power Off !"); delay(2000);
    M5.Axp.PowerOff();
  }
 // Bloquer la mesure
  if (M5.BtnA.wasPressed() && digitalRead(M5_LED) == 1) {
    M5.Lcd.setCursor(30, 25, 4);// ligne2
    M5.Lcd.println("Mesure bloquee");digitalWrite(M5_LED, LOW); // Eteindre
  } else if (M5.BtnA.wasPressed() && digitalRead(M5_LED) == 0) {
    M5.Lcd.setCursor(30, 25, 4); M5.Lcd.println("Mesure continue");
    digitalWrite(M5_LED, HIGH); // Allumer
  }
  if (digitalRead(M5_LED) == 1) {
    temp_Ambient = mlx.readAmbientTempC();temp_Object = mlx.readObjectTempC();
  } else if (digitalRead(M5_LED) == 0) {
    temp_Ambient = old_temp_Ambient;temp_Object = old_temp_Object;
  }
  M5.Lcd.setCursor(40, 75, 4);// ligne3
  M5.Lcd.println("AMB " + String(temp_Ambient) + " *C");
  M5.Lcd.setCursor(40, 100, 4);// ligne4
  M5.Lcd.println("OBJ " + String(temp_Object) + " *C");
  old_temp_Ambient  = temp_Ambient; old_temp_Object = temp_Object;
 Serial.println(); delay(100);
} // End loop
//**********************************************/loop**********************************
```

Programmation du **M5 StickC Plus** pour les mesures des températures

issues du capteur **MLX90614** et affichées sur :

- L’écran du M5StickC Plus
- Un téléphone Android à l’aide d’une connexion Bluetooth

Connexion du M5 StickC Plus à un téléphone Android par Bluetooth

Dans la mesure ou tous les **ESP 32** disposent d'une connexion Bluetooth pourquoi ne pas en profiter !

Nous allons ajouter une connexion Bluetooth à ce projet.

Pour créer l'interface graphique nous allons utiliser une application nommée « **Bluetooth Electronics** », elle est très simple d'utilisation. Elle est disponible sur le PlayStore pour Android.

Cette application gratuite est vraiment très bien faite, elle dispose de tout ce dont on peut avoir besoin pour afficher diverses mesures issues de montages électroniques ou contrôler des paramètres.

Le concepteur a même développé une astuce que je trouve **géniale :**
On peut créer l'interface graphique du téléphone connecté en lui envoyant via Bluetooth le code qui va générer cette interface graphique (**simplement génial** !).

En bref une pression sur le bouton « M5 » du M5StickC Plus suffit pour créer l'interface graphique du téléphone Android.

Code qui va créer l'interface graphique sur le client Android connecté.

```
static void sendifaceBT() {
  SerialBT.println("*.kwl");
  SerialBT.println("clear_panel()");
  SerialBT.println("set_grid_size(13,6)");
  SerialBT.println("add_text(6,3,xlarge,R,Temperature ambiante,245,240,245,)");
  SerialBT.println("add_text(6,4,xlarge,R,Temperature objet,245,240,245,)");
  SerialBT.println("add_text(6,2,xlarge,R,Bloquer la mesure,245,240,245,D)");
  SerialBT.println("add_text(9,3,xlarge,L,DegC,245,240,245,)");
  SerialBT.println("add_text(9,4,xlarge,L,DegC,245,240,245,)");
  SerialBT.println("add_text_box(7,4,2,L,,245,240,245,C)");
  SerialBT.println("add_text_box(7,3,2,L,,245,240,245,B)");
  SerialBT.println("add_button(7,1,29,A,a)");
  SerialBT.println("set_panel_notes(-,,,)");
  SerialBT.println("run()");
  SerialBT.println("*");
}
```

L'interface graphique est générée par le M5+ au départ de la connexion Bluetooth vers le téléphone Android.

C'est simplement **génial** (oui je me répète).

Le gros bouton vert permet de bloquer la mesure

Sketch complet de mesure de température avec un téléphone Android

```
/*
  M5StickC_MLX90614
  https://fr.aliexpress.com/item/4000889394595.html
*/

#include <M5StickC.h>
//#include "M5StickCPlus.h"
#include <Wire.h>
#include <Adafruit_MLX90614.h>
#include "BluetoothSerial.h"

#include <WiFi.h>
#include <WiFiClient.h>
#include <WebServer.h>
#include <ESPmDNS.h>
#include <Update.h>

double temp_Ambient;
double temp_Object;
double old_temp_Ambient;
double old_temp_Object;

#include "modota.h"

Adafruit_MLX90614 mlx = Adafruit_MLX90614();


BluetoothSerial SerialBT;
int connectBT = 0;
int old_connectBT = 0;
int update_interval = 100; // time interval in ms for updating panel indicators
unsigned long last_time = 0; // time of last update
char data_in; // data received from serial link
String text = "0"; // String for text elements
String textBouton = "Mesure continue"; // String for text elements

void callback(esp_spp_cb_event_t event, esp_spp_cb_param_t *param) {
  if (event == ESP_SPP_SRV_OPEN_EVT) {
    sendifaceBT(); // Envoie interface Bluetooth au client (passe pas totalement ... bug ??)
    Serial.println("Client Connected");
    connectBT = 1;
  }
  if (event == ESP_SPP_CLOSE_EVT ) {
    Serial.println("Client disconnected");
    connectBT = 0;
  }
}


//**********************************************sendifaceBT**********************
static void sendifaceBT() {
  SerialBT.println("*.kwl");
  SerialBT.println("clear_panel()");
  SerialBT.println("set_grid_size(13,6)");
  SerialBT.println("add_text(6,3,xlarge,R,Temperature ambiante,245,240,245,)");
  SerialBT.println("add_text(6,4,xlarge,R,Temperature objet,245,240,245,)");
  SerialBT.println("add_text(6,2,xlarge,R,Bloquer la mesure,245,240,245,D)");
  SerialBT.println("add_text(9,3,xlarge,L,DegC,245,240,245,)");
  SerialBT.println("add_text(9,4,xlarge,L,DegC,245,240,245,)");
  SerialBT.println("add_text_box(7,4,2,L,,245,240,245,C)");
  SerialBT.println("add_text_box(7,3,2,L,,245,240,245,B)");
  SerialBT.println("add_button(7,1,29,A,a)");
  SerialBT.println("set_panel_notes(-,,,)");
  SerialBT.println("run()");
  SerialBT.println("*");
}
//**********************************************/sendifaceBT**********************

//**********************************************readBT**********************
void readBT() {
  //////////////   Receive and Process Data
  if (SerialBT.available()) {
    data_in = SerialBT.read(); //Get next character

    if (data_in == 'A') { //Button Pressed
      M5.Lcd.setCursor(30, 15, 2);// ligne1
      M5.Lcd.println("Mesure bloquee                    ");
      digitalWrite(M5_LED, LOW); // Eteindre
      textBouton = "Mesure bloquee";
    }
    if (data_in == 'a') { // Button Released
      M5.Lcd.setCursor(30, 15, 2);// ligne1
      M5.Lcd.println("Mesure continue                   ");
      digitalWrite(M5_LED, HIGH); // Allumer
      textBouton = "Mesure continue";
    }
  }
  //////////////  Send Data to Android device
  unsigned long t = millis();
  if ((t - last_time) > update_interval) {
    last_time = t;

    // Update Text Element
    text = temp_Ambient; // <--- Set text to send here
    SerialBT.print("*B" + text + "*");

    // Update Text Element
    text = temp_Object; // <--- Set text to send here
    SerialBT.print("*C" + text + "*");

    // Update Text Element
    SerialBT.print("*D" + textBouton + "*");

  }

}
//**********************************************readBT**********************


//**********************************************setup**********************
void setup() {
  Serial.begin(115200);
  Serial.println("Adafruit MLX90614 test");
  M5.begin();
  delay(100);
```

Sketch complet de mesure de température avec un téléphone Android

```
117
118     modota(); // Programation par OTA
119
120     /* Bluetooth Connexion */
121     //SerialBT.setPin("1342");
122     if (!SerialBT.begin("BT-M5+Temperature")) {
123       Serial.println("Une erreur s'est produite lors de l'initialisation du
        Bluetooth");
124     } else {
125       Serial.println("Bluetooth initialisé");
126     }
127     SerialBT.register_callback(callback);
128
129     sendifaceBT(); // Envoi interface Bluetooth à client
130
131       // init sensor
132       if (!mlx.begin()) {
133         Serial.println("Erreur de connexion au capteur MLX. Vérifier le câblage.");
134         while (1);
135       };
136
137     pinMode(M5_LED, OUTPUT);
138     digitalWrite(M5_LED, HIGH); // Eteindre
139
140     M5.Lcd.setRotation(1);
141     M5.Lcd.setTextColor(WHITE, RED);//Couleur texte, Couleur fond texte
142     M5.Lcd.fillScreen(RED);
143     M5.Lcd.setCursor(20, 0, 2);// ligne2
144     M5.Lcd.println("DNS OTA: " + String(host) + ".local");
145     M5.Lcd.setCursor(30, 15, 2);// ligne1
146     M5.Lcd.println("Mesure continue                  ");
147
148   } // End setup
149   //**************************************/setup*****************************
      **********
150
151
152   //**************************************loop*****************************
      ***********
153   void loop() {
154     M5.update(); //  indispensable pour boutons M5.BtnA/B/C ...
155     server.handleClient();
156
157     // On envoie interface Bluetooth
158     if (M5.BtnA.wasPressed()) {
159       Serial.println("On envoie interface Bluetooth");
160       sendifaceBT();
161     }
162     // On envoie interface au client Bluetooth a chaque connexion !
163     if (old_connectBT  != connectBT && connectBT == 1) {
164       sendifaceBT();
165     }
166     old_connectBT = connectBT;
167     Serial.println("connectBT: " + String(connectBT));
168
169     readBT(); // Lecture de Bluetooth
170
171     // PowerOff via bouton M5
172     if (M5.BtnA.pressedFor(2000)) {
173       M5.Lcd.fillScreen(RED); // Nettoyer ecran
174       M5.Lcd.setCursor(15, 30, 4);
175       M5.Lcd.print("Power Off !");
176       delay(2000);
177       M5.Axp.PowerOff();
178     }
179
180     // Bloquer la mesure
181     if (M5.BtnA.wasPressed() && digitalRead(M5_LED) == 1) {
182       M5.Lcd.setCursor(30, 15, 2);// ligne1
183       M5.Lcd.println("Mesure bloquee                  ");
184       digitalWrite(M5_LED, LOW); // Eteindre
185     } else if (M5.BtnA.wasPressed() && digitalRead(M5_LED) == 0) {
186       M5.Lcd.setCursor(30, 15, 2);// ligne1
187       M5.Lcd.println("Mesure continue                 ");
188       digitalWrite(M5_LED, HIGH); // Allumer
189     }
190
191     if (digitalRead(M5_LED) == 1) {
192       temp_Ambient = mlx.readAmbientTempC();
193       temp_Object = mlx.readObjectTempC();
194     } else if (digitalRead(M5_LED) == 0) {
195       temp_Ambient = old_temp_Ambient;
196       temp_Object = old_temp_Object;
197     }
198
199     M5.Lcd.setCursor(0, 30, 4);// ligne2
200     M5.Lcd.println("AMB " + String(temp_Ambient) + " *C");
201     M5.Lcd.setCursor(0, 55, 4);// ligne3
202     M5.Lcd.println("OBJ " + String(temp_Object) + " *C");
203     Serial.println("M5_LED = " + String(digitalRead(M5_LED)));
204     Serial.print("Ambient = "); Serial.print(mlx.readAmbientTempC());
205     Serial.print("*C\tObject = "); Serial.print(mlx.readObjectTempC());
        Serial.println("*C");
206     // Serial.print("Ambient = "); Serial.print(mlx.readAmbientTempF());
207     //Serial.print("*F\tObject = "); Serial.print(mlx.readObjectTempF());
        Serial.println("*F");
208     old_temp_Ambient  = temp_Ambient;
209     old_temp_Object = temp_Object;
210
211     Serial.println();
212     delay(100);
213   } // End loop
214   //**************************************/loop*****************************
      **********
215
```

Programmation du **M5 StickC Plus** pour les mesures des températures

issues du capteur **MLX90614** et affichées sur :

- L’écran du M5StickC Plus
- Un PC à l’aide d’une connexion WiFI

Transfert d’un fichier binaire compilé par WiFi.

Mode OTA (Over The Air)

Utilisation du mode OTA (Over The Air) avec M5 StickC Plus

Le but est de vous faire découvrir le mode OTA qui est souvent méconnu.

Le mode OTA offre la possibilité de transférer un **sketch Arduino compilé en binaire** via une connexion WiFi.

Nous allons voir comment implémenter le mode OTA dans notre M5+.

Ce mode vous évitera de vous connecter en filaire à votre M5+ pour transférer le Sketch Arduino. Il existe 2 exemples dans l'IDE Arduino pour les M5+ **BasicOTA** et **OTAWebUpdate**r, ils sont accessibles par le menu : Fichier Exemples /Pour M5Stick-C-Plus/Exemples/ ArduinoOTA

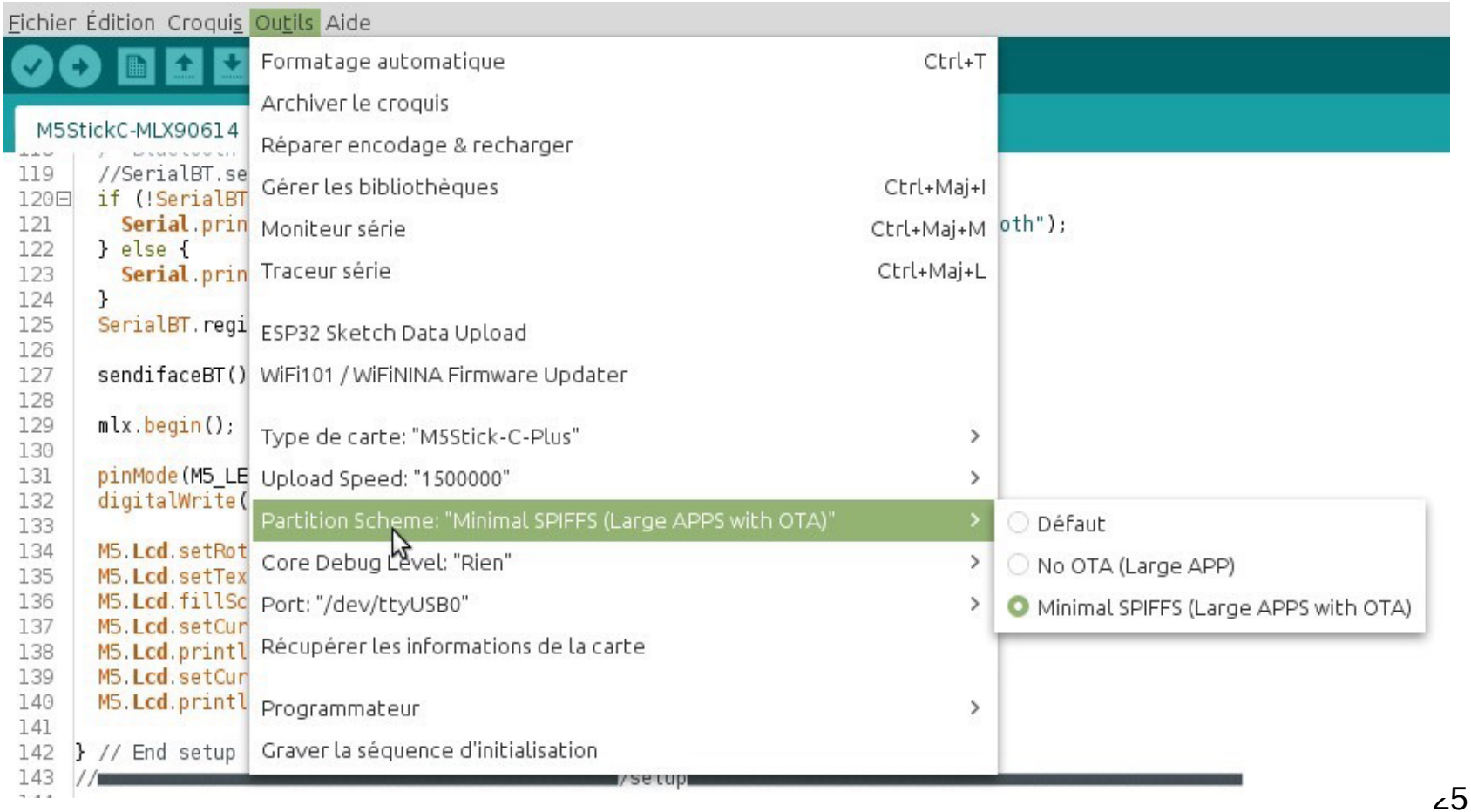
BasicOTA vous permettra de faire le transfert en WiFi de manière traditionnelle par l'IDE Arduino via une connexion WiFi. Une fois le sketch transféré dans votre M5+ un nouveau port va apparaître dans le menu « Ports Réseau »

OTAWebUpdater va permettre de faire le transfert par un client internet, c'est ce mode que j'ai implémenté sur mon M5+.

Il faut pour que ce mode de transfert fonctionne soit avoir transféré notre Sketck Arduino, soit avoir transféré un des deux exemples disponibles.

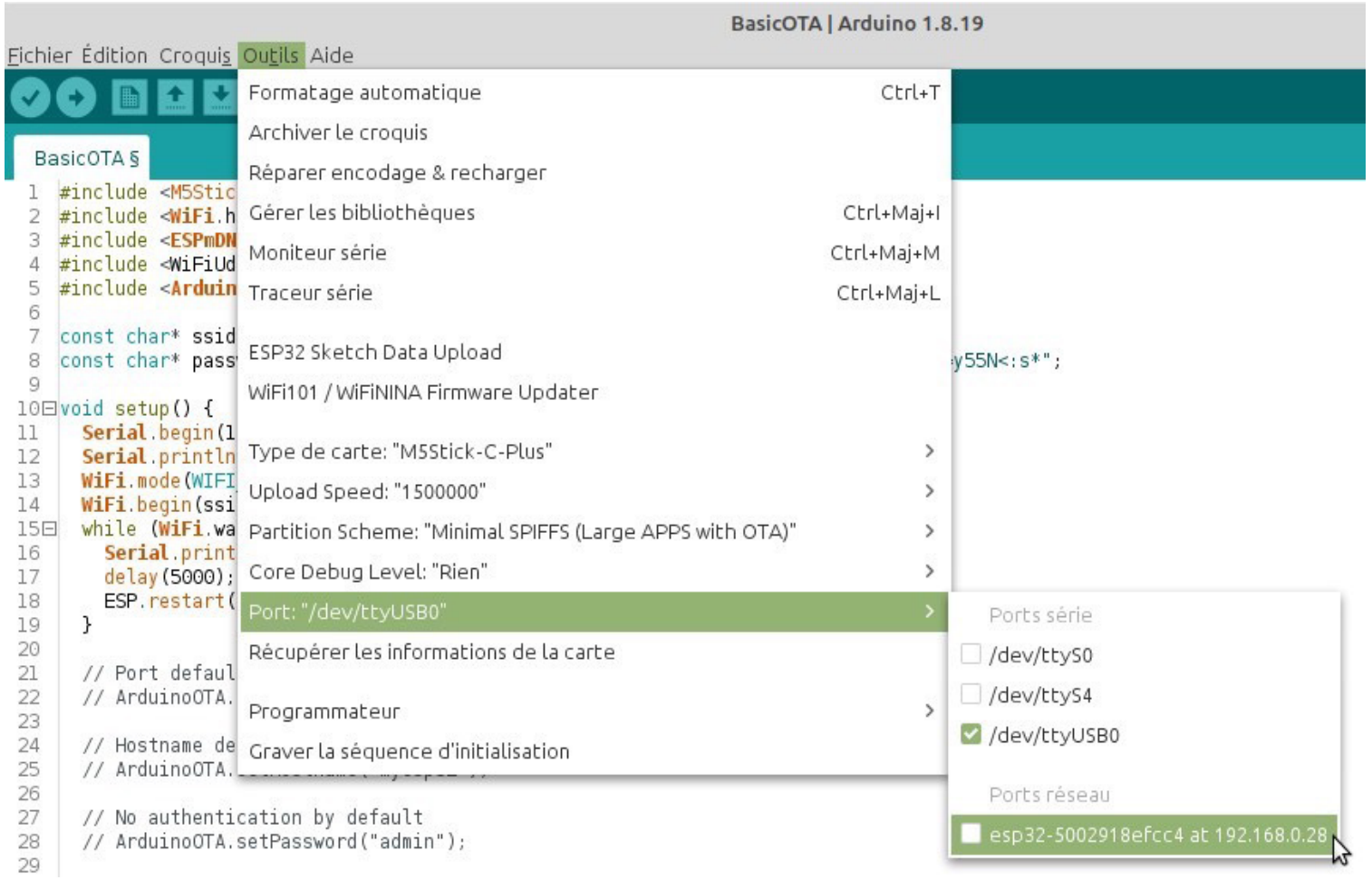
Utilisation du mode OTA (Over The Air) avec M5 StickC Plus

Pour le mode OTA, en premier lieu, comme indiqué sur l'écran ci-dessous, il faut optimiser la mémoire flash du M5+ à fin de pouvoir transférer notre Sketch avec le mode OTA.



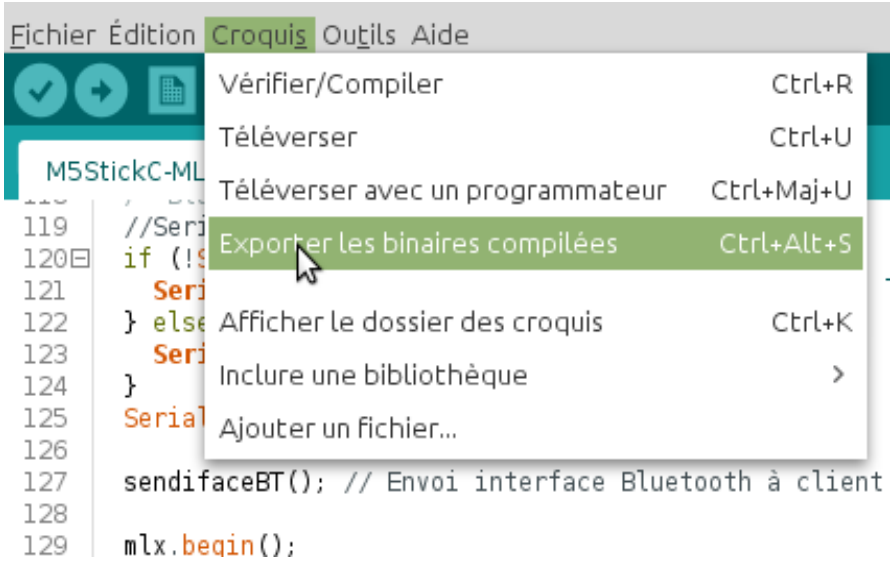
Utilisation du mode OTA (Over The Air) avec M5 StickC Plus

Une fois q'un Sketch a été transféré avec **BasicOTA** un nouveau « Ports Réseau » est disponible.

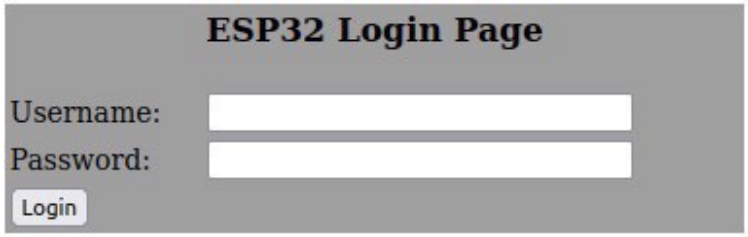


Utilisation du mode OTA (Over The Air) avec M5 StickC Plus

En WiFi on exporte que des fichiers binaires compilés. L'écran ci-dessous explique comment obtenir ce type de fichier.

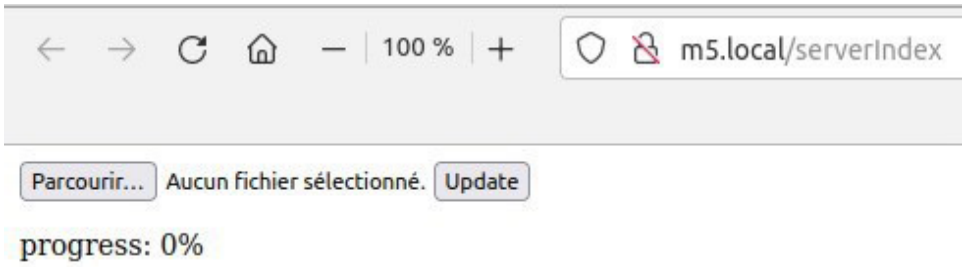


J'ai intégré le Sketch OTAWebUpdater (onglet modota.h) dans mon code, j'ai modifié la variable « host » en « const char* host = "**m5**"; »
Cela va permettre de se connecter facilement à notre M5+ a l'aide du DNS local : http://m5.local
On arrive sur cette page, par défaut Username: admin et Password: admin, **à modifier bien sur !**

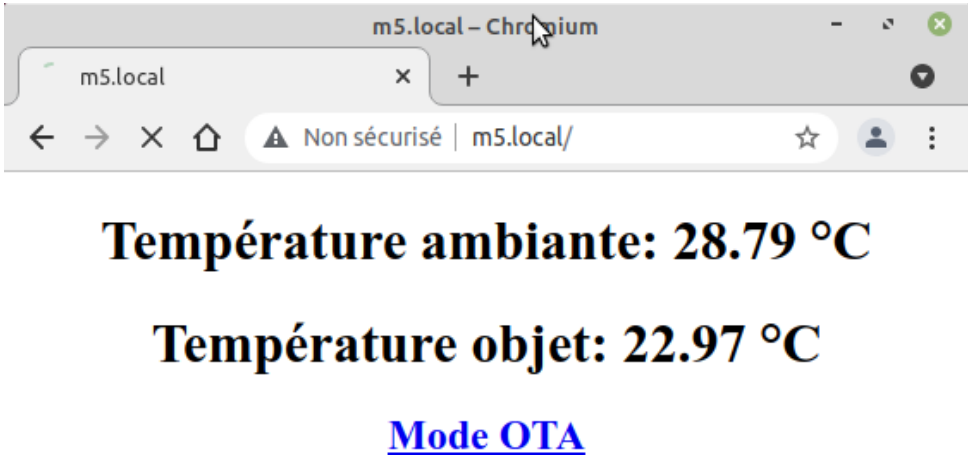


Utilisation du mode OTA (Over The Air) avec M5 StickC Plus

Une fois logué on va pouvoir sélectionner notre binaire compilé (disponible dans le dossier du sketch) et le transférer !

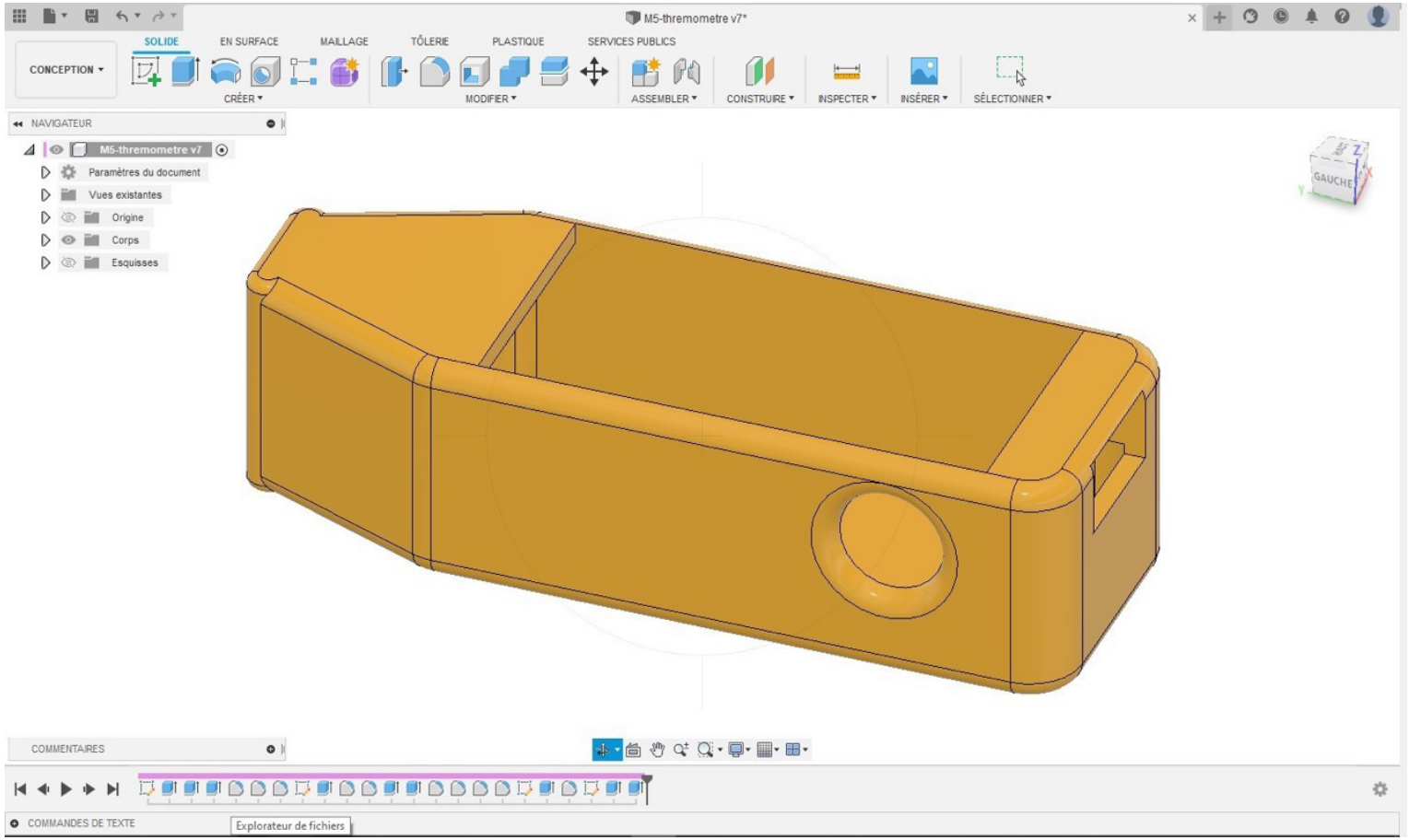


En ajoutant le mode « OTA » nous avons installé un serveur Web, alors pourquoi s'en priver on va ajouter une page de monitoring des températures qui sera accessible directement par le DNS http://m5.local/



Conception d’un boitier contenant le thermomètre et le M5 StickC Plus à l’aide du logiciel de CAO « Fusion 360 »

Téléchargement : https://a360.co/3ARqN8L



Boitier réalisé par une I3d en FDM

Conclusion

J’espère vous avoir intéressé dans cet exemple d’utilisation d’un M5+.
Je suis à votre disposition pour vous présenter d’autres projets utilisant le M5+

Présentation réalisée pour le FABLAB SAINT GELY-DU-FESC PIC SAINT LOUP

Par François Fabre
Email : frafaster@gmail.com