## Identifiants

ID : ETU

Mdp : PolyTech24

Box 4G : EPU-Borne4G-4

## Etapes d’installation

Suivre les instructions suivantes :

Dans un terminal installer les librairies nécessaires via les commandes suivantes :

* sudo apt-get update
* sudo apt-get install git-core
* sudo apt-get install libssl-dev
* sudo apt install python3
* sudo apt install -y jq
* curl -SsL https://packages.httpie.io/deb/KEY.gpg | sudo gpg --dearmor -o /usr/share/keyrings/httpie.gpg
* sudo echo "deb [arch=amd64 signed-by=/usr/share/keyrings/httpie.gpg] [https://packages.httpie.io/deb ./](https://packages.httpie.io/deb%20./)"
* sudo apt install httpie

Créer un dossier libraries

Suivre instruction pour installer les librairies :

<https://raspberry-projects.com/pi/programming-in-c/io-pins/bcm2835-by-mike-mccauley>

<https://eclipse.dev/paho/index.php?page=clients/c/index.php>

## Liens github

Projet version TP public :

<https://github.com/AvidsenRD/TP_Polytech>

Projet version Complet privé

<https://github.com/AvidsenRD/TP_Polytech_Complet>

## Compiler programme en C

Pour compiler un fichier .c, il faut exécuter cette ligne de commande :

gcc [nom\_fichier].c -o [nom\_executable] -l [librairie1] -l [librairie2]

Pour compiler plusieurs fichiers il est possible de les rajouter directement dans la commande après [nom\_fichier].c

Attention à bien mettre le chemin relatif au fichier s’il se situe à un autre emplacement.

-lm permet de compiler maths.h

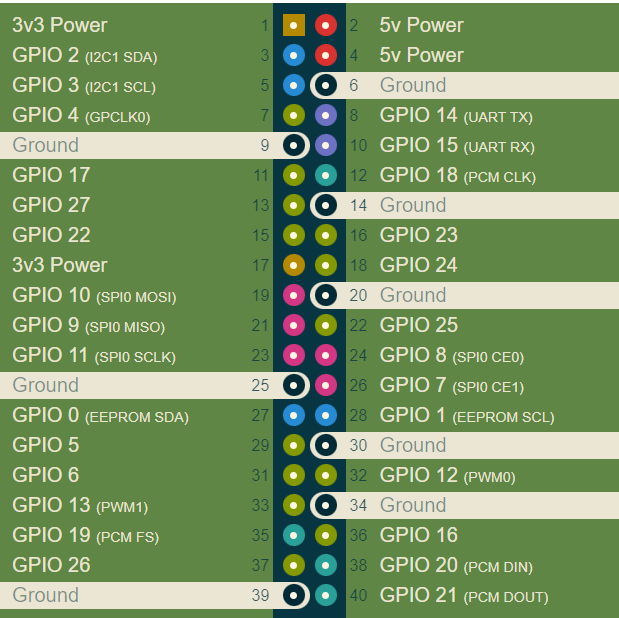
|  |  |
| --- | --- |
| Nom librairie | Description |
| bcm2835 | Gestion des GPIO |
| paho-mqtt3a | Communication MQTT asynchrone |
| paho-mqtt3c | Communication MQTT synchrone |

Lancer le script build\_script.sh permet de compiler et d’exécuter le programme.

Il peut arriver qu’il faille changer les droits de ce fichier, chmod 777 build\_script.sh, permet de donner les droits voulus.

Utiliser sudo lorsqu’il faut utiliser les PWM (pour le servo moteur ou le chandelier).

## Disposition pins Raspberry



LedExtGauche

LedExtDroite

Lustre

Bouton

ADC\_DATAOUT

ADC\_SELPIN

ADC\_DATAIN

ADC\_SPICLOCK

Chauffage

Capteur garage

Capteur Lum

Capteur rampe

Porte avant

Garage

## Etiquetage des connectiques de la maison

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nom | Pin | 5V | GND | Pin Arduino |
| LED ext porte | bleu c bleu f |  | bleu c rouge | 13 |
| LED lustre | rose blanc |  | rose orange | 12 |
| LED ext garage | bleu f |  | bleu f blanc | 11 |
| Porte avant | Bleu c noir | Rouge blanc | Rouge noir | 10 |
| Porte garage | Vert c | Rose bleu f | Vert c noir | 6 |
| Capteur rampe | Bleu c | Vert c rouge | Vert f | 5 |
| Moteur garage | blanc | rouge | noir | 3 |
| Capteur Lum | Vert c | azur vert |  | A0 |

## Câblage ADC

L’adc utilisé est un MCP3208

## Nom produits serveur

Ne pas mettre même ID

|  |  |  |  |
| --- | --- | --- | --- |
| user\_name | mqtt\_client\_id | realName (<object\_type>/<object\_reference>/<mqtt\_client\_id>) | object\_type\_id |
| Light Ext Droite | IDmaison0 | light/light\_2ch/IDmaison0  **ou** light/ 2ch/IDmaison0 | AtHome2ChLight |
| Motor Garage | IDmaison1 | shutter/micrs/IDmaison1 | AtHomeModuleShutter |
| Chandelier | IDmaison2 | Light/light\_rgb/IDmaison2 | AtHomeLight |
| Luminosity | IDmaison3 | heater/boiler/IDmaison3 | AtHomeBoiler |
| Rampe | IDmaison4 | heater/boiler/IDmaison4 | AtHomeBoiler |
| Porte Garage | IDmaison5 | heater/boiler/IDmaison5 | AtHomeBoiler |
| Porte Avant | IDmaison6 | heater/boiler/IDmaison6 | AtHomeBoiler |
| Thermostat | IDmaison7 | sensor/temp\_sensor/IDmaison7 | AtHomeProbe |

## Gestion des produits sur le serveur

Pour ajouter un produit, ouvrir un terminal :

* ./moria.sh <https://donkey.athemium.com>
* dain login '<user\_login>' '<user\_password>'
* durin
* post /my/objects name='<user\_name>' realName='<object\_type>/<object\_reference>/<mqtt\_client\_id>' typeName='<object\_type\_id>'

Par exemple :

* post /my/objects name='Porte Avant' realName='heater/boiler/IDmaison6' typeName='AtHomeBoiler'

Pour supprimer un produit, ouvrir un terminal :

* ./moria.sh <https://donkey.athemium.com>
* dain login '<user\_login>' '<user\_password>'
* durin
* delete /my/objects/<object\_id\_bdd>

Pour trouver <object\_id\_bdd> :

* ./moria.sh <https://donkey.athemium.com>
* dain login '<user\_login>' '<user\_password>'
* durin
* get /my/objects

*si l’objet ne s’affiche pas, il est possible qu’il soit dans la deuxième page :*

* get /my/objects?page=2  
  *Récupérer la valeur du champ « id »*

## Topics

Chaque objet a des topics pour communiquer avec le serveur :

|  |  |  |
| --- | --- | --- |
| Objet | topic | description |
| light/light\_2ch | s*n* | Etat de la lumière  *n* = numéro du channel |
| light/light\_rgb | lu | Etat lumière |
| light/light\_rgb | te | Valeur intensité lumineuse |
| shutter/micrs | st | Niveau de fermeture |
| heater/boiler | st | Etat du capteur |
| sensor/temp\_sensor | tt | Température consigne |
| sensor/temp\_sensor | mt | Température mesurée |

Attention, les topic provenant du serveur commencent tous par un u.

## MQTTX

Pour simuler l’envoie de commande MQTT et les objets, il est possible d’utiliser l’application MQTTX.

Les objets sont à connecter avec les paramètres suivants :

Name : *Nom de l’objet*  
Client ID : *mqtt\_client\_id*  
Host : mqtt:// donkey.athemium.com  
Port : 1883  
Username : *realName* (sans ID)  
password : *pass*

MQTT version : 3.1.1  
  
Pour envoyer les commandes, il faut créer un client spécifique :

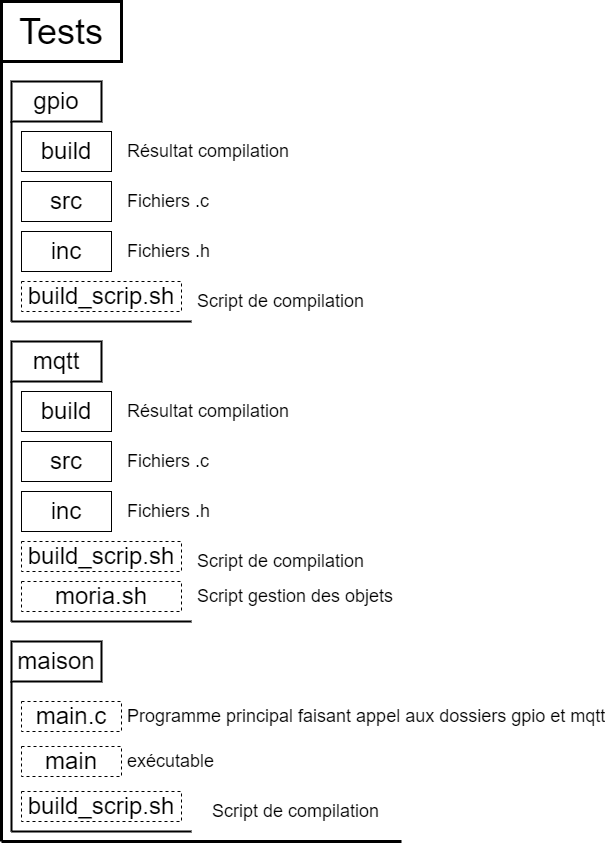
Name : *Nom*  
Client ID : mqttx\_c4317cec   
Host : mqtts:// donkey.athemium.com  
Port : 8883  
Username : virtual/probe  
Password : pass  
SSL/TLS : OK  
CA File : *certificat serveur*

MQTT version : 3.1.1

Pour envoyer les commandes avec l’objet virtual/probe, il faut se mettre en mode Hex, le topic est composé de : *ID*/*topic*  
La commande doit être composée d’octet, pour envoyer 1, il faut envoyer 01, pour 0, il faut envoyer 00.

Exemple pour allumer une

## Architecture du projet



Dans le code, chaque élément de la maison est géré via une structure qui lui est propre. Les éclairages sont gérés dans ***api\_led*,** les capteurs de porte et le capteur de luminosité dans ***api\_hall***, le servo moteur dans ***api\_servo***, la connexion MQTT dans ***api\_mqtt*,** le bouton du toit dans ***api\_button***. Le thermostat via l’***api\_thermostat*** fait appel à l’***api\_adc*** pour mesurer la température.

Avant l’utilisation de chaque élément, il faut impérativement l’initialiser en utilisant la fonction d’initialisation de l’api.

L’***api\_mqtt*** regroupe chaque élément sous un seul type de structure nommée ***Object***. Elle contient toutes les informations nécessaires à la communication MQTT de l’objet.

Il faut impérativement que chaque objet ait un MQTTClient unique pour éviter les conflits.

## Liens utiles

|  |  |
| --- | --- |
| Description | Lien |
| Documentation librairie paho | https://eclipse.dev/paho/files/mqttdoc/MQTTClient/html/index.html |
| Documentation bcm2835 | https://www.airspayce.com/mikem/bcm2835/index.html |
| Pin raspberry | https://pinout.xyz/pinout/ground |
| Doc NTC (NCP18WF104F) | https://files.seeedstudio.com/wiki/Grove-Temperature\_Sensor\_V1.2/res/NCP18WF104F03RC.pdf |
| Calcul steinhart hart | https://www.thinksrs.com/downloads/programs/therm%20calc/ntccalibrator/ntccalculator.html |