

SAÉ 304 : Découvrir et mettre en place un réseau IoT

Jalon 2: Implémentation des Super-Pouvoirs

Ce jalon correspond à la [UserStory2](#). L'objectif principal était de permettre aux joueurs d'activer des super-pouvoirs ou des malus via des à un microcontrôleur ESP8266 que va simuler une gant e envoyer des messages MQTT au système qui déclenche l'activation du super-pouvoir ou du malus dans Minecraft. Pour réaliser cela, nous avons utilisé l'IDE Arduino pour configurer l'ESP8266, ainsi que des scripts Python pour intégrer ces fonctionnalités à Minecraft.

1. Configuration de Mosquitto (Client et Serveur)

La première étape consistait à installer et configurer Mosquitto, un broker MQTT, sur un Raspberry Pi. Une fois le client et le serveur installés, nous avons testé leur bon fonctionnement à l'aide de commandes simples de publication et de souscription.

```
lidian@raspberrypi:/etc/apt/sources.list.d $ sudo service mosquitto status
Failed to get journal cutoff time: Message invalide
● mosquitto.service - Mosquitto MQTT Broker
  Loaded: loaded (/lib/systemd/system/mosquitto.service; enabled; vendor preset: enabled)
  Active: active (running) since Tue 2025-01-07 10:31:05 CET; 3min 14s ago
    Docs: man:mosquitto.conf(5)
          man:mosquitto(8)
  Process: 2995 ExecStartPre=/bin/mkdir -m 740 -p /var/log/mosquitto (code=exited, >
  Process: 2996 ExecStartPre=/bin/chown mosquitto /var/log/mosquitto (code=exited, >
  Process: 2997 ExecStartPre=/bin/mkdir -m 740 -p /run/mosquitto (code=exited, stat>
  Process: 2998 ExecStartPre=/bin/chown mosquitto /run/mosquitto (code=exited, stat>
Main PID: 2999 (mosquitto)
  Tasks: 1 (limit: 3933)
     CPU: 148ms
    CGroup: /system.slice/mosquitto.service
           └─2999 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf

janv. 07 10:31:05 raspberrypi systemd[1]: Starting Mosquitto MQTT Broker...
janv. 07 10:31:05 raspberrypi systemd[1]: Started Mosquitto MQTT Broker.
[lines 1-18/18 (END)]
```

Image 1 : Service mosquitto actif

```

lidian@raspberrypi:~ $ mosquitto_sub -h localhost -t "testtopic" -v
testtopic Testing

```



```

lidian@raspberrypi:~ $ mosquitto_pub -h localhost -t "testtopic" -m "Testing"
lidian@raspberrypi:~ $

```

Image 2 : Teste de publication et souscription sur le broker MQTT

2.Création du script Python MQTT

Une fois le serveur Mosquitto opérationnel, nous avons développé un script Python permettant de gérer les abonnements et publications MQTT sur le Raspberry Pi. Ce script établit le lien entre les messages MQTT reçus et les actions correspondantes dans Minecraft.

Cela a permis d'associer chaque message publié par le microcontrôleur ESP8266 à un super-pouvoir ou malus spécifique. Ce processus garantit une communication fluide entre le système IoT et l'univers Minecraft.

```

lidian@raspberrypi:~ $ mosquitto_pub -h localhost -t "testtopic" -m "Testing"

```



```

lidian@raspberrypi:~ $ python3 test_mqtt_broker.py
Connected with result code Success
testtopic b'Testing'

```

Image 3 : Teste du fonctionnement du script python

```

lidian@raspberrypi:~ $ mosquitto_pub -h localhost -t "gant1/index" -m "appuie index"

```



```

lidian@raspberrypi:~ $ python3 mqtt_pt.py
Connected with result code Success
Message reçu : gant1/index
Traceback (most recent call last):

```

Image 4 : Teste association du script avec superpouvoirs

Script pour l'appel des pouvoirs – `appel_pouvoirs.py`

L'une des premières étapes du projet a consisté à ajouter différents super-pouvoirs et malus afin d'enrichir l'interaction avec Minecraft. Pour chaque super-pouvoir ou malus, une action spécifique a été définie. Par exemple, un super-pouvoir de lave a été créé en utilisant un bloc spécifique de Minecraft (bloc 11 pour la lave), permettant de générer un mur de lave autour de l'adversaire. Pour éviter une utilisation trop fréquente des malus, on a ajouté un délai de 10 secondes avant de relancer le malus.



Image 5 : Message minecraft délai 10 secondes pour réactiver malus

Développement des pouvoirs

Les pouvoirs ont été testés de manière itérative, en commençant par des tests simples en ligne de commande via l'outil Mosquitto. Nous avons utilisé des commandes `mosquitto_pub` pour publier des messages simulés. Cela nous a permis de nous assurer que chaque pouvoir fonctionnait comme prévu avant de passer à l'intégration matérielle.

Voici un aperçu des super-pouvoirs et malus développés :

1. **Creuser un trou sous le joueur (index)** : Ce pouvoir permet au joueur de creuser un trou sous lui, en activant la commande `mc.setBlock(x, y, z, 0)` pour créer un espace vide sous ses pieds, et ainsi trouver le diamant si c'est en dessous.



Image 6 : screenshot minecraft creuser un trou

2. **Créer un mur de lave autour de l'adversaire (majeur)** : En appuyant sur le bouton "majeur", un malus est appliqué à l'adversaire. Un mur de lave est généré autour de lui, ce qui complique sa survie. Ce pouvoir utilise une boucle pour remplir un carré autour de l'adversaire avec des blocs de lave. `mc.setBlock(x, adversaire_pos.y, z, 11)`



Image 7 : Activation malus lave sur minecraft

3. **Créer un ascenseur d'eau sous le joueur (annulaire)** : Ce pouvoir permet de faire remonter le joueur à la surface en plaçant des blocs d'eau sous lui, créant ainsi un ascenseur d'eau naturel. `mc.setBlock(pos.x, pos.y + 1, pos.z, 8)`



Image 8: Activation superpouvoir ascenseur d'eau

4. **Créer un mur de TNT autour de l'adversaire (auriculaire)** : Lorsque le bouton "auriculaire" est pressé, un malus encore plus dangereux est appliqué sous forme d'un mur de TNT entourant l'adversaire.

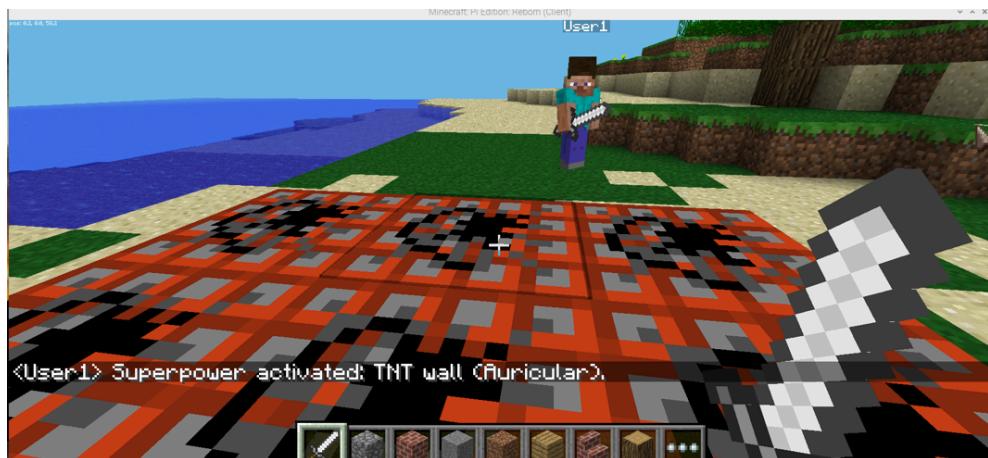


Image 9 : screenshot activation malus TNT sur minecraft

5. **Méga pouvoir (Gagnant)** : Une fois un joueur déclaré gagnant, un méga pouvoir est activé, et un anneau de glowstone est placé autour de lui pour symboliser sa victoire.



Image 10 : victoire user 1 activation anneau de glowstone autour de lui



Image 11: vue de gagnant : pour lui l'anneau est invisible

Mise en place du ESP8266

Après avoir testé les super-pouvoirs à l'aide de commandes **mosquitto_pub**, nous avons intégré la communication MQTT en utilisant le microcontrôleur ESP8266. Ce dernier a été configuré via l'IDE Arduino pour publier des messages MQTT à chaque pression de bouton.

- **Configuration initiale**

La première étape consistait à configurer l'IDE Arduino pour qu'il reconnaisse la carte ESP8266. Cela a été fait en ajoutant l'URL de gestion des cartes ESP8266 dans les Préférences de l'IDE Arduino. Ensuite, nous avons installé le support pour la carte *Adafruit HUZZAH ESP8266* via le gestionnaire de cartes.

- **Test de la connexion Wi-Fi**

Un programme simple a été utilisé pour vérifier la connexion Wi-Fi. Ce test affichait dans le moniteur série l'état de la connexion ainsi que l'adresse IP attribuée. Cette étape a confirmé que la carte pouvait se connecter au réseau local avec succès.

```
wifi $ 
#include <ESP8266WiFi.h>

// Configuration Wi-Fi
const char* ssid = "SAE304-LDA";
const char* password = "lidian20";

WiFiClient espClient;
void setup() {
    Serial.begin(115200);
    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(1000);
        Serial.println("Connexion au WiFi...");
    }
    Serial.print("Connecté au WiFi ");
    Serial.println(WiFi.SSID());
    Serial.print("IP: ");
    Serial.println(WiFi.localIP());
}

void loop() {
```

Connexion au WiFi...
Connexion au WiFi...
Connexion au WiFi...
Connecté au WiFi SAE304-LDA
IP: 192.168.1.142

Autoscroll Show timestamp

Image 12: Test de connexion wifi ESP

Création du code de publication MQTT

Une fois la connexion Wi-Fi validée, nous avons développé un programme en utilisant la bibliothèque **PubSubClient**. Ce programme permet à l'ESP8266 de publier des messages MQTT. Comme les boutons poussoirs n'étaient pas encore disponibles, nous avons testé la publication des topics lorsque l'ESP réussissait à se connecter au Broker.

- **Configuration du Broker MQTT**

Par défaut, le MQTT n'accepte pas les connexions d'utilisateurs anonymes. Pour contourner ce problème, nous avons ajouté la ligne `allow_anonymous true` au fichier de configuration du Broker MQTT situé à `/etc/mosquitto/mosquitto.conf`. Après cette modification, l'ESP8266 a pu se connecter avec succès au broker MQTT.

7

rl1ff|rlf| l b|fffffrfbf bfnnnlnnnffff bpfflrl Connexion au WiFi... Connexion au WiFi... Connexion au WiFi... Connecté au WiFi Connexion au MQTT Broker comme esp8266-client-B Connecté au MQTT broker Gant1 BP_Index pressé Gant1 BP_majeur pressé Gant1 BP_annuaire pressé Gant1 BP_auriculaire pressé

liandi@raspberrypi:~ \$ python3 mqtt.py Connecté avec le code de résultat Success Message reçu Topic: gant1/index superpouvoir Creuser un trou Message reçu Topic: gant1/majeur Malus Muro de Lava Message reçu Topic: gant1/annulaire superpouvoir ANCESTEUR D'EAU Message reçu Topic: gant1/auriculaire Malus Muro de TNT

Image 13:Publication de l'ESP au Broker et Subscription du Raspberry Pi

Tests finaux

Pour valider l'ensemble du système, nous avons vérifié si l'ESP8266 pouvait se connecter au broker MQTT et publier des messages correctement. Ces tests ont confirmé une communication fluide entre le microcontrôleur et le serveur Mosquitto, permettant ainsi l'interaction directe avec Minecraft et l'activation des pouvoirs ou malus.