

Projet IOT

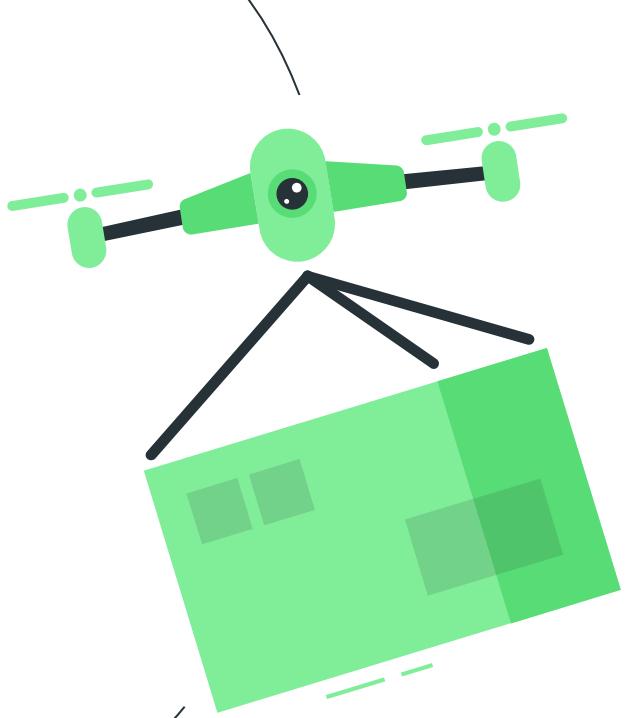
Robot

M1 MIAGE - Promo 2020-2021

Steven BOUCHE
Armand PREVOT
Pierre GRISERI
Lina BELKFARFA
Corentin GARNIER
Margaux PARENT

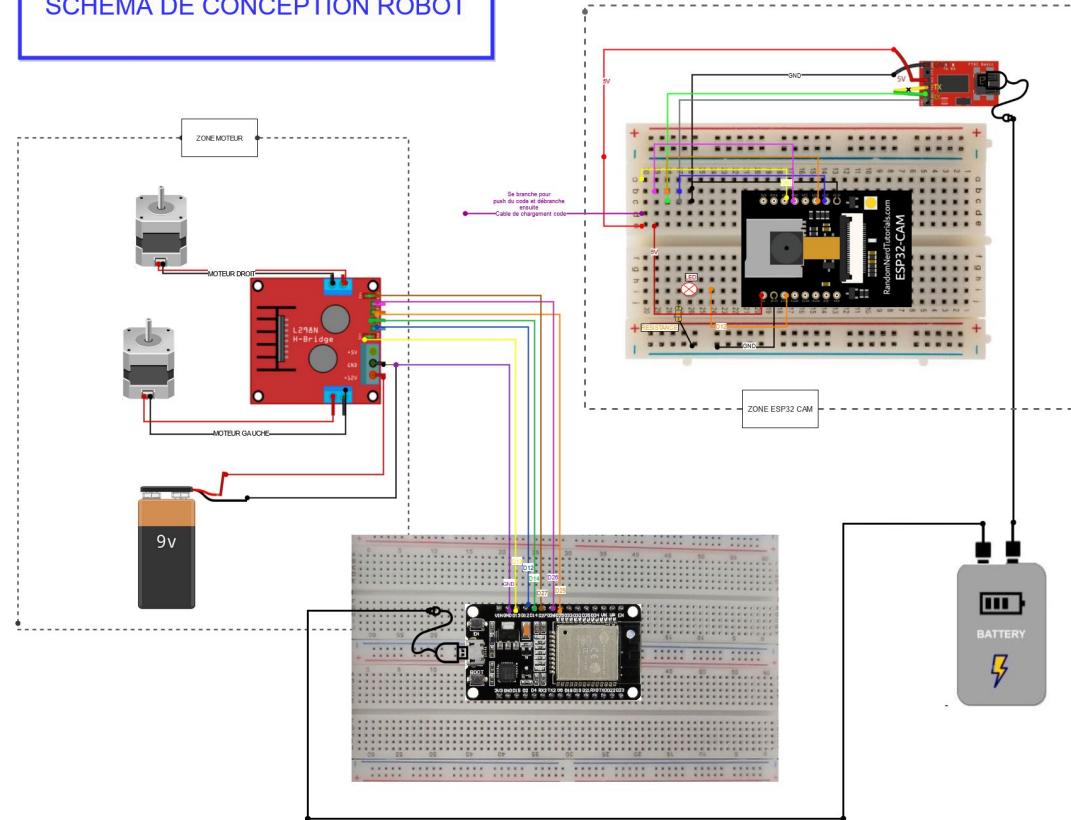


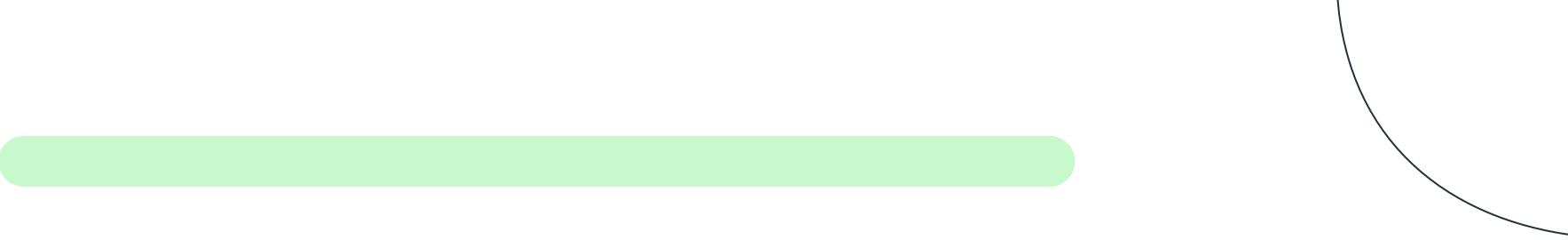
Introduction



1. Conception du robot

SCHEMA DE CONCEPTION ROBOT

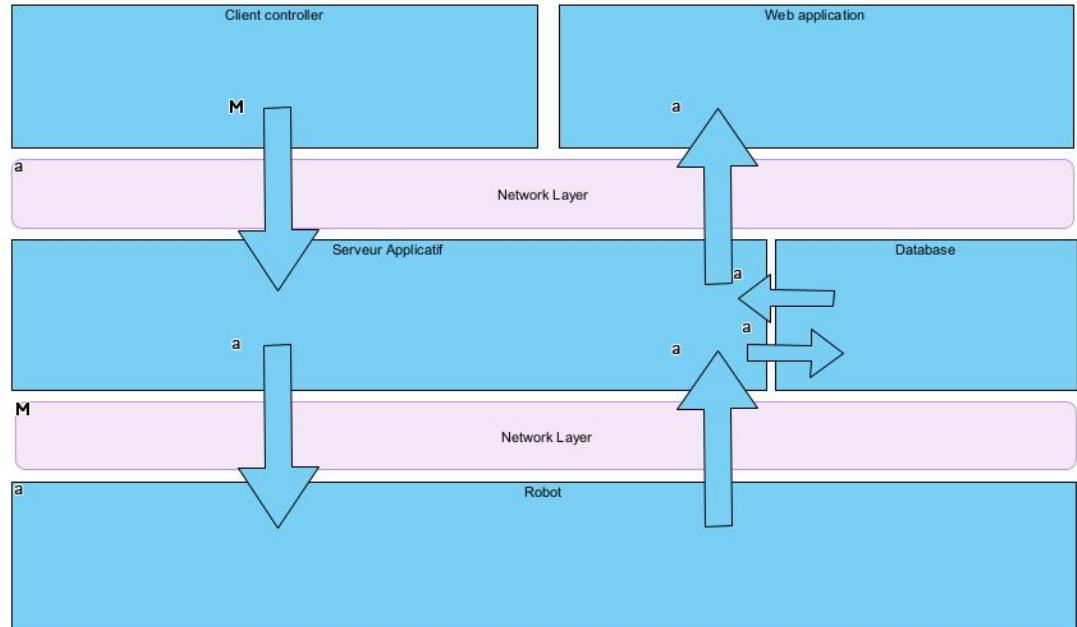
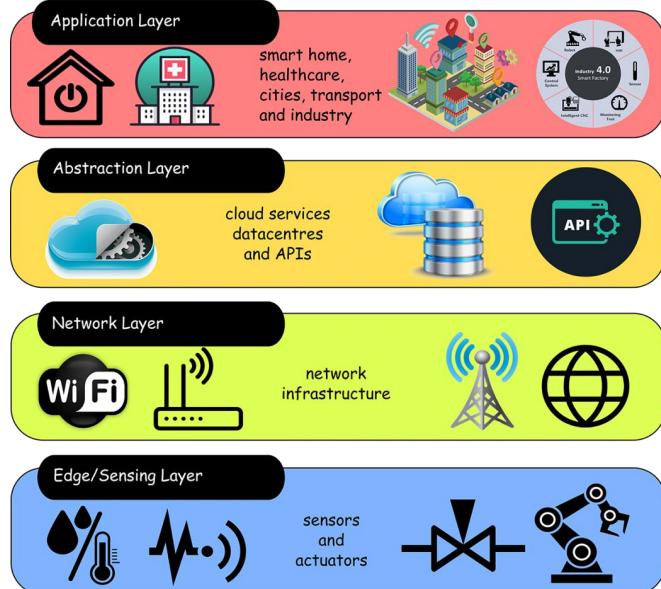




DEMO

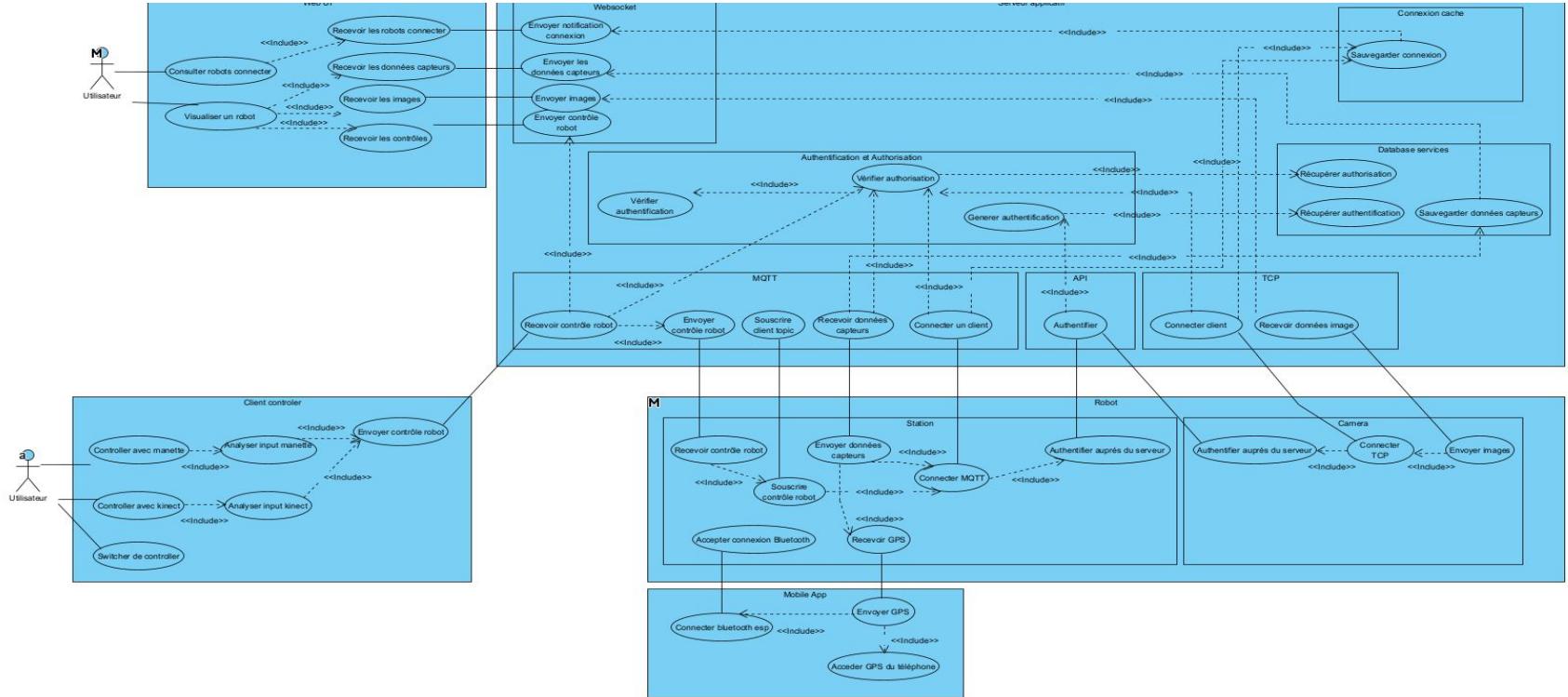
2. Conception/Architecture globale

2.1- Architecture



2. Conception/Architecture globale

2.2- Conception



3. Robot

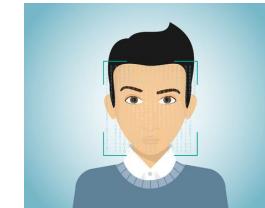
3.1 Présentation

Plusieurs fonctionnalités :

- Capteurs de luminosité et température.
- Valeurs sont consultables sur une interface web.
- Contrôle du robot : Grâce à la "kinect" et manette de jeu.
- Le robot détecte une personne grâce à la reconnaissance d'image.
- Le robot est localisable via une balise GPS
- Visualisation en temps réel via la caméra embarquée du robot.

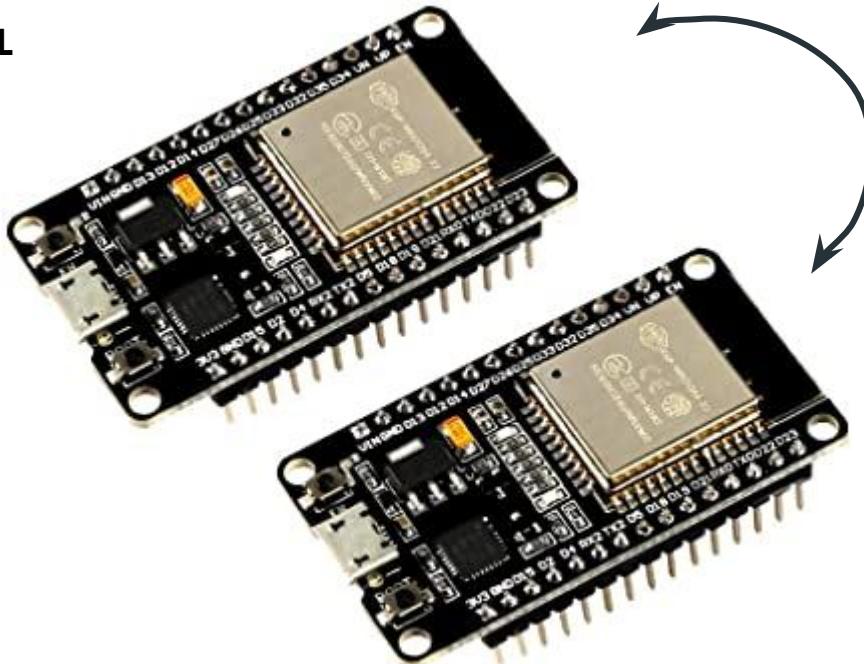
Composants matériels :

- Microcontrôleur
- Deux moteurs pour les déplacements avec roue libre
- Capteurs de luminosité et de température
- Balise GPS
- Caméra embarquée



3. Robot

1. PRINCIPAL



2. CAMERA

3. Robot

3.2 Environnement de développement

Utilisation de PlatformIO.

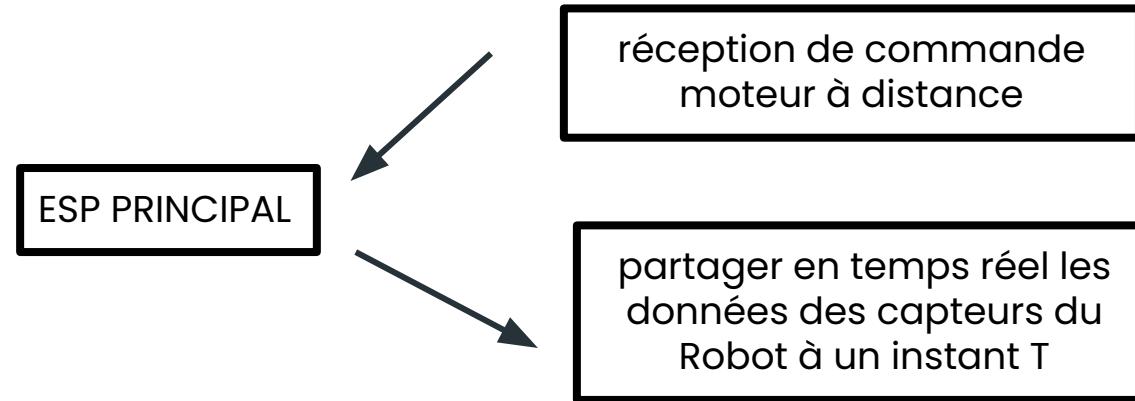


Architecture d'un projet

```
▼ SolutionESPCam
  > .vscode
  > include
  > lib
  > src
  > test
  ◇ .gitignore
  🐛 platformio.ini
```

3. Robot

3.3 ESP Principal



Protocole de communication réseau



Protocole de transmission de données



Protocole de communication (GPS)



3. Robot

3.3 ESP Principal

3.3.1 Les modules

Le programme est divisé en plusieurs modules utilisables par le programme principal pour mieux diviser la responsabilité des différentes tâches qu'il doit réaliser.

- Gestion parallèle des tâches
- Gestion de la connexion WiFi
- Gestion MQTT
- Gestion Bluetooth
- Gestion des moteurs
- Gestion des capteurs
- Représentation des données utilisés

3. Robot

3.3.1.1- Traitement parallèle

L'ESP est composé de 2 Cœurs.

Il est alors possible de répartir certaines tâches entre les deux cœur avec une certaine priorité. Nous avons donc créé une abstraction supplémentaire comme module pour nous permettre d'écrire des tâches qui s'exécutent simultanément.

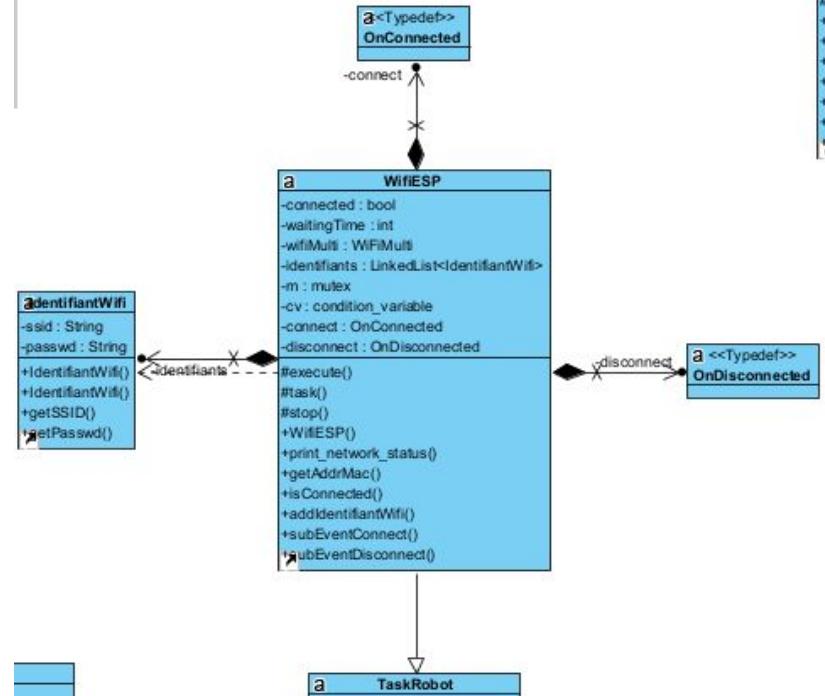
Toutefois, si plusieurs tâches sont exécutées sur le même cœur, alors le niveau de priorité de la tâche déterminera quelle tâche aura la priorité d'exécution.

TaskRobot
-core : int
-handler : TaskHandle_t
-name : string
-stackDepth : int
-taskFunction : TaskFunction_t
+TaskRobot(name : string, stackDepth : int, core : int)
+executeTask()
+stopTask()
#execute()
#stop()
#task()
-onTask()

3. Robot

3.3.1.2- WIFI

- Interaction avec le Cloud
- Utilisation de librairie WifiMulti
- Hérite de TaskRobot
- Est exécuté sur le cœur 1
- Gère la connexion à des AP
- Permet la réaction du programme principal avec des events



3. Robot

3.3.1.2- WIFI

```
void WiFiESP::execute(){

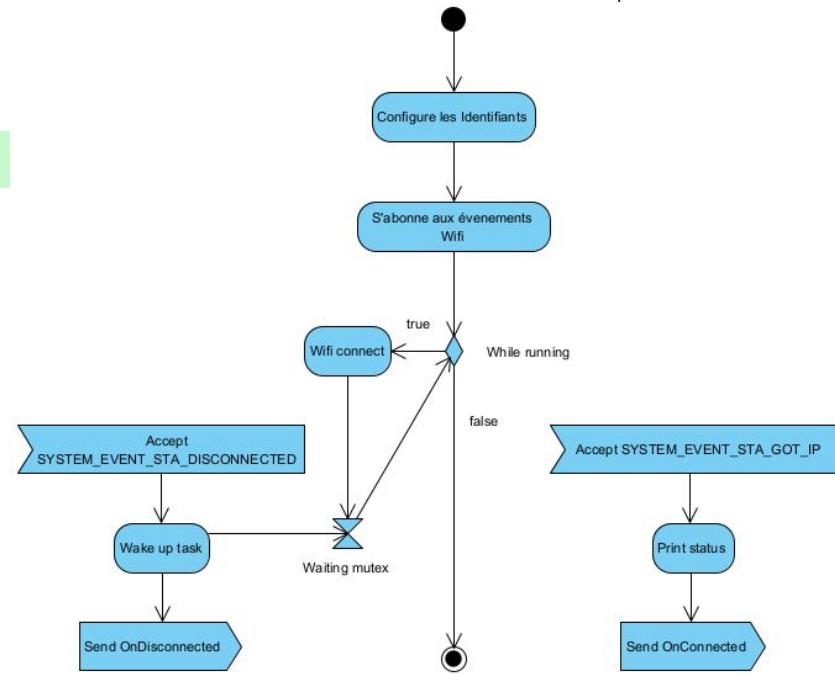
    WiFi.mode(WIFI_STA);
    //esp_wifi_set_ps(WIFI_PS_NONE);

    //for all identifiants, add ssid and password to multi wifi
    for(int i = 0; i < identifiants.size(); i++){
        IdentifiantWifi id = identifiants.get(i);
        wifiMulti.addAP(id.getSSID().c_str(), id.getPassword().c_str());
    }

    //when disconnected wake lock for retry connection and callback disconnected
    auto lambdaDisconnected = [this] (WiFiEvent_t event, WiFiEventInfo_t info) {
        this->connected = false;
        cv.notify_all();
        if(this->disconnect)
            this->disconnect();
    };

    //when have an address ip callback connect
    auto lambdaGotIP = [this] (WiFiEvent_t event, WiFiEventInfo_t info) {
        this->print_network_status();
        this->connected = true;
        if(this->connect)
            this->connect();
    };

    //subscribe event
    WiFi.onEvent(lambdaDisconnected, SYSTEM_EVENT_STA_DISCONNECTED);
    WiFi.onEvent(lambdaConnected, SYSTEM_EVENT_STA_CONNECTED);
    WiFi.onEvent(lambdaGotIP, SYSTEM_EVENT_STA_GOT_IP);
}
```



```
while(this->running){

    Serial.println("Try to connect to an Wifi network.");

    //try connection while not connected
    while(this->wifiMulti.run() != WL_CONNECTED){
        vTaskDelay(this->waitingTime);
    }

    //wait status connected
    while(WiFi.status() != WL_CONNECTED){
        vTaskDelay(1000); // ms
        Serial.print(".");
    }

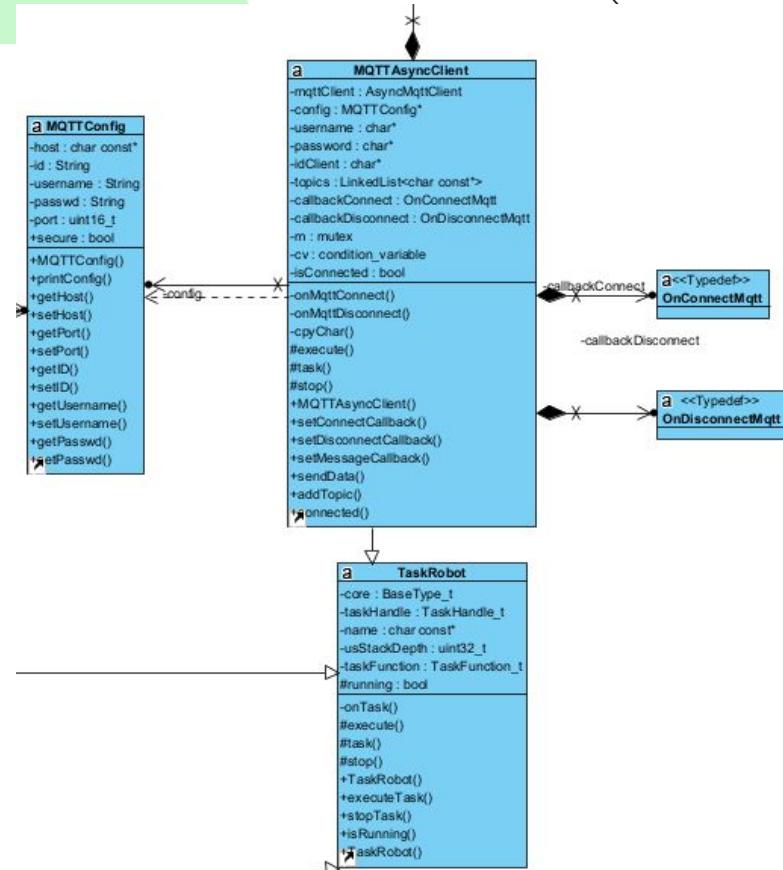
    //thread to sleep waiting deconnection for retry connection
    std::unique_lock<std::mutex> verrou(this->m);
    this->cv.wait(verrou);
}

Serial.println("Stop task wifi.");
```

3. Robot

3.3.1.3- MQTT

- Protocole d'échange de donnée basé sur TCP
- Hérite de TaskRobot
- Est exécuté sur le cœur 0
- Asynchrone
- Permet la réaction du programme principal à des events



3. Robot

3.3.1.3- MQTT

```
void MQTTAsyncClient::execute(){  
  
    mqttClient.setKeepAlive(30);  
    mqttClient.setCleanSession(true);  
  
    if(this->username) free(this->username);  
    if(this->password) free(this->password);  
    if(this->idClient) free(this->idClient);  
  
    this->username = this->cpyChar(this->config->getUsername());  
    this->password = this->cpyChar(this->config->getPasswd());  
    this->idClient = this->cpyChar(this->config->getID());  
  
    mqttClient.setCredentials(this->username, this->password); //const char* username, const  
    mqttClient.setClientId(this->idClient);  
    mqttClient.setServer(this->config->getHost(), this->config->getPort());  
  
#if ASYNC_TCP_SSL_ENABLED  
    mqttClient.setSecure(this->config->secure);  
    if (this->config->secure) {  
        mqttClient.addServerFingerprint((const uint8_t[])MQTT_SERVER_FINGERPRINT);  
    }  
#endif  
}
```

```
void MQTTAsyncClient::task(){  
  
    while(this->running){  
  
        Serial.println("Try to connect to Mqtt broker.");  
  
        mqttClient.connect();  
  
        std::unique_lock<std::mutex> verrou{this->m};  
        this->cv.wait(verrou);  
  
        vTaskDelay(2000);  
    }  
    Serial.println("Stop task mqtt.");  
}
```

```
void MQTTAsyncClient::onMqttConnect(bool sessionPresent){  
    this->isConnected = true;  
    Serial.println("Connected to Mqtt broker.");  
}  
  
void MQTTAsyncClient::onMqttDisconnect(AsyncMqttClientDisconnectReason reason){  
    this->isConnected = false;  
    Serial.println("Disconnected to Mqtt broker.");  
    this->cv.notify_all();  
}
```

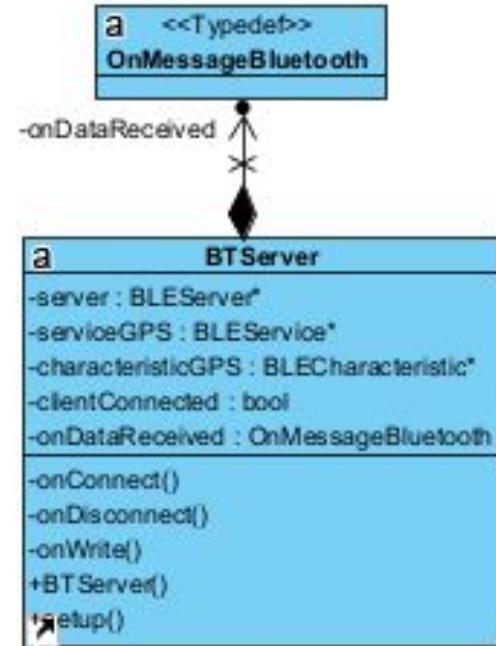
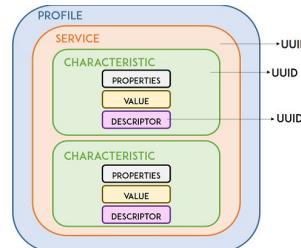
```
void setMessageCallback(AsyncMqttClientInternals::OnMessageUserCallback callback){  
    this->mqttClient.onMessage(callback);  
};
```

3. Robot

3.3.1.4- BLUETOOTH

L'ESP agit comme serveur Bluetooth afin que le téléphone puisse se connecter et écrire les coordonnées GPS. Pour cela nous avons écrit un module Bluetooth utilisant le Bluetooth Low Energy.

GATT définit la manière dont deux appareils BLE envoient et reçoivent des messages standard.



3. Robot

3.3.1.4- BLUETOOTH

```
BLEDevice::init("ESP32-BLE");

this->server = BLEDevice::createServer();

this->server->setCallbacks(this);

this->serviceGPS = this->server->createService(SERVICE_UUID);
```

```
this->characteristicGPS = this->serviceGPS->createCharacteristic(
    CHARACTERISTIC_UUID,
    BLECharacteristic::PROPERTY_WRITE
);

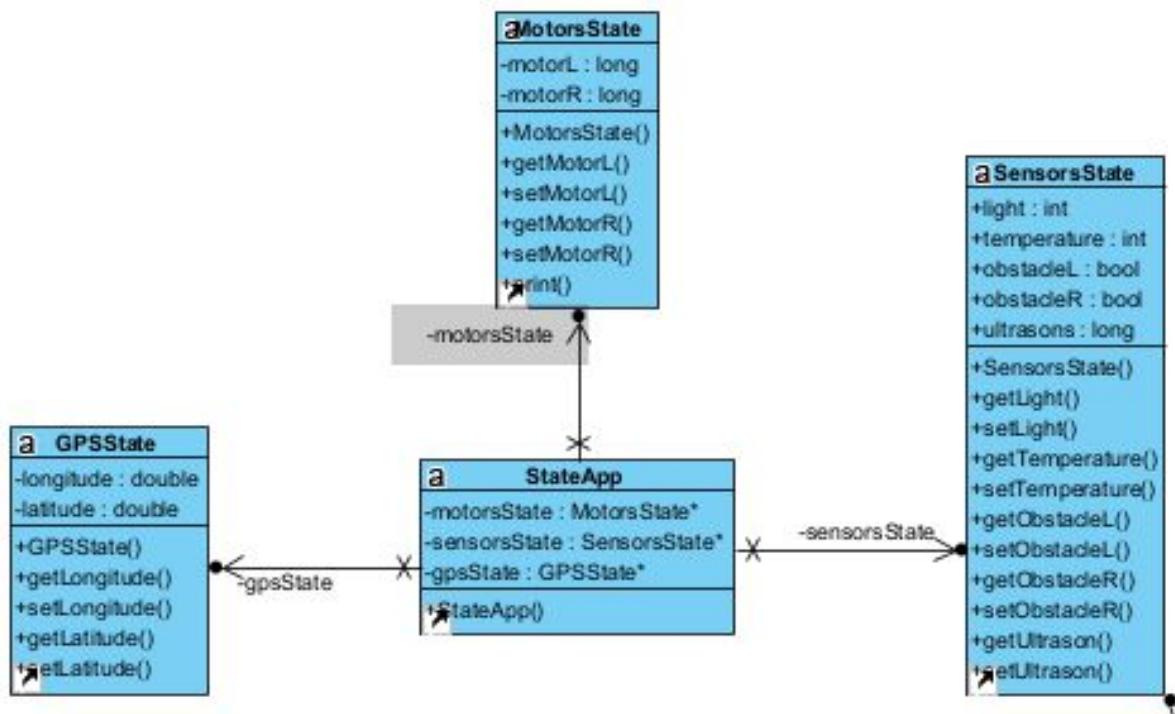
this->characteristicGPS->setCallbacks(this);
```

```
this->serviceGPS->start();

// BLEAdvertising *pAdvertising = pServer->getAdvertising(); // this still is working for backward compatibility
BLEAdvertising *pAdvertising = BLEDevice::getAdvertising();
pAdvertising->addServiceUUID(SERVICE_UUID);
pAdvertising->setScanResponse(true);
pAdvertising->setMinPreferred(0x06); // functions that help with iPhone connections issue
pAdvertising->setMinPreferred(0x12);
BLEDevice::startAdvertising();
```

3. Robot

3.3.1.5 – Représentation des données utilisés



Puissance moteurs

Les ETATS

Valeurs robot

BROKER MQTT

3. Robot

3.3.1.6 – Les moteurs

Au démarrage, il configure les différents PIN de sorties de l'ESP vers les moteurs :

- IN1 et IN2 : broche GPIO OUTPUT du moteur gauche
 - IN3 et IN4 : broche GPIO OUTPUT du moteur droit
 - ENA : broche PWM OUTPUT du moteur gauche
 - ENB : broche PWM OUTPUT du moteur droit
- Utilise les outputs GPIO pour le sens de rotation
→ Utilise PWM pour la vitesse de rotation.

```
void MotorHandler::init(){  
  
    pinMode(this->IN1, OUTPUT);  
    pinMode(this->IN2, OUTPUT);  
    pinMode(this->ENA, OUTPUT);  
  
    if(this->dualMotor)  
    {  
        pinMode(this->IN3, OUTPUT);  
        pinMode(this->IN4, OUTPUT);  
        pinMode(this->ENB, OUTPUT);  
    }  
}
```

MotorHandler
-IN1 : int
-IN2 : int
-ENA : int
-IN3 : int
-IN4 : int
-ENB : int
-dualMotor : bool
-mapVitesse()
+MotorHandler()
+MotorHandler()
+init()
+rotateMotor()
+rotateMotors()
+breakMotor()
+breakMotors()

3. Robot

3.3.1.6 – Les moteurs suite

```
✓ void MotorHandler::rotateMotors(long valueMotor1, long valueMotor2){  
    rotateMotor(MOTOR1,valueMotor1);  
    rotateMotor(MOTOR2,valueMotor2);  
}
```

	IN1	IN2
VAL > 0	1	0
VAL < 0	0	1
VAL = 0	0	0

```
void MotorHandler::rotateMotor(motor_num_t motor, long value){  
  
    int in1, in2, en;  
  
    if(motor==MOTOR1){  
        in1 = this->IN1;  
        in2 = this->IN2;  
        en = this->ENA;  
    } else if(motor==MOTOR2 && this->dualMotor) {  
        in1 = this->IN3;  
        in2 = this->IN4;  
        en = this->ENB;  
    }  
    else return;  
  
    uint32_t level1 = value > 0 ? HIGH : LOW;  
    uint32_t level2 = value < 0 ? HIGH : LOW;  
  
    value = abs(value);  
  
    value = value < MIN_SPEED ? MIN_SPEED : value > MAX_SPEED ? MAX_SPEED : value;  
  
    digitalWrite(in1, level1);  
    digitalWrite(in2, level2);  
    digitalWrite(en, LOW);  
    analogWrite(en, mapVitesse(value));  
}  
  
}
```

```
#define MIN_SPEED_PWM 200  
#define MAX_SPEED_PWM 255  
#define MIN_SPEED 0  
#define MAX_SPEED 100  
  
typedef enum {  
    MOTOR1 = 1,  
    MOTOR2 = 2  
} motor_num_t;
```

```
int MotorHandler::mapVitesse(long v){  
    return map(v, MIN_SPEED, MAX_SPEED, MIN_SPEED_PWM, MAX_SPEED_PWM);  
}
```

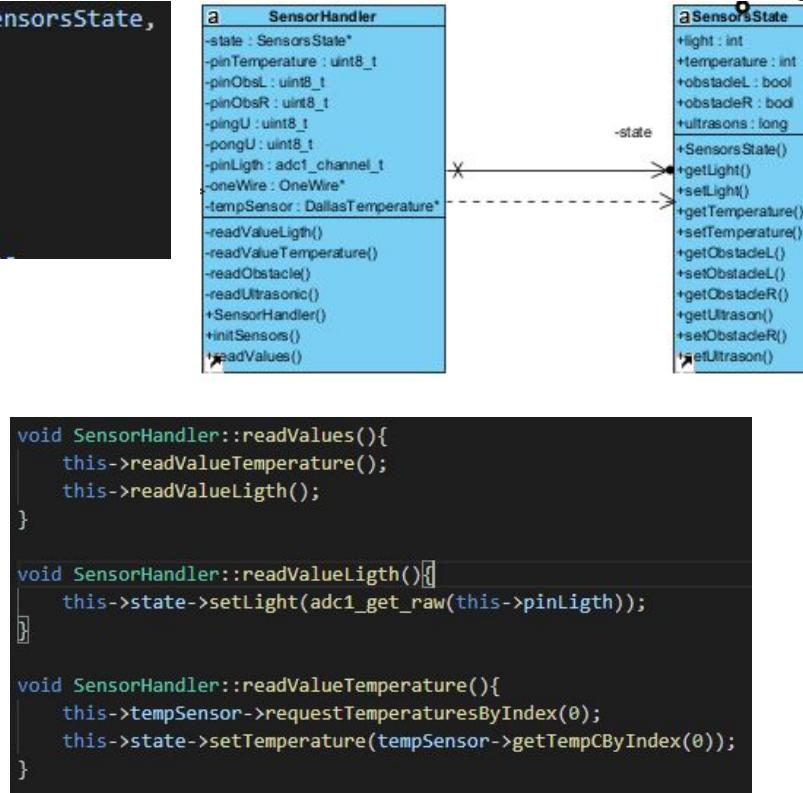
3. Robot

3.3.1.7 – Les capteurs

```
void SensorHandler::initSensors(){  
  
    //init Ligth  
    adc1_config_width(ADC_WIDTH_BIT_12); //value on 12 bits  
    adc1_config_channel_atten(this->pinLigth, ADC_ATTEN_DB_0);  
  
    //init Temperature  
    this->oneWire = new OneWire(this->pinTemperature);  
    this->tempSensor = new DallasTemperature(this->oneWire);  
  
    //Obstacle  
    //pinMode(this->pinObsL, INPUT);  
    //pinMode(this->pinObsR, INPUT);  
  
    //Ultrasonic  
    //pinMode(pingU, OUTPUT);  
    //pinMode(pongU, INPUT);  
}
```

```
SensorHandler sensors(state.sensorsState,  
                      PIN_SENSOR_TEMPERATURE,  
                      PIN_SENSOR_LIGTH,  
                      OBSTACLE_L,  
                      OBSTACLE_R,  
                      ULTRASONIC_PING,  
                      ULTRASONIC_PONG);
```

```
#define PIN_SENSOR_LIGTH (adc1_channel_t) ADC1_CHANNEL_5 //D34 INPUT  
#define PIN_SENSOR_TEMPERATURE (uint8_t) 35 //D35 INPUT
```



3. Robot

3.3 ESP Principal

3.3.2 Le programme principal - Configuration

```
#pragma region IdDefinition
const String idEquipment = "Esp32Robot";
#pragma endregion IdDefinition
```

```
IdentifiantWifi ids[] = {
    IdentifiantWifi("Bbox-4DD70ADE", "551F54E2D72A27CA1EA44567F149E1"),
    IdentifiantWifi("iPhone de Steven", "iobwgn7obkf91")
};
```

```
const char* HostMqtt = "62.35.150.64";
const char* DataTopicName = "IOT/Data";
const String ControllerTopicNameSub = "IOT/Controller/" + idEquipment;
```

```
const String urlAuth = "http://62.35.150.64:8000/api/AuthEquipment/auth";
```

JSON

```
{}
  └─ IdEquipment : "Esp32Robot"
  └─ Password : "25Zjqgr8AQxxZsyz"
  └─ TypeEquipment : "Robot"
  └─ Role : "Station"
```

```
#if ASYNC_TCP_SSL_ENABLED
bool mqttSecure = true;
const uint16_t PortMqtt = 8884;
#else
bool mqttSecure = false;
const uint16_t PortMqtt = 1883;
#endif
```

3. Robot

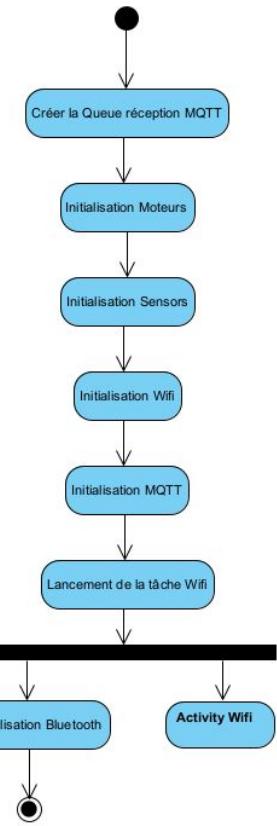
3.3 ESP Principal

3.3.2 Le programme principal - Initialisation

- Moteurs et des capteurs pour attacher les différents PIN aux différentes broches de l'ESP.
- Wifi pour ajouter les identifiants multi AP ainsi que s'abonné aux différents events.
- MQTT pour définir les abonnements aux différents topics et s'abonner aux events.
- Bluetooth pour lancer le service et s'abonner à la réception de données.

```
/**  
 * Init WiFi. Push all identifiants wifi for multi AP.  
 * Subscribe events WiFi whith different callbacks and execute task WiFi.  
 */  
void initWiFi(){  
  
    Serial.println("Init WiFi connection.");  
  
    int sizeIds = *(&ids + 1) - ids;  
    for(int i = 0; i < sizeIds; i++){  
        wifi.addIdentifiantWifi(ids[i]);  
    }  
  
    wifi.subEventConnect(onConnectWifi);  
    wifi.subEventDisconnect(onDisconnectWifi);  
}
```

```
/**  
 * Init MQTT Client.  
 * Create actions for subscribe topics and execute a callback associate with topic.  
 */  
void initMQTT(){  
  
    Serial.println("Init MQTT connection.");  
  
    MQTTAction actionToto = MQTTAction(ControllerTopicNameSub, receivedActionController);  
    actions.addAction(actionToto);  
    mqttAsyncClient.setConnectCallback(onConnectMqtt);  
    mqttAsyncClient.setDisconnectCallback(onDisconnectMqtt);  
    mqttAsyncClient.setMessageCallback(mqtt_pubcallback_async);  
    mqttAsyncClient.addTopic(ControllerTopicNameSub.c_str());  
}
```

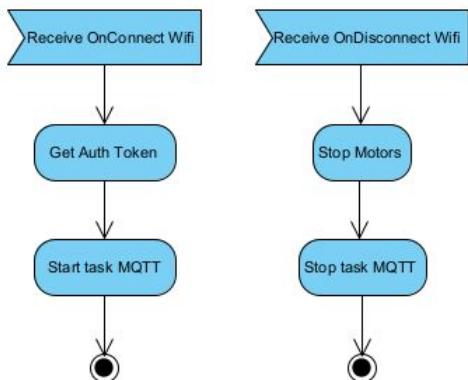


```
/**  
 * Init Bluetooth Server with callback on receive message.  
 */  
void initBLE(){  
    Serial.println("Init Bluetooth server.");  
    btServer.setup(receivedActionBluetooth);  
    Serial.println("End init Bluetooth server.");  
}
```

3. Robot

3.3 ESP Principal

3.3.2 Le programme principal - Réaction aux événements Wifi



```
/** * Callback WiFi connected. Execute task MQTT Client when connected. */  
void onConnectWifi(){  
    Serial.println("Connected WiFi event trigger.");  
    timeClient.begin();  
    tryRefreshAuthorizationForMqtt();  
    mqttAsyncClient.executeTask();  
}  
  
/** * Callback WiFi disconnected. Stop task MQTT Client and stop motors. */  
void onDisconnectWifi(){  
    Serial.println("Disconnected WiFi event trigger.");  
    timeClient.end();  
    motor.breakMotors();  
    mqttAsyncClient.stopTask();  
}
```

```
bool authProcess(unsigned long currentTime){  
  
    Serial.println("Start authentication.");  
  
    bool result = false;  
    String payload = "";  
  
    http.begin(AuthUrl);  
  
    http.addHeader("Content-Type", "application/json");  
    int httpResponseCode = http.POST(AuthJson);  
  
    Serial.println("HTTP Response code: " + String(httpResponseCode));  
  
    if (httpResponseCode > 0) {  
        payload = http.getString();  
        result = setToken(payload,currentTime);  
    }  
  
    http.end();  
    return result;  
}
```

3. Robot

3.3 ESP Principal

3.3.2 Le programme principal – Envoi des données au broker MQTT

```
void loop() {
    sendDataIOT();
    delay(LoopDelay);
}
```

```
void sendDataIOT(){

    //actualize data before send MQTT broker
    sensors.readValues();

    //Serialize data
    String output;
    DynamicJsonDocument doc(512);

    doc[IdEspJson] = idEquipment.c_str();
    doc[LatitudeJson] = state.gpsState->getLatitude();
    doc[LongitudeJson] = state.gpsState->getLongitude();
    doc[LighthJson] = state.sensorsState->getLight();
    doc[TemperatureJson] = state.sensorsState->getTemperature();

    serializeJson(doc, output);

    //Serial.println(output);
    //send data to broker
    mqttAsyncClient.sendData(DataTopicName, output);
}
```

3. Robot

3.3 ESP Principal

3.3.2 Le programme principal – Réception des données au broker MQTT

```
void mqtt_pubcallback_async(char* topic, char* payload, AsyncMqttClientMessageProps
size_t lenTopic = strlen(topic)+1;
DataMQTT_t * data = reinterpret_cast<DataMQTT_t*>(pvPortMalloc(sizeof(DataMQTT_t
data->topic = (char*)pvPortMalloc(lenTopic*sizeof(char));
data->message = (byte*)pvPortMalloc(len*sizeof(byte));
memcpy(data->message,payload,len);
strcpy(data->topic,topic);
xQueueSend(queue, (void*)&data, portMAX_DELAY);
}
```

```
#pragma region QueueMqtt
typedef struct DataMQTT {
    char * topic;
    byte * message;
    unsigned int length;
} DataMQTT_t;

QueueHandle_t queue;
#pragma endregion QueueMqtt
```

```
void taskHandlePubCallback(void * args{
    DataMQTT_t *data;
    while(true){
        BaseType_t t = xQueueReceive(queue, &data, portMAX_DELAY);
        if(t == pdPASS)
        {
            for(int i = 0; i < actions.size(); i++){
                if(actions.get(i).getTopic() == data->topic){
                    actions.get(i).getHandler()(const byte* data->message,data->length);
                    break;
                }
            }
            vPortFree(data);
        }
        vTaskDelete(NULL);
    }
}
```

```
/*
 * Callback when received data from MQTT broker from topic controller.
 *
 * @param message message received
 * @param length size of message
 */
void receivedActionController(const byte* message, unsigned int length){

    StaticJsonDocument<256> doc;
    DeserializationError error = deserializeJson(doc, message, length);

    if(error != DeserializationError::Ok){
        Serial.println("Error deserialisation"); Serial.println(error.c_str());
        return;
    }
    else if(doc.containsKey(MotorLJson) && doc.containsKey(MotorRJson)){
        state.motorsState->setMotorL(doc[MotorLJson]);
        state.motorsState->setMotorR(doc[MotorRJson]);
        //state.motorsState->print();
        motor.rotateMotor(MOTOR_L, state.motorsState->getMotorL());
        motor.rotateMotor(MOTOR_R, state.motorsState->getMotorR());
    } else {
        Serial.println("Error json parser");
    }
}
```

3. Robot

3.3 ESP Principal

3.3.2 Le programme principal – Réception des données GPS

```
/*
 * Callback when received data from Bluetooth client.
 *
 * @param message message received
 */
void receivedActionBluetooth(String *message){

    StaticJsonDocument<256> doc;
    DeserializationError error = deserializeJson(doc, message->c_str());

    if(error != DeserializationError::Ok)
        return;

    if(doc.containsKey(LatitudeJson) && doc.containsKey(LongitudeJson)){
        state.gpsState->setLongitude(doc[LongitudeJson]);
        state.gpsState->setLatitude(doc[LatitudeJson]);
    }
}
```

3. Robot

3.4 ESP Camera

3.4.1 Les modules

Principalement composés de deux modules :

- Le module de traitement parallèle présenté plus haut pour l'ESP principal
- Un module de récupération de l'image et traitement



Protocole de transmission
de données



Protocole de communication réseau

ESP CAMERA



envoi d'images prises par la
caméra au serveur
applicatif

3. Robot

3.4 ESP Camera

3.4.1.1 Les modules - Caméra

```
typedef struct DataVideo {
    size_t size;
    const byte *buffer;
} DataVideo_t ;
```

```
CameraStream() : TaskRobot("Camera", 32768, 0) {
    queue = xQueueCreate(5, sizeof(struct DataVideo));
}
```

```
✓ void CameraStream::sendData(camera_fb_t * fb){
    DataVideo_t * data = reinterpret_cast<DataVideo*>(pvPortMalloc(sizeof(struct DataVideo)));
    data->size = fb->len;
    data->buffer = (const byte *)fb->buf;
    xQueueSend(queue, (void*) &data, portMAX_DELAY);
}
```

3. Robot

3.4 ESP Camera

3.4.2 Programme principal - Configuration

```
WifiESP wifi;
const IdentifiantWifi ids[] = {
    IdentifiantWifi("Bbox-4DD70ADE", "551F54E2D72A27CA1EA44567F149E1"),
    IdentifiantWifi("iPhone de Steven", "iobwgn7obkf91")
};
const char* SSID = "Bbox-4DD70ADE";
const char* SSID_PASSWORD = "551F54E2D72A27CA1EA44567F149E1";

String token = "";
long tokenExpiration = 0;
const uint16_t portTCP = 11000;
const char* urlTCP = "62.35.150.64";
const String urlAuth = "http://62.35.150.64:8000/api/AuthEquipment/auth";
const String jsonAuth = "{ \"IdEquipment\": \"Esp32Robot\", \"Password\": \"25Zjqgr8AQxxZsyz\", \"TypeEquipment\": \"Robot\", \"Role\": \"Camera\" }";
```

3. Robot

3.4 ESP Camera

3.4.2 Programme principal - Gestion d'envoi des images

```
xTaskCreatePinnedToCore(taskQueue, "QueueTCP", 16384, NULL, 1, NULL, 1);
```

```
void taskQueue(void * args){  
    DataVideo_t *data;  
  
    while(true){  
  
        BaseType_t t = xQueueReceive(stream.queue, &data, portMAX_DELAY);  
        if(t == pdPASS)  
        {  
            if(client.connected())  
            {  
                size_t len = data->size;  
                byte *message = (byte*) pvPortMalloc(4*sizeof(byte)+len*sizeof(byte));  
                memcpy(message, &len, 4);  
                memcpy(message + 4, data->buffer, len);  
                client.write(message, len + 4);  
                free(message);  
            } else {  
                ESP.restart();  
            }  
        }  
        //Serial.println(t);  
        vPortFree(data);  
    }  
    vTaskDelete(NULL);  
}
```

4. Serveur Applicatif

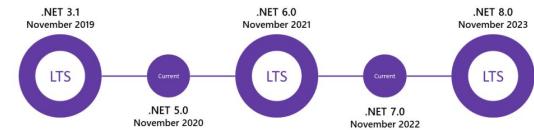
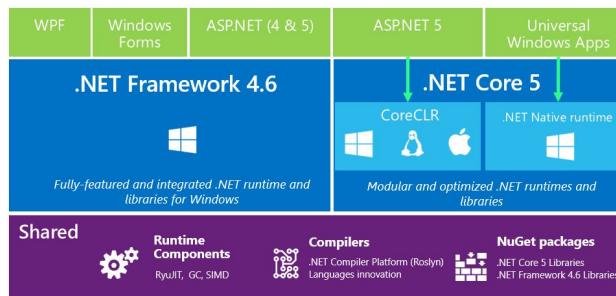
4.1 Présentation - 4.2 Environnement de développement

Le serveur d'application **est le cœur du système**

Comme environnement de développement pour le serveur applicatif, nous utilisons **Visual Studio 2019**.

Le serveur est une application ASP.NET qui cible le Framework .NET 5 de Microsoft. Ce Framework nous permet de créer des applications web, client lourd, ou encore des applications mobiles.

ASP.NET Core est une infrastructure multiplateforme, à hautes performances et Open source pour la création d'applications modernes, basées sur le Cloud et connectées à Internet.



4. Serveur Applicatif

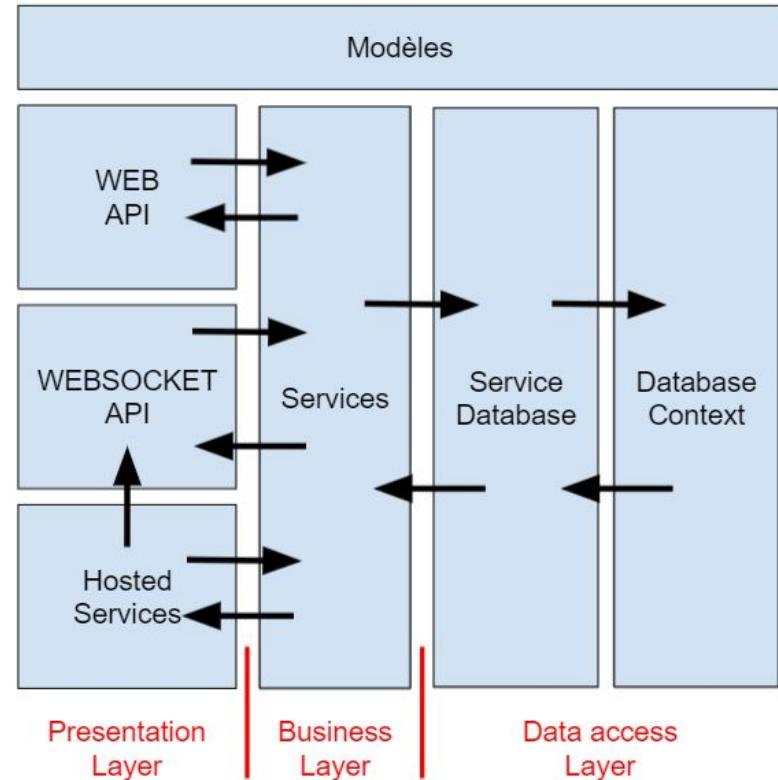
4.3 Architecture globale

Pour une meilleure maintenabilité et responsabilité des composants du serveur applicatif, nous avons mis en place une architecture 3-tiers.

Dans ce type d'architecture nous divisons l'application en différentes couches principalement :

- Présentation
- Métier
- Données

Nous permettant de mettre en place une division des responsabilités.



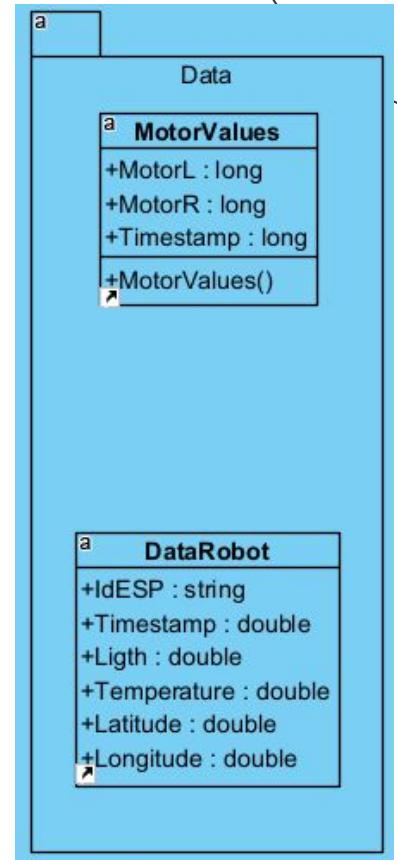
4. Serveur Applicatif

4.4 Modèles

4.4.1 Modèle de données robot

Le modèle des données robot sont des objets représentant la structure de données utilisée par le robot.

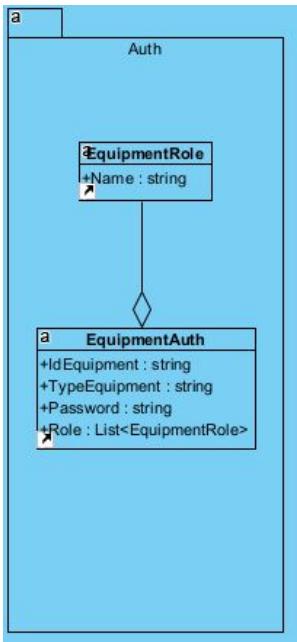
- **La classe « MotorValues »**
- **La classe « DataRobot »**



4. Serveur Applicatif

4.4 Modèles

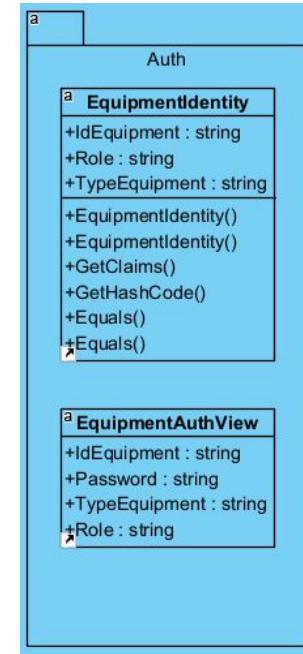
4.4.2 Modèle de données d'authentification



Modèle en base

Infos sur le robot :

- Type
- Role
- Mot de passe



Modèle interne (Controller et ESP)

EquipmentIdentity :

- Identité d'un équipement
- Stock des revendications (token)

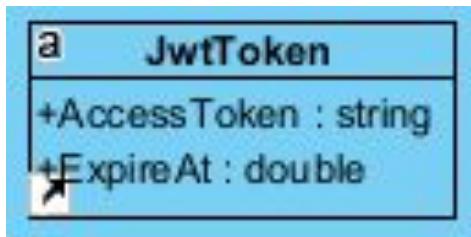
EquipmentAuthView :

- Contient les données d'identifications

4. Serveur Applicatif

4.4 Modèles

4.4.2 Modèle de données d'authentification

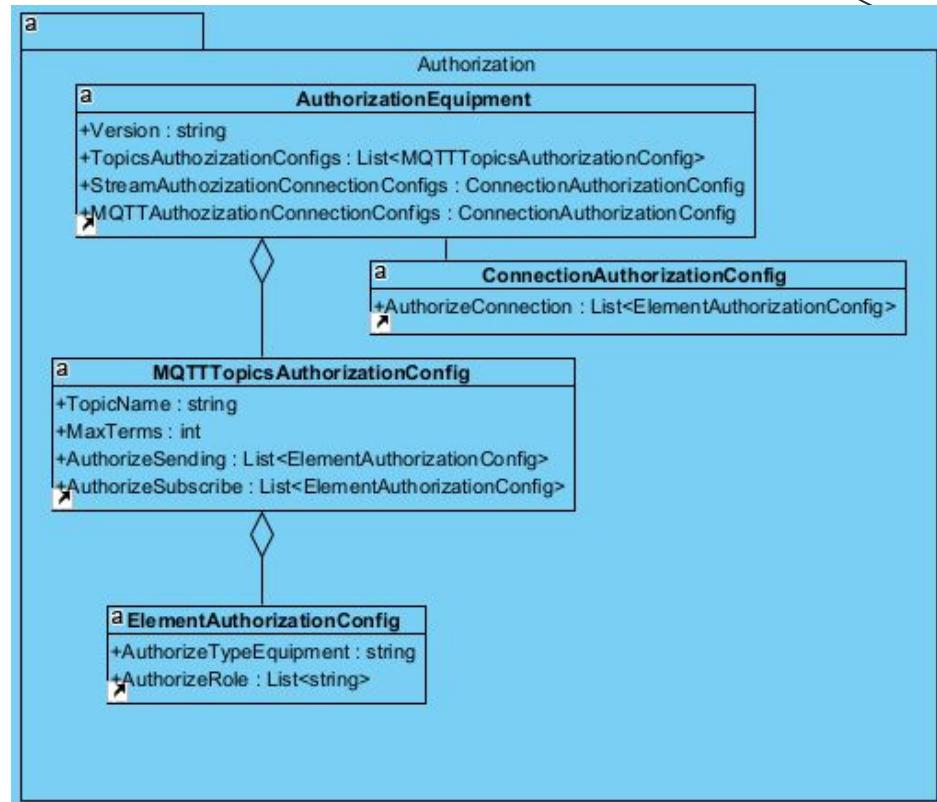


Permet de stocker un token et le temps d'expiration de celui-ci.

4. Serveur Applicatif

4.4 Modèles

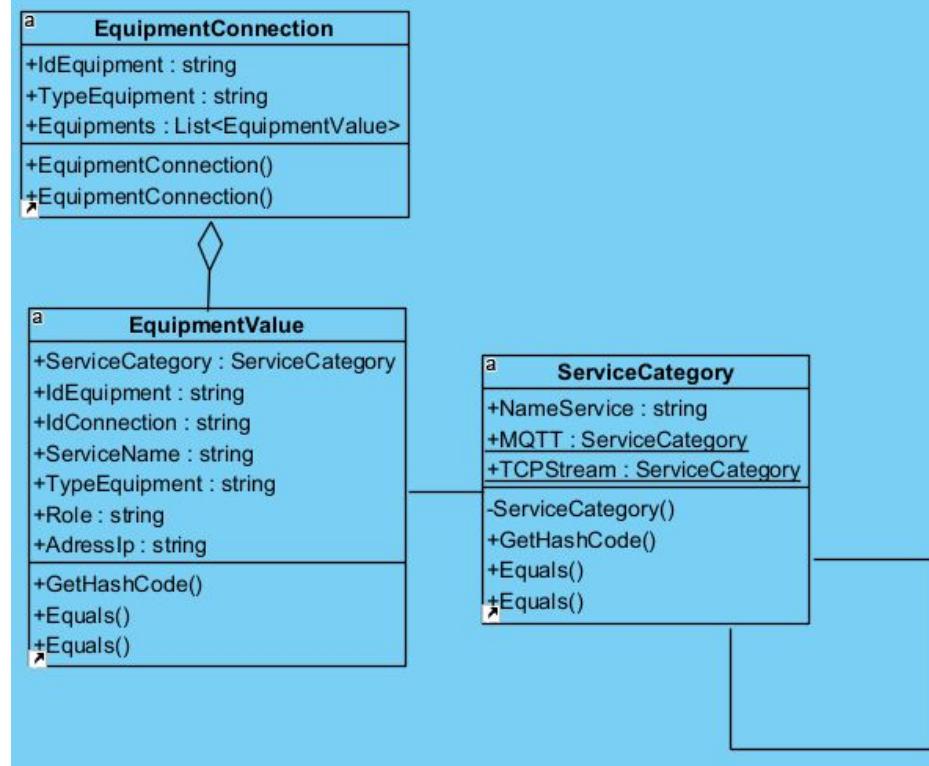
4.4.3 Modèle de données d'autorisation



4. Serveur Applicatif

4.4 Modèles

4.4.4 Modèle de données de connexion



4. Serveur Applicatif

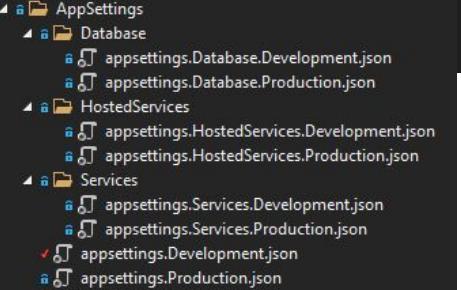
4.5 Configuration

```
0 références
public Startup(IWebHostEnvironment env)
{
    var basePath = $"{ env.ContentRootPath}/AppSettings";

    Configuration = ConfigurationBuilder(basePath, env.EnvironmentName, "");
    ConfigurationServices = ConfigurationBuilder(basePath, env.EnvironmentName, "Services");
    ConfigurationDatabase = ConfigurationBuilder(basePath, env.EnvironmentName, "Database");
    ConfigurationHostedServices = ConfigurationBuilder(basePath, env.EnvironmentName, "HostedServices");
}

4 références
private static IConfiguration ConfigurationBuilder(string basePath, string environmentName, string name)
{
    var path = string.IsNullOrEmpty(name) ? basePath : $"{basePath}/{name}";
    var file = string.IsNullOrEmpty(name) ? $"appsettings.{environmentName}.json" : $"appsettings.{name}.{environmentName}.json";

    return new ConfigurationBuilder()
        .SetBasePath(path)
        .AddJsonFile(file, optional: false, reloadOnChange: true)
        .AddEnvironmentVariables()
        .Build();
}
```



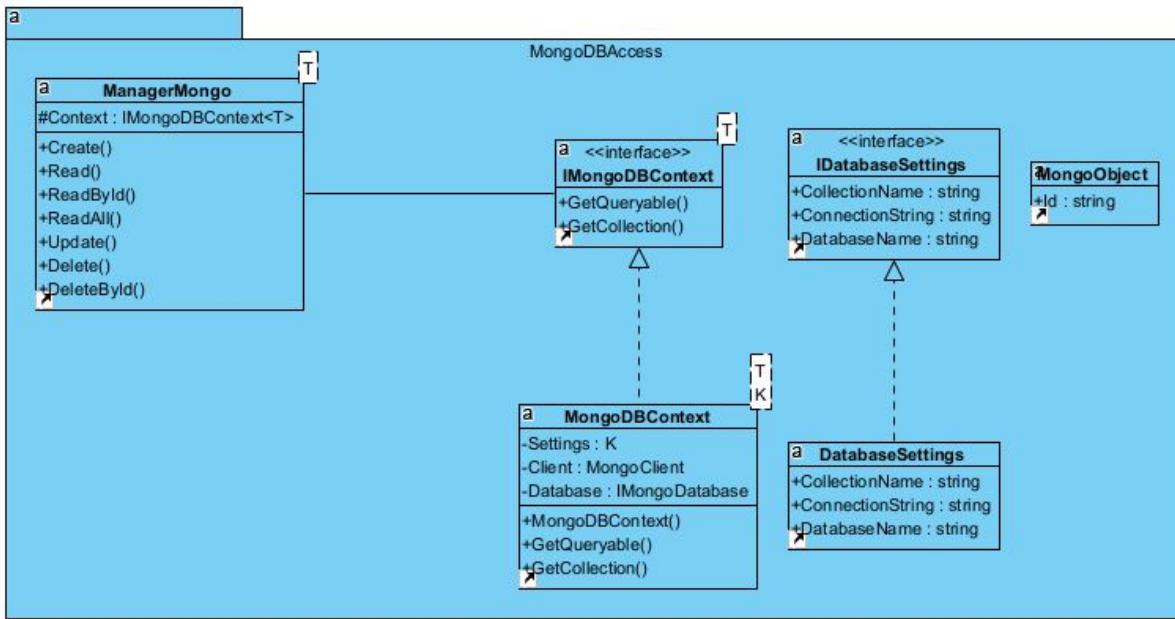
```
{ "MQTTServiceConfig": {
    "MQTTConfigBroker": {
        "Port": 1883,
        "EncryptedPort": 8884
    },
    "TopicNameDataRobot": "IOT/Data",
    "TopicNameMotorDataRobot": "IOT/Controller"
}, }
```

```
{
    "UsersDatabaseSettings": {
        "CollectionName": "users",
        "ConnectionString": "mongodb://mongoApp:27017",
        "DatabaseName": "IOT"
    },
    "DataDatabaseSettings": {
        "CollectionName": "data",
        "ConnectionString": "mongodb://mongoApp:27017",
        "DatabaseName": "IOT"
    },
    "AuthorizationDatabaseSettings": {
        "CollectionName": "authorization",
        "ConnectionString": "mongodb://mongoApp:27017",
        "DatabaseName": "IOT"
    }
}
```

4. Serveur Applicatif

4.6 Couches d'accès aux données (Data Access Layer)

4.6.1 Contexte d'accès aux données



```
/> Références
public interface IDatabaseSettings
{
    7 références
    public string CollectionName { get; set; }
    7 références
    public string ConnectionString { get; set; }
    7 références
    public string DatabaseName { get; set; }
}

/> Références
public class MongoDBContext<T, K> : IMongoDbContext<T> where K : IDatabaseSettings
{
    0 références
    private K Settings;
    private readonly MongoClient Client;
    private readonly IMongoDatabase Database;

    2 références
    public MongoDBContext(K settings)
    {
        this.Settings = settings;
        this.Client = new MongoClient(this.Settings.ConnectionString);
        this.Database = this.Client.GetDatabase(this.Settings.DatabaseName);
    }

    9 références
    public IQueryable<T> GetQueryable()
    {
        return this.GetCollection().AsQueryable();
    }

    9 références
    public IMongoCollection<T> GetCollection()
    {
        return this.Database.GetCollection<T>(this.Settings.CollectionName);
    }
}
```

4. Serveur Applicatif

4.6 Couches d'accès aux données (Data Access Layer)

4.6.2 Services Database

Nous avons principalement deux services :

- DataRobotService : données des différents robots
- UserService : données des différents utilisateurs de l'application

```
public class DataRobotService : ManagerMongo<DataRobot>
{
}

public class UserService : ManagerMongo<EquipmentAuth>
{
    public UserService(IMongoDBContext<EquipmentAuth> Context)
    {
        this.Context = Context;
    }

    public EquipmentAuth ReadByIdEquipment(string id)
    {
        return this.Context.GetQueryable().FirstOrDefault(element => element.IdEquipment.Equals(id));
    }
}
```

4. Serveur Applicatif

4.7 Services (Business Layer)

L'application est composée de différents services. Généralement ce sont des composants de niveau intermédiaire, typiquement ce sont :

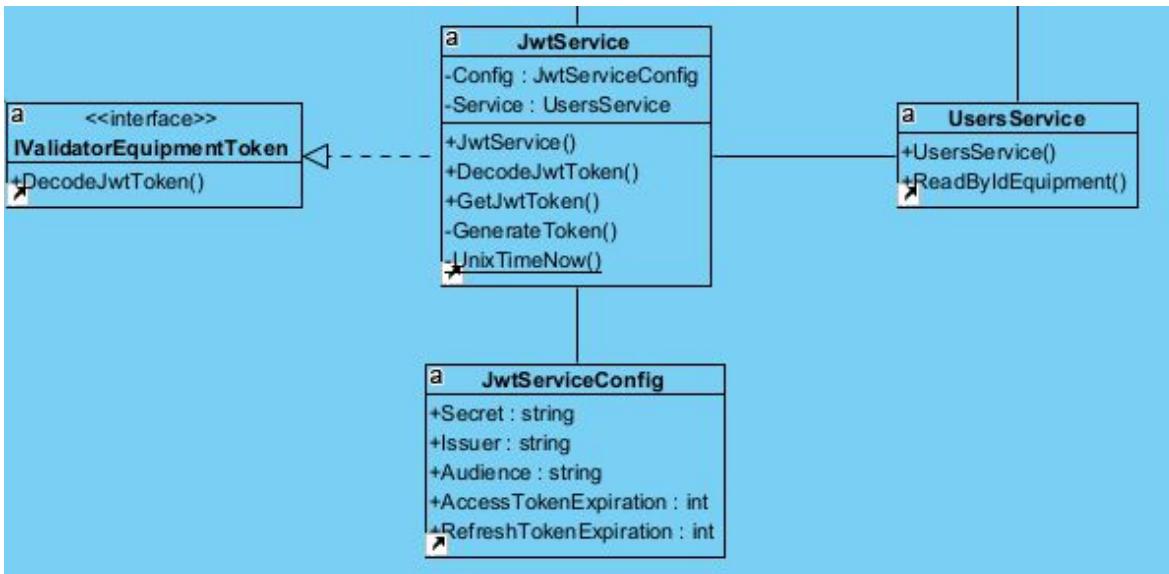
- Logique métier, ou validation de données
- Composants et logique d'accès aux données

Nous avons **principalement trois services** dans notre application, deux qui concernent **l'authentification et l'autorisation pour l'utilisation des ressources de notre application**. Enfin un dernier service, nous permettant de **faire persister dans un cache, l'état des connexions actuel des clients de l'application ainsi que certaines données**.

4. Serveur Applicatif

4.7 Services (Business Layer)

4.7.1 Authentification



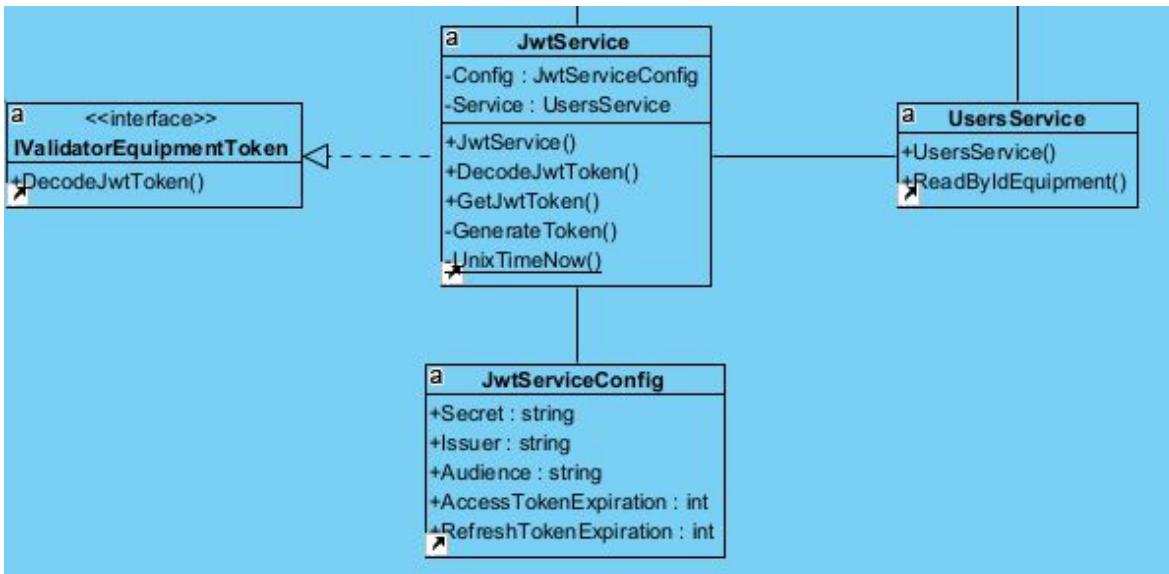
Génération des tokens :

- Contient une clef privée qui est convertie en hash,
- Ce hash est utilisé comme clef de chiffrement pour l'identité d'un équipement,
- Le token est ainsi généré à partir d'une identité et du hash de la clef privée
- Ce token sera alors envoyé aux ESP et à l'application de contrôle (MQTT et TCP)

4. Serveur Applicatif

4.7 Services (Business Layer)

4.7.1 Authentification



Décodage des tokens :

- Nous pouvons déchiffrer une identité en re-calculant le hash,
- Nous n'avons pas besoin de vérifier l'identité fournie par le token car nous sommes sûrs de son authenticité et de son intégrité (grâce à HMAC).

4. Serveur Applicatif

4.7 Services (Business Layer)

4.7.1 Authentification

Précision sur le HMAC:

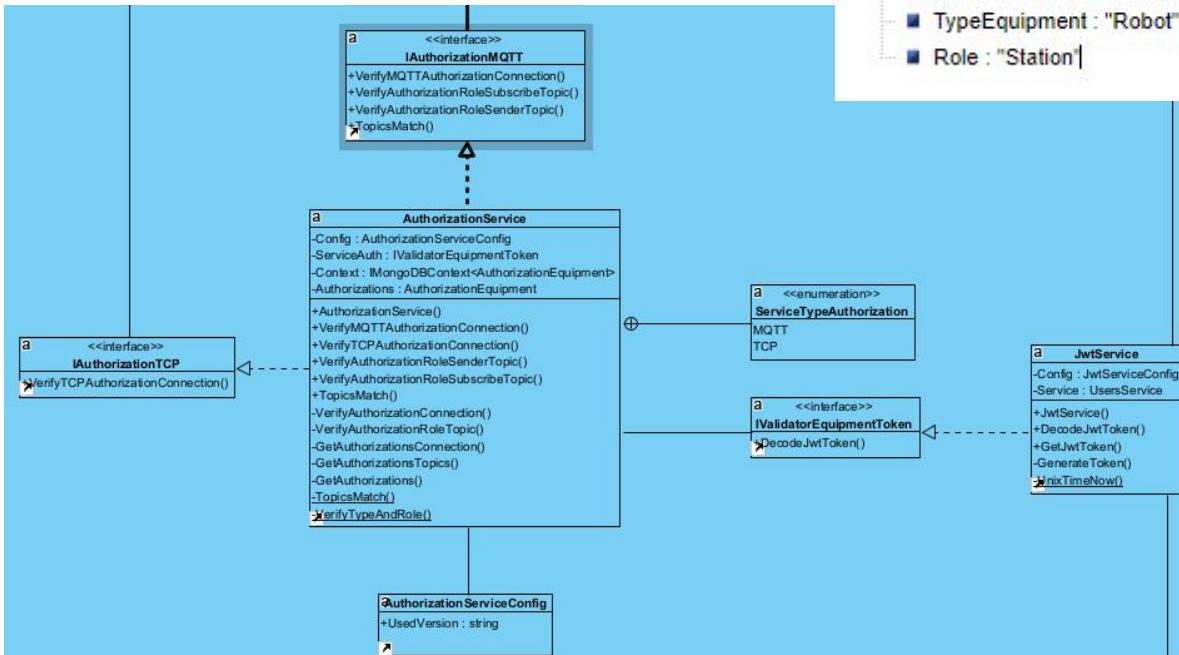
Le hash que nous utilisons est un code d'authentification de message (MAC) impliquant une fonction de hachage cryptographique et une clé cryptographique secrète. Nous l'utilisons pour vérifier simultanément l'intégrité des données et l'authenticité d'un message.

Cette méthode nous permet de déchiffrer un token plus simplement et plus rapidement, en évitant de vérifier une deuxième fois (au moment du déchiffrement) l'intégrité et l'authenticité du token.

4. Serveur Applicatif

4.7 Services (Business Layer)

4.7.2 Autorisation



JSON

```

{
  "IdEquipment": "Esp32Robot",
  "Password": "25Zjqgr8AQxxZsy",
  "TypeEquipment": "Robot",
  "Role": "Station"
}
  
```

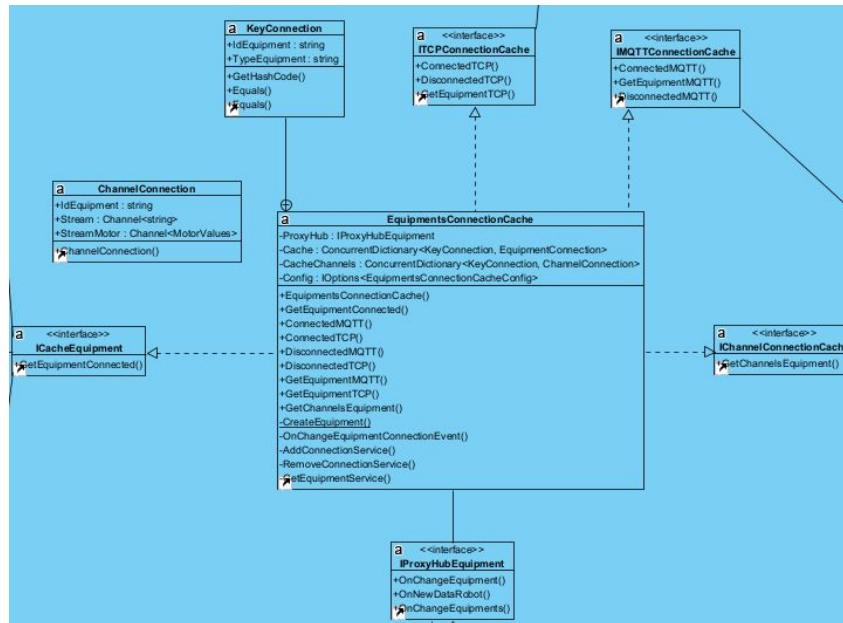
```

    "AuthorizationEquipment": {
      "Version": "1",
      "TopicsAuthorizationConfigs": [
        {
          "TopicName": "IOT/Data",
          "MaxTerms": 2,
          "AuthorizeSending": [
            {
              "AuthorizeTypeEquipment": "Robot",
              "AuthorizeRole": [ "Station" ]
            }
          ],
          "AuthorizeSubscribe": []
        },
        {
          "TopicName": "IOT/Controller",
          "MaxTerms": 3,
          "AuthorizeSending": [
            {
              "AuthorizeTypeEquipment": "Application",
              "AuthorizeRole": [ "Controller" ]
            }
          ],
          "AuthorizeSubscribe": [
            {
              "AuthorizeTypeEquipment": "Robot",
              "AuthorizeRole": [ "Station" ]
            }
          ]
        }
      ],
      "StreamAuthorizationConnectionConfigs": {
        "AuthorizeConnection": [
          {
            "AuthorizeTypeEquipment": "Robot",
            "AuthorizeRole": [ "Camera" ]
          }
        ]
      },
      "MQTTAuthorizationConnectionConfigs": {
        "AuthorizeConnection": [
          {
            "AuthorizeTypeEquipment": "Robot",
            "AuthorizeRole": [ "Station" ]
          },
          {
            "AuthorizeTypeEquipment": "Application",
            "AuthorizeRole": [ "Controller" ]
          }
        ]
      }
    }
  
```

4. Serveur Applicatif

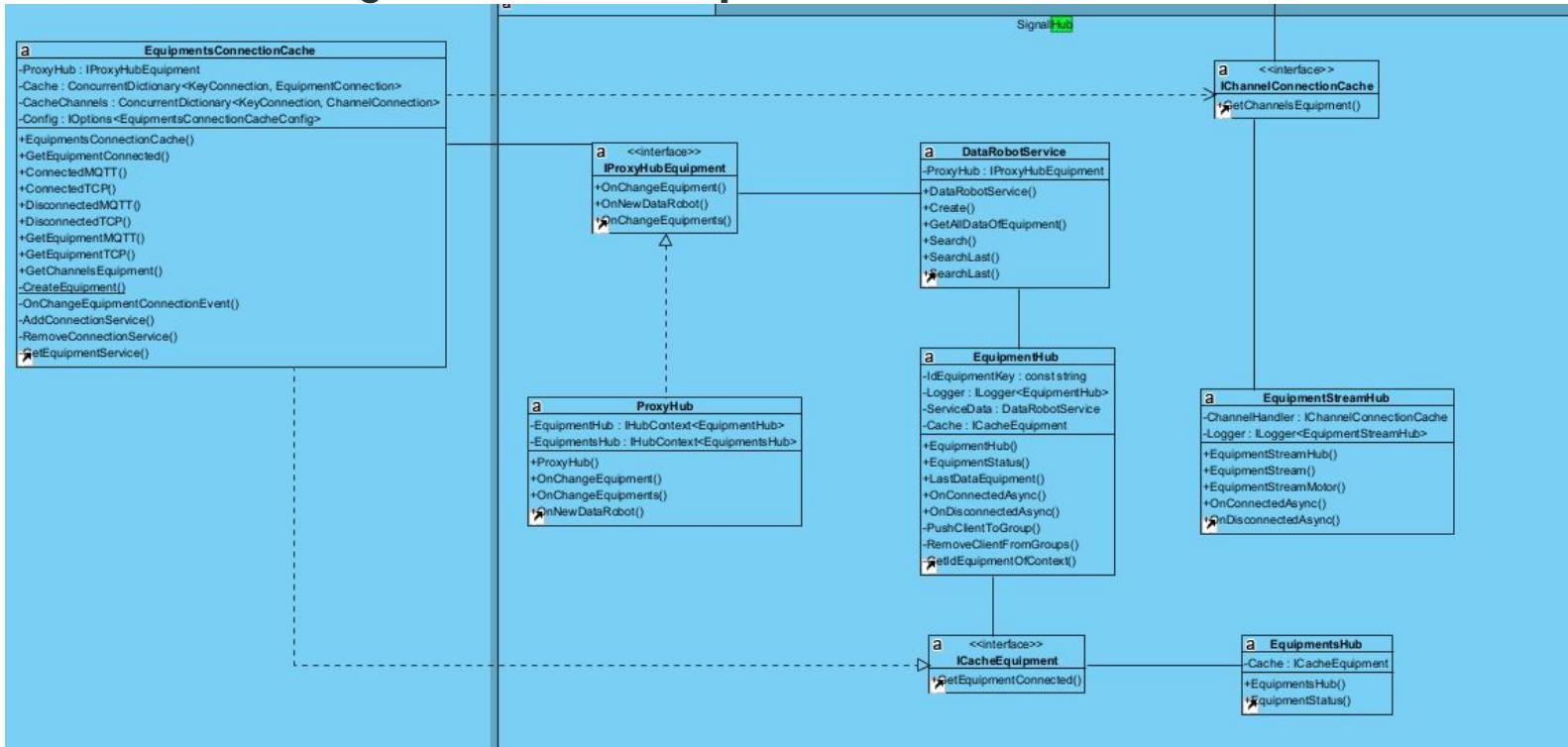
4.7 Services (Business Layer)

4.7.3 Cache de connexion et données



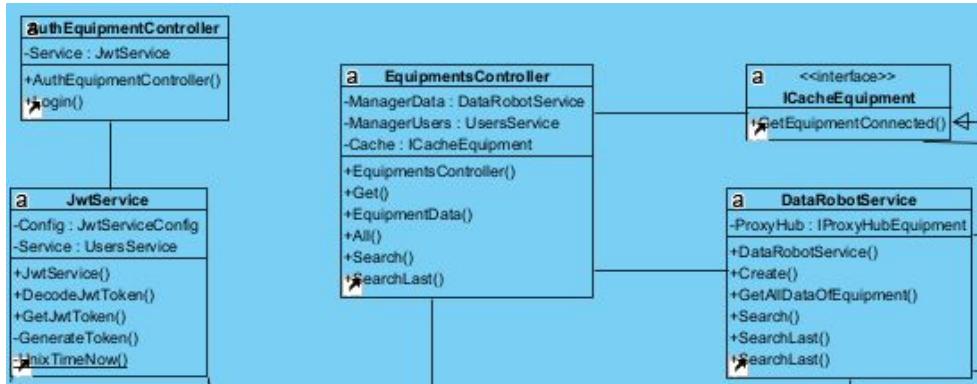
4. Serveur Applicatif

4.8 Service d'échange de données temps réel



4. Serveur Applicatif

4.9 API



```
[Route("api/[controller]")]
[ApiController]
1 référence
public class AuthEquipmentController : ControllerBase
{
    private readonly JwtService Service;

    0 références
    public AuthEquipmentController(JwtService Service)
    {
        this.Service = Service;
    }

    [DisableCors]
    [AllowAnonymous]
    [HttpPost("auth")]
    0 références
    public ActionResult<JwtToken> Login([FromBody] EquipmentAuthView equipment)
    {

        JwtToken token = Service.GetJwtToken(equipment);

        if (token is null)
            return BadRequest("{\"reason\": \"Bad credential\"}");

        return Ok(token);
    }
}
```

4. Serveur Applicatif

4.10 Services hébergés

Nous avons principalement 2 services hébergés :

- **Broker MQTT**
- **Serveur TCP**

4. Serveur Applicatif

4.10 Services hébergés

4.10.1 Service MQTT

Le service MQTT est un service hébergé dans le serveur applicatif. C'est un broker MQTT permettant la connexion d'autres applications pour permettre l'échange de données entre elles.

```
MQTTServiceConfig": {  
    "MQTTConfigBroker": {  
        "Port": 1883,  
        "EncryptedPort": 8884  
    },  
    "TopicNameDataRobot": "IOT/Data",  
    "TopicNameMotorDataRobot": "IOT/Controler"  
},  
    "HTTP": {  
        "Port": 8080  
    }  
}
```

Ici nous pouvons observer la configuration de notre serveur MQTT, à savoir :

Port : 1883 (le seul utilisé)

Port crypté pour SSL: 8884 (structure mise en place mais ne fonctionnement à cause d'une bibliothèque manquante sur ESP32).

4. Serveur Applicatif

4.10 Services hébergés

4.10.1 Service MQTT

Pour chaque événement arrivant, nous avons associé une fonction. Chaque fonction effectue des traitements, notamment : déserialiser, sauvegarder en base et propager au serveur applicatif.

Lors de la tentative de connexion au service MQTT, le client doit spécifier un token d'authentification qu'il à obtenu précédemment en s'identifiant auprès du serveur via l'API, suite à quoi, chaque requête comportera le token en question.

Aussi, le service est lancé au démarrage du service applicatif. Il est également responsable de contrôle de la connexion, de l'abonnement et de l'envoi de messages sur des topics.

4. Serveur Applicatif

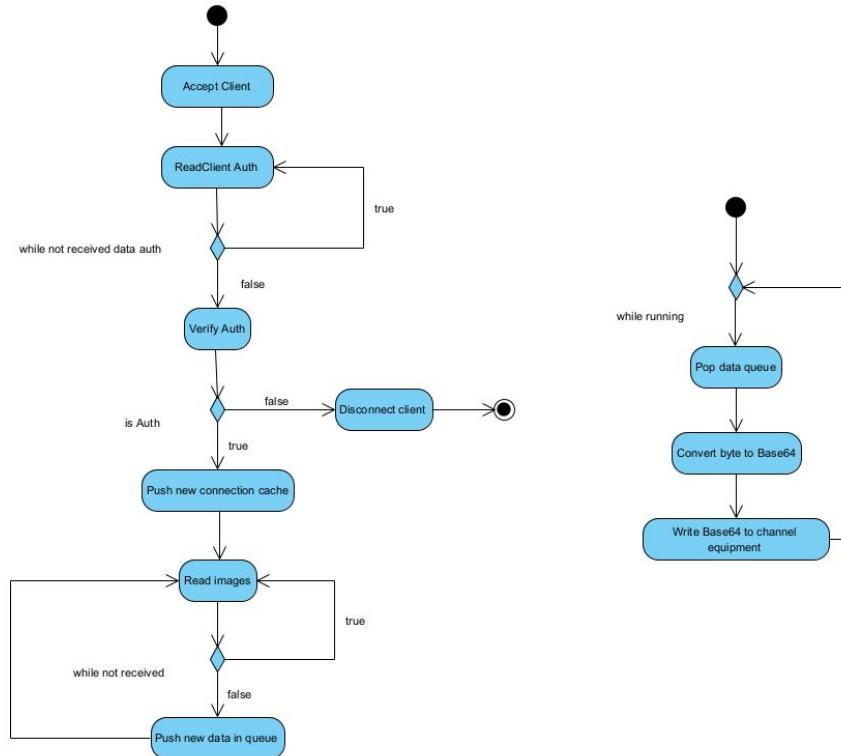
4.10 Services hébergés

4.10.2 Service TCP

- Asynchrone
- Protocol applicatif

```
"TCPServiceConfig": {  
    "Port": 11000  
}
```

```
public interface IStateBuffer  
{  
    7 références  
    public int CurrentSizeAttempting { get; set; }  
    5 références  
    public int RestSize { get => CurrentSizeAttempting - CurrentSizeBuffer; }  
    7 références  
    public int CurrentSizeBuffer { get; set; }  
    6 références  
    public byte[] Buffer { get; set; }  
    8 références  
    public byte[] Temp { get; set; }  
}
```



4. Serveur Applicatif

4.11 Injection de données en base

Les données d'authentification et d'identification par défaut qui sont insérées en base :

```
"EquipmentsAllowed": {  
    "Allowed": [  
        {  
            "IdEquipment": "Esp32Robot",  
            "TypeEquipment": "Robot",  
            "Password": "25Zjqgr8AQxxZsyz",  
            "Role": [  
                { "Name": "Station" },  
                { "Name": "Camera" }  
            ]  
        },  
        {  
            "IdEquipment": "WPFControler",  
            "TypeEquipment": "Application",  
            "Password": "25Zjqgr8AQxxZsyz",  
            "Role": [  
                { "Name": "Controler" }  
            ]  
        }  
    ],  
},
```

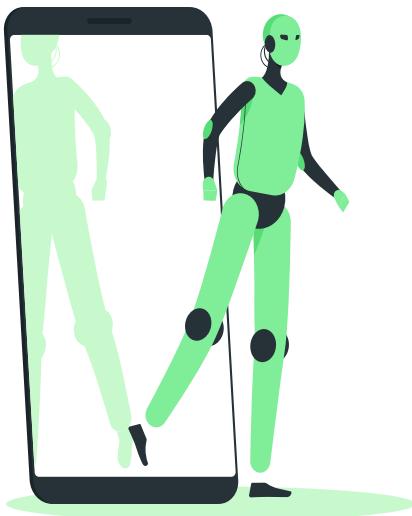
Données par défaut des autorisations pour les utilisateurs insérer en base :

```
,  
    "AuthorizationEquipment": {  
        "Version": "1",  
        "TopicsAuthorizationConfigs": [  
            {  
                "TopicName": "IOT/Data",  
                "MaxTerms": 2,  
                "AuthorizeSending": [  
                    {  
                        "AuthorizeTypeEquipment": "Robot",  
                        "AuthorizeRole": [ "Station" ]  
                    }  
                ],  
                "AuthorizeSubscribe": []  
            },  
            {  
                "TopicName": "IOT/Controler",  
                "MaxTerms": 3,  
                "AuthorizeSending": [  
                    {  
                        "AuthorizeTypeEquipment": "Application",  
                        "AuthorizeRole": [ "Controler" ]  
                    }  
                ],  
                "AuthorizeSubscribe": [  
                    {  
                        "AuthorizeTypeEquipment": "Robot",  
                        "AuthorizeRole": [ "Station" ]  
                    }  
                ]  
            }  
        ],  
        "StreamAuthorizationConnectionConfigs": {  
            "AuthorizeConnection": [  
                {  
                    "AuthorizeTypeEquipment": "Robot",  
                    "AuthorizeRole": [ "Camera" ]  
                }  
            ]  
        },  
        "MQTTAuthorizationConnectionConfigs": {  
            "AuthorizeConnection": [  
                {  
                    "AuthorizeTypeEquipment": "Robot",  
                    "AuthorizeRole": [ "Station" ]  
                },  
                {  
                    "AuthorizeTypeEquipment": "Application",  
                    "AuthorizeRole": [ "Controler" ]  
                }  
            ]  
        }  
    }
```

5. Application mobile

5.1 Présentation

- Application Xamarin.Forms : cross platform
- Utilisation du Bluetooth
 - Scan des appareils à proximité
 - Connexion
 - Envoie de données
- ESP Serveur Bluetooth
- Téléphone Balise GPS
- Sérialisation des données en JSON
- Service et Characteristic

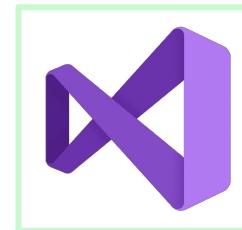


{ j s o n }

5. Application mobile

5.2 Environnement de développement

- Microsoft Visual Studio 2019 (Community Edition)
- C# et XAML
- Framework Newtonsoft (<https://www.newtonsoft.com/json>)
- Plugin Geolocator
(<https://github.com/jamesmontemagno/GeolocatorPlugin>)
- Plugin BLE (<https://github.com/xabre/xamarin-bluetooth-le>)
- Cross platform : iOS et Android
- Samsung Galaxy J3 6
 - Android 5.1
 - API 22



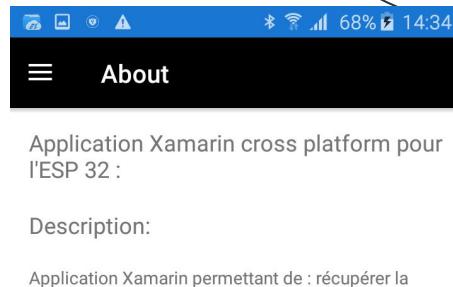
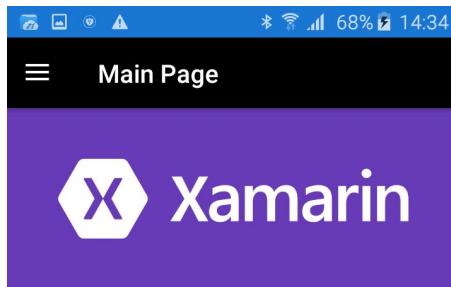
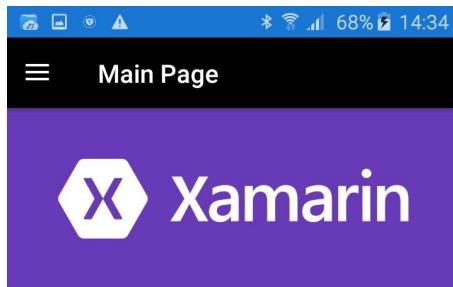
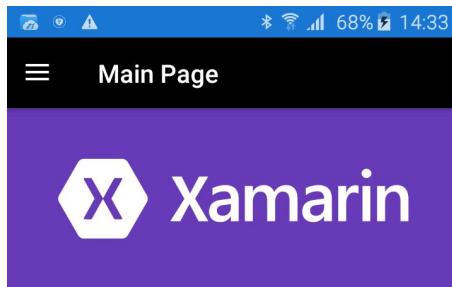
5. Application mobile

```
Init Bluetooth server.  
Setup BLE Server.  
[0;32mI (30) BTDM_INIT: BT controller compile version [c1cbe45][0m  
[0;32mI (886) system_api: Base MAC address is not set, read default base MAC address from BLK0 of EFUSE[0m  
[0;32mI (996) phy: phy_version: 4180, cb3948e, Sep 12 2019, 16:39:13, 0, 0[0m  
Setup BLE Server finished.  
End init Bluetooth server.
```

```
Coord c = new Coord(pos.Latitude, pos.Longitude);  
string msg = JsonConvert.SerializeObject(c);  
data = Encoding.UTF8.GetBytes(msg);  
var service = await device.GetServiceAsync(new Guid("4fafc201-1fb5-459e-8fcc-c5c9c331914b"));  
var charatecistics = await service.GetCharacteristicAsync(new Guid("beb5483e-36e1-4688-b7f5-ea07361b26a8"));  
await charatecistics.WriteAsync(data);
```

```
Event connection BLEReceive data from BLE{"Latitude":43.799515725061994,"Longitude":7.20858664473405}  
Receive data from BLE{"Latitude":43.799570146715233,"Longitude":7.2085328383894831}  
Receive data from BLE{"Latitude":43.799531766023634,"Longitude":7.20856687270479}  
Receive data from BLE{"Latitude":43.799531766023634,"Longitude":7.20856687270479}  
Receive data from BLE{"Latitude":43.799531766023634,"Longitude":7.20856687270479}  
Receive data from BLE{"Latitude":43.799462982644521,"Longitude":7.2084828757514421}  
Receive data from BLE{"Latitude":43.799462982644521,"Longitude":7.2084828757514421}
```

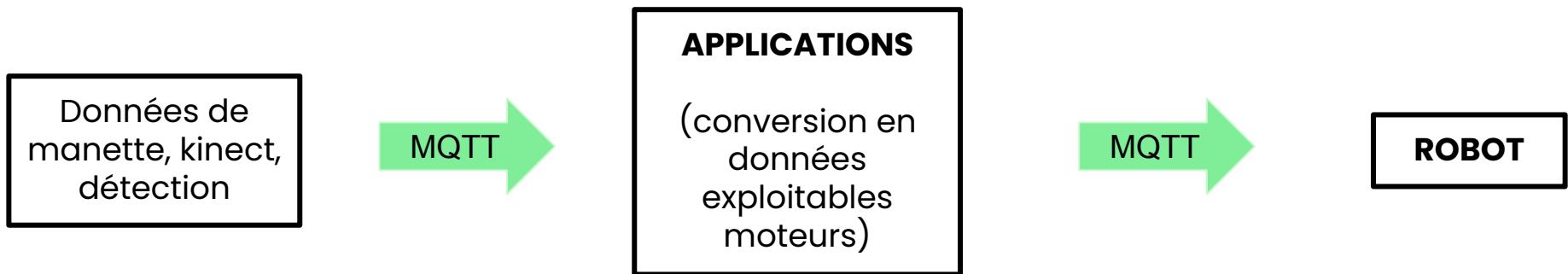
5. Application mobile



6. Contrôle du robot

6.1 Présentation

- Application de contrôle par manette / kinect
- Application de contrôle par reconnaissance d'images



Plage de valeurs [-100;100] pour représenter la puissance et le sens des différents moteurs pour toutes les applications.

6. Contrôle du robot

6.2 Application WPF (manette/kinect)

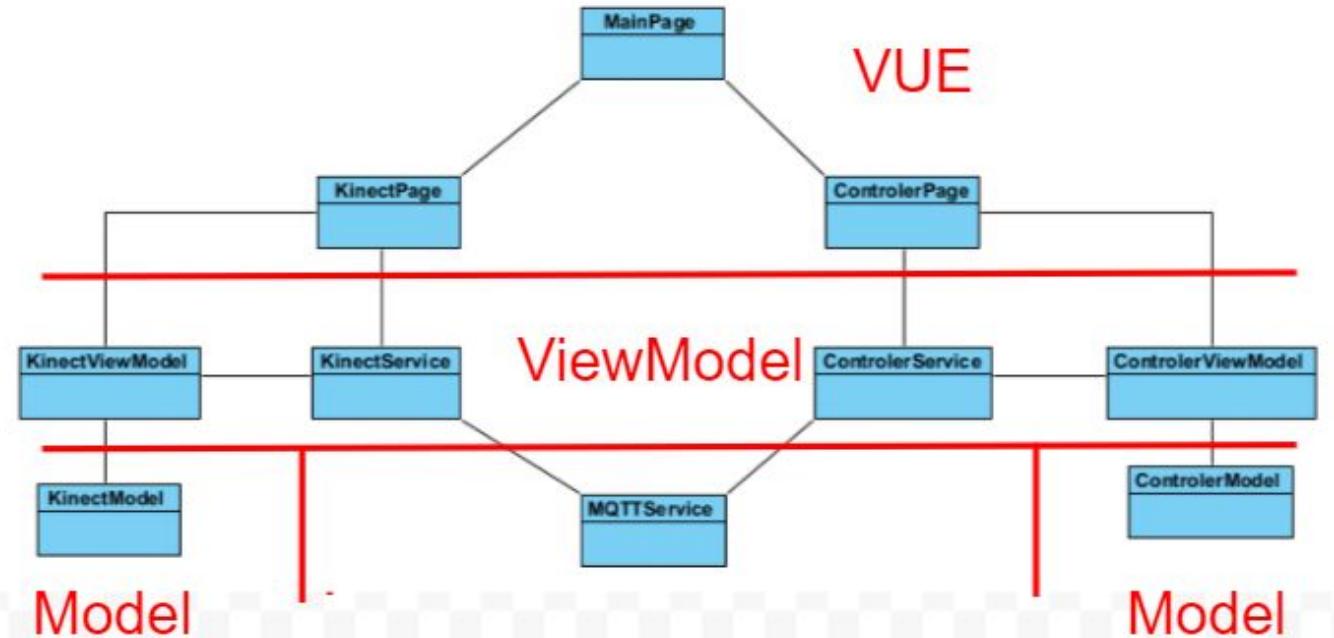
6.2.1 Environnement de développement



6. Contrôle du robot

6.2 Application WPF (manette/kinect)

6.2.2 Architecture



6. Contrôle du robot

6.2 Application WPF (manette/kinect)

6.2.3 Service MQTT Client

```
1 référence
private void OnDisconnect(MqttClientDisconnectedEventArgs args)
{
    DisconnectedEvents?.Invoke(args);
}

1 référence
private void OnConnect(MqttClientConnectedEventArgs args)
{
    ConnectedEvents?.Invoke(args);
}

1 référence
public async Task<Task> Connect()
{
    var json = JsonConvert.SerializeObject(this.ConfigTopic.EquipmentAuth);
    var data = new StringContent(json, Encoding.UTF8, "application/json");

    var response = await HttpClient.PostAsync(this.ConfigTopic.ApiURL, data);
    string result = response.Content.ReadAsStringAsync().Result;

    var jwt = JsonConvert.DeserializeObject<JwtToken>(result);

    return this.Client.Connect(jwt.AccessToken, "");
}

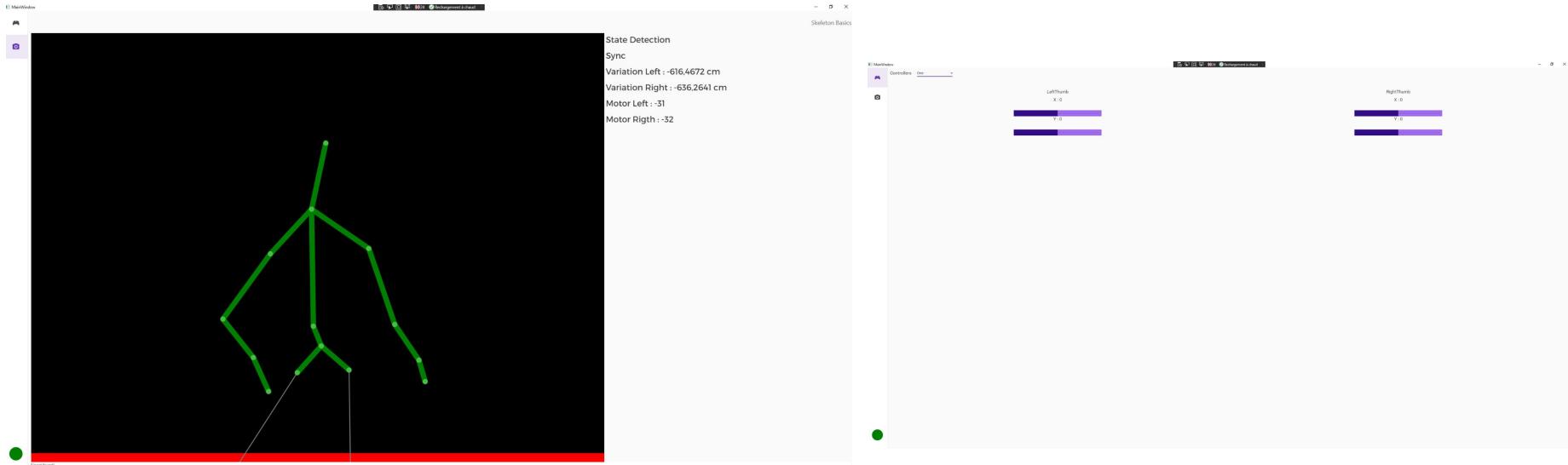
1 référence
public Task Disconnect()
{
    return this.Client.Disconnect();
}

4 références
public Task SendDataMotors(MotorValues message)
{
    return this.Client.Publish(ConfigTopic.TopicSendControl, JsonConvert.SerializeObject(message));
}
```

6. Contrôle du robot

6.2 Application WPF (manette/kinect)

6.2.4 Présentation du dashboard



6. Contrôle du robot

6.2 Application WPF (manette/kinect)

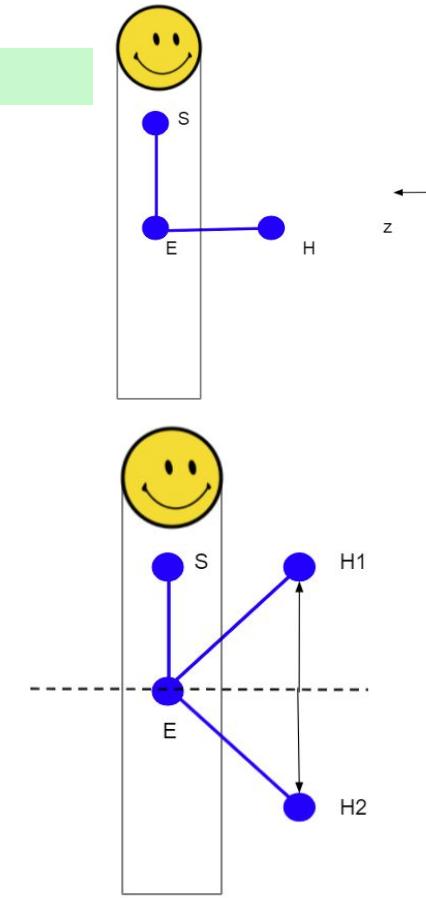
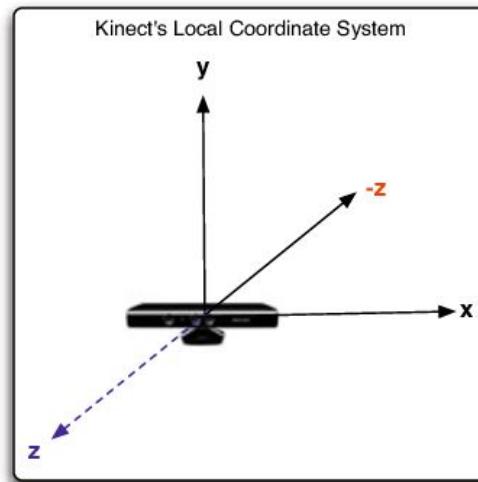
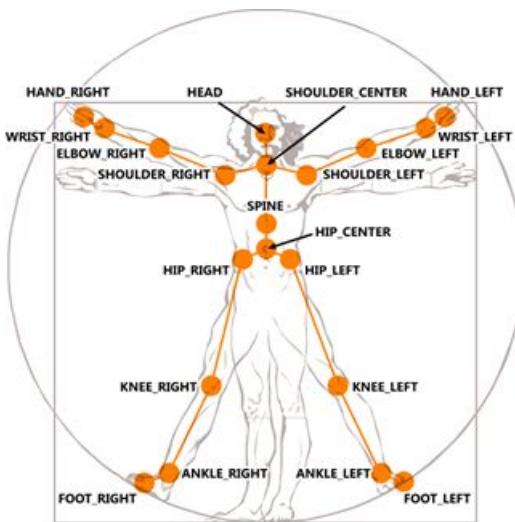
6.2.5 Contrôle par manette



6. Contrôle du robot

6.2 Application WPF (manette/kinect)

6.2.6 Contrôle par Kinect



6. Contrôle du robot

6.3 Contrôle OpenCV



```

# Display the results
for (top, right, bottom, left), name in zip(face_locations, face_names):
    top *= 4
    right *= 4
    bottom *= 4
    left *= 4

    # Draw a rectangle around the face
    cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)
    # Input text label with a name below the face
    cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0, 255), cv2.FILLED)
    font = cv2.FONT_HERSHEY_DUPLEX
    cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255, 255, 255), 1)

font          = cv2.FONT_HERSHEY_SIMPLEX
topLeftCornerOfText = (10,25)
fontScale     = 0.6
fontColor     = (255,255,255)
lineType      = 2

cv2.putText(frame,'Detected:', topLeftCornerOfText, font, fontScale,fontColor,lineType)
cv2.putText(frame,'Exit code: q', (500,25), font, 0.6,fontColor,lineType)

index=25
for i in names :
    if name == i :
        if name not in detection :
            detection.append(name)

    if name == str("Unknown") :
        nbInconnus=nbInconnus+1
        name = ""

detection=[]
nbInconnus=0
name=""

while True:
    ret, frame = video_capture.read() ...

```

Dessine le rectangle autour des visages détecté ou identifier.
Si le visage est identifié, son nom est écrit sous le rectangle.
Si le visage est détecté mais pas identifié, le script écrit "Unknown" sous le rectangle.

Cette partie permet d'enregistrer et d'afficher les noms des personnes détectés pendant le stream en les cumulant
Le script compte également le nombre d'image avec un "inconnu" détecté.
Ces résultats sont affichés dans le stream.

Présentation d'une partie du code

7. Interface web de visualisation

7.1 Présentation

Visualiser les données des équipements en temps réel (en l'occurrence les ESP32) :

- température de l'environnement,
- luminosité de l'environnement,
- caméra en direct du robot,
- localisation du robot via une carte,
- ESP connectés en direct,
- utilisation des moteurs de l'ESP en direct.

Cette interface sert alors de dashboard, accessible à l'adresse suivante :

<https://51.38.226.68/dashboard>

7. Interface web de visualisation

7.2 Environnement de développement



Visual Studio Code

Nos scripts permettent 4 types d'exécution :

- **local** : permettant de lancer l'environnement dev en HTTP
- **localSsl** : permettant de lancer l'environnement dev en HTTPS
- **dockerSsl** : permettant l'exécution dans un environnement Docker avec HTTPS en dev
- **dockerSslProd** : permettant l'exécution dans un environnement Docker avec HTTPS en prod

7. Interface web de visualisation

The screenshot displays a web-based interface titled "IOT Application". On the left, there is a vertical purple sidebar with a "Dashboard" icon and a three-line menu icon. The main content area has a white background with a purple header bar. In the top right corner of the main area, the text "Connected robots" is displayed. Below this, there is a card with a purple header containing the text "Esp32Robot - Robot" and a small Wi-Fi signal icon. At the bottom of this card is a purple "Show" button. To the left of the card, the word "Connected robots" is repeated.



Détails : Esp32Robot - Robot

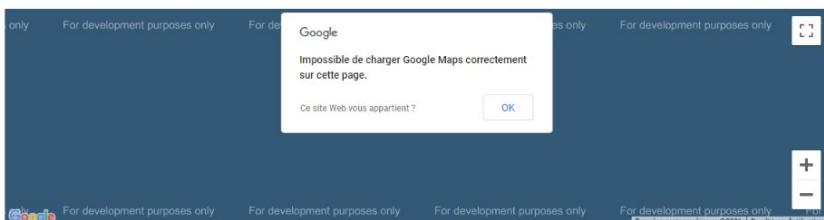
Latest Data

Temperature	25
Ligth	1149
Latitude	0
Longitude	0
Timestamp	12/06/2021, 20:01:40

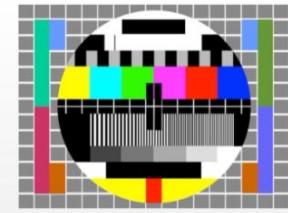
Services

Name	IdConnection	Role	AddressIp
MQTT	24:0A:C4:61:91:F4	Station	172.19.0.1:5372

Geolocation



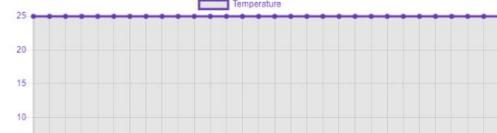
Stream



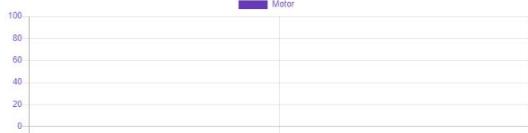
Line chart Ligth



Line chart Temperature



Bar chart Motor



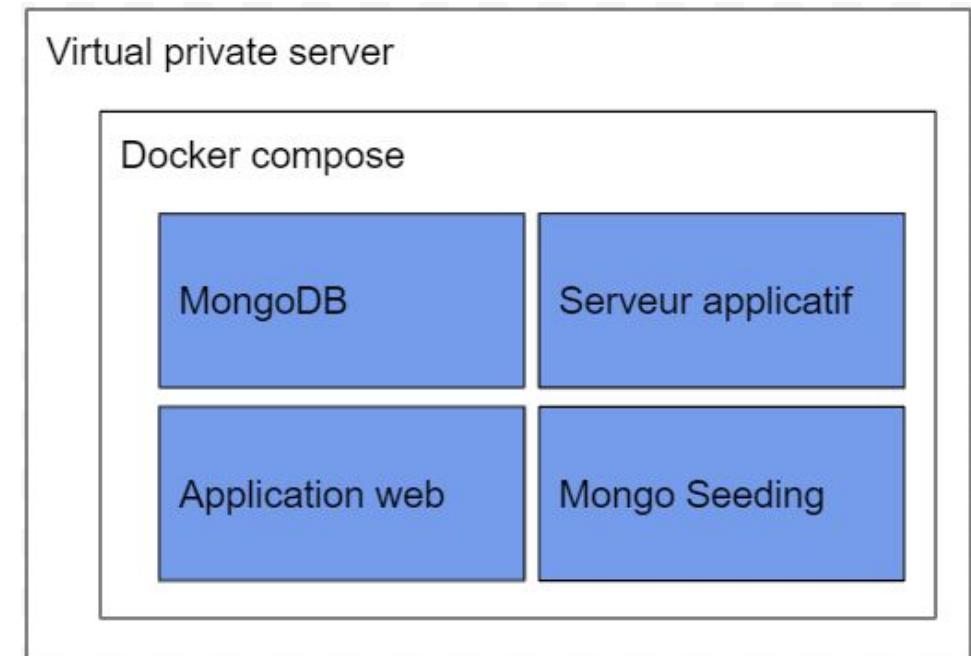
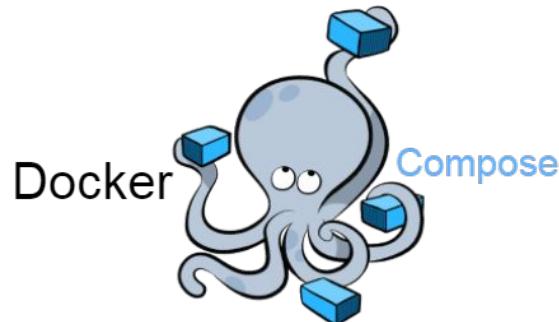
8. Sécurité

- **SSL/TLS pour HTTPS (OpenSSL):**
 - API
 - Websocket
 - MQTT
- **Mise en place d'authentification via une identification des applications et autorisation d'accès :**
 - Serveur TCP pour la connexion
 - Serveur MQTT pour la connexion, publication et l'abonnement

9. Déploiement de l'application



docker



9. Déploiement de l'application

```
👉 Dockerfile
1  FROM node:12.16.1-alpine
2
3  RUN npm install -g @angular/cli@11.2.12
4
5  WORKDIR /app
6
7  COPY ./package.json ./package-lock.json .
8  RUN npm install
9
10 COPY ..
11
12 ENTRYPOINT ["npm", "run", "dockerSslProd"]
```

```
FROM mcr.microsoft.com/dotnet/aspnet:5.0 AS base
WORKDIR /app
EXPOSE 80
EXPOSE 443

FROM mcr.microsoft.com/dotnet/sdk:5.0 AS build
WORKDIR /src
COPY ["Projects/APIRobot/APIRobot.csproj", "Projects/APIRobot/"]
COPY ["Libraries/MongoDBAccess/MongoDBAccess.csproj", "Libraries/MongoDBAccess/"]
COPY ["Libraries/ConfigPolicy/ConfigPolicy.csproj", "Libraries/ConfigPolicy/"]
COPY ["Projects/SharedModels/SharedModels.csproj", "Projects/SharedModels/"]
RUN dotnet restore "Projects/APIRobot/APIRobot.csproj"
COPY ..
WORKDIR "/src/Projects/APIRobot"
RUN dotnet build "APIRobot.csproj" -c Release -o /app/build

FROM build AS publish
RUN dotnet publish "APIRobot.csproj" -c Release -o /app/publish

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "APIRobot.dll"]
```

9. Déploiement de l'application

```
version: '3.4'

services:
  mongoApp:
    container_name: mongoApp
    image: mongo:latest
    restart: always
    volumes:
      - ./mongodb/db:/data/db
    ports:
      - "27017:27017"

  mongoseeding:
    image: ${DOCKER_REGISTRY}-mongoseeding
    build:
      context: ../../SolutionIOT
      dockerfile: Projects/MongoSeeding/Dockerfile
    links:
      - mongoApp

  apirobot:
    image: ${DOCKER_REGISTRY}-apirobot
    restart: always
    build:
      context: ../../SolutionIOT
      dockerfile: Projects/APIRobot/Dockerfile
    links:
      - mongoApp

  frontrobot:
    container_name: FrontRobot
    image: ${DOCKER_REGISTRY}-frontrobot
    restart: always
    build:
      context: ../../Angular
      dockerfile: Dockerfile
    links:
      - apirobot
```

```
version: '3.4'

services:
  apirobot:
    environment:
      - ASPNETCORE_URLS=https://+:8001;http://+:8000
      - ASPNETCORE_HTTPS_PORT=5001
      - ASPNETCORE_Kestrel_Certificates_Default_Password=0099669
      - ASPNETCORE_Kestrel_Certificates_Default_Path=/https/certificat.pfx
    ports:
      - "8000:8000"
      - "8001:8001"
      - "1883:1883"
      - "11000:11000"
      - "8884:8884"
    volumes:
      - ./Certificates/self/https:/https/
    frontrobot:
      ports:
        - "443:443"
      volumes:
        - ./Certificates/self/https/
```

```
version: '3.4'

services:
  apirobot:
    environment:
      - ASPNETCORE_ENVIRONMENT=Production
```

```
#!/bin/bash

mv ../Project/DockerCompose
docker-compose -f docker-compose.yml -f docker-compose.override.yml -f docker-compose.vs.release.yml build
docker-compose -f docker-compose.yml -f docker-compose.override.yml -f docker-compose.vs.release.yml up -d
```

```
root@vps-b75ef108:/home/ubuntu/github_repo/prod/Projet-IOT-MIAGE-M1/Projet/DockerCompose# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
1f11b42aa1cc        frontrobot          "npm run dockerSslPr..."   12 days ago        Up 12 days         0.0.0.0:443->443/tcp, ::1:443->443/tcp
3d34286a70f7        apirobot           "dotnet APIRobot.dll"   12 days ago        Up 12 days         80/tcp, 0.0.0.0:1883->1883/tcp, ::1:1883->1883/tcp
t_1
181634f9bd9f        mongo:latest        "docker-entrypoint.s..."  12 days ago        Up 12 days         0.0.0.0:27017->27017/tcp, ::1:27017->27017/tcp
root@vps-b75ef108:/home/ubuntu/github_repo/prod/Projet-IOT-MIAGE-M1/Projet/DockerCompose#
```

BILAN

1. Résultats

Nous avons donc :

- Un robot composé d'un ESP pour contrôler les moteurs, récolte les données des capteurs, de l'ESP pour gérer la caméra embarqué, les deux pouvant interagir avec le Cloud.
- Un serveur applicatif récolte les données des robots, et les sauvegarde. Sert de proxy entre l'application de contrôle et le robot, et entre la caméra embarquée et le site web.
- Le serveur applicatif met en œuvre l'identification, l'authentification, les autorisations et le chiffrement.
- L'application de contrôle permet le contrôle des robots via une manette ou kinect.
- Nous avons une application de visualisation du robot pour les utilisateurs.

Notre robot envoie alors les images de la caméra ainsi que ses données au serveur, qui les sauvegarde et les redirige vers l'application de visualisation. Il est contrôlable grâce à notre application de contrôle via le Cloud et localisable.

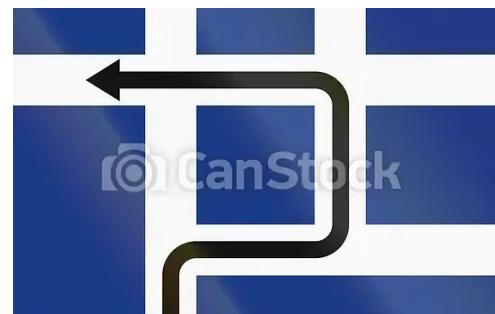
BILAN

2. Difficultés rencontrés

- Pas accès au matériel tous en même temps à cause de la distance (télétravail etc..)
- Difficile de trouver des créneaux pour se réunir et bosser en physique sur le robot
- Matériel manquant

BILAN

3. Evolutions possibles





**Merci pour votre
attention !**

Des questions ?