Rapport de Sarah Hamdi et Amar Hadjeres

## Description du projet

Ce projet se base sur le travail précèdent. En effet, après avoir récolté les informations utiles au contrôle de la porte d’une serre, on envoi sous la forme d’un message ces renseignements et on les stock localement. C’est à l’Iot hub d’azure que le message est envoyé en premier pour y être stocké. Une application web va donc ensuite chercher ces messages et afficher les informations correspondantes au bon endroit.

Le but est de pouvoir contrôler la porte à partir de l’application web et d’afficher les commandes exécutées dans le Raspberry pi dans la page web à l’aide des fonctionnalités cloud d’Azure.

En plus des mêmes informations à afficher et les mêmes boutons, la page web contient un bouton qui illustre sous la forme d’un graphique les variations de température et de pourcentage selon le temps. Une Alerte est également ajoutée pour signaler si la porte n’est pas assez ouverte/ fermée.

## La description de toutes les étapes de configuration de l’infonuagique

Une image contenant table

Description générée automatiquement

* Dans iot hub, on ajoute un nouvel appareil. Dans la machine, il y a un code qui utilise la clé de connexion principale pour pouvoir envoyer des messages de cet appareil. (AzureCloud.py)
* Les messages peuvent être lus par l’application web grâce à un consumer group et la clé de connexion primaire (server.js)
* Création d’un serveur de base de données mariaDB, le connecter au Workbench et y accéder grâce au lien du serveur pour récupérer les informations dans la BD (server graph.js)
* Création d’une application de fonction Azure : <https://tp2-gestion.azurewebsites.net>

## La description des codes des applications

**Raspberry pi :**

Classe Main : S’occupe du UI et du déroulement du programme, elle appelle toutes les autres classes pour les instancier sauf ADCDevice.

Classe UltrasonicRanging : S’occupe d’envoyer des pulsions et les recevoir pour déterminer la distance.

Classe Thermometer : S’occupe d’obtenir le voltage et l’ajouter dans un calcul pour obtenir la température en Celsius. Elle appelle la classe ADCDevice.

Classe ADCDevice : Classe qui s’occupe de l’ADC et qui est utilisé par le thermistor

Classe SteppingMotor: Classe qui s’occupe de faire tourner le moteur à une vitesse de 2 tours / minutes dans une direction donné.

Classe AzureCloud : Classe qui communique avec l’iot hub pour envoyer un message

Classe Database : Classe qui ajoute dans la BD locale les informations des messages avant de l’envoyer

**App web :**

Server.js : création du serveur web, va chercher les messages d’Azure Iot hub et les envoie à la page html

Server graph.js : création d’un serveur web et se connecte au serveur de bd de maria dB pour récupérer les informations d’une table

Iot-hub-connection-string.js : utilisé par le server.js pour récupérer les messages à partir du iot hub

Event-hub-reader.js : code qui lit les messages dans Azure

Index.html : Page web principale qui affiche toutes les informations

Graph.html : Page web qui affiche les graphiques

Style.css : Contient le style de l’application web

## Un manuel d’utilisation de l’application

1 – S’assurer d’avoir assemblé les composant électroniques

2 – Dans un terminal, exécuter les commandes suivantes :

* sudo python3 -m pip install --upgrade requests
* pip3 install azure-iot-device
* piip3 install mariadb

3 – Ouvrir le fichier Main.py sur le Raspberry pi et l’exécuter

4 – Ajouter les variables d’environnement suivantes :

* IotHubConnectionString 🡪 HostName=ObjetsConnectesHub.azure-devices.net;SharedAccessKeyName=service;SharedAccessKey=GyH0kvk7bYcQKlg/A8wIneGPxHEMDITk7YqYUd1PUR0=
* EventHubConsumerGroup 🡪 consumer\_group

5 – Ouvrir l’application web dans Visual studio code. Y ouvrir un terminal, s’assurer d’être dans le dossier ***projet azure,*** d’avoir node et exécuter ces commandes : *npm install* puis *npm start*

6 – Dans un navigateur web, se rendre sur [localhost:3000](http://localhost:3000/)