29/05/2021

SYSTÈME D'ÉCLAIRAGE ET D'IRRIGATION INTELLIGENT

IOT

2020-2021

RETIF ROMAIN
COLSON KEVIN
SIAUD MATHIEU

SOMMAIRE

03 CONCEPTION DU PROJET

- DESCRIPTION
- MATERIELS
- TECHNOLOGIES
- DIAGRAMMES
- MAQUETTES DE L'APPLICATION
- CODE SOURCE

04 DEFINITION DE L'API REST

- DÉFINITION ET DESCRIPTION DES ROUTES

05 REALISATION OBJET NON CONNECTÉ

- PLAN DE MONTAGE
- AMÉLIORATIONS POSSIBLES

15 REALISATION OBJET CONNECTÉ

- COMMUNICATION AVEC L'OBJET
- LIBRAIRIES
- DIFFICULTÉS RENCONTRÉES

15 MISE EN PLACE SERVEUR API / APPLICATION WEB

- NODE JS ET CODE METIER
- PRODUIT FINAL

CONCEPTION DU PROJET

DESCRIPTION

Le projet va permettre de gérer l'éclairage et l'arrosage d'une plante en fonction de la luminosité et de la température et de l'humidité de l'air.
Les seuils de déclenchement pourront être paramétrés par l'utilisateur. Enfin, l'historique des actions pourra être consultable.

MATERIELS

Pour mener à bien ce projet, nous avons utilisé le matériel suivant :

Starter Kit Feather HUZZAH32,
Capteur de luminosité ,
Capteur de temperature,
Panneau de LEDS

TECHNOLOGIES

Au niveau de l'API, nous avons opté pour la technologie NodeJs qui propose des modules déjà conçus pour gérer les requêtes HTTP.
Pour l'application client, nous avons utilisés les technologies classiques de Web : HTML, CSS et JavaScript.
Pour le microcontrôleur, nous avons utilisé Arduino et le langage C++, et la base de données est sous MySQL.

DIAGRAMMES

Diagramme de donnee

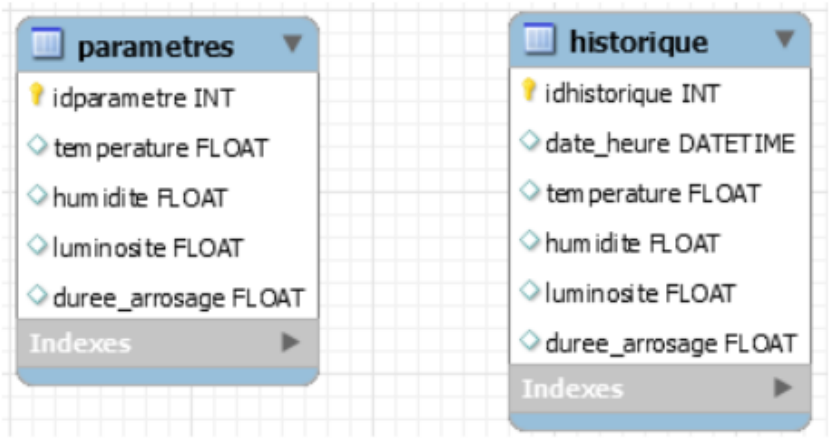
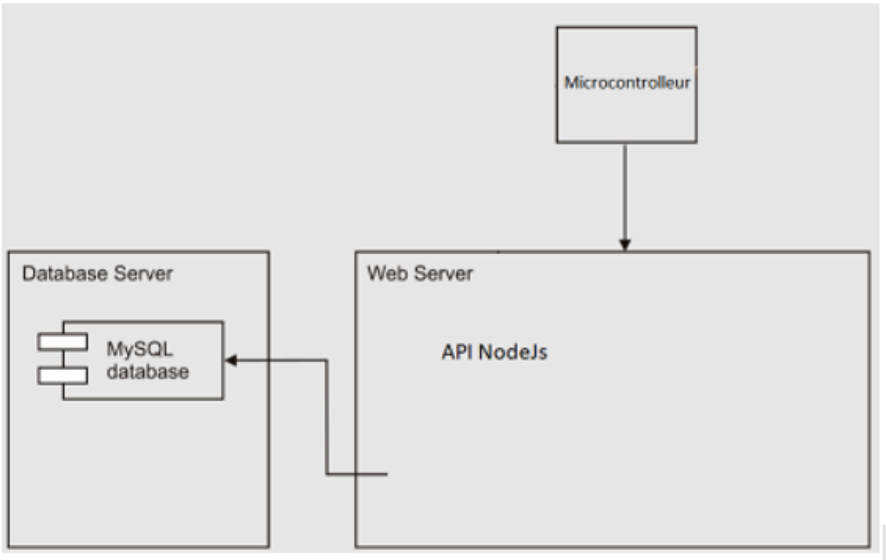
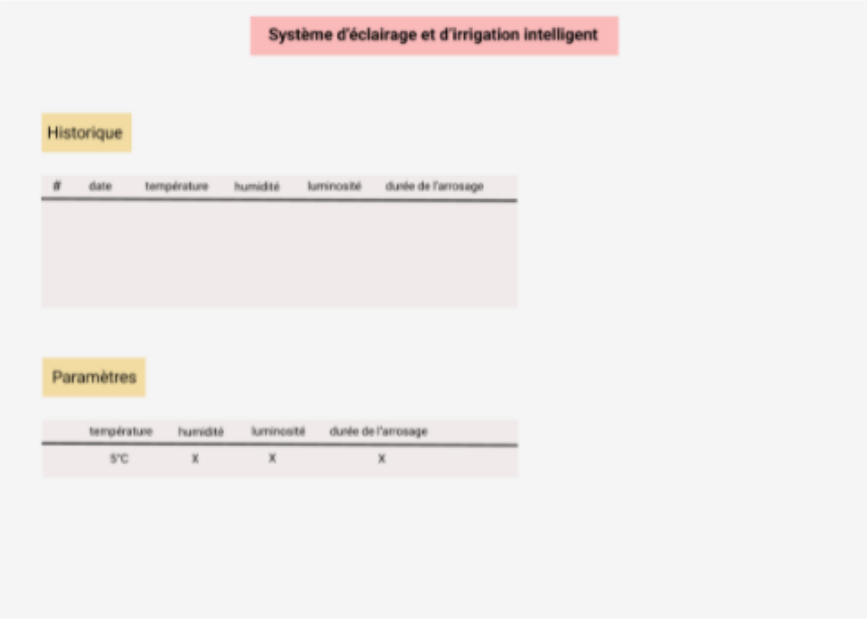


Diagramme de déploiement



MAQUETTES DE L'APPLICATION



CODE SOURCE

https://github.com/RomainRETIF/iot-arrosoir-api

DEFINITION DE L'API REST

L'api comprends plusieurs routes :

GET /historique
retourne l'historique des déclenchements (date, température, humidité, luminosité, durée de l'arrosage) sous forme de JSON.

POST /historique/add
ajoute les date, température, humidité, luminosité, durée de l'arrosage, présents dans le body de la requête, dans la base de données.

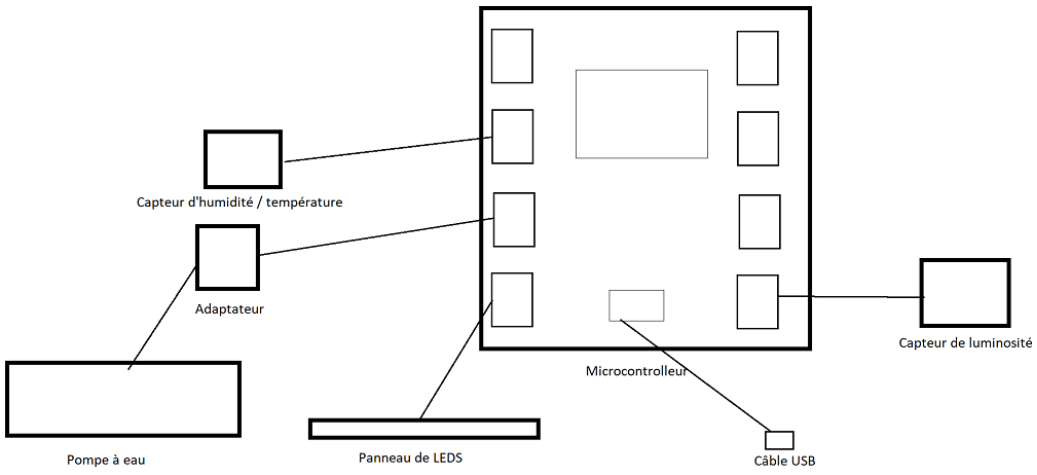
GET /parametres
retourne les température, humidité, luminosité, durée de l'arrosage entrée par l'utilisateur (ce sont les seuils) sous forme de JSON

PUT /parametres/update
met à jour les température, humidité, luminosité, durée de l'arrosage présents dans le body, dans la base de données

Les échanges se font via des requêtes HTTP. Le microcontrôleur envoie des requêtes HTTP à notre API NodeJS qui tourne sur l'ordinateur hôte.

DEFINITION DE L'OBJET NON CONNECTÉ

L'objet est composé du microcontrôleur auquel sont attachés les capteurs de luminosité, d' humidité et de température de l'air, ainsi que la pompe et le panneau de LEDs. Le microcontrolleur est lui même branché en USB.



Plan de montage du produit final

AMÉLIORATIONS POSSIBLES

Une amélioration possible aurait été d'ajouter un capteur qui mesurerait l'humidité de la terre et qui pourrait déclencher l'arrosage si celle-ci est trop faible. Par manque de port disponible sur le microcontrôleur, nous n'avons pas pu le brancher, bien que le code soit écrit.

RÉALISATION DE L'OBJET CONNECTÉ

Nous avons utilisé les librairies suivantes :

Pour C++ :
Adafruit_NeoPixel.h
avr/power.h
Arduino.h
WiFi.h
WiFiMulti.h
HTTPClient.h
math.h
Adafruit_Sensor.h
DHT.h
DHT_U.h
ArduinoJson.h

Pour Node.js :
require('express');
require("mysql");
require('fs');
require('../joi_request_format');

Pour communiquer via le Wi-Fi, nous nous sommes servi de la libraire WiFi (https://www.arduino.cc/en/Reference/WiFi).
Pour communiquer avec l'API, nous avons utilisé la librairie HTTPClient (https://www.arduino.cc/reference/en/libraries/httpclient/) qui permet au microcontrolleur de faire des requêtes HTTP. Le microcontrôleur se connecte via un réseau Wi-Fi, et devient un client HTTP. Lors de requêtes POST ou pour obtenir les réponses, les données sont converties au format JSON grâce à la librairie ArduinoJson (https://arduinojson.org/), pour être utilisées dans l'application.

DIFFICULTÉS RENCONTRÉES

Nous avions essayé de réaliser l'API sous Java Springboot, mais nous avons finalement décidé de passer sur NodeJs à la suite de bug et de problèmes que nous n'arrivons pas à résoudre.

NODE JS ET CODE METIER

La technologie que nous avons utilisé pour faire l'API est Node.js qui est un environnement d'exécution permettant d'utiliser le JavaScript côté serveur. Grâce à son fonctionnement non bloquant, il permet de concevoir des applications en réseau performantes, telles qu'un serveur web, une API ou un job CRON. Créé en 2009 par Ryan Dahl, ce runtime JavaScript open source et cross-platform avait pour but premier de remédier aux limites de la programmation séquentielle et des serveurs web (tels qu'Apache HTTP Server). Ces technologies atteignaient effectivement leurs limites lorsqu'elles devaient gérer de très nombreuses connexions simultanées.

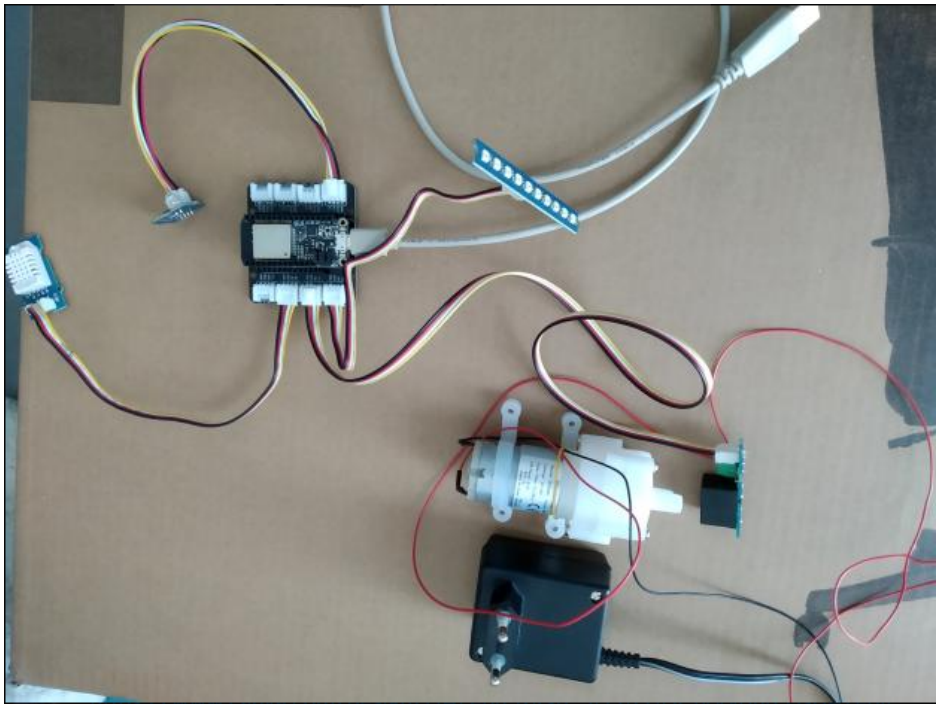
Le code métier au niveau du microcontrôleur se trouve dans le fichier client_http.ino.

Durant la phase de setup, le programme va se connecter au réseau Wi-Fi avec le nom du réseau et le mot de passe donné.

Durant la phase qui se répète en boucle :

- toutes les minutes, le programme fait une requête HTTP GET pour récupérer les seuils de déclenchements définis par l'utilisateur.
- toutes les secondes, le programme vérifie si la luminosité est inférieure au seuil défini. Si c'est le cas, le panneau de LEDS va s'allumer, et la requête POST /historique/add sera envoyée.
- toutes les secondes, le programme vérifie si la température et l'humidité sont inférieures au seuil défini. Si c'est le cas, la pompe va se mettre en marche, et la requête POST /historique/add sera envoyée.

PHOTO DU PRODUIT FINAL



Système d'éclairage et d'irrigation intelligent

Historique

Date	Température	Humidité	Luminosité	Durée de l'arrosage
29/05/2021	25°C	X	X	X

Paramètres

Température	Humidité	Luminosité	Durée de l'arrosage
25°C	X	X	X