Nicolas Soulard-Bouchard

Sébastien Jean

**TP3 : Station Météo**

Travail présenté à monsieur Alain Parent & Pierre-François Léon

Applications mobiles et objets connectés.

420-W46-SF

Département de la formation continue

Programmation, bases de données et serveurs

Cégep de Sainte-Foy

1er Mars 2021

Contexte

Dans le cadre du cours d’applications mobiles et objets connectés, nous avons reçu le mandat de concevoir une station qui permettra de prélever, et envoyer des données concernant diverses informations relatives à la météo et ce en temps réel. Les informations qui devront être collectés sont la température, la pression atmosphérique ainsi que le taux d’humidité dans l’air. La station devra être réalisée à partir d’un micro-contrôleur ESP32, et d’un module BME280.

La station en question devra être capable de se connecter à un réseau Wifi afin d’envoyer les données qui seront collectés dans un courtier de messages, pour ensuite être récupérées et utilisés par un logiciel de domotique quelconque.

Des appareils devront ensuite être capable d’utiliser ces informations via le logiciel de domotique, afin de performer certaines actions ou tâches que nous pourrons nous-mêmes choisir.

Planification & attribution des tâches

|  |  |  |
| --- | --- | --- |
| **Planification et registre des heures\*** | | |
| **Tâche** | **Nom du membre d’équipe** | **Durée de réalisation** |
| **Documentation** | | |
| Rédaction de la liste de tâches | Nicolas Soulard-Bouchard | 60 minutes |
| Rédaction page de présentation du document | Sébastien Jean | 15 minutes |
| Rédaction du contexte | Sébastien Jean | 30 minutes |
| Inventaire des pièces | Nicolas Soulard-Bouchard | 20 minutes |
| Réalisation du diagramme de classes | Sébastien Jean et Nicolas Soulard-Bouchard | 90 minutes |
| Réalisation et montage du vidéo explicatif | Sébastien Jean | 60 minutes |
| Consultation de la documentation | Nicolas Soulard-Bouchard et Sébastien Jean | Au total pour les deux 900 minutes |
| Écriture du manuel d’utilisation pour configurer la station météo | Nicolas Soulard-Bouchard | 120 minutes |
| Explication du schéma | Nicolas Soulard-Bouchard | 60 minutes |
| **Montage schéma et plan** | | |
| Schéma sur Fritzing | Nicolas Soulard-Bouchard | 45 minutes |
| Plan sur Fritzing | Nicolas Soulard-Bouchard | 45 minutes |
| **Programmation** | | |
| Coder classe Bouton | Sébastien Jean | 15 minutes |
| Coder classe LED | Sébastien Jean | 15 minutes |
| Coder classe Fenêtre | Sébastien Jean | 15 minutes |
| Coder classe TemoinsLumineux | Sébastien Jean | 30 minutes |
| Coder classe ControlesDuPanneau | Sébastien Jean | 30 minutes |
| Coder classe EvaluateurConditionsMeteo | Sébastien Jean | 45 minutes |
| Coder classe EtatPanneauDeControle | Sébastien Jean | 15 minutes |
| Coder classe PanneauControle | Sébastien Jean | 120 minutes |
| Coder classe ClientCourtierDeMessage | Sébastien Jean | 15 minutes |
| Coder classe GestionnaireDeWifi | Sébastien Jean | 120 minutes |
| Coder classe StationMeteo | Sébastien Jean | 30 minutes |
| Écriture du code procédurale | Nicolas Soulard-Bouchard | 120 minutes |
| Restructuration du code procédurale afin de découper en méthodes | Nicolas Soulard-Bouchard | 30 minutes |
| **Configuration des VM** | | |
| Installation et configuration de Home Assistance | Nicolas Soulard-Bouchard | 240 minutes \*\* |
| Installation et configuration de Home Assistance | Sébastien Jean | 30 minutes |
| Débuggage | Nicolas Soulard-Bouchard | 60 minutes |
| Débuggage et refactorisation | Sébastien Jean | 1800 minutes |
| \*Le registre des heures fait également office de liste de tâches et de registre des heures. Nous avons premièrement fait la liste de tâche et avons ajouté le temps nécessaire pour chacune d’elles lorsque celles-ci étaient complétés.  \*\* Après avoir essayé pendant environ 3 heures (seul et avec Alain Parent) à faire l’installation de Home Assistance sur un VMware Workstation Pro, j’ai finalement fait l’installation sur Oracle VM VirtualBox suite au conseil reçu de mes collègues de classe. | | |

Diagramme de classes

Inventaire des pièces

|  |  |
| --- | --- |
| **Inventaire des pièces** | |
| **Pièce** | **Quantité** |
| Microcontroleur ESP32 | 1 |
| Capteur BME280 | 1 |
| Bouton poussoir | 3 |
| Résistances 1000Ω | 3 |
| Résistances 220Ω | 3 |
| LED (1x Rouge, 1x Verte, 1x Bleu) | 3 |
| Platine d’essaie | 1 |
| Cables cavalier ( longueurs & couleurs variées) | 20 |

Consommation d’énergie

En ce qui concerne la consommation en énergie, nous avons deux composants principaux, soit le microcontrôleur et le capteur BME280. À cela s’ajoute les composants visuels qui sont des LED, des résistances et des boutons. En nous fiant à la documentation sur internet et fournit par les professeurs, nous avons

Calcul de l’intensité des résistances :

Pour celles de 220 ohm :

**U = 3.3 volts R = 220 ohm I=? I = U/R 🡪 I = 3.3/220 = 0,015 A ou 15 mA**

**P = U \* I 🡪 P = 3.3 \* 0.015 P= 0.0495 W**

Pour celles de 1000 ohm :

**U = 5 volts R = 1000 ohm I=? I = U/R 🡪 I = 5/1000 = 0,0050 A ou 5 mA**

**P = U \* I 🡪 P = 5 \* 0,0050 P= 0.025 W**

Calcul de la consommation par jour:

Nombre de DELs : 1

**1 \* 0.0495 = 0.0495 W 0.0495 \* 24 = 1.188 W/jour 0.0000495 kWh**

Nombre de bouton : 3

**3\*0.025 = 0.075 W 0.075\*24 = 1.8 W/jour 0.000075 kWh**

Nombre de BME280 : 1

Intensité : 1.0 mA = 0.001 A

**P = U \* I 🡪 P = 3.3 \* 0.001 P= 0.0033 W**

**1\*0.0033 = 0.0033 W 0.0033 \*24 = 0.013200 W/jour 0.0033 KWh**

Nombre d’Arduino Esp32 : 1

Pour une intensité de 160 mA = 0.160 A pour une tension de 3.3 V

**P = U \* I 🡪 P = 3.3 \* 0.160 P= 0.528 W**

**1\*0.528 = 0.528 W 0.528 \*24 = 12.672 W/jour 0.000528 KWh**

Pour une intensité de 260 mA = 0.0260 A pour une tension de 3.3 V

**P = U \* I 🡪 P = 3.3 \* 0.260 P= 0.858 W**

**1\*0.858 = 0.858 W 0.858 \*24 = 20.592 W/jour 0.000858 KWh**

Calcul de la consommation annuel:

Pour la DEL :

**((1.188 W/jour \* 365) / 1000) = 0.43362 KW / an**

Pour les boutons:

**((1.8 W/jour \* 365) / 1000) = 0.6570 KW / an**

Pour le capteur BME280 :

**(0.0132 W/jour \* 365) / 1000 = 0.004818 KW / an**

Pour le microcontrôleur d’Arduino Esp32 :

**Pour 3.3 V : (12.672 W/jour \* 365) / 1000 = 4.6253 KW / an**

**Pour 3.3 V : (20.592 W/jour \* 365) / 1000 = 7.5161 KW / an**

|  |  |  |  |
| --- | --- | --- | --- |
| **Consommation** | **Journalière (en Watt)** | **Mensuel**  **(en watt)** | **Annuel** |
| Pièce |  |  |  |
| LED | 1.188 |  |  |
| Bouton | 1.80 |  |  |
| BME280 | 0.0132 |  |  |
| ESP32 minimum | 12.672 |  |  |
| ESP32 maximum | 20.592 |  |  |
| Total minimum | 15.6732 | 470.196 |  |
| Total maximum | 23.5932 | 707.796 |  |

Schéma Technique

Manuel d’utilisation

Référence

<https://lastminuteengineers.com/bme280-esp32-weather-station/>

<https://randomnerdtutorials.com/cloud-weather-station-esp32-esp8266/>

<https://randomnerdtutorials.com/build-an-all-in-one-esp32-weather-station-shield/>

<https://custom-one.fr/station-meteo-qualite-de-lair-diy-arduino-mqtt/>

<https://forums.adafruit.com/viewtopic.php?f=19&t=138414>

<https://projetsdiy.fr/esp8266-dht22-mqtt-projet-objet-connecte/>

<https://projetsdiy.fr/esp32-test-librairie-wifimanager-gerer-connexions-wifi/>

<https://github.com/tzapu/WiFiManager>

<https://github.com/tzapu/WiFiManager/blob/master/examples/DEV/OnDemandConfigPortal/OnDemandConfigPortal.ino>

<https://www.instructables.com/ESP8266-and-ESP32-With-WiFiManager/>

<https://lastminuteengineers.com/creating-esp32-web-server-arduino-ide/>

<http://www.esp32learning.com/code/publishing-messages-to-mqtt-topic-using-an-esp32.php>

<https://iotdesignpro.com/projects/how-to-connect-esp32-mqtt-broker>

<https://randomnerdtutorials.com/esp32-mqtt-publish-subscribe-arduino-ide/>

<https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/protocols/mqtt.html>

<https://learn.sparkfun.com/tutorials/introduction-to-mqtt/all>

<https://techtutorialsx.com/2017/04/24/esp32-publishing-messages-to-mqtt-topic/>

<https://www.home-assistant.io/integrations/mqtt/>

<https://www.home-assistant.io/docs/configuration/devices/>

<https://www.home-assistant.io/getting-started/configuration/>

<https://www.home-assistant.io/docs/mqtt/broker>

<https://www.home-assistant.io/docs/configuration/yaml/>

<https://randomnerdtutorials.com/wifimanager-with-esp8266-autoconnect-custom-parameter-and-manage-your-ssid-and-password/>

<https://github.com/zhouhan0126/WIFIMANAGER-ESP32>

<https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/>

<https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/>

<https://randomnerdtutorials.com/esp32-mqtt-publish-bme280-arduino/>

<https://randomnerdtutorials.com/esp32-web-server-with-bme280-mini-weather-station/>

<https://github.com/bblanchon/ArduinoJson>

<https://projetsdiy.fr/esp32-code-arduino-sommeil-deep-sleep-reveils-timer-touch-pad-gpio/>

<https://letmeknow.fr/blog/category/wifi/>

<https://randomnerdtutorials.com/esp32-deep-sleep-arduino-ide-wake-up-sources/>

<https://phmarduino.wordpress.com/2018/07/18/station-de-mesures-v1/>

<https://gist.github.com/Schm1tz1/d5f4d34492509611846862cfdc786b66>

<https://forum.arduino.cc/index.php?topic=627063.0>

<https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/sleep_modes.html>

<https://diyi0t.com/reduce-the-esp32-power-consumption/>

<https://github.com/espressif/arduino-esp32/issues/882>

<https://randomnerdtutorials.com/esp32-timer-wake-up-deep-sleep/>

<https://lastminuteengineers.com/datasheets/esp32-datasheet-en.pdf>

<http://empa.com/empaiot/esp-wroom-32d_esp32-wroom-32u_datasheet_en.pdf>

<https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/sleep_modes.html>

<https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf>

<https://github.com/tzapu/WiFiManager/issues/1050>

<https://github.com/tzapu/WiFiManager/tree/master/examples>

<https://arduinojson.org/>

https://projetsdiy.fr/esp8266-webserveur-partie4-arduinojson-charger-enregistrer-fichiers-spiffs/

<https://byfeel.info/eeprom-ou-spiffs/>

<https://arduinojson.org/v6/example/config/>

Reste à faire

Consommation d’énergie Nicolas

<https://abra-electronics.com/sensors/sensors-temperature-en/sens-64-5v-bme280-i2c-pressure-humidity-temperature-sensor-5vdc.html?sl=fr>

* Consommation: 3,6 µA (0,1µA en veille)

<https://www.learnrobotics.org/blog/bme280-arduino-tutorial/>

<https://lastminuteengineers.com/bme280-arduino-tutorial/>

## Puissance requise

Le module est livré avec un régulateur embarqué [LM6206](https://www.mikrocontroller.net/attachment/193855/LM6206N3.pdf) 3,3 [V](https://www.mikrocontroller.net/attachment/193855/LM6206N3.pdf) et un traducteur de niveau de tension I2C, vous pouvez donc l'utiliser avec un microcontrôleur logique 3,3 V ou 5 V comme Arduino sans souci.

Le BME280 consomme moins de 1 mA pendant les mesures et seulement 5 μA au repos. Cette faible consommation d'énergie permet la mise en œuvre dans des appareils alimentés par batterie tels que des combinés, des modules GPS ou des montres.

MicroControlleur

<https://www.universal-solder.ca/product/espduino-32-esp32-wemos-d1-r32-wifi-bluetooth-ble-dual-core/>

Le courant de veille de la puce ESP32 est inférieur à 5 µA, ce qui la rend adaptée aux applications électroniques alimentées par batterie et portables.

Manuel d’utilisation Nicolas

<http://emery.claude.free.fr/esp32-wifimanager.html>

Lundi 1 mars de 12h00 à 16h00 local A304