

BE Systèmes Embarqués

Conception et Développement d'une Application en Temps Réel
avec Gestion Multi-Tâches.



Réalisé par :

Ahmed ABDELJELIL

Wissal ABID

3AGE1

Supervisé par :

M. Lotfi CHARAABI

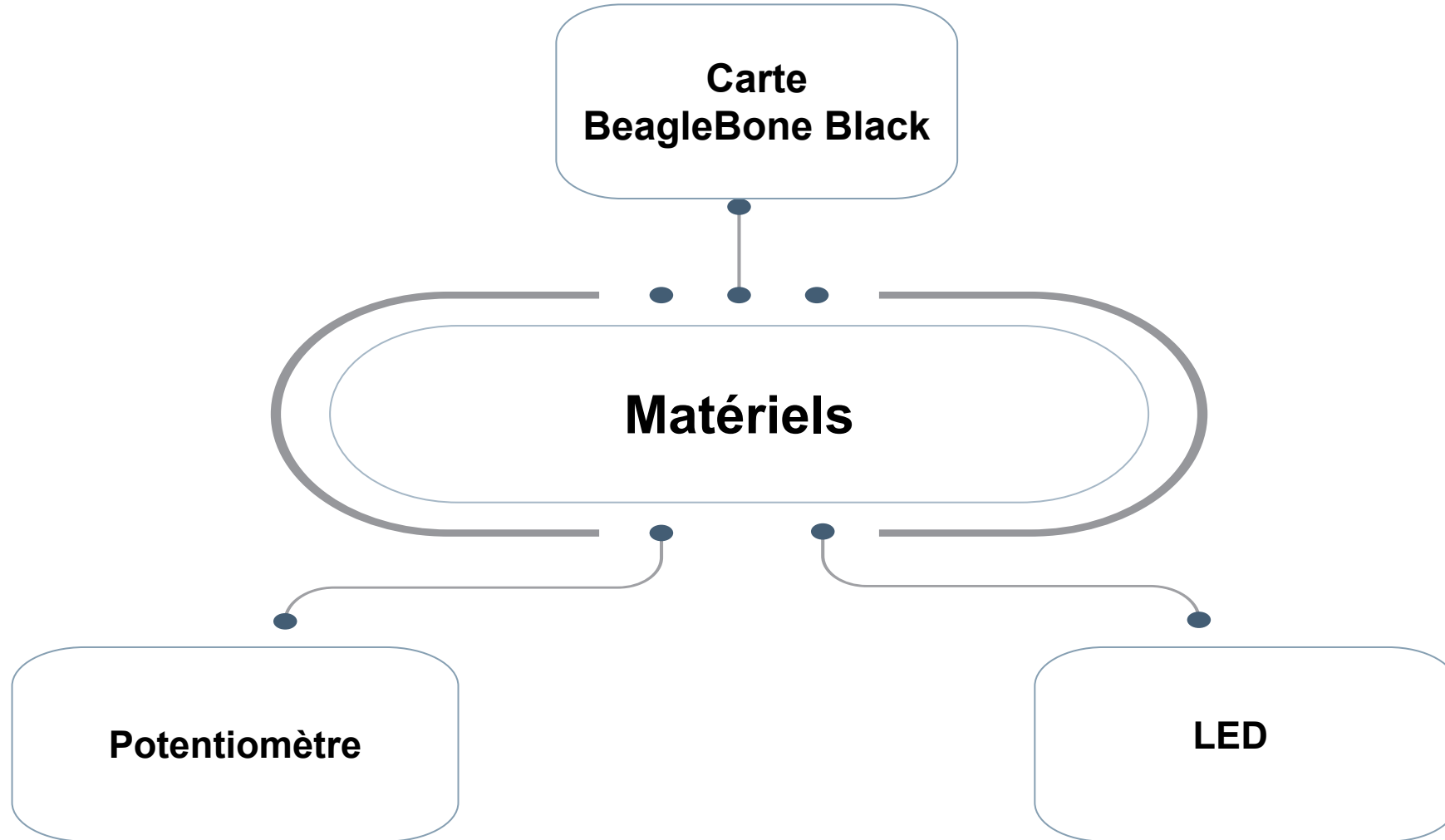
Plan

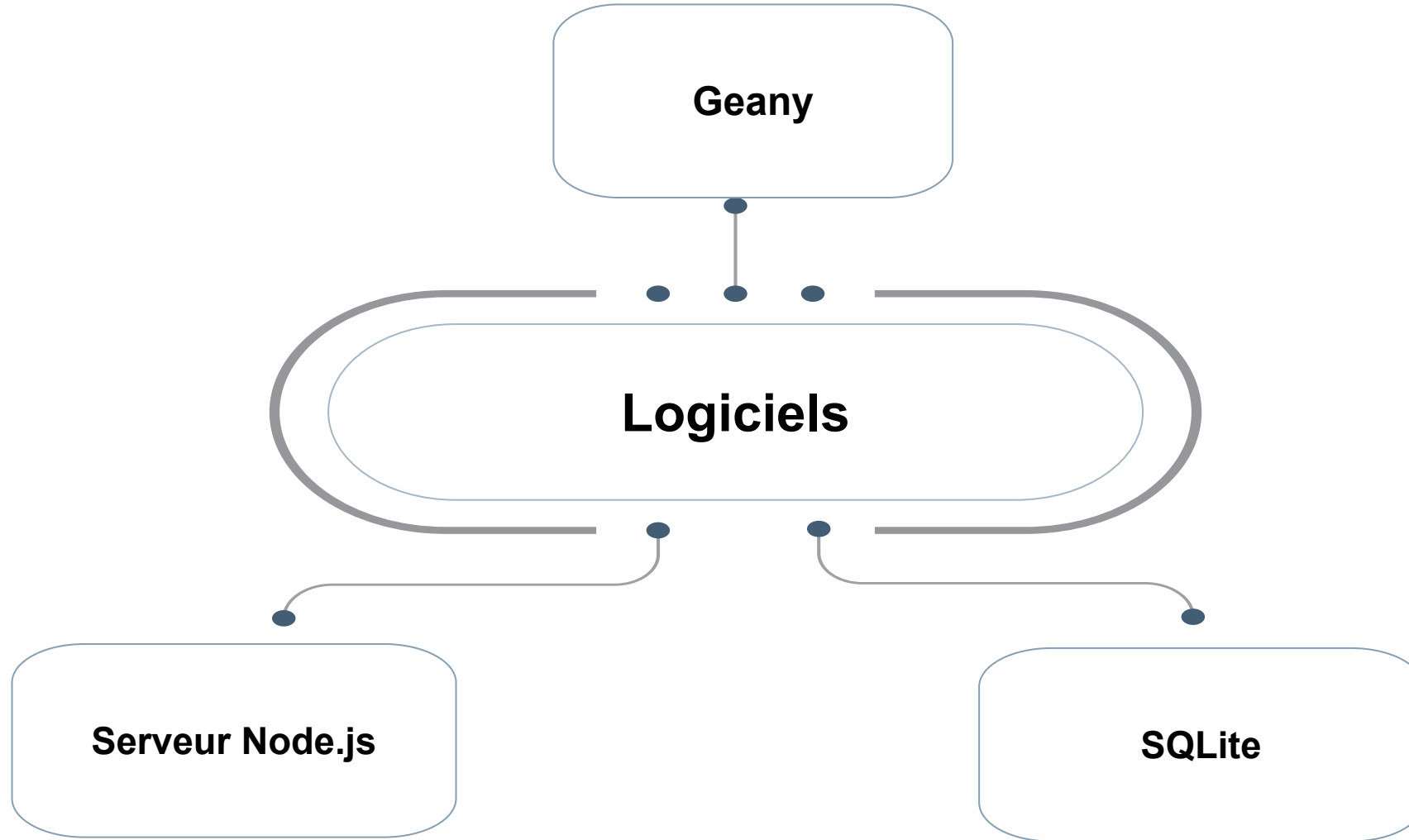
- Introduction
- Matériels
- Logiciels
- Contrôle d'une LED en Temps Réel
- Acquisition de Données en Temps Réel
- Contrôle de LED avec timer sur la BeagleBone Black(BBB) en C.
- Application Multithread avec Acquisition ADC, Contrôle PWM et Création de la base de données.
- Conclusion

Introduction

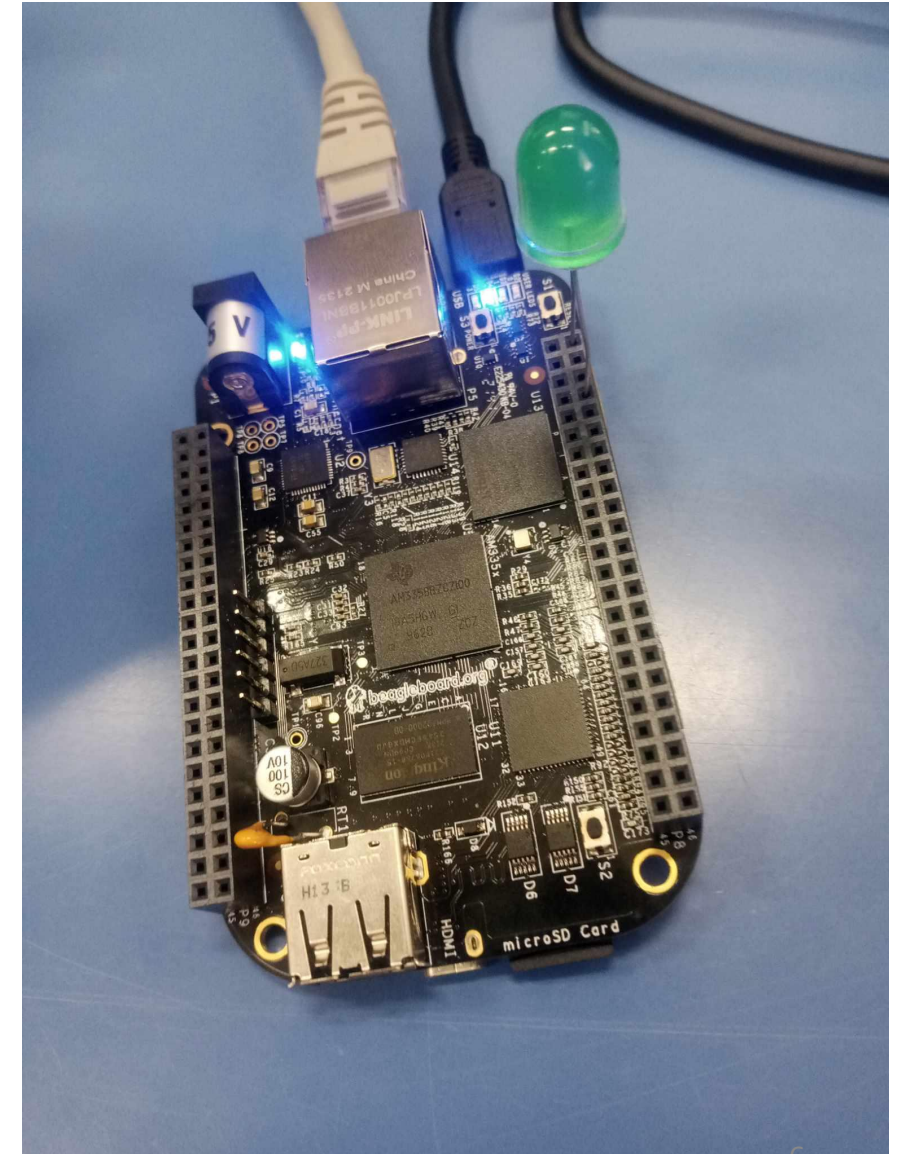
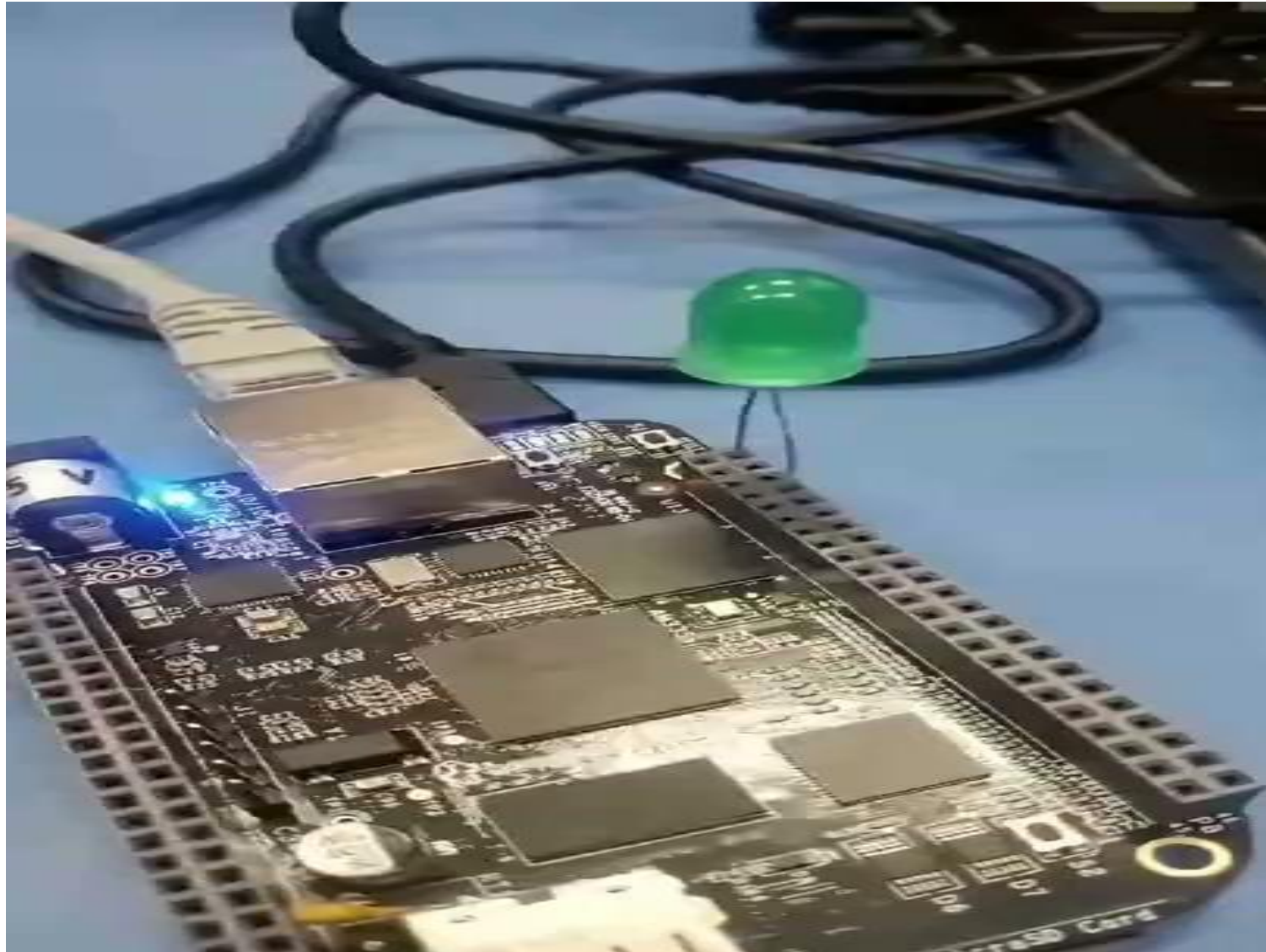
Notre projet se focalise sur la BeagleBone Black (BBB), une plateforme embarquée, pour créer une application innovante.

Notre objectif central est d'établir une communication en temps réel entre plusieurs threads, intégrant le contrôle de périphériques, l'acquisition de données analogiques, et la gestion d'une base de données. À travers cette aventure, nous explorons les capacités avancées de la BBB, démontrant notre engagement envers l'innovation dans le domaine des systèmes embarqués.

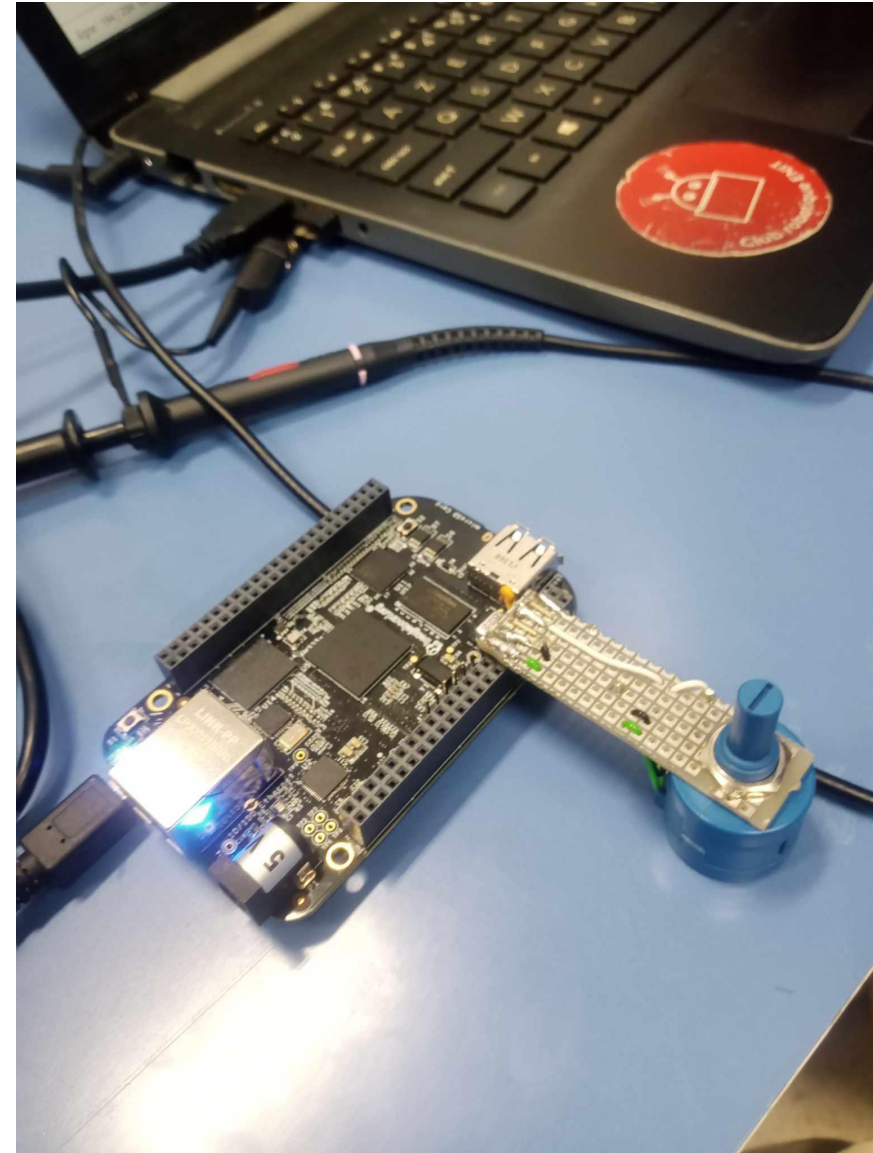
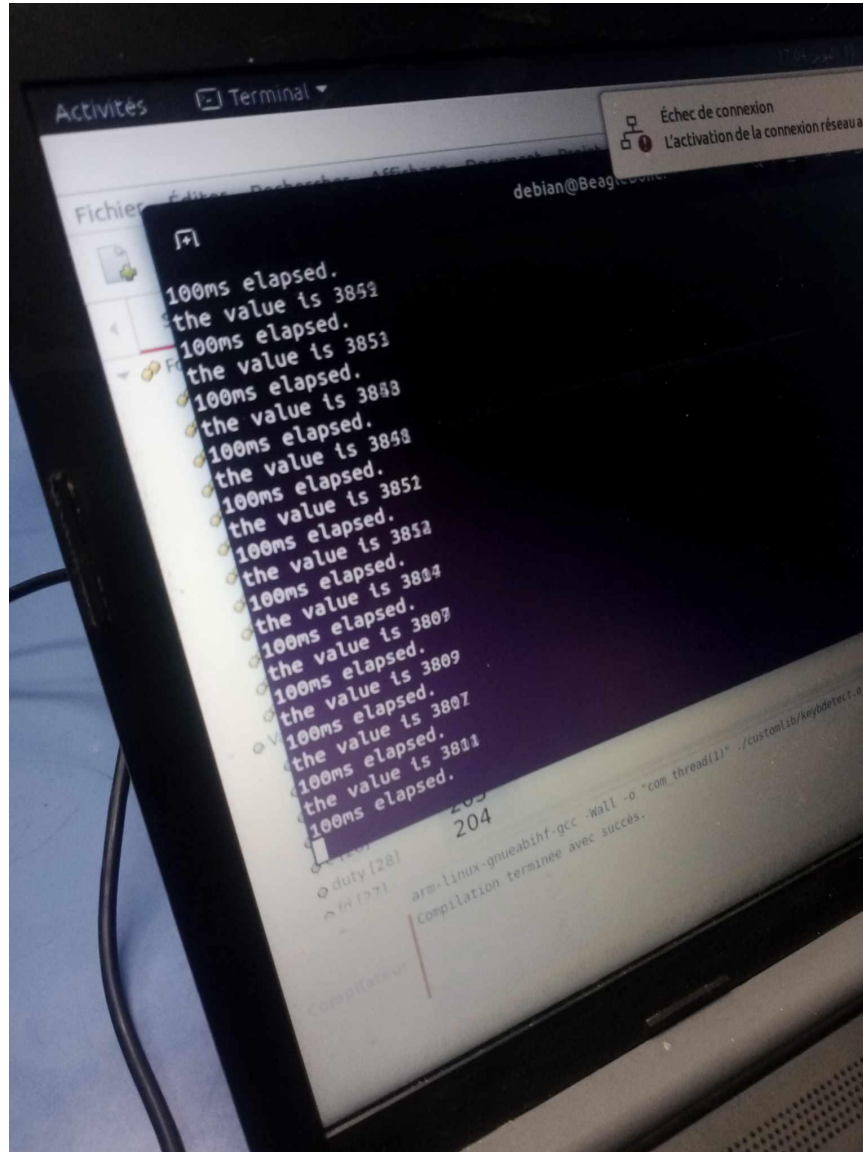




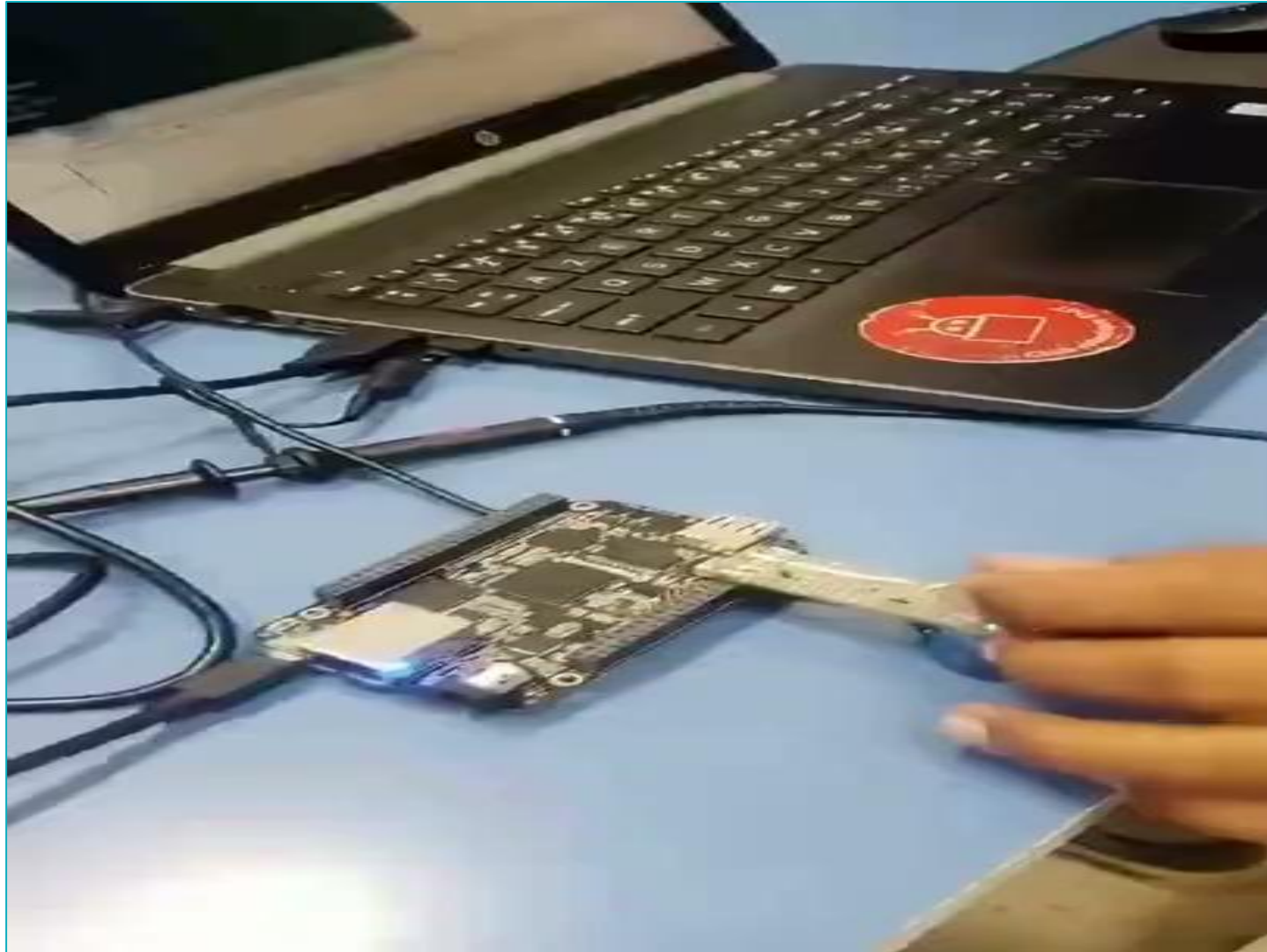
Contrôle d'une LED en Temps Réel



Acquisition de données en Temps Réel



Acquisition de Données en Temps Réel avec un Capteur Analogique



Application Multithread avec Acquisition ADC, Contrôle PWM

- **Déclarations et Initialisations :**

Le code déclare différentes variables, structures et pointeurs, notamment pour la gestion des threads, des timers, des GPIOs, et des paramètres du serveur.

- **Fonctions pour l'initialisation du timer et du PWM :**

- init_timer: Initialise un timer en utilisant les fonctions POSIX pour les threads et les timers.
- init_pwm: Initialise le PWM en écrivant dans les fichiers correspondants dans le système de fichiers de la BeagleBone Black.

- **Threads :** Trois threads sont créés pour différentes tâches :

- server_func: Gère la fonctionnalité du serveur en écoutant les connexions des clients, recevant des données de la base de données SQLite et les envoyant au client
- thread_adc: Lit la valeur ADC à partir du fichier "adc5/in_voltage3_raw" et l'insère dans la base de données SQLite.
- thread_pwm: Calcule la valeur du cycle de service PWM basée sur la valeur ADC et écrit cette valeur dans le fichier "pwm/duty_cycle".

- **Fonctions pour la gestion de la base de données SQLite :**

- db_open: Ouvre ou crée une base de données SQLite.
- db_create_table: Crée une table dans la base de données pour stocker les paramètres de l'énergie
- db_insert: Insère des valeurs dans la table de la base de données.
- db_read: Lit les valeurs d'une ligne spécifiée dans la table de la base de données.

- **Fonctions pour la gestion du GPIO :**

Le code initialise et contrôle un GPIO en utilisant la bibliothèque gpiod.h.

- **Boucle Principale :**

La boucle principale utilise une fonction (kbhit()) pour détecter l'appui sur la touche "q" et maintient le programme en cours d'exécution jusqu'à ce que l'utilisateur appuie sur "q".

Commandes nécessaires pour exécuter le code sur carte BBB(compile cross)

Définir les commandes de construction

#	Étiquette	Commande	Dossier de travail	Remettre à zéro
Commandes pour C				
1.	Compile	arm-linux-gnueabihf-gcc -l		
2.	Build	arm-linux-gnueabihf-gcc -l		
3.	Lint	cppcheck --language=c -e		
Expression régulière pour les erreurs :				
Commandes indépendantes				
1.	Make	make		
2.	Make Custom Target...	make		
3.	Make Object	make %e.o		
4.				
Expression régulière pour les erreurs :				
Note : le deuxième élément ouvre une boîte de dialogue et ajoute la réponse à la commande.				
Commandes d'exécution				
1.	Execute	sshpass -p "temppwd" scp		
2.				

%d, %e, %f, %p et %l sont remplacés dans les dossiers et les commandes, voir le manuel pour plus de détails.

AnnulerValider

Compile: arm-linux-gnueabihf-gcc -Wall -c "%f"
-I./libgpiod/rootfs/include -I
./sqlite-autoconf-3160200

Build:arm-linux-gnueabihf-gcc -Wall -o "%e"
./customlib/keybdetect.o "%f"
-I./libgpiod/rootfs/include
-L./libgpiod/rootfs/lib -lgpiod -lpthread -lrt -l
./sqlite-autoconf-3160200 -L
./sqlite-autoconf-3160200/.libs -lsqlite3

Site :
<https://medium.com/geekculture/start-using-the-sqlite-database-while-programming-with-c-on-beaglebone-black-arm-and-embedded-809e5eea2a21>

```
debian@BeagleBone: ~  
debian@BeagleBone:~$ ls  
adc5  adc_pwm_pwm  pwmmux  
debian@BeagleBone:~$
```

Voici le fichier à été bien
générer ensuite nous avons
exécuter pour voir les valeurs
d'adc et pwm

```
debian@BeagleBone: ~  
THREAD SERVEUR : IP programmée = 127.0.0.0  
THREAD SERVEUR : Numéro de port programmé = 1880  
THREAD SERVEUR : Nombre maximal de clients pris en charge dans la file d'attente = 5  
THREAD SERVEUR : Socket creee avec succès  
THREAD SERVEUR : Socket liée avec succès  
THREAD SERVEUR : La socket écoute les clients maintenant  
@=> Database values inserted successfully!!  
the value duty is 99  
the value ADC is 4092  
100ms elapsed.  
@=> Database values inserted successfully!!  
the value duty is 99  
the value ADC is 4093  
100ms elapsed.  
@=> Database values inserted successfully!!  
the value duty is 99  
the value ADC is 4092  
100ms elapsed.  
@=> Database values inserted successfully!!  
the value duty is 99  
the value ADC is 4091  
100ms elapsed.  
@=> Database values inserted successfully!!  
the value duty is 99  
the value ADC is 4093  
100ms elapsed.  
@=> Database values inserted successfully!!  
the value duty is 99  
the value ADC is 4094  
100ms elapsed.  
@=> Database values inserted successfully!!  
the value duty is 99  
the value ADC is 4093  
100ms elapsed.  
@=> Database values inserted successfully!!  
the value duty is 99  
the value ADC is 3559  
100ms elapsed.  
@=> Database values inserted successfully!!
```

Nous remarquons deux choses qui ont été générées par notre code : un serveur dont l'objectif est d'afficher les valeurs de l'ADC qui seront stockées dans la base de données.

```
debian@BeagleBone: ~  
debian@BeagleBone:~$ ls  
ADC.db  adc5  adc_pwm  pwm  pwmmux  
debian@BeagleBone:~$
```

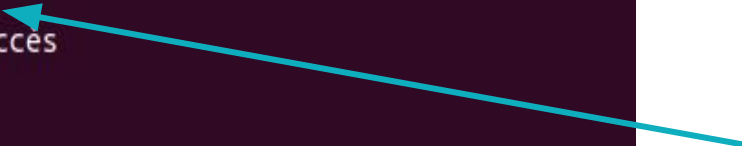
Voici la base de données qui a été configurée correctement.
Nous pouvons la déplacer pour le traitement.
Pour ce faire, nous avons essayé de créer un site web à l'extérieur du serveur BBB en utilisant Node.js .


```
debian@BeagleBone: ~
ahmed@ahmed-idoumou-HP-Laptop-15-da2xxx:~/node-jquery-app$ ls
ADC.db  app.js  node_modules  package-lock.json  public  views
ahmed@ahmed-idoumou-HP-Laptop-15-da2xxx:~/node-jquery-app$
```

```
ahmed@ahmed-idoumou-HP-Laptop-15-da2xxx:~/node-jquery-app$ cd views
ahmed@ahmed-idoumou-HP-Laptop-15-da2xxx:~/node-jquery-app/views$ ls
adc.ejs
ahmed@ahmed-idoumou-HP-Laptop-15-da2xxx:~/node-jquery-app/views$
ahmed@ahmed-idoumou-HP-Laptop-15-da2xxx:~/node-jquery-app$ cd public
ahmed@ahmed-idoumou-HP-Laptop-15-da2xxx:~/node-jquery-app/public$ ls
chart.html  chart.js  style.css
ahmed@ahmed-idoumou-HP-Laptop-15-da2xxx:~/node-jquery-app/public$
```

Après l'installation de Node.js, nous avons mis en place deux répertoires : "public" qui contient des ressources statiques comme des fichiers CSS et JavaScript, et "views" destiné aux modèles utilisés par l'application. Enfin, nous avons créé un fichier "app.js" qui sert de point d'entrée de l'application, et c'est celui que nous exécuterons pour démarrer le serveur.

```
ahmed@ahmed-idoumou-HP-Laptop-15-da2xxx:~/node-jquery-app$ ls
ADC.db  app.js  node_modules  package-lock.json  public  views
ahmed@ahmed-idoumou-HP-Laptop-15-da2xxx:~/node-jquery-app$ node app.js
node-pre-gyp info This Node instance does not support builds for Node-API version 6
node-pre-gyp info This Node instance does not support builds for Node-API version 6
Serveur en cours d'exécution sur le port 1880
Connexion à la base de données établie avec succès
```

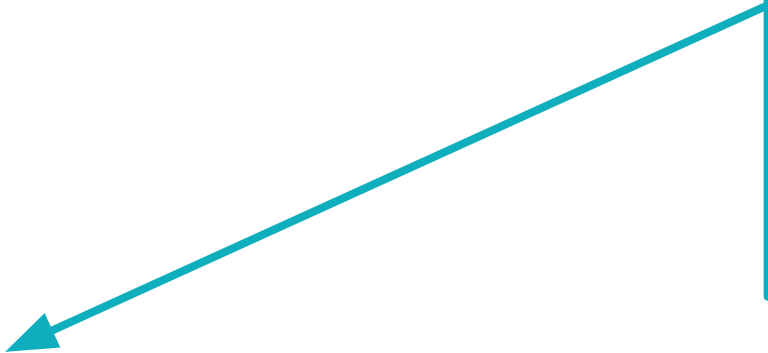


Pour lancer l'application, exécutez la commande "node app.js". Cela démarrera le serveur, et vous pourrez voir le numéro de port sur lequel l'application est en cours d'exécution.

Tableau de données ADC

ID ADC	Time
1 2853	02/01/2024 23:39:07
2 2961	02/01/2024 23:39:07
3 2960	02/01/2024 23:39:07
4 2959	02/01/2024 23:39:08
5 2960	02/01/2024 23:39:08
6 2979	02/01/2024 23:39:08
7 3073	02/01/2024 23:39:08
8 3225	02/01/2024 23:39:08
9 3465	02/01/2024 23:39:08
10 3823	02/01/2024 23:39:08
11 4092	02/01/2024 23:39:08
12 4090	02/01/2024 23:39:08
13 4094	02/01/2024 23:39:08
14 4093	02/01/2024 23:39:09
15 4094	02/01/2024 23:39:09
16 4092	02/01/2024 23:39:09
17 4094	02/01/2024 23:39:09
18 4094	02/01/2024 23:39:09
19 4092	02/01/2024 23:39:09
20 4094	02/01/2024 23:39:09
21 4094	02/01/2024 23:39:09
22 4093	02/01/2024 23:39:09

Ainsi, pour confirmer la configuration correcte de la base de données, nous utilisons la base de données générée dans notre projet.



Conclusion

- En résumé, nous avons établi une communication en temps réel entre plusieurs threads, puis nous avons stocké les valeurs de l'ADC dans une base de données pour pouvoir les utiliser ultérieurement.
- L'acquis principal a été la connexion au serveur BBB (BeagleBone Black).



**MERCI
POUR VOTRE ATTENTION**