Cours- Plateformes de développement de l'IOT : Cloud et OS, Communication logicielle

TP3 : TP Linux embarqué et temps réels

Compilation du kernel

Nous avons commencé (au bout de 3 essais) par dézipper le fichier « Noyau.zip » via 7zip. Suite à cela, nous avons copier, renommer le fichier « config-linux-qemu » en « .config », patcher, configurer (changement du cross-compiler tool prefix) puis compiler le noyau.

Commandes	Résultat
7z x linux-rpi-3.10.y.zip	Décompression de l'archive
mv config-linux-qemu .config	Renomme le fichier
patch -p1 -d linux-rpi-3.10.y / < linux-arm.patch	Patch à la racine
make ARCH=arm oldconfig	Configure la compilation du noyau
make ARCH=arm menuconfig	Ouvre un menu de configuration Kernel
make ARCH=arm	Compile le noyau

Test du noyau en simulateur

Pour cela nous avons utilisé QEMU afin de lancer l'émulation du noyau. Nous avons pu constater une erreur venant d'un fichier de programmation c : « do_mount.c »

Commandes	Résultat
qemu-system-arm -kernel ./arch/arm/boot/zImage -cpu arm1176 -m 256 -M versatilepb -no-reboot	Lancement de l'émulateur QEMU

TP 4 : TP photorésistance

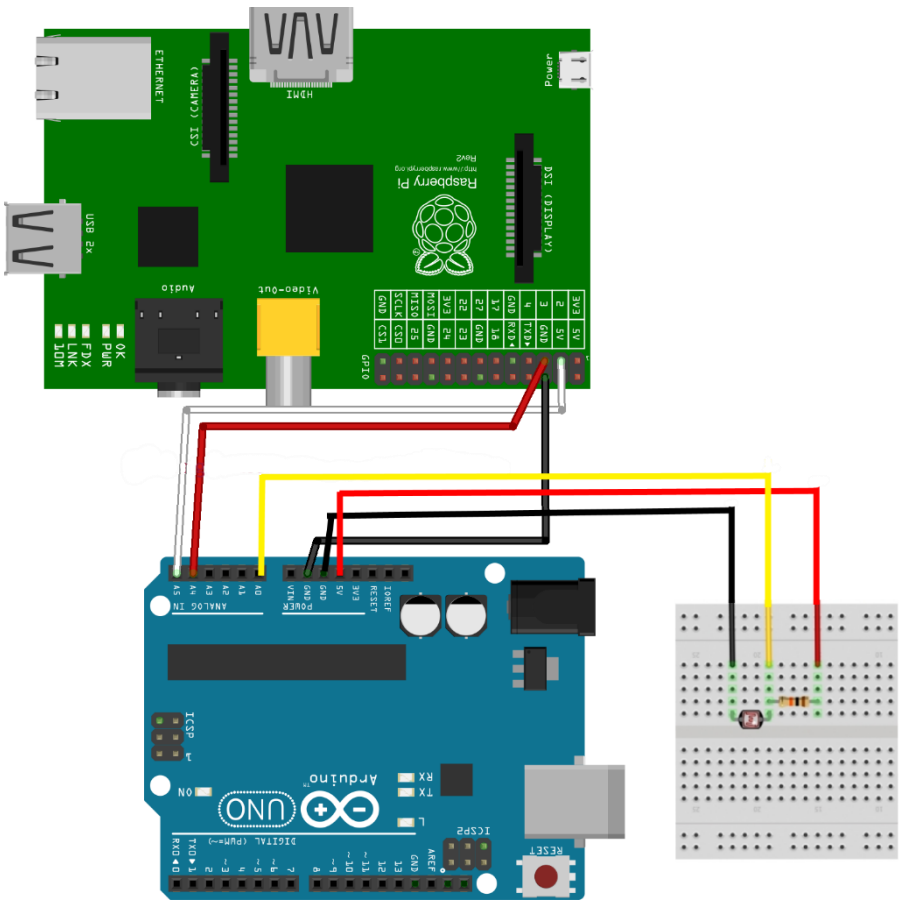
Objectif

Nous devons récupérer la valeur lumineuse d'une photorésistance afin de l'afficher sur une application mobile.

Méthode de résolution

Pour cela nous avons eu besoin d'une photorésistance, une bread board, une carte Arduino Uno, une Raspberry Pi et un smartphone. La photorésistance récupère la valeur lumineuse ambiante, la Raspberry Pi récupère cette valeur toutes les 30 secondes via la Arduino puis envoi cette valeur sur une base de données Firebase. Le smartphone récupère les 10 dernières valeurs enregistrer dans la base de données puis les affiches sous forme de graphique avec la date et l'heure du dernière enregistrement.

Schéma du montage



Code Arduino Uno

```
#include <Wire.h>

#define SLAVE_ADDRESS 0x12

int diode = 0;
unsigned int a;

void setup()
{
  Serial.begin(9600); // start serial for output

  // initialise la connexion i2c
  Wire.begin(SLAVE_ADDRESS);

  // Utilise la méthode d'envoi des données
  Wire.onRequest(sendData);
  Serial.println("Communication start!");
}

void loop()
{
  delay(100);
}

void sendData()
{
  a = analogRead(diode)/4; // Lis la valeur et la divise par 4 pour que a < 255
  Wire.write(a); // Envoi les données en i2c
  Serial.println("Data sent");
}
```

Script Python Raspberry Pi

```
Import smbus
Import time
Import datetime
Import json
From firebase import firebase

bus = smbus.SMbus(1)
DEVICE = 0x12

firebase = firebase.FirebaseApplication("https://datatransfersensor.firebaseio.com", None)

rd=bus.read_byte(DEVICE)
laDate = datetime.datetime.now().strftime("%d-%m-%Y %H :%M :%S")
If int(rd) >= 0 :
        Firebase.post(/data, {"photocell" : int(rd), "date" : laDate}
```

Ce script python récupère les valeurs transmises en i2c puis les enregistres en BDD. Il est appelé toutes les 30 secondes grâce au crontab.

Rendu Android