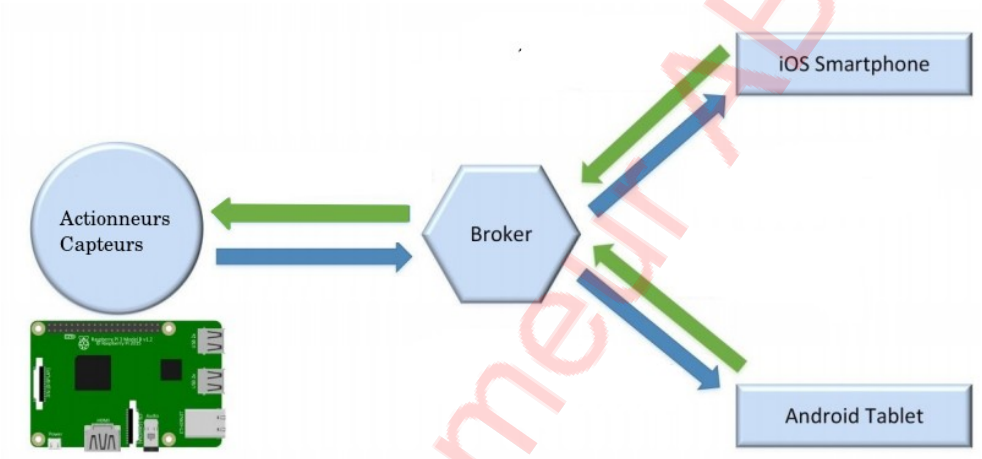
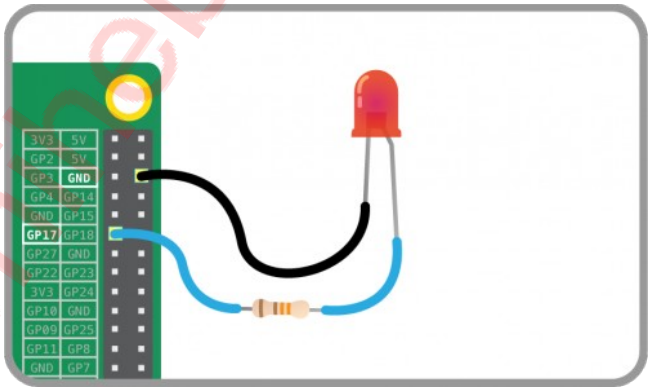
TP Corrigé
Développement des objets connectés

Environnement matériel : Raspberry Pi 3
Environnement de développement : Eclipse IDE
Langage de programmation : C++17
Bibliothèques : pigpio et PAHO MQTT C++

On se propose de réaliser un objet connecté (nœud) sur la RPi pour commander une diode LED à l'aide des messages MQTT. Un message START permet de démarrer le clignotement de la diode, le message STOP arrête le clignotement et le message EXIT pour arrêter le programme.



Le montage de la connexion de la diode avec la raspberry Pi 3 est le suivant :



On utilise la bibliothèque pigpio et l'API C++ du projet PAHO MQTT.

Pour effectuer la compilation croisée avec Eclipse IDE, il est nécessaire de spécifier au compilateur le chemin $ROOTFS/ ($ROOTFS doit être remplacé par le chemin absolu du votre répertoire rootfs utilisé contenant une copie des fichiers bibliothèques installés sur la RPi).

Pour l'éditeur de liens, il faut utiliser les bibliothèques suivantes :

```
paho-mqttpp3
pigpio
crypto
pthread
ssl
paho-mqtt3a
paho-mqtt3as
```

Les emplacements à passer à l'éditeur de liens `(Library search paths (-L))` sont :

```
$ROOTFS/usr/local/lib
/opt/cross-pi-gcc/arm-linux-gnueabihf/lib
$ROOTFS/usr/lib/arm-linux-gnueabihf
```

L'idée de base du programme proposé est de créer deux threads. Le premier est utilisé pour recevoir les messages MQTT. Tandis que le deuxième est utilisé pour clignoter la diode lorsque le clignotement est activé.

```
#include "mqtt/async_client.h"
#include "mqtt/client.h"
#include "pigpio.h"
#include <iostream>
#include <string>
#include <cstring>
#include <thread>

using namespace std;

const string SERVER_ADDRESS { "tcp://Your broker ip:1883" };
const string CLIENT_ID { "user1" };
const string TOPIC { "controlLED" };
const int QOS = 1;
const int O_LED = 17;
////////////////////////////////////////////////////////////////////////////
volatile bool flash = false;
void flasher();
int main(int argc, char *argv[]) {
	if (gpioInitialise() < 0) {
		std::cerr << "Can't initialize pigpio..." << std::endl;
		exit(-1);
	}
	gpioSetMode(O_LED, PI_OUTPUT);
	std::thread myTh(flasher);
	mqtt::async_client cli(SERVER_ADDRESS, CLIENT_ID);
```

```
	try {
		cli.start_consuming();
		auto rsp = cli.connect()->get_connect_response();
		if (!rsp.is_session_present())
			cli.subscribe(TOPIC, QOS)->wait();
	} catch (const mqtt::exception &exc) {
		cerr << "\n  " << exc << endl;
		return 1;
	}
	while (true) { // Loop to consume read messages
		auto msg = cli.consume_message();
		if (!msg)
			break;
		if (msg->get_topic() == "controlLED") {
			if (msg->to_string() == "START")
				flash = true;
			else if (msg->to_string() == "STOP")
				flash = false;
			else if (msg->to_string() == "EXIT")
				break;
		}
	}

	try {
		if (cli.is_connected()) {
			cli.unsubscribe(TOPIC)->wait();
			cli.stop_consuming();
			cli.disconnect()->wait();
		}
	} catch (const mqtt::exception &exc) {
		cerr << "\n  " << exc << endl;
		return 1;
	}

	return 0;
}

/**
 * Flash the led
 **/
void flasher() {
	while (1) {
		while (flash) {
			gpioWrite(O_LED, !gpioRead(O_LED));
			std::this_thread::sleep_for(1s);
		}
	}
}
```