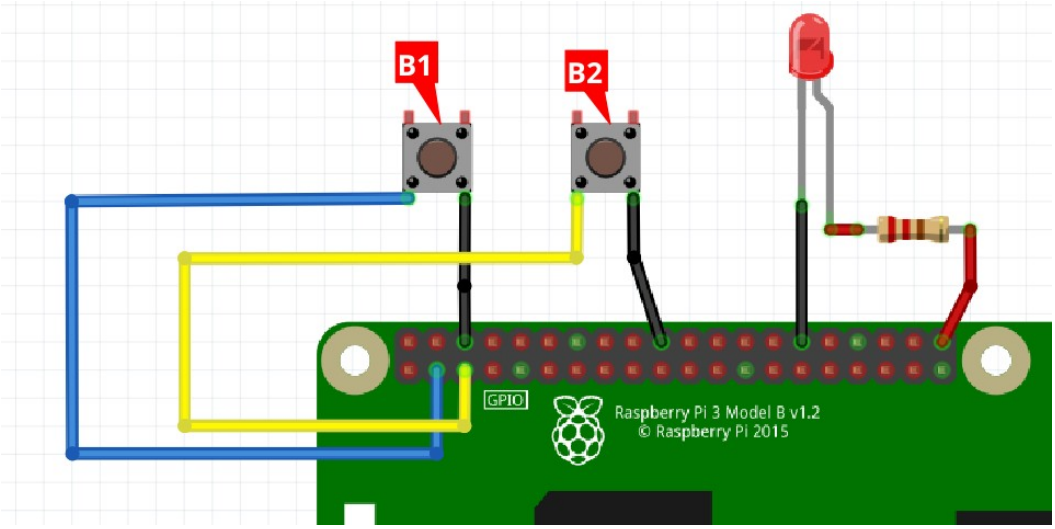
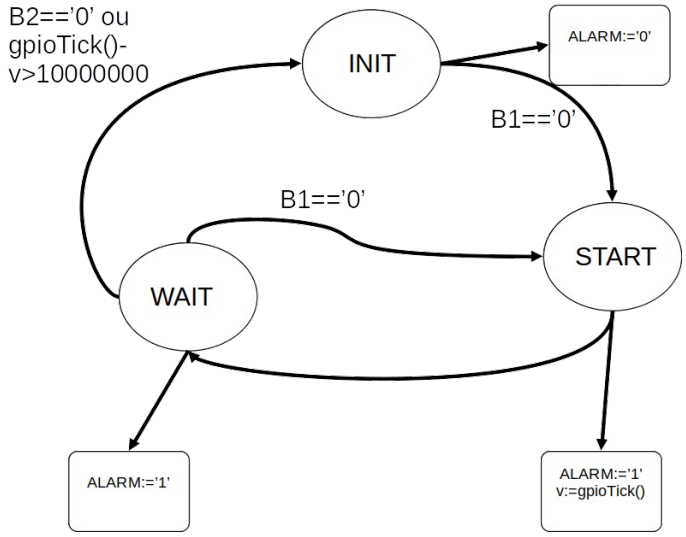
Université Tunis El-Manar	Année Universitaire : 2021-2022
Faculté des Sciences de Tunis	Module : Conception des objets connectés
Section : LIOT3	Enseignant : C.A. ABID

Exercice 1. (correction)
On se propose de simuler un système d'alarme dont le montage est présenté comme suit :



L'alarme se déclenche lorsque le bouton poussoir B1 est appuyé. Cela se fait en flashant la diode LED en sortie pendant 10 secondes. Si le bouton B1 est appuyé encore une fois, la durée de l'alarme est ré-initialisée. Par contre, si le bouton B2 est appuyé, l'alarme s'arrête immédiatement.

1) Donner la machine à états modélisant le système d'alarme.



2) Écrire le programme permettant d'implémenter ce système d'alarme.

```
#include <unistd.h>
#include <pigpio.h>

constexpr uint8_t B1=2; // B1 connecté à BCM2
constexpr uint8_t B2=3; // B1 connecté à BCM3
constexpr uint8_t ALARM=21; // diode LED connecté à BCM21

int main(int argc, char *argv[]) {
	gpioInitialise();
	gpioSetMode(B1, PI_INPUT);
	gpioSetMode(B2,PI_INPUT);
	gpioSetPullUpDown(B1,PI_PUD_UP);
	gpioSetPullUpDown(B2,PI_PUD_UP);

	gpioSetMode(ALARM,PI_OUTPUT);
	gpioWrite(ALARM,0);
	enum class state {INIT,START,WAIT};
	state etat=state::INIT;
	uint32_t v;
	while (1) {
		switch(etat) {
		case state::INIT:
			gpioWrite(ALARM,0);
			if (gpioRead(B1)==0) {
				while (gpioRead(B1)==0);
				etat=state::START;
			}
			break;
		case state::START:
			gpioWrite(ALARM,1);
			v=gpioTick();
			etat=state::WAIT;
			break;
		case state::WAIT:
			gpioWrite(ALARM,1);
			if (gpioRead(B1)==0) {
				while (gpioRead(B1)==0);
				etat=state::START;
			}
			else if (gpioRead(B2)==0 || gpioTick()-v>10000000) {
				while (gpioRead(B2)==0);
				etat=state::INIT;
			}
			break;
		}
	}
	gpioTerminate();
	return 0;
}
```

3) Reprendre le même programme en utilisant les interruptions pour la la vérification de l'état des boutons poussoirs.

```
#include <unistd.h>
#include <pigpio.h>

constexpr uint8_t B1=2; // B1 connecté à BCM2
constexpr uint8_t B2=3; // B1 connecté à BCM3
constexpr uint8_t ALARM=21; // diode LED connecté à BCM21
volatile bool B1_evt=false,B2_evt=false;
void ISR(int gpio, int level, uint32_t   tick) {
      if (gpio==B1) B1_evt=true;
      else if (gpio==B2) B2_evt=true;
}
int main(int argc, char *argv[]) {
      gpioInitialise();
      gpioSetMode(B1, PI_INPUT);
      gpioSetMode(B2,PI_INPUT);
      gpioSetPullUpDown(B1,PI_PUD_UP);
      gpioSetPullUpDown(B2,PI_PUD_UP);
      gpioSetISRFunc(B1,FALLING_EDGE,-1,ISR);
      gpioSetISRFunc(B2,FALLING_EDGE,-1,ISR);
      gpioSetMode(ALARM,PI_OUTPUT);
      gpioWrite(ALARM,0);
      enum class state {INIT,START,WAIT};
      state etat=state::INIT;
      uint32_t v;
      while (1) {
            switch(etat) {
            case state::INIT:
                  gpioWrite(ALARM,0);
                  if (B1_evt) {
                        etat=state::START;
                  }
                  B1_evt=B2_evt=false;
                  break;
            case state::START:
                  gpioWrite(ALARM,1);
                  v=gpioTick();
                  etat=state::WAIT;
                  B1_evt=B2_evt=false;
                  break;
            case state::WAIT:
                  gpioWrite(ALARM,1);
                  if (B1_evt) {
                        etat=state::START;
                  }
                  else if (B2_evt || gpioTick()-v>10000000) {
                        etat=state::INIT;
                  }
                  B1_evt=B2_evt=false;
                  break;
            }
      }
      gpioTerminate();
      return 0;
}
```

4) Réécrire le programme en autorisant l'arrêt du système d'alarme suite à la réception du message « Stop » dans le topic « Alarme »

```
#include "mqtt/async_client.h"
#include <unistd.h>
#include <pigpio.h>
#include <string>
const std::string SERVER_ADDRESS { "tcp://Your broker IP" };
const std::string CLIENT_ID { "user1" };

constexpr uint8_t B1 = 2; // B1 connecté à BCM2
constexpr uint8_t B2 = 3; // B1 connecté à BCM3
constexpr uint8_t ALARM = 21; // diode LED connecté à BCM21
volatile bool B1_evt = false, B2_evt = false;
void ISR(int gpio, int level, uint32_t tick) {
	if (gpio == B1)
		B1_evt = true;
	else if (gpio == B2)
		B2_evt = true;
}
int main(int argc, char *argv[]) {
	gpioInitialise();
	gpioSetMode(B1, PI_INPUT);
	gpioSetMode(B2, PI_INPUT);
	gpioSetPullUpDown(B1, PI_PUD_UP);
	gpioSetPullUpDown(B2, PI_PUD_UP);
	gpioSetISRFunc(B1, FALLING_EDGE, -1, ISR);
	gpioSetISRFunc(B2, FALLING_EDGE, -1, ISR);
	gpioSetMode(ALARM, PI_OUTPUT);
	gpioWrite(ALARM, 0);

	mqtt::async_client cli(SERVER_ADDRESS, CLIENT_ID);
	cli.start_consuming();
	cli.connect()->wait();
	cli.subscribe("Alarme", 1)->wait();
	enum class state {
		INIT, START, WAIT
	};
	state etat = state::INIT;
	uint32_t v;
	while (1) {
		switch (etat) {
		case state::INIT:
			gpioWrite(ALARM, 0);
			if (B1_evt) {
				etat = state::START;
			}
			B1_evt = B2_evt = false;
			break;
		case state::START:
			gpioWrite(ALARM, 1);
			v = gpioTick();
			etat = state::WAIT;
			B1_evt = B2_evt = false;
			break;
		case state::WAIT:
			gpioWrite(ALARM, 1);
			mqtt::const_message_ptr msg;
			if (cli.try_consume_message(&msg)) {
				if (msg->get_topic() == "Alarme"
```

```
                                    && msg->get_payload() == "Stop") {
                                etat = state::INIT;
                            }
                        } else if (B1_evt) {
                            etat = state::START;
                        } else if (B2_evt || gpioTick() - v > 10000000) {
                            etat = state::INIT;
                        }
                        B1_evt = B2_evt = false;
                        break;
                }
        }
        gpioTerminate();
        return 0;
}
```