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 2.

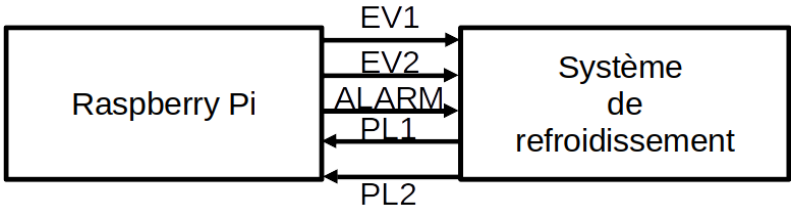
Dans un laboratoire de recherche, une expérience nécessite le maintient d'une salle à une température fixe. Le refroidissement ne doit être en aucun cas interrompu.
Pour ce faire, on utilise deux bouteilles de gaz réfrigérant B1 et B2 utilisées en alternance. Dès qu'une bouteille est vide, le système doit commuter immédiatement sur l'autre et vice-versa. Cela laisse ainsi le temps de changer la bouteille vide pendant que l'autre est utilisée. Si par mégarde, les deux bouteilles venaient à être vides, une alarme se déclenche, afin d'alerter tout le monde sur l'urgence de la situation.

Ce système de surveillance est réalisé à l'aide d'un module admettant les entrées/sorties suivantes :

• PL1,PL2 : Informations issues de capteurs de pression positionnées sur les bouteilles. Ces signaux passent à '1' si la bouteille est vide
• EV1,EV2 : Actionneurs reliés aux électrovannes des bouteilles.
La bouteille i est ouverte et délivre son gaz si `EVi = '1'`
• ALARM : Commande de l'alarme du système, active au niveau haut.

Les contraintes du cahier des charges sont les suivantes :
• À l'état initial, les deux bouteilles sont fermées. On cherchera à utiliser prioritairement la bouteille 1.
• Une bouteille de gaz est utilisée jusqu'à ce qu'elle soit vide. À ce moment uniquement, on commute sur l'autre bouteille.
• L'alarme ne peut être désactivée que par un reset asynchrone du système.
• Après le déclenchement de l'alarme, le système doit reprendre le refroidissement, sans désactiver l'alarme, dès qu'une bouteille pleine soit disponible.

1) Donner la représentation en bloc de fonction du module.



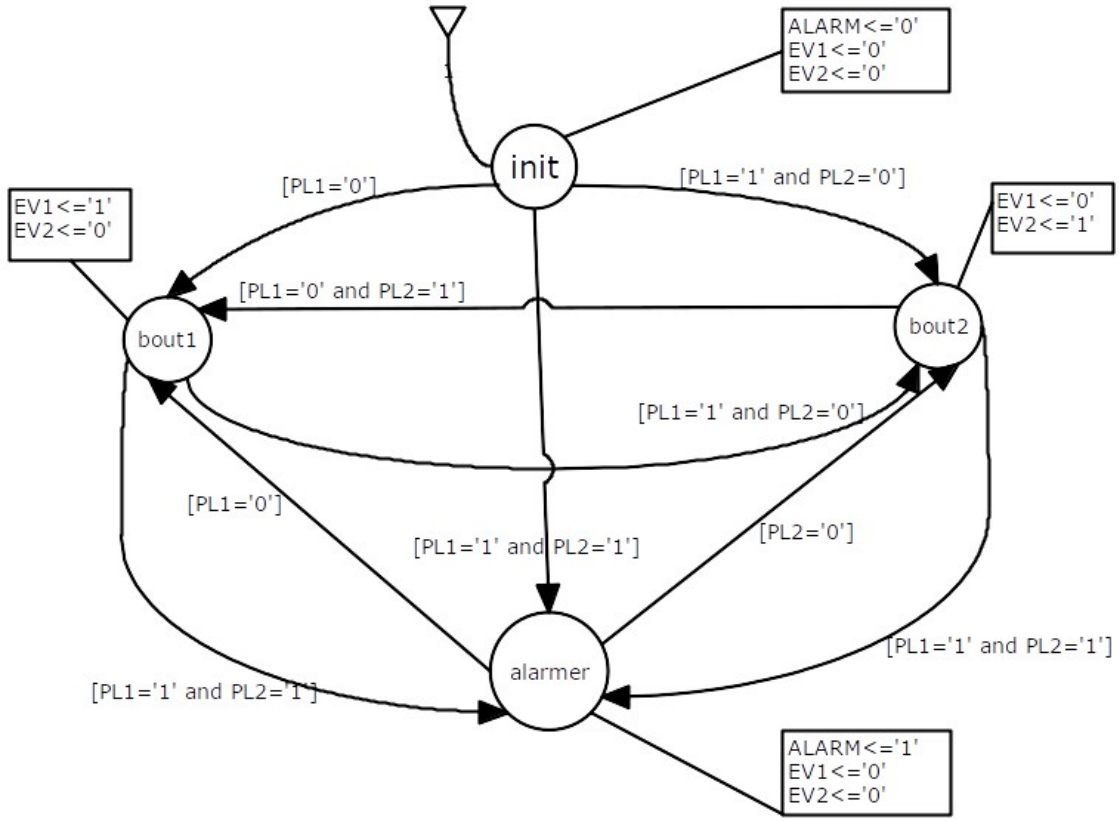
2) Donner le code C++ de la fonction `void initialize()` qui permet de configurer les entrées/sorties.

On propose les connexions suivantes : EV1 ↔ BCM14, EV2 ↔ BCM15, ALARM ↔ BCM16, PL1 ↔ BCM17, PL2 ↔ BCM18

```
constexpr uint8_t EV1 = 14;
constexpr uint8_t EV2 = 15;
constexpr uint8_t ALARM = 16;
constexpr uint8_t PL1 = 17;
constexpr uint8_t PL2 = 18;

void initialize() {
      gpioSetMode(EV1, PI_OUTPUT);
      gpioWrite(EV1,0);
      gpioSetMode(EV2, PI_OUTPUT);
      gpioWrite(EV2,0);
      gpioSetMode(ALARM, PI_OUTPUT);
      gpioWrite(ALARM, 0);
      gpioSetMode(PL1, PI_INPUT);
      gpioSetMode(PL2, PI_INPUT);
}
```

3) Donner la machine à états décrivant le comportement du module.



4) Traduire la machine à états en code C++.

```
int main() {
	gpioInitialise();
	initialize();

	enum class state {
		init, bout1, bout2, alarmer
	};
	state etat = state::init;

	while (1) {
		switch (etat) {
		case state::init:
			gpioWrite(EV1, 0);
			gpioWrite(EV2, 0);
			gpioWrite(ALARM, 0);
			if (gpioRead(PL1) == 0) {
				etat = state::bout1;
			} else if (gpioRead(PL1) == 1 && gpioRead(PL2) == 0) {
				etat = state::bout2;
			} else if (gpioRead(PL1) == 1 && gpioRead(PL2) == 1) {
				etat = state::alarmer;
			}
			break;
		case state::bout1:
			gpioWrite(EV1, 1);
			gpioWrite(EV2, 0);

			if (gpioRead(PL1) == 1 && gpioRead(PL2) == 0) {
				etat = state::bout2;
			} else if (gpioRead(PL1) == 1 && gpioRead(PL2) == 1) {
				etat = state::alarmer;
			}

			break;
		case state::bout2:
			gpioWrite(EV1, 0);
			gpioWrite(EV2, 1);
			if (gpioRead(PL1) == 0 && gpioRead(PL2) == 1) {
				etat = state::bout1;
			} else if (gpioRead(PL1) == 1 && gpioRead(PL2) == 1) {
				etat = state::alarmer;
			}

			break;
		case state::alarmer:
			gpioWrite(EV1, 0);
			gpioWrite(EV2, 0);
			gpioWrite(ALARM, 1);
			if (gpioRead(PL1) == 0) {
				etat = state::bout1;
			} else if (gpioRead(PL2) == 0) {
				etat = state::bout2;
			}
			break;
		}
	}
	gpioTerminate();
	return 0;
}
```

5) On souhaite permettre la ré-initialisation du système de surveillant à son état initial à travers la réception d'un message MQTT « STOP » dans le topic « ALARME ». On réalise l'implémentation de la réception dans un thread séparé.
Donner le code du thread, et le code traduisant la machine à états en tenant compte la ré-initialisation à travers un message MQTT.

```
const std::string SERVER_ADDRESS { "tcp://Your broker IP" };
const std::string CLIENT_ID { "user1" };
volatile bool ALARM_EVT = false;

void receiveMQTT() {
	mqtt::async_client cli(SERVER_ADDRESS, CLIENT_ID);
	cli.start_consuming();
	cli.connect()->wait();
	cli.subscribe("Alarme", 1)->wait();
	while (1) {
		auto msg = cli.consume_message();
		if (msg) {
			if (msg->get_topic() == "ALARME" && msg->get_payload() ==
"STOP") {
				ALARM_EVT = true;
			}
		}
	}
}

int main() {
	gpioInitialise();
	initialize();

	enum class state {
		init, bout1, bout2, alarmer
	};
	state etat = state::init;
	std::thread th(receiveMQTT);
	while (1) {
		switch (etat) {
		case state::init:
			gpioWrite(EV1, 0);
			gpioWrite(EV2, 0);
			gpioWrite(ALARM, 0);
			if (gpioRead(PL1) == 0) {
				etat = state::bout1;
			} else if (gpioRead(PL1) == 1 && gpioRead(PL2) == 0) {
				etat = state::bout2;
			} else if (gpioRead(PL1) == 1 && gpioRead(PL2) == 1) {
				etat = state::alarmer;
			}
			break;
		case state::bout1:
			gpioWrite(EV1, 1);
			gpioWrite(EV2, 0);

			if (gpioRead(PL1) == 1 && gpioRead(PL2) == 0) {
```

```
                        etat = state::bout2;
                  } else if (gpioRead(PL1) == 1 && gpioRead(PL2) == 1) {
                        etat = state::alarmer;
                  } else if (gpioRead(ALARM)==1 && ALARM_EVT) {
                        ALARM_EVT = false;
                        etat = state::init;
                  }
                  break;
            case state::bout2:
                  gpioWrite(EV1, 0);
                  gpioWrite(EV2, 1);
                  if (gpioRead(PL1) == 0 && gpioRead(PL2) == 1) {
                        etat = state::bout1;
                  } else if (gpioRead(PL1) == 1 && gpioRead(PL2) == 1) {
                        etat = state::alarmer;
                  } else if (gpioRead(ALARM)==1 && ALARM_EVT) {
                        ALARM_EVT = false;
                        etat = state::init;
                  }
                  break;
            case state::alarmer:
                  gpioWrite(EV1, 0);
                  gpioWrite(EV2, 0);
                  gpioWrite(ALARM, 1);
                  if (gpioRead(PL1) == 0) {
                        etat = state::bout1;
                  } else if (gpioRead(PL2) == 0) {
                        etat = state::bout2;
                  } else
                        ALARM_EVT = false;
                  break;
            }
      }
      th.join();
      gpioTerminate();
      return 0;
}
```