Université Tunis El-Manar	**Année Universitaire** : 2023-2024
Faculté des Sciences de Tunis	**Module :** Conception des objets connectés
Section : LIOT3	**Enseignants :** C.A. ABID

TD1 : Entrées/Sorties GPIO

Compilation :
g++ nomProgramme.cpp -o nomExecutable -std=c++17
Exécution :
sudo ./nomExecutable

Exercice 1.
```
#include <iostream>
#include <pigpio.h>
#include <unistd.h>
constexpr uint8_t trigPin=23,echoPin=24;
constexpr int MAX_DISTANCE=220;
constexpr int timeOut=MAX_DISTANCE*60;

uint32_t pulseIn(uint8_t pin, uint32_t timeout) {
	auto t0 = gpioTick();
	while (gpioRead(pin) != 1) {
		if ((gpioTick()-t0) > timeout) return 0;
	}
	t0 = gpioTick();
	while (gpioRead(pin) == 1) {
			if ((gpioTick()-t0) > timeout) return 0;
	}
	return gpioTick()-t0;
}

float getSonar() {
	float distance;
	gpioWrite(trigPin, 1); //trigPin send 10us high level
	usleep(10);
	gpioWrite(trigPin, 0);
	auto pingTime = pulseIn(echoPin, timeOut);
	distance = (float) pingTime * (340.0 *100.0)/ 2.0 /
1000000.0; // the sound speed is 340m/s
	return distance;
}

int main() {
	if (gpioInitialise() < 0) {
		exit(-1);
	}
	float distance = 0;
	gpioSetMode(trigPin, PI_OUTPUT);
	gpioSetMode(echoPin, PI_INPUT);
	while (1) {
		distance = getSonar();
```

```
            std::cout << "The distance is : " << distance << "\n";
            sleep(1);
        }
        return 1;
}
```

Exercice 2.

```
#include <iostream>
#include <pigpio.h>
#include <thread>
#include <chrono>
#include <tuple>
constexpr uint32_t FREQUENCY=10000000;
constexpr uint8_t _gpioPin=15;

std::tuple<uint8_t,uint8_t> ProcessData(uint64_t Data) {
        uint8_t HumidityHigh = (Data >> 32) & 0xFF;
        uint8_t TemperatureHigh = (Data >> 16) & 0xFF;
        return std::make_tuple(TemperatureHigh,HumidityHigh);
}

int WaitForLow() {
        auto StartTime = gpioTick();
        while (gpioRead(_gpioPin)) {
            if (FREQUENCY < gpioTick() - StartTime) return 0;
        }
        return gpioTick() - StartTime;
}

int WaitForHigh() {
        auto StartTime = gpioTick();
        while (!gpioRead(_gpioPin)) {
            if (FREQUENCY < gpioTick()- StartTime) return 0;
        }
        return gpioTick() - StartTime;
}

void SendStartSignal() {
        gpioWrite(_gpioPin, 0);
        std::this_thread::sleep_for(std::chrono::milliseconds(20));
        gpioWrite(_gpioPin, 1);
}

std::tuple<uint8_t,uint8_t> Measure() {
        uint64_t data = 0;
        SendStartSignal();
        gpioSetMode(_gpioPin, PI_INPUT);

            WaitForLow();
            WaitForHigh();
            WaitForLow();
```

```
          for (int i = 0; i < 40; i++) {
               data <<= 1;
               int LowTime = WaitForHigh();
               int HighTime = WaitForLow();
               if (LowTime < HighTime) {
                    data |= 0x1;
               }
          }
          WaitForHigh();
          gpioSetMode(_gpioPin, PI_OUTPUT);
     gpioWrite(_gpioPin, 1);
     return ProcessData(data);
}

int main(int argc, char *argv[]) {
     if (gpioInitialise() < 0) {
          std::cout << "Can't initialize pigpio..." << std::endl;
          exit(-1);
     }
     gpioSetMode(_gpioPin, PI_OUTPUT);
     gpioWrite(_gpioPin, 1);
     for (int i = 0; i < 10; ++i) {
          std::this_thread::sleep_for(std::chrono::seconds(5));
          std::tuple<int,int> data = Measure();
          std::cout << "Temperature: " << std::get<0>(data) << " 
°C" << std::endl;
          std::cout << "Humidity: " << std::get<1>(data) << "%" << 
std::endl;
          std::cout << std::endl;
     }

     gpioTerminate();
     return 0;
}
```