## Laboratori 1 - Introducció al mbed LPC1728

**A)** En aquest codi el que fem és a través de dues variables comprovar si el botó es prem o es desprem i anar canviant l’estat del led respectivament.

#include "mbed.h"

int main()

{

// Initialise the digital pin LED1 as an output

DigitalOut led(LED1);

DigitalIn enable(p17);

bool activa = 0;

bool pulsat = 0;

while (true) {

if(!enable) { // Botó premut

if(!pulsat) { // Comprovem que l'acció de premer sigui una nova, és a dir, que per molt que estigui el botó premut, el led no canvii d'estat fins que no despremem el botó

activa = !activa;

pulsat = 1;

}

}

else { // Indiquem que el botó ha sigut despulsat

pulsat = 0;

}

led = activa; // Assignem el valor que toca al led

}

}

**B)** En aquest codi treballem amb un timer (t) el qual controla el temps que el botó es premut. Si és premut més de 0,5 segons canviarà l’estat del led.

#include "mbed.h"

#include "Timer.h"

#define PULSE\_TIME 500

using namespace std::chrono;

int main()

{

// Initialise the digital pin LED1 as an output

DigitalOut led(LED1);

DigitalIn enable(p17);

bool activa = 0;

bool pulsat = 0;

Timer t;

bool restart = 1;

while (true) {

if(!enable) { // Botó premut

if(restart) { // Iniciem el timer del botó

t.start();

restart = 0;

}

int time\_elapsed = duration\_cast<milliseconds>(t.elapsed\_time()).count();

if(!pulsat && time\_elapsed >= PULSE\_TIME) { // Comprovem si el botó està pulsat i si porta més de PULSE\_TIME pulsat canviem l'estat del led

activa = !activa;

pulsat = 1;

}

}

else { // El botó no està premut i per tant reiniciem el timer

t.stop();

t.reset();

pulsat = 0;

restart = 1;

}

led = activa; // Assignem el valor que toca al led

}

}

**C)** En aquest codi treballem amb dos timers diferents, Un per controlar el temps que el botó ha estat premut i un altre per controlar el temps que porta fent una intermitència. La intermitència acabarà o bé quan hagin passat 10 segons o bé si es torna a prémer el botó durant 0,5 segons.

#include "mbed.h"

#include "Timer.h"

#include <chrono>

using namespace std::chrono;

#define PULSE\_DURATION 500

#define BLINK\_DURATION 10000

#define BLINK\_INTERVAL 500

int main()

{

// Initialise the digital pin LED1 as an output

DigitalOut led(LED1);

DigitalIn enable(p17);

Timer t, blinking\_timer;

bool restart = true;

bool pulsat = false;

bool blinking = false;

while (true) {

if(!enable) { // Botó premut

if(restart) { // Iniciem el timer del botó

t.start();

restart = false;

}

int time\_elapsed = duration\_cast<milliseconds>(t.elapsed\_time()).count();

if(time\_elapsed >= PULSE\_DURATION){ // Si el temps és major a PULSE\_DURATION volem comprovar si hem de iniciar la intermitència o apagar-la

if(!pulsat) { // Iniciem la intermitència

pulsat = true;

blinking = true;

blinking\_timer.start();

t.stop();

t.reset();

}

else{ // Apaguem la intermitència

pulsat = false;

blinking = false;

led = false;

t.stop();

t.reset();

blinking\_timer.stop();

blinking\_timer.reset();

}

}

}

else { // El botó no està premut

t.stop();

t.reset();

restart = true;

}

if(blinking){ // Comprovem si hem de seguir provocant una intermitència

int blinking\_time = duration\_cast<milliseconds>(blinking\_timer.elapsed\_time()).count();

if(blinking\_time > BLINK\_DURATION) { // Comprovem si ja portem més del temps que ha de durar la intermitència, en el cas que el temps sigui major l'apaguem

blinking= false;

led = false;

restart = true;

pulsat = false;

blinking\_timer.stop();

blinking\_timer.reset();

}

else if (blinking\_time % BLINK\_INTERVAL < (BLINK\_INTERVAL / 2)) { // Provoquem la intermitència del led

led = true;

}

else {

led = false;

}

}

}

}