```
#define TIMER_H

#include <thread>
#include <chrono>
#include <functional>
#include <future>
#include <cstdio>
#include <iostream>

class Timer
{
    public:
        //constructor de la clase
         Timer(){};
        //crea template de la clase
         template <class callable>
        //método connect, se encarga de gestionar la pausa
         void connect(callable&& f)
         {
                       //crea el hilo
                       std::thread([=]()
            {
                 while(true)
                 {
                //obtiene el contenido booleano guardado en go e invoca a f si go es true
                                    if(go.load())
                                            std::invoke(f);
//espera el tiempo guardado en period con el método start
std::this_thread::sleep_for(std::chrono::milliseconds(period.load()));
                 }
            }).detach();
         };
         //inicializa el timer, almacena el tiempo p pasado por parámetro
         void start(int p)
         {
                       //almacena en period el tiempo p
                       period.store(p);
                       //almacena true en la variable go
                       go.store(true);
         };
         //método que finaliza el timer
         void stop() { go.store(!go); };
                //método para modificar el período del timer
                void setPeriod(int p) { period.store(p) ;};

    private:
        //declaraciones de go y de period
         std::atomic_bool go = false;
          std::atomic_int period = 0;


};

#endif // TIMER_H
```