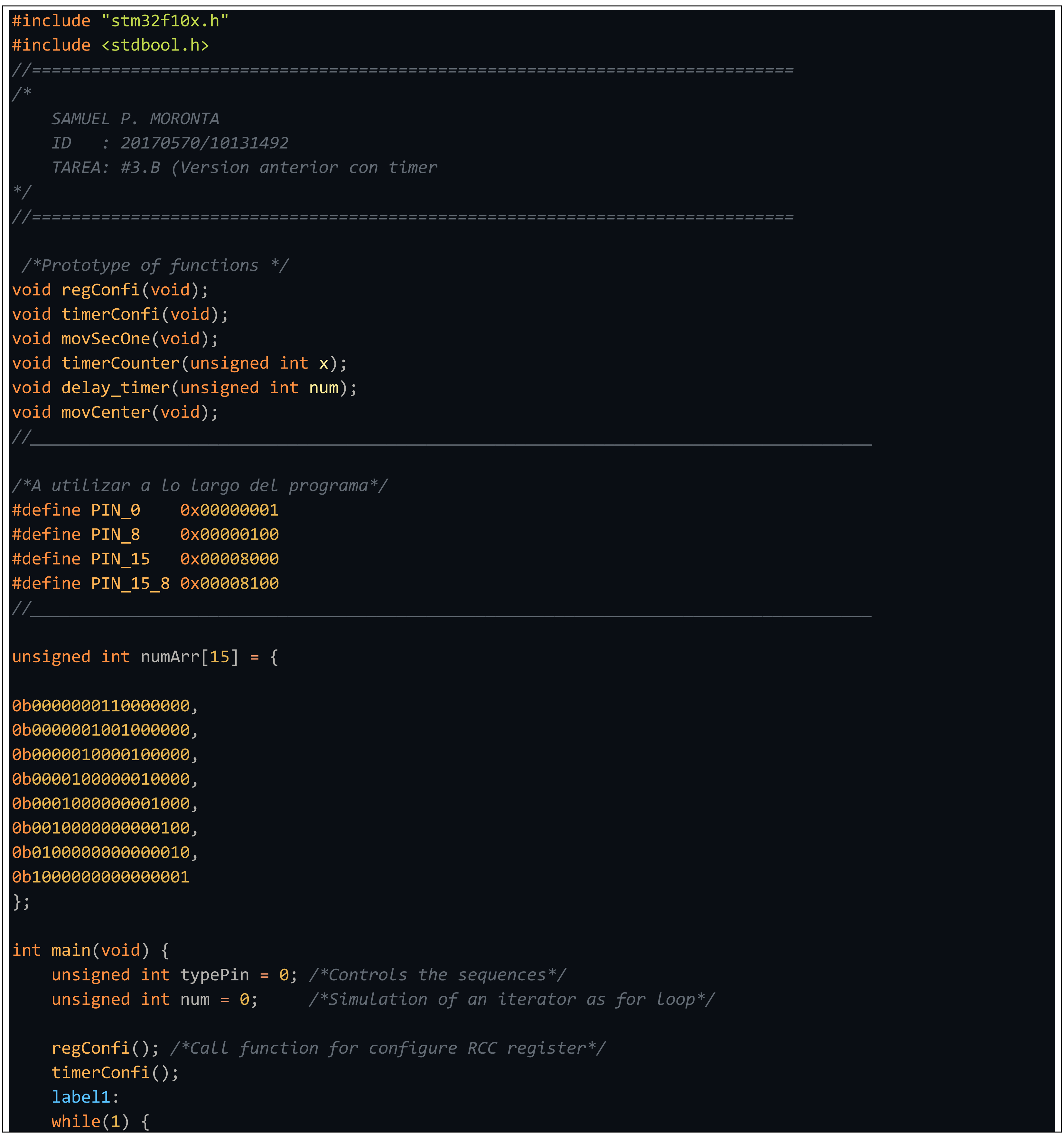
TAREA # 3.B MICROPROCESDORES Y MICROCONTROLADORES

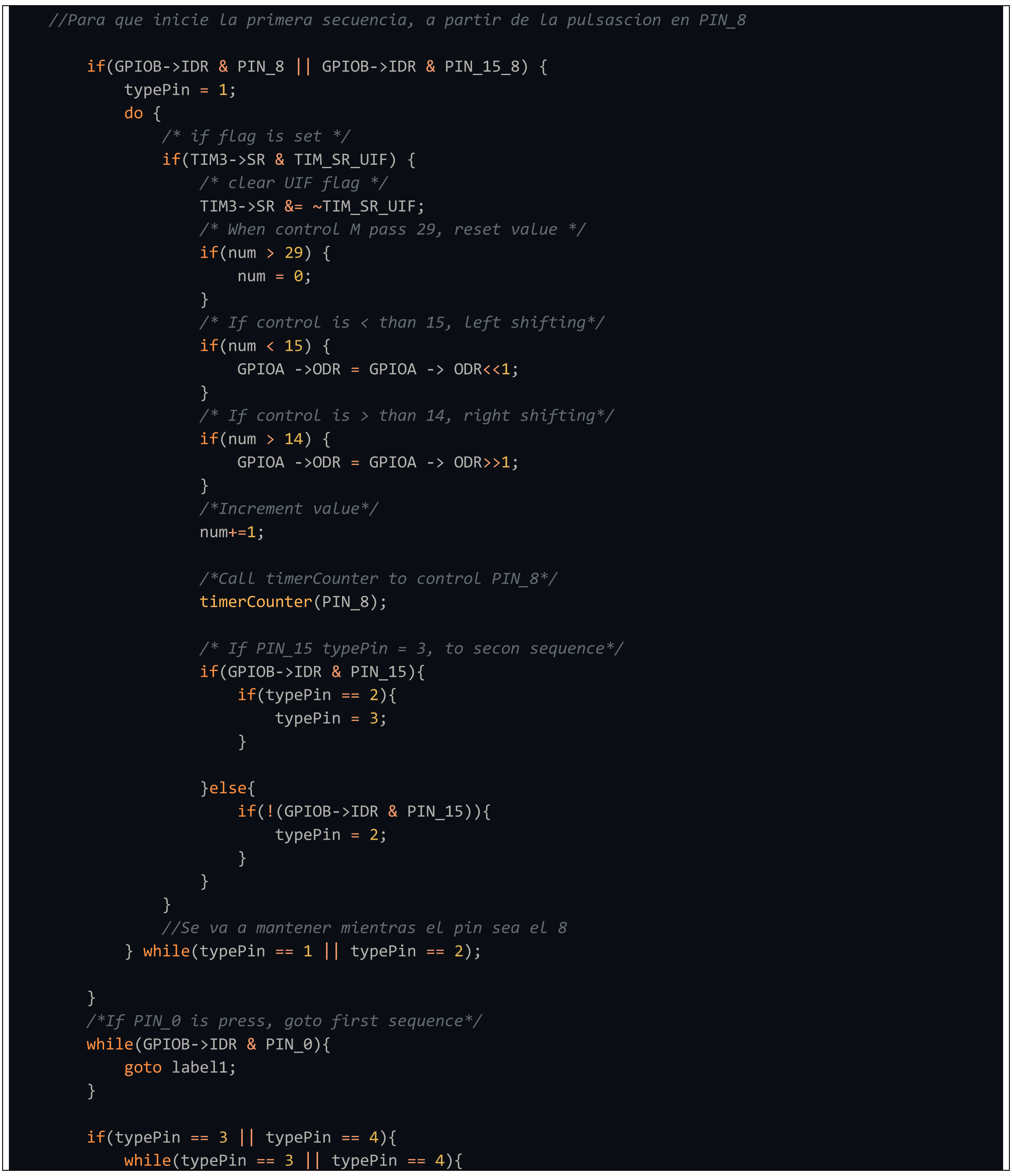
SAMUEL PEÑA MORONTA 20170570-10131492



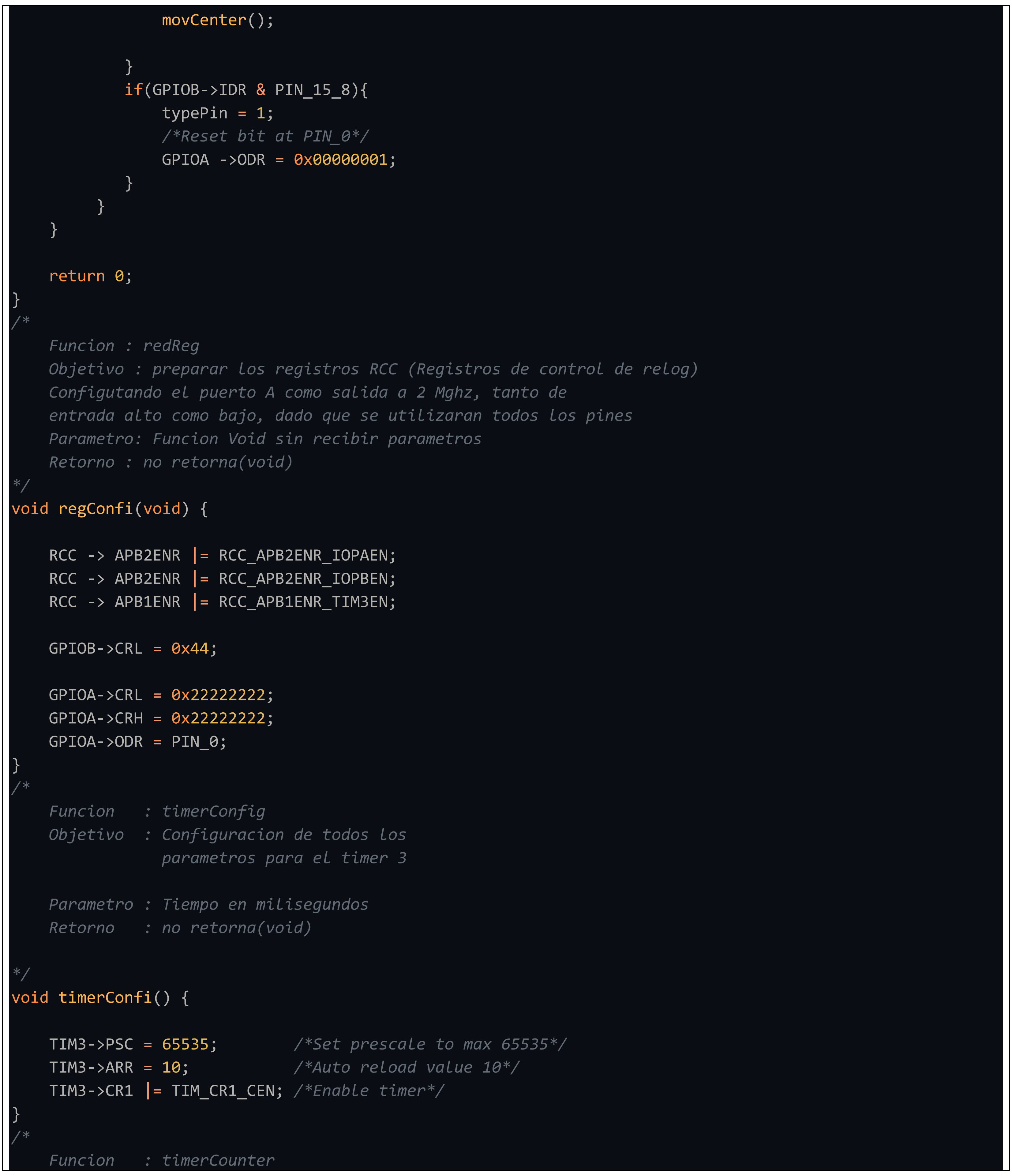
```c
#include "stm32f10x.h"
#include <stdbool.h>
//=============================================================================
/*
    SAMUEL P. MORONTA
    ID   : 20170570/10131492
    TAREA: #3.B (Version anterior con timer
*/
//=============================================================================

 /*Prototype of functions */
void regConfi(void);
void timerConfi(void);
void movSecOne(void);
void timerCounter(unsigned int x);
void delay_timer(unsigned int num);
void movCenter(void);
//_____________________________________________________________________________________

/*A utilizar a lo largo del programa*/
#define PIN_0    0x00000001
#define PIN_8    0x00000100
#define PIN_15   0x00008000
#define PIN_15_8 0x00008100
//_____________________________________________________________________________________

unsigned int numArr[15] = {

0b0000000110000000,
0b0000001001000000,
0b0000010000100000,
0b0000100000010000,
0b0001000000001000,
0b0010000000000100,
0b0100000000000010,
0b1000000000000001
};

int main(void) {
    unsigned int typePin = 0; /*Controls the sequences*/
    unsigned int num = 0;     /*Simulation of an iterator as for loop*/

    regConfi(); /*Call function for configure RCC register*/
    timerConfi();
    label1:
    while(1) {
```

TAREA # 3.B MICROPROCESDORES Y MICROCONTROLADORES

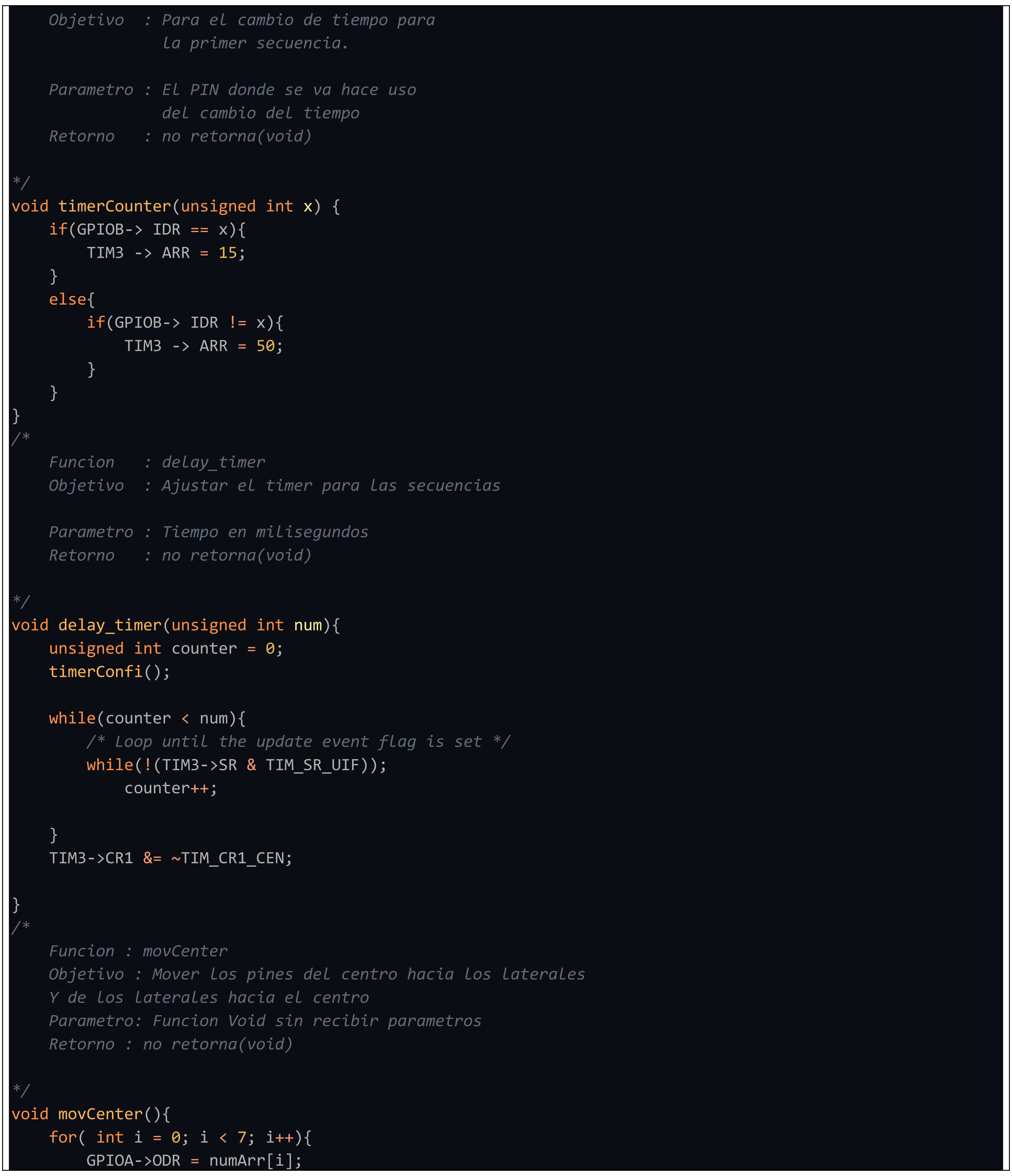
SAMUEL PEÑA MORONTA 20170570-10131492



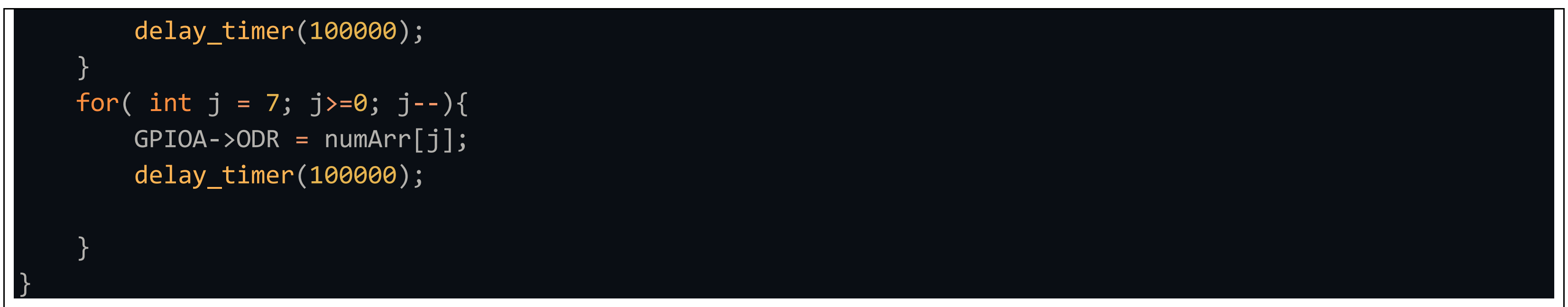
```
//Para que inicie la primera secuencia, a partir de la pulsascion en PIN_8

    if(GPIOB->IDR & PIN_8 || GPIOB->IDR & PIN_15_8) {
        typePin = 1;
        do {
            /* if flag is set */
            if(TIM3->SR & TIM_SR_UIF) {
                /* clear UIF flag */
                TIM3->SR &= ~TIM_SR_UIF;
                /* When control M pass 29, reset value */
                if(num > 29) {
                    num = 0;
                }
                /* If control is < than 15, left shifting*/
                if(num < 15) {
                    GPIOA ->ODR = GPIOA -> ODR<<1;
                }
                /* If control is > than 14, right shifting*/
                if(num > 14) {
                    GPIOA ->ODR = GPIOA -> ODR>>1;
                }
                /*Increment value*/
                num+=1;

                /*Call timerCounter to control PIN_8*/
                timerCounter(PIN_8);

                /* If PIN_15 typePin = 3, to secon sequence*/
                if(GPIOB->IDR & PIN_15){
                    if(typePin == 2){
                        typePin = 3;
                    }

                }else{
                    if(!(GPIOB->IDR & PIN_15)){
                        typePin = 2;
                    }
                }
            }
            //Se va a mantener mientras el pin sea el 8
        } while(typePin == 1 || typePin == 2);

    }
    /*If PIN_0 is press, goto first sequence*/
    while(GPIOB->IDR & PIN_0){
        goto label1;
    }

    if(typePin == 3 || typePin == 4){
        while(typePin == 3 || typePin == 4){
```



```
                movCenter();

            }
            if(GPIOB->IDR & PIN_15_8){
                typePin = 1;
                /*Reset bit at PIN_0*/
                GPIOA ->ODR = 0x00000001;
            }
        }
    }

    return 0;
}
/*
    Funcion : redReg
    Objetivo : preparar los registros RCC (Registros de control de relog)
    Configutando el puerto A como salida a 2 Mghz, tanto de
    entrada alto como bajo, dado que se utilizaran todos los pines
    Parametro: Funcion Void sin recibir parametros
    Retorno : no retorna(void)
*/
void regConfi(void) {

    RCC -> APB2ENR |= RCC_APB2ENR_IOPAEN;
    RCC -> APB2ENR |= RCC_APB2ENR_IOPBEN;
    RCC -> APB1ENR |= RCC_APB1ENR_TIM3EN;

    GPIOB->CRL = 0x44;

    GPIOA->CRL = 0x22222222;
    GPIOA->CRH = 0x22222222;
    GPIOA->ODR = PIN_0;
}
/*
    Funcion   : timerConfig
    Objetivo  : Configuracion de todos los
                parametros para el timer 3

    Parametro : Tiempo en milisegundos
    Retorno   : no retorna(void)

*/
void timerConfi() {

    TIM3->PSC = 65535;        /*Set prescale to max 65535*/
    TIM3->ARR = 10;           /*Auto reload value 10*/
    TIM3->CR1 |= TIM_CR1_CEN; /*Enable timer*/
}
/*
    Funcion   : timerCounter
```



```
    Objetivo  : Para el cambio de tiempo para
                la primer secuencia.

    Parametro : El PIN donde se va hace uso
                del cambio del tiempo
    Retorno   : no retorna(void)

*/
void timerCounter(unsigned int x) {
    if(GPIOB-> IDR == x){
        TIM3 -> ARR = 15;
    }
    else{
        if(GPIOB-> IDR != x){
            TIM3 -> ARR = 50;
        }
    }
}
/*
    Funcion   : delay_timer
    Objetivo  : Ajustar el timer para las secuencias

    Parametro : Tiempo en milisegundos
    Retorno   : no retorna(void)

*/
void delay_timer(unsigned int num){
    unsigned int counter = 0;
    timerConfi();

    while(counter < num){
        /* Loop until the update event flag is set */
        while(!(TIM3->SR & TIM_SR_UIF));
            counter++;

    }
    TIM3->CR1 &= ~TIM_CR1_CEN;

}
/*
    Funcion : movCenter
    Objetivo : Mover los pines del centro hacia los laterales
    Y de los laterales hacia el centro
    Parametro: Funcion Void sin recibir parametros
    Retorno : no retorna(void)

*/
void movCenter(){
    for( int i = 0; i < 7; i++){
        GPIOA->ODR = numArr[i];
```



```
        delay_timer(100000);
    }
    for( int j = 7; j>=0; j--){
        GPIOA->ODR = numArr[j];
        delay_timer(100000);

    }
}
```