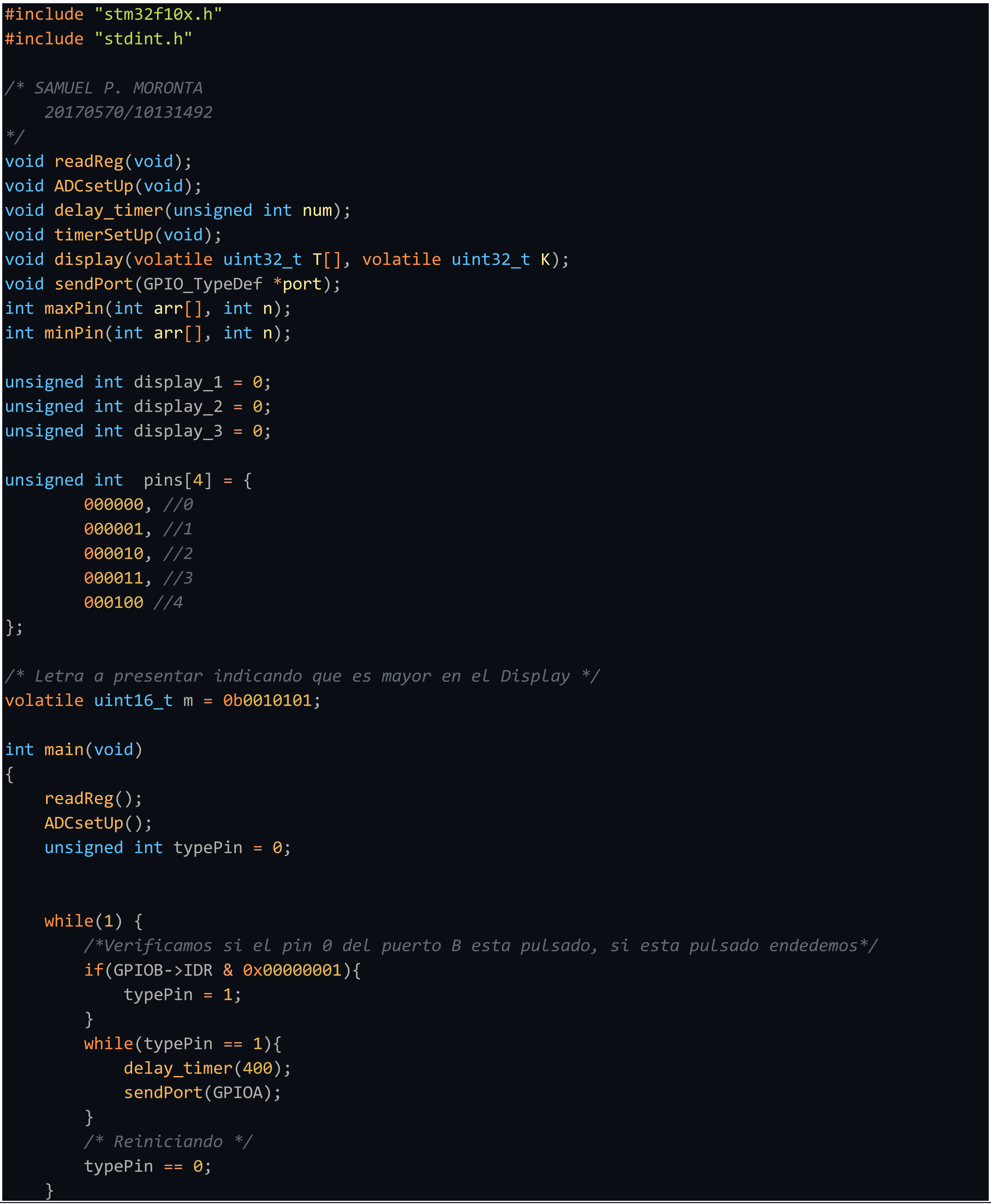
SAMUEL P. MORONTA 20170570/10131492



```c
#include "stm32f10x.h"
#include "stdint.h"

/* SAMUEL P. MORONTA
    20170570/10131492
*/
void readReg(void);
void ADCsetUp(void);
void delay_timer(unsigned int num);
void timerSetUp(void);
void display(volatile uint32_t T[], volatile uint32_t K);
void sendPort(GPIO_TypeDef *port);
int maxPin(int arr[], int n);
int minPin(int arr[], int n);

unsigned int display_1 = 0;
unsigned int display_2 = 0;
unsigned int display_3 = 0;

unsigned int  pins[4] = {
        000000, //0
        000001, //1
        000010, //2
        000011, //3
        000100 //4
};

/* Letra a presentar indicando que es mayor en el Display */
volatile uint16_t m = 0b0010101;

int main(void)
{
    readReg();
    ADCsetUp();
    unsigned int typePin = 0;


    while(1) {
        /*Verificamos si el pin 0 del puerto B esta pulsado, si esta pulsado endedemos*/
        if(GPIOB->IDR & 0x00000001){
            typePin = 1;
        }
        while(typePin == 1){
            delay_timer(400);
            sendPort(GPIOA);
        }
        /* Reiniciando */
        typePin == 0;
    }
```

```c
}

void readReg(void){

     RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;
     RCC->APB2ENR |= RCC_APB2ENR_IOPBEN;
     GPIOB->CRL = 0x4444444;
     GPIOA->CRL = 0x22222222; //2
     GPIOA->CRH = 0x22222222;
     //Registro de lectura/
     GPIOA->ODR = 0x00000001;
}
void ADCsetUp(void){

    /* Habilitando AFIO */
    RCC->APB2ENR |= RCC_APB2ENR_AFIOEN;
    /* Remapear el TIM2*/
    AFIO->MAPR |= AFIO_MAPR_TIM1_REMAP_PARTIALREMAP;
    /*ADC CONFIGURACION */
    ADC1->CR2   = ADC_CR2_ADON | ADC_CR2_CONT;
    /* Cambiando prescale para no exceder los 14Mhz */
    RCC->CFGR |= RCC_CFGR_ADCPRE_DIV6;

    RCC->CFGR =0x8;
    ADC1->CR1 = 0x00;
    /*Configurando el Sampleo */
    ADC1->SMPR1 = 0x00;
    ADC1->SQR3 = 1;

}
void timerSetUp(void){
    /*Set prescale to max 65535*/
     TIM3->PSC = 65535;
    /*Auto reload value 10*/
     TIM3->ARR = 10;
    /*Enable timer*/
     TIM3->CR1 |= TIM_CR1_CEN;
}
void delay_timer(unsigned int num){
     unsigned int counter = 0;

 while(counter < num){
    /* Loop until the update event flag is set */
    while(!(TIM3->SR & TIM_SR_UIF));
        counter++;
    }
    TIM3->CR1 &= ~TIM_CR1_CEN;
}
void display(volatile uint32_t T[], volatile uint32_t K){
```

SAMUEL P. MORONTA 20170570/10131492

```c
/* Mostrando primer display, la primera comparacion, es decir
   tenemos 3 display [*][*][*], Mostramos el voltaje maximo
   en el primer display, y el voltaje minimo en el segundo,
   en el 3 display mostramos  la letra que corresponde al
   displa mayor
*/
    display_1 = K/100;

    display_2 = ((K-display_1 *100)/10);

    display_3 = (K - display_1 * 100 - display_2 * 10);

    GPIOC->ODR = T[display_3-1];
    GPIOD->ODR = T[display_2-1];
}
/* Obtenemos el maximo de pines en un arreglo */
int maxPin(int arr[], int n)
{
    int i;

    int max = arr[0];
    for (i = 1; i < n; i++)
        if (arr[i] > max)
            max = arr[i];

    return max;
}
/* Obtenemos el minimo de pines en un arreglo */
int minPin(int arr[], int n)
{
    int i;

    int min = arr[0];
    for (i = 1; i < n; i++)
        if (arr[i] < min)
            min = arr[i];

    return min;
}
/* Enviamos el puerto de acuerdo a los paraemtros ya verifcados
    es decir, tenemos el puerto A, verificando con anterioridad
    cuanl de los arreglos de lo poines mandados al puerto es el
    mayor en ese arreglo
*/
void sendPort(GPIO_TypeDef *port){
    /*Reccoremos la cantidad de entradas en el puerto, 4 entradas*/
    for(unsigned int i = 0; i < 4; i++){
        /*Si estamos en el puerto A*/
        if(port == GPIOA){
            /*Comprobamos el maximo, y le asigamos en el lugae co
              correspondiente en el display correspindiente*/
            if(maxPin(pins[i], 4)== 1){
                display((maxPin(pins[i], 4)),100);
                GPIOA->ODR = m;
            }
            else if (minPin(pins[i],4) == 1){
                display((minPin(pins[i], 4)),100);
                GPIOA->ODR = m;
            }
        }
    }
}
```

SAMUEL P. MORONTA 20170570/10131492