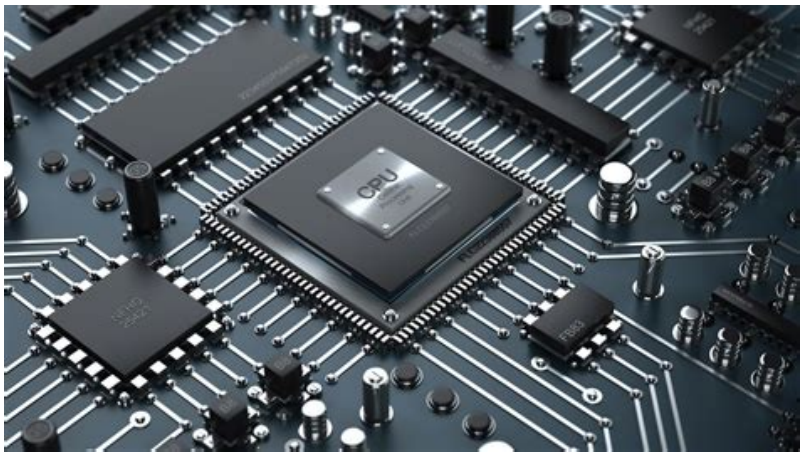
Trabajo Práctico Número 3

Informe técnico

Un trabajo presentado para la materia de
Aplicaciones de electrónica digital



Krapp Ramiro – Golmar Elias – Pisacane Juan Cruz
Instituto tecnológico San Bonifacio
Departamento de electrónica
31 de octubre de 2021

Hecho en LaTeX

Índice

Actividad

Se desea realizar una cerradura electrónica.

La misma debe contener como entrada un control de acceso por teclado (matricial); como salidas dos señales de 12V para el accionamiento de un solenoide (consumo máximo 150mA) y un zumbador (consumo máximo 50mA).

El circuito debe permitir 3 (tres) intentos de ingreso de código, activándose la alarma (zumbador) luego del tercer intento erróneo. También debe poseer tres LEDs, uno que indique que el equipo está encendido (el microcontrolador está iniciado), otro que indique el correcto ingreso del código y otro que se encienda cuando se dispara la alarma.

El código debe contener etapas de direccionamiento indirecto donde fuese necesario sin excepción.

Se pide:

a) Dibujar circuito eléctrico.

b) Realizar diagrama de flujo.

c) Realizar código en lenguaje assembler.

d) Realizar código en lenguaje C.

e) Realizar bitácoras personales

f) Simular en PROTEUS.

Circuito Eléctrico

Este es el circuito electrico:

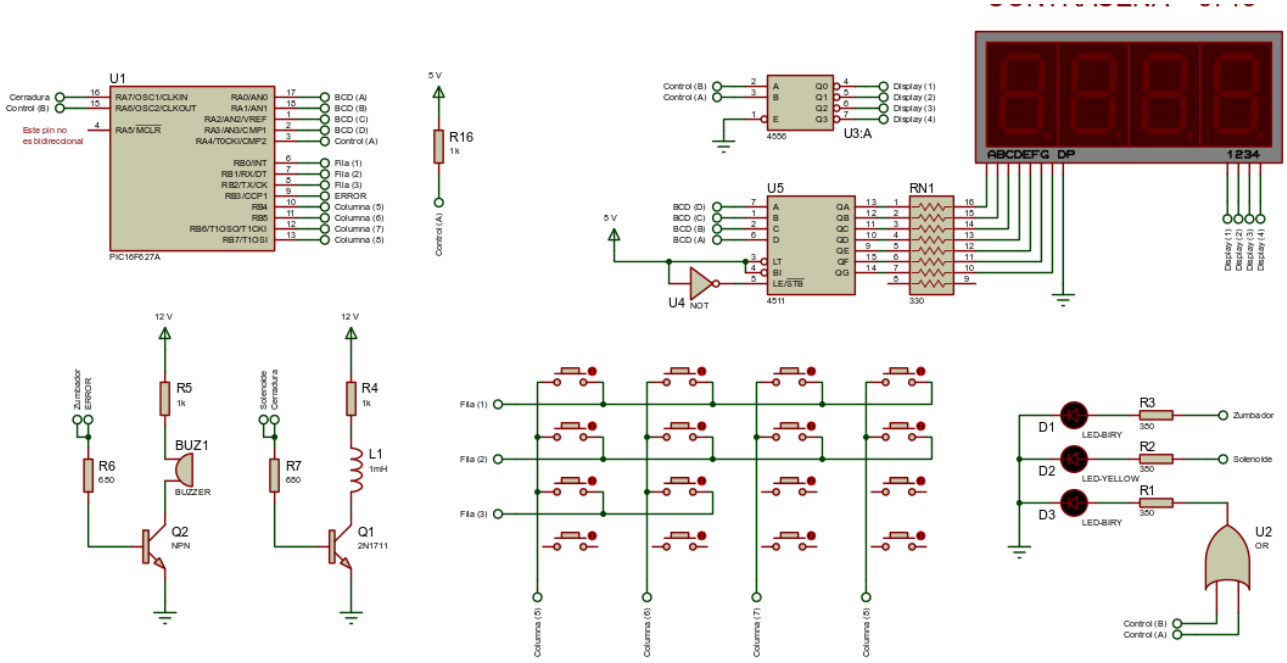


Figura 1: Diagrama esquematico hecho en Proteus

Diagrama de flujo

Este es el diagrama de flujo

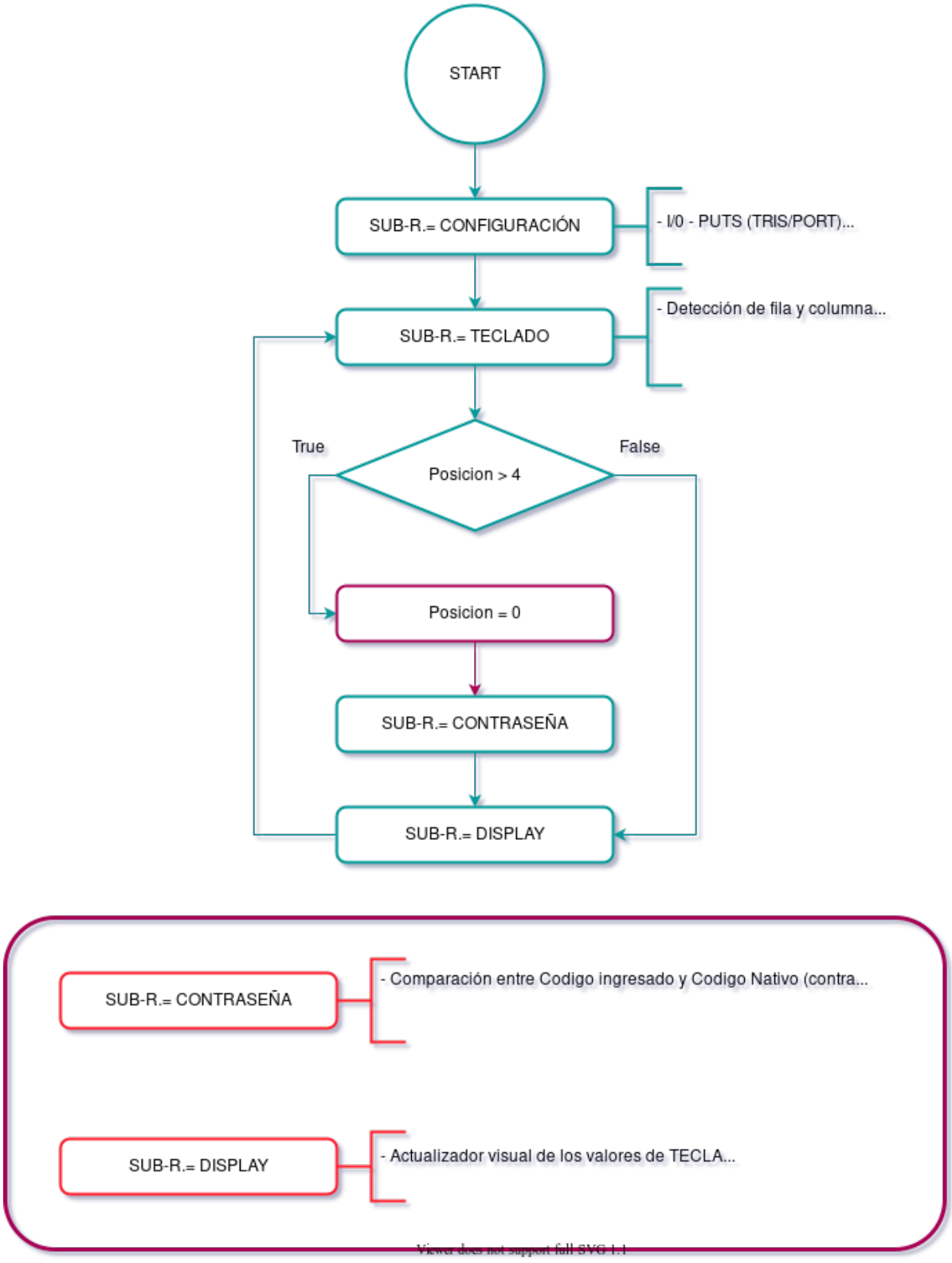


Figura 2: Diagrama de flujo simplificado de assembler

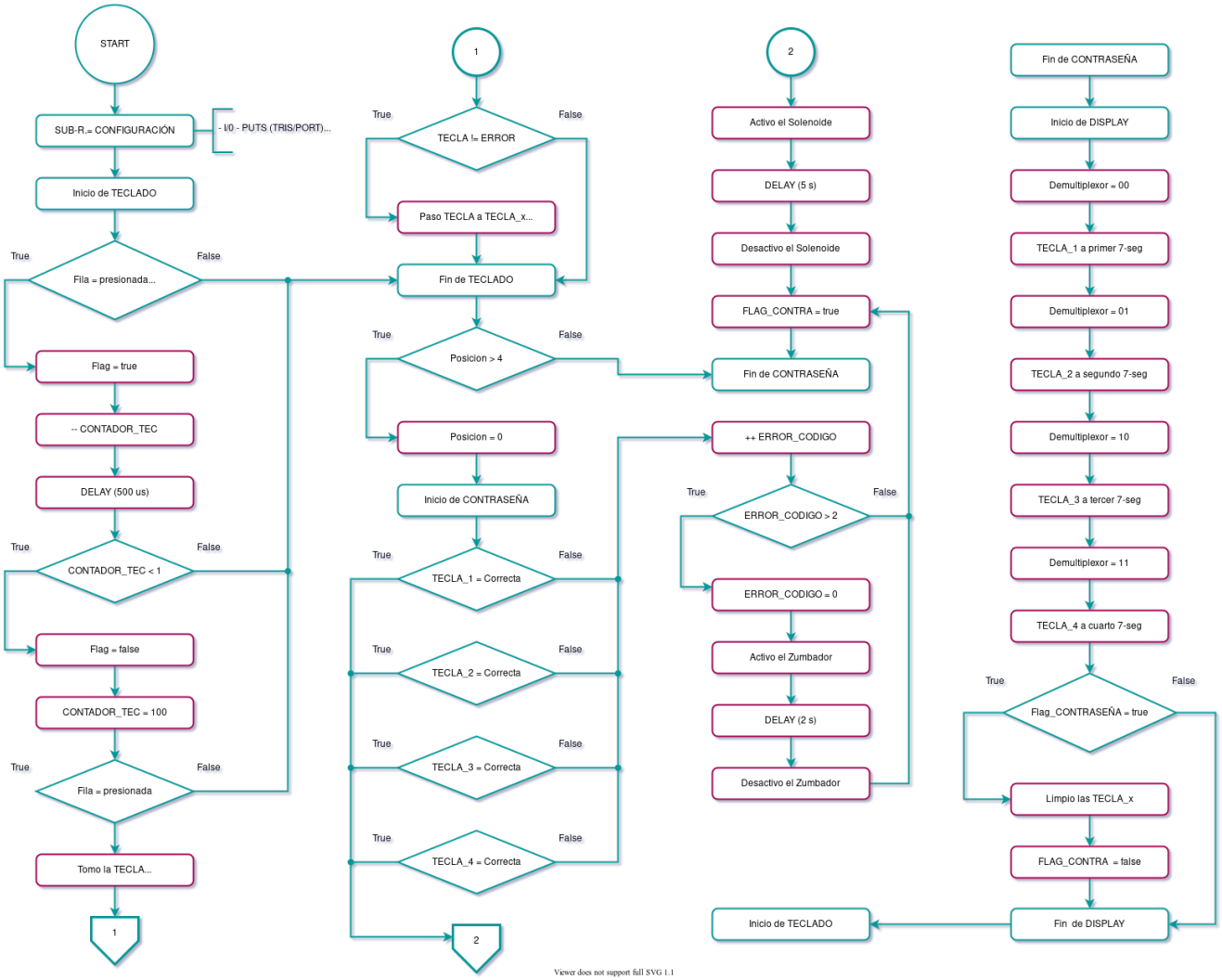


Figura 3: Flujo completo de assembler

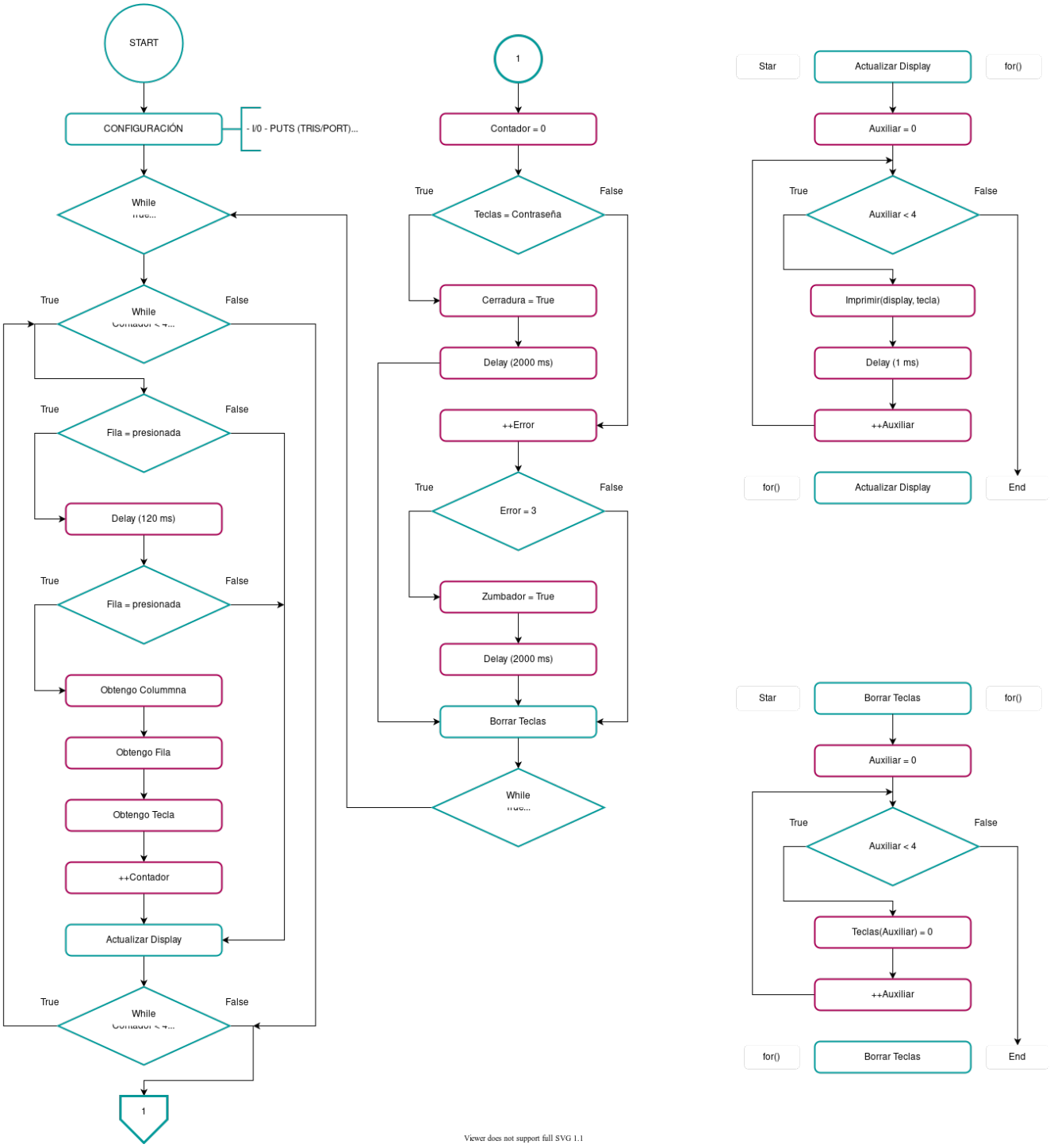


Figura 4: Flujo de C

Código en assembler

Codigo principal

```
;------------------------------------------------------------------
 Links de Interes:
- https://drive.google.com/file/d/1ZZ_PkKAMHkiPFsBQH1RG-QCuwhHDrB09/view?usp=sharing    [2021/04/29 (Subrutina)]
- https://www.youtube.com/watch?v=P2d1vcJXiKo                        [Teclado Matricial (N°1)]
- https://www.youtube.com/watch?v=FKH4VzvnSIA                        [Teclado Matricial (N°2)]
- https://drive.google.com/file/d/1gRnxLs02jr1anoHdlwI4Nu9IDc3Q3f7H/view?usp=sharing    [2021/06/24
↪  (Direccionamiento indirecto)]
- https://drive.google.com/file/d/1MCJOSUEMAwAOi8mT45195ngJOqF3UO4N/view?usp=sharing    [2021/06/10 (Display
↪  7-segmentos)]
- https://blog.ars-electronica.com.ar/2017/08/cd4511-decodificador-para-display-7.html    [Display 7 - seg con
↪  CD4511 (Decodificador BCD)]
- https://pdf1.alldatasheet.es/datasheet-pdf/view/66469/INTERSIL/CD4555.html          [Datasheet CD4555
↪  (Demultiplexor 1 a 4)]
- https://www.youtube.com/watch?v=XElUJawQXho                       [Multiplexado de displays]
;------------------------------------------------------------------

Datos:

- Recordatorios:
   TRISX, 1 = Entrada.
   TRISX, 0 = Salida.
   PORTX, 1 = HIGH.
   PORTX, 0 = LOW.

- La funcion SWAPF "registro", 0 ---> cambia los primeros 4 valores
  por los ultimos 4 valores y guarda la combinacion en el "acumulador" (W)
  Pero el "file" (F) queda igual.
- La funcion SWAPF "registro", 1 ---> cambia los primeros 4 valores
  por los ultimos 4 valores y sobreescribe la combinacion en "file" (F).

- Teclado Matricial 4x4:
    - Tecla oprimida = Fila + Columna
    - Cada pin de "fila" tiene un peso que va de 0 a 12.
    - Cada pin de "columna" tiene un peso que va de 0 a 3.
    - Para detectar Fila:
        * Pongo los pines de columna a 0 (salida).
          * Pongo los pines de fila a 1, o "nada" y activo el "pull up" de estos pines (entrada)
          * Cuando se activa una tecla, el pin de fila se pondra a 0 (pasa de 1 [pull up] a 0).
          * Valores posibles en binario para fila (en decimal): 0000-1110 (.14) / 0000-1101 (.13) / 0000-1011 (.11)
          ↪  / 0000-0111 (.7)

    - Para detectar Columna: swapear tanto el Tris como los pull ups o estados logicos
        * Pongo los pines de columna a 1, o "nada" y activo el "pull up" de estos pines (entrada).
          * Pongo los pines de fila a 0 (salida)
          * Cuando se activa una tecla, el pin de columna se pondra a 0 (pasa de 1 [pull up] a 0).
          * Swapear el TRISx y el PORTx(LOS DECIMALES):
           Valores posibles en binario para columna (en decimal): 0000-0001 (.1) / 0000-0010 (.2) / 0000-0100 (.4) /
           ↪  0000-1000 (.8)
           El caso anterior es swapear y complementar los valores de Fila.

;-------------------------------------------------------------------------------------------------------------

#include "configurationBits.h"            ; Incluyo la configuración para los pines RB4, RA6 y RA7 (I/O).

;-------------------------------------------------------------------------------------------------------------
; Las siguentes asignaciones corresponden a REGISTROS deL programa:
PCL               equ     0x02      ; Direccion dentro de "DATA MEMORY" para PCL   (banco 0, posicion 0x02).
STATUS            equ     0x03      ; Direccion dentro de "DATA MEMORY" para STATUS (banco 0, posicion 0x03).
PORTA             equ     0x05      ; Direccion dentro de "DATA MEMORY" para PORTAx (banco 0, posicion 0x05).
PORTB             equ     0x06      ; Direccion dentro de "DATA MEMORY" para PORTBx (banco 0, posicion 0x06).
TRISA             equ     0x85      ; Direccion dentro de "DATA MEMORY" para TRISAx (banco 1, posicion 0x85).
TRISB             equ     0x86      ; Direccion dentro de "DATA MEMORY" para TRISBx (banco 1, posicion 0x86).
OPTION_REG          equ     0x81      ; Direccion para acceder al pull up del puerto B interno del PIC.

;-------------------------------------------------------------------------------------------------------------
; Las siguentes asignaciones corresponden a VARIABLES de la aplicación:
_AUX1             equ     0x20      ; Direcciones para variables auxiliares necesarias para el DELAY original.
```

```
62  _AUX2             equ    0x21
63  _AUX3             equ    0x22
64  TECLA            equ    0x23      ; Direccion para acceder a la tecla [fila + columna].
65  ERROR_ACUM         equ    0x24      ; Direccion para acceder al contador de errores.
66  TECLA_1          equ    0x25      ; Direcciones para variables auxiliares necesarias para el DISPLAY.
67  TECLA_2          equ    0x26
68  TECLA_3          equ    0x27
69  TECLA_4          equ    0x28
70  FLAG             equ    0x29      ; Direccion para acceder a las banderas (para TECLADO y CONTRASEÑA).
71  AUXILIAR         equ      0x30      ; Direccion para acceder al auxiliar (para todo tipo de cuentas/procesos).
72  CONTADOR_DPY      equ      0x31      ; Direccion para acceder al contador que multiplica el DELAY sobre un el
    ↪  DISPLAY.
73  POSICION         equ      0x33      ; Direccion para acceder a la posicion de la tecla [1 a 5].
74  CODIGO_1         equ    0x34      ; Direccion para acceder al primer numero de la contraseña.
75  CODIGO_2         equ    0x35      ; Direccion para acceder al segundo numero de la contraseña.
76  CODIGO_3         equ    0x36      ; Direccion para acceder al tercer numero de la contraseña.
77  CODIGO_4         equ    0x37      ; Direccion para acceder al cuarto numero de la contraseña.
78  _AUX11            equ    0x38      ; Direcciones para variables auxiliares necesarias para el DELAY modificado.
79  _AUX33            equ    0x39
80  _AUX22            equ      0x40
81  ;-------------------------------------------------------------------------------------------------------------------
82  ; Las siguientes asignaciones corresponden a MACROS de la aplicación:
83  SAVE_IN_W         equ    0          ; Destino.
84  SAVE_IN_F         equ    1          ; Destino.
85  DELAY            equ      0          ; Flag nuevo.
86  PASSWORD_VERIFIED   equ      1          ; Flag nuevo.
87  ERROR_TECLA         equ      2          ; Flag nuevo.
88  CARRY            equ      0          ; Flag.
89  ZERO            equ      2          ; Flag.
90  RP0               equ    5
91  RP1               equ    6
92  RBPU             equ    7
93  ;-------------------------------------------------------------------------------------------------------------------
94  ; Las siguientes asignaciones corresponden a ESPECIFICACIONES deL programa:
95  RES_VECT         CODE   0x0000
96                GOTO   INICIO
97  MAIN_PROG         CODE
98
99  ;-------------------------------------------------------------------------------------------------------------------
100 ; Subrutina de Configuracion
101
102 CONFIGURACION      BCF         STATUS, RP1              ; Banco 1 para Trisx.
103               BSF         STATUS, RP0              ; Banco 1 para Trisx.
104               MOVLW      0x20                ; Pongo como salida de RA0 a RA4 + RA6 a RA7 (RA5 = NO
                  ↪  BIDIRECCIONAL).
105               MOVWF      TRISA
106               BCF         OPTION_REG, RBPU         ; Activo los pull-ups internos del PIC del puerto B (pines RBx).
107               BCF         STATUS, RP0
108               ;----------------------------
109               CLRF      FLAG
110               CLRF      ERROR_ACUM
111               CLRF      POSICION
112               CLRF      TECLA_1
113               CLRF      TECLA_2
114               CLRF      TECLA_3
115               CLRF      TECLA_4
116               CLRF      CONTADOR_DPY
117               CLRW                           ; Limpio el acumulador.
118               MOVLW      .5                     ; Valor del CODIGO_1.
119               MOVWF      CODIGO_1
120               MOVLW      .7                     ; Valor del CODIGO_2.
121               MOVWF      CODIGO_2
122               MOVLW      .1                     ; Valor del CODIGO_3.
123               MOVWF      CODIGO_3
124               MOVLW      .3                     ; Valor del CODIGO_4.
125               MOVWF      CODIGO_4
126               CLRW
127               RETURN
128
129 ;-------------------------------------------------------------------------------------------------------------------
130 ; Subrutina del Teclado Matricial (10 teclas, 3x4 modificado)
```

```
131
132 TECLAS            CLRF      TECLA
133                 CLRW
134                 BSF         STATUS, RP0
135                 MOVLW       0xF0                ; High PORTB como entrada.
136                 MOVWF       TRISB
137                 BCF         STATUS, RP0
138                 MOVLW       0x00                ; Escribir 1 en Low PORTB.
139                 MOVWF       PORTB
140                 ;---------------------              ; Antirebote.
141                 BTFSS       PORTB, 4
142                 GOTO        QUIZAS_TOCO_TECLA
143                 BTFSS       PORTB, 5
144                 GOTO        QUIZAS_TOCO_TECLA
145                 BTFSS       PORTB, 6
146                 GOTO        QUIZAS_TOCO_TECLA
147                 BTFSS       PORTB, 7
148                 GOTO        QUIZAS_TOCO_TECLA
149                 GOTO          NO_TOCO_TECLA
150                 ;---------------------
151 QUIZAS_TOCO_TECLA   CALL      DELAY_100ms              ; CALIBRADO PARA PROTEUS.
152                 ;---------------------
153                 BTFSS       PORTB, 4
154                 GOTO        TOCO_TECLA
155                 BTFSS       PORTB, 5
156                 GOTO        TOCO_TECLA
157                 BTFSS       PORTB, 6
158                 GOTO        TOCO_TECLA
159                 BTFSS       PORTB, 7
160                 GOTO        TOCO_TECLA
161                 GOTO          NO_TOCO_TECLA
162                 ;---------------------              ; Detecto Columna.
163 TOCO_TECLA          CLRW
164                 CLRF        TECLA
165                 ADDWF       PORTB, SAVE_IN_W
166                 MOVWF       AUXILIAR
167                 COMF        AUXILIAR, SAVE_IN_F         ; Complemente después de leer puerto.
168                 CLRW
169                 SWAPF       AUXILIAR, SAVE_IN_W
170                 CLRF        AUXILIAR
171                 ANDLW       0x0F
172                 CALL        PESO_COLUMNA
173                 ADDWF       TECLA, SAVE_IN_F
174                 CLRW
175                 ;---------------------
176                 BTFSC       FLAG, ERROR_TECLA
177                 GOTO        NO_TOCO_TECLA           ; Error de doble tecla.
178                 ;---------------------              ; Detecto Fila.
179                 CALL        TECLADO_SWAP
180                 ADDWF       PORTB, SAVE_IN_W
181                 MOVWF       AUXILIAR
182                 COMF        AUXILIAR, SAVE_IN_F         ; Complemento después de leer puerto.
183                 CLRW
184                 ADDWF       AUXILIAR, SAVE_IN_W
185                 CLRF        AUXILIAR
186                 ANDLW       0x07
187                 CALL        PESO_FILA
188                 ;---------------------
189                 BTFSC       FLAG, ERROR_TECLA
190                 GOTO        NO_TOCO_TECLA           ; Error de doble tecla.
191                 ;---------------------
192                 ADDWF       TECLA, SAVE_IN_F         ; TECLA = COLUMNA + FILA
193                 CLRW
194                 ;---------------------
195                 ADDWF       POSICION, SAVE_IN_W       ; Paso el contenido de POSICION al acumulador (sin perder
                    ↪ POSICION).
196                 CALL        PESO_DERIVADO           ; Asigna el valor de TECLA a otra variable que corresponde con su
                    ↪ posición.
197                 INCF        POSICION, SAVE_IN_F       ; Incremento la posición para la siguiente tecla.
198                 ;---------------------
199 NO_TOCO_TECLA       BCF         FLAG, ERROR_TECLA
```

```
200                 RETURN
201
202   ;-------------------------------------------------------------------------------------------------------------------
203   ; Subrutina (procedimiento) para SWAPEAR el registro TRISB y PORTB.
204   ; Solo se usa 1 vez por TECLADO cuando se quiere detectar FILA.
205
206   TECLADO_SWAP      BSF         STATUS, RP0               ; Cambio de banco para el TRISx.
207                   MOVLW       0x0F
208                   MOVWF       TRISB
209                   BCF         STATUS, RP0             ; Cambio de banco para el PORTx.
210                   MOVLW       0x00
211                   MOVWF       PORTB
212                   CLRW
213                   RETURN
214
215   ;-------------------------------------------------------------------------------------------------------------------
216   ; Sub-subrutina de Tabla de PESO de COLUMNA (anidada en TECLAS).
217   ; NOTA = esta pensada de forma en que PC tiene la direccion actual.
218
219   PESO_COLUMNA      ADDWF       PCL, SAVE_IN_F            ; Lo que esta en el acumulador (W) se lo sumo a PCL (F).
220                   GOTO        DOBLE_TECLA           ; 0       ; ERROR.
221                   RETLW       .0                ; 1       ; Columna 1
222                   RETLW       .1                ; 2       ; Columna 2
223                   GOTO        DOBLE_TECLA           ; 3       ; ERROR.
224                   RETLW       .2                ; 4       ; Columna 3
225                   GOTO        DOBLE_TECLA           ; 5       ; ERROR.
226                   GOTO        DOBLE_TECLA           ; 6       ; ERROR.
227                   GOTO        DOBLE_TECLA           ; 7       ; ERROR.
228                   RETLW       .3                ; 8       ; Columna 4
229                   GOTO        DOBLE_TECLA           ; 9       ; ERROR.
230                   GOTO        DOBLE_TECLA           ; 10    ; ERROR.
231                   GOTO        DOBLE_TECLA           ; 11    ; Fila 3.
232                   GOTO        DOBLE_TECLA           ; 12    ; ERROR.
233                   GOTO        DOBLE_TECLA           ; 13    ; Fila 2.
234                   GOTO        DOBLE_TECLA           ; 14    ; Fila 1.
235   DOBLE_TECLA         BSF         FLAG, 2             ; 15    ; ERROR. Seteo FLAG de error de doble
      ↪  tecla.
236                   RETURN
237
238   ;-------------------------------------------------------------------------------------------------------------------
239   ; Sub-subrutina de Tabla de PESO de FILA (anidada en TECLAS).
240   ; NOTA = esta pensada de forma en que PC tiene la direccion actual.
241
242   PESO_FILA         ADDWF       PCL, 1                    ; Lo que esta en el acumulador (W) se lo sumo a PCL (F).
243                   GOTO        DOBLE_TECLA           ; 0       ; ERROR.
244                   RETLW       .0                ; 1       ; Fila 1.
245                   RETLW       .4                ; 2       ; Fila 2.
246                   GOTO        DOBLE_TECLA           ; 3       ; ERROR.
247                   RETLW       .8                ; 4       ; Fila 3.
248                   GOTO        DOBLE_TECLA           ; 5     ; ERROR.
249                   GOTO        DOBLE_TECLA           ; 6     ; ERROR.
250                   RETLW       .0                ; 7       ; Si es un error, muestro un 0 por defecto.
251                   BSF         FLAG, 2                   ; Seteo FLAG de error de doble tecla.
252                   RETURN
253
254   ;-------------------------------------------------------------------------------------------------------------------
255   ; Sub-subrutina de Tabla de PESO de TECLA DERIVADO (anidada en TECLADO).
256   ; NOTA = esta pensada de forma en que PC tiene la direccion actual.
257
258   PESO_DERIVADO       ADDWF       PCL, 1                    ; Lo que esta en el acumulador (W) se lo sumo a PCL (F).
259                   GOTO        TECLA_N1
260                   GOTO        TECLA_N2
261                   GOTO        TECLA_N3
262                   GOTO        TECLA_N4
263                   ;---------------------
264   TECLA_N1          CLRW                            ; Limpio el acumulador.
265                   ADDWF       TECLA, SAVE_IN_W          ; Paso el contenido de TECLA al acumulador (sin perder TECLA).
266                   ADDWF       TECLA_1, SAVE_IN_F        ; Paso el contenido del acumulador a la TECLA_1.
267                   GOTO        END_PESO_DERIVADO
268                   ;---------------------
269   TECLA_N2          CLRW                            ; Limpio el acumulador.
```

```
270                 ADDWF       TECLA, SAVE_IN_W          ; Paso el contenido de TECLA al acumulador (sin perder TECLA).
271                 ADDWF       TECLA_2, SAVE_IN_F        ; Paso el contenido del acumulador a la TECLA_2.
272                 GOTO        END_PESO_DERIVADO
273                 ;---------------------
274 TECLA_N3           CLRW                            ; Limpio el acumulador.
275                 ADDWF       TECLA, SAVE_IN_W          ; Paso el contenido de TECLA al acumulador (sin perder TECLA).
276                 ADDWF       TECLA_3, SAVE_IN_F          ; Paso el contenido del acumulador a la TECLA_3.
277                 GOTO        END_PESO_DERIVADO
278                 ;---------------------
279 TECLA_N4           CLRW                            ; Limpio el acumulador.
280                 ADDWF       TECLA, SAVE_IN_W          ; Paso el contenido de TECLA al acumulador (sin perder TECLA).
281                 ADDWF       TECLA_4, SAVE_IN_F        ; Paso el contenido del acumulador a la TECLA_4.
282                 ;---------------------
283 END_PESO_DERIVADO   RETURN
284
285 ;*****************************************************************************************************************
286 ; Subrutina de CONTRASEÑA.
287
288 CONTRASEÑA          ;---------------------               ; Este indica si CONTRASEÑA se ignora (en ese caso, sigue
    ↪ DISPLAY).
289                 CLRW                              ; Limpio el acumulador.
290                 ADDWF       POSICION, SAVE_IN_W       ; Paso el contenido de POSICION al acumulador (sin perder
                    ↪ POSICION).
291                 SUBLW       .4                     ; Testeo si el literal sustraido del acumulador es igual a 0.
292                 BTFSS       STATUS, ZERO              ; Reviso si el "flag Zero" se activo (skip if (ZERO) == 1 / next
                    ↪ if (ZERO) == 0).
293                 GOTO        END_CONTRASEÑA
294                 CLRF        POSICION                ; Borro el contenido de POSICION.
295                 ;---------------------                ; Dirreccionamiento indirecto.
296                 CLRW
297                 CLRF        FSR
298                 MOVLW       0x34
299                 MOVWF       FSR
300                 ;---------------------
301                 CLRW                              ; Limpio el acumulador.
302                 ADDWF       TECLA_1, SAVE_IN_W          ; Paso el contenido de TECLA_1 al acumulador (sin perder
                    ↪ TECLA_1).
303                 SUBWF       INDF, SAVE_IN_W           ; Testeo si el literal sustraido del acumulador es igual a 0.
304                 BTFSS       STATUS, ZERO
305                 GOTO        CONTRASEÑA_FALSA
306                 ;---------------------
307                 CLRW
308                 INCF        FSR
309                 ADDWF       TECLA_2, SAVE_IN_W          ; Paso el contenido de TECLA_2 al acumulador (sin perder
                    ↪ TECLA_2).
310                 SUBWF       INDF, SAVE_IN_W           ; Testeo si el literal sustraido del acumulador es igual a 0.
311                 BTFSS       STATUS, ZERO
312                 GOTO        CONTRASEÑA_FALSA
313                 ;---------------------
314                 CLRW
315                 INCF        FSR
316                 ADDWF       TECLA_3, SAVE_IN_W          ; Paso el contenido de TECLA_3 al acumulador (sin perder
                    ↪ TECLA_3).
317                 SUBWF       INDF, SAVE_IN_W           ; Testeo si el literal sustraido del acumulador es igual a 0.
318                 BTFSS       STATUS, ZERO
319                 GOTO        CONTRASEÑA_FALSA
320                 ;---------------------
321                 CLRW                              ; Limpio el acumulador.
322                 INCF        FSR
323                 ADDWF       TECLA_4, SAVE_IN_W          ; Paso el contenido de TECLA_4 al acumulador (sin perder
                    ↪ TECLA_4).
324                 SUBWF       INDF, SAVE_IN_W           ; Testeo si el literal sustraido del acumulador es igual a 0.
325                 BTFSS       STATUS, ZERO
326                 GOTO        CONTRASEÑA_FALSA
327                 ;---------------------                ; La contraseña es igual al codigo ingresado.
328                 BSF         PORTA, 7                ; Activo el Solenoide o Cerradura...
329                 CALL        DELAY_500ms
330                 CALL        DELAY_500ms
331                 BCF         PORTA, 7                ; Desactivo el Solenoide.
332                 CLRF        ERROR_ACUM
333                 GOTO        FLAG_P_V
```

```
334                 ;---------------------              ; Testeo si la contraseña fue errada 3 veces.
335   CONTRASEÑA_FALSA    CLRW
336                 INCF      ERROR_ACUM, SAVE_IN_F      ; Incremento el contador de errores de ingreso de codigo.
337                 ADDWF      ERROR_ACUM, SAVE_IN_W      ; Paso el contenido de ERROR_ACUM al acumulador (sin perder
                    ↪  ERROR_ACUM).
338                 SUBLW      .3                     ; Testeo si el literal sustraido del acumulador es igual a 0.
339                 BTFSS      STATUS, ZERO              ; Reviso si el "flag Zero" se activo (skip if (ZERO) == 1 / next
                    ↪  if (ZERO) == 0.
340                 GOTO       FLAG_P_V
341                 ;---------------------              ; Contraseña equívoca 3 veces.
342                 CLRF      ERROR_ACUM
343                 BSF         STATUS, RP0              ; Banco 1 para Trisx.
344                 MOVLW      0xF7                  ; Pongo como salida RB3 (1111-0111).
345                 MOVWF      TRISB
346                 BCF         STATUS, RP0
347                 BSF         PORTB, 3               ; Activo el Zumbador.
348                 CALL      DELAY_500ms
349                 CALL      DELAY_500ms
350                 BCF         PORTB, 3               ; Desactivo el Zumbador.
351                 ;---------------------
352   FLAG_P_V          BSF       FLAG, PASSWORD_VERIFIED      ; Activo un Flag que indica que el codigo llego hasta
    ↪  aqui.
353   END_CONTRASEÑA      RETURN
354
355   ;*****************************************************************************************************************
356   ; Subrutina de DISPLAY.
357
358   DISPLAY             BCF        PORTA, 4               ; Primer display.
359                 BCF         PORTA, 6
360                 CLRW
361                 ADDWF      TECLA_1, SAVE_IN_W          ; Paso el contenido de TECLA_1 al acumulador (sin perder
                    ↪  TECLA_1).
362                 CALL       NUMERO_DISPLAY            ; Elije la configuracion de pines para mostrar el numero.
363                 CALL      DELAY_500us
364                 CLRW
365                 ;---------------------
366                 BCF         PORTA, 4                ; Segundo display.
367                 BSF         PORTA, 6
368                 ADDWF      TECLA_2, SAVE_IN_W          ; Paso el contenido de TECLA_2 al acumulador (sin perder
                    ↪  TECLA_2).
369                 CALL       NUMERO_DISPLAY            ; Elije la configuracion de pines para mostrar el numero.
370                 CALL      DELAY_500us
371                 CLRW
372                 ;---------------------
373                 BSF         PORTA, 4                ; Tercer display.
374                 BCF         PORTA, 6
375                 ADDWF      TECLA_3, SAVE_IN_W          ; Paso el contenido de TECLA_3 al acumulador (sin perder
                    ↪  TECLA_3).
376                 CALL       NUMERO_DISPLAY            ; Elije la configuracion de pines para mostrar el numero.
377                 CALL      DELAY_500us
378                 CLRW
379                 ;---------------------
380                 BSF         PORTA, 4                Cuarto display.
381                 BSF         PORTA, 6
382                 ADDWF      TECLA_4, SAVE_IN_W          ; Paso el contenido de TECLA_4 al acumulador (sin perder
                    ↪  TECLA_4).
383                 CALL       NUMERO_DISPLAY            ; Elije la configuracion de pines para mostrar el numero.
384                 CALL      DELAY_500us
385                 CLRW
386                 ;---------------------              ; Testeo el estado del FLAG que me indica si el codigo paso por
                    ↪  CONTRASEÑA.
387                 BTFSS      FLAG, PASSWORD_VERIFIED      ; Reviso si FLAG se activo (skip if (FLAG, 1) == 0 / next if
                    ↪  (FLAG, 1) == 1.)
388                 GOTO       END_DISPLAY
389                 ;---------------------              ; El siguiente codigo hace que se retrase por un tiempo
                    ↪  determinado
390                 INCF      CONTADOR_DPY, SAVE_IN_F    ; el "borrado" de los displays (sin que pare todo el programa).
                    ↪  Cuando
391                 CALL      DELAY_2_5ms               ; termina ese delay, se actualiza a 0 todos los digitos.
392                 ADDWF      CONTADOR_DPY, SAVE_IN_W
393                 SUBLW      .150
```

```
394                 BTFSS       STATUS, ZERO
395                 GOTO        END_DISPLAY
396                 CLRF        CONTADOR_DPY            ; Borro por las dudas el contador.
397                 ;---------------------
398                 CLRF        TECLA_1                 ; Limpio el contenido de las TECLA_X.
399                 CLRF        TECLA_2
400                 CLRF        TECLA_3
401                 CLRF        TECLA_4
402                 CLRW
403                 BCF           FLAG, PASSWORD_VERIFIED
404 END_DISPLAY          RETURN
405
406 ;-------------------------------------------------------------------------------------------------------------------
407 ; Sub-subrutina de numeros para display (se encarga de eligir la configuracion de pines para
408 ; mostrar el numero)
409 ; La siguiente subrutina esta pensada de forma en que PC tiene la direccion actual.
410
411 NUMERO_DISPLAY      ADDWF       PCL, 1                  ; Lo que esta en el acumulador (W) se lo sumo a PCL (F).
412                 GOTO        N0_DISPLAY
413                 GOTO        N1_DISPLAY
414                 GOTO        N2_DISPLAY
415                 GOTO        N3_DISPLAY
416                 GOTO        N4_DISPLAY
417                 GOTO        N5_DISPLAY
418                 GOTO        N6_DISPLAY
419                 GOTO        N7_DISPLAY
420                 GOTO        N8_DISPLAY
421                 GOTO        N9_DISPLAY
422                 ;---------------------              ; Numero 0.
423 N0_DISPLAY          BCF         PORTA, 0
424                 BCF           PORTA, 1
425                 BCF           PORTA, 2
426                 BCF           PORTA, 3
427                 GOTO        END_NUM
428                 ;---------------------              ; Numero 1.
429 N1_DISPLAY          BCF         PORTA, 0
430                 BCF           PORTA, 1
431                 BCF           PORTA, 2
432                 BSF           PORTA, 3
433                 GOTO        END_NUM
434                 ;---------------------              ; Numero 2.
435 N2_DISPLAY          BCF         PORTA, 0
436                 BCF           PORTA, 1
437                 BSF           PORTA, 2
438                 BCF           PORTA, 3
439                 GOTO        END_NUM
440                 ;---------------------              ; Numero 3.
441 N3_DISPLAY          BCF         PORTA, 0
442                 BCF           PORTA, 1
443                 BSF           PORTA, 2
444                 BSF           PORTA, 3
445                 GOTO        END_NUM
446                 ;---------------------              ; Numero 4.
447 N4_DISPLAY          BCF         PORTA, 0
448                 BSF           PORTA, 1
449                 BCF           PORTA, 2
450                 BCF           PORTA, 3
451                 GOTO        END_NUM
452                 ;---------------------              ; Numero 5.
453 N5_DISPLAY          BCF         PORTA, 0
454                 BSF           PORTA, 1
455                 BCF           PORTA, 2
456                 BSF           PORTA, 3
457                 GOTO        END_NUM
458                 ;---------------------              ; Numero 6.
459 N6_DISPLAY          BCF         PORTA, 0
460                 BSF           PORTA, 1
461                 BSF           PORTA, 2
462                 BCF           PORTA, 3
463                 GOTO        END_NUM
464                 ;---------------------              ; Numero 7.
```

```
N7_DISPLAY          BCF         PORTA, 0
                BSF         PORTA, 1
                BSF         PORTA, 2
                BSF         PORTA, 3
                GOTO      END_NUM
                ;--------------------              ; Numero 8.
N8_DISPLAY          BSF         PORTA, 0
                BCF         PORTA, 1
                BCF         PORTA, 2
                BCF         PORTA, 3
                GOTO      END_NUM
                ;--------------------              ; Numero 9.
N9_DISPLAY          BSF         PORTA, 0
                BCF         PORTA, 1
                BCF         PORTA, 2
                BSF         PORTA, 3
                ;--------------------
END_NUM             RETURN

;-----------------------------------------------------------------------------------------------------------------------
; Subrutina del Delay (base de 500 ms).

DELAY_500ms          MOVLW      .5
                MOVWF      _AUX11
_LOOP11             CALL      DELAY_100ms
                DECFSZ      _AUX11, 1
                GOTO      _LOOP11
                RETURN

;-----------------------------------------------------------------------------------------------------------------------
; Subrutina del Delay (base de 100 ms)

DELAY_100ms          MOVLW      .40
                MOVWF      _AUX1
_LOOP1             CALL      DELAY_2_5ms              ; El guion bajo lo tomo como un punto o una coma (2,5 ms).
                DECFSZ      _AUX1, 1
                GOTO      _LOOP1
                RETURN

;-----------------------------------------------------------------------------------------------------------------------
; Subrutina del Delay (base de 25 ms)

DELAY_25ms          MOVLW      .10
                MOVWF      _AUX22
_LOOP22             CALL      DELAY_2_5ms              ; El guion bajo lo tomo como un punto o una coma (2,5 ms).
                DECFSZ      _AUX22, 1
                GOTO      _LOOP22
                RETURN

;-----------------------------------------------------------------------------------------------------------------------
; Subrutina del Delay (base de 2,5 ms)

DELAY_2_5ms          MOVLW      .250
                MOVWF      _AUX2
_LOOP2             CALL      DELAY_10us               ; Retardo de 10 micro segundos (us).
                DECFSZ      _AUX2, 1
                GOTO      _LOOP2
                RETURN

;-----------------------------------------------------------------------------------------------------------------------
; Subrutina del Delay (base de 500 us)

DELAY_500us          MOVLW      .50
                MOVWF      _AUX33
_LOOP33             CALL      DELAY_10us
                DECFSZ      _AUX33, 1
                GOTO      _LOOP33
                RETURN

;-----------------------------------------------------------------------------------------------------------------------
; Subrutina del Delay (base de 10 us)
```

```
536
537 DELAY_10us          MOVLW       .2
538                 MOVWF       _AUX3
539 _LOOP3             DECFSZ      _AUX3, 1
540                 GOTO        _LOOP3
541                 NOP
542                 RETURN
543
544 ;-------------------------------------------------------------------------------------------------------------------
545 ; Rutina Principal
546
547 INICIO              CALL        CONFIGURACION
548 LOOP               CALL        TECLAS
549                 CALL        CONTRASEÑA
550 DPY                 CALL        DISPLAY
551                 GOTO        LOOP
552                 END
553
554
555 ------------------------------------------------------------
```

Código en C

Codigo principal

```
//-----------------------------------------------------------------------------------------------
//-----------------------------------------------------------------------------------------------
//-----------------------------------------------------------------------------------------------
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include "ConfigurationBitsC.h"

#define _XTAL_FREQ 4000000
#define fila1       RB0
#define fila2       RB1
#define fila3       RB2
#define ERROR       RB3
#define columna1    RB4
#define columna2    RB5
#define columna3    RB6
#define columna4    RB7
#define BCD_A       RA0
#define BCD_B       RA1
#define BCD_C       RA2
#define BCD_D       RA3
#define control_A   RA4
#define control_B   RA6
#define cerradura   RA7

void config() {
   // en el TRIS 1--> ENTRADA, 0 -- Salida
   TRISA = 0x20; // Pongo como salida de RA0 a RA4 + RA6.
   TRISB = 0xF0; // Pongo como salida de RB0 a RB3. Pongo como entrada de RB4 a RB7.
   PORTA = 0x00; // Pongo en LOW el estado logico de RA0 a RA7.
   PORTB = 0x00; // Pongo en LOW el estado logico de RB0 a RB7.
   nRBPU = 0;
}

void delay_ms(int milisegundos) { //hice mi propio delay ms porque queria poder usar delay_ms() con variables, este
↪  delay_ms() si acepta un delay con variables
   while (milisegundos > 0) {
      __delay_ms(1);
      milisegundos--;
   }
}

void imprimirNumero(int display, int numero) { //imprime un numero en el display indicado
   /*
   esta funcion tiene un funcionamiento sencillo, hay que pasarle dos parametros.
   El primer parametro es a que display queres imprimir
   El segundo parametro es que numero queres imprimir.
   O sea que si queres imprimir el numero 5 en el display 1, seria asi:
   imprimirNumero(1, 5)
   1 siendo el display
   5 siendo el numero a imprimir
    */
   switch (display) {
      case 0:
         control_A = 0;
         control_B = 0;
         break;
      case 1:
         control_A = 0;
         control_B = 1;
         break;
      case 2:
         control_A = 1;
         control_B = 0;
         break;
      case 3:
         control_A = 1;
```

```
67            control_B = 1;
68            break;
69        }
70        switch (numero) {
71           case 0:
72              BCD_A = 0;
73              BCD_B = 0;
74              BCD_C = 0;
75              BCD_D = 0;
76              break;
77           case 1:
78              BCD_A = 0;
79              BCD_B = 0;
80              BCD_C = 0;
81              BCD_D = 1;
82              break;
83           case 2:
84              BCD_A = 0;
85              BCD_B = 0;
86              BCD_C = 1;
87              BCD_D = 0;
88              break;
89           case 3:
90              BCD_A = 0;
91              BCD_B = 0;
92              BCD_C = 1;
93              BCD_D = 1;
94              break;
95           case 4:
96              BCD_A = 0;
97              BCD_B = 1;
98              BCD_C = 0;
99              BCD_D = 0;
100             break;
101          case 5:
102             BCD_A = 0;
103             BCD_B = 1;
104             BCD_C = 0;
105             BCD_D = 1;
106             break;
107          case 6:
108             BCD_A = 0;
109             BCD_B = 1;
110             BCD_C = 1;
111             BCD_D = 0;
112             break;
113          case 7:
114             BCD_A = 0;
115             BCD_B = 1;
116             BCD_C = 1;
117             BCD_D = 1;
118             break;
119          case 8:
120             BCD_A = 1;
121             BCD_B = 0;
122             BCD_C = 0;
123             BCD_D = 0;
124             break;
125          case 9:
126             BCD_A = 1;
127             BCD_B = 0;
128             BCD_C = 0;
129             BCD_D = 1;
130             break;
131       }
132    }
133
134
135    void activarAlarma(int tiempo_ms) { //esta funcion activa la alarma durante 1 segundo, es cuando te equivocas 3 veces
       ↪  en el codigo
136       ERROR = 1;
```

```
137     delay_ms(tiempo_ms);
138     ERROR = 0;
139  }
140
141  void activarCerradura(int tiempo_ms) { //activa la cerradura, es cuando pones bien el codigo
142     cerradura = true;
143     delay_ms(tiempo_ms);
144     cerradura = false;
145  }
146
147  int consultarFila() { //consulta que fila se pulso del teclado matricial y la retorna
148     int filaPulsada;
149     if (fila1 == false) {
150        filaPulsada = 0;
151     }
152      else if (fila2 == false) {
153        filaPulsada = 1;
154     }
155      else if (fila3 == false) {
156        filaPulsada = 2;
157     }
158      else {
159        ERROR = 1;
160     }
161     return filaPulsada;
162  }
163
164  int consultarColumna() { //consulta que columna se pulso del teclado matricial y la retorna
165     int columnaPulsada;
166     if (columna1 == false) {
167        columnaPulsada = 0;
168     }
169      else if (columna2 == false) {
170        columnaPulsada = 1;
171     }
172      else if (columna3 == false) {
173        columnaPulsada = 2;
174     }
175      else if (columna4 == false) {
176        columnaPulsada = 3;
177     }
178      else {
179        ERROR = 1;
180     }
181     return columnaPulsada;
182  }
183
184  int main() {
185     config();
186     //----------------------------
187     // Declaracion de variables
188     int tecladoMatricial[3][4] = {
189        {0, 1, 2, 3},
190        {4, 5, 6, 7},
191        {8, 9, -2, -2}
192     };
193     int teclaPulsada[4] = {0, 0, 0, 0};
194     int intentosFallidos = 0;
195     int cantidadMaximaIntentos = 3;
196     int contrasena[4] = {5, 7, 1, 3};
197      int contador = 0;
198     //----------------------------
199      while (1){ // Programa sin fin.
200          while(contador < 4) { // Hasta que se ingrese 4 digitos.
201              if ((fila1 == false) || (fila2 == false) || (fila3 == false)) { // Antirrebote.
202                  delay_ms(120);
203                  if ((fila1 == false) || (fila2 == false) || (fila3 == false)) { // Verificacón de Antirrebote.
204                      TRISB = 0xF0;
205                      PORTB = 0x00;
206                      int columnaPulsada = consultarColumna(); // Consigo el valor correspondiente de la columna para
↪  la matriz.
```

```
207                     TRISB = 0x07; // En binario 0000-0111.
208                     PORTB = 0x00;
209                     int filaPulsada = consultarFila(); // Consigo el valor correspondiente de la fila para la matriz.
210                     teclaPulsada[contador] = tecladoMatricial[filaPulsada][columnaPulsada]; // Consigo el valor de
    ↪ tecla.
211                     ++contador;
212                 }
213             }
214             for(int i = 0; i < 4; ++i){ // Siempre actualizo el display (hasta que se ingrese 4 digitos).
215                 imprimirNumero(i, teclaPulsada[i]);
216                 __delay_ms(1);
217             }
218         }
219         contador = 0;
220         if ((teclaPulsada[0] == contrasena[0]) && (teclaPulsada[1] == contrasena[1]) && (teclaPulsada[2] ==
    ↪ contrasena[2]) && (teclaPulsada[3] == contrasena[3])) {
221          activarCerradura(2000);
222       }
223         else { // Contraseña incorrecta.
224          ++intentosFallidos;
225             if (intentosFallidos == cantidadMaximaIntentos) { // 3 Errores.
226                 activarAlarma(2000);
227             }
228         }
229         for(int i = 0; i < 4; ++i){ // Borro las teclas.
230             teclaPulsada[i] = 0;
231         }
232     }
233    return (EXIT_SUCCESS);
234 }
```

Bitacoras personales

Con respecto al código en C, se empezó diseñando las funciones que serían de utilidad a la hora de realizar el código principal.

Las que primero se diseñaron fueron las funciones que se utilizan para imprimir en el display de 7 segmentos, la cual fue diseñada para tener versatilidad para poder imprimir cualquier numero en cualquier display. Lo único que le falta a esa función es poder alternar entre modo cátodo común y ánodo común.

Luego se diseñó el código que detecta que tecla se pulsó, y en un momento se hizo una función de delay_ms() personalizada, que permite pasar variables como argumentos.

Hubo problemas con la función de imprimir en los displays, y sucedió que teníamos los pines del decodificador mal colocados, entonces no recibía correctamente la información.

Se hicieron las funciones pensando que tenían que ser lo más facil de entender posible, con nombres descriptivos y claros. Una cosa que se tuvo en mente es usar la filosofía UNIX, que es hacer una instrucción que haga una sola cosa y que la haga bien.

El codigo en assembler fue dividido en multiples instrucciones, fue creado en base a un código preliminar hecho en pseudo-código.

Simulaciones en Proteus

Assembler

- https://www.loom.com/share/57d7351ca74c480face7ed5f56e2720c?sharedAppSource=personal_library

- https://www.loom.com/share/4943de01fee7459e9b8c88f70fd844cd?sharedAppSource=personal_library

C

- https://www.loom.com/share/e2184dde78ff4f12a691476db8c95cd1