Universidad Nacional Autónoma de México

Facultad de Ingeniería

Microcomputadoras

Profesor: M.I. Rubén Anaya García

Proyecto Final. Cerradura Electrónica

Suárez Espinoza Mario Alberto

Semestre 2020-2

Entrega: viernes 14 de agosto de 2020

Requerimientos:

Requerimientos mínimos del proyecto

1. Utilizar al menos una fuente de interrupción.
   Se utiliza una fuente de interrupción por recepción de datos por puerto serial, para abrir la cerradura sin necesidad de contraseña.
2. Empleo de tres periféricos distintos.
   Se utilizan como periféricos:
   - Teclado matricial 4x4
   - Pantalla LCD 16x2
   - Servomotor
   - Teléfono celular con llave para abrir cerradura sin contraseña

Requerimientos propios de la cerradura electrónica

1. La cerradura electrónica permite abrir o cerrar una cerradura de puerta mediante el uso de un servomotor.
2. Para poder abrir la cerradura, el sistema solicita una contraseña. En caso de que esta sea válida, la cerradura se abre y muestra un mensaje de bienvenida.
3. Para ingresar los datos del usuario y de la contraseña se usará un teclado matricial de 4x4.
4. Para desplegar los mensajes del sistema, se usará una pantalla LCD 16x2.
5. El sistema contará con la opción de que el usuario cambe su contraseña.
6. El sistema contará adicionalmente con comunicación bluetooth para poder abrir la cerradura mediante una aplicación en Android sin la necesidad de tener contraseña.

Diseño:

Descripción de la solución

El programa se basa en dos estados. Estado “Contraseña Valida” y “Contraseña Inválida”. En el estado “Contraseña Inválida” se configura al servomotor en -90° y se pregunta por la contraseña, mientras que en el estado “Contraseña Valida” se configura al servomotor en 90° y se permite cambiar la contraseña.

El servomotor estará conectado físicamente a la cerradura. El servomotor configurado a -90° implica tener la cerradura cerrada, mientras que a 90° implica tenerla abierta.

Se ingresan los datos mediante un teclado matricial 4x4.

Además, estando en cualquier estado, se puede conectar la aplicación Android mediante bluetooth, y al momento de presionar el candado de la app, permite configurar el sistema para que se cambie al estado de “Contraseña Válida”.

Pseudocódigo del sistema

```
Inicio
Configura servo a -90°
Imprime “Password:”
Mientras longitud(Password_ingresado[]) ≤ 4
  Lee el teclado
  Si la entrada = A
    Enqueue entrada_anterior en Password_ingresado[]
    Muestra entrada_anterior en LCD
  Si la entrada = B
    Vacía Password_ingresado[]
Salta a Valida_password
Fin inicio
```

```
Valida_password
  Si Password_ingresado[] = Password_guardado[]
    Imprime “Welcome:”
    Configura servo a 90°
    Lee el teclado
    Si la entrada = A
      Salta a Inicio
    Si la entrada = B
      Salta a Cambia_contraseña
  En caso contrario
    Imprime “Invalid Password”
    Salta a Inicio
Fin Valida_password

Cambia_contraseña
  Imprime “Change Password:”
Mientras longitud(Password_ingresado[]) ≤ 4
  Lee el teclado
  Si la entrada = A
    Enqueue entrada_anterior en Password_ingresado[]
    Muestra entrada_anterior en LCD
  Si la entrada = B
    Vacía Password_ingresado[]
Password_guardado[] ← Password_ingresado[]
Salta a inicio
Fin Cambia_contraseña

Interrupción_Recibe_Datos_Serial
  Password_ingresado[] ← Password_guardado[]
Fin Interrupción_Recibe_Datos_Serial
```

Manual de operación

Uso de la cerradura

La cerradura posee una contraseña de longitud 4 (4 caracteres).

En un principio la cerradura se mantiene cerrada hasta que se ingrese la contraseña.

La contraseña se ingresa mediante un teclado matricial 4x4 y posee las siguientes teclas: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, * y #.

Las teclas A y B son especiales.

A significa “Aceptar”, y permite aceptar tanto los mensajes, como aceptar que se quiere ingresar un caracter.

B significa “Borrar”, y permite borrar los caracteres ingresados hasta el momento.

La contraseña por defecto es 1 2 3 4

Una vez ingresada la cerradura muestra un mensaje de bienvenida “Welcome!”.

Para salir de este mensaje se puede presionar la tecla A para aceptar y volver a cerrar la cerradura, o bien, la tecla B para cambiar la contraseña actual.

En caso de seleccionar B, se solicitará ingresar la nueva contraseña.

Olvido de contraseña – Apertura sin contraseña

En caso de olvidar la contraseña, se puede utilizar la app de recuperación. Esta app debe estar instalada en el dispositivo Android.

Para abrir la cerradura se abre la aplicación y se presiona el botón de bluetooth.

Dentro de la lista de dispositivos se selecciona a “Bluetooth HC05 masues”.

A continuación, se presiona el botón del candado. Esto además de abrir la cerradura sin necesidad de ingresar la contraseña, permite también cambiarla por una nueva siguiendo los pasos descritos previamente.

Comentarios:

El proceso para seleccionar un proyecto adecuado fue un poco difícil, debido a que tenia que ser un proyecto que pudiese realizarse en un periodo de tiempo no muy largo y además tenia que poder realizarse con los componentes electrónicos que tengo disponible en mi casa. Seleccione realizar una cerradura electrónica porque consideraba que en ella podría aplicar la mayoría de los conceptos relacionados al funcionamiento del PIC16F877A, tales como los puertos paralelos, la pantalla LCD 16x2, PWM (para el control del servomotor), comunicación serial (para la comunicación con la llave Android) e interrupciones. Las cerraduras electrónicas son proyectos comunes de electrónica y microcontroladores, pero, en la mayoría de estos proyectos se ocupan plataformas con lenguajes de alto nivel, el más famoso es Arduino, así que realizar el proyecto en ensamblador además de significar un reto por el propio lenguaje, permite ser un proyecto distinto a la mayoría que se pueden encontrar en internet.

La implementación de algoritmos en lenguaje ensamblador no es directa, debido a la limitada cantidad de instrucciones que existen, sin embargo, gracias al trabajo realizado durante el semestre con proyectos anteriores, considero que obtuve la experiencia necesaria para afrontar proyectos de esta índole.

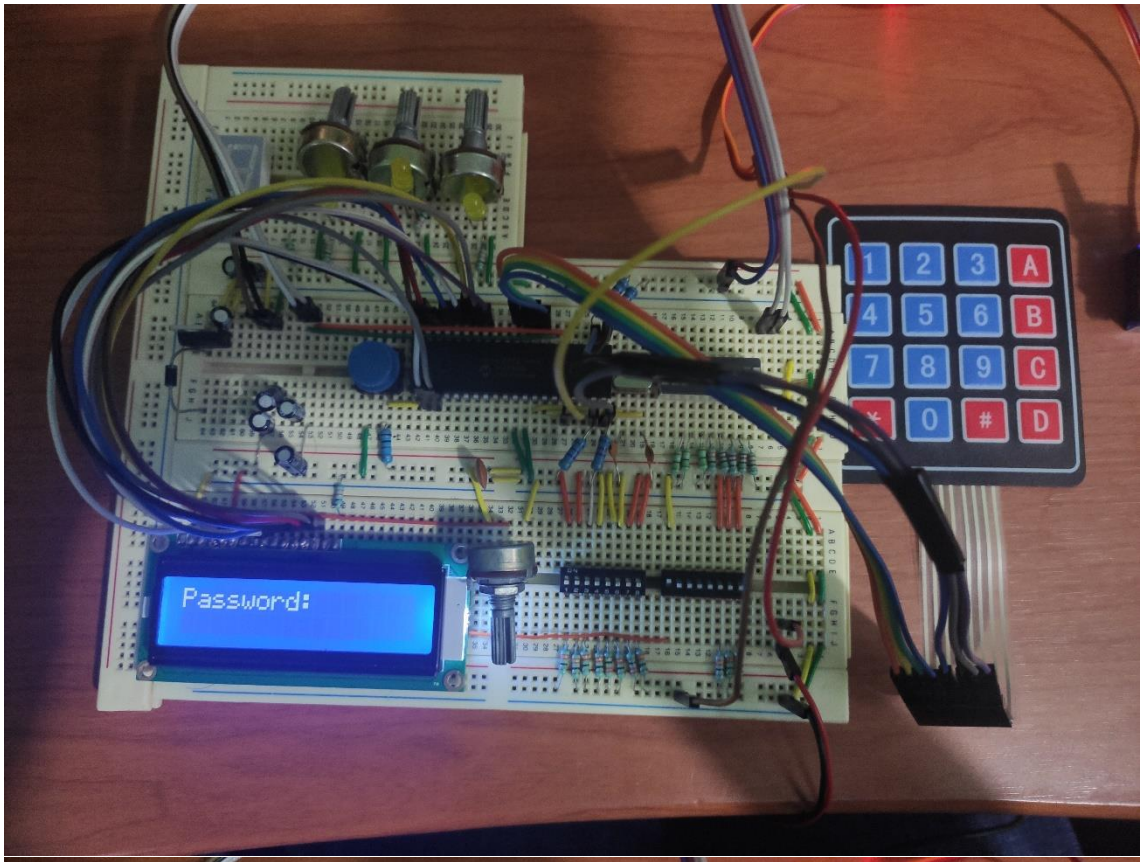
Dentro de algunos desafíos que me encontré al desarrollar el proyecto fue investigar sobre el funcionamiento y uso del teclado matricial 4x4, pues este nunca lo había ocupado, a pesar de ello, pude trasladar su funcionamiento a ensamblador con éxito.

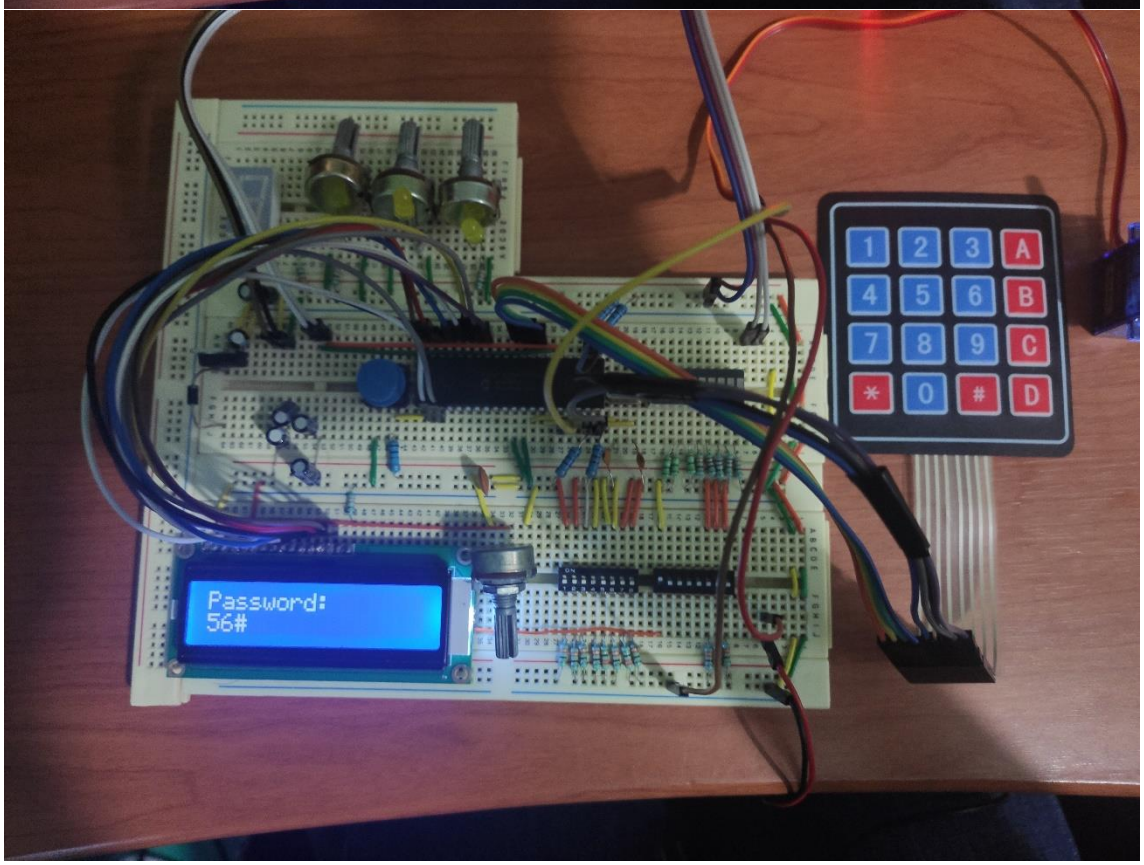
El enfoque de desarrollo que implementé fue incremental, lo que quiere decir que fui agregando de una en una las funciones de la cerradura, y para ello me apoyé de la herramienta Git. Considero que este enfoque es uno de mejores para proyectos de este estilo, pues permite tener entregas de valor en tiempos cortos.

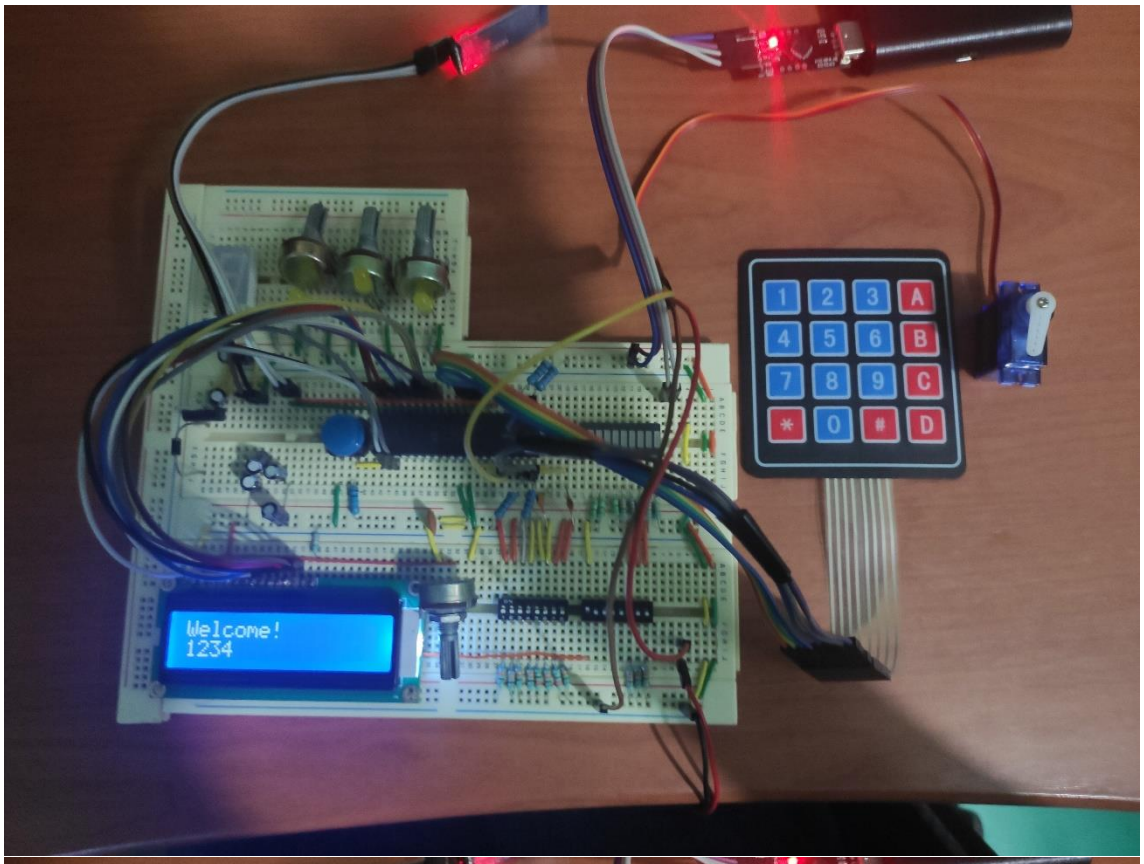
Por último, me gustaría decir que el proyecto aún puede escalarse y mejorase. Una de estas mejoras es guardar la contraseña en la memoria EEPROM, para que esta no se pierda cuando se desconecta la alimentación, mientras que otra mejora sería la implementación en un entorno real (en una puerta).

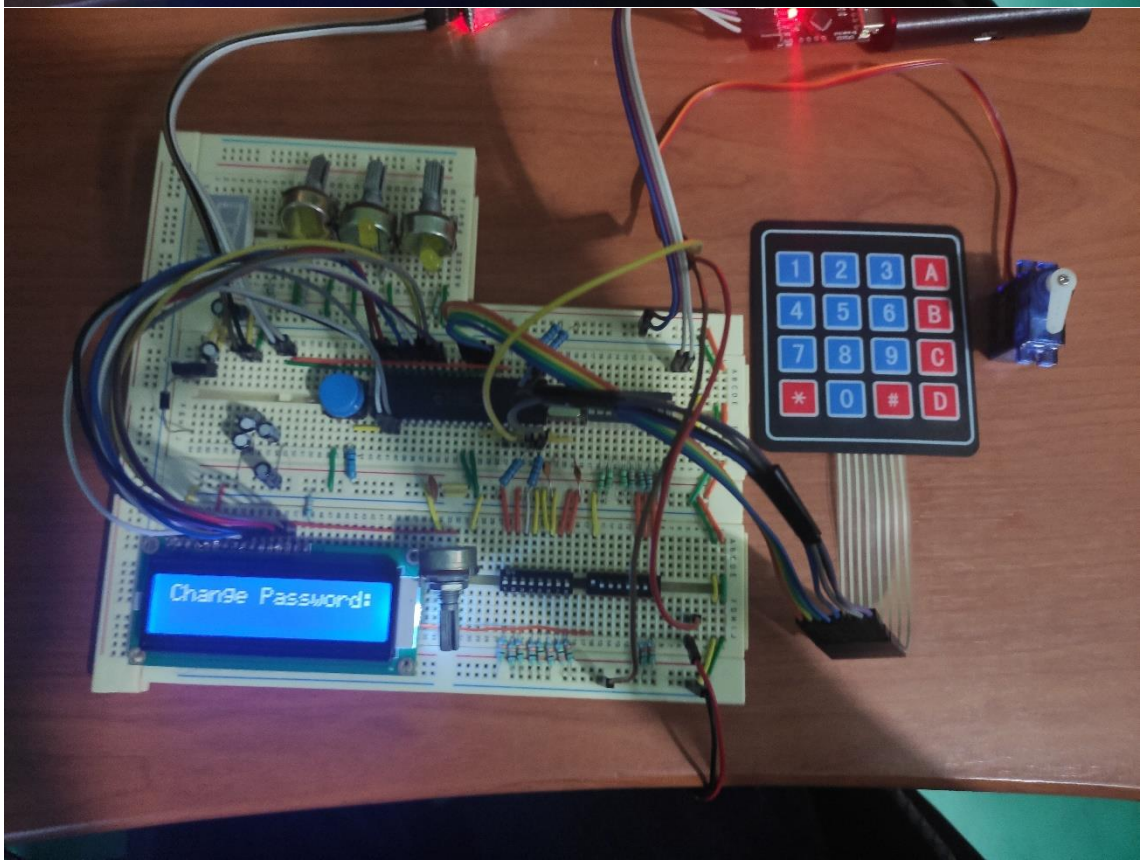
Resultados obtenidos:

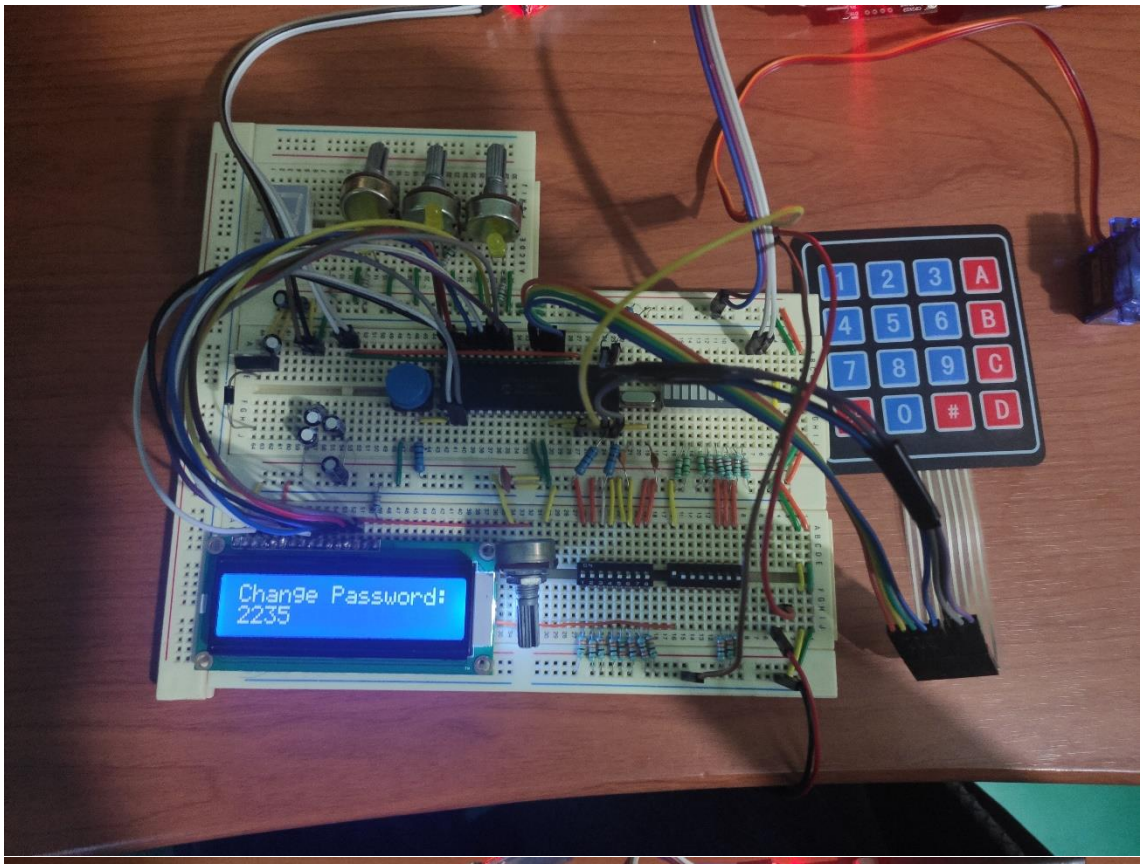
Se agregan imágenes de su funcionamiento

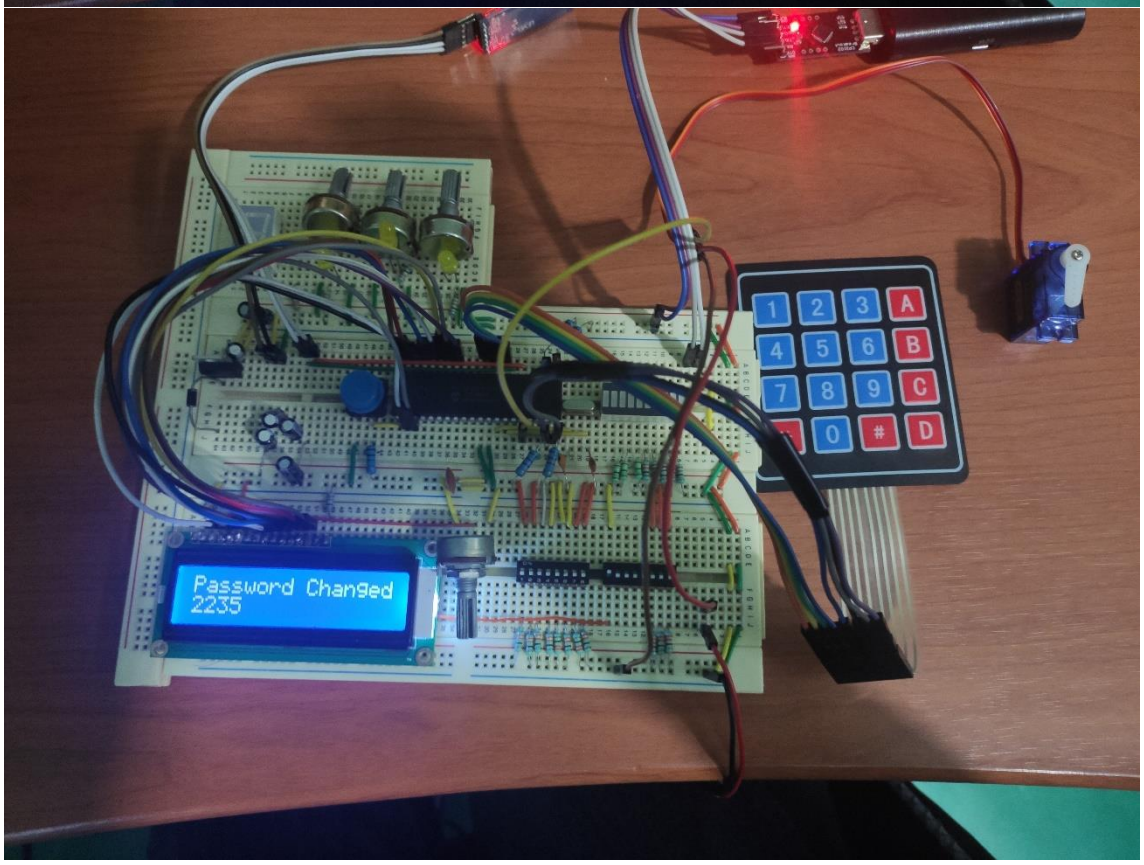


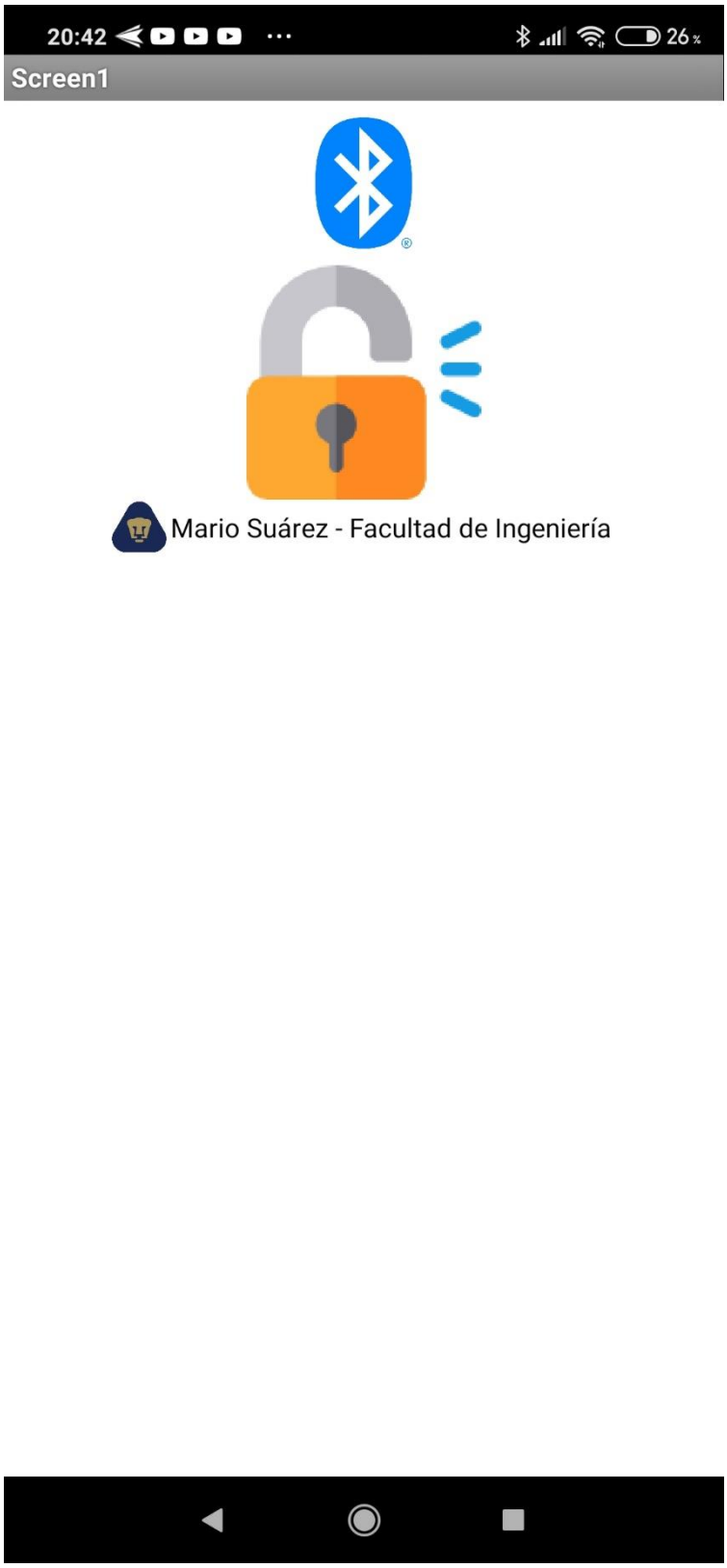



Welcome!
1234
1 2 3 A
4 5 6 B
7 8 9 C
* 0 # D


Change Password:
1 2 3 A
4 5 6 B
7 8 9 C
* 0 # D


Change Password:
2235
1 2 3 A
4 5 6 B
7 8 9 C
0 # D


Password Changed
2235
1 2 3 A
4 5 6 B
7 8 9 C
0 # D



Se puede ver una demostración de su funcionamiento en el siguiente enlace: https://youtu.be/ZjBsDhLZr7E

Código en ensamblador:

```
; Mario Suárez
; masues64@gmail.com
; PORTB BUS DE DATOS B0-D0 ... B7-D7
; RS - A0
; E - A1
; R/W - GND

;Puerto A: Control display (2 pines de 6)
;Puerto B: Datos display (8 pines de 8)
;Puerto C: CCP2 para PWM, Puerto serie RX y TX  (3 pines de 8)

  processor 16f877
  include<p16f877.inc>

  cblock h'20'
  ; LCD
  valorRet    ; Valores utilizados para generar un retardo por ciclos
  valorRet1
  valorRet2

  ; Keypad
  tecla ; Registro utilizado para almacenar la tecla que fue presionada
  tecla_anterior; Registro que almacena la última tecla presionada

  ; Password
  pass1 ; Almacena teclas ingresadas por el usuario para el password
  pass2
  pass3
  pass4
  long_pass

  ;Registros para almacenar el password secreto
  mem_pass1
  mem_pass2
  mem_pass3
  mem_pass4

  endc

  org 0
  goto inicio ; Vector de reset
  org 4
  goto interrupciones
  org 5
inicio
  bsf STATUS, RP0 ; Cambia al banco 1
```

```
  ; Configuración de puertos paralelos
  clrf TRISB ; Puerto B como salida
  movlw h'0F'; PORTD_H como salida y PORTD_L como entrada
  movwf TRISD
  movlw h'07'
  movwf ADCON1 ;Configura a PORTA como puerto digital
  clrf TRISA  ;Configura a PORTA como salida
  bcf TRISC,1; coloca el bit 1 del puerto C como salida (pin CCP2)

  ;Configuración de PWM
  movlw D'255'
  movwf PR2; PR2 <- d'255'

  ;Configuración de puerto serie asíncrono
  bsf TXSTA,BRGH ; BRGH <- 1. Selecciona alta velocidad de baudios
  movlw D'129' ; Coloca d'129' en el registro SPBRG. Registro para config
urar
  ; la velocidad de comunicación. En este caso, se trata de 9600[bauds]
  movwf SPBRG
  bcf TXSTA,SYNC ; SYNC <- 0. Configura el modo asíncrono
  bsf TXSTA,TXEN ; TXEN <- 1. Activa la transmisión

  ;Configuración de interrupciones
  bsf INTCON,GIE; Habilita interrupciones generales
  bsf INTCON,PEIE; Habilita interrupción de segundo bloque
  bsf PIE1,RCIE; Habilita interrupción por recepción de datos serial


  bcf STATUS, RP0 ; Regresa al banco 0

  ;Configuración de contraseña en memoria de datos
  movlw a'1'
  movwf mem_pass1
  movlw a'2'
  movwf mem_pass2
  movlw a'3'
  movwf mem_pass3
  movlw a'4'
  movwf mem_pass4

  ;Configura a CCP2 como PWM
  movlw B'00001100'
  movwf CCP2CON; CCP2CON <- b'00001100'

  ;Configura a al timer2 para PWM
  ;Configrua un postescala de 1:1
  ;Enciende al timer2
  ;Configura una preescala de 16
```

```
  movlw B'00000111'
  movwf T2CON; T2CON <- b'00000111'

  ; Confiugración de puerto serie asíncrono
  bsf RCSTA,SPEN ; Habilita el puerto serie (configura RX y TX como puert
os serie)
  bsf RCSTA,CREN ; Configura la recepción continua de datos

  call inicia_lcd

inicio_conf:
  clrf tecla; inicializa a tecla con 0
  clrf tecla_anterior
  clrf long_pass
  movlw pass1
  movwf FSR; FSR <- dirección de pass1
  call ret100ms; Retardo de 200 milisegundo para evitar rebotes en el tec
lado
  call ret100ms

  call despliega_mensaje

  ;Configura el tiempo en alto de la señal PWM en CCP2
  MOVLW D'20'
  MOVWF CCPR2L

  movlw h'c0'
  call comando ;Se posiciona en la segunda fila de la LCD para iniciar es
critura

main:
  ; Primero valida que se hayan ingresado 4 caracteres del password
  movlw d'4'
  xorwf long_pass, 0
  btfsc STATUS, Z; Si long_pass != d'4' salta
  goto valida_pass; Si la longitud es 4, salta a validar password

  ; Si aun no se ingresan los 4 caracteres, lee el teclado
  call lee_teclado
  movlw a'B'
  xorwf tecla, 0
  btfsc STATUS, Z; Si tecla != a'B' salta
  goto inicio_conf; tecla = a'B', borra la entrada actual de datos
  movlw a'A'
  xorwf tecla, 0
  btfsc STATUS, Z; Si tecla != a'A' salta
  goto w_lcd; tecla = a'A', escribe la tecla anterior
  movf tecla, 0
  movwf tecla_anterior
```

```
  goto main

valida_pass:
  movf mem_pass1, 0
  xorwf pass1, 0
  btfss STATUS, Z; Si pass1 = mem_pass1 salta para verificar el siguiente
 número
  goto pass_invalido

  movf mem_pass2, 0
  xorwf pass2, 0
  btfss STATUS, Z; Si pass1 = mem_pass2 salta para verificar el siguiente
 número
  goto pass_invalido

  movf mem_pass3, 0
  xorwf pass3, 0
  btfss STATUS, Z; Si pass1 = mem_pass3 salta para verificar el siguiente
 número
  goto pass_invalido

  movf mem_pass4, 0
  xorwf pass4, 0
  btfss STATUS, Z; Si pass1 = mem_pass4 salta para verificar el siguiente
 número
  goto pass_invalido

  ; En caso de que sea válido el password, muestra un mensaje y espera
  movlw h'80'
  call comando
  movlw a'W'
  call datos
  movlw a'e'
  call datos
  movlw a'l'
  call datos
  movlw a'c'
  call datos
  movlw a'o'
  call datos
  movlw a'm'
  call datos
  movlw a'e'
  call datos
  movlw a'!'
  call datos
  movlw a' '
  call datos
```

```
  MOVLW D'250'
  MOVWF CCPR2L

  call lee_teclado
  movlw a'B'
  xorwf tecla, 0
  btfsc STATUS, Z; Si tecla != a'B' salta
  goto cambia_password
  movlw a'A'
  xorwf tecla, 0
  btfss STATUS, Z; Si tecla = a'A' salta
  goto $-8; Se queda en un loop hasta que tecla = a'A' o tecla = a'B'
  goto inicio_conf

pass_invalido:
  ; En caso de que sea inválido el password, muestra un mensaje y espera
  movlw h'80'
  call comando
  movlw a'I'
  call datos
  movlw a'n'
  call datos
  movlw a'v'
  call datos
  movlw a'a'
  call datos
  movlw a'l'
  call datos
  movlw a'i'
  call datos
  movlw a'd'
  call datos
  movlw a' '
  call datos
  movlw a'P'
  call datos
  movlw a'a'
  call datos
  movlw a's'
  call datos
  movlw a's'
  call datos
  movlw a'w'
  call datos
  movlw a'o'
  call datos
  movlw a'r'
  call datos
  movlw a'd'
```

```
  call datos


  call lee_teclado
  movlw a'A'
  xorwf tecla, 0
  btfss STATUS, Z; Si tecla = a'A' salta
  goto $-4; Se queda en un loop hasta que tecla = a'A'
  goto inicio_conf

w_lcd: ; Escribe el contenido de la tecla anterior
  ; Guarda el contenido de la tecla anterior en pass1, pass2, ... pass3
  movf tecla_anterior, 0
  movwf INDF; INDF <- tecla_anterior
  incf FSR
  incf long_pass

  movf tecla_anterior, 0
  call datos

  call lee_teclado
  movlw a'A'
  xorwf tecla, 0
  btfsc STATUS, Z; Si tecla != a'A' salta
  goto $-4; Se queda en un loop hasta que tecla != a'A'
  goto main

despliega_mensaje:
  movlw h'80'
  call comando
  movlw a'P'
  call datos
  movlw a'a'
  call datos
  movlw a's'
  call datos
  movlw a's'
  call datos
  movlw a'w'
  call datos
  movlw a'o'
  call datos
  movlw a'r'
  call datos
  movlw a'd'
  call datos
  movlw a':'
  call datos
  movlw a' '
```

```
  call datos
  movlw a' '
  call datos
  movlw a' '
  call datos
  movlw a' '
  call datos
  movlw a' '
  call datos
  movlw a' '
  call datos
  movlw a' '
  call datos
  movlw h'c0'
  call comando
  movlw a' '
  call datos
  movlw a' '
  call datos
  movlw a' '
  call datos
  movlw a' '
  call datos
  return

cambia_password:
  clrf tecla; inicializa a tecla con 0
  clrf tecla_anterior
  movlw mem_pass1
  movwf FSR; FSR <- dirección de mem_pass1
  clrf long_pass

  movlw h'80'
  call comando
  movlw a'C'
  call datos
  movlw a'h'
  call datos
  movlw a'a'
  call datos
  movlw a'n'
  call datos
  movlw a'g'
  call datos
  movlw a'e'
  call datos
  movlw a' '
  call datos
  movlw a'P'
```

```
  call datos
  movlw a'a'
  call datos
  movlw a's'
  call datos
  movlw a's'
  call datos
  movlw a'w'
  call datos
  movlw a'o'
  call datos
  movlw a'r'
  call datos
  movlw a'd'
  call datos
  movlw a':'
  call datos

  movlw h'c0'
  call comando
  ; Borra el password anterior de la pantalla
  movlw a' '
  call datos
  movlw a' '
  call datos
  movlw a' '
  call datos
  movlw a' '
  call datos
  movlw h'c0'
  call comando

ciclo_cambia_password:
  ; Primero valida que se hayan ingresado 4 caracteres del password
  movlw d'4'
  xorwf long_pass, 0
  btfsc STATUS, Z; Si long_pass != d'4' salta
  goto mensaje_passChanged; Si la longitud es 4, muestra un mensaje sobre
 pass cambiado

  ; Si aun no se ingresan los 4 caracteres, lee el teclado
  call lee_teclado
  movlw a'B'
  xorwf tecla, 0
  btfsc STATUS, Z; Si tecla != a'B' salta
  goto cambia_password; tecla = a'B', borra la entrada actual de datos
  movlw a'A'
  xorwf tecla, 0
  btfsc STATUS, Z; Si tecla != a'A' salta
```

```
  goto w_memPass; tecla = a'A', escribe la tecla anterior
  movf tecla, 0
  movwf tecla_anterior
  goto ciclo_cambia_password

w_memPass:
  ; Guarda el contenido de la tecla anterior en mem_pass1,  ... mem_pass3
  movf tecla_anterior, 0
  movwf INDF; INDF <- tecla_anterior
  incf FSR
  incf long_pass

  movf tecla_anterior, 0
  call datos

  call lee_teclado
  movlw a'A'
  xorwf tecla, 0
  btfsc STATUS, Z; Si tecla != a'A' salta
  goto $-4; Se queda en un loop hasta que tecla != a'A'
  goto ciclo_cambia_password

mensaje_passChanged:
  movlw h'80'
  call comando
  movlw a'P'
  call datos
  movlw a'a'
  call datos
  movlw a's'
  call datos
  movlw a's'
  call datos
  movlw a'w'
  call datos
  movlw a'o'
  call datos
  movlw a'r'
  call datos
  movlw a'd'
  call datos
  movlw a' '
  call datos
  movlw a'C'
  call datos
  movlw a'h'
  call datos
  movlw a'a'
  call datos
```

```
  movlw a'n'
  call datos
  movlw a'g'
  call datos
  movlw a'e'
  call datos
  movlw a'd'
  call datos

  call lee_teclado
  movlw a'A'
  xorwf tecla, 0
  btfss STATUS, Z; Si tecla = a'A' salta
  goto $-4; Se queda en un loop hasta que tecla = a'A'
  goto inicio_conf

inicia_lcd
  movlw h'30'
  call comando
  call ret100ms
  movlw h'30'
  call comando
  call ret100ms
  movlw h'38'
  call comando
  movlw h'0c'
  call comando
  movlw h'01'
  call comando
  movlw h'06'
  call comando
  movlw h'02'
  call comando
  return

comando
  movwf PORTB
  call ret200us
  bcf PORTA,0
  bsf PORTA,1
  call ret200us
  bcf PORTA,1
  return

datos
  movwf PORTB
  call ret200us
  bsf PORTA,0
  bsf PORTA,1
```

```
  call ret200us
  bcf PORTA,1
  call ret200us
  call ret200us
  return

ret200us
  movlw h'02'
  movwf valorRet1
r200us_lp
  movlw d'164'
  movwf valorRet
r200us_lp1
  decfsz valorRet,1
  goto r200us_lp1
  decfsz valorRet1,1
  goto r200us_lp
  return

ret100ms
  movlw h'03'
  movwf valorRet
r100ms_lp3
  movlw h'ff'
  movwf valorRet1
r100ms_lp2
  movlw h'ff'
  movwf valorRet2
r100ms_lp1
  decfsz valorRet2
  goto r100ms_lp1
  decfsz valorRet1
  goto r100ms_lp2
  decfsz valorRet
  goto r100ms_lp3
  return

lee_teclado:
    movf tecla, 0; w <- tecla
    ; Sirve para que en caso de no presionar tecla alguna, se mantenga
    ; la última que se presionó anteriormente

    ;Fila 1
    bsf PORTD, 7; Pin 7 equivale a fila 1 del Keypad. Inicia el escaneo
    btfsc PORTD, 3; Si PIN3 = 1, entonces se presionó tecla 1
    movlw a'1'
    btfsc PORTD, 2; Si PIN2 = 1, entonces se presionó tecla 2
    movlw a'2'
    btfsc PORTD, 1; Si PIN1 = 1, entonces se presionó tecla 3
```

```
    movlw a'3'
    btfsc PORTD, 0; Si PIN0 = 1, entonces se presionó tecla A
    movlw a'A'
    bcf PORTD, 7; Termina el escaneo de la fila 1

    ;Fila 2
    bsf PORTD, 6; Pin 6 equivale a fila 2 del Keypad. Inicia el escaneo
    btfsc PORTD, 3; Si PIN3 = 1, entonces se presionó tecla 4
    movlw a'4'
    btfsc PORTD, 2; Si PIN2 = 1, entonces se presionó tecla 5
    movlw a'5'
    btfsc PORTD, 1; Si PIN1 = 1, entonces se presionó tecla 6
    movlw a'6'
    btfsc PORTD, 0; Si PIN0 = 1, entonces se presionó tecla B
    movlw a'B'
    bcf PORTD, 6; Termina el escaneo de la fila 2

    ;Fila 3
    bsf PORTD, 5; Pin 5 equivale a fila 3 del Keypad. Inicia el escaneo
    btfsc PORTD, 3; Si PIN3 = 1, entonces se presionó tecla 7
    movlw a'7'
    btfsc PORTD, 2; Si PIN2 = 1, entonces se presionó tecla 8
    movlw a'8'
    btfsc PORTD, 1; Si PIN1 = 1, entonces se presionó tecla 9
    movlw a'9'
    btfsc PORTD, 0; Si PIN0 = 1, entonces se presionó tecla C
    movlw a'C'
    bcf PORTD, 5; Termina el escaneo de la fila 3

    ;Fila 4
    bsf PORTD, 4; Pin 4 equivale a fila 4 del Keypad. Inicia el escaneo
    btfsc PORTD, 3; Si PIN3 = 1, entonces se presionó tecla *. Por el mom
ento *=E
    movlw a'*'
    btfsc PORTD, 2; Si PIN2 = 1, entonces se presionó tecla 0
    movlw a'0'
    btfsc PORTD, 1; Si PIN1 = 1, entonces se presionó tecla #. Por el mom
ento #=F
    movlw a'#'
    btfsc PORTD, 0; Si PIN0 = 1, entonces se presionó tecla D
    movlw a'D'
    bcf PORTD, 4; Termina el escaneo de la fila 4

    movwf tecla; tecla <- W. W contiene la tecla leida

  return

interrupciones:
  btfss PIR1, RCIF; Pregunta si se ha terminado la recepción de datos
```

```
  retfie; No fue interrupción por recepción de datos, retorna interrupció
n

  movlw a'A'
  xorwf RCREG, 0;
  ; RCREG (datos recibidos durante la comunicación serial)
  btfss STATUS,Z; ¿RCREG = a'A'?
  retfie; No recibió una a'A'

  ;pone a longpass en 4 y configura los registros para que sea un pass vá
lido
  movlw d'4'
  movwf long_pass
  movf mem_pass1,0
  movwf pass1
  movf mem_pass2,0
  movwf pass2
  movf mem_pass3,0
  movwf pass3
  movf mem_pass4,0
  movwf pass4

  retfie

  end
```