

UNIVERSIDAD VERACRUZANA

Facultad de ingeniería Eléctrica y Electrónica



Actividad

Reporte Proyecto Final

Docente

Sánchez Vidal Adrián

Integrantes

Octavio Yael Giménez Nicolas

Eduardo Garcia Lopez

Miguel Ángel Rosas Serna

10 DE JUNIO DE 2023

REPORTE FINAL EQUIPO 9

Índice

| | |
|---|----|
| Activación y monitoreo remoto de un sistema domótico didáctico (tres variables) | 2 |
| Resumen..... | 2 |
| Introducción | 2 |
| Diagrama circuito | 2 |
| Materiales..... | 3 |
| Proceso de creación. | 3 |
| Código fuente | 6 |
| Conclusiones..... | 10 |
| Evidencia | 10 |
| Bibliografía..... | 11 |

Activación y monitoreo remoto de un sistema domótico didáctico (tres variables).

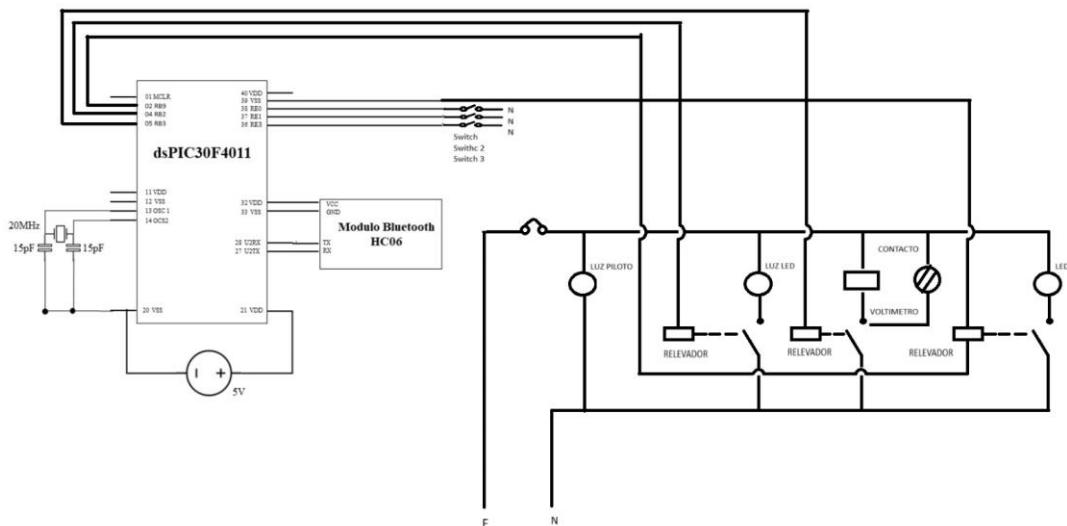
Resumen

Esta documentación detalla de manera general el armado, programada, y funcionamiento del proyecto final de la EE microprocesadores y microcontroladores, teniendo como fin el poder tener una idea general sobre el funcionamiento de los microcontroladores, su comportamiento con diferentes tipos de circuitos y componentes electrónicos, además de saber lo que se puede realizar con ellos.

Introducción

Para dicho proyecto lo que se llevó a cabo fue el análisis de la actividad de controlar 3 variables de manera remota, para posteriormente plasmar el circuito en un esquemático para posteriormente ser armado siguiendo la lógica del esquemático realizado, y así poder continuar con el lado de la programación usando como principal herramienta el módulo UART en conjunto con un dispositivo de bluetooth HC-06, y así cargarlo en el microcontrolador DSPIC30F4011, para al final hacer pruebas finales del funcionamiento correcto de cada uno de los dispositivos usados para el controlado remoto.

Diagrama circuito



Materiales

- 1 interruptor doble termomagnético de 16 amperios
- 5 metros de cable calibre 16 y 14 cada uno
- 2 regletas de terminales
- 1 toma de corriente de 110 V
- 1 socket
- 1 foco 110 V
- 2 barras led 110 V
- 1 indicador de voltaje digital
- 1 foco piloto (para verificar el energizado del circuito)
- 1 clavija
- 1 cable para usarlo como extensión con la clavija
- 3 switches
- 2 protoboard
- 3 relevadores
- 1 caja de conexiones

Proceso de creación.

Como primer paso se comenzaron a conseguir todo el material necesario para el armado del circuito el cual se menciona anteriormente, una vez conseguido se comenzó a analizar el esquemático para el armado del circuito tanto del protoboard como del cableado necesario para cada elemento que iba a ser controlado.

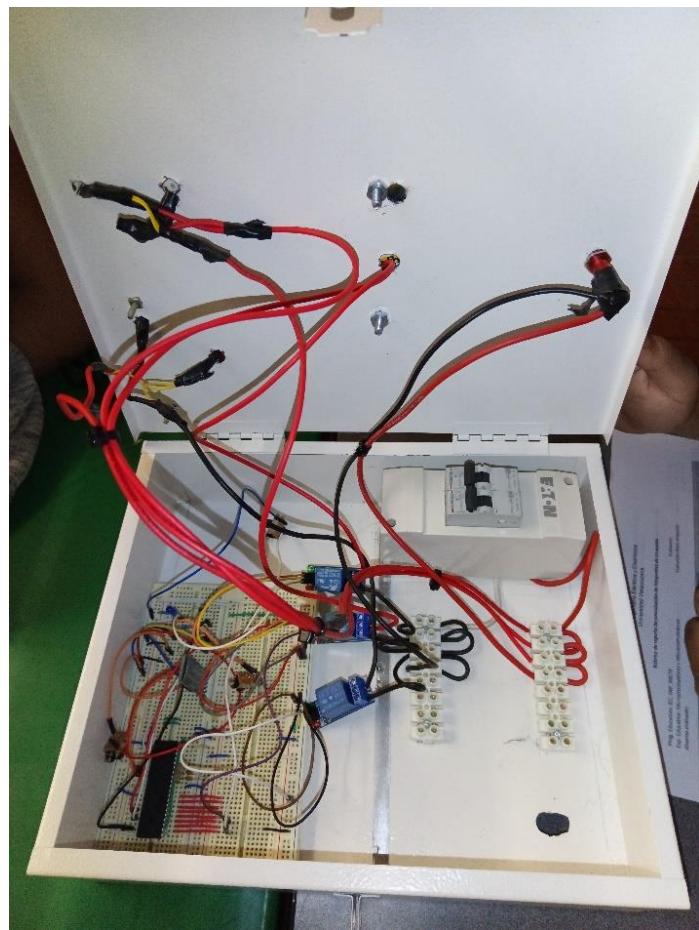
Se realizo el añadir en el protoboard las conexiones necesarias en cuando a los relevadores colocando cada pin del relevador en lugar correcto y el pin que se planeaba utilizar del microcontrolador en para la programación.

Se hicieron las conexiones del cableado de cada elemento a controlar con el cable de calibre 16 y 14, tomando en cuenta el neutro y fase, ya que para conectarlo al relevador se tenía que hacer un puenteo entre el relevador en el puerto comúnmente abierto directo a neutro del elemento a controlar, y en el comúnmente cerrado el neutro de la extensión, y de la extensión la fase directo al elemento a controlar, repitiendo lo mismo para cada elemento que se utilizó, con el añadido de que en uno de los elementos que fue un contacto se conectó al cableado del mismo, para cuando el relevador se activase este marcara el total de voltaje que estaba recibiendo.

Ya conectado el cableado correspondiente a lo necesario sin hacer conexiones directas a la extensión lo que siguió fue hacer los agujeros por donde pasaría el cableado para montar todo en la caja de conexiones montando tambien, el interruptor termomagnético junto con el foco piloto.

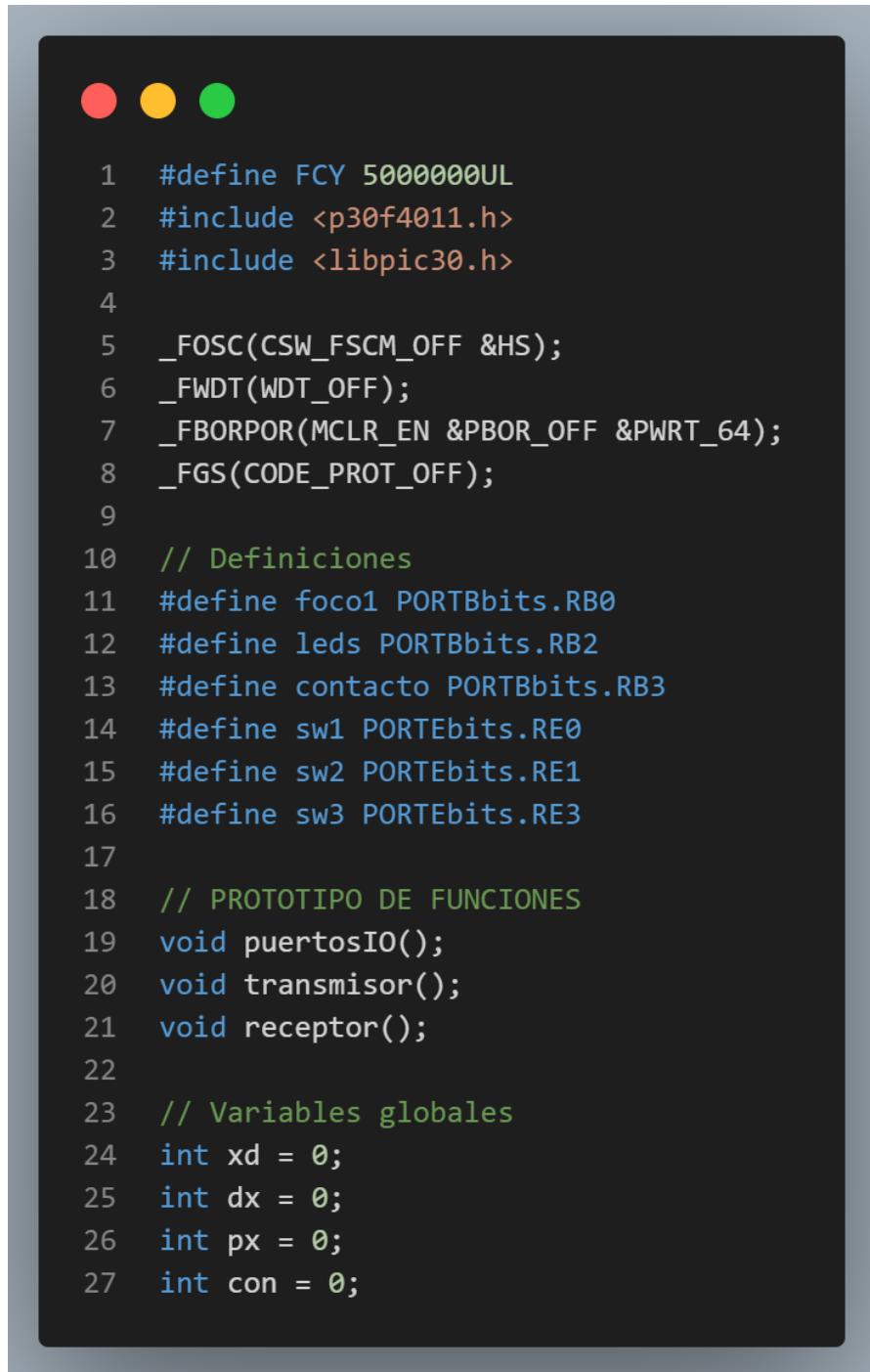
Una vez montado todo y atornillado se procedió a ahora si hacer la conexión del interruptor con la extensión directa, usando una parte de la regleta, el puenteo en los relevadores y con ayuda del interruptor termomagnético y el resto de las regletas, se hizo las conexiones de cada uno de los elementos que se usaron de manera que en teoría estuviera directa a la extensión como se mencionó antes.

REPORTE FINAL EQUIPO 9



Para finalizar se comenzó con el desarrollo del código el cual nos guiamos de prácticas realizadas en clases, las cuales fueron el uso del transmisor y receptor del módulo UART quedando de la siguiente manera.

Código fuente



```
1 #define FCY 5000000UL
2 #include <p30f4011.h>
3 #include <libpic30.h>
4
5 _FOSC(CSW_FSCM_OFF &HS);
6 _FWDT(WDT_OFF);
7 _FBORPOR(MCLR_EN &PBOR_OFF &PWRT_64);
8 _FGS(CODE_PROT_OFF);
9
10 // Definiciones
11 #define foco1 PORTBbits.RB0
12 #define leds PORTBbits.RB2
13 #define contacto PORTBbits.RB3
14 #define sw1 PORTEbits.RE0
15 #define sw2 PORTEbits.RE1
16 #define sw3 PORTEbits.RE3
17
18 // PROTOTIPO DE FUNCIONES
19 void puertosIO();
20 void transmisor();
21 void receptor();
22
23 // Variables globales
24 int xd = 0;
25 int dx = 0;
26 int px = 0;
27 int con = 0;
```

REPORTE FINAL EQUIPO 9

```
1 // FUNCION PRINCIPAL
2 void main()
3 {
4     // EJECUCION DE FUNCIONES
5     puertosIO();
6     transmisor();
7     receptor();
8
9     foco1 = 1; //foco apagado
10    leds = 0; //leds apagados
11    contacto = 1; //contacto cerrado
12
13    // BUCLE REPETITIVO
14    while (1)
15    {
16        // SELECCION DE OPCIONES RECIVIDAS EN EL U2RXREG
17        switch (xd)
18        {
19            // PRIMER VALOR PARA SELECCIONAR EL MODO DE OPERACION REMOTO
20            case 100:
21                // SELECCION DE OPCIONES RECIVIDAS EN EL U2RXREG PARA EL CONTROLADO DE LOS DISPOSITIVOS
22                switch (px)
23                {
24                    // CONTROL DE FOCO
25                    case 40:
26                        foco1 = 0;
27                        dx = 10;
28                        break;
29                    case 45:
30                        foco1 = 1;
31                        dx = 15;
32                        break;
33                    // CONTROL DE LEDS
34                    case 50:
35                        leds = 0;
36                        dx = 20;
37                        break;
38                    case 55:
39                        leds = 1;
40                        dx = 25;
41                        break;
42                    // CONTROL DE CONTACTO
43                    case 60:
44                        contacto = 0;
45                        dx = 30;
46                        break;
47                    case 65:
48                        contacto = 1;
49                        dx = 35;
50                        break;
51                }
52                break;
53        }
54    }
55}
```

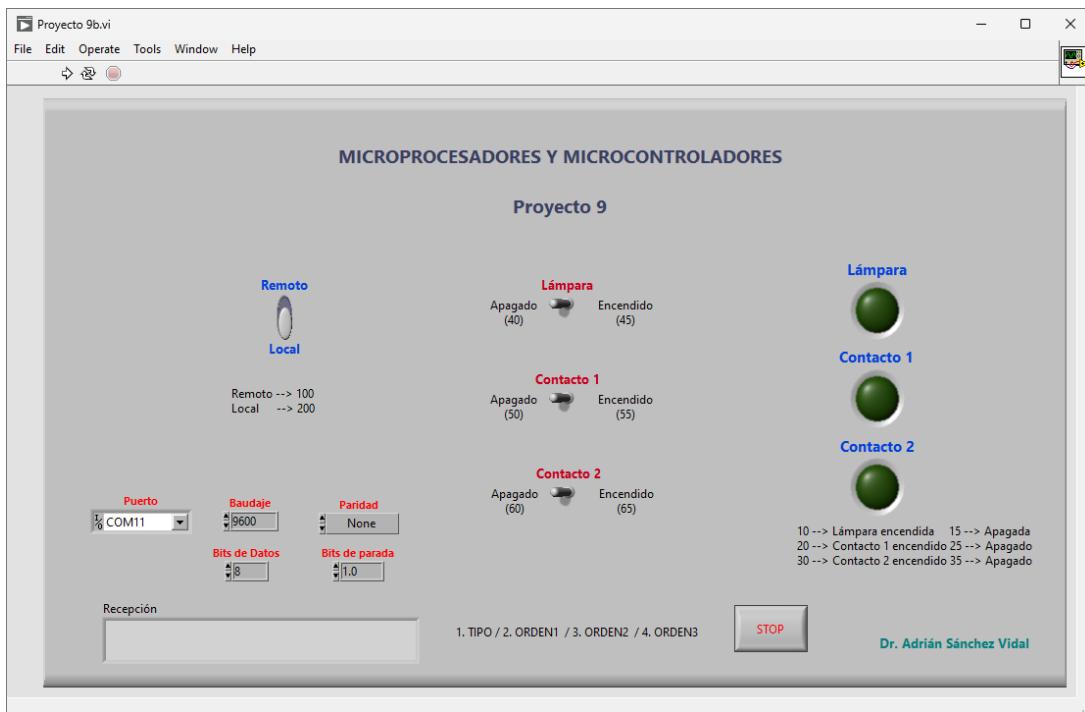
REPORTE FINAL EQUIPO 9

```
1 // SEGUNDO VALOR PARA SELECCIONAR EL MODO DE OPERACION LOCAL
2 case 200:
3
4     // CONTROL DEL FOCO
5     if (sw1 == 1)
6     {
7         foco1 = 0;
8         dx = 10;
9     }
10    else if (sw1 == 0)
11    {
12        foco1 = 1;
13        dx = 15;
14    }
15
16    // CONTROL DE LEDS
17    else if (sw2 == 1)
18    {
19        leds = 0;
20        dx = 20;
21    }
22    else if (sw2 == 0)
23    {
24        leds = 1;
25        dx = 25;
26    }
27
28    // CONTROL DE CONTACTO
29    else if (sw3 == 1)
30    {
31        contacto = 1;
32        dx = 30;
33    }
34    else if (sw3 == 0)
35    {
36        contacto = 0;
37        dx = 35;
38    }
39    break;
40}
41}
42}
```

REPORTE FINAL EQUIPO 9

```
● ○ ● ●
1 void puertosIO()
2 {
3     // relay's
4     TRISBbits.TRISB0 = 0; // PUERTO B0 COMO SALIDA
5     ADPCFGbits.PCFG0 = 1; // PUERTO B0 COMO DIGITAL
6     TRISBbits.TRISB2 = 0; // PUERTO B2 COMO SALIDA
7     ADPCFGbits.PCFG2 = 1; // PUERTO B2 COMO DIGITAL
8     TRISBbits.TRISB3 = 0; // PUERTO B3 COMO SALIDA
9     ADPCFGbits.PCFG3 = 1; // PUERTO B3 COMO DIGITAL
10
11    // sw
12    TRISEbits.TRISE0 = 1; // PUERTO E0 COMO ENTRADA
13    TRISEbits.TRISE1 = 1; // PUERTO E1 COMO ENTRADA
14    TRISEbits.TRISE3 = 1; // PUERTO E3 COMO ENTRADA
15 }
16
17 // FUNCION DE CONFIGURACION DE TRANSMISOR UART
18 void transmisor()
19 {
20     // Configuración del transmisor UART
21     U2MODE = 0;           // LIMPIAR EL REGISTRO U2MODE
22     U2STA = 0;           // LIMPIAR EL REGISTRO U2STA
23     U2BRG = 32;          // BAUD RATE DE 9600
24     U2MODEbits.PDSEL = 0; // 8 BITS DE DATO, SIN PARIDAD
25     U2MODEbits.STSEL = 0; // UN BIT DE PARO
26     IEC1bits.U2TXIE = 1; // HABILITAR INTERRUPCION DE TRANSMISOR
27     IPC6bits.U2TXIP = 4; // PRIORIDAD DE INTERRUPCION DE TRANSMISOR
28     U2STAbits.UTXISEL = 0; // INTERRUPCION DE TRANSMISOR POR CADA CARACTER
29     IFS1bits.U2TXIF = 0; // LIMPIAR BANDERA DE INTERRUPCION DE TRANSMISOR
30     U2MODEbits.UARTEN = 1; // HABILITAR EL MODULO UART
31     U2STAbits.UTXEN = 1; // HABILITAR TRANSMISOR
32 }
33
34 // FUNCION DE CONFIGURACION DE RECEPTOR UART
35 void receptor()
36 {
37     // Configuración del receptor UART
38     U2MODE = 0;           // LIMPIAR EL REGISTRO U2MODE
39     U2MODEbits.PDSEL = 0; // 8 BITS DE DATO, SIN PARIDAD
40     U2BRG = 32;          // BAUD RATE DE 9600
41     U2STA = 0;           // LIMPIAR EL REGISTRO U2STA
42     U2STAbits.URXISEL = 0; // INTERRUPCION DE RECEPTOR POR CADA CARACTER
43     U2MODEbits.STSEL = 0; // UN BIT DE PARO
44     IEC1bits.U2RXIE = 1; // HABILITAR INTERRUPCION DE RECEPTOR
45     IPC6bits.U2RXIP = 4; // PRIORIDAD DE INTERRUPCION DE RECEPTOR
46     IFS1bits.U2RXIF = 0; // LIMPIAR BANDERA DE INTERRUPCION DE RECEPTOR
47     U2MODEbits.UARTEN = 1; // HABILITAR EL MODULO UART
48     U2STAbits.UTXEN = 0; // DESHABILITAR TRANSMISOR
49 }
50
51 // ISR - RUTINAS DE SERVICIO A LA INTERRUPCION
52 // ISR TRANSMISOR
53 void __attribute__((interrupt, no_auto_psv)) _U2TXInterrupt(void)
54 {
55     U2TXREG = dx;
56     IFS1bits.U2TXIF = 0; // LIMPIAR BANDERA DE INTERRUPCION DE TRANSMISOR
57 }
58
59 // ISR RECEPTOR
60 void __attribute__((interrupt, no_auto_psv)) _U2RXInterrupt(void)
61 {
62     xd = U2RXREG;        // xd es la variable que se utiliza para procesar el primer valor
63     px = U2RXREG;        // px es la variable que se utiliza para procesar el segundo valor
64     IFS1bits.U2RXIF = 0; // LIMPIAR BANDERA DE INTERRUPCION DE RECEPTOR
65 }
```

Para finalizar se realizaron pruebas en nuestro circuito con ayuda de una interfaz que nos permitía conectarnos a un módulo bluetooth y en la cual seleccionar el modo remoto dentro del código entraba a la sección del código correspondiente para así poder encender o apagar cada relevador que hacía pasar o cortar la corriente para el funcionamiento e cada elemento montado en la caja, la interfaz fue diseñada con ayuda del profesor a cargo de la EE quedando de la siguiente manera.



Conclusiones

Como conclusión aunque en el proceso se encontraron muchos problemas para la conexión del módulo bluetooth con la interfaz, y con la configuración en la programación para cada relevador y en cableado general, aun así se concluyó el proyecto con la mayor cantidad de funcionamiento esperado, quedándonos con la experiencia de como con las herramientas adecuadas y un circuito lo suficientemente bueno y funcional podemos realizar diversas cosas con solo modificar un poco el circuito y el código fuente que se le carga al micro.

Evidencia

[Evidencia Proyecto final.mp4](#)

Microprocesadores Equipo 9

Bibliografía

(S/f-a). Microchip.com. Recuperado el 16 de diciembre de 2023, de <https://www.microchip.com/downloads/en/devicedoc/70135g.pdf>

(S/f-b). Microchip.com. Recuperado el 16 de diciembre de 2023, de <https://www.microchip.com/downloads/en/devicedoc/70046e.pdf>

Castaño Giraldo, S. A. [@SergioACGiraldo]. (2020, julio 27). Relé con Microcontrolador

PIC  [Manejo de CARGAS Grandes] # 053. Youtube.

<https://www.youtube.com/watch?v=F1vyiIDa2gc>