

Proyecto 27/2007
Estudio y diseño de dispositivos de ayuda en la acción a
distancia para Personas Mayores y Personas
Discapacitadas

INFORME FINAL

Investigador Principal: José María González de Durana y García

10 de diciembre de 2008

Resumen

El presente informe contiene en primer lugar una memoria final del trabajo realizado e informe final de resultados, incluyendo en un anexo una guía de utilización del equipo diseñado y construido para que pueda ser utilizada por los técnicos del CEAPAT para la puesta a punto de mandos a distancia adaptados a personas discapacitadas, en segundo lugar una memoria justificativa para dar cumplimiento lo establecido en la OM TAS 1588 de 2005 I+D en lo que se refiere a justificar la *liquidación de la subvención* (disposiciones decimonovena y vigésima) y en tercer lugar los correspondientes justificantes de gastos.

1. Informe final de resultados.

Memoria final del trabajo realizado

1.1 Objetivos previstos

El objetivo principal del proyecto ha sido el diseño de un MD que pueda ser operado por una persona con actuación manual limitada a través de un pulsador diseñado a la medida de su discapacidad, aprovechando en lo posible los dispositivos creados por el Area de Desarrollo Tecnológico del CEAPAT.

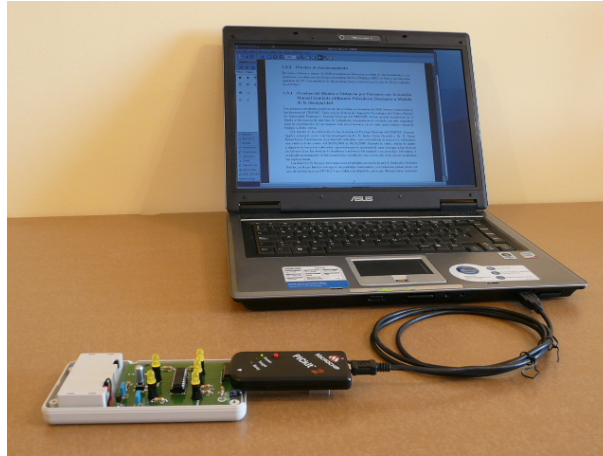
1.2 Metodología

La metodología empleada ha sido la que se emplea habitualmente en el ámbito de los desarrollos de aplicaciones con microprocesadores. El microprocesador empleado ha sido el microcontrolador PIC 16F628A de Microchip y para el desarrollo del hardware y software se ha utilizado el equipo de desarrollo PICkit-2 de Microchip (en vez del ICD2DV164006 que en principio habíamos previsto en el proyecto inicial) debido a su sencillez de uso y a su reducido coste, con el software de desarrollo MPLAB, suministrado por el mismo fabricante. Si bien para las primeras pruebas se utilizó un programador de Velleman VM134 conectado al puerto serie RS-232, pero pronto se pasó a utilizar el equipo PICkit-2 de Microchip, que aventaja al anterior por su simplicidad de conexión al ordenador y su facilidad de uso.

Este sistema ofrece la posibilidad de programar el PIC “en circuito”, es decir, permite la carga del programa en el PIC, tantas veces como se desee, desde el PICkit-2 al PIC conectado en el circuito, sin necesidad de sacarlo para ello. Esto representa una gran comodidad y fiabilidad práctica, ya que redundante en una mayor rapidez en la operación de carga del programa y elimina la posibilidad de dañar el PIC en las operaciones de extracción e inserción.

El equipo para programarlo ha sido un ordenador portátil en el que se ha instalado el software, suministrado por Microchip, necesario para programar el PIC.

También se han utilizado, para la medida de algunas señales eléctricas, 2 Osciloscopios digitales PicoScope 3224 PC, conectados al puerto USB.



Conexión del PICkit2 “en circuito”

El diseño ha tenido una dificultad adicional a la habitual de todo diseño con microprocesador, que es la del diseño de todas las funciones de un mando convencional utilizando un único pulsador. En esta parte nos fue de gran ayuda el modelado del sistema digital con Cartas de Estado (*Statecharts de D. Harel*), mediante el programa AnyLogic. Este método nos permitió la simulación del sistema completo en el ordenador y de este modo sólo montar el circuito cuando tuvimos la seguridad de que su funcionamiento era el adecuado.

1.3 Desarrollo del proyecto

El proyecto ha sido realizado entre julio de 2007 y diciembre de 2008 y su desarrollo ha seguido las fases que se indican a continuación.

1.3.1 Actualización de información

Durante el mes de julio de 2007 se procedió en primer lugar a la selección de dos alumnos “becarios” para, bajo la supervisión del investigador principal, formar el equipo humano encargado de la realización del proyecto. Estas personas son:

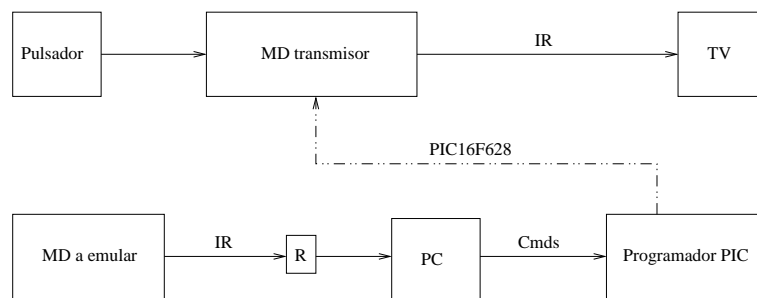
D. Alejandro Grandes López

D. Miguel Angel Rayo Cerrato

La primera tarea consistió en la búsqueda y actualización de información existente sobre el tema, o estado del arte, y la elección del método de trabajo para el diseño y desarrollo del equipo. Se buscó con denodado ahinco la publicación de documentos sobre la realización de equipos de comando a distancia adaptados a personas discapacitadas, encontrando algunas empresas dedicadas a ello, entre las que podemos destacar la australiana *NovitaTech-Regency*

Park, que suministra diversos equipos de mando a distancia adaptados a personas discapacitadas.

Entre las posibles alternativas de diseño, se escogió un desarrollo con microcontrolador tipo PIC de Microchip. Esta elección se debió a la gran versatilidad de estos dispositivos, a su facilidad de programación y a su reducido coste (del orden de unos 2 o 3 euros por chip). Las primeras pruebas las realizamos utilizando el programador Velleman VM-134, adquirido con fondos de nuestro departamento, y sirvieron para adquirir cierta destreza en el manejo de los dispositivos PIC. El PIC16F84 fue el primer candidato y con él se realizaron las primeras pruebas de placas de circuito con sencillas aplicaciones con pulsadores y diodos LED.



Durante el mes de septiembre nos dedicamos a experimentar con con señales de mandos a distancia de TV de diferentes marcas. Comparar las señales y grabarlas para luego tratar de reproducirlas. Pronto abandonamos la idea inicial de emular el funcionamiento de un mando a distancia cualquiera, debido a la enorme diversidad de códigos existentes y la imprecisión en la obtención de los mismos en algunos mandos. Por el contrario nos decantamos por diseñar un mando sólo válido para los códigos universales más difundidos: el RC5 de Phillips y el código SIRC de Sony. Pensamos que es mejor diseñar un mando que funcione bien con uno de estos códigos que diseñar un mando universal que funcione mal o a medias con muchos de ellos. La decisión fue debida en gran medida al hallazgo de la publicación de Micrichip AN1064, IR Remote Control Transmitter, de Tom Perme (de fecha 8 de Diciembre de 2006). En ella, su autor describe cómo realizar un transmisor de control remoto utilizando un PIC10F206, utilizando los protocolos RC5 de Phillips y SIRC de Sony.

1.3.2 Montaje del primer prototipo

Durante los meses de septiembre y octubre el equipo investigador procedió al diseño y montaje de los primeros diseños y se realizaron las primeras pruebas de hardware y software.

El 31 de octubre recibimos un e-mail del IMSERSO indicando la resolución provisional de concesión de 11.000 euros. El importe solicitado había sido de 16.640 euros. Durante el mes de noviembre continuamos realizando diferentes pruebas y a primeros de diciembre hicimos un desarrollo basado en las ideas de Tom Perme en la nota de aplicación de Microchip AN1064. Se montó una placa de circuito con un PIC16F628, cinco diodos LED, un diodo IR y un pulsador. La adaptación del código a este PIC no fue trivial, pero tras diversos intentos lo

conseguimos. El sistema se diseñó primero en simulación mediante el programa AnyLogic, utilizando una carta de estados (statechart de D.Harel) para emular el comportamiento del sistema de eventos discretos concebido. En el apéndice A puede verse la carta de estados.



Primer prototipo

Tras numerosas pruebas, el 27 de diciembre el primer prototipo funcionó adecuadamente. El 31 se notificó el hecho a J.A. Redondo, director del área técnica del CEAPAT.

1.3.3 Pruebas de funcionamiento

En enero, febrero y marzo de 2008 se realizaron diferentes pruebas de funcionamiento y, en particular, pruebas con los códigos universales IRC5 de Phillips y SIRC de Sony y en diferentes aparatos de TV. Los resultados de las pruebas fueron positivos por lo que se dió por validado al prototipo.

1.3.4 Pruebas del Mando a Distancia por Personas con Actuación Manual Limitada utilizando Pulsadores Diseñados a Medida de la discapacidad

Los primeros resultados positivos de las pruebas, a comienzos de 2008, fueron comunicados a los técnicos del CEAPAT. Dado que en el Area de Desarrollo Tecnológico del Centro Estatal de Autonomía Personal y Ayudas Técnicas del IMSERSO tienen amplia experiencia en el diseño y fabricación de este tipo de pulsadores, nos pusimos en contacto con este organismo para la coordinación de los ensayos más importantes y, en su caso, para realizar visitas de trabajo a dicho centro.

Con motivo de la celebración de las Jornadas de Puertas Abiertas del CEAPAT, el investigador principal, junto con los investigadores Dr. D. Isidro Calvo Gordillo y Dr. D. Oscar Barambones Caramazana (previamente adscritos como colaboradores al proyecto), realizamos

una visita a dicho centro del 28/05/2008 al 30/05/2008. Durante la visita, además de asistir a algunos de los actos celebrados, aprovechamos la oportunidad para entregar a los técnicos un informe (con los detalles de hardware y software del equipo) y un prototipo del mismo, y explicales someramente el funcionamiento, acordando una nueva cita en la que se ampliarían las explicaciones.

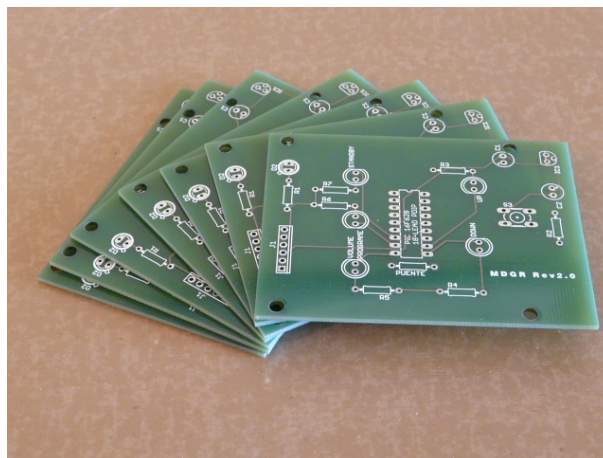
Los días 19 y 20 de junio tuve una nueva entrevista con los técnicos J.A. Redondo y Roberto Gaitán, en la que les hice entrega de un prototipo desmontado, con todas sus piezas, junto con uno de los dos equipos PIC-Kit-2 que habíamos adquirido, para que ellos pudieran montarlo, cargar el programa, y probarlo. También les indiqué la conveniencia de que el prototipo que ya les habíamos entregado montado (y/o el otro que ellos podrían montar) fuera probado por personas discapacitadas y utilizando los pulsadores desarrollados por el área de desarrollo del CEAPAT, para después, con los resultados de las pruebas, poder ajustar los tiempos encendido y apagado de los diodos LED adaptándolos a la medida de la discapacidad.

Por su parte, los citados técnicos manifestaron algunas sugerencias sobre algunas opciones que deseaban añadir al funcionamiento del equipo.

Hay que tener en cuenta que este tipo de pruebas puede resultar muy costoso en tiempo debido a la dificultad añadida de tener que realizarlas, al menos algunas de ellas, contando con la presencia de alguna persona con actuación manual limitada y que los resultados de las mismas no siempre tienen porqué ser aceptables. Es por ello que se dispuso de un amplio periodo para realizarlas.

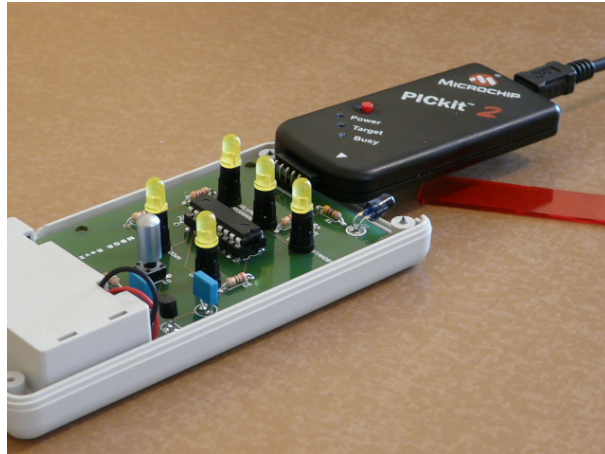
1.3.5 Finalización del proyecto

Durante los meses de septiembre y octubre se realizaron las pruebas finales y se procedió al montaje definitivo del prototipo, incorporando algunas mejoras respecto al prototipo original así como las opciones que nos habían sugerido los técnicos del CEAPAT. La fabricación de los circuitos impresos, una vez incorporados todos estos detalles, se encargó a la empresa Circuitos Impresos S.A. (CISA) de Madrid.



Placas de circuito impreso

Con estos circuitos impresos y con los componentes electrónicos detallados en el informe, esperamos que los técnicos del CEAPAT puedan montar una pequeña serie de unidades del mando.



Carga del programa en una de las placas

1.3.6 Documentación

La documentación se ha ido realizando durante el desarrollo del proyecto, y en esta fase, durante el mes de noviembre, se procedió a su redacción final incluyendo las explicaciones necesarias para su comprensión.

1.3.7 Entrega del proyecto

Durante el mes de diciembre de 2008 se procede a la entrega del proyecto al IMSERSO.

1.4 Resultados obtenidos

Tras la realización de varios prototipos del mando a distancia en las sucesivas fases del proyecto, tras diferentes pruebas de funcionamiento, el resultado del proyecto ha sido la construcción del mando a distancia definitivo, el producto a entregar al IMSERSO: un mando a distancia para TV adaptado al uso por Personas con Actuación Manual Limitada.



Producto a entregar al imserso

1.5 Conclusiones

- El presupuesto solicitado ha resultado adecuado para conseguir resultados esperados.
- Se ha cumplido el principal objetivo: diseño de un Mando a Distancia que pueda ser operado por una persona con actuación manual limitada a través de un pulsador diseñado a la medida de su discapacidad.
- Por medio de las placas de circuito impreso realizadas es posible realizar una primera pequeña serie de mandos. Y si el IMSERSO lo considerara oportuno se podrían realizar series mayores encargando a la empresa CISA la fabricación todas las copias de los circuitos impresos que se precisaran.
- Quedamos a disposición del IMSERSO y de los técnicos del CEAPAT cualquier cuestión relativa a la fabricación bien sea de la primera serie y de posibles posteriores series del mando.

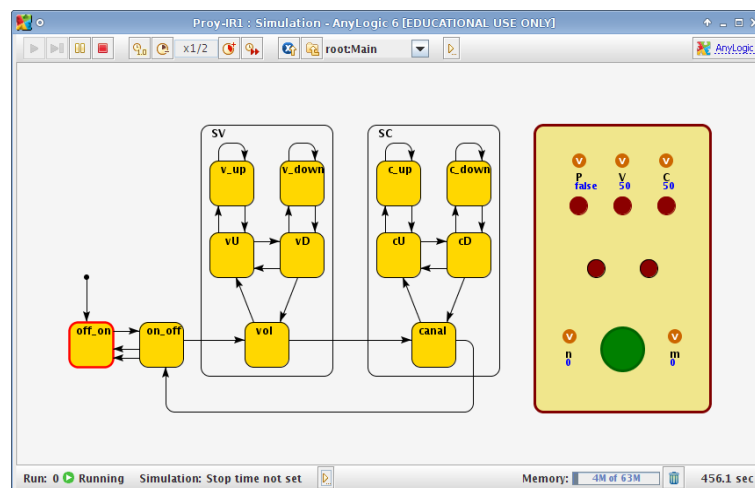
Apéndice A

Guía de utilización

1 Equipos

2 Descripción

El mando a distancia diseñado es un sistema secuencial, basado en el microcontrolador PIC-16F628A. En su diseño se ha utilizado el modelo de Cartas de Estado (*Statecharts*) que aparece en la figura. Este modelo nos va a servir para explicar su funcionamiento.

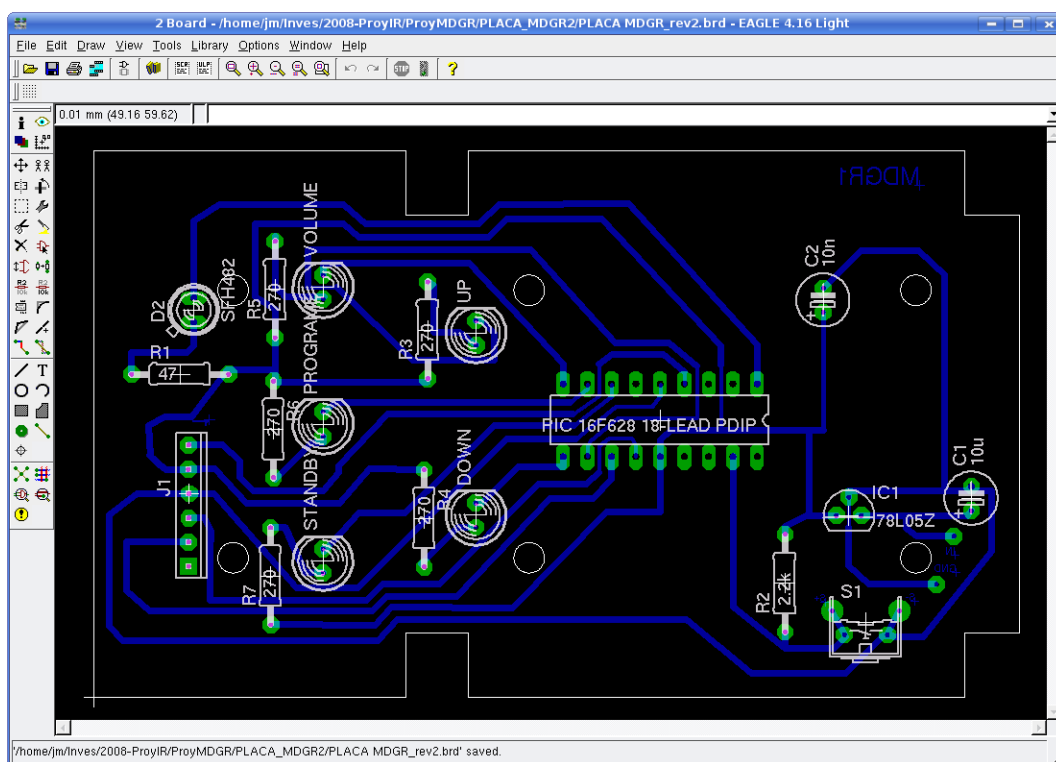


El sistema tiene cuatro estados: *off_on* (apagado), *on_off* (encendido), *SV* (volumen) y *SC* (canal). El estado inicial, en reposo, es *off_on*. Al pulsar el botón el sistema evoluciona en forma cíclica entre los tres estados *off_on*, *SV* y *SC*, pudiendo desde cada uno de ellos saltar a los estados indicados por las flechas (transiciones), en función del estado del botón: si el botón está pulsado el sistema salta al estado asociado a la correspondiente acción, y si no, al cabo de un tiempo, al estado siguiente. Si en los estados *SV* o *SC* se pulsa el botón, entonces el

sistema evoluciona a través de los estados en ellos contenidos, pudiendo, en cada caso, subir y bajar el volumen o incrementar y decrementar el número del canal sintonizado.

3 Construcción

El circuito del mando se ha diseñado con el programa Eagle, obteniendo el trazado del fotolito para la realización de una placa de circuito impreso a una sola cara, para la realización del primer prototipo, y a dos caras para la realización del circuito impreso final. La figura nos muestra la ventana de Eagle en la que se puede ver dicho trazado, incluidos los componentes.

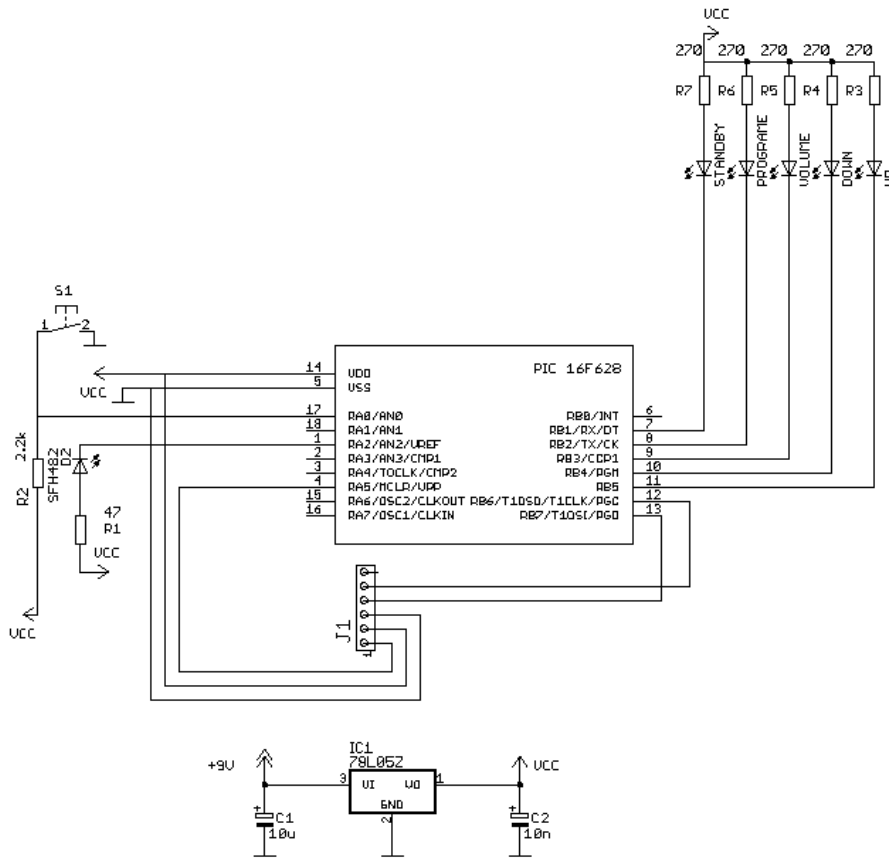


Los componentes utilizados son:

- 1 microcontrolador PIC 16F628A
- 1 diodo IR (emisor infrarrojo)
- 5 diodos LED
- 1 regulador de voltaje 78L05Z
- 1 resistencia de $47\ \Omega$ $\frac{1}{4}$ w

- 1 resistencia de $2.2\text{ k}\Omega$ $\frac{1}{4}\text{ w}$
- 5 resistencias de $270\ \Omega$ $\frac{1}{4}\text{ w}$
- 2 condensadores de 10 nF
- 1 conector AMP MTA-100 6 pin
- 1 conector para Jack de 3.5 mm

La figura



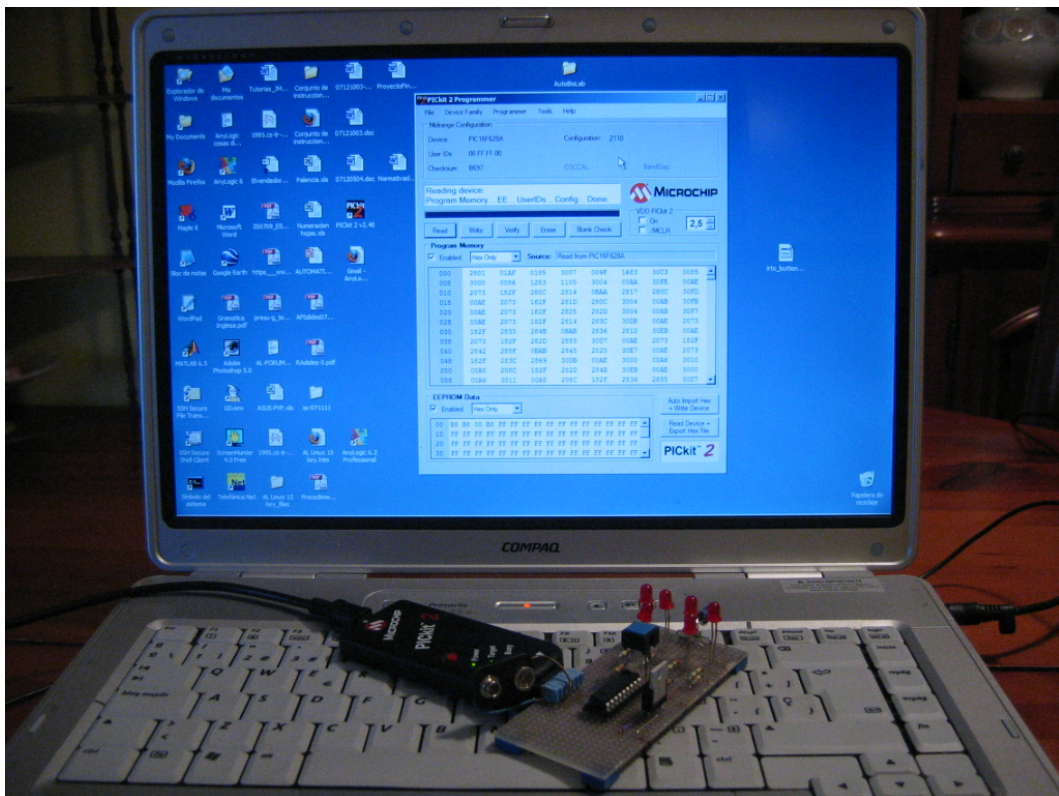
muestra el esquema del circuito. La siguiente figura,

4.1 Programa en ensamblador

Siguiendo el método anterior resulta sencillo entender el programa realizado en lenguaje ensamblador MASM de Microchip, que aparece en el apéndice B. Para facilitar la comprensión del programa, en el apéndice C se ha incluido un resumen de las instrucciones de la familia de microcontroladores PIC de microchip.

4.2 Carga del programa en la placa

La carga del programa se realiza fácilmente desde un ordenador tipo PC. Para ello se ejecuta el paquete MPLAB de Microchip. Este paquete contiene el programa MPASM, programa ensamblador, con el cual ensamblaremos nuestro programa y obtendremos como resultado un archivo con extensión .hex para cargar en la placa. Conectaremos el equipo PICKit-2 de Microchip a un puerto USB del ordenador y ejecutaremos su software. Seleccionaremos el PIC a utilizar (posiblemente lo autodetecte), y se carga el programa .hex. En la figura vemos una foto del PICKit-2 conectado a un ordenador portátil.



Apéndice B

Programa en ensamblador

```
; SOFTWARE ADAPTATION
;
; FILE:      ir_5L_1P.asm
; AUTHORS:   Alejandro Grandes, Miguel Angel Rayo,
;           Jose Maria Gonzalez de Durana
; COMPANY:   EUI - UPV/EHU Vitoria, Spain
; DEVICE:    PIC16F628A
; UPDATED:   13/01/2008
; DESCRIPT:  Software adaptation for PIC16F628A running on an
;           only one push-button IR remote comand
;
; ORIGINAL SOFTWARE:
;
; FILE:      ir_tx_RC5.asm
; AUTHOR:    Tom Perme
; COMPANY:   Microchip Technology, Inc.
; DEVICE:    10F206
; CREATED:   10/08/2006
;
; DESCRIP:   Application Note example file to illustrate RC5
;           protocol being transmitted over an infrared LED.
;
; Software License Agreement:
;
; The software supplied herewith by Microchip Technology Incorporated
; (the ‘‘Company’’) for its PICmicro Microcontroller is intended and
; supplied to you, the Company’s customer, for use solely and
; exclusively on Microchip PICmicro Microcontroller products. The
; software is owned by the Company and/or its supplier, and is
; protected under applicable copyright laws. All rights are reserved.
```

```

; Any use in violation of the foregoing restrictions may subject the
; user to criminal sanctions under applicable laws, as well as to
; civil liability for the breach of the terms and conditions of this
; license.
;
; THIS SOFTWARE IS PROVIDED IN AN ‘AS IS’ CONDITION. NO WARRANTIES,
; WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED
; TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
; PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. THE COMPANY SHALL NOT,
; IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL OR
; CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
;-----;
    errorlevel          -227
    list                p=16f628a
    radix               hex
#include <p16f628a.inc>
; Config Bits para P16f628 no son igual que para P16f628A
    --                ---                -----
; CP |  |  |  |  CPD LVP BOREN MCLRE FOSC2 PWRTE WDTE FOSC1 FOSCO
;-----+-----+-----+-----+-----+-----+-----+-----+
; 1 0 0 0 0 1 0 0 0 1 0 0 0 0
;-----+-----+-----+-----+-----+-----+-----+-----+
    __config           0x2110
; System Inputs
#define BBTN           PORTA, 0           ; One button which input
; System Outputs
#define OUTPUT_LED     PORTA, 2           ; Define the OUTPUT IR-LED PIN
#define LEDS           PORTB             ; State LEDS pines
; CONSTANTS
#define TV1            0x00              ; Address for device TV1
#define VOLUP_CMD      0x10              ; Command for volume up
#define VOLDOWN_CMD    0x11              ; and volume down.
#define CHUP_CMD       0x20              ; Command for channel up
#define CHDOWN_CMD     0x21              ; and channel down.
#define STNDBY_CMD     0x0C              ; Command for RC5 Data ‘Standby’
; Register Assignments
#define Delay_Count    0x20              ; Define registers for
#define Delay_Count2   0x21              ; delay routines (0x1, 0x11 for 10F20x)
#define DataByte       0x22              ; Define a byte to use for RC5 Data
#define AddrByte       0x23              ; Define a byte to use for RC5 address
#define ToggByte       0x24              ; Define a byte to determine toggle low or high
#define DataByteSave   0x25              ; Define a byte to save DataByte
#define AddrByteSave   0x26              ; Define a byte to save AddrByte

```

```

#define Delay_n      0x27      ; Delay register n
#define Delay_n1     0x28      ; Delay register n1
#define Delay_n2     0x29      ; Delay register n2
#define n            0x2A      ; Counter
#define m            0x2B      ; Counter
#define LEDsave      0x2E      ; LEDsave register to save LEDS
; FLAGS
#define FLAGS        0x2F      ; Define a byte to save some flags
#define BP           FLAGS, 0   ; Button pushed flag

; -----
; PROGRAM CODE
; -----

org      0                ; Processor Reset vector

goto     Main             ; redirect flow on powerup to Main label

Main

Init
    clrf    FLAGS
    clrf    PORTA          ; initialize output buffers
    movlw   b'00000111'
    movwf   CMCON          ; comparators off, just digital I/O
    bsf     STATUS,RP0     ; (RP1,RP0) = (0,1) ==> Bank 1 (TRISA & TRISB)
    movlw   b'11000011'
    movwf   TRISA          ; PORTA<0,1,6,7> inputs, PORTA<2,3,4,5> outputs
    movlw   0x00
    movwf   TRISB          ; all PORTB bits become outputs
    bcf     STATUS,RP0     ; (RP1,RP0) = (0,0) ==> Bank 0 again
    bcf     OUTPUT_LED     ; Init output off

MainLoop:

s0:
    movlw   0x04
    movwf   n              ; n=4
    movlw   b'11111111'    ; Load s0 LEDS parameter
    movwf   LEDS

aqui:

```

```

        btfsc    BTN
        goto    aqui
        goto    s1

s1:
        decfsz  n,1          ; n--
        goto    s11         ; if n>0
        goto    s0         ; if n==0

s11:
        movlw   b'11111101' ; Load s1 LEDs parameter
        movwf   LEDsave     ; Save LEDs
        call    time_button_leds
        btfsc   BP          ; P has been pushed => BP=0 => skip next
        goto    s2         ; BP=1 => has not been pushed (T)
        goto    stby1      ; BP=0 => has been pushed (P)

s2:
        movlw   0x04
        movwf   m           ; m=4
        movlw   b'11111011' ; Load s2 LEDs parameter
        movwf   LEDsave     ; Save LEDs
        call    time_button_leds
        btfsc   BP          ; P has been pushed => BP=0 => skip next
        goto    s3         ; BP=1 => has not been pushed (T)
        goto    vu         ; BP=0 => has been pushed (P)

s3:
        movlw   0x04
        movwf   m           ; m=4
        movlw   b'11110111' ; Load s3 LEDs parameter
        movwf   LEDsave     ; Save LEDs
        call    time_button_leds
        btfsc   BP          ; P has been pushed => BP=0 => skip next
        goto    s1         ; BP=1 => has not been pushed (T)
        goto    cu         ; BP=0 => has been pushed (P)

vu:
        movlw   b'11011011' ; Load vu LEDs parameter
        movwf   LEDsave     ; Save LEDs
        call    time_button_leds
        btfsc   BP          ; P has been pushed => BP=0 => skip next
        goto    vd1        ; BP=1 => has not been pushed (T)

```

```

        goto        v_up                ; BP=0 => has been pushed (P)

vd1:
    decfsz        m,1                ; Para que no lo haga que 3 veces
    goto         vd                  ; vu <--> vd
    goto         s2

vd:
    movlw        b'11101011'        ; Load vd LEDs parameter
    movwf        LEDsave            ; Save LEDs
    call         time_button_leds
    btfsc        BP                  ; P has been pushed => BP=0 => skip next
    goto         vu                  ; BP=1 => has not been pushed (T)
    goto         v_down              ; BP=0 => has been pushed (P)

cu:
    movlw        b'11010111'        ; Load cu LEDs parameter
    movwf        LEDsave            ; Save LEDs
    call         time_button_leds
    btfsc        BP                  ; P has been pushed => BP=0 => skip next
    goto         cd1                 ; BP=1 => has not been pushed (T)
    goto         c_up                ; BP=0 => has been pushed (P)

cd1:
    decfsz        m,1                ; Para que no lo haga mas que 3 veces
    goto         cd                  ; el baile cu <--> cd
    goto         s3

cd:
    movlw        b'11100111'        ; Load cd LEDs parameter
    movwf        LEDsave            ; Save LEDs
    call         time_button_leds
    btfsc        BP                  ; P has been pushed => BP=0 => skip next
    goto         cu                  ; BP=1 => has not been pushed (T)
    goto         c_down              ; BP=0 => has been pushed (P)

v_up:
    movlw        b'11011011'        ; Load vu LEDs parameter
    movwf        LEDsave            ; Save LEDs
    movlw        TV1                 ; Load device
    movwf        AddrByteSave        ; Save Device Address
    movlw        VOLUP_CMD           ; Load command
    movwf        DataByteSave        ; Save Data byte with command
    call         time_button_send
    btfsc        BP                  ; P has been pushed => BP=0 => skip next

```

```

        goto      vu          ; BP=1 => has not been pushed (T)
        goto      v_up       ; BP=0 => has been pushed (P)

v_down:
        movlw     b'11101011' ; Load vd LEDs parameter
        movwf     LEDsave     ; Save LEDs
        movlw     TV1        ; Load device
        movwf     AddrByteSave ; Save Device Address
        movlw     VOLDOWN_CMD ; Load command
        movwf     DataByteSave ; Save Data byte with command
        call      time_button_send
        btfsc     BP         ; P has been pushed => BP=0 => skip next
        goto      vd        ; BP=1 => has not been pushed (T)
        goto      v_down    ; BP=0 => has been pushed (P)

c_up:
        movlw     b'11010111' ; Load cu LEDs parameter
        movwf     LEDsave     ; Save LEDs
        movlw     TV1        ; Load device
        movwf     AddrByteSave ; Save Device Address
        movlw     CHUP_CMD    ; Load command
        movwf     DataByteSave ; Save Data byte with command
        call      time_button_send
        btfsc     BP         ; P has been pushed => BP=0 => skip next
        goto      cu        ; BP=1 => has not been pushed (T)
        goto      c_up      ; BP=0 => has been pushed (P)

c_down:
        movlw     b'11100111' ; Load cd LEDs parameter
        movwf     LEDsave     ; Save LEDs
        movlw     TV1        ; Load device
        movwf     AddrByteSave ; Save Device Address
        movlw     CHDOWN_CMD  ; Load command
        movwf     DataByteSave ; Save Data byte with command
        call      time_button_send
        btfsc     BP         ; P has been pushed => BP=0 => skip next
        goto      cd        ; BP=1 => has not been pushed (T)
        goto      c_down    ; BP=0 => has been pushed (P)

st1:
        movlw     b'11111101' ; Load vu LEDs parameter
        movwf     LEDsave     ; Save LEDs
        call      time_button_leds

```

```

        btfsc    BP                ; P has been pushed => BP=0 => skip next
        goto    stby1             ; BP=1 => has not been pushed (T)
        goto    stby_act          ; BP=0 => has been pushed (P)

stby1:
        decfsz  m,1               ; Para que no lo haga que 3 veces
        goto    st2               ; vu <--> vd
        goto    s1

st2:
        movlw   b'11111101'       ; Load vd LEDs parameter
        movwf   LEDsave           ; Save LEDs
        call    time_button_leds
        btfsc   BP                ; P has been pushed => BP=0 => skip next
        goto    st1               ; BP=1 => has not been pushed (T)
        goto    stby_act          ; BP=0 => has been pushed (P)

stby_act:
        movlw   b'11001101'       ; Load cd LEDs parameter
        movwf   LEDsave           ; Save LEDs
        movlw   TV1               ; Load device
        movwf   AddrByteSave      ; Save Device Address
        movlw   STNDBY_CMD        ; Load command
        movwf   DataByteSave      ; Save Data byte with command
        call    time_button_send
        btfsc   BP                ; P has been pushed => BP=0 => skip next
        goto    s1                ; BP=1 => has not been pushed (T)
        goto    s1                ; BP=0 => has been pushed (P)

; End MainLoop:

;*****
;*****
;
;   SUB-ROUTINES
;
;*****
;*****
;
; TIME_BUTTON_LEDS

```

```

;
; Description:
; If button is up (BTTN=1) during the given TIME then return BP = 1
; If button is pressed (BTTN=0) then the ACTION is taken and return BP = 0
;
; Input parameters:
; Before calling, preload LEDsave register with desired LED bits,
; for example:
; movlw    b'01011000'
; movwf    LEDsave      ; Save LEDs
;
; Output parameter:
; BP = 0  if button has been pushed
; BP = 1  if button has not been pushed
;
;-----
time_button_leds:
    movf    LEDsave,0      ; get LEDsave
    movwf   LEDS           ; to light leds
    movlw   0x06           ; aprox 1s
    movwf   Delay_n2      ; Load Delay register Delay_n2 with time
tbl_n2:
    movlw   0xff
    movwf   Delay_n1      ; Load Delay register Delay_n1 with 0xff
tbl_n1:
    movlw   0xff
    movwf   Delay_n       ; Load Delay register Delay_n with 0xff
tbl_n:
    btfsc   BTTN          ; Check if button is pushed down
    goto    skip_action1  ; button is up, skip action code
    call    DebounceDelay ; Short debounce delay
    btfsc   BTTN          ; check if button is up (false indicator)
    goto    skip_action1  ; bttN is up, skip action code
; Detected the button as pressed. Send keydown code.
; Button's Action Code
button_action1:
    movf    LEDsave,0      ; w = LEDsave
    movwf   LEDS           ; LEDS = w
    bcf     BP             ; Save BT = 0 (button pushed)
    goto    action_return1
skip_action1:
    decfsz  Delay_n,1      ; Delay_n = Delay_n-1
    goto    tbl_n         ; if Delay_n>0

```

```

        decfsz    Delay_n1,1      ; Delay_n1 = Delay_n1-1
        goto     tbl_n1          ; if Delay_n1>0
        decfsz    Delay_n2,1      ; Delay_n2 = Delay_n2-1
        goto     tbl_n2          ; if Delay_n2>0
        bsf      BP              ; Save BP = 1 (button not pushed)
action_return1:
        return

;-----
;
; TIME_BUTTON_SEND
;
; Description:
; If button is up (BTTN=1) during the given TIME then return BP = 1
; If button is pressed (BTTN=0) then the ACTION is taken and return BP = 0
;
; Input parameters:
; Before calling, preload registers DataByteSave and AddrByteSave with
; appropriate values to be sent, for example:
; movlw    TV1
; movwf    AddrByteSave          ; Save Device Address
; movlw    VOLDDOWN_CMD
; movwf    DataByteSave          ; Save Data byte with command
;
; Output parameter:
; BP = 0  if button has been pushed and action has been taken or
; BP = 1  if button has not been pushed
;
;-----
time_button_send:

        movf     LEDsave,0        ; get LEDsave
        movwf    LEDS             ; to light leds
        movlw    0x0D
        movwf    Delay_n2        ; Load Delay register Delay_n2 with time
tbs_n2:
        movlw    0xff
        movwf    Delay_n1        ; Load Delay register Delay_n1 with 0xff
tbs_n1:
        movlw    0xff
        movwf    Delay_n         ; Load Delay register Delay_n with 0xff
tbs_n:
        btfsc    BTTN            ; Check if button is pushed down

```

```

goto      skip_action      ; button is up, skip action code
call      DebounceDelay    ; Short debounce delay
btfsc    BBTN              ; check if button is up (false indicator)
goto      skip_action      ; bbtn is up, skip action code
; Detected the button as pressed.  Send keydown code.
; Button's Action Code
;
; Repeatedly send RC5 transmission with toggle bit = 0.

```

button_action:

```

; Load Device Address
movf      AddrByteSave,0    ; w = AddrByteSave
movwf    AddrByte          ; AddrByte = w
; Load Data byte with command
movf      DataByteSave,0    ; w = DataByteSave
movwf    DataByte          ; DataByte = w
;
movlw    0x00
movwf    ToggByte          ; Send Toggle=0 for button down.
call     SendRC5
btfss   BBTN              ; On button release, send one toggle+
goto     button_action     ; Keep sending RC5 code while bbtn down
; Send a final transmission with toggle bit = 1.
; Load Device Address
movf      AddrByteSave,0    ; w = AddrByteSave
movwf    AddrByte          ; AddrByte = w
; Load Data byte with command
;      movf      DataByteSave,0    ; w = DataByteSave
;      movwf    DataByte          ; DataByte = w
movlw    0x4b              ; w = DataByteSave (0x4b=75= "NOP en RC5")
movwf    DataByte          ; DataByte = w
;
movlw    0xFF
movwf    ToggByte          ; Send Toggle=1 for button released
call     SendRC5          ; Send button released.
bcf     BP                  ; Save BT = 0 (button pushed)
goto     action_return

```

skip_action:

```

decfsz   Delay_n,1        ; Delay_n = Delay_n-1
goto     tbs_n             ; if Delay_n>0

decfsz   Delay_n1,1       ; Delay_n1 = Delay_n1-1

```

```

        goto        tbs_n1            ; if Delay_n1>0

        decfsz     Delay_n2,1        ; Delay_n2 = Delay_n2-1
        goto        tbs_n2            ; if Delay_n2>0

        bsf        BP                ; Save BP = 1 (button not pushed)
action_return:
        return

;-----
; SendRC5
;
;     Before calling, preload registers DataByte and AddrByte with
;     appropriate values to be sent.
;
;     DataByte = 6 bits of data to send (upper 2 bits ignored)
;     AddrByte = 5 bits of addr to identify target (upper 3 bits ignored)
;-----
SendRC5:

; Pre-shift Addr Byte
    rlf           AddrByte, F        ; Must be rotated left 3 bits
    rlf           AddrByte, F        ;
    rlf           AddrByte, F        ; MSB now is MSB of 5 bit #
; Pre-shift Data Byte
    rlf           DataByte, F        ; Must be rotated left 2 bits
    rlf           DataByte, F        ; MSB now is MSB of 6 bit #

; SEND PREAMBLE
    call          SendOne            ; S1          Start 1
    call          SendOne            ; S2          Start 2
; SEND TOGGLE
    btfss        ToggByte, 0        ; if toggle is one, skip instr
    call          SendZero           ; Send a 0, toggle byte is zero
    btfsc        ToggByte, 0        ;
    call          SendOne            ; toggle byte is set, need to send a 1

; SEND DATA IN SPEED EFFICIENT MANNER

; SEND ADDRESS
; Begin shifting out address

```

```

; bit 4
  rlf      AddrByte, F      ; Shift out MSB.. C = MSB
  btfss   STATUS, C        ; if bit is 1, skip next instr.
  call    SendZero        ; bit is 0, send a zero
  btfsc   STATUS, C
  call    SendOne         ; bit is 1, send a one
; bit 3
  rlf      AddrByte, F      ; Shift out MSB.. C = MSB
  btfss   STATUS, C        ; if bit is 1, skip next instr.
  call    SendZero        ; bit is 0, send a zero
  btfsc   STATUS, C
  call    SendOne         ; bit is 1, send a one
; bit 2
  rlf      AddrByte, F      ; Shift out MSB.. C = MSB
  btfss   STATUS, C        ; if bit is 1, skip next instr.
  call    SendZero        ; bit is 0, send a zero
  btfsc   STATUS, C
  call    SendOne         ; bit is 1, send a one
; bit 1
  rlf      AddrByte, F      ; Shift out MSB.. C = MSB
  btfss   STATUS, C        ; if bit is 1, skip next instr.
  call    SendZero        ; bit is 0, send a zero
  btfsc   STATUS, C
  call    SendOne         ; bit is 1, send a one
; bit 0
  rlf      AddrByte, F      ; Shift out MSB.. C = MSB
  btfss   STATUS, C        ; if bit is 1, skip next instr.
  call    SendZero        ; bit is 0, send a zero
  btfsc   STATUS, C
  call    SendOne         ; bit is 1, send a one

; SEND DATA
; Shift Out DataByte
; bit 5
  rlf      DataByte, F     ; Shift out MSB.. C = MSB
  btfss   STATUS, C        ; if bit is 1, skip next instr.
  call    SendZero        ; bit is 0, send a zero
  btfsc   STATUS, C
  call    SendOne         ; bit is 1, send a one
; bit 4
  rlf      DataByte, F     ; Shift out MSB.. C = MSB
  btfss   STATUS, C        ; if bit is 1, skip next instr.
  call    SendZero        ; bit is 0, send a zero

```

```

    btfsc    STATUS, C
    call     SendOne          ; bit is 1, send a one
; bit 3
    rlf     DataByte, F      ; Shift out MSB.. C = MSB
    btfss   STATUS, C        ; if bit is 1, skip next instr.
    call    SendZero        ; bit is 0, send a zero
    btfsc   STATUS, C
    call    SendOne          ; bit is 1, send a one
; bit 2
    rlf     DataByte, F      ; Shift out MSB.. C = MSB
    btfss   STATUS, C        ; if bit is 1, skip next instr.
    call    SendZero        ; bit is 0, send a zero
    btfsc   STATUS, C
    call    SendOne          ; bit is 1, send a one
; bit 1
    rlf     DataByte, F      ; Shift out MSB.. C = MSB
    btfss   STATUS, C        ; if bit is 1, skip next instr.
    call    SendZero        ; bit is 0, send a zero
    btfsc   STATUS, C
    call    SendOne          ; bit is 1, send a one
; bit 0
    rlf     DataByte, F      ; Shift out MSB.. C = MSB
    btfss   STATUS, C        ; if bit is 1, skip next instr.
    call    SendZero        ; bit is 0, send a zero
    btfsc   STATUS, C
    call    SendOne          ; bit is 1, send a one

; 25 ms passed

; Delay remaining time so that repetitive calls to SendRC5
; occur at 114ms intervals as per RC5 spec.

; 114-25 = 89ms
    bcf     OUTPUT_LED      ; Set output low for off time

    movlw   d'89'
    movwf   Delay_Count2    ; outer loop delay counter

    call    delay_1ms        ; Delay 1ms ea. time through loop
    decfsz  Delay_Count2, F  ; Go through loop
    goto    $-2             ; if count not 0, keep looping

    return    ; Return from SendRC5 routine

```

```

;-----
; SendOne
;
;     A '1' in Manchester goes from Low-High over the bit period
;     Timing based off 4MHz internal clock with 1MHz Instruction
;     cycle. During the pulsing period, the carrier frequency should
;     be 36kHz, and the duty cycle of the carrier should be about 1/4.
;-----
SendOne:

; LOW HALF (889us = 889 instr cycles)
bcf     OUTPUT_LED           ; Turn off LED
; 1 --> 888us                ; (-1) instr cycle from total needed
;
movlw   0xFF                 ; (-1) Move 0xFF (255) into w
movwf   Delay_Count         ; (-1) Move w -> Delay_Count
decfsz  Delay_Count, F      ; (-1) Decrement F, skip if result = 0
goto    $-1                 ; (-2) Go back 1, keep decrementing until 0
; Loop Eq. = 3*N-1 cycles
; 3*254 = 764us completed in loop, + 3 cycles beforehand..
; 767us completed --> 122us to go.
;
movlw   d'39'                ; -1 (Load to finish time accurately)
movwf   Delay_Count         ; -1
decfsz  Delay_Count, F      ; -1
goto    $-1                 ; -2

; NOTE: there are two cycles following this

;     before pulsing will start.. so take 2 cycles off desired

; 1 + 1 + 3*N-1 = 122 - 2 --> N=39.66

; Choose N = 39, gives 116 cycles in loop, +2 setup, +2 lagging

; = 116+2+2 = 120 --> need 2 nops, or 1 goto $+1

goto    $+1                 ; -2
;
; HIGH HALF (889us)
; Toggle 7us on, 21us off for 35.7kHz (~36kHz) for the duration
; Pulsing 7 on, 21 off yields a 1/4 time duty cycle for carrier.

```

```

; Has 32 pulses of these periods 32*28us = 896us (~889us)
;
; These two clock cycles contribute to LOW TIME
movlw    d'32'      ; -1    (2 addit'l low cycles on low time)
movwf    Delay_Count2 ; -1    num pulses counter

```

CarrierLoopOne:

```

    bsf    OUTPUT_LED      ; -1    (BEGIN ON TIME)
    goto   $+1             ; -2us
    goto   $+1             ; -2us
    goto   $+1             ; -2us delayed 7us
    bcf    OUTPUT_LED      ; -1    (BEGIN OFF TIME)
    movlw  d'5'            ; -1 (Load to finish time accurately)
    movwf  Delay_Count     ; -1
    decfsz Delay_Count, F  ; -1
    goto   $-1             ; -2
; 1 + 1 + 1 + 3*N-1 = 21-3 --> x = 5.33
; Choose N=5, 1+1+1 + LOOP=14 =17 --> need 1 nop
    nop
    decfsz Delay_Count2, F ; -1 3us tacked on each pulse xcept last one
    goto   CarrierLoopOne ; -2 TAKE OFF OF ABOVE CALC

; DONE Sending a one
    return ; -2 return from subroutine

```

```

;-----
; SendZero
;
; A '0' in Manchester goes from High-Low over the bit period.
; The high period is a series of pulses of duty cycle 1/4 at a
; frequency of 36kHz. This implementation yields 35.71kHz.
;-----

```

SendZero:

```

; HIGH HALF (889us)
; Toggle 7us on, 21us off for 35.7kHz (~36kHz) for the duration
; Pulsing 7 on, 21 off yields a 1/4 time duty cycle for carrier.
; Has 32 pulses of these periods 32*28us = 896us (~889us)
;
; These two clock cycles contribute to LOW TIME
movlw    d'32'      ; -1    (2 addit'l low cycles on low time)
movwf    Delay_Count2 ; -1    num pulses counter

```

CarrierLoopZero:

```

    bsf      OUTPUT_LED      ; -1    (BEGIN ON TIME)
    goto     $+1             ; -2us
    goto     $+1             ; -2us
    goto     $+1             ; -2us  delayed 7us
    bcf      OUTPUT_LED      ; -1    (BEGIN OFF TIME)
    movlw   d'5'            ; -1 (Load to finish time accurately)
    movwf   Delay_Count     ; -1
    decfsz  Delay_Count, F   ; -1
    goto     $-1             ; -2
; 1 + 1 + 1 + 3*N-1 = 21-3 --> x = 5.33
; Choose N=5, 1+1+1 + LOOP=14 =17 --> need 1 nop
    nop
    decfsz  Delay_Count2, F   ; -1  3us tacked on each pulse xcept last one
    goto     CarrierLoopZero ; -2  TAKE OFF OF ABOVE CALC

; Last pulse needs its off time
; that it misses from goto CarrierLoop
    goto     $+1             ; -2

; LOW HALF (889us = 889 instr cycles)
    bcf      OUTPUT_LED      ; Turn off LED
; 1 --> 888us          ; (-1) instr cycle from total needed
;
    movlw   0xFF             ; (-1) Move 0xFF (255) into w
    movwf   Delay_Count     ; (-1) Move w -> Delay_Count
    decfsz  Delay_Count, F   ; (-1) Decrement F, skip if result = 0
    goto     $-1             ; (-2) Go back 1, keep decrementing until 0
; Loop Eq. = 3*N-1 cycles
; 3*254 = 764us completed in loop, + 3 cycles beforehand..
; 767us completed --> 122us to go.
;
    movlw   d'39'           ; -1 (Load to finish time accurately)
    movwf   Delay_Count     ; -1
    decfsz  Delay_Count, F   ; -1
    goto     $-1             ; -2
; NOTE: there are two cycles following this (return)
;       before next bit may be sent.
; 1 + 1 + 3*N-1 = 122 - 2 --> N=39.66
; Choose N = 39, gives 116 cycles in loop, +2 setup, +2 lagging
; = 116+2+2 = 120 --> return finishes the last 2 instr cycl.

```

```
return      ; -2
```

```
-----  
; delay_1ms  
;  
;     Precise delay.  Actually delays 999us, but this allows for  
;     any loops calling it to incur 1us of overhead.  
-----  
delay_1ms:  
    movlw    0xFF                ; +1 us  
    movwf    Delay_Count         ; +1  
    decfsz   Delay_Count, F      ; Decrement F, skip if result = 0  
    goto     $-1                 ; Go back 1, keep decrementing until 0  
; Loop = 3*N-1 us = 3*255-1  
; +764  
; Need to delay 236us more.  2 of which will be used in return  
    movlw    d'76'              ; +1  
    movwf    Delay_Count         ; +1 234 remain, but only acct for 232  
    decfsz   Delay_Count, F      ; 3*N-1 = 232 --> N = 77.66  
    goto     $-1                ; Choose N=77  
  
    return   ; +2 Return program flow
```

```
-----  
; DebounceDelay  
;  
;     Small and simple delay to provide time for bouncing from  
;     a button to settle.  
-----  
DebounceDelay:  
    movlw    d'3'                ; Move 0xFF into w (count, N)  
    movwf    Delay_Count         ; Move w -> Delay_Count  
    decfsz   Delay_Count, F      ; Decrement F, skip if result = 0  
    goto     $-1                 ; Go back 1, keep decrementing until 0  
    ; Loop delay = 3*N-1  
    return   ; Return program flow  
    ; TOTAL DELAY ~12us
```

```
;,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,  
end
```


Apéndice C

Conjunto de instrucciones del PIC 16F628

Todos los PICs de gama media de Microchip comparten el mismo juego de 35 instrucciones.

Notación utilizada en las instrucciones:

f: un registro determinado de la fila de registros

W: registro de trabajo (acumulador)

b: posición de un bit en el registro "f"

d: destino, puede ser "f" (d=1) o "W" (d=0)

etiqueta: grupo de caracteres que marcan el inicio de una parte del programa

[]: opcional

: posición de un bit dentro de un registro

Instrucciones orientadas a registros

ADDWF : Suma W y f

Suma el contenido del registro "W" y el registro "f". Si "d" es 0, el resultado se almacena en el registro W. Si "d" es 1 el resultado se almacena en el registro "f". Sintaxis: [etiqueta] ADDWF f,d Operación: (W) + (f) \Rightarrow (destino) Flags afectados: C, DC, Z
Ejemplo: ADDWF REG,1 Antes de la instrucción: W=0x03, REG=0x04 Después de la instrucción: W=0x03, REG=0x07

ANDWF : W AND f

Realiza la operación lógica AND entre el registro W y el registro "f". Si "d" es 0 el resultado se almacena en el registro W. Si "d" es 1, el resultado se almacena en el registro "f". Sintaxis: [etiqueta] ANDWF f,d
Operación: (W) AND (f) \Rightarrow (destino)
Flags afectados: Z
Ejemplo: ANDWF REG,1

Antes de la instrucción: W=0x17, REG= 0xC2
Después de la instrucción: W=0x17, REG= 0x02

CLRF : Borra un registro

Borra el contenido del registro “f” y se activa el flag Z.

Sintaxis: [etiqueta] CLRF f

Flags afectados: Z

Ejemplo: CLRF REG

Antes de la instrucción: REG=0x54

Después de la instrucción: REG=0x00, Z=1

CLRW : Borra el registro de trabajo W

Borra el contenido del registro “W” y se activa el flag Z. Esta instrucción no tiene operandos.

Sintaxis: [etiqueta] CLRW

Flags afectados: Z

Ejemplo: CLRW

Antes de la instrucción: W=0x54

Después de la instrucción: W=0x00, Z=1

COMF : Complementa el registro f

El contenido del registro “f” se complementa. Si d=0 el resultado se almacena en el registro W. Si d=1 el resultado se almacena en el registro “f”.

Sintaxis: [etiqueta] COMF f,d

Flags afectados: Z

Ejemplo: COMF REG,0

Antes de la instrucción: REG=0x13

Después de la instrucción: REG=0x13, W=0xEC

DECF : Decrementa f en 1

De decrementa en uno el contenido del registro “f”. Si d=0, el resultado se almacena en W. Si d=1, el resultado se almacena en “f”.

Sintaxis: [etiqueta] DECF f,d

Flags afectados: Z

Ejemplo: DEC CONT, 1

Antes de la instrucción: CONT=0x01, Z=0

Después de la instrucción: CONT=0x00, Z=1

DECFSZ : Decrementa en 1 y salta si el resultado es 0

El contenido del registro “f” se decrementa. Si “d”=0, el resultado se coloca en el registro W. Si d=1, el resultado se coloca en el registro “f”. Si el resultado es 0, se salta la siguiente instrucción y se continúa con la ejecución.

Sintaxis: [etiqueta] DECFSZ f,d

Flags afectados: Ninguno

INCF : Incrementa el registro f

Incrementa en uno el contenido del registro “f”. Si d=0, el resultado se almacena en W. Si d=1, el resultado se almacena en “f”.

Sintaxis: [label] INCF f,d

Flags afectados: Z

Ejemplo: INCF CONT,1

Antes de la instrucción: CONT=0xFF, Z=0

Después de la instrucción: CONT=0x00, Z=1

INCFSZ : Incrementa en 1 y salta si el resultado es 0

El contenido del registro “f” se incrementa. Si d=0, el resultado se coloca en el registro W. Si d=1, el resultado se coloca en el registro “f”. Si el resultado es 0, se salta la siguiente instrucción y se continúa con la ejecución.

Sintaxis: [etiqueta] DECFSZ f,d

Flags afectados: Ninguno

IORWF : W OR f

Realiza la operación lógica OR entre el registro W y el registro “f”. Si d=0 el resultado se almacena en el registro W. Si d=1, el resultado se almacena en el registro “f”.

Sintaxis: [etiqueta] IORWF f,d

Flags afectados: Z

Ejemplo: IORWF REG,0

Antes de la instrucción: REG=0x13, W=0x91

Después de la instrucción: REG=0x13, W=0x93

MOVF : Mover el registro f

El contenido del registro “f” se mueve al destino “d”. Si d=0, el destino es el registro W. Si d=1, el destino es el propio registro “f”.

Sintaxis: [etiqueta] MOVF f,d

Flags afectados: Z

Ejemplo: MOVF REG,0

Después de la instrucción: W=REG

MOVWF : Mover el registro w a f

RLF : Rota el registro f a la izquierda

El contenido del registro “f” se rota una posición a la izquierda. El bit de más peso pasa al carry y el carry se introduce por el bit de menos peso de “f”. Si d=0, el resultado se coloca en el registro W. Si d=1, el resultado queda en el registro “f”.

Sintaxis: [etiqueta] RLF f,d

Flags afectados: C

Ejemplo: RLF REG,1

Antes de la instrucción: REG=b“11100110”, C=0

Después de la instrucción: REG=b“11001100”, C=1

RRF : Rota el registro f a la derecha

El contenido del registro “f” se rota una posición a la derecha. El bit de menos peso pasa al carry y el carry se introduce por el bit de más peso de “f”. Si d=0, el resultado se coloca en el registro W. Si d=1, el resultado queda en el registro “f”.

Sintaxis: [etiqueta] RLF f,d

Flags afectados: C

Ejemplo: RLF REG,1

Antes de la instrucción: REG=b“11100110”, C=0

Después de la instrucción: REG=b“01110011”, C=0

SUBWF : Resta f - W

Resta el contenido del registro “f” menos el contenido del registro W. Si d=0, el resultado se almacena en el registro W. Si d=1, el resultado se almacena en el registro “f”.

Sintaxis: [etiqueta] SUBWF f,d

Flags afectados: C, DC, Z

Ejemplo: SUBWF REG,1

Antes de la instrucción: REG=0x01, W=0x02

Después de la instrucción: REG=0xFF, W=0x02

SWAPF : Intercambio de f

El nibble bajo del registro “f” se intercambia con el nibble alto del mismo. Si d=0, el resultado se coloca en el registro W. Si d=1, el resultado queda en el registro “f”.

Sintaxis: [etiqueta] SWAPF f,d

Flags afectados: Ninguno

Ejemplo: SWAPF REG,1

Antes de la instrucción: REG=0x54

Después de la instrucción: REG=0x45

XORWF : W XOR f

Realiza la función lógica OR exclusiva entre el contenido del registro W y el registro “f”. Si d=0, el resultado se almacena en el registro W. Si d=1 el resultado se almacena en el registro “f”.

Sintaxis: [etiqueta] XORWF f,d

Flags afectados: Z

Ejemplo: XORWF REG,1

Antes de la instrucción: REG=0xAF, W=0xB5

Después de la instrucción: REG=0x1A, W=0xB5

NOP : No operacion

No realiza ninguna operacion, solo consume un ciclo de reloj

Sintaxis: [etiqueta] NOP

Instrucciones orientadas a bits

BCF : Borra un bit

Borra el bit “b” del registro “f”

Sintaxis: [etiqueta] BCF f,b

Ejemplo: BCF REG,0

Antes de la instrucción: REG=b“01101101”

Después de la instrucción: REG=b“01101100”?

BSF : Activa un bit

Activa el bit “b” del registro “f”

Sintaxis: [etiqueta] BSF f,b

Ejemplo: BSF REG,2

Antes de la instrucción: REG=b“01001001”

Después de la instrucción: REG=b“01001011”

BTFSC : Chequea un bit y salta si es 0

Si el bit “b” del registro “f” es 0, se salta una instrucción y se continúa con la ejecución.

Sintaxis: [etiqueta] BTFSC f,b

BTFSS : Chequea un bit y salta si es 1

Si el bit “b” del registro “f” es 1, se salta una instrucción y se continúa con la ejecución.

Sintaxis: [etiqueta] BTFSS f,b

Instrucciones orientadas a constantes y de control

ANDLW : W AND literal

Realiza la operación lógica AND entre el registro W y la constante “k”. El resultado se almacena en el registro W.

Sintaxis: [label] ANDWL k

Flags afectados: Z

Ejemplo: ANDLW 0x5F

Antes de la instrucción: W=0xA3

Después de la instrucción: W=0x03

CALL : Llamada a subrutina

Llamada y salto a subrutina. La dirección de retorno se guarda en el stack. La constante “k” de 8 bits forma la dirección de salto y se carga en los bits del PC. Los bits del PC se cargan con los bits del registro “STATUS”. PC se pone a 0.

Sintaxis: [etiqueta] CALL k

Ejemplo: INICIO CALL DESTINO

Antes de la instrucción: PC=INICIO

Después de la instrucción: PC=DESTINO

CLRWDT : Borra el watchdog timer

Esta instrucción borra tanto el “watchdog” como el prescaler. Los bits TO y PD del registro de estado se ponen a “1”.

Sintaxis: [label] CLRWDT

Flags afectados: TO, PD

GOTO : Salto incondicional

Se trata de un salto incondicional. Los 9 bits de la constante “k” que forman la instrucción, se cargan en los bits del PC y forman la dirección de salto. Los bits del PC se cargan con los bits del registro de estado.

Sintaxis: [etiqueta] GOTO k

Ejemplo: INICIO GOTO DESTINO

Antes de la instrucción: PC=0

Después de la instrucción: PC=DESTINO

IORLW : W OR literal

Se realiza la función lógica OR entre el registro W y la constante “k”.

El resultado se almacena en el registro W.

Sintaxis: [etiqueta] IORLW k

Flags afectados: Z

Ejemplo: IORLW 0x35

Antes de la instrucción: W=0x9A
Después de la instrucción: W=0xBF

MOVLW : Carga un literal en W

El registro W se carga con el valor de 8 bits expresado mediante el literal “k”.

Sintaxis: [etiqueta] MOVLW k
Ejemplo: MOVLW 0x5A
Después de la instrucción: W=0x5A

RETLW : Regresa de una subrutina y carga el valor K en W

El programa regresa de la subrutina y carga el valor de 8 bits del registro k en el registro W

Sintaxis: [etiqueta] RETLW,k
Ejemplo: RETLW,0x45
Antes de la instrucción: W=xx
Después de la instrucción: W=0x45

RETFIE : Regresa de la rutina de servicio

Sintaxis: [etiqueta] RETFIE

RETURN : Regresa de una subrutina

El programa regresa de la subrutina y ejecuta la instrucción que sigue a CALL

Sintaxis: [etiqueta] RETURN

SLEEP : Entra en estado de reposo

Al salir, activa el bit de estado TO y borra el PD. El WDT y el prescaler se borran. Al entrar en el modo SLEEP, se detiene el oscilador.

Sintaxis: [etiqueta] SLEEP
Flags afectados: TO, PD, GPWUF

SUBLW : Resta L - W

A una constante “k” de 8 bits se le resta el registro W. El resultado es guardado en el mismo registro W.

Sintaxis: [etiqueta] SUBLW k
Flags afectados: C,DC,Z

XORLW : W XOR literal

Realiza la función lógica OR exclusiva entre el contenido del registro W y la constante “k” de 8 bits. El resultado se almacena en el registro W.

Sintaxis: [etiqueta] XORLW k

Flags afectados: Z

Ejemplo: XORLW 0xAF

Antes de la instrucción: W = 0xB5

Después de la instrucción: W = 0x1A

2. Memoria justificativa

a) Investigador Principal

José María González de Durana y García

b) Denominación del proyecto

Estudio y diseño de dispositivos de ayuda en la acción a distancia para Personas Mayores y Personas Discapacitadas.

c) Periodo de ejecución

El proyecto ha sido realizado entre julio de 2007 y diciembre de 2008 y su desarrollado ha seguido las fases que se indican a continuación.

Actualización de información. Durante el mes de julio de 2007 se procedió en primer lugar a la selección de dos alumnos “becarios” para, bajo la supervisión del investigador principal, formar el equipo humano encargado de la realización del proyecto. Estas personas son:

D. Alejandro Grandes López

D. Miguel Angel Rayo Cerrato

Pruebas de funcionamiento. En enero, febrero y marzo de 2008 se realizaron diferentes pruebas de funcionamiento. Los resultados de las pruebas fueron positivos por lo que se dió por validado al prototipo.

Pruebas del Mando a Distancia por Personas con Actuación Manual Limitada utilizando Pulsadores Diseñados a Medida de la discapacidad.

Los primeros resultados positivos de las pruebas, a comienzos de 2008, fueron comunicados a los técnicos del CEAPAT. Dado que en el Area de Desarrollo Tecnológico del

Centro Estatal de Autonomía Personal y Ayudas Técnicas del IMSERSO tienen amplia experiencia en el diseño y fabricación de este tipo de pulsadores, nos pusimos en contacto con este organismo para la coordinación de los ensayos más importantes y, en su caso, para realizar visitas de trabajo a dicho centro.

Con motivo de la celebración de las Jornadas de Puertas Abiertas del CEAPAT, el investigador principal, junto con los investigadores Dr. D. Isidro Calvo Gordillo y Dr. D. Oscar Barambones Caramazana (previamente adscritos como colaboradores al proyecto), realizamos una visita a dicho centro del 28/05/2008 al 30/05/2008. Durante la visita, además de asistir a algunos de los actos celebrados, aprovechamos la oportunidad para entregar a los técnicos un informe (con los detalles de hardware y software del equipo) y un prototipo del mismo, y explicarles someramente el funcionamiento, acordando una nueva cita en la que se ampliarían las explicaciones.

Los días 19 y 20 de junio tuve una nueva entrevista con los técnicos J.A. Redondo y Roberto Gaitán, en la que les hice entrega de un prototipo desmontado, con todas sus piezas, junto con uno de los dos equipos PIC-Kit-2 que habíamos adquirido, para que ellos pudieran montarlo, cargar el programa, y probarlo. También les indiqué la conveniencia de que el prototipo que ya les habíamos entregado montado (y/o el otro que ellos podrían montar) fuera probado por personas discapacitadas y utilizando los pulsadores desarrollados por el área de desarrollo del CEAPAT, para después, con los resultados de las pruebas, poder ajustar los tiempos encendido y apagado de los diodos LED adaptándolos a la medida de la discapacidad.

Por su parte, los citados técnicos manifestaron algunas sugerencias sobre algunas opciones que deseaban añadir al funcionamiento del equipo.

Hay que tener en cuenta que este tipo de pruebas puede resultar muy costoso en tiempo debido a la dificultad añadida de tener que realizarlas, al menos algunas de ellas, contando con la presencia de alguna persona con actuación manual limitada y que los resultados de las mismas no siempre tienen por qué ser aceptables. Es por ello que se dispuso de un amplio periodo para realizarlas.

Finalización del proyecto. Durante los meses de septiembre y octubre se realizaron las pruebas finales y se procedió al montaje definitivo del prototipo, incorporando algunas mejoras respecto al prototipo original así como las opciones que nos habían sugerido los técnicos del CEAPAT. La fabricación de los circuitos impresos, una vez incorporados todos estos detalles, se encargó a la empresa Circuitos Impresos S.A. (CISA) de Madrid. Con estos circuitos impresos y con los componentes electrónicos detallados en el informe, esperamos que los técnicos del CEAPAT puedan montar una pequeña serie (del orden de 10) de unidades del mando.

Documentación. La documentación se ha ido realizando durante el desarrollo del proyecto, y en esta fase, durante el mes de noviembre, se procedió a su redacción final incluyendo

las explicaciones necesarias para su comprensión.

Entrega del proyecto. Durante el mes de diciembre de 2008 se procede a la entrega del proyecto al IMSERSO.

d) Resumen económico

El origen de la financiación ha sido únicamente la subvención al proyecto denominado *Estudio y diseño de dispositivos de ayuda en la acción a distancia para Personas Mayores y Personas Discapacitadas* cuyo importe, por Resolución de 21 de noviembre de 2007, de la Dirección General del Imsero de concesión de subvenciones para la realización de proyectos de investigación científica, desarrollo e innovación tecnológica, convocatoria de 11 de mayo de 2007 (BOE de 22 de mayo), fué de 11.000€. En la tabla siguiente se indica el importe subvencionado y el estado de la liquidación desglosado por conceptos de gasto.

Concepto 2320: material fungible

Fecha	Acreedor	Nºfactura	Descripción	Debe	Haber
					2000,00
11.12.07	RS Amidata	69094332	10 PIC 16F690A-I/P	34,57	
15.12.07	IP Electrónica	C00874	Material electrónico	30,25	
25.01.08	IP Electrónica	C01138	6 PIC 16F690	22,27	
31.01.08	SEL Logroño	00800222	Material electrónico	183,53	
09.02.08	IP Electrónica	C01291	Material electrónico	49,43	
23.02.08	IP Electrónica	C01430	Material electrónico	79,53	
29.02.08	IP Electrónica	C01515	Material electrónico	64,31	
07.03.08	RS Amidata	69130840	10 cajas ABS 140x65	59,25	
18.03.08	XLAN	80596	4 Memorias USB - 2GB	92,80	
14.04.08	IP Electrónica	C01806	Material electrónico	33,57	
06.05.08	IP Electrónica	C01972	Material electrónico	54,64	
15.05.08	IP Electrónica	C02026	Material electrónico	34,11	
30.05.08	IP Electrónica	C02131	Material electrónico	121,73	
08.07.08	CISA Madrid	085248	12 C.I. (realización)	505,78	
15.07.08	IP Electrónica	C02483	Material electrónico	43,15	
19.09.08	RS Amidata	69207582	2 PICkit 2 Flash	101,27	
Gasto total en material fungible				1510,19	

Concepto 6420: Inmovilizado

Fecha	Acreedor	Nºfactura	Descripción	Debe	Haber
					3640,00
23.11.07	RS Amidata	69086756	2 PicoScope	1502,57	
23.11.07	RS Amidata	69086757	2 PICkit-2	101,27	
14.03.08	XLAN	80563	2 Portátil ASUS F3E	1879,20	
Gasto total en material inventariable				3483,04	

Concepto 2600: Viajes y dietas

Fecha	Acreedor	Nºfactura	Descripción	Debe	Haber
					2000,00
30.05.08	Viajes Iberia	50813/F550	3 Viaje Madrid	775,44	
30.05.08	Viajes Iberia	50956/F550	Reemb fra 50813/F550	-347,50	
30.05.08	Viajes Iberia	50957/F550	3 billetes tren M-VI	92,62	
02.06.08	O Barambones		Dietas Viaje Madrid	84,00	
02.06.08	Isidro Calvo		Dietas Viaje Madrid	101,00	
02.06.08	JMG Durana		Dietas Viaje Madrid	128,00	
16.06.08	Viajes Iberia	51158/F550	Viaje Madrid y hotel	260,43	
22.06.08	JMG Durana		Dietas Viaje Madrid	105,00	
Gasto total en viajes y dietas				1198,99	

Concepto 2500: Subcontratación

Fecha	Acreedor	Nºfactura	Descripción	Debe	Haber
					1360,00
Gasto total en subcontratación				0000,00	

Concepto 2490: Otros gastos

Fecha	Acreedor	Nºfactura	Descripción	Debe	Haber
			pasados a fungible		2000,00
Gasto total en otros gastos				0000,00	

e) Metodología

La metodología empleada ha sido la que se emplea habitualmente en el ámbito de las aplicaciones con microprocesadores: desarrollo del hardware y software utilizando equipos especiales suministrados por el fabricante del microprocesador, en nuestro caso el equipo de desarrollo PICkit-2 Microchip, conectado al puerto USB de un ordenador portátil.

También se han utilizado, para la medida de algunas señales eléctricas, los 2 Osciloscopios digitales PicoScope 3224 PC, conectados al puerto USB.

f) Objetivos previstos

La Ley 39/2006, de 14 de diciembre, en su artículo 3.1 define la autonomía como "la capacidad de controlar, afrontar y tomar, por propia iniciativa, decisiones personales acerca de cómo vivir de acuerdo con las normas y preferencias propias así como de desarrollar las actividades básicas de la vida diaria" en el 3.3 las Actividades Básicas de la Vida Diaria como "las tareas más elementales de la persona, que le permiten desenvolverse con un mínimo de autonomía e independencia, tales como: el cuidado personal, las actividades domésticas básicas, la movilidad esencial, reconocer personas y objetos, orientarse, entender y ejecutar órdenes o tareas sencillas". También, en su disposición adicional tercera (ayudas económicas para facilitar la autonomía personal) habla de las Ayudas económicas para facilitar la autonomía personal: "la Administración General del Estado y las administraciones de las Comunidades Autónomas podrán, de conformidad con sus disponibilidades presupuestarias, establecer acuerdos específicos para la concesión de ayudas económicas con el fin de facilitar la autonomía personal". Dice también que las ayudas tendrán la condición de subvención e irán destinadas: (a) a apoyar a la persona con ayudas técnicas o instrumentos necesarios para el normal desenvolvimiento de su vida ordinaria, (b) A facilitar la accesibilidad y adaptaciones en el hogar que contribuyan a mejorar su capacidad de desplazamiento en la vivienda.

El presente proyecto se ha centrado especialmente en el estudio y desarrollo de algunos métodos y dispositivos o aparatos (aludidos en el apartado (a) de dicha disposición tercera), que sirvan "para el normal desenvolvimiento de su vida ordinaria". Y en particular nos hemos centrado en los dispositivos de acción a distancia, más comúnmente llamados *mandos a distancia* (MD). Si todos hemos comprobado cómo los MD sirven para aumentar el confort a las personas sin discapacidad, parece que su utilidad podría de ser mayor para personas que sufren alguna discapacidad porque, quizás tras oportunos rediseños o adaptaciones, podrían facilitarles la realización algunas de las actividades básicas de la vida diaria".

El objetivo principal del proyecto ha sido el diseño de un MD que pueda ser operado por una persona con actuación manual limitada a través de un pulsador diseñado a la medida de su discapacidad, aprovechando en lo posible los dispositivos creados por el Area de Desarrollo Tecnológico.

g) Resultados obtenidos

Tras la realización de varios prototipos del mando a distancia en las sucesivas fases del proyecto, tras diferentes pruebas de funcionamiento, el resultado del proyecto ha sido la construcción del mando a distancia definitivo, el producto a entregar al IMSERSO: un mando a distancia

para TV adaptado al uso por Personas con Actuación Manual Limitada.



Producto a entregar al imsero

h) Conclusiones

- El presupuesto solicitado ha resultado adecuado para conseguir resultados esperados.
- Se ha cumplido el principal objetivo: diseño de un Mando a Distancia que pueda ser operado por una persona con actuación manual limitada a través de un pulsador diseñado a la medida de su discapacidad.
- Por medio de las placas de circuito impreso realizadas es posible realizar una primera pequeña serie de mandos. Y si el IMSERSO lo considerara oportuno se podrían realizar series mayores encargando a la empresa CISA la fabricación todas las copias de los circuitos impresos que se precisaran.
- Quedamos a disposición del IMSERSO y de los técnicos del CEAPAT cualquier cuestión relativa a la fabricación bien sea de la primera serie y de posibles posteriores series del mando.

3. Justificantes de gastos

<p>Proyecto 27/2007 Mando a distancia E.U.I. Vitoria-Gasteiz Universidad del País Vasco</p>	<p>Proyecto 27/2007 Mando a distancia E.U.I. Vitoria-Gasteiz Universidad del País Vasco</p>	<p>Proyecto 27/2007 Mando a distancia E.U.I. Vitoria-Gasteiz Universidad del País Vasco</p>
<p>Proyecto 27/2007 Mando a distancia E.U.I. Vitoria-Gasteiz Universidad del País Vasco</p>	<p>Proyecto 27/2007 Mando a distancia E.U.I. Vitoria-Gasteiz Universidad del País Vasco</p>	<p>Proyecto 27/2007 Mando a distancia E.U.I. Vitoria-Gasteiz Universidad del País Vasco</p>
<p>Proyecto 27/2007 Mando a distancia E.U.I. Vitoria-Gasteiz Universidad del País Vasco</p>	<p>Proyecto 27/2007 Mando a distancia E.U.I. Vitoria-Gasteiz Universidad del País Vasco</p>	<p>Proyecto 27/2007 Mando a distancia E.U.I. Vitoria-Gasteiz Universidad del País Vasco</p>
<p>Proyecto 27/2007 Mando a distancia E.U.I. Vitoria-Gasteiz Universidad del País Vasco</p>	<p>Proyecto 27/2007 Mando a distancia E.U.I. Vitoria-Gasteiz Universidad del País Vasco</p>	<p>Proyecto 27/2007 Mando a distancia E.U.I. Vitoria-Gasteiz Universidad del País Vasco</p>