# Smart Medicine Box

**By,**

**V. Shambavi**

**V. Shanthini**

**Introduction**

Our ECE 4760 final project is to build a microcontroller based smart medicine box. Our

medicine box is targeted on users who regularly take drugs or vitamin supplements or nurses

who take care of the older or patients. The medicine box is programmable that allows nurses

or users to specify the pill quantity to take and the serve time for each day. The smart medicine

box contains seven separate sub-boxes. Therefore, nurses or users can set information for

seven different pills. When the pill quantity and serve time has been set, the medicine box will

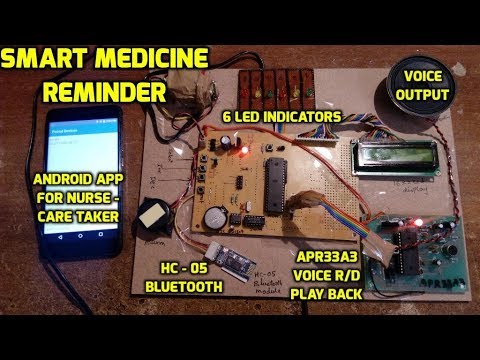
remind users or patients to take pills using sound and light. The specific number of pills needs

to be taken will be displayed by a seven segment led display placed on the corresponding box.

Compared with the traditional pillbox that requires users or nurses to load the box every day or

every week. Our smart medicine box would significantly release nurses or users’ burden on

frequently preloading pills for patients or users.





**1.2 Logical Structure**

Our device uses the state machine and real time clock (RTC) to provide real-time functionality.

This state machine determines which key has been pressed and provides keypad debounce

functionality. We use the 16MHz external oscillator to build a real time clock for the device.

Since the accuracy for the external crystal oscillator has a very high accuracy, and based on our

calculation, our RTC should only delay about several minutes in a week. Such error is tolerable,

since we not particularly used the RTC as an alarm clock.

The device's logic structure contains three major stages: 1. User initialization stage. 2.

Comparison stage. 3. Reminder stage. In the user initialization stage, the user enters the

current time, date and pill information (including amount and serve time for each type of pill).

After the user finishing entering all the information, the device will enter the comparison stage

unless the initialization button is pressed. During the comparison stage, the system compares

the pill information for each of the sub-box with the time counted by RTC. Once the information

entered by the user matches the RTC time, the system will jump out of comparison stage and

enter the reminder stage. In the reminder stage, the device will continuously play synthesized

voice, and the seven segments LED display will show the number of pills needs to be taken on each of the sub box.

**II. Software Design**

**2.1 Overall Software design**

We built a medicine box with an integrated software system running in the MCU. The

programming platform is AVRStudio4.0, and programming language is standard C and

WINAVR/GCC compiler. Generally, our software system can be divided into four parts, including

real time clock, user interface, LED control and sound generation. We could get information for

each medicine boxes from the user input and store the information in structure variables. The

real time clock would keep running once user finishes initialization. After all the information has

been entered. The system would enter comparison status. The comparison function would

detect if there were medicine should be taken at that time. When it finds medicine should be

taken, the audio will broadcast. After the user response to the system by pushing certain button,

LEDs would indicate the amount of medicines for each box. Our system flow chart is as follow.



**III. Hardware Design**

**3.1 Microcontroller**

The microcontroller used for the smart medicine box is the ATmega1284 mounted on a custom

PCB. We used port A for outputting numbers on led displays; port B for controlling the switches

and speaker module; port C for the LCD module; and Port D for keypad.

**3.2 LCD module**

The LCD module used in our project is a 16-characters, 2 lines Microtivity IM161 (with back

light). Considering its small size, ease of use and its yellow back, we think it is the best

candidate for our project. We found the yellow back light make it easier for the user to see the

characters displayed on it， even in the dark environment. Currently, we didn't have the back

light adjustment feature in our circuit. In the future, we will add this feature to our device so that

the user can dim the backlight during certain circumstances.

Pin 1 of the LCD module is connected to the ground. Pin2 is connected to the power supply of

the MCU. Pin 3 connects to the wiper of the 10k trimpot. Pin 4 is the register select, which is

connected to the C.0. Pin 5 is the data read/write, which is connected to C.1. Pin 6 is the enable

signal, which is connected to C.2. Pin 11 to pin 14 are the data bus, which are connected to

C.3-C.7. Pin15 and pin16 are the LED power and ground for the backlight.

**3.3 Keypad**

The keypad we used for our device is a 3x4 12-button keypad, which is purchased from all

electronics. In the software design we will explain the key scan algorithm in detail.

**3.4 Seven segment led displays**

We used Kingbright SC56-11EWA seven segment led displays for displaying the number of pills

the user need to take from each of the sub-box. SC56-11EWA is a common cathode led

display, which has seven pins corresponding to seven different segments on the display and

two pins as the ground. We designed circuits that allow us to use only 10 pins of the

microcontroller to control all seven units of these led displays (See figure below). The concept

behind this circuitry design is to use transistors as switches to turn the led displays on and off

sequentially. Such method can be realized with the use of a 3 to 8 decoder, which uses 3 pins

from the microcontroller to send 7 bits output to control the transistors. And the rest seven pins

from the microcontroller are used to send binary outputs to control the seven segments led

display one at a time. In our first prototype circuits, we used pnp transistors as switches, since

the 3 to 8 decoder can only generate one low output each time. We tested the circuits on a

breadboard with three led displays, and they all worked properly. However, after we soldered

the first prototype circuits with all seven led displays on it, we found the last three led displays

were always much dimmer than the rest led displays. After we talked with Bruce about our

circuit problem, he suggested us to redesign the circuits using npn transistors and inverters

instead. We then built our second prototype circuits on the breadboards due to the time.

**3.5 Speaker Modules and Audio Amplification Circuitry**

We used a speaker module acquired from ECE digital lab, but the output from the

microcontroller was not large enough to drive it. Therefore, we used a sound amplification

circuitry based on a LM386 N-1 audio amplifier to amplify the sound from the microcontroller

(see figure below). After we implemented the audio amplification circuitry, we were able to

generate a much louder sound from the speaker module, and we were also able t adjust the

sound amplitude.

**Code:**

//all headfiles;

#include <stdio.h>

#include <inttypes.h>

#include <avr/io.h>

#include <avr/interrupt.h>

#include <avr/pgmspace.h>

#include <stdlib.h>

#include <string.h>

#include <util/delay.h> // needed for lcd\_lib

#include "lcd\_lib.h"

#include <math.h>

#define F\_CPU 16000000UL

#define begin {

#define end }

#define t1 20 //statemachine repeattime

#define t2 100//

#define t3 1000 // 1s base for the RTC

#define t4 1 // 4ms for led

#define t5 1000 //1 min for CompareF

16

//for audio

#define TableSize 2920 //refers to the following incl file

//Contains the packed 2-bit codes for syntehsis

//Generated by the program Make2code476.m

#include "DPCMAllDigits.h"

//reconstruction differentials

// PCMvalue[4] = {-78, -16, 16, 78};

volatile signed char PCMvalue[4] = {-20, -4, 4, 20};

volatile unsigned int outI, tableI; //indexes

volatile unsigned char cycle ; //decode phase counter

volatile signed char out, lastout; //output values

volatile unsigned char p1, p2, p3, p4; //hold 4 differentials

volatile unsigned char packed ; //byte containing 4 2-bit values

int firstenter=1; // for the sound

//=====================================================

int countdisplay=0;

//Box information

struct box

{

int dayofweek[7];

int times2eat;

int amount2eat;

int flag;

};

struct box box[7];

int boxnum=0;

// RTC PARAMETERS

int second1,second2,minute1,minute2,hour,weekdays;

second1=0;

second2=0;

minute1=9;

minute2=5;

hour=7;

weekdays=1;//1 represents Monday and so on

//fake time====

second11=0;

second22=0;

minute11=0;

minute22=0;

hour1=0;

int runflag;//flag for running

volatile int hitflag;

volatile int responseb;

//State machine state names

#define NoPush 1

#define MaybePush 2

#define Detect 3

#define StillType 4

#define Release 5

#define StillTerm 6

#define DebounceTerm 7

#define Done 8

#define RunState 9

int8\_t InputString[17]; // The string of numbers we entered

unsigned char PushState; //state machine

volatile unsigned char timeofstatemachine, timeofbuttonRes,timedisplay;

volatile int timeCompareF; //for ComparaF

volatile int timesound;

volatile int trtc; //for real time clock

//for keypad scan============================================

#define maxkeys 12

#define PORTDIR DDRD

#define PORTDATA PORTD

#define PORTIN PIND

// The raw keyscan

unsigned char key;

// The decoded button number

unsigned int butnum,position,i,value,multi;

// the last key pushed

unsigned char lastbutnum;

//key pad scan table

unsigned char keytbl[16]=

{0xee, 0xde,0xbe,0xed, //{1 2 3 4

0xdd, 0xbd,0xeb, 0xdb, // 5 6 7 8

0xbb, 0xe7, 0xd7, 0xb7, //9 \* 0 #

};

//============================================================

//LED display library

unsigned char number[10]=

{

0b1111110,

0b1001000,

0b0111101,

0b1101101,

0b1001011,

0b1100111,

0b1110111,

0b1001100,

0b1111111,

0b1101111

};

//LCD display

int8\_t lcd\_buffer[17];

int Pointer=0;

int OldPointer=8; //for the position of the pointer when setting the

days of the week at the starting of the system

const int8\_t LCD\_initialize[] PROGMEM = "LCD Initialized\0";

const int8\_t LCD\_p1[] PROGMEM = "m t w t f s s";

const int8\_t LCD\_p2[] PROGMEM = "Set minutes: ";

const int8\_t LCD\_p3[] PROGMEM = "Set Hours: ";

const int8\_t LCD\_p4[] PROGMEM = "BOX1 Date: ";

const int8\_t LCD\_p5[] PROGMEM = "BOX1 Time: ";

const int8\_t LCD\_p6[] PROGMEM = "BOX1 Amount:";

const int8\_t LCD\_p7[] PROGMEM = "BOX2 Date: ";

const int8\_t LCD\_p8[] PROGMEM = "BOX2 Time: ";

const int8\_t LCD\_p9[] PROGMEM = "BOX2 Amount: ";

const int8\_t LCD\_p10[] PROGMEM = "BOX3 Date: ";

const int8\_t LCD\_p11[] PROGMEM = "BOX3 Time: ";

const int8\_t LCD\_p12[] PROGMEM = "BOX3 Amount: ";

const int8\_t LCD\_p13[] PROGMEM = "BOX4 Date: ";

const int8\_t LCD\_p14[] PROGMEM = "BOX4 Time: ";

const int8\_t LCD\_p15[] PROGMEM = "BOX4 Amount: ";

const int8\_t LCD\_p16[] PROGMEM = "BOX5 Date: ";

const int8\_t LCD\_p17[] PROGMEM = "BOX5 Time: ";

const int8\_t LCD\_p18[] PROGMEM = "BOX5 Amount: ";

const int8\_t LCD\_p19[] PROGMEM = "BOX6 Date: ";

const int8\_t LCD\_p20[] PROGMEM = "BOX6 Time: ";

const int8\_t LCD\_p21[] PROGMEM = "BOX6 Amount: ";

const int8\_t LCD\_p22[] PROGMEM = "BOX7 Date: ";

const int8\_t LCD\_p23[] PROGMEM = "BOX7 Time: ";

const int8\_t LCD\_p24[] PROGMEM = "BOX7 Amount: ";

const int8\_t LCD\_p25[] PROGMEM = "Time to eat";

const int8\_t LCD\_space[] PROGMEM = " ";

const int8\_t Monday[] PROGMEM = "MON";

const int8\_t Tuesday[] PROGMEM = "TUS";

19

const int8\_t Wednesday[] PROGMEM = "WED";

const int8\_t Thursday[] PROGMEM = "THU";

const int8\_t Friday[] PROGMEM = "FRI";

const int8\_t Saturday[] PROGMEM = "SAT";

const int8\_t Sunday[] PROGMEM = "SUN";

unsigned int paranum=0; //For parameter input and LCD showing staff

unsigned int lock=0;

//keypad scanf function===================================

void scanfkeypad()

begin

//get lower nibble

PORTDIR = 0x0f;

PORTDATA = 0xf0;

\_delay\_us(5);

key = PORTIN;

//get upper nibble

PORTDIR = 0xf0;

PORTDATA = 0x0f;

\_delay\_us(5);

key = key | PORTIN;

butnum=0;

//find matching keycode in keytbl

if (key != 0xff)

begin

for (butnum=0; butnum<maxkeys; butnum++)

begin

if (keytbl[butnum]==key)

break; // break when keyscan finds the pressed key

end

if (butnum==maxkeys)

butnum=0; // detect more than one key is pushed

else butnum++; // adjust to 1-16

end // end the search

else butnum=0;

end //end keyscan

//==============real time clock =================

void rtc()

{

trtc=t3; //reset t3

second1++;

20

if (second1>9)

{

second2++;

second1=0;

}

if(second2==6)

{

minute1++;

second2=0;

}

if(minute1>9)

{

minute2++;

minute1=0;

}

if(minute2==6)

{

hour++;

minute2=0;

}

if(hour==24)

{

hour=0;

weekdays++;

}

if(weekdays==8)

{

weekdays=1;

}

}

//====================================ISR================================

ISR (TIMER1\_COMPA\_vect)

begin

if(timeofstatemachine>0)timeofstatemachine--; //statemachine start every 25

ms

if(timeofbuttonRes>0)timeofbuttonRes--; //screen responds function excute

every 100ms

if (trtc>0) trtc--;

if (timesound>0) timesound--;

if (timedisplay>0) timedisplay--;

if(timeCompareF>0) timeCompareF--;

21

end

//generate waveform at 7812 scamples/sec

ISR (TIMER2\_OVF\_vect)

begin

//compute next sample

cycle = outI & 3; // outI modulo 4

if (cycle==0) //do we need to unpack more data?

begin

if (tableI<TableSize) //end of stored wave?

begin

//unpack a table entry into 2-bit indexs

// pgm\_read\_byte (address\_short)

packed = pgm\_read\_byte(&DPCMAllDigits[tableI]) ;

//packed = DPCMAllDigits[tableI];

p1 = (packed>>6) & 3 ;

p2 = (packed>>4) & 3 ;

p3 = (packed>>2) & 3 ;

p4 = (packed & 3);

tableI++ ;

end //end unpack table entry

//compute the output and send to PWM

out = lastout + PCMvalue[p1] - (lastout>>3) ;

end

else if (cycle==1) //don't need to unpack yet--just ouput

out = lastout + PCMvalue[p2] - (lastout>>3) ;

else if (cycle==2)

out = lastout + PCMvalue[p3] - (lastout>>3) ;

else if (cycle==3)

out = lastout + PCMvalue[p4] - (lastout>>3) ;

//update outputs

OCR0A = out + 128;

lastout = out;

outI++;

//at end, turn off TCCRO

if (tableI==TableSize) TCCR0B = 0;

end //ISR

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// LCD setup

void init\_lcd(void)

begin

LCDinit(); //initialize the display

22

LCDcursorOFF();

LCDclr(); //clear the display

LCDGotoXY(0,0);

CopyStringtoLCD(LCD\_initialize, 0, 0); // display initialize to test the function of LCD

end

//===========================================================

//Set it all up

void initialize(void)

begin

init\_lcd();

//for audio==================================================

DDRB=(1<<PORTB3);

// turn on pwm with period= 256 cycles

// (62,500 samples/sec) in fast PWM mode.

// BUT OCR0A update is done using timer2 at 7800/sec

// timer 0 runs at full rate set in MAIN loop; TCCR0B = 1 ;

// turn on fast PWM and OC0A output

// 16 microsec per PWM cycle sample time

TCCR0A = (1<<COM0A0) | (1<<COM0A1) | (1<<WGM00) | (1<<WGM01) ;

OCR0A = 128 ; // set PWM to half full scale

// turn on timer2 set to overflow at 7812 Hz

// (prescaler set to divide by 8)

TCCR2B = 2;

// turn on overflow interrupt

TIMSK2 = (1<<TOIE2);

///============================================================

DDRA=0xff; //set A as the output of LED number

DDRB=0x0F;

//B.0-B2 as output for choosing LED

//set up timer 1 for 1 mSec timebase for fast pwm mode and full speed

TIMSK1 = 2; //turn on timer 1 cmp match ISR

OCR1A = 249; //set the compare reg to 250 time ticks

//TCCR1A = 0b00000010; // turn on clear-on-match

TCCR1B = 0b00001011; // clock prescalar to 64 and turn on CTC

//initialize time variables

timeofstatemachine=t1;

timeofbuttonRes=t2;

trtc=t3;

timedisplay=t4;

timeCompareF=t5;

//set flag

runflag=0;

hitflag=0;

23

responseb=0;

int i;

for(int i=0;i<7;i++){

box[i].flag=0;

}

//init the state machine

PushState = NoPush;

position = 0; // the count for the input number to the string buffer

multi = 1; // used for converting char to integer

value = 0;

paranum=1;

//crank up the ISRs

sei() ;

end

//===============================================================

//show the time

void showtime(int pos)

{

LCDGotoXY(0,pos);

sprintf(lcd\_buffer, "%d:%d%d:%d%d",hour,

minute2,minute1,second2,second1);

LCDstring(lcd\_buffer, strlen(lcd\_buffer));

switch (weekdays)

{

case 1:

CopyStringtoLCD(Monday, 11,pos);

break;

case 2:

CopyStringtoLCD(Tuesday, 11,pos);

break;

case 3:

CopyStringtoLCD(Wednesday, 11,pos);

break;

case 4:

CopyStringtoLCD(Thursday, 11,pos);

break;

case 5:

CopyStringtoLCD(Friday, 11,pos);

24

break;

case 6:

CopyStringtoLCD(Saturday, 11,pos);

break;

case 7:

CopyStringtoLCD(Sunday, 11,pos);

break;

}

}

//=====================LDE DISPALY===============================

void led()

begin

timedisplay=t4;

if((countdisplay==0)&&(box[0].flag==1))

{

PORTB = (0<<PINB0)|(0<<PINB1)|(0<<PINB2);

PORTA=number[box[0].amount2eat];

}

if((countdisplay==1)&&(box[1].flag==1))

{ PORTB = (1<<PINB0)|(0<<PINB1)|(0<<PINB2);

PORTA=number[box[1].amount2eat];

}

if((countdisplay==2)&&(box[2].flag==1))

{ PORTB=(0<<PINB0)|(1<<PINB1)|(0<<PINB2);

PORTA=number[box[2].amount2eat];

}

if((countdisplay==3)&&(box[3].flag==1))

{ PORTB=(1<<PINB0)|(1<<PINB1)|(0<<PINB2);

PORTA=number[box[3].amount2eat];

}

if((countdisplay==4)&&(box[4].flag==1))

{ PORTB=(0<<PINB0)|(0<<PINB1)|(1<<PINB2);

PORTA=number[box[4].amount2eat];

}

if((countdisplay==5)&&(box[5].flag==1))

{ PORTB=(1<<PINB0)|(0<<PINB1)|(1<<PINB2);

PORTA=number[box[5].amount2eat];

}

if((countdisplay==6)&&(box[6].flag==1))

25

{ PORTB=(0<<PINB0)|(1<<PINB1)|(1<<PINB2);

PORTA=number[box[6].amount2eat];

}

countdisplay++;

if(countdisplay>6) countdisplay=0;

end

//=====================COMPARE FUNcTION===============================

void CompareF()

begin

timeCompareF=t5;

int i;

int j;

for(i=0;i<7;i++)

{

for(j=0;j<7;j++)

{

if(box[i].dayofweek[j]==weekdays&&minute1==0&&minute2==0&&second1==0&&second2==0)

{

LCDGotoXY(12,1); // location for the pointer

sprintf(lcd\_buffer, "%d",box[0].dayofweek[0]);

LCDstring(lcd\_buffer, strlen(lcd\_buffer));

switch (hour)

{

case 8:

if(box[i].times2eat==2||box[i].times2eat==3)

box[i].flag=1;

hitflag=1;

lock=0;

//send messge to turn on led

break;

case 12:

if (box[i].times2eat==1||box[i].times2eat==3)

box[i].flag=1;

hitflag=1;

lock=0;

break;

case 18:

26

if (box[i].times2eat==2||box[i].times2eat==3)

box[i].flag=1;

hitflag=1;

lock=0;

break;

}//switch

} //if

}//for

}

scanfkeypad();

if(butnum==10)

{

PushState=NoPush;

paranum=4;

lock=0;

runflag=0;

boxnum=0;

}

if(butnum==12)

{

hitflag=0;

LCDclr();

responseb=0;

PORTA=0b00000000;

for(i=0;i<7;i++)box[i].flag=0;

}

end

//==================================================================

void StaticString()

begin

switch(paranum)

begin

27

case 0:

if(lock==0)

{

LCDclr();

showtime(0);

lock=1;

}

break;

case 1:

if(lock==0)

{

LCDclr();

CopyStringtoLCD(LCD\_p1,0,0); // print out m t w...

lock=1;

}

break;

case 2:

if (lock==0)

{

LCDclr();

CopyStringtoLCD(LCD\_p2,0,0);

lock=1;

}

break;

case 3:

if (lock==0)

{

LCDclr();

CopyStringtoLCD(LCD\_p3,0,0);

lock=1;

}

break;

case 4:

if (lock==0)

{

LCDclr();

CopyStringtoLCD(LCD\_p4,0,0);

lock=1;

}

28

break;

case 5:

if(lock==0)

{

LCDclr();

CopyStringtoLCD(LCD\_p5,0,0);

lock=1;

}

break;

case 6:

if (lock==0)

{

LCDclr();

CopyStringtoLCD(LCD\_p6,0,0);

lock=1;

}

break;

case 7:

if (lock==0)

{

LCDclr();

CopyStringtoLCD(LCD\_p7,0,0);

lock=1;

}

break;

case 8:

if (lock==0)

{

LCDclr();

CopyStringtoLCD(LCD\_p8,0,0);

lock=1;

}

break;

case 9:

if (lock==0)

{

LCDclr();

CopyStringtoLCD(LCD\_p9,0,0);

lock=1;

}

break;

case 10:

if (lock==0)

{

29

LCDclr();

CopyStringtoLCD(LCD\_p10,0,0);

lock=1;

}

break;

case 11:

if (lock==0)

{

LCDclr();

CopyStringtoLCD(LCD\_p11,0,0);

lock=1;

}

break;

case 12:

if (lock==0)

{

LCDclr();

CopyStringtoLCD(LCD\_p12,0,0);

lock=1;

}

break;

case 13:

if (lock==0)

{

LCDclr();

CopyStringtoLCD(LCD\_p13,0,0);

lock=1;

}

break;

case 14:

if (lock==0)

{

LCDclr();

CopyStringtoLCD(LCD\_p14,0,0);

lock=1;

}

break;

case 15:

if (lock==0)

{

LCDclr();

CopyStringtoLCD(LCD\_p15,0,0);

lock=1;

}

30

break;

case 16:

if (lock==0)

{

LCDclr();

CopyStringtoLCD(LCD\_p16,0,0);

lock=1;

}

break;

case 17:

if (lock==0)

{

LCDclr();

CopyStringtoLCD(LCD\_p17,0,0);

lock=1;

}

break;

case 18:

if (lock==0)

{

LCDclr();

CopyStringtoLCD(LCD\_p18,0,0);

lock=1;

}

break;

case 19:

if (lock==0)

{

LCDclr();

CopyStringtoLCD(LCD\_p19,0,0);

lock=1;

}

break;

case 20:

if (lock==0)

{

LCDclr();

CopyStringtoLCD(LCD\_p20,0,0);

lock=1;

}

break;

case 21:

if (lock==0)

31

{

LCDclr();

CopyStringtoLCD(LCD\_p21,0,0);

lock=1;

}

break;

case 22:

if (lock==0)

{

LCDclr();

CopyStringtoLCD(LCD\_p22,0,0);

lock=1;

}

break;

case 23:

if (lock==0)

{

LCDclr();

CopyStringtoLCD(LCD\_p23,0,0);

lock=1;

}

break;

case 24:

if (lock==0)

{

LCDclr();

CopyStringtoLCD(LCD\_p24,0,0);

lock=1;

}

break;

case 25:

if(hitflag==0)

showtime(0);

else if(lock==0)

{

LCDclr();

CopyStringtoLCD(LCD\_p25,0,0);

lock=1;

}

break;

end // switch

end

32

//==================================================================

void buttonResponse() // refresh lcd every 100ms and

begin

timeofbuttonRes=t2;

switch(paranum)

{

case 0:

showtime(0);

break;

case 1:

//CopyStringtoLCD(LCD\_p1,0,0); // print out m t w...

LCDGotoXY(Pointer,1); // location for the pointer

sprintf(lcd\_buffer, "%c",94);

LCDstring(lcd\_buffer, strlen(lcd\_buffer)); // display the new pointer

LCDGotoXY(OldPointer,1);

sprintf(lcd\_buffer, "%s"," ");

LCDstring(lcd\_buffer, strlen(lcd\_buffer)); // erase the old pointer

break;

case 2:

showtime(1);

break;

case 3:

showtime(1);

break;

case 4: // box1 date

LCDGotoXY(0,1);

LCDstring(InputString, strlen(InputString));

break;

case 5: ///box1 time

LCDGotoXY(0,1);

LCDstring(InputString, strlen(InputString));

break;

case 6:

LCDGotoXY(0,1);

LCDstring(InputString, strlen(InputString));

break;

case 7:

LCDGotoXY(0,1);

33

LCDstring(InputString, strlen(InputString));

break;

case 8:

LCDGotoXY(0,1);

LCDstring(InputString, strlen(InputString));

break;

case 9:

LCDGotoXY(0,1);

LCDstring(InputString, strlen(InputString));

break;

case 10:

LCDGotoXY(0,1);

LCDstring(InputString, strlen(InputString));

break;

case 11:

LCDGotoXY(0,1);

LCDstring(InputString, strlen(InputString));

break;

case 12:

LCDGotoXY(0,1);

LCDstring(InputString, strlen(InputString));

break;

case 13:

LCDGotoXY(0,1);

LCDstring(InputString, strlen(InputString));

break;

case 14:

LCDGotoXY(0,1);

LCDstring(InputString, strlen(InputString));

break;

case 15:

LCDGotoXY(0,1);

LCDstring(InputString, strlen(InputString));

break;

case 16:

34

LCDGotoXY(0,1);

LCDstring(InputString, strlen(InputString));

break;

case 17:

LCDGotoXY(0,1);

LCDstring(InputString, strlen(InputString));

break;

case 18:

LCDGotoXY(0,1);

LCDstring(InputString, strlen(InputString));

break;

case 19:

LCDGotoXY(0,1);

LCDstring(InputString, strlen(InputString));

break;

case 20:

LCDGotoXY(0,1);

LCDstring(InputString, strlen(InputString));

break;

case 21:

LCDGotoXY(0,1);

LCDstring(InputString, strlen(InputString));

break;

case 22:

LCDGotoXY(0,1);

LCDstring(InputString, strlen(InputString));

break;

case 23:

LCDGotoXY(0,1);

LCDstring(InputString, strlen(InputString));

break;

case 24:

LCDGotoXY(0,1);

LCDstring(InputString, strlen(InputString));

break;

35

}

end

//=================================================================

void statemachine(void)

begin

timeofstatemachine=t1; //reset the task timer

switch (PushState)

begin

case NoPush:

scanfkeypad(); // keypad scan

if (butnum!=0)

begin

PushState=MaybePush; // goes to maybepush when butnum not 0

lastbutnum=butnum;

end

else

PushState=NoPush;

break;

case MaybePush:

scanfkeypad();

if (butnum==lastbutnum)

begin

PushState=Detect; //when button is still pushed go to detect

whether "enter" key is being pressed

end

else

PushState=NoPush;

break;

case Detect:

if(butnum==12)//enter key

{ PushState = StillTerm;

}

if (PushState == StillTerm) break;

//set the system weekdays

if(paranum==1)

{

OldPointer=Pointer;

if (butnum==8)Pointer=Pointer-2;

if(butnum==9)Pointer=Pointer+2;

if(Pointer<0)Pointer=12;

if(Pointer>12)Pointer=0;

36

PushState = StillType;

}

//set the system time

if(paranum>1&&paranum<4)

{

PushState = NoPush;

switch(paranum)

{

case 2:

if(butnum==5)//first up

minute2++;

if(minute2>5)

minute2=0;

if(butnum==8)//first down

minute2--;

if(minute2<0)

minute2=9;

if(butnum==6)//second up

minute1++;

if(minute1>9)

minute1=0;

if(butnum==9)//second down

minute1--;

if(minute1<0)

minute1=5;

break;

case 3:

if(butnum==5)//first up

hour++;

if(hour>24)

hour=0;

if(butnum==8)//first down

hour--;

if(hour<0)

hour=24;

break;

}

}

//Box information

if(paranum>=4)

{

PushState = StillType;

if (butnum!=10&&butnum!=12)

37

{

InputString[position] = butnum+'0';

position++;

}

if( butnum==10)

{

position--;

InputString[position]=' ';

}

}

break;

case StillType:

scanfkeypad();

if (butnum == lastbutnum)

begin

PushState = StillType;// the button is still pressed

end

else

PushState = Release; // the button does released

break;

case Release:

scanfkeypad();

if (butnum == lastbutnum)

PushState = StillType; // to remove debounce

else

PushState = NoPush; //go to the first state and press a new character

break;

case StillTerm:

scanfkeypad();

if (butnum == lastbutnum)

PushState = StillTerm; // it's the debounce step, so if the button is

still pressetd it goes to itself

else

PushState = DebounceTerm;

break;

case DebounceTerm:

scanfkeypad();

if (butnum == lastbutnum)

PushState = StillTerm; // if it's still pressed go to the last state to

scan again

else

PushState = Done;

break;

38

case Done:

lock=0; //reset lock ===========

if(paranum==1) weekdays=Pointer/2+1;

if(paranum>=4)

{

for(int k=0;k<position-1;k++)//just translate the input string into numberial

value

begin

multi=multi\*10;

end

for (i = 0;i < position ;i++)

begin

value = value+((int)(InputString[i] - '0') \* multi);

multi = multi / 10;

end

int sw=paranum-3-boxnum\*3;

switch (sw)

{

case 1:

for(i = 0;i < position ;i++)

box[boxnum].dayofweek[i]=InputString[i]-'0';

break;

case 2:

box[boxnum].times2eat=value;

break;

case 3:

box[boxnum].amount2eat=value;

boxnum++;

break;

}

for(int i=0;i<position;i++)//clear the buffer for the input string

InputString[i]=' ';

position=0;//clear variables for the next parameter input

multi=1;

value=0;

39

}

paranum++;

if(paranum>=25)

{

runflag=1;

LCDclr();

}

else

PushState=NoPush;

break;

end

end

//main==============================================================

int main(void)

begin

initialize();

LCDclr();

while(1)

begin

if(trtc==0) rtc();

StaticString();

if(runflag==0)

{

if(timeofbuttonRes==0) buttonResponse();// excute buttonResonse every 100 ms

if(timeofstatemachine==0) statemachine();

}

else

{

if(timeCompareF==0) CompareF();

}

if(hitflag==1)

{

//init the output indexes

if(responseb==0)

{

switch (firstenter)

40

{

case 1:

outI = 0;

tableI = 0;

//init the ouptut value

lastout = 0;

// turn on PWM

TCCR0B = 1;

firstenter=2;

break;

case 2:

//wait until the speech is done then

//time delay the next utterance.

if(TCCR0B==0)

{\_delay\_ms(1000);

firstenter=1;

}

break;

}// switch

} // if

else

{

if(timedisplay==0) led();

}

scanfkeypad();

if(butnum==11)

{

responseb=1;

}

}

end

end