Activity Analyzer with voice-Guidance for Independent Living Environments (AAGILE)

Josh Harvey, Tanya Wang

11/2/2012

Abstract

The purpose of the project is to develop an interactive, portable device capable of monitoring motion as well as provide user feedback to promote continued exercise. The intended utility of the Activity Analyzer is testing whether or not audio and digital feedback promotes higher levels of exercise in the elderly population, particularly when the audio feedback is comprised of personalized messages recorded by loved ones. The device uses a microprocessor interfaced with a voice record chip, as well as a three dimensional accelerometer for motion input and detection. The onboard microprocessor is used to evaluate physical performance, determine which messages to be played at specified times, as well as save all messages and scores (including a time stamp) to the onboard EEPROM. A prototype of the Activity Analyzer has been successfully constructed.

Origin of problem

The idea behind the project was presented to us by Dr.Burbank from the nursing department at URI for an elderly family member. Today's technology driven world has provided us with many ways of monitoring activity, ranging from pedometers, GPS enabled distance tracker apps, and even Nike+ shoe inserts. The neglected aspect of this activity monitoring is that the elderly population doesn't get integrated into these devices very easily. The elderly population has more trouble understanding technology, and can be very forgetful. The idea behind AAGILE is to target the needs of the elderly patient in the following ways: Loud and customized voice feedback to make reminders more personal, single-button interface making control of the device extremely user friendly, Sensitivity controls targeting the patient's activity threshold, and customized GUI interface for easy time-based activity monitoring.

Realistic Constraints

The greatest concern for this project is to increase mobility and overall health of the elderly population. With this in mind, the health, and safety of this product is the number one concern. The power source and all equipment are enclosed in a self-contained unit with no leads to the outside. The input provided to the unit is through a 3 axis accelerometer, as well as a single push-button input, and the output is a series of custom pre-recorded voice recording. The recordings are set at a reasonable volume within safe constraints. The mobile unit is powered by a single 9V rechargeable battery, and is regulated with a series of voltage regulators. The docking unit has a separate charging unit which also meets all safety constraints. The entirety of one mobile unit is estimated to cost under $70 and with mass manufacturing will that price.

International Electrotechnical Commission Safety Standards

IEC provides constraints safety constraints IEC 60601-1-6, (Third Edition)

The following clauses are related to the project: (4.1, 4.1.1, 4.1.2, 5, 5.1, 5.2, 5.5, 5.8, 6, 7)

The AAGILE must be usable under normal use, and possible user error.

Provide documentation relating to usage, transportation, maintenance, installation.

Provide documentation of usability and performance provided in usability engineering file.

Usability specifications determining user interaction, such as daily use/charging times, possible user scenarios, possible user errors, primary operating functions.

Medical device is to be compared to usability validation plan.

Necessary training materials provided for operating functions

Bill of Materials

Processor	PIC18F452	$8.20	1
Accelerometer	LIS302SG	$7.17	1
Voice chip	ISD1750PY	$7.60	1
Serial LCD 16x2	Sparkfun LCD-09066	$24.95	1
USB to serial	Sparkfun BOB-00718	$14.95	1
Battery/charger	Batterymart C-RLI-9600-SET	$28.95	1
Box (AAVID)	Digikey 377-1167-ND: 4.1x2.23x2	$4.50	1
Box (docking)	Jameco 18914: 4.875x2.5x1.5	$3.95	1
Prototyping boards	X 2	$6.00	1
Connector 15-pos	Female 609-2802-ND	$2.50	1
Connector 15-pos	Male 609-4042-ND	$1.34	1
Switches	Power, sliding switches, IDC connectors	$10.00	1
Others	Other electronic components	$5.00	
JK flip flop	4027		1
555 timer			1
inverter	4069		1
5V regulator	LM7805		1
Multiplexor	MC14066		1
4MHz Crystal			1
Op Amps	LM386n03		2
3.3V regulator	3302e		1
Mechanical Switch	EDR292A0500		1
Diode	Zener		1
Speaker			1

Project Management

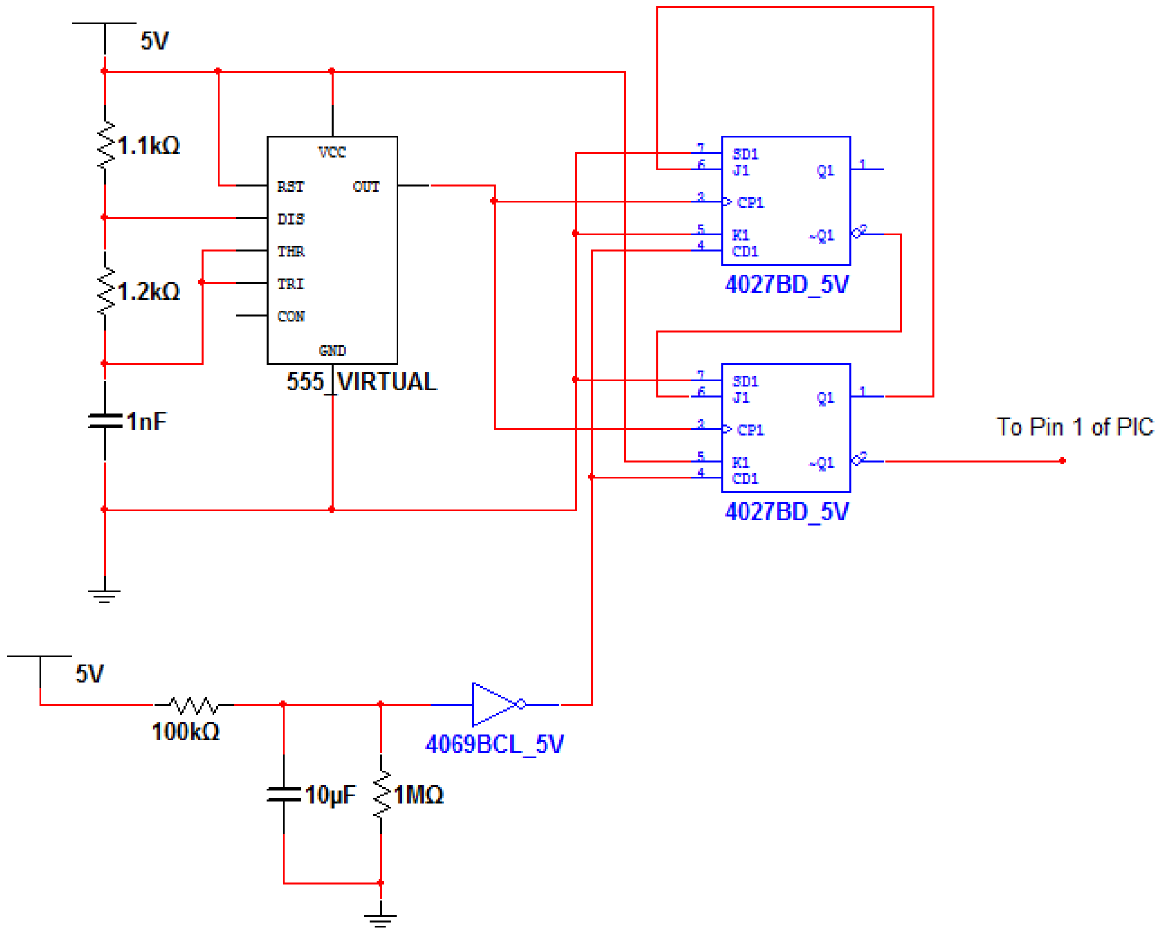
The overall management of the project is balanced between both team members, Josh Harvey and Tanya Wang. The troubleshooting of software problems is Josh’s expertise, and that of hardware is Tanya’s. Each member is responsible for creating a functional mobile unit.

IRB approval

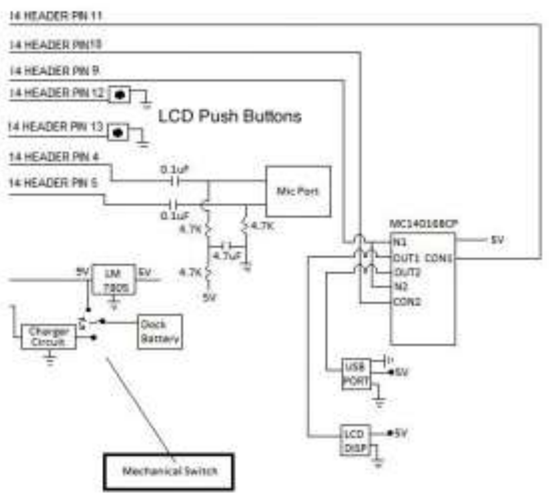
The IRB approval is required for trials with healthy young individuals, as well as testing with the targeted elder population. Documentation is provided in the appendix.

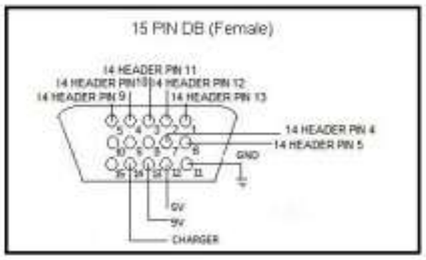
Schematics

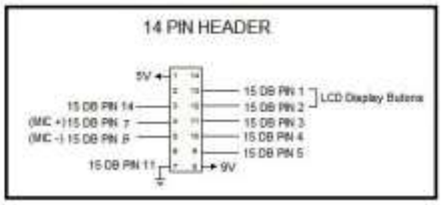
Start-up Correction Schematic

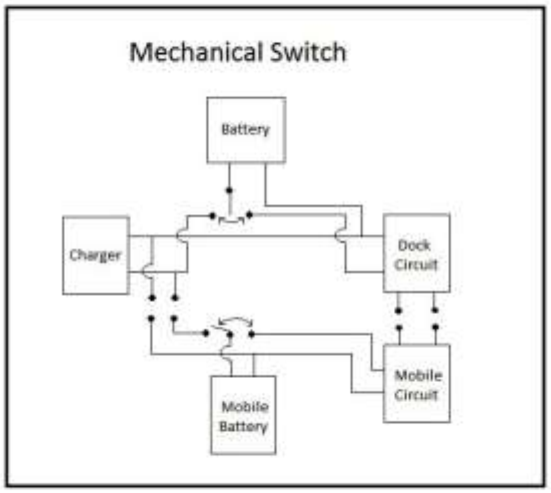


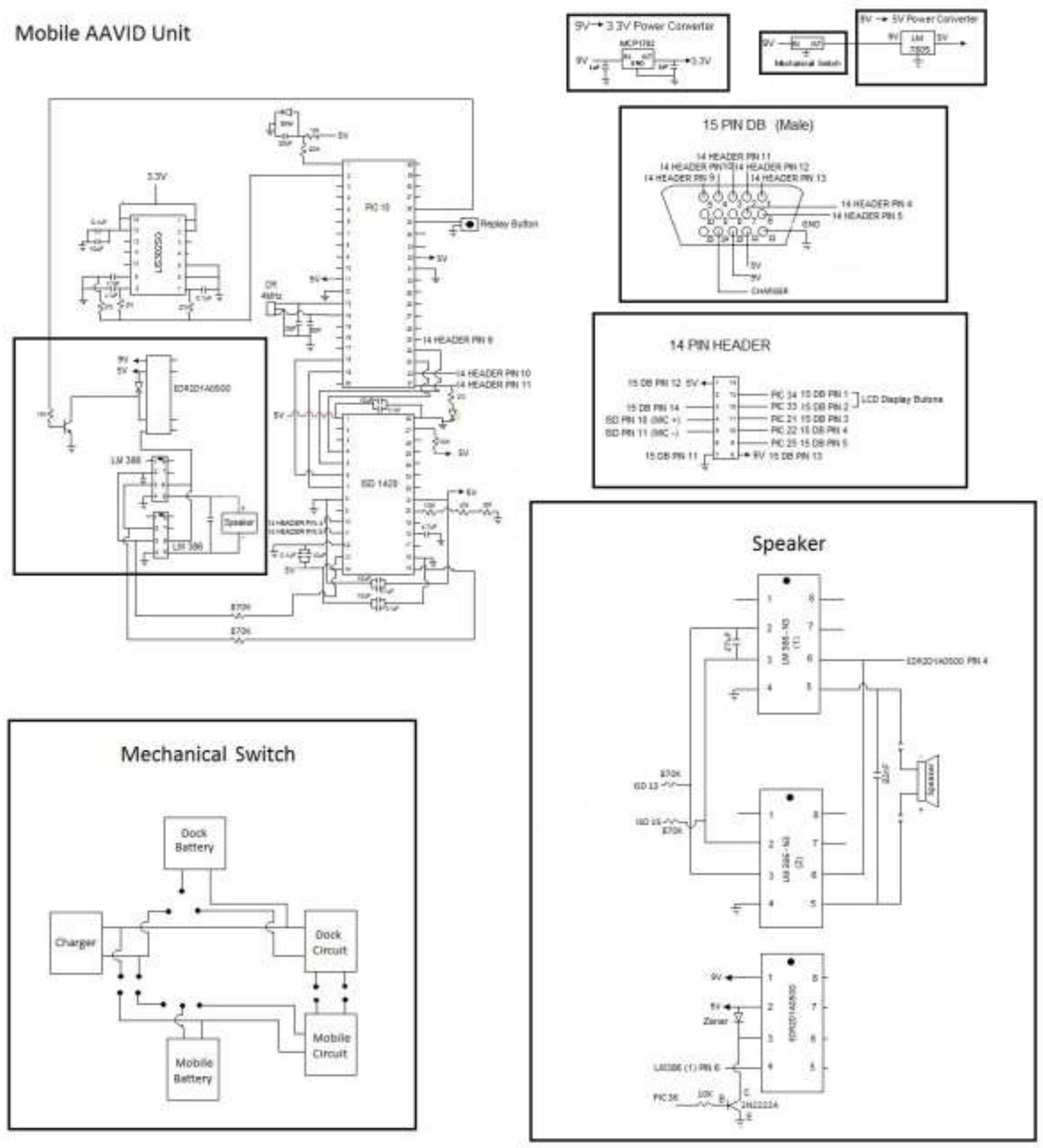
Docking Unit Schematic











Methods

The microprocessor of choice for this device is the PIC18F452 (Microchip Technology, Chandler, AZ). This processor is interfaced with the ISD1750 (Winbond) voice record (VR) chip, which is capable of recording up to 60 to 100 seconds worth of message time (dependent on the load resistor chosen). To enable an efficient communication, a serial-parallel interface (SPI) is established between the processor and VR chip.

For detecting motion a three-dimensional accelero-meter (LIS302sg, STmicro-electronics) is used. The accelerometer is capable of outputting motion signals in the x, y, and z directions. To eliminate discontinuities from the varying x, y, and z baselines, the outputs are averaged together through a resistive network and sent to the analog-to-digital converter (ADC) of the PIC processor.

There are two components to this project, the Activity Analyzer mobile unit and the docking system (both depicted individually in Fig. 1). The mobile unit contains the accelerometer, PIC processor, VR chip, and speaker. The docking system contains an LCD display, push buttons to interact with the mobile unit while docked, digital switch, USB device, and microphone input. While docked, the docking system is used to alter settings such as time of day, when messages are played, which messages are played, and how sensitive the accelerometer decoding algorithm is (all through a DB15 connection). The user can also record new messages, erase old ones, and send the data in the EEPROM of the PIC to a computer via USB.

B. Software Development

The software implemented for the PIC processor is comprised of four major components: the user menu controls, SPI communication, motion detection and scoring algorithm, and finally the memory storage algorithm. Figure 2 depicts a flow graph of these four major software components, as well as how they interact amongst one another in the code.

For ease of use, it is important for the device to have an intuitive menu interface. The user menu control is set up in correspondence with two exterior push buttons: one to scroll and one to select. The user can select several different menus from the LCD screen, including a menu to set time, a menu to record, erase, play, or set play times, a menu to send the data via USB to a computer, and a menu to set the sensitivity of motion detection to name a few.

By scrolling and selecting (once you enter a menu, the user has the option to select exit), the user can easily maneuver through all the option available to them. It is also worth noting

that most of these opera-tions are controlled by sub functions outside of the main loop in code. This means adding functions or changing current functions is easier for the developer as the framework to do so is already in place.

Both the processor and the VR chip are capable of SPI communication. The processor requires most significant bit first and the VR chip requires least significant bit first. To maintain compatibility, the reversed bits are defined as protocol functions in the code such that the VR chip receives LSB first. In order to use the VR chip, the power up command and clear INT command must be sent prior to sending the SPI command for the desired action. After the action is implemented (play, rec, erase, forward message, set_play, set_rec, set_erase) the power down command should be sent. When a message is recorded on the VR chip the device stores starting and ending points which can be read via play and rec pointers. Alternatively, set_rec, set_play, and set_erase commands can be implemented to record, play, and erase messages from memory location A to memory location B. This method allows for more control over message length and can be used to optimize memory usage of the VR chip.

An integral part of the device involves the data received by the processor from the 3D accelerometer. One major issue involved with accelerometers is the drift of baseline voltage: this baseline voltage will drift both with age and orientation. To counter drift, an algorithm using a 16-point averaging filter can be used. The PIC will fill a 16-point array with 16 bytes of data from the ADC and then average the value (baseline voltage) of the sixteen points. The sampling rate is set at 1150 Hz. Thus, the 16-point buffer represents a time frame of 14 ms. This average value will then be able to act as threshold for motion detection, i.e. any peaks of voltage over the average voltage plus some voltage (threshold) will cause the PIC to recognize motion and increment a scoring counter. This array is then emptied and refilled periodically, allowing the threshold to change with the baseline voltage and any possible drift that arises.

Another functional component of the Activity Analyzer algorithm is data storage: saving the scores and messages played (with a time stamp) to be extracted and analyzed at the end of the day. However, the PIC18F452 is only comprised of 256 bytes of EEPROM, making one day's worth of storage impossible, considering each score would need 3 bytes: one for the score, one for the hour, and one for the minute. The solution to this problem was to 'split' every bite in two, using the 4 most significant bits and 4 least significant bits individually. This, although complex and tedious, allowed us to essentially double the memory storage capability of the processor while maintaining a five-minute temporal resolution for the scoring algorithm.

Finally, the user can upload data from the EEPROM of the PIC onto a personal computer (PC) via the USB port. A program on the PC, created by using the C# language, is able to take in all the data, split the incoming bytes, and plot all the scores on a score vs. time scale. This allows the user to visualize the amount of exercise throughout the day.

References

[1] P.M. Burbank and D. Riebe. *Promoting Exercise and Behavior Change in Older Adults: Interventions and Transtheoretical Model*. NY: Springer, 2002.

[2] K. Rafferty, T. Alberg, H. Greene, Y. Sun, and P.M. Burbank. Development of an activity analyzer with voice directions for exercises. *38th Annual Northeast Bioengineering Conference, Temple University, Philadelphia, PA, March 16-18, 2012.*

[3] P.M Burbank,. Y Sun, and P.M. Burbank. Evaluation of the Device Effectiveness of the Activity Analyzer with Voice Individualized Direction. 2012

PIC 18F452 Code

```
// ________________________________________________________________________________ /
// AAVID Project code: Version 1.0                                   /
// Instructor: Dr. Ying Sun
                                     /
// Contributors: Kyle Rafferty, Timothy  Alberg, Ron Greene
/
// Completed: 18 August 2011
                                     /
//                                                                              /
//   The following code uses a PIC18f452 microcontroller and a ISD1750                      /
//   voice recording chip.  This code establishes an accurate clock (gains approx/
//   1 second every 2.5 hours) and a series of menus that may be navigated by the/
//   use of two push buttons, one for scroll and one for select.  Menu features  /
//   include setting the clock, recording 8 messages, and the ability to program /
//   times at which to play any of the first 7 messages (message 8 is the        /
//   reminder message and automatically plays after an hour of inactivity.                    /
//   The PIC also has an algorithm to take in data from the accelerometer and        /
//        save it to the EEPROM. This save will be in the form of a score and a time  /
//   stamp (hour and minute). At the end of the day, the data can be retrieved   /
//   via usb to the AAVID GUI and will be plotted. Along with the motion data    /
//   being saved, the messages will also be saved to memory along with the time  /
//    they were played to see if there is any correlation to messages being played/
//    and an increase in motion. We have also extensively commented the following /
//    code in hopes that, if and when this code needs to be updated, it will be   /
//            understood and therefore easy to update.
                        /
//            Future work would include making functions for the redundant message code   /
//            (i.e. the same code copied 8 times for all 8 messages) and calling the      /
//            functions in the main to make the code more efficient.                        /
// ______________________________________________________________________________ /

// _______________________ Declare Chip Type to Compiler ________________________ /
#pragma chip PIC18f452
#define PU            0x8000
#define CLR_INT       0x2000
#define PD            0xE000
#define SET_PLAY      0x0100
#define SET_REC                 0x8100
#define SET_ERASE      0x4100
#define message1_start 0x0800
#define message1_stop  0xEA00
#define message2_start 0x1A00
#define message2_stop  0x2500
#define message3_start 0xA500
#define message3_stop  0x8F00
#define message4_start 0x4F00
#define message4_stop  0x7C80
#define message5_start 0xFC80
#define message5_stop  0xEA80
#define message6_start 0x1A80
#define message6_stop  0xF680
#define message7_start 0x0E80
#define message7_stop  0xE180
#define message8_start 0x1180
#define message8_stop  0xF980
#define START_ADDR 0x00;
// ________________________ Define prototype functions __________________________ /
void _highPriorityInt (void);
char Read_adc(void);
void Delay_ms(char x);

// _____________________________ Global variables _______________________________ /
bit pm, message_timer, update_display, scroll, select, usb, replay, no_play, relay_flag, relay_flag1;
char second, minute, hour, message_timer_count_short, message_timer_count_long,
            button_delay, ad_input, scroll1;
uns8 relay_time, relay_time1;
unsigned long millisec, time_return, counter;

// _____________ Define High Priority Interrupt Service Routine ________________ /
```

```
#pragma origin 0x8
interrupt highPriorityInterrupt (void) {
        #pragma fastMode
        _highPriorityInt (); }
void _highPriorityInt (void) {
checkflags:
        if (TMR0IF == 1) {
                TMR0IE = 0;
                TMR0IF = 0;
                TMR0H = 0xFC;
                TMR0L = 0x36;
                ad_input = Read_adc();
                if(counter != 0) counter --;
                if (button_delay != 0) button_delay--;
                if (millisec < 999) millisec++;
                else if (second < 59){
                        update_display = 1;
                        millisec = 0;
                        second++;
                        relay_time++;
                        relay_time1++;
                if (message_timer) {
                        message_timer_count_short++;
                        message_timer_count_long++; }
                else message_timer_count_short = message_timer_count_long = 0;}
                else if (minute < 59) {
                        update_display = 1;
                        millisec = 0;
                        second = 0;
                        minute++;
                        no_play = 1;}
                else if (hour < 12) {
                        update_display = 1;
                        millisec = 0;
                        second = 0;
                        minute = 0;
                        hour++;
                        if (hour == 12) pm = !pm; }
                else {
                        update_display = 1;
                        millisec = 0;
                        second = 0;
                        minute = 0;
                        hour = 1; }
                TMR0IE = 1; }
        if(INT0IF == 1){
                INT0IF = 0;
                if(button_delay == 0){
                        usb = !usb;
                        scroll = !scroll;
                        button_delay = 200;
                        goto checkflags;}}
        if(INT1IF == 1){
                INT1IF = 0;
                if(button_delay == 0){
                        select = !select;
                        button_delay = 200;
                        goto checkflags;}}
        if(INT2IF == 1){
                INT2IF = 0;
                if(button_delay == 0){
                        replay = !replay;
                        button_delay = 200;
                        goto checkflags;}}}

// ___________________________ Define SetupADC Function ___________________________ /
void Setup_adc (char channel) {
        TRISA = 0xFF;
        ADCON1          = 0x00;
        ADCON0 = (channel << 3) + 0x41;
        ADIE = 0;
        ADIF = 0; }

// ___________________________ Define Read_adc Function ___________________________ /
char Read_adc () {
        char adc_value;
        GO = 1;
        while(!ADIF) continue;
        adc_value = ADRESH;
        return adc_value; }

// __________________________ Define Setup_spi Function ___________________________ /
void Setup_spi () {
        SSPEN = 0;
        SSPSTAT = 0xC0;
        SSPCON1 = 0x20;
        SSPIE = 0;
        TRISC = 0x90; }

// ________________________ Define Spi_read_byte Function _________________________ /
char Spi_read_byte () {
        char value;
        value = SSPBUF;
        return value; }

// ________________________ Define Spi_write_byte Function ________________________ /
```

```
void Spi_write_byte (char value) {
            SSPBUF = value;
            while (SSPIF == 0);
            SSPIF = 0; }

// _______________________ Define Spi_write_2bytes Function _______________________ /
void Spi_write_2bytes (unsigned long value) {
            char byte1, byte2;
            unsigned long value1, value2;
            value1 = value & 0xFF00;
            byte1 = value1 >> 8;
            value2 = value & 0x00FF;
            byte2 = value2;
            SSPBUF = byte1;
            while (SSPIF == 0);
            SSPIF = 0;
            SSPBUF = byte2;
            while (SSPIF == 0);
            SSPIF = 0; }

// _________________________ Define Setup_USART Function __________________________ /
void Setup_USART () {
            TRISC = 0x80;
            SPBRG = 25;
            TXEN = 1;
            SYNC = 0;
            CREN = 1;
            SPEN = 1;
            BRGH = 1; }

// __________________________ Define Transmit Function _____________________________ /
void Transmit (char value) {
            while (!TXIF) continue;
            TXREG = value;
            while (!TXIF) continue; }

// ________________________ Define Clear_screen Function __________________________ /
void Clear_screen () {
            Transmit (254);
            Transmit (0x01); }

// __________________________ Define Backlight Function ___________________________ /
void Backlight (char state) {
            Transmit (124);
            if (state) Transmit (0x9D);
            else Transmit (0x81); }

// ________________________ Define Set_position Function __________________________ /
```

```
void Set_position (char position) {
        Transmit(254);
        Transmit(128 + position); }

// _________________________ Define Print_line Function ___________________________ /
void Print_line (const * string, char num_chars) {
        char counter;
        for (counter = 0; counter < num_chars; counter++) Transmit (string [counter]); }

// __________________________ Define EEPROM READ __________________________________ /
char readEEPROM (void) {
        #asm                                    //Command tells compiler following code is assembly
        BCF EECON1, EEPGD;              //Clear bit 7 of EECON1 register: access EEPROM
        BCF EECON1, CFGS;               //Clear bit 6 of EECON1 register: access EEPROM memory
        BSF EECON1, RD;                 //Set bit 0 of EECON1 register: initiates EEPROM read
        MOVF   EEDATA, 0;               //Move EEDATA from EEPROM memory into buffer register W
        #endasm                                         //End assembly code
        return EEDATA; }                //Return the value of EEDATA (data saved in memory)

// __________________________ Define EEPROM WRITE _________________________________ /
void writeEEPROM (char value) {
        EECON1 = 0x04;                  //EECON1 = 0000x0100: set WREN bit to allow write
        EEDATA = value;                         //Store EEDATA to buffer value
        GIE = 0;                        //Disable all interrupts: can't have write sequence interrupted
        EECON2 = 0x55;                  //Set EECON2 = 0101.0101 (0x55) and then EECON2 =
        EECON2 = 0xAA;                          //1010.1010 (AA) as handshake procedure
        WR = 1;                                 //Set write bit = 1 to initiate write of value into memory
        WREN = 0;                               //End write sequence once write complete
        Delay_ms(10);                   //Allow time to complete
        GIE = 1; }                              //Enable global interrupts again

// __________________________ Define EEPROM Location Increment ____________________ /
char INCEEPROM (void) {
        #asm                                    //Command tells compiler following code is assembly
        BCF EECON1, EEPGD;              //Clear bit 7 of EECON1 register: access EEPROM
        BCF EECON1, CFGS;               //Clear bit 6 of EECON1 register: access EEPROM memory
        BSF EECON1, RD;                 //Set bit 0 of EECON1 register: initiates EEPROM read
        MOVF   EEDATA, 0;               //Move EEDATA from EEPROM memory into buffer register W
        INCFSZ  EEADR, 1;               //Increment memory address
        #endasm                                         //End assembly code
        return EEADR; }                 //Return EEADR

// __________________________ Define Specify Location Increment ___________________ /
char SpecifyEEPROM (char value) {
        EEADR = value;                  //Input hex value to move to that value in memory
        return EEADR; }                 //Ex: SpecifyEEPROM(0x05) will move EEADR to 0x05 in mem.

// _______________________ Define Print_2dig_num Function _________________________ /
```

```
void Print_2dig_num (char value, char position) {
          char tens, ones;
          Set_position (position);
          tens = value / 10;
          Transmit (tens + 48);
          position++;
          Set_position (position);
          ones = value - tens * 10;
          Transmit (ones + 48); }

// __________________________ Define Delay_ms Function _________________________ /
void Delay_ms (char x) {
          char y;
          for ( ; x > 0 ; x--)
   for ( y = 0; y < 165 ; y++); }

// __________________________ Define Delay_sec Function _________________________ /
void Delay_sec (char x) {
          char magicmicro;
          unsigned long milli;
          for ( ;x > 0; x--)
          for (milli = 0; milli < 1000; milli++)
          for (magicmicro = 0; magicmicro < 158; magicmicro++); }

// __________________________ Define Set_time Function___________________________ /
void Set_time () {
          bit set_am_pm, am_pm_choice, set_hour, set_minute;
          char set_hour_mode, hour_choice, set_minute_mode, minute_choice;
          uns16 am_pm_selection, hour_selection, minute_selection;
          set_am_pm = 1;
          update_display = 1;
          am_pm_choice = set_hour = set_minute = 0;
          set_minute_mode = minute_choice = 0;
          am_pm_selection = minute_selection = 0;
          set_hour_mode = hour_choice = hour_selection = 12;
          while (set_am_pm) {
                    if (update_display) {
                      Clear_screen ();
                      Set_position (0);
                      Print_line ("Set AM/PM:", 9);
                      Set_position (64);
                      Print_line ("    :   m",9);
                      Print_2dig_num (minute_selection,69);
                      Print_2dig_num (hour_selection,66);
                      Set_position (71);
                      if (am_pm_choice) Print_line ("p",1);
                      else Print_line ("a",1);
                      update_display = 0; }
```

```
            if (scroll) {
              scroll = 0;
              am_pm_choice = !am_pm_choice;
              update_display = 1; }
            if (select) {
              select = 0;
              set_am_pm = 0;
              if (am_pm_choice) am_pm_selection = 1;
              else am_pm_selection = 0;
              am_pm_choice = 0;
              set_hour = 1;
              update_display = 1;}}
while (set_hour) {
            if (update_display) {
              Clear_screen ();
              Set_position (0);
              Print_line ("Set Hours:", 10);
              hour_choice = set_hour_mode;
              Set_position (64);
              Print_line ("    :   m",9);
              Print_2dig_num (minute_selection,69);
              Print_2dig_num (hour_choice,66);
              Set_position (71);
              if (am_pm_selection == 1) Print_line ("p",1);
              else Print_line ("a",1);
              update_display = 0; }
            if (scroll) {
              scroll = 0;
              set_hour_mode++;
              if (set_hour_mode == 13) set_hour_mode = 1;
              update_display = 1; }
            if (select) {
              select = 0;
              set_hour = 0;
              hour_selection = set_hour_mode;
              set_hour_mode = 1;
              set_minute = 1;
              update_display = 1;}}
while (set_minute) {
            if (update_display) {
              Clear_screen ();
              Set_position (0);
              Print_line ("Set Minutes:", 12);
              minute_choice = set_minute_mode;
              Set_position (64);
              Print_line ("    :   m",9);
              Print_2dig_num (minute_choice,69);
              Print_2dig_num (hour_selection,66);
```

```
                    Set_position (71);
                    if (am_pm_selection == 1) Print_line ("p",1);
                    else Print_line("a",1);
                    update_display = 0; }
                  if (scroll) {
                    scroll = 0;
                    set_minute_mode++;
                    if (set_minute_mode == 60) set_minute_mode = 0;
                    update_display = 1; }
                  if (select) {
                    select = 0;
                    set_minute = 0;
                    minute_selection = set_minute_mode;
                    set_minute_mode = 0;
                    update_display = 1;}}
  am_pm_selection = am_pm_selection << 12;
  hour_selection = hour_selection << 8;
  time_return = am_pm_selection + hour_selection;
  time_return = time_return + minute_selection; }

// __________________________ Define Message_select Function ______________________ /
char Message_select() {
          bit message_select;
          char message_select_mode, message_selection;
          message_select = 1;
          message_select_mode = 0;
          while (message_select) {
                  if (update_display) {
                    Clear_screen ();
                    Set_position (0);
                    Print_line ("Select Message", 14);
                    Set_position (64);
                    if (message_select_mode == 0) Print_line ("Exit", 4);
                    else {
                      Print_line ("Message", 7);
                      Print_2dig_num (message_select_mode, 73); }
                    update_display = 0; }
                  if (scroll) {
                    scroll = 0;
                    message_select_mode++;
                    if (message_select_mode == 8) message_select_mode = 0;
                    update_display = 1; }
                  if (select) {
                    select = 0;
                    message_select = 0;
                    if (message_select_mode == 0) message_selection = 1;
                    else message_selection = message_select_mode;
                    update_display = 1; } }
```

```
        return message_selection; }

// ________________________ Define Play_message Function ________________________ /
void Play_message (unsigned long message_start, unsigned long message_stop) {
        PORTB.3 = relay_flag = 1;
        relay_time = 0;
        PORTD.0 = 0;
        Spi_write_2bytes (PU);
        PORTD.0 = 1;
        PORTD.0 = 0;
        Spi_write_2bytes (CLR_INT);
        PORTD.0 = 1;
        PORTD.0 = 0;
        Spi_write_2bytes (SET_PLAY);
        Spi_write_2bytes (message_start);
        Spi_write_2bytes (message_stop);
        Spi_write_byte (0x00);
        PORTD.0 = 1;
        PORTD.0 = 0;
        Spi_write_2bytes (PD);
        PORTD.0 = 1; }

// _______________________ Define Record_message Function _______________________ /
void Record_message (unsigned long message_start, unsigned long message_stop) {
        PORTD.0 = 0;
        Spi_write_2bytes (PU);
        PORTD.0 = 1;
        PORTD.0 = 0;
        Spi_write_2bytes (CLR_INT);
        PORTD.0 = 1;
        PORTD.0 = 0;
        Spi_write_2bytes (SET_REC);
        Spi_write_2bytes (message_start);
        Spi_write_2bytes (message_stop);
        Spi_write_byte (0x00);
        PORTD.0 = 1;
        PORTD.0 = 0;
        Spi_write_2bytes (PD);
        PORTD.0 = 1; }

// ________________________ Define Erase_message Function ________________________ /
void Erase_message (unsigned long message_start, unsigned long message_stop) {
        PORTD.0 = 0;
        Spi_write_2bytes (PU);
        PORTD.0 = 1;
        PORTD.0 = 0;
        Spi_write_2bytes (CLR_INT);
        PORTD.0 = 1;
        PORTD.0 = 0;
        Spi_write_2bytes (SET_ERASE);
        Spi_write_2bytes (message_start);
        Spi_write_2bytes (message_stop);
        Spi_write_byte (0x00);
        PORTD.0 = 1;
        PORTD.0 = 0;
        Spi_write_2bytes (PD);
        PORTD.0 = 1; }

// ________________________________ main program ________________________________ /
void main () {
        T0CON = 0x88;
        INTCON = 0xA0;
        TRISB = 0x07;
        TRISD = 0x00;
        INTEDG0 = INTEDG1 = INTEDG2 = 0;
        INT0IE = INT1IE = INT2IE = 1;
        Setup_adc(0);
        char i, hour1, minute1, ARHG, array[16], motion, data, data0, data1, score,
                messagenum, exercise, messageaddr, messageaddr1, messageaddr2, next_mess;
        uns8 set_sensitivity_mode;
        unsigned long average, threshold, total;
        bit low, medium, high, set_sensitivity;
        bit TimeF, baseline, mincheck, exercise1f, exercise2f, stop_save;
        bit play1, play2, play3, play4, play5, play6, play7, play8, messageplay;
        uns16 replay_count, exercisecounter;
        total = average = threshold = button_delay = 0;
        motion = baseline = mincheck = usb = TimeF = 0;
        data = data0 = data1 = counter = next_mess = hour1 = minute1 = exercise = scroll1 = 0;
        low = medium = high = set_sensitivity = 0;
        bit clock, main_menu, set_clock,
                set_play_times, edit_messages, message, view_data, message_edit_timer,
                play_time1, play_time1_clock, play_time1_message_select, play_time1_pm,
                play_time2, play_time2_clock, play_time2_message_select, play_time2_pm,
                play_time3, play_time3_clock, play_time3_message_select, play_time3_pm,
                play_time4, play_time4_clock, play_time4_message_select, play_time4_pm,
                play_time5, play_time5_clock, play_time5_message_select, play_time5_pm,
                play_time6, play_time6_clock, play_time6_message_select, play_time6_pm,
                play_time7, play_time7_clock, play_time7_message_select, play_time7_pm,
                play_time8, play_time8_clock, play_time8_message_select, play_time8_pm,
                start_edit, slot1, slot2, slot3, slot4, slot5, slot6, slot7, slot8;
        char main_menu_mode, set_play_times_mode,
                play_time1_mode, play_time1_hour, play_time1_minute,
play_time1_message_selection,
                play_time2_mode, play_time2_hour, play_time2_minute,
play_time2_message_selection,
                    play_time3_mode, play_time3_hour, play_time3_minute,
play_time3_message_selection,
                    play_time4_mode, play_time4_hour, play_time4_minute,
play_time4_message_selection,
                    play_time5_mode, play_time5_hour, play_time5_minute,
play_time5_message_selection,
                    play_time6_mode, play_time6_hour, play_time6_minute,
play_time6_message_selection,
                    play_time7_mode, play_time7_hour, play_time7_minute,
play_time7_message_selection,
                    play_time8_mode, play_time8_hour, play_time8_minute,
play_time8_message_selection,
                    edit_messages_mode, message_mode,
                    view_data_mode, dummy_delay, start_byte1, start_byte2, end_byte1, end_byte2;
          unsigned long time, time_pm_copy, edit_message_start, edit_message_stop;
          edit_message_start = edit_message_stop = 0;
          scroll = select = main_menu = set_clock = pm =
          set_play_times = edit_messages = message = message_edit_timer = 0;
          view_data  = play_time1 = play_time2 = play_time3 = play_time4 = play_time5 = play_time6 =
play_time7 = play_time8 =
          play_time1_clock = play_time1_message_select =  play_time2_clock = 0;
          play_time2_message_select = play_time3_clock =
          play_time3_message_select = play_time4_clock =
          play_time4_message_select = play_time5_clock =
          play_time5_message_select = play_time6_clock =
          play_time6_message_select = play_time7_clock =
          play_time7_message_select = play_time8_clock =
          play_time8_message_select = 0;
          clock = slot1 = slot2 = slot3 = slot4 = slot5 = slot6 = slot7 = slot8 = 1;
          main_menu_mode = set_play_times_mode = set_play_times_mode =
          edit_messages_mode = play_time1_mode = messagenum = 0;
          play1 = play2 = play3 = play4 = play5 = play6 = play7 = play8 = exercise1f = exercise2f = 0;
          replay_count = exercisecounter = 0;
          replay = messageplay = relay_flag = relay_flag1 = 0;
          set_sensitivity_mode = 0;
///Set Clock///
          hour = 12;
          minute = 4;
          second = 30;
///////////////
          messageaddr = messageaddr1 = messageaddr2 = 0x00;
          play_time1_minute = play_time1_message_selection =
          play_time2_mode = play_time2_minute = play_time2_message_selection = 0;
          play_time3_mode = play_time3_minute = play_time3_message_selection = 0;
          play_time4_mode = play_time4_minute = play_time4_message_selection = 0;
          play_time5_mode = play_time5_minute = play_time5_message_selection = 0;
          play_time6_mode = play_time6_minute = play_time6_message_selection = 0;
          play_time7_mode = play_time7_minute = play_time7_message_selection = 0;
```

```
        play_time8_mode = play_time8_minute = play_time8_message_selection = 0;
        message_mode = message_timer_count_short = message_timer_count_long = 0;
        view_data_mode = 0;
        play_time1_hour = play_time2_hour = play_time3_hour = play_time4_hour =
        play_time5_hour = play_time6_hour = play_time7_hour = play_time8_hour = 12;
        time = 0;
        no_play = 1;
        stop_save = 0;
        Setup_USART();
        Backlight(0);
        Setup_spi();
        Delay_sec (2);
        Clear_screen();
        Set_position(0);
        Print_line(" URI BME 2011", 13);
        Set_position(64);
        Print_line(" AAVID Project", 14);
        Delay_sec (2);
        PORTB.3 = 0;

// ____________________ Return to last mem. loc. if turn off___________________ /
//When the device powers on after being turned off, we want to make sure we return
//to the correct spot in memory so as not to lose any data that may be already
//saved into the EEPROM. To do this, we read in the values stored into memory
//from 3 different locations; 0x00, 0xFE, and 0xFF. If all three values are 0xFF,
//then we know there is nothing saved to memory, and we make the pointer value
//(mem loc 0xFF) equal to 0x00. If mem loc 0xFF equals 0xFF and memory locations
//0x00 and 0xFE do not (i.e. there is data stored there) then we can assume that
//the memory is full and has not been erased. We notify the user of this and do
//save anything to memory until the user has erased memory. Lastly, it could just
//be that the device turned off. If this is the case, we return to the mem loc that
//the pointer is pointing to.
        SpecifyEEPROM(0xFF);
        messageaddr = readEEPROM();
        Delay_ms(100);
        SpecifyEEPROM(0x00);
        messageaddr1 = readEEPROM();
        Delay_ms(100);
        SpecifyEEPROM(0xFE);
        messageaddr2 = readEEPROM();
        if (messageaddr == 0xFF && messageaddr1 == 0xFF && messageaddr2 == 0xFF){
                SpecifyEEPROM(0xFF);
                writeEEPROM(0x00);
                SpecifyEEPROM(0x00);
                messageaddr = 0x00;}
        else if ((messageaddr == 0xFF) && (messageaddr1 != 0xFF) && (messageaddr2 != 0xFF))
stop_save = 1;
        else SpecifyEEPROM(messageaddr);
```

```
// ____________________________ Start of While Loop ___________________________ /
        while(1) {
                PORTD.3 = 0;     //PORTD.2 and PORTD.3 are for Digital Switch: This allows
                PORTD.2 = 1;     //for usb upload as both LCD and usb use same line from PIC
                if (messageplay == 1) replay_count ++;

// ___________________ Turn off PORTB when no message play ____________________ /
        if (relay_flag == 1 && relay_flag1 == 0) {
                if (relay_time > 20) {
                        PORTB.3 = 0;
                        relay_flag = relay_flag1 = 0;}}
        if (relay_flag == 1 && relay_flag1 == 1) {
                if (relay_time > 120) {
                        PORTB.3 = 0;
                        relay_flag = relay_flag1 = 0;}}
//      if (relay_flag == 1){
//              relay_time = 0;
//              if (relay_time > 20){PORTB.3 = relay_flag = 0;}}
//      if (relay_flag1 == 1){
//              PORTB.3 = 1;
//              relay_time1 = 0;
//              if (relay_time1 > 120){PORTB.3 = relay_flag1 = 0;}}

// __________________ Stop saving to memory when reach 0xFF ___________________ /
//If data storage ever reaches memory location 0xFF, we want to stop it from
//saving anything else to memory as it will wrap around to 0x00 and overwrite
//any previous memory. Bit stop_save has to be 0 to save anythinf to memory.
                if
((minute==1||minute==6||minute==11||minute==16||minute==21||minute==26||minute==31||min
ute==36||minute==41||
                        minute==46||minute==51||minute==56)&&second==30){
                        readEEPROM();
                        messageaddr = EEADR;
                        if (messageaddr == 0xFF) stop_save = 1;}

// ____________________________ Setup for Scoring ______________________________ /
//Motion detection algorithm: Every few seconds, a 16 point array will be populated
//with 16 consecutive data points from the accelerometer. These points are then
//averaged into the value 'average'. We then compare all incoming data to this
//average and compare it to a threshold. If data is 3 points higher than threshold,
//then we increase motion counter. This algorithm takes drift of accelerometer into
//account as average is constantly recalculated around the current acc. data
                if(counter == 0){
                        total = 0;
                        for(i=0; i<16; i++) array[i] = ad_input;
                        for(i=0; i<16; i++) total += array[i];
                        average = total/16;
```

```
                    counter = 250; }
            data0 = ad_input;
            threshold = average + 3;
            if(data0 < threshold)baseline = 0;
            if((data0 > threshold) && baseline == 0){
                    baseline = 1;
                    motion++;
                    Delay_ms(70); }

    if((minute==1||minute==6||minute==11||minute==16||minute==21||minute==26||minute==
31||minute==36||minute==41||
                    minute==46||minute==51||minute==56)&&TimeF==1)TimeF = exercise2f = 0;

// _____________________________ Score/Save Motion ______________________________ /
//Save to memory every minutes: we had to choose five minute resolution for mem.
//space reasons. Every time we save data to memory, we need to save the score,
//hour, and minute for data analysis at the end of the day. The EEPROM of the
//PIC has 256 bytes of memory. Using a full byte of memory for score, hour, and
//minute would only allow for 7 hours of storage. To get around this, we created
//an algorithm that only uses half of a byte for each, or one and a half bytes
//per storage which doubles the memory storage capability.

    if((minute==0||minute==5||minute==10||minute==15||minute==20||minute==25||minute==
30||minute==35||minute==40||
                    minute==45||minute==50||minute==55)&& TimeF==0 && stop_save == 0){
                    TimeF = 1;          //Flag used to make sure we only save data once
                    counter = 0;
                    Delay_ms(70);
//The following are different sensitivity settings for people with more motion
                    if (high == 0 && medium == 0 && low == 0) high = 1;
                    if (high == 1){
                            if(motion >= 0 && motion <= 30) motion = 1;
                            if(motion > 30 && motion <= 60) motion = 2;
                            if(motion > 60 && motion <= 90) motion = 3;
                            if(motion > 90 && motion <= 120) motion = 4;
                            if(motion > 120 && motion <= 150) motion = 5;
                            if(motion > 150 && motion <= 180) motion = 6;
                            if(motion > 180 && motion <= 210) motion = 7;
                            if(motion > 210 && motion <= 240) motion = 8;
                            if(motion > 240 && motion <= 270) motion = 9;
                            if(motion > 270) motion = 10;
                            score = motion;}
                    if (medium == 1){
                            if(motion >= 0 && motion <= 45) motion = 1;
                            if(motion > 45 && motion <= 90) motion = 2;
                            if(motion > 90 && motion <= 135) motion = 3;
                            if(motion > 135 && motion <= 180) motion = 4;
                            if(motion > 180 && motion <= 225) motion = 5;
```

```
                                        if(motion > 225 && motion <= 270) motion = 6;
                                        if(motion > 270 && motion <= 315) motion = 7;
                                        if(motion > 315 && motion <= 360) motion = 8;
                                        if(motion > 360 && motion <= 405) motion = 9;
                                        if(motion > 405) motion = 10;
                                        score = motion;}
                                if (low == 1){
                                        if(motion >= 0 && motion <= 60) motion = 1;
                                        if(motion > 60 && motion <= 120) motion = 2;
                                        if(motion > 120 && motion <= 180) motion = 3;
                                        if(motion > 180 && motion <= 240) motion = 4;
                                        if(motion > 240 && motion <= 300) motion = 5;
                                        if(motion > 300 && motion <= 360) motion = 6;
                                        if(motion > 360 && motion <= 420) motion = 7;
                                        if(motion > 420 && motion <= 480) motion = 8;
                                        if(motion > 480 && motion <= 540) motion = 9;
                                        if(motion > 540) motion = 10;
                                        score = motion;}
//The following code is used to determine what should be save to memory and
//the actual saving of that data. Explaining how it works can be complicated
//so the variables are named to to represent their function. Going through
//the code and writing a flow chart is the best way to understand what is
//being done. There is a lot of manipulation of memory and data to break the
//bytes in memory in half. Also, it should be noted that the scores and times
//are stored in decimal form. This allowed us to debug more efficiently as we
//did not have to convert everything from hex to decimal. Last note: The mem.
//is stored as follows: score, hour, minute. to find the minute, multiply
//whatever is stored in memory by five.
//Example from memory: 1C 34 C4. This would mean that their was a score of 1
//at 12 (C) fifteen (3*5) and a score of 4 at 12 (C) 20 (4*5)
        readEEPROM();
        if(EEDATA == 255){
                                        mincheck = 0;
                                        hour1 = hour; minute1 = minute;
                                        if(score == 1) score = 16;
                                        if(score == 2) score = 32;
                                        if(score == 3) score = 48;
                                        if(score == 4) score = 64;
                                        if(score == 5) score = 80;
                                        if(score == 6) score = 96;
                                        if(score == 7) score = 112;
                                        if(score == 8) score = 128;
                                        if(score == 9) score = 144;
                                        if(score == 10) score = 160;
                                        data = score + hour1;
                                        if(score == 16) exercise++; }
                                if(EEDATA != 255){
                                        mincheck = 1;
        hour1 = hour; minute1 = minute;
        readEEPROM();
        data = (EEDATA - 15) + score;
        if(score == 1) exercise++; }
writeEEPROM(data);
Delay_ms(100);
INCEEPROM();
Delay_ms(100);
readEEPROM();
if(EEDATA == 255 && mincheck == 0){
        hour1 = hour; minute1 = minute;
        if(minute1 == 0) minute1 = 15;
        if(minute1 == 5) minute1 = 31;
        if(minute1 == 10) minute1 = 47;
        if(minute1 == 15) minute1 = 63;
        if(minute1 == 20) minute1 = 79;
        if(minute1 == 25) minute1 = 95;
        if(minute1 == 30) minute1 = 111;
        if(minute1 == 35) minute1 = 127;
        if(minute1 == 40) minute1 = 143;
        if(minute1 == 45) minute1 = 159;
        if(minute1 == 50) minute1 = 175;
        if(minute1 == 55) minute1 = 191;
        data = minute1; }
if(EEDATA == 255 && mincheck == 1){
        hour1 = hour; minute1 = minute;
        if(hour1 == 1) hour1 = 16;
        if(hour1 == 2) hour1 = 32;
        if(hour1 == 3) hour1 = 48;
        if(hour1 == 4) hour1 = 64;
        if(hour1 == 5) hour1 = 80;
        if(hour1 == 6) hour1 = 96;
        if(hour1 == 7) hour1 = 112;
        if(hour1 == 8) hour1 = 128;
        if(hour1 == 9) hour1 = 144;
        if(hour1 == 10) hour1 = 160;
        if(hour1 == 11) hour1 = 176;
        if(hour1 == 12) hour1 = 192;
        if(minute1 == 0) minute1 = 0;
        if(minute1 == 5) minute1 = 1;
        if(minute1 == 10) minute1 = 2;
        if(minute1 == 15) minute1 = 3;
        if(minute1 == 20) minute1 = 4;
        if(minute1 == 25) minute1 = 5;
        if(minute1 == 30) minute1 = 6;
        if(minute1 == 35) minute1 = 7;
        if(minute1 == 40) minute1 = 8;
        if(minute1 == 45) minute1 = 9;
```

```
                    if(minute1 == 50) minute1 = 10;
                    if(minute1 == 55) minute1 = 11;
                    data1 = hour1 + minute1; }
                if(mincheck == 0)writeEEPROM(data);
                if(mincheck == 1){
                    writeEEPROM(data1);
                    Delay_ms(100);
                    INCEEPROM(); }
                messageaddr = EEADR;
                SpecifyEEPROM(0xFF);
                Delay_ms(100);
                writeEEPROM(messageaddr);
                Delay_ms(100);
                SpecifyEEPROM(messageaddr);
                Delay_ms(100);
                motion = 0; }

//_________________________Play message if no motion/1 hr________________________ /
//Everytime the PIC stores a score of 1 to memory, the exercise counter is incremented.
//If ever exercise is greater than 11 (or one hour of no motion), then message 8
//will be played as a reminder to exercise. If played, the message number will be
//saved to memory. To differentiate motion data from message data in the EEPROM, an
//FF will always preceed the message in memory. For Ex: 11 01 11 FF 83 12 FF ...
//This would say score of 1 @ 1:00, 1 @ 1:05, then message 8 was played sometime
//@ 1:05, and the motion then went up to 3 @ 1:10. If, however, their is exercise
//within the hour (i.e. score > 1), the exercise counter will be reset to 0.
//*Note: their are several flags in this code (exercise1f & exercise2f) to make sure
//the message is only played once. For more information on how this done, you will
//need to trace them through the code.
            if (exercise > 11){
                exercise1f = 1;
                exercisecounter++;}
            if (exercisecounter == 1) exercise2f = 1;
            if (exercisecounter != 1) exercise2f = 0;
            if (exercise > 11 && exercise1f == 1 && exercise2f == 1 && stop_save == 0 &&
(minute==0||minute==5||minute==10||minute==15||

        minute==20||minute==25||minute==30||minute==35||minute==40||minute==45||minute==
50||minute==55)){
                Delay_ms(100);
                exercise2f = 0;
                Play_message (message8_start, message8_stop);
                messageaddr = EEADR;
                SpecifyEEPROM(0xFF);
                Delay_ms(100);
                writeEEPROM(messageaddr);
                Delay_ms(100);
                SpecifyEEPROM(messageaddr);
```

```
                    readEEPROM();
                    if(EEDATA == 255){
                            INCEEPROM();
                            data = 143;
                            Delay_ms(100);
                            writeEEPROM(data);}
                    if(EEDATA != 255){
                            INCEEPROM();
                            data = 248;
                            writeEEPROM(data);
                            Delay_ms(100);
                            INCEEPROM();}}

//______________________________Reset Exercise Counter to Zero_____________________________ /
            if (score > 1) exercise = exercise1f = exercisecounter = 0;

// ________________________________ if (clock) sequence __________________________________ /
            if (clock) {
    if (update_display) {
          Clear_screen();
          Set_position(0);
          Print_line("  Clock Mode",12);
          Set_position(64);
          Print_line("    :  :   m",12);
          Print_2dig_num(second,72);
          Print_2dig_num(minute,69);
          Print_2dig_num(hour,66);
          Set_position(74);
          if (pm) Print_line ("p",1);
          else Print_line("a",1);
          update_display = 0; }
                    if (scroll) {
                            scroll = 0;
                            clock = 0;
                            main_menu = 1;
                            update_display = 1; }
                    if (select) {
                      select = 0;
                      clock = 0;
                      main_menu = 1;
                      update_display = 1;}}

// _____________________________ if (main_menu) sequence _____________________________ /
            if (main_menu) {
                    if (update_display) {
          Clear_screen();
          Set_position(0);
          Print_line("Main Menu", 9);
          Set_position(64);
          switch (main_menu_mode) {
          case 0:
                    Print_line("Exit", 4);
                    break;
          case 1:
                    Print_line("Set Clock", 9);
                    break;
          case 2:
                    Print_line("Edit Play Times", 14);
                    break;
          case 3:
                    Print_line("Edit Messages", 13);
                    break;
          case 4:
                    Print_line("USB Data Upload", 15);
                    break;
          case 5:
                    Print_line("Erase Memory", 12);
                    break;
          case 6:
                    Print_line("Sensitivity", 11);
                    break;}
                      update_display = 0; }
                      if (scroll) {
                                scroll = 0;
                                main_menu_mode++;
                                if (main_menu_mode == 7) main_menu_mode = 0;
                                update_display = 1; }
                      if (select) {
          select = 0;
          main_menu = 0;
switch (main_menu_mode) {
case 0:
          clock = 1;
          break;
          case 1:
          set_clock = 1;
          break;
   case 2:
          set_play_times = 1;
          break;
case 3:
          edit_messages = 1;
          break;
case 4:
                                        Clear_screen();
                    Set_position(0);
```

```
                                        Print_line("Sending Data...", 15);
                                        while(EEADR != 0xFF){
                                                PORTD.3 = 1;
                                                PORTD.2 = 0;
                                                Delay_ms(10);
                                                ARHG = readEEPROM();
                                                Transmit(ARHG);
                                                INCEEPROM(); }
                                        Delay_ms(10);
                                        SpecifyEEPROM(0x00);
                                        usb = 0;
                 main_menu = 1;
                 break;
                                case 5:
                                        Clear_screen();
                        Set_position(0);
                                        Print_line("Erasing...", 10);
                                        SpecifyEEPROM(0x00);
                                        while(EEADR != 0xFF){
                                                writeEEPROM(0xFF);
                                                INCEEPROM();}
                                        SpecifyEEPROM(0xFF);
                                        writeEEPROM(0x00);
                                        SpecifyEEPROM(0x00);
                                        messageaddr = 0x00;
                                        stop_save = 0;
                 main_menu = 1;
                                  break;
                                case 6:
                                        set_sensitivity = 1;
                                  break;}
                main_menu_mode = 0;
                update_display = 1;}}

// ___________________________ if (set_clock) sequence ___________________________ /
                if (set_clock) {
                        Set_time();
                        time_pm_copy = time_return >> 12;
                        if (time_pm_copy == 1) pm = 1;
                        else pm = 0;
                        hour = time_return >> 8;
                        hour = hour & 0x0F;
                        minute = time_return;
                        minute = time_return & 0x3F;
                        second = 0;
                        set_clock = 0;
                        main_menu = 1;
                        update_display = 1;}
```

```
// ______________________ if (set_play_times) sequence _____________________________ /
                    if (set_play_times) {
          if (update_display) {
                  Clear_screen ();
                  Set_position (0);
                  switch (set_play_times_mode) {
                  case 0:
                            Print_line ("Edit Play Times", 15);
                            Set_position (64);
                            Print_line ("Exit", 4);
                            break;
          case 1:
                                                Print_line ("Edit Play Time 1", 16);
                                                Set_position (64);
                                                Print_line ("  :   m",7);
                                                Print_2dig_num (play_time1_minute,67);
                                                Print_2dig_num (play_time1_hour,64);
                                                Set_position (69);
                                                if (play_time1_pm == 1) Print_line ("p",1);
                                                else Print_line("a",1);
                                                Set_position (74);
                                                Print_line ("Msg", 3);
                                                Print_2dig_num (play_time1_message_selection, 78);
                                                break;
                                        case 2:
                                                Print_line ("Edit Play Time 2", 16);
                                                Set_position (64);
                                                Print_line ("  :   m",7);
                                                Print_2dig_num (play_time2_minute,67);
                                                Print_2dig_num (play_time2_hour,64);
                                                Set_position (69);
                                                if (play_time2_pm == 1) Print_line ("p",1);
                                                else Print_line("a",1);
                                                Set_position (74);
                                                Print_line ("Msg", 3);
                                                Print_2dig_num (play_time2_message_selection, 78);
                                                break;
                                        case 3:
                                                Print_line ("Edit Play Time 3", 16);
                                                Set_position (64);
                                                Print_line ("  :   m",7);
                                                Print_2dig_num (play_time3_minute,67);
                                                Print_2dig_num (play_time3_hour,64);
                                                Set_position (69);
                                                if (play_time3_pm == 1) Print_line ("p",1);
                                                else Print_line("a",1);
                                                Set_position (74);
```

```
            Print_line ("Msg", 3);
            Print_2dig_num (play_time3_message_selection, 78);
            break;
    case 4:
            Print_line ("Edit Play Time 4", 16);
            Set_position (64);
            Print_line ("  :   m",7);
            Print_2dig_num (play_time4_minute,67);
            Print_2dig_num (play_time4_hour,64);
            Set_position (69);
            if (play_time4_pm == 1) Print_line ("p",1);
            else Print_line("a",1);
            Set_position (74);
            Print_line ("Msg", 3);
            Print_2dig_num (play_time4_message_selection, 78);
            break;
    case 5:
            Print_line ("Edit Play Time 5", 16);
            Set_position (64);
            Print_line ("  :   m",7);
            Print_2dig_num (play_time5_minute,67);
            Print_2dig_num (play_time5_hour,64);
            Set_position (69);
            if (play_time5_pm == 1) Print_line ("p",1);
            else Print_line("a",1);
            Set_position (74);
            Print_line ("Msg", 3);
            Print_2dig_num (play_time5_message_selection, 78);
            break;
    case 6:
            Print_line ("Edit Play Time 6", 16);
            Set_position (64);
            Print_line ("  :   m",7);
            Print_2dig_num (play_time6_minute,67);
            Print_2dig_num (play_time6_hour,64);
            Set_position (69);
            if (play_time6_pm == 1) Print_line ("p",1);
            else Print_line("a",1);
            Set_position (74);
            Print_line ("Msg", 3);
            Print_2dig_num (play_time6_message_selection, 78);
            break;
    case 7:
            Print_line ("Edit Play Time 7", 16);
            Set_position (64);
            Print_line ("  :   m",7);
            Print_2dig_num (play_time7_minute,67);
            Print_2dig_num (play_time7_hour,64);
```

```
                    Set_position (69);
                    if (play_time7_pm == 1) Print_line ("p",1);
                    else Print_line("a",1);
                    Set_position (74);
                    Print_line ("Msg", 3);
                    Print_2dig_num (play_time7_message_selection, 78);
                    break;
            case 8:
                    Print_line ("Edit Play Time 8", 16);
                    Set_position (64);
                    Print_line ("  :   m",7);
                    Print_2dig_num (play_time8_minute,67);
                    Print_2dig_num (play_time8_hour,64);
                    Set_position (69);
                    if (play_time8_pm == 1) Print_line ("p",1);
                    else Print_line("a",1);
                    Set_position (74);
                    Print_line ("Msg", 3);
                    Print_2dig_num (play_time8_message_selection, 78);
                    break; }
        update_display = 0; }
        if (scroll) {
            scroll = 0;
            set_play_times_mode++;
            if (set_play_times_mode == 9) set_play_times_mode = 0;
            update_display = 1; }
        if (select) {
            select = 0;
            set_play_times = 0;
            switch (set_play_times_mode) {
            case 0:
main_menu = 1;
break;
            case 1:
                    play_time1 = 1;
                    break;
            case 2:
                    play_time2 = 1;
                    break;
            case 3:
                    play_time3 = 1;
                    break;
            case 4:
                    play_time4 = 1;
                    break;
            case 5:
                    play_time5 = 1;
                    break;
```

```
                        case 6:
                                play_time6 = 1;
                                break;
                        case 7:
                                play_time7 = 1;
                                break;
                        case 8:
                                play_time8 = 1;
                                break;}
                set_play_times_mode = 0;
                update_display = 1;}}

// _________________________ if (play_time1) sequence _____________________________ /
                if (play_time1) {
                        if (update_display) {
               Clear_screen ();
               Set_position (0);
               Print_line ("Edit Play Time 1", 16);
               Set_position (64);
               if (play_time1_mode == 0 ) Print_line ("Exit", 4);
               else if (play_time1_mode == 1) Print_line ("Edit Time", 9);
               else Print_line ("Select Message", 14);
               update_display = 0;}
                        if (scroll) {
                                scroll = 0;
                                play_time1_mode++;
                                if (play_time1_mode == 3) play_time1_mode = 0;
                                update_display = 1; }
                        if (select) {
                                select = 0;
                                play_time1 = 0;
                                if (play_time1_mode == 0) set_play_times = 1;
                                else if (play_time1_mode == 1) play_time1_clock = 1;
                                else play_time1_message_select = 1;
                                play_time1_mode = 0;
                                update_display = 1;}}

// _________________________ if (play_time1_clock) sequence _______________________ /
                if (play_time1_clock) {
                        Set_time();
                        time_pm_copy = time_return >> 12;
                        if (time_pm_copy == 1) play_time1_pm = 1;
                        else play_time1_pm = 0;
                        play_time1_hour = time_return >> 8;
                        play_time1_hour = play_time1_hour & 0x0F;
                        play_time1_minute = time_return;
                        play_time1_minute = time_return & 0x3F;
                        play_time1_clock = 0;
                        play_time1 = 1;
                        update_display = 1; }

        // ____________________ if (play_time1_message_select) sequence _________________ /
                if (play_time1_message_select) {
                        play_time1_message_selection = Message_select();
                        play_time1_message_select = 0;
                        play_time1 = 1; }

        // _________________________ if (play_time2) sequence ____________________________ /
                if (play_time2) {
                        if (update_display) {
                                Clear_screen ();
                                Set_position (0);
                                Print_line ("Edit Play Time 2", 16);
                                Set_position (64);
                                if (play_time2_mode == 0 ) Print_line ("Exit", 4);
                                else if (play_time2_mode == 1) Print_line ("Edit Time", 9);
                                else Print_line ("Select Message", 14);
                                update_display = 0; }
                        if (scroll) {
                                scroll = 0;
                                play_time2_mode++;
                                if (play_time2_mode == 3) play_time2_mode = 0;
                                update_display = 1; }
                        if (select) {
                                select = 0;
                                play_time2 = 0;
                                if (play_time2_mode == 0) set_play_times = 1;
                                else if (play_time2_mode == 1) play_time2_clock = 1;
                                else play_time2_message_select = 1;
                                play_time2_mode = 0;
                                update_display = 1;}}

        // ________________________ if (play_time2_clock) sequence ______________________ /
                if (play_time2_clock) {
                        Set_time();
                        time_pm_copy = time_return >> 12;
                        if (time_pm_copy == 1) play_time2_pm = 1;
                        else play_time2_pm = 0;
                        play_time2_hour = time_return >> 8;
                        play_time2_hour = play_time2_hour & 0x0F;
                        play_time2_minute = time_return;
                        play_time2_minute = time_return & 0x3F;
                        play_time2_clock = 0;
                        play_time2 = 1;
                        update_display = 1; }
```

```
// ____________________ if (play_time2_message_select) sequence _________________ /
		if (play_time2_message_select) {
			play_time2_message_selection = Message_select();
			play_time2_message_select = 0;
			play_time2 = 1; }

// _________________________ if (play_time3) sequence ___________________________ /
		if (play_time3) {
			if (update_display) {
				Clear_screen ();
				Set_position (0);
				Print_line ("Edit Play Time 3", 16);
				Set_position (64);
				if (play_time3_mode == 0 ) Print_line ("Exit", 4);
				else if (play_time3_mode == 1) Print_line ("Edit Time", 9);
				else Print_line ("Select Message", 14);
				update_display = 0; }
			if (scroll) {
				scroll = 0;
				play_time3_mode++;
				if (play_time3_mode == 3) play_time3_mode = 0;
				update_display = 1; }
			if (select) {
				select = 0;
				play_time3 = 0;
				if (play_time3_mode == 0) set_play_times = 1;
				else if (play_time3_mode == 1) play_time3_clock = 1;
				else play_time3_message_select = 1;
				play_time3_mode = 0;
				update_display = 1;}}

// ________________________ if (play_time3_clock) sequence ______________________ /
		if (play_time3_clock) {
			Set_time();
			time_pm_copy = time_return >> 12;
			if (time_pm_copy == 1) play_time3_pm = 1;
			else play_time3_pm = 0;
			play_time3_hour = time_return >> 8;
			play_time3_hour = play_time3_hour & 0x0F;
			play_time3_minute = time_return;
			play_time3_minute = time_return & 0x3F;
			play_time3_clock = 0;
			play_time3 = 1;
			update_display = 1; }

// ____________________ if (play_time3_message_select) sequence _________________ /
		if (play_time3_message_select) {
			play_time3_message_selection = Message_select();
```

```
                        play_time3_message_select = 0;
                        play_time3 = 1; }

// ___________________________ if (play_time4) sequence ___________________________ /
                if (play_time4) {
                        if (update_display) {
                                Clear_screen ();
                                Set_position (0);
                                Print_line ("Edit Play Time 4", 16);
                                Set_position (64);
                                if (play_time4_mode == 0 ) Print_line ("Exit", 4);
                                else if (play_time4_mode == 1) Print_line ("Edit Time", 9);
                                else Print_line ("Select Message", 14);
                                update_display = 0; }
                        if (scroll) {
                                scroll = 0;
                                play_time4_mode++;
                                if (play_time4_mode == 3) play_time4_mode = 0;
                                update_display = 1; }
                        if (select) {
                                select = 0;
                                play_time4 = 0;
                                if (play_time4_mode == 0) set_play_times = 1;
                                else if (play_time4_mode == 1) play_time4_clock = 1;
                                else play_time4_message_select = 1;
                                play_time4_mode = 0;
                                update_display = 1;}}

// _________________________ if (play_time4_clock) sequence _______________________ /
                if (play_time4_clock) {
                        Set_time();
                        time_pm_copy = time_return >> 12;
                        if (time_pm_copy == 1) play_time4_pm = 1;
                        else play_time4_pm = 0;
                        play_time4_hour = time_return >> 8;
                        play_time4_hour = play_time4_hour & 0x0F;
                        play_time4_minute = time_return;
                        play_time4_minute = time_return & 0x3F;
                        play_time4_clock = 0;
                        play_time4 = 1;
                        update_display = 1; }

// _____________________ if (play_time4_message_select) sequence __________________ /
                if (play_time4_message_select) {
                        play_time4_message_selection = Message_select();
                        play_time4_message_select = 0;
                        play_time4 = 1; }
```

```
// ________________________ if (play_time5) sequence ____________________________ /
            if (play_time5) {
                  if (update_display) {
                        Clear_screen ();
                        Set_position (0);
                        Print_line ("Edit Play Time 5", 16);
                        Set_position (64);
                        if (play_time5_mode == 0 ) Print_line ("Exit", 4);
                        else if (play_time5_mode == 1) Print_line ("Edit Time", 9);
                        else Print_line ("Select Message", 14);
                        update_display = 0; }
                  if (scroll) {
                        scroll = 0;
                        play_time5_mode++;
                        if (play_time5_mode == 3) play_time5_mode = 0;
                        update_display = 1; }
                  if (select) {
                        select = 0;
                        play_time5 = 0;
                        if (play_time5_mode == 0) set_play_times = 1;
                        else if (play_time5_mode == 1) play_time5_clock = 1;
                        else play_time5_message_select = 1;
                        play_time5_mode = 0;
                        update_display = 1;}}

// ________________________ if (play_time5_clock) sequence ______________________ /
            if (play_time5_clock) {
                  Set_time();
                  time_pm_copy = time_return >> 12;
                  if (time_pm_copy == 1) play_time5_pm = 1;
                  else play_time5_pm = 0;
                  play_time5_hour = time_return >> 8;
                  play_time5_hour = play_time5_hour & 0x0F;
                  play_time5_minute = time_return;
                  play_time5_minute = time_return & 0x3F;
                  play_time5_clock = 0;
                  play_time5 = 1;
                  update_display = 1; }

// ____________________ if (play_time5_message_select) sequence _________________ /
            if (play_time5_message_select) {
                  play_time5_message_selection = Message_select();
                  play_time5_message_select = 0;
                  play_time5 = 1; }

// ________________________ if (play_time6) sequence ____________________________ /
            if (play_time6) {
                  if (update_display) {
                        Clear_screen ();
                        Set_position (0);
                        Print_line ("Edit Play Time 6", 16);
                        Set_position (64);
                        if (play_time6_mode == 0 ) Print_line ("Exit", 4);
                        else if (play_time6_mode == 1) Print_line ("Edit Time", 9);
                        else Print_line ("Select Message", 14);
                        update_display = 0; }
                if (scroll) {
                        scroll = 0;
                        play_time6_mode++;
                        if (play_time6_mode == 3) play_time6_mode = 0;
                        update_display = 1; }
                if (select) {
                        select = 0;
                        play_time6 = 0;
                        if (play_time6_mode == 0) set_play_times = 1;
                        else if (play_time6_mode == 1) play_time6_clock = 1;
                        else play_time6_message_select = 1;
                        play_time6_mode = 0;
                        update_display = 1;}}

// _________________________ if (play_time6_clock) sequence _______________________ /
        if (play_time6_clock) {
                Set_time();
                time_pm_copy = time_return >> 12;
                if (time_pm_copy == 1) play_time6_pm = 1;
                else play_time6_pm = 0;
                play_time6_hour = time_return >> 8;
                play_time6_hour = play_time6_hour & 0x0F;
                play_time6_minute = time_return;
                play_time6_minute = time_return & 0x3F;
                play_time6_clock = 0;
                play_time6 = 1;
                update_display = 1; }

// _____________________ if (play_time6_message_select) sequence _________________ /
        if (play_time6_message_select) {
                play_time6_message_selection = Message_select();
                play_time6_message_select = 0;
                play_time6 = 1; }

// _________________________ if (play_time7) sequence _____________________________ /
        if (play_time7) {
                if (update_display) {
                        Clear_screen ();
                        Set_position (0);
                        Print_line ("Edit Play Time 7", 16);
```

```
                                        Set_position (64);
                                        if (play_time7_mode == 0 ) Print_line ("Exit", 4);
                                        else if (play_time7_mode == 1) Print_line ("Edit Time", 9);
                                        else Print_line ("Select Message", 14);
                                        update_display = 0; }
                                if (scroll) {
                                        scroll = 0;
                                        play_time7_mode++;
                                        if (play_time7_mode == 3) play_time7_mode = 0;
                                        update_display = 1; }
                                if (select) {
                                        select = 0;
                                        play_time7 = 0;
                                        if (play_time7_mode == 0) set_play_times = 1;
                                        else if (play_time7_mode == 1) play_time7_clock = 1;
                                        else play_time7_message_select = 1;
                                        play_time7_mode = 0;
                                        update_display = 1;}}

// ________________________ if (play_time7_clock) sequence ______________________ /
                        if (play_time7_clock) {
                                Set_time();
                                time_pm_copy = time_return >> 12;
                                if (time_pm_copy == 1) play_time7_pm = 1;
                                else play_time7_pm = 0;
                                play_time7_hour = time_return >> 8;
                                play_time7_hour = play_time7_hour & 0x0F;
                                play_time7_minute = time_return;
                                play_time7_minute = time_return & 0x3F;
                                play_time7_clock = 0;
                                play_time7 = 1;
                                update_display = 1; }

// ____________________ if (play_time7_message_select) sequence _________________ /
                        if (play_time7_message_select) {
                                play_time7_message_selection = Message_select();
                                play_time7_message_select = 0;
                                play_time7 = 1; }

// ________________________ if (play_time8) sequence ____________________________ /
                        if (play_time8) {
                                if (update_display) {
                                        Clear_screen ();
                                        Set_position (0);
                                        Print_line ("Edit Play Time 8", 16);
                                        Set_position (64);
                                        if (play_time8_mode == 0 ) Print_line ("Exit", 4);
                                        else if (play_time8_mode == 1) Print_line ("Edit Time", 9);
                                        else Print_line ("Select Message", 14);
                                        update_display = 0; }
                                if (scroll) {
                                        scroll = 0;
                                        play_time8_mode++;
                                        if (play_time8_mode == 3) play_time8_mode = 0;
                                        update_display = 1; }
                                if (select) {
                                        select = 0;
                                        play_time8 = 0;
                                        if (play_time8_mode == 0) set_play_times = 1;
                                        else if (play_time8_mode == 1) play_time8_clock = 1;
                                        else play_time8_message_select = 1;
                                        play_time8_mode = 0;
                                        update_display = 1;}}

// _________________________ if (play_time8_clock) sequence _______________________ /
                        if (play_time8_clock) {
                                Set_time();
                                time_pm_copy = time_return >> 12;
                                if (time_pm_copy == 1) play_time8_pm = 1;
                                else play_time8_pm = 0;
                                play_time8_hour = time_return >> 8;
                                play_time8_hour = play_time8_hour & 0x0F;
                                play_time8_minute = time_return;
                                play_time8_minute = time_return & 0x3F;
                                play_time8_clock = 0;
                                play_time8 = 1;
                                update_display = 1; }

// _____________________ if (play_time8_message_select) sequence _________________ /
                        if (play_time8_message_select) {
                                play_time8_message_selection = Message_select();
                                play_time8_message_select = 0;
                                play_time8 = 1; }

// _________________________check for play times __________________________________ /
                        if (pm == play_time1_pm && hour == play_time1_hour && minute ==
                                play_time1_minute && second == 0 && no_play == 1) {
                                no_play = 0;
                                if (play_time1_message_selection == 1){
                                        Play_message (message1_start, message1_stop);
                                        messagenum = play1 = messageplay = 1;
                                        replay_count = 0;}
                                if (play_time1_message_selection == 2){
                                        Play_message (message2_start, message2_stop);
                                        messagenum = 1; play2 = 1; messageplay = 1;
                                        replay_count = 0;}
```

```
		if (play_time1_message_selection == 3){
			Play_message (message3_start, message3_stop);
			messagenum = 1; play3 = 1; messageplay = 1;
			replay_count = 0;}
		if (play_time1_message_selection == 4){
			Play_message (message4_start, message4_stop);
			messagenum = 1; play4 = 1; messageplay = 1;
			replay_count = 0;}
		if (play_time1_message_selection == 5){
			Play_message (message5_start, message5_stop);
			messagenum = 1; play5 = 1; messageplay = 1;
			replay_count = 0;}
		if (play_time1_message_selection == 6){
			Play_message (message6_start, message6_stop);
			messagenum = 1; play6 = 1; messageplay = 1;
			replay_count = 0;}
		if (play_time1_message_selection == 7){
			Play_message (message7_start, message7_stop);
			messagenum = 1; play7 = 1; messageplay = 1;
			replay_count = 0;}
		if (play_time1_message_selection == 8){
			Play_message (message8_start, message8_stop);
			messagenum = 1; play8 = 1; messageplay = 1;
			replay_count = 0;}
		Delay_ms(100);}
	if (pm == play_time2_pm && hour == play_time2_hour && minute ==
		play_time2_minute && second == 0 && no_play == 0) {
		no_play = 1;
		if (play_time2_message_selection == 1){
			Play_message (message1_start, message1_stop);
			messagenum = 2; play1 = 1; messageplay = 1;
			replay_count = 0;}
		if (play_time2_message_selection == 2){
			Play_message (message2_start, message2_stop);
			messagenum = 2; play2 = 1; messageplay = 1;
			replay_count = 0;}
		if (play_time2_message_selection == 3){
			Play_message (message3_start, message3_stop);
			messagenum = 2; play3 = 1; messageplay = 1;
			replay_count = 0;}
		if (play_time2_message_selection == 4){
			Play_message (message4_start, message4_stop);
			messagenum = 2; play4 = 1; messageplay = 1;
			replay_count = 0;}
		if (play_time2_message_selection == 5){
			Play_message (message5_start, message5_stop);
			messagenum = 2; play5 = 1; messageplay = 1;
			replay_count = 0;}
		if (play_time2_message_selection == 6){
			Play_message (message6_start, message6_stop);
			messagenum = 2; play6 = 1; messageplay = 1;
			replay_count = 0;}
		if (play_time2_message_selection == 7){
			Play_message (message7_start, message7_stop);
			messagenum = 2; play7 = 1; messageplay = 1;
			replay_count = 0;}
		if (play_time2_message_selection == 8){
			Play_message (message8_start, message8_stop);
			messagenum = 2; play8 = 1; messageplay = 1;
			replay_count = 0;}
		Delay_ms(100);}
	if (pm == play_time3_pm && hour == play_time3_hour && minute ==
		play_time3_minute && second == 0 && no_play == 0) {
		no_play = 1;
		if (play_time3_message_selection == 1){
			Play_message (message1_start, message1_stop);
			messagenum = 3; play1 = 1; messageplay = 1;
			replay_count = 0;}
		if (play_time3_message_selection == 2){
			Play_message (message2_start, message2_stop);
			messagenum = 3; play2 = 1; messageplay = 1;
			replay_count = 0;}
		if (play_time3_message_selection == 3){
			Play_message (message3_start, message3_stop);
			messagenum = 3; play3 = 1; messageplay = 1;
			replay_count = 0;}
		if (play_time3_message_selection == 4){
			Play_message (message4_start, message4_stop);
			messagenum = 3; play4 = 1; messageplay = 1;
			replay_count = 0;}
		if (play_time3_message_selection == 5){
			Play_message (message5_start, message5_stop);
			messagenum = 3; play5 = 1; messageplay = 1;
			replay_count = 0;}
		if (play_time3_message_selection == 6){
			Play_message (message6_start, message6_stop);
			messagenum = 3; play6 = 1; messageplay = 1;
			replay_count = 0;}
		if (play_time3_message_selection == 7){
			Play_message (message7_start, message7_stop);
			messagenum = 3; play7 = 1; messageplay = 1;
			replay_count = 0;}
		if (play_time3_message_selection == 8){
			Play_message (message8_start, message8_stop);
			messagenum = 3; play8 = 1; messageplay = 1;
			replay_count = 0;}
```

```
            Delay_ms(100);}
    if (pm == play_time4_pm && hour == play_time4_hour && minute ==
            play_time4_minute && second == 0 && no_play == 0) {
            no_play = 1;
            if (play_time4_message_selection == 1){
                        Play_message (message1_start, message1_stop);
                        messagenum = 4; play1 = 1; messageplay = 1;
                        replay_count = 0;}
            if (play_time4_message_selection == 2){
                        Play_message (message2_start, message2_stop);
                        messagenum = 4; play2 = 1; messageplay = 1;
                        replay_count = 0;}
            if (play_time4_message_selection == 3){
                        Play_message (message3_start, message3_stop);
                        messagenum = 4; play3 = 1; messageplay = 1;
                        replay_count = 0;}
            if (play_time4_message_selection == 4){
                        Play_message (message4_start, message4_stop);
                        messagenum = 4; play4 = 1; messageplay = 1;
                        replay_count = 0;}
            if (play_time4_message_selection == 5){
                        Play_message (message5_start, message5_stop);
                        messagenum = 4; play5 = 1; messageplay = 1;
                        replay_count = 0;}
            if (play_time4_message_selection == 6){
                        Play_message (message6_start, message6_stop);
                        messagenum = 4; play6 = 1; messageplay = 1;
                        replay_count = 0;}
            if (play_time4_message_selection == 7){
                        Play_message (message7_start, message7_stop);
                        messagenum = 4; play7 = 1; messageplay = 1;
                        replay_count = 0;}
            if (play_time4_message_selection == 8){
                        Play_message (message8_start, message8_stop);
                        messagenum = 4; play8 = 1; messageplay = 1;
                        replay_count = 0;}
            Delay_ms(100);}
    if (pm == play_time5_pm && hour == play_time5_hour && minute ==
            play_time5_minute && second == 0 && no_play == 0) {
            no_play = 1;
            if (play_time5_message_selection == 1){
                        Play_message (message1_start, message1_stop);
                        messagenum = 5; play1 = 1; messageplay = 1;
                        replay_count = 0;}
            if (play_time5_message_selection == 2){
                        Play_message (message2_start, message2_stop);
                        messagenum = 5; play2 = 1; messageplay = 1;
                        replay_count = 0;}
```

```
		if (play_time5_message_selection == 3){
			Play_message (message3_start, message3_stop);
			messagenum = 5; play3 = 1; messageplay = 1;
			replay_count = 0;}
		if (play_time5_message_selection == 4){
			Play_message (message4_start, message4_stop);
			messagenum = 5; play4 = 1; messageplay = 1;
			replay_count = 0;}
		if (play_time5_message_selection == 5){
			Play_message (message5_start, message5_stop);
			messagenum = 5; play5 = 1; messageplay = 1;
			replay_count = 0;}
		if (play_time5_message_selection == 6){
			Play_message (message6_start, message6_stop);
			messagenum = 5; play6 = 1; messageplay = 1;
			replay_count = 0;}
		if (play_time5_message_selection == 7){
			Play_message (message7_start, message7_stop);
			messagenum = 5; play7 = 1; messageplay = 1;
			replay_count = 0;}
		if (play_time5_message_selection == 8){
			Play_message (message8_start, message8_stop);
			messagenum = 5; play8 = 1; messageplay = 1;
			replay_count = 0;}
		Delay_ms(100);}
	if (pm == play_time6_pm && hour == play_time6_hour && minute ==
		play_time6_minute && second == 0 && no_play == 0) {
		no_play = 1;
		if (play_time6_message_selection == 1){
			Play_message (message1_start, message1_stop);
			messagenum = 6; play1 = 1; messageplay = 1;
			replay_count = 0;}
		if (play_time6_message_selection == 2){
			Play_message (message2_start, message2_stop);
			messagenum = 6; play2 = 1; messageplay = 1;
			replay_count = 0;}
		if (play_time6_message_selection == 3){
			Play_message (message3_start, message3_stop);
			messagenum = 6; play3 = 1; messageplay = 1;
			replay_count = 0;}
		if (play_time6_message_selection == 4){
			Play_message (message4_start, message4_stop);
			messagenum = 6; play4 = 1; messageplay = 1;
			replay_count = 0;}
		if (play_time6_message_selection == 5){
			Play_message (message5_start, message5_stop);
			messagenum = 6; play5 = 1; messageplay = 1;
			replay_count = 0;}
```

```
        if (play_time6_message_selection == 6){
                Play_message (message6_start, message6_stop);
                messagenum = 6; play6 = 1; messageplay = 1;
                replay_count = 0;}
        if (play_time6_message_selection == 7){
                Play_message (message7_start, message7_stop);
                messagenum = 6; play7 = 1; messageplay = 1;
                replay_count = 0;}
        if (play_time6_message_selection == 8){
                Play_message (message8_start, message8_stop);
                messagenum = 6; play8 = 1; messageplay = 1;
                replay_count = 0;}
        Delay_ms(100);}
if (pm == play_time7_pm && hour == play_time7_hour && minute ==
        play_time7_minute && second == 0 && no_play == 0) {
        no_play = 1;
        if (play_time7_message_selection == 1){
                Play_message (message1_start, message1_stop);
                messagenum = 7; play1 = 1; messageplay = 1;
                replay_count = 0;}
        if (play_time7_message_selection == 2){
                Play_message (message2_start, message2_stop);
                messagenum = 7; play2 = 1; messageplay = 1;
                replay_count = 0;}
        if (play_time7_message_selection == 3){
                Play_message (message3_start, message3_stop);
                messagenum = 7; play3 = 1; messageplay = 1;
                replay_count = 0;}
        if (play_time7_message_selection == 4){
                Play_message (message4_start, message4_stop);
                messagenum = 7; play4 = 1; messageplay = 1;
                replay_count = 0;}
        if (play_time7_message_selection == 5){
                Play_message (message5_start, message5_stop);
                messagenum = 7; play5 = 1; messageplay = 1;
                replay_count = 0;}
        if (play_time7_message_selection == 6){
                Play_message (message6_start, message6_stop);
                messagenum = 7; play6 = 1; messageplay = 1;
                replay_count = 0;}
        if (play_time7_message_selection == 7){
                Play_message (message7_start, message7_stop);
                messagenum = 7; play7 = 1; messageplay = 1;
                replay_count = 0;}
        if (play_time7_message_selection == 8){
                Play_message (message8_start, message8_stop);
                messagenum = 7; play8 = 1; messageplay = 1;
                replay_count = 0;}
```

```
				Delay_ms(100);}
			if (pm == play_time8_pm && hour == play_time8_hour && minute ==
				play_time8_minute && second == 0 && no_play == 0) {
				no_play = 1;
				if (play_time8_message_selection == 1){
					Play_message (message1_start, message1_stop);
					messagenum = 8; play1 = 1; messageplay = 1;
					replay_count = 0;}
				if (play_time8_message_selection == 2){
					Play_message (message2_start, message2_stop);
					messagenum = 8; play2 = 1; messageplay = 1;
					replay_count = 0;}
				if (play_time8_message_selection == 3){
					Play_message (message3_start, message3_stop);
					messagenum = 8; play3 = 1; messageplay = 1;
					replay_count = 0;}
				if (play_time8_message_selection == 4){
					Play_message (message4_start, message4_stop);
					messagenum = 8; play4 = 1; messageplay = 1;
					replay_count = 0;}
				if (play_time8_message_selection == 5){
					Play_message (message5_start, message5_stop);
					messagenum = 8; play5 = 1; messageplay = 1;
					replay_count = 0;}
				if (play_time8_message_selection == 6){
					Play_message (message6_start, message6_stop);
					messagenum = 8; play6 = 1; messageplay = 1;
					replay_count = 0;}
				if (play_time8_message_selection == 7){
					Play_message (message7_start, message7_stop);
					messagenum = 8; play7 = 1; messageplay = 1;
					replay_count = 0;}
				if (play_time8_message_selection == 8){
					Play_message (message8_start, message8_stop);
					messagenum = 8; play8 = 1; messageplay = 1;
					replay_count = 0;}
				Delay_ms(100);}

// ________________________ Replay Message (3rd button) _________________________ /
//The play# variables are bits that are set when their corresponding messages
//have been played. For example, if message 2 is played, then play2 will be
//set to 1. If, once the message has been played, the replay button has been
//pressed (replay = 1), then the corresponding message will be replayed.
//			if (play1 == 1 && replay == 1){
//				replay = play1 = messageplay = 0 ;
//		Play_message (message1_start, message1_stop);
//				Delay_ms(100);}
//			if (play2 == 1 && replay == 1){
```

```
//                              replay = play2 = messageplay = 0;
//                  Play_message (message2_start, message2_stop);
//                              Delay_ms(100);}
//                      if (play3 == 1 && replay == 1){
//                              replay = play3 = messageplay = 0;
//                  Play_message (message3_start, message3_stop);
//                              Delay_ms(100);}
//                      if (play4 == 1 && replay == 1){
//                              replay = play4 = messageplay = 0;
//                  Play_message (message4_start, message4_stop);
//                              Delay_ms(100);}
//                      if (play5 == 1 && replay == 1){
//                              replay = play5 = messageplay = 0;
//                  Play_message (message5_start, message5_stop);
//                              Delay_ms(100);}
//                      if (play6 == 1 && replay == 1){
//                              replay = play6 = messageplay = 0;
//                  Play_message (message6_start, message6_stop);
//                              Delay_ms(100);}
//                      if (play7 == 1 && replay == 1){
//                              replay = play7 = messageplay = 0;
//                  Play_message (message7_start, message7_stop);
//                              Delay_ms(100);}
//                      if (play8 == 1 && replay == 1){
//                              replay = play8 = messageplay = 0;
//                  Play_message (message8_start, message8_stop);
//                              Delay_ms(100);}

// ______________________ Play all Messages for Dr. Sun _________________________ /
                        if (replay == 1){
                                relay_flag1 = 1;
                                Play_message (message1_start, message1_stop);
                                Delay_ms(200);
                                replay = 0;}

// __________________________ Save message play time _____________________________ /
//The following code is the algorithm to save the message number to EEPROM. The
//PIC first has to check whether the current memory location is empty (FF) or
// half full (#F) and then decide what number to save to memory. As described
//earlier in the code, the message number is always preceeded by FF. Once again,
//the best way to understand this algorithm is to go through it line by line
//and to draw out a flow chart.
                        if(messagenum != 0 && stop_save == 0){
                                readEEPROM();
                                if(EEDATA == 255){
                                        INCEEPROM();
                                        if(messagenum == 1) messagenum = 31;
                                        if(messagenum == 2) messagenum = 47;
                                        if(messagenum == 3) messagenum = 63;
                                        if(messagenum == 4) messagenum = 79;
                                        if(messagenum == 5) messagenum = 95;
                                        if(messagenum == 6) messagenum = 111;
                                        if(messagenum == 7) messagenum = 127;
                                        if(messagenum == 8) messagenum = 143;
                                        data = messagenum;
                                        Delay_ms(100);
                                        writeEEPROM(data);}
                                messagenum = 0;
                                readEEPROM();
                                messageaddr = EEADR;
                                SpecifyEEPROM(0xFF);
                                Delay_ms(100);
                                writeEEPROM(messageaddr);
                                Delay_ms(100);
                                SpecifyEEPROM(messageaddr);}
                        if(messagenum != 0 && stop_save == 0){
                                readEEPROM();
                                if(EEDATA != 255){
                                        INCEEPROM();
                                        if(messagenum == 1) messagenum = 241;
                                        if(messagenum == 2) messagenum = 242;
                                        if(messagenum == 3) messagenum = 243;
                                        if(messagenum == 4) messagenum = 244;
                                        if(messagenum == 5) messagenum = 245;
                                        if(messagenum == 6) messagenum = 246;
                                        if(messagenum == 7) messagenum = 247;
                                        if(messagenum == 8) messagenum = 248;
                                        data = messagenum;
                                        writeEEPROM(data);
                                        Delay_ms(100);
                                        INCEEPROM();}
                                messagenum = 0;
                                readEEPROM();
                                messageaddr = EEADR;
                                SpecifyEEPROM(0xFF);
                                Delay_ms(100);
                                writeEEPROM(messageaddr);
                                Delay_ms(100);
                                SpecifyEEPROM(messageaddr);}

                // __________________________ if (edit_messages) sequence _______________________ /
                        if (edit_messages) {
                                if (update_display) {
                                        Clear_screen ();
                                        Set_position (0);
                                        Print_line ("Edit Messages", 13);
```

```
                                        Set_position (64);
                                        if (edit_messages_mode == 0) Print_line ("Exit", 4);
                                        else {
                                                  Print_line ("Message", 7);
                                                  Print_2dig_num (edit_messages_mode, 73); }
                                        update_display = 0; }
                              if (scroll) {
                                        scroll = 0;
                                        edit_messages_mode++;
                                        if (edit_messages_mode == 9) edit_messages_mode = 0;
                                        update_display = 1; }
                              if (select) {
                                        select = 0;
                                        edit_messages = 0;
                                        if (edit_messages_mode == 0) main_menu = 1;
                                        else message = 1;
                                        update_display = 1;}}

// ___________________________ if (set_sensitivity) sequence ______________________ /
                    if (set_sensitivity) {
                              if (update_display) {
                                        Clear_screen ();
                                        Set_position (0);
                                        Print_line ("Set Sensitivity", 15);
                                        Set_position (64);
                                        if (set_sensitivity_mode == 0) Print_line ("Exit", 4);
                                        if (set_sensitivity_mode == 1) {Print_line ("High", 4); medium = low = 0;
high = 1;}
                                        if (set_sensitivity_mode == 2) {Print_line ("Mid ", 4); high = low = 0;
medium = 1;}
                                        if (set_sensitivity_mode == 3) {Print_line ("Low ", 4); high = medium = 0;
low = 1;}
                                        update_display = 0; }
                              if (scroll) {
                                        scroll = 0;
                                        set_sensitivity_mode++;
                                        if (set_sensitivity_mode == 4) set_sensitivity_mode = 0;
                                        update_display = 1; }
                              if (select) {
                                        select = 0;
                                        set_sensitivity = 0;
                                        main_menu = 1;
                                        update_display = 1; }}

// _______________________________ if (message) sequence __________________________ /
                    if (message) {
                              if (update_display) {
                                        Clear_screen ();
```

```
        Set_position (0);
        Print_line ("Message", 7);
        Print_2dig_num (edit_messages_mode, 9);
        Set_position (64);
        switch (message_mode) {
        case 0:
                Print_line ("Exit", 4);
                break;
        case 1:
                Print_line ("  Play", 6);
                break;
        case 2:
                if ((edit_messages_mode == 1 && slot1 == 0) ||
                        (edit_messages_mode == 2 && slot2 == 0) ||
                        (edit_messages_mode == 3 && slot3 == 0) ||
                        (edit_messages_mode == 4 && slot4 == 0) ||
                        (edit_messages_mode == 5 && slot5 == 0) ||
                        (edit_messages_mode == 6 && slot6 == 0) ||
                        (edit_messages_mode == 7 && slot7 == 0) ||
                        (edit_messages_mode == 8 && slot8 == 0))
                        Print_line ("Record", 6);
                else Print_line (" Erase", 6);
                break; }
update_display = 0; }
if (scroll) {
        scroll = 0;
        message_mode++;
        if (message_mode == 3) message_mode = 0;
        update_display = 1; }
if (select) {
        select = 0;
        switch (edit_messages_mode) {
        case 0: case 1:
                edit_message_start = message1_start;
                edit_message_stop = message1_stop;
                break;
        case 2:
                edit_message_start = message2_start;
                edit_message_stop = message2_stop;
                break;
        case 3:
                edit_message_start = message3_start;
                edit_message_stop = message3_stop;
                break;
        case 4:
                edit_message_start = message4_start;
                edit_message_stop = message4_stop;
                break;
                case 5:
                        edit_message_start = message5_start;
                        edit_message_stop = message5_stop;
                        break;
                case 6:
                        edit_message_start = message6_start;
                        edit_message_stop = message6_stop;
                        break;
                case 7:
                        edit_message_start = message7_start;
                        edit_message_stop = message7_stop;
                case 8:
                        edit_message_start = message8_start;
                        edit_message_stop = message8_stop;
                        break; }
                switch (message_mode) {
                case 0:
                        edit_messages_mode = 0;
                        edit_messages = 1;
                        message = 0;
                        update_display = 1;
                        break;
                case 1:
                        Play_message (edit_message_start, edit_message_stop);
                        message_edit_timer = 1;
                        message = 0;
                        update_display = 1;
                        break;
                case 2:
                        if ((edit_messages_mode == 1 && slot1 == 0) ||
                                (edit_messages_mode == 2 && slot2 == 0) ||
                                (edit_messages_mode == 3 && slot3 == 0) ||
                                (edit_messages_mode == 4 && slot4 == 0) ||
                                (edit_messages_mode == 5 && slot5 == 0) ||
                                (edit_messages_mode == 6 && slot6 == 0) ||
                                (edit_messages_mode == 7 && slot7 == 0) ||
                                (edit_messages_mode == 8 && slot8 == 0)) {
                                Record_message (edit_message_start,
edit_message_stop);
                                if (edit_messages_mode == 1) slot1 = 1;
                                if (edit_messages_mode == 2) slot2 = 1;
                                if (edit_messages_mode == 3) slot3 = 1;
                                if (edit_messages_mode == 4) slot4 = 1;
                                if (edit_messages_mode == 5) slot5 = 1;
                                if (edit_messages_mode == 6) slot6 = 1;
                                if (edit_messages_mode == 7) slot7 = 1;
                                if (edit_messages_mode == 8) slot8 = 1;
                                message_edit_timer = 1;
```

```
                                        message = 0; }
                                else {
                                        Erase_message (edit_message_start,
edit_message_stop);
                                        if (edit_messages_mode == 1) slot1 = 0;
                                        if (edit_messages_mode == 2) slot2 = 0;
                                        if (edit_messages_mode == 3) slot3 = 0;
                                        if (edit_messages_mode == 4) slot4 = 0;
                                        if (edit_messages_mode == 5) slot5 = 0;
                                        if (edit_messages_mode == 6) slot6 = 0;
                                        if (edit_messages_mode == 7) slot7 = 0;
                                        if (edit_messages_mode == 8) slot8 = 0; }
                        update_display = 1;
                        break; }}}

// ______________________ if (message_edit_timer) sequence ________________________ /
        if (message_edit_timer) {
                message_timer = 1;
                if (update_display) {
                        Clear_screen ();
                        Set_position (0);
                        Print_line ("Message", 7);
                        Print_2dig_num (edit_messages_mode, 9);
                        Set_position (64);
                        if (message_mode == 1) Print_line ("  Play:   sec", 13);
                        if (message_mode == 2) Print_line ("Record:   sec", 13);
                        if ((edit_messages_mode == 1 || edit_messages_mode == 2 ||
edit_messages_mode == 3 || edit_messages_mode == 4) && message_timer_count_long < 19)
Print_2dig_num (message_timer_count_long,72);
                        else if ((edit_messages_mode == 5 || edit_messages_mode == 6 ||
edit_messages_mode == 7 || edit_messages_mode == 8) && message_timer_count_short < 7)
Print_2dig_num (message_timer_count_short,72);
                        else {
                                message_timer = 0;
                                message_timer_count_short = 0;
                                message_timer_count_long = 0;
                                message_edit_timer = 0;
                                message = 1;
                                message_mode = 1;
                                update_display = 1; }
                update_display = 0;}}}}
```

Evaluation of the Device Effectiveness of the Activity Analyzer with Voice Individualized Direction

Dr. Patricia Burbank
University of Rhode Island College of Nursing
Dr. Ying Sun
University of Rhode Island Department of Computer, Electrical, and Biomedical Engineering
Tanya Wang, Joshua Harvey
University of Rhode Island Department of Computer, Electrical, and Biomedical Engineering
Rachel Gingras
University of Rhode Island College of Nursing

Background

The AAVID device was designed to create a portable tool that would provide feedback to the elderly population and help enhance their daily physical activity. The purpose of this project is to collect data and record the volunteer's activity and inactivity and determine if the digital and audio feedback plays correctly and the device is comfortable enough to wear on a daily basis. The device is constructed of a microprocessor interfaced with a voice recording chip and an accelerometer for audio and physical activity feedback. The microprocessor is programmed to record the patient's rate of daily motion and allows for personal messages to be played at specific times.

The microprocessor chosen for this device is the PIC18F452 8-bit microcontroller and is interfaced with the Winbond voice recording chip ISD1750. The Winbond chip was chosen because it has the capability of recording a message up to 100 seconds at a time. The chip also allows for the external push button (the standalone mode) or the serial peripheral interface mode which allow for communication between the microprocessor and the voice chip. The third chip chosen is the LIS302SG three dimensional accelerometer from STmicroelectronics. These components have been constructed on a breadboard and tested as a working circuit. The microprocessor has also been programmed using C++ language using the MPLab development tool. The purpose of the software is to constantly analyze the data of the rate of activity and inactivity as well as the magnitude of motions taken from the accelerometer. It was also programmed to have the opportunity to record voice messages from care takers or loved ones when a period of immobility has occurred. An I/O port has been included on the unit to allow for uploading the daily activity to a household computer, although it is not necessary for daily use.

Significance

The AAVID device was designed and built for the use of regulating and enhancing frequency of physical activity in the elderly population. The hopes for this project is perfect the circuit so that it will positively impact the lives of our elderly population by encouraging a healthy lifestyle through the use of smart voice playback technology and activity sensors. Some uses in the future may also be for prerecorded reminders to take medications and have loved ones record instructions for simple tasks like operating the stove. These uses not only will help the individual enhance their lifestyle with increased exercise, but could also help overcome some of the challenges the elderly face while continuing to live independently.

Specific Aims

- debug the circuit
- improve the functionality of the AAVID by upgrading the PIC processor
- Build the prototype
- Construct 3 prototypes for use in the study
- Obtain IRB approval for human study
- Conduct preliminary human study using volunteer URI engineering and nursing students that are at least 18 years of age and in good physical health.
- Analyze results and review the participants' data and completed questionnaire to conclude whether or not the AAVID device works correctly and can be worn comfortably on a daily basis.

Preliminary Results:

The result of the AAVID device thus far is that the circuitry on the breadboard including the preprogrammed PIC processor is currently performing the desired tasks, and we have successfully corrected the issues we were experiencing with the power to the circuit. The clock in microprocessor is working, allowing a prerecorded message to be played at the desired set time. We added a larger capacitor and a switch to correct the delay we were experiencing with the 9 volt battery when the power was turned on. Also the speaker being used now is too large to fit in the prototype, so another issue were are working on resolving is finding a smaller speaker and correcting the speaker circuit to play the recording loud enough and at a tone low enough for an elderly person with hearing loss to hear. At this stage in the project we are building a working prototype to match the circuit on the breadboard.

Project Description:

The main circuitry and coding for the microprocessor have been completed for the AAVID device. This part of the project will include building three prototypes matching the circuit on the breadboard to be tested by the volunteer URI students in the study. Once the mechanics of the device are finished, and the IRB is approved, we will finish building the devices to be worn in a fanny pack around the waist by the volunteers and begin the preliminary study. The researchers will include the Principle Investigator, Nursing Professor Patricia Burbank, DNSC, RN, Co-Investigator and Biomedical Engineering professor, Dr. Ying Sun, as well as three senior nursing students and three senior biomedical engineering students for this step of the project. For the user trial we will be examining the activity results of ten volunteer students who are at least 18 years of age and in moderate physical health, five of which will be from the URI Engineering Department and five will be from the URI College of Nursing.

We will be recruiting students with the use of a flyer that will be hung in White Hall and emailed to all undergraduate engineering students through the use of the URI Ugrads engineering email list. The advertisement was designed to give a brief overview of the project and reassure anyone interested that their participation is entirely voluntary. The students must be willing to wear a small, portable, box-like device at the waistline, and able to engage in low-level physical activity to participate in this research study.As the researchers, we will be looking for a diverse group, both male and females within that age range who are in moderate physical shape where any slight exercise requests would not cause any overexertion.

To begin we will obtain their daily schedule to personalize the messages at times that will not interfere with their sleep schedule, class schedule, and other activities throughout the day. The device will be programmed to play a number of messages throughout the day (only the researchers will know the preset number) at preset times or when a period of inactivity has occurred, encouraging the user to do moderate activity. These messages may be recorded using multiple voices in order to test the device's functionality. The content of the messages may vary and include several types of messages such as: encouragement to engage in physical activity, congratulations for engaging in physical activity, or instructions on exactly what type of moderate physical activity to participate in. The AAVID will be kept in a fanny pack and worn around the waist, and on the day the volunteer chooses to participate in the study, they must continue to wear the device all day, keep it in the ON mode, and refrain from tampering with any part of the device, as this may alter the data. At the end of the day the participants must report back to the researchers so they may collect the data of their daily activity and fill out a simple questionnaire about their experience with the device.

The clock and accelerometer in the circuit will record the duration of the activity throughout the day. With these results we can conclude whether or not the messages are played at the correct preset times and if the device is comfortable enough to be worn around the waist for an entire day. We will collect the data of the daily physical activity using a predesigned computer program. Using the previously constructed program, we will enter our data from the AAVID I/O port to a graduate student's computer so the data will remain confidential. From the questionnaire and collected data, we will conclude whether or not any changes need to be made to the circuitry or the way the device is worn.

Resources:

For our portion of the project we are using the AAVID data and circuitry previously constructed by Kyle Raferty and Tim Alberg as well as the PIC processor code completed by Gabe Ausfresser. We are also using the IRB manual found on the University of Rhode Island website to fully understand and complete all the paperwork and approvals necessary prior to performing the human study.

References:

- Sun, Ying. Scope of Work: Development and Testing of an Activity Analyzer with Voice Direction for Exercise.

- Rafferty, Kyle and Tim Alberg. Development of an Activity Analyzer with Voice Direction for Exercise.

Development of an Activity Analyzer with Voice Directions for Exercises

Kyle Rafferty, Timothy Alberg, Harold Greene, Gabriel Ausfresser, Ying Sun, PhD, Patricia Burbank, DNSc, RN*

University of Rhode Island, Dept. of Electrical, Computer & Biomedical Engineering, Kingston, RI 02881, USA; *University of Rhode Island, College of Nursing, Kingston, RI 02881, USA

Abstract – **The purpose of the project is to develop an interactive, portable device capable of monitoring motion as well as provide user feedback to promote continued exercise. The intended utility of the Activity Analyzer is testing whether or not audio and digital feedback promotes higher levels of exercise in the elderly population, particularly when the audio feedback is comprised of personalized messages recorded by loved ones. The device uses a microprocessor interfaced with a voice record chip, as well as a three dimensional accelerometer for motion input and detection. The onboard microprocessor is used to evaluate physical performance, determine which messages to be played at specified times, as well as save all messages and scores (including a time stamp) to the onboard EEPROM. A prototype of the Actitivity Analyzer has been successfully constructed.**

I. Introduction

As the elderly population in the United States continues to grow, one question that needs to be asked is whether increased life expectancy will result in a decline of independence of our older citizens. It is also well documented that physical activity in older adults will increase one's health and quality of life, while decreasing the probability of chronic diseases. A major problem that arises, however, is that more than 40% of adults over the age of 65 (in the United States) do not participate in any physical activity.

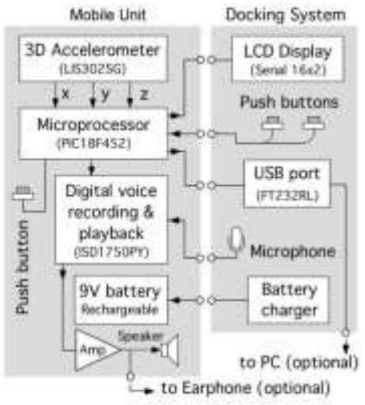


Fig. 1. Hardware schematic

The Activity Analyzer looks to encourage exercises by providing individualized voice messages from loved ones, along with daily feedback on the amount of exercise completed throughout the day. Messages can be prerecorded by family members to encourage older adults to get up and exercise during times of inactivity. Research has shown that, when intervening with older adults, it is essential that family-level perspective be incorporated throughout the process of a physical activity program: this model is referred to as the transtheoretical model of behavior change (TTM) [1]. Thus, the purpose of this device is to incorporate the TTM research into a portable, wearable device to test whether family voiced encouragement will increase activity and promote healthy living amongst older adults [1].

II. Methods

A. Hardware Development

The microprocessor of choice for this device is the PIC18F452 (Microchip Technology, Chandler, AZ). This processor is interfaced with the ISD1750 (Winbond) voice record (VR) chip, which is capable of recording up to 60 to 100 seconds worth of message time (dependent on the load resistor chosen). To enable an efficient communication, a serial-parallel interface (SPI) is established between the processor and VR chip.

For detecting motion a three-dimensional accelero-meter (LIS302sg, STmicro-electronics) is used. The accelerometer is capable of outputting motion signals in the x, y, and z directions. To eliminate discontinuities from the varying x, y, and z baselines, the outputs are averaged together through a resistive network and sent to the analog-to-digital converter (ADC) of the PIC processor.

There are two components to this project, the Activity Analyzer mobile unit and the docking system (both depicted individually in Fig. 1). The mobile unit contains the accelerometer, PIC processor, VR chip, and speaker. The docking system contains an LCD display, push buttons to interact with the mobile unit while docked, digital switch, USB device, and microphone input. While docked, the docking system is used to alter settings such as time of day, when messages are played, which messages are played, and how sensitive the accelerometer decoding algorithm is (all through a DB15 connection). The user can also record new messages, erase old ones, and send the data in the EEPROM of the PIC to a computer via USB.

B. Software Development

The software implemented for the PIC processor is comprised of four major components: the user menu controls, SPI communication, motion detection and scoring algorithm, and finally the memory storage algorithm. Figure 2 depicts a flow graph of these four major software components, as well as how they interact amongst one another in the code.

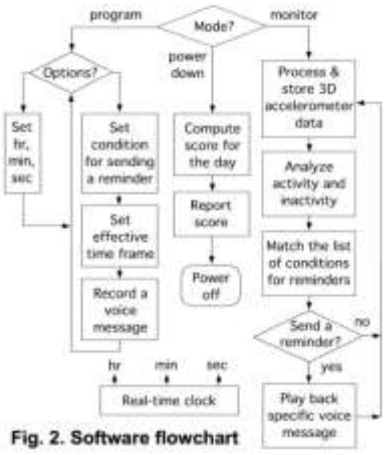

program
Mode?
monitor
Options?
power down
Process & store 3D accelerometer data
Set hr, min, sec
Set condition for sending a reminder
Compute score for the day
Analyze activity and inactivity
Set effective time frame
Report score
Match the list of conditions for reminders
Power off
Record a voice message
Send a reminder?
no
yes
hr
min
sec
Real-time clock
Play back specific voice message

Fig. 2. Software flowchart

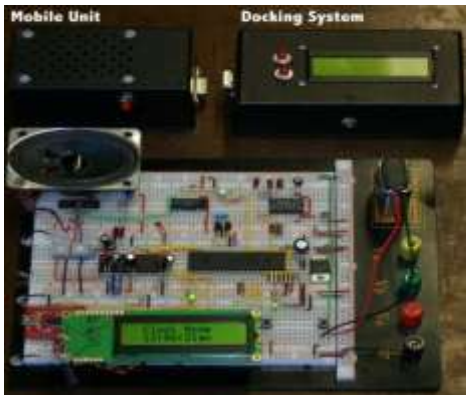


Fig. 3. The bread-board prototype (bottom) and the soldered prototype packaged into the mobile unit and the docking system (top).

For ease of use, it is important for the device to have an intuitive menu interface. The user menu control is set up in correspondence with two exterior push buttons: one to scroll and one to select. The user can select several different menus from the LCD screen, including a menu to set time, a menu to record, erase, play, or set play times, a menu to send the data via USB to a computer, and a menu to set the sensitivity of motion detection to name a few.

By scrolling and selecting (once you enter a menu, the user has the option to select exit), the user can easily maneuver through all the option available to them. It is also worth noting

that most of these opera-tions are controlled by sub functions outside of the main loop in code. This means adding functions or changing current functions is easier for the developer as the framework to do so is already in place.

Both the processor and the VR chip are capable of SPI communication. The processor requires most significant bit first and the VR chip requires least significant bit first. To maintain compatibility, the reversed bits are defined as protocol functions in the code such that the VR chip receives LSB first. In order to use the VR chip, the power up command and clear INT command must be sent prior to sending the SPI command for the desired action. After the action is implemented (play, rec, erase, forward message, set_play, set_rec, set_erase) the power down command should be sent. When a message is recorded on the VR chip the device stores starting and ending points which can be read via play and rec pointers. Alternatively, set_rec, set_play, and set_erase commands can be implemented to record, play, and erase messages from memory location A to memory location B. This method allows for more control over message length and can be used to optimize memory usage of the VR chip.

An integral part of the device involves the data received by the processor from the 3D accelerometer. One major issue involved with accelerometers is the drift of baseline voltage: this baseline voltage will drift both with age and orientation. To counter drift, an algorithm using a 16-point averaging filter can be used. The PIC will fill a 16-point array with 16 bytes of data from the ADC and then average the value (baseline voltage) of the sixteen points. The sampling rate is set at 1150 Hz. Thus, the 16-point buffer represents a time frame of 14 ms. This average value will then be able to act as threshold for motion detection, i.e. any peaks of voltage over the average voltage plus some voltage (threshold) will cause the PIC to recognize motion and increment a scoring counter. This array is then emptied and refilled periodically, allowing the threshold to change with the baseline voltage and any possible drift that arises.

Another functional component of the Activity Analyzer algorithm is data storage: saving the scores and messages played (with a time stamp) to be extracted and analyzed at the end of the day. However, the PIC18F452 is only comprised of 256 bytes of EEPROM, making one day's worth of storage impossible, considering each score would need 3 bytes: one for the score, one for the hour, and one for the minute. The solution to this problem was to 'split' every bite in two, using the 4 most significant bits and 4 least significant bits individually. This, although complex and tedious, allowed us to essentially double the memory storage capability of the processor while maintaining a five-minute temporal resolution for the scoring algorithm.

Finally, the user can upload data from the EEPROM of the PIC onto a personal computer (PC) via the USB port. A program on the PC, created by using the C# language, is able to take in all the data, split the incoming bytes, and plot all the scores on a score vs. time scale. This allows the user to visualize the amount of exercise throughout the day.

III. Results

As shown in Fig. 3, the hardware of the Activity Analyzer was first implemented on a breadboard to allow for ease of testing and debugging. The four software components of the Activity Analyzer were successfully implemented. Once all components of the device functioned as intended, including the data capture on a computer, the device was then prototyped into the docking system and wearable mobile unit. The preliminary test results on human subjects were obtained in a separate study [2].

IV. Discussion

This project involved the development of the Activity Analyzer; a device that is capable of providing individualized voice encouragement for older adults to exercise. The Activity Analyzer was designed around the transtheoretical model of behavior change [1] and looks to corroborate past research that indicates it is essential to provide family-level perspective throughout the process of a physical activity program. This wearable device was the result of an interdisciplinary collaboration between engineering and nursing. Future work will include building and testing more prototypes and eventually marketing this product for household use.

Acknowledgements

This study was supported in part by the Rhode Island Center for Nursing Excellence (RICNE) and a grant from the National Institutes of Health (2R44NS048682-02A1, PI: Sun).

References

[1] P.M. Burbank and D. Riebe. *Promoting Exercise and Behavior Change in Older Adults: Interventions and Transtheoretical Model*. NY: Springer, 2002.

[2] H. Greene, C. Dulude, A. Neves, Y. Sun, and P.M. Burbank. Performance evaluation of the activity analyzer. 38th Annual Northeast Bioengineering Conference, Temple University, Philadelphia, PA, March 16-18, 2012.

Performance Evaluation of the Activity Analyzer

Harold Greene, Courtney Dulude, Amanda Neves, Ying Sun, PhD, Patricia Burbank, DNSc, RN*
Department of Electrical, Computer and Biomedical Engineering, University of Rhode Island, Kingston, RI 02881, USA;
*College of Nursing, University of Rhode Island, Kingston, RI, 02881, USA

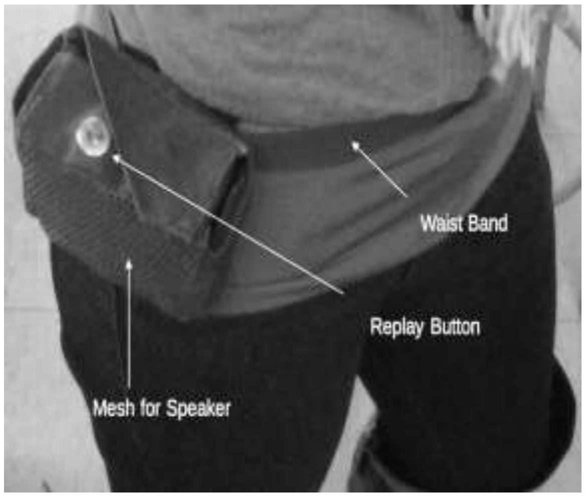


Fig. 1. The mobile unit of the Activity Analyzer in a carrying pouch strapped around the waist.

Abstract – **The purpose of this study is to evaluate the performance of the Activity Analyzer used to monitor and encourage physical activities of older adults. The Activity Analyzer is a wearable device designed to monitor the daily activity and inactivity according to a pre-programmed time schedule. The device is also capable of playing back pre-recorded messages to encourage exercises if periods of inactivity are detected under predefined conditions. At the end of the day, the daily activity data with a time resolution of 5 minutes can be retrieved from the mobile unit via a docking station for further analysis. The study provided preliminary test results from human subjects to evaluate the performance and optimize parameters of the device. The result is useful as a guideline for a larger-scale human study to assess the effectiveness of the device in terms of improving the daily activities for older adults.**

I. Introduction

As baby boomers now represent nearly one-third of the total United States population, researchers are prompted to find new ways to encourage a healthy lifestyle through increased physical activity in older adults. Research has shown that older adults have the ability to adapt to exercise in much the same way as they did in their younger years, and a regular exercise program can often slow age-related declines in health by reducing the risk of heart disease, cancer, osteoporosis, bone fractures, high blood pressure, as well as help with arthritis and depression.

According to the Transtheoretical Model of Behavior Change (TTM) [1], helping relationships such as that of a family member, play an important role in supporting behavior change. The Activity Analyzer developed in this study uses family member participation to encourage older adults to stay active, through tracking of physical activity levels, positive reinforcement, and voice personalization.

The purpose of this study is to conduct a pilot test for performance evaluation and parameter optimization before the Activity Analyzer can be used in a larger-scale study involving older adults.

II. Methods

A. Device Development

This device contains two main components – a mobile Activity Analyzer unit that is designed to be worn in a pouch around the individual's waist and a docking station. The mobile unit contains a microprocessor (PIC18F452, Microchip Technology, Chandler, AZ) used to store data and interface with all other major components including a voice recording integrated circuit chip (Winbond ISD1750) that can record and play back voice messages for a total of duration of 60-100 seconds. An accelerometer (STmicroelectronics lis302sg) is used to detect 3-dimensional motions (x, y, z directions). The x, y, and z signals are combined into one signal before it is acquired by the PIC processor. The mobile unit also contains an amplifier and speaker used to output pre-recorded messages, as well as a replay button in case the user needs to have a message repeated.

The second component, the docking station, contains a microphone that can be used by family members to record messages, as well as an LCD screen for displaying a manual that can be navigated using two pushbuttons. Using the LCD screen, caregivers can set the clock, record and erase messages, set the times at which these messages are to be displayed, and listen to previously recorded messages. Another important part to the docking station is the battery charger. This is used in combination with a series of switches to charge both the batteries for the mobile Activity Analyzer unit and the docking stations. This allows both components to run continuously without having to be turned off to charge. The docking station also contains a USB port that can be connected to a host computer. Through this USB port, the data from the accelerometer stored in the EEPROM of the PIC processor can be uploaded to the computer. The mobile Activity Analyzer unit and the docking station are connected through a 15-pin DB connector and must remain connected when using the push buttons and LCD display in the docking station to program the mobile unit [2].

B. *Human Subject Testing*

To develop an effective and ergonomic way for carrying the Activity Analyzer, we consulted faculty and students from the College of Nursing who have worked with the older adults. It was important to consider factors affecting functionality, convenience, appearance, comfort level, and user-friendliness for the older adult population. As shown in Fig. 1, the Activity Analyzer is worn around the waist in a custom pouch with a strap and clip to make it easy to be taken on and off. The design included durable fabric for the sides with a mesh material in the front for the speaker, as well as a metal ring on the top so the replay button would be easily accessible.

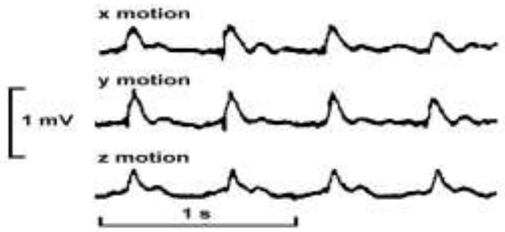


Fig. 2. Analog signals from the accelerometer showing motions in the x, y, and z axes before combining.

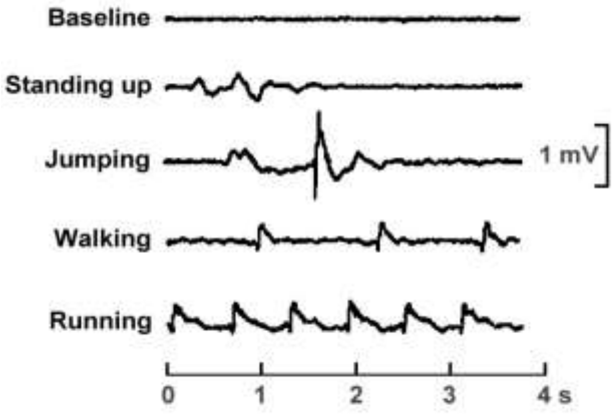


Fig. 3. The combined x-y-z motion signal shown for various types of activities.

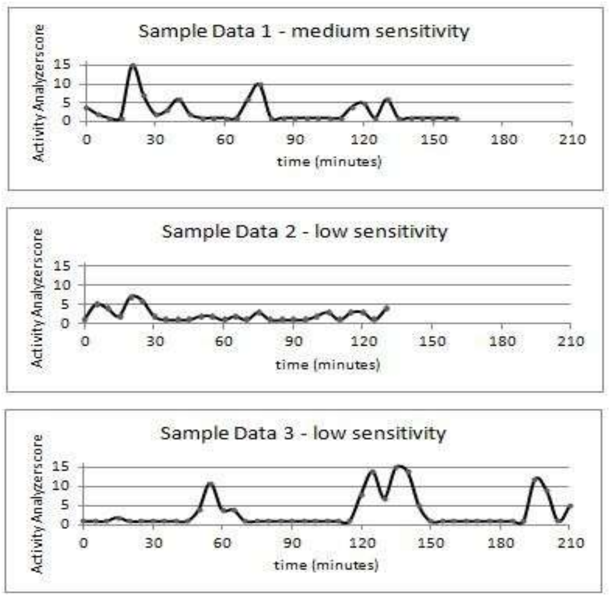


Fig. 4. Three samples of the activity data at the 5-minute temporal resolution retrieved from the Activity Analyzer.

Prior to the testing with human subjects, an approval from the Institutional Review Board (IRB) at the University of Rhode Island was obtained. The longer-term research objective is to test the hypothesis that the Activity Analyzer with voice messages recorded by loved ones can improve the activity level for community-dwelling older adults. The present study conducts a preliminary test on a small set of young, healthy subjects to determine if the basic function and comfort was sufficient before moving on to test older adults. The study also provides information about parameter settings such as the threshold for motion detection, which affects the sensitivity of the Activity Analyzer.

III. Results

Figure 2 shows the motion signals from the accelerometer in the x, y, and z directions. After summing up the three motions signals, the combined signal is acquired by the PIC processor to assess the activity level. Figure 3 shows the combined signal for various types of activities. The initial sampling rate is sufficiently high to reveal the shapes of the activity waveforms and to perform tasks such as the implementation of a pedometer, if needed. Then, the activity level is integrated at a 5-minute interval to produce an activity score, which is between 0 and 15. The data with the 5-minute temporal resolution are stored in the EEPROM, which can be retrieved later via the docking station. Figure 4 shows three sample data sets illustrating the typical data collected by the Activity Analyzer during activity testing. Data sets 1 and 2 display moderate household activity at two different sensitivity settings. A medium sensitivity setting results in an activity score of 1-2 for sitting, 3-12 for short periods of household activity, such as getting a glass of water or going to

the restroom, and a value of 13-15 for lengthier periods of activity, like preparing dinner or doing laundry. A low sensitivity setting results in scores of 1, 2-5, and 6-13 respectively for the same activities. Data set 3 describes longer periods of activity recorded at low sensitivity. The values for the peaks of 14 and 15 in the data set demonstrate sustained periods of higher activity, such as walking for five minutes.

IV. Discussion

The result of this study indicates that the performance of the Activity Analyzer is satisfactory and meets the original design goals. The activity data stored at the 5-minute temporal resolution are sufficiently sensitive to discern low, medium, and high levels of activities. Future work will include a larger scale study with an older adult population to assess the effectiveness of improving the daily activity level with voice messages recorded by the loved ones.

ACKNOWLEDGEMENTS

The authors thank the three nursing students of the University of Rhode Island: Jessica Ball, Alexandra Perone, and Heather Slevin, for their collaboration and helpful input to the design of the carrying pouch for the Activity Analyzer. This study was supported in part by an undergraduate research grant from the University of Rhode Island Division of Research & Economic Development, the Office of the Provost and VP for Academic Affairs, and the Honors Program and a grant from the National Institutes of Health (2R44NS048682-02A1, PI: Sun).

REFERENCES

[1] P.M. Burbank and D. Riebe. *Promoting Exercise and Behavior Change in Older Adults: Interventions and Transtheoretical Model*. NY: Springer, 2002.

[2] K. Rafferty, T. Alberg, H. Greene, Y. Sun, and P.M. Burbank. Development of an activity analyzer with voice directions for exercises. *38th Annual Northeast Bioengineering Conference, Temple University, Philadelphia, PA, March 16-18, 2012.*