SMART PILL DISPENSER

Submitted by

Jestina Peris

Pooja Rao

Under the guidance of

Prof. Rashmi Ravikumar

DEPARTMENT OF BIOMEDICAL ENGINEERING

YEAR 2016-2017

APPROVAL SHEET

This is to certify that

Jestina Peris

Pooja Rao

have submitted their project report on " **Smart Pill Dispenser**" as a part of the fulfilment of the B.E., Semester VII, Biomedical Engineering curriculum of the University of Mumbai.

Internal Examiner External Examiner

Project Guide Head of Department

ACKNOWLEDGEMENT

We wish to express our profound thanks to Dwarkadas J. Sanghvi College of Engineering (Biomedical Department) for providing us with all facilities and making the project possible.

We also take this opportunity to thank our Prof. Rashmi Ravikumar, our project guide, for her constant guidance and support throughout the course of this project.

We would like to thank our Head of Department, Dr. (Mrs.) Manali Godse, whose guidance and co-operation proved valuable during the project. We also thank other faculty members and fellow students for their support, encouragement and advice. Last but not the least, we would like to thank the non-teaching staff of the biomedical engineering department.

JESTINA PERIS (60013130018)

POOJA RAO (60013130020)

ABSTRACT

With an increase in the number of health problems, there is a need for a device to maintain and monitor the medicines of an individual. Management of prescribed medicines can be difficult, especially for senior citizens. We have put forth a design of a smart pill dispenser which will remind patients to take their medications on time and avoid underdose, overdose and skipping medications. The device is a compact, inexpensive and portable assembly. It makes the process of taking medicines comparatively easier. By taking advantage of flexible scheduling provided by medication directions, the device makes the user's medication schedule easy to adhere and tolerant to tardiness whenever possible. The report describes the action-oriented design, major components, hardware, and software structures of the Smart Pill Dispenser.

TABLE OF CONTENTS

LIST OF FIGURES

4.5	Circuit encased in a box (sealed)	19

LIST OF TABLES

1. INTRODUCTION

The fast-paced life of people has always taken a toll on the health of the people. We tend to neglect our health even though we are prescribed with the correct dosage. To counter various health issues, timely medication and course therapy for curing is required. The busy life schedule of the people often let down the best procedure. Most common reason for the failure of a method of cure is the failure of the patient to administer the dosage in the right proportion and at right time. Caring of the aged is of a serious concern in the developing countries. Family members are responsible for the care and management of the old. When the family members are away from home, it is utmost important that the aged or patients with various memory problems, remember to take their medication on time. Smart pill dispenser serves this purpose. [9]

The pill dispenser will be developed with off-the-shelf technology for the design and implementation of the project. The end goal is not to develop any new technologies associated with current manufactured dispensers. Rather, the goal is to design a unit with the same basic functionality, but for a much cheaper price.

1.1 Objectives:

i. Construct a device that is relatively small and lightweight.

ii. Develop the software in such a way that patients receive their medication reliably and safely as prescribed by their physician.

iii. Use as much off the shelf technology with basic electronic components to keep costs low.

iv. Develop a device that can perform all the necessary functions as stated in the project abstract.

1.2 Product Expectations:

i. An audio and visual alarm to notify patient that medication needs to be taken.

ii. A microprocessor controlled system that will automatically dispense medications at the preset time of day.

iii. Software that will monitor and record the time that medications have been taken by the patient.

iv. Software that will automatically adjust future medication dispensation based on when the medication was taken by the patient.

v. A mechanical locking mechanism that will keep the patient from over medicating.

vi. An LED display that will give pertinent instructions about the medication to the patient.

The following table shows the problems with the current technology and our solution to the problems.

Table 1.1. Difference between existing and proposed methodology [2]

Problems in existing methodology	**Solution in proposed methodology**
In RMAIS- RFID based Medication Adherence Intelligent System, expensive materials are used for system architecture and it is non-portable.	In Smart Pill Dispenser, inexpensive equipments are used and it is portable.
An Android based Medication Reminder Systemis not suitable for elderly people as it is difficult to operate for them.	The Proposed Smart Pill Dispenser prototype consists of a simple process. It can be operated easily. Cost wise, it is more economical and efficient system.

Hence we have seen the problems faced in existing methodology and how Smart Pill Dispenser can provide an effective solution for the same. As the cost of in-home medical care rises, it has become more and more incumbent among individuals to opt for a device that effectively takes care of their medications. Thus, the Smart Pill Dispenser is one such approach to help the elderly take their medicines efficiently.

2. Literature survey

In this chapter, we have given a run-down forthe need of a pill dispenser for combating issues related to missing dosage due to old age, mental illness, etc. andhow a pill dispenser is an invaluable aid to medical adherence after reviewing several technical papers.By relieving the users from the error-prone tasks of interpreting medication directions and administrating medications accordingly, the device can improve rigor in compliance and prevent serious medication errors.

2.1 Introduction:

According to World Health Organization, over 80% of the people above the age of 60 years are prescribed medicines that are to be administered 2 - 4 times a day. With the increase in Cardio vascular diseases and Diabetes among the peer group regular medicine administration has become a necessity but among this, 40-60% have issues related to forgetting to take medicines at the right time [10]. In their busy schedule, often, people neglect their health and forget to take their daily medication. Better self-management of medication enables people to enjoy improved quality of life and remain independent from home for longer.

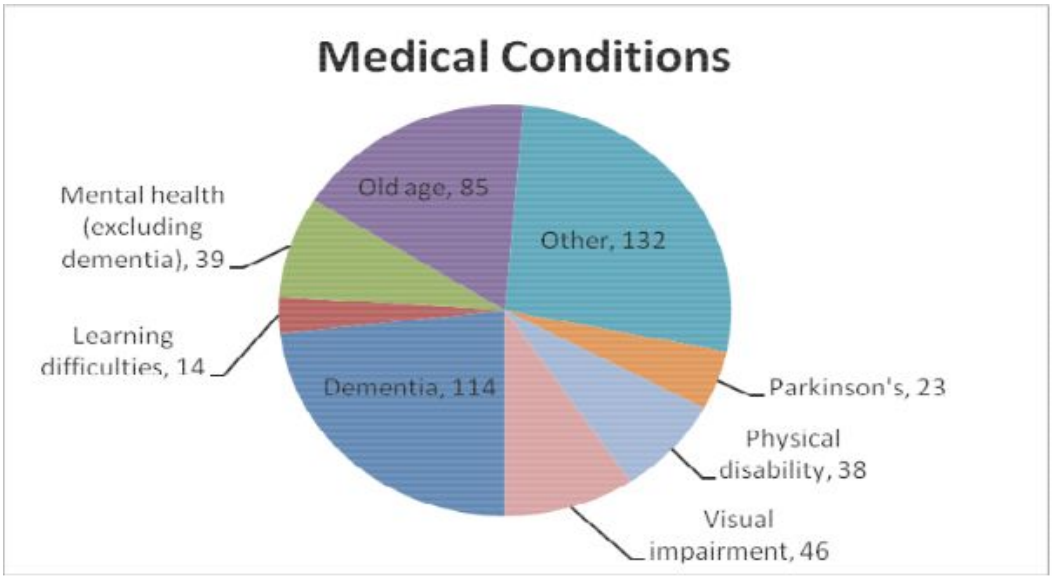

Figure 2.1 Pie-chart showing percentage of people who miss their medication

http://www.pivotell.co.uk/downloads/WMPDReport.pdf

Smart pill dispenser aims to distribute pills to the patients or elders at the desired time easily and efficiently. Not taking pills at the required time and dosage leads to health complications especially diabetic patients whose need to follow an accurate time for intake of pills.

According to administer of ageing, one of the main leaders admitting to hospital is mismanagement of medicines [10]. Pill dispenser allows for an in-home care provider to regulate patient's medication without constantly monitoring the patient. It caters the need of people who suffer from impaired ability to adhere to their prescribed medication regime. Since many patients are required to take multiple pills, a pill dispenser needs separate compartments either arranging pills for a particular time or separate compartments for each pill. Our project, the smart pill dispenser consists of four such compartments for morning, afternoon, evening and night dosage which are to be pre-filled in the respective slots, according to the prescription.

Need for a pill dispenser is most important for patients that take in multiple pills. Chances are high that patients consuming pills might get easily distracted due to phone calls, knock on door etc. These distractions confuse the patient as whether or not they have taken a particular dose of their medication; if a pill still remains in its compartment, it is apparent that it has not yet been taken, whereas if it is missing, it has already been taken. [8] Pill dispenser reduces time on reading labels which are difficult to read and often written in small letters.

The graph gives the average number of doses patients take throughout the day. [4]

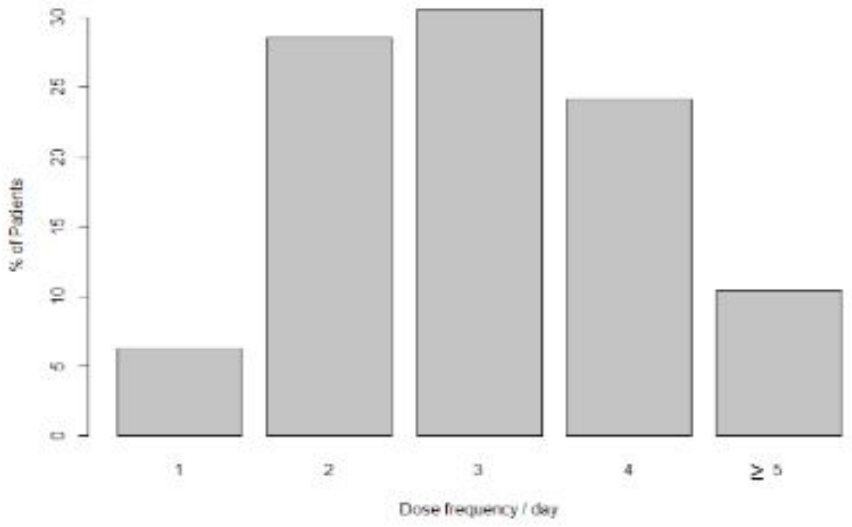

Figure 2.2 Percentage of patients by medication intake

http://www.usa.philips.com/b-dam/corporate/healthysociety/v2/Medido_WhitePaper_FINAL.pdf

2.2Existing Medication dispensing system

Figure 2.3. Pill tray without alarm www.careco.co.uk

There is a large variety of medication administration assistance devices for non-professional users. Most of them are manual, providing multiple compartments called pill trays. The cost of this pill tray is around 200INR. The pill tray has number of square-shaped compartments for each day of the week, although other more compact and discreet versions have come to market, including cylindrical and pen-shaped cases that can be filled with medication. Each compartment can hold different sizes and combination of medicines. The user is required to take the medicine from each tray but it does not provide any alarm to indicate the time of taking the medicine. [5]

The other automate pill dispenser available in the market stores dispenses medicines by a push of a button but is expensive and bulky. The dispenser has a locking system that avoids spillage of the medicines. When the time comes to take the medication, the device automatically releases the pre-measured dose into a small compartment that is easily opened and sounds a loud signal that it is time to take the

Figure 2.4. Pill dispenser with alarm and locking system www.pivotell.co.uk

medication The average cost of such automated dispenser is 10,000INR.[3]

Another type of automated pill dispenser is user-friendly giving intelligent notifications, security updates and for multiple users. Typical features on these dispensers include automatic pill dispensing at regular intervals, audible warnings, as well as a connection to either a phone line or the internet for monitoring purposes by the medical care provider. The dispenser is however bulky, non-portable, very expensive and not available in the Indian market. [12]

Figure 2.5.HERO Pill dispenser www.herohealth.com

Pill Dispensers are viewed as a way to prevent or reduce medication errors on the part of the patient. Thus, we can summarize the need of a cost effective, portable and handy device for managing daily dosage at a regular basis. After reviewing several pill dispensers in the market, we decided to come up with a pill dispenser that can perform a simple task for reminding patients to take the correct dose at the correct time and on the other hand, is inexpensive and light weighted than other devices in the market.

3. Design and Working of the Proposed Smart Pill Dispenser

In this chapter, we put forth the design of the Smart Pill and discuss the hardware components used and the flow of the software used to set an alarm. We propose to achieve a compact and user friendly device that will be easy to operate for senior citizens as well as for commercial use. We have also designed thisdevice with components that are easily available in the market.

3.1 Architecture of the Proposed Smart Pill Dispenser:

The following figure is the block diagram of the Smart Pill Dispenser

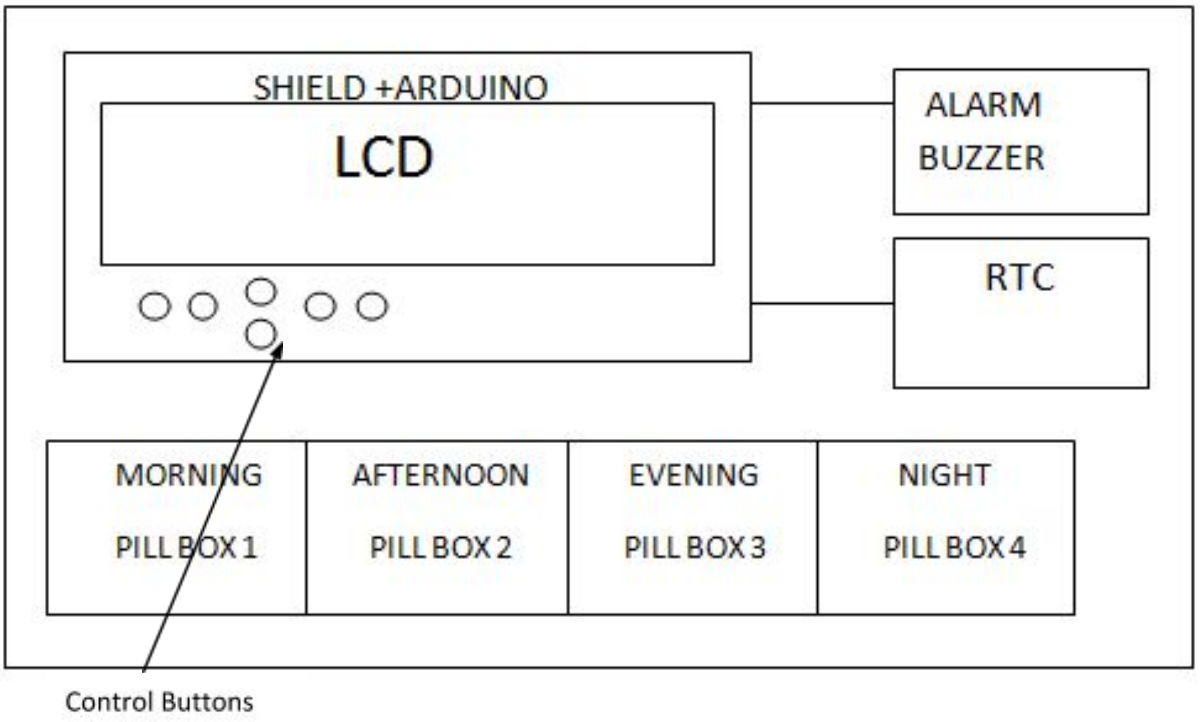

Fig 3. 1 Block diagram of the pill dispenser

The smart pill dispenser is encased in a box which has four major components:

i. Arduino Uno board
ii. RTC module DS1307
iii. LCD Keypad Shield for Arduino
iv. Active buzzer alarm module

Arduino Uno board:

Arduino Uno is a microcontroller board based on the ATmega328P (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. [3]

RTC ModuleDS1307:

DS1307 is a low power serial real time clock with full binary coded decimal (BCD) clock/calendar plus 56 bytes of NV SRAM. The RTC provides year, month, date, hour, minute and second information. It works on I2C protocol. Real time clock means it runs even after power failure. When power is reconnected, it displays the real time irespective to the time and duration it was in off state.The battery, which is connected to the RTC is a separate

one and is not related or connected to the main power supply. It is connected so that the IC can automatically switch to the backup supply in case of power failure. A 32.768 KHz crystal is connected to the oscillator terminal of DS1307 for 1 Hz oscillations. [3]

LCD Keypad Shield for Arduino:

It includes a 2x16 LCD display and 6 momentary push buttons. Pins 4, 5, 6, 7, 8, 9 and 10 are used to interface with the LCD. Just one Analog Pin 0 is used to read the five pushbuttons. The LCD shield supports contrast adjustment and back-lit on/off functions. This shield has 5 keys — select, up, right, down and left which allow you move through menus and make selections. The LCD & Keypad Shield requires a good 5V power supply to ensure the backlight fully illuminates and the display contrast is high. [3]

Active buzzer alarm module:

An active buzzer will generate a tone using an internal oscillator, so all that is needed is a DC voltage. The buzzer stays on once the signal is written (keeps playing, unlike a passive buzzer which stops when the signal stops). It has an Input Voltage between 3.5 to 5.5V. [3]

The schematic diagram for the 'Smart Pill Dispenser' is given below (Figure 3.2). The Smart Pill Dispenser is encased in a box, no bigger than 12 X 6 square inches and a width of about 3 inches. The RTC and the Arduino board assembly is hidden and only the LCD screen and the control buttons are visible. A small opening is given for the alarm buzzer. There are four small boxes attached to the main box, two on each side. These boxes act as pill containers for four dosages, namely, MORNING, AFTERNOON, EVENING and NIGHT.

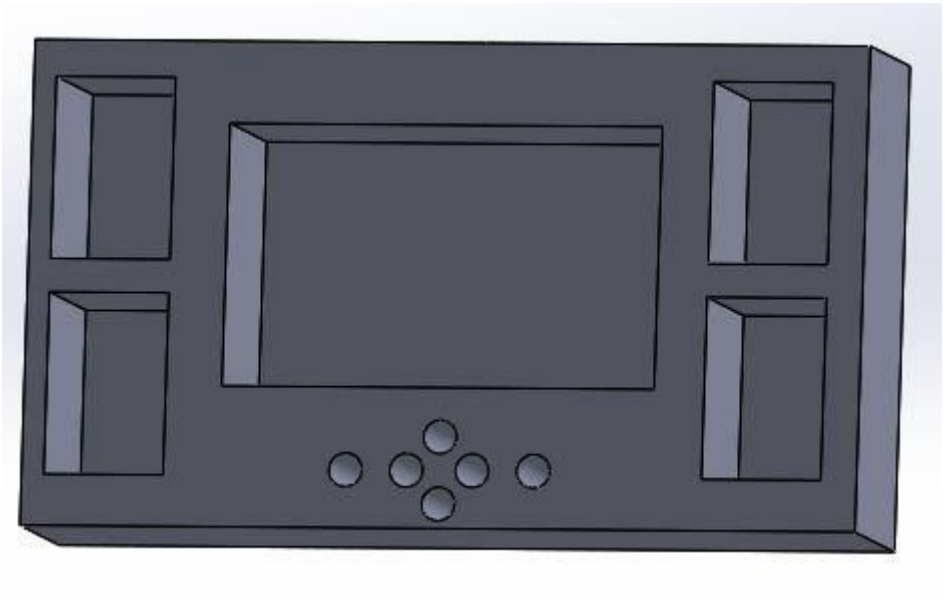

Figure 3.2 Schematic of the Smart Pill Dispenser

We have also attached LEDs which serve as a visual indiaction

3.2 Working the Proposed Smart Pill Dispenser:

The entire code is divided into states. Therefore, when a button is pressed, it transitions between states depending on the function. We have made use of predefined Finite State Machine (FSM) states like 'ALARM_TIME_MET' and 'TIME_OUT' and the rest of the states are user-defined functions which are called when a certain criteria is fulfilled.

We start at the 'show time' state [1] [9] [17]. Here, the date and time appear on the screen. The following buttons have the following functions.

RIGHT - The alarm turns on and an ALARM indication appears on the screen. By pressing the RIGHT button again, the alarm turns off. [2] [10] [18]

LEFT - Shows the alarm time. Also used to turn off the alarm. [3] [11] [19]

SELECT - Set Alarm1 in hours. By pressing it again, you can set minutes. By pressing the LEFT button again, you can set Alarm 2. The same process continues till Alarm4 [4] [5] [6] [12] [13] [14] [21] [22] [23] [30] [31] [32]

UP - Used to set alarm time by increasing the number. Also used to snooze the alarm [20] [28] [29]

DOWN - Used to set alarm time by decreasing the number.Also used to snooze the alarm for ten minutes [20] [28] [29]

The following diagram shows the flowchart of working of the alarm:

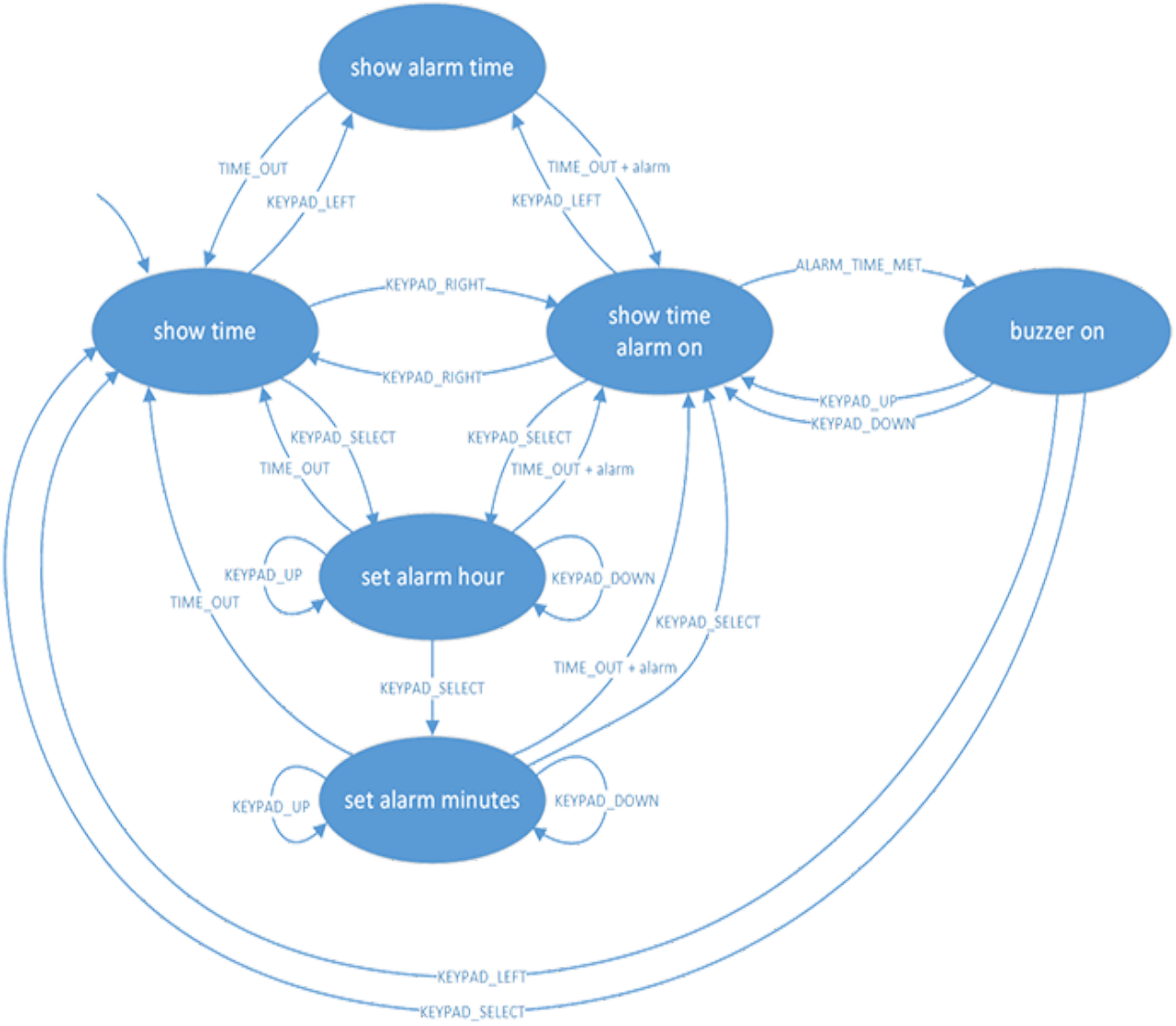

Figure 3.3 States of the alarm

http://blog.codebender.cc/2014/01/16/alarmclock/[3]

Thus we designed the analog buttons of the LCD shield to act as control buttons and also programmed the setup to set an alarm.The device opts to manage the timing of dosage in a

very effective way. As seen in this chapter, the complexity of the circuit has been significantly reduced thus making it user friendly, especially for senior citizens.

4. Results

In this chapter, we show the results that we achieved in stage 1 and 2 after we dumped the code on the Arduino board. We have displayed the current date and time on the LCD and set an alarm in hours and minutes. We were also able to overcome problems that we faced in Stage 1.

Results of Stage 1:

- We interfaced the Arduino board and the LCD Keypad Shield with the Real Time Clock (RTC) module to display the current date and time.
- We tried to set the alarm for a particular time and make the buzzer beep. We were successfully able to set the alarm, but the alarm buzzer did not give us the results that we expected. Instead, it buzzed as soon as the buzzer made contact with the port pin.

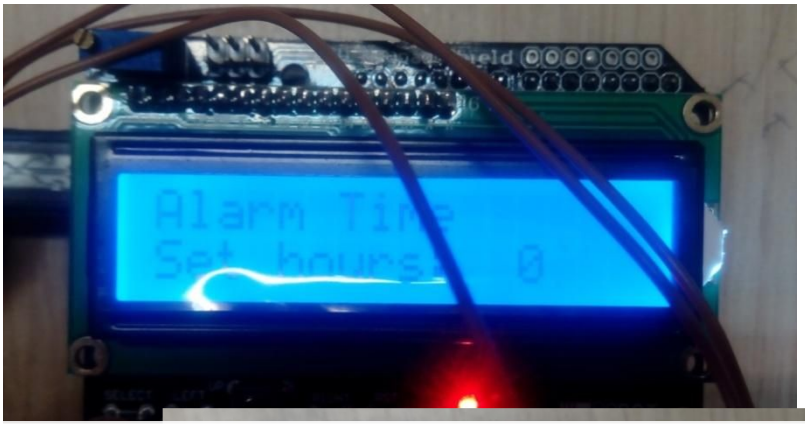

Figure 4.1 Setting alarm in hours

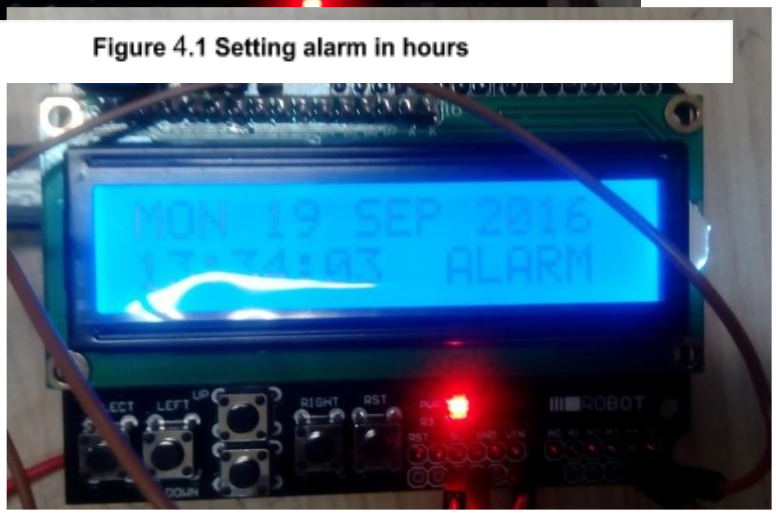

Figure 4.2 Displaying current date and time

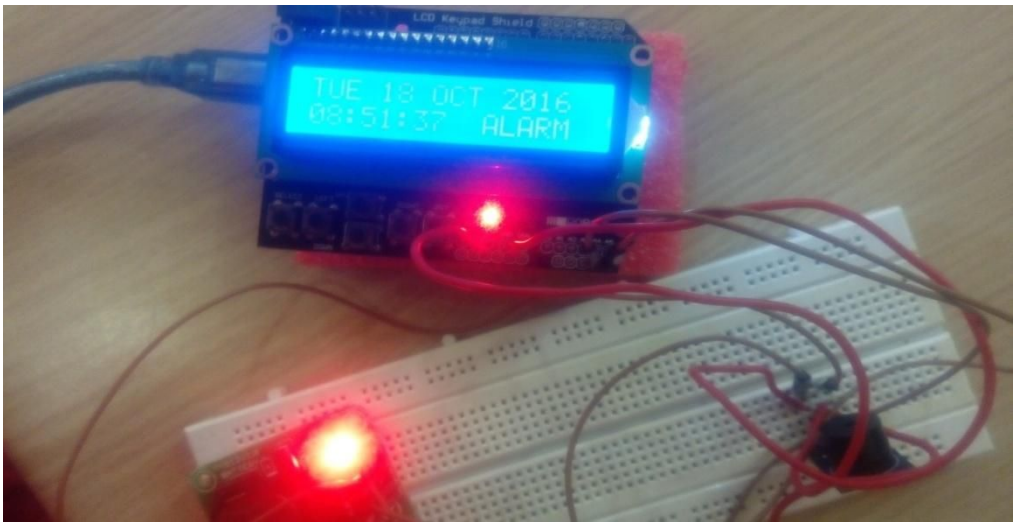

Fig. 4.3 Implementation of the circuit on breadboard

We made a few changes in the code blocks and implemented the circuit on a breadboard changing the port pins and received a loud, audible beep. In this circuit, we faced the same problem, i.e. the alarm went off as soon as it was connected to digital pin 3 or 11, which are the unused PWM pins of the Arduino board (pins 5, 6, 9 10 and 11 are used up by the LCD Keypad module).

Figure 4.4 Circuit encased in a box (not sealed)

Figure 4.5 Circuit encased in a box (sealed)

Results of Stage 2:

In order to get the circuit to meet our expectations, we took a few courses on Arduino programming to understand and improve our project.

- We were able to set multiple alarms for a day and also a timely alert to re-fill the boxes with pills.
- We made the LEDs glow according to the alarm time
- We were able to stop the continuous ringing of the buzzer.

Figure 4.6 Alarm 4 in minutes

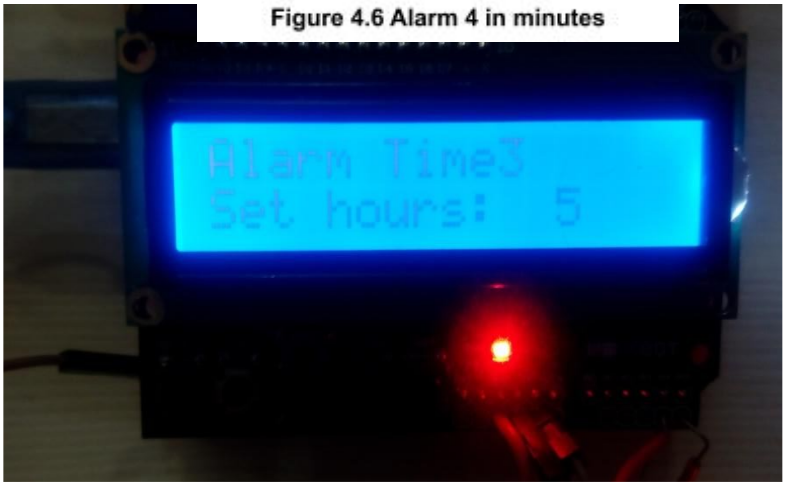

Figure 4.7 Alarm3 in hours

Thus, we finally put the entire assembly into a box of desirable dimensions (the dimensions used were almost same to the one we planned and designed on SolidWorks) with four pill boxes for storing the pills for the respective times of the day. The box has openings only for the LCD screen and ssembly serves the purpose of the 'Smart Pill Dispenser'.

5. Project Evaluation

In this chapter, we discuss the cost estimation and time frame of the project and our plans to be executed in the later stages. The Smart Pill Dispenser makes use of simple components that

are easily available in the market. The elderly population are greatly benefited from this device as it does not require expensive in-home HealthCare.

5.1 Cost Estimation:

One of the goals of this design was to produce a product comparable to current market pill dispensers but at a more affordable cost. The lowest cost unit that was found through internet research was approximately 8,000 INR[6]. The cost for the project can be seen in the chart below.

Table 5.1. Cost of the project

COMPONENT	COST(Rs.)
Arduino Uno	463
LCD Keypad Shield	354
Alarm Buzzer	240
RTCModule DS1307	246
Wires	25
Breadboard (only for testing)	-

Hence, the estimated cost of the project is **Rs.1328**

5.2 Project Plan:

Table 5.2. Plan of the project

WORK DISTRIBUTION	TIME FRAME
Software implementation	December 2016 -February 2017
Technical paper	February 2017
Minor Changes and beautification of the box	March 2017
Final Report and Blackbook	March 2017 – April 2017

Total time: December 2016 - April 2017

6. Future Plan

We have planned to make the following changes:

- Changes in the code - function to repeat the alarms without making the user to input the values again.
- Keeping a fixed time interval between two medications.
- In the future, we might consider making the pill dispenser a rotating assembly attached to a motor, which aligns the respective slots at the time it has to be taken.
- The current prototype needs the user to retrieve doses manually. We plan to make incorporate a system that can dispense the doses from the containers automatically.
- We also plan to design an application to keep the records of the medication taken and the history of the time when it's taken, to adjust the next medication according to the previous medication, to give access to the caretaker of the patient logs.
- We plan to work on the power supply to make it more effective and smaller.
- The entire design can be downsized and made into small VLSI chips, thus reducing the size further.

Thus we hope to bring some innovation to the existing project to add to the attraction to the device.

7. Conclusion

Timely medication is very necessary for the cure of any disease. By relieving the user from the error-prone tasks of interpreting medication directions and administrating medications accordingly, the proposed Smart Pill Dispensercan improve the adherence to the treatment and prevent serious medication errors. The devicewould remind the patient to take the daily medication on time by giving audio and visual signals. It would ensure that patients take their medicines on time. This system would be light-weight, compact, cost-effective and portable, such that it can be used by everyone. An advantage of this design is that new functions can be added and existing ones removed or revised with little or no need to modify the dispenser control structure. Much work on testing, experimentation and trial use (by actual users) of the proof-of-concept prototype remains to done to turn the prototype into a mature product.

8. References

[1]Aakash Sunil Salgia*, K. Ganesan and AshwinRaghunath,Smart Pill Box- Indian Journal of Science and Technology, *Vol 8(S2), 189–194, January 2015*

[2] Albert Jaison, Anu Simon, ArunChristin, Neethu John, Nisha Varghese, Yuvaraj.V, Robotic Pill Dispenser, IOSR Journal of Pharmacy and Biological Sciences (IOSR)Volume 9, Issue 4 (Jul -Aug. 2014)

[3]AlarmClock "http://blog.codebender.cc/2014/01/16/alarmclock/#comment-41737"

[4] Automatic Pill Dispenser "ww.add.ece.ufl.edu/4924/docs/Fall06_Good_Report3.pdf"

[5] Automatic pill dispenser http://www.epill.com/medtime.html

[6] Ben Anderson,KyleSmith,MattStrasser,JasonTerhune,Yao "Annie" Yao,Medicine dispensing device, October 2012

[7]JuGeon Pak and KeeHyun Park Construction of a Smart Medication Dispenser with High Degree of Scalability and Remote Manageability-J BIOMED BIOTECHNOL ,July 2012

[8]Mosca, Lori (2005*)*. "Heart to Heart: A Personal Plan for Creating a Heart-Healthy Family: Your Guide to the Good Life". Health & Fitness. Retrieved April 10, 2011.

[9] P. H. Tsai, C. Y. Yu, C. S. Shih, and J. W. S. Liu, Smart Medication Dispenser: Design, Architecture and Implementation, Institute of Information Science,October 2008

[10] R. Priyadharshini, S. Ramya, S. Kalaiyarasi, S. Mithuna, Manivannan. L, SuthanthiraVanitha. N,A Novel Approach of Microcontroller based Automatic Medication Reminder (AMR) System for Patients-International Journal of Engineering Research & Technology (IJERT)www.ijert.orgVol. 4 Issue 04, April-2015

[11]The Automated Pill Dispenser Project http://www.pivotell.co.uk/downloads/Report.pdf

[12]http://www.usa.philips.com/bdam/corporate/healthysociety/v2/Medido_WhitePaper_FINAL.pdf

APPENDIX

Datasheets:

ATmega48A/PA/88A/PA/168A/PA/328/P

ATMEL 8-BIT MICROCONTROLLER WITH 4/8/16/32KBYTES

IN-SYSTEM PROGRAMMABLE FLASH

Features

- High Performance, Low Power Atmel®AVR® 8-Bit Microcontroller Family
- Advanced RISC Architecture

– 131 Powerful Instructions – Most Single Clock Cycle Execution

– 32 x 8 General Purpose Working Registers

– Fully Static Operation

– Up to 20 MIPS Throughput at 20MHz

– On-chip 2-cycle Multiplier

- High Endurance Non-volatile Memory Segments

– 4/8/16/32KBytes of In-System Self-Programmable Flash program memory

– 256/512/512/1KBytes EEPROM

– 512/1K/1K/2KBytes Internal SRAM

– Write/Erase Cycles: 10,000 Flash/100,000 EEPROM

– Data retention: 20 years at 85°C/100 years at 25°C(1)

– Optional Boot Code Section with Independent Lock Bits

- In-System Programming by On-chip Boot Program
- True Read-While-Write Operation

– Programming Lock for Software Security

✦ Atmel® QTouch® library support

– Capacitive touch buttons, sliders and wheels

– QTouch and QMatrix® acquisition

– Up to 64 sense channels

✦ Peripheral Features

– Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode

– One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode

– Real Time Counter with Separate Oscillator

– Six PWM Channels

– 8-channel 10-bit ADC in TQFP and QFN/MLF package

✦ Temperature Measurement

– 6-channel 10-bit ADC in PDIP Package

✦ Temperature Measurement

– Programmable Serial USART

– Master/Slave SPI Serial Interface

– Byte-oriented 2-wire Serial Interface (Philips I2C compatible)

– Programmable Watchdog Timer with Separate On-chip Oscillator

– On-chip Analog Comparator

– Interrupt and Wake-up on Pin Change

ATmega48A/PA/88A/PA/168A/PA/328/P

ATMEL 8-BIT MICROCONTROLLER WITH 4/8/16/32KBYTES

IN-SYSTEM PROGRAMMABLE FLASH

Atmel-8271J-AVR- ATmega-Datasheet_11/2015

✦ Special Microcontroller Features

– Power-on Reset and Programmable Brown-out Detection

– Internal Calibrated Oscillator

– External and Internal Interrupt Sources

– Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby

✦ I/O and Packages

– 23 Programmable I/O Lines

– 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF

✦ Operating Voltage:

– 1.8 - 5.5V

✦ Temperature Range:

– -40℃ to 85℃

✦ Speed Grade:

– 0 - 4MHz@1.8 - 5.5V, 0 - 10MHz@2.7 - 5.5.V, 0 - 20MHz @ 4.5 - 5.5V

✦ Power Consumption at 1MHz, 1.8V, 25℃

– Active Mode: 0.2mA

– Power-down Mode: 0.1µA

– Power-save Mode: 0.75µA (Including 32kHz RTC)

Code:

```
#include <Wire.h>  // Required by RTClib
#include <LiquidCrystal.h>  // Required by LCDKeypad
#include <LCDKeypad.h>
#include "RTClib.h"

#define TIME_OUT 5  // One of the system's FSM transitions
#define ALARM_TIME_MET 6  // One of the system's FSM transitions

#define BUZZER_PIN 3 // Output PWM pin for the buzzer
#define SNOOZE 10  // Minutes to snooze
```

```
#define led1 11
#define led2 12
#define led3 13
// The different states of the system
enum states
{
  SHOW_TIME,          //[1]  Displays the time and date
  SHOW_TIME_ALARM_ON,  //[2] Displays the time and date, and alarm is on
   SHOW_ALARM_TIME,      //[3] Displays the alarm time and goes back to time and date after 3 seconds
   SET_ALARM_HOUR1,      // [4] Option for setting the alarm hours. If provided, it moves on to alarm minutes.
                         //   Otherwise, it times out after 5 seconds and returns to time and date
  SET_ALARM_MINUTES1,   // Option for setting the alarm minutes. If provided, it finally sets the alarm time and alarm.
                         //   Otherwise, it times out after 5 seconds and returns to time and date
   SET_ALARM_HOUR2,      // [5] Option for setting the alarm hours. If provided, it moves on to alarm minutes.
                         //   Otherwise, it times out after 5 seconds and returns to time and date
  SET_ALARM_MINUTES2,
   SET_ALARM_HOUR3,      // [6] Option for setting the alarm hours. If provided, it moves on to alarm minutes.
                         //   Otherwise, it times out after 5 seconds and returns to time and date
  SET_ALARM_MINUTES3,
   SET_ALARM_HOUR4,      // [7] Option for setting the alarm hours. If provided, it moves on to alarm minutes.
                         //   Otherwise, it times out after 5 seconds and returns to time and date
  SET_ALARM_MINUTES4,
  BUZZER_ON           // [8] Displays the time and date, and buzzer is on (alarm time met)
};

// Creates an LCDKeypad instance
```

```
// It handles the LCD screen and buttons on the shield
LCDKeypad lcd;

// Creates an RTC_DS1307 instance
// It handles the DS1307 Real-Time Clock
RTC_DS1307 RTC;

states state;  // Holds the current state of the system
int8_t button;  // Holds the current button pressed
uint8_t alarmHours1 = 0, alarmMinutes1 = 0;  // Holds the current alarm time
uint8_t alarmHours2 = 0, alarmMinutes2 = 0;
uint8_t alarmHours3 = 0, alarmMinutes3 = 0;
uint8_t alarmHours4 = 0, alarmMinutes4 = 0;
uint8_t tmpHours1;
uint8_t tmpHours2;
uint8_t tmpHours3;
uint8_t tmpHours4;
uint8_t tmpMinutes1;
uint8_t tmpMinutes2;
uint8_t tmpMinutes3;
uint8_t tmpMinutes4;
boolean alarm = false;  // Holds the current state of the alarm
unsigned long timeRef;
DateTime now;  // Holds the current date and time information



void setup()
{
pinMode(BUZZER_PIN, OUTPUT);  // Buzzer pin
pinMode(led1, OUTPUT);
pinMode(led2, OUTPUT);
```

```
pinMode(led3, OUTPUT);
   // Initializes the LCD and RTC instances
lcd.begin(16, 2);
Wire.begin();
RTC.begin();
digitalWrite(led1, LOW);
digitalWrite(led2, LOW);
digitalWrite(led3, LOW);
analogWrite(BUZZER_PIN, 255);

state = SHOW_TIME;  // Initial state of the FSM

   // Uncomment this to set the current time on the RTC module
   RTC.adjust(DateTime(__DATE__, __TIME__));
}

void loop()
{
timeRef = millis();

   // Uses the current state to decide what to process
switch (state)
   {
case SHOW_TIME: //[9]
showTime();
break;
case SHOW_TIME_ALARM_ON: //[10]
showTime();
checkAlarmTime();
break;
case SHOW_ALARM_TIME: //[11]
showAlarmTime();
```

```
break;
case SET_ALARM_HOUR1:// [12]
setAlarmHours1();
if ( state != SET_ALARM_MINUTES1 )
break;
case SET_ALARM_MINUTES1:
setAlarmMinutes1();
break;
case SET_ALARM_HOUR2: //[13]
setAlarmHours2();
if ( state != SET_ALARM_MINUTES2 )
break;
case SET_ALARM_MINUTES2:
setAlarmMinutes2();
break;
case SET_ALARM_HOUR3: //[14]
setAlarmHours3();
if ( state != SET_ALARM_MINUTES3 )
break;
case SET_ALARM_MINUTES3:
setAlarmMinutes3();
break;
case SET_ALARM_HOUR4: //[15]
setAlarmHours4();
if ( state != SET_ALARM_MINUTES4 )
break;
case SET_ALARM_MINUTES4:
setAlarmMinutes4();
break;
case BUZZER_ON: //[16]
showTime();
break;
```

```
    }

    // Waits about 1 sec for events (button presses)
    // If a button is pressed, it blocks until the button is released
    // and then it performs the applicable state transition
while ( (unsigned long)(millis() - timeRef) < 970 )
    {
if ( (button = lcd.button()) != KEYPAD_NONE )
        {
while ( lcd.button() != KEYPAD_NONE ) ;
transition(button);
break;
        }
    }
}

// Looks at the provided trigger (event)
// and performs the appropriate state transition
// If necessary, sets secondary variables
void transition(uint8_t trigger)
{

switch (state)
    {
case SHOW_TIME: //[17]
if ( trigger == KEYPAD_LEFT )  state = SHOW_ALARM_TIME;
else if ( trigger == KEYPAD_RIGHT )
{
 alarm = true;
state = SHOW_TIME_ALARM_ON; //[18]
}
else if ( trigger == KEYPAD_SELECT ) state = SET_ALARM_HOUR1;
```

```
break;
case SHOW_TIME_ALARM_ON: //[19]
if ( trigger == KEYPAD_LEFT ) state = SHOW_ALARM_TIME;
else if ( trigger == KEYPAD_RIGHT )
{
 alarm = false; state = SHOW_TIME;
}
else if ( trigger == KEYPAD_SELECT ) state = SET_ALARM_HOUR1;
else if ( trigger == ALARM_TIME_MET )
 {
analogWrite(BUZZER_PIN,0); state = BUZZER_ON;
 }
break;
case SHOW_ALARM_TIME: //[20]
if ( trigger == TIME_OUT )
{
 if ( !alarm ) state = SHOW_TIME;
else state = SHOW_TIME_ALARM_ON;
 }
break;
case SET_ALARM_HOUR1: // [21]
if ( trigger == KEYPAD_SELECT ) state = SET_ALARM_MINUTES1;
else if ( trigger == TIME_OUT )
{
 if ( !alarm ) state = SHOW_TIME;
else state = SHOW_TIME_ALARM_ON;
}
break;
case SET_ALARM_MINUTES1:
if ( trigger == KEYPAD_SELECT )
 {
 alarm = true; state = SET_ALARM_HOUR2;
```

```
 }
else if ( trigger == TIME_OUT )
 {
 if ( !alarm ) state = SHOW_TIME;
else state = SHOW_TIME_ALARM_ON;
}
break;
case SET_ALARM_HOUR2: //[22]
if ( trigger == KEYPAD_SELECT ) state = SET_ALARM_MINUTES2;
else if ( trigger == TIME_OUT )
{
 if ( !alarm ) state = SHOW_TIME;
else state = SHOW_TIME_ALARM_ON;
 }
break;
case SET_ALARM_MINUTES2:
if ( trigger == KEYPAD_SELECT ) { alarm = true; state = SET_ALARM_HOUR3; }
else if ( trigger == TIME_OUT )
{
 if ( !alarm ) state = SHOW_TIME;
else state = SHOW_TIME_ALARM_ON;
 }
break;
case SET_ALARM_HOUR3: //[23]
if ( trigger == KEYPAD_SELECT ) state = SET_ALARM_MINUTES3;
else if ( trigger == TIME_OUT )
 {
 if ( !alarm ) state = SHOW_TIME;
else state = SHOW_TIME_ALARM_ON;
}
break;
case SET_ALARM_MINUTES3:
```

```
if ( trigger == KEYPAD_SELECT )
 {
 alarm = true; state = SET_ALARM_HOUR4;
}
else if ( trigger == TIME_OUT )
 {
 if ( !alarm ) state = SHOW_TIME;
else state = SHOW_TIME_ALARM_ON;
 }
break;
case SET_ALARM_HOUR4: //[24]
if ( trigger == KEYPAD_SELECT ) state = SET_ALARM_MINUTES4;
else if ( trigger == TIME_OUT )
 {
 if ( !alarm ) state = SHOW_TIME;
else state = SHOW_TIME_ALARM_ON;
 }
break;
case SET_ALARM_MINUTES4:
if ( trigger == KEYPAD_SELECT )
 {
 alarm = true; state = SHOW_TIME_ALARM_ON;
}
else if ( trigger == TIME_OUT )
 {
 if ( !alarm ) state = SHOW_TIME;
else state = SHOW_TIME_ALARM_ON;
 }
break;
case BUZZER_ON: //[25]
if ( trigger == KEYPAD_UP || trigger == KEYPAD_DOWN )
{
```

```
 analogWrite(BUZZER_PIN, 255);
digitalWrite(led1, LOW);
digitalWrite(led2, LOW);
 digitalWrite(led3, LOW);
snooze();
 state = SHOW_TIME_ALARM_ON;
 }
if ( trigger == KEYPAD_SELECT || trigger == KEYPAD_LEFT )
 {
 analogWrite(BUZZER_PIN, 255);
 digitalWrite(led1, LOW);
digitalWrite(led2, LOW);
digitalWrite(led3, LOW);
alarm = false;
state = SHOW_TIME;
 }
break;
   }
}

// Displays the current date and time, and also an alarm indication
// e.g. SAT 04 JAN 2014, 22:59:10  ALARM
void showTime() //[26]
{
now = RTC.now();
const char* dayName[] = { "SUN", "MON", "TUE", "WED", "THU", "FRI", "SAT" };
const char* monthName[] = { "JAN", "FEB", "MAR", "APR", "MAY", "JUN", "JUL",
"AUG", "SEP", "OCT", "NOV", "DEC" };

lcd.clear();
lcd.print(String(dayName[now.dayOfTheWeek()]) + " " +
          (now.day() < 10 ? "0" : "") + now.day() + " " +
```

```
monthName[now.month()-1] + " " + now.year());
lcd.setCursor(0,1);
lcd.print((now.hour() < 10 ? "0" : "") + String(now.hour()) + ":" +
          (now.minute() < 10 ? "0" : "") + now.minute() + ":" +
          (now.second() < 10 ? "0" : "") + now.second() + (alarm ? "  ALARM" : ""));
}

// Displays the current alarm time and transitions back to show
// date and time after 2 sec (+ 1 sec delay from inside the loop function)
// e.g. Alarm Time HOUR: 08 MIN: 20
void showAlarmTime() //[27]
{
lcd.clear();
lcd.print("Alarm Time1");
lcd.setCursor(0,1);
lcd.print(String("HOUR: ") + ( alarmHours1 < 9 ? "0" : "" ) + alarmHours1 +
              " MIN: " + ( alarmMinutes1 < 9 ? "0" : "" ) + alarmMinutes1);
delay(2000);
transition(TIME_OUT);
}

// Checks if the alarm time has been met,
// and if so initiates a state transition
void checkAlarmTime() //[28]
{
if( now.hour() == alarmHours1 &&now.minute() == alarmMinutes1 )
  {
  digitalWrite(led1, HIGH);
  transition(ALARM_TIME_MET);
  }

if( now.hour() == alarmHours2 &&now.minute() == alarmMinutes2 )
```

```
  {
    digitalWrite(led2, HIGH);
  transition(ALARM_TIME_MET);
  }

if ( now.hour() == alarmHours3 &&now.minute() == alarmMinutes3 )
  {
    digitalWrite(led3, HIGH);
  transition(ALARM_TIME_MET);
  }
if( now.hour() == alarmHours4 &&now.minute() == alarmMinutes4 )
transition(ALARM_TIME_MET);
}

// When the buzzer is ringing, by pressing the UP or DOWN buttons,
// a SNOOZE (default is 10) minutes delay on the alarm time happens
void snooze() //[29]
{
   alarmMinutes1 += SNOOZE;
if ( alarmMinutes1 > 59 )
   {
      alarmHours1 += alarmMinutes1 / 60;
      alarmMinutes1 = alarmMinutes1 % 60;
   }
}

// The first of a 2 part process for setting the alarm time
// Receives the alarm time hour. If not provided within 5 sec,
// times out and returns to a previous (time and date) state

void setAlarmHours1() //[30]
{
```

```
unsigned long timeRef;
boolean timeOut = true;

lcd.clear();
lcd.print("Alarm Time1");

   tmpHours1 = 0;
timeRef = millis();
lcd.setCursor(0,1);
lcd.print("Set hours:  0");

while ( (unsigned long)(millis() - timeRef) < 5000 )
   {
uint8_t button = lcd.button();

if ( button == KEYPAD_UP )
      {
         tmpHours1 = tmpHours1 <23 ? tmpHours1 + 1 : tmpHours1;
lcd.setCursor(11,1);
lcd.print("  ");
lcd.setCursor(11,1);
if ( tmpHours1 < 10 ) lcd.print(" ");
lcd.print(tmpHours1);
timeRef = millis();
      }
else if ( button == KEYPAD_DOWN )
      {
         tmpHours1 = tmpHours1 >0 ? tmpHours1 - 1 : tmpHours1;
lcd.setCursor(11,1);
lcd.print("  ");
lcd.setCursor(11,1);
if ( tmpHours1 < 10 ) lcd.print(" ");
```

```
lcd.print(tmpHours1);
timeRef = millis();
      }
else if ( button == KEYPAD_SELECT )
      {
while ( lcd.button() != KEYPAD_NONE ) ;
timeOut = false;
break;
      }
delay(150);
   }

if ( !timeOut ) transition(KEYPAD_SELECT);
else transition(TIME_OUT);
}

// The second of a 2 part process for setting the alarm time
// Receives the alarm time minutes. If not provided within 5 sec,
// times out and returns to a previous (time and date) state
// If minutes are provided, sets the alarm time and turns the alarm on
void setAlarmMinutes1()
{
unsigned long timeRef;
boolean timeOut = true;
   uint8_t tmpMinutes1 = 0;

lcd.clear();
lcd.print("Alarm Time1");

timeRef = millis();
lcd.setCursor(0,1);
lcd.print("Set minutes:  0");
```

```
while ( (unsigned long)(millis() - timeRef) < 5000 )
   {
uint8_t button = lcd.button();

if ( button == KEYPAD_UP )
      {
         tmpMinutes1 = tmpMinutes1 <59 ? tmpMinutes1 + 1 : tmpMinutes1;
lcd.setCursor(13,1);
lcd.print("  ");
lcd.setCursor(13,1);
if ( tmpMinutes1 < 10 ) lcd.print(" ");
lcd.print(tmpMinutes1);
timeRef = millis();
      }
else if ( button == KEYPAD_DOWN )
      {
         tmpMinutes1 = tmpMinutes1 >0 ? tmpMinutes1 - 1 : tmpMinutes1;
lcd.setCursor(13,1);
lcd.print("  ");
lcd.setCursor(13,1);
if ( tmpMinutes1 < 10 ) lcd.print(" ");
lcd.print(tmpMinutes1);
timeRef = millis();
      }
else if ( button == KEYPAD_SELECT )
      {
while ( lcd.button() != KEYPAD_NONE ) ;
timeOut = false;
break;
      }
delay(150);
```

```
   }

if ( !timeOut )
   {
      alarmHours1 = tmpHours1;
      alarmMinutes1 = tmpMinutes1;
transition(KEYPAD_SELECT);
   }
else transition(TIME_OUT);
}
void setAlarmHours2() //[31]
{
unsigned long timeRef;
boolean timeOut = true;

lcd.clear();
lcd.print("Alarm Time2");

   tmpHours2 = 0;
timeRef = millis();
lcd.setCursor(0,1);
lcd.print("Set hours:  0");

while ( (unsigned long)(millis() - timeRef) < 5000 )
   {
uint8_t button = lcd.button();

if ( button == KEYPAD_UP )
      {
         tmpHours2 = tmpHours2 <23 ? tmpHours2 + 1 : tmpHours2;
lcd.setCursor(11,1);
lcd.print("  ");
```

```
lcd.setCursor(11,1);
if ( tmpHours2 < 10 ) lcd.print(" ");
lcd.print(tmpHours2);
timeRef = millis();
      }
else if ( button == KEYPAD_DOWN )
      {
         tmpHours2 = tmpHours2 >0 ? tmpHours2 - 1 : tmpHours2;
lcd.setCursor(11,1);
lcd.print("  ");
lcd.setCursor(11,1);
if ( tmpHours2 < 10 ) lcd.print(" ");
lcd.print(tmpHours2);
timeRef = millis();
      }
else if ( button == KEYPAD_SELECT )
      {
while ( lcd.button() != KEYPAD_NONE ) ;
timeOut = false;
break;
      }
delay(150);
   }

if ( !timeOut ) transition(KEYPAD_SELECT);
else transition(TIME_OUT);
}
void setAlarmMinutes2()
{
unsigned long timeRef;
boolean timeOut = true;
   uint8_t tmpMinutes2 = 0;
```

```
lcd.clear();
lcd.print("Alarm Time2");

timeRef = millis();
lcd.setCursor(0,1);
lcd.print("Set minutes:  0");

while ( (unsigned long)(millis() - timeRef) < 5000 )
   {
uint8_t button = lcd.button();

if ( button == KEYPAD_UP )
      {
         tmpMinutes2 = tmpMinutes2 <59 ? tmpMinutes2 + 1 : tmpMinutes2;
lcd.setCursor(13,1);
lcd.print("  ");
lcd.setCursor(13,1);
if ( tmpMinutes2 < 10 ) lcd.print(" ");
lcd.print(tmpMinutes2);
timeRef = millis();
      }
else if ( button == KEYPAD_DOWN )
      {
         tmpMinutes2 = tmpMinutes2 >0 ? tmpMinutes2 - 1 : tmpMinutes2;
lcd.setCursor(13,1);
lcd.print("  ");
lcd.setCursor(13,1);
if ( tmpMinutes2 < 10 ) lcd.print(" ");
lcd.print(tmpMinutes2);
timeRef = millis();
      }
```

```
else if ( button == KEYPAD_SELECT )
        {
while ( lcd.button() != KEYPAD_NONE ) ;
timeOut = false;
break;
        }
delay(150);
    }

if ( !timeOut )
    {
        alarmHours2 = tmpHours2;
        alarmMinutes2 = tmpMinutes2;
transition(KEYPAD_SELECT);
    }
else transition(TIME_OUT);
}
void setAlarmHours3() //[32]
{
unsigned long timeRef;
boolean timeOut = true;

lcd.clear();
lcd.print("Alarm Time3");

    tmpHours3 = 0;
timeRef = millis();
lcd.setCursor(0,1);
lcd.print("Set hours:  0");

while ( (unsigned long)(millis() - timeRef) < 5000 )
    {
```

```
uint8_t button = lcd.button();

if ( button == KEYPAD_UP )
      {
        tmpHours3 = tmpHours3 <23 ? tmpHours3 + 1 : tmpHours3;
lcd.setCursor(11,1);
lcd.print("  ");
lcd.setCursor(11,1);
if ( tmpHours3 < 10 ) lcd.print(" ");
lcd.print(tmpHours3);
timeRef = millis();
      }
else if ( button == KEYPAD_DOWN )
      {
        tmpHours3 = tmpHours3 >0 ? tmpHours3 - 1 : tmpHours3;
lcd.setCursor(11,1);
lcd.print("  ");
lcd.setCursor(11,1);
if ( tmpHours3 < 10 ) lcd.print(" ");
lcd.print(tmpHours3);
timeRef = millis();
      }
else if ( button == KEYPAD_SELECT )
      {
while ( lcd.button() != KEYPAD_NONE ) ;
timeOut = false;
break;
      }
delay(150);
   }

if ( !timeOut ) transition(KEYPAD_SELECT);
```

```
else transition(TIME_OUT);
}
void setAlarmMinutes3()
{
unsigned long timeRef;
boolean timeOut = true;
   uint8_t tmpMinutes3 = 0;

lcd.clear();
lcd.print("Alarm Time3");

timeRef = millis();
lcd.setCursor(0,1);
lcd.print("Set minutes:  0");

while ( (unsigned long)(millis() - timeRef) < 5000 )
   {
uint8_t button = lcd.button();

if ( button == KEYPAD_UP )
      {
         tmpMinutes3 = tmpMinutes3 <59 ? tmpMinutes3 + 1 : tmpMinutes3;
lcd.setCursor(13,1);
lcd.print("  ");
lcd.setCursor(13,1);
if ( tmpMinutes3 < 10 ) lcd.print(" ");
lcd.print(tmpMinutes3);
timeRef = millis();
      }
else if ( button == KEYPAD_DOWN )
      {
         tmpMinutes3 = tmpMinutes3 >0 ? tmpMinutes3 - 1 : tmpMinutes3;
```

```
lcd.setCursor(13,1);
lcd.print("  ");
lcd.setCursor(13,1);
if ( tmpMinutes3 < 10 ) lcd.print(" ");
lcd.print(tmpMinutes3);
timeRef = millis();
      }
else if ( button == KEYPAD_SELECT )
      {
while ( lcd.button() != KEYPAD_NONE ) ;
timeOut = false;
break;
      }
delay(150);
   }

if ( !timeOut )
   {
      alarmHours3 = tmpHours3;
      alarmMinutes3 = tmpMinutes3;
transition(KEYPAD_SELECT);
   }
else transition(TIME_OUT);
}
void setAlarmHours4() //[33]
{
unsigned long timeRef;
boolean timeOut = true;

lcd.clear();
lcd.print("Alarm Time4");
```

```
   tmpHours4 = 0;
timeRef = millis();
lcd.setCursor(0,1);
lcd.print("Set hours:  0");

while ( (unsigned long)(millis() - timeRef) < 5000 )
   {
uint8_t button = lcd.button();

if ( button == KEYPAD_UP )
      {
         tmpHours4 = tmpHours4 <23 ? tmpHours4 + 1 : tmpHours4;
lcd.setCursor(11,1);
lcd.print("  ");
lcd.setCursor(11,1);
if ( tmpHours4 < 10 ) lcd.print(" ");
lcd.print(tmpHours4);
timeRef = millis();
      }
else if ( button == KEYPAD_DOWN )
      {
         tmpHours4 = tmpHours4 >0 ? tmpHours4 - 1 : tmpHours4;
lcd.setCursor(11,1);
lcd.print("  ");
lcd.setCursor(11,1);
if ( tmpHours4 < 10 ) lcd.print(" ");
lcd.print(tmpHours4);
timeRef = millis();
      }
else if ( button == KEYPAD_SELECT )
      {
while ( lcd.button() != KEYPAD_NONE ) ;
```

```
timeOut = false;
break;
      }
delay(150);
   }

if ( !timeOut ) transition(KEYPAD_SELECT);
else transition(TIME_OUT);
}
void setAlarmMinutes4()
{
unsigned long timeRef;
boolean timeOut = true;
   uint8_t tmpMinutes4 = 0;

lcd.clear();
lcd.print("Alarm Time4");

timeRef = millis();
lcd.setCursor(0,1);
lcd.print("Set minutes:  0");

while ( (unsigned long)(millis() - timeRef) < 5000 )
   {
uint8_t button = lcd.button();

if ( button == KEYPAD_UP )
      {
         tmpMinutes4 = tmpMinutes4 <59 ? tmpMinutes4 + 1 : tmpMinutes4;
lcd.setCursor(13,1);
lcd.print("  ");
lcd.setCursor(13,1);
```

```
if ( tmpMinutes4 < 10 ) lcd.print(" ");
lcd.print(tmpMinutes4);
timeRef = millis();
      }
else if ( button == KEYPAD_DOWN )
      {
         tmpMinutes4 = tmpMinutes4 >0 ? tmpMinutes4 - 1 : tmpMinutes4;
lcd.setCursor(13,1);
lcd.print("  ");
lcd.setCursor(13,1);
if ( tmpMinutes4 < 10 ) lcd.print(" ");
lcd.print(tmpMinutes4);
timeRef = millis();
      }
else if ( button == KEYPAD_SELECT )
      {
while ( lcd.button() != KEYPAD_NONE ) ;
timeOut = false;
break;
      }
delay(150);
   }

if ( !timeOut )
   {
      alarmHours4 = tmpHours4;
      alarmMinutes4 = tmpMinutes4;
transition(KEYPAD_SELECT);
   }
else transition(TIME_OUT);
}
```