

Smart Medicine Holder

CPEG 480- Capstone Design Project II



Project Members

Sabika Al-Asfour (S00030784)
Ruaa Al-Yaqoub (S00032192)
Naemah Al-Ali (S00032195)
Heba Al-Nasser (S00034543)

Project Supervisor

Dr. Said Esmaili

Department of Electrical & Computer Engineering

AMERICAN UNIVERSITY *of* KUWAIT

May 15, 2019

Smart Medicine Holder

Project Members

Sabika Al-Asfour (S00030784)

Ruaa Al-Yaqoub (S00032192)

Naemah Al-Ali (S00032195)

Heba Al-Nasser (S00034543)

The capstone project report is being submitted in partial

Fulfillment of the requirements for the degree of

Bachelor of Engineering in Electrical/ Computer Engineering

Project Supervisor

Dr. Said Esmaeili

Supervisor's Signature: _____

Department of Electrical & Computer Engineering

AMERICAN UNIVERSITY *of* KUWAIT

May 15, 2019

Declaration

We certify that this project work titled “*Smart Medicine Holder*” is our own work. The work has not been presented elsewhere for assessment. The material that has been used from other sources has been properly acknowledged / referred.

Project

Members

Sabika Al-Asfour (S00030784)

Ruaa Al-Yaqoub (S00032192)

Naemah Al-Ali (S00032195)

Heba Al-Nasser (S00034543)

Plagiarism Certificate (Turnitin Report)

This thesis has been checked for Plagiarism. Turnitin report endorsed by the Supervisor(s) is attached.

Project

Members

Sabika Al-Asfour (S00030784)

Ruaa Al-Yaqoub (S00032192)

Naemah Al-Ali (S00032195)

Heba Al-Nasser (S00034543)

Signature of
Supervisor(s)

Copyright Statement

- Copyright in text of this project report rests with the student authors. Copies (by any process) either in full, or of extracts, may be made only in accordance with instructions given by the authors and lodged in the Library of AUK. Details may be obtained by the Librarian. This page must form part of any such copies made. Further copies (by any process) may not be made without the permission (in writing) of the authors.
- The ownership of any intellectual property rights which may be described in this project report is vested in AUK's Department of Electrical & Computer Engineering, subject to any prior agreement to the contrary, and may not be made available for use by third parties without the written permission of the AUK's Department of Electrical & Computer Engineering, which will prescribe the terms and conditions of any such agreement.
- Further information on the conditions under which disclosures and exploitation may take place is available from the Library of AUK, Kuwait

Acknowledgements

This page can be used to acknowledge any technical or financial support and collaboration with academic, industrial or funding partners, support from supervisor, other faculty members, and/ or other fellow students. Students may also include acknowledgements of family members. Font, line spacing, and page margins must not be changed throughout this section and other sections in the thesis. Example texts for acknowledgements are given below.

Sabika Al-Asfour (S00030784)

Ruaa Al-Yaqoub (S00032192)

Naemah Al-Ali (S00032195)

Heba Al-Nasser (S00034543)

“Dedicated to our parents, family members and friends who supported us and cooperated leading us to produce this accomplishment.”

Abstract

The study of medicine is evolving just like any other scientific field in the world; as the study is evolving, it is becoming more versatile and with growing fields and study, testing, and experimentation. The progression of these studies has helped compose and construct new types of medications for different types of diseases. With the improvement of medicine, we shed the light on another aspect, the dispenser of the pills, including the time, days, and doses.

The idea behind this project is to create a fully functional medicine holder that can dispense the correct medicine at its specified time. The medicine holder is mainly targeted towards the elder citizens, the blind and the disabled, where this is at least one feature within the smart medicine holder that aids each targeted population of this project. Hard thinking and planning went into the special features of this medicine holder in order to maintain the highest standards and the most effective performance. The smart medicine holder is equipped with a sound system that alerts that patient before and during the time of the specified medicine; as well as a complex system that assures that no mistake will occur when dispensing multiple pills of different types of pills at the same time. The medicine holder is also made with special type of material that is known to be antibacterial; this feature guarantees that the pills are kept clean and hygienic as well as many other features that are discussed within the report. The smart medicine holder is able to dispense different types of pills from different containers at the same time. The team worked on making the overall system very user-friendly so that no user encounters difficulty programming the settings to their specified medical regime.

The main purpose of this project is to organize the patient's dose intake by alerting the user when the medicine's time comes. The holder also organizes the pill intake by dispensing the exact number of pills at their specified time. This in turn will avoid mixing up of the pills or neglecting the prescriptions.

Key Words: *Medicine holder, Sanitary pills, servo motor.*

Table of Contents

CHAPTER 1: INTRODUCTION	13
1.1 Problem Statement.....	13
1.2 Solution	14
1.3 Goals	15
1.4 SWOT analysis	15
1.5 Conclusion.....	16
CHAPTER 2: LITTERATURE REVIEW.....	17
2.1: Smart Pill Dispenser For Dependent People [1].....	17
2.1.1 General Overview.....	17
2.1.2 Implementation	17
2.1.3 Performance	22
2.2 Construction of a Smart Medicine Dispenser with High Degree of Scalability and Remote Manageability [2]	22
2.2.1 General Overview.....	22
2.2.2 Implementation	23
2.2.3 Performance	26
2.3 Medicine Container [3]	27
2.3.1 General Overview.....	27
2.3.2 Implementation	27
2.3.3 Performance	28
2.4 Smart Pill Box [4].....	28
2.4.1 General Overview.....	28
2.4.2 Implementation	29
2.4.3 Performance	31
2.5 Smart Medicine Box [5].....	31
2.5.1 General overview	31
2.5.2 Implementation	33
2.5.3 Performance	34
2.6 Intelligent pill box [6]	35
2.6.1 General overview	35
2.6.2 Implementation	35
2.6.3 Performance	36
2.7: SMART MEDICINE CONTAINER [7].....	37
2.7.1 General Overview.....	37
2.7.2 Implementation	37
2.7.3 Performance	39
2.8 Our Project	40
2.9 Comparison between the Implemented Project and Our Project	41
2.10: Conclusion.....	41
CHAPTER 3: METHODOLOGY, DESIGN AND ANALYSIS	42
3.1 Requirements	42
3.1.1 Functional Requirements.....	42
3.1.2 Non-Functional Requirements.....	43
3.2 System Architecture.....	44
3.3 Software.....	46
3.4 Hardware Components.....	46
3.4.1 Arduino Leonardo	46
3.4.2 Red Board	47
3.4.3 Arduino Uno (R3)	47
3.4.4 Lily Pad Arduino	47
3.4.5 Arduino Mega (R3)	47

3.4.6 Team's Decision	48
3.4.7 RTC	48
3.4.8 Temperature and humidity sensor	49
3.4.9 MZ80 IR sensor	49
3.4.10 MP3 DF player mini	50
3.4.11 Push button.....	50
3.4.12 LCD Screen	50
3.4.13 Easy driver	51
3.4.14 Speaker	51
3.4.15 Servomotor	51
3.4.16 Stepper motor.....	52
3.4.17 Switch	52
3.5 Budget.....	53
3.6: Conclusion	53
CHAPTER 4: IMPLEMENTATION	54
4.1 Sound system	54
4.2 MP3 and card	55
4.3 The smart medicine holder	55
4.4 The Stepper motor mechanism	55
4.6 The six alarms	56
4.7 Temperature and humidity sensor	57
4.8 Final Implementation.....	58
4.9 Conclusion.....	62
CHAPTER 5: EVALUATION	63
5.1 Economic aspect.....	63
5.2 Environmental impact.....	64
5.3 Social Impact	65
5.4 Survey	65
5.5 Engineering Ethics.....	70
5.6 Project Evaluation.....	71
5.7 Conclusion.....	72
CHAPTER 6: CONCLUSION AND FUTURE WORK	73
6.1 Summaries.....	73
6.2 The Current Progress	74
6.3 Future Work	74
6.4 Problems faced.....	75
REFERENCES.....	76
APPENDIX A.....	77

List of Figures

Figure 1: Pill storage [1].....	19
Figure 2 : Pill Hatch [1].....	19
Figure 3:Pill chamber [1].....	20
Figure 4:Pill pipe[1].....	20
Figure 5: Management operations.....	23
Figure 6: Overview of the medication monitoring system.....	23
Figure 7: Hardware architecture of the smart medicine dispenser.....	24
Figure 8: Software architecture of the smart medication dispenser.....	25
Figure 9: Prototype of smart medication dispenser.....	26
Figure 10: Flow chart of operations.....	30
Figure 11: Smart pill box device from the back.....	32
Figure 12: Smart pill box device from the front.....	32
Figure 13: High level block diagram.....	33
Figure 14: Software design flowchart.....	34
Figure 15: Intelligent pill box flowchart.....	36
Figure 16: The collecting conveyor.....	38
Figure 17: The counting compartment.....	39
Figure 18: System Architecture of our project.....	45
Figure 19: The type of Arduino we used.....	48
Figure 20: Real time clock used in our project.....	48
Figure 21: The temperature and humidity sensor.....	49
Figure 22: MZ80 Sensor.....	49
Figure 23: Push buttons used to enter data input.....	50
Figure 24: LCD screen used.....	50
Figure 25: Stepper motor driver.....	51
Figure 26: Speaker to output the alarms.....	51
Figure 27: Servo motor.....	52
Figure 28: Stepper motor for the cycles.....	52
Figure 29: Switch for turning on and off.....	52
Figure 30: Audio converter.....	54
Figure 31: Overall view of the Smart medicine holder.....	58
Figure 32: Side view of the holder (switch, Adaptor, injections holder).....	58
Figure 33: Side view of the holder (LED, speaker).....	59
Figure 34: Back view of the holder (the circuit).....	59
Figure 35: LCD screen and push buttons.....	60
Figure 36: The medicine containers.....	60
Figure 37: IR sensor and opened dispenser basket.....	61
Figure 38: Results of the first question.....	65
Figure 39: Results of the second question.....	66
Figure 40: Results of the third question.....	66
Figure 41: Results of the fourth question.....	67
Figure 42: Results of the fifth question.....	67
Figure 43: Results of the sixth question.....	68
Figure 44: Results of the seventh question.....	68
Figure 45: Results of the eighth question.....	69
Figure 46: Results of the ninth question.....	69

List of Tables

<i>Table 1: Project Swot Analysis</i>	15
<i>Table 2: Team Swot Analysis</i>	16
<i>Table 3: Comparing our project with other projects</i>	41
<i>Table 4: Functional Requirements</i>	42
<i>Table 5: Non-functional Requirements</i>	43
<i>Table 6: Budget of our project</i>	53
<i>Table 7: Table of standards and the status of completion</i>	71

CHAPTER 1: INTRODUCTION

Advances in medicine are enhancing the longevity and quality of the human life; various types of different viruses and diseases are now treated using multiple drugs. In many cases, patients have to remember to take a dose of medication at a prescribed time. Many treatments require taking a combination of one or more types of medication including pills, injectable or liquids. Following the prescribed treatment is essential in assuring full recovery to the patients, but the different combination of medication within the treatment can get confusing for some while some can simply forget to take the pill at the prescribed time therefore decreasing the effectiveness of the treatment.

The elderly, the blind, and people with mental health issues make up most of the categories that may not be able to follow simple instructions concerning their treatment. Failing to follow the instructions of their medication might not only extend their recovery but will be a reason for more health related problems. Care takers are assigned to patients but those care takers may have more than one patient to attend to, meaning that this care taker may be exposed to forgetting to give the medicine or simply mixing up the patient's treatments so even the care takers sometimes needs assistance.

Failing to follow the instructions of one's treatment is a serious problem and is the primary inspiration that triggered the idea behind this project. The smart medicine holder will be the reason why most users if not all follow their treatments and hopefully recover from their health problems. The purpose of this project is to provide all necessary features to the caretakers and the users themselves to assure their medical regime is followed properly.

1.1 Problem Statement

According to the World Health Organization, over 80% of people above the age of 60 years are prescribed medicines that are to be administered two to four times a day. With the intense treatment a patient has comes the great importance of remembering to take the medication at the right time with the right quantity in order to assure full recovery for the patient.

Old people have a hard time remembering when to take the medication, what the required dose is or if they even took the medication in the first place, this is the primary reason why people of old age are assigned care takers. The job of those care takers is to assure that the patient benefits entirely from the prescribed treatment, this primarily relays on taking the medication in the specified timings.

Patients with special needs like the blind or people with mental health problems usually face this problem where they cant take the specified dose just because they don't know which medicine they have at hand specially when they are prescribed more that one type of medicine, so it is common for them to mix up their medicine, take one before the other or take the same one over and over again, this elongates and hardens the healing process of the patient and might probably create more health problems because of the inconsistency in medicine intake. Therefore, the elderly, the blind, special needs patients and many more will be exposed to various health problems if they don't abide to the treatment properly, this is the main objective and inspiration for this project, the smart medicine holder.

1.2 Solution

The smart medicine holder has been around for some time, people throughout time update the medicine holder by adding features that help ease following the treatment's instructions. The main purpose of this project is to enhance the medicine holder, add new features using technology in both the electrical and computer engineering departments in order to attain the most optimum medicine holder.

The medicine holder proposed has separate storage compartments where each type of medicine is stored separately; the compartments are equipped with temperature and humidity sensor used to ensure that the medicine is stored according to the specified temperature. The compartments also contain a feature that is directed to the blind, the smart medicine holder compartments has speaker sound system that is activated at any time the compartment has been opened. The main objective of having the sound system is to inform the medicine container's user of what compartment has been opened and what medicine it contains.

The medicine container has a simple system using push button switches for the users for the purpose of entering data into the system, data like the time of the first dose taken, and how many times must the medication be taken.

1.3 Goals

The main goals of this project are:

- To aid the elderly, the blind and the special needs in maintaining a stable medical regime.
- To alert the caretaker and user of the specified medication timing
- To connect the caretaker and user to the smart medicine holder in order to keep track of the required treatment.
- To keep all the prescribed medicine whether it is prescription pills or injectable in a safe place monitored with temperature and humidity sensor.
- To dispense one pill at the specified time.

1.4 SWOT analysis

Table 1: Project Swot Analysis

Project SWOT Analysis	
<p style="text-align: center;">Strengths</p> <ul style="list-style-type: none"> • Combines several components in one machine. • Convenient • Relatively fast 	<p style="text-align: center;">Weaknesses</p> <ul style="list-style-type: none"> • Speed of the container • Accuracy • Energy consumption, and acceptability • Lifetime, data fidelity, and user's comfort
<p style="text-align: center;">Opportunities</p> <ul style="list-style-type: none"> • New design (currently unavailable) 	<p style="text-align: center;">Threats</p> <ul style="list-style-type: none"> • Including the components in one machine

Table 2: Team Swot Analysis

Team SWOT Analysis	
Strengths <ul style="list-style-type: none">• Ability to work under pressure• Stick to deadlines• Good communication skills• Highly active workers	Weaknesses <ul style="list-style-type: none">• Difficulty in meeting because of different time tables• Limited budget
Opportunities <ul style="list-style-type: none">• The chance to gain experience in assembling and coding in general	Threats <ul style="list-style-type: none">• Arriving the components late

1.5 Conclusion

In this chapter we introduced the problem faced by many people when they are prescribed medication. Our project addressed the main needs of deaf or old people when they are prescribed medication and they have to take many pills in different times. Then we introduced the solutions we came up with to solve the problem of medicine time management. We defined the goals of our project and we verified the reasons why the project is important.

CHAPTER 2: LITTERATURE REVIEW

Patients of various diseases have shown the tendencies to have hardship in following the doctor's prescribed medicine course leading to not obtaining the ultimate benefit and total recovery from that disease. The idea of this project is to create an efficient medicine holder that aids old people, the blind, people with psychiatric problems and to help the doctors, nurses and caretakers within hospitals. This medicine box will be programmed based on the patient's personal and medical data where information like the number of courses and medicine pills to be taken is recorded and an alarm is set to notify the patient or the nurses when the pill is due.

2.1: Smart Pill Dispenser For Dependent People [1]

2.1.1 General Overview

Smart Pill Dispenser for Dependent People was designed by Guillermo de Juan Grau[1]. This pill dispenser is used to prevent forgetting when and what medicines should be taken at a certain time which can be problematic for most people like the elderly, the blind and people with illness due to the negative effects that they cause to themselves or others. For example, the death behind giving pills to the wrong person or at wrong time. Therefore, Guillermo de Juan Grau came up with this creation to do the things that has not been done accurately. First, his smart pill dispenser is designed in an easy and simple way that requires the owner or any dependent person to only log the name of the pill, numbers and hours at which each pill is taken versus the time it should be taken, and the device should be connected to the wireless network to be powered on.

2.1.2 Implementation

The pillbox will possess the physical characteristics which are a big, easy to see button, one or various LEDs to signal different working modes or notifications, a small simple speaker which will be used as an acoustic notifier, a small hole from which the pills will be dispensed at the correct time, one or more power/data ports from which the pillbox will be charged and some other functions may be implemented, a small hidden buttons for pillbox, a removable cap from where to access the pill compartments, allowing for a quick and

efficient refilling, a charger specially made to charge the pillbox, and a program which will allow management functions for the pillbox.

Pillbox will work under three different working modes: normal, charging or refill modes and management modes. During the normal mode the pillbox remains in standby till the user must take a pill/dose. Charging mode will initially be similar in every possible way to normal mode for the exception of a different color led which will be on to indicate that charging is underway. Charging mode will only happen when the pillbox is inserted into its charging base. The final mode is management mode, which will be engaged when the pillbox is being managed by the management program installed on a computer. The connection is done wirelessly and once the device enters management mode the charging LED will blink twice at the start to indicate the pillbox has established correctly the connection with the managing program.

The mechanism design of the pill dispensing consists in three different stages: default stage, the second stage being the pill load stage and the third stage, which is the pill release stage. After the third stage, the system will reset or go back to its default stage. The system is activated every time a pill has to be taken, starting on the first stage, the default stage, then the pill get loaded and finally dispensed for the user to take. The four parts involved in the dispensing mechanism are pill storage, pill hatch, pill chamber and pill pipe.

All of the parts of the pill storage on the top of it there is a circular hole where the cables for circuits will pass when the pillbox is assembled. These compartments will be storing and separating the pills of the same type, refer to figure 1. The second part is pill hatch, which will be a fixed part which allows communicating the pill storage part through a small hole with the pill chamber. It consists on a very thin circular piece with an opening big enough for a pill to pass through it and a small ramp connecting the opening to the upper level of the cylinder, as shown in figure 2.

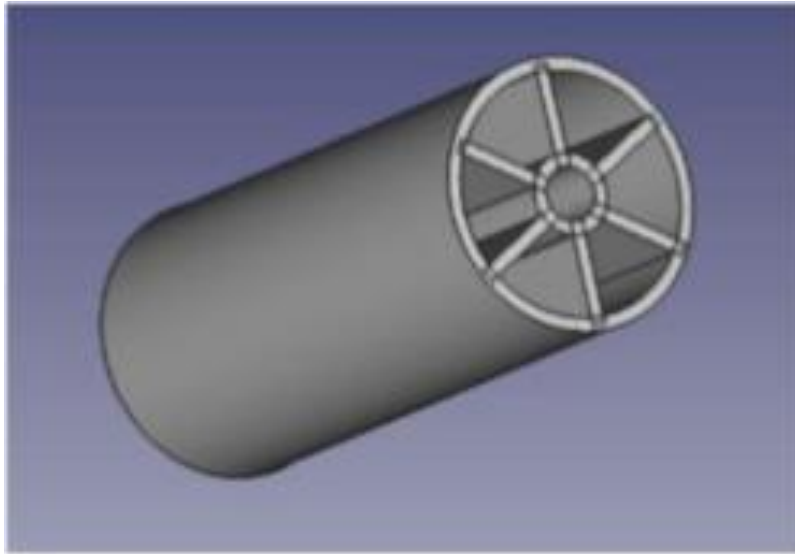


Figure 1: Pill storage [1]

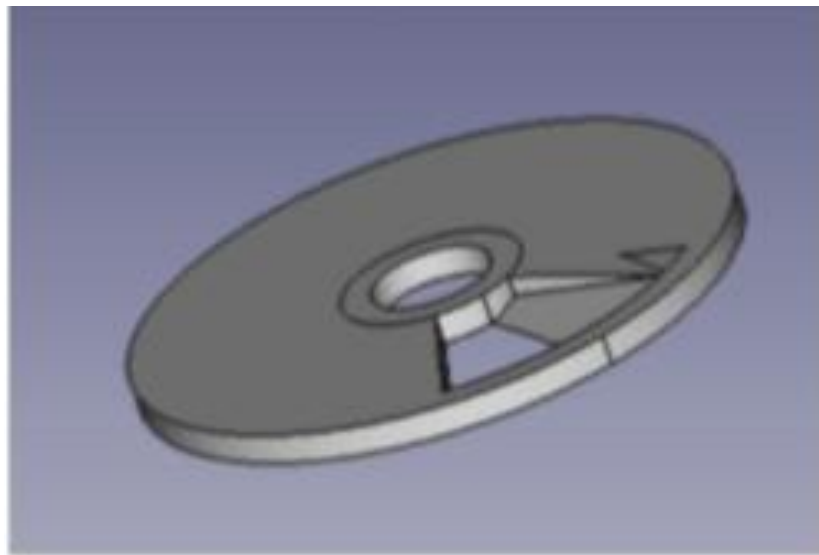


Figure 2 : Pill Hatch [1]

The pill chamber consists of a cylinder with a circular hole at the center and another passage beside the hole. This passage is wide enough to hold a pill in an upright vertical position. The height also is the same as the model pill which will be used in this pillbox, therefore this part can only hold one pill standing in an upright position in its storage space, as shown in figure 3. Finally, The pill pipe part consists of a semi-circular piece with a large pipe of a considerable greater height than the circular part protruding from its side, connecting with a hole the upper part of the circular part with the end of the pipe, see figure 4. All of the parts have a circular opening, which is helpful when the cables are being inserted into the pillbox during assembly.

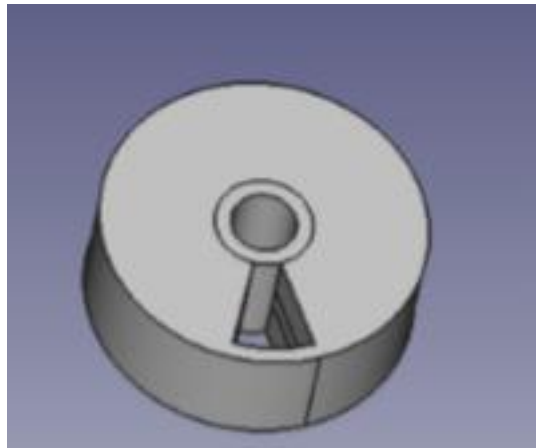


Figure 3: Pill chamber [1]

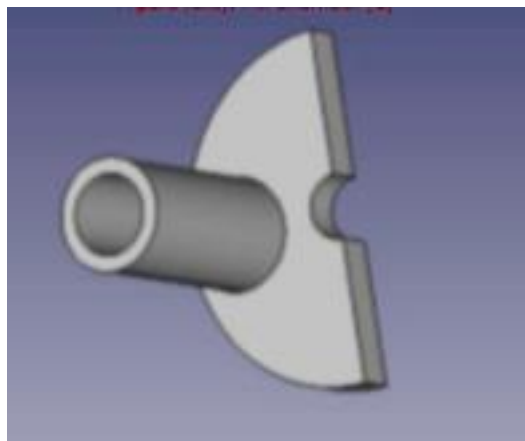


Figure 4: Pill pipe[1]

Moreover, the different parts of the device were designed using open-source software with no cost assigned to it. This software is the program Free CAD. The creator used many electronic devices that were mentioned and will perform different functions that are Arduino, servos, LEDs, Speaker, RTC (real-time clock) module, Bluetooth module, Arduino's EEPROM memory, and External battery.

The Arduino selected for this project is a Funduino Pro Mini, a version of the Arduino Pro Mini, which comes at a lower price offering the same functionality as the Arduino Pro Mini board. The author soldered the board because it comes without the head pins, which is needed for uploading the code and powering up the device, as well as pins A4 and A5 which are needed for the communication with the Real Time Clock module. The Arduino controls the device and different electronic modules. Two servos motor are used in this creation to help in rotating the inner parts of the pillbox.

The Bluetooth module that was chosen for the project is the official Arduino Bluetooth shield, which has 4 pins, 2 for powering the device one for receiving data from the Arduino board and sending it via Bluetooth and another one for transmitting the data received from Bluetooth to the Arduino board.

An RTC module was acquired, a Tiny RTC Module, which communicates, with the Arduino board via I2C communications and keeps track of time. Another electronic device EEPROM memory this project uses (Arduino Pro Mini) has 1kB of memory. A single entry in the pill schedule will take up a total of four bytes in memory. The Arduino Pro Mini memory will keep logs of different pills intake. Furthermore, the piezo speaker is connected directly to an output pin and can control the acoustic notifications while the LEDs, the switches and buttons will relate to 220 ohms resistors to either output or input pins. Last electronic device was required for this project is the battery used. The chose battery is the LiPo battery that will be connected to the RAW and the GND pins in the Arduino board powering it. This battery will be able to power the pillbox for at least a day and a half approximately.

2.1.3 Performance

Unfortunately, the smart pill dispenser for depending people was not implemented as a real-life functioning apparatus. Therefore, factual numbers concerning power consumption, maintenance, rate of failure, or operating time could not be achieved. However, Guillermo de Juan Grau made a small prototype, which worked very successfully.

2.2 Construction of a Smart Medicine Dispenser with High Degree of Scalability and Remote Manageability [2]

2.2.1 General Overview

The Construction of a Smart Medication Dispenser with High Degree of Scalability and Remote Manageability was invented by JuGeon and KeeHyun Park. In this project, they Considered that users of medication dispensers are typically in the older age group or are mostly patients with chronic diseases, so they come up with an invention trying to achieve and solve the problems caused by other medication dispensers. The dispenser operates as follows: when the clock reaches the predetermined medication time and the user presses the dispense button at that time, the predetermined medication is dispensed from the medication dispensing tray (MDT). In the proposed dispenser, the medication for each patient is stored in an MDT. One smart medication dispenser contains mainly one MDT, but it can be extended to include more MDTs to allow medical staff and system administrators, instead of end users, to manage medication dispensers, thus leading to cost efficiency and safe operation of the device. For remote management, the smart medication dispenser corrects a patient's medication state and transmits the corrected data to the medication-monitoring server. Further, system settings, embedded programs, and operational errors can be remotely managed by medical staff and system administrators figure 5.

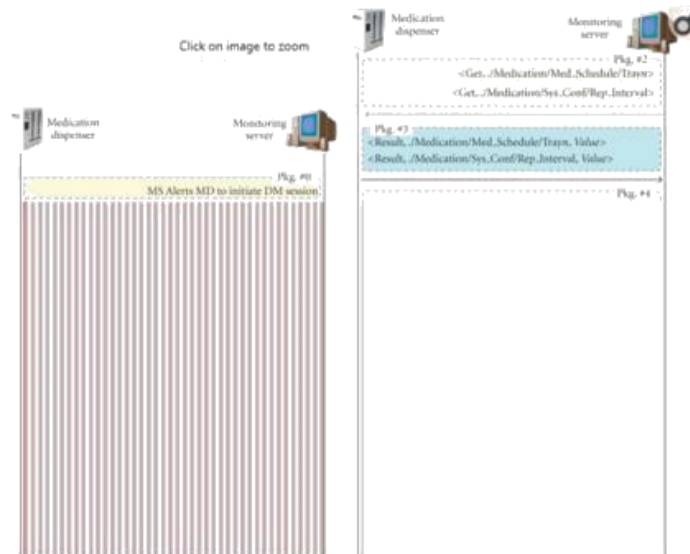


Figure 5: Management operations

2.2.2 Implementation

The Construction of a Smart Medication Dispenser with High Degree of Scalability and Remote Manageability can accomplish a quick and a safe procedure at a much higher efficiency and speed than what any other machine could do before, due to its working mechanism and the components used in its operation. It uses a variety of components, including, a component of a medication monitoring system. The medication monitoring system is comprised of smart medication dispensers and a medication-monitoring server. Figure 6.

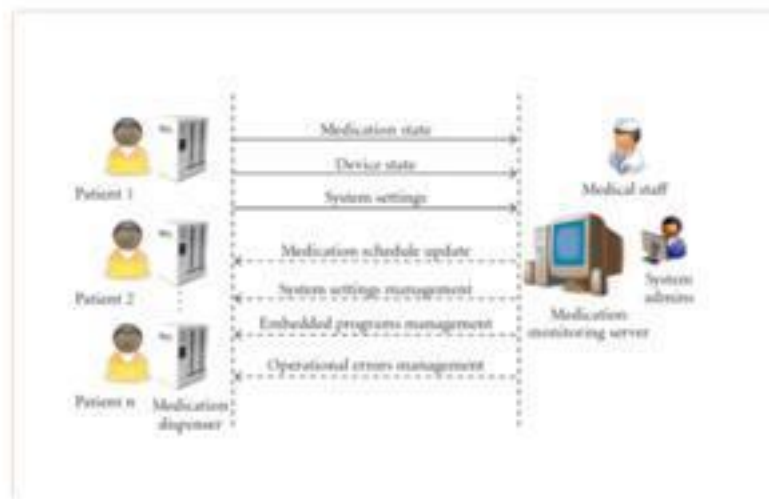


Figure 6: Overview of the medication monitoring system

The server medication monitoring, which the dispenser sends the data to, analyzes device state and system settings and generates operations for managing system settings, embedded programs, and operational errors. Further, the smart medication dispenser developed through Hardware architecture and software architecture. For the hardware architecture, it was developed on the WinCE platform figure 7.

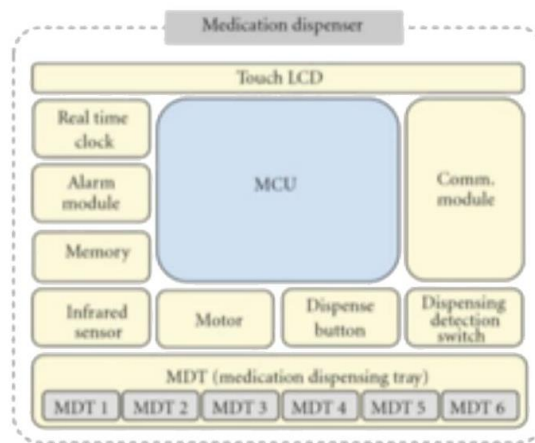


Figure 7: Hardware architecture of the smart medicine dispenser

Figure 7 has divided into eight parts, and each part is responsible for a certain thing in the medication dispenser. The MCU controls all system functions, Touch LCD displays the medication information and schedule, and the MDT is the container of the pills to be taken. Alarm module reminds the user by a buzzer that it is time to take pills. Dispense button it works only one-time during predetermined cycle period and it is used to dispense the user's medication. Infrared sensor checks how many pills left in the MDT, and if the pills is running out it sends message to the monitoring server. Real-time clock ensures the machine alerts the user to take its pills at the right time.

Lastly, the communication module is used to communicate with a MS or PC. RS232 Serial communication and a local area network (LAN) are provided.

Moving to software Architecture of the smart medication dispenser, if some problems occur such as a shortage of medication, medication jam, memory overload, software error, or non-adherence occurs, the problem is transmitted immediately. All these operations are conducted automatically without the intervention of patients through software installed in the smart medication dispenser Figure 8.

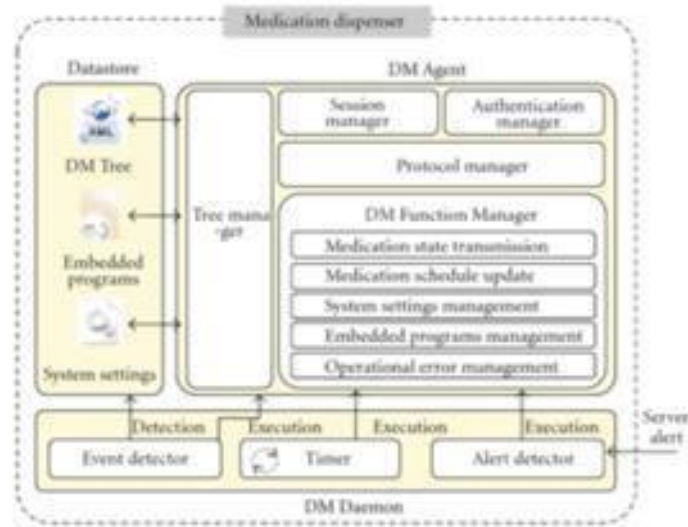


Figure 8: Software architecture of the smart medication dispenser

The figure divided into three main parts: DM Agent, Data store and DM Daemon. The DM Agents manages the dispenser according to the operations of the medication-monitoring server and it consists of Session Manager, Authentication Manager, Protocol Manager, DM Function Manager, and Tree Manager. The Session Manager is ensuring the connections with the server till the session done. The Authentication Manager provides the user's information when connected to the server and confirm who the user is before operating the device. The Protocol Manager set up a message from the user's pills and device statues to evolve management operations from a received message.

2.2.3 Performance

Unfortunately, the Construction of a Smart Medication Dispenser with High Degree of Scalability and Remote Manageability could not be implemented as a real-life functioning apparatus. Therefore, factual numbers concerning power consumption, maintenance, rate of failure, or operating time could not be achieved. However, JuGeon and KeeHyun Park made a small prototype that worked very successfully figure 9.

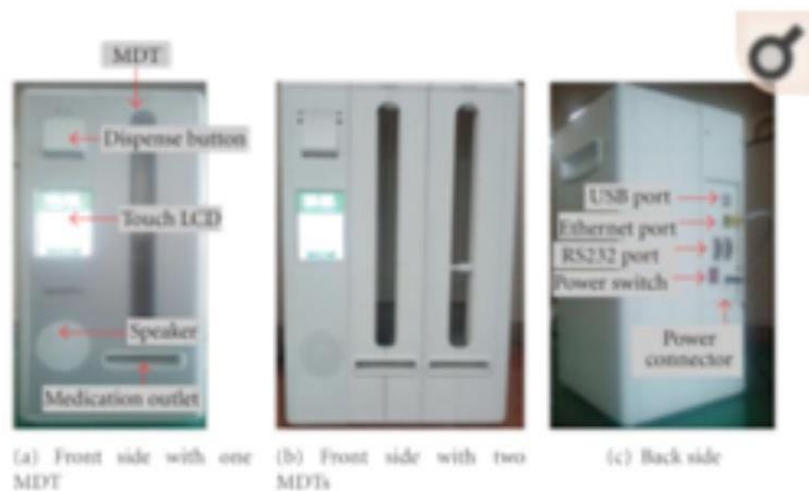


Figure 9: Prototype of smart medication dispenser

This button may have the function of scrolling down the contents list that cannot be shown fully on the display element. The actuating button can be a piezo key that supplies energy to the display element. The piezo key actuation may help by updating the display or the data related to the medicine at the communication interface. The medicine container does not need a power supply to function.

2.3 Medicine Container [3]

2.3.1 General Overview

The Medicine container was invented by Axel Gerlt, Furth (DE); Robert Kagermeier, Nurnberg (DE), and they received the patent for their invention on the 8th of May 2008 [1]. The Medicine container is used to let patients remember to take their medicine in the right time and amount. People with lots of medications distributed all over the day with different timings and quantity may forget to take their medication that can be problematic for most people, due to the negative effects that they may cause to their health. The medicine container operates using a specific mechanism that helps the patient to organize their medicine intake throughout the day.

2.3.2 Implementation

The medicine container has embodiments and each one has a specific task it includes at least one closable compartment that accommodates the medicine. There is a memory element that has the task of storing data related to the medications taken. There is also another element that can do the same task and this is the display element that is made of digital paper. The data stored can be seen on the display using digital paper in a simple manner and in real time. A benefit of Digital paper is that it may be an energy-saving display of a foil-type character. The other benefit is that it does not include liquid crystals. Electro chromic displays include a current-controlled electro migration between two control electrodes. The medicine-related data that is to be shown on the display maybe prespecified as it may include the time at which the medicine is to be taken and/or the dose meaning the quantity. This display can be used by the patient him self or the nursing staff for disable patients. An electro chromic display may include either a plate or foil-type carrier. This carrier includes electrically- conductive plastic and small spheres with hdyeparticles that react chemically to an electrical voltage; this is called electro chromic effect. Though the voltage and in an application of direct current the color of the electro chromic will change. In one embodiment there must be at least one actuating button that is placed on the medicine container.

Finally, The DM function manager of provides the five functions of medication state transmission, medication scheduling, system settings, embedded programs management, and operational errors management.

Moving to Data store, the DM Tree (Tree file), embedded programs, and system settings (Configuration file) are stored in the Data store. However, if an event occurs, then the DM Daemon runs the DM Agent to establish a management session with the medication-monitoring server.

2.3.3 Performance

The medicine container was only theoretical not in real life terms they didn't implement it or test it and saw how it works. The article specified all the implementation part and what are the benefits of this device if it was done in real life. It would have helped hospitals, nursing staff at home for disabled people or even normal patients to help them organize their medication intake.

2.4 Smart Pill Box [4]

2.4.1 General Overview

The Smart pillbox was designed by Aakash Sunil Salgia, K. Ganesan and Ashwin Raghunath and was on January 2015. The remembering to take the medication at the right time and quantity is an important factor for old and even young people where some patients are prescribed medicine that needs to be taken two to four times a day. There is a device to help solve this problem that is the intelligent pillbox. This means there is an available automated alert system for the chemist to send the re-fill of the tablets. The GSM technology will be used and by using it there can be connections to most people around the planet. The benefit of GSM being used in this device is that it operates as a link between the patient and the chemist and can be anywhere in the world as the device is portable.

2.4.2 Implementation

The normal alarm in a pillbox that is used in the market doesn't have the function of checking for overdose and wrong dosage among the patients. It only uses a clock, and setting the time to generate the alarm. The timely alerting for the re-filling of the pillbox to the patient using it is also absent resulting often in breaks in the course of therapy. Regarding the Sensing methodology and based on various references the slot sensing is used depending on three main methods:

- Milligram measuring Load Cell

- Light based sensing

- Capacitive based sensing

The characteristics are firstly checked within the load cell by analyzing and studying the bigger version. This will be the most accurate method provided as it is connected within the center of the box. The light based sensing is the second method used and it is relatively cheap and more suitable method of sensing. Next each slot will be sensed with one sensor individually and the sensor used is Light Dependent Resistor (LDR). Maximum output voltage will be generated from the light blockade of the LDR. The tablets placed in the slot will produce a varying dielectric value leading to varying frequency than when it is empty. The sensor gives a pulse output with variable frequency for varying capacitances of the sensor. This frequency output can be monitored by a microcontroller that will count the corresponding number of pulses in a stipulated amount of time. Figure 10 shows the flow of operation of the system.



Figure 10: Flow chart of operations

The Software algorithm is that the smart pillbox is designed to assist the users who are generally in the age group of a 50+ who are most likely to forget their periodical medicinal intake. There are 3 basic modes for this system:

- Set Time And Date
- Set Medicine Frequency And Alarm
- Display Mode

The Set Time and Date mode function is used to let the user adjust the time and date of the display system according to which country the user. Is In this option, the values are incremented by using a loop and check function. The set medicine frequency module is used to enable the user to set the frequency of the medicine in a day and also to set the time for the alarms.

2.4.3 Performance

This system was tested in office and home environments and passed the test easily; it was tested for 10mg tablets. Below 10mg, the results were marred by interferences from the surrounding. For a higher sensitivity sensor, the result was in cross talk between the sensors. The milligram measuring Load Cell is expensive making it and not suited for a commercial product. Limiting network problems or engaged/busy lines of communication can be done by this system. These additional solutions will be more expensive and unfeasible keeping in mind the portability of the system. The development of an advanced technology called the Smart pillbox is the solution of this complication. Combining the two vintage experiments derives this technology. The smartness of this pillbox is achieved using cost effective slot sensing techniques; this is like capacitance-based slot sensing. These simple techniques are supported by GSM technology to bridge the gap in communication between the chemist and the patient.

2.5 Smart Medicine Box [5]

2.5.1 General overview

This smart medicine box was designed by two students Mingyuan Huang and Jie Zhang from Cornell University as a project for their Digital System Design micro controller class. The medicine box is targeted for users who regularly take drugs or vitamin supplements or nurses who take care of elder patients. The medicine box is programmed for different users by the nurse or the user himself. The pillbox can be programmed for different users according to their health conditions and needs. It is divided into seven sub-boxes to represent the seven days of the week and it alerts the user who intakes the medicine by light and sound signals. The quantity intake and instructions are displayed on a seven segment LED. The main advantage of this medicine box is that it is not necessary to fill the pillbox daily but weekly.

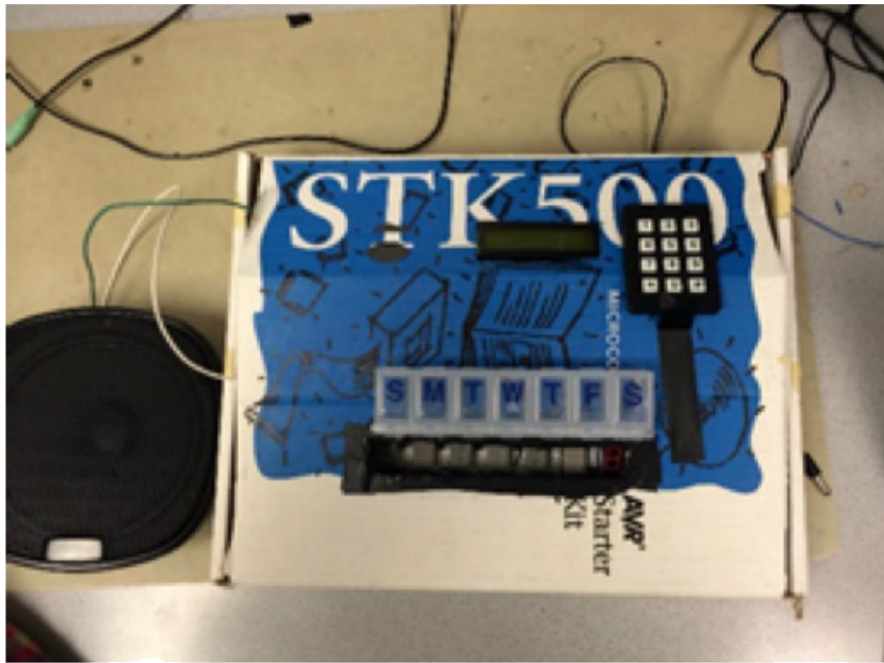


Figure 11: Smart pill box device from the back

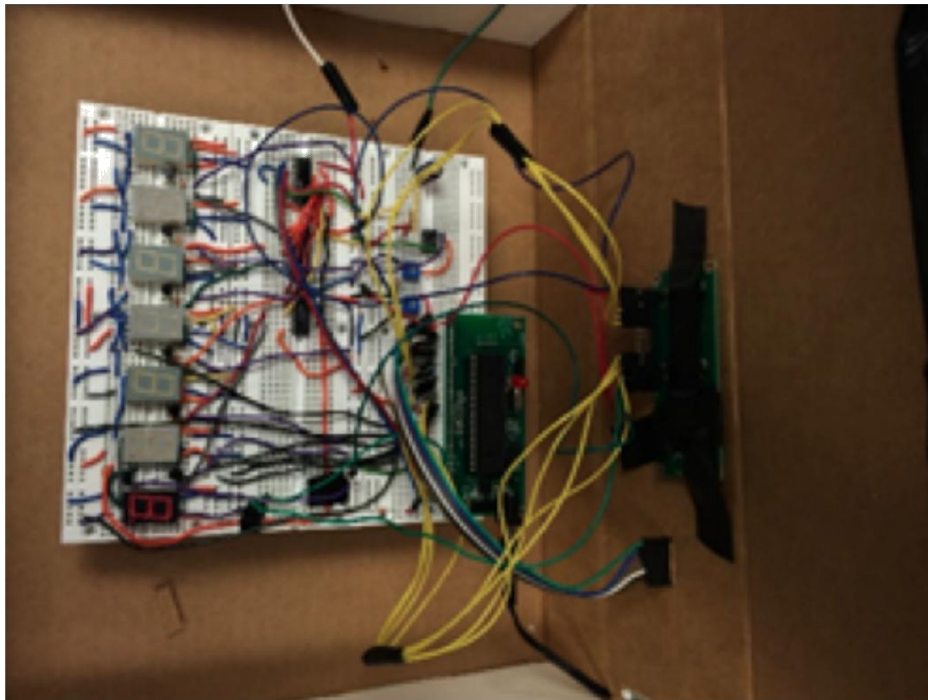


Figure 12: Smart pill box device from the front

2.5.2 Implementation

High-level design

There are five major components in the device, a speaker module; a 3x4 keypad; an Atmel 1284p microcontroller; seven segment LED display; and a 2x16 characters LCD screen. The device uses state machine and real time clock to provide the functionality of the real-time. The state machine determines the key that has been pressed. The device uses a 16MHz external oscillator as a real time clock for the device. The real time clock is used also as an alarm clock in the device.

The device has three major stages in its logical structure: 1- user initialization stage, 2- comparison stage, 3- reminder stage. In the first stage (user initialization), the user enters the current time, date and pill information. Next is the comparison stage, where the system compares the pill information for each sub-box with continuously emits sound, and the seven segments LED screen will display the dosage for the user.

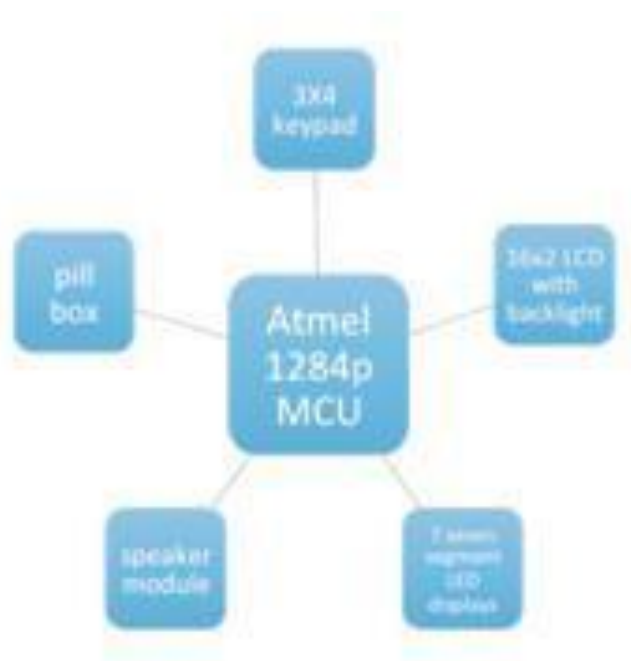


Figure 13: High level block diagram

Software design

The machine box is built with an integrated software system running in the MCU. The programming platforms used are: 1- AVRStudio, 2-the programming language is the standard C, 3- WINAVR/GCC compiler is used. The software system of the device can be divided into, real time clock, user interface, LED control and sound generation. The input information is stored in structural variables. For reach sub- box there is a structure to store information. The structure has four variables that indicate the day a medicine should be taken, the number of times per day the pill should be taken and the dosage that should be taken each time. The user interface consists of two main components, user input and system output. The user input method is the keypad typing and the system output methods consist of one of the three following: 1- LCD displays, 2- LED digits display, 3-audio broadcast.

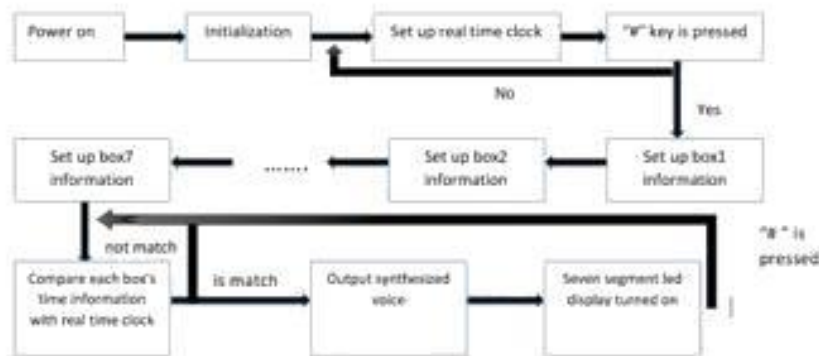


Figure 14: Software design flowchart

2.5.3 Performance

The overall performance of the medicine box was good, even though the design of the device had a paper box that was large as shown in the figure above but its design helped cover all the wires and electrical components so that the user is not confused by all the mess. The large paper box helped as it insulated electricity. The LCD used displays characters with the yellow backlight allowing the user to read even in the dark environments. The keypad responded accordingly when buttons were pressed therefore it was successful. The seven-segment display was embedded

on the breadboard, the light intensity varied in the lab with bright lights, but this didn't affect the message itself. The sound was also clear and it alarmed users as programmed and is stopped when the # key is pressed.

2.6 Intelligent pill box [6]

2.6.1 General overview

This device is designed by Kulkarni. The main reason behind creating this device is patient non-compliance problem, which is an important problem that is key to the health of millions of patients. The SmartPillBox device offers a solution to the patient non-compliance issue. The device helps keep track of how many pills the patients have taken and the timing of the dosage intake. The main functionality of the SmartPillBox device is based on a weight and a microprocessor which keeps track of the time and the pills left at any time in the device.

The device stores the patient's schedule for taking medication. The SmartPillBox uses the information to determine whether the patient is following his treatment or not. If he is not following his treatment the SmartPillBox will remind him to take his medication. Nurse or user fills the SmartPillBox with the prescribed medicine and the schedule for the medicine intake is stored in the device. The weight of the pills is measured by the device automatically and is stored. Each time a pill is taken, the SmartPillBox registers the change in weight. This information is used to mark the time the pills were taken and the change of weight of the remaining pills will be calculated to determine how many pills were taken.

The SmartPillBox reminds the patient to take his medication. The SmartPillBox also sends information using a telephone line to external systems to store the information of the patient's compliance. The system alerts a nurse or physician if the patient is not complying with his medication.

2.6.2 Implementation

A real-time clock is used making sure the device alarms the user notifying him to take his medication. The real-time clock and 32 MHz crystal are used and

connected to the microcontroller through an I²C bus. The device alarms the user visually and by emitting sound because some users have hearing problems therefore it is better to alarm using both ways. The device is powered by a 9 VDC source. The LCD is programmed to displays a menu for the user to input the required medicine information. The information inputted is medicine name, dosage and time. The keyboard used for this device is the Membrane/Tactile PDA keyboard. The device uses a Cypress AT89C52 microcontroller.

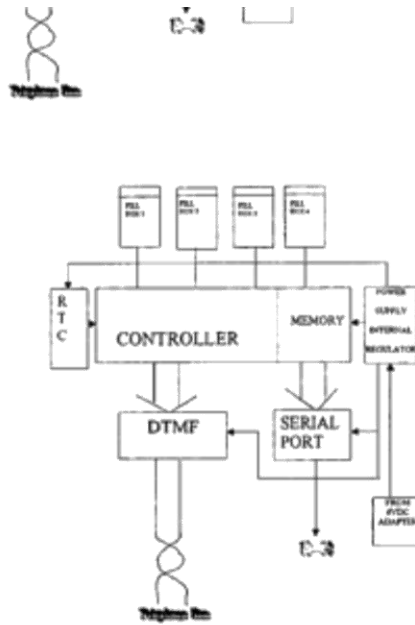


Figure 15: Intelligent pill box flowchart

2.6.3 Performance

A compiler and debugger/simulator will be used as verification that the LCD code is correct. After the LCD code is compiled and tested in software, it will be uploaded to the Atmel AT89C52 micro controller. The LCD will be tested to display the following: medication name, the correct dosage, dosage time and how the number of hours left to take the next dosage. Voltage across the LED is measured by a digital multimeter and resistance to get current and therefore determine whether it is within the tolerance of the LED's.

2.7: SMART MEDICINE CONTAINER [7]

2.7.1 General Overview

Dr.Nitesh Ratnakar is a gastroenterologist in Elkins, West Virginia and he is also the inventor of the Smart Medicine Container. This invention is directed to people that have difficulty remembering to take medicine at specific times; remembering that they have already taken their dose or the recommended dose to be taken.

These mistakes may have great consequences to the patient. This is true for patients who can't follow a specific regimen like the elderly, disabled, and patients with psychiatric disorders. People with faint eyesight or the blind will also benefit from such a project. Patients with the above situations take doses that are either too low to affect the course, or too high in which it might cause an overdose. Studies have been conducted that show a number of factors that are proven to be unsatisfactory to the effectiveness of the course, these factors include: 1) poor compliance with medicine regimen. 2) Frequent need to go to the pharmacist for refills and education. 3) Under or over dosing of medicine. The sole purpose of this invention is to prevent all these factors. Dr.Nitesh's invention basically separates pills into different compartments. The first compartment is used for storage; the second is used to dispense the pills. Several dispensation ports are provided with different speeds. The pills are separated due to a system and the conveyer belts. After the dispensation cycle is done the recovered pills go back into the storage compartments for use in the future.

2.7.2 Implementation

The project has an inbuilt pill dispensing assembly. The smart medicine container is made up of three compartments. The first compartment is the storage compartment, the second is the counting compartment, and the third is the dispensing compartment. All the compartments are stacked on top of each other as shown in the figure below. The storage compartment is placed on top; it usually

has a “V” shape and has an opening at the bottom to allow the pills to pass through to the second compartment, which is the counting compartment that is controlled by a regulating wheel. This invention has a pill counting gear for the sole purpose of automatically dispensing the desired number of pills at the desired time. The pill-dispensing tool is made up of a regulating wheel with two pill receptacles, a collecting conveyor, a dispensing conveyor, and motors to power the conveyors, the regulating wheel and a multitude of photoelectric sensors placed along the path of the pills. A printed circuit board regulates the entire pill dispensing process. The pill receptacles collect pills when facing the storage compartment and discharge them onto the collecting conveyor. The collecting conveyor has a “Y” shaped pill organizer, which aligns the pills in one column to assure an orderly discharge of the pills. After the pills are released from the counting compartment, they are passed down to the counting compartment through the collecting conveyor. Then the dispensing conveyor collects the pills from the collecting conveyor and discharges them to the last compartment which is the dispensing compartment. The dispensing conveyor is constructed in the same manner as the collecting conveyor. The passage way from the dispensing conveyor to the dispensing compartment us photoelectric sensors. The last step within the process includes the dispensing compartment that is located at the bottom and has an outlet door through which the pills are dispensed to the patient. The outlet door has sensors that control that operation.

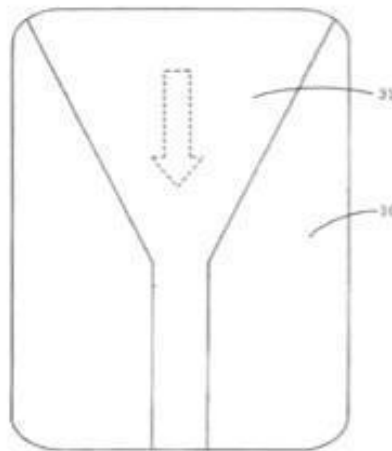


Figure 16: The collecting conveyor

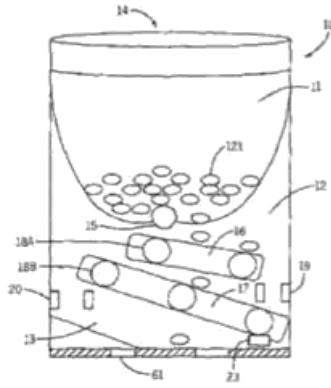


Figure 17: The counting compartment

The medicine container has been supplied with a modem and three communication ports, this gives the medicine box the feature of sending and receiving information and communication from external sources like a personal computer, wireless network, the internet and a telephone line just to name a few. The smart medicine container is covered on top by a cap that has a safety lock that adds a safety feature to the invention. The safety lock is coupled with a radio frequency identification reader (RFID), only users with the RFID tag will be authorized to control the safety lock. The smart medicine container has an internal printed circuit board and a memory chip located in the cap. The printed circuit board is programmed to carry out most of the invention's applications like data analysis, operational control and external communication. The memory chip stores operational data, information about the medication and any other information about the patient. The smart medicine container can communicate with the pharmacy wirelessly using the wireless transceiver. The pharmacist can program the medicine container with the required information about the patient, the medicine courses to be taken, when the first dose is going to be and what time the next dose is going to be, refills and expiration date of the medicine and many more.

2.7.3 Performance

This project has not been implemented in real life where the project's report is theoretical. We would hope to see this project implemented in real life.

2.8 Our Project

For our project, the smart medicine holder will be a modified version of all the other previous projects out there; the team will implement the most efficient features of the previous projects to provide the users with the ultimate smart medicine box. The main idea behind our project is to connect four different compartments to one conveyor that ultimately end up in the container that opens up to the users during the specified medication time. The four compartments are shaped in a rotating wheel manner with 16 holes that houses almost all types of pills. The rotating wheel is manufactured to have a whole size of 14 mm which is proven to be the ideal size, due to that the rotating wheel accepts almost every size and weight a pill can be and still dispense it effectively. The four rotating wheels are placed in the corners of a square shape, and in the middle, the stepper motor will be placed. The stepper motor is responsible for rotating the specified wheel with the help of the handle placed on the stepper motor. After dispensing, all the pills will fall into the main container, this container is equipped with an IR sensor that is responsible for automatically opening the main container anytime movement has been sensed during the specified pill dispensing times. The smart medicine holder will also contain temperature and humidity sensor that will be responsible for making sure that all the pills are stored within the correct cooling temperature and humidity. This feature is added just to prevent the pills from spoiling due to the exposure to undesirable temperature conditions. In the back of the smart medicine holder, the arduino is housed giving it a secure place that provides that the arduino will not be exposed to anything that may spoil it. Another important feature that has been added to this project is a complete sound system. The sound system is a feature that is new to the smart medicine holder's line. This feature is mainly useful for the blind and the elderly because every compartment has an audio recording prepared with information about the compartment number, the medicine name, the number of times a day it needs to be taken, and the dose of the medicine. An LCD screen is put in a very obvious place in the middle of the medicine container, and the LCD screen will print out all of the information previously mentioned within the sound system.

2.9 Comparison between the Implemented Project and Our Project

Table 3: Comparing our project with other projects

Project names	Microcontroller	Motor	Power supply	Sensor	Screen
Pill dispenser for dependent people	Funduino Pro mini	NA	Power/Data ports	NA	LEDs
Construction of a smart medicine dispenser	WinCE	NA	NA	NA	LCD Display
Intelligent Pill box	Cyprus AT89C52	NA	9VDC	Real Time Clock	LCD Screen + LED
Smart pill box					
Smart medicine container	Internal printed circuit board +memory chip	NA	NA	Photoelectric sensors	NA
Smart medicine box	Amtel 1284p	NA	16MHZ oscillator	Light based sensing + capacitive based sensing	LCD screen + LED lights
Our project	Arduino Uno	Stepper motor + servo motor	NA	IR sensor + Temperature and humidity sensor	LCD screen

2.10: Conclusion

In this chapter we researched the available and similar medicine boxes that have been already implemented. We discussed 7 research papers that discuss the projects that have been implemented; we also explained the main aspects of each project and the way each project works. Finally, in 2.9 we highlighted how our project differs from all the available projects and we stated that it will be the most modified version of all the available medicine boxes.

CHAPTER 3: METHODOLOGY, DESIGN AND ANALYSIS

In this chapter, we are going to discuss the design of our enforcement, and the components that we used in order to process it. The chapter of design and analysis also includes the functional and non-functional requirements of our project, which distinguishes between the main tasks of the project and any extra features which can be added to it. Moreover, the components are written down with their specifications, and what are the best components that suit our project, in terms of budget, delivering the task, etc.

3.1 Requirements

In the following section, we are going to admonish the requirements of our system, which divided into two things: the functional requirements and the non-functional requirements. Each one of the requirements clarifies a specific type of the system features, and a table to illustrated more.

3.1.1 Functional Requirements

Functional requirements are the tasks or the main parts that the project should give. In other words, functional requirements are things that the user expects from the implementation to be done with minimal malfunctioning cases. Requirements are considered very important features of the system and without them; the system will not perform according to the standards that were given. Table 4 illustrates the functional requirements of our project with their priorities:

Table 4: Functional Requirements

Functional Requirements	Priority	Completion
Push Button Switch	High	Yes
Medicine container	High	Yes
Temperature/ humidity sensor	High	Yes
LCD	High	Yes

3.1.2 Non-Functional Requirements

The non-functional requirements are the requirements that illustrate how the system work and implement the tasks it is expected to do, rather than explaining what the system should do. These can be features that do not affect the main functions of the project, such as efficiency, time, power consumption, and portability. Table 5 includes the non-functional requirements of our project and their properties:

Table 5:Non-functional Requirements

Non-Functional Requirement	Description
Performance	It is a measure of how efficient the system is, and for the project in hand, performance is a high priority.
Reliability	It is a measure of stability and how strong the system should be.
Availability	The system should be available whenever needed. It is a high priority requirement.
Safety	It is a measure of how well the system works with causing minimal damage. Safety should be of high priority.

3.2 System Architecture

The system architecture is the model that defines the structure and behavior of the system. The system architecture contains a description of the hardware, software and interface of the system; it also demonstrates how all the components work together. A formal description or the “architecture” is a representation of the system in a way that supports the reasoning or the flow of the system’s procedures. Our system architecture is shown in figure 18. It starts with the hardware and is mostly contained within it. The smart medicine holder starts out with push button switch where the user may input the appropriate information about the medication schedule the patient will be following, information like the number of different pills to be taken, and the times in which the pills are supposed to be consumed. After programming the medicine container, it is ready to use, the user can easily open the medicine containers and dispose the stash of pills in each compartment. The holder is equipped with a temperature and humidity sensor to monitor the humidity and heat in the compartments, a sound system that is a feature that is directed to patients with faint eye sight and the blind where whenever the compartment has been opened the sound system is activated speaking out the name of the medicine within this specific compartment. The pills are released to the patient with the help of the sensor and the motors in the container. The RTC module saves the date and time and all the inputs by the user.

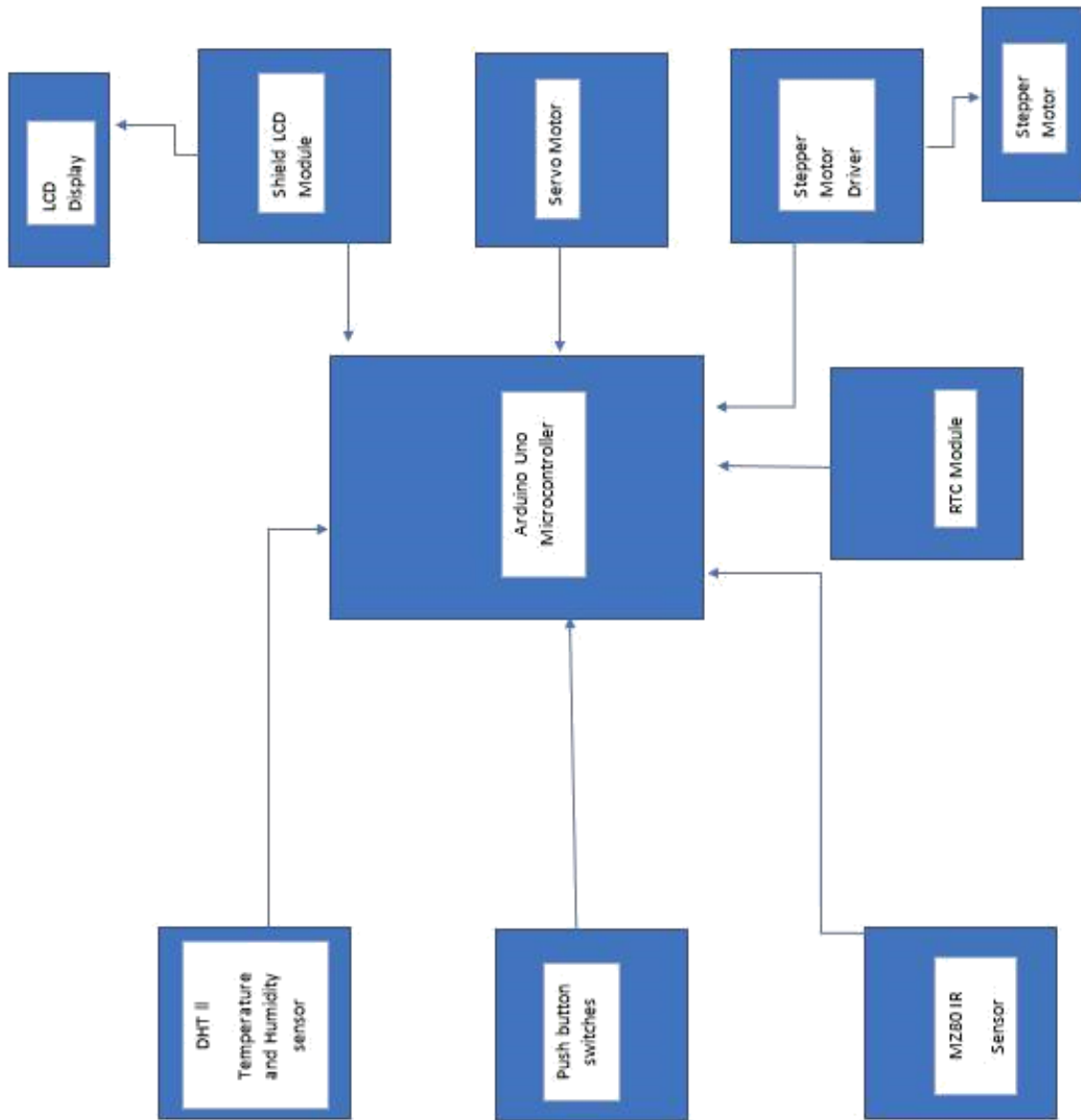


Figure 18: System Architecture of our project

3.3 Software

The Arduino Software (IDE) is an open source software where IDE stands for Integrated Development Environment. It makes it easy for the user to write their code and upload it to the board. Arduino IDE is an official software introduced by Arduino.cc that is used mainly for writing, compiling and uploading the code in the Arduino Device. It is a cross-platform application that runs on Windows, Mac OS X, and Linux. The environment is written in Java and the source code for the integrated development environment is released under the GNU that stands for general public license. The advantage of this software is that it can be used with any Arduino board. It supports languages like C and C++ and this is done by special rules of code structuring. There is a program called avrdude that the Arduino IDE have and it employs this program to convert the executable code into a text file in hexadecimal encoding. Then this file is loaded in to the Arduino board by a loader program in the board's firmware. In our project we choose the Uno Arduino board and choose the Arduino IDE software to write our code in. We saw that it is the best software suitable for us from simplicity perspective and that it is an open source software.

3.4 Hardware Components

In this section, we will explain the different hardware components used in our project we will explain them and also compare between similar components. There are five different types of Arduino boards:

3.4.1 Arduino Leonardo

This is the first development board of an Arduino; it uses one microcontroller with the USB. This tells us that it can be very simple and cheap because this board handles USB directly; program libraries are obtainable, which let the Arduino board to follow a keyboard of the computer, mouse and others. The Arduino Leonardo uses a 16MHz ATmega32u4 processor, 2.5KB SRAM, 32KB flash. Digital inputs/outputs are twenty but for the analog there are twelve inputs with zero outputs.

3.4.2 Red Board

This is the second type of Arduino that uses its IDE to program using a Mini-B USB cable. It will work on Windows 8 easily without having to change your security settings. The advantage of it is that creating it is very simple to utilize in the project design. You can do so by plugging the board; selecting the menu option to choose an Arduino UNO and after that you are ready to upload the program. You can also control the Red Board over USB cable using the barrel jack.

3.4.3 Arduino Uno (R3)

The third type is the Uno that is a huge option for initial Arduino. Digital inputs/outputs are fourteen pins; six pins can be used as PWM (Pulse width modulation outputs). Analog inputs are six that are classified as a reset button, a power jack, a USB connection and more. This type includes everything required to hold up the microcontroller; we can use it by simply attaching it to a PC by connecting a USB cable and giving the supply to get started with an AC-to-DC adapter or battery.

3.4.4 Lily Pad Arduino

This board is a wearable e-textile technology expanded by Leah “Buechley” and designed by “Leah and Spark Fun”. Every board design consists of a huge connecting pads & a smooth back. This helps them to be sewn into clothing using conductive thread. This Arduino has many advantages like comprising of the inputs and outputs power. Also the sensor boards that are built especially for e-textiles are an advantage because they are even washable.

3.4.5 Arduino Mega (R3)

The Arduino Mega and UNO are similar to each other. This type includes 54 digital inputs and output pins (14-pins of those can be used as a pulse width modulation), sixteen analog inputs, a reset button, a power jack, a USB connection and a reset button. It has a 16MHz Amega2560 processor, 8KB SRAM, 256KB flash memory.

These are all things required to hold up the microcontroller and works by simply attaching it to a PC by connecting a USB cable and give the supply to get started with an AC-to-DC adapter or battery.

3.4.6 Team's Decision

In our project we will be using the microcontroller Arduino that is called Uno. We choose this type because it has many advantages that we can benefit from in designing our project. The main advantage is that it has an open source software. The Arduino software is published as open source tools, available for extension by experienced programmers. It has the right number of input pins that we need and this makes this Arduino board very helpful for designing our project, as we need a bunch of digital inputs and outputs pins. Also the language we will be using can be expanded through C++ libraries.



Figure 19: The type of Arduino we used

3.4.7 RTC

In our project we used the Real time clock (RTC) to save all the inputs the user will enter on the EEPROM. It is the same as the hand watch and the microcontroller reads the user inputs and compare it with what is in the RTC using the EEPROM. The IC we used in the real time is DS1307. The RTC has 4 pins to read the time SDA, SCL and the VCC and ground. In our project the SDA is connected to A4 and the SCL is connected to A5.

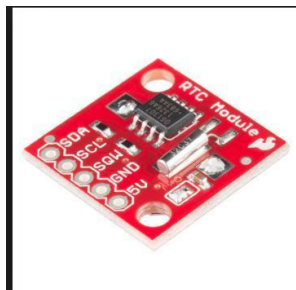


Figure 20: Real time clock used in our project

3.4.8 Temperature and humidity sensor

In our project we are using a temperature and humidity sensor to detect if the temperature and the humidity of the medicine tablets stored is suitable. We choose to use the DHT11 temperature and humidity sensor. The two temperature sensors that are mainly used for the Arduino are the DHT11 and the DHT22, both sense temperature and humidity but the DHT22 model is more accurate. In our project the temperature sensor uses one pin only in the Arduino.

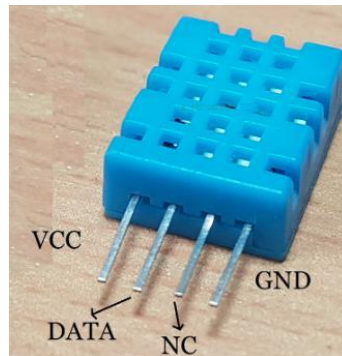


Figure 21: The temperature and humidity sensor

3.4.9 MZ80 IR sensor

The Infrared sensor is used in our project for detecting when the users hand is near the holder. We choose to use the IR sensor that is the MZ80 because it is simple, needs low range and is less expensive than the Ultrasonic sensor. We choose the MZ80 sensor because it has the distance of 80cm that is more than the normal IR sensor. The connection of the Mz80 can be analog or digital and it has VCC, ground and the one pin connected to Arduino.



Figure 22: MZ80 Sensor

3.4.10 MP3 DF player mini

We replaced the SD micro card reader with the MP3 because the SD micro card reader uses 5 pins and we want to save pins to still use the Arduino Uno. The MP3 uses only two pins which are the transmitter and receiver because it works serially. The MP3 is connected directly to the speaker.

3.4.11 Push button

We are using three push buttons as a user interface. It is used to take input from the user allowing information like the medication timings and date to be entered. We are using 12 mm Waterproof push buttons. The push buttons are digital not analog and they are connected to the ground so when a push button is pressed this pin will be low.



Figure 23: Push buttons used to enter data input

3.4.12 LCD Screen

In our project the LCD allows the user to see the information of each alarm and the time and date. The LCD takes 9 wires but we don't want to switch to Arduino Mega so we bought a Shield to save pins. We are using an LCD 1602 shield also called 2C. It has the same connection as the RTC and it helps us in our project by reading the data from the screen. It display the data saved on the RTC on the LCD and the thing that connects them is the LCD shield all of this is using the microcontroller Arduino.



Figure 24: LCD screen used

3.4.13 Easy driver

The easy driver for the stepper motor we are using in our project has two files A and B and the motor in between. In the driver we connected file A and B and there is something called speed and step. The step tells how many steps to move and the direction is specified by being positive for clockwise or negative for anticlockwise. The step is the cycles and each step is 4500 that indicates 90 degrees. If the speed is 1 it is fast, if it is 0.1 this is the slowest, if 0.5 then medium. We put it as 0.2 because if we increased it the holders won't turn only the stepper motor. We will take 3 pins for the Arduino, 2 for the power supply, four for the stepper motor and two for the step and speed. The step and speed will be assigned to pins 2 and 3.



Figure 25: Stepper motor driver

3.4.14 Speaker

In our project we are using a speaker for the sound that will be outputted to the user. We are using a 5-10 Watt Arduino speaker.

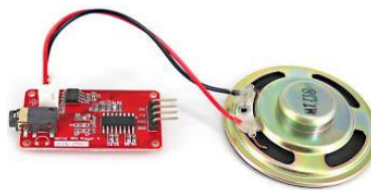


Figure 26: Speaker to output the alarms

3.4.15 Servomotor

We are using a Servomotor in our project for mechanical movement purposes. A servomotor is a closed-loop system that uses the position feedback to control its motion and its final position. There are many types and we choose a 12V servomotor that needs electricity instead of batteries. In our project the servomotor is connected directly through digital pin 9. The servo motor has two rules to operate one is the users hands must be near the holder and detected by the sensor and there

must be a tablet in the final container to open. It moves 50 degrees to open the final tray for the user to get his/her medication then 50 degrees to close it.



Figure 27: Servo motor

3.4.16 Stepper motor

There are three main types of motors; Brushless, Brushed and Stepper motors. We need these motors for mechanical motion. We choose the stepper motor because it has the best position control out of the three types. The stepper motor is not the most efficient type but because it has the best position control that is the main reason we choose it.

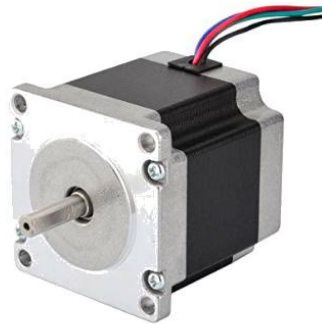


Figure 28: Stepper motor for the cycles

3.4.17 Switch

We used a switch in our project to switch on and off the holder.



Figure 29: Switch for turning on and off

3.5 Budget

Table 6: Budget of our project

Component Name	Availability	Quantity	Price
Switch	yes	1	0.250 KD
Adaptor 12v 1.5A	Yes	1	3.750 KD
Mp3 Module	Yes	1	3.750 KD
Stepper motor	yes	1	1.500 KD
Temperature and humidity sensor	Yes	1	1.750 KD
Stepper motor driver	Yes	1	2.750 KD
Push button switch	Yes	3	1.200 KD
MZ80 IR sensor	Yes	1	3.000 KD
Speaker (sound system)	Yes	1	4.000 KD
Servo motor	Yes	1	1.250 KD
LCD	Yes	1	2.500 KD
L2C LCD shield	Yes	1	1.750 KD
Arduino UNO	Yes	1	4.000 KD
RTC module	Yes	1	2.000 KD
Plastic	Yes	10	80.000 KD
Anti-bacterial material	Yes	5	100.000 KD
Brakes	Yes	4	10.000 KD
Power input of adaptor	yes	1	0.150 KD
LED	Yes	1	0.050 KD
Screws	yes	30	0.500 KD
Cables	yes	45	2.000 KD
Injections holder	Yes	1	5.000 KD
Price in Dollars	-	-	850 USD
Price in dinars	-	-	258.200 KD

3.6: Conclusion

In this chapter we take a closer look at the components chosen in this project. The medicine container is made mostly of hardware but also contains software in which everything used is listed. In this chapter, we explore all of the options available in the world and specify the chosen resources and why we chose them to seek the most optimal solution to provide the best version for this project.

CHAPTER 4: IMPLEMENTATION

This chapter of this report will elaborate on the details of how the team approached the Implementation of the current functionality in hand. The chapter will also talk about the assembly details of the machine overall.

4.1 Sound system

The main idea of the smart medicine holder is to aid the elderly; the blind and the mentally disabled fulfill their prescribed treatments in order to attain the fullest benefit for their medical program. One of the important features in this smart medicine holder is an installed sound system. This sound system will be the guide for the medicine holder's users.

The first step is to record all of the desired phrases and upload them to the computer and then to the arduino. Before uploading the audio sound messages to the arduino, there needs to be some adjustments done on the voice messages. The arduino only accepts voice recordings in the (.WAV) format. With the use of a audio converter website called (online-convert.com) making sure that the PCM format is on the PCM unsigned 8-bit option.



Figure 30: Audio converter

4.2 MP3 and card

The Mp3 is used to store all the information we needed to upload to the arduino. Before the SD card can be used, the format of the card needs to be fixed in order to match the language of the arduino. The program used to convert the format of the SD card is SD format for aduino (FAT32). After that we will put what is on the SD micro reader on the MP3 then to the arduino. In the end, all the data needed for the sound system to be complete will be stored and numbered in the SD card memory.

4.3 The smart medicine holder

The medicine holder contains four compartments; each compartment can hold up to 15 pills, the whole size is 14 mm large. In the middle of the board in between the four compartments, the stepper motor will be placed. The stepper motor will be responsible for moving the compartments in order to drop the specified pill in the desired time. The medicine holder is created in a matter where it has a specific place in the back that is supposed to house the arduino and all the sensors, this area will be hidden so it cant be seen by any user. At the bottom of the holder, there will be the final container where all the dispensed pills finally go to, all four compartments are connected to this container. After the pills have been dispensed this container opens up to the user where they can pick up their pill. This compartment will not open up except if two rules are satisfied one is the user hand is detected by the sensor and its time for a tablet which is ready in the final container. The final container will open for 9 seconds delay that is 9000 in our code then close again and wont open until both the two rules are satisfied again.

4.4 The Stepper motor mechanism

We have 4 holders A, B, C and D each holder represent a compartment. Depending on the input that we enter the holder will be called. We designed the turning of the compartments done by the stepper motor to be in cycles either clockwise or anticlockwise depending on the sign we enter either positive or negative steps.

The starting point must be the same for each compartment. For the holder A the stepper motor will turn quarter of a cycle anticlockwise this means -4500 steps in order to drop the specified pill in the final container. This is because each quarter of a cycle represents 4500 and the negative for the direction which is anticlockwise. Then the stepper motor moves quarter of a cycle clockwise to return to the starting point because we want all of the holders to start at the same position for simplicity. For holder B the stepper motor moves half a cycle anticlockwise (-9000) then quarter of a cycle clockwise (4500) in order to drop the specified pill. Then the stepper motor will turn three quarters of a cycle anticlockwise to return to the starting point. For holder C the stepper motor will turn three quarters of a cycle anticlockwise then a quarter of a cycle clockwise to drop the pill wanted. Then the stepper motor will turn half a cycle anticlockwise to return to the starting point. For holder D the stepper motor will turn quarter of a cycle clockwise to drop the pill then quarter of a cycle anticlockwise to return to the starting point. In each of the four compartments we used an antibacterial plastic to be safer when placing the pills on them. The antibacterial plastic can be pulled off the compartment to be washed then returned back.

4.6 The six alarms

We designed six alarms; we said that each holder is dedicated for one alarm but some medications requires patient to take a tablet one before and after eating so 6 alarms was to be safe. Each alarm has 4 choices (time, day, duration and holder name) for example we can assign a specific medicine to be at 3pm, every other day, its duration is 2 seconds and assigned for holder C. The first thing to do when we start the holder is to put in the date and time to be saved in the RTC so even if we closed the holder and reopen it the time and date will still be saved. After that we start choosing our alarms and enter the 4 inputs of information for each alarm. The switch has many menus; when we start the first menu that will appear is the main menu screen that has day, date and time. Then we will press on the switch for the submenu to appear which has the data. The third press has the time then after that we will enter the alarms and input the four things required. This means four presses for

each alarm and we have 6 alarms so 24 then 3 more for the main menu, data and time; this is a total of 27 submenus. We can make some alarms off by not assigning any day to it. We have 3 push buttons the middle one for the menu that is pin 2; the left button to decrease and the right is to increase, they are called the direction and stage pins assigned to pins 10 and 8. For example if we want 2pm when we reach it by the left and right buttons we press menu that is the middle button. If we want to go between one alarm to another we use the left and right button and if we press menu we enter inside the alarm and start to enter the four inputs. We press the left button to go to the other alarm and as long as we press menu we are in the same holder or the same day or the same time; we must press the left button to get out. As long as the cursor is available we press menu and we are still adjusting the data once it is gone we can press the left button. The servomotor has a delay with each cycle and the same delay between all of them. The servomotor has two rules, one we implemented in the code that has the cycles. The other one is that the user must put his hand near the final container for it to open with the help of the sensor. We designed that there is a specific time for the final container to open that is 9 seconds and then it will close automatically.

4.7 Temperature and humidity sensor

In our project we used the temperature and humidity sensor of type DHT11 that uses one analog pin in the arduino board. This sensor is used so that if the temperature of the container gets higher than 29 or the humidity is higher than 60 an LED light will turn on and an alarm that says that it is not suitable for either the tablets or the injections.

4.8 Final Implementation



Figure 31: Overall view of the Smart medicine holder



Figure 32: Side view of the holder (switch, Adaptor, injections holder)



Figure 33: Side view of the holder (LED, speaker)

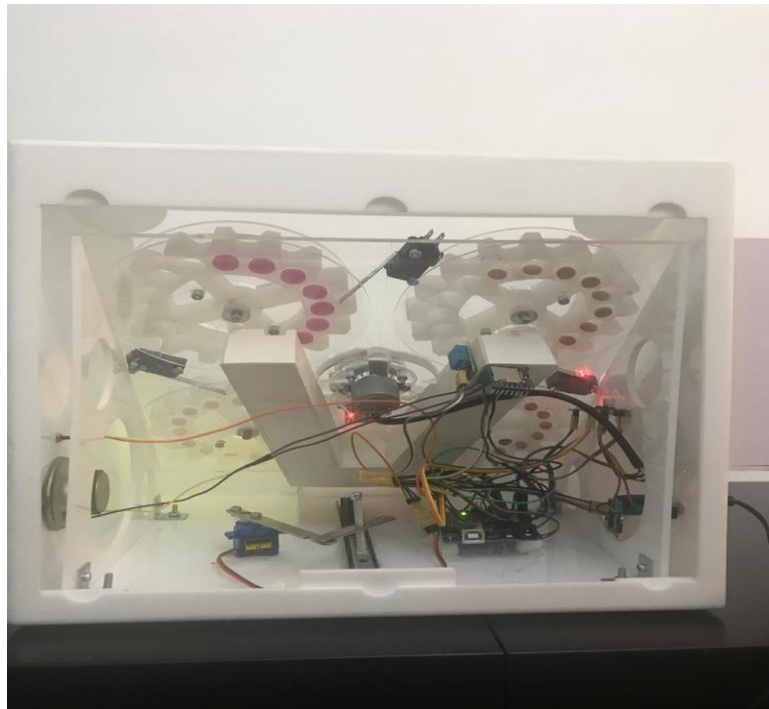


Figure 34: Back view of the holder (the circuit)

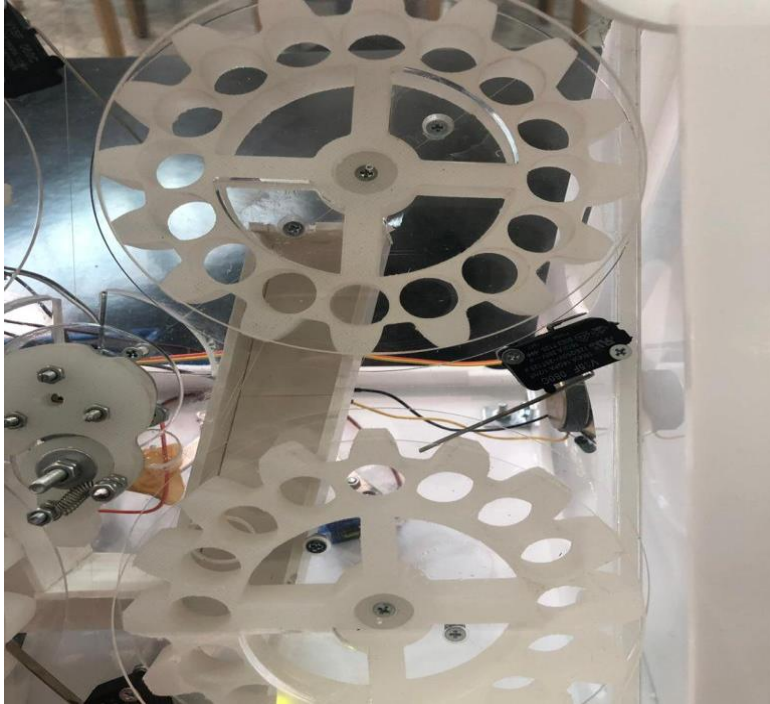


Figure 36: The medicine containers



Figure 35: LCD screen and push buttons



Figure 37: IR sensor and opened dispenser basket

4.9 Conclusion

This chapter was well thought of by the team to make sure that the implementation of the project is perfect. This chapter explains the mechanism of the stepper motor in order to move the compartments, how the six alarms work and also the motion sensor role in our project. Chapter four also taught us how to handle different hardware and software components like the sound system to implement a smart medicine container.

CHAPTER 5: EVALUATION

In this chapter, we shed the light on the impact and evaluation of our project, and how this project affects the community we live in, we will be concentrating mainly on the economic, environmental and social impact for this project. Every project has its strengths and weaknesses and is all-different in the way the projects contribute to society. Overall, our project's positives overweight its negatives where, it fixes a serious problem within a very sensitive area, which is the medical field.

Throughout this project, we will speak in details about the specific types of impacts this project have on the community, we will look into the economic aspect, which talks mainly about how this project contributes in the economical field, the medical field; the environmental aspect which describes how the project affects the environment, and how the material used in the making of this project impacts the environment. The social impact will talk about how our project affects the society and people around us. Lastly, we are going to include the results of a survey we conducted and the overall percentage of each question asked.

5.1 Economic aspect

The medicine holder project will enhance many economic aspects in the field of the quality of how medicines are stored and arranged. One of the goals that this project will outcome is building a cost effective holder that can accumulate more than one type of medicine instead of specifying a holder for each type of medication. This will save money because although it is more expensive than a holder that accepts one type of medicine but a slight increase in price will help you accommodate all the medications you have throughout the day. When we choose our components we choose low power consumption and cheap components so that we minimize the budget for implementing this holder. This will be helpful as it saves money for the user.

The medicine holder material will be efficient and ready to be used for all kinds of people and stays for a long time without being damaged and needing to buy a new one. Although there is a slightly small percent that it can be damaged and if so the amount to repair it will be small and it is better to repair one holder that contains all your medications every year then repairing a holder containing one medicine every month. Nevertheless, this project requires people that have experience in electrical, computer and even mechanical engineering. Those people must work hard in implementing both the hardware and software part efficiently. The advantage is once they are done with the holder the user can deal with it without the help of anyone. This means a salary is not necessary needed to be paid for someone to keep an eye on the holder because the user can deal with it alone. They are only needed to design the project and deliver it to the user working successfully without any errors. This saves money because salaries will go to other people who are importantly needed for certain projects.

5.2 Environmental impact

The smart medicine holder project we are working on can have an effective part in protecting the environment for a number of reasons. Our holder doesn't contain toxic fluid or out coming a gas that can harm the Ozone's sphere and cause air pollution. Some machines implemented nowadays produce harmful gases that destroys the environment and effects people/users health. Our smart medicine holder will help the environment very much as it is made of components that are not harmful for the user and can be used by any age. The holder is also portable and lightly weighted so it wont take place and wont hurt anything when placed above it. Noise pollution is another aspect that we took care of in our project. The smart medicine holder doesn't include any chemical substances that can react with high temperatures and if so our holder has a temperature sensor that will produce an alarm if the temperature went high to protect the medications stored.

5.3 Social Impact

The medicine box has great social impact as it targets everybody that takes prescribed medication. Social impact is how our project effects the community and weather it is helpful to the community or not. The smart medicine box helps pregnant women take all their necessary medication, old people who might forget or mix up medications, deaf men and women who have prescribed medications and much more therefore it targets a lot of consumers and helps them all with their health. Our project helps nurses with the patients and unsupervised old people. The health sector will benefit from our project as it solves many problem regarding the timing, intake and distribution of medications. Many families will benefit from our project therefore the whole society is affected positively.

5.4 Survey

In order to get an unbiased opinion about the project in hand, the appointed team decided to conduct a survey to help gather as much information as possible from the public domain. The survey questions target some important details that are crucial to understanding the level of public satisfaction between the current measures used for repairing the smart pill medicine and the attempted design. In addition, the survey has shed light on the most important features that the public expects the design in hand to own.

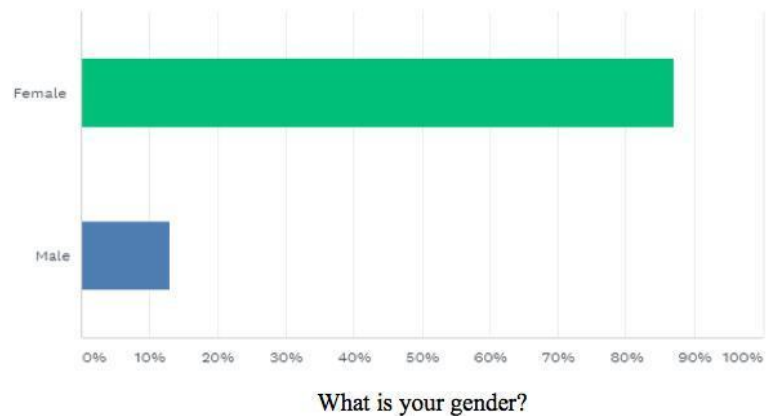


Figure 38: Results of the first question

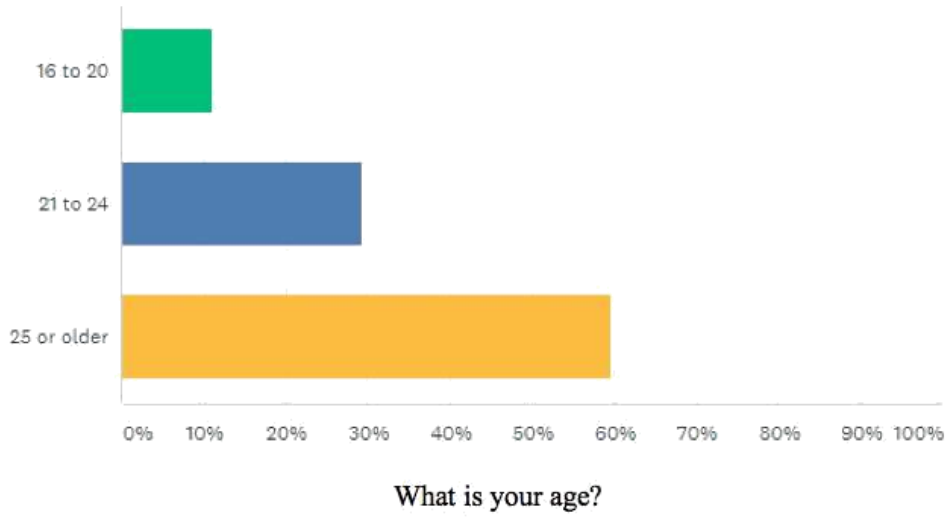
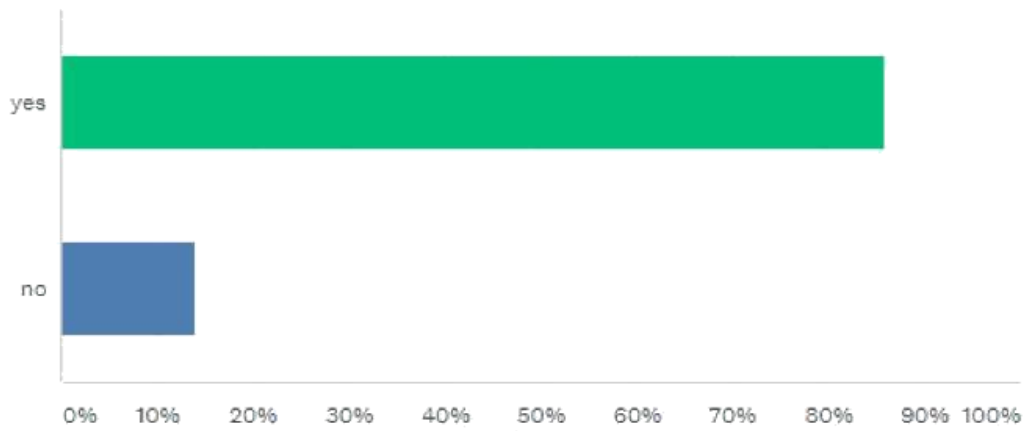
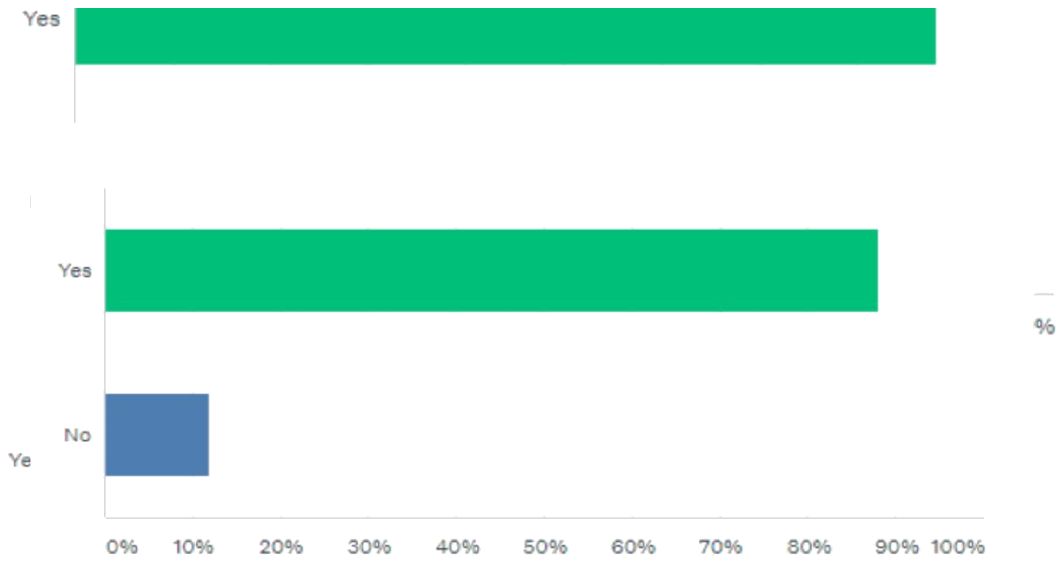


Figure 39: Results of the second question



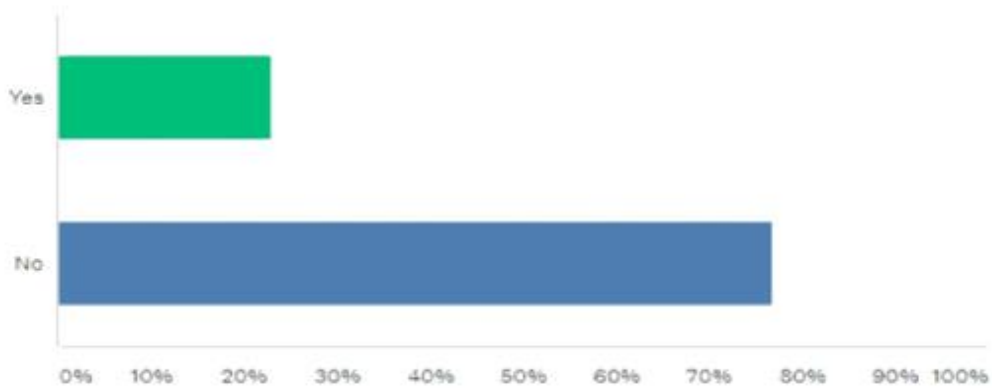
Do you know someone who is old, suffering from a disease, blind, or suffering from anything else that requires taking medicine regularly?

Figure 40: Results of the third question



Have you ever forgotten to take your medicine on time, or what type of medicine you should take, or even the quantity of the medicine that you should be taking?

Figure 41: Results of the fourth question



Have you ever heard about the smart medicine holder? “a programmable holder that reminds the users which specific medicine to take at particular times of day and serves at those times each day”

Figure 42: Results of the fifth question

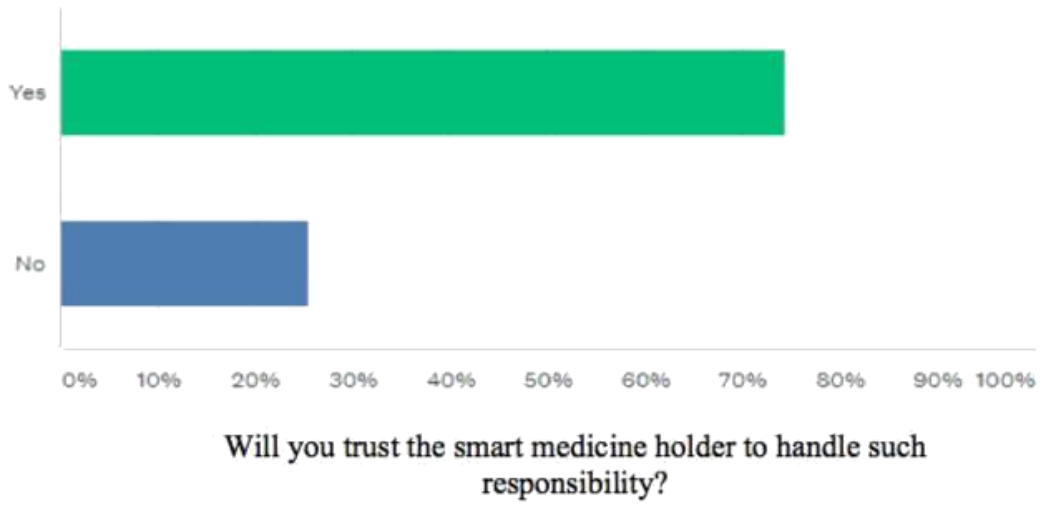


Figure 43: Results of the sixth question

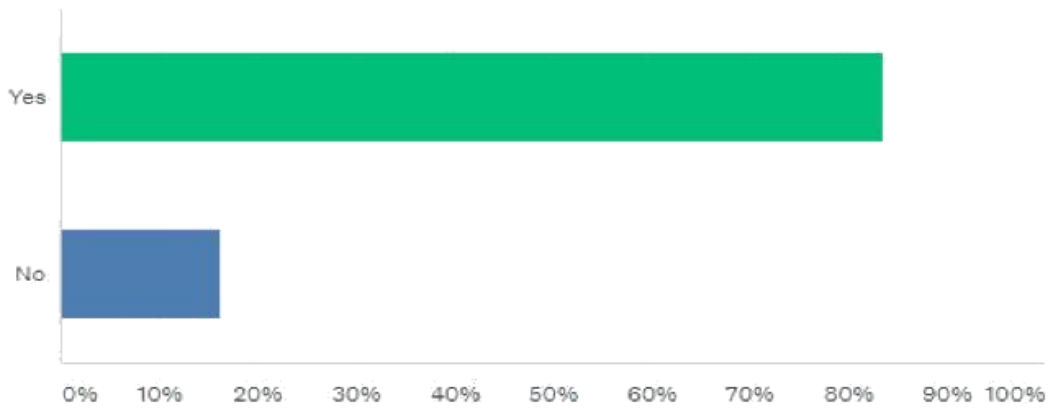
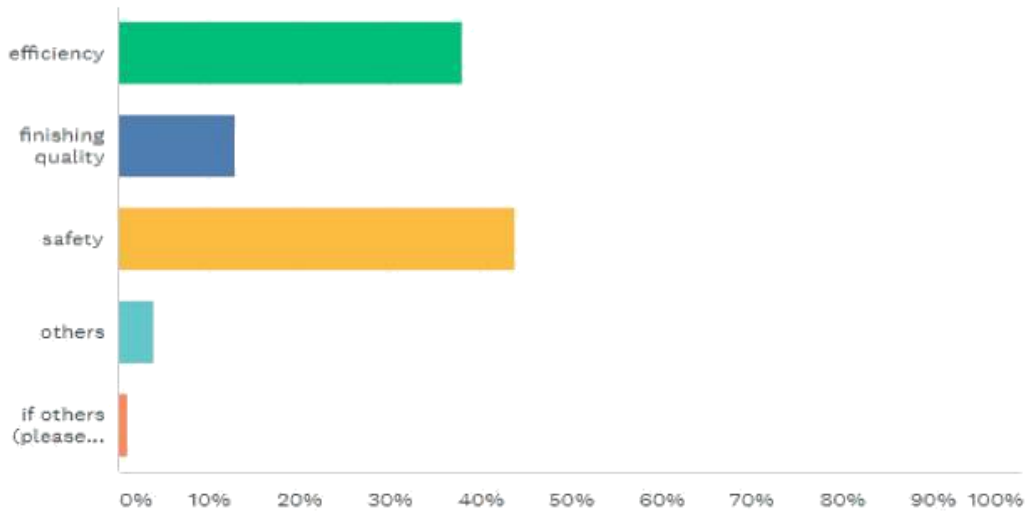
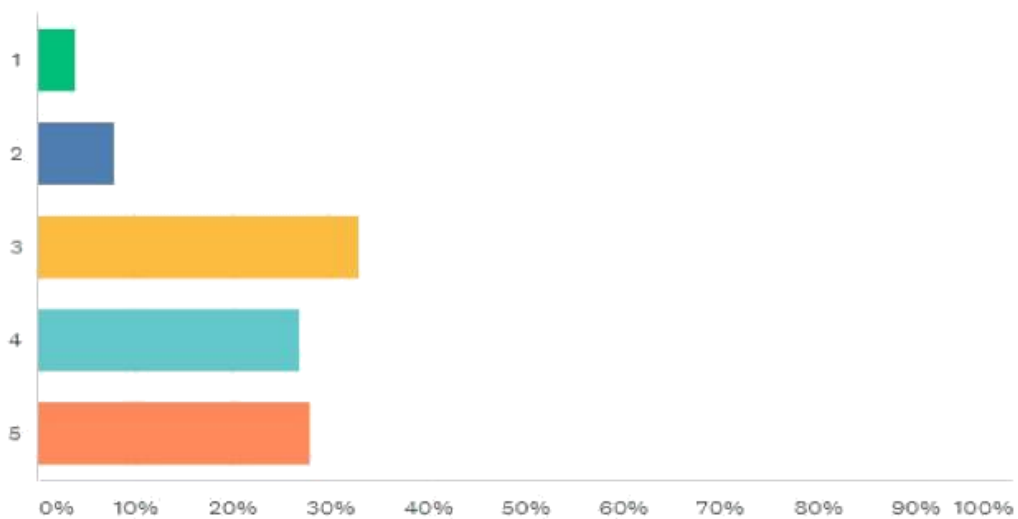


Figure 44: Results of the seventh question



What is the most important feature that will distinguish the smart medicine holder?

Figure 45: Results of the eighth question



On a scale from 1 – 5 how useful do you think the smart medicine holder will be?

Figure 46: Results of the ninth question

The results obtained from the survey were of great help to the project, and luckily in favor of the proposed design, as the majority of people reported that they know someone who is old, suffering from a disease, blind, or suffering from anything else that requires taking medicine regularly. Also, most people stated that Have forgotten to take their medicine on time, or what type of medicine they should take, or even the quantity of the medicine that they should be taking However, when asked about the smart medicine container, most of the people doesn't know it. Some key results questions that showed the team the importance of the design in hand was the dissatisfaction that the public exhibited when asked about their opinion in saving smart medicine holder lives. Thus, when asked about their opinion in handling the smart medicine holder such responsibility and trusting it, the response was highly in favor of the development of such holder or container. Through some other results obtained, the team has realized that the most important feature that should distinguish the implemented design is the container's efficiency it. Moreover, when the public was asked about how efficient the featured design could be if provided to our users or hospitals, most rated it to be very helpful if implemented in our countries. Overall, the survey was a great tool to gather direct and honest information that has offered the team great insight on how to tackle the project in hand.

5.5 Engineering Ethics

Engineering ethics are basically the moral code of conduct that each engineer must have towards his/her carrier. The engineering carrier has direct impacts on human lives, so these ethics set the standards that all engineers must follow to ensure the quality of work being produced. When it comes to the project in hand, many ethical values were taken into consideration. However, the team decided to choose the NSPE Code of Ethics

[34] as a guideline for the ethical principles that will govern all decisions and behaviors conducted during technical work. The following are the most critical values that the team adhered to:

1. To hold paramount the safety, health, and welfare of the public
2. We shall issue public statements only in an objective and truthful manner.
- 3 To be guided in all our relations by the highest standards of honesty and integrity. Honest and realistic in stating claims or estimates based on available data
- 4 To strive to serve the public interest in other words to improve the understanding by individuals and society of the capabilities and societal implications of conventional and emerging technologies, including intelligent systems;
- 5 To maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations;
- 6 To seek, accept personal responsibility for our professional activities, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others;
7. To avoid injuring others, their property, reputation, or employment by false or malicious action.

5.6 Project Evaluation

As mentioned before and preciously in chapter 3 where the functional requirements were highlighted, and they were basically the main tasks to be fulfilled by our project. Therefore, after the project implementation was completed, it was at most important that once again we touched to see if the project standards and requirements were achieved. The table below lists their standards with their status of completion.

Table 7: Table of standards and the status of completion

Functional Requirements	Priority	Completion
Push Button Switch	High	Yes
Medicine container	High	Yes
Temperature sensor	High	Yes
LCD	High	Yes

5.7 Conclusion

This chapter is all about discovering the main and different aspects in which our project affects. The smart medicine holder is constructed for the medical field. Within our study for this chapter we came to a conclusion that the smart medicine holder fulfills the purpose it was created for and helps make sure that the patient gets the maximum effectiveness from the treatment prescribed for them. The survey results proves that people with specific and complicated treatments who are the target population for this project would be happy to have this within their possession.

CHAPTER 6: CONCLUSION AND FUTURE WORK

During this course our Smart Medicine Holder project was successfully constructed and tested. In this chapter section 6.1 summarizes the accomplishments of previous chapters. Section 6.2 discusses the current progress of the Smart Medicine Holder, and in Section 6.3 we are talking about what future work can be done to enhance the functionality of the project. Section 6.4 talks about the problems we faced throughout the semester.

6.1 Summaries

All group members choose our capstone project idea as we all felt that the idea was important and that it is very beneficial. The medicine box project is divided into two main parts, the first part is that the box is made of sub divided compartments that contain the medicine and whenever the user has to take the medication the device alarms the user. All this was briefly mentioned in chapter 1 as the chapter constructed by all team members discusses the main objective of the project and the goals to be achieved by the device. For chapter two our team read multiple articles to gather and acquire information regarding our idea. This helped us further understand concepts regarding our project. In chapter three, our team had a good knowledge base therefore we started comparing different components and we choose the best components in orders to build a prototype. Chapter four included aspects concerning the buildup of the hardware and software implementations in the project. After that, in chapter five which is the evaluation chapter our team discussed the impact of the project on different aspects of life such as the economic, social, environmental and so on.

As a team we also explained the engineering ethics that we obeyed, and the analysis done on the conducted survey. Finally, chapter six is the conclusions where we reinforced on our project idea. We also discussed our final prototype and explained our future plans.

6.2 The Current Progress

By the end of capstone 2, our team had successfully managed to satisfy all goals that were proposed in all stages. First, we successfully managed to implement the hardware part that is a smart medicine holder with separate compartments inside to accompany different categories of tablets. Then, all compartments were combined at the end to output a single tablet that the user needed at a specific time. The user can adjust the date and time of the medication, this is something we worked on and was verified when tested. Other aspects like alarming the user when the temperature gets high and having the tablet to drop using a sensor are goals that our team achieved. In addition, our project consists of a medicine box that is made up of four pill compartments that revolve in different cycles to dispense the pills for the user. Our project has six different alarms for the four medicines and two extras just in case a medication is supposed to be taken twice a day. Moreover, in the software part our team wrote a code for a program to manage the time and amount of the medication for the user based on the inputs he/she entered.

6.3 Future Work

Although we managed to satisfy all our goals and tried to improve it as much as possible but there must be some aspects we didn't cover. Our smart medicine holder project can be improved and some of these improvements include:

1. We can improve the prototype of our medicine holder by using a more suitable and efficient material.
2. We can add a new box inside the holder for ampules so not only tablets are stored.

3. We can let the holder check if the ampule is expired or no and if so it can alarm the user.

4. As a future plan we can also put two compartments on top of each other. Each compartment has 16 holes, but we use 15 because the last one is to be safe. So when we put two on top of each other there will be 32 holes so instead of 15 pills giving one week this will give us double the duration for example 2 weeks so there is no need to refill the compartments every few days.

6.4 Problems faced

- We bought an LCD screen, but it didn't work because all the buttons work on one analog pin, which is A0. So to go from one alarm to another we must press on another pin for the device to understand to go out to the next alarm. It worked only for one alarm not all, so we bought a new one.
- The MP3 was not available in Kuwait but RTC just bought it instead of us ordering it online.
- The reader uses 6 pins so we must use Arduino mega instead of Uno or we must save pins and bring something that works with less pins. MP3 works serially so it takes 2 pins and the LCD takes 6 pins and the reader takes 6 pins and with the easy driver and the servo motor and temperature sensor the total number of pins are 20. The problem is the Arduino has only 13 pins so to still use Arduino Uno and not replace it with mega we replaced the reader with MP3 and we bought a shield for the screen to save 4 pins.
- The stepper motor we bought was weak and we noticed that because when there is weight on it then it stops working. One of the reasons it was weak because it acts as a servo and stepper motor so we bought another one that acts as a stepper motor only.

REFERENCES

- [1] J. Pak and K. Park, "Construction of a Smart Medication Dispenser with High Degree of Scalability and Remote Manageability," *Journal of Biomedicine and Biotechnology*, vol. 2012, pp. 1–10, 2012.
- [2] "US7877268B2 - Intelligent Pill Box - Google Patents." *Patents.google.com*. N.p., 2018. Web. 20 Nov. 2018. <<https://patents.google.com/patent/US7877268B2/en>>.
- [3] Gerlt, Axel, and Robert Kagermeier. "MEDICINE CONTAINER. Patent Application Publication, 2008, MEDICINE CONTAINER, patentimages.storage.googleapis.com/72/62/2c/d34dc7ecddc84b/US20080109510A1.pdf .
- [4] Huang, Mingyuan, and Jie Zhang . "Smart Medicine Box". *Smart Medicine Box*, people.ece.cornell.edu/land/courses/ece4760/FinalProjects/f2014/mh2239/finalreport/finalreportweb2/FinalReport.pdf.
- [5] Nitesh , Ratnakar. "Smart Medicine Container". Patent Application Publication, 2010, pp.1–30, 2010.
- [6] Salgia, Aakash Sunil, et al. "Smart Pill Box". *Indian Journal of Science and Technology*, 2015, Smart Pill Box, www.indjst.org/index.php/indjst/article/view/58744.
- [7] jDe Juan Grau, Guillermo. "Smart Pill Dispenser For Dependent People". 2015, core.ac.uk/download/pdf/79176691.pdf

APPENDIX A

```
// Smart Medicine Holder Project code
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
library////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
#include "Wire.h"
#include "LiquidCrystal_I2C.h"
#include "math.h"
#include "EEPROM.h"
#include "Servo.h"
#include "Arduino.h"
#include <SoftwareSerial.h>
#include <DFPlayer_Mini_Mp3.h>
#include "dht.h"

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////arduino pinout
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
#define dht_apin A0 // Analog Pin sensor is connected to
#define DS1307_I2C_ADDRESS 0x68
#define PIN_STG 8 // pushbutton switches
#define PIN_MENU 2
#define PIN_DR 10

#define PIN_COMMAND_A 7 //medicine holder A
#define PIN_COMMAND_B 3 //medicine holder B
#define PIN_COMMAND_C 11 //medicine holder C
#define PIN_COMMAND_D 12 //medicine holder D

#define OPEN LOW // holder stop
#define CLOSED HIGH //holder run

#define NR_ALARME 6 ////////////////////////////////////////////////////
#define MAX_MENU_POS NR_ALARME*4+3 //must be NR_ALARME*4+3

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////// stepper motor////////////////////////////////
int smDirectionPin = 5; //Direction pin
int smStepPin = 6; //Stepper pin

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////// MZ80 ir sensor////////////////////////////////
int MzSensorPin_1 = A1; // select the input pin for the MZ80 IR sensor
int MzSensorValue_1 = 0; // variable to store the value coming from the sensor
int a=0;

Servo myservo; // create servo object to control a servo
dht DHT;
```

```

// Set the LCD address to 0x27 for a 16 chars and 2 line display
LiquidCrystal_I2C lcd(0x27, 16, 2);
byte menuPos=0;
byte subMenuPos = 0;
byte cursorPos = -1;
char line1[17];
char line2[17];
char line[17];
boolean dataDirty = false;    // data
boolean oraDirty = false;    //time
long lastMillis=0;
boolean bAlarma = false;    //alarm

typedef struct alarm{
  byte hour;
  byte minute;
  byte days;
  byte duration;
  byte relee;
  byte unitAlarma;
};

typedef struct ds1307_time{
  byte second;
  byte minute;
  byte hour;
  byte dayOfWeek;
  byte dayOfMonth;
  byte month;
  byte year;
};

struct ds1307_time time;

alarm* alarme = (alarm*) malloc(sizeof(alarm) * NR_ALARME);

byte relee[]={PIN_COMMAND_D,PIN_COMMAND_C,PIN_COMMAND_B,PIN_COMMAND_A};
//////////

byte decToBcd(byte val){
  return ((val/10*16)+(val%10));
}

byte bcdToDec(byte val){
  return ((val/16*10)+(val%16));
}

void setDateDs1307(struct ds1307_time time)          // set data
{
  Wire.beginTransmission(DS1307_I2C_ADDRESS);
  Wire.write(0);
  Wire.write(decToBcd(time.second));
  Wire.write(decToBcd(time.minute));
}

```

```

Wire.write(decToBcd(time.hour));

Wire.write(decToBcd(time.dayOfWeek));
Wire.write(decToBcd(time.dayOfMonth));
Wire.write(decToBcd(time.month));
Wire.write(decToBcd(time.year));

Wire.endTransmission();

}

void getDateDs1307(struct ds1307_time *time)
{
Wire.beginTransmission(DS1307_I2C_ADDRESS);
Wire.write(0);
Wire.endTransmission();

Wire.requestFrom(DS1307_I2C_ADDRESS,7);

(*time).second = bcdToDec(Wire.read() & 0x7f);
(*time).minute = bcdToDec(Wire.read());
(*time).hour = bcdToDec(Wire.read() & 0x3f);
(*time).dayOfWeek = bcdToDec(Wire.read());
(*time).dayOfMonth = bcdToDec(Wire.read());
(*time).month = bcdToDec(Wire.read());
(*time).year = bcdToDec(Wire.read());
}

void showMenu(){
char *days[7];
days[0]="M\0"; //days
days[1]="T\0";
days[2]="W\0";
days[3]="T\0";
days[4]="F\0";
days[5]="S\0";
days[6]="S\0";
if(menuPos==1){
strcpy(line1,"Date\0");
strcpy(line2,date(time));
strcat(line2," ");
strcat(line2,getZiuaSaptamaniiShort(time.dayOfWeek)); //get Short Weekday
}else if(menuPos==2){
strcpy(line1,"Time\0");
strcpy(line2,time(time));
}else if((menuPos+1)%4==0){ //starting with 3, from 4 to 4 we have the alarm time
strcpy(line1,"Alarm ");
line1[7] = 48+(byte)((menuPos+1)/4)/10;
line1[8] = 48+(byte)(((menuPos+1)/4)-((menuPos+1)/4)/10*10);
line1[9]='\0';
strcat(line1," time\0");
strcpy(line2,getFmt2Char((alarme[(menuPos+1)/4-1]).hour));
}
}

```

```

strcat(line2,":");
strcat(line2,getFmt2Char((alarme[(menuPos+1)/4-1].minute));
strcat(line2," ");
}else if((menuPos)%4==0){ //starting with 4, out of 4 in 4 we have the alarms
strcpy(line1,"Alarm ");
line1[7] = 48+(byte)((menuPos)/4)/10;
line1[8] = 48+(byte)((menuPos)/4-(menuPos)/4/10*10);
line1[9]='\0';
strcat(line1," days\0");
strcpy(line2,"M");
for(int i=6;i>=0;i--){
if((alarme[(menuPos)/4-1].days & (byte)ceil(pow(2,i))){
strcat(line2,"X");
}else{
strcat(line2," ");
}
if(i>0)strcat(line2,days[7-i]);
}
}else if((menuPos-1)%4==0){ // starting with 5, out of 4 in 4 we have the alarm duration
strcpy(line1,"Alarm ");
line1[7] = 48+(byte)((menuPos-1)/4)/10;
line1[8] = 48+(byte)((menuPos-1)/4-(menuPos-1)/4/10*10);
line1[9]='\0';
strcat(line1," durate\0");
itoa(alarme[(menuPos-1)/4-1].duration,line2,10);
if(alarme[(menuPos-1)/4-1].unitAlarma){
strcat(line2," min");
}else{
strcat(line2," sec");
}
}
}else if((menuPos-2)%4==0){ // starting with 6 of 4 in 4 we have relays activated by the alarm
strcpy(line1,"Holder alarm ");
line1[13]= 48+(byte)((menuPos-2)/4)/10;
line1[14]= 48+(byte)((menuPos-2)/4-(menuPos-2)/4/10*10);
line1[15]='\0';
strcpy(line2,"");

for(int i=3;i>=0;i--){ //////////////// number of holder ////////////////
char chBuff[2];
chBuff[0]=3-i+65;
chBuff[1]='\0';
strcat(line2,chBuff);

if((alarme[(menuPos-2)/4-1].relee & (byte)ceil(pow(2,i))){
strcat(line2,"X");
}else{
strcat(line2," ");
}
//if(i>0)strcat(line2,zile[7-i]);
}
}
}
}

```

```

// place the cursor according to our location in the submenu
void showSubMenu(){
  if(menuPos==1 || menuPos==2){
    switch(subMenuPos){
      case 1://days
        cursorPos = 1;
        break;
      case 2://months
        cursorPos = 4;
        break;
      case 3://years
        cursorPos = 7;
        break;
      case 4://day of the week
        if(menuPos==1){ // Only for the date we have the day of the week
          cursorPos=12;
          break;
        }
        default:
          subMenuPos = 0;
          cursorPos = -1;
          break;
    }
  }
  else if((menuPos+1)%4==0){ //alarm time
    switch(subMenuPos){
      case 1: //hour
        cursorPos = 1;
        break;
      case 2: //minutes
        cursorPos = 4;
        break;
      default:
        subMenuPos = 0;
        cursorPos = -1;
        break;
    }
  }
  else if((menuPos)%4==0){ // alarm days
    if(subMenuPos>0 && subMenuPos<8){
      cursorPos = subMenuPos*2-1;
    }else{
      subMenuPos = 0;
      cursorPos = -1;
    }
  }
  else if((menuPos-1)%4==0){ // duration of alarms
    if(subMenuPos==1){
      cursorPos = 0;
    }else if(subMenuPos==2){
      cursorPos=4;
    }else{
      subMenuPos=0;
      cursorPos = -1;
    }
  }
}

```

```

    }
    }else if((menuPos-2)%4==0){ //alarm holders
        if(subMenuPos>0 && subMenuPos<5){
            holder////////////////////////////////////
            cursorPos = subMenuPos*2-1;
        }else{
            subMenuPos = 0;
            cursorPos = -1;
        }
    }
}

void menu(){
    if((millis()-lastMillis)>150){
        if(digitalRead(PIN_MENU)==LOW && menuPos==0){
            menuPos=(menuPos+1)%(MAX_MENU_POS);
            showMenu();
        }else if(digitalRead(PIN_MENU)==LOW){
            subMenuPos++;
            showSubMenu();
        }
    }
    lastMillis=millis();
}

void readValues(){
    // we read the initial values in the eeprom.
    for(int i=0;i<NR_ALARME;i++){
        byte b = EEPROM.read(i*5+1);
        if(b==0xFF){
            b = 0;
            EEPROM.write(i*5+1,b);
        }
        (alarme[i]).hour = b & 31;    // positions 1, 6, 11 etc.
        (alarme[i]).unitAlarma = b >> 6;

        b = EEPROM.read(i*5+2);
        if(b==0xFF){
            b = 0;
            EEPROM.write(i*5+2,b);
        }
        (alarme[i]).minute = b;    // positions 2, 7, 12 etc.
        b = EEPROM.read(i*5+3);
        if(b==0xFF){
            b = 0;
            EEPROM.write(i*5+3,b);
        }
        (alarme[i]).days = b;    // positions 3, 8, 13 etc.
        b = EEPROM.read(i*5+4);
        if(b==0xFF){
            b = 0;
            EEPROM.write(i*5+4,b);
        }
    }
}

```

```

(alarme[i]).duration = b; // positions 4, 9, 14 etc.
b = EEPROM.read(i*5+5);
if(b==0xFF){
  b = 0;
  EEPROM.write(i*5+5,b);
}
(alarme[i]).relee = b; // positions 5, 10, 15 etc.
}
}

////////////////////////////////////// put your setup code here, to run
once://////////////////////////////////////

void setup(){

  pinMode(PIN_MENU,INPUT); // btn stg
  pinMode(PIN_DR,INPUT); // menu
  pinMode(PIN_STG,INPUT); // btn Dr

  for(int i=0;i<4;i++){
//////////////////////////////////////holder//////////////////////////////////////
    pinMode(relee[i],OUTPUT);
    digitalWrite(relee[i],OPEN);
  }
  digitalWrite(PIN_MENU,HIGH);
  digitalWrite(PIN_STG,HIGH);
  digitalWrite(PIN_DR,HIGH);
  attachInterrupt(0,menu,CHANGE);

////////////////////////////////////// stepper motor//////////////////////////////////////
  pinMode(smDirectionPin, OUTPUT);
  pinMode(smStepPin, OUTPUT);
//////////////////////////////////////
myservo.attach(9); // attaches the servo on pin 9 to the servo object
myservo.write(0);
pinMode(13,OUTPUT);
digitalWrite(13,LOW);

//alarmele
readValues();

//initialize DS_1307
Wire.begin();
lcd.begin();

// Turn on the backlight and print a message.
lcd.backlight();
Serial.begin(9600);
struct ds1307_time time;
time.second = 15;
time.minute = 55;

```

```

time.hour = 11;
time.dayOfWeek = 5;
time.dayOfMonth = 14;
time.month = 3;
time.year = 14;

//setDateDs1307(time);
////////////////////////////////////mp3////////////////////////////////////
/////
Serial.begin (9600);
mp3_set_serial (Serial); // DFPlayer-mini mp3 module
delay(1);
mp3_set_volume (30); // volume 0~30
////////////////////////////////////mp3////////////////////////////////////
////////////////////////////////////
}

char* getFmt2Char(byte val){
char *buf = "00\0";
buf[0] = 48+val/10;
buf[1] = 48+val%10;
return buf;
}

char* getDate(struct ds1307_time time){
char *buf=" \0";
buf[0]=48+time.dayOfMonth/10;
buf[1]=48+time.dayOfMonth%10;
buf[2]='.';
buf[3]=48+time.month/10;
buf[4]=48+time.month%10;
buf[5]='.';
buf[6]=48+time.year/10;
buf[7]=48+time.year%10;
return buf;
}

char* getTime(struct ds1307_time time){
char *buf=" \0";
buf[0]=48+time.hour/10;
buf[1]=48+time.hour%10;
buf[2]='.';
buf[3]=48+time.minute/10;
buf[4]=48+time.minute%10;
buf[5]='.';
buf[6]=48+time.second/10;
buf[7]=48+time.second%10;
return buf;
}

char* getZiuaSaptamaniiShort(byte val){
switch(val){

```

```

case 1:
    return "Mon\0"; //Mon
    break;
case 2:
    return "Tue\0"; //Tue
    break;
case 3:
    return "Wen\0"; //Wen
    break;
case 4:
    return "Thu\0"; //Thu
    break;
case 5:
    return "Fri\0"; //Fri
    break;
case 6:
    return "Sat\0"; //Sat
    break;
case 7:
    return "Sun\0"; //Sun
    break;
default:
    return "?\0";
    break;
}
}

```

```

char* getZiuaSaptamanii(byte val){
    switch(val){
        case 1:
            return "Monday\0"; //replace with Monday if you want
            break;
        case 2:
            return "Tuesday\0"; //replace with Tuesday if you want
            break;
        case 3:
            return "Wednesday\0"; //replace with Wednesday if you want
            break;
        case 4:
            return "Thursday\0"; //replace with Thursday if you want
            break;
        case 5:
            return "Friday\0"; //replace with Friday if you want
            break;
        case 6:
            return "Saturday\0"; //replace with Saturday if you want
            break;
        case 7:
            return "Sunday\0"; //replace with Sunday if you want
            break;
        default:
            return "?\0";
            break;
    }
}

```

```

}
}

void minusData(){
    if(subMenuPos==1 && time.dayOfMonth>1){ //days
        time.dayOfMonth--;
    }else if(subMenuPos==2 && time.month>1){
        time.month--;
    }else if(subMenuPos==3 && time.year>1){
        time.year--;
    }else if(subMenuPos==4 && time.dayOfWeek>1){
        time.dayOfWeek--;
    }
    dataDirty=true;
    showMenu();
}

void plusData(){
    if(subMenuPos==1){ // days here the user must be careful not to put more days dact has the moon, I only limited
to 31
        time.dayOfMonth=(time.dayOfMonth+1)%32;
        if(time.dayOfMonth==0){
            time.dayOfMonth=1;
        }
    }else if(subMenuPos==2){
        time.month=(time.month+1)%13;
        if(time.month==0){
            time.month=1;
        }
    }else if(subMenuPos==3){
        time.year=(time.year+1)%99;
    }else if(subMenuPos==4){
        time.dayOfWeek=(time.dayOfWeek+1)%8;
        if(time.dayOfWeek==0){
            time.dayOfWeek=1;
        }
    }
    dataDirty=true;
    showMenu();
}

void minusOra(){
    if(subMenuPos==1 && time.hour>0){ //days
        time.hour=time.hour-1;
    }else if(subMenuPos==2 && time.minute>0){
        time.minute=time.minute-1;
    }else if(subMenuPos==3 && time.second>1){
        time.second=time.second-1;
    }
    oraDirty = true;
    showMenu();
}

```

```

void plusOra(){
  if(subMenuPos==1){ //days
    time.hour=(time.hour+1)%24;
  }else if(subMenuPos==2){
    time.minute=(time.minute+1)%60;
  }else if(subMenuPos==3){
    time.second=(time.second+1)%60;
  }
  oraDirty = true;
  showMenu();
}

void minusAlarma(int pos){
  alarme[pos-1];
  if(subMenuPos==1 && alarme[pos-1].hour>0){ //zilele
    alarme[pos-1].hour--;
  }else if(subMenuPos==2 && alarme[pos-1].minute>0){
    alarme[pos-1].minute--;
  }
  showMenu();
}

void plusAlarma(byte pos){
  if(subMenuPos==1){
    alarme[pos-1].hour=(alarme[pos-1].hour+1)%24;
  }else if(subMenuPos==2){
    alarme[pos-1].minute=(alarme[pos-1].minute+1)%60;
  }
  showMenu();
}

void minusZileAlarma(byte pos){ // minus day of alarm

  Serial.println(alarme[pos-1].days);
  alarme[pos-1].days =alarme[pos-1].days ^ (byte)ceil(pow(2,(7-subMenuPos)));
  Serial.println(alarme[pos-1].days);

  showMenu();
}

void plusReleeAlarma(byte pos){ // plus alarm of holder
  minusReleeAlarma(pos);
}

void minusReleeAlarma(byte pos){ // minus alarm of holder

  Serial.println(alarme[pos-1].relee);
  alarme[pos-1].relee = alarme[pos-1].relee ^ (byte)ceil(pow(2,(4-subMenuPos)));
  //////////////////////////////////4444444444444444////////////////////////////////
  Serial.println(alarme[pos-1].relee);

  showMenu();
}

```

```

}

void plusZileAlarma(byte pos){
  minusZileAlarma(pos);
}

void minusUnitAlarma(byte pos){
  plusUnitAlarma(pos);
}

void plusUnitAlarma(byte pos){
  if(subMenuPos==2){
    alarme[pos-1].unitAlarma=(alarme[pos-1].unitAlarma+1)%2;
  }
}

void minusDurataAlarma(byte pos){ // minus duration of alarm
  if(alarme[pos-1].duration>0 && subMenuPos==1){
    alarme[pos-1].duration--;
  }else if(subMenuPos==2){
    minusUnitAlarma(pos);
  }
  showMenu();
}

void plusDurataAlarma(byte pos){ // plus duration of alarm
  if(subMenuPos==1){
    alarme[pos-1].duration=(alarme[pos-1].duration+1)%240;
  }else if(subMenuPos==2){
    plusUnitAlarma(pos);
  }
  showMenu();
}

void saveValues(){
  for(int i=0;i<NR_ALARME;i++){
    EEPROM.write(i*5+1,(alarme[i]).hour | (alarme[i].unitAlarma << 6));
    EEPROM.write(i*5+2,(alarme[i]).minute);
    EEPROM.write(i*5+3,(alarme[i]).days);
    EEPROM.write(i*5+4,(alarme[i]).duration);
    EEPROM.write(i*5+5,(alarme[i]).relee);
  }
}

void applyNewDateHour(){
  if(dataDirty || oraDirty){
    if(oraDirty){
      Serial.println("with time");
      setDateDs1307(time);
    }else{ // if the time has not changed and we leave it as it is
      Serial.println("no time");
      struct ds1307_time tmp;
    }
  }
}

```

```

    getDateDs1307(&tmp);
    tmp.dayOfMonth = time.dayOfMonth;
        tmp.dayOfWeek = time.dayOfWeek;
    tmp.month = time.month;
    tmp.year = time.year;
    setDateDs1307(tmp);
}
dataDirty=false;
oraDirty = false;
}
saveValues();
}
long valAlarmaMin(byte index){
    return (alarme[index]).hour*3600L+(alarme[index]).minute*60L;
}
long valAlarmaMax(byte index){
    byte multipl = (alarme[index]).unitAlarma?60:1;
    return (alarme[index]).hour*3600L+(alarme[index]).minute*60L+multipl*(alarme[index]).duration;
}
void checkAlarms(){
    long valComp = time.hour*3600L+time.minute*60L+time.second;
    bAlarma=false;
    strcpy(line,getTime(time));
    strcat(line,";A");
    char cAl[4];
    long timeToFinish=0,vAlMax=0;
    boolean tipTimp=0;
    byte statusRelee[4];
holder////////////////////////////////////
    for(int i=0;i<4;i++){
        statusRelee[i]=0;
    }
    for(int i=0;i<NR_ALARME;i++){
        struct alarm alarma = alarme[i];
        vAlMax=valAlarmaMax(i);
        if(valComp>=valAlarmaMin(i) && valComp<=vAlMax){ // Compare the minimum and maximum values of each
alarm to current time
            if((byte)ceil(pow(2,7-time.dayOfWeek)) & (alarma).days){
                bAlarma=true;
                if((timeToFinish==0 && vAlMax-valComp>timeToFinish) || (vAlMax-valComp<timeToFinish)){// set the
duration until the end of the first alarm
                    tipTimp = alarma.unitAlarma;
                    if(tipTimp && vAlMax-valComp>=60){
                        timeToFinish=(vAlMax-valComp)/60;
                    }else{
                        timeToFinish=vAlMax-valComp;
                        tipTimp=0;
                    }
                }
            }
            for(int j=3;j>=0;j--){
holder////////////////////////////////////
                if((alarma).relee & (byte)(ceil(pow(2,j)))){
                    statusRelee[j]=statusRelee[j] | 1;

```

```

    }
}

    itoa((i+1),cAl,10);
    if(strlen(line)<11){
        strcat(line,cAl);
    }
}
}
}
for(int i=0;i<4;i++){
//////////holder////////////////////////////////////
    if(statusRelee[i]){
        digitalWrite(relee[i],CLOSED);

    }else{
        digitalWrite(relee[i],OPEN);          //////////////////////////////////////////holder run
        //////////////////////////////////////////
    }
}
if(bAlarma){
    itoa(timeToFinish,cAl,10);
    strcat(line," ");
    strcat(line,cAl);
    if(tipTimp){
        strcat(line,"m");
    }else{
        strcat(line,"s");
    }
}
}

////////// 4 holder movment
//////////

void motion(){
    //////////////////////////////////////////stepper
    motor////////////////////////////////////
    if(digitalRead(PIN_COMMAND_A) == HIGH){
        Serial.println("Holder A move ");
        /*Here we are calling the rotate function to turn the stepper motor*/
        rotate(4500, 0.2); //The motor rotates 800 steps clockwise with a speed of 0.1 (slow)
        delay(3000);
        rotate(-4500, 0.2); //The motor rotates 1600 steps clockwise with a speed of 0.5 (medium)
        delay(3000);
        mp3_play (1); // sd:/mp3/0001.mp3
        a=1;
        delay(3000);
    }

    if(digitalRead(PIN_COMMAND_B) == HIGH){
        Serial.println("Holder B move ");
    }
}

```

```
    /*Here we are calling the rotate function to turn the stepper motor*/
    rotate(8500, 0.2); //The motor rotates 800 steps clockwise with a speed of 0.1 (slow)
    delay(3000);
    rotate(-4500, 0.2); //The motor rotates 1600 steps clockwise with a speed of 0.5 (medium)
    delay(3000);
    rotate(12500, 0.2); //The motor rotates 1600 steps counter clockwise with a speed of 1 (fast)
    delay(3000);
    mp3_play (2); // sd:/mp3/0002.mp3
    a=1;
    delay(3000);
    }

    if(digitalRead(PIN_COMMAND_C) == HIGH){
        Serial.println("Holder C move ");
        /*Here we are calling the rotate function to turn the stepper motor*/
        rotate(12500, 0.2); //The motor rotates 800 steps clockwise with a speed of 0.1 (slow)
        delay(3000);
        rotate(-4500, 0.2); //The motor rotates 1600 steps clockwise with a speed of 0.5 (medium)
        delay(3000);
        rotate(8500, 0.2); //The motor rotates 1600 steps counter clockwise with a speed of 1 (fast)
        delay(3000);
        mp3_play (3); // sd:/mp3/0003.mp3
        a=1;
        delay(3000);
        }

    if(digitalRead(PIN_COMMAND_D) == HIGH){
        Serial.println("Holder D move ");
        /*Here we are calling the rotate function to turn the stepper motor*/
        rotate(-4500, 0.2); //The motor rotates 1600 steps clockwise with a speed of 0.5 (medium)
        delay(3000);
        rotate(4500, 0.2); //The motor rotates 1600 steps counter clockwise with a speed of 1 (fast)
        delay(3000);
        mp3_play (4); // sd:/mp3/0004.mp3
        a=1;
        delay(3000);
        }

    ////////////////////////////////////////
}

////////////////////////////////////// stepper
motor//////////////////////////////////////
/*The rotate function turns the stepper motor. Tt accepts two arguments: 'steps' and 'speed'*/
void rotate(int steps, float speed){
    /*This section looks at the 'steps' argument and stores 'HIGH' in the 'direction' variable if */
    /*'steps' contains a positive number and 'LOW' if it contains a negative.*/
```

```

int direction;

if (steps > 0){
  direction = HIGH;
}else{
  direction = LOW;
}

speed = 1/speed * 70; //Calculating speed
steps = abs(steps); //Stores the absolute value of the content in 'steps' back into the 'steps' variable

digitalWrite(smDirectionPin, direction); //Writes the direction (from our if statement above), to the EasyDriver
DIR pin

/*Steppin'*/
for (int i = 0; i < steps; i++){
  digitalWrite(smStepPin, HIGH);
  delayMicroseconds(speed);
  digitalWrite(smStepPin, LOW);
  delayMicroseconds(speed);
}
}
////////////////////////////////////check Temperture and
humidity////////////////////////////////////

void checkTemp(){
  DHT.read11(dht_apin);

  Serial.print("Current humidity = ");
  Serial.print(DHT.humidity);
  Serial.print("% ");
  Serial.print("temperature = ");
  Serial.print(DHT.temperature);
  Serial.println("C ");

  if ((DHT.temperature) > 29.00 ){
    Serial.print("temperature High ");
    digitalWrite(13,HIGH);
    mp3_play (5); // sd:/mp3/0005.mp3
    delay(3000);
  }else{
    digitalWrite(13,LOW);
  }

  if ((DHT.humidity) > 60.00 ){
    Serial.print("humidity High ");
    mp3_play (6); // sd:/mp3/0006.mp3
    digitalWrite(13,HIGH);
    delay(3000);
  }else{
    digitalWrite(13,LOW);
  }
}
}

```

```

////////////////////////////////////
locker open and close////////////////////////////////////

```

```

void Locker(){
  // read the sensor:
  MzSensorValue_1 = analogRead(MzSensorPin_1);

  Serial.println(MzSensorValue_1);

```

```

if ( MzSensorValue_1 < 100 && a==1 )
{
  Serial.println(" Locker open");
  myservo.write(50);
  delay(9000);
  myservo.write(0);
  delay(100);
  a=0;
}

```

```

//////////////////////////////////// main code to run repeatedly
////////////////////////////////////

```

```

void loop(){
  // If we have the menu pressed, we show what we need and do not show the current date and time
  if(menuPos>0){
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print(line1);
    lcd.setCursor(0,1);
    lcd.print(line2);
    if(digitalRead(PIN_STG)==LOW && menuPos){ // we hit minus //the arduino pin used for the left button in the
menu
      delay(100);
      if(subMenuPos){
        if(menuPos==1){ //data
          minusData();
        }else if(menuPos==2){
          minusOra();
        }else if((menuPos+1)%4==0){
          minusAlarma((menuPos+1)/4);
        }else if((menuPos)%4==0){ //days alarm
          minusZileAlarma((menuPos)/4);
        }
      }
    }
  }
}

```

```

    }else if((menuPos-1)%4==0){ //duration alarm
        minusDurataAlarma((menuPos-1)/4);
    }else if((menuPos-2)%4==0){ //holder alarm
        minusReleeAlarma((menuPos-2)/4);
    }
}
else{
    if(menuPos==1){
        applyNewDateHour();
    }
    menuPos--;
    showMenu();
}
}else if(digitalRead(PIN_DR)==LOW && menuPos){ // we hit plus
    delay(200);
    if(subMenuPos){
        if(menuPos==1){ //data
            plusData();
        }else if(menuPos==2){
            plusOra();
        }else if((menuPos+1)%4==0){ //time alarm
            plusAlarma((menuPos+1)/4);
        }else if((menuPos)%4==0){ //days alarm
            plusZileAlarma((menuPos)/4);
        }else if((menuPos-1)%4==0){ //duration alarm
            plusDurataAlarma((menuPos-1)/4);
        }else if((menuPos-2)%4==0){ //holder alarm
            plusReleeAlarma((menuPos-2)/4);
        }
    }
}
else if(menuPos<MAX_MENU_POS-1){
    ++menuPos;
    Serial.print("Menu pos: ");
    Serial.print(menuPos);
    showMenu();
}else{
    applyNewDateHour();
    menuPos=0;
}
}else if(subMenuPos>0 && cursorPos>-1){

//    Serial.println(cursorPos);

    lcd.cursor();
    lcd.setCursor(cursorPos,1);
}else{

    lcd.noCursor();

}

delay(100);
return;

```

```

}

lcd.clear();

char buf[3];
getDateDs1307(&time);
Serial.print(getTime(time));
Serial.print(" ");
Serial.print(getDate(time));
Serial.print(":");
Serial.print("Day of the week: ");
Serial.println(time.dayOfWeek, DEC);

lcd.setCursor(0, 0);
lcd.print(getZiuaSaptamanii(time.dayOfWeek));
lcd.setCursor(8,0);
lcd.print(getDate(time));
if(bAlarma){
  lcd.setCursor(0,1);
  lcd.print(line);
}else{
  lcd.setCursor(4,1);
  lcd.print(getTime(time));
  lcd.setCursor(12,1);
  lcd.print(" ");
  lcd.noCursor();
}
checkAlarms();
motion();
checkTemp();
Locker();

delay(1000);

```