Table of contents

INTRODUCTION

In this era of modern medicine where humans are largely dependent on the use of pills/tablets we know at least one or more who have taken their medication in the form of pills/tablets in order to live a healthy life.

This is a pill dispenser robot able to provide the patient the correct amount and type of medicine pills. The dosing of the pill is performed automatically at the correct time of the day, preceded by an alarm. When empty, the machine is easily refilled by the user. The dispensing and refilling mechanism is controlled by means of an application connected via Bluetooth to the robot and by means of buttons.

In this project, we focus on an automated machine that will help a person to take his/her pills on time according to a schedule and mainly focus on making sure that your loved ones who are either old aged or having memory loss or have difficulty in remembering the medicine schedule take their pills on time from the touch of your phone around the world.

The project includes designing and fabricating the body and the parts of the final product. Design will be done using software's like Solid Works or AutoCAD. The physical product will have PCB circuits and a Bi-directional rotating mechanism which will be enclosed inside the body of the product. This product will also be Bluetooth enabled and has a dedicated smart app which can be downloaded on the phone. It also has a coloured LCD panel screen which is mounted on the body and be accessed with the buttons provided on the body.

LITERATURE REVIEW

In this section we are going to see the Literature study on our concept that was previously done by some engineers, scientists and other personalities. Our literature study includes mainly 4 articles they were:

1. Smart Pill Box Health Care System.

2. Improving healthcare using Smart Pill Box for Medicine Reminder and Monitoring System.

3. A Comprehensive Approach for a Smart Medication Dispenser.

4. Design and Development of Smart Medicine Box.

1. **A Comprehensive Approach for a Smart Medication Dispenser (July 2019)**
   By Abdallah Kassem , Wissam Antoun , Mustapha Hamad and Chady El-Moucary.

   This paper presents a comprehensive approach for a Smart Medicine Dispenser (SMD) prototype. The main purpose of the proposed system is to help patients, mainly seniors and elderly people, take their medications on time in an easy manner without the possibility of skipping pills and thus reducing the risk of accidental over/under dose treatment. An Android application is developed that is responsible for controlling the whole system as it constitutes a data base awaited to be synchronized and on synchronization the data is sent by the application that determines which motor should be rotated.

2. **Improving healthcare using Smart Pill Box for Medicine Reminder and Monitoring System** by Diaa Salama, Abdul Minaam.

   This paper consists on the conception, design and creation of a pillbox prototype intended to solve the deficiency in the medical area as it has the ability of sorting out the pills by itself. The medication pill box is focused on patients who frequently take medications or vitamin supplements, or attendants who deal with the more seasoned or patients. SMART AUTOMATED PILL DISPENSER 2020-

2021 Dept. of Mechanical Engg. Dr.TTIT 6 It has 9 compartment boxes as the previous paper consists three and it has alert remainder set through android application. The pillbox will remind clients or patients to take pills utilizing sound and light.

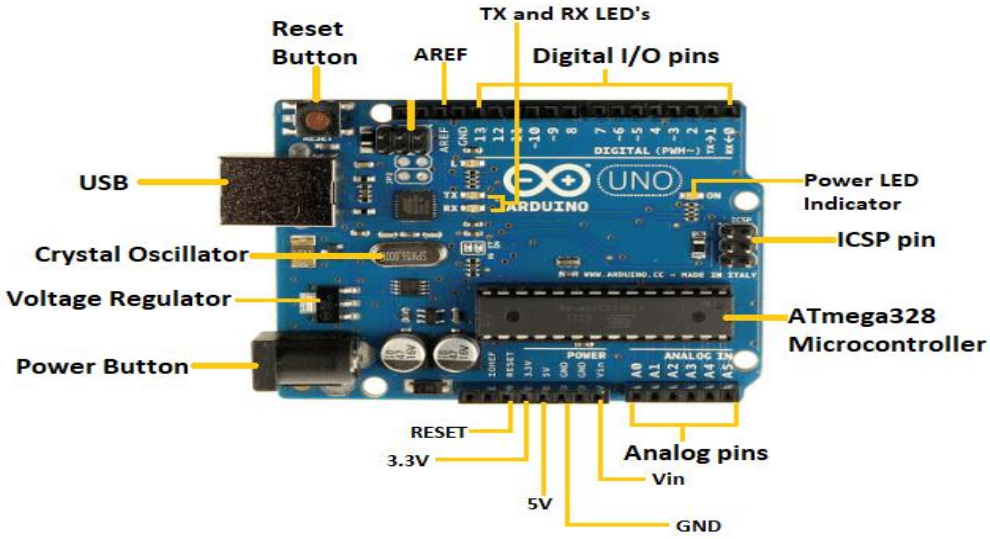
COMPONENT REQUIRED

1. Arduino uno R3
2. Stepper motor
3. Motor driver
4. Real time clock
5. Buzzer
6. Humidity temperature sensor
7. LCD
8. PIR sensor
9. Push Button
10. Jumper wire
11. Bread board
12. LEDs

→Briefly explain about all the components which are used in this project.

Arduino Uno

The Arduino UNO is a standard board of Arduino. Here UNO means 'one' in Italian. It was named as UNO to label the first release of Arduino Software. It was also the first USB board released by Arduino. It is considered as the powerful board used in various projects. Arduino.cc developed the Arduino UNO board.



Arduino UNO is based on an ATmega328P microcontroller It is easy to use compared to other boards, such as the Arduino Mega board, etc. The board consists of digital and analog Input/Output pins (I/O), shields, and other circuits.

The Arduino UNO includes 6 analog pin inputs, 14 digital pins, a USB connector, a power jack, and an ICSP (In-Circuit Serial Programming) header. It is programmed based on IDE, which stands for Integrated Development Environment. It can run on both online and offline platforms.

The IDE is common to all available boards of Arduino.

Components of Arduino in detail:

- **ATmega328 Microcontroller**- It is a single chip Microcontroller of the ATmel family. The processor code inside it is of 8-bit. It combines Memory (SRAM, EEPROM, and Flash), Analog to Digital Converter, SPI serial ports, I/O lines, registers, timer, external and internal interrupts, and oscillator**.**
- **ICSP pin** - The In-Circuit Serial Programming pin allows the user to program using the firmware of the Arduino board.
- **Power LED Indicator**- The ON status of LED shows the power is activated. When the power is OFF, the LED will not light up.
- **Digital I/O pins**- The digital pins have the value HIGH or LOW. The pins numbered from D0 to D13 are digital pins.
- **TX and RX LED's**- The successful flow of data is represented by the lighting of these LED's.
- **AREF-** The Analog Reference (AREF) pin is used to feed a reference voltage to the Arduino UNO board from the external power supply.
- **Reset button**- It is used to add a Reset button to the connection.

- **USB**- It allows the board to connect to the computer. It is essential for the programming of the Arduino UNO board.
- **Crystal Oscillator**- The Crystal oscillator has a frequency of 16MHz, which makes the Arduino UNO a powerful board.
- **Voltage Regulator**- The voltage regulator converts the input voltage to 5V.
- **GND**- Ground pins. The ground pin acts as a pin with zero voltage.
- **Vin**- It is the input voltage.
- **Analog Pins**- The pins numbered from A0 to A5 are analog pins. The function of Analog pins is to read the analog sensor used in the connection. It can also act as GPIO (General Purpose Input Output) pins.

Arduino Uno Specifications

The specifications of Arduino Uno is as given in the table below.

Arduino Uno Specifications Table

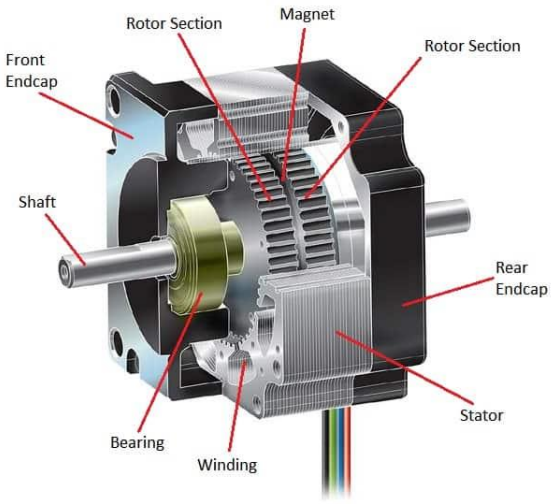
Microcontroller	ATmega38P – 8 bit AVR family microcontroller
Operating Voltage	5V
Recommended Input Voltage	7-12V
Input Voltage Limits	6-20V
Analog Input Pins	6 (A0-A5)
Digital I/O Pins	14 (Out of which 6 provide PWM output)
DC Current on I/O Pins	40mA
DC Current on 3.3V Pin	50mA
Flash Memory	32 KB (0.5 KB is used for Bootloader)
SRAM	2kB
EEPROM	1kB
Frequency (Clock Speed)	16MHz

Stepper Motor

A stepper motor is an electric motor whose main feature is that its shaft rotates by performing steps, that is, by moving by a fixed number of degrees. This feature is obtained thanks to the internal structure of the motor, and allows to know the

exact angular position of the shaft by simply counting how may steps have been performed, with no need for a sensor. This feature also makes it fit for a wide range of applications.

The **stepper motor working principle** is Electro-Magnetism. It includes a rotor which is made with a permanent magnet whereas a stator is with electromagnets. Once the supply is provided to the winding of the stator then the magnetic field will be developed within the stator. Now rotor in the motor will start to move with the rotating magnetic field of the stator. So, this is the fundamental working principle of this motor.



In this motor, there is a soft iron that is enclosed through the electromagnetic stators. The poles of the stator as well as the rotor don't depend on the kind of stepper. Once the stators of this motor are energized then the rotor will rotate to line up itself with the stator otherwise turns to have the least gap through the stator. In this way, the stators are activated in a series to revolve the stepper motor.

STEPPER MOTAR SPECIFICATION

Rated voltage ——- 5VDC

Number of Phase—— 4

Speed Variation Ratio ——1/64

Stride Angle ————5.625° /64

Frequency ————-100HzDC

resistance————- 50Ω±7%(25°C)

Idle In-traction Frequency—— > 600HzIdle

Out-traction Frequency —–> 1000Hz

In-traction Torque ————————->34.3mN.m(120Hz)

Self-positioning Torque ————->34.3mN.m

Friction torque———————— 600-1200 gf.cm

Pull in torque————————- 300 gf.cm

Insulated resistance ————————>10MΩ(500V)

Insulated electricity power ——600VAC/1mA/1s

Insulation grade————————- A

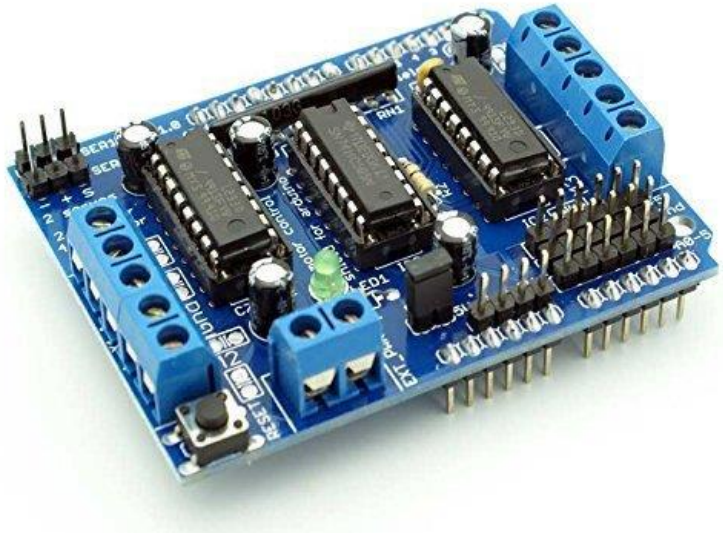
Rise in Temperature————— <40K(120Hz)

Noise ————————————-<35dB(120Hz,No load,10cm)

Model ————————————28BYJ-48 – 5V

Arduino Motor Driver

The L298N Motor Driver is a controller that uses an H-Bridge to easily control the direction and speed of up to 2 DC motors. The module can drive DC motors that have voltages between 5 and 35V, with a peak current up to 2A.



Instructions of Using motor driver:

1.) The controller can take up to 2 motors. Plug one motor into the terminal labelled OUT1 and OUT2. Plug the second motor into the terminal labelled OUT3 and OUT4:
2.) The row of pins on the bottom right of the L298N control the speed and direction of the motors. IN1 and IN2 control the direction of the motor connected to OUT1 and OUT2. IN3 and IN4 control the direction of the motor connected to OUT3 and OUT4
3.) You can power the L298N with up to 12V by plugging your power source into the pin on the L298N labelled "12V". The pin labelled "5V" is a 5V output that you can use to power your Arduino
4.) Write the code. Setting IN1 to HIGH and IN2 to LOW will cause the left motor to turn a direction. Setting IN1 to LOW and IN2 to HIGH will cause the left motor to spin the other direction. The same applies to IN3 and IN4.
5.) You can change the speed with the EN pins using PWM. ENA controls the speed of the left motor and ENB controls the speed of the right motor. Here I plugged them into pins 9 and 10 on the Arduino. This is optional and the motors will still run if you don't do this.
6.) To change the speed in the code, use the analogWrite() function (more info) on the ENA and ENB pins.

Real Time Clock

DS3231 is a low-cost, extremely accurate I2C real-time clock (RTC) with an integrated temperature-compensated crystal oscillator (TCXO) and crystal. The device incorporates a battery input, and maintains accurate timekeeping when main power to the device is interrupted.

The module can work on either 3.3 or 5 V which makes it suitable for many development platforms or microcontrollers. The battery input is 3V and a

typical CR2032 3V battery can power the module and maintain the information for more than a year.

Specification of RTC Module

- Highly Accurate RTC Completely Manages All Timekeeping Functions
- Real-Time Clock Counts Seconds, Minutes, Hours, Date of the Month, Month, Day of the Week, and Year, with Leap-Year Compensation Valid Up to 2100
- Accuracy ±2ppm from 0°C to +40°C
- Accuracy ±3.5ppm from -40°C to +85°C
- Digital Temp Sensor Output: ±3°C Accuracy
- Two Time-of-Day Alarms
- Programmable Square-Wave Output Signal
- Simple Serial Interface Connects to Most Microcontrollers
- Fast (400kHz) I2C Interface
- Battery-Backup Input for Continuous Timekeeping
- Low Power Operation Extends Battery-Backup Run Time
- 3.3V Operation

HOW DOES IT WORK?

The DS3231 real time clock module keeps track of the time even when the module is not powered. It has a built-in 3V battery which keeps updating the time. We will get the time and date from the RTC module using the library functions and then we will compare this time with the alarm time that we have set in the code.

Most RTC's use an external 32kHz timing crystal that is used to keep time with low current draw. And that's all well and good, but those crystals have slight drift, particularly when the temperature changes (the temperature changes the oscillation frequency very very very slightly but it does add up!) This RTC is in a beefy package because the crystal is inside the chip! And right next to the integrated crystal is a temperature sensor. That sensor compensates for the frequency changes by adding or removing clock ticks so that the timekeeping stays on schedule

Buzzer

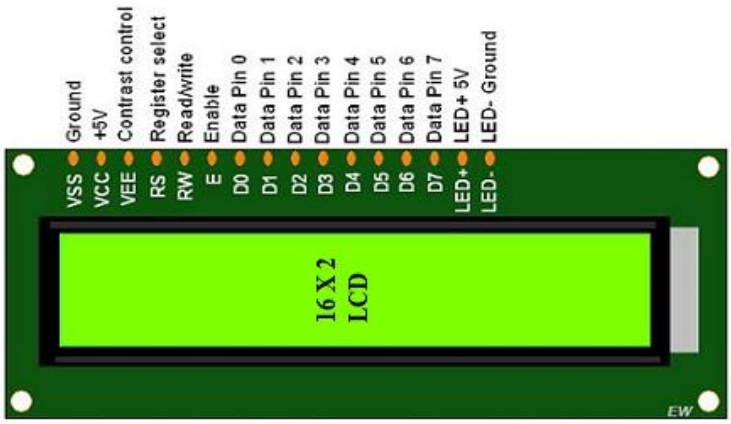
There are many ways to communicate between the user and a product. One of the best ways is audio communication using a buzzer.

An audio signalling device like a beeper or buzzer may be electromechanical or piezoelectric or mechanical type. The main function of this is to convert the signal from audio to sound. Generally, it is powered through DC voltage and used in timers, alarm devices, printers, alarms, computers, etc. Based on the various designs, it can generate different sounds like alarm, music, bell & siren.

The pin configuration of the buzzer is shown. It includes two pins namely positive and negative. The positive terminal of this is represented with the '+' symbol or a longer terminal. This terminal is powered through 6Volts whereas the negative terminal is represented with the '-'symbol or short terminal and it is connected to the GND terminal.

LCD

The term LCD stands for liquid crystal display. It is one kind of electronic display module used in an extensive range of applications like various circuits & devices like mobile phones, calculators, computers, TV sets, etc. These displays are mainly preferred for multi-segment light-emitting diodes and seven segments. The main benefits of using this module are inexpensive; simply programmable, animations, and there are no limitations for displaying custom characters, special and even animations, etc.



The LCD 16×2 working principle is, it blocks the light rather than dissipate. A 16×2 LCD has two registers like data register and command register. The RS (register select) is mainly used to change from one register to another. When the register set is '0', then it is known as command register. Similarly, when the register set is '1', then it is known as data register.

LCD 16×2 Pin Diagram

The 16×2 LCD pinout is shown above.

- Pin1 (Ground/Source Pin): This is a GND pin of display, used to connect the GND terminal of the microcontroller unit or power source.
- Pin2 (VCC/Source Pin): This is the voltage supply pin of the display, used to connect the supply pin of the power source.
- Pin3 (V0/VEE/Control Pin): This pin regulates the difference of the display, used to connect a changeable POT that can supply 0 to 5V.
- Pin4 (Register Select/Control Pin): This pin toggles among command or data register, used to connect a microcontroller unit pin and obtains either 0 or 1(0 = data mode, and 1 = command mode).
- Pin5 (Read/Write/Control Pin): This pin toggles the display among the read or writes operation, and it is connected to a microcontroller unit pin to get either 0 or 1 (0 = Write Operation, and 1 = Read Operation).
- Pin 6 (Enable/Control Pin): This pin should be held high to execute Read/Write process, and it is connected to the microcontroller unit & constantly held high.
- Pins 7-14 (Data Pins): These pins are used to send data to the display. These pins are connected in two-wire modes like 4-wire mode and 8-wire mode. In 4-wire mode, only four pins are connected to the

microcontroller unit like 0 to 3, whereas in 8-wire mode, 8-pins are connected to microcontroller unit like 0 to 7.

- Pin15 (+ve pin of the LED): This pin is connected to +5V
- Pin 16 (-ve pin of the LED): This pin is connected to GND.

PIR Sensor

The electronic sensor used to detect the movement of human being within a certain range of the sensor is called as PIR sensor or passive infrared sensor (approximately have an average value of 10m, but 5m to 12m is the actual detection range of the sensor). Fundamentally, pyroelectric sensors that detect the levels of infrared radiation are used to make PIR sensors.

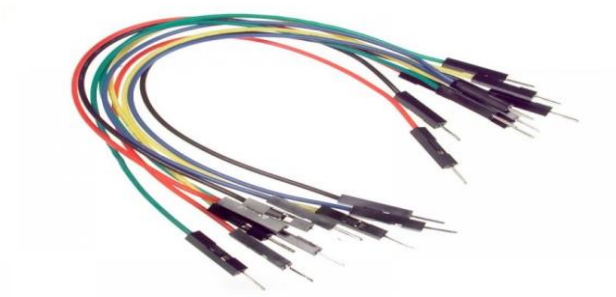
The PIR sensor circuit is used in numerous electronics projects which are used to discover a human being entering or leaving the particular area or room. These passive infrared sensors are flat control, consists of a wide range of lens, and PIR sensors can be easily interfaced with electronics circuits.

PIR sensor consists of three pins, ground, signal, and power at the side or bottom. Generally, the PIR sensor power is up to 5V, but, the large size PIR modules operate a relay instead of direct output. It is very simple and easy to interface the sensor with a microcontroller. The output of the PIR is (usually digital output) either low or high.

Jumper Wires

Jumper wires are simply wires that have connector pins at each end, allowing them to be used to connect two points to each other without soldering. Jumper wires are typically used with breadboards and other prototyping tools in order to

make it easy to change a circuit as needed. Fairly simple. In fact, it doesn't get much more basic than jumper wires.



Things can get messy fast, especially when you start to use a lot of components or build complicated circuits. Using jumper wires in a smart way can help take some of the headaches out of sanity checking your circuits.

Using color coding is breathtaking simple but will save you so much time in the long run.

The two most normal conventions are:

- Black (or sometimes Blue): This denotes a wire running directly to Ground (GND)
- Red: This is connected to a source of power. Normally running from the 3.3V on your Particle board.

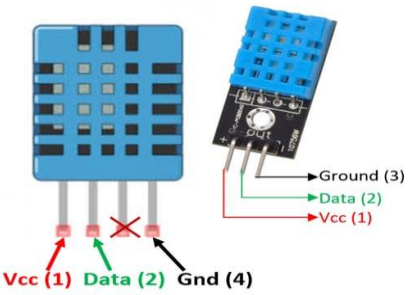
When working with other components, choose a consistent color. For example, if working with an RGB led, you might use a Green, Orange (because red is for power) and blue cable to connect to each of the color terminals on the LED.

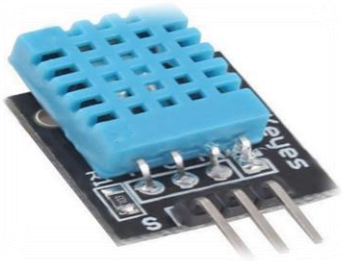
Servos normally have three wired connections, brown/black, orange/red and yellow. Match the colors. Match the jumper wires connecting to your breadboard to make it quick and easy to check connections.

Humidity Temperature Sensor

A temperature sensor is an electronic device that measures the temperature of its environment and converts the input data into electronic data to record monitor or signal temperature changes. There are many different types of temperature sensors. Here you can get a variety of Temperature & Humidity sensor which includes Digital Microcomputer Thermostat Switch, Humidity Controller Module, high-temperature resistance Probe, Moisture Sensor and many more module

The **DHT11** is a commonly used **Temperature and humidity sensor that** comes with a dedicated NTC to measure temperature and an 8-bit microcontroller to output the values of temperature and humidity as serial data.





DHT11 Specifications

- Operating Voltage: 3.5V to 5.5V
- Operating current: 0.3mA (measuring) 60uA (standby)
- Output: Serial data
- Temperature Range: 0°C to 50°C
- Humidity Range: 20% to 90%
- Resolution: Temperature and Humidity both are 16-bit
- Accuracy: ±1°C and ±1%

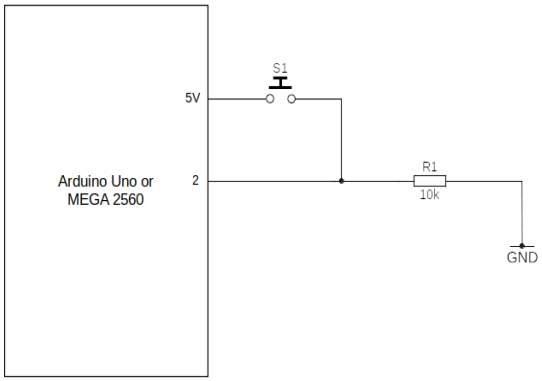
Push Button

A push button switch called a *momentary push button switch* is used in this tutorial. Momentary means that the switch stays closed only while pushed. When the switch is released, the contacts open. The image below shows examples of this type of switch. Use a wire link instead if you do not have a push button switch. Connect the wire link to the breadboard circuit to simulate closing the switch, and disconnect the wire link to simulate opening the switch.

The following image is a circuit diagram of the previous two breadboard circuits. R1 is a 10k resistor that pulls Arduino pin 2 to GND. With the switch S1 open, a voltage level of 0V is read on pin 2 by the Arduino. When the switch is closed, 5V is attached to pin 2 of the Arduino. In this case, the Arduino sees 5V on pin 2. A sketch reads 0V and 5V on a digital pin as 0 and 1 logic levels. An article on

the basics of switches for beginners shows different symbols used for switches in circuit diagrams.

Circuit Diagram of Arduino Push Button Switch



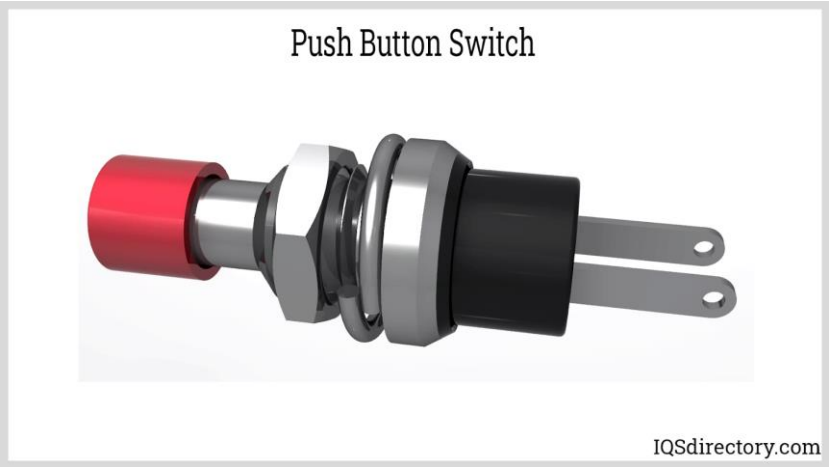
Push button switch

A push button switch is a simple mechanism that initiates power to a machine, device, or appliance. The body is typically metal or plastic, featuring a durable and ergonomic design. With switches available that provide latching or momentary action, there are push button switches for virtually any relevant application

How Does a Push Button Switch Work?

Push button switches can serve many purposes, but each type operates on a similar principle. When an operator applies pressure to the push button actuator, an internal spring depresses. Contacts attached to the spring connect with the electrical contacts on the switch's lower end to open or close the electrical circuit. Depending on the switch, releasing the button or pressing it again will retract the spring and return the circuit to its original state.

The initial state of the push button circuit can be open or closed, depending on the application. The type of switch will also depend on the application.



Push button switch types

Push button switches come in two types: momentary switches or push-pull switches.

- **Momentary:** The momentary push button switch has a single pole and is initially in an off state. When the operator presses the push button switch, it changes to on. There are also double-pole momentary switches that provide an additional state of functionality.
- **Push-pull:** Push-pull switches are typically in an off state until an operator presses the button and engages the actuator. The machine or device will remain on until an operator pulls the actuator to its initial position.

LED

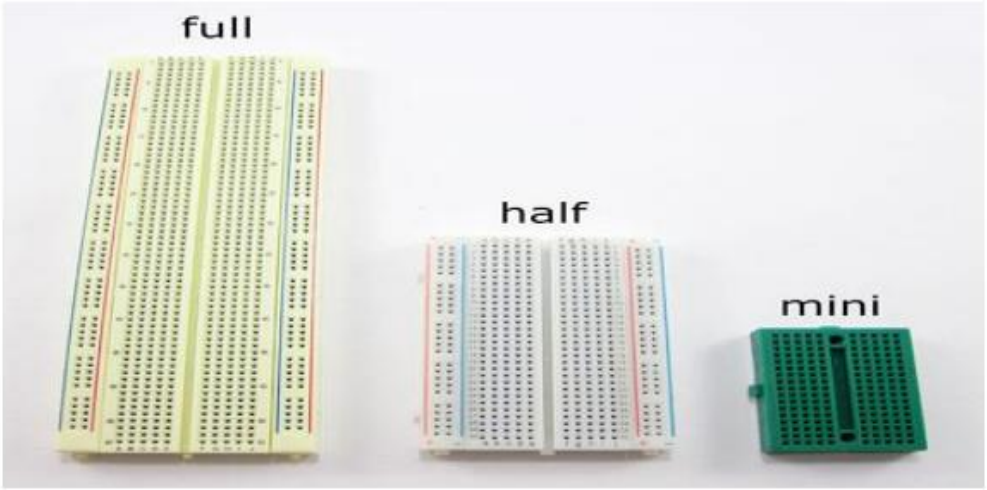
The light emitting diode is the most visible type of semiconductor diode. They emit a fairly narrow bandwidth of either visible light at different coloured wavelengths, invisible infra-red light for remote controls or laser type light when a forward current is passed through them.

The "**Light Emitting Diode**" or LED as it is more commonly called, is basically just a specialised type of diode as they have very similar electrical characteristics to a PN junction diode. This means that an LED will pass current in its forward direction but block the flow of current in the reverse direction.

Light emitting diodes are made from a very thin layer of fairly heavily doped semiconductor material and depending on the semiconductor material used and the amount of doping, when forward biased an LED will emit a coloured light at a particular spectral wavelength.

Bread board

A breadboard is a rectangular plastic board with a bunch of tiny holes in it. These holes let you easily insert electronic components to prototype (meaning to build and test an early version of) an electronic circuit, like this one with a battery, switch, resistor, and an LED (light-emitting diode). Modern breadboards are made from plastic, and come in all shapes, sizes, and even different colors. While larger and smaller sizes are available, the most common sizes you will probably see are "full-size," "half-size," and "mini" breadboards. Most breadboards also come with tabs and notches on the sides that allow you to snap multiple boards together. However, a single half-sized breadboard is sufficient for many beginner-level projects.



Technical Hints on the Choice of the Components

The mechanisms of dispensing and of refilling require great precision and little movements of the wheels that contain the pills. For this reason, we decide to use two stepper motors.

Stepper Motors are stables, can drive a wide range of frictional and inertial loads, don't need feedback. The motor is also a position transducer: sensors of position and speed are not required. Moreover, they have excellent repeatability and return to the same location accurately

A Motor Shield drives the two stepper motors. It contains 4 H-Bridge that allow to control both direction and speed of the motors. Using a motor shield, we

increase the number of free pins. To be sure that pills are always in good conditions.

A Humidity and Temperature sensors measures constantly the temperature and humidity inside the dispenser. To notify to the user that it is time to take his therapy we built an alarm with a Buzzer and a Real-Time Clock.

The RTC module runs on a battery and can keep track of the time even if we reprogram the microcontroller or disconnect the main power.

Two Buttons and a RGB Liquid Crystal Display permit to the user to interact with the dispenser.

The user can also set his therapy and the dispensing time through an App for smartphone. He can link his personal device via Bluetooth connection (a Bluetooth module is connected to Arduino).

A PIR sensor detects a movement if the user takes his medicine and gives feedback of the correct work of the dispenser. Because of its great sensibility and its wide range of detection, it is intentionally obstacle in some directions to avoid useless measurements.

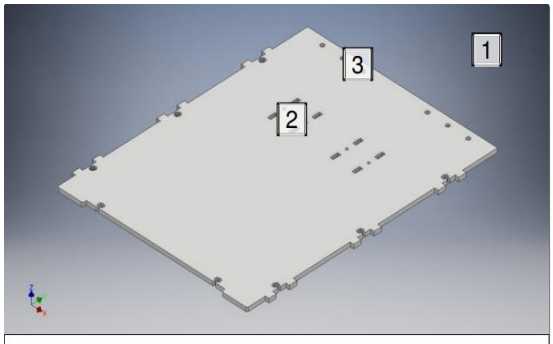
Technical Drawings for box container

The assembly of the box is design in order to avoid the use of glue. This allow to realize a cleaner work and, if needed, disassembly can be done to fix some issues. In particular, the assembly is performed by means of bolts and nuts. In a hole of a proper geometry, a bolt from one side, and a nut from the other side, perfectly fit in in order to have a strong connection between all the mdf plates. In particular for what regards the various plates:

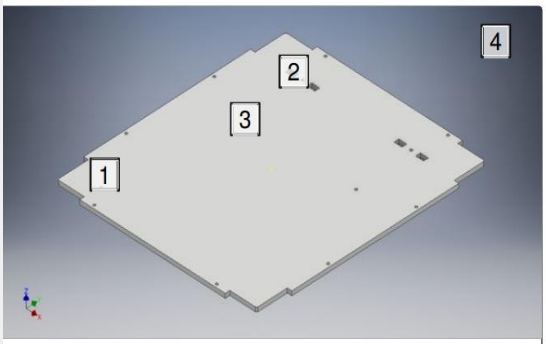
- The lateral plate has a hole positioned in order to let the cable passing through so as to have a connection between the Arduino and the computer.
- The frontal plate has 2 apertures. The lowest one is intended to be used when the person has to take the glass where the pill has been dispensed. The other one is used when it's time for refill. In this particular situation there is a plug (see later the design) that can close the aperture on the cap of the wheel from below. The positioning of this cap is indeed performed by exploiting this second aperture. Once the plug is positioned, using the buttons or the app, the person can let the wheel rotate one section at a time and place a pill in each section.
- The sustain plate is positioned in order to have a vertical support for the rails where the wheel and the cap are positioned so as to have a more reliable and stiff structure.

- The opening plate is designed as the word says in order to facilitate the refilling mechanism by the user

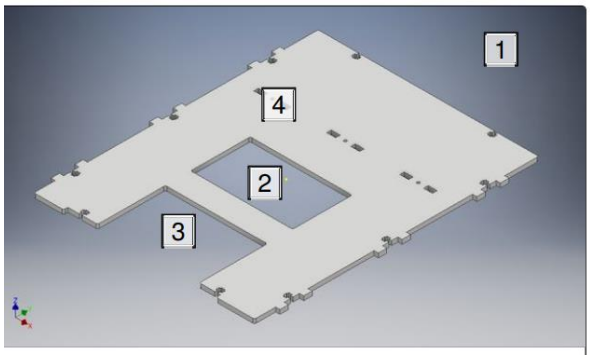
- The top plate, as can be seen from the picture, is done in plexiglass in order to enable from outside the vision of what's happening inside.



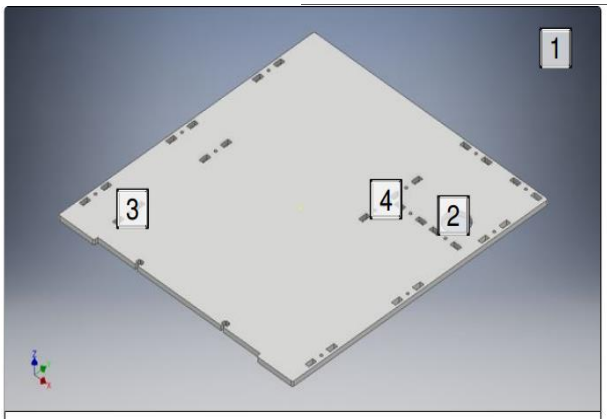
1. Back plate
2. Holes for the connection with the rails supporting the wheel and the wheel cap
3. holes for the connection with the opening plate through a classical door hinge



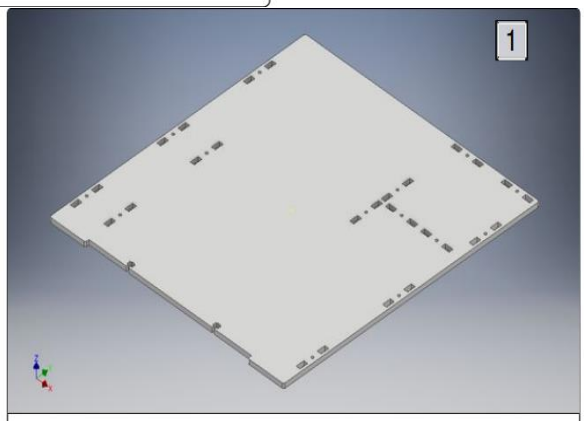
1. Holes for PIR holder connection
2. connection with the sustain plate
3. holes for connection with the funnel
4. Base plate



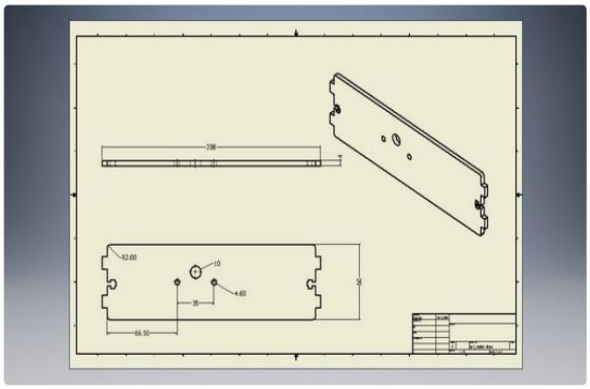
1. Frontal Plate
2. Aperture for the refill
3. Aperture to take the pill
4. Connection with the Arduino plate

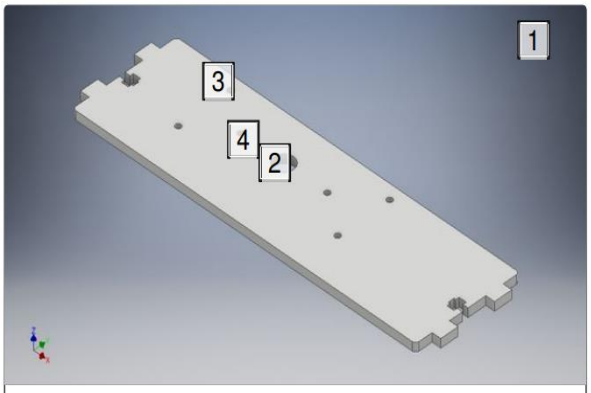


1. Lateral plate (with the hole for the Arduino)
2. Hole for powering the Arduino
3. Connection with the sustain plate
4. Connections with the Arduino plate and the connector plate



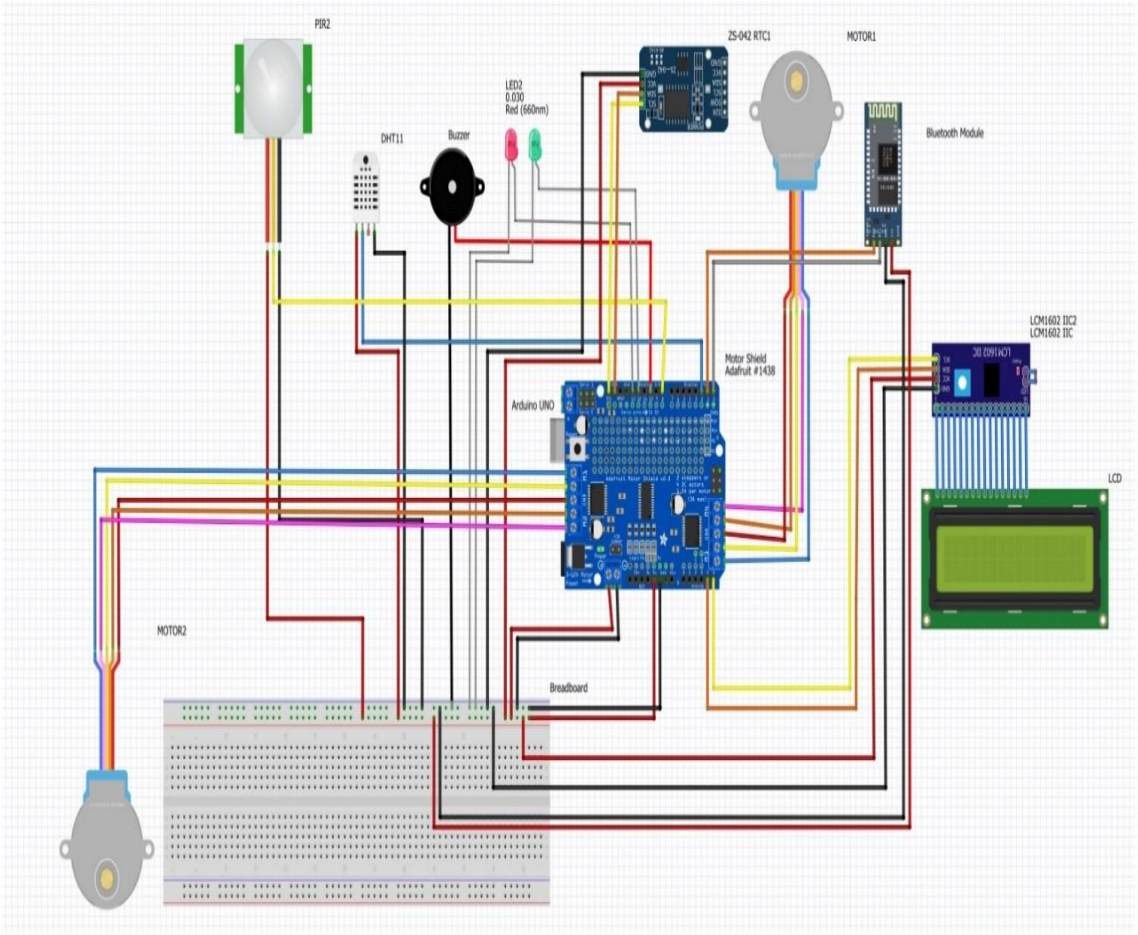
1. Lateral plate



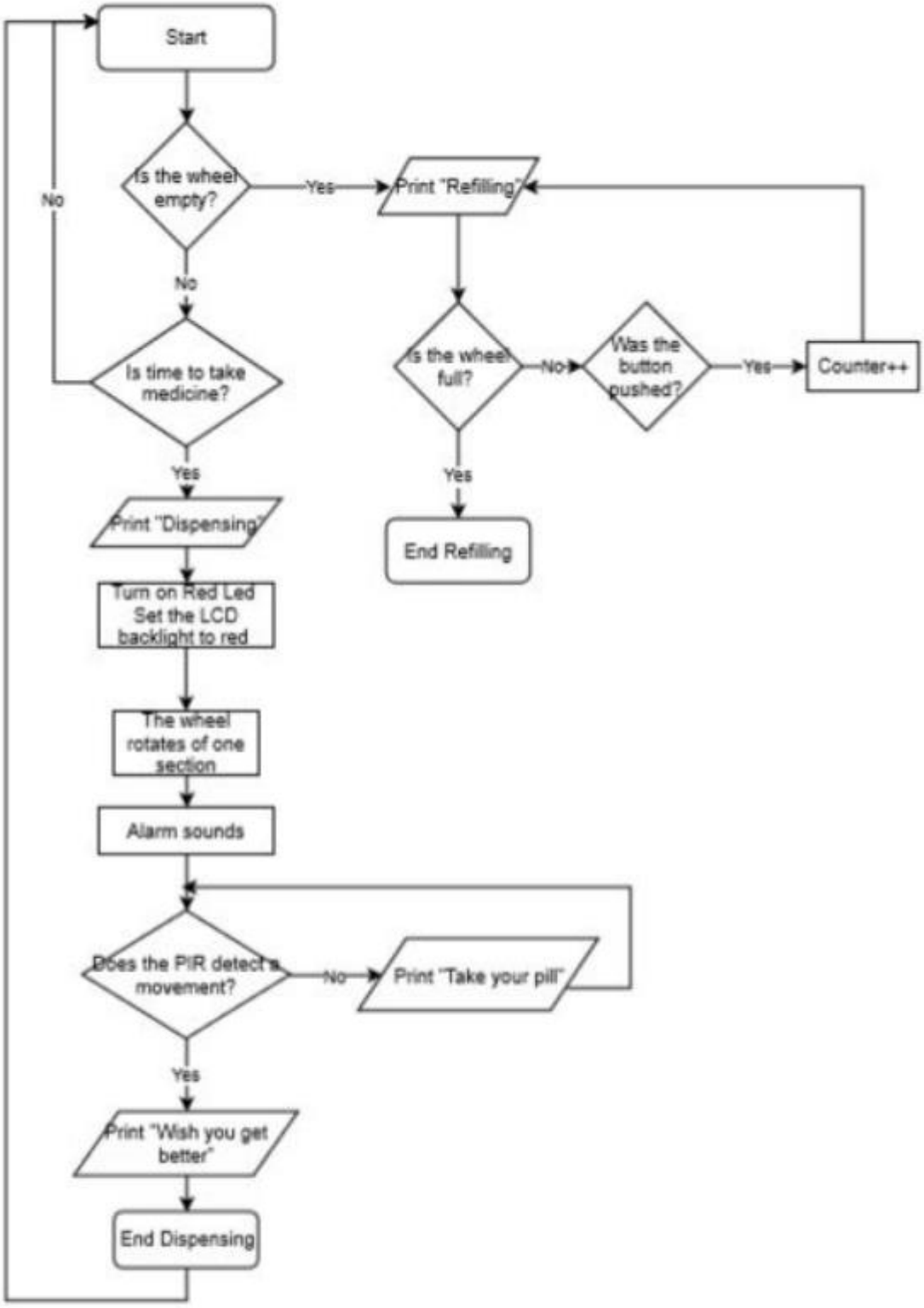


1. plate for the connection of the cap of the wheel and the bearing holder
2. holes for letting the wheel shaft passing through
3. 4 holes for the cap connection
4. 2 holes for bearing holder connection

Circuit Diagram



Program flowchart



Programming Part

```
#include <Wire.h>   //including the wire library
#include "RTClib.h" //including the adafruit real time clock library
#include <Adafruit_MotorShield.h> //include the adafruit motor shield library

#include <LiquidCrystal_I2C.h> //note in case you have an I2c liquid crystal driver you can use this library instead of the rgb library
#include <DHT.h> //including the temperature and humidity sensor library

int tx=1; //connected to RXD bluetooth pin
int rx=0; //connected to TVD bluetooth pin
int ledPinR = 13; //definig integer for the pin for red led pin
int ledPinG = 12; //definig integer for the pin for green led pin
int blue; //defining integer for storing the value of recieved by bluetooth module
int pirPin = 8; // defining integer for the pin for the PIR sensor
int pirval = 0; // defining integer to store the PIR measurements
int pinDHT = 2; // defining integer for the pin for the DHT11 sensor
DHT dht(2,DHT11); //definig the type of the temperature and humidity sensor which is the DHT11
LiquidCrystal_I2C lcd(0x27,16,2);  // set the LCD address to 0x27 for a 16 chars and 2 line display (in case of an I2C lcd driver was used)
const int buzzer = 10; //Define pin 10, can use other PWM pins  (5,6 or 9)
const int songspeed = 1.5; //Change to 2 for a slower version of the song, the bigger the number the slower the song
#define NOTE_C4  262   //Defining note frequency
#define NOTE_D4  294
#define NOTE_E4  330
#define NOTE_F4  349
#define NOTE_G4  392
#define NOTE_A4  440
#define NOTE_B4  494
#define NOTE_C5  523
#define NOTE_D5  587
#define NOTE_E5  659
#define NOTE_F5  698
#define NOTE_G5  784
#define NOTE_A5  880
#define NOTE_B5  988
RTC_DS3231 rtc;     // defining type of RTC
Adafruit_MotorShield AFMS = Adafruit_MotorShield();
Adafruit_StepperMotor*myMotor2 = AFMS.getStepper(28,2); //assigning address 2 to the stepper motor on the motor shield
Adafruit_StepperMotor*myMotor1 = AFMS.getStepper(28,1); //assigning address 1 to the stepper motor on the motor shield
```

```
char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday",
"Wednesday", "Thursday", "Friday", "Saturday"}; //definig a string that
contains the days of the week
int pilltime ; //defining an integer to store the value of the current hour
int counterone=0; //definig a counter that follows the  position of the first wheel
int check1=0;
int check2=0;
int pillone; // integer that stores the time implemented by the user for the first
wheel
int pillalarm; //integer that stores the time of the alarm to alert the user to take
pill (for example each 10 min)
int pilltwo; // integer that stores the time implemented by the user for the second
wheel
int m ;
int safetytwo=0 ; //safety integer to ensure that only one pill is dispensed at the
required time for the second wheel
int safetyone=0; //safety integer to ensure that only one pill is dispensed at the
required time for the first wheel
int c;
int countertwo=0; //definig a counter that follows the  position of the second
wheel
int notes[] = {       //Note of the song, 0 is a rest/pulse
  NOTE_E4, NOTE_G4, NOTE_A4, NOTE_A4, 0,NOTE_A4, NOTE_B4,
NOTE_C5, NOTE_C5, 0, NOTE_C5, NOTE_D5, NOTE_B4, NOTE_B4, 0,
  NOTE_A4, NOTE_G4, NOTE_A4, 0,NOTE_E4, NOTE_G4, NOTE_A4,
NOTE_A4, 0, NOTE_A4, NOTE_B4, NOTE_C5, NOTE_C5, 0,
  NOTE_C5, NOTE_D5, NOTE_B4, NOTE_B4, 0, NOTE_A4, NOTE_G4,
NOTE_A4, 0,NOTE_E4, NOTE_G4, NOTE_A4, NOTE_A4, 0,  NOTE_A4,
NOTE_C5, NOTE_D5, NOTE_D5, 0,
  NOTE_D5, NOTE_E5, NOTE_F5, NOTE_F5, 0, NOTE_E5, NOTE_D5,
NOTE_E5, NOTE_A4, 0,
  NOTE_A4, NOTE_B4, NOTE_C5, NOTE_C5, 0, NOTE_D5, NOTE_E5,
NOTE_A4, 0,  NOTE_A4, NOTE_C5, NOTE_B4, NOTE_B4, 0,
  NOTE_C5, NOTE_A4, NOTE_B4, 0, NOTE_A4, NOTE_A4,
  }; //arrangment of the notes of Pirates of the Caribbean music
int duration[] = {        //duration of each note (in ms) Quarter Note is set to 250
ms
125, 125, 250, 125, 125, 125, 125, 250, 125, 125,125, 125, 250, 125, 125,125,
125, 375, 125,125, 125, 250, 125, 125,125, 125, 250, 125, 125,125, 125, 250,
125, 125,125, 125, 375, 125,125, 125, 250, 125, 125,125, 125, 250, 125,
125,125, 125, 250, 125, 125,125, 125, 125, 250, 125,125, 125, 250, 125, 125,
250, 125, 250, 125,125, 125, 250, 125, 125,125, 125, 375, 375,250, 125};
/*const int colorR = 255; // assignement of first parameter for the lcd color
const int colorG = 0; // assignement of second parameter for the lcd color
const int colorB = 0; // assignement of third parameter for the lcd color*/
```

```
////////////void setup/////////////v o i d   s e t u p//////////////v o i d s e t u p///////////////
void setup () {
  pinMode(ledPinR, OUTPUT); // defining the mode of the Red led
  pinMode(ledPinG, OUTPUT); // defining the mode of the Green led
  counterone=4;    //should be equal to 13 if the first wheel was initially empty, but here it is assumed that the first wheel containes 13-4= 9 pills (it means it dispensed 4 pills)
  countertwo=5;    //same as previous line but for wheel two (here it is assumed that 5 pills where dispensed)
 m=0;
  pinMode(pirPin, INPUT); // defining the mode of the PIR sensor
AFMS.begin(); //initialize the motor shield
#ifndef ESP8266
while (!Serial); // for Leonardo/Micro/Zero
#endif
pilltwo = 20; // arbitrary chosen value for time of dispensing of first wheel
pillone = 15; // arbitrary chosen value for time of dispensing of second wheel
pinMode(4, INPUT_PULLUP); //defines the mode of work for the first button
pinMode(5, INPUT_PULLUP); //defines the mode of work for the second button
lcd.init(); // should be used in case of using I2C LCD normal module
 lcd.begin(16, 2);

 lcd.backlight(); //in case of normal I2C LCD module
  Serial.begin(9600);


  delay(3000); // wait for console opening
  pinMode(tx, OUTPUT); //defines the mode of work for tx pin
pinMode(rx, INPUT); //defines the mode of work for rx pin

  if (! rtc.begin()) {
    Serial.println("Couldn't find RTC");
    while (1);  } //check the existence of the Real Time Clock

  myMotor2->setSpeed(100); //set speed of motor two
  myMotor1->setSpeed(100); //set speed of motor one

  if (rtc.lostPower()) {
    Serial.println("RTC lost power, lets set the time!");
    // following line sets the RTC to the date & time this sketch was compiled
    //rtc.adjust(DateTime(F(_DATE), F(TIME_)));
    // This line sets the RTC with an explicit date & time, for example to set
    // January 21, 2014 at 3am you would call:
    rtc.adjust(DateTime(2023, 5, 14, 10, 55, 0));
  }
```

```
  //rtc.adjust(DateTime(2018, 12, 18, 20, 22, 0)); //Fast reseting of the RTC Date
and Time
}
//////////////v o i d l o o p//////////////v o i d l o o p////////////////
void loop () {
//  int touchValue = digitalRead(touchPin);
  int buttonone = digitalRead(4);
  int buttontwo = digitalRead(5);
   float h = dht.readHumidity();
   float t = dht.readTemperature();

  lcd.clear();
  lcd.setCursor(0,0); //set cursor of the LCD on first line
  Serial.print(F("Sample OK: "));//asure that the sample is Ok in case that the
previous error message was not printed too
  lcd.print("temp:");
  lcd.print(t); lcd.print(" *C, "); //print the temperature on the LCD screen
  lcd.setCursor(0,1); //set cursor of the LCD on second line
  lcd.print("humidity:");
  lcd.print(h); //lcd.println(" H"); //print the humidity on the LCD screen
   DateTime now = rtc.now();

    pilltime=now.hour(); //store the current hour as an integer
    pillalarm=now.minute();
    Serial.print(pilltime,DEC);
    //the following lines can be used in case the user wants to check the time on
the computer on the serial display
    Serial.print(now.year(), DEC);
    Serial.print('/ ');
    Serial.print(now.month(), DEC);
    Serial.print('/');
    Serial.print(now.day(), DEC);
    Serial.print(" (");
    Serial.print(daysOfTheWeek[now.dayOfTheWeek()]);
    Serial.print(") ");
    Serial.print(now.hour(), DEC);
    Serial.print(':');
    Serial.print(now.minute(), DEC);
    Serial.print(':');
    Serial.print(now.second(), DEC);
    Serial.println();
    delay(1000);
    ////////////////////wheel one dispense/////////////////////wheel one dispense
    if(pilltime==pillone && safetyone==0 && counterone != 13 )//fill later;//pill
dispensed at 11:00 am
   {
```

```
  lcd.clear();
  //following lines print amessage on the LCD and change its backlight while dispensing the pill depending on stage of dispensing
  lcd.setCursor(0,0); //set cursor of the LCD on first line
  lcd.print("wheel one");
  lcd.setCursor(0,1); //set cursor of the LCD on second line
  lcd.print("dispensing");
   digitalWrite(ledPinR, HIGH);//turn the red led on
  myMotor1->step(146.285714,FORWARD,SINGLE); //the motor rotates exactly one step that should dispense one and only one pill
  delay(3000);
   digitalWrite(ledPinR, LOW); //turn the red led off
    digitalWrite(ledPinG, HIGH); //turn the green led on
  for (int i=0;i<78;i++){             //203 is the total number of music notes in the song
  int wait = duration[i] * songspeed;
  tone(buzzer,notes[i],wait);          //tone(pin,frequency,duration)
  delay(wait);}                       //delay is used so it doesn't go to the next loop before tone is finished playing

  counterone++; //increment counterone because a pill was dispensed and the position of the wheel changed
  safetyone++;  //increment safetyone to guarantee that  only one pill is dispensed at the required time
  check1++;
   digitalWrite(ledPinG, LOW); //turn the green led off
  }
  if (pilltime == pillone+1)
  {safetyone==0; //reset the value of safetyone to zero in order to dispense the pill when the required time it achieved again
   check1=0;
   }
   ////////////////////wheel two dispense//////////////////////wheel two dispense
  //the folowing if condition contains the same procedure as for wheel one written previously
  if( pilltime==pilltwo && safetytwo==0 && countertwo !=13 )//pill dispensed at 11:00 am
  {
 lcd.clear();

 lcd.setCursor(0,0);
  lcd.print("wheel Two");
  lcd.setCursor(0,1);
  lcd.print("dispensing");
   digitalWrite(ledPinR, HIGH);
  myMotor2->step(146.285714,FORWARD,SINGLE);
```

```
   digitalWrite(ledPinR, LOW);
    digitalWrite(ledPinG, HIGH);
   delay(3000);
  for (int i=0;i<78;i++){             //203 is the total number of music notes in the
song
  int wait = duration[i] * songspeed;
  tone(buzzer,notes[i],wait);          //tone(pin,frequency,duration)
  delay(wait);                        //delay is used so it doesn't go to the next loop
before tone is finished playing
  }
  countertwo++;
  safetytwo++;
  check2++;

   digitalWrite(ledPinG, LOW);
   }
   if ( pilltime== pilltwo)
   {safetytwo==0;
   check2=0;
   }
   pirval = digitalRead(pirPin);  // read input value from the PIR sensor and
storre it in a variable
  if (pirval == HIGH && pilltime!=pillone && pilltime!=pilltwo) {  //in case of
the hand of the patient of the patient was detected the code enters this if loop
    //and prints on the LCD screen Wish You get better
    lcd.clear();
       lcd.setCursor(0,0);
      lcd.print("Wish you get better");
      lcd.setCursor(0,1);
      lcd.print("   SWETA   ");
      Serial.println("Motion detected!");
      delay(5000);
    }
    if (pirval == HIGH && pilltime==pillone) {  //in case of the hand of the
patient of the patient was detected the code enters this if loop
    //and prints on the LCD screen Wish You get better
    lcd.clear();
       lcd.setCursor(0,0);
      lcd.print("Wish you get better ");
      lcd.setCursor(0,1);
      lcd.print("   SWETA   ");
      Serial.println("Motion detected!");
      delay(5000);
      check1=0;

  }
```

```
  if (pirval == HIGH && pilltime==pilltwo) {  //in case of the hand of the
patient of the patient was detected the code enters this if loop
    //and prints on the LCD screen Wish You get better
    lcd.clear();
      lcd.setCursor(0,0);
     lcd.print("Wish you get better ");
     lcd.setCursor(0,1);
     lcd.print("   SWETA   ");
     Serial.println("Motion detected!");
     delay(5000);
     check2=0;
     }

  if (pirval != HIGH && pilltime==pillone && (pillalarm==10 || pillalarm ==
20 || pillalarm ==30 || pillalarm==40 || pillalarm==50 || pillalarm ==59 ))
  {
     lcd.clear();
     lcd.setCursor(0,0);
     lcd.print("REMEMBER");
     lcd.setCursor(0,1);
     lcd.print("YOUR PILL");
     for (int i=0;i<78;i++){             //203 is the total number of music notes in
the song
   int wait = duration[i] * songspeed;
   tone(buzzer,notes[i],wait);          //tone(pin,frequency,duration)
   delay(wait);}                       //delay is used so it doesn't go to the next loop
before tone is finished playing

     Serial.println("Motion detected!");
     delay(5000);

  }
  if ( pirval != HIGH && pilltime==pilltwo && ( pillalarm==10 || pillalarm ==
20 || pillalarm ==30 || pillalarm==40 || pillalarm==50 || pillalarm ==59 ) )
  {
     lcd.clear();
     lcd.setCursor(0,0);
     lcd.print("REMEMBER ");
     lcd.setCursor(0,1);
     lcd.print("YOUR PILL");
     for (int i=0;i<78;i++){             //203 is the total number of music notes in
the song
   int wait = duration[i] * songspeed;
   tone(buzzer,notes[i],wait);          //tone(pin,frequency,duration)
```

```
  delay(wait);}                    //delay is used so it doesn't go to the next loop
before tone is finished playing

     Serial.println("Motion detected!");
     delay(5000);

  }
  ////////////////////BLUETOOTH/////////////////BLUETOOTH//////////////////////////
  if (Serial.available() > 0) { //condition that something is recieved by the
bluetooth module

blue= Serial.read();//assign the value recieved by th Bluetooth module to a
variable
Serial.print(blue);
if (blue==49)//in case of wheel one test option was chosen on the application
{ myMotor1->step(146.285714,FORWARD,SINGLE);}
if (blue==50) //in case of wheel two test option was chosen on the application
{ myMotor2->step(146.285714,FORWARD,SINGLE);}// for the other values
read by the bluetooth they assign time in a way that the first number on the left
//corresponds to the wheel number and the two other digits correspond for the
time setted by the user, example if blue=112 that means that user wants wheel
one
//to dispense pill at 11:00 am
if (blue==101){pillone=1;
safetyone==0;
}
if (blue==102){pillone=2;
safetyone==0;}
if (blue==103){pillone=3;
safetyone==0;}
if (blue==104){pillone=4;safetyone=0;}
if (blue==105){pillone=5;safetyone=0;}
if (blue==106){pillone=6;safetyone=0;}
if (blue==107){pillone=7;safetyone=0;}
if (blue==108){pillone=8;safetyone=0;}
if (blue==109){pillone=9;safetyone=0;}
if (blue==110){pillone=10;safetyone=0;}
if (blue==111){pillone=11;safetyone=0;}
if (blue==112){pillone=12;safetyone=0;}
if (blue==113){pillone=13;safetyone=0;}
if (blue==114){pillone=14;safetyone=0;}
if (blue==115){pillone=15;safetyone=0;}
if (blue==116){pillone=16;safetyone=0;}
if (blue==117){pillone=17;safetyone=0;}
if (blue==118){pillone=18;safetyone=0;}
if (blue==119){pillone=19;safetyone=0;}
```

```
if (blue==120){pillone=20;safetyone=0;}
if (blue==121){pillone=21;safetyone=0;}
if (blue==122){pillone=22;safetyone=0;}
if (blue==123){pillone=23;safetyone=0;}
if (blue==124){pillone=24;safetyone=0;}
if (blue==201){pilltwo=1;safetytwo=0;}
if (blue==202){pilltwo=2;safetytwo=0;}
if (blue==203){pilltwo=3;safetytwo=0;}
if (blue==204){pilltwo=4;safetytwo=0;}
if (blue==205){pilltwo=5;safetytwo=0;}
if (blue==206){pilltwo=6;safetytwo=0;}
if (blue==207){pilltwo=7;safetytwo=0;}
if (blue==208){pilltwo=8;safetytwo=0;}
if (blue==209){pilltwo=9;safetytwo=0;}
if (blue==210){pilltwo=10;safetytwo=0;}
if (blue==211){pilltwo=11;safetytwo=0;}
if (blue==212){pilltwo=12;safetytwo=0;}
if (blue==213){pilltwo=13;safetytwo=0;}
if (blue==214){pilltwo=14;safetytwo=0;}
if (blue==215){pilltwo=15;safetytwo=0;}
if (blue==216){pilltwo=16;safetytwo=0;}
if (blue==217){pilltwo=17;safetytwo=0;}
if (blue==218){pilltwo=18;safetytwo=0;}
if (blue==219){pilltwo=19;safetytwo=0;}
if (blue==220){pilltwo=20;safetytwo=0;}
if (blue==221){pilltwo=21;safetytwo=0;}
if (blue==222){pilltwo=22;safetytwo=0;}
if (blue==223){pilltwo=23;safetytwo=0;}
if (blue==224){pilltwo=24;safetytwo=0;}
}

   //////////////////Refillement warning//////////////////Refillement warning
  if(counterone==13 && countertwo==13)// if the two wheels were empty the
LCD will display a refilling message for both wheels
  {lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Refill wheel 1");
  lcd.setCursor(0,1);
  lcd.print("and wheel 2");

  delay(5000);

  }
```

```
  if (counterone==13 && countertwo!=13 ) // if the first wheel was empty the
LCD will display a refilling message for first wheel
  {lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Refill wheel 1");
  delay(5000);

  }
  if(countertwo==13 && counterone!=13) // if the second wheel was empty the
LCD will display a refilling message for second wheel
  {lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Refill wheel 2");
  delay(5000);
  }
  ////////////////////////////Refilling Mechanism//////////////////Refilling
 if ((buttonone == LOW)) //if button one was pressed
  {if((counterone!=1) && (counterone!=0))//if there are still more than one pill
to get the whole wheel filled
   {myMotor1->step(146.285714,BACKWARD,SINGLE);
   counterone--; //counter decremented
   }
   if(counterone==1)
   {myMotor1->step(146.285714,BACKWARD,SINGLE); //if there is still one
more pill to get the whole wheel filled
   lcd.print("This is the last pill"); //messege printed on the screen
   counterone--; //counter decremented
   }
   if(counterone==0)
   {myMotor1->step(146.285714,BACKWARD,SINGLE);} //if the wheel is
completly filled but the user is still pressing the button, wheel will rotate but
counter will remain null
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("do not take the ");
  lcd.setCursor(0,1);
  lcd.print("pill that drop");
  }
 if ((buttontwo == LOW)) //same precedure as for button one
  {if((countertwo!=1) && (countertwo!=0))
   {myMotor2->step(146.285714,BACKWARD,SINGLE);
   countertwo--;
   }
   if(countertwo==1)
   {myMotor2->step(146.285714,BACKWARD,SINGLE);
   lcd.print("This is the last pill");
```

```
    countertwo--;
    }
    if(countertwo==0)
    {myMotor2->step(146.285714,BACKWARD,SINGLE);}
   lcd.clear();
   lcd.setCursor(0,0);
   lcd.print("do not take the ");
   lcd.setCursor(0,1);
   lcd.print("pill that drop");
   delay(300);
   }

   delay(300);
}
```

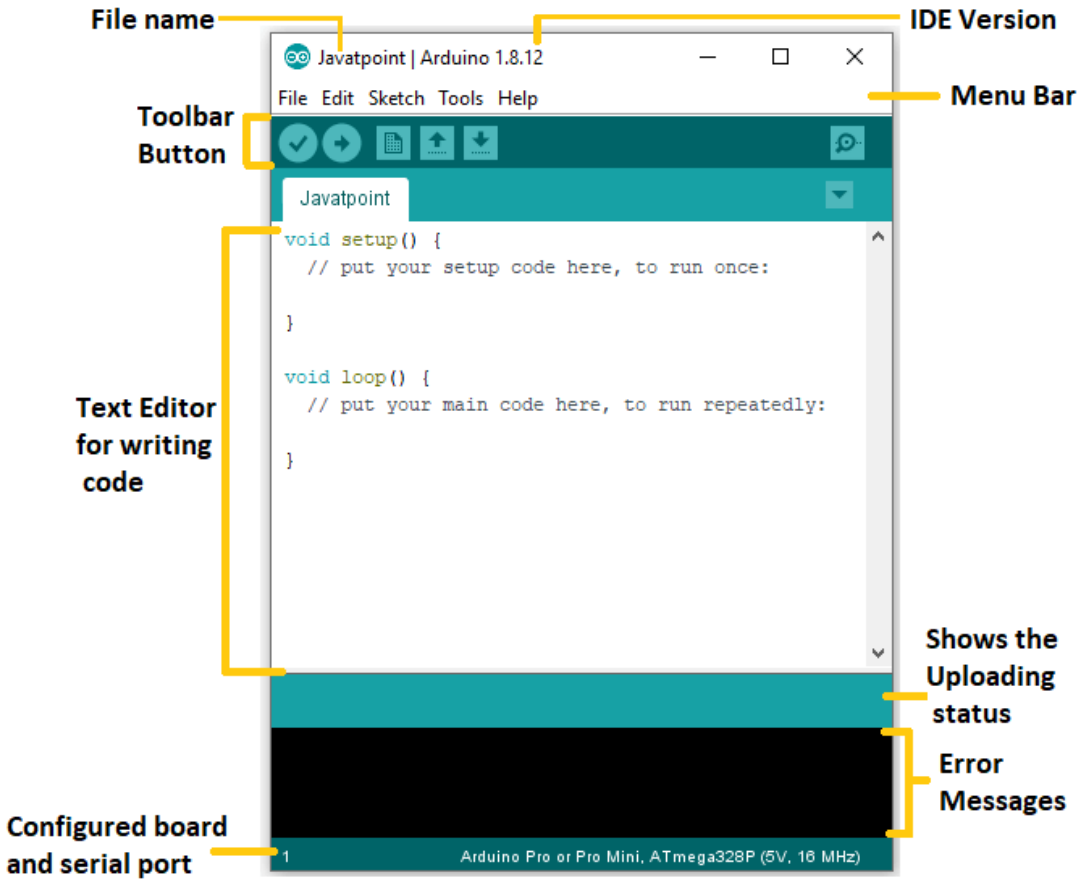
Software used in the project (Automatic Pill Dispenser)

1.Arduino IDE
2.Autodesk CAD
3.Fritzing

Arduino IDE

The Arduino IDE is an open-source software, which is used to write and upload code to the Arduino boards. The IDE application is suitable for different operating systems such as **Windows, Mac OS X, and Linux**. It supports the programming languages C and C++. Here, IDE stands for **Integrated Development Environment**.

The program or code written in the Arduino IDE is often called as sketching. We need to connect the Genuino and Arduino board with the IDE to upload the sketch written in the Arduino IDE software. The sketch is saved with the extension '.ino.'



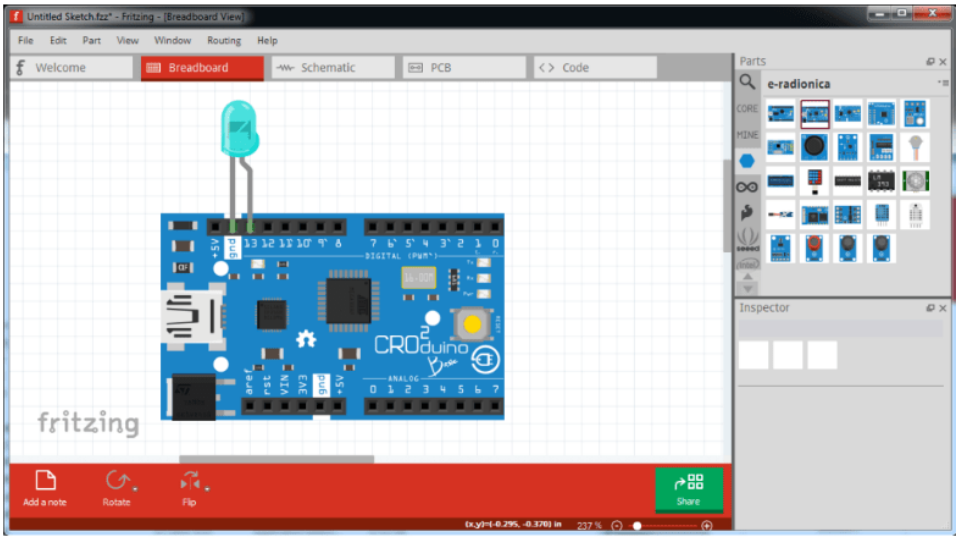
Autodesk CAD

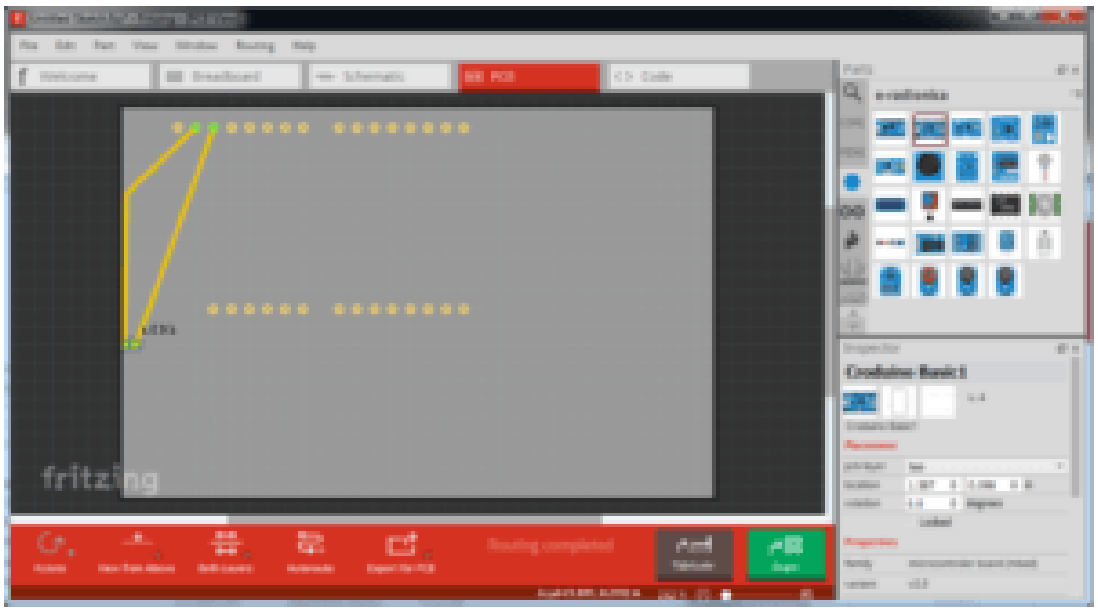
Autodesk software is used to design, make, and use things. Today it is more important than ever to stay competitive through product design innovation while at the same time tackle the big challenges of our time by producing and using things in a more sustainable manner. Businesses across multiple industries are changing the way they design and produce things and Autodesk is equipping them with the tools needed to turn the change into opportunities.

"Autodesk is changing the future of making things through computer aided design (CAD) and 3D modeling"

Fritzing

Fritzing is an open-source initiative to develop amateur or hobby CAD software for the design of electronics hardware, intended to allow designers and artists to build more permanent circuits from prototypes. It was developed at the University of Applied Sciences Potsdam. Fritzing is free software under the GPL 3.0 or later license, with the source code available on GitHub and the binaries at a monetary cost, which is allowed by the GPL.





Smartphone Application Connection

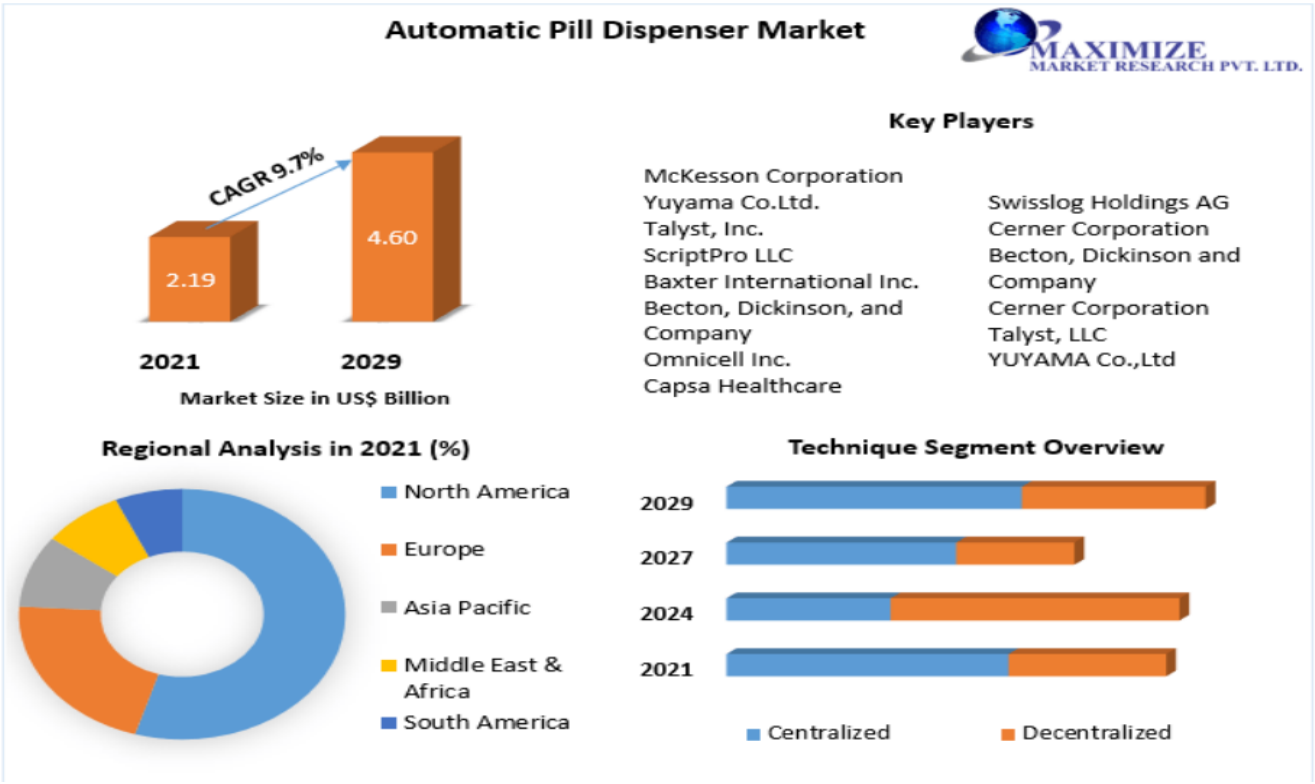
As already said, the communication with the robot is ensured by an smartphone application connected via a bluetooth module to the robot. The following images represent the functioning of the app. The first one represents the application's icon while the second and the third, deal with the manual dispensing mechanism and the setting time menu respectively. In the latter case, the dispensing mechanism is performed automatically at the time selected by the user.

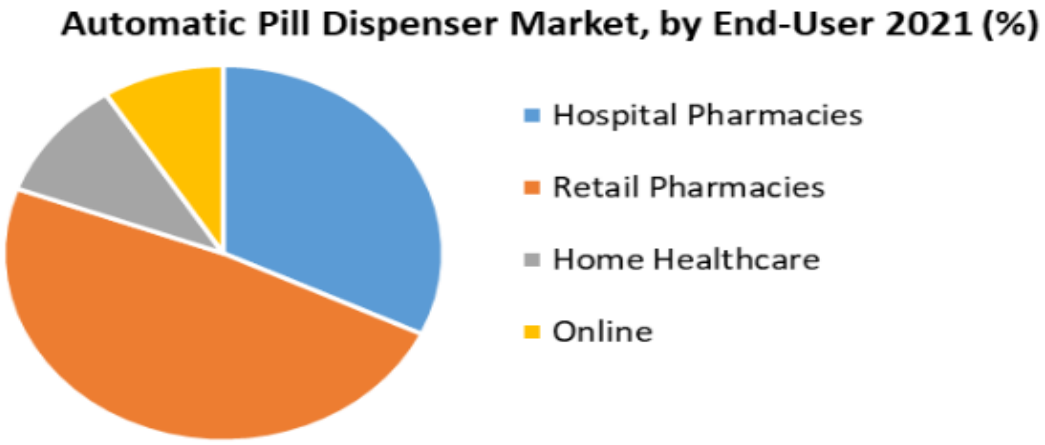
Future scope/Market of Automatic Pill Dispenser System

Automatic Pill Dispenser Market size was valued at US$ 2.19 Bn. in 2021 and the total revenue is expected to grow at CAGR 9.7 % through 2022 to 2029, reaching nearly US$ 4.60 Bn.

In inpatient care units, automated dispensing equipment allows for secure medication storage as well as computerized tracking of narcotic and other restricted drug usage. Reports can be created to aid in the detection and prevention of possible deviation from medication course. By removing the need for manual end-of-shift narcotic counts in patient care units, automated dispensing machines save nurses time. The capacity to detect and proactively monitor drug consumption patterns is another therapeutic function of automated dispensing equipment. This is accomplished by developing clinical indicators during medication removal. One example of this method was the use of aprotinin, a medicine provided by injection to minimize bleeding and the need for blood transfusions during difficult surgery. This medicine is costly, and its usage is restricted; as a result, it was regarded appropriate for assessing the use of clinical indicators.

Finally, automated dispensing equipment saves pharmacists' dispensing time since inventory management is driven by pre-set minimum and maximum levels and is managed only by pharmacy technicians. As a result, pharmacists have more time to devote to direct patient care and patient safety measures. Automated dispensing devices have unquestionably satisfied the standards for an effective drug delivery system in this age of technological advancement. Implementing automated dispensing devices is a step toward improving patient safety when cabinet design and use are well planned and the cabinets are used to their maximum capacity.





ıional Insights:

natic Pill Dispenser Market with the largest market share of 79%

Automatic Pill Dispenser Market			
Report Coverage	Details		
Base Year:	2021	**Forecast Period:**	2022-2029
Historical Data:	2017 to 2021	**Market Size in 2021:**	US $ 2.19 Bn.
Forecast Period 2022 to 2029 CAGR:	9.7%	**Market Size in 2029:**	US $ 4.60 Bn.
Segments Covered:	by Technique	• Centralized o Robotic o Carousels • Decentralized o Pharmacy Based o Ward based o Unit-Dose based	
	by End Users	• Hospital Pharmacies • Retail Pharmacies • Home Healthcare • Online	

Automatic Pill Dispenser Market, by Region

Automatic Pill Dispenser Market Key Players

- McKesson Corporation
- Yuyama Co.Ltd. • Talyst, Inc.
- ScriptPro LLC • Baxter International Inc.
- Becton, Dickinson, and Company
- Omnicell Inc.
- Capsa Healthcare
- Swisslog Holdings AG
- Cerner Corporation
- Becton, Dickinson and Company
- Cerner Corporation
- Talyst, LLC • YUYAMA Co.,Ltd

OBJECTIVES OF THE WORK

- The Objectives of the work can be broadly viewed as follows:
- To select appropriate materials used to design the SAPD.
- To select appropriate mechanism and program for actuating the device.
- To design and Fabricate Automated Pill Dispenser.
- To develop a working prototype
- To improve and simplify medical adherence

Result

- The design of the Automated Pill Dispenser has been successfully designed using Autodesk Inventor 2024.
- The render of the Android/IOS mobile application is completed.
- The bi - directional mechanism for tablet slot rotation is prototyped in Prototype.

CONCLUSION

- As of now the SAPD concept design is explored and some of its specifications are listed.
- The applications of SAPD is described in the problem definition.
- The Purpose of the idea and the survey of the idea is determined in the report.
- The scope of work and conceptual selection is also explained clearly.
- The basic driving mechanisms have been finalized and have been tested.
- The Smart Phone App is rendered and is under testing for 100% function without any bugs so that it can be published on the Play Store as well as the App Store for customer usage

REFERENCES

[1] Daniels DJ (ed.) (2004). Ground Penetrating Radar (2nd ed.). Knoval (Institution of Engineering and Technology). pp. 1–4. ISBN 978-0-86341-360-5

[2] B.Sklar, “Microcontrollers Fundamentals and Applications”, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall PTR, 2001.

[3] System Generator for DSP. Getting Started Guide. Arduino.

[4] S.O. Popescu, G. Budura, A.S. Gontean, “Review of PSK and QAM – Digital Modulation Techniques on FPGA”, International Joint Conference on Computational Cybernatics and Technical Informatics (ICCC-CONTI), Romania, 2010, pp.327-332.

[5] X.J. Eberlein, N.H.M. Wilson, and D. Bernstein. The holding problem with real-time information

[6] Automatic pill dispenser by anamolino

[7] Research gate