Smart Liquid Dispenser
Project Report

Presented by
Nishantkumar V Patel

Abstract

- As the technology has been developing since many years and it remains welfare upto better level in people's life. Many kind of liquid faucets and dispensers for instance, sensor based water taps developed and gradually with time few of small technological changes like design, efficiency, etc have been applied onto it.
- However, until now it can be seen that such dispensers and faucets stay beneficial upto some level. They can not resolve the water scarcity problems completely. On the other hand, my designed dispenser can play the peculiar role by providing a good amount of technological change to ramp-up the improvements in this sector.

- ***Key Words: Water Industry, Fluid Mechanics, Arduino UNO Microcontroller, Servo-motor, Infrared Sensor, Keypad.***

INDEX

1. Introduction

- Since few decades, world has been facing the crisis for water. This proposed dispenser can contribute by its vital performance. By implemantation of this device one can save the water on his/ her daily basis needs. It has special ability to dispense the water or any liquid as per your desirations and necessities. The water; people use to drink, to wash clothes, to wash utensils, to fill up the containers or tank on daily basis in the households can be usually found out its wastages. Not only this device can be used in the realm of domestics but can be proved helping factor at industrial level also.
- Here, the report presents device's working mechanism, its component details, bit amount of fluid mechanics and C/ C++ codes which were used for the microcontroller to make device smart enough from other dispenser and faucets which are already available in market.

2. Problem Description

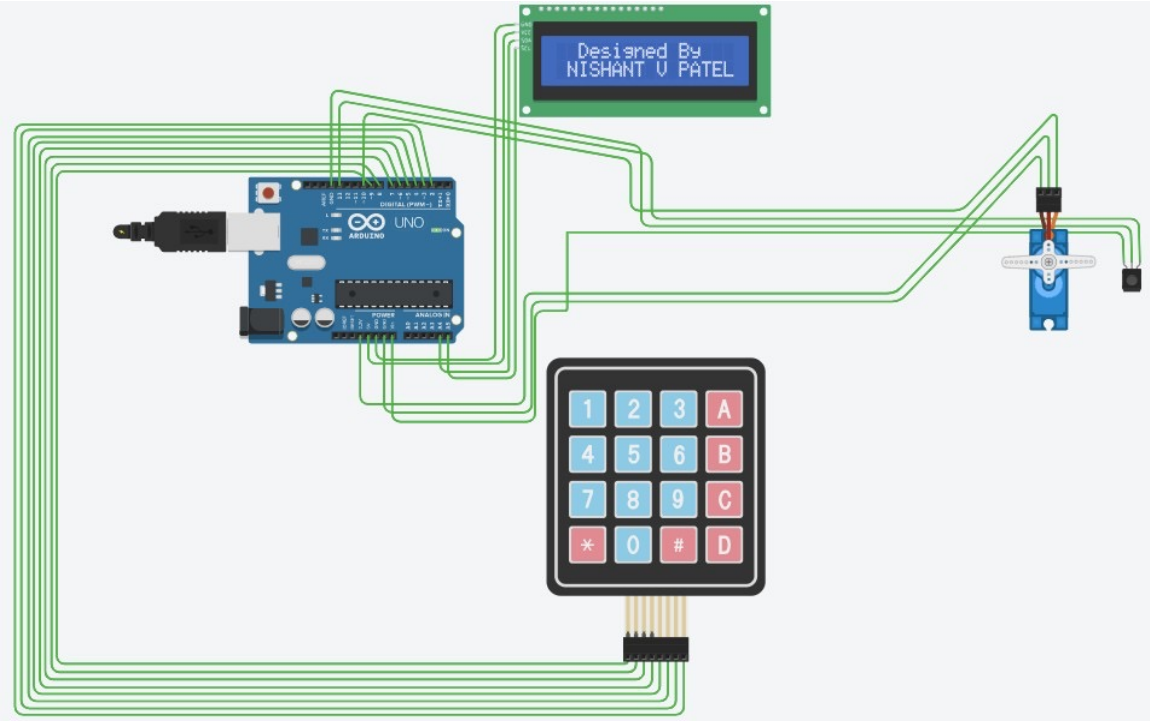
- Nowadays, conventional or traditional water taps are most commonly used at households, industry nearly everywhere in the world. Unfortunately, the sensored dispenser has its own limitations. When a person lay off his/ her hands in front of water tap then sensor detects the object and allows tap to flow the water through it. The second type of water tap is sensor based which is also immensly used specifically in coporate offices, headquarters.
- Thus, these 2 types can be seen mostly anywhere. But, this two kind of taps have its own limitations.
- The conventional water tap can not be turned off automatically once the targeted/ desired amount of water gets filled-up into water container. (the container for which one had turned on the tap)
- In additon, sensor based tap can not discharge the water/ liquid upto certain discharge rate or the person who wants tap to discharge the water/ liquid at his/ her desiration or need. As result, more water get discharged and ultimately go to wastage.
- Therefore, in order to eliminate such disabilities my Smart Liquid Dispenser enables the users to get the liquid as per his comfort and accordance.

3. Advantages of Proposed Design

- The proposed design allows the user to offer the amount of water or any other liquid as per his/ her neccessity.
- It can get automatically turned off itself when the targeted amount of water gets filled up in container as Smart Liquid Dispenser also offers the Auto-stop feature.
- Moreover, it has configurations of the conventional and sensor based mode also as used in today's date.
- This device occupied not so much of space.

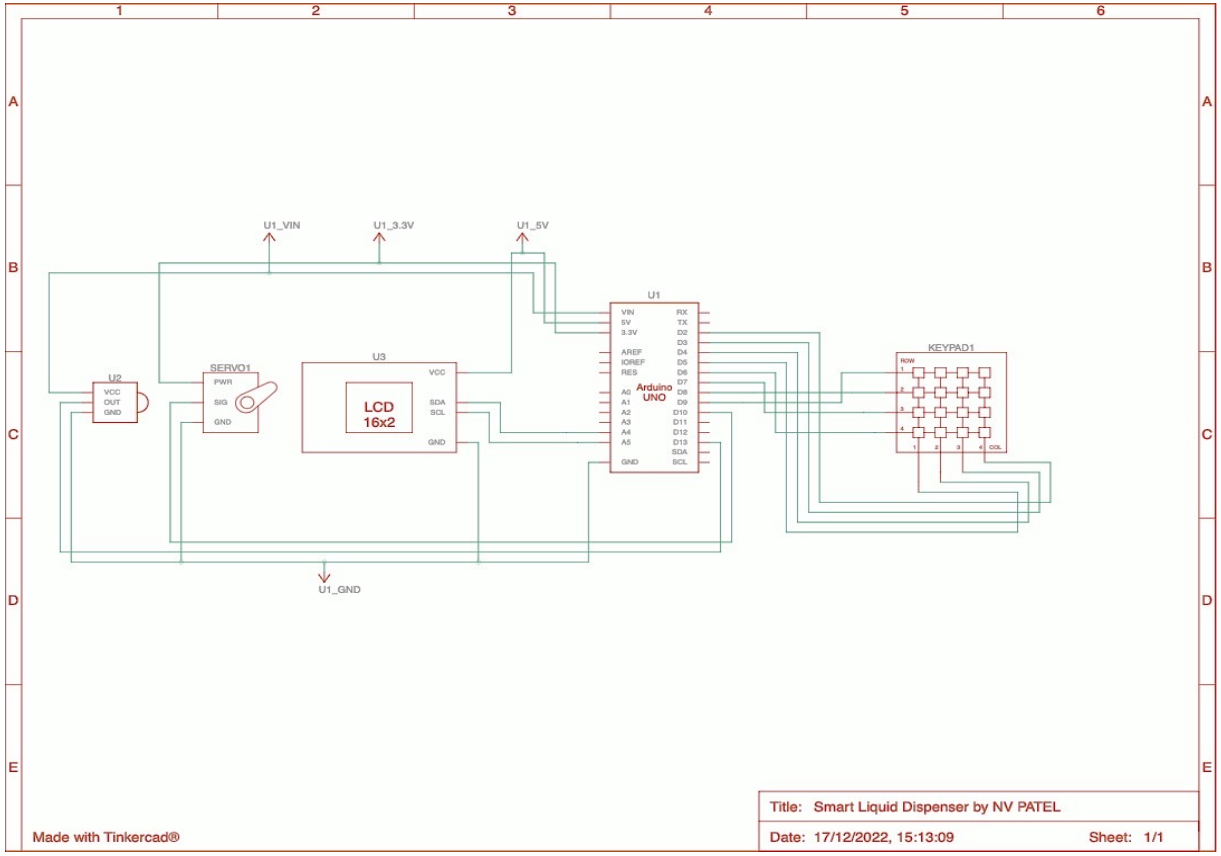
4. Prototype Diagrams

4.1 Component Diagram:



- This photo was taken from Tinker Card®.
- The Simulation with code have been done on Autodesk Tinker Card® online software.
- The photo diagram contains the Matrix Keypad, Aurduino UNO, Micro Servo-motor, Infrared Sensor, and LCD Display.

4.2 Circuit/ Wire Diagram:



- This photo was also taken from TinkerCrad®. This diagram depicts the wiring connections among these compoenents.

- <u>Denotations used:</u>

  PWR= power pin
  SIG/ OUT= signal pin
  GND= ground pin
  VCC= voltage pin
  A0-A5= analog pins
  D2-D13= digital pins
  ROW 1-4 & COL 1-4= keypad pins

5. Working Function

- The working function is very simple.
- Keypad and IR Sensor are input device to microcontroller whereas LCD display and Servo-motor are output device for this particular prototype system.
- The Microcontroller acts as a median between input & output devices. This microcontroller Arduino takes in the different particular set of commands from keypad (which is ultimately given by user) and from IR sensor (when the user hand is being detected by sensor).
- Then According to the code written for this device, Arduino acts upon.
- Arduino then sends the signals to Servo-motor and LCD display.
- According to the signals received, these output devices will act and outputs the result which user had exactly expected from the device.

- The servo motor connected to ball valve and as the servo rotate the ball valve rotate itself accordingly then allows/ restricts the liquid to flow/ not to flow throguh itself during opening and closing phases.
- The servo motor is connected to ball valve in 2 efficient ways.
- First way is gear connection and seconds one is only-shaft/ direct connection.
- The next 2 pictorial representations will clear more of how actually this device works.

Manual Mode:

- In manual mode, user give the valve turn on and turn off command to the device.
- Here in this case User have to manually give the inputs of turn On as well as turn Off for the dispenser.

Sensor Mode:

- It is the default mode in device.
- There is no need from user to give specific input by keypad.
- This mode is active all time and forever similarly as today's sensor based facuets.
- Infra red sensor module is used for the prototype.
- So, when there is object detected (user's hand) dispenser automatically discharge the water and stop discharging when nothing detected.

Auto Stop Mode:

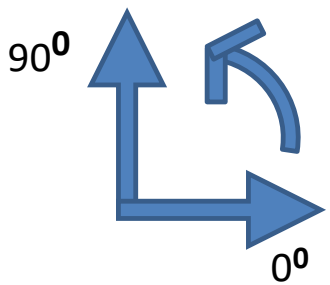
- Its more like the hybrid version or modified of Manual Mode.
- As in case of Manual mode, user has to give intputs for valve On and Off as well, here in this mode user is only required to give the amount of water He/ She wants from the dispenser to discharge.
- Once the user give the input value then even user will not there to stop the dispenser (closed the valve) dispenser will automatically stop the discharging operation when the targeted amount of liquid (which user wants) once get filled inside vessel or glass or any container of liquid.

Let us understand this mode with one example here,

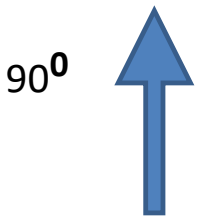
Suppose, you have installed a liquid dispenser assembly and your dispenser has a discharge or flow rate of 25 ml/sec (it can be any flow rate depending upon your pipe dimeter but here I assumed to explain you). Whatever value there is, according to that flow rate value, set the data/ parameters in Arduino code.

- Now, for instace, you want to drink a water and you are putting the glass beneath the dispenser and you want to fill a glass with 500 ml water. So, simple fluid mechanics calculation comes into picture and calculate the time takes by dipenser to fill up that glass with 500 ml quantity of water.
- So, according to calculations (500 ml) / (25 ml/sec)= 20 seconds.
- It takes 20 seconds to fill the targeted amount of water which user wants.
- So, here Servo motor rotate at opening rotation (0^0 to 90^0) and stays at 90^0 with 20 seconds of delay to allows the water to flow out then rotate back from 90^0 to 0^0 to close the valve.

(in C/ C++ code delay function is used for Servo motor rotation between 2 FOR Loops of servo motor rotations, 0^0 to 90^0 opening rotation For loop and 90^0 to 0^0 closing rotation For loop)

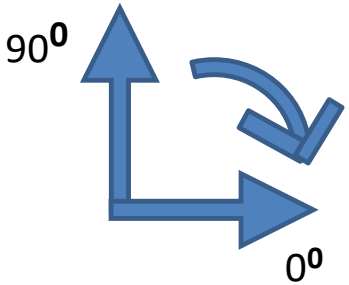


Opening rotation



Delay of 20 seconds

Water discharging for 20 seconds



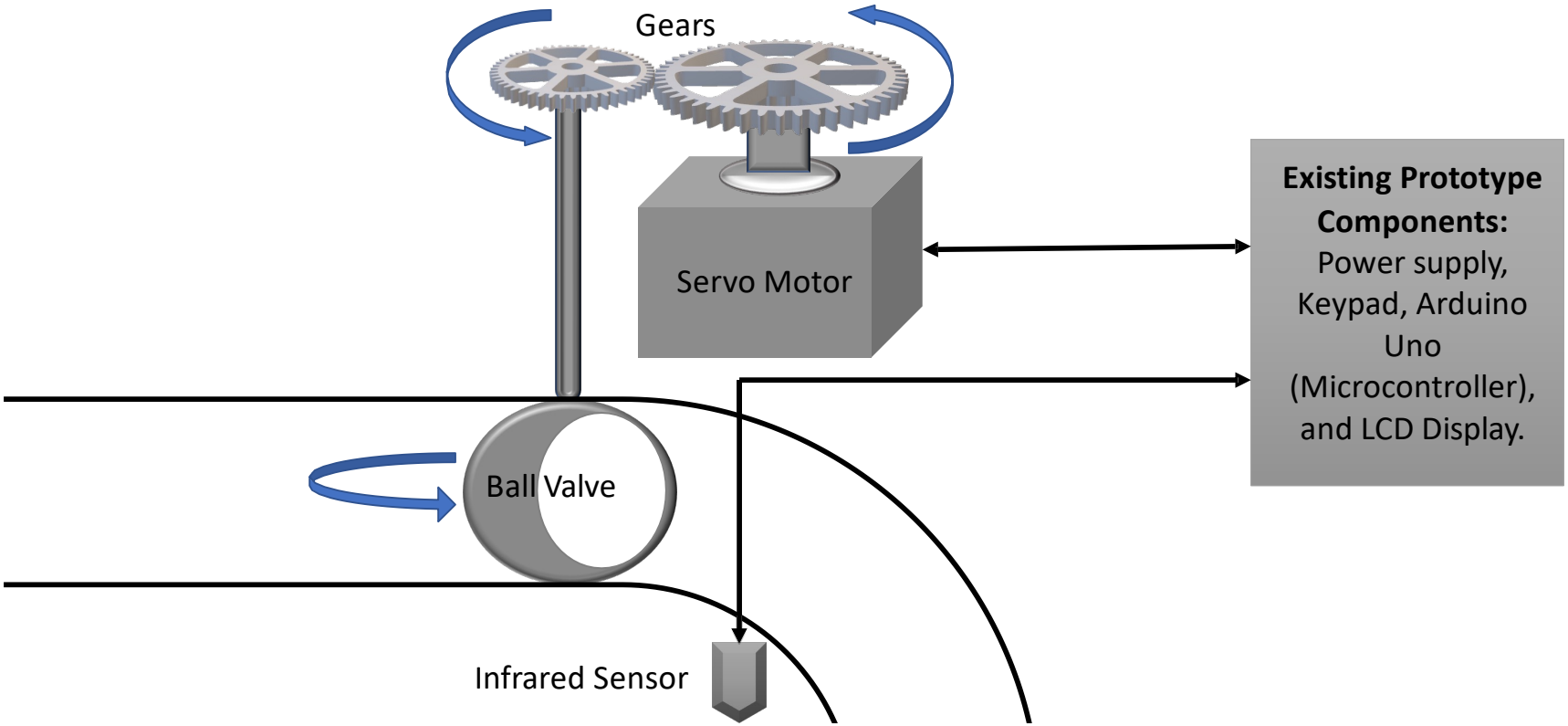
Closing Rotation

Partial Openings Mode:

- Partial opening mode has 5 different servo motor discrete rotation to complete one whole opening rotation of 0^0 to 90^0. It has 5 opening with 18^0 each.
- Once the user is pressed B first 0^0 to 18^0 rotation occurred then again by pressing B 18^0 to 36^0 rotation takes place and cosecutively up tp 5 times B key pressing/ 90^0.
- It is very helpful in case of show jet to set the required quantity of flow rate for bathing.
- It is also useful for Latrine Jets.

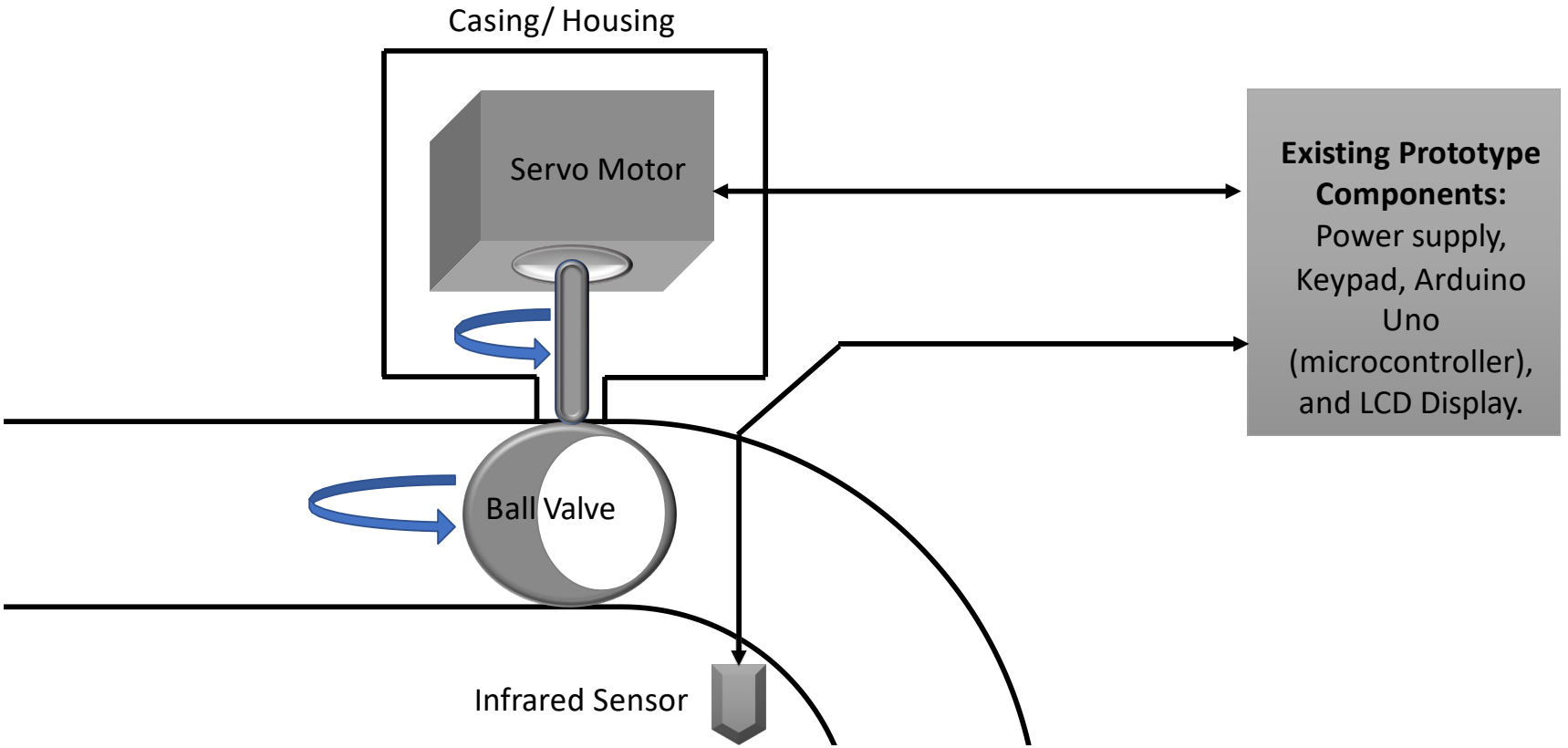
5.1 Connection of Proposed Assembly to Normal Dispenser

5.1.1) Gear Connection:

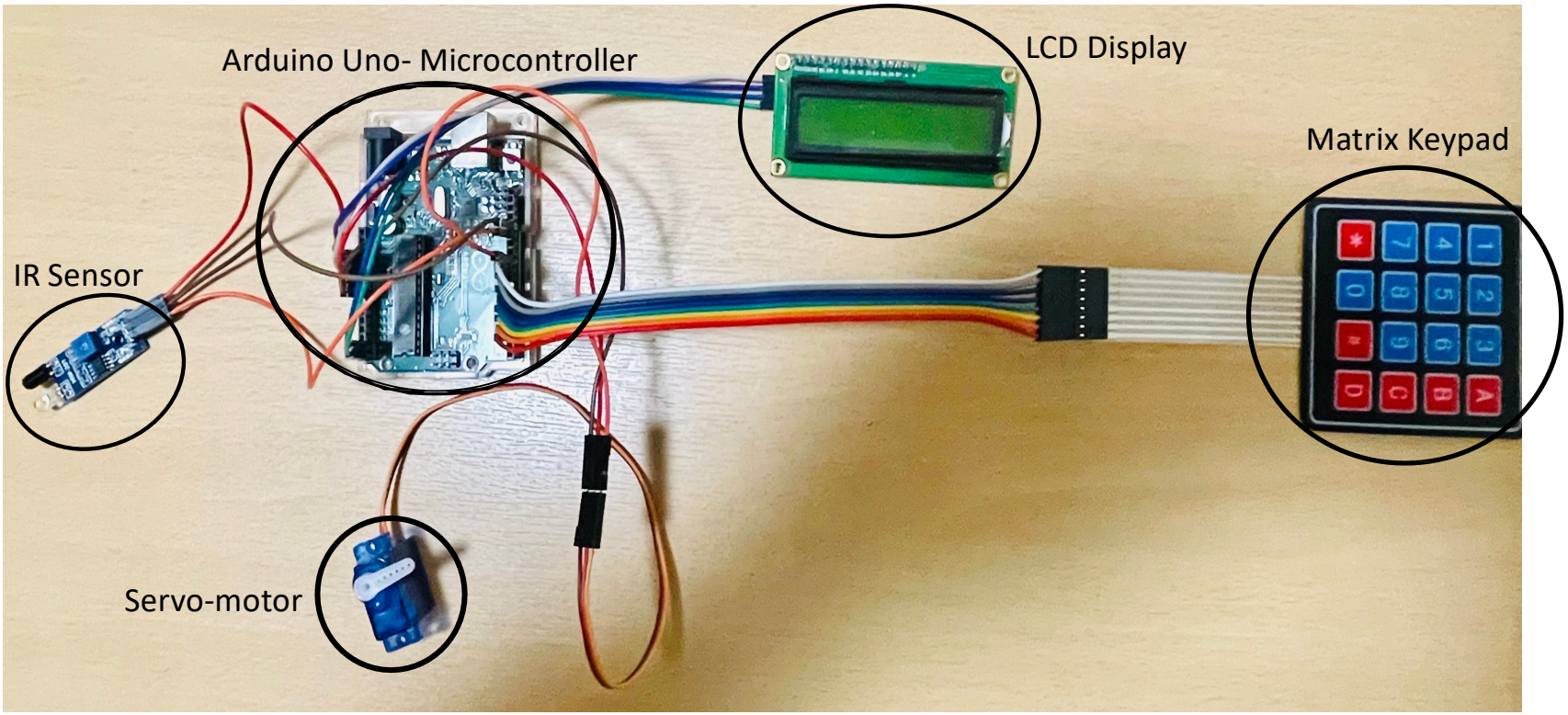


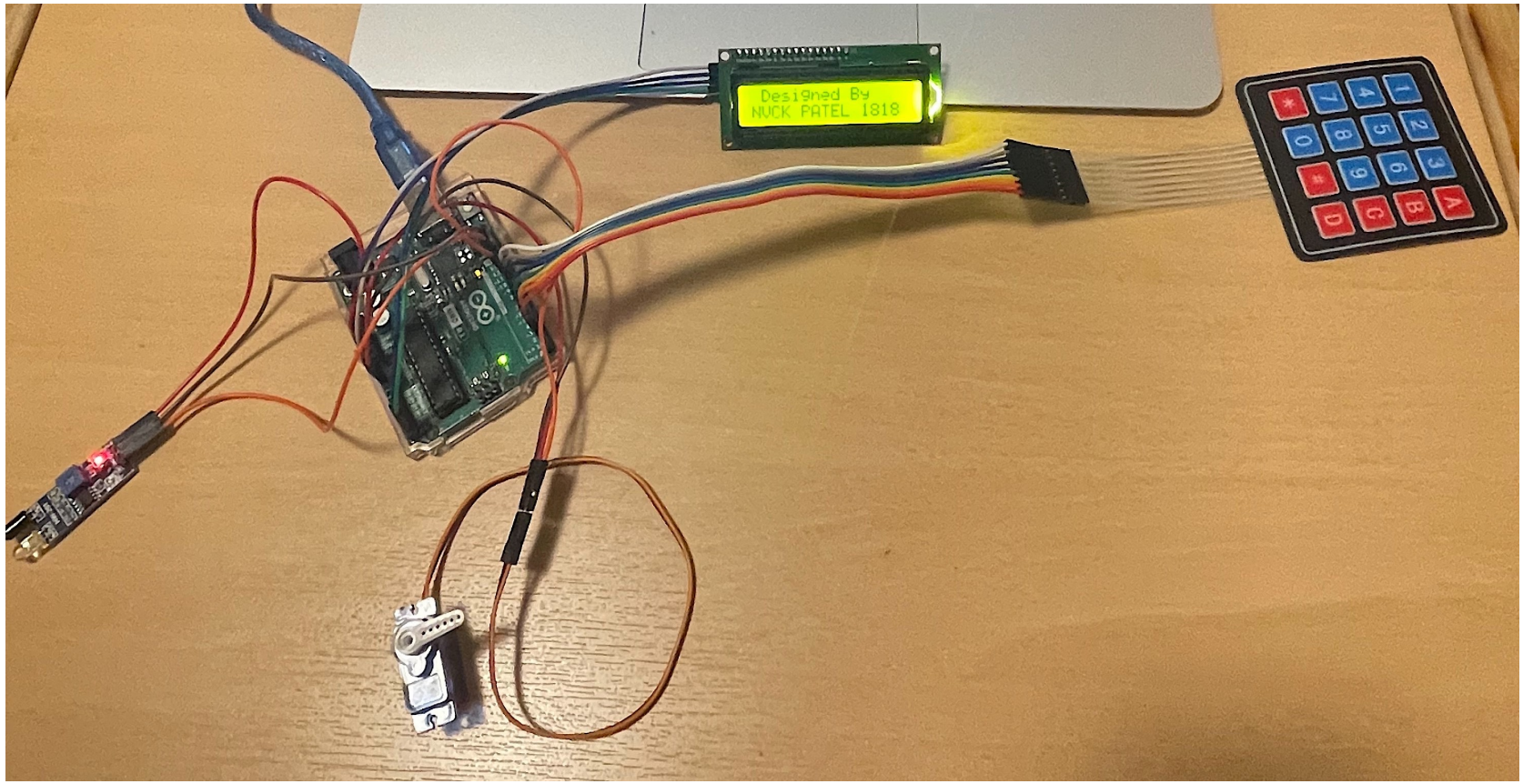
- In the gear connection, although the servo motor provides good enough torque for the ball valve to rotate. To stay at safe side increasing the torque output, the gear connection is best option to go with. In order to have higher torque, the driven gear should be of smaller diameter than driving gear.
- The servo motor degree rotation of 180^0 means the ball valve rotate to 90^0 as the gear ratio is 0.5. Therefore, the rotational frequency will be reduced by half. On the contrary, the torque will become doubled.

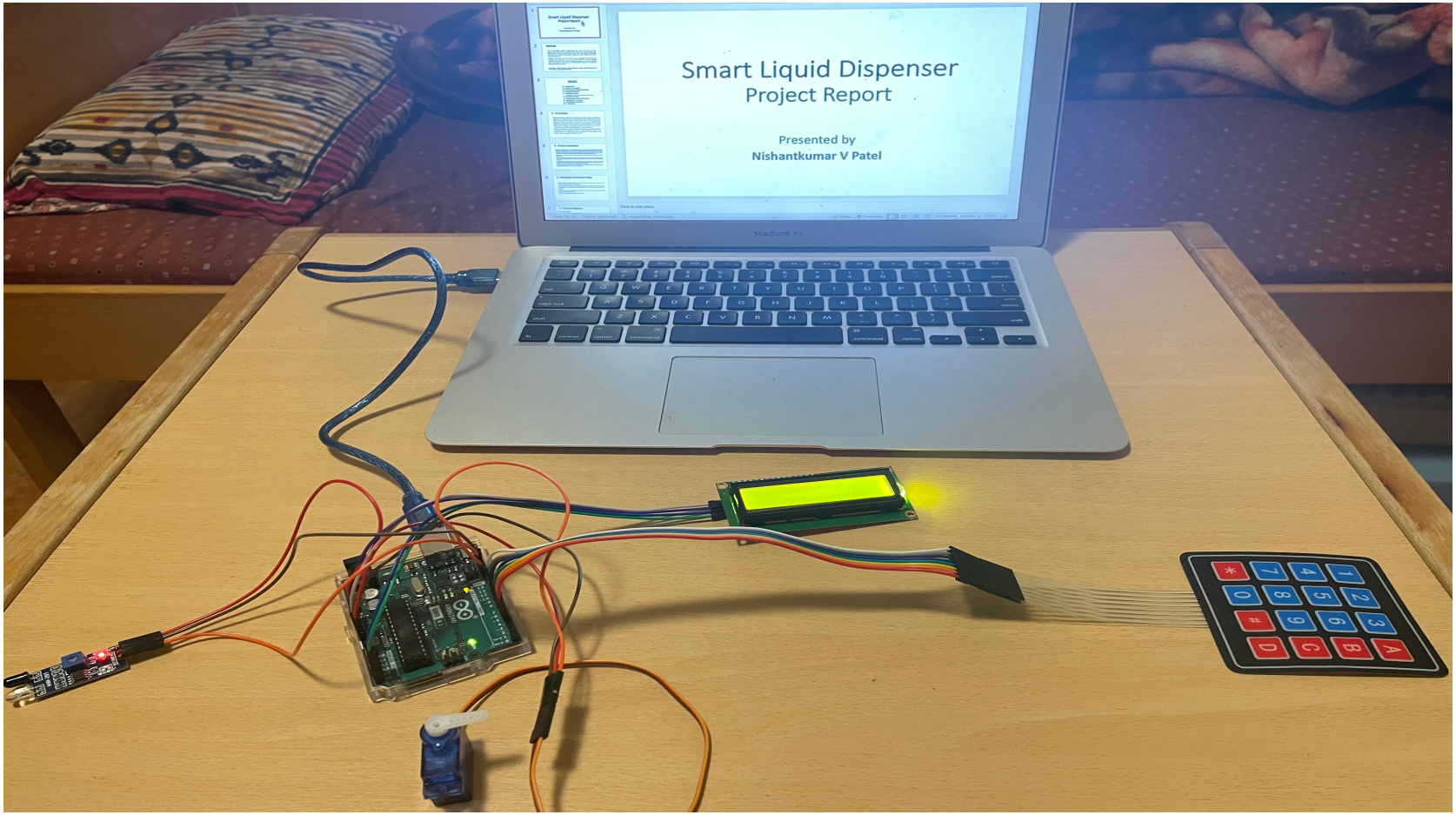
5.1.2) Only-shaft/ Direct Connection:



6. Prototype Images




Designed By
NUCK PATEL 1818


Smart Liquid Dispenser
Project Report
Presented by
Nishantkumar V Patel

7. Components for Final Design

- Note: The components which actually will be used for the final assembly are listed and described. Here, no componenets are mentioned which were used for the existing assembly.

- **Servo Motor:** Standard Servo motor will be used to provide the torque required for the ball valve to open and close.
- **Gears:** Two gears (driving & driven) of different diameters are used.
- **IR Sensor:** Infrared sensor module is used to detect the user hands for closing and opening the valve.
- **Jumper Wires:** These wires are used to connect all the components with each other for sending, receiving the signals and providing power to motor, lcd and sensor.

- **Microcontroller:** Arduino Uno R3 or Arduino Nano can be used as a microcontroller.
- **Keypad:** 4×3 matrix keypad is utilize send the inputs to microcontroller.
- **LCD Display:** 16× 2 lcd display with I2C module is used to displaying the information and updations.

8. Arduino (C/ C++) Code

- The Arduino code has been created where basically the C/ C++ languages are used.
- Different IoT Liabraries are included in the code for dealing with its functions.
- Plenty of If_else if_else control or conditional statements, for loops for servo motor degree rotations, keypad functions, and LCD display's functions are used.
- The comments inside this program has been written out, which persuade and justify the different kind of logics/ methods used and applied to this code.

```
/*
    The Code and Logic used behind this project is
independently designed by
                *******   NISHANTKUMAR V PATEL    ********
 */

#include <Servo.h>                  // Different Libraries
Included.
#include <Keypad.h>
#include <LiquidCrystal_I2C.h>
#include <Wire.h>

int current;       // Variable Declarations.
int previous;
String inputString;
long inputInt;
String binput;
const float flowrate=25;

void ServoMotor_Rotation_FUNCTION ();         // Function
Prototypes Declarations.
void Discrete_ServoMotor_Rotation_FUNCTION ();
void IR_Sensor_FUNCTION ();
void Device_Information_FUNCTION ();
void User_Manual_FUNCTION ();

const int ROW_NUM = 4;
const int COLUMN_NUM = 4;
char keys[ROW_NUM][COLUMN_NUM] =          // different key
declarations
{
  {'1','2','3', 'A'},
  {'4','5','6', 'B'},           // it is 4x3 matrix keypad
Array.
  {'7','8','9', 'C'},
  {'*','0','#', 'D'}
};
byte row_pins[ROW_NUM] = {9, 8, 7, 6};          // to setup
the pin numbers.
```

```
byte col_pins[COLUMN_NUM] = {5, 4, 3, 2};

Keypad keypad = Keypad(makeKeymap(keys), row_pins, col_pins,
ROW_NUM, COLUMN_NUM);      // create the Keypad Object.
LiquidCrystal_I2C lcd(0x27,16,2);       // create the LCD
Display Object with I2C chip address with 16 characters and 2
rows or lines.
Servo s1;


void setup()
{
  lcd.init();  // to initialize the LCD.
  lcd.backlight();
  pinMode(13,INPUT);    // input pin for reading IR Sensor
signals.
  s1.attach(10);       // pin for sending the signals to
servo-motor.
  inputString.reserve(9);
}



void loop()    // --Main Function--           // to run
Infinitely....
{

  {
    IR_Sensor_Mode_FUNCTION ();
  }

  char key = keypad.getKey();         // pressed key accepted
as input for keypad.
  if (key)        // if any key is pressed then condition
becomes true and enter inside the below conditional
statements.
  {
    if (key >= '0' && key <= '9')
    {
```

```
      inputString += key;                    // storing the
multiple key input in a single string variable.
      lcd.setCursor(1,0);
      lcd.print("Your Input is");
      lcd.setCursor(1,1);
      lcd.print(inputString);
      lcd.setCursor(12,1);
      lcd.print("(ml)");
      delay(100);
      inputInt = inputString.toInt();           // converting
from String to Integer data type.
    }

    else if (key == '#')           // to execute the multiple
key input.
    {
      ServoMotor_Rotation_FUNCTION ();
    }

    else if (key == '*')            // to clear the multiple
key input.
    {
      inputString= "";
      inputInt= 0;
      binput="";
    }

    else if (key == 'A')        // if A is pressed then servo-
motor starts to rotate to Open the Dispenser.
    {
      lcd.clear();
      lcd.setCursor(5,0);
      lcd.print("Opened");
      for (int i=0; i<=102; i+=1)           // to initialize
the loop with conditions.
      {
        s1.write(i);        // to rotate the servo motor.
        delay(10);
      }
```

```
    }

    else if (key == 'B')          // if B is pressed then
servo-motor starts to rotate discretely to Open the Dispnser
partially.
    {
      binput+=key;      // store the multiple keys in a single
variable- binput.
      Discrete_ServoMotor_Rotation_FUNCTION ();
    }

    else if (key == 'C')        // if C is pressed then servo-
motor starts to rotate back to Close the Dispenser.
    {
      lcd.clear();            // to clear the LCD display
      lcd.setCursor(5,0);
      lcd.print("Closed");        // to print the string or
character on LCD display.
      for (int i=102; i>=0; i-=1)
      {
        s1.write(i);
        delay(10);
      }
      delay(1800);
      lcd.clear();
    }

    else if (key=='D')
    {
      Device_Information_FUNCTION ();
      delay(1500);                        // delay(milliseconds)
      User_Manual_FUNCTION ();            // Every Function
Prototypes are called back in main infinitely runing void
loop().
    }
  }

}
```

```
  /* Function Prototyping of Different Functions for Modes.
  or Function Prototypes; which is call back in main code
structure or void loop() */

void ServoMotor_Rotation_FUNCTION ()

{
  flowrate;        // Flowrate of Actual Liquid Dispenser.
Flowrate is in (ml/second) unit.
  float time=inputInt/flowrate;     //Fluid mechanics
calculations to get the time in seconds.
  float seconds= time*1000;

  if (inputInt> 0 && inputInt!=0)
  {
    lcd.clear();
    lcd.setCursor(3,0);
    lcd.print("Started!");
    for (int i=0; i<=102; i+=1)     // for loop with initial
and ending degree angles for servo-motor.
    {
      s1.write(i);
      delay(10);
    }
    lcd.clear();
    lcd.setCursor(3,0);
    lcd.print("Running...");

    {
      delay(seconds-400);       //give the delay of amount of
seconds which is derived from the fluid mechanics
calculations.
    }

    for (int i=102; i>=0; i-=1)       //it tells to servo-
motor to rotate up to how much of angles and with speed
increments
```

```
    {
      s1.write(i);
      delay(10);
    }
    lcd.clear();
    lcd.setCursor(3,0);
    lcd.print("Completed!");
    delay(2000);
    lcd.clear();
  }
}


void Discrete_ServoMotor_Rotation_FUNCTION ()

{
  lcd.clear();
  lcd.setCursor(0,0);           //setting up the location of
the cursor in lcd display.
  lcd.print("Partial Openings");
  if (binput == "B")
  {
    lcd.setCursor(5,1);
    lcd.print("@ 1X");
    for (int i=0; i<=20; i+=1)        // if B key is pressed
one time, this loop will execute.
    {
      s1.write(i);
      delay(10);
    }
  }
  else if (binput == "BB")
  {
     lcd.setCursor(5,1);
     lcd.print("@ 2X");
     for (int i=20; i<=40; i+=1)      // if B key is pressed
consecutively two times, this loop will run.
     {
        s1.write(i);
```

```
        delay(10);
      }
  }
  else if (binput == "BBB")
  {
     lcd.setCursor(5,1);
     lcd.print("@ 3X");
     for (int i=40; i<=60; i+=1)
     {
        s1.write(i);
        delay(10);
     }
  }
  else if (binput == "BBBB")
  {
     lcd.setCursor(5,1);
     lcd.print("@ 4X");
     for (int i=60; i<=80; i+=1)
     {
        s1.write(i);
        delay(10);
     }
  }
  else if (binput == "BBBBB")       // if B key is pressed
consecutively five times, below loop will run.
  {
    lcd.setCursor(5,1);
    lcd.print("@ 5X");
    for (int i=80; i<=102; i+=1)
    {
      s1.write(i);
      delay(10);
    }
  }
}


void IR_Sensor_Mode_FUNCTION ()
 {
```

```
  current= digitalRead(13);             // 0 means, if object
is detected. 1 means, if object is not detected.
  if (current==0)            // if something is detected
(human hands for washing) then servo-motor rotate to Open the
Dispenser.
  {
    if (previous==1)
    {
      for (int i=0; i<=102; i+=1)
      {
        s1.write(i);
        delay(7);
      }
      delay(1000);
    }
  }

  else if(current==1)           // if something is not
detected then servo motor rotate back (reverse) to close the
dispenser.
  {
    if (previous==0)
    {
      for (int i=102; i>=0; i-=1)           // for
(Initialization; Condition; Updation)
      {
          s1.write(i);
          delay(7);
      }
    }
  }
  previous=current;         // assign the value of current
into previous variable for next iteration.
  delay(50);
}


void Device_Information_FUNCTION ()         // this function
```

```
is made up of entirely upon LCD codes.

{
  lcd.clear();
  lcd.setCursor(2,0);
  lcd.print("Hello World");
  delay(1500);
  lcd.clear();

  lcd.setCursor(4,0);
  lcd.print("I AM"); delay(1000);
  lcd.clear();
  lcd.setCursor(5,0); lcd.print("SMART");
  lcd.setCursor(0,1); lcd.print("LIQUID DISPENSER");
  delay(3500);
  lcd.clear();

  lcd.setCursor(1,0); lcd.print("Designed By");
  delay(500); lcd.setCursor(0,1);
  lcd.print("NVCK PATEL 1818");
  delay(3500);
  lcd.clear();

  lcd.setCursor(2,0); lcd.print("Save Water!");
  delay(2000);
  lcd.clear();
}

void User_Manual_FUNCTION ()        //  this function also is
made up of entirely upon LCD codes.

{
  lcd.clear(); lcd.setCursor(3,0);
  lcd.print("Welcome to"); lcd.setCursor(2,1);
  lcd.print("User Manual"); delay(3000);
  lcd.clear();

  lcd.setCursor(4,0); lcd.print("Mode-1");
  lcd.setCursor(5,1); lcd.print("Manual");
```

```
delay(3000);
lcd.clear();
lcd.setCursor(0,0); lcd.print("Press A to Open");
lcd.setCursor(0,1); lcd.print("Press C to Close");
delay(3500);
lcd.clear();

lcd.setCursor(4,0); lcd.print("Mode-2");
lcd.setCursor(0,1); lcd.print("Partial Openings");
delay(3500);
lcd.clear();
lcd.setCursor(0,0); lcd.print("Press *");
lcd.setCursor(0,1); lcd.print("to start Mode-2");
delay(3500);
lcd.clear(); lcd.setCursor(0,0);
lcd.print("Press B to set"); lcd.setCursor(0,1);
lcd.print("openings upto 5X"); delay(3500);
lcd.clear();

lcd.setCursor(4,0); lcd.print("Mode-3");
lcd.setCursor(3,1); lcd.print("Auto-Stop");
delay(3500);
lcd.clear();
lcd.setCursor(0,0); lcd.print("Press 0 to 9 keys");
lcd.setCursor(0,1); lcd.print("as Input in ml");
delay(3500);
lcd.clear();
lcd.setCursor(0,0); lcd.print("Press *");
lcd.setCursor(2,1); lcd.print("to clear Input");
delay(3500);
lcd.clear();
lcd.setCursor(0,0); lcd.print("Press #");
lcd.setCursor(2,1); lcd.print("to start Input");
delay(3500);
lcd.clear();

lcd.setCursor(4,0); lcd.print("Mode-4");
lcd.setCursor(0,1); lcd.print("Sensor/ Default");
delay(3500);
```

```
  lcd.clear();
}
```

9. Operation Instructions

- The different keypad buttons stands for different mode
- For ‘Manual Mode’, to turn on and turn off the dispenser Press A and Press C respcetively.
- Press * to start the ‘Partial Openings Mode’ then press B button to open the dispenser discretely up to 5 times. Means, when user press this specific button B one time valve opens at 18^0, then again by pressing the same button valve further open 18^0, means upto now dispenser opens at 36^0 in total. So, one can partially open in such manner upto 5 times. (0^0-18^0, 18^0-36^0, 36^0-54^0, 54^0-72^0, 72^0-90^0)
- Press D button for User Manual and Device Information.
- The ‘Sensor Mode’ is default therefore no keypad key is needed to be pressed for activating, as this mode is always ON and whenever the hands are detected it will start to implement.

- Press * to start the Auto-Stop Mode.
- Here one needs to press the numeric values. So, after pressing * press numeric keys 0 to 9 as the liquid quantity [only in Mili-Liter (ml)] user wants.
- The simply press # to start device or to start the dispensing process.
- In between if user wants to change the quantity of liquid, then press * to clear/ erase the previous entered input value.

10. Conclusion

- In conclusion, it is clearly seen that the proposed assembly can help and establish the better framework for discharges the required and specific amount of liquid whatever user wants.
- As a consequence, it can be proved more efficient as compared to other dispenser.
- Auto-stop and Partial-openings, these two features bring the advantages in effectiveness of device's performance.
- Therefore, Smart Liquid Dispenser has more things to offer than traditional taps and faucets. Its like all in one agenda.