ZERO CONTACT HAND SANITIZER DISPENSER

By

Harshal Shirke	TE-5-67
Tirth Thoria	TE-5-75
Vivek Vadhiya	TE-5-78

Guide

Prof. Archana Chaugule

Department of Information Technology

Shah & Anchor Kutchhi Engineering College, Mumbai

2020-21

Approval for IoT mini Project Report for Third Year semester V

This IOT mini-project report entitled “ZERO CONTACT HAND SANITIZER DISPENSER” by Harshal Shirke, Tirth Thoria, and Vivek Vadhiya is approved for semester V in partial fulfillment of the requirement for the IoT Lab of T.E. Engineering.

Examiners

1.____________________________

2.____________________________

Guide

1.____________________________

2.____________________________

Table of Contents

List of Figures

Zero Contact Hand Sanitizer Dispenser

Abstract:

Coronavirus (Covid-19) has created havoc and it is spreading rapidly. To bring it under control, it is important to maintain a healthy hygiene by constantly cleaning hands with hand sanitizers since hands have the highest probability of getting infected. But, the main problem is the manual operation of the hand sanitizer, especially in public places where there is a higher chance that an infected person might come in contact with it. To overcome this issue, we are introducing a system through which hands can be automatically sanitized thereby reducing the rate of people getting infected with the virus.

The main objective of Zero Contact Hand Sanitizer is to maintain social distancing and bring in automation in order to avoid spreading of the virus proactively. Therefore, the Model uses an ultrasonic sensor which can detect the hand with the intended range and signal it to the servo motor to press the cap of the sanitizer bottle. Also, it acquaints the user if the bottle gets empty. The process requires zero use of any physical part of the human body. The advancement in Internet of Things also enables to educate users about the intensity of Covid-19 in their area.

Chapter 1: Introduction

Definition

A dynamic network infrastructure with self-configuring capabilities based on the standard and interoperable communication protocols where physical and virtual "things" have identities, physical attributes, and virtual personalities and use intelligent interfaces, and are seamlessly integrated into the information network, often communicate data associated with users and their environments.

What is IoT?

The Internet of Things (IoT) describes the network of physical objects—"things"—that are embedded with sensors, software, and other technologies to connect and exchange data with other devices and systems over the internet. These devices range from ordinary household objects to sophisticated industrial tools.

Industrial IoT (IIoT) refers to the application of IoT technology in industrial settings, especially concerning instrumentation and control of sensors and devices that engage cloud technologies. Recently, industries have used machine-to-machine communication (M2M) to achieve wireless automation and control. But with the emergence of cloud and allied technologies (such as analytics and machine learning), industries can achieve a new automation layer and it creates new revenue and business models. IIoT is sometimes called the fourth wave of the industrial revolution, or Industry 4.0.

Characteristics of IoT

Some of the general and key characteristics identified during the research study are as follows:

1. Intelligence- IoT comes with the combination of algorithms and computation, software & hardware that makes it smart. Ambient intelligence in IoT enhances its capabilities which facilitate the things to respond in an intelligent way to a particular situation and supports them in carrying out specific tasks. Despite all the popularity of smart technologies, intelligence in IoT is only concerned as a means of interaction between devices, while user and device interaction is achieved by standard input methods and graphical user interface.

2. Connectivity- Connectivity empowers the Internet of Things by bringing together everyday objects. The connectivity of these objects is pivotal because simple object-level interactions contribute towards collective intelligence in the IoT network. It enables network accessibility and compatibility in things. With this connectivity, new market opportunities for the Internet of things can be created by the networking of smart things and applications.

3. Dynamic Nature- The primary activity of the Internet of Things is to collect data from its environment, this is achieved with the dynamic changes that take place around the devices. The state of these devices changes dynamically, for example sleeping and waking up, connected and/or disconnected as well as the context of devices including temperature, location, and speed. In addition to the state of the device, the number of devices also changes dynamically with a person, place, and time.

4. Enormous scale- The number of devices that need to be managed and that communicate with each other will be much larger than the devices connected to the current Internet. The management of data generated from these devices and their interpretation for application purposes becomes more critical. Gartner (2015) confirms the enormous scale of IoT in the estimated report where it stated that 5.5 million new things will get connected every day and 6.4 billion connected things will be in use worldwide in 2016, which is up by 30 percent from 2015. The report also forecasts that the number of connected devices will reach 20.8 billion by 2020.

5. Sensing- IoT wouldn't be possible without sensors that will detect or measure any changes in the environment to generate data that can report on their status or even interact with the environment. Sensing technologies provide the means to create capabilities that reflect a true awareness of the physical world and the people in it. The sensing information is simply the analog input from the physical world, but it can provide a rich understanding of our complex world.

6. Heterogeneity- Heterogeneity in the Internet of Things is one of the key characteristics. Devices in IoT are based on different hardware platforms and networks and can interact with other devices or service platforms through different networks. IoT architecture should support direct network connectivity between heterogeneous networks. The key design requirements for heterogeneous things and their environments in IoT are scalabilities, modularity, extensibility, and interoperability.

7. Security- IoT devices are naturally vulnerable to security threats. As we gain efficiencies, novel experiences, and other benefits from the IoT, it would be a mistake to forget about security concerns associated with it. There is a high level of transparency and privacy issues with IoT. It is important to secure the endpoints, the networks, and the data that is transferred across all of it means creating a security paradigm.

Physical and Logical design of IoT

Physical Design:

The things in IoT generally refer to IoT devices that have unique identifiers and can perform remote sensing, actuating, and monitoring capabilities. IoT devices can exchange data with other connected devices and applications, or collect data from other devices and

process the data either locally or send the data to centralized servers and other tasks within IoT infrastructure, based on temporal and space constraints.

An IoT device may consist of several interfaces for connection to several devices both wired and wireless. These include

1. I/O interfaces for sensors

2. Interfaces for Internet connectivity

3. Memory and storage interfaces

4. Audio/Video interfaces.

An IoT device can collect various types of data board or attached sensors, such as temperature, humidity, and light intensity. Sensed data can be communicated either to other devices or cloud-based storage/servers. IoT devices can be connected to actuators that allow them to interact with other physical entities in the vicinity of the device.

Logical Design:

The logical design of an IoT system refers to an abstract representation of the entities and processes without going into the low-level specifics of the implementation.

IOT Functional Blocks:

1. **DEVICES**: An IoT system comprises devices that provide sensing, actuation, monitoring, and control functions.
2. **COMMUNICATION**: The communication block handles the communication of the IoT system.
3. **SERVICES**: An IoT system uses various types of IoT services such as services for device monitoring, device control services, data publishing services, and services for device discovery.
4. **MANAGEMENT**: Management functional block provides various functions to govern the IoT system.
5. **SECURITY**: Security functional block secures the IoT system and by providing functions such as authentication, authorization, message and content integrity, and data security.
6. **APPLICATION**: IoT applications provide an interface that users can use to control and monitor various aspects of the IoT system. Applications also allow users to view the system status and view or analyze the processed data.

IoT Levels 1-7

Layer 1: The Things Layer

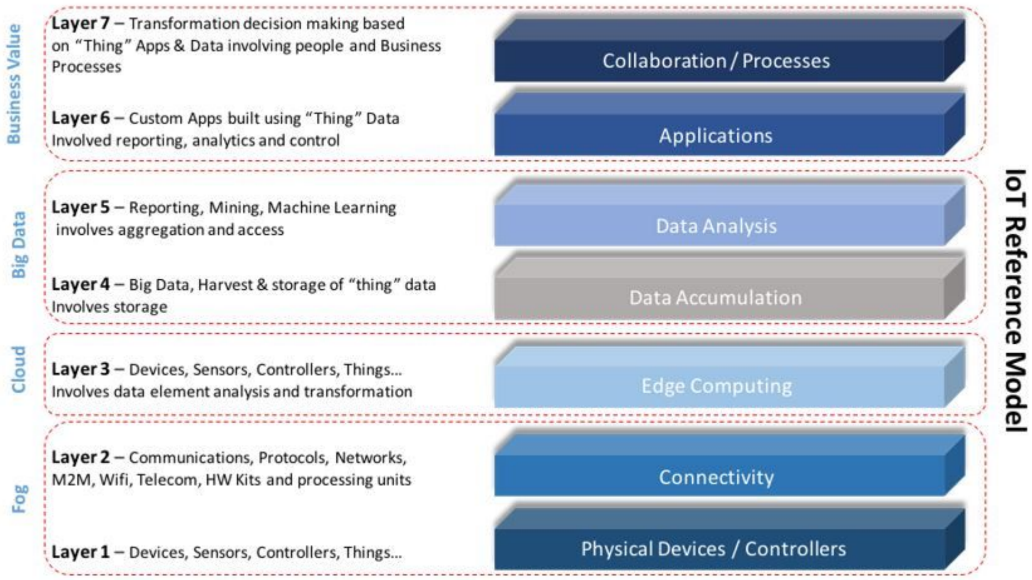

Fig 1.0 - IOT Reference Model

This layer of IoT comprises devices, sensors, and controllers. Connected devices are what enable the IoT environment. These devices include mobile devices such as smartphones or tablets, microcontroller units, and single-board computers. The connected devices are the real endpoint for IoT.

The IoT Reference Model starts with Level 1: physical devices and controllers that might control multiple devices. These are the "things" in the IoT, and they include a wide range of endpoint devices that send and receive information. Today, the list of devices is already extensive.

It will become almost unlimited as more equipment is added to the IoT over time. Devices are diverse, and there are no rules about size, location, form factor, or origin. Some devices will be the size of a silicon chip. Some will be as large as vehicles. The IoT must support the entire range. Dozens or hundreds of equipment manufacturers will produce IoT devices.

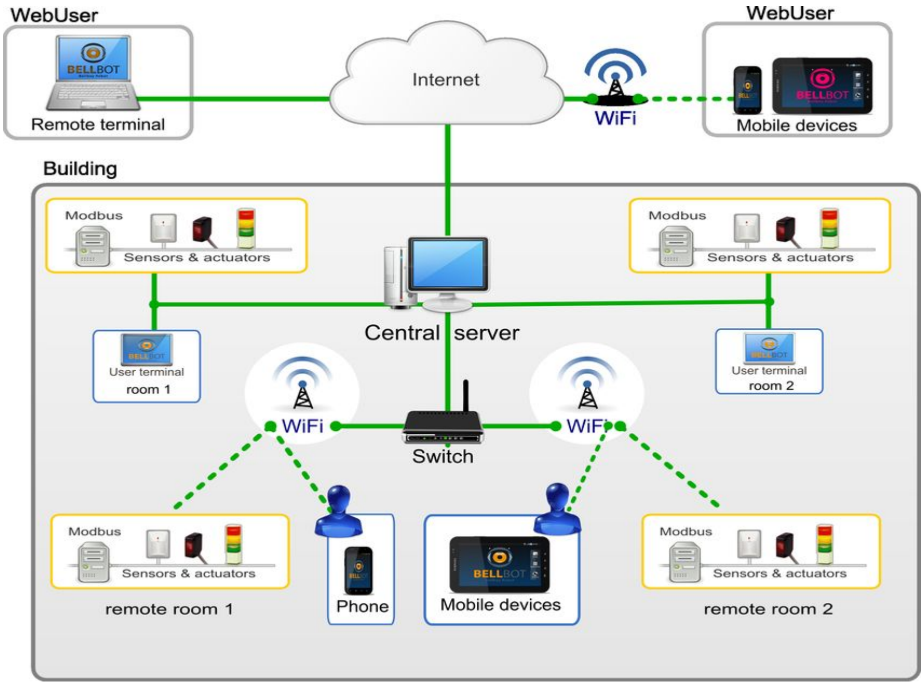

Fig 1.1 - Physical Devices and Controllers

IoT "devices" are capable of:

- Analog to digital conversion, as required
- Generating data
- Being queried/controlled over-the-net

<u>Layer 2: Connectivity/Edge Computing Layer</u>

Layer 2 is the connectivity/ edge computing layer, which defines the various communication protocols and networks used for connectivity and edge computing. It is a distributed architecture where IoT data is processed at the edge of the network.

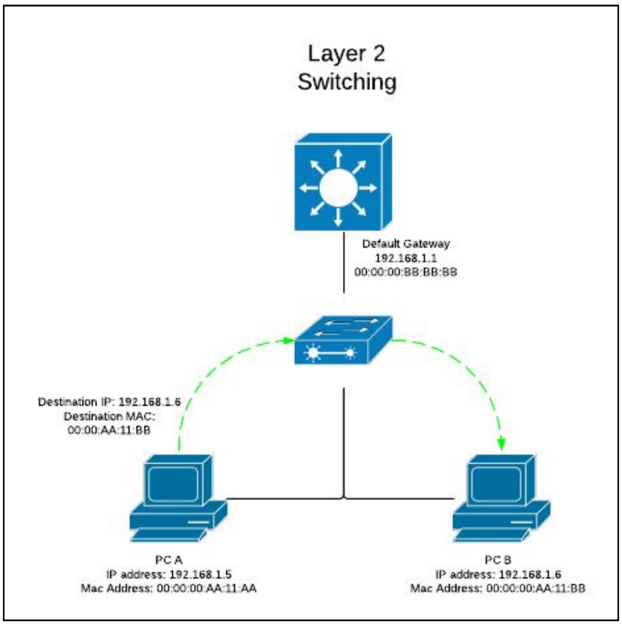

Fig 1.2 - Connectivity/Edge Computing layer

Communications and connectivity are concentrated on one level—Level 2. The most important function of Level 2 is reliable, timely information transmission.

This includes transmissions:

- Between devices (Level 1) and the network
- across networks (east-west)
- Between the network (Level 2) and low-level information processing occurring at Level 3

Traditional data communication networks have multiple functions, as evidenced by the International Organization for Standardization (ISO) 7-layer reference model. However, a complete IoT system contains many levels in addition to the communications network. One objective of the IoT Reference Model

Layer 3: Global Infrastructure Layer

Layer 3 is the global infrastructure layer, which is typically implemented in cloud infrastructure. Most of the IoT solutions integrate with cloud services. A comprehensive set of integrated services, the IoT cloud can provide businesses with useful insights and perspectives on customers.

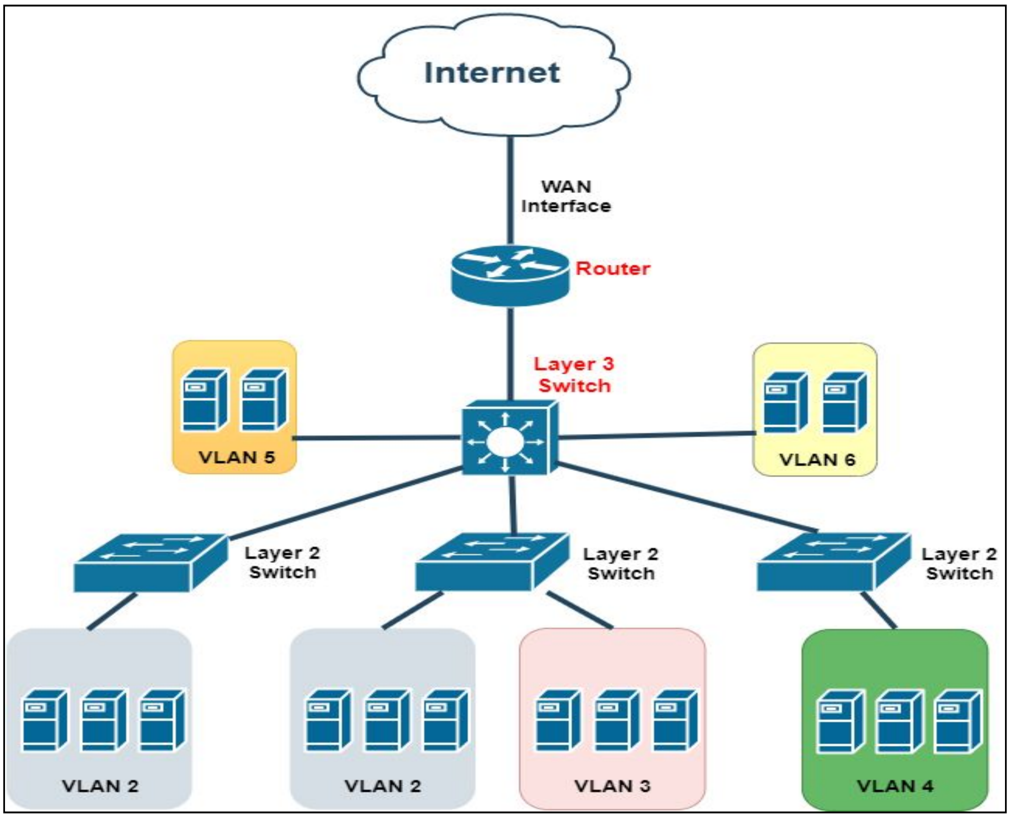

Fig 1.3 - Global Infrastructure connection layer

Given that data is usually submitted to the connectivity level (Level 2) networking equipment by devices in small units, Level 3 processing is performed on a packet-by-packet basis. This processing is limited because there is only awareness of data units—not “sessions” or “transactions.” Level 3 processing can encompass many examples, such as

- Evaluation: Evaluating data for criteria as to whether it should be processed at a higher level
- Formatting: Reformatting data for consistent higher-level processing
- Expanding/decoding: Handling cryptic data with additional context (such as the origin)
- Distillation/reduction: Reducing and/or summarizing data to minimize the impact of data and traffic on the network and higher-level processing systems
- Assessment: Determining whether data represents a threshold or alert; this could include redirecting data to additional destinations

Layer 4: Data Ingestion Layer

Layer 4 is the data ingestion layer, which includes big data, cleansing, streaming, and storage of data.

Most applications cannot, or do not need to, process data at network wire speed. Applications typically assume that data is "at rest"—or unchanging—in memory or on disk. At Level 4, Data Accumulation, data in motion is converted to data at rest. Level 4 determines:

- If data is of interest to higher levels: If so, Level 4 processing is the first level that is configured to serve the specific needs of a higher level.
- If data must be persisted: Should data be kept on disk in a non-volatile state or accumulated in memory for short-term use?
- The type of storage needed: Does persistency require a file system, big data system, or relational database?
- If data is organized properly: Is the data appropriately organized for the required storage system?
- If data must be recombined or recomputed: Data might be combined, recomputed, or aggregated with previously stored information, some of which may have come from non-IoT sources.

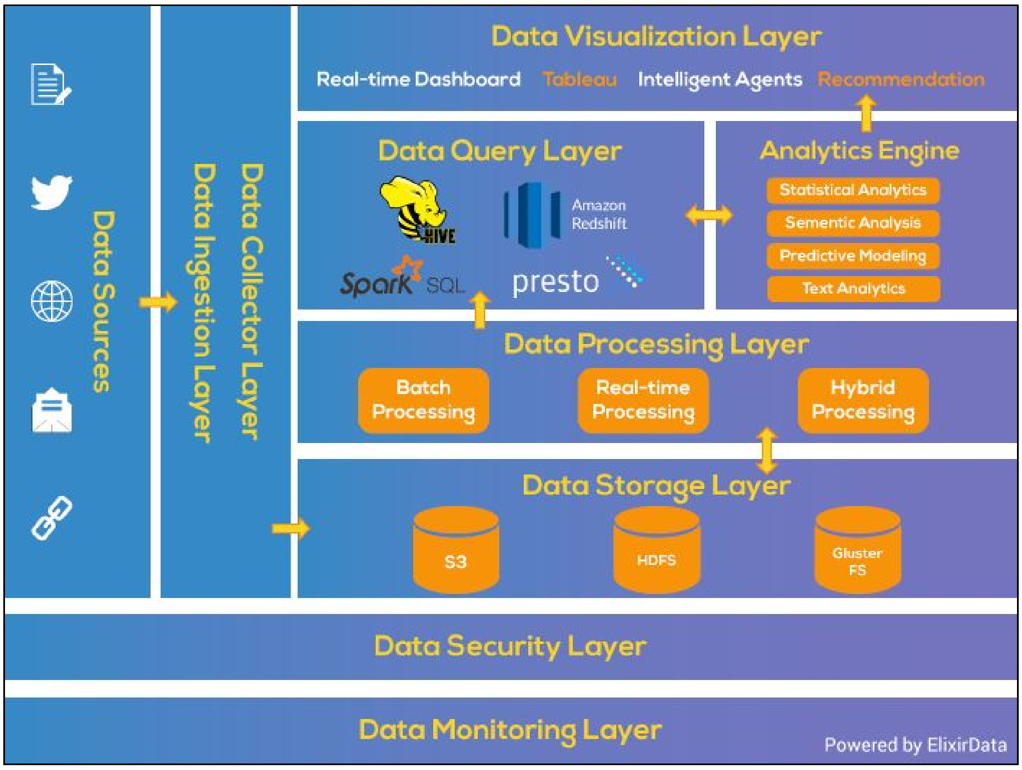

Fig 1.4 - Data Accumulation Layer

Layer 5: Data Analysis Layer

Layer 5 is the data analysis layer and relates to data reporting, mining, machine learning, etc.

The data abstraction functions of Level 5 are focused on rendering data and its storage in ways that enable developing simpler, performance-enhanced applications. With multiple devices generating data, there are many reasons why this data may not land in the same data storage:

- There might be too much data to put in one place.
- Moving data into a database might consume too much processing power so that retrieving it must be separated from the data generation process. This is done today with online transaction processing (OLTP) databases and data warehouses.
- Devices might be geographically separated, and processing is optimized locally.
- Levels 3 and 4 might separate "continuous streams of raw data" from "data that represents an event." Data storage for streaming data may be a big data system, such as Hadoop. Storage for event data may be a relational database management system (RDBMS) with faster query times.
- Different kinds of data processing might be required. For example, in-store processing will focus on different things than across-all-stores summary processing.

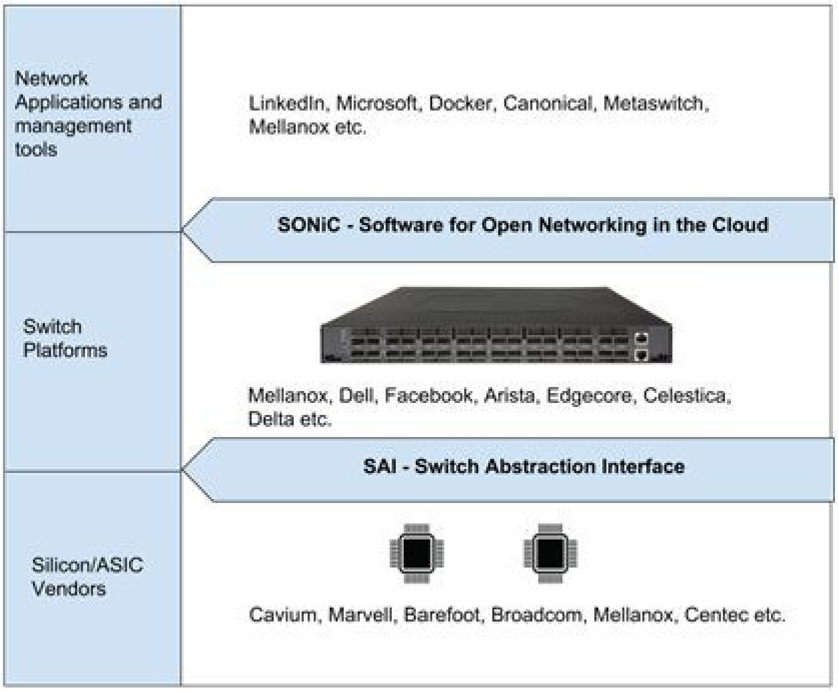

Fig 1.5 - Data Abstraction Function Layer

Layer 6: The Application Layer

Layer 6 is the application layer, which comprises the custom applications that are actually making use of the data

The IoT Reference Model does not strictly define an application. Applications vary based on vertical markets, the nature of device data, and business needs. For example, some applications will focus on monitoring device data. Some will focus on controlling devices. Some will combine device and non-device data. Monitoring and control applications represent many different application models, programming patterns, and software stacks, leading to discussions of operating systems, mobility, application servers, hypervisors, multi-threading, multi-tenancy, etc. These topics are beyond the scope of the IoT Reference Model discussion. Suffice it to say that application complexity will vary widely. Examples include

- Mission-critical business applications, such as generalized ERP or specialized industry solutions
- Mobile applications that handle simple interactions
- Business intelligence reports, where the application is the BI server

- Analytic applications that interpret data for business decisions © 2014 Cisco and/or its affiliates. All rights reserved. This Draft document is currently Cisco Confidential. Page 10 of 12
- System management/control center applications that control the IoT system itself and don't act on the data produced by it

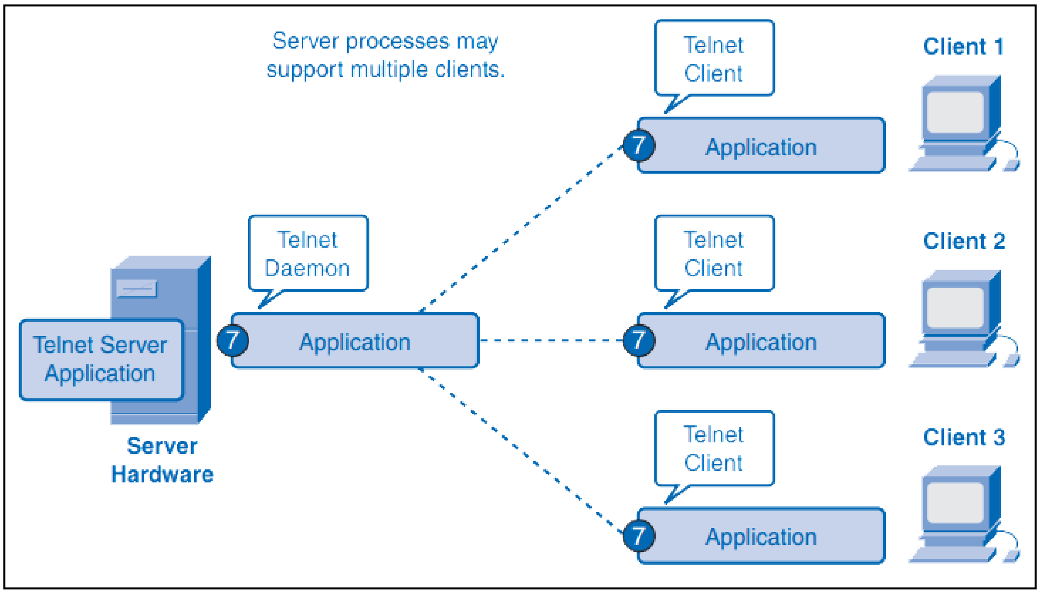

Fig 1.6 - Application Layer

Layer 7: People and Process Layer

Layer 7 is the people and the process layer. This includes people, businesses, collaboration, and decision making based on the information derived from IoT computing.

Applications execute a business logic to empower people. People use applications and associated data for their specific needs. Often, multiple people use the same application for a range of different purposes. So the objective is not the application—it is to empower people to do their work better.

Applications (Level 6) give business people the right data, at the right time, so they can do the right thing. But frequently, the action needed requires more than one person. People must be able to communicate and collaborate, sometimes using the traditional Internet, to make the IoT useful. Communication and collaboration often require multiple steps. And it usually transcends multiple applications. This is why Level 7 represents a higher level than a single application.

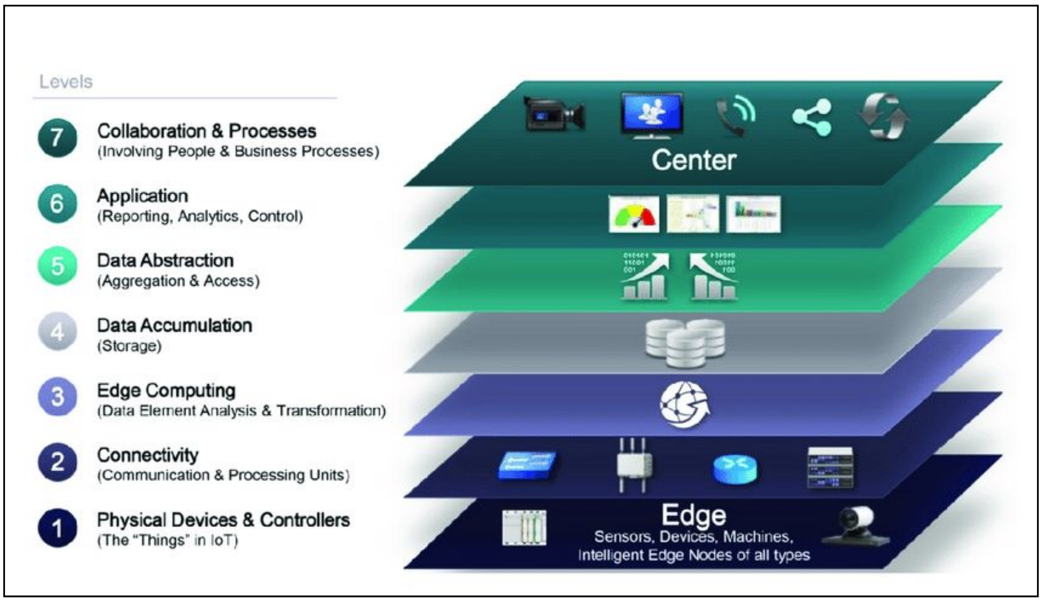

Fig 1.7 - Involving People and Business Processes

IoT design methodology

Step 1: Purpose & Requirements Specification

- The first step in the IoT system design methodology is to define the purpose and requirements of the system. In this step, the system purpose, behavior, and requirements (such as data collection requirements, data analysis requirements, system management requirements, data privacy and security requirements, user interface requirements, ...) are captured.

Step 2: Process Specification

- The second step in the IoT design methodology is to define the process specification. In this step, the use cases of the IoT system are formally described based on and derived from the purpose and requirement specifications.

Step 3: Domain Model Specification

- The third step in the IoT design methodology is to define the Domain Model. The domain model describes the main concepts, entities, and objects in the domain of IoT systems to be designed. The domain model defines the attributes of the objects and relationships between objects. The domain model provides an abstract representation of the concepts, objects, and entities in the IoT domain, independent of any specific technology or platform. With the domain model, the IoT system designers can get an understanding of the IoT domain for which the system is to be designed.

Step 4: Information Model Specification

- The fourth step in the IoT design methodology is to define the Information Model. Information Model defines the structure of all the information in the IoT system, for example, attributes of Virtual Entities, relations, etc. The information model does not describe the specifics of how the information is represented or stored. To define the information model, we first list the Virtual Entities defined in the Domain Model. The information model adds more details to the Virtual Entities by defining their attributes and relations.

Step 5: Service Specifications

- The fifth step in the IoT design methodology is to define the service specifications. Service specifications define the services in the IoT system, service types, service inputs/output, service endpoints, service schedules, service preconditions, and service effects.

Step 6: IoT Level Specification

- The sixth step in the IoT design methodology is to define the IoT level for the system. In Chapter-1, we defined five IoT deployment levels.

Step 7: Functional View Specification

- The seventh step in the IoT design methodology is to define the Functional View. The Functional View (FV) defines the functions of the IoT systems grouped into various Functional Groups (FGs). Each Functional Group either provides functionalities for interacting with instances of concepts defined in the Domain Model or provides information related to these concepts.

Step 8: Operational View Specification

- The eighth step in the IoT design methodology is to define the Operational View Specifications. In this step, various options pertaining to the IoT system deployment and operation are defined, such as service hosting options, storage options, device options, application hosting options, etc

Step 9: Device & Component Integration

- The ninth step in the IoT design methodology is the integration of the devices and components.

Step 10: Application Development

- The final step in the IoT design methodology is to develop the IoT application.

Chapter 2: Review of Literature

The emergence of Coronavirus (Covid-19) has caused unexpected challenges to the health of the people in this world. Thus, the proposed system aims at reducing the transmission rate of the pandemic. For ages, hand washing with cleanser and water has been viewed as a proportion of individual cleanliness. The proposed system mainly highlights the usage of automatic hand sanitizer dispensers over manually operated sanitizers in order to bring automation and at the same time reduce the probability of getting infected to Covid-19[2].

The proposed system showed effectiveness of contactless hand sanitizers, which reduced the Covid-19 infection rates by a whopping 35%. The users were given a significant ounce of sanitizer(liquid) which mainly consisted of germs fighting chemicals such as ethanol and isopropanol[4]. Further, for a 10 month period of using automatic hand sanitizers showed a result of 46.1% infection reduction. Several antibiotics were experimented in order to completely disinfect the hands from covid-19. After a proper research, the World Health Organization (WHO) recommended the use of alcohol-based sanitizers which marginally had negative association with the germs. Considering this fact, the proposed system includes these constituents in the sanitizer with a view to prove effective[1].

Thus, the model gives a complete comparison between manually operated hand sanitizers and automatic hand sanitizer dispensers and also underlines an important point that bringing in as much as automation in the Health and Medicine sector would help in maintaining good health hygiene and social distancing to win over this challenging pandemic. Hence, Zero Contact Hand Sanitizer Dispenser is an attempt to contribute in providing safety and prioritize automation over manual operations.

Chapter 3: Problem Statement

In this current scenario of a global outbreak, the World Health Organization (WHO) should maintain healthy hand wash and sanitization habits. The main problem in health hand wash is the way we do it, especially in public places, wherein we physically touch the bottle cap. This poses some of the risk factors, where there might be a possibility that an infected person might have come in contact with the sanitizer dispenser. If proper care is not taken, this might prove dangerous for other people as the virus is known to spread via physical contact.

Therefore, it is of utmost importance to develop and implement measures to reduce any of the physical contact possible. It is also highly encouraged to implement automation using the various provisions provided by the Internet of Things in this particular problem scenario.

Chapter 4: Objectives

- To maintain Social Distancing.
- To reduce spreading of covid-19 proactively.
- To emphasize the need of automation.
- To educate users about the intensity of Covid-19 cases in their area.
- To introduce automation in simple day to day necessities using IoT.
- To create cost efficient IoT based devices that can be used anywhere, and acquaint users of live covid-19 statistics in areas with internet connectivity.

Chapter 5: Project Details/ Flow

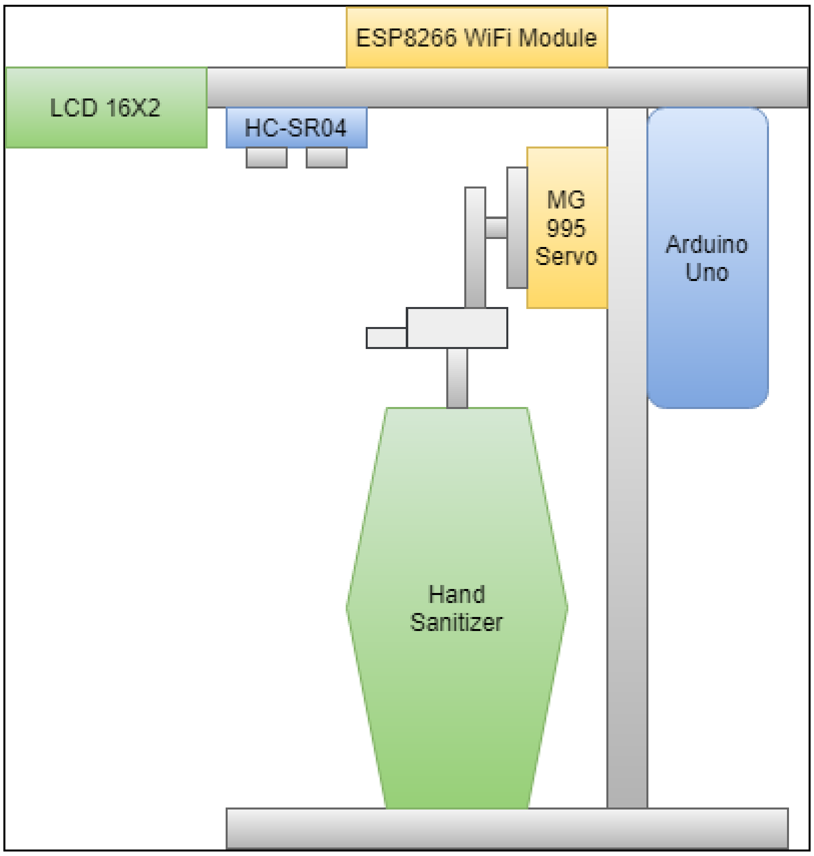

Fig 5.0 - Intended Design of the Apparatus

The Zero Contact Hand Sanitizer dispenser consists of an apparatus which houses the components required for the project according to the above diagram. There is a hand sanitizer bottle with a dispenser cap resting on a surface, in front of the column which houses the Arduino UNO microcontroller and the MG995 servo motor. Above the column, there is a beam which houses the ESP8266 Wi-Fi shield, the ultrasonic sensor and the LCD display. The ESP8266 connects to a Wi-Fi network and fetches the Covid-19 statistics from the internet and displays it on the LCD when the device is in an idle state.

Whenever the user brings their hands in the range of the ultrasonic sensor, the device activates, meaning the servo motor will rotate 90 degrees and in the process, it will press the cap of the hand sanitizer thus dispensing the liquid on the user’s palms. This design ensures that there is no contact between the apparatus and the user, thus maintaining a hygienic environment in the apparatus and a lower chance of contamination.

The block diagram of the expected functioning of the apparatus is given below.

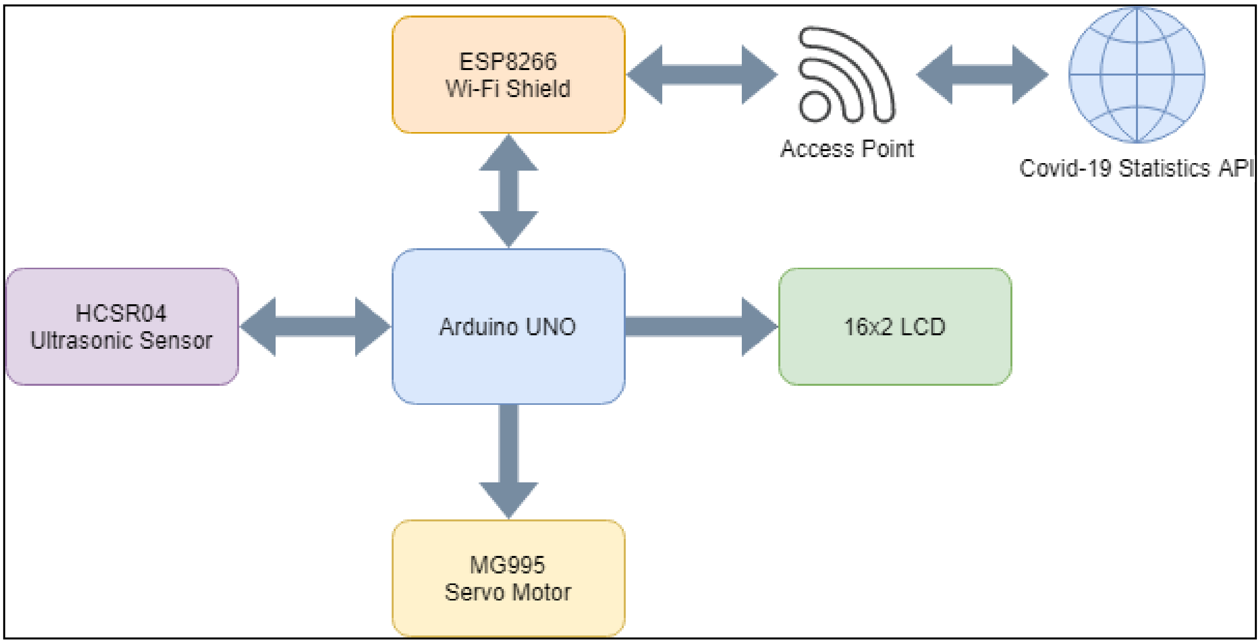

Fig 5.1 - Block Diagram of the Zero Contact Hand Sanitizer Dispenser

The Arduino UNO Microcontroller manages the various IoT based components and sensors as shown. The ESP8266 is first connected to a wireless access point, from where it accesses the internet and fetches data from a Covid-19 statistics API.

The microcontroller controls the LCD and the Servo Motor. The Ultrasonic Sensor sends its readings to the microcontroller and the timings of the wave impulses are managed by the microcontroller. The ESP8266 Wi-Fi shield sends the fetched JSON from the API to the microcontroller where it is parsed and the required information is printed on the LCD.

The working of the ZCHSD is explained with the help of a flowchart below.

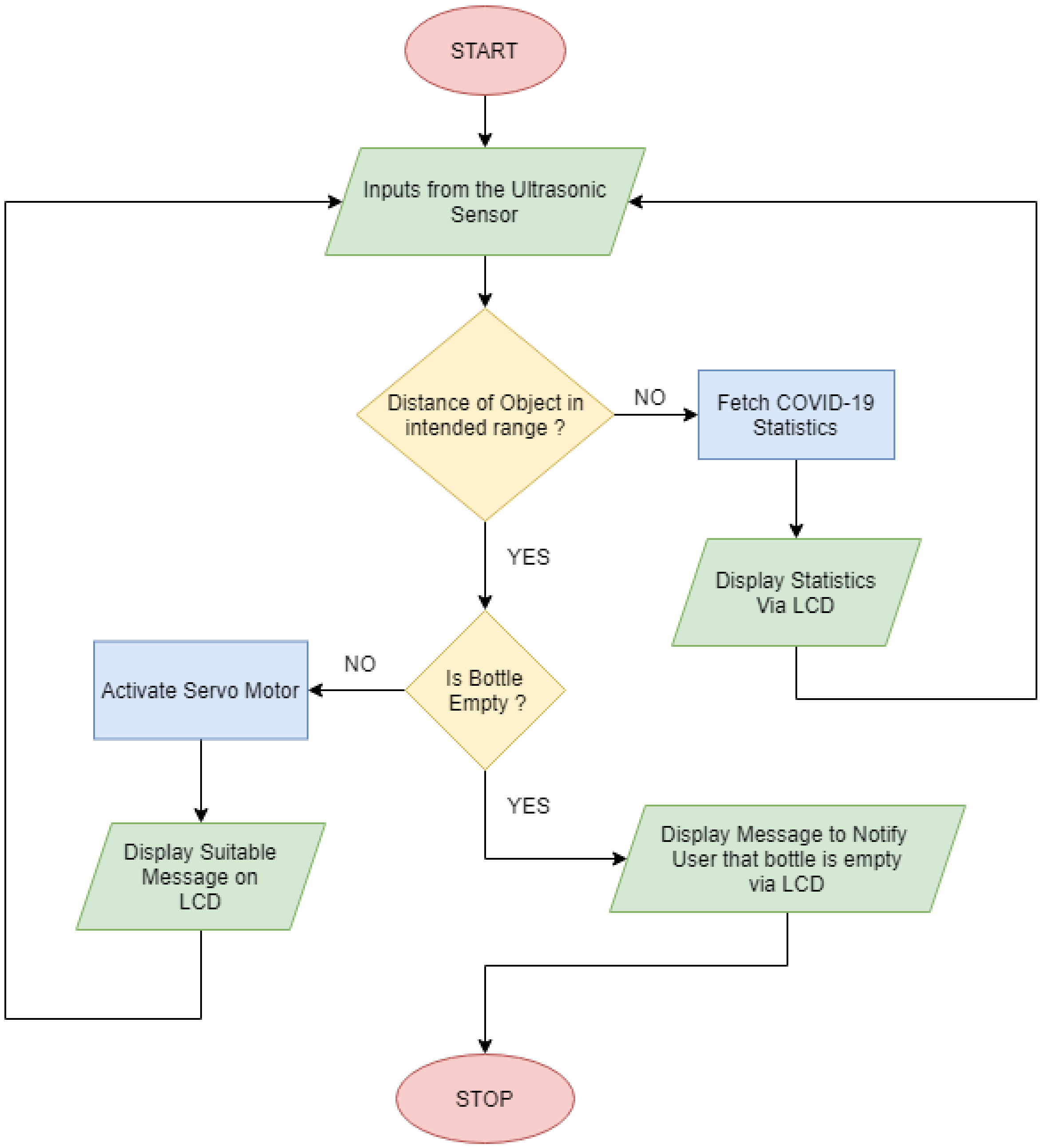

Fig 5.2 - Flowchart

Chapter 6: Circuit Diagram

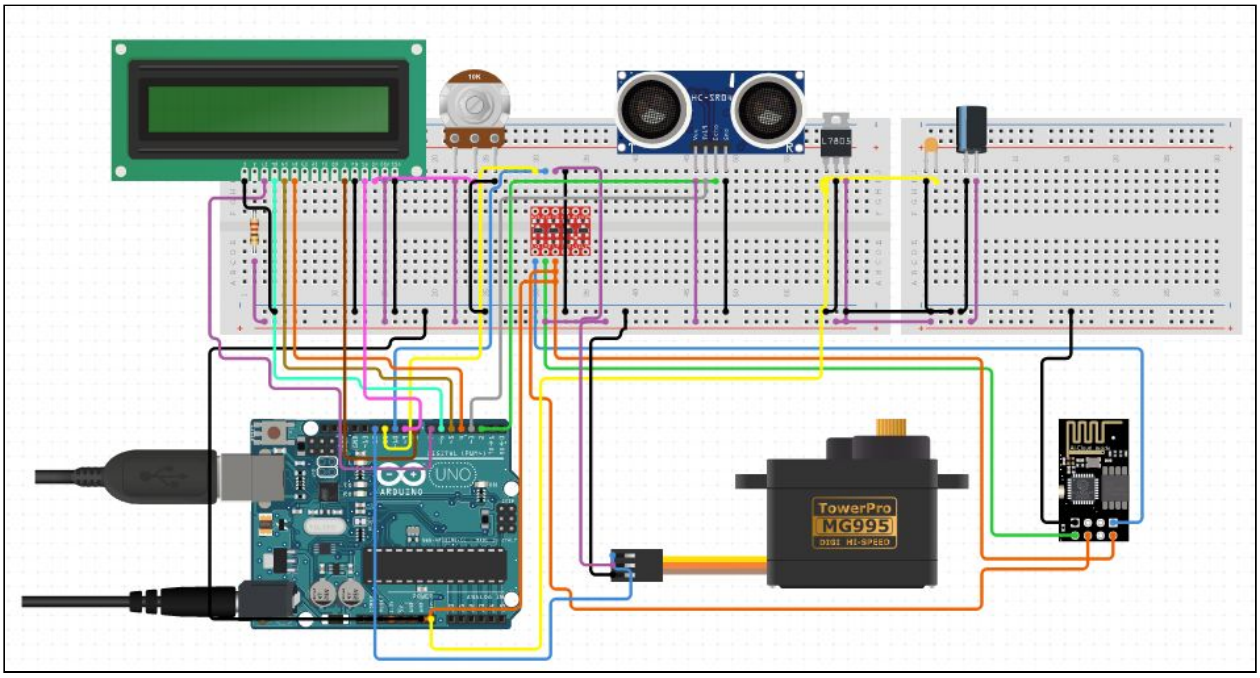

Fig 6.0 - Circuit diagram of Zero Contact Hand Sanitizer Dispenser

Ultrasonic sensor: VCC Pin is connected to the 5V input of the Arduino board.GND Pin is connected to the Ground of the Arduino. The TRIG pin is connected to the Arduino at pin 3 as the output of the sensor and the ECHO pin is connected to the Arduino at pin 2 as the input of the ultrasonic sensor.

Servo Motor: The POWER of the servo motor is connected to the 5V power supply of the breadboard.GND Pin is connected to the Ground of the Arduino and The SIGNAL of the servo motor is connected to the pin 10 of the Arduino.

LCD Display: Pin1(GND) of LCD is connected to the Ground of the Arduino, Pin2(VCC) is connected to the 5V power supply of the breadboard, Pin3(V0) a contrast pin is grounded with the 10-kilo ohm potentiometer, Pin4(RS) is connected to the pin 9 of the Arduino, Pin5(RW) read/write pin is connected to the ground of the breadboard, Pin6(E) enable pin is connected to the pin 8 of the Arduino, Pin D4, D5, D6, D7 is connected to the pin 4, 5, 6, 7 of the Arduino respectively.

Anode pin (A) is given a 5V power supply through a 220-ohm resistor to the breadboard and the Cathode pin (K) is grounded to the breadboard.

Chapter 7: Requirement of Software and Hardware

Hardware Requirements

The research was done for various components, their uses, and their simulations on various electronic circuit simulation software. The hardware components and working of the prototype are given below.

1. Arduino UNO
2. LCD 16X2
3. HC-SR04 Ultrasonic Sensor
4. MG995 Metal Gear Servo Motor
5. Generic Hand Sanitizer Bottle with a dispenser cap

A. <u>Arduino UNO</u>

Arduino Uno is a microcontroller. It is a board based on an 8-bit ATmega328P microcontroller. Along with ATmega328P, it consists of other components such as crystal oscillator, serial communication, voltage regulator, etc. to support the microcontroller. Arduino Uno has 14 digital input/output pins (out of which 6 can be used as PWM outputs), 6 analog input pins, a USB connection, A Power barrel jack, an ICSP header and a reset button.

Fig 7.0 - Arduino UNO Microcontroller

Following are the specifications for the chosen microcontroller:

Microcontroller	ATmega328P – 8-bit AVR family microcontroller
Recommended Input Voltage	7-12V
Operating Voltage	5V
Analog Input Pins	6(A0 – A5)
Input voltage limits	6 – 20V
Digital I/O Pins	14 (Out of which 6 provide PWM output)
DC Current on I/O Pins	40 mA
DC Current on 3.3V Pin	50mA
SRAM	520kB
EEPROM	1Kb
Frequency (Clock Speed)	16MHz
Bluetooth	V4.2-Supports BLE and Classic Bluetooth

Flash memory	32 KB (0.5 KB is used for Bootloader

B. HC-SR04 Ultrasonic Sensor

HC-SR04 Ultrasonic sensor is a 4-pin module, whose pin names are Vcc, Trigger, Echo and Ground individually. This sensor is a mainstream sensor utilized in numerous applications where estimating separation or detecting objects are required. The module has two eyes like activities in the front which frames the Ultrasonic transmitter and Receiver. The sensor works with the simple high school formula that

$$Distance = Speed \times Time$$

The Ultrasonic transmitter sends an ultrasonic wave, this wave goes in air and when it gets protested by any material it gets reflected back toward the sensor this reflected wave is seen by the Ultrasonic collector module.

Fig 7.1 - HCSR04 Ultrasonic Sensor

Following are the features of the HC-SR04:

- Operating voltage: +5V
- Theoretical Measuring Distance: 2cm to 450cm
- Practical Measuring Distance: 2cm to 80cm
- Accuracy: 3mm
- Measuring angle covered: <15°
- Operating Current: <15mA
- Operating Frequency: 40Hz. [8]

C. <u>MG995 Metal Gear Servo Motor</u>

A servomotor is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration. It consists of a suitable motor coupled to a sensor for position feedback. The servo is suited for designing robotic arms in which wear and tear of the motor is high.

Being metal outfitted, the servo has a long life and can be introduced on a framework like a mechanical arm where engine work is gigantic. Servomotors are utilized in application, for example, advanced mechanics, CNC hardware or robotized production.

Fig 7.2 - MG995 Metal Gear Servo Motor

Following are its features and characteristics:

- Metal geared servo for more life
- Stable and shock proof double ball bearing design
- High speed rotation for quick response
- Fast control response
- Constant torque throughout the servo travel range
- Excellent holding power
- Weight: 55 g
- Dimension: 40.7×19.7×42.9mm
- Operating voltage range: 4.8 V to 7.2 V
- Stall torque: 9.4kg/cm (4.8v); 11kg/cm (6v)
- Operating speed: 0.2 s/60º (4.8 V), 0.16 s/60º (6 V)
- Rotational degree: 180º

- Dead band width: 5 μs
- Operating temperature range: 0ºC to +55ºC
- Current draw at idle: 10mA
- No load operating current draw: 170mA
- Current at maximum load: 1200mA

D. <u>LCD 16X2</u>

LCD modules are very commonly used in most embedded projects, the reason being its cheap price, availability and programmer friendly. The 16×2 LCD is named so because; it has 16 Columns and 2 Rows (Fig 4). So, it will have (16×2=32) 32 characters in total and each character will be made of 5×8 Pixel Dots.

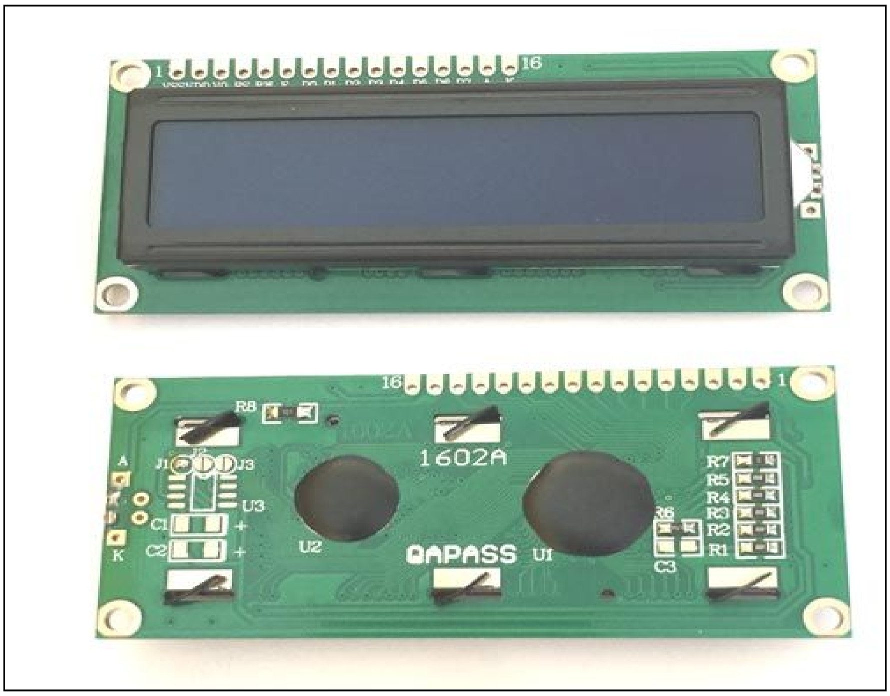

Fig 7.3 - LCD 16X2

Features of LCD 16x2 module:

- Operating Voltage is 4.7V to 5.3V
- Current consumption is 1mA without backlight
- Alphanumeric LCD display module, meaning can display alphabets and numbers

- Consists of two rows and each row can print 16 characters.
- Each character is built by a 5×8-pixel box
- Can work on both 8-bit and 4-bit mode
- It can also display any custom generated characters
- Available in Green and Blue Backlight.[8]

E. ESP8266 WiFi Module:

The ESP8266 is a very user friendly and low-cost device to provide internet connectivity to your projects. The module can work both as an access point (can create hotspot) and as a station (can connect to Wi-Fi), hence it can easily fetch data and upload it to the internet making Internet of Things as easy as possible. It can likewise get information from web utilizing API's henceforth your undertaking could get to any data that is accessible in the web, in this manner making it more brilliant. Another energizing element of this module is that it tends to be customized utilizing the Arduino IDE which makes it much more easy to use.

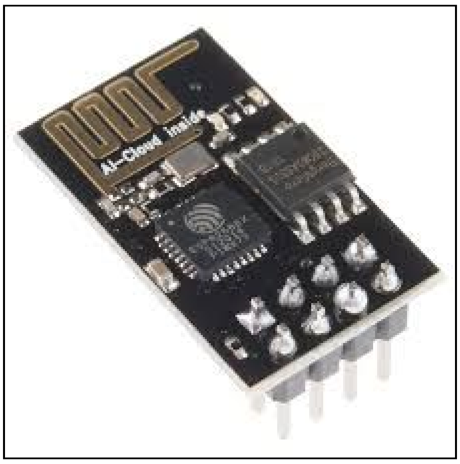

Fig 7.4 - ESP8266 Wi-Fi Module

Following are its applications:

- IOT Projects
- Access Point Portals
- Wireless Data logging
- Smart Home Automation
- Learn basics of networking
- Portable Electronics
- Smart bulbs and Sockets

Software Requirements

Arduino Integrated Development Environment (IDE)

The Arduino Integrated progress Environment - or Arduino Software (IDE) - It includes a text editor to write a program, a message field, a text area, a toolbar which has buttons for a common method and a series of menus. It connects with the Arduino and Genuino hardware to load the program and establish communication with them. These sketches are to be written in the text editor and are saved with the extension .ino.The editor has characteristics for cutting/pasting and for searching/replacing text. The message area gives information while saving and also shows errors. The console shows text output by the Arduino Software (IDE), which includes complete error messages and other information. The lowermost right-hand corner of the window shows the configured board and serial port. The toolbar buttons allow you to validate and upload programs, create, open, and save sketches, and open the serial monitor.

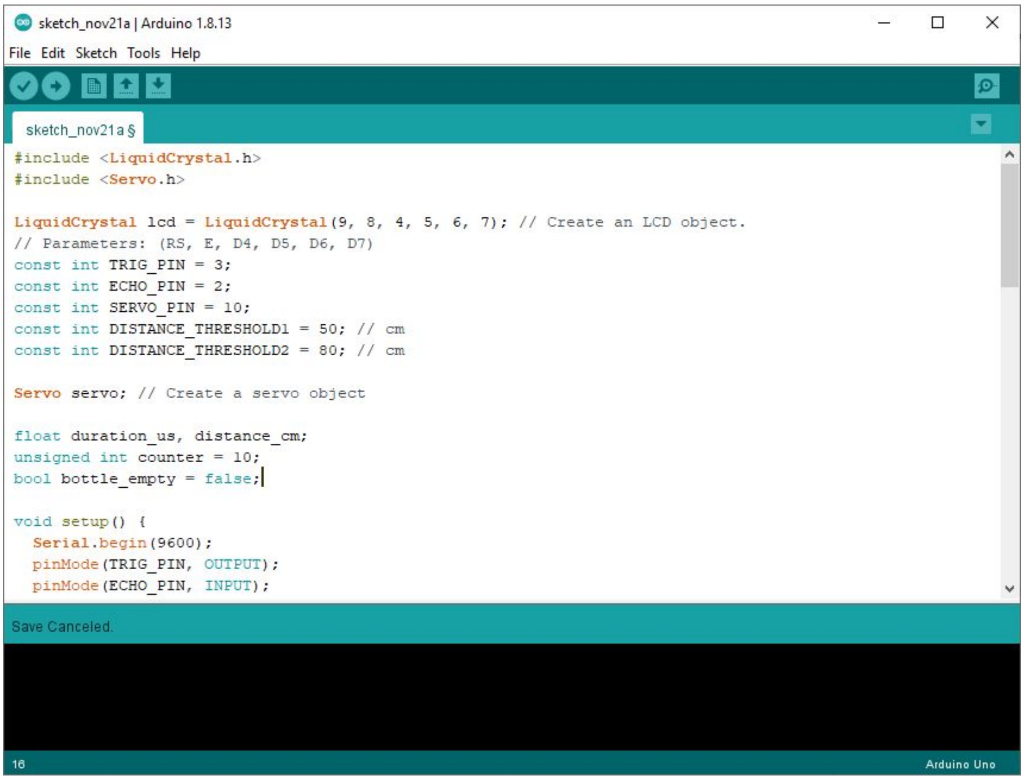

Fig 7.5 - Arduino IDE

Chapter 8: Implementation

CODE FOR ZCHSD:

```
#include <LiquidCrystal.h>
#include <Servo.h>

LiquidCrystal lcd = LiquidCrystal(9, 8, 4, 5, 6, 7);
// Create an LCD object.
// Parameters: (RS, E, D4, D5, D6, D7)
const int TRIG_PIN = 3;
const int ECHO_PIN = 2;
const int SERVO_PIN = 10;
const int DISTANCE_THRESHOLD1 = 50;
const int DISTANCE_THRESHOLD2 = 80; //cm

Servo servo; //create a servo object

float duration_us, distance_cm;
unsigned int counter = 10;
bool bottle_empty = false;

void setup() {
  Serial.begin(9600);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  servo.attach(10);
  servo.write(0);
  lcd.begin(16, 2);
}

void loop() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  duration_us = pulseIn(ECHO_PIN, HIGH);
```

```
  distance_cm = 0.017 * duration_us; //calculate the
distance


  if (!bottle_empty) {
    if (distance_cm >= DISTANCE_THRESHOLD1 &&
distance_cm <= DISTANCE_THRESHOLD2 && counter != 0) {

    if (servo.read() != 0) {
    Serial.print("counter: ");
    Serial.print(counter);
    Serial.print("\n");
        servo.write(0);
    counter--;
    lcd.setCursor(0, 0);
    lcd.print("Thank You  ");
    lcd.setCursor(0, 1);
    lcd.print("Stay Safe   ");
    }
  } else {
    servo.write(90);
      lcd.setCursor(0, 0);
    lcd.print("Bring Your");
    lcd.setCursor(0, 1);
    lcd.print("Hands Closer");
      }
  }
  // For LCD Display

  if (counter == 0) {
    bottle_empty = true;
    lcd.setCursor(0, 0);
    lcd.print("Bottle Empty! ");
    lcd.setCursor(0, 1);
    lcd.print("Please Refill.");
  }
  delay(500);

}
```

Chapter 9: Testing

Test 1: The object is ***not*** in range of the ultrasonic sensor and the bottle is ***not empty***.

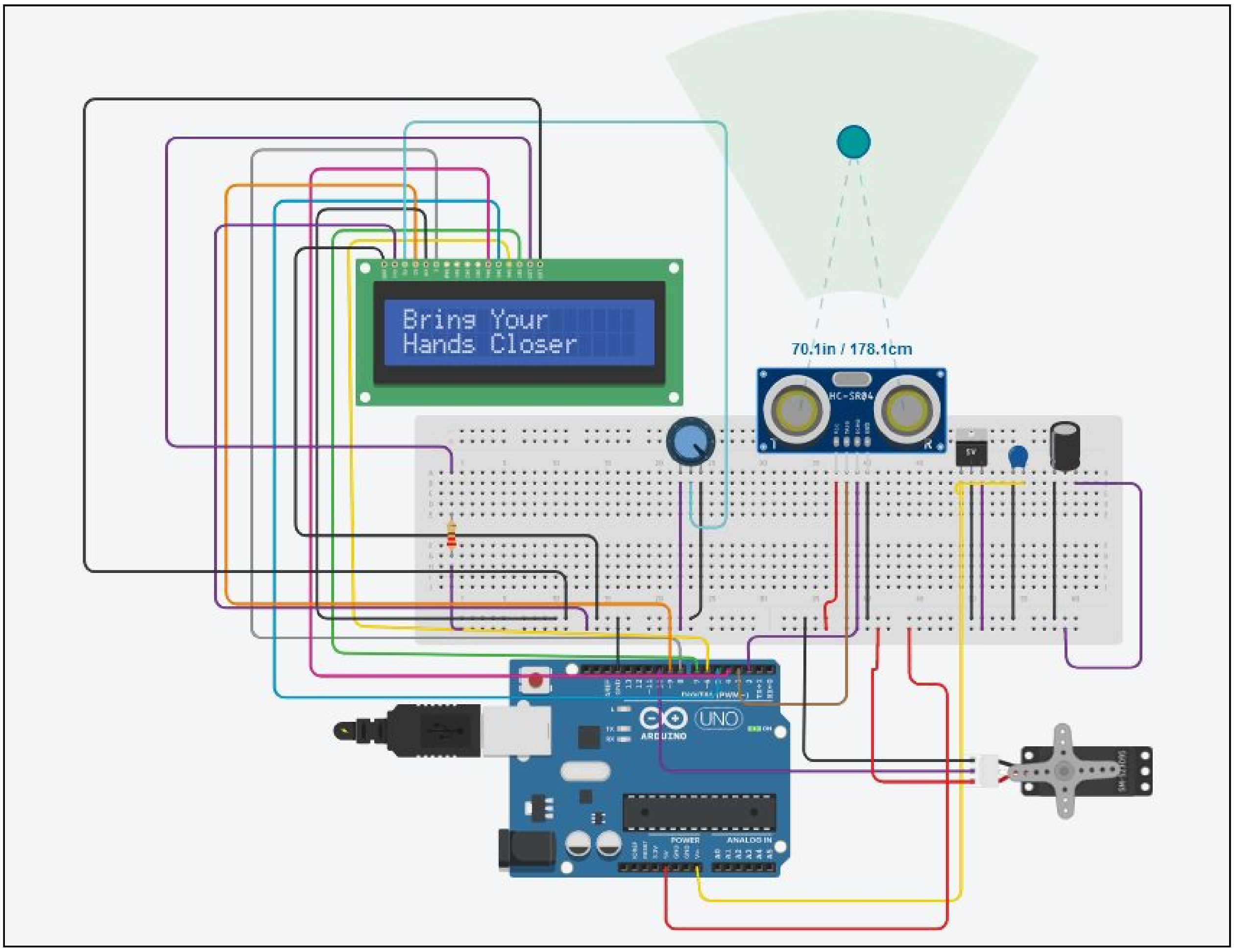

Fig 9.0 - Test 1

Test 2: The object ***is*** in range of the ultrasonic sensor and the bottle is ***not empty***.

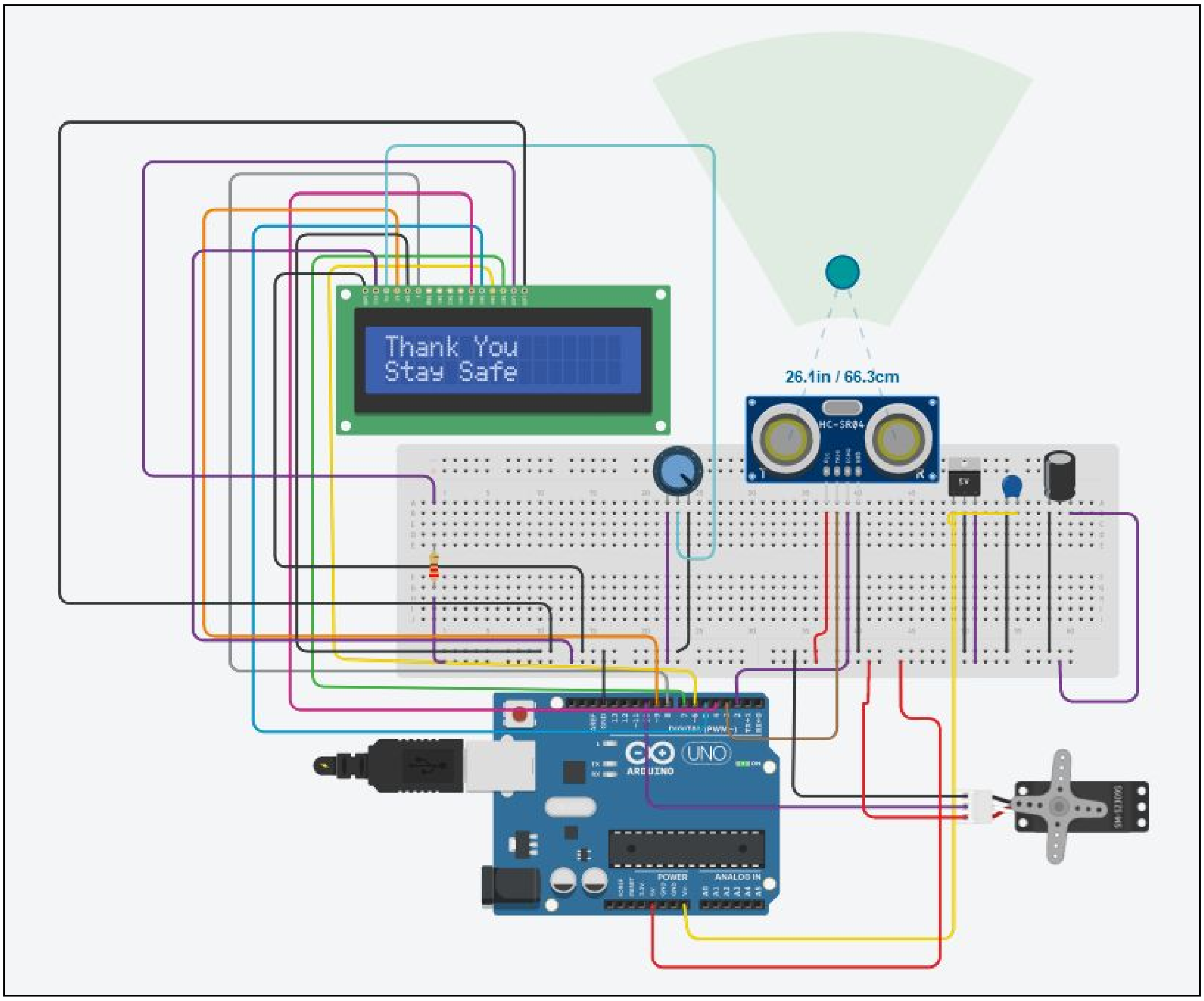

Fig 9.1 - Test 2

Test 3: The object ***is not*** in range of the ultrasonic sensor and the bottle ***is empty***.

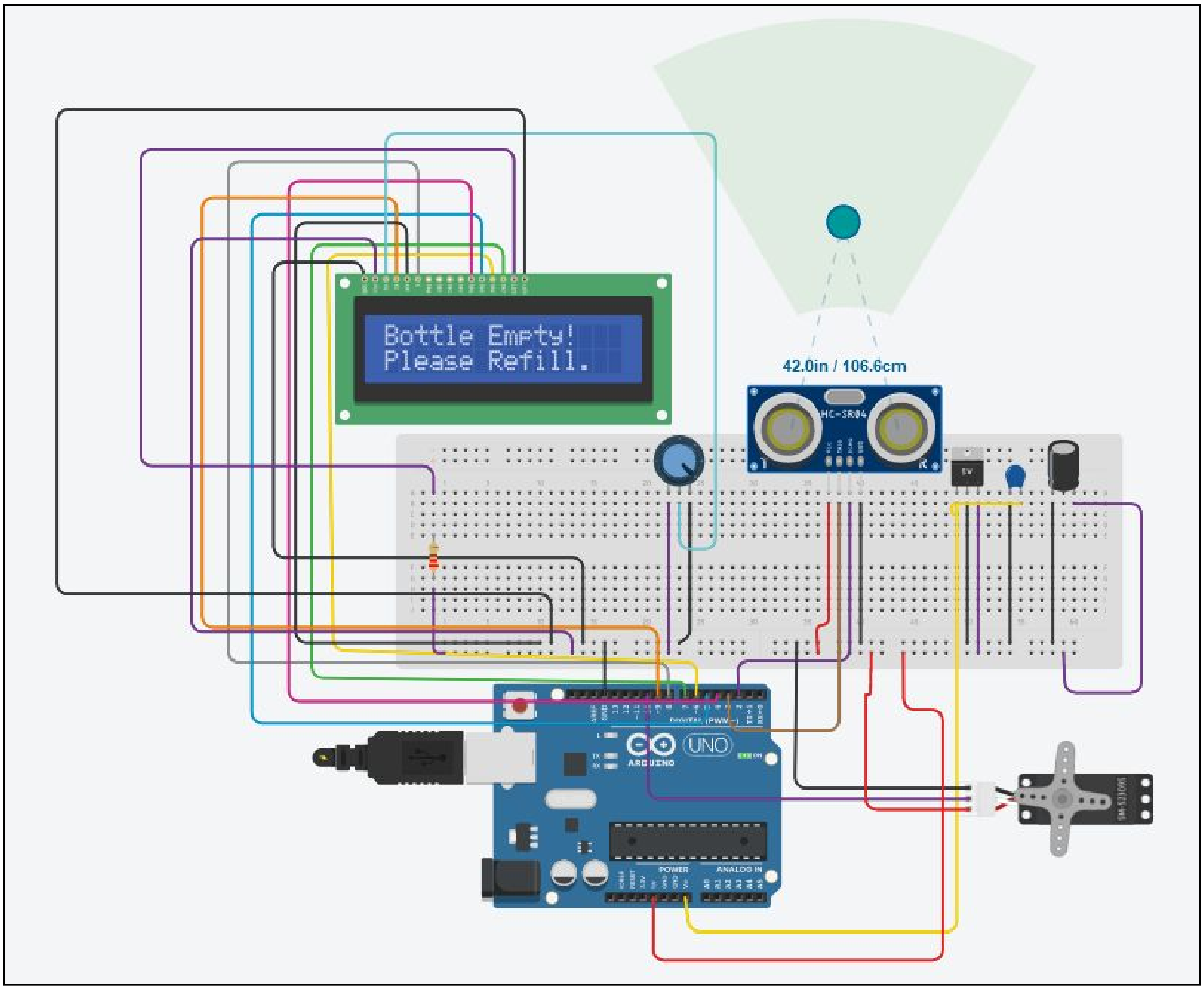

Fig 9.2 - Test 3

Test 4: The object ***is*** in range of the ultrasonic sensor and the bottle ***is empty***.

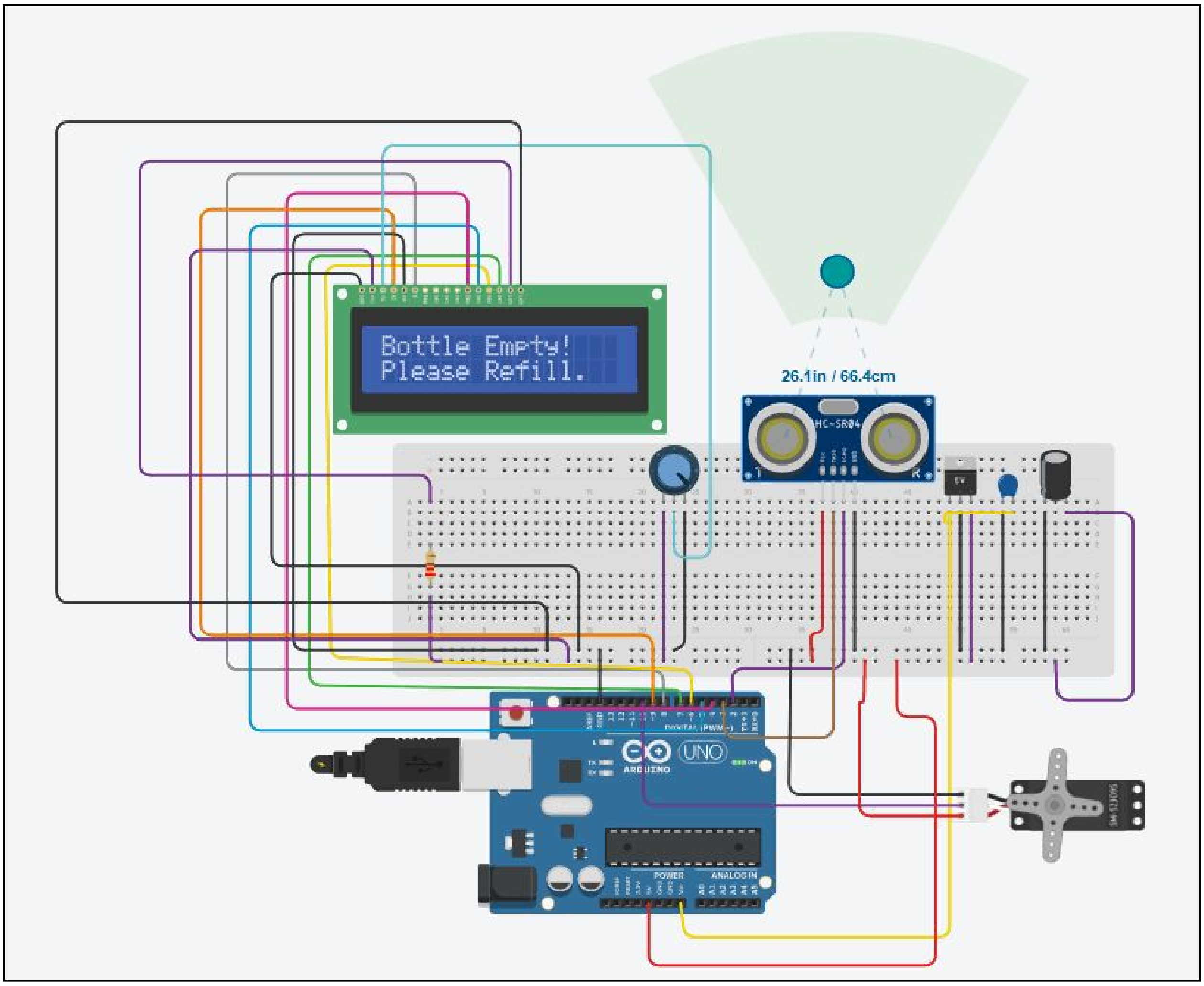

Fig 9.3 - Test 4

Chapter 10: Results

Results of Test 1:
The servo motor stays at its initial position and the message “Bring Your Hands Closer” is displayed on the LCD screen indicating that the bottle is not empty and the device needs the user to bring their hands closer so that they are in the range defined.

Results of Test 2:
The servo motor rotates 90 degrees and the message “Thank You Stay Safe” is displayed on the LCD screen indicating that the user has brought their hands in the range of the ultrasonic sensor and the hand sanitizer was dispensed successfully.

Results of Test 3:
The servo motor does not activate and the message “Bottle Empty Please Refill” is shown on the LCD indicating that the bottle is empty and needs to be replaced/refilled. After refilling or replacing the bottle the user can restart the apparatus to reset the counter of the bottle usage.

Results of Test 4:
The servo motor does not activate and the message “Bottle Empty Please Refill” is shown on the LCD indicating that the bottle is empty and needs to be replaced/refilled. The system basically goes into a dead state unless the bottle is not refilled/replaced and the system is restarted.

Chapter 11: Conclusion

The coronavirus COVID-19 pandemic is the defining global health crisis and the greatest challenge we have faced since World War-II. It is wreaking havoc in the world. Since its emergence in Asia late last year, the virus has spread to every possible continent in the world. The World Health Organization (WHO) has already announced it as a pandemic disease and recommended alcohol-based hand sanitizers for frequent hand hygiene, which are mainly made up of ethanol, isopropyl alcohol, hydrogen peroxides in different combinations. According to the Worldometers statistics, the figures of death rate and people getting affected is increasing rapidly. The USA, Brazil, India, Russia are the most affected countries in the entire world. Several countries have announced a lockdown where people can't step out of their homes and are suffering through various aspects of life.

This ZCHSD is thus an attempt to reduce the impact of coronavirus by automation and a hands-free approach of using hand sanitizers in public places. Due to the modern appearance of our system, it can attract the attention of the people which will directly increase the need for automatic hand sanitizer dispensers. Also, our system delivers an adequate amount of dose of hand soap or sanitizer to the user. The key highlight aspect of the system is to have a Zero Contact with the user thereby eliminating the chances of getting affected by the virus through hands. This task is done with the help of MG995 Metal Gear Servo Motor which can rotate at an angle of 180 degrees by pressing the dispenser cap. Taking advantage of the ESP8266 Wi-Fi shield module connected to Arduino Uno, the users are also made aware of the intensity of cases in their area by fetching data through Application Programming Interface(API) and displaying it on the LCD screen embedded with our system. Thus, the automatic hand sanitizer device proposed is expected to contribute to contactless hand disinfection in public places and virus infection prevention.

Although this system might prove effective against germs, it's not a complete substitute when needing to remove any bodily fluids or dirt from hands. These types of substances need to be completely removed. Hence, this instigates the need to add more features to the ZCHSD which will satisfy the limitation shortly.

Future Scope :

Due to the on-going COVID pandemic, people's perception regarding health hygiene has seen a positive change thereby raising the health care expenditure and high investments by companies in the health sector. This ZCHSD is cost-effective with lower maintenance costs and most importantly user-friendly which ultimately will result in the growth of the automatic hand sanitizer dispenser market soon. If the budget is not the concern, then a thermal sensor can also be installed which will display the body temperature of the user and eventually alert the user if any abnormal symptoms are seen. So, to accommodate the ultrasonic sensor, as well as the infrared heat sensor, the Arduino Uno microcontroller, is used which has 6 analog pin support, which means it can support more than one analog sensor.

Chapter 12: References

[1] Ashish Chaudhary, "https://circuitdigest.com/microcontrollerprojects/automatic-hand-sanitizer-dispenser-with-covid19-liveupdates", website.

[2] herolivechannel, "https://create.arduino.cc/projecthub/herolivechannel/fighting-corona-multichoice-automatic-hand-sanitizer-b3df30", website.

[3] Alberto Berardi, Diego Perinelli, Hamid Merchant, Lorina Bisharat, Iman Basheti, Giulia Bonacucina, Marco Crespi, Giovanni Palmieri, "Hand sanitizers amid CoViD-19: A critical review of alcohol-based products on the market and formulation approaches to respond to an increasing demand", published.

[4] Mert Bal, Reza Abrishambaf, "A system for monitoring hand hygiene compliance based-on Internet-of-Things", published.

[5] Juhu Lee, Jin-Young Lee, Sung-Min Cho, Ki-Cheol Yoon, Young Jae Kim, Kwang Gi Kim, "Design of Automatic Hand Sanitizer System Compatible with Various Containers", published.

[6] Adeel Mahmood, Maryam Eagan, Saher Pervez, Huda Ahmed Alghamdi, Amtul Bari Tabinda, Abdullah Yasar, Kathirvel Brindhadevi, Arivalagan Pugazhendhi, "COVID-19 and frequent use of hand sanitizers; human health and environmental hazards by exposure pathways", published.

[7] Arduino, "https://store.arduino.cc/usa/arduino-uno-rev3", website.

[8] Components101, "https://components101.com/microcontrollers/arduino-uno", website.

[9] Enerst Edozie, Wantimba Janat, ZainaKalyankolo, " Design and Implementation of a Smart Hand Sanitizer Dispenser with Door Controller using ATMEGA328P", published.