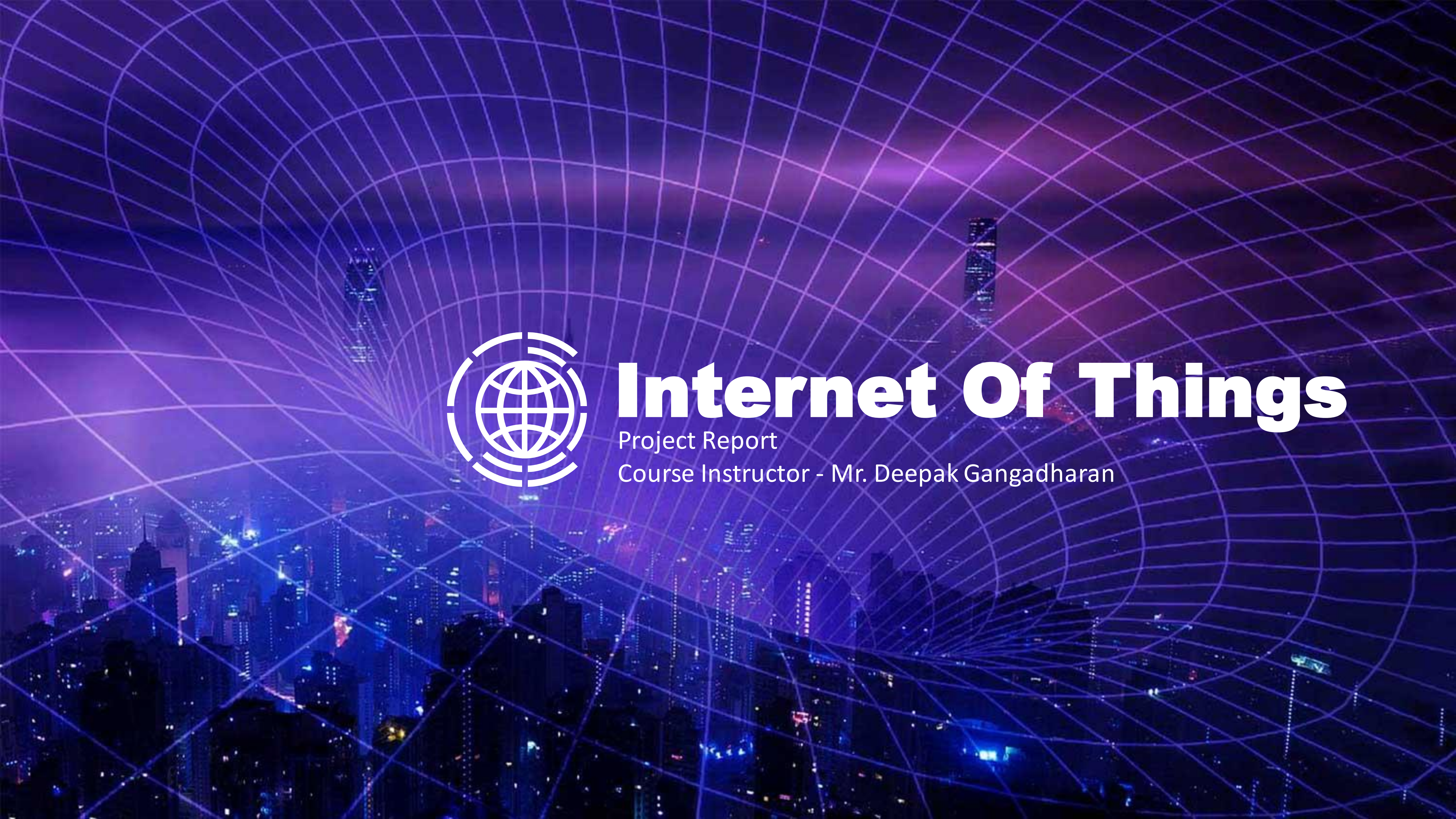


Internet Of Things

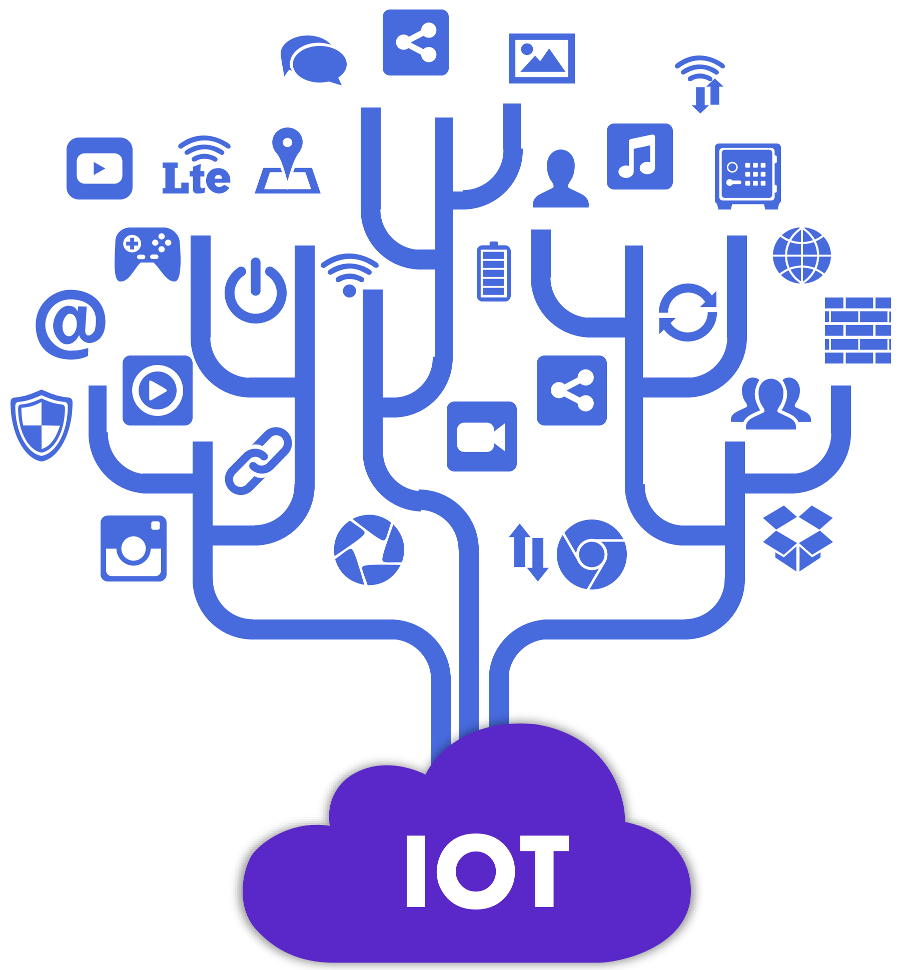
Project Report

Course Instructor - Mr. Deepak Gangadharan

Internet Of Things

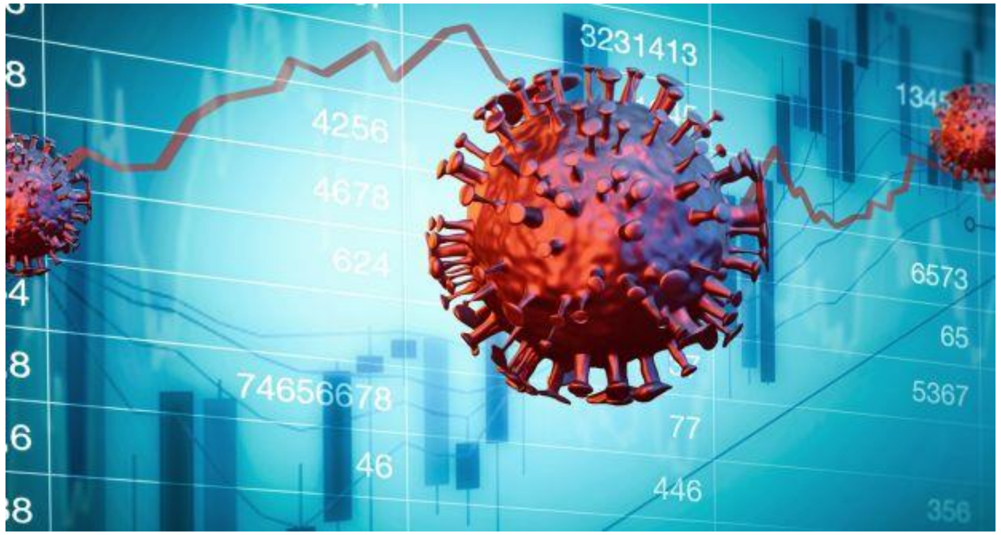
IOT

The Internet of Things (IoT) describes **the network of physical objects**—"things"—that are embedded with sensors, software, and other technologies for the purpose of connecting and exchanging data with other devices and systems over the internet.



CONTACTLESS HAND SANITIZER DISPENSER

Introduction



Coronavirus disease has spread to more than 213 countries infecting more than 26 Crore people and killing over more than 5 million globally, according to data compiled by world meter (as on Dec 2, 2021).

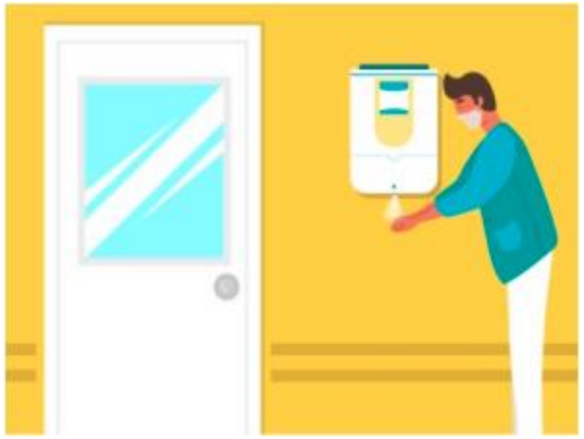
The world is fighting the Covid19 pandemic. There are many essential equipment needed to fight against Coronavirus. One of such essentials is Sanitizer. Before Sanitizer was not mandatory for everyone but as the day progressed scientists and Doctors have recommended everyone to use Sanitizer.

Despite the vaccine, the second wave of the virus has come and affected. Almost all countries- imposed lockdown. So in such circumstances, social distancing and use of contact less hand sanitizers is an effective way to tackle the deadly coronavirus.

As the pandemic slowly settles and such sectors become eager to resume in-person work, individuals are still skeptical of getting back to the office. Multiple studies have shown that the use of Sanitizers reduces the risk of viral transmission as well as provides a sense of protection. However, it is infeasible to manually enforce such a policy on large premises and track any violations. Contact-less Hand-Sanitizer Dispenser provides a better alternative.

Motivation

To limit Corona Virus spread, social distancing and observing hygiene standards like compulsory wearing of mask, use of hand gloves, face shield, and use of sanitizer is very important. So, we are planning to develop a completely **contact less hand-sanitizer dispenser.**



Features

Automatic Hand triggered dispenser

Daily Log Record of People using Dispenser (Count people)

Push and Email notification for Bottle refilling via Blynk Server

Temperature and humidity of the current location gets updated every few seconds.

Analyzing real-time data on ThingSpeak

Buzzer beep at low-to-critical levels for offline alerts.

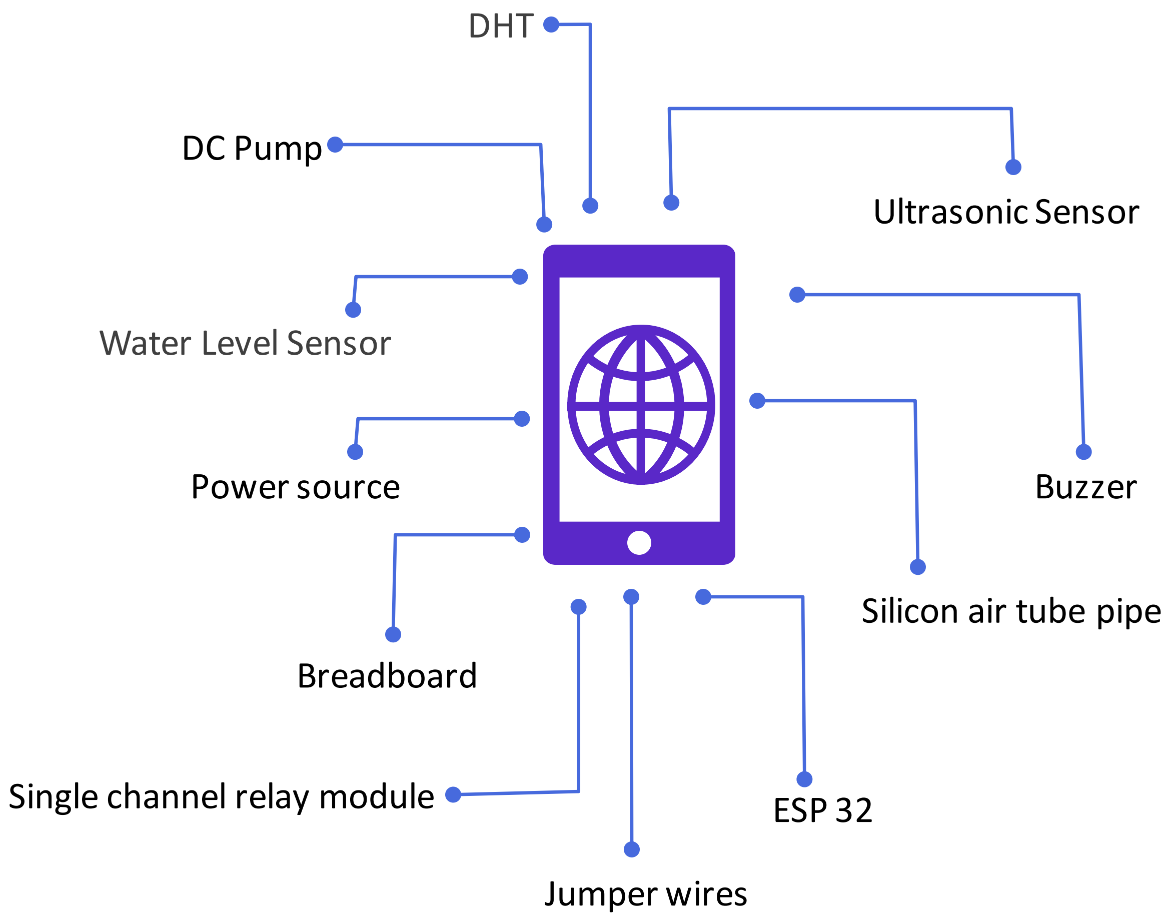
Contactless Hand Dispenser

Fluid Level Detection

Shows count of uses after each refill

System Overview

Hardware Components Used



DC Pump

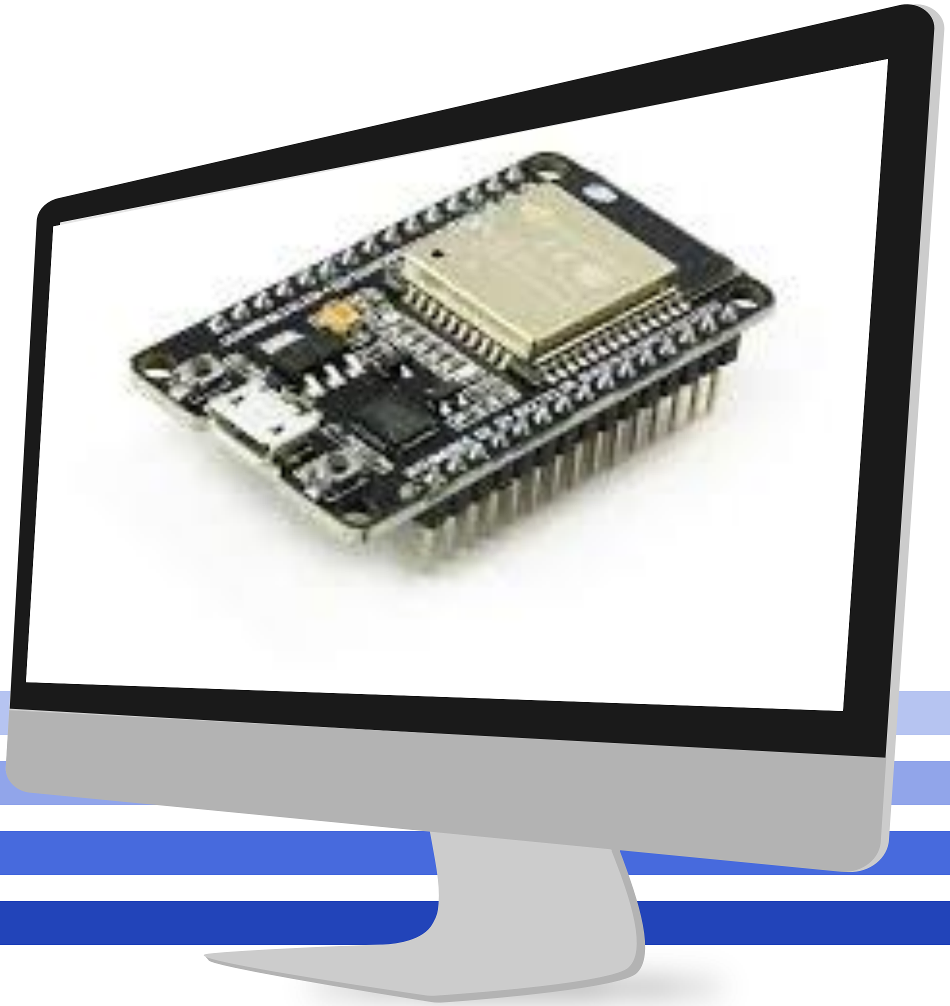
A High Performance non submersible dc water pump is a device which has a hermetically sealed motor close coupled to the pump body. Some part of assembly is submerged in the fluid to be pumped. The main advantage of this type of pump is that it prevents pump cavitation a problem associated with a high elevation difference between pump and the fluid surface. Submersible pumps push fluid to the surface as opposed to jet pumps having to pull fluids. The non-submersible pumps used in ESP installations are multistage centrifugal pumps operating in a vertical position. Although their constructional and operational features underwent a continuous evolution over the years, their basic operational principle remained the same. Condition: new: a brand-new, unused, unopened, undamaged item in its original packaging Mini Submersible Pump Motor: This is a low cost, small size Submersible Pump Motor. Just connect tube pipe to the motor outlet, submerge it in water and power it.

Ultrasonic Sensor

Ultrasonic sensors measure distance by using ultrasonic waves. The sensor head emits an ultrasonic wave and receives the wave reflected from the target. Ultrasonic Sensors measure the distance to the target by measuring the time between the emission and reception. HC-SR04 distance sensor is commonly used with both microcontroller and microprocessor platforms like Arduino, ARM, PIC, Raspberry Pie etc. The following guide is universally since it must be followed irrespective of the type of computational device used. Once the wave is returned after it getting reflected by any object the Echo pin goes high for a particular amount of time which will be equal to the time taken for the wave to return to the sensor. Application :- 1. Used to avoid and detect obstacles with robots like biped robot, obstacle avoider robot, path finding robot etc. 2. Can be used to map the objects surrounding the sensor by rotating it. 3. Depth of certain places like wells, pits etc. can be measured since the waves can penetrate through water

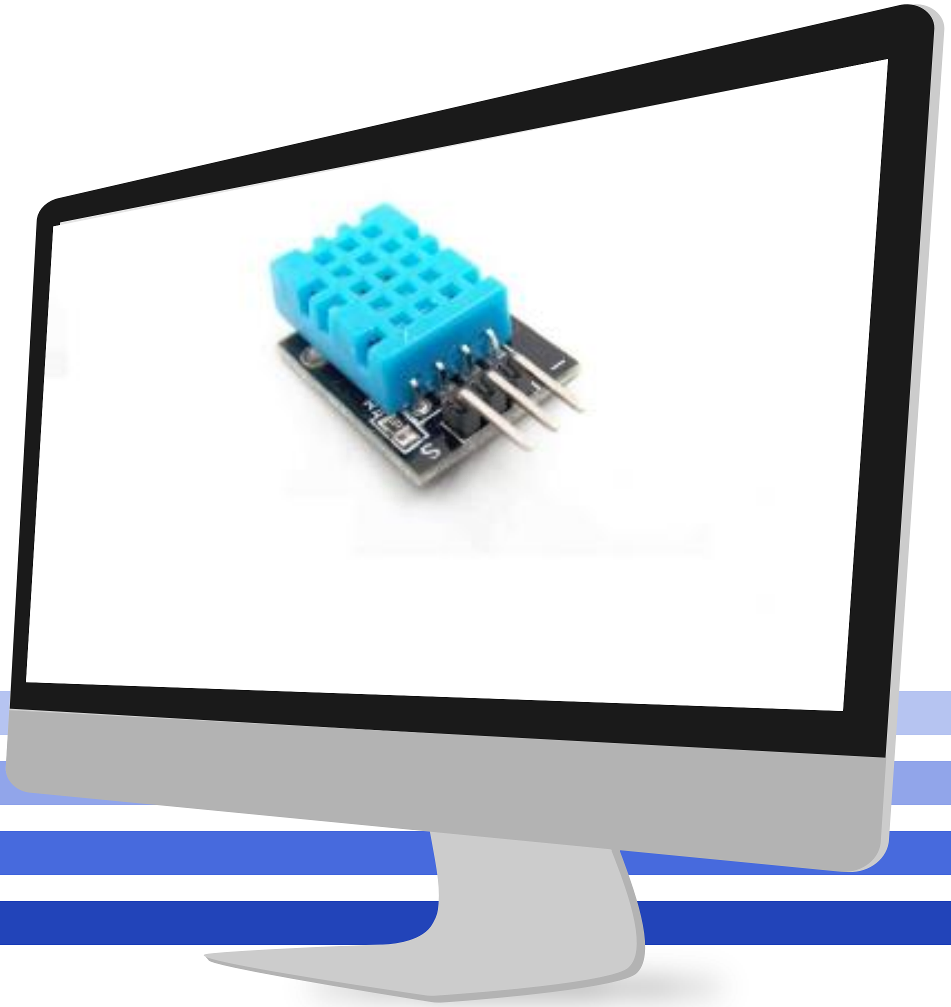
ESP32

- ESP32 is power packed with hardware features. The high-speed dual core processors along with the numerous built-in peripherals it is set to replace micro-controllers in connected products.
- The ESP32 is dual core, this means it has 2 processors.
- It has Wi-Fi and Bluetooth built-in.
- It runs 32-bit programs.
- The clock frequency can go up to 240MHz and it has a 512 kB RAM.
- This particular board has 36 pins, 15 in each row.
- It also has wide variety of peripherals available, like capacitive touch, ADCs, DACs, UART, SPI, I2C and much more.
- It comes with built-in hall effect sensor and built-in temperature sensor



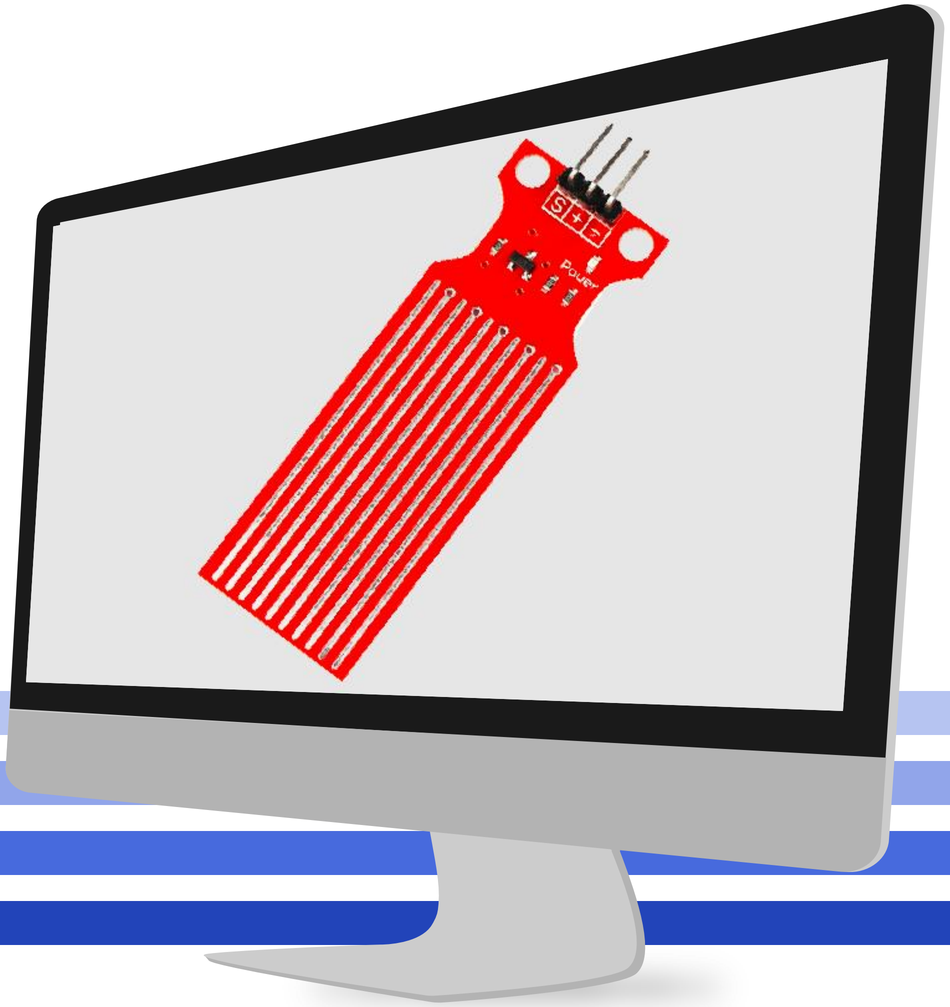
DHT

- DHT11 Temperature & Humidity Sensor features a temperature & humidity sensor complex with a calibrated digital signal output.
- It uses a capacitive humidity sensor and a thermistor to measure the surrounding air and spits out a digital signal on the data pin (no analog input pins needed).
- DHT11's power supply is 3-5.5V DC.
- Good for 20-80% humidity readings with 5% accuracy
- Good for 0-50°C temperature readings ±2°C accuracy
- Operating current: 0.3mA (measuring) 60uA (standby)



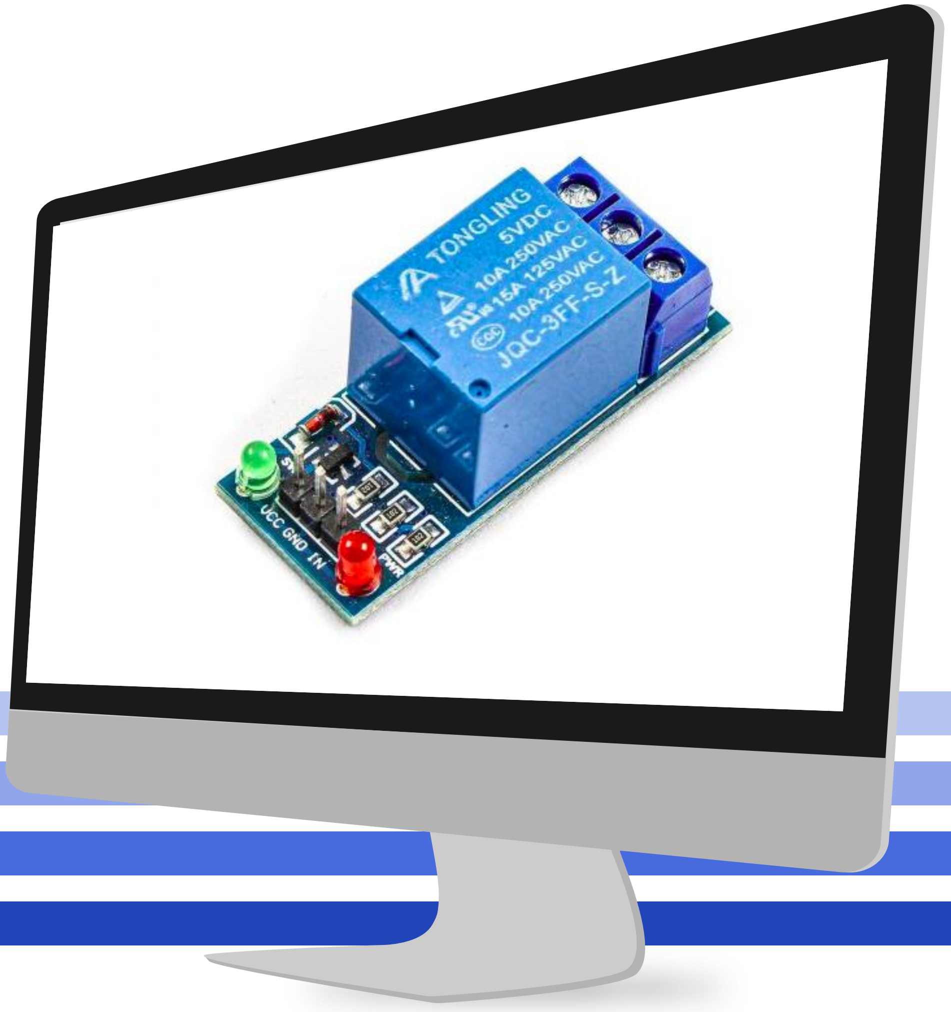
Water Level Sensor

- The sensor can be used to measure the water level, monitor a sump pit, detect rainfall or detect leakage.
- The sensor has a series of ten exposed copper traces, five of which are power traces and five are sense traces. (alternate traces)
- There's a Power LED on the board which will light up when the board is powered.
- The series of exposed parallel conductors, together acts as a variable resistor (just like a potentiometer) whose resistance varies according to the water level. The change in resistance corresponds to the distance from the top of the sensor to the surface of the water and resistance is inversely proportional to the height of the water.
- Its power supply is 3-5.5V DC.
- The water level sensor has 3 pins to connect:- +(VCC),-(GND) and S(analog output .



Single Channel Relay

- Relay is an electro-mechanical device which acts as a switch.
- The Single Channel Relay Module is a **convenient board which can be used to control high voltage, high current load** such as motor, solenoid valves, lamps and AC load.
- It also comes with two LED to indicate the status of relay i.e., the **Power LED** and **Status LED**.
- The Power LED will light up when the module is powered. The Status LED will light up when the relay is activated.
- There are three pins – a Ground pin and a VCC pin to power the module and an input pin IN to control the relay.
- There are three channels of the relay: common (COM), normally closed (NC), and normally open (NO).



Software Requirements

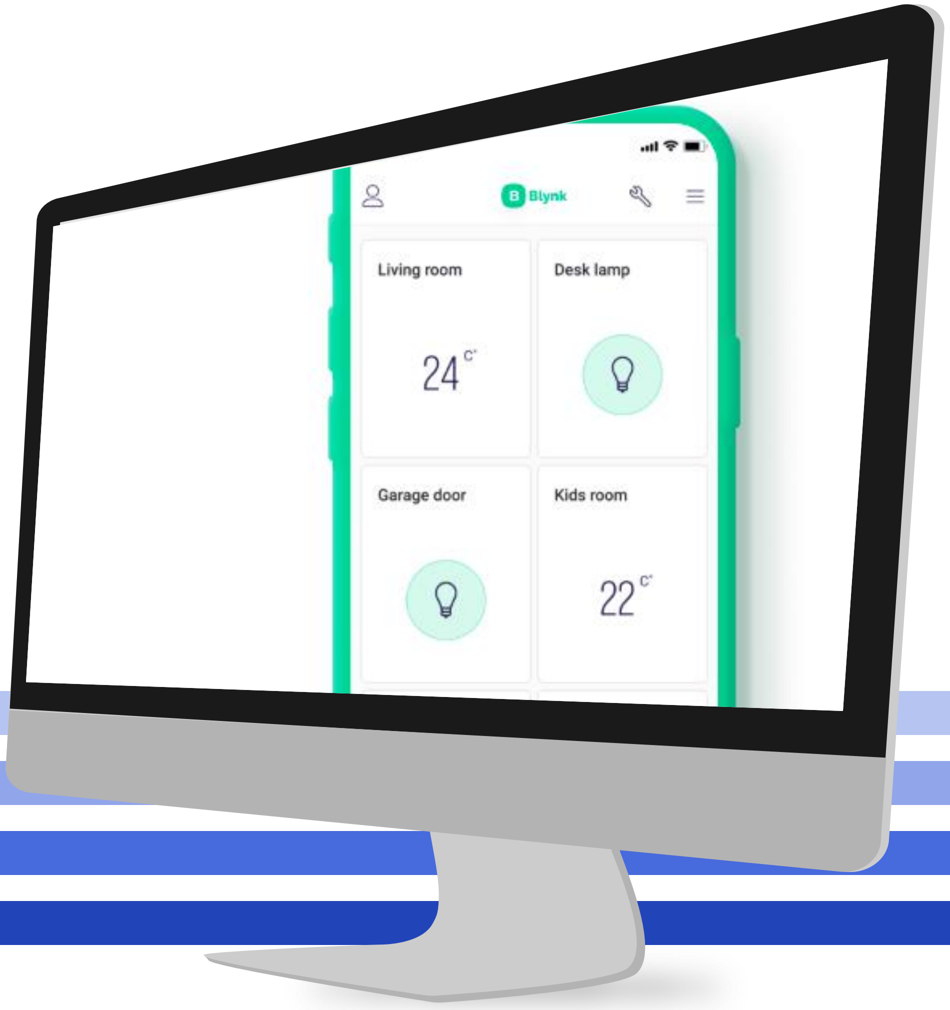
- Blynk Server and Blynk App
- ThingSpeak Integration
- OM2M Integration
- Arduino IDE

Blynk Server

- Blynk Server platform enables you to monitor and control device from anywhere in the world.
- It is responsible for all the communications between the smartphone and hardware.
- You can use our Blynk Cloud or run your private Blynk server locally
- It's open-source, could easily handle thousands of devices
- Blynk application is used to create a graphical interface or human machine interface (HMI) by compiling and providing the appropriate address on the available widgets.

Blynk App

- Blynk app allows you to create amazing interfaces for your projects using various widgets we provide.
- It is a IOT platform to connect your devices to the cloud, design apps to control them. It analyze telemetry data and manage your deployed product at scale and handle easily.
- The features of blynk app is that it has Similar API UI for all supported hardware devices Connection to the cloud using: WIFI, Bluetooth BLE, Ethernet, USB (Serial), GSM and Easy to integrate and add new functionality using virtual pins.
- Now imagine: every time you press a Button in the Blynk app, the message travels to space the Blynk Cloud, where it magically finds its way to your hardware.
- It works the same in the opposite direction and everything happens in a blynk of an eye.



Arduino IDE

- The Arduino Integrated Development Environment is a cross platform application that is written in function from C and C++.
- It is used to write and upload programs to Arduino compatible boards programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .ino .
- The editor has features for cutting/pasting and for searching/replacing text. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.
- Arduino IDE is a derivative of the Processing IDE, however as of version 2.0, the Processing IDE will be replaced with the Visual Studio Code-based Eclipse Theia IDE framework.

OM2M

- oneM2M was launched in 2012.
- It is a global initiative that develops specifications to ensure the most efficient deployment of **Machine-to-Machine** (M2M) communications systems and the Internet of Things (IoT).
- It **allows developers to mix and match components from different vendors**.
- In machine-to-machine communications, a **remote sensor gathers data and sends it wirelessly to a network, where it's next routed, often through the Internet, to a server such** as a personal computer.
- At that point, the data is analyzed and acted upon, according to the software in place.

- The **OM2M** project is an open-source implementation of the SmartM2M standard.
- It provides a framework for developing services independently of the underlying network and aims to facilitate deployment of vertical applications and heterogeneous devices.

ThingSpeak

- ThingSpeak is an IoT analytics platform service that allows you to aggregate, visualize, and analyze live data streams in the cloud.
- You can send data to ThingSpeak from your devices, create instant visualization of live data, and send alerts.
- Once you send data to ThingSpeak from your devices, you can create instant visualizations of live data without having to write any code.
- With the ability to execute MATLAB® code in ThingSpeak you can perform online analysis and processing of the data as it comes in. ThingSpeak is often used for prototyping and proof of concept IoT systems that require analytics.
- The Integration with The Things Network allows you to seamlessly forward data from The Things Network to ThingSpeak for analysis and visualization

Methodology

- ***Technique to automate hand sanitizer dispenser.***

When we place our hand in front of ultrasonic sensor, ESP32 will read the signals from these sensors and if hands are in a desired range, it will trigger and then it will turn on the DC motor after receiving signal from ESP32 and people will be able to sanitize their hands.

- ***Technique to get Email & mobile notification reminder for refilling sanitizer bottle via Blynk Server and app.***

So ESP32 is interfacing with WIFI Module. Using blynk app we can set an alert for mobile notification for refiling sanitizer bottle as soon as the fluid level reaches minimum value.

- ***Technique for Fluid Level Detection and Log record of people using Dispenser.***

So ESP32 is interfacing with WIFI Module. using blynk app we can set a value i.e. suppose at this
count the sanitizer fluid will be at the empty point and need to refill it. So, as people starts sanitizing their hands and a s the count of no of people sanitizing their hands increases, fluid level indicator starts decreasing. People can live monitor the amount of fluid present in the sanitizer bottle, the daily count of people sanitizing their hands and log record of people used sanitizer after every refill with the help of blynk server on mobile app.

- ***Technique for Temperature and Humidity readings on ThingSpeak as well as on Blynk***

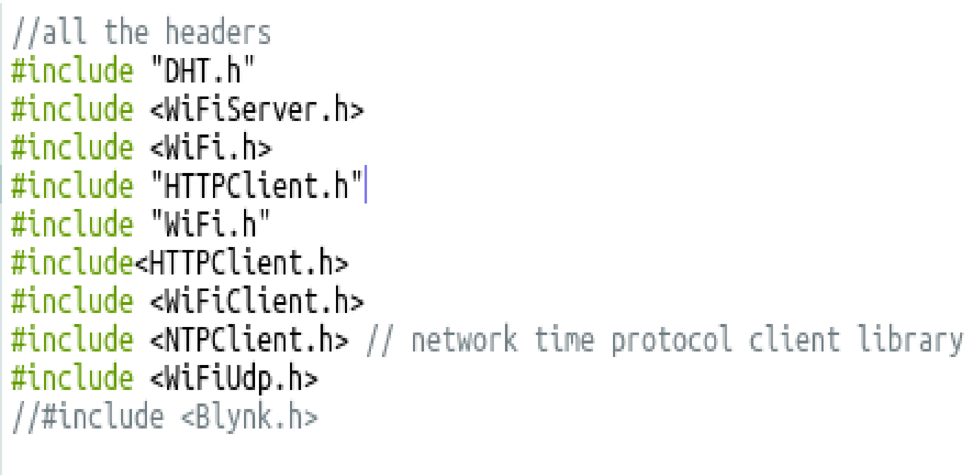
So, ESP32 is interfacing with WIFI module. Using ThingSpeak API key and server client, current temperature and humidity readings are being updated on ThingSpeak every few seconds. The same readings can also be accessed via blynk server on mobile apps.

- **Technique for OM2M Integration**

For doing OM2M integration, we have created three application entities namely DHT, Uses and Sanitizer_level. In these, DHT have two containers namely Temperature and Humidity, Uses also have two containers namely Daily_Count as well as Use_After_Refill. The third application entity is for Sanitizer_level which has only one container I.e., Current_level. They all are initiated with an empty string and a new instance is added to every container with their respective data field each time the sanitizer is ejected.

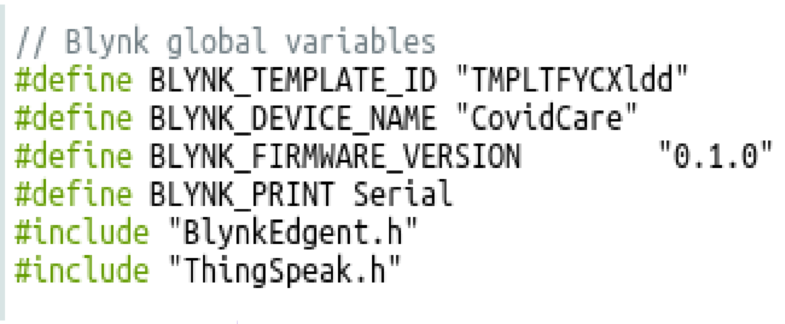
CODE

- Headers



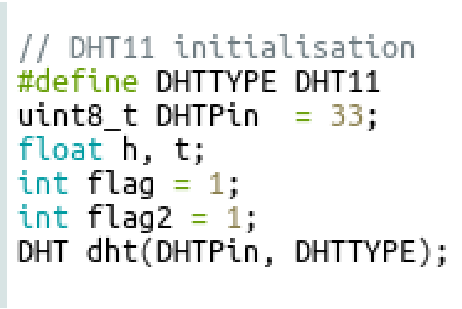
```
//all the headers
#include "DHT.h"
#include <WiFiServer.h>
#include <WiFi.h>
#include "HTTPClient.h"
#include "WiFi.h"
#include<HTTPClient.h>
#include <WiFiClient.h>
#include <NTPClient.h> // network time protocol client library
#include <WiFiUdp.h>
//#include <Blynk.h>
```

- Defining Blynk Variables



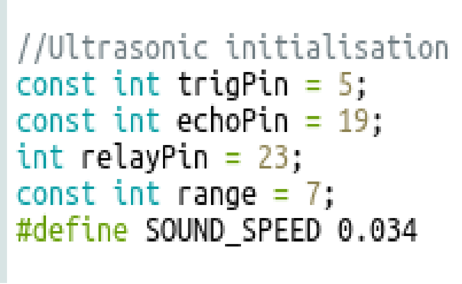
```
// Blynk global variables
#define BLYNK_TEMPLATE_ID "TMPLTFYCXldd"
#define BLYNK_DEVICE_NAME "CovidCare"
#define BLYNK_FIRMWARE_VERSION        "0.1.0"
#define BLYNK_PRINT Serial
#include "BlynkEdgent.h"
#include "ThingSpeak.h"
```

- DHT11 initialization



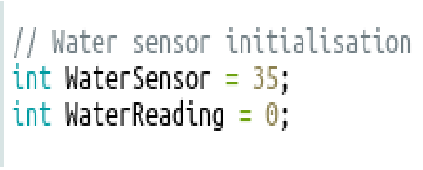
```
// DHT11 initialisation
#define DHTTYPE DHT11
uint8_t DHTPin  = 33;
float h, t;
int flag = 1;
int flag2 = 1;
DHT dht(DHTPin, DHTTYPE);
```

- Ultrasonic initialization



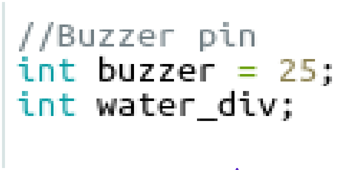
```
//Ultrasonic initialisation
const int trigPin = 5;
const int echoPin = 19;
int relayPin = 23;
const int range = 7;
#define SOUND_SPEED 0.034
```

- Water sensor initialization



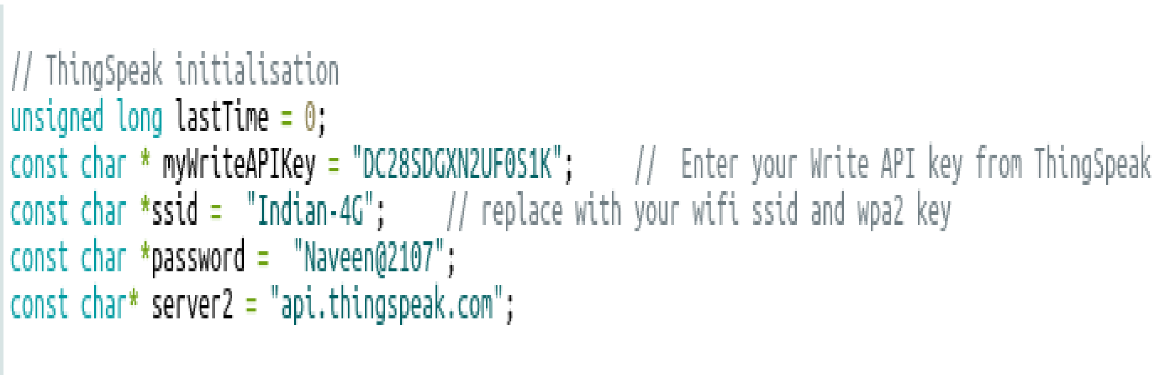
```
// Water sensor initialisation
int WaterSensor = 35;
int WaterReading = 0;
```

- Buzzer pin



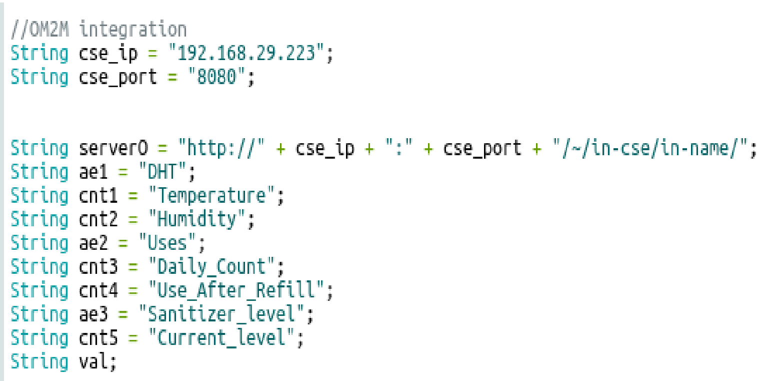
```
//Buzzer pin
int buzzer = 25;
int water_div;
```

- ThingSpeak initialization



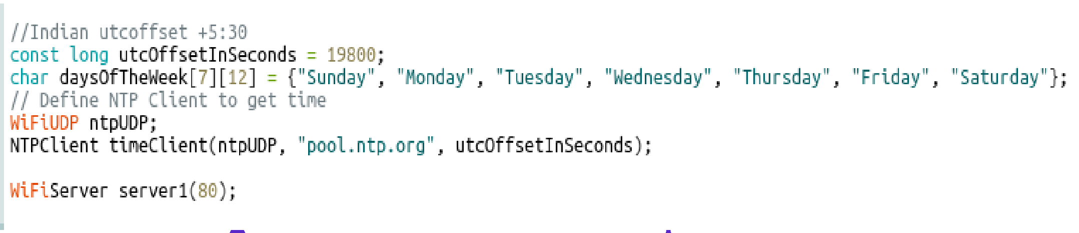
```
// ThingSpeak initialisation
unsigned long lastTime = 0;
const char * myWriteAPIKey = "DC28SDGXN2UF0S1K";    //  Enter your Write API key from ThingSpeak
const char *ssid =  "Indian-4G";    // replace with your wifi ssid and wpa2 key
const char *password =  "Naveen@2107";
const char* server2 = "api.thingspeak.com";
```

- OM2M integration



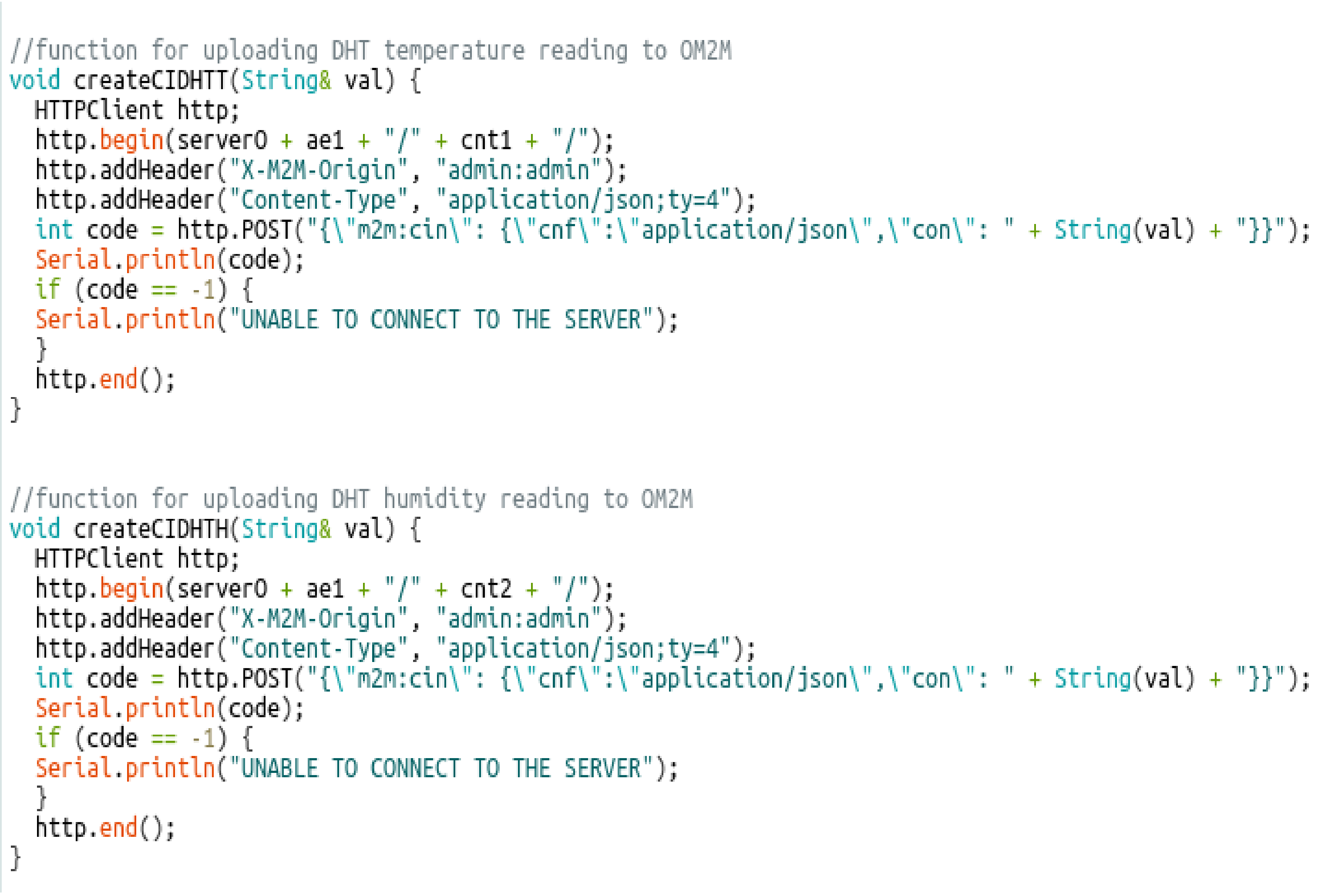
```
//OM2M integration
String cse_ip = "192.168.29.223";
String cse_port = "8080";


String server0 = "http://" + cse_ip + ":" + cse_port + "/~/in-cse/in-name/";
String ae1 = "DHT";
String cnt1 = "Temperature";
String cnt2 = "Humidity";
String ae2 = "Uses";
String cnt3 = "Daily_Count";
String cnt4 = "Use_After_Refill";
String ae3 = "Sanitizer_level";
String cnt5 = "Current_level";
String val;
```

- Indian utcoffset (Converting the time to IST)



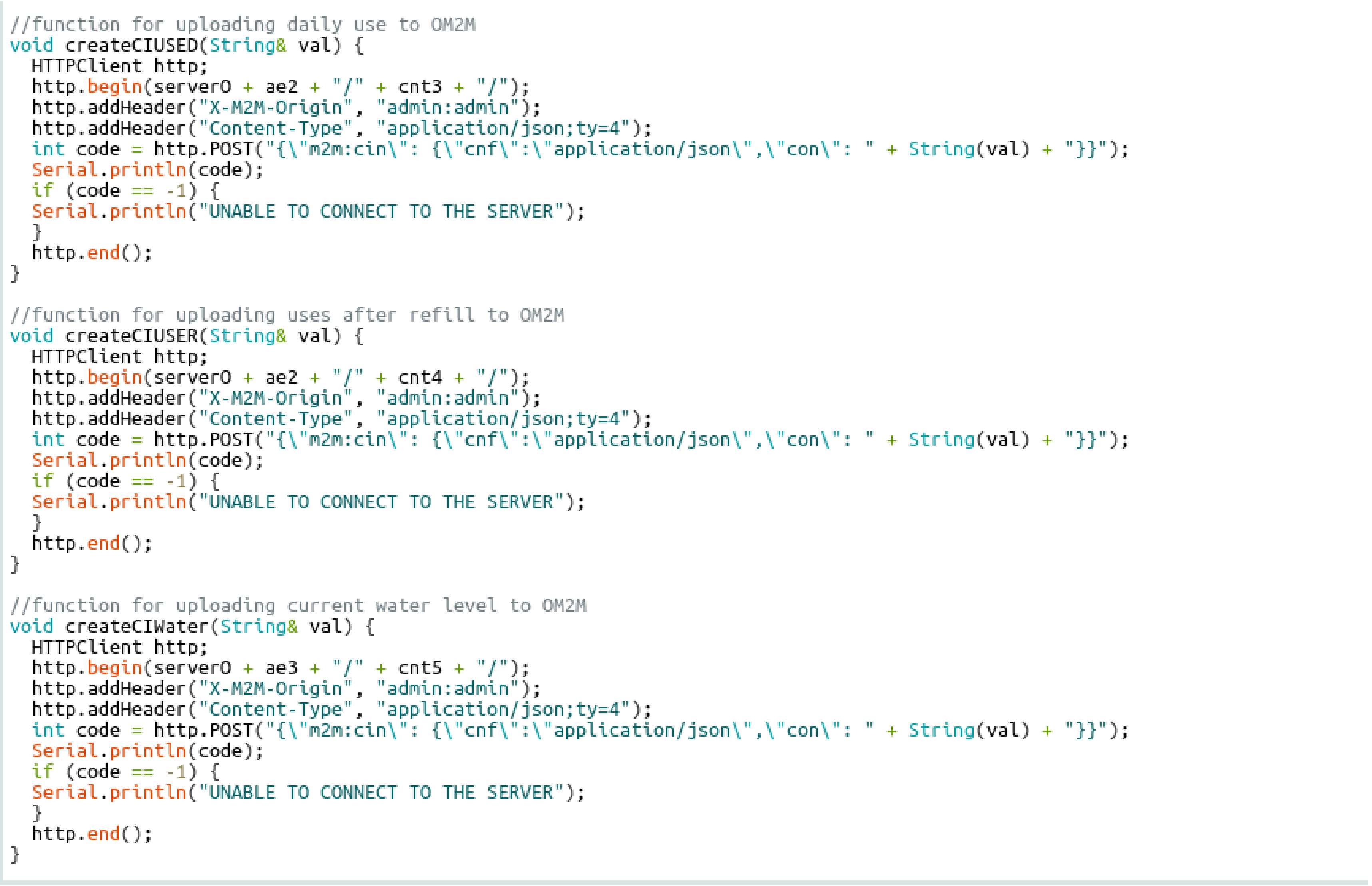
```
//Indian utcoffset +5:30
const long utcOffsetInSeconds = 19800;
char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};
// Define NTP Client to get time
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "pool.ntp.org", utcOffsetInSeconds);

WiFiServer server1(80);
```

- Function for uploading sensor data of OM2M



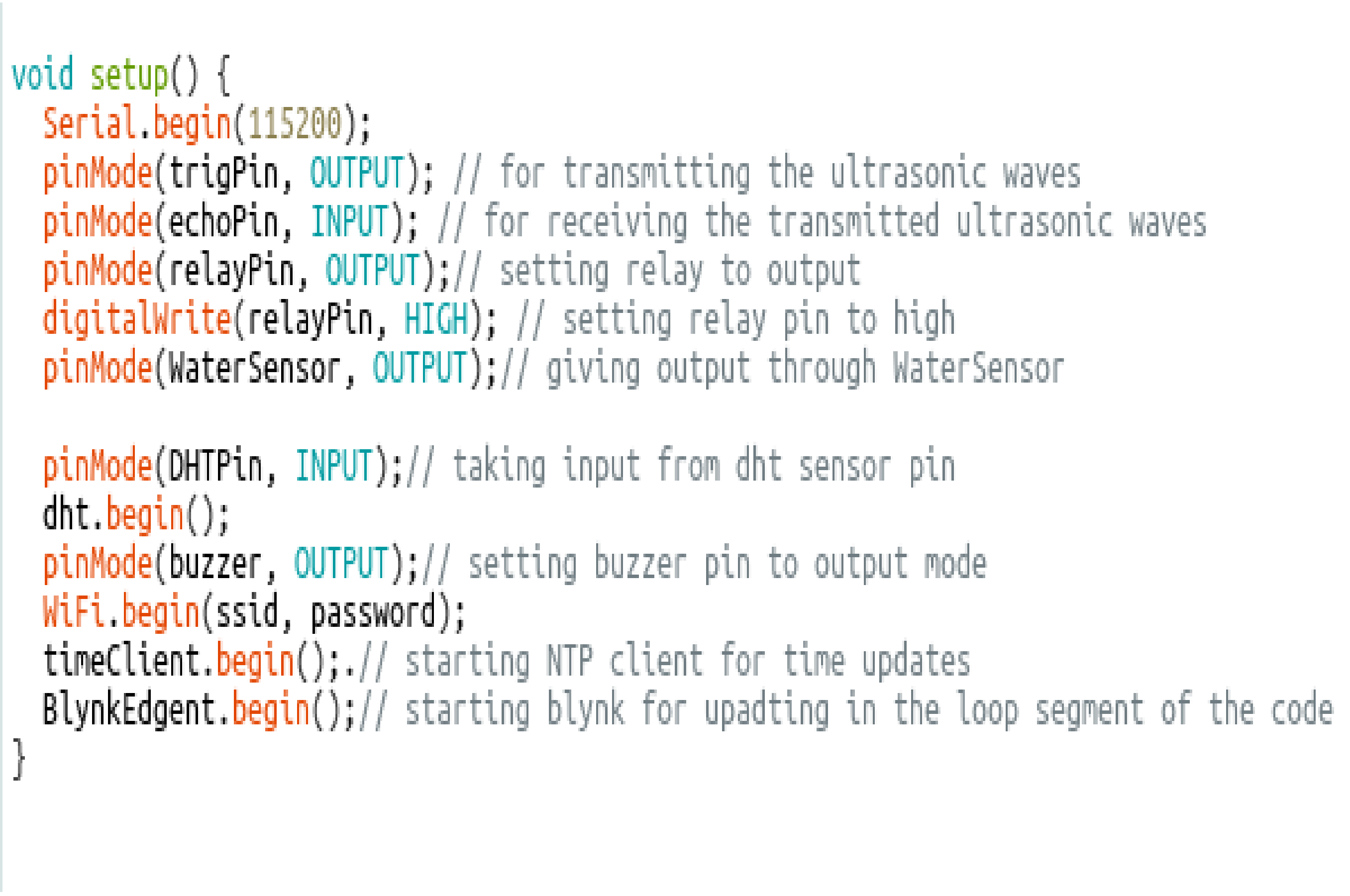
```
//function for uploading DHT temperature reading to OM2M
void createCIDHTT(String& val) {
  HTTPClient http;
  http.begin(server0 + ae1 + "/" + cnt1 + "/");
  http.addHeader("X-M2M-Origin", "admin:admin");
  http.addHeader("Content-Type", "application/json;ty=4");
  int code = http.POST("{\"m2m:cin\": {\"cnf\":\"application/json\",\"con\": " + String(val) + "}}");
  Serial.println(code);
  if (code == -1) {
  Serial.println("UNABLE TO CONNECT TO THE SERVER");
  }
  http.end();
}


//function for uploading DHT humidity reading to OM2M
void createCIDHTH(String& val) {
  HTTPClient http;
  http.begin(server0 + ae1 + "/" + cnt2 + "/");
  http.addHeader("X-M2M-Origin", "admin:admin");
  http.addHeader("Content-Type", "application/json;ty=4");
  int code = http.POST("{\"m2m:cin\": {\"cnf\":\"application/json\",\"con\": " + String(val) + "}}");
  Serial.println(code);
  if (code == -1) {
  Serial.println("UNABLE TO CONNECT TO THE SERVER");
  }
  http.end();
}
```

- Function for uploading sensor data of OM2M



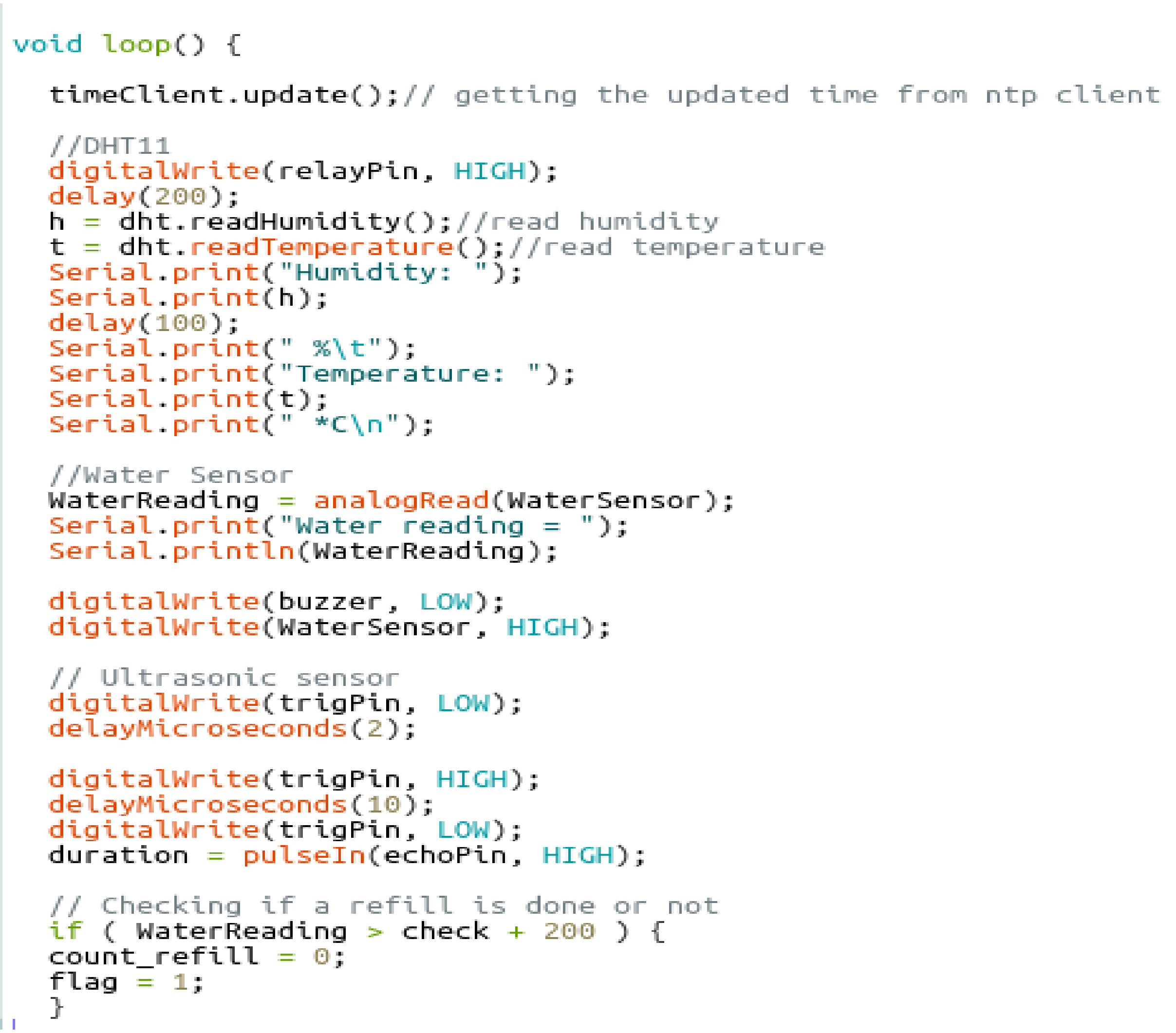
```
//function for uploading daily use to OM2M
void createCIUSED(String& val) {
  HTTPClient http;
  http.begin(server0 + ae2 + "/" + cnt3 + "/");
  http.addHeader("X-M2M-Origin", "admin:admin");
  http.addHeader("Content-Type", "application/json;ty=4");
  int code = http.POST("{\"m2m:cin\": {\"cnf\":\"application/json\",\"con\": " + String(val) + "}}");
  Serial.println(code);
  if (code == -1) {
  Serial.println("UNABLE TO CONNECT TO THE SERVER");
  }
  http.end();
}

//function for uploading uses after refill to OM2M
void createCIUSER(String& val) {
  HTTPClient http;
  http.begin(server0 + ae2 + "/" + cnt4 + "/");
  http.addHeader("X-M2M-Origin", "admin:admin");
  http.addHeader("Content-Type", "application/json;ty=4");
  int code = http.POST("{\"m2m:cin\": {\"cnf\":\"application/json\",\"con\": " + String(val) + "}}");
  Serial.println(code);
  if (code == -1) {
  Serial.println("UNABLE TO CONNECT TO THE SERVER");
  }
  http.end();
}

//function for uploading current water level to OM2M
void createCIWater(String& val) {
  HTTPClient http;
  http.begin(server0 + ae3 + "/" + cnt5 + "/");
  http.addHeader("X-M2M-Origin", "admin:admin");
  http.addHeader("Content-Type", "application/json;ty=4");
  int code = http.POST("{\"m2m:cin\": {\"cnf\":\"application/json\",\"con\": " + String(val) + "}}");
  Serial.println(code);
  if (code == -1) {
  Serial.println("UNABLE TO CONNECT TO THE SERVER");
  }
  http.end();
}
```

- Configurize pins into Input and Output



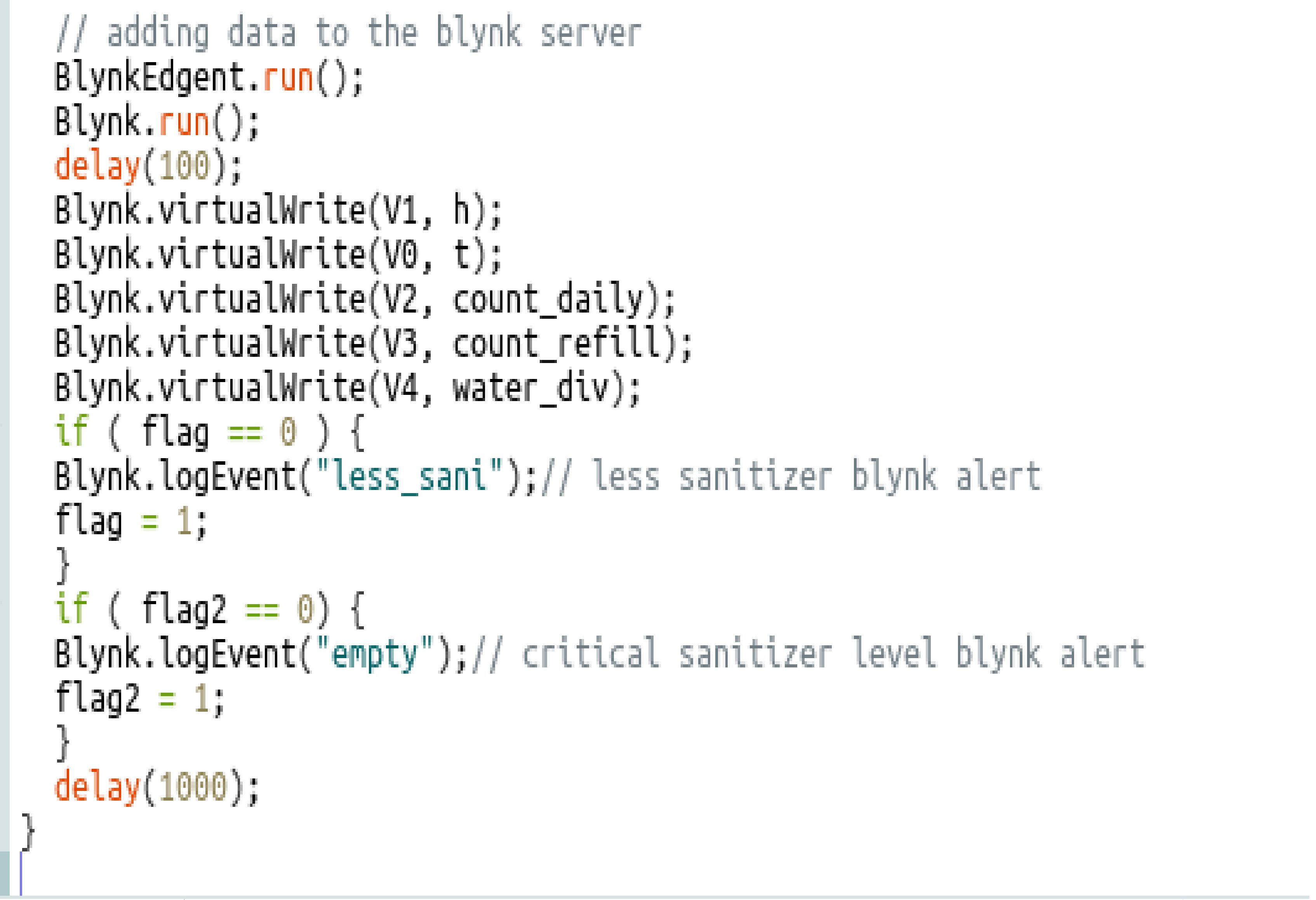
```
void setup() {
  Serial.begin(115200);
  pinMode(trigPin, OUTPUT); // for transmitting the ultrasonic waves
  pinMode(echoPin, INPUT); // for receiving the transmitted ultrasonic waves
  pinMode(relayPin, OUTPUT);// setting relay to output
  digitalWrite(relayPin, HIGH); // setting relay pin to high
  pinMode(WaterSensor, OUTPUT);// giving output through WaterSensor

  pinMode(DHTPin, INPUT);// taking input from dht sensor pin
  dht.begin();
  pinMode(buzzer, OUTPUT);// setting buzzer pin to output mode
  WiFi.begin(ssid, password);
  timeClient.begin();.// starting NTP client for time updates
  BlynkEdgent.begin();// starting blynk for upadting in the loop segment of the code
}
```

- Conditions



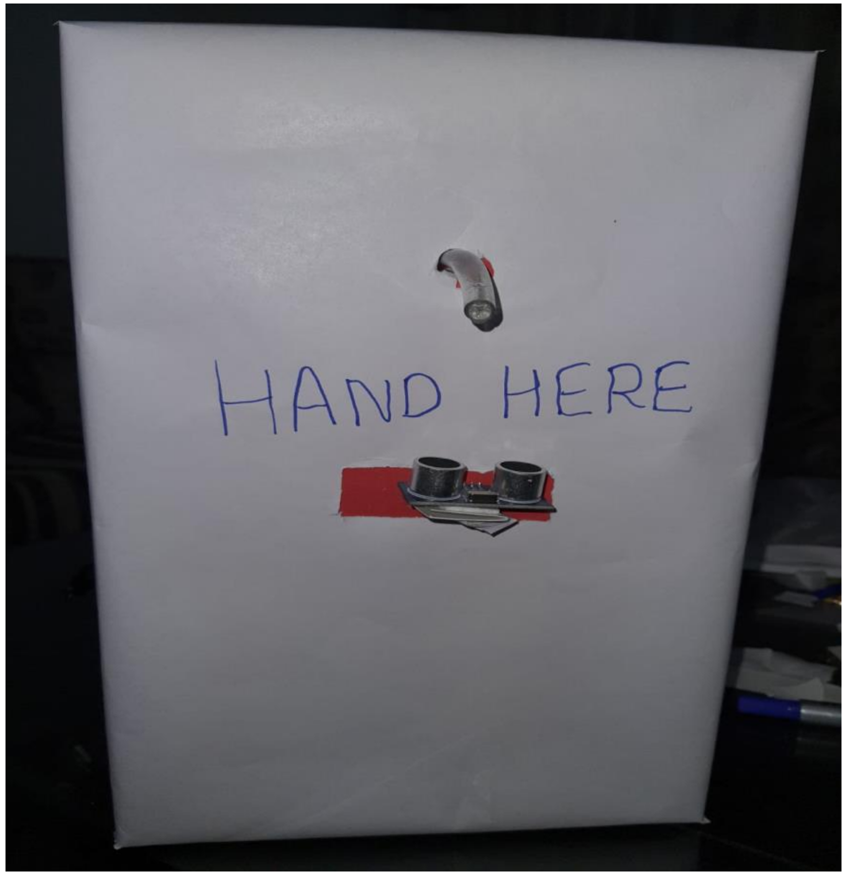
```
void loop() {

  timeClient.update();// getting the updated time from ntp client

  //DHT11
  digitalWrite(relayPin, HIGH);
  delay(200);
  h = dht.readHumidity();//read humidity
  t = dht.readTemperature();//read temperature
  Serial.print("Humidity: ");
  Serial.print(h);
  delay(100);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print(" *C\n");

  //Water Sensor
  WaterReading = analogRead(WaterSensor);
  Serial.print("Water reading = ");
  Serial.println(WaterReading);

  digitalWrite(buzzer, LOW);
  digitalWrite(WaterSensor, HIGH);

  // Ultrasonic sensor
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);

  // Checking if a refill is done or not
  if ( WaterReading > check + 200 ) {
  count_refill = 0;
  flag = 1;
  }
```

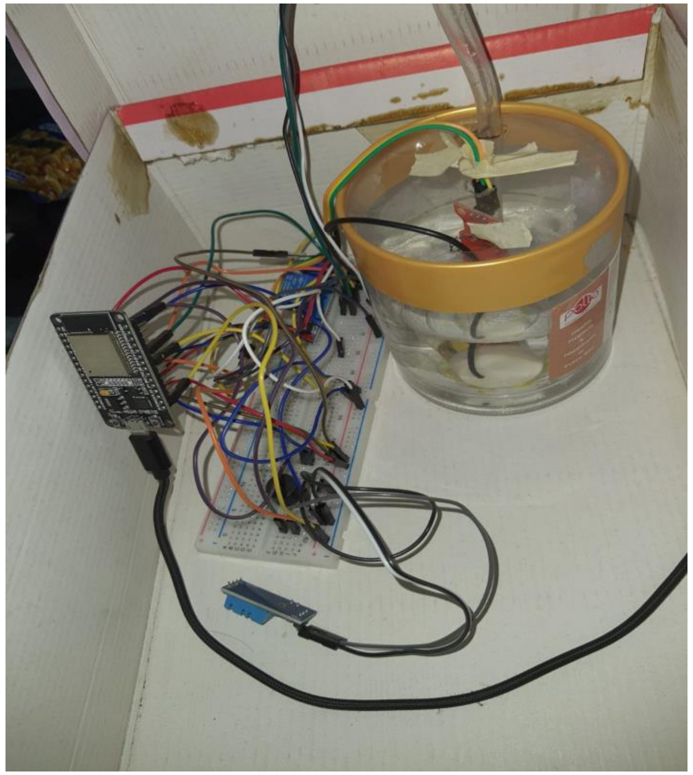
```
// calculating the distance using the formula s=vt.
distanceCm = duration * SOUND_SPEED / 2;

//getting the water reading on a scale of 0-100
water_div = WaterReading / 28;

//thingspeak field setup and uploading the data
ThingSpeak.begin(client);
ThingSpeak.setField(1, t);
ThingSpeak.setField(2, h);
ThingSpeak.setField(3, WaterReading);
ThingSpeak.setField(4, distanceCm);
int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
Serial.print("Distance (cm): ");
Serial.println(distanceCm);

//getting time from NTP client
Serial.print("Time = ")
Serial.print(timeClient.getHours());
Serial.print(" : ");
Serial.println(timeClient.getMinutes());

//Setting daily count to 0 when it clocks 12:00
if ( (timeClient.getHours()) == 0 && (timeClient.getMinutes()) < 1 )
{
count_daily = 0;
}

// proximity check as well as water level check if present eject sanitizer
if ( distanceCm < range && water_div > 0 )
{
digitalWrite(relayPin, LOW);
delay(250);
digitalWrite(relayPin, HIGH);
count_daily = count_daily + 1;
count_refill = count_refill + 1;
// uploading data on OM2M server using the above defined functions.
val = t;
createCIDHTT(val);
val = h;
createCIDHTH(val);
val = count_daily;
delay(20);
createCIUSED(val);
val = count_refill;
createCIUSER(val);
val = water_div;
createCIWater(val);
check = WaterReading;

//indicating low water level
if ( WaterReading < 700 ) {
  digitalWrite(buzzer, HIGH);
  delay(300);// smaller beep indicating low level of sanitizer
  flag = 0;// setting flag for blynk alert for low level
  digitalWrite(buzzer, LOW);
}
delay(8000); // delay for preventing next use till 8 seconds
}
else {
if ( distanceCm < range ) {
  digitalWrite(buzzer, HIGH);
  delay(600);// longer beep indicating critically low levels of sanitizer
  digitalWrite(buzzer, LOW);
  flag2 = 0;// setting flag for blynk alert for critical level alert
}
}
```



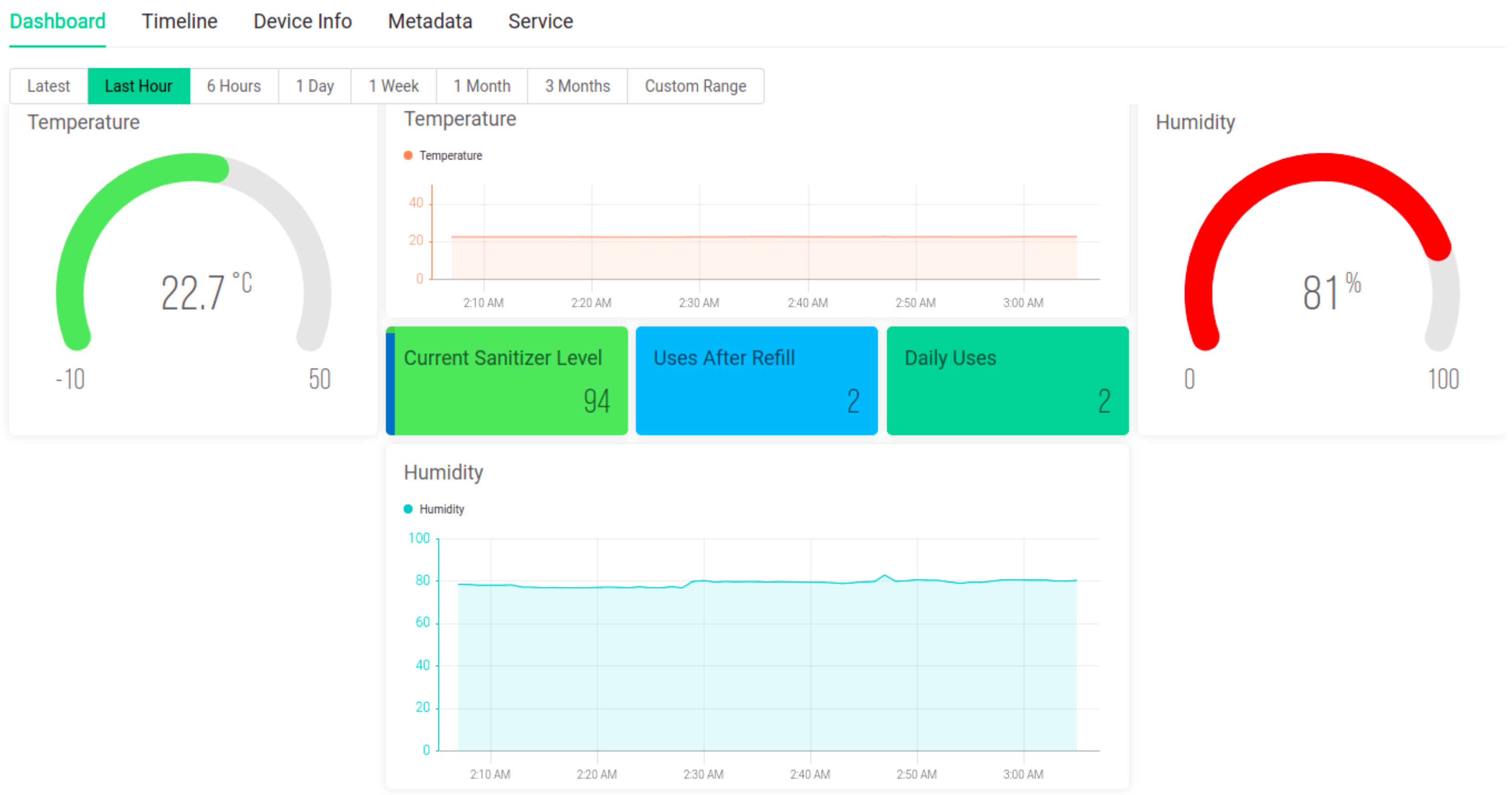
```
  // adding data to the blynk server
  BlynkEdgent.run();
  Blynk.run();
  delay(100);
  Blynk.virtualWrite(V1, h);
  Blynk.virtualWrite(V0, t);
  Blynk.virtualWrite(V2, count_daily);
  Blynk.virtualWrite(V3, count_refill);
  Blynk.virtualWrite(V4, water_div);
  if ( flag == 0 ) {
  Blynk.logEvent("less_sani");// less sanitizer blynk alert
  flag = 1;
  }
  if ( flag2 == 0) {
  Blynk.logEvent("empty");// critical sanitizer level blynk alert
  flag2 = 1;
  }
  delay(1000);
}
```

Hardware Design

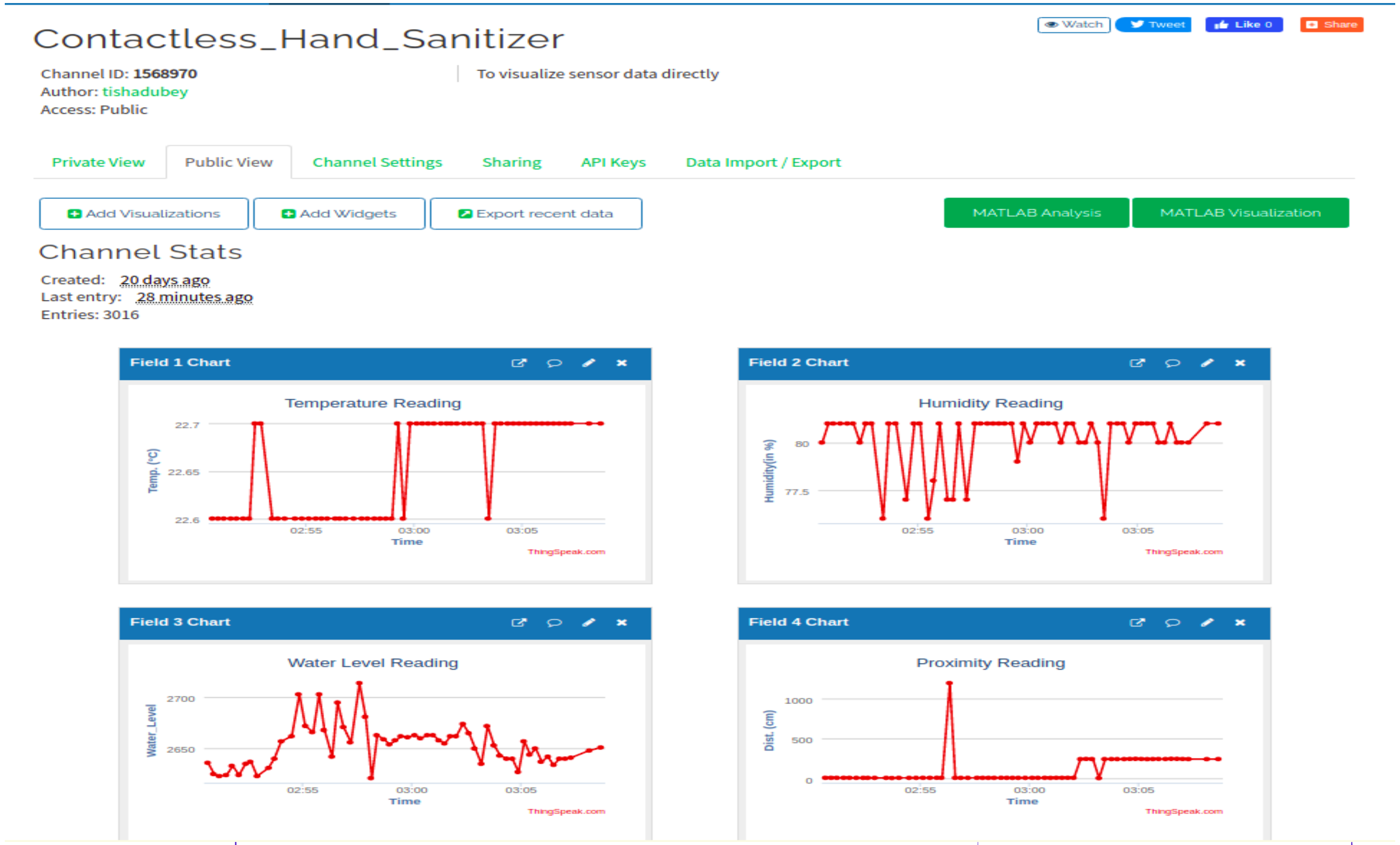




User Interface - Blynk



Analytics - ThingSpeak



Contactless_Hand_Sanitizer

Channel ID: **1568970**
Author: tishadubey
Access: Public

To visualize sensor data directly

Private View | Public View | Channel Settings | Sharing | API Keys | Data Import / Export

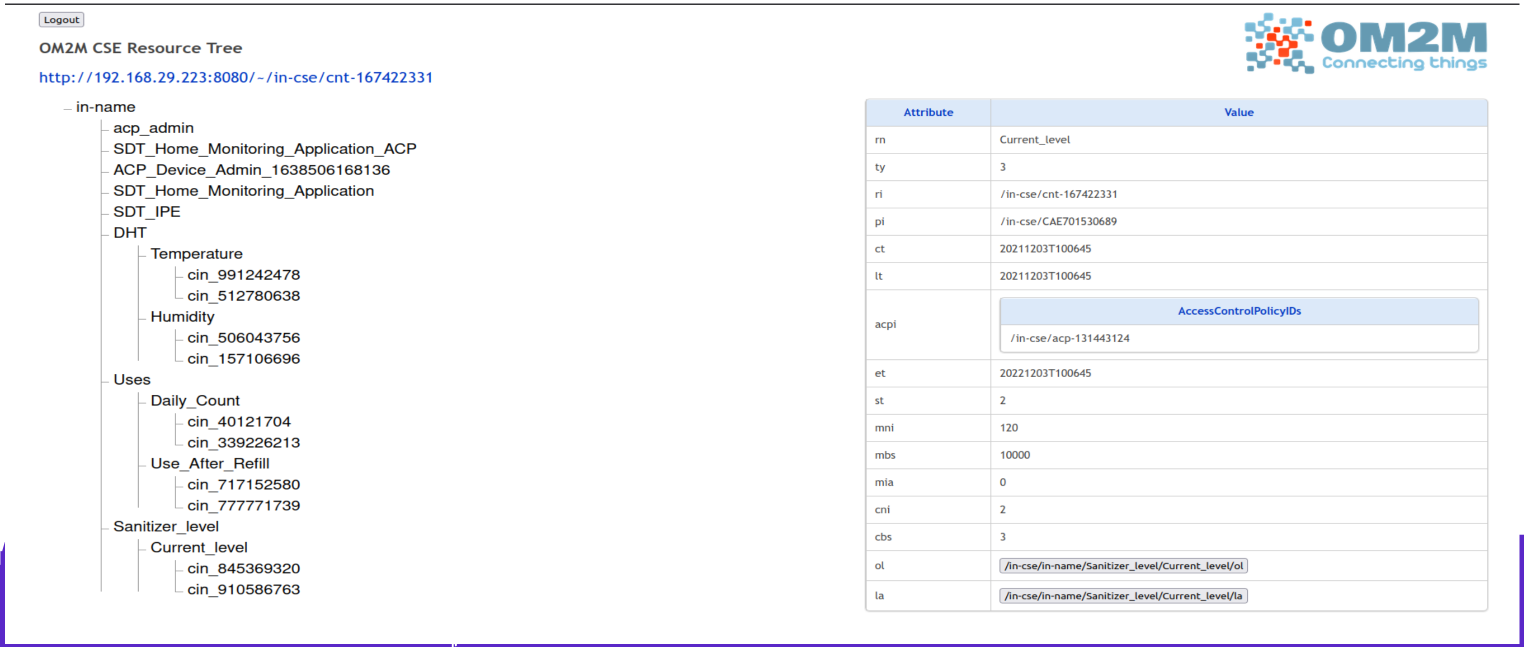
Channel Stats

Created: 20 days ago
Last entry: 28 minutes ago
Entries: 3016

Working Model

RESULTS

As soon as the ESP32 gets connects and ultrasonic gets triggered, readings from various sensors starts updating on ThingSpeak as well as on Blynk server and app.
The readings stored inside the containers as a content instance in their respective AE's(application entity) in which AE are the DHT, Uses and Sanitizer_level. The result is the real-time values of the parameters like temperature, humidity, daily uses, uses after refill and current sanitizer level are being updated in the OM2M as a content instance in the respective containers forming a regular onem2m resource tree.


Logout

OM2M CSE Resource Tree

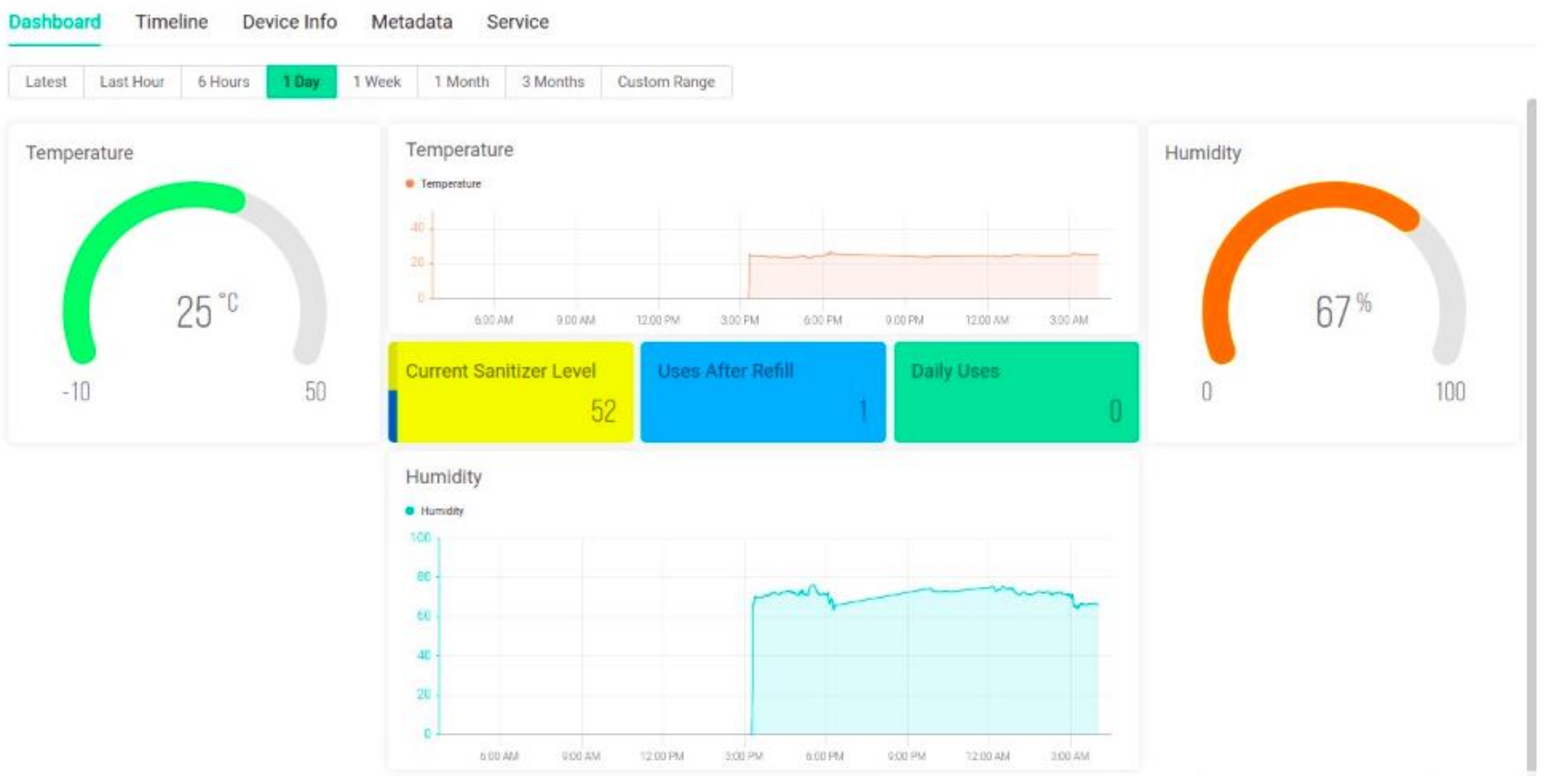
http://192.168.29.223:8080/~/in-cse/cnt-167422331

```
in-name
    acp_admin
    SDT_Home_Monitoring_Application_ACP
    ACP_Device_Admin_1638506168136
    SDT_Home_Monitoring_Application
    SDT_IPE
    DHT
        Temperature
            cin_991242478
            cin_512780638
        Humidity
            cin_506043756
            cin_157106696
    Uses
        Daily_Count
            cin_40121704
            cin_339226213
        Use_After_Refill
            cin_717152580
            cin_777771739
    Sanitizer_level
        Current_level
            cin_845369320
            cin_910586763
```

OM2M Connecting things

Attribute	Value
rn	Current_level
ty	3
ri	/in-cse/cnt-167422331
pi	/in-cse/CAE701530689
ct	20211203T100645
lt	20211203T100645
acpi	AccessControlPolicyIDs: /in-cse/acp-131443124
et	20221203T100645
st	2
mni	120
mbs	10000
mia	0
cni	2
cbs	3
ol	/in-cse/in-name/Sanitizer_level/Current_level/ol
la	/in-cse/in-name/Sanitizer_level/Current_level/la

As the soon as the day ends the daily count refill sets to 0 again at 00:00.



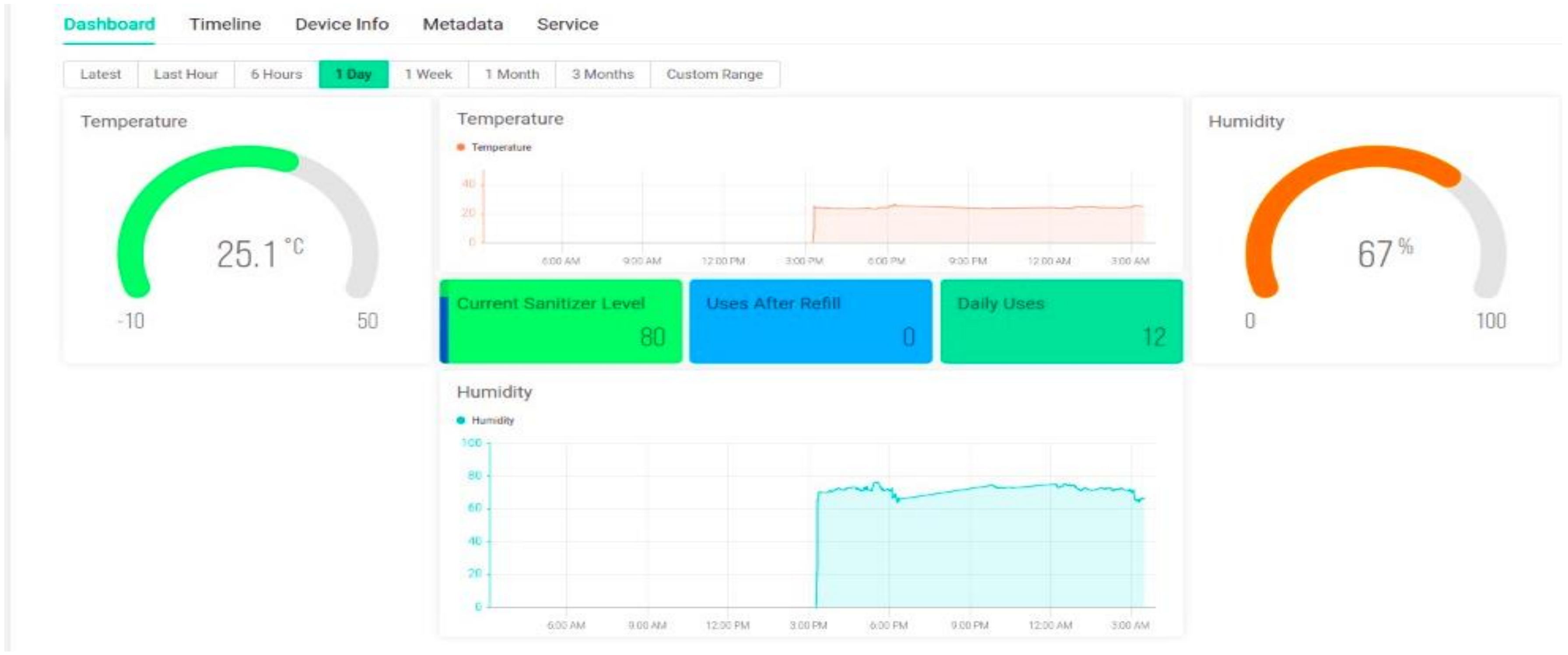
Current Sanitizer Level 52

Uses After Refill 1

Daily Uses 0

In this, the daily count resets to 0 between 00:00 to 00:01

After every refill, the uses after refill (i.e., count_refill) gets updated and sets to 0 so that we can get an count as how many people have used the sanitizer in a particular refill.

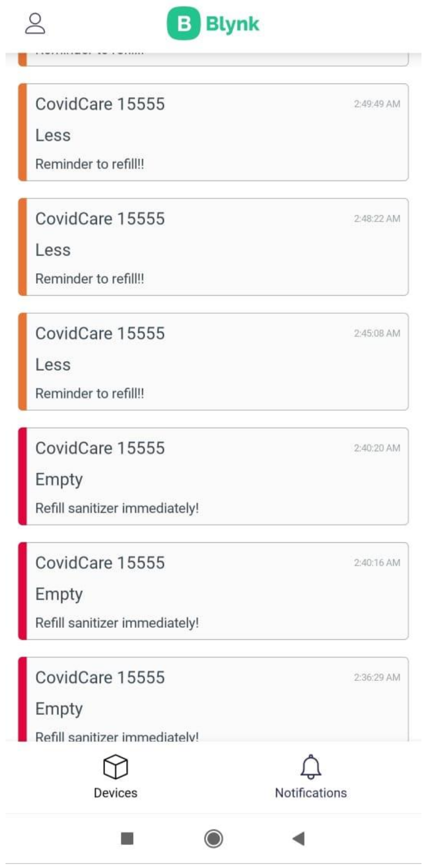


Current Sanitizer Level 80

Uses After Refill 0

Daily Uses 12

Whenever the fluid level reading reaches below a certain level, we receives push notification as well as email notification reminding us to refill the sanitizer. Also, when there is no more fluid left in tank, we receive an critical alert warning as a push notification to refill sanitizer immediately.



CovidCare 15555 2:49:49 AM
Less
Reminder to refill!!

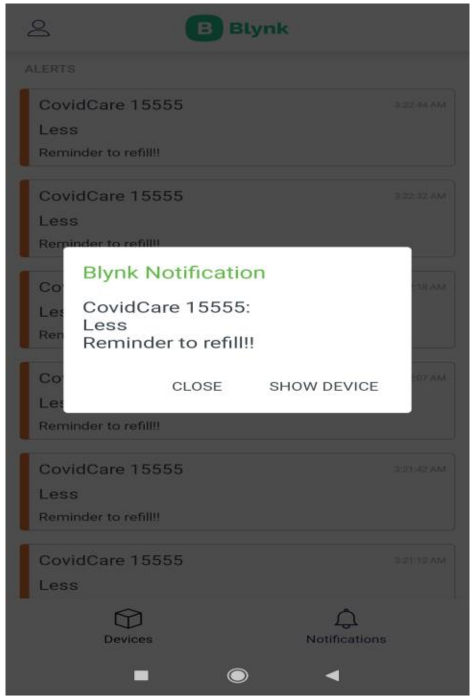
CovidCare 15555 2:48:22 AM
Less
Reminder to refill!!

CovidCare 15555 2:45:08 AM
Less
Reminder to refill!!

CovidCare 15555 2:40:20 AM
Empty
Refill sanitizer immediately!

CovidCare 15555 2:40:16 AM
Empty
Refill sanitizer immediately!

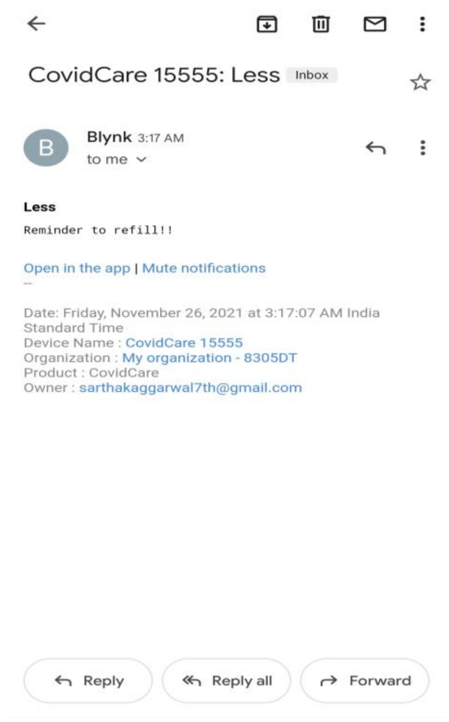
CovidCare 15555 2:36:29 AM
Empty
Refill sanitizer immediately!



Blynk Notification

CovidCare 15555:
Less
Reminder to refill!!

Push notification via blynk app



CovidCare 15555: Less Inbox

Blynk 3:17 AM
to me

Less
Reminder to refill!!

Open in the app | Mute notifications

Date: Friday, November 26, 2021 at 3:17:07 AM India Standard Time
Device Name : CovidCare 15555
Organization : My organization - 8305DT
Product : CovidCare
Owner : sarthakaggarwal7th@gmail.com

Email alert

Team - **JATS**

Sarthak Aggarwal
2020101008

Hardware implementation and code

Tisha Dubey
2020101101

Online web interface, (ThingSpeak and Blynk), code, and assistance

Aarushi Mittal
2020111008

Assistance in code writing for sensors

Janardan
2020111005

Coding part for OM2M and added the two parameters(count_daily and count_refill)

THANK YOU