Smart Kitchen using IBM Cloud

Introduction:

Overview:

In the past years we are continuosly monitering the grocerie levels if the quantity of groceries is less then we have to buy at store. Now a days the people are busy with a lot of works and the persons dont have any time to monitor the grocerie levels and they forgot to buy for the people this project is very usefull. The amount of groceries in the jars and lpg gas levels in the kitchen can be visualized through the mobile application and web application. The decreasing levels in the jars will be sensed by ultrasonic sensors and orders will be placed, if lpg weight goes below some threshold value an message alert will be sent to the user. The controlling devices of the system are Python Code with IBM CLOUD and Sensors. The data updated in cloud will be visualized in android application developed in MIT App Inventer and web page developed in Node-Red Service. And LPG is one of the clean fuel. It is most widely being used in India. In case of household use it is mostly used in kitchen. In the past years the demand for use of LPG have increased sustainably and will continue to rise. But with the increase in demand of use of LPG the rate of accidents caused due to it have also increased in the past few years. The majority of the accidents are due to the explosion of LPG cylinder. But, sometimes very small quantity of gas leakage is unnoticed and is responsible for the major accident. This paper discusses the solution to it.

Purpose:

By this project there is no need to monitor the grocerie Jar levels, LPG level and Gas leakage Detection and this project helps us when the grocerie jar levels below the throushold levels the the sensors sends the data to the android application as well as website and also we will get an SMS notification and it detects the gas leakage at the Gas cylinder if any gas is detected by the sensor then we will get a notification like "Gas Leakage Detected" so we can get an alert.

LITERATURE SURVEY:

EXISTING PROBLEM:

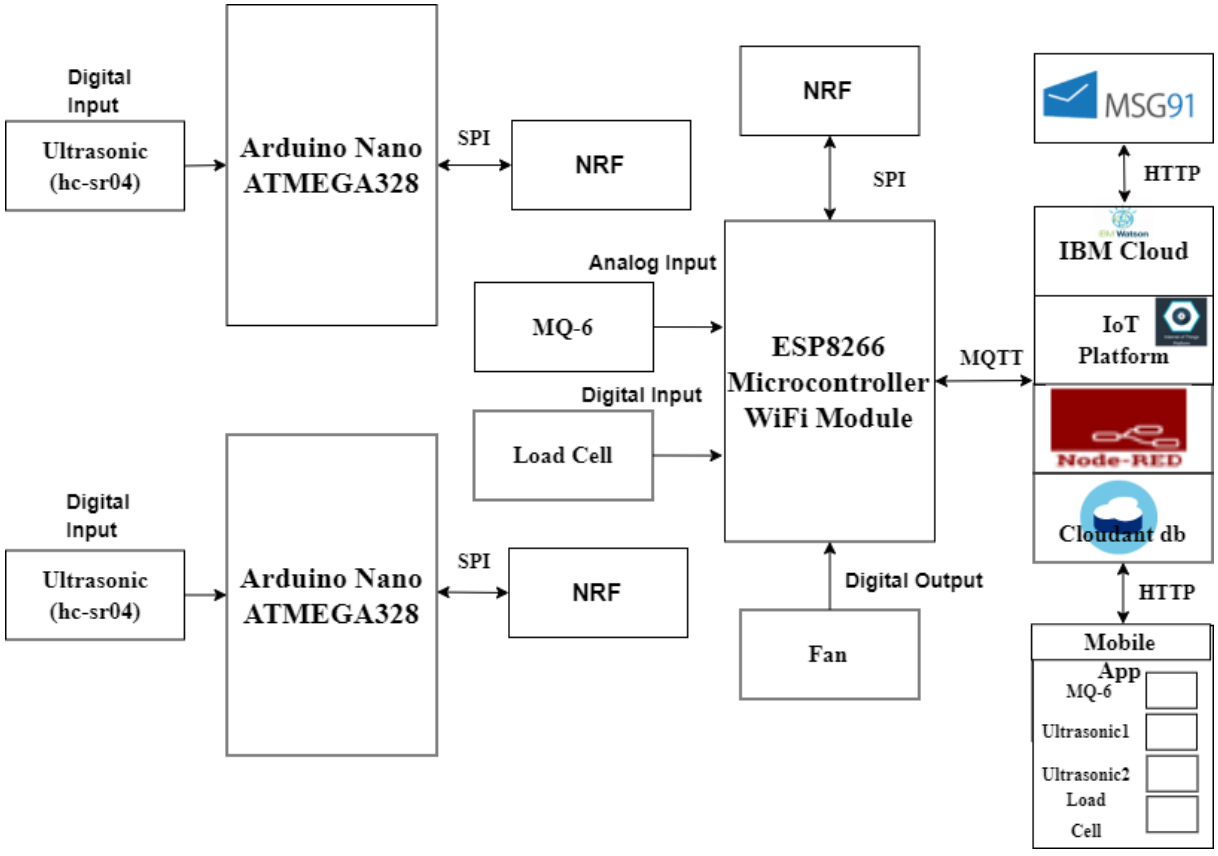
In the past years we are continuosly monitering the grocerie levels if the quantity of groceries is less then we go to the store and buy what we need. Now a days the people are busy with a lot of works and the persons dont have any time to monitor the grocerie levels and they forgot to buy and this is the existing problem

PROPOSED SOLUTION:

Previously there is a human need to monitor the jar levels By this project there is no human need to monitor the grocerie Jar levels, LPG level and Gas leakage Detection and this project helps us We can replace all the regular storage jars with the smart jars, which sends an alert when the jar gets empty or the measured sensor value is below the threshold. These jars communicate with the controller through Nrf communication. The cylinder is attached with a leakage sensor that detects the leakage from the cylinder and sends a notification if any leakage is detected. If any leakage is detected the exhaust fans are automatically switched ON. Cylinder weight is also measured and sends an alert when it is empty, based on the empty cylinder weight. All these parameters can be monitored by both Mobile App and Web App.

Theoritical Analysis:

Block Diagram:



SOFTWARE DESIGNING:

In the software designing part create a IBM cloud platform. In this design the raspberrypi model is used .the software should be design by taking a random values and then sent to the IBM IOT cloud services and then the data send to the mobile application which was developed using MIT app inventor.Here we use python language for coding,Node-Red,etc.

Steps for software designing

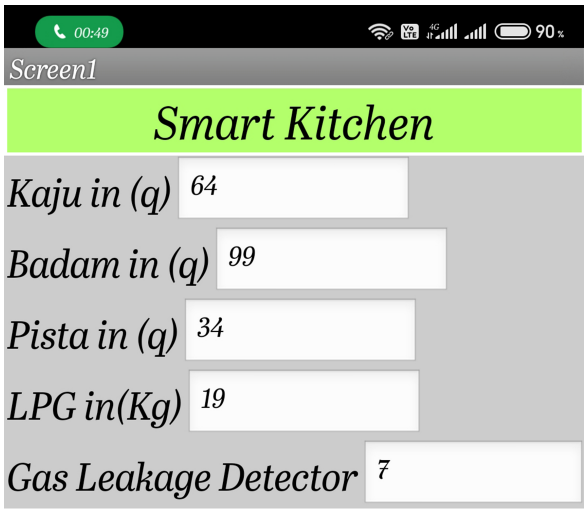
1.Setup Environment

2.Setup Hardware And Develop The Code

3.Building a Web App

4.Buinding a Mobile Application

Result:



ADVANTAGES AND DISADVANTAGES OF SMART PARKING SYSTEM

ADVANTAGES:-

Real-Time Data and Insights:- For to become a Kitchen as smart – Smart Kitchen provides you with rich data-sets that can be used to identify the grocerie levels,Gas Cylinder level indications for booking instently and it reduces the gas leak explosons by getting alert from gas leakage detector.

Reduced Monitoring:- By using this project there is no need to monitor the grocerie jars we can do our works when the jars quantity reaches below the throushold level then automatically we get an notification so reduces monitoring.

Safety:- The increase in demand of use of LPG the rate of accidents caused due to it have also increased in the past few years. The majority of the accidents are due to the explosion of LPG cylinder. So the gas leakage detector can detect tha gas in all time and turn on the fan automatically. By using using this Project we can reduce the gas cylinder explosion accidents.

DISADVANTAGES:-

1. It consists of more sensors so cost is high
2. Use of notification service will result in a greater cost
3. It may be a bit confusion for unfamiliar users
4. It is not required for high peak hour volume facilities
5. There may be a fear of break down

APPLICATIONS:-

1. The smart kitchen can be implemented in
    - Home kitchens
    - Grocerie stores
    - Small Food Industries

CONCLUSION:-

2. The project focuses implementatiom of grocerie jar levels, gas cylinder level based on weight and gas leak detection using internet of things
3. The system benefits of smart kitchen go well beyond avoiding gas leakages
4. Developing a smart kitchen solutions with reduces the gas leak explosons
5. Time saving
6. Thus,the proposed system can detect the gas leakages at gas cylinder and useful to avoid accidents and Time wasting issues.

FUTURE SCOPE:-

1. The future of the smart kitchen is expected to be significantly influenced by the arrival of automated smart kitchens.
2. It increases more applications for more and different problems and it increases oppurtunities and decreases problems.

APPENDIX:

```
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random
import requests
#Provide your IBM Watson Device Credentials
organization = "9u9gxc"
deviceType = "Raspberrypi"
deviceId = "123456"
authMethod = "token"
authToken = "12345678"

def myCommandCallback(cmd):
      print("Command received: %s" % cmd.data)#Commands
try:
        deviceOptions = {"org": organization, "type": deviceType, "id": deviceId,
"auth-method": authMethod, "auth-token": authToken}
        deviceCli = ibmiotf.device.Client(deviceOptions)
        #.....

except Exception as e:
        print("Caught exception connecting device: %s" % str(e))
        sys.exit()

# Connect and send a datapoint "hello" with value "world" into the cloud as an event
of type "greeting" 10 times
deviceCli.connect()
while True:
```

```
    jar1=random.randint(0, 100)
    if jar1 < 10:
        print("Kaju needs to refill")
        url = "https://www.fast2sms.com/dev/bulk"
        querystring =
{"authorization":"I4yrX8QM5S7TPCxRsEufdemLFq6nw0ObptBGoNDvaH1WhKUcj3PqhLwzya62flxE8XpRBeOsm3CVKTiY","sender_id":"FSTSMS","message":"Kaju needs to refill","language":"english","route":"p","numbers":"7013831615"}
        headers = {
            'cache-control': "no-cache"
            }
        response = requests.request("GET", url, headers=headers,
params=querystring)
        print(response.text)
    elif jar1 < 50:
        print("Kaju at 50%")
    elif jar1 <80:
        print("Kaju is almost full")
    else:
        print("Kaju is full")
    jar2=random.randint(0, 100)
    if jar2 < 10:
        print("Badam needs to refill")
        url = "https://www.fast2sms.com/dev/bulk"
        querystring =
{"authorization":"I4yrX8QM5S7TPCxRsEufdemLFq6nw0ObptBGoNDvaH1WhKUcj3PqhLwzya62flxE8XpRBeOsm3CVKTiY","sender_id":"FSTSMS","message":"Badam needs to refill","language":"english","route":"p","numbers":"7013831615"}
        headers = {
            'cache-control': "no-cache"
            }
        response = requests.request("GET", url, headers=headers,
params=querystring)
        print(response.text)
    elif jar2 < 50:
        print("Badam at 50%")
    elif jar2 <80:
        print("Badam is almost full")
    else:
        print("Badam is full")
```

```
    jar3=random.randint(0, 100)
    if jar3 < 10:
        print("Pista needs to refill")
        url = "https://www.fast2sms.com/dev/bulk"
        querystring =
{"authorization":"I4yrX8QM5S7TPCxRsEufdemLFq6nw0ObptBGoNDvaH1WhKUcj3PqhLwzya62flxE8XpRBeOsm3CVKTiY","sender_id":"FSTSMS","message":"Pista needs to refill","language":"english","route":"p","numbers":"7013831615"}
        headers = {
            'cache-control': "no-cache"
            }
        response = requests.request("GET", url, headers=headers,
params=querystring)
        print(response.text)
    elif jar3 < 50:
        print("Pista at 50%")
    elif jar3 <80:
        print("Pista is almost full")
    else:
        print("Pista is full")

    lpg=random.randint(10, 20)
    if lpg < 15:
        print("lpg at 50%")
    elif lpg < 12:
        print("lpg at low")
    elif lpg >=10:
        print("LPG needs to refill")
        url = "https://www.fast2sms.com/dev/bulk"
        querystring =
{"authorization":"I4yrX8QM5S7TPCxRsEufdemLFq6nw0ObptBGoNDvaH1WhKUcj3PqhLwzya62flxE8XpRBeOsm3CVKTiY","sender_id":"FSTSMS","message":"LPG needs to refill","language":"english","route":"p","numbers":"7013831615"}
        headers = {
            'cache-control': "no-cache"
            }
        response = requests.request("GET", url, headers=headers,
params=querystring)
        print(response.text)
    else:
```

```
            print("lpg is full")
        gas_leak=random.randint(0,10)
        if  gas_leak > 0:
            print("Gas leakage detected at LPG")
            url = "https://www.fast2sms.com/dev/bulk"
            querystring =
{"authorization":"I4yrX8QM5S7TPCxRsEufdemLFq6nw0ObptBGoNDvaH1WhKUcj3Pq
hLwzya62flxE8XpRBeOsm3CVKTiY","sender_id":"FSTSMS","message":"Gas leakage
detected at LPG","language":"english","route":"p","numbers":"7013831615"}
            headers = {
                'cache-control': "no-cache"
                }
            response = requests.request("GET", url, headers=headers,
params=querystring)
            print(response.text)
        else:
            print("No gas leakage detected")

        #Send jar1 & jar2 & jar3 & lpg & gas_leakage_sensor to IBM Watson
        data = { 'Kaju' : jar1, 'Badam': jar2, 'Pista': jar3, 'lpg': lpg, 'gas_leakage_detector':
gas_leak, }
        #print (data)
        def myOnPublishCallback():
          print ("Published Kaju = %sQ" % jar1, "Badam = %sQ" % jar2, "Pista = %sQ" %
jar3, "lpg = %sKg" % lpg, "gas_leakage_detector = %s " % gas_leak, "to IBM Watson")
        success = deviceCli.publishEvent("smartkitchen", "json", data, qos=0,
on_publish=myOnPublishCallback)
        if not success:
          print("Not connected to IoTF")
        time.sleep(2)
        deviceCli.commandCallback = myCommandCallback
# Disconnect the device and application from the cloud
deviceCli.disconnect()
```