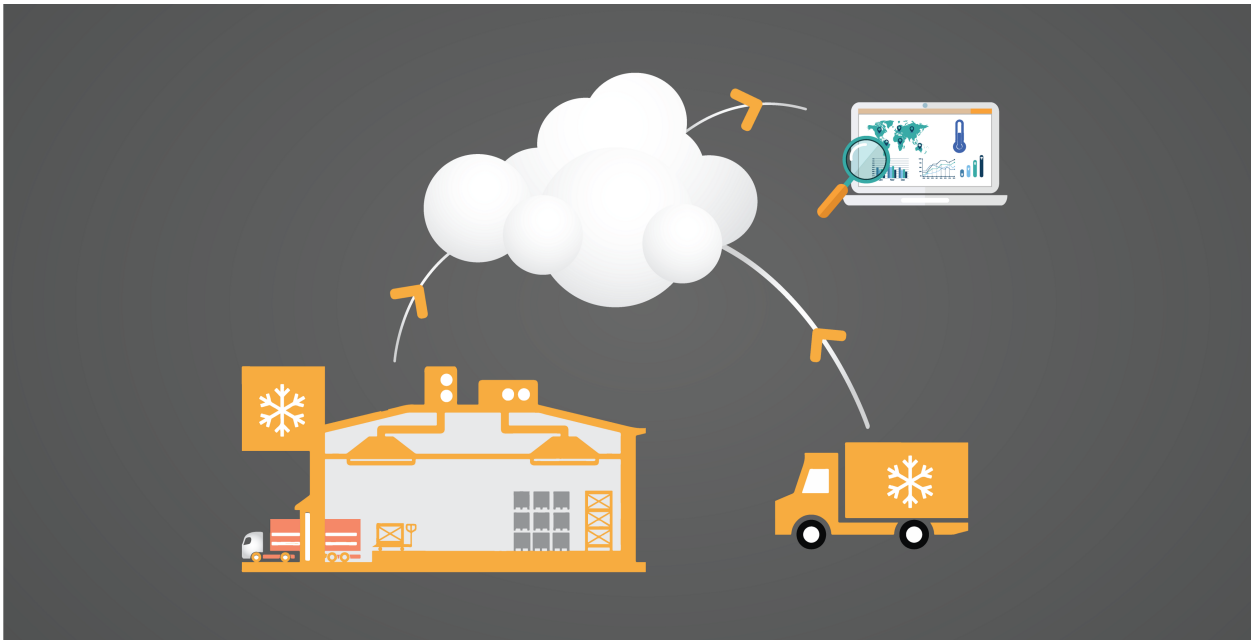
PROJECT REPORT

PROJECT TITLE :COLD STORAGE MONITORING USING IOT



Submitted by,

IMMANUEL D A

RAKSHITH S

1.INTRODUCTION

1.1 Overview :

Perishable goods are consumed by almost everyone on a daily basis. These products include fruits, vegetables, dairy products, meat and poultry, fresh food, frozen food, seafood and even pharmaceutical products. Since these goods are temperature-sensitive and their storage time varies from a few days to a few years, these items are stored in temperature-controlled rooms. This is essential to ensure their safety and quality. Moreover, it is also important to ensure that the temperature of the room must never exceed the optimal temperature.
The cold storage business started around the 1800s, but during that time the only industry using it were the breweries and they mainly rely on the unsanitary practice of ice-harvesting. By the 1900s, the idea of cold storage had dribbled into the meat packing business, until finally in the mid 20th century; these refrigeration facilities have been installed on trucks which were used to transport perishable foods to long distances. India's cold chain sector is anticipated to reach Rs. 624 billion ($13 billion) in 2020, according to 2017 estimates. Barely 5 percent of this is organised. Almost 40 percent of India's total food production is lost in transit, wasted by consumers or damaged (according to the United Nations Development Program), and about 75 percent of medicines produced lose their potency due to inferior temperature controls during transport.

1.2 Purpose :

Industries dealing with perishable goods have been using cold storage facilities for decades now. These facilities require critical monitoring as well as control of temperature and humidity. Monitoring these facilities has not been an easy task, but the advent of new technologies has made it quite convenient.

Instrumenting cold storage facility with an IoT-enabled cold storage monitoring solution helps you to record, monitor and maintain the conditions inside the facility on a regular basis.

2.LITERATURE SURVEY

2.1 Existing Problem :

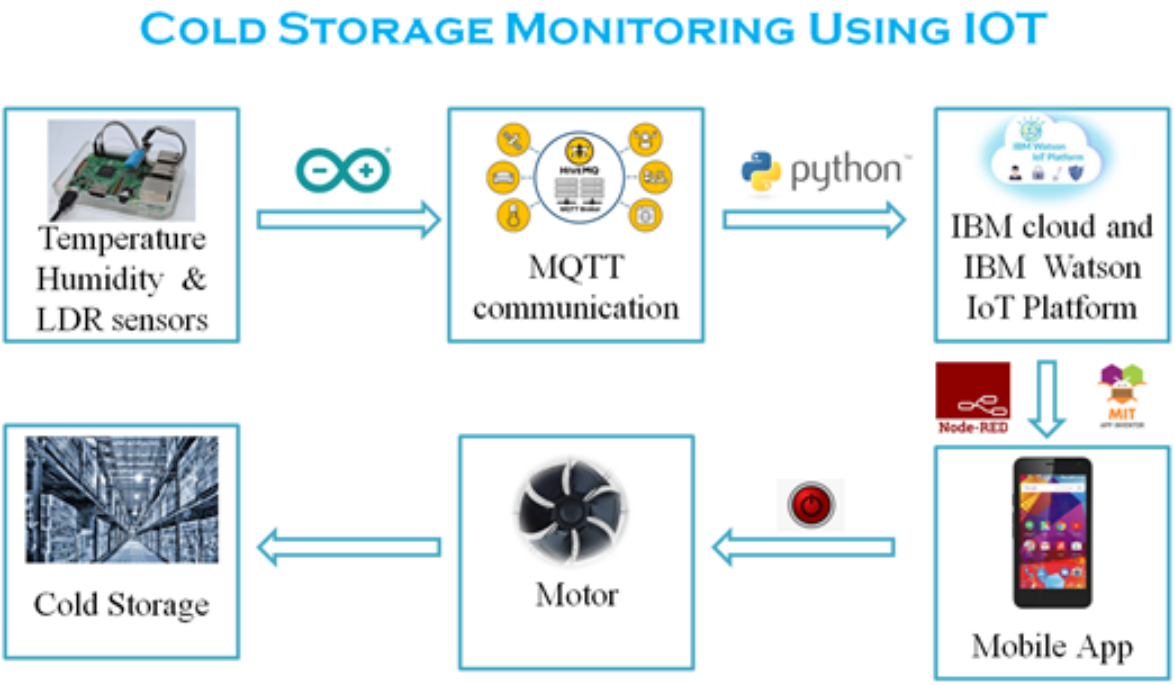
Various products that are kept in different sections of one facility require unique temperatures to be maintained to prevent decay and extend its shelf-life. Along with constant monitoring, whenever a deviation is detected, adjustments must be made immediately as weather changes and temperature fluctuations affect the environment and can have adverse effects on the stock.

2.2 Proposed Solution:

Implementation of an IoT solution for cold storage facilities helps to monitor the necessary parameters and adjusts them when deviation occurs from their preset values. This helps to prevent food decay. The solution also sends alerts via SMS text and email whenever an anomaly is detected. Hence, preserving the item and maintaining regulatory compliance becomes easy. A cold storage temperature monitoring solution includes thermostats and sensors that constantly measure the temperature of a closed system, capture data and send it to a centralized platform over a network. This helps the logistics manager to monitor the shipment remotely and ensure the maintenance of optimum temperature.
The implementation of a cold storage temperature monitoring solution is helpful for climate-sensitive perishable items. Smart warehouse solution is easy to incorporate, convenient to use and ensures that the quality of goods does not degrade in warehouse and shipping.

3.THEORITICAL ANALYSIS

3.1 Block Diagram :



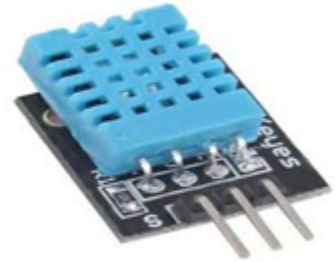
3.2 Hardware & Software Designing :

Hardware Components used :

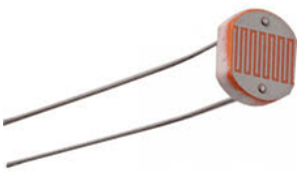
- Arduino Uno
- dHT11 Temperature and Humidity Sensor
- Light Dependent Resistor Sensor
- Jumper Wires

Arduino Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

DHT11 is used as Temperature and humidity sensor. The sensor comes with a dedicated NTC to measure temperature and an 8-bit microcontroller to output the values of temperature and humidity as serial data. The sensor is also factory calibrated and hence easy to interface with other microcontrollers.



A **Light Dependent Resistor (LDR**) is also called a photoresistor or a cadmium sulfide (CdS) cell. It is also called a photoconductor. It is basically a photocell that works on the principle of photoconductivity. The passive component is basically a resistor whose resistance value decreases when the intensity of light decreases. This optoelectronic device is mostly used in light varying sensor circuit, and light and dark activated switching circuits. Some of its applications include camera light meters, street lights, clock radios, light beam alarms, reflective smoke alarms, and outdoor clocks.



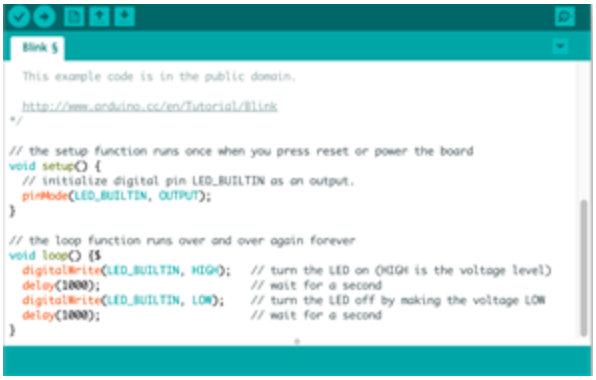
Software Used:

- Arduino IDE
- Python Shell
- IBM Cloud
- Node-RED
- MIT App Inventor

Arduino IDE :

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.

This software can be used with any Arduino board.



Python Shell:

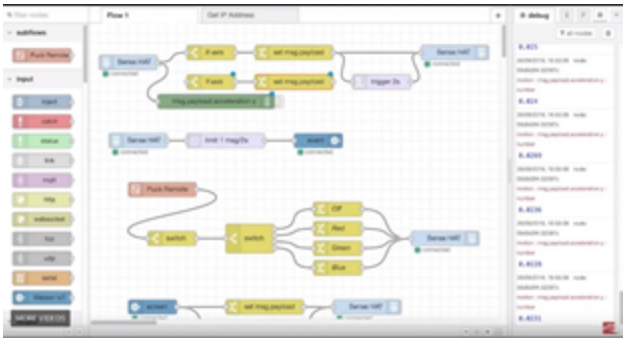
Python provides a Python Shell (also known as Python Interactive Shell) which is used to execute a single Python command and get the result. Python Shell waits for the input command from the user. To open the Python Shell on Windows, open the command prompt, write python and press enter.

IBM Cloud

IBM Cloud is a suite of cloud computing services from IBM that offers both platform as a service (PaaS) and infrastructure as a service (IaaS). With IBM Cloud IaaS, organizations can deploy and access virtualized IT resources -- such as compute power, storage and networking -- over the internet.

Node-RED

Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways. It provides a browser-based editor that makes it easy to wire together flows using the wide range of nodes in the palette that can be deployed to its runtime in a single-click.



MIT App Inventor

MIT App Inventor is an online platform designed to teach computational thinking concepts through development of mobile applications. Students create applications by dragging and dropping components into a design view and using a visual blocks language to program application behavior.

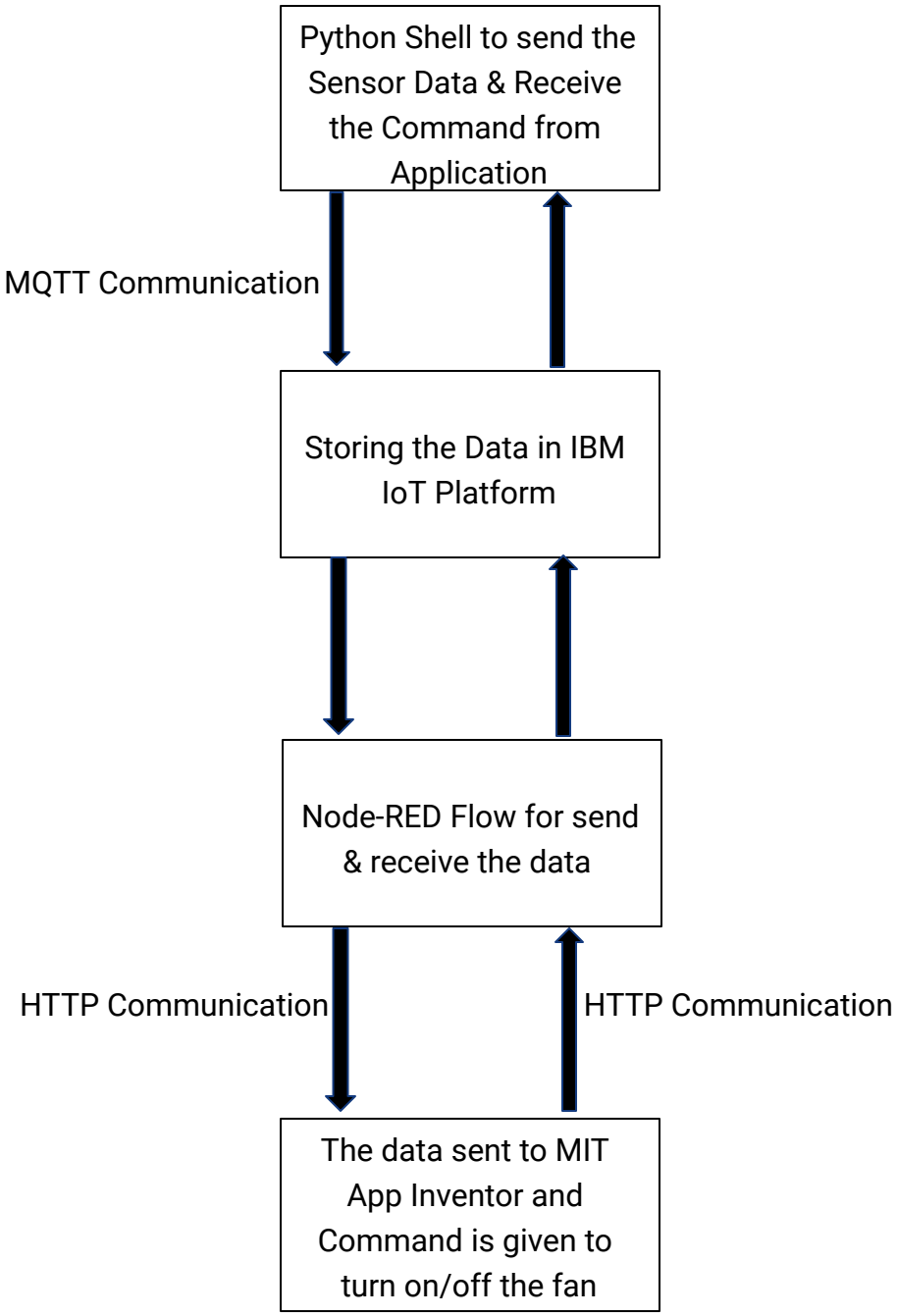
4.Experimental Investigation

The miniature IoT based Cold Storage Maintenance system is structured as:

- Obtaining the sensor values from the DHT11 and LDR sensors through the Arduino snippet codes.
- These values are run through the Python code developed using MQTT client-server Publishing.
- The values are sent to the IBM cloud storage device through the specific credentials required.
- Using Node-red and MIT App Inventor a mobile application has been designed.
- Generally when not looked-into, the temperature of the facility keeps increasing steadily beyond the optimum level(say 12◦C) . The humidity and Light intensity values are thus followed.
- The values of the present time conditions of the facility is diplayed in run time on the app.
- Even the increase is noted down and when the value exceeds the threshold(say 21◦C) a notification is sent through sms as well as notified through the app.
- Depending upon the action taken the Cold storage fan is run inorder to get the facility to its optimum conditions.

- As Humidity and Light intensity is dependant on Temperature they need not be monitored seperately. The humidity for given range, levels from (75% to 95%).
- After reaching the optimum conditions the user will be notified to turn off the fan.
- Keeping the fan on even after that might lead to technical issues or problem in the preservation of the objects of the facility.
- When the fan is off, fluctuations in the surrounding environment might again lead to variation in temperature which can again be monitored is the similar way mentioned above.

5.FlowChart



6.RESULT

The Cold storage monitoring System using IoT was successfully Built Using the dHT11 Temperature and Humidity sensor and LDR sensor where the data was sent using MQTT communication protocol to the IBM IoT Platform Cloud then to Node-RED flow, from Node-RED using HTTP Communication protocol to the Appliction where the user can monitor and controls the Fan using the Mobile Application.

7.Advantages & Disadvantages

Advantages :

- Real time maintenance of the facility.
- Providing optimum environmental conditions to specific items and monitoring all at the same time.
- Minimizes the intervention of Humans.
- Minimizes wastage.
- IoT-enabled systems maintain a detailed record of every activity to improve the security of cold storage facilities.

Disadvantages :

- Delay in retrieving data from the cloud.
- Unexpected crashing down of Mobile/Web Application.
- Issues with Motor fans.
- Higher financial investment.
- Inability to maintain optimum conditions during extreme repeated fluctuations.

8.Applications

A cold storage monitoring solution would be most beneficial for:

- Agriculture industry
- Blood banks
- Food & beverage industry
- Healthcare industry
- Pharmaceutical facilities
- Restaurant chains
- Food manufacturing facilities
- Educational institutions that provide meals to students, etc.

9.Conclusion

All in all, the implementation of an IoT-based cold storage monitoring system leads to the optimum utilization of space and resources. It helps to track the usage pattern and power consumption of devices, minimize wastage, detect anomalies within the facility and monitor and control the intensity of light as per the changes in daylight. An IoT-enabled monitoring solution brings terrific value to businesses and enhances profitability.

10.Future scope

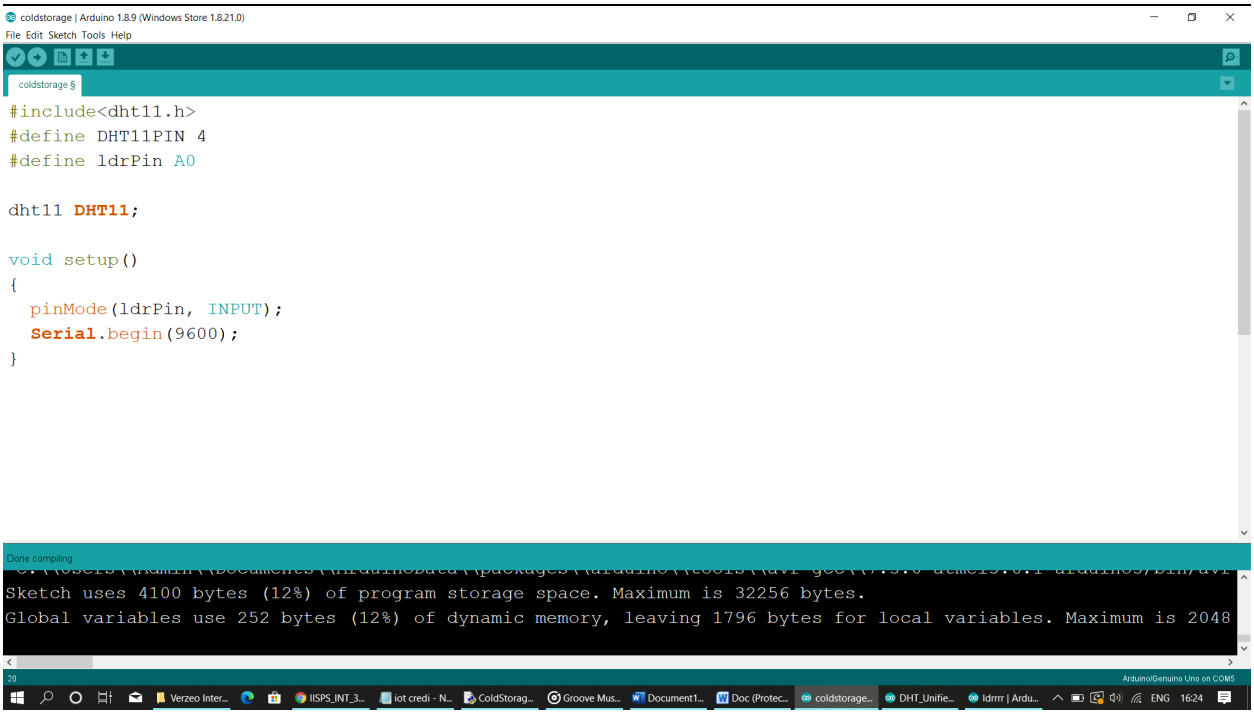
The growing demand for processed foods as a result of higher disposable incomes would lead to a requirement of robust cold chain distribution system.
Growth of retail market in India and the entry of multinational retail giants will be a major driving force.
Increasing production of horticultural products that require cold storage facilities.
Establishment of new and modern cold storage facility will necessarily push up demand for refrigerated vehicles.The cold chain monitoring space in the country has seen growth and the emergence of multiple players.

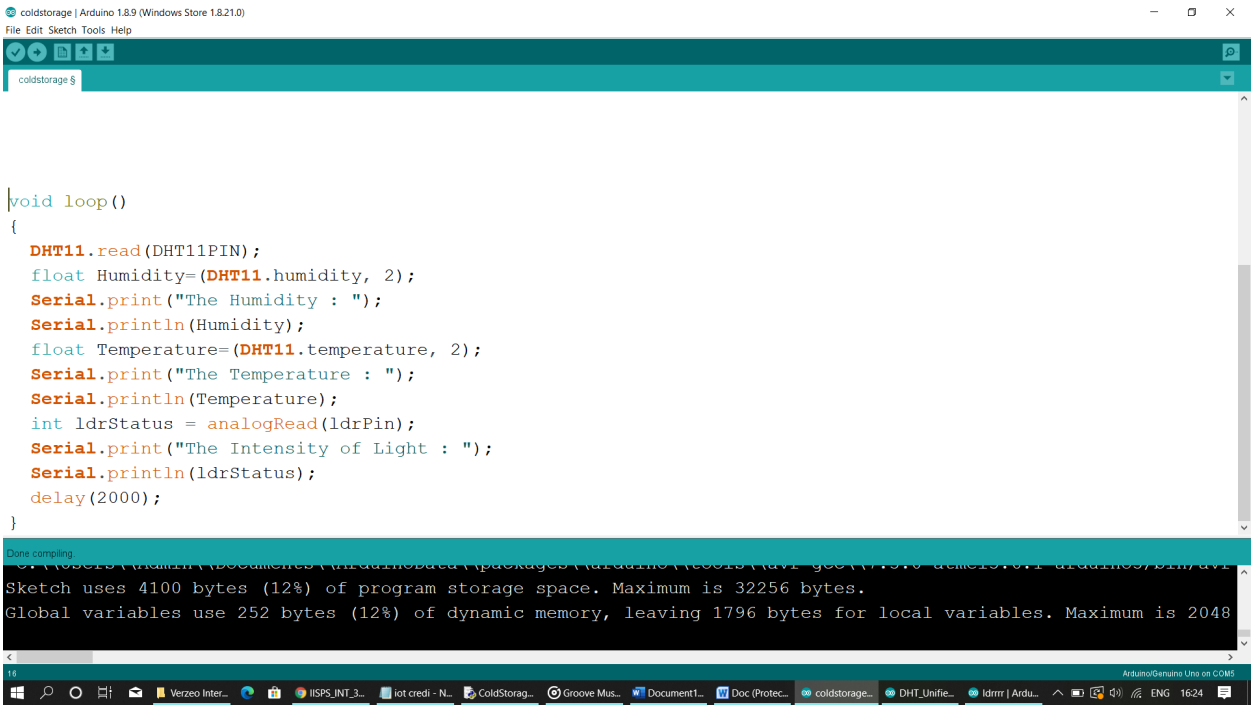
Along with IoT, AI can be integrated to a cold storage management system that keeps you informed about the empty spaces in your cold storage for its optimal usage. Also, as the IoT sensors collect data and record the movement of assets and other items, it can be used to send alerts in case they detect any abnormal movement suspecting a theft. Not only that, but the system also sends alerts about the expiry of a product to make sure that it is moved out for selling in time. Smart Warehouse concept can be implemented.

Appendix:

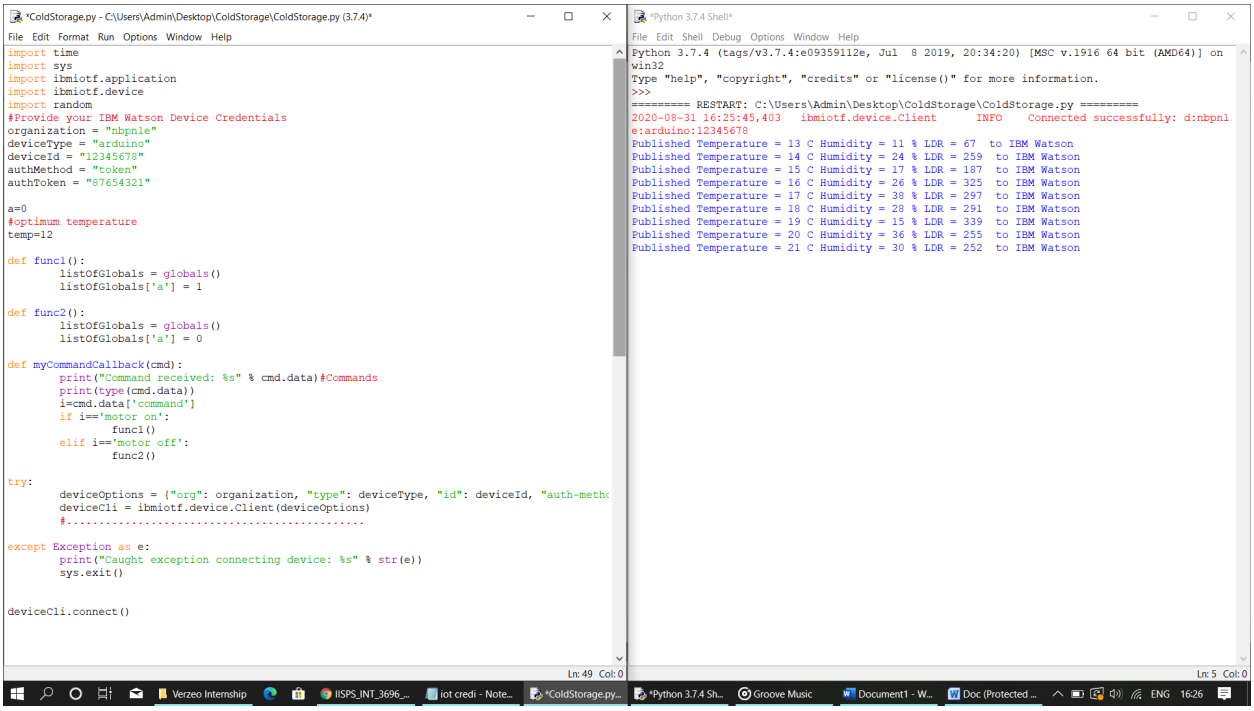
- **“Cold Storage in India: Present Scenario and Future Directions”** by ***Rakesh Pandey*** of Indian Council of Agricultural Research
- https://www.foodlogistics.com/warehousing/article/21047818/the-rise-of-hightech-cold-storage
- https://encyclopedia2.thefreedictionary.com/Cold-Storage+Facility
- **“An IoT-based Occupational Safety Management System in Cold Storage Facilities”** by *G.T.s. Ho*

Arduino Code for dHT11 sensor and LDR sensor to sense :





Python Code for send the data and receive the command from IBM IoT Platform using MQTT communication protocol :



ColdStorage.py - C:\Users\Admin\Desktop\ColdStorage\ColdStorage.py (3.7.4)

File Edit Format Run Options Window Help

```
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random
#Provide your IBM Watson Device Credentials
organization = "nbpnle"
deviceType = "arduino"
deviceId = "12345678"
authMethod = "token"
authToken = "87654321"

a=0
#optimum temperature
temp=12

def func1():
        listOfGlobals = globals()
        listOfGlobals['a'] = 1

def func2():
        listOfGlobals = globals()
        listOfGlobals['a'] = 0

def myCommandCallback(cmd):
        print("Command received: %s" % cmd.data)#Commands
        print(type(cmd.data))
        i=cmd.data['command']
        if i=='motor on':
                func1()
        elif i=='motor off':
                func2()

try:
        deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-metho
        deviceCli = ibmiotf.device.Client(deviceOptions)
        #.............................................

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


deviceCli.connect()
```

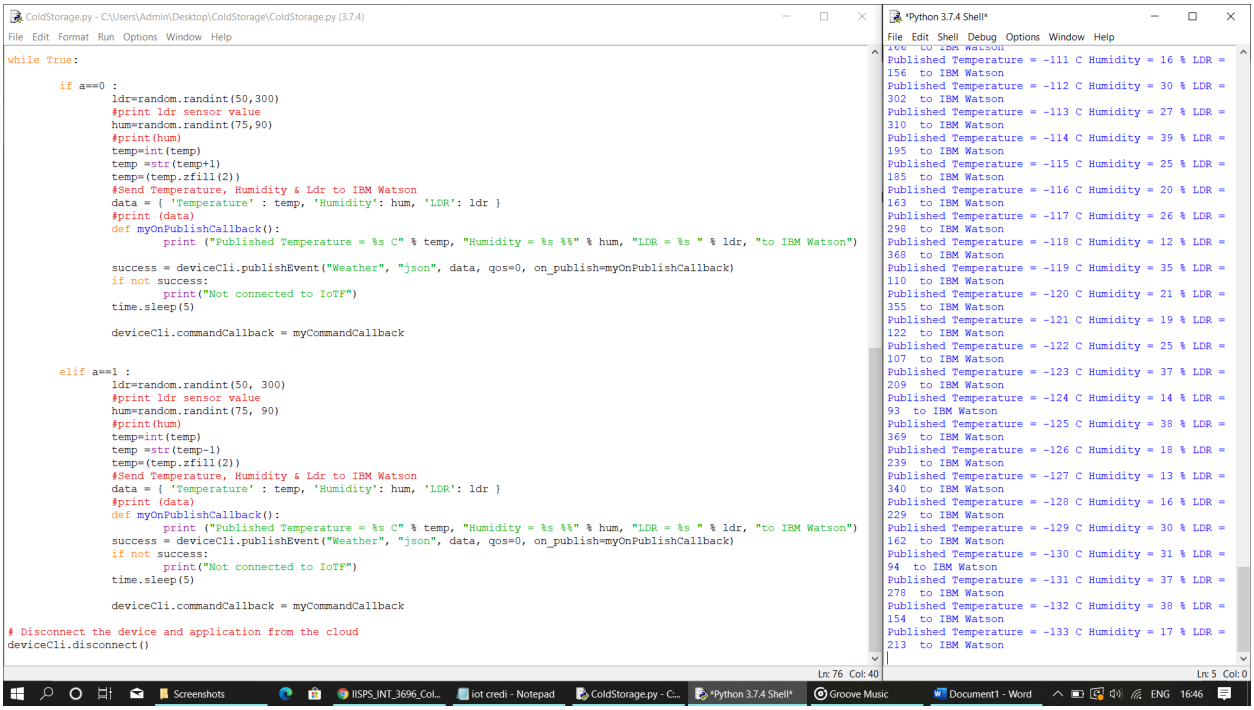
Ln: 49 Col: 0

Python 3.7.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit (AMD64)] on
win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========= RESTART: C:\Users\Admin\Desktop\ColdStorage\ColdStorage.py =========
2020-08-31 16:25:45,403   ibmiotf.device.Client      INFO    Connected successfully: d:nbpn1
e:arduino:12345678
Published Temperature = 13 C Humidity = 11 % LDR = 67  to IBM Watson
Published Temperature = 14 C Humidity = 24 % LDR = 259  to IBM Watson
Published Temperature = 15 C Humidity = 17 % LDR = 187  to IBM Watson
Published Temperature = 16 C Humidity = 26 % LDR = 325  to IBM Watson
Published Temperature = 17 C Humidity = 38 % LDR = 297  to IBM Watson
Published Temperature = 18 C Humidity = 28 % LDR = 291  to IBM Watson
Published Temperature = 19 C Humidity = 15 % LDR = 339  to IBM Watson
Published Temperature = 20 C Humidity = 36 % LDR = 255  to IBM Watson
Published Temperature = 21 C Humidity = 30 % LDR = 252  to IBM Watson
```

Ln: 5 Col: 0



ColdStorage.py - C:\Users\Admin\Desktop\ColdStorage\ColdStorage.py (3.7.4)

File Edit Format Run Options Window Help

```
while True:

        if a==0 :
                ldr=random.randint(50,300)
                #print ldr sensor value
                hum=random.randint(75,90)
                #print(hum)
                temp=int(temp)
                temp =str(temp+1)
                temp=(temp.zfill(2))
                #Send Temperature, Humidity & Ldr to IBM Watson
                data = { 'Temperature' : temp, 'Humidity': hum, 'LDR': ldr }
                #print (data)
                def myOnPublishCallback():
                        print ("Published Temperature = %s C" % temp, "Humidity = %s %%" % hum, "LDR = %s " % ldr, "to IBM Watson")

                success = deviceCli.publishEvent("Weather", "json", data, qos=0, on_publish=myOnPublishCallback)
                if not success:
                        print("Not connected to IoTF")
                time.sleep(5)

                deviceCli.commandCallback = myCommandCallback


        elif a==1 :
                ldr=random.randint(50, 300)
                #print ldr sensor value
                hum=random.randint(75, 90)
                #print(hum)
                temp=int(temp)
                temp =str(temp-1)
                temp=(temp.zfill(2))
                #Send Temperature, Humidity & Ldr to IBM Watson
                data = { 'Temperature' : temp, 'Humidity': hum, 'LDR': ldr }
                #print (data)
                def myOnPublishCallback():
                        print ("Published Temperature = %s C" % temp, "Humidity = %s %%" % hum, "LDR = %s " % ldr, "to IBM Watson")
                success = deviceCli.publishEvent("Weather", "json", data, qos=0, on_publish=myOnPublishCallback)
                if not success:
                        print("Not connected to IoTF")
                time.sleep(5)

                deviceCli.commandCallback = myCommandCallback

# Disconnect the device and application from the cloud
deviceCli.disconnect()
```

Ln: 76 Col: 40

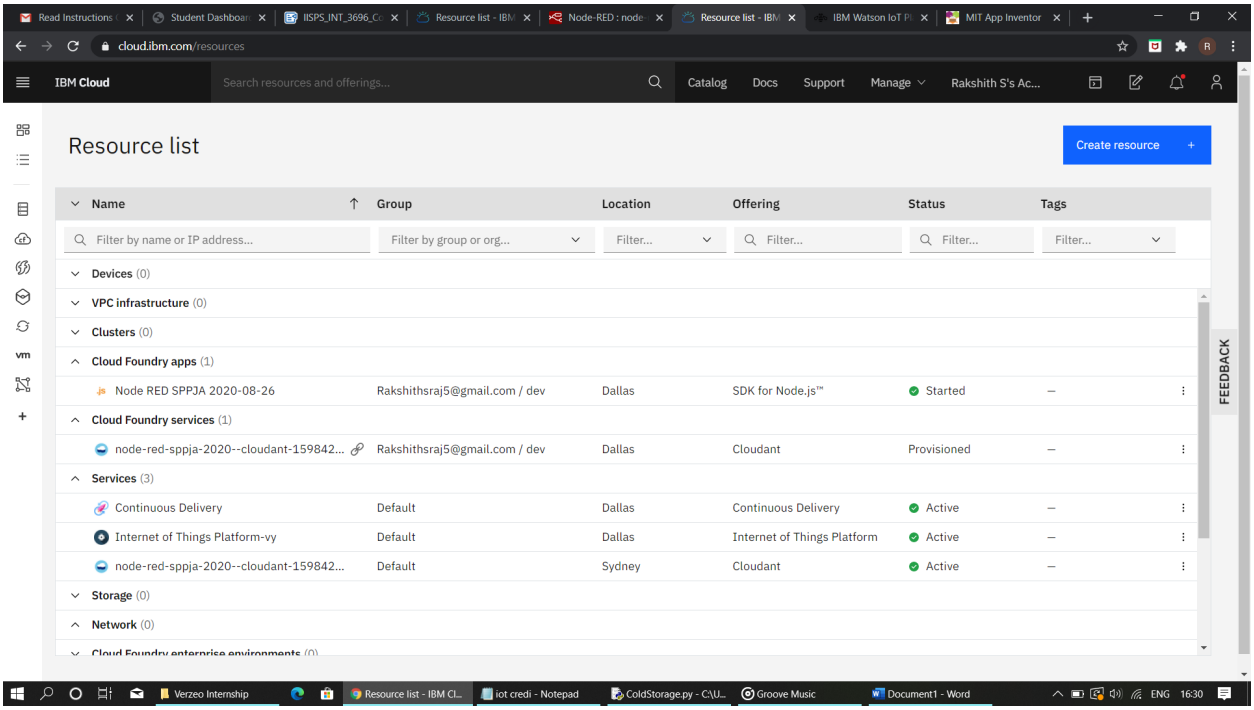
Python 3.7.4 Shell

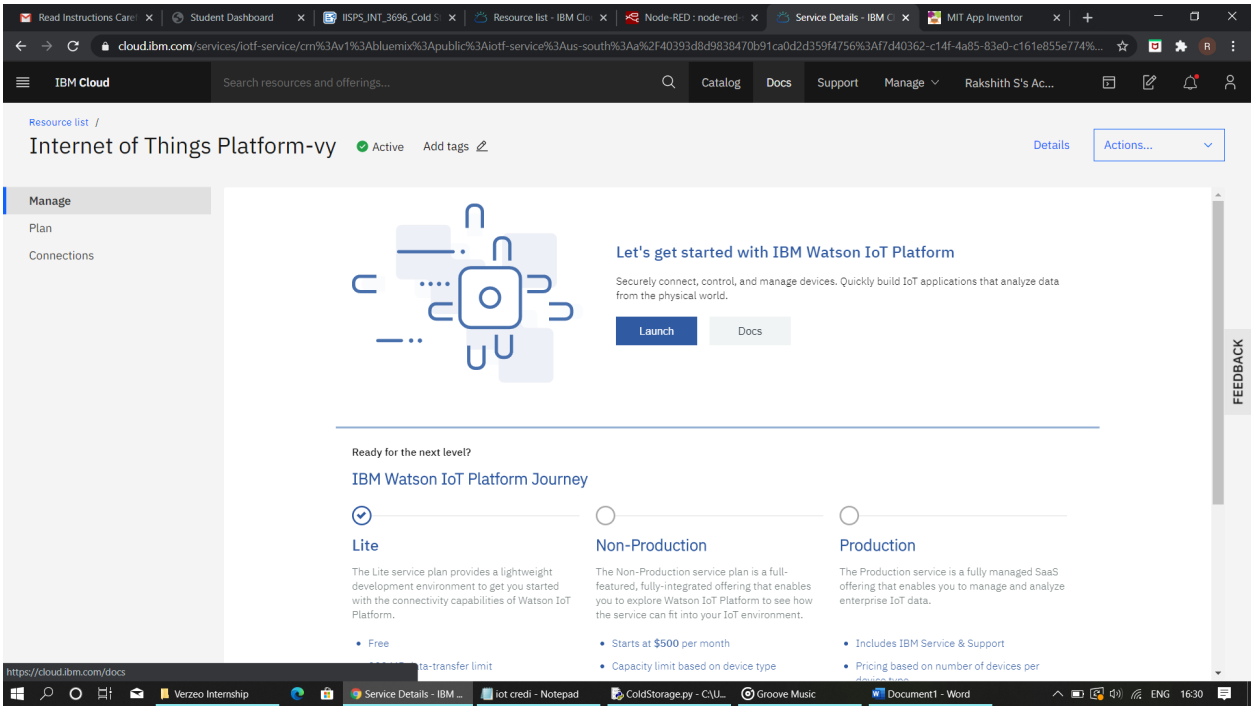
File Edit Shell Debug Options Window Help

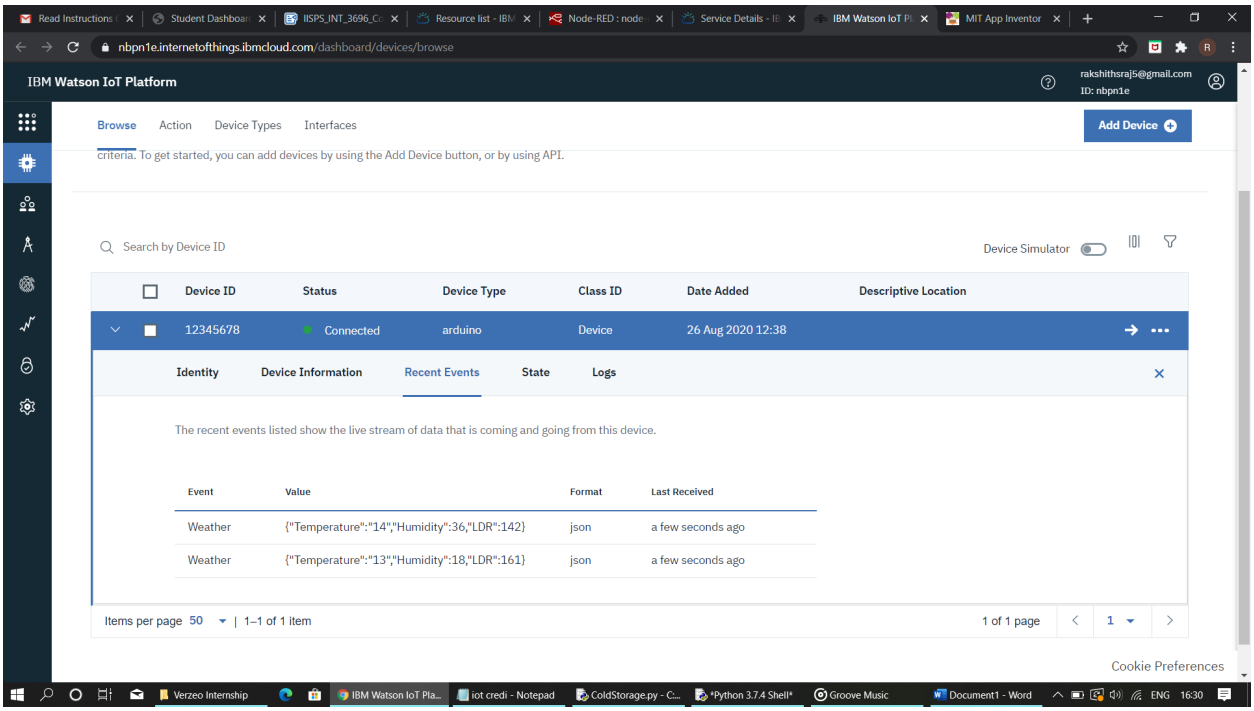
```
Published Temperature = -111 C Humidity = 16 % LDR =
156  to IBM Watson
Published Temperature = -112 C Humidity = 30 % LDR =
302  to IBM Watson
Published Temperature = -113 C Humidity = 27 % LDR =
310  to IBM Watson
Published Temperature = -114 C Humidity = 39 % LDR =
195  to IBM Watson
Published Temperature = -115 C Humidity = 25 % LDR =
185  to IBM Watson
Published Temperature = -116 C Humidity = 20 % LDR =
163  to IBM Watson
Published Temperature = -117 C Humidity = 26 % LDR =
298  to IBM Watson
Published Temperature = -118 C Humidity = 12 % LDR =
368  to IBM Watson
Published Temperature = -119 C Humidity = 35 % LDR =
110  to IBM Watson
Published Temperature = -120 C Humidity = 21 % LDR =
355  to IBM Watson
Published Temperature = -121 C Humidity = 19 % LDR =
122  to IBM Watson
Published Temperature = -122 C Humidity = 25 % LDR =
107  to IBM Watson
Published Temperature = -123 C Humidity = 37 % LDR =
209  to IBM Watson
Published Temperature = -124 C Humidity = 14 % LDR =
93  to IBM Watson
Published Temperature = -125 C Humidity = 38 % LDR =
369  to IBM Watson
Published Temperature = -126 C Humidity = 18 % LDR =
239  to IBM Watson
Published Temperature = -127 C Humidity = 13 % LDR =
340  to IBM Watson
Published Temperature = -128 C Humidity = 16 % LDR =
229  to IBM Watson
Published Temperature = -129 C Humidity = 30 % LDR =
162  to IBM Watson
Published Temperature = -130 C Humidity = 31 % LDR =
94  to IBM Watson
Published Temperature = -131 C Humidity = 37 % LDR =
278  to IBM Watson
Published Temperature = -132 C Humidity = 38 % LDR =
154  to IBM Watson
Published Temperature = -133 C Humidity = 17 % LDR =
213  to IBM Watson
```

Ln: 5 Col: 0

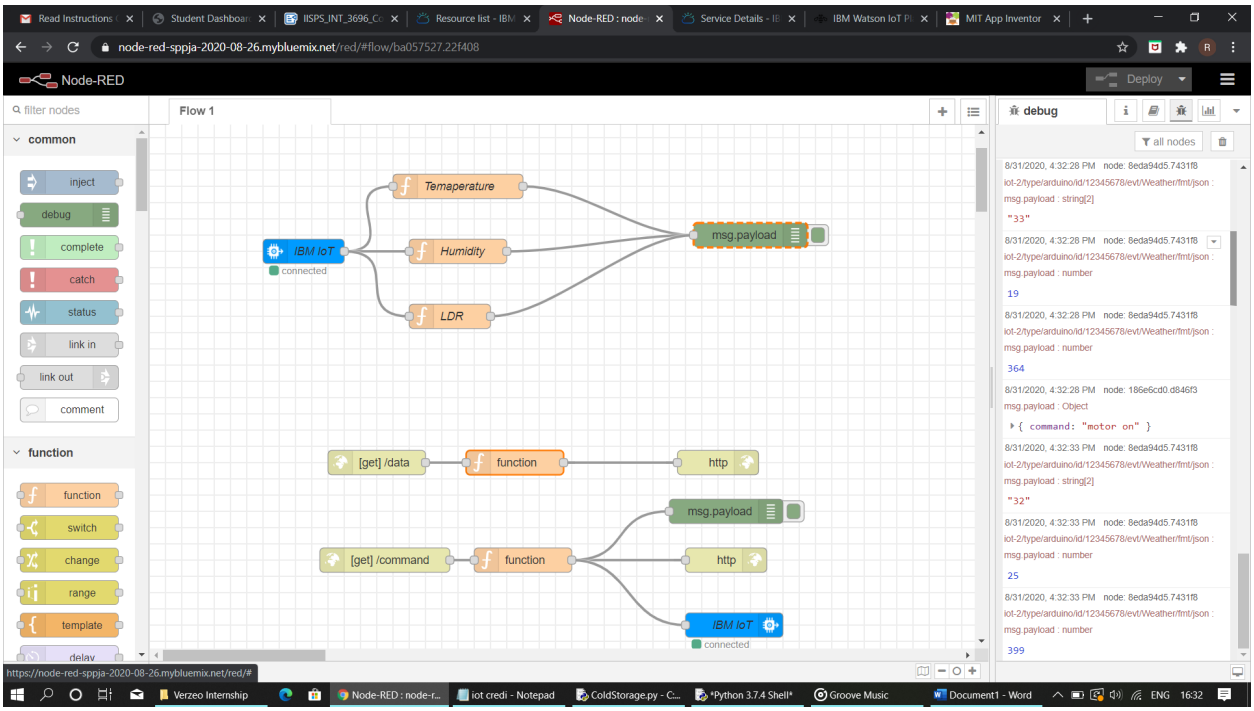
IBM IoT Platform to store the Data:



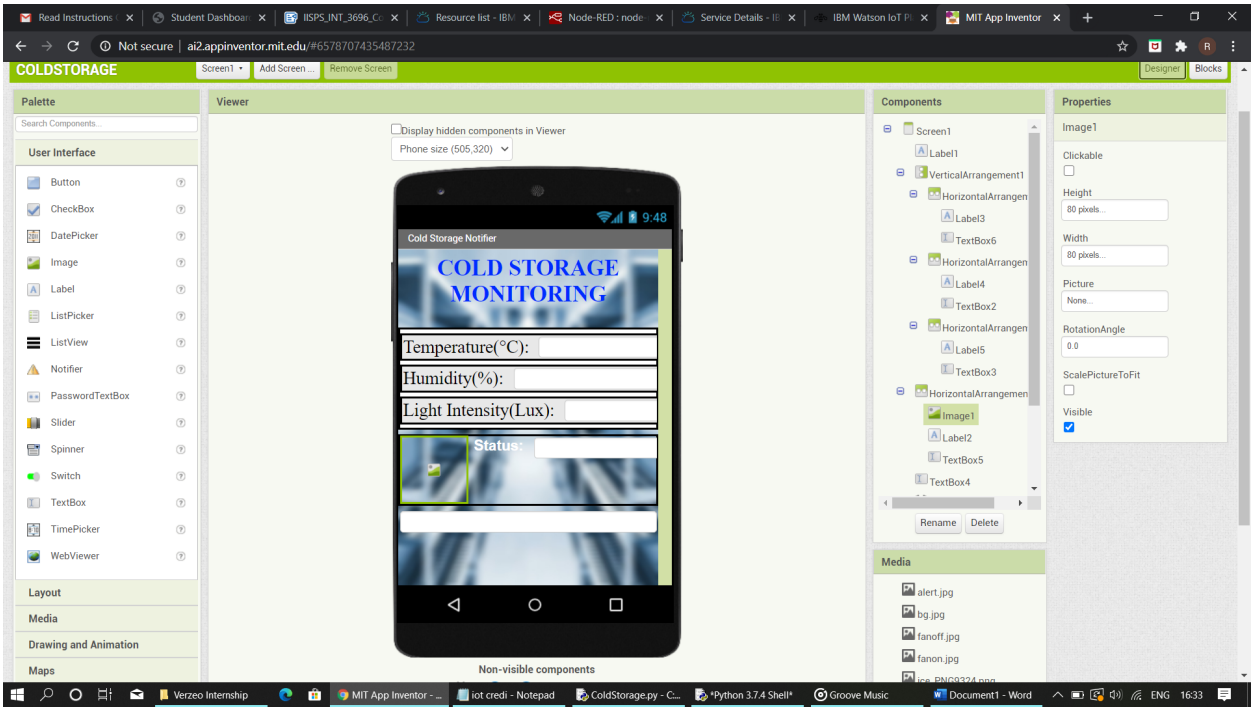


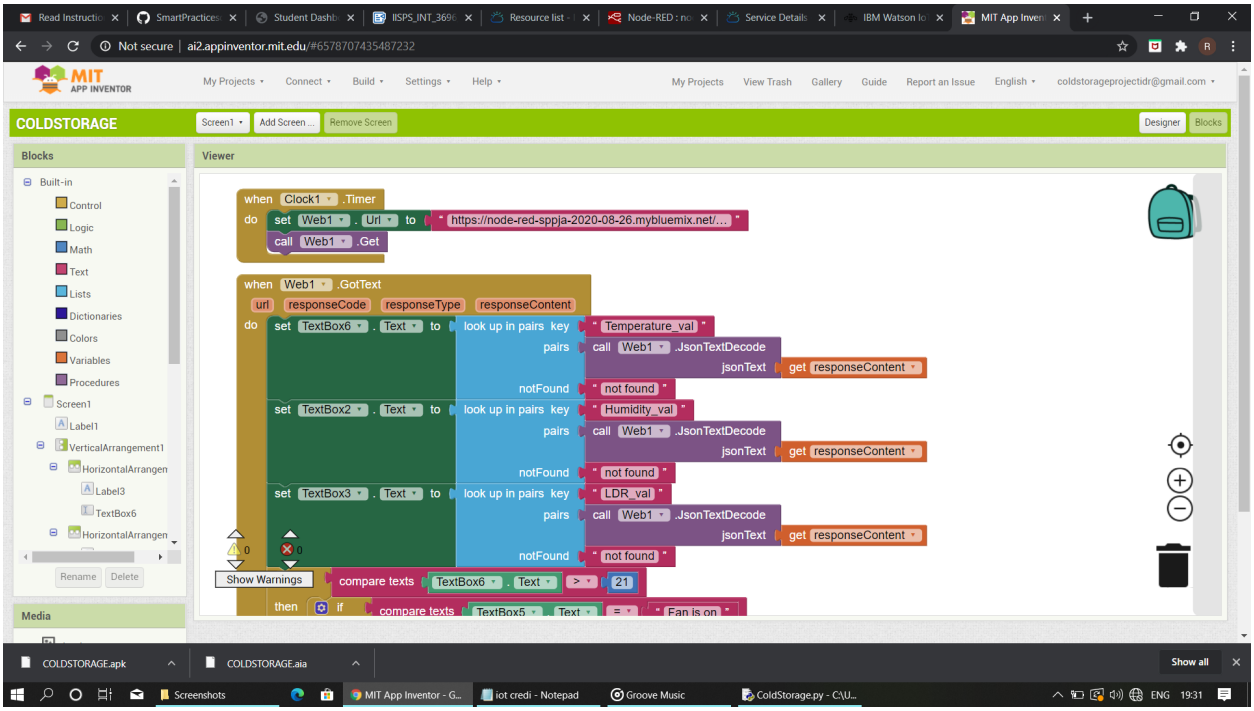


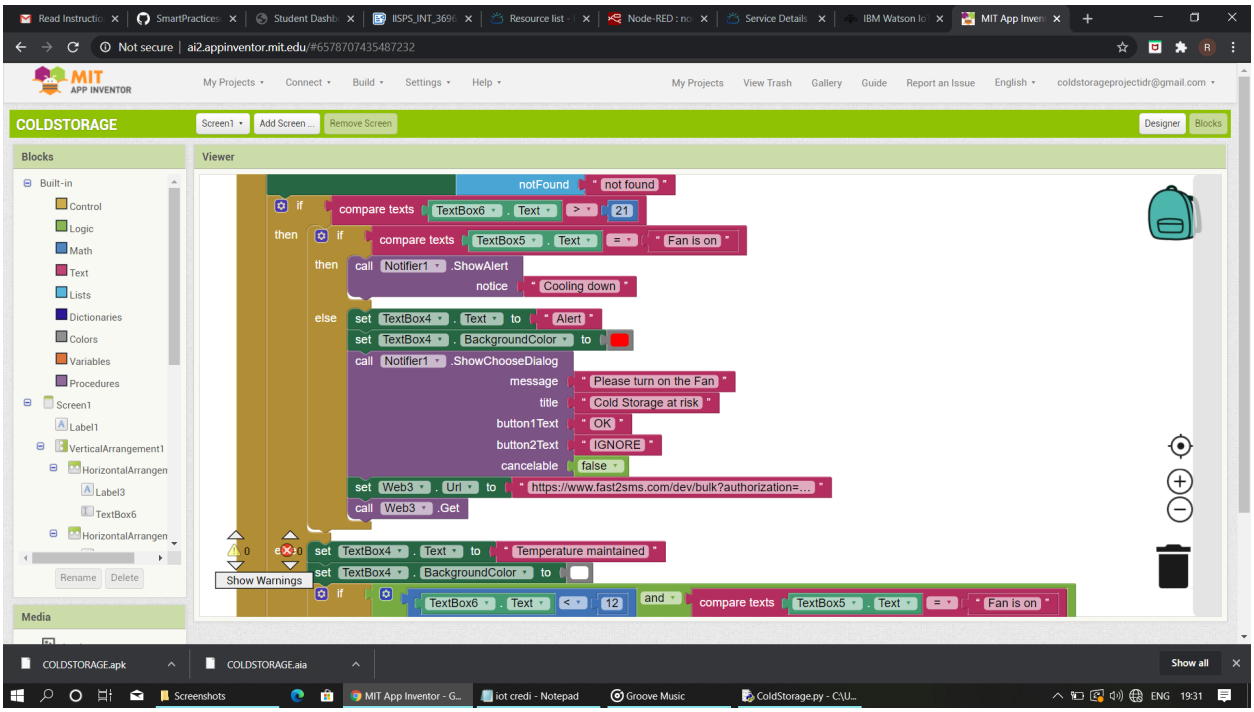
Node-RED flow : Send and receive the data & command from Mobile Application using HTTP Communication protocol

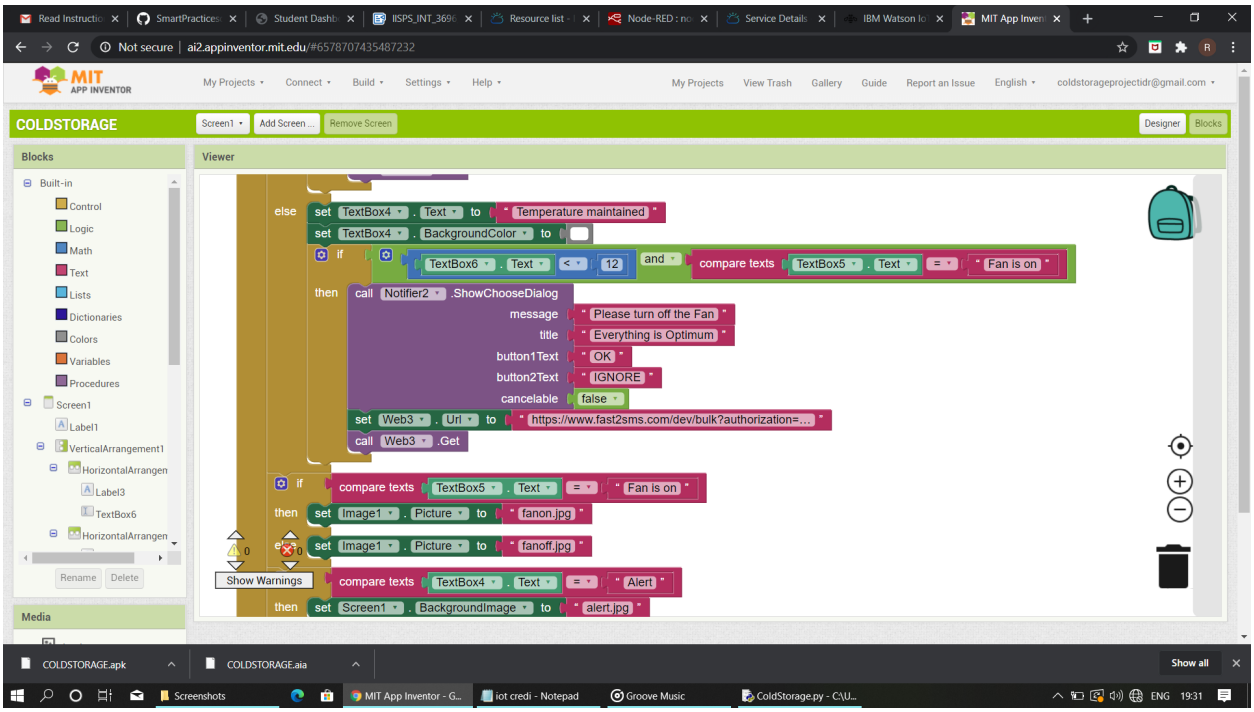


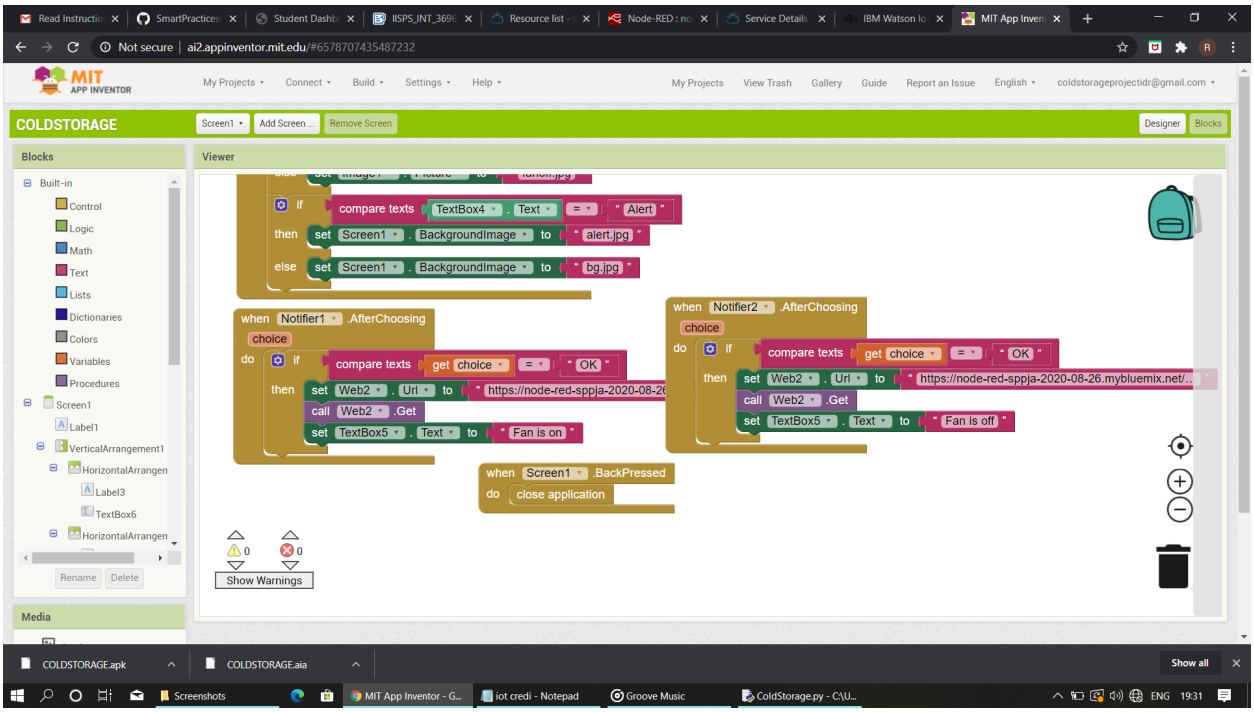
MIT App Inventor :






Not secure | ai2.appinventor.mit.edu/#6578707435487232
MIT APP INVENTOR
My Projects Connect Build Settings Help
My Projects View Trash Gallery Guide Report an Issue English coldstorageprojectidr@gmail.com
COLDSTORAGE
Screen1 Add Screen ... Remove Screen
Designer Blocks
Blocks
Viewer
Built-in
Control
Logic
Math
Text
Lists
Dictionaries
Colors
Variables
Procedures
Screen1
Label1
VerticalArrangement1
HorizontalArrangen
Label3
TextBox6
HorizontalArrangen
Rename Delete
Media
notFound " not found "
if compare texts TextBox6 . Text > 21
then if compare texts TextBox5 . Text = " Fan is on "
then call Notifier1 .ShowAlert
notice " Cooling down "
else set TextBox4 . Text to " Alert "
set TextBox4 . BackgroundColor to
call Notifier1 .ShowChooseDialog
message " Please turn on the Fan "
title " Cold Storage at risk "
button1Text " OK "
button2Text " IGNORE "
cancelable false
set Web3 . Url to " https://www.fast2sms.com/dev/bulk?authorization=... "
call Web3 .Get
Show Warnings
set TextBox4 . Text to " Temperature maintained "
set TextBox4 . BackgroundColor to
if TextBox6 . Text < 12 and compare texts TextBox5 . Text = " Fan is on "
COLDSTORAGE.apk
COLDSTORAGE.aia
Show all
Screenshots
MIT App Inventor - G...
iot credi - Notepad
Groove Music
ColdStorage.py - C:\U...
ENG 19:31


Not secure | ai2.appinventor.mit.edu/#6578707435487232
MIT APP INVENTOR
My Projects Connect Build Settings Help
My Projects View Trash Gallery Guide Report an Issue English coldstorageprojectidr@gmail.com
COLDSTORAGE
Screen1 Add Screen ... Remove Screen
Designer Blocks
Blocks
Viewer
Built-in
Control
Logic
Math
Text
Lists
Dictionaries
Colors
Variables
Procedures
Screen1
Label1
VerticalArrangement1
HorizontalArrangen
Label3
TextBox6
HorizontalArrangen
Rename Delete
Media
else set TextBox4 . Text to " Temperature maintained "
set TextBox4 . BackgroundColor to
if TextBox6 . Text < 12 and compare texts TextBox5 . Text = " Fan is on "
then call Notifier2 .ShowChooseDialog
message " Please turn off the Fan "
title " Everything is Optimum "
button1Text " OK "
button2Text " IGNORE "
cancelable false
set Web3 . Url to " https://www.fast2sms.com/dev/bulk?authorization=... "
call Web3 .Get
if compare texts TextBox5 . Text = " Fan is on "
then set Image1 . Picture to " fanon.jpg "
set Image1 . Picture to " fanoff.jpg "
Show Warnings
compare texts TextBox4 . Text = " Alert "
then set Screen1 . BackgroundImage to " alert.jpg "
COLDSTORAGE.apk
COLDSTORAGE.aia
Show all
Screenshots
MIT App Inventor - G...
iot credi - Notepad
Groove Music
ColdStorage.py - C:\U...
ENG 19:31


MIT APP INVENTOR
COLDSTORAGE
Screen1
Add Screen ...
Remove Screen
Designer
Blocks
Viewer
Built-in
Control
Logic
Math
Text
Lists
Dictionaries
Colors
Variables
Procedures
Screen1
Label1
VerticalArrangement1
Label3
TextBox6
Rename
Delete
Media
if compare texts TextBox4 . Text = " Alert "
then set Screen1 . BackgroundImage to " alert.jpg "
else set Screen1 . BackgroundImage to " bg.jpg "
when Notifier1 .AfterChoosing
choice
do if compare texts get choice = " OK "
then set Web2 . Url to https://node-red-sppja-2020-08-2
call Web2 .Get
set TextBox5 . Text to " Fan is on "
when Notifier2 .AfterChoosing
choice
do if compare texts get choice = " OK "
then set Web2 . Url to " https://node-red-sppja-2020-08-26.mybluemix.net/..
call Web2 .Get
set TextBox5 . Text to " Fan is off "
when Screen1 .BackPressed
do close application
Show Warnings
COLDSTORAGE.apk
COLDSTORAGE.aia
Show all
Screenshots
MIT App Inventor - G...
iot credi - Notepad
Groove Music
ColdStorage.py - C:\U...
ENG
19:31

THANK YOU