Temperature Monitoring and Alert System using IoT & ML

Created by: Nisha M.Bharti

1. Components:

Bolt Module

LM35 Temperature Sensor

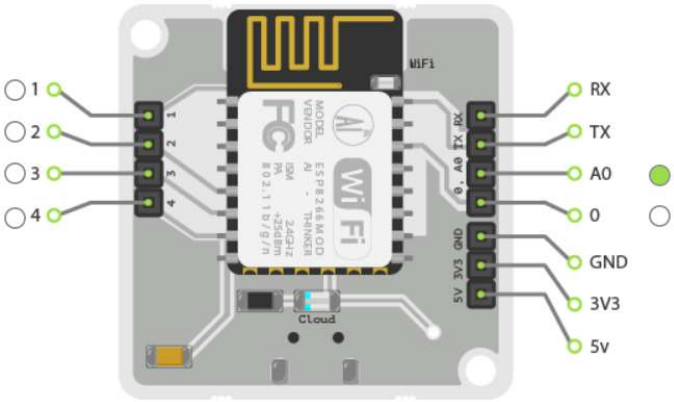
Jumper Cables

2. Apps and Softwares or Online platforms

Bolt Cloud

- VmWare Virtual Workstation
- Mailgun for mail alert
- Twilio for sms alert

3. Connections/ Circuit:



LM35	Bolt WiFi module
VCC ⟷	5 V
Output ⟷	A0
GND ⟷	GND

USB cable connected to power adapter or power bank.

4. Operation:

- First a product named PharmaTempMonitAlert is created on Bolt Cloud and is linked to bolt Wifi module, as shown in Fig. 1

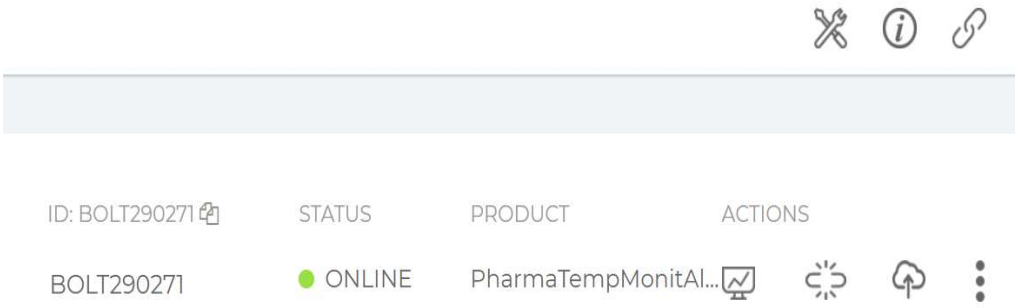


Fig. 1

The product code is updated for polynomial regression algorithm as shown in Fig. 2, to predict the data points for temperature.

Filename Extension

PharmaTempMonitAlert js

```
setChartLibrary('google-chart');
setChartTitle('Polynomial Regression');
setChartType('predictionGraph');
setAxisName('time_stamp','temp_op');
mul(0.15);
plotChart('time_stamp','temp_op');
```

Fig. 2

- Using the Bolt WiFi module, the data (temperature detected) is sent to Bolt IoT cloud which connects to VMware virtual server. For this, the API key and device ID are configured for the Bolt module, Twilio and Mailgun.

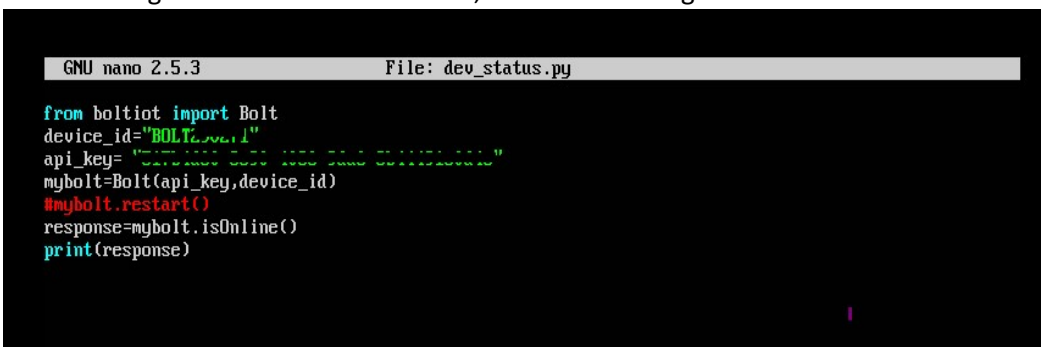


Fig. 3

- The twi_conf.py has the configuration of SID, authorization token and sender/receiver number:

SID = 'You can find SID in your Twilio Dashboard'
AUTH_TOKEN = 'You can find on your Twilio Dashboard'
FROM_NO = 'This is the no. generated by Twilio. You can find this on your Twilio Dashboard'
TO_NO = 'This is your number. Make sure you are adding +91 in beginning'

- The mail_conf.py has the api key, sandbox url, sender and receiver mail id:

mailgun_api = 'This is the private API key which you can find on your Mailgun Dashboard'
sandbox_url= 'You can find this on your Mailgun Dashboard'
sender_mail = 'test@' + SANDBOX_URL # No need to modify this. The sandbox URL is of the format test@YOUR_SANDBOX_URL
rec_mail = 'Enter your Email ID Here'

- At the server, the python scripts have also the anomaly detection algorithm. It first checks whether enough data as required by parameter frame-size has been accumulated or not. After the data accumulation, Z-score (Zn) is calculated. Zn

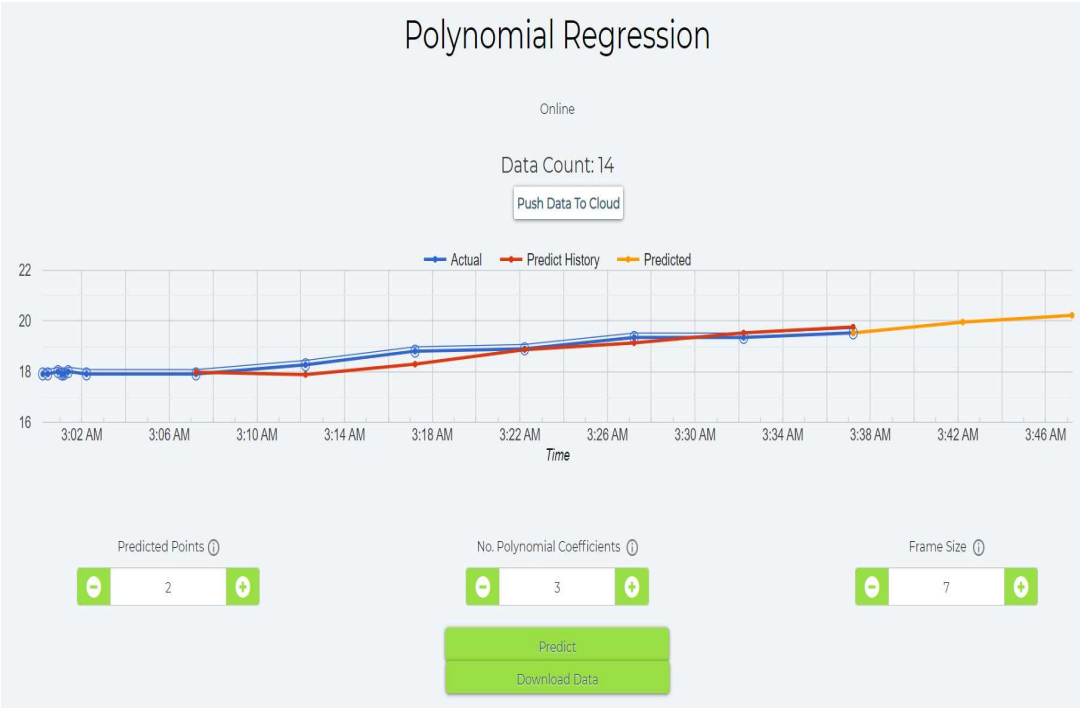
```
def compute_bounds(history_data,frame_size,factor):
        if len(history_data)<frame_size:
                return None
        if len(history_data)>frame_size:
                del history_data[0:len(history_data)-frame_size]
        Mn=statistics.mean(history_data)
        Variance=0
        for data in history_data:
                Variance += math.pow((data-Mn),2)

        Zn = factor * math.sqrt(Variance / frame_size)
        High_bound = history_data[frame_size -1] + Zn
        Low_bound= history_data[frame_size-1]-Zn
        return [High_bound,Low_bound]
```

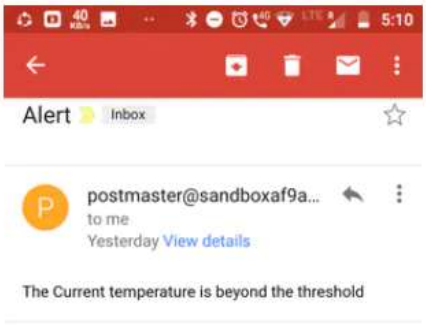
is calculated to find out the upper and lower threshold bounds required to check if a new data point is normal or anomalous.

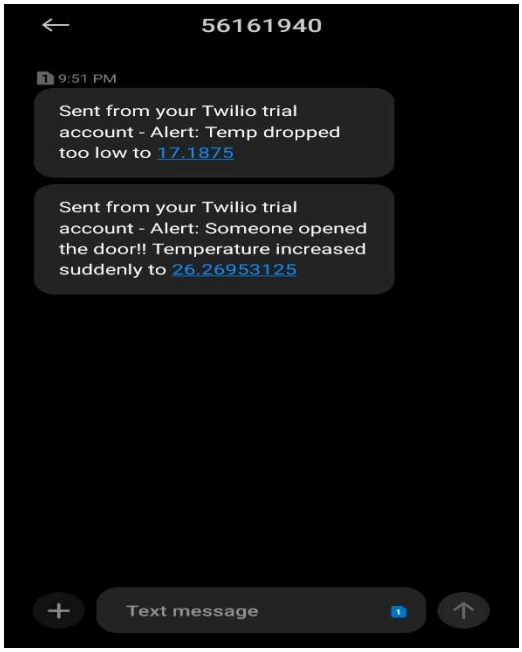
- The temperature range for operation was taken as 20 to 25 deg C, which can also be changed as per requirement. If anomaly is detected in terms of rise of temperature, then e-mail and sms alert is sent to the user using Mailgun/ Twilio API.

5. Output:




```
Reading sensor value:
Storage Temperature is: 26.171875
Reading sensor value:
Storage Temperature is: 26.171875
Reading sensor value:
Storage Temperature is: 26.26953125
Temp level increased suddenly. Sending alert
Response from Email <Response [200]>
Response from SMS <Twilio.Api.V2010.MessageInstance sid=SM54d9673ec63b44fc9ae6af7d33a23f6e account_s
id=AC3ed23dace9171a4d4a9e82afcb4c2545>
```





6. Applications

This system setup can be used for applications where continuous temperature monitoring is required. For example, pharmacy storage, industrial equipments, food production facilities, etc.

7. Main Code:

```
 GNU nano 2.5.3                File: temp_monit_mail_sms.py                      Modified


import mail_conf
import twi_conf

from boltiot import Sms,Email,Bolt
import json,time,math,statistics

def compute_bounds(history_data,frame_size,factor):
        if len(history_data)<frame_size:
                return None
        if len(history_data)>frame_size:
                del history_data[0:len(history_data)-frame_size]
        Mn=statistics.mean(history_data)
        Variance=0
        for data in history_data:
                Variance += math.pow((data-Mn),2)

        Zn = factor * math.sqrt(Variance / frame_size)
        High_bound = history_data[frame_size -1] + Zn
        Low_bound= history_data[frame_size-1]-Zn
        return [High_bound,Low_bound]



min_lim=215
max_lim=256
fram_size=8
mul_factor=4

import dev_status

mybolt=Bolt(dev_status.api_key,dev_status.device_id)
mailer=Email(mail_conf.mailgun_api,mail_conf.sandbox_url,mail_conf.sender_mail,mail_conf.rec_mail)
sms= Sms(twi_conf.SID,twi_conf.AUTH_TOKEN,twi_conf.TO_NO,twi_conf.FROM_NO)

history_data=[]


while True:
        print("Reading sensor value: ")
        response = mybolt.analogRead('A0')
        data = json.loads(response)
        temp_val=(int(data['value']))/10.24
        print("Storage Temperature is: "+str(temp_val))
        sensor_value=0
        try:
                sensor_value=int(data['value'])
                temp=sensor_value/10.24
                if sensor_value > max_lim or sensor_value < min_lim:
                        response=mailer.send_email("Alert!", "Temperature of refrigerator is: "+str$
                        response_text=json.loads(response.text)
                        print("Response from mailgun: ",str(response_text["message"]))
        except Exception as e:
                print("Error occured: details ",e)
                continue
        bound = compute_bounds(history_data,fram_size,mul_factor)
        if not bound:
                required_data_count = fram_size-len(history_data)
                print("Not enough data to get z-score. Need ",required_data_count," more data point$
                history_data.append(int(data['value']))
                time.sleep(5)
                continue
    try:
            if sensor_value >bound[0] or sensor_value < bound[1]:
                    temp_val=sensor_value/10.24
                    print("Temp level increased suddenly. Sending alert")
                    response1 = mailer.send_email("Alert!","Someone opened the door and tempera$
                    respons1_text=json.loads(response1.text)
                    print("Response from Email: ",+str(response1_text['message']))
                    response2=sms.send_sms("Alert: Someone opened the door!! Temperature increa$
                    print("Response from SMS",response2)
            if sensor_value <bound[1]:
                    temp_val=sensor_value/10.24
                    #print("Temp level dropped too low")
                    response2=sms.send_sms("Alert:Someone opened the door!! Temp dropped too lo$
                    print("Response from sms",response2)
            history_data.append(sensor_value)
    except Exception as e:
            print("Error",e)
    time.sleep(5)
```