**Temperature Monitor Using Bolt IoT**

Introduction

We are using a Bolt IoT device to design a warning system that detects the current temperature and sends a warning through Telegram when the temperature crosses a given threshold or shows an anomaly (sudden increase or decrease in temperature).

Components

Bolt IoT Device, LM35 Temperature Sensor, LED, Connecting Wires, Resistor, Wi-Fi with Internet Connection, etc.

Connection

Hold the sensor in a manner such that you can read LM35 written on it. In this position, identify the sensor pins as VCC, Output and GND from your left to right. The VCC pin of the LM35 connects to 5v of the Bolt module. The Output pin of the LM35 connects to the Analog input, and the GND pin of the LM35 to the GND of the Bolt module.

Python Codes

We are using the functions available in the “*boltiot*” Python library to design our system.

The codes can be found at: *https://github.com/ep20btech11004/projects/tree/main/Temp\_Monitor*

The Python file “*conf.py*” contains all the required credentials for the Bolt device, Telegram, and the temperature threshold.

“*device\_status.py*” is used to check the working status of the Bolt IoT device.

“*device\_restart.py*” is used to restart our Bolt IoT device.

“*telegram\_alert.py*” is our main code. It gets the temperature reading by the temperature sensor through Bolt Cloud, checks it against the given threshold, and sends a warning through telegram using a telegram api if the temperature crosses it. It also checks if there is an anomaly in the temperature curve using Z-Score Analysis and sends a warning if found.

Analysing - “*telegram\_alert.py*”

We have declared a function “get\_sensor\_value\_from\_pin()” which returns the sensor value from the selected pin. In case of any error, it returns -999 which we have declared as an error response. It takes one parameter (pin of the Bolt module to which the sensor is connected) and makes a request to the Bolt Cloud to fetch the latest sensor value from the mentioned pin using `mybolt.analogRead()` . The Bolt Cloud returns a status of 1 if the request was made successfully and anything else apart from 1 means that the request has failed. We can finally get the sensor value from the "value” field inside the response, convert it to an integer and return it. The function is encased inside a try-except block to handle any exceptions and errors. In case of any error or exception inside the try block, the except block is executed. This will return a response of -999.

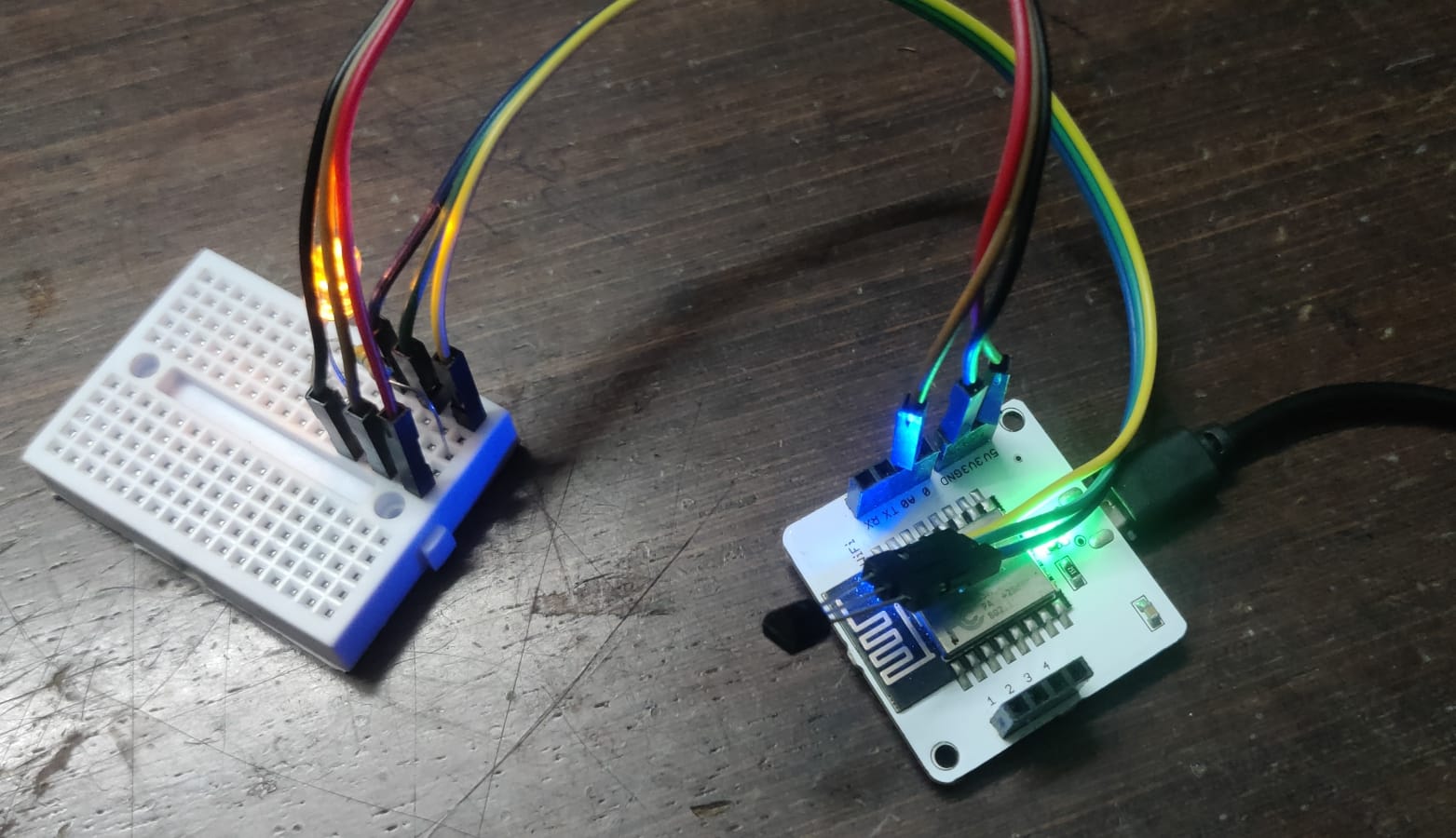
We have created a function to send a message via Telegram. The function `send\_telegram\_message()` takes one parameter i.e. the message to send. First, we build the URL so that Telegram knows which bot it must send the message to. The *telegram\_bot\_id* is required for this purpose. In the next step, we need to make a HTTP request to the Telegram servers using the URL we have built earlier. The request is a "POST" request which contains all the relevant data like URL and the data to be contained in the request. The status of the request is stored in the "ok" field of the “*telegram\_data”* variable and we are returning it. The "ok" field will always contain a boolean value i.e., True/False and is True if the message has been sent. The function is encased in a try-except block so that any errors are caught and it returns a False if any error is present in the try block.

We can now evaluate the Z-score using the “*compute\_bounds()*” function. It accepts the sensor data, frame-size, and the multiplication factor as parameters. We first check if enough data (*frame-size*) has been accumulated to calculate the Z-score, and if there is too much data, then the code deletes the older data. We calculate the mean and variance of the selected data. Then we calculate Z-score with “*Zn = factor \* math.sqrt(Variance / frame\_size)*”. We use this to evaluate the higher and lower bound by adding and subtracting it to the previous sensor value respectively and is returned by the function.

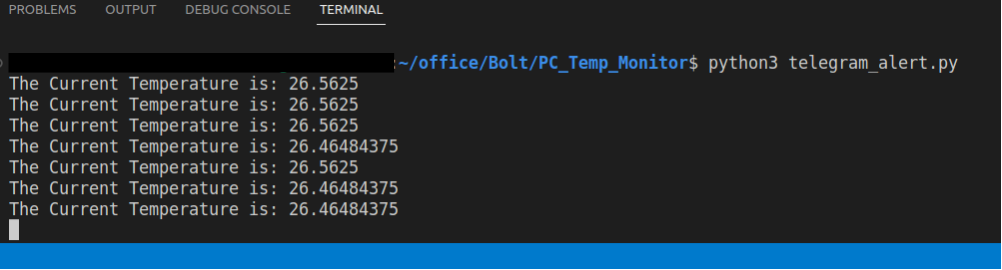
We have declared a While loop to run infinitely. Inside the while loop, we have used the function `get\_sensor\_value\_from\_pin()` to get the value from the pin A0 of the Bolt device. Now, we need to first check if the response is valid by checking if the response is equal to -999. If the response is exactly -999, we know an error has occurred. At this point we can print a message and skip the reading. The program waits for 10 seconds and then continues to the next iteration of the while loop. When we get a valid response, we need to check if the sensor value has exceeded the threshold that we have defined in the conf.py file. If crossed, we print a message saying that the threshold has been breached, and construct a message to be sent and invokes the function to send the message through Telegram. We also send an anomaly warning if the sensor data crosses the bounds given by Z-score analysis. The program waits for 10 seconds and then continues to the next iteration of the while loop.

Figures

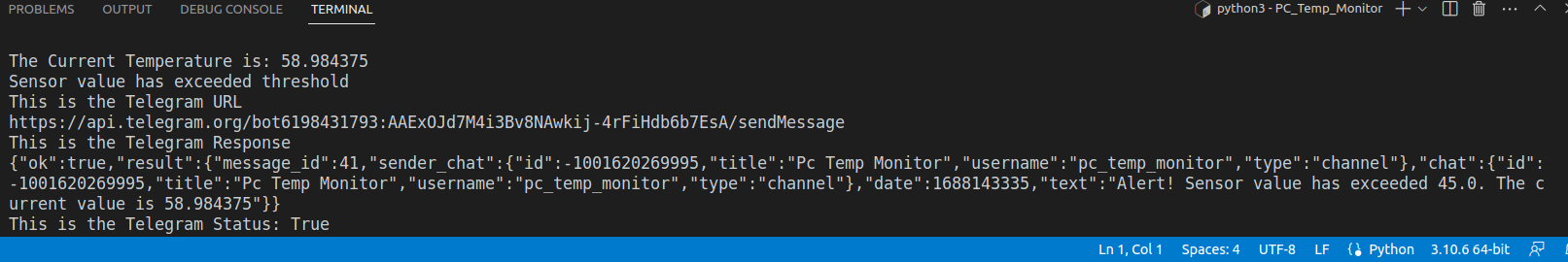
The Hardware with connections is shown



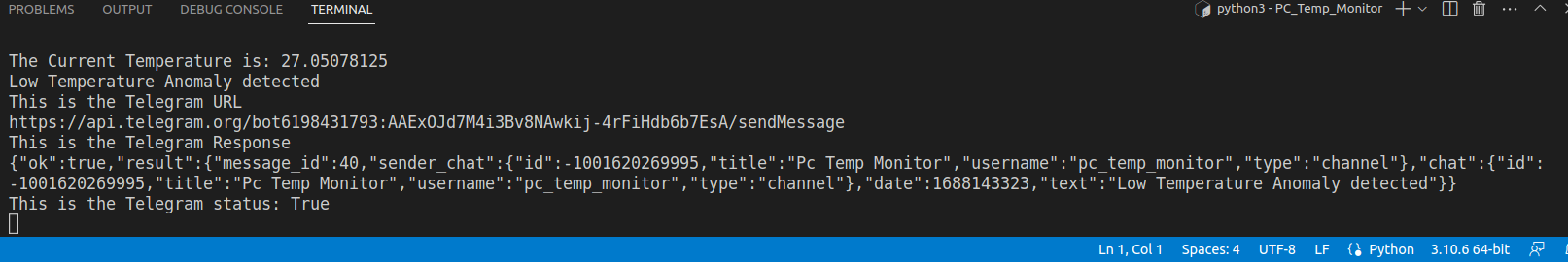
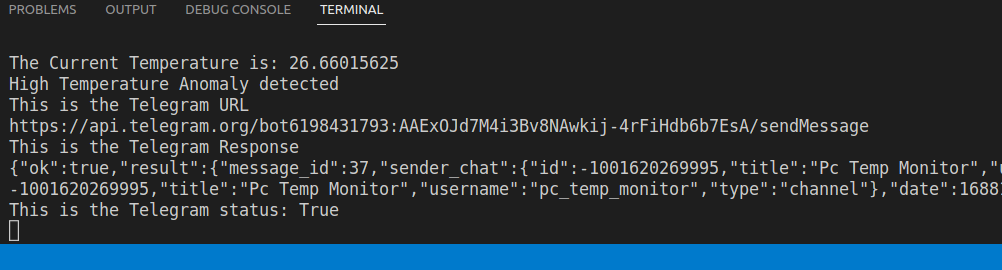
Running the program



Shows the event of temperature crossing threshold

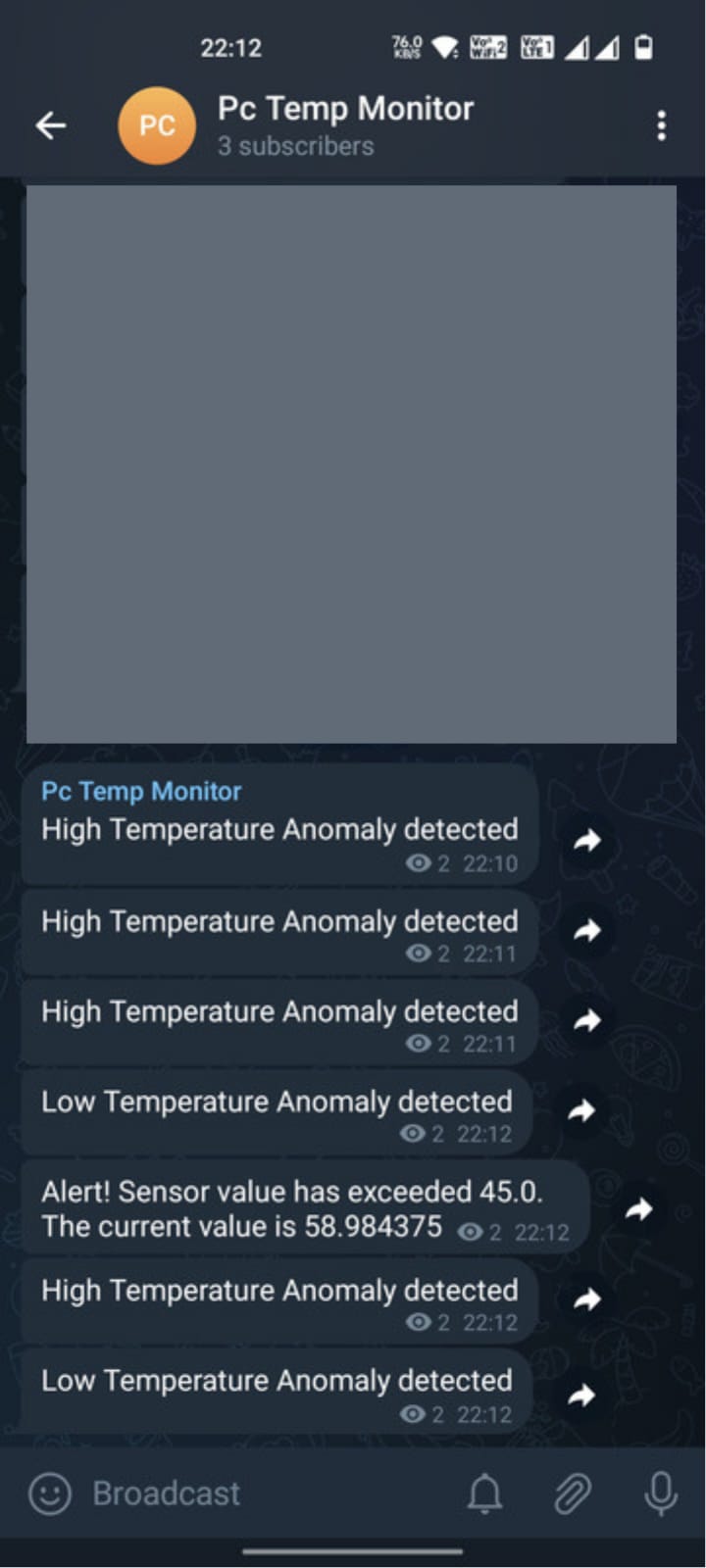


Shows the event of high temperature anomaly

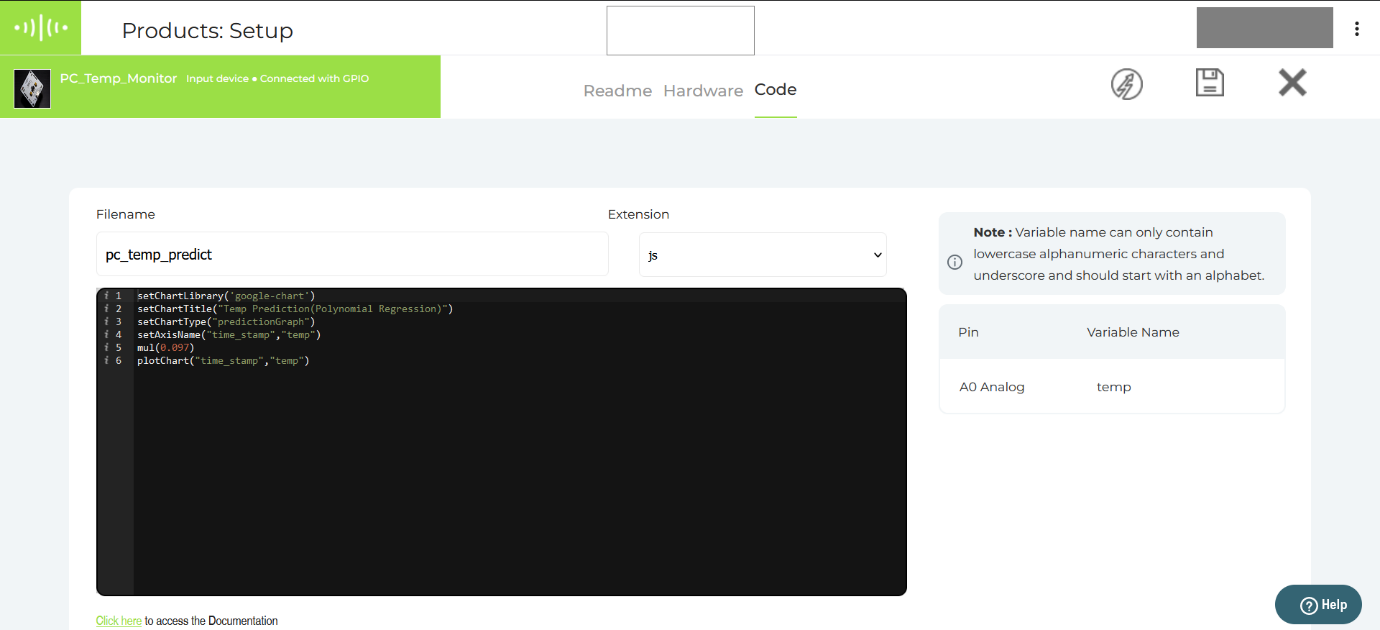


Shows the event of low temperature anomaly

The Telegram warning messages are shown



Product setup in Bolt cloud with code for plotting data



Data Plot with value prediction using Polynomial Regression (Bolt Cloud)

