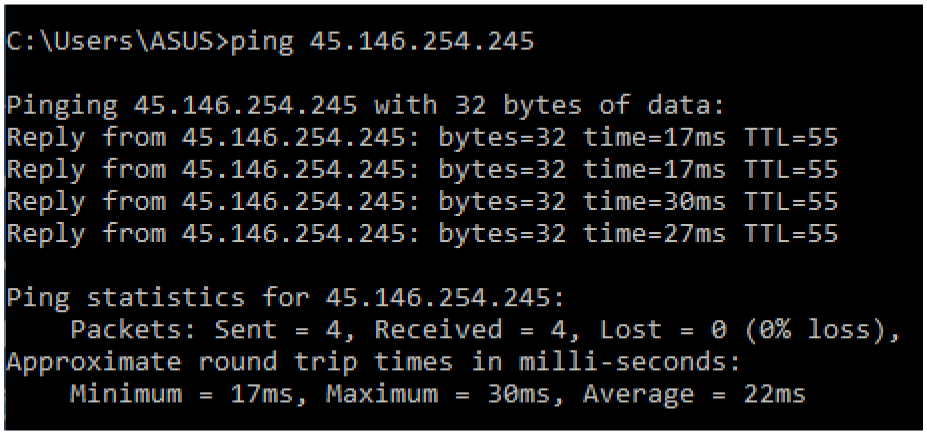
Testing the System

1. Connectivity testing:

This testing is done to check network connectivity performance between two endpoints in the network. The round trip time(RTT) is found less than 300ms which is considered as a considerable RTT while connectivity testing. Here, RTT is denoted as time it takes for a packet to travel from a user(customer) to server (ThingsBoard server) and back to user(customer). In addition, RTT encompasses queueing delay, propagation delay and processing delay. Therefore, after proper network connection, the presence of telemetry data is detected. When there is no sudden connection, data gets restored in database and latest telemetry is shown in dashboard. However, it is ensured that all the devices integrated with internet of things testing are able to register to the network. By applying ping command, RTT is evaluated as below:



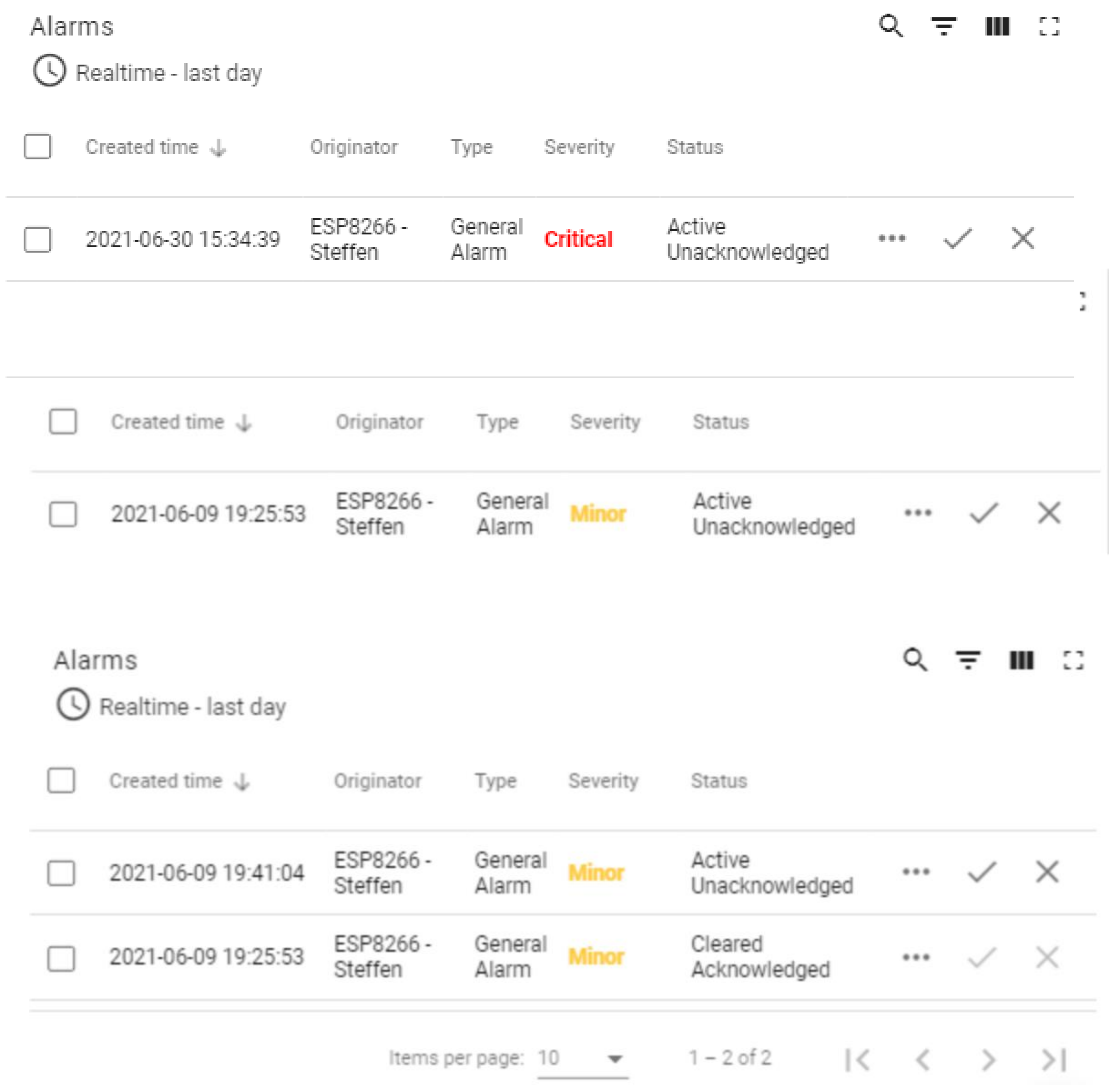
```
C:\Users\ASUS>ping 45.146.254.245

Pinging 45.146.254.245 with 32 bytes of data:
Reply from 45.146.254.245: bytes=32 time=17ms TTL=55
Reply from 45.146.254.245: bytes=32 time=17ms TTL=55
Reply from 45.146.254.245: bytes=32 time=30ms TTL=55
Reply from 45.146.254.245: bytes=32 time=27ms TTL=55

Ping statistics for 45.146.254.245:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 17ms, Maximum = 30ms, Average = 22ms
```

2. Functionality testing:

For functional testing, some input values have been changed to observe respective expected outputs to verify if the system is working perfectly or not. In rule engine, the values in scripting are modified to get corresponding outputs which gives errors also in case of inappropriate inputs. Here, it is acknowledged that when scripting function as well as heat index is changed with critical value from 40 to 10, a "minor" severity is showing as output alarm as presumed for providing low input value. Consequently, when heat index value is set to 42 as critical value by adding another node in rule engine, another new alarm is generated which named as "critical". Therefore, the rule engine functions are working properly with any changing values by delivering distinct outputs.


Alarms

Realtime - last day

	Created time	Originator	Type	Severity	Status
	2021-06-30 15:34:39	ESP8266 - Steffen	General Alarm	Critical	Active Unacknowledged

	Created time	Originator	Type	Severity	Status
	2021-06-09 19:25:53	ESP8266 - Steffen	General Alarm	Minor	Active Unacknowledged

Alarms

Realtime - last day

	Created time	Originator	Type	Severity	Status
	2021-06-09 19:41:04	ESP8266 - Steffen	General Alarm	Minor	Active Unacknowledged
	2021-06-09 19:25:53	ESP8266 - Steffen	General Alarm	Minor	Cleared Acknowledged

Items per page: 10 1 – 2 of 2

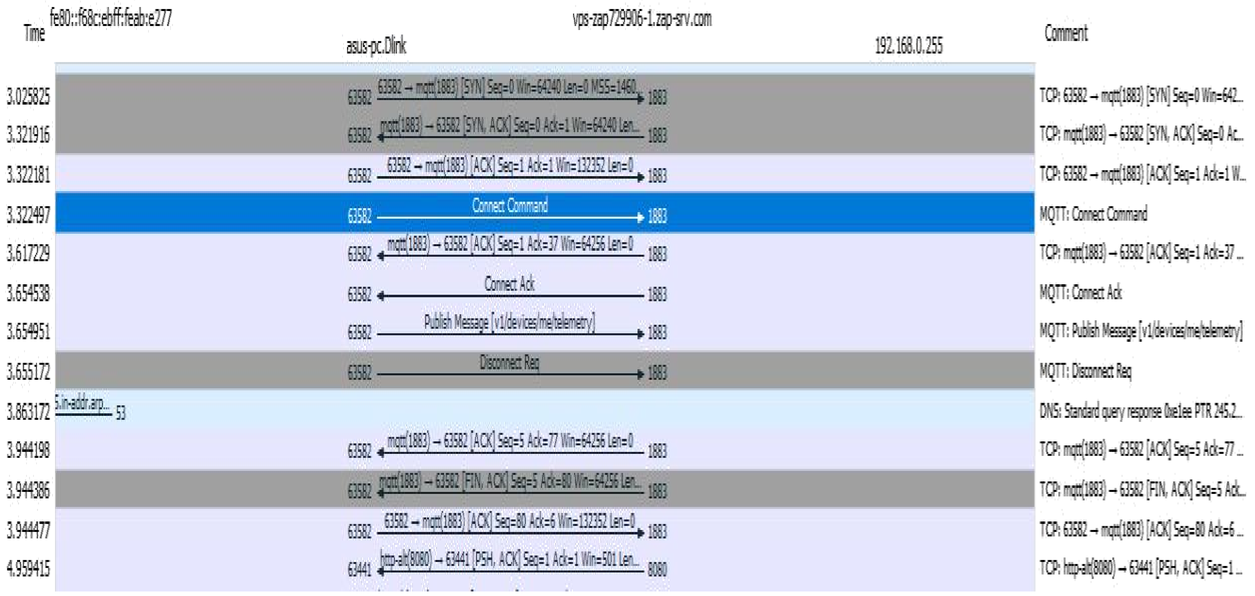
Figure 7: Dashboard changes with Function changing test

3. Protocol testing:

For devices to be connected with ThingsBoard, transport protocols are used such as MQTT, HTTP which flow back and forth between devices and ThingsBoard Web UI (Rule Engine). This Rule Engine is connected with other core services such as with devices and their credentials, Rule Chains and Rule Nodes, tenants and customers, widgets and dashboard and alarms and events. So if transport protocols MQTT, HTTP do not transfer data, there will be no output data or visualization from ThingsBoard platform. Here, MQTT protocol is transferring data by publish-subscribe method by applying following command as:

mosquitto_pub -h 45.146.254.245 -u l5mC2Vds66TcEIwyx27v -t v1/devices/me/telemetry -m "{temparature":45}"

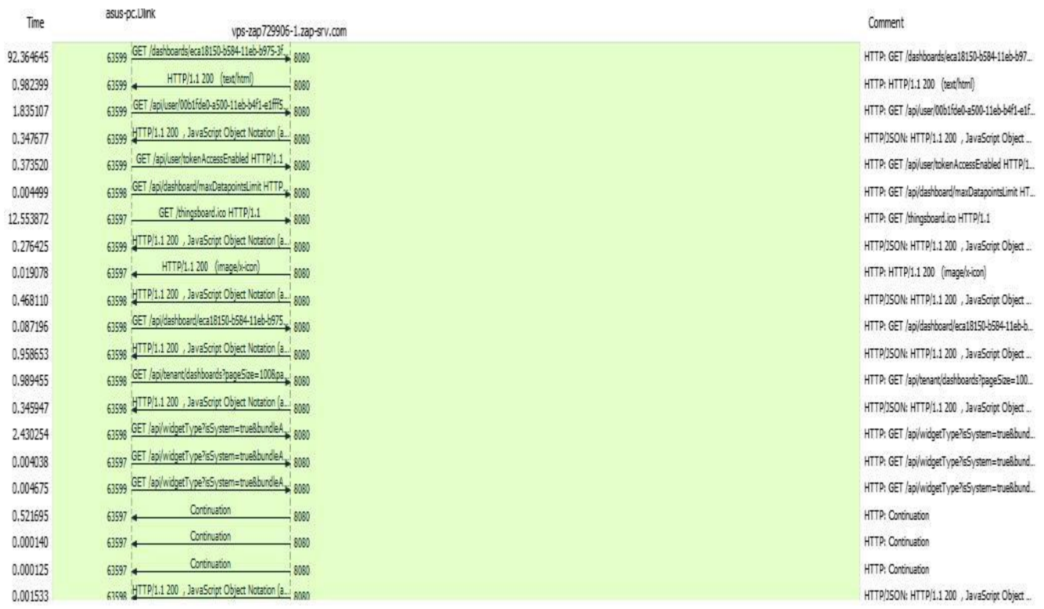
mosquitto_pub -h 45.146.254.245 -u l5mC2Vds66TcEIwyx27v -t v1/devices/me/telemetry -m "{humidity":30}"



Time	fe80::f68c:ebff:feab:e277	asus-pc.Dlink	vps-zap729906-1.zap-srv.com	192.168.0.255	Comment
3.025825		63582	63582 → mqtt(1883) [SYN] Seq=0 Win=64240 Len=0 MSS=1460 → 1883		TCP: 63582 → mqtt(1883) [SYN] Seq=0 Win=642...
3.321916		63582	mqtt(1883) → 63582 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len... ← 1883		TCP: mqtt(1883) → 63582 [SYN, ACK] Seq=0 Ac...
3.322181		63582	63582 → mqtt(1883) [ACK] Seq=1 Ack=1 Win=132352 Len=0 → 1883		TCP: 63582 → mqtt(1883) [ACK] Seq=1 Ack=1 W...
3.322497		63582	Connect Command → 1883		MQTT: Connect Command
3.617229		63582	mqtt(1883) → 63582 [ACK] Seq=1 Ack=37 Win=64256 Len=0 ← 1883		TCP: mqtt(1883) → 63582 [ACK] Seq=1 Ack=37 ...
3.654538		63582	Connect Ack ← 1883		MQTT: Connect Ack
3.654951		63582	Publish Message [v1/devices/me/telemetry] → 1883		MQTT: Publish Message [v1/devices/me/telemetry]
3.655172		63582	Disconnect Req → 1883		MQTT: Disconnect Req
3.863172	5.in-addr.arp... 53				DNS: Standard query response 0xe1ee PTR 245.2...
3.944198		63582	mqtt(1883) → 63582 [ACK] Seq=5 Ack=77 Win=64256 Len=0 ← 1883		TCP: mqtt(1883) → 63582 [ACK] Seq=5 Ack=77 ...
3.944386		63582	mqtt(1883) → 63582 [FIN, ACK] Seq=5 Ack=80 Win=64256 Len... ← 1883		TCP: mqtt(1883) → 63582 [FIN, ACK] Seq=5 Ack...
3.944477		63582	63582 → mqtt(1883) [ACK] Seq=80 Ack=6 Win=132352 Len=0 → 1883		TCP: 63582 → mqtt(1883) [ACK] Seq=80 Ack=6 ...
4.959415		63441	http-alt(8080) → 63441 [PSH, ACK] Seq=1 Ack=1 Win=501 Len... ← 8080		TCP: http-alt(8080) → 63441 [PSH, ACK] Seq=1 ...

Figure 8: *IoT-Client to MQTT Broker*

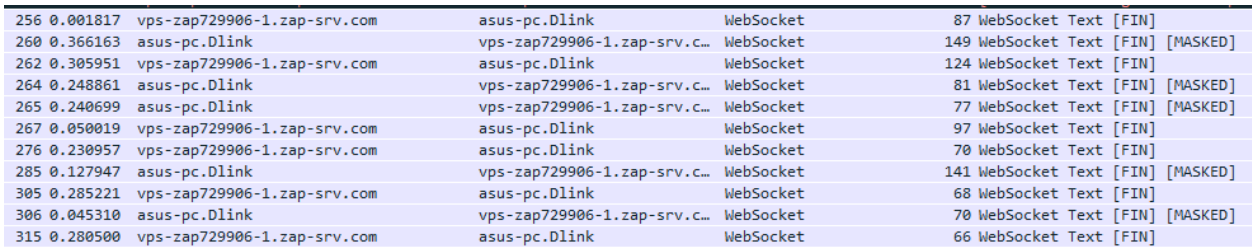
Fig:8 shows the flow graph where after the TCP connection setup, the MQTT connection command and MQTT messages are transmitted. When commands like above are applied, data gets published in MQTT Broker which is in built in 45.146.254.245 (ThingsBoard server). Then customers get accessed to this new published data from their respective dashboards. Whereas, there is HTTP protocol which is the basic protocol for transportation of data in network between client and server. The flow graph of this protocol in *Fig:9* is shown below:



```
Time        asus-pc.Dlink                                                      Comment
                    vps-zap729906-1.zap-srv.com
92.364645   63599  GET /dashboards/eca18150-b584-11eb-b975-3f...      8080   HTTP: GET /dashboards/eca18150-b584-11eb-b97...
0.982399    63599  HTTP/1.1 200  (text/html)                          8080   HTTP: HTTP/1.1 200  (text/html)
1.835107    63599  GET /api/user/00b1fde0-a500-11eb-b4f1-e1ff5...     8080   HTTP: GET /api/user/00b1fde0-a500-11eb-b4f1-e1f...
0.347677    63599  HTTP/1.1 200 , JavaScript Object Notation (a...    8080   HTTP/JSON: HTTP/1.1 200 , JavaScript Object ...
0.373520    63599  GET /api/user/tokenAccessEnabled HTTP/1.1          8080   HTTP: GET /api/user/tokenAccessEnabled HTTP/1...
0.004499    63598  GET /api/dashboard/maxDatapointsLimit HTTP...      8080   HTTP: GET /api/dashboard/maxDatapointsLimit HT...
12.553872   63597  GET /thingsboard.ico HTTP/1.1                      8080   HTTP: GET /thingsboard.ico HTTP/1.1
0.276425    63599  HTTP/1.1 200 , JavaScript Object Notation (a...    8080   HTTP/JSON: HTTP/1.1 200 , JavaScript Object ...
0.019078    63597  HTTP/1.1 200  (image/x-icon)                       8080   HTTP: HTTP/1.1 200  (image/x-icon)
0.468110    63598  HTTP/1.1 200 , JavaScript Object Notation (a...    8080   HTTP/JSON: HTTP/1.1 200 , JavaScript Object ...
0.087196    63598  GET /api/dashboard/eca18150-b584-11eb-b975...      8080   HTTP: GET /api/dashboard/eca18150-b584-11eb-b...
0.958653    63598  HTTP/1.1 200 , JavaScript Object Notation (a...    8080   HTTP/JSON: HTTP/1.1 200 , JavaScript Object ...
0.989455    63598  GET /api/tenant/dashboards?pageSize=100&pa...      8080   HTTP: GET /api/tenant/dashboards?pageSize=100...
0.345947    63598  HTTP/1.1 200 , JavaScript Object Notation (a...    8080   HTTP/JSON: HTTP/1.1 200 , JavaScript Object ...
2.430254    63598  GET /api/widgetType?isSystem=true&bundleA...       8080   HTTP: GET /api/widgetType?isSystem=true&bund...
0.004038    63597  GET /api/widgetType?isSystem=true&bundleA...       8080   HTTP: GET /api/widgetType?isSystem=true&bund...
0.004675    63599  GET /api/widgetType?isSystem=true&bundleA...       8080   HTTP: GET /api/widgetType?isSystem=true&bund...
0.521695    63597  Continuation                                       8080   HTTP: Continuation
0.000140    63597  Continuation                                       8080   HTTP: Continuation
0.000125    63597  Continuation                                       8080   HTTP: Continuation
0.001533    63598  HTTP/1.1 200 , JavaScript Object Notation (a...    8080   HTTP/JSON: HTTP/1.1 200 , JavaScript Object ...
```

Figure 9: IoT-Client to MQTT Broker

Furthermore, there is another protocol named WebSocket which is responsible for visualization of real-time analytics or time-series table which has collected data from devices using different protocols HTTP, MQTT and has stored those time-series data in Cassandra database as shown below:



```
256 0.001817  vps-zap729906-1.zap-srv.com   asus-pc.Dlink                 WebSocket    87 WebSocket Text [FIN]
260 0.366163  asus-pc.Dlink                 vps-zap729906-1.zap-srv.c…    WebSocket   149 WebSocket Text [FIN] [MASKED]
262 0.305951  vps-zap729906-1.zap-srv.com   asus-pc.Dlink                 WebSocket   124 WebSocket Text [FIN]
264 0.248861  asus-pc.Dlink                 vps-zap729906-1.zap-srv.c…    WebSocket    81 WebSocket Text [FIN] [MASKED]
265 0.240699  asus-pc.Dlink                 vps-zap729906-1.zap-srv.c…    WebSocket    77 WebSocket Text [FIN] [MASKED]
267 0.050019  vps-zap729906-1.zap-srv.com   asus-pc.Dlink                 WebSocket    97 WebSocket Text [FIN]
276 0.230957  vps-zap729906-1.zap-srv.com   asus-pc.Dlink                 WebSocket    70 WebSocket Text [FIN]
285 0.127947  asus-pc.Dlink                 vps-zap729906-1.zap-srv.c…    WebSocket   141 WebSocket Text [FIN] [MASKED]
305 0.285221  vps-zap729906-1.zap-srv.com   asus-pc.Dlink                 WebSocket    68 WebSocket Text [FIN]
306 0.045310  asus-pc.Dlink                 vps-zap729906-1.zap-srv.c…    WebSocket    70 WebSocket Text [FIN] [MASKED]
315 0.280500  vps-zap729906-1.zap-srv.com   asus-pc.Dlink                 WebSocket    66 WebSocket Text [FIN]
```

Figure 10: Dashboard Visualization for WebSocket protocol

4. Regression testing:

This test assured that any changes to the rule engine (new function implementation, code changes, debug mode) has not distorted the prevailed functions or has not created any kind of instabilities in ThingsBoard platform. After changing input functions in rule engine, the ThingsBoard platform always shows stable output constantly regardless of transformations. For instance, in sunset/sunrise rule engine, if boolean value is changed from true to false, the light from pin 40 will be switched on as per false command which was previously set to true as shown in *Fig 11 and 12*:

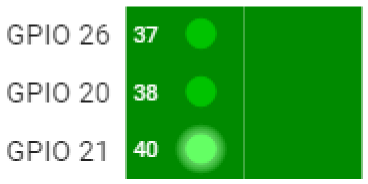


Figure 11: Dashboard Visualization (switch on) for false condition under daylight time

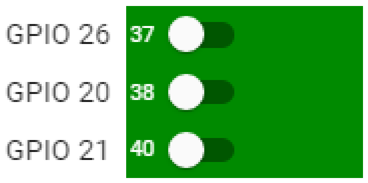
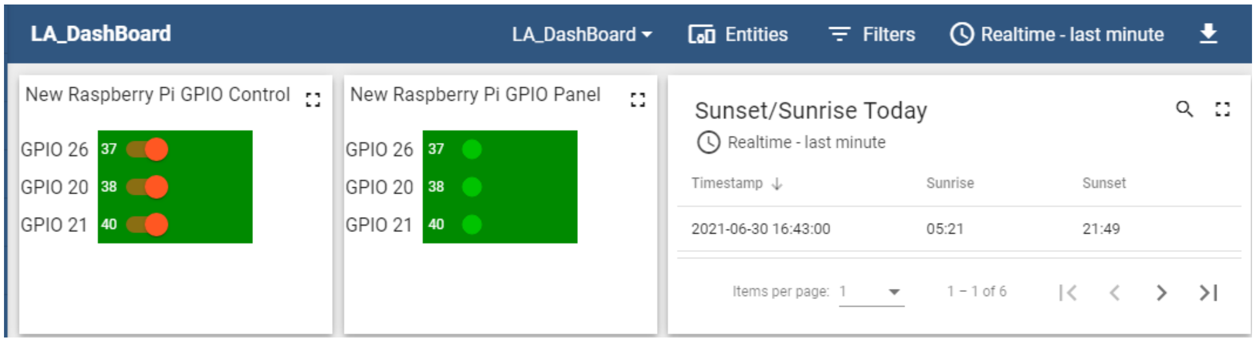


Figure 12: Dashboard Visualization (switch off) for true condition under daylight time

As seen above, the visualization output is working according to admin inputs which is working reversely. Previously, it was set to switch OFF the lights while it's the daylight time from 5:21 A.M. to 21.48 P.M and was defined to set the lights ON when it crosses 21.49 P.M. as per the data get from rest api calls as shown below:



Timestamp	Sunrise	Sunset
2021-06-30 16:43:00	05:21	21:49

Figure 13: Real dashboard visualization under daylight time

5.Usability testing:

Usability testing is done to gauge user satisfaction with the product interface by the customers. After creation of final product, customers are assigned with defined credentials with which they can access ThingsBoard platform according to their need to visualize and observe so that they can take actions accordingly. Also after having users' feedback, required improvements can be implemented through scripting and coding.

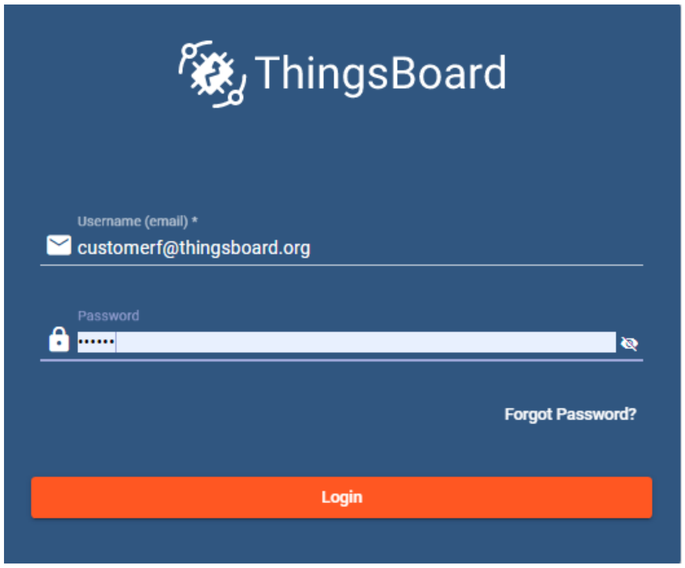
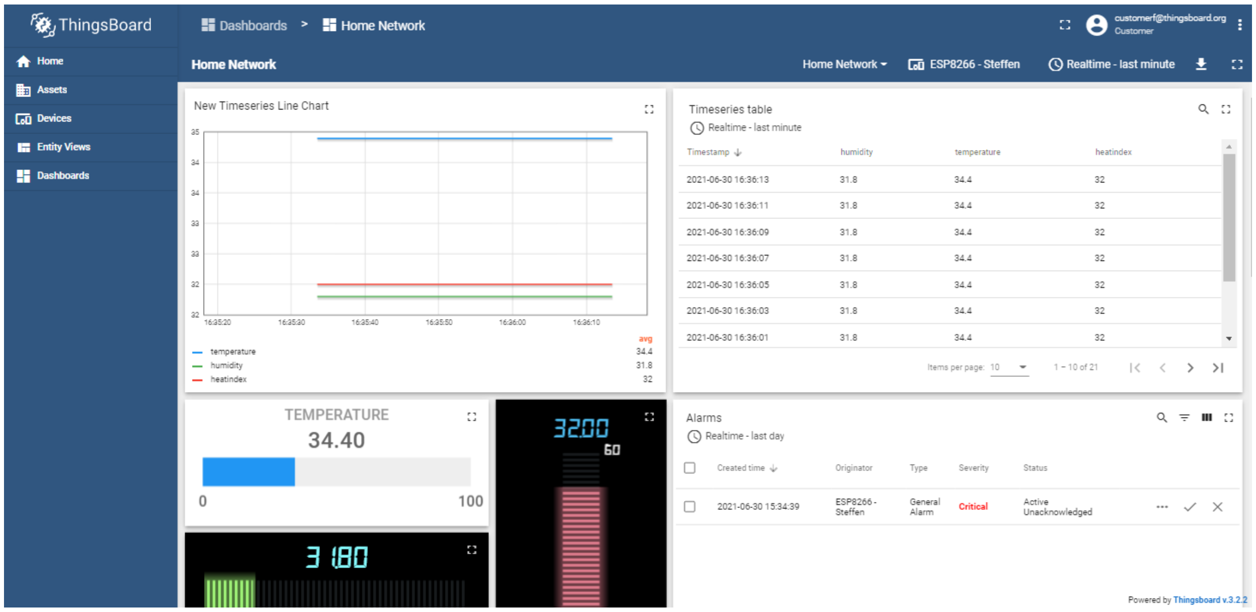


Figure 14: Customer login interface with activation link



Timeseries table
Realtime - last minute

Timestamp	humidity	temperature	heatindex
2021-06-30 16:36:13	31.8	34.4	32
2021-06-30 16:36:11	31.8	34.4	32
2021-06-30 16:36:09	31.8	34.4	32
2021-06-30 16:36:07	31.8	34.4	32
2021-06-30 16:36:05	31.8	34.4	32
2021-06-30 16:36:03	31.8	34.4	32
2021-06-30 16:36:01	31.8	34.4	32

Items per page: 10 1 – 10 of 21

Alarms
Realtime - last day

Created time	Originator	Type	Severity	Status
2021-06-30 15:34:39	ESP8266 - Steffen	General Alarm	Critical	Active Unacknowledged

Figure 15: Customer dashboard interface same as admin

Observations:

In this project, we have learnt and studied about construction of a smart home IoT network by the help of ThingsBoard (open source IoT Platform) using MQTT, HTTP protocols. The system design has been implemented on Raspberry Pi 4, ESP8266 and DTH21 sensors accordingly by obtaining real-time data visualization, accuracy and responsiveness. The goal of the project was acknowledged having some definite outputs except some issues which are presented as challenges in community edition in this platform. Furthermore, the home network system was implemented with two individual scenarios by applying and scripting proper rule engines in ThingsBoard platform. In addition, by the usage of MQTT, HTTP, WebSocket protocols, the data fetching and visualization were successful. Therefore, the smart home environment shows expected outputs maintaining user accessibility and proper functioning with data storage facility through Cassandra database.

For the future of this project, several integrations from professional edition should be examined and upgraded.