SOFE 4610 Fall 2021 - Design & Analysis of IoT

Project Title: Temperature-based Smart Fan

Project Group No: 11

Group Members:

Esam Uddin - 100711116
Ashwin Shanmugam - 100700236
Mihir Patel - 100702168

Reference Architecture :

The Figure below is the reference architecture that we have used as a basis to design the architecture for our Temperature-based smart fan. The reference architecture provides a set of instructions that shows the recommended structure and integration of IoT components and services to arrive at a proven solution. The reference architecture is an accepted industry best practice standard typically used to choose an optimal method for integrating IoT technologies.

The figure below is a combination of class and decision diagram to demonstrate the service specific architecture interaction between the IoT system, services types, service input/output, services endpoint, services precondition and and services effects.

Figure 1.1 Reference architecture [Service Specifications]

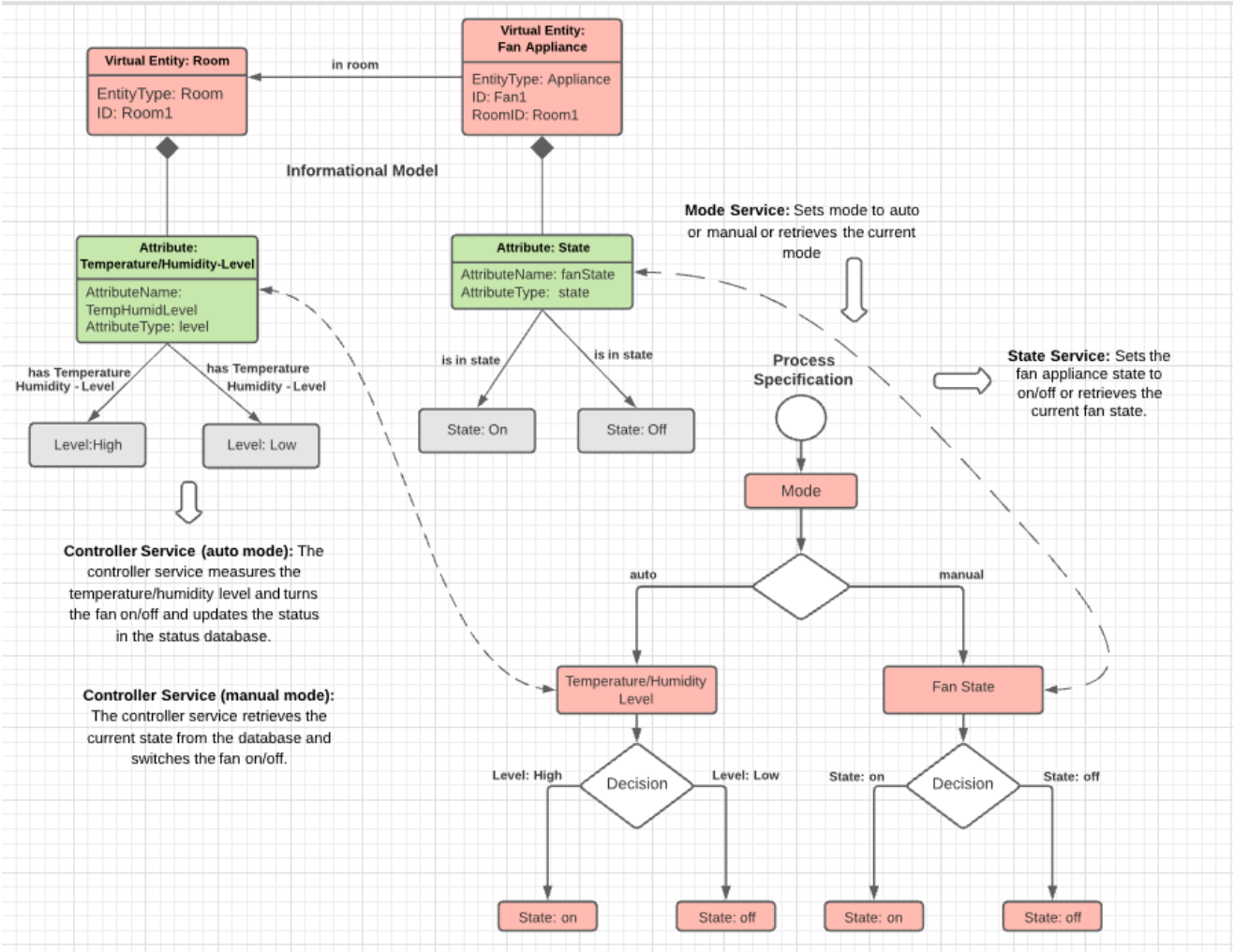
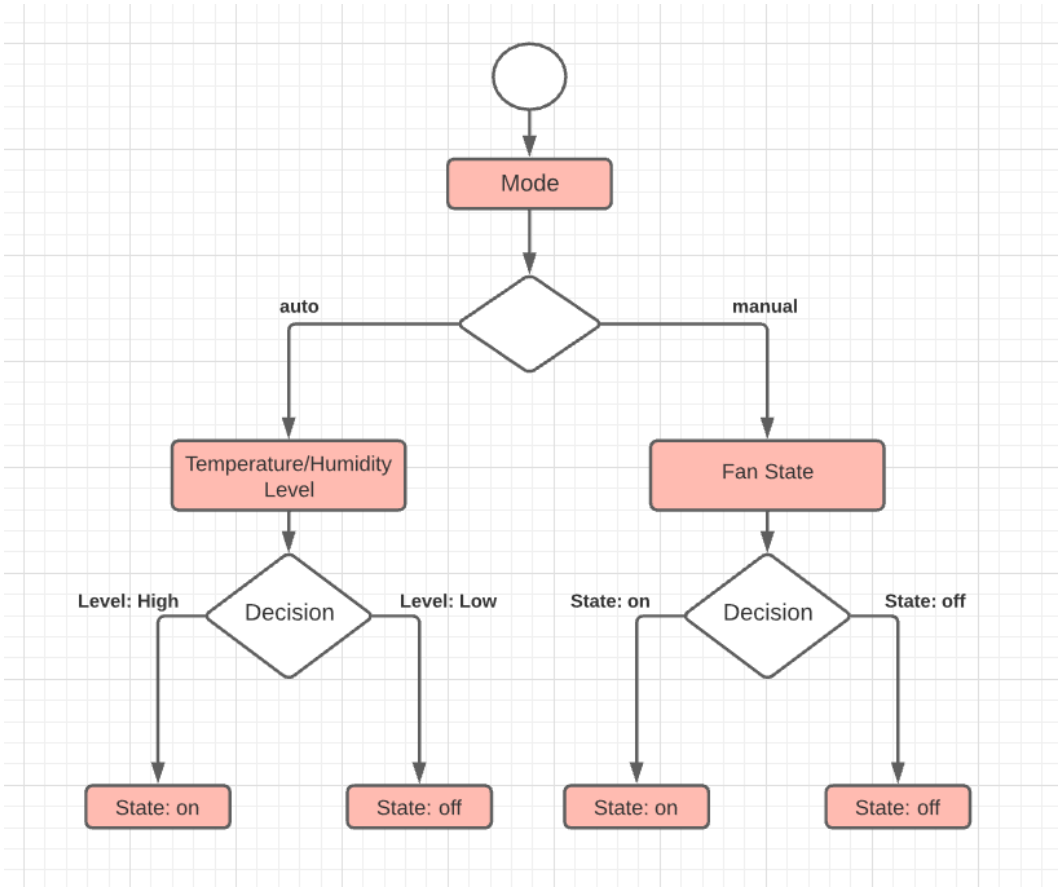


Figure 1.2 Reference architecture [Decision Diagram]

The image below is a decision diagram that describes the different state of the system and the precondition required to arrive at the given state. Initially the system starts with the auto mode to sense the room temperature to determine the state of the fan. In the case of manual mode the user can enter that through the online web application that is connected to the system to change the state of the fan.

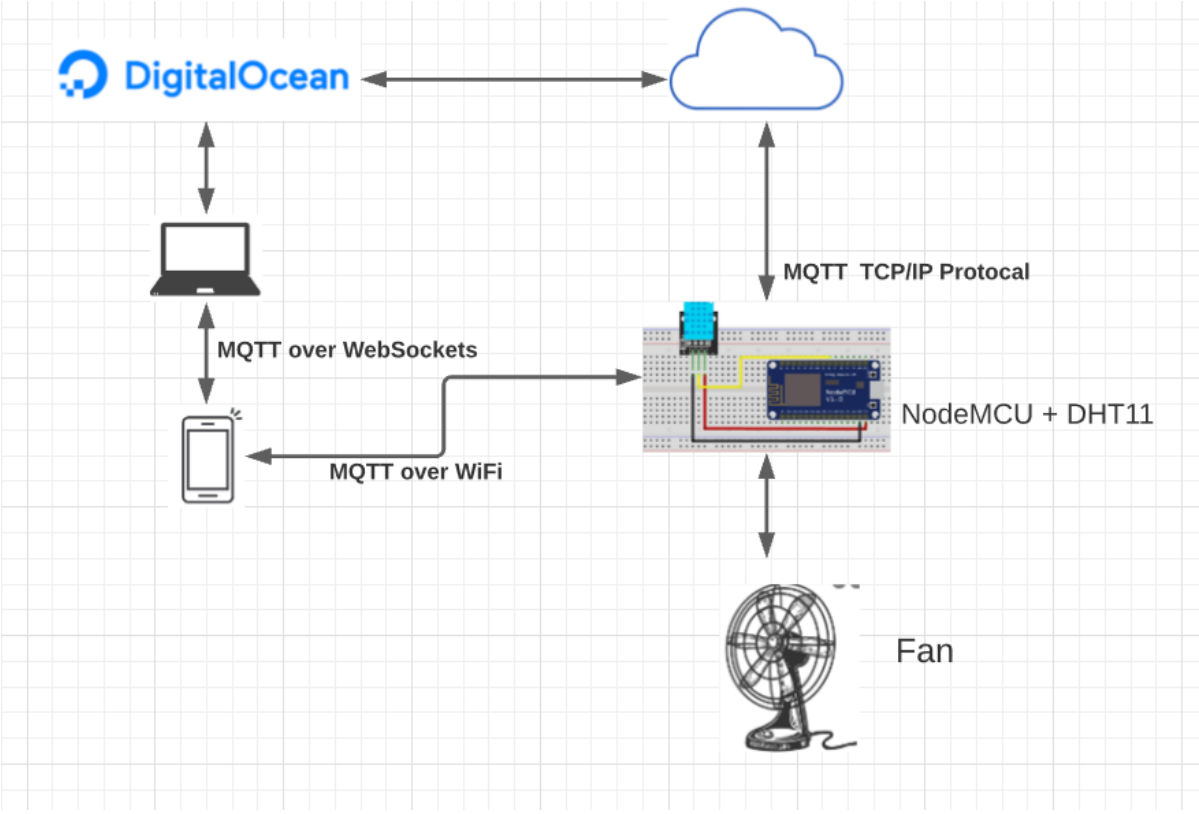


Applied Architecture:

The applied architecture with the help of a reference architecture serves as a guide for creating the actual system solution. This is also known as a subset of the reference architecture produced as an instance. Below are the screenshots of different flows created using the reference architecture.

The below architecture diagram demonstrates the interactions between different components of our application. A cloud computing vendor named Digital Ocean was used to create our cloud infrastructure to deploy our services which can be accessed through web Sockets. The Node MCU interacts with the cloud platform and other devices using the MQTT protocol.

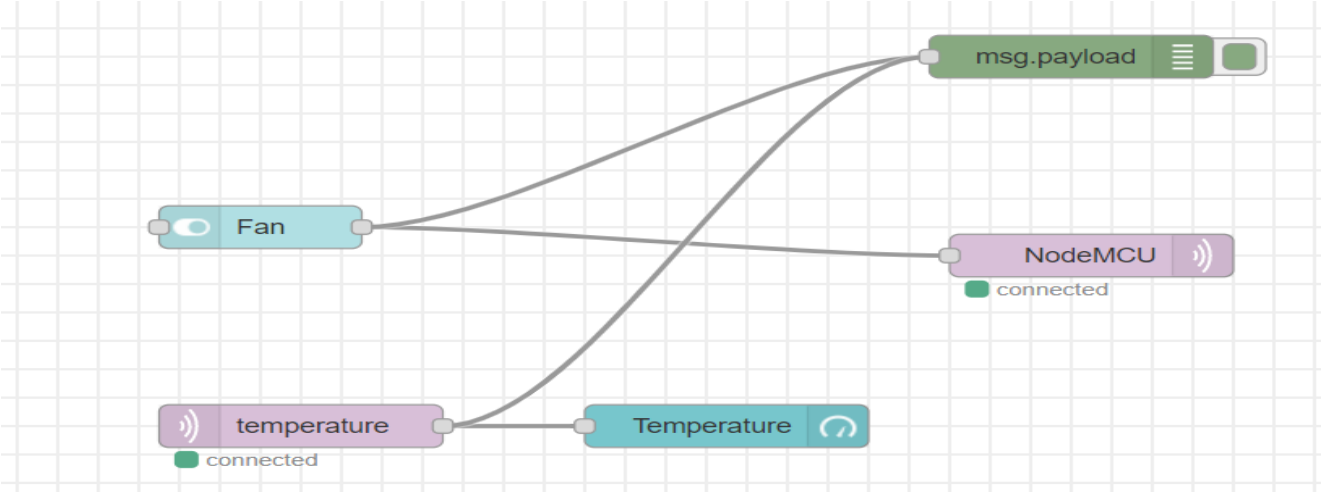
Figure 2.1



Manual Mode:

This diagram is a flow graph created using node-red implementing one of the key functionalities of the application. This flow provides the user to manually turn on/off the fan according to their convenience overriding the auto mode. The sensor data from the Node MCU is published to the topic ‘temperature’ which is subscribed by the cloud instance. A fan switch was created using a node-red dashboard to provide the user with an interface to change the fan status. Within the node-red user interface, the user is displayed with the current room temperature based on the sensor data and a switch to manually operate the fan. The user choice is sent to the NodeMCU to allow the microcontroller to act accordingly.

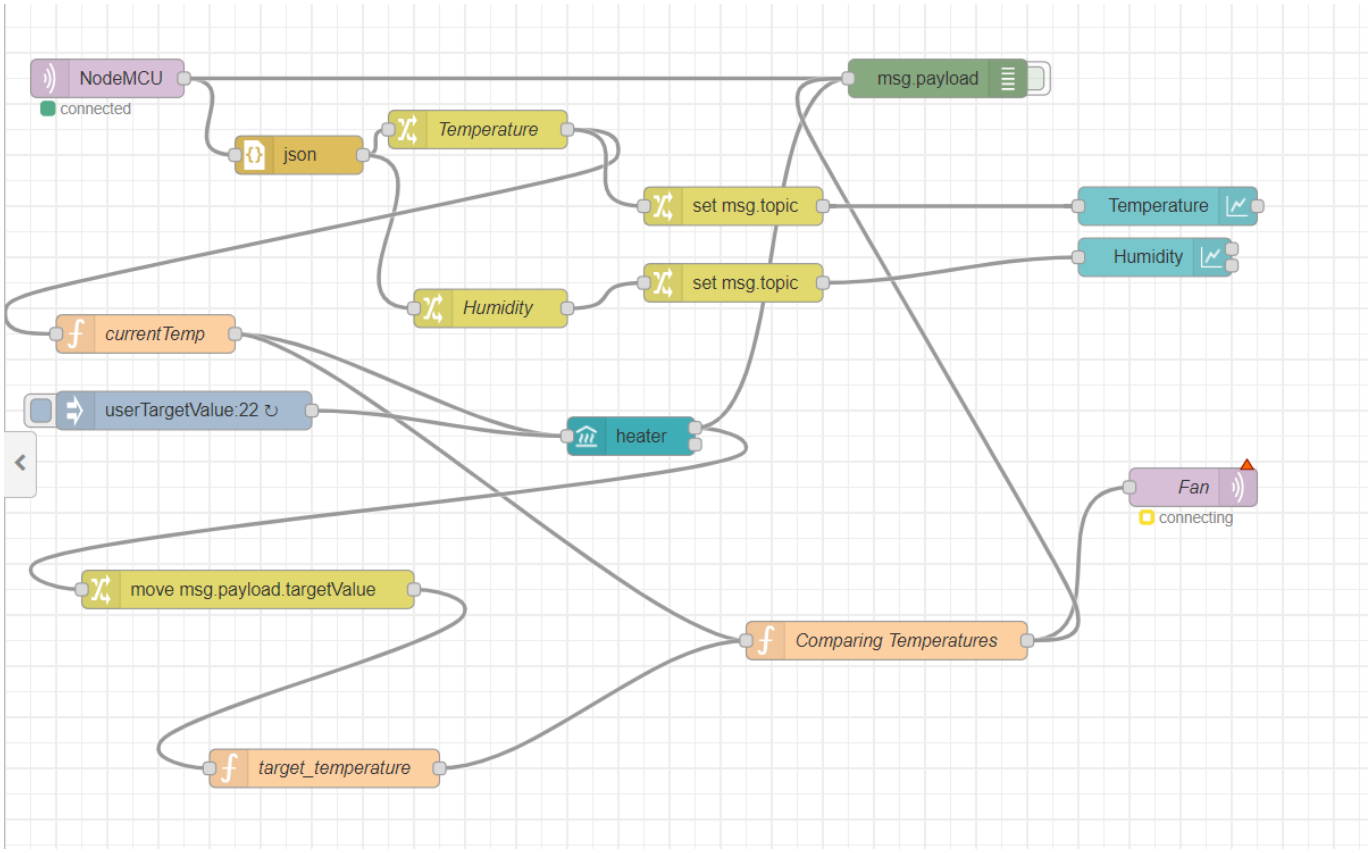
Figure 2.2



Auto Mode :

This diagram is a flow graph created using node-red implementing the automatic mode to control fan status. Firstly, the sensor data is published and subscribed through the NodeMCU, which is converted into Javascript objects to differentiate temperature and humidity sensor readings. The sensor data is then passed to the user interface charts of the node-red dashboard to display the live sensor readings on the IoT platform. The temperature reading is also passed to the heater UI interface which sends the data to the function ‘Comparing Temperatures’ where it is compared against the target temperature value of 22°C. If the current temperature was greater than target temperature, it will send a message to the NodeMCU to turn ON the fan automatically and vice versa.

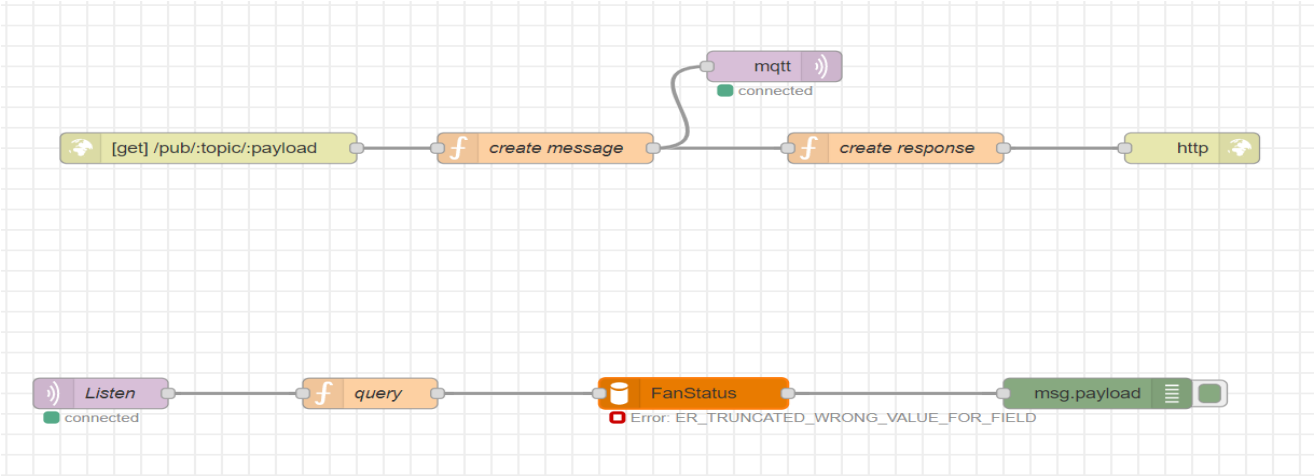
Figure 2.3



REST API :

Below flow graph demonstrates on how to create a REST API to add entries to the FanStatus database. A get request is passed to the URL of our platform. The topic and payload passed as a parameter to the URL are temperature data and fan status. This parameter information is then inserted into the database using the query function.

Figure 2.3


mqtt
connected
[get] /pub/:topic/:payload
create message
create response
http
Listen
connected
query
FanStatus
Error: ER_TRUNCATED_WRONG_VALUE_FOR_FIELD
msg.payload