IOT LAB

FUNDAMENTAL DEVELOPMENT WITH THINGWORX

STUDENT MANUAL

DesignTech
Technology for designing the future

GTTC

ptc

Internet of Things (IoT)

- In this lab we use Thing Worx for creation of Thing shapes, Thing Template and Mashup, composer to communicate to Thing over the internet.
- IOT is the networking of physical objects that contain electronics embedded within their architecture in order to communicate and sense interactions amongst each other or with respect to the external environment.
- In the upcoming years, IoT-based technology will offer advanced levels of services and practically change the way people lead their daily lives. Advancements in medicine, power, gene therapies, agriculture, smart cities, and smart homes are just a very few of the categorical examples where IoT is strongly established. Over 9 billion 'Things' (physical objects) are currently connected to the Internet, as of now.
- In the near future, this number is expected to rise to a whopping 20 billion.

Internet of Things (IoT) is a system of interconnected objects, usually called smart devices, through the Internet. The object can be a heart monitor, a remote, or an automobile with built-in sensors. That is objects that have been assigned an IP address and have the capability to collect and transfer data over a network.

FUNDAMENTAL OF IOT DEVELOPMENT WITH THINGWORX

Internet of Things (IoT) is the networking of physical objects that contain electronics embedded within their architecture in order to communicate and sense interactions amongst each other or with respect to the external environment. In the upcoming years, IoT-based technology will offer advanced levels of services and practicallychange the way people lead their daily lives. Advancements in medicine, power, gene therapies, agriculture, smart cities, and smart homes are just a very few of the categorical examples where IoT is strongly established

- **IoT Lab view and Course offered Details:**

- **The list of courses offered**

S. No	Name of the Course	Duration
1	Fundamentals of IoT Development with Thingworx	50 Hours
2	IoT Modeling With Thingworx	40 Hours

Table of Contents

Some communication devices in IoT 15
Architecture of Internet of Things (IoT): 16
Industry 4.0: 18
what is Industry 4.0? 18
Industry 4.0 and the fourth industrial revolution (4IR) 18
Sectors that can benefit the Most from IoT Development: 21
1. Agriculture 22
2. Energy 24
3. Finance 25
4. Healthcare 26
5. Manufacturing 27
6. Retail 28
7. Hospitality 30
8. Transportation and Logistics 31
What Are Some of the Most Significant Advantages of IoT? 32
1. Improve Customer Service and Experience 32
2. Increase New Business and Revenue Opportunities 32
3. Enhance Business Safety and Security 32
4. Improve Productivity and Competence 32
5. Cut Down Operational Costs 33
6. Collect Data for Business Decisions 33
7. Add Mobility and Agility to Business 33
What is ThingWorx Platform? 36
IoT-ThingWorx 37
Benefits of ThingWorx IoT Platform: 38
Components of ThingWorx IoT 38

Advantages of ThingWorx 39
Different Services Provided for ThingWorx 39
1. ThingWorx Foundation 39
2. ThingWorx Analytics 39
3. ThingWorx for Industry 40
System Requirements for installing Thingworx 8.5v 40
ThingWorx Server Requirements 40
Hardware Sizing 40
Operating System 42
Required Software 43
Database Options 44
ThingWorx Development Process 47
The Experience Stage 48
The Model Stage 49
What is Thing in Modeling? 52
Analyze, Connect, and Build Stages 53
The Deploy Stage 54
Capabilities of Thingworx 54
MASHUP CREATION 56
1. Plan Screen Layout and Functionality 56
2. Implement the Layout Using Containers and Container Widgets 56
3. Add Widgets 58
4. Bind Mashup Events and Data 59
5. Test the Mashup 61
Exercise session 61
Excerise1 61
How to LOG IN TO THINGWORX? 62

Exercise 2 63
ThingWorx Composer 63
Exercise 3 64
Thing Creation 64
Exercise 4 65
Application Key Creation 65
Exercise 5 68
Project Creation 68
Exercise 6 69
Industrial connection 69
Exercise 7 72
How to Create mashup? 72
Exercise 8 73
How to Design the Mashup? 73
2.The Model Stage; 82
Experience Complete IoT Connectivity 127
IoT connectivity isn't one size fits all 128
Connectivity in IoT is challenging—and critical to success 128
THINGWORX KEPWARE SERVER: PURPOSE-BUILT CAPABILITIES 129
Build the Data Model 132
Data Model Introduction 132
Learn how the Thing Worx Data Model makes your IoT application flexible and scalable. 132
Benefits 133
Entities: 133
Inheritance Model 134
Scenario 135

Implement Services, Events, and Subscriptions 137
Automate business processes with Services, Events and Subscriptions. 137
Create Events 138
Create Subscriptions 140
1. Operationalize an Analytics Model 142
Use Analytics Manager and Thing Predictor to automatically perform analytical calculations. 142
Analytics Architecture 143
Build Model 143
Operationalize Model 143
2. Build a Predictive Analytics Model 144
Generate predictions and gain insight into your data with machine learning 144
Generate predictions and gain insight into your data with machine learning 145
1. Application Layout (UI) 146
Convey information about IoT data effectively by customizing style definitions and implementing event-based logic 146
2. How to Display Data in Charts? 146
Graphically display multiple data points in a various chart. 146
3. Reusable Components 148
Organize Your UI with the Collection Widget 148
Use the Collection Widget to organize visual elements of your application. 148
Create a Data shape 149
First Definition 150
Second Definition 151
Third Definition 152
Create a Thing 153
Add Info Table Property 154
First Default 155

Second Default 157
Third Default 158
Basic Mashup Widgets 160
Use UI Widgets used by most Mashups 160
Create Mashup 161
Build Mashup 161
BUTTON 164
SLIDER 167
Toggle Button 171
Add Value Display: 172
GAUGE 174
Node MCU Development Board Pinout Configuration 240
Applications 241
DHT11 Sensor: 251
Working Principle of DHT11 Sensor: 251
Force Sensor: 252
Working Principle 252
Pin Name 253
Pin Configuration 253
GND 254
Ground Pin 254
VCC 254
Power Supply pin (3V to 6V) 254
CS 254
Chip Select Pin 254
INT1 254
Interrupt 1 Output 254

INT2 254
Interrupt 2 Output 254
SDO 254
Serial Data Output 254
SDA 254
Serial Data Input & Output 254
SDL 254
Serial Communication Clock 254
How to Use the ADXL345 Accelerometer Module? 255
Ultrasonic Sensor (HC-SR04) 255
Install ESP8266 Add-on in Arduino IDE 258
Installation of all Sensors Libraries 261
How to add library files for different sensors in Arduino ide platform: 261

What is IOT?

Internet of Things (IoT) is the networking of physical objects that contain electronics embedded within their architecture in order to communicate and sense interactions amongst each other or with respect to the external environment. In the upcoming years, IoT-based technology will offer advanced levels of services and practically change the way people lead their daily lives. Advancements in medicine, power, gene therapies, agriculture, smart cities, and smart homes are just a very few of the categorical examples where IoT is strongly established.
Over 9 billion 'Things' (physical objects) are currently connected to the Internet, as of now. In the near future, this number is expected to rise to a whopping 20 billion.

There are four main components used in IoT:

Low-power embedded systems: Less battery consumption, high performance arethe inverse factors that play a significant role during the design of electronic systems.

Cloud computing: Data collected through IoT devices is massive and this data has to be stored on a reliable storage server. This is where cloud computing comes into play. The data is processed and learned, giving more room for us to discover things.

Networking connection: In order to communicate, internet connectivity is a must where each physical object is represented by an IP address. However, there are only a limited number of addresses available according to the IP naming. Due to the growing number of devices, this naming system will not be feasible anymore. Therefore, researchers are looking for another alternative naming system to represent each physical object

There are two ways of building IoT:

- ✓ Form a separate internetwork including only physical objects.
- ✓ Make the Internet ever more expansive, but this requires hard-core technologies such as rigorous cloud computing and rapid big data storage (expensive).

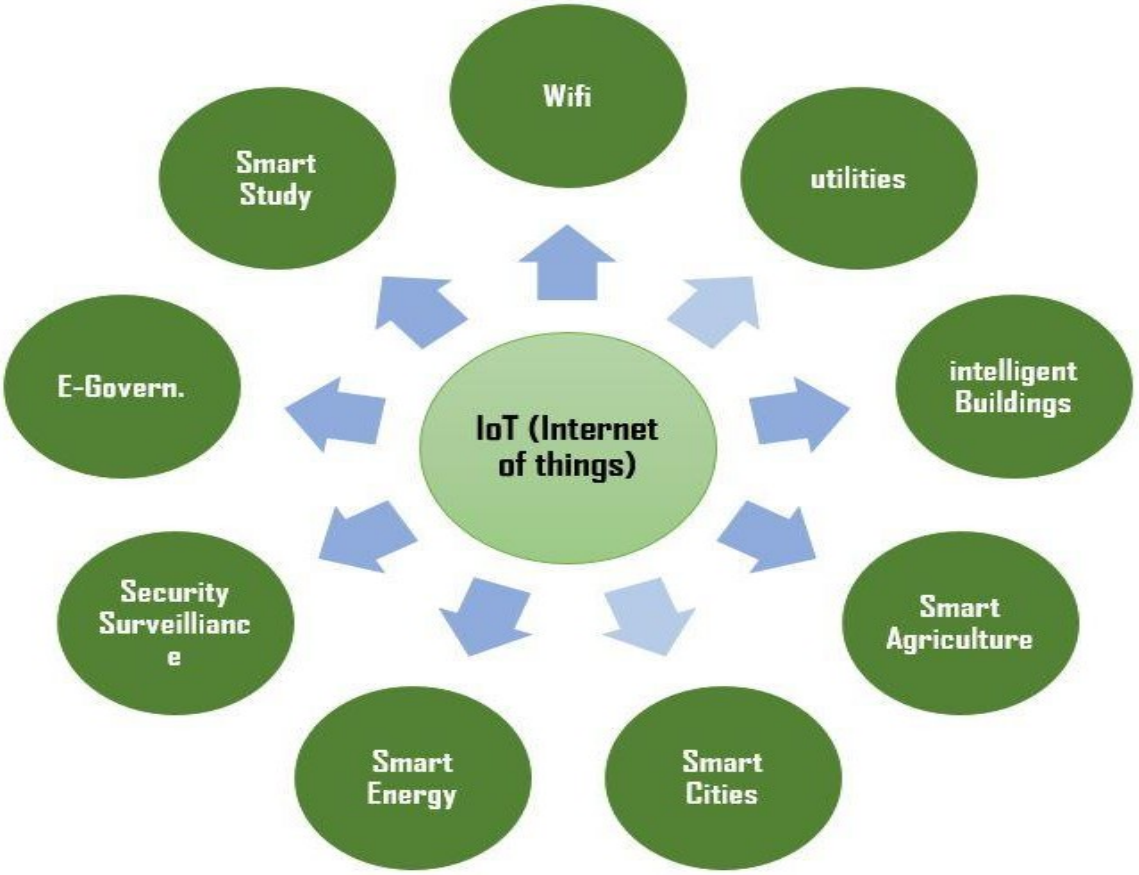
In the near future, IoT will become broader and more complex in terms of scope. It will change the world in terms of anytime, anyplace, anything in connectivity.

IoT Enablers:

- **RFIDs:** uses radio waves in order to electronically track the tags attached to each physical object.
- **Sensors:** devices that are able to detect changes in an environment (ex: motion detectors).
- **Nanotechnology:** as the name suggests, these are extremely small devices with dimensions usually less than a hundred nano meters.

Internet of Things (IoT) is a system of interconnected objects, usually called smart devices, through the Internet. The object can be a heart monitor, a remote, or an automobile with built-in sensors. That is objects that have been assigned an IP address and have the capability to collect and transfer data over a network. The objects interact with the external environment with the help of embedded technology, which helps them in taking decisions. Since these devices can now represent themselves digitally.

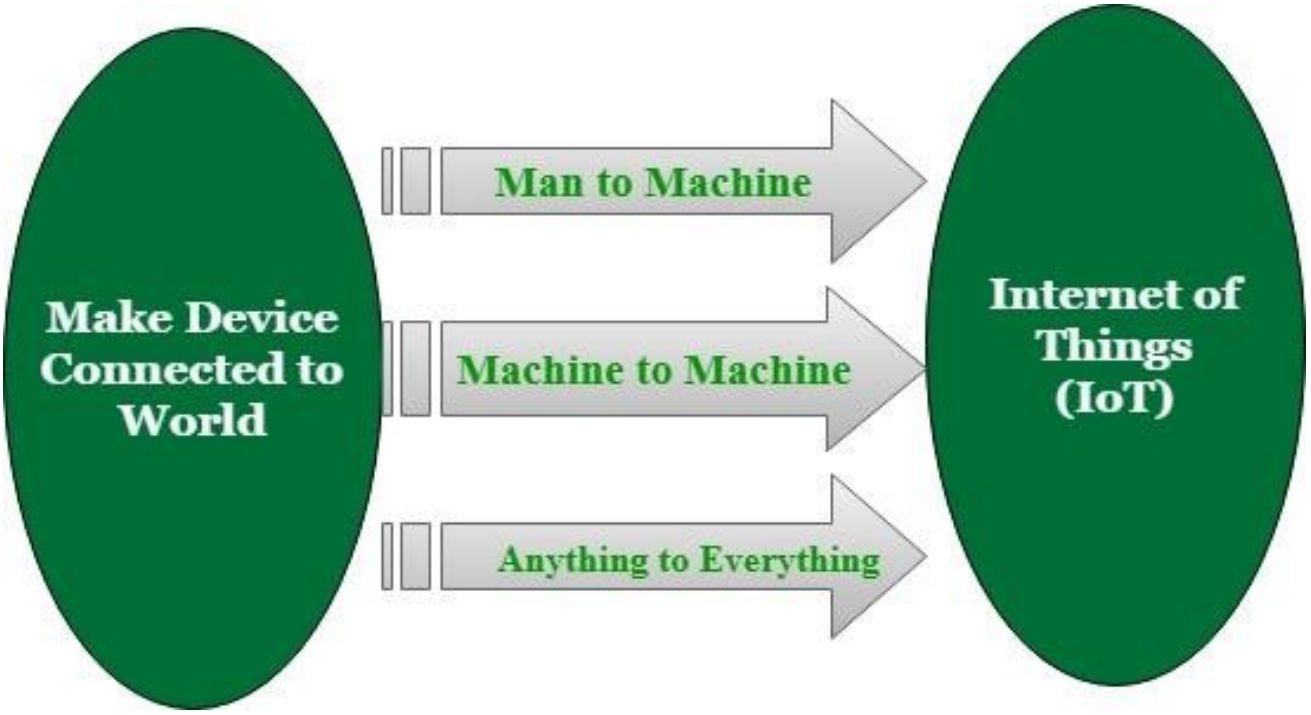
other words "The globally ruling technology acts as a single key to shrinking this whole universe to a tiny globally connected village, whereas IoT comprises just two words that precisely depict its definition.



Internet: Inter connectivity-For global connection
Things: Embedded system devices-sensors, actuators, RFID tags, QR codes and so many.

- For sensing the data
- Collecting the data
- Sending the data

Thus, on the whole, the Internet of Things is the technology that enables everything to communicate by themselves over the internet through devices without the use of computers. Here comes the most essential and prevalent term in IoTcalled **'Smart'** which means ***Automation*** – the process of decreasing human intervention or involvement thereby increasing the machine intelligence to perform every task by itself, which could be done by IoT.
IoT makes an intertwined network of artificial things like physical devices, vehicles, home appliances and even to connect with natural living beings like plants, animals, and so on.



Characteristics of IoT:

- Massively scalable and efficient
- IP-based addressing will no longer be suitable in the upcoming future.
- An abundance of physical objects is present that do not use IP, so IoT is made possible.
- Devices typically consume less power. When not in use, they should be automatically programmed to sleep.
- A device that is connected to another device right now may not be connected in another instant of time.
- Intermittent connectivity – IoT devices aren't always connected. In order to save bandwidth and battery consumption, devices will be powered off periodicallywhen not in use. Otherwise, connections might turn unreliable and thus prove to be inefficient.

As a quick note, IoT incorporates trillions of sensors, billions of smart systems, and millions of applications.

Modern Applications:

1. Smart Grids and energy saving
2. Smart cities
3. Smart homes
4. Healthcare
5. Earthquake detection
6. Radiation detection/hazardous gas detection
7. Smartphone detection
8. Water flow monitoring
9. Traffic monitoring
10. Wearables

Some communication devices in IoT:

1. Sensors: Devices that convert physical parameters like temperature, motion, etc.… into electrical signals. Smart sensors are the indispensable enablers of IoT.
Imagine a scenario of automated monitoring of a farm such that it will just indicate the current situation of crops like "4 crops need water, Now I'm going to pour it" and then it will satisfy the crop's need.
This wonder is because of the IoT technology behind it,

1. The temperature sensor connected with the plant pot detects the low temperature.
2. Then it triggers the microprocessor platforms such as Raspberry-Pi, Arduino boards.
3. It receives the sensor signals through internet pathways such as Wi-Fi, Bluetooth.
4. Then it notifies the user and the motion sensor connected to the tap which turns on to pour it.

2. Actuators: Devices which is a contrast to sensors. It transforms electrical signals into physical movements. Both sensors and actuators are transducers that convert one form of energy to another. The **exchange of data** is the most important key factor in IoT. Hence sensors and actuators play a vital role here.

3. RFID Tags: Wireless microchips are used for automatic ***unique identification*** of anything by tagging it over them. You have been seen it in credit cards, automobile. Since interconnection of things is the main goal of IoT, the RFID tags get hand-shaken with IoT technology and are used to provide the unique id for the connected "things" in IoT.

Applications of IoT:

Smart kitchen appliances: – a smart kitchen that would make the house more functional and more appealing to the people buying the house. A few of the features in the smart kitchen include: -

- **Smart plate: -** will be equipped with Wi-Fi, weight sensors, and cameras. The dish will be watching what you eat. If you overload it sends you an alert and it can sync to your personal fitness plan on your mobile.

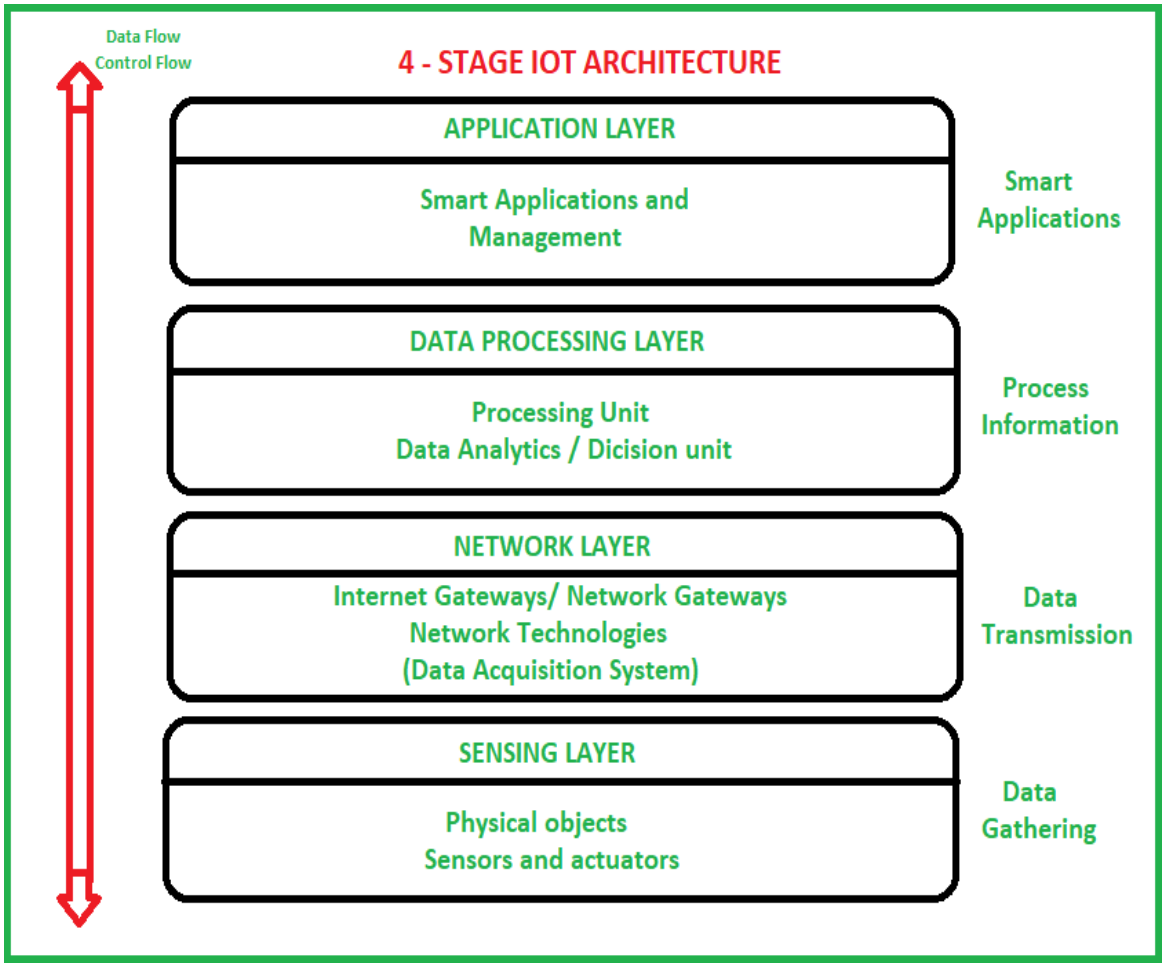
- **Drop: -** allows the selection of the dish you wish to cook and with the help of a smart scale, you can put together the recipe for your liking. Further, the recipes Drop suggests involving the use of one bowl most of the time. Meaning, it also ensures less cleaning up after cooking.

- **Smart Fridge: -** this would involve an Artificial Intelligence built into the fridge, which could communicate with other devices in the smart kitchen or in the smart home. There's a 29-inch front screen taking notes or inputting the specifics about the fridge contents.

- **Smart Cooker: -** allows you to adjust your cooking settings like cooking temperature, cook time enabling you to warm up or turn off the device irrespective of where you are – no more burnt breakfast when this comes into the market!!

Architecture of Internet of Things (IoT): -

Internet of Things (IoT) technology has a wide variety of applications and use of Internet of Things is growing so faster. Depending upon different application areas of Internet of Things, it works accordingly as per it has been designed/developed. But it has not a standard defined architecture of working which is strictly followed universally. The architecture of IoT depends upon its functionality and implementation in different sectors. Still, there is a basic process flow based on whichIoT is built.



From the above image it is clear that there is 4 layers are present that can be divided as follows: Sensing Layer, Network Layer, Data processing Layer, and Application Layer.

These are explained as following below.

1. Sensing Layer –

Sensors, actuators, devices are present in this Sensing layer. These Sensors or Actuators accepts data (physical/environmental parameters), processes data and emits data over network.

2. Network Layer –

Internet/Network gateways, Data Acquisition System (DAS) are present in this

layer. DAS performs data aggregation and conversion function (Collecting data and aggregating data then converting analog data of sensors to digital data etc). Advanced gateways which mainly opens up connection between Sensor networks and Internet also performs many basic gateway functionalities like malware protection, and filtering also sometimes decision making based on inputted data and data management services, etc.

3. Data processing Layer –

This is processing unit of IoT ecosystem. Here data is analyzed and pre-processed before sending it to data center from where data is accessed by software applications often termed as business applications where data is monitored and managed and further actions are also prepared. So here Edge IT or edge analytics comes into picture.

4. Application Layer –

This is last layer of 4 stages of IoT architecture. Data centers or cloud is management stage of data where data is managed and is used by end-user applications like agriculture, health care, aerospace, farming, defense, etc.

Industry 4.0:

What is Industry 4.0?

Industry 4.0 has been defined as “a name for the current trend of automation and data exchange in manufacturing technologies, including cyber-physical systems, the Internet of things, cloud computing and cognitive computing and creating the smart factory”.

Industry 4.0 and the fourth industrial revolution (4IR)
As a reminder the classic view of these four industrial revolutions, as Industry 4.0 became increasingly popular, was:

1. **The first industrial revolution**, which REALLY was a revolution, and, among others thanks to invention of steam machines, the usage of water and steam power and all sorts of other machines, would lead to the industrial

transformation of society with trains, mechanization of manufacturing and loads of smog.

2. **The second industrial revolution** is typically seen as the period where electricity and new manufacturing 'inventions' which it enabled, such as the assembly line, led to the area of mass production and to some extent to automation.
3. **The third industrial revolution** had everything to do with the rise of computers, computer networks (WAN, LAN, MAN,…), the rise of robotics in manufacturing, connectivity and obviously the birth of the Internet, that big game changer in the ways information is handled and shared, and the evolutions to e-anything versions of previously brick and mortar environments only, with far more automation.
4. **In the fourth industrial revolution** we move from 'just' the Internet and the client-server model to ubiquitous mobility, the bridging of digital and physical environments *(in manufacturing referred to as Cyber Physical Systems*), the convergence of IT and OT, and all the previously mentioned technologies *(Internet of Things, Big Data, cloud, etc.)* with additional accelerators such as advanced robotics and AI/cognitive which enable Industry 4.0 with automation and optimization in entirely new ways that lead to ample opportunities to innovate and truly fully automate and bring the industry to the next level.

Industry 4.0 is the **digital transformation** of manufacturing/production and related industries and value creation processes.

Industry 4.0 is used interchangeably with the fourth industrial revolution and represents a new stage in the organization and control of the industrial value chain.

Cyber-physical systems form the basis of Industry 4.0 *(e.g., 'smart machines')*. They use modern control systems, have embedded software systems and dispose of an Internet address to connect and be addressed via IoT *(the Internet of Things)*.

This way, products and means of production get networked and can 'communicate', enabling new ways of production, value creation, and real-time optimization. Cyber-physical systems create the capabilities needed for smart factories. These are the same capabilities we know from the Industrial Internet of Things like remote monitoring or track and trace, to mention two.

Industry 4.0 has been defined as "a name for the current trend of automation and data exchange in manufacturing technologies, including cyber-physical systems, the Internet of things, cloud computing and cognitive computing and creating the smart factory".

Industry 4.0 is often used interchangeably with the notion of the fourth industrial revolution. It is characterized by, among others,

- even more automation than in the third industrial revolution,
- the bridging of the physical and digital world through cyber-physical systems, enabled by Industrial IoT,
- a shift from a central industrial control system to one where smart product define the production steps,
- closed-loop data models and control systems and
- Personalization/customization of products.

The goal is to enable autonomous decision-making processes, monitor assets and processes in real-time, and enable equally real-time connected value creation networks

Enabling more direct models of personalized production, servicing, as well as customer/consumer interaction *(including gaining real-time data from actual product usage)* and cutting the inefficiencies, irrelevance and costs of intermediaries in a digital supply chain model, where possible, are some goals of Industry 4.0 in this customer-centric sense of increasingly demanding customers who value speed, *(cost)* efficiencies and value-added innovative services.

In the end, it remains business – with the innovative twist of innovation and transformation of business models and processes: increase profit, decrease costs, enhance customer experience, optimize customer lifetime value and where possible customer loyalty, sell more, and innovate to grow and remain relevant.

Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2030

The number of Internet of Things (IoT) devices worldwide is forecast to almost triple from 8.74 billion in 2020 to more than 25.4 billion IoT devices in 2030. In 2020, the highest number of IoT devices is found in China with 3.17 billion devices. IoT devices are used in all types of industry verticals and consumer markets, with the consumer segment accounting for around 60 percent of all IoT connected devices in 2020. This share is projected to stay at this level over the next ten years.

IOT Sectors and Applications

Sectors that can benefit the Most from IoT Development**: -**

As the IoT continues to unroll globally and spread across the sectors, some industries are leading investments in this revolutionary technology, changing how we live and work.

1. Agriculture

The Agriculture industry has always been very receptive to technical innovation. It has become quite technology-driven and industrialized by embracing the IoT.

The growth of IoT use in Agriculture is seen from the report by Statista, which presents the worldwide IoT market size of the Agriculture sector in 2018 – 14.79 billion USD. It is projected to reach almost 30 billion USD by 2030

The term, **Smart agriculture**, denotes the application of IoT solutions in agriculture. It uses IoT sensors to collect environmental and machine metrics. The data can help farmers make informed decisions and improve just about every area of their work – from livestock to crop farming.

Use Cases of IoT in Agriculture

Let us check few IoT use cases in agricultural processes.

Agricultural Drones

Agricultural drones are equipped with sensors and cameras. They are used for imaging, mapping, and surveying the farms. There are two types of drones: ground- based drones and aerial drones. Ground drones are bots that survey the fields on wheels.

In comparison, Aerial drones are unmanned aerial vehicles or flying robots. Drones can be remotely controlled.

Besides the surveillance capabilities, drones can perform a varying number of tasks that previously required human labour: planting crops, fighting pests, agriculture spraying, and crop monitoring, etc.

Greenhouse Automation

Farmers usually use manual intervention to control the greenhouse environment. IoT sensors help them get accurate real-time information on greenhouse conditions like lighting, temperature, soil condition, and humidity.

Monitoring climate conditions

Weather stations combined with smart farming sensors collect data from the environment and send it to the cloud for analysis. It is further used to map the climate conditions, choose the right crops, and improve their capacity.

Cattle monitoring

Just as in crop monitoring, these IoT agriculture sensors perform livestock tracking.

Increased control over the internal processes and lower the production risks

The ability to check the output of your production in advance aids in planning for improved product distribution.

If you know precisely how many crops you will harvest, you can make sure no surplus product will lie around unsold and thus control the risk.

2. Energy

Statistically speaking, the report shows that the global value of IoT in the Energy market is 20.2 billion USD in 2020. It is expected to increase to 35.2 USD by 2025

IoT has started revolutionizing most aspects of the Energy sector from generation to transmission to distribution and impacting how energy companies and customers interact.

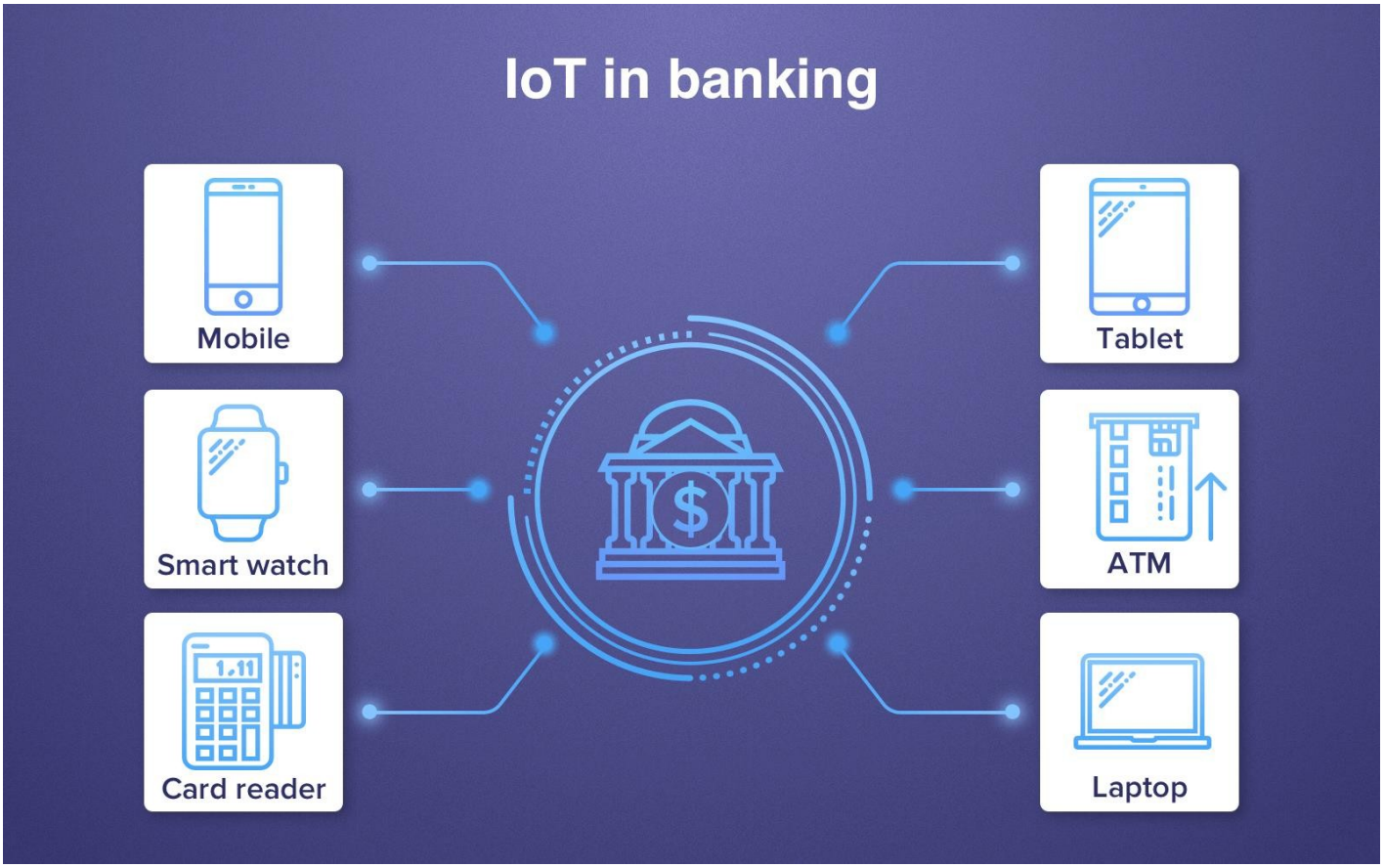
Here's what numbers from another report say about the growth of IoT in the energy industry.

- Almost **47%** of Energy sector executives indicate they have implemented IoT across functions.
- Data sources for Energy include the use of machinery **(49%)** and Robots **(46%)**.
- IoT is deployed by **45%** of Energy companies to monitor asset performance.
- IoT is used by **43%** of Energy companies to enhance customer experiences.
- **40%** of Energy companies reported an overall boost in productivity.

There are many areas where IoT can be very impactful in transforming the Energy sector:

- Remote assets monitoring and management
- Grid balancing
- Load forecasting
- Smart decision making
- Innovative power solutions

3. Finance



The IoT is becoming increasingly secure. Banks and customers have become accustomed to managing financial transactions through different connected devices.

It is not uncommon to see smart cashpoints with connected cash vending machines.

As the amount of data transferred and gathered is huge with IoT, financial businesses can measure risk accurately.

With time, banks will start using sensors and data analytics to collect a lot more information about customers and thus offer personalized services. It will help the banks understand how their customers buy and spend their money.

Implementing IoT in banking can provide more benefits:

- Connected devices can help users cultivate good financial habits and deal with any spending indulgent behaviour.
- It can improve the quality of the banking experience.
- Interactive credit card for customers.
- It has automated business processes for banks.

4. Healthcare

The Healthcare sector has started making optimum use of IoT technologies.

IoT technology in Healthcare is represented by tracking systems and real-time health systems and is responsible for improving patient treatment, diagnosis, medical and diagnostic equipment maintenance, and remote surgeries.

Connected Healthcare is necessary for correct decisions, right actions, smart treatment, and thus patient satisfaction.

Here's a list of advantages that IoT devices can provide healthcare organizations with:

- Reduced cost for patients
- Remote monitoring of patient's health
- Reduce the length of the patient's stay at the hospital
- Devices in wearable fitness bands, blood pressure, heart rate monitoring cuffs, and glucometer give patients personalized attention. These devices can also remind of calorie count, physician appointments, and exercise checks.

- IoT devices with sensors are used to track real-time locations of medical equipment
- Asset management

5. Manufacturing

The rise in IIoT or Industrial Internet of Things is the main driving force of the fourth industrial revolution.

The World Economic Forum's Centre for the Fourth Industrial Revolution has been working on some IoT initiatives in Denmark. These initiatives are producing incredible societal efficiencies.

Currently, the manufacturing industry is the most prominent investors in the Internet of things.

The leading IoT applications in manufacturing include:

- Monitoring of the Production flow leads to flow optimization, waste reduction, and the minimization of unnecessary work in process inventory.
- Remote equipment management, allowing for tracking and maintaining of equipment's performance and cost reduction.
- Condition-based maintenance notifications, helping optimize machine availability.
- Supply chains, where IoT solutions help track vehicles and assets, improve manufacturing and supply chain operations' efficiency.
-

6. Retail

The Internet of Things technology helps retailers reduce operational costs and enhance customers' experiences through new and innovative use cases.

Here are some advantages that retail stores gain by applying IoT.

- **Improved customer experience**

This is done by using data from surveillance cameras and social media, which allow retailers to accurately predict customer behavior and habits.

- **Better supply-chain management**

GPS sensors and RFID tags offer a complete journey of the good's movement from manufacturing to the store's shelf to customers.

- **Automated checkout**

When the store is crowded, checkout remains the most unpleasant process for the customers. IoT solutions can automate and personalize checkouts.

- **Fully automated stores**

They allow customers to pick up the items they wish to purchase after walking into the store.

Customers can walk out with the items, and payment is processed automatically, and the customers get a receipt on their phone. The technology relies on full camera coverage of the store, facial recognition technologies.

7. Hospitality

The hospitality industry has transformed itself with the use of IoT. The IoT technology helps deliver a personalized experience for guests and reduce their waiting time.

With the help of IoT applications, hotels can create super personalized rooms. The room's features can be controlled through the guest's mobile phone or the provided tablet. The IoT-integrated rooms would:

- Greet the guest by their name when they enter the room
- Send location data to guests for a smoother arrival
- Allow guests to control room temperature and ventilation, and save their preferences for future check-ins
- Predict repairs and maintenance of devices and alert the maintenance staff on time

There are several creative ways the hospitality industry can leverage IoT technology.

8. Transportation and Logistics

The IoT transportation market was valued at $135 billion in 2016 and is expected to reach $328 billion in 2023.

Speed and timing are two priority challenges for Transportation and logistics. Internet-of-things devices are set to make transportation companies smarter and more efficient by automation and business processes optimization.

IoT provides actionable insights on the data collected and thus aids the industry in the following aspects:

1. Cost-saving and increased profitability
2. Making travel more efficient
3. Improved operational performance
4. Reduced energy use and congestion
5. Greater safety
6. Ensures real-time visibility
7. Improves fleet management

8. Supports in the maintenance of vehicle's health
9. Warehouse management

What Are Some of the Most Significant Advantages of IoT?

Let's check out the various ways companies can benefit from using the IoT – Internet of Things and its cutting-edge technology to improve different aspects of the business.

1. Improve Customer Service and Experience

Equipment and devices that your customers use every day become extraordinary when integrated well with the IoT. Businesses should look at their products & services and check if they can be improved with IoT.

IoT-integrated cars can calculate the length of the trip, check traffic situations, and fuel consumption. A smart doorbell that can show who is at the door, even when you are not home.

Refrigerators can tell if you are low on supplies. An AC system that can correlate between the weather outside and the preferred temperature of the house owner.

2. Increase New Business and Revenue Opportunities

IoT introduces new ways by which a company can interact with customers andbusiness flow and thus open the door to new business opportunities.

3. Enhance Business Safety and Security

IoT devices can monitor buildings and offices to ensure people's safety and protect against physical threats.

By adding IoT devices integrated with sensors, video cameras, and smart locks, businesses can monitor office premises and protect assets. Employees that work in high-risk areas like manufacturing, real estate, or construction can be regularly tracked and alerted about dangers.

4. Improve Productivity and Competence

Repetitive tasks at companies can be assigned to smart IoT devices, and they sure will do a better job because they don't take a break, sleep, eat, have mood swings, and fall sick.

5. Cut Down Operational Costs

Improved productivity, process efficiency, and asset utilization will help save some expenditures.

For example, predictive analytics and real-time diagnostics can help trim down maintenance costs in various industries.

6. Collect Data for Business Decisions

With IoT, companies can measure and gather essential sets of data that can improve their services and products.

This data can provide crucial insights into business analysis and decision-making.

7. Add Mobility and Agility to Business

IoT technology allows businesses to let their staff and workers conduct their work from literally anywhere.

This flexibility can help many companies – cost saving as office leases are expensive, hiring many remote employees.

IoT Application Development Challenges

- Integration of hardware and software from several vendors
- Highly dispersed device environment
- Lack of Agility
- Technical Complexity
- Volume, velocity, and variety of IoT data
- Project requirements outweigh current resources and development tools

How an IOT Platform simplifies Application Development

A broad range of capabilities of IoT platform simplifies the application development such as:

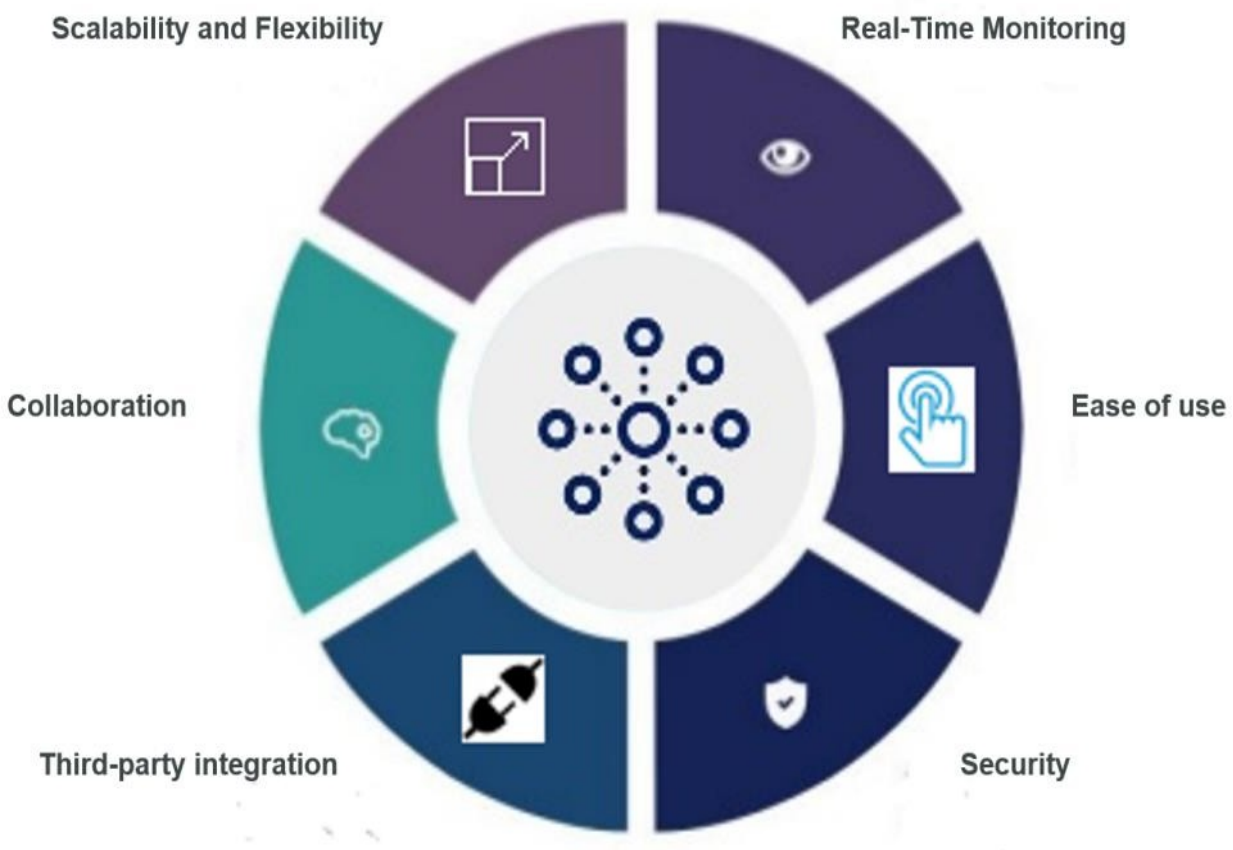
- Quickly connect any device in the ecosystem to the platform.
- Quickly and easily automate complex big data analytics with integrated machine learning.
- Remove the complexity and develop IoT applications and solutions without limits.

- Experience Things in a whole new way.
- Extend and share innovation with other IoT developers.

Capabilities of an IOT Platform

- Scalability and Flexibility
- Real-Time Monitoring
- Collaboration
- Third Party Integration
- Ease of use
- Security

Capabilities of an IOT Platform



What is Thing Worx Platform?

The **ThingWorx platform** is a complete, end-to-end technology platform designed for the industrial Internet of Things (IIoT).

It delivers tools and technologies that empower businesses to rapidly develop and deploy powerful applications and augmented reality (AR) experiences.

ThingWorx empowers enterprises to create smart, connected products, operations and software that:

- Improve customer experience
- Optimize business processes
- Differentiate product and service offerings
- Drive new revenue streams

The ThingWorx IoT platform is a collection of modules that deliver the flexibility, capability, and agility establishment required to implement IoT applications.

ThingWorx empowers businesses to develop and deploy powerful applications rapidly and augmented reality (AR) experiences.

ThingWorx is the first platform that connects the people, systems, things, connection operations, connected products, connected applications, etc. ThingWorx reduces the time, cost, and risk which are required to build the IoT applications. It deploys the application 10-time faster with model-based development.

ThingWorx allows you to deploy how you like by providing the complete application design, runtime, and intelligent environment. The ThingWorx IoT platform also has flexibility and scalability to adapt that application in future.

Industrial companies face pressing challenges that require IIoT solutions. To address a wide range of manufacturing, service, and engineering use cases, PTC has spent years innovating theThingWorx IIoT platform.

From remote monitoring and service to workforce efficiency and asset optimization, ThingWorx solves common challenges across different industries.

Because building **Industrial IoT solutions** is often cited as a pain point, ThingWorx is designed to reduce these barriers. Cruise from pilots to enterprise-scale solutions, using pre-built applications and developer tools.

IoT-ThingWorx

The ThingWorx is an IoT platform that helps commercial corporations assess and release the value of IoT. It promises the tools and technology necessary to developed. It implements a robust and daily augmented reality (AR) applications.

The ThingWorx IoT platform includes well-suited modules. It provides the capacity, flexibility, and agility that organizations need to implement IoT applications and AR studies. It includes business connectivity, software activation, analysis, and AR creation.

Benefits of Thing Worx IoT Platform:

- Reason-Constructed Platform
- Rapid Improvement
- Flexibility

Reason-Constructed Platform: The platform offers features specifically designed for protection as well as scalability to develop as the commercial enterprise grows.

Rapid Improvement: It includes platform modules that join through Thing Model. Thing Model is a direct digital illustration of an element of the body. It allows us to enable experiences, studies, and applications quickly and easily.

Flexibility: The platform has the flexibility to be deployed on premise, in the cloud or hybrid. Experiences are available to customers in different formats, such as laptops, Internet, mobile, and AR. It Integrates with external records and simplifies procedures that guarantee more significant results.

Components of ThingWorx IoT

Thingworx offers many key tools to create applications. These tools include Mashup Builder, Composer, search engine, storage, collaboration, and connectivity.

The composer proposes modeling environments for design verification.

Mashup Builder allows the construction of a smooth dashboard through components. For example, lists, wikis, buttons, meters, etc.

Thingworx uses a search engine called SQUEAL. SQUEAL stands for search, query, and analysis. The clients rent SQUEAL for filtering and analyzing records, and search for the information.

Advantages of ThingWorx

Few advantages of using ThingWorx solutions

- To improve customer service and experience
- Maximum utilization of IoT eco-system
- Increase the revenue of the commercial establishment
- Automating and optimizing business process

Different Services Provided for ThingWorx

ThingWorx has numerous services available for the users based upon their requirements. One can easily install it on their desktop using 90 days of a free trial.

Let's look at the various services provided:

1. **ThingWorx Foundation**

ThingWorx Foundation uses the least amount of coding and uses a mash builder which is a drag and drop tool to carry out the operations. Creating a web page or mobile GUI's can be easily accomplished with the help of its foundation. Monitoring and managing connected assets can be easily done with ThingWorx Foundation, moreover accessing critical information and historical data can be quickly accessed.

2. **ThingWorx Analytics**

ThingWorx Analytics provides a platform for performing complex analytical and mathematical operations without any prior statistical experience. Machine learning and AI technology embedded in this Analytics solution automate most of the complex processes. The predictive modelling algorithm quickly analyses the data obtained from the connected devices to forecast and detect the pattern in the data.

3. ThingWorx for Industry

ThingWorx for Industrial connectivity manages, monitors and machines used in the factories and the software application used to run them. It is the industrial solution that connects to a large number of devices and systems in a wide range of industries, including manufacturing, power & utilities, and oil & gas. The Industrial Connectivity offers a real-time and historical view of OPC incidents and diagnostics of communications.

System Requirements for installing Thingworx 8.5v

ThingWorx Server Requirements

The following are minimum recommendations for production. Actual sizing will depend on the application and usage scenarios.

Hardware Sizing

Aspect	Value	Comment
Memory	16 GB	Configuration will require a percent of dedicated memory tobe allocated to the Java VM.
CPU(s)	4 Cores	Virtualized environments may have their own terminology for specifying how many CPUs/Cores are being provided. This must be taken into account when determining if the environment meets the minimum requirements. The amount of concurrent query activity, indexing and searching, as well as the amount of internal event and/or

		property data- change may warrant increased CPUresources.
Disk Type	*	Server-class hardware is recommended.
Disk Space	100GB	This size will accommodate the default ThingWorx webapplication installed in Tomcat, along with the initial Thingworx Storage (DB) directory and some initialconfiguration data. • This does NOT include the space required for theOS or other prerequisite software. • An additional 100GB of space is required if ThingWorx Flow is also installed (for a total minimum of 200GB of disk space). • The total disk space required for any given application depends on the amount of configuration and runtime data that will be maintained. Customersshould work with Sales and Field Enablement to estimate space needs according to their application requirements.

Disk Speed	10000 RPM or SSD	Speed is important if you are storing data on the ThingWorx Core. ThingWorx recommends the fastest disk(s) you can afford, but you should not use disksslower than 10000 RPM. • You may wish to consider a RAID configuration to increase disk performance • If you will be executing regular system backups, it isHIGHLY recommended to use an additional physical disk on its own controller - not simply a separate partition on the same physical disk.

Operating System

Supported OS (64-bit only)	Notes
Windows Server 2016	
Windows Server 2012 R2	Not supported for use with Thing Worx Flow
Red Hat Enterprise Linux (RHEL) 7.6 and 8.2	
Ubuntu 16.04 LTS, 18.04 LTS	Not supported for use with ThingWorx Flow

The software in the table below requires downloading/installing the proper OS-specific version.

In general, you may install later minor versions of the software, but new major versions are not supported unless explicitly stated here (i.e. JDK 1.7.0.17 would be a minor version, but JDK 1.8 would be considered a major version). ThingWorx may still properly run on a newer major version but will not be officially supported until tested and certified.

Required Software

Prerequisite Software	Versions Tested	Comments
Oracle JavaSE DevelopmentKit (JDK)	Java SE Development Kit 8, Update 141 or later, 1.8.0_141-bxx (64-bit)	• ThingWorx is only supported on64-bit operating systems. • Open JDK is not currently supported.
Tomcat (Linux)	• 8.5.66 (64-bit) • 9.0.46 (64-bit)	• Installed manually as the standardpackage management does not typically have the latest versions available. • ThingWorx may still properly run on anewer major version but will not be officially supported until tested and certified.

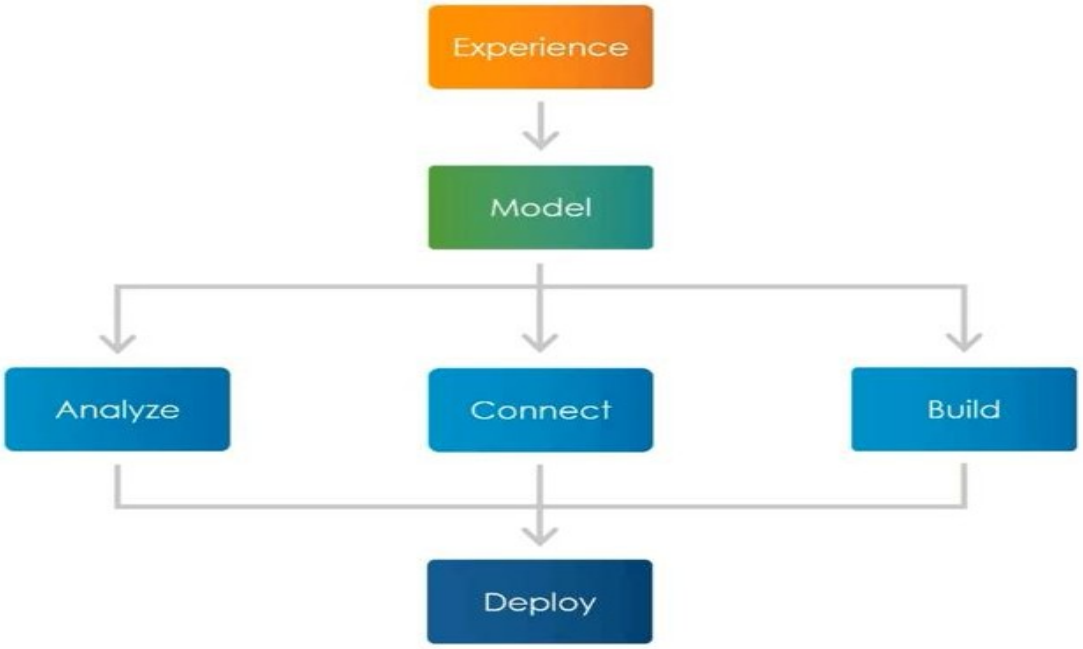
Tomcat (Windows)	• 8.5.66 (64-bit) • 9.0.46 (64-bit)	• Installed using the Windows serviceinstaller. • Thing Worx may still properly run on anewer major version but will not be officially supported until tested and certified.
Ping Federate	• 9.3.3 Patch 5	• Ping Federate is the PTC-supportedmethod for conducting Single Sign-on(SSO) to Thing Worx. Ping Federate is only required forcustomers implementing SSO. • PTC supports the Versions Testedand all subsequent builds within the same minor version (such as 9.3.x).

Database Option

Database	Supported Version	Comment
PostgreSQL	9.6.12, 10.8	Additional information is located in Using PostgreSQL as the Persistence Provider
DataStax Enterpris eEdition (DSE)	6.7	DSE is no longer for sale and will not be supported in a future release. Reference the Endof Sale article for more information. Not supported for use with ThingWorx Flow
		Additional information is located in the Getting Started with DataStax Enterprise andThingWorx Guide.
Microsoft SQL Server	2016 and 2017	Additional information is located in Using Microsoft SQL Server as the PersistenceProvider.
Azure SQL	Azure SQL Logical Server- V12	For information on the latest Azure SQL version, refer to https://docs.microsoft.com/en-us/azure/sql-database/sql-database-faq.
	Azure SQL Managed Instance- V12	Not supported for use with ThingWorx Flow . Additional information is located in UsingAzure SQL as the Persistence Provider
Influx DB	1.7.7	Not supported for use with ThingWorx Flow Additional information is located in UsingInflux DB as the Persistence Provider.

ThingWorx Development Process

The ThingWorx Development process is divided into six stages:

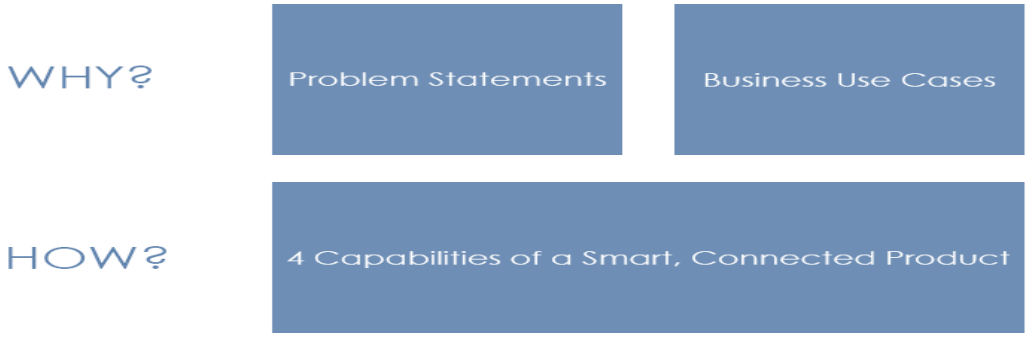


- Experience - define the problem to solve and the value the application provides users.
- Model - create the schema to store the IoT data and functionality, which is the backbone that later stages rely upon.
- Analyze - aggregate and optimize the IoT data to make it easier to understand and more valuable.
- Connect - establish the connection to IoT devices and enterprise systems.
- Build - create a user interface for your users.
- Deploy - move the IoT application into a reliable, scalable, and secure production system.

The Experience Stage

The experience stage is the most important in ThingWorx development because it informs all other stages.

The purpose of the experience stage is to define what the IoT system will do and the value it provides the users. Without this stage, you are merely collecting IoT data without considering users, which will result in a system that doesn't provide users value and is unlikely to be used.



The experience stage is accomplished by answering some critical questions. At this point, you only need a rough answer to these questions; more detail will be added in later stages. However, it is important to define the scope of your application and know why you are building it.

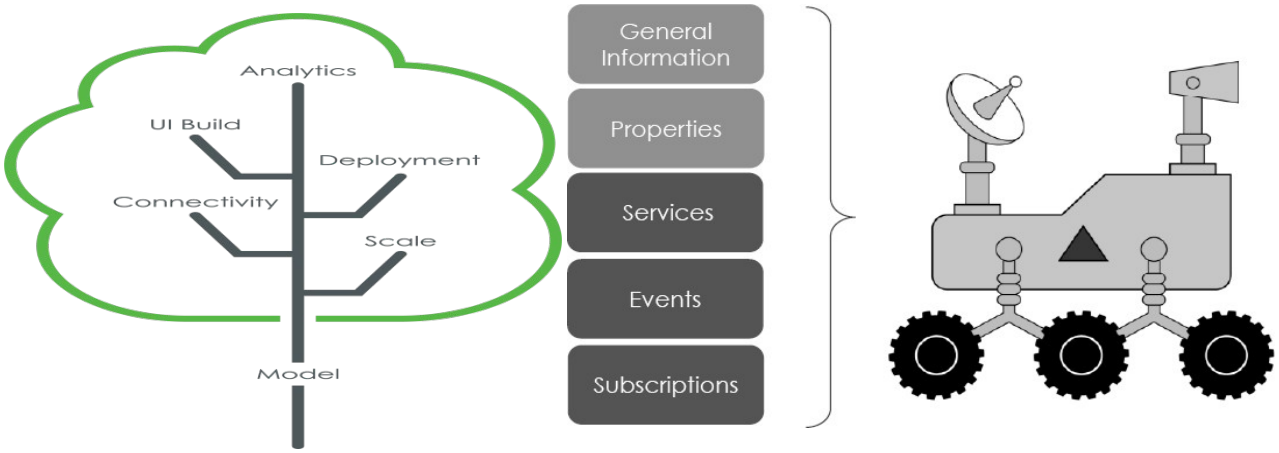
- Who are the users of this IoT application?
- What problems do the users have that this IoT application will solve?
- How will this IoT application enable the users to do their job more efficiently or accurately?
- How will the application access the IoT devices and enterprise systems?
- What will the application do to the IoT devices?

- Monitor: Every IoT application monitors device status. What properties will it monitor?
- Control: Can the user control the device? If so, what control commands can the application send, and how does that help the users?
- Optimize: Does the IoT application utilize the monitored data from the devices? Will it transform data to make it easier for the user to digest? Will it proactively notify the user when under specified criteria? Will it analyze history and suggest optimal settings?
- Automate: Will the IoT system automatically send commands to the device? If so, under what conditions would this occur, and should a user be able to override or shut off this automation?

- How will the users access the application? What devices will they use, and what should the user interface be structured?

Examine these questions from the perspective of the end user. If the answer to the question doesn't help the user do their job, it should not be part of the IoT application.

The Model Stage



The ThingWorx model is the schema of your IoT system, defining what IoT data it accesses and what it will do with that data.
A good analogy is to think of the model as the trunk and root of a tree, the central structure that all other parts of the tree rely upon.

Devices connect to the model, providing the model with data. These devices may also receive control requests from the model.

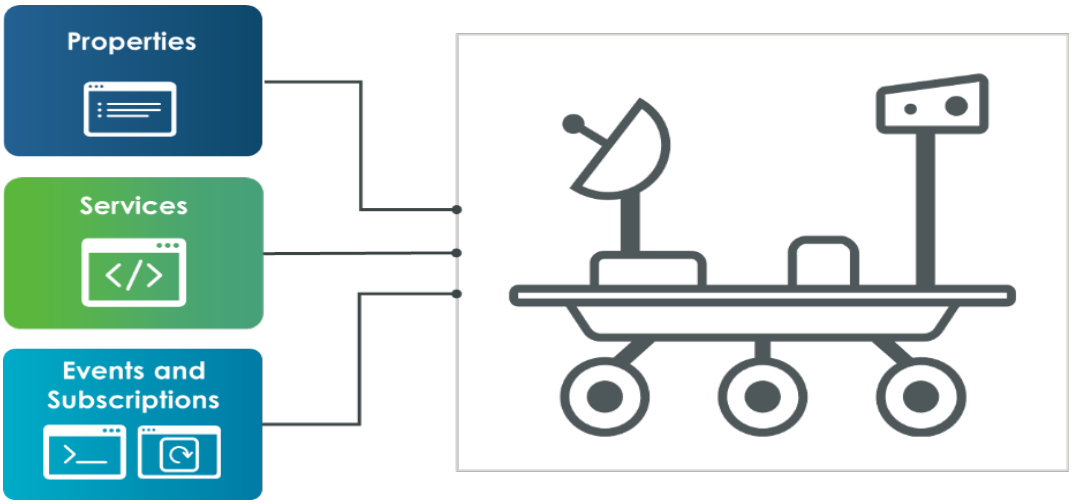
User interfaces are used to display data in the model and provide the user with the ability to send data or commands to the model.

Analytics systems access the model to manipulate and optimize the data in numerous ways, such as predicting when a device will fail based on its properties and how similar devices have behaved in the past.

Lastly, a good model must be built with deployment and scalability in mind. You must consider the model's users and access control scheme to deploy it effectively, making sure the devices and users have the access they need to accomplish their tasks but are prohibited from more restricted tasks and actions.

What is Thing in Modeling?

The ThingWorx model is based on a data structure called the ThingWorx Thing.



A ThingWorx Thing represents a real set of information and data in the ThingWorx system. Different things may represent different types of data. It may represent:

- A physical device, summarize its properties and the commands that can be done to it.
- An enterprise system, gathering data from a remote database or enterprise application, such as **Salesforce**, **SAP**, or a **mail server**.
- A set of internal commands and functions that control the IoT application.

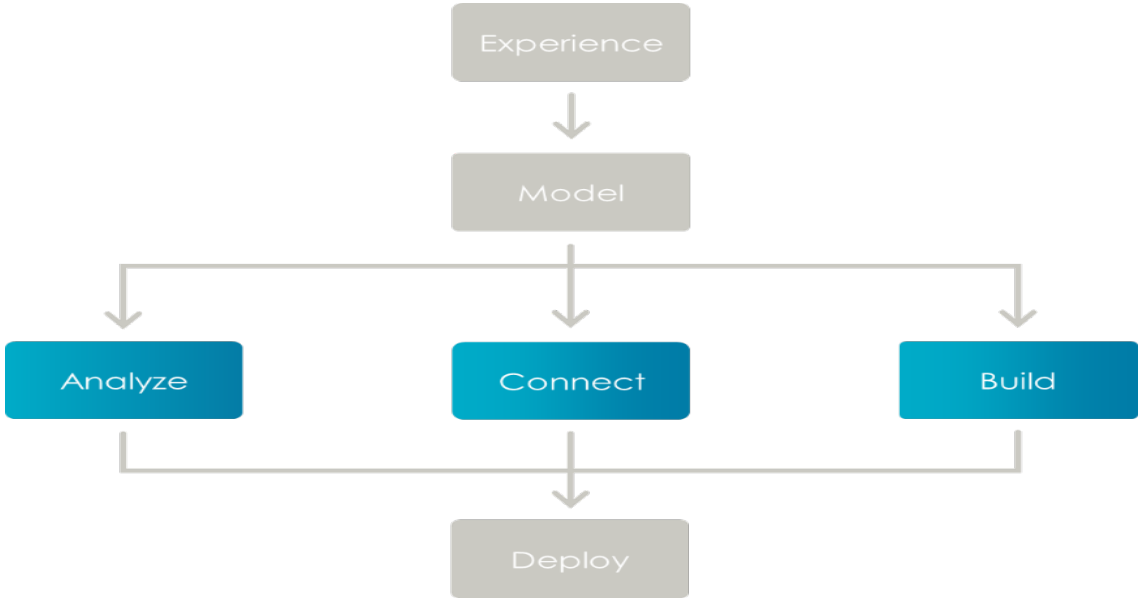
Each Thing may have attributes to support it. **Four of these attributes are:**

- **Properties**, which are status variables. They may be IoT data, such as oil pressure from a device, or internally computed data, such as the optimal speed of an assembly line as determined by advanced analytics.

- **Services,** which are commands that the user may issue to the Thing. For example, if a user can change the speed of an assembly line or add a user to the system, they would do so by executing a service.
- **Events and Subscriptions** are linked together. Subscriptions are similar to services. However, instead of a user executing them, they are executed automatically when an event occurs. For example, if ThingWorx sends an SMS message to the operator of a device when the oil pressure exceeds a certain level, that is a subscription executed automatically upon an oil pressure event.

Throughout the model stage, it is critically important to remember the experience stage. Consider your users and consider access control. Each thing and attribute of the thing is subject to access control, permitting some users to perform the action while denying other users the same action.

Analyze, Connect, and Build Stages



The next three stages, analyze, connect, and build, may be accomplished in any order or in parallel. However, they cannot be accomplished without a model.

- You cannot analyze data that is not defined in the ThingWorx model.
- Devices require a modeled thing to connect to, such as a Database Thing or a Device Thing.
- The build stage creates a user interface informed by data in the model. This user interface is usually, but not always, a Web-based user interface. It may be an augmented reality UI based on Vuforia, or it may provide notifications using email or SMS.

The Deploy Stage

The deploy stage covers tasks involved when your IoT application is built and needs to be placed into production. Topics, such as access control testing, scalability testing, user acceptance testing, selecting the correct hardware, scaling the production system appropriately, setting up redundancy, and high availability functionality, securing the system with encryption, setting up a backup scheme, and migration are all deploy stage topics.

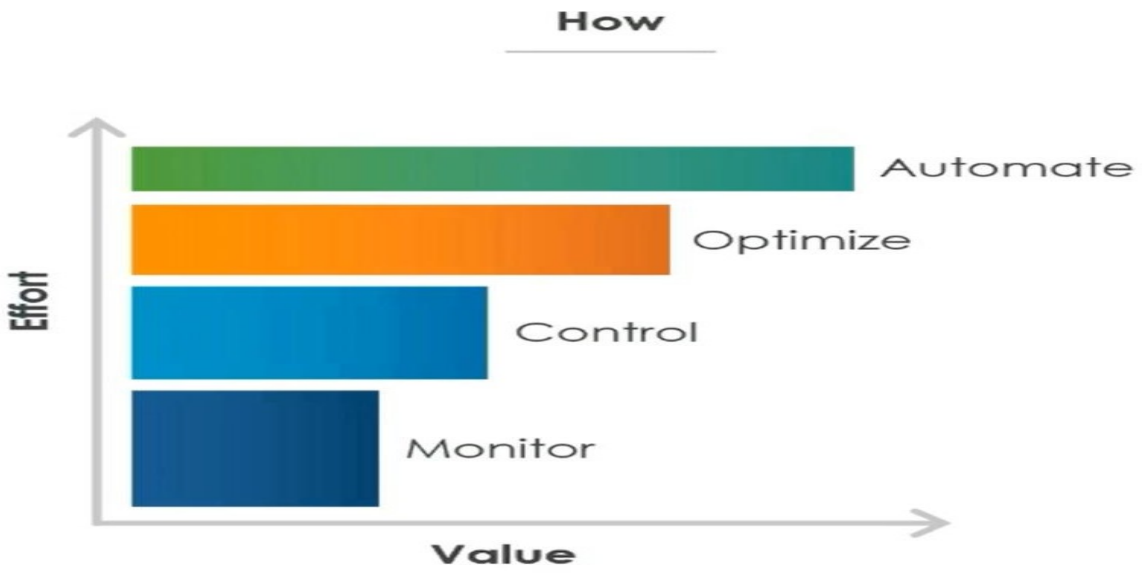
Capabilities of Thingworx

With Thingworx's platform capabilities, you can easily manage the development lifecycle for your IoT applications in one centralized location. With ThingWorx, you can rapidly connect devices, analyze data, build, and deploy solutions
Generally, in IoT development, there are four capabilities in escalating stages of complexity:

1. Monitor: How do I monitor IoT devices and dashboards that retrieve information?
2. Control: Once I can monitor a device, how can I control it? This involves sending instructions to the device that will be able to change its behavior.

3. Optimize: Now that I can control it, how can I provide it with information to consistently generating a particular outcome? At this point you'll find it useful to leverage data analytics.

4. Automate: How can I ensure that my desired outcomes happen automatically without user intervention?



Note that not all applications need all four capabilities. Some applications simply monitor whereas others may monitor and control. However, you cannot achieve more complex capabilities without the ones underneath it. Optimizing a device that you can't monitor is not possible.

In addition to teaching about the ThingWorx Development Process, Adam also covered the benefits of our ThingWorx LEARN Online Subscription. Because all stages in the process depend upon the Experience Stage, PTC University has broken out the supporting topics into a distinct Fundamentals Learning Path that is comprised of three-hour, live instructor-led courses that cover each stage. In addition, taking these courses in their recommended sequence aptly prepares you to take the ThingWorx Fundamentals Exam where you can earn your Fundamentals Certification. Passing this exam showcases your knowledge and expertise in the product and is the only authorized and verified credential for ThingWorx Fundamental knowledge.

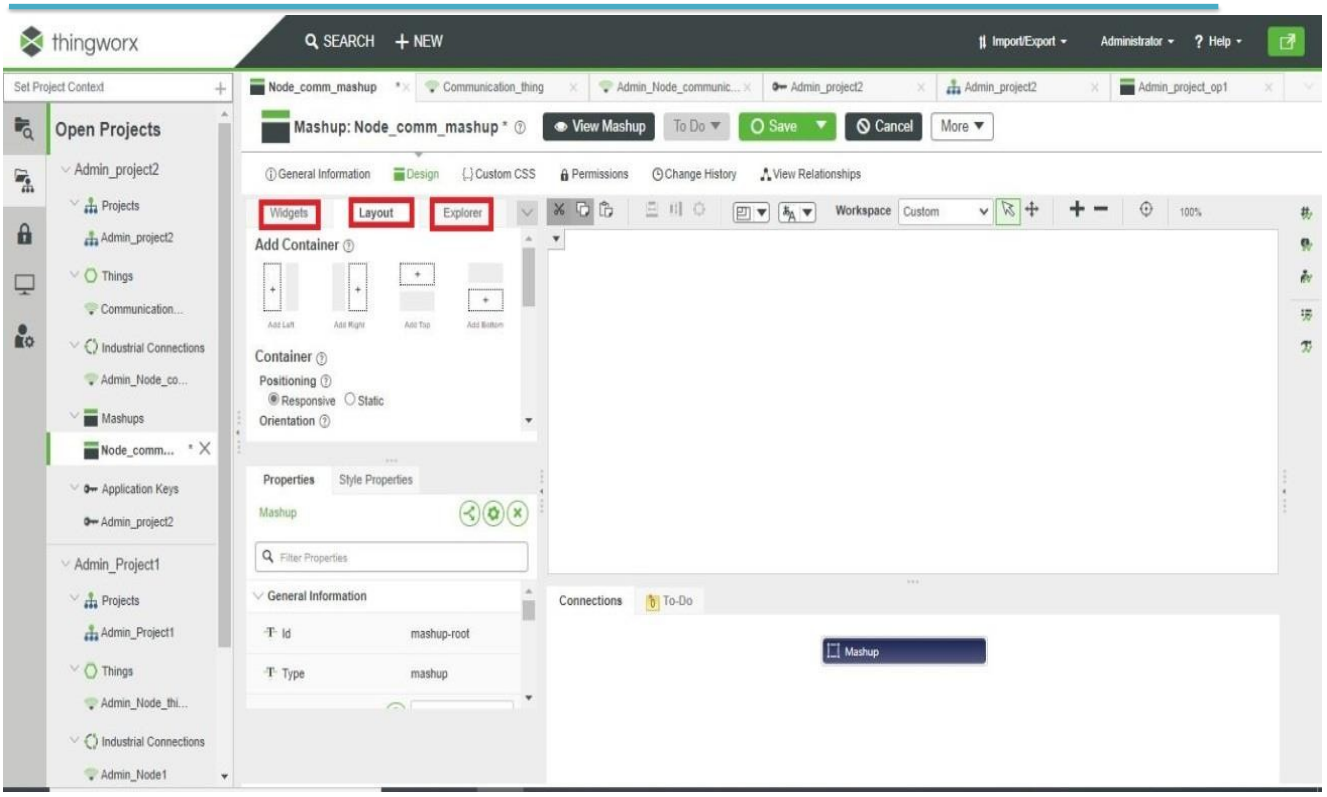
MASHUP CREATION

1. Plan Screen Layout and Functionality

Before using the mashup builder, plan your mashup on paper. When planning the mashup, you should consider:

- The expected screen resolutions of your users. This will drive the size and positioning of the containers and widgets. It also drives the decision to use responsive or static container sizing and widget positioning.
- The relative sizing of each container and widget in the user interface.
- The data and services required to drive the widgets. Although you do not need these to begin UI development, you will need them to complete the mashup. Advanced mashups may also require contained mashups.

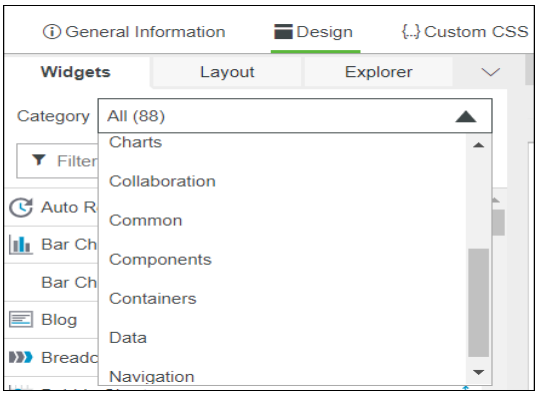
2. Implement the Layout Using Containers and Container Widgets

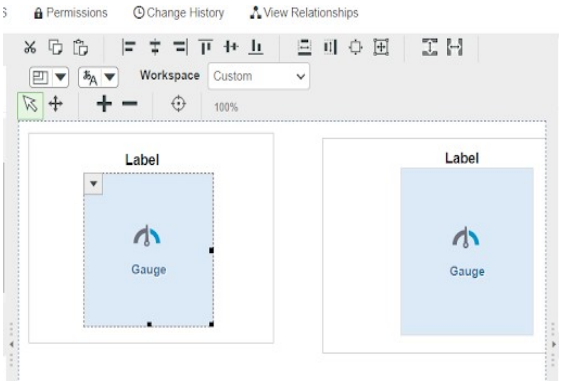


After planning the user interface, open the mashup builder and add the containers that create your layout. Set container sizing. The layout should be complete before adding widgets. You should:

- Use containers to divide the user interface into areas with different widgets and functionality. Some containers may have multiple widgets, while others may only have one or may be empty to provide whitespace.
- Consider whether widget positioning is responsive, following alignment and distribution rules, or static, with you setting the number of pixels each widget is offset from the edge of the container.
- Add container widgets. These are widgets designed to contain other widgets and enhance the layout, such as the Tabs widget, Field set widget and the Dynamic Panel widget. However, widgets that bind and display data should be deferred until the layout is complete.

3. Add Widgets



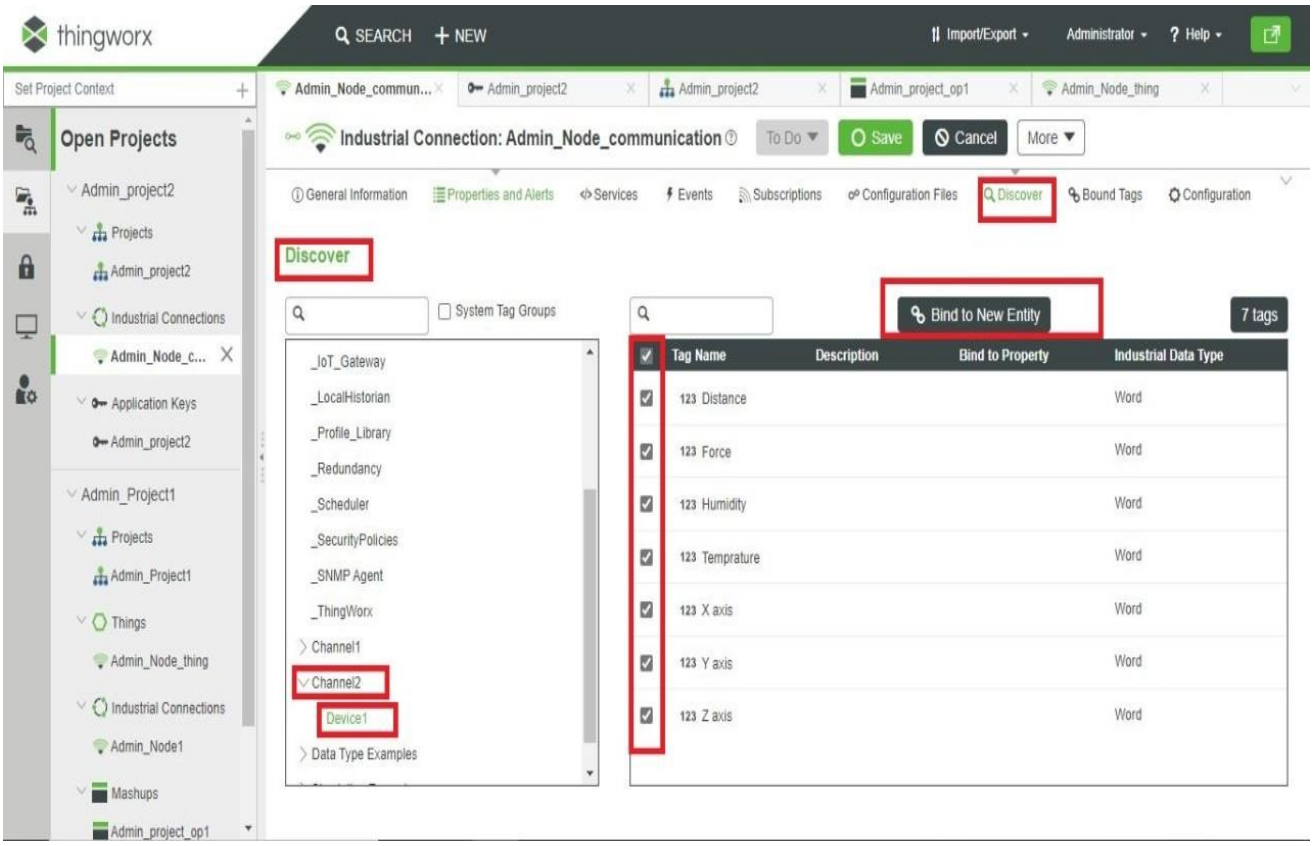


Drag the widgets into the containers and position them in the containers. A few points to remember are:

- If the container size has static positioning, you must set the position of the widget manually.
- If the container is responsively sized or has responsive positioning, consider all resolutions that users access the user interface. Verify that scroll bars do not appear at any expected screen resolution.
- Styling may be done now or deferred until later. However, fonts should be approximately the correct size to verify that all components fit.
- Until you have bound data, set default text to a string approximating the length of the data.
- Some widget properties may be set now, but others must be deferred until you have bound data. For example, a list widget can't specify the data field it will display until it is bound to an info table containing the available fields.

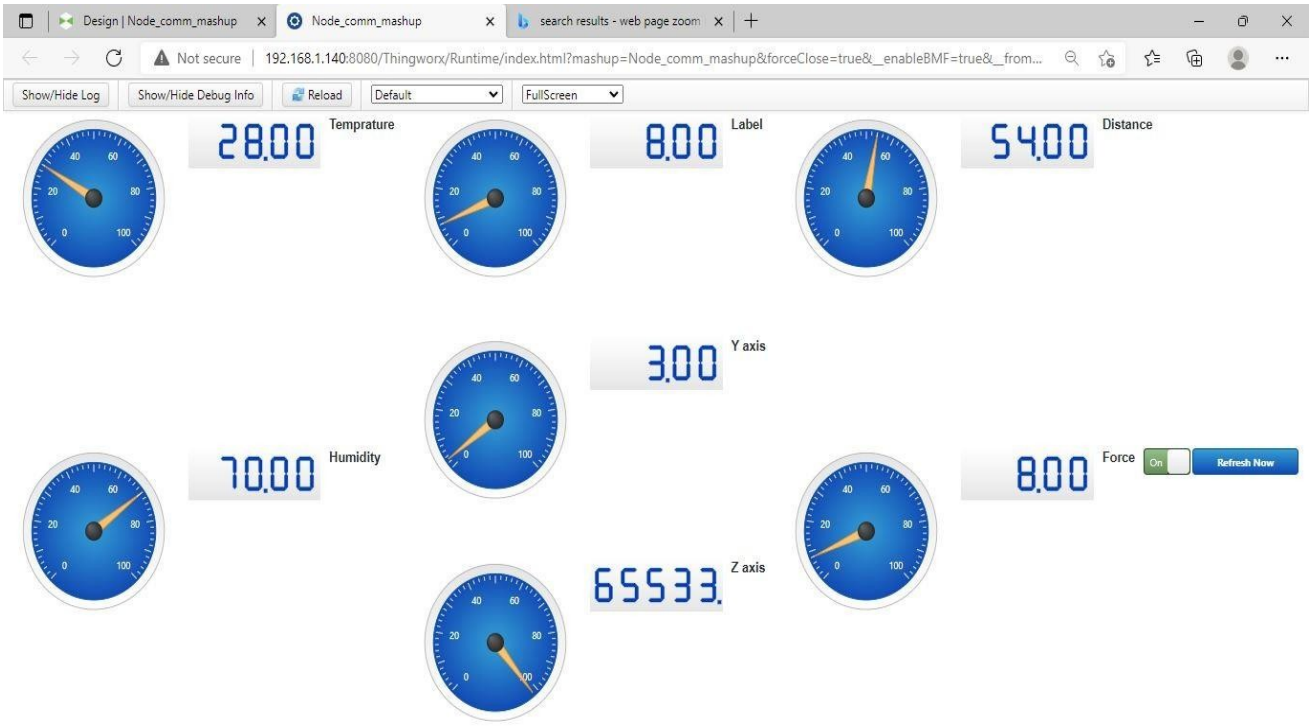
4. Bind Mashup Events and Data

In this stage, use the Data tab to:



- Select the services to execute.
- Select the mashup events that trigger the services.
- Bind the result of the service to widgets.

5. Test the Mashup



As you develop the mashup, test functionality, making sure widgets appear where they are expected, buttons and events are working properly, and incoming data is as expected. Initially, you may test as an administrative user.

However, it is very important to test as a non-administrative user that belongs to the same ThingWorx access control groups as the end user. This will uncover any access control issues with the mashup or the underlying services. If something works as the Administrator, but not as an end user, the issue is not with the mashup itself but with access control settings.

Exercise session

Excerise1

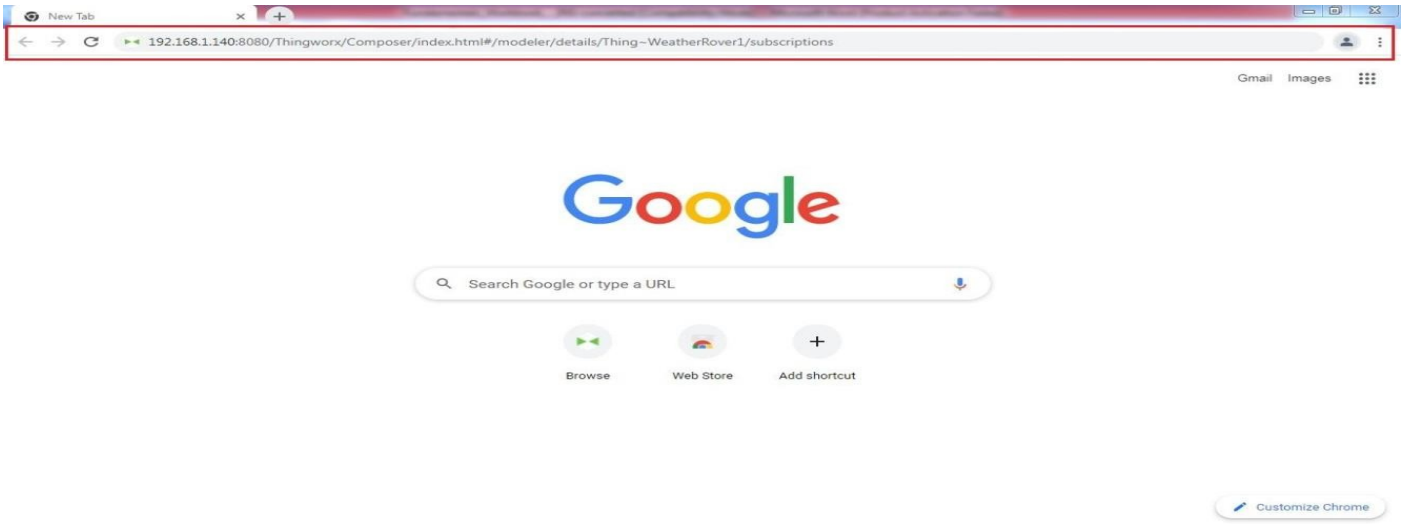
How to LOG IN TO THINGWORX?

Step1: Check Your PC is connected to the server PC if not connect via LAN cable

Step2: Check Your Internet connection

Step2: Open the browser

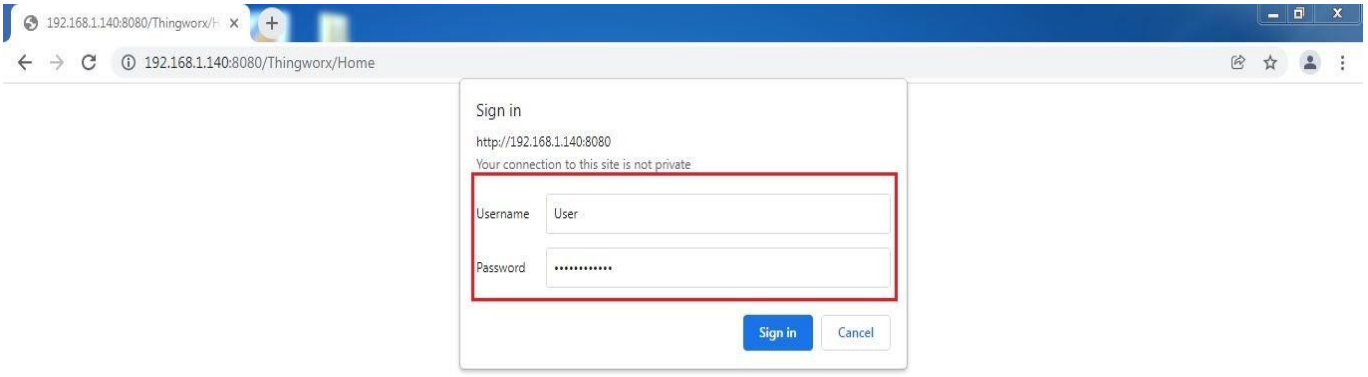
Step2: Enter the Local Host URL (192.168.1.140:8080/Thingworx/Home) with host PC IP Address



Step3: Enter to Thingworx using login Credentials

EX: - User Name: User

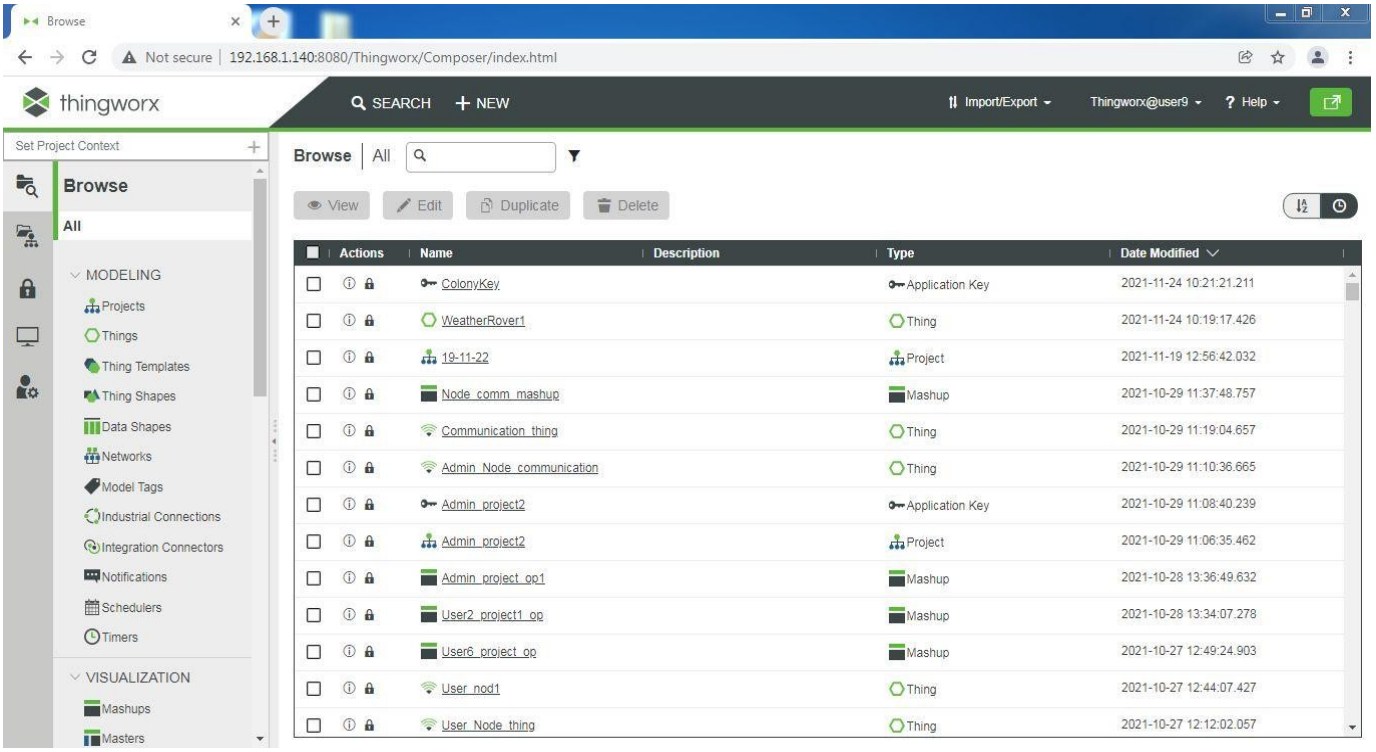
Password: User123



Exercise 2

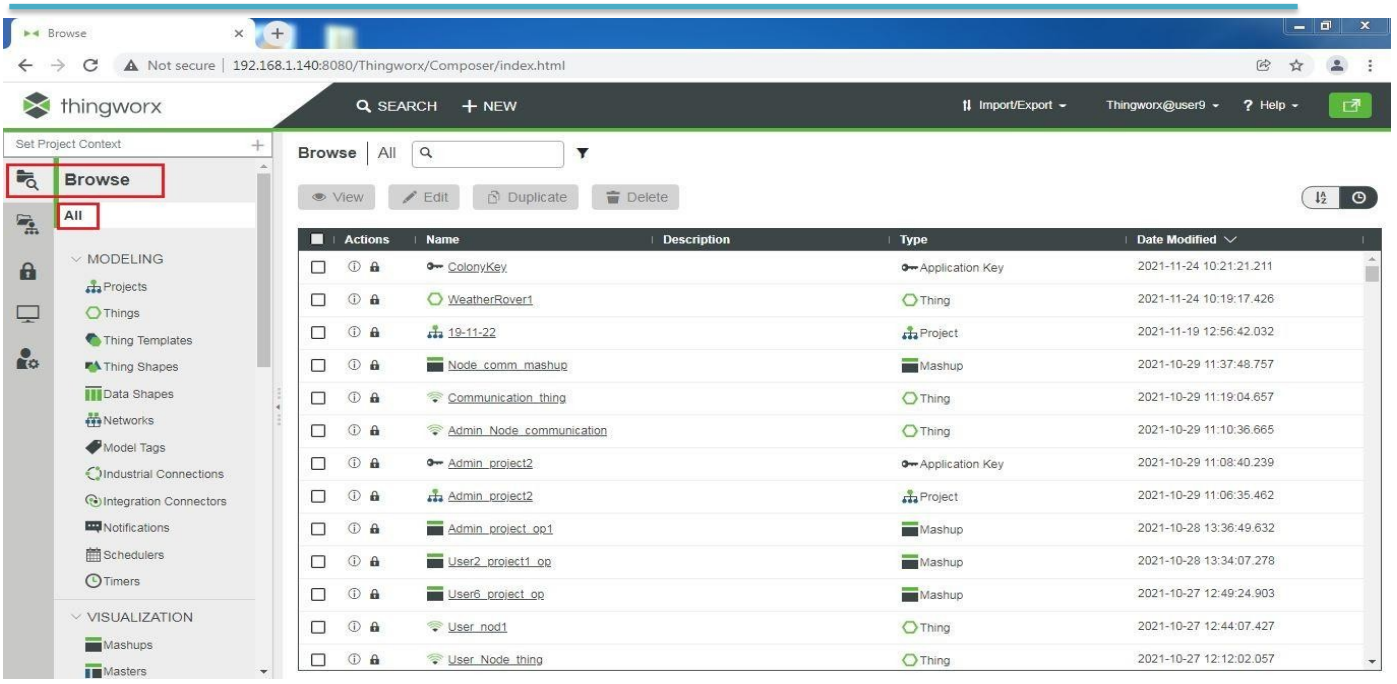
ThingWorx Composer

Step1: After the login to Thingworx The Thingworx Composer window will open



Step 2 Click on the  Browse Icon in top right corner of Composer window to Browse

Step 3 In Browse click on All to see all the Options available in the ThingWorx composer

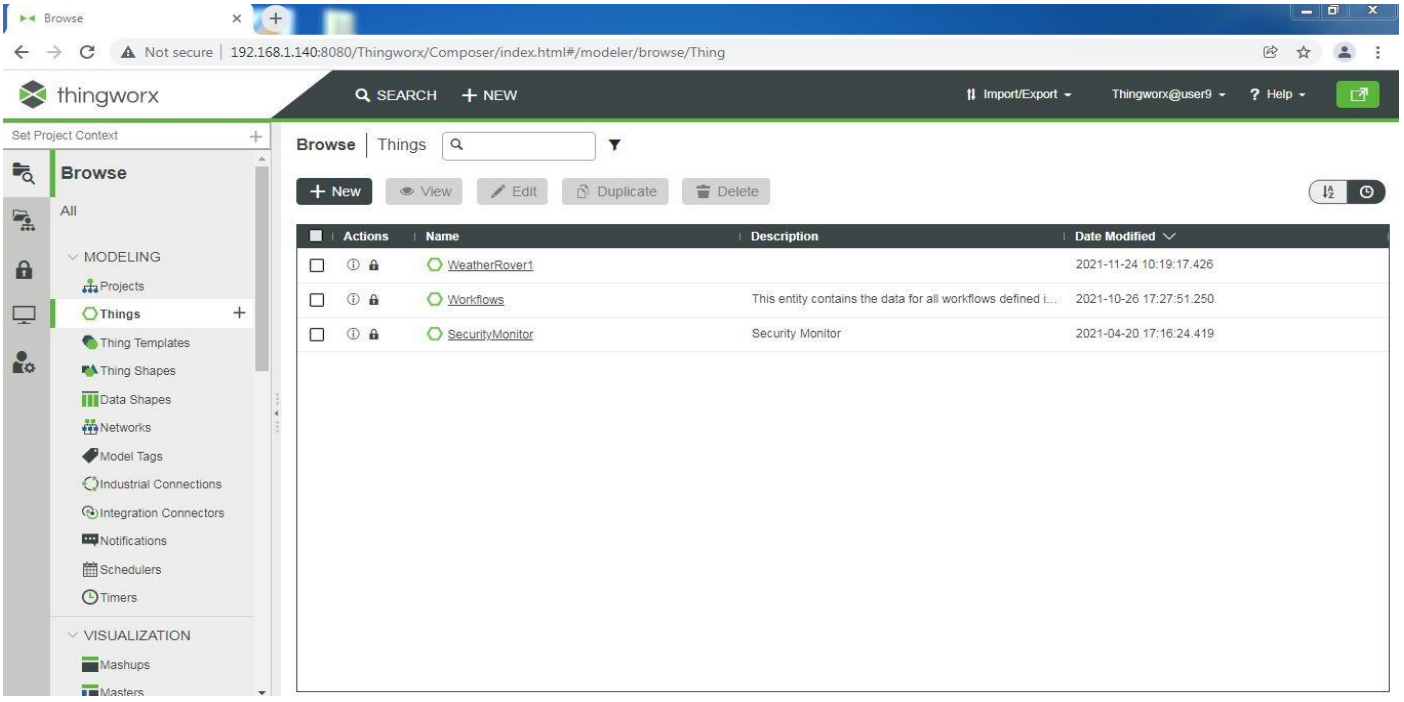


Exercise 3

Thing Creation

Step 1 In Browse Go to Modelling Option

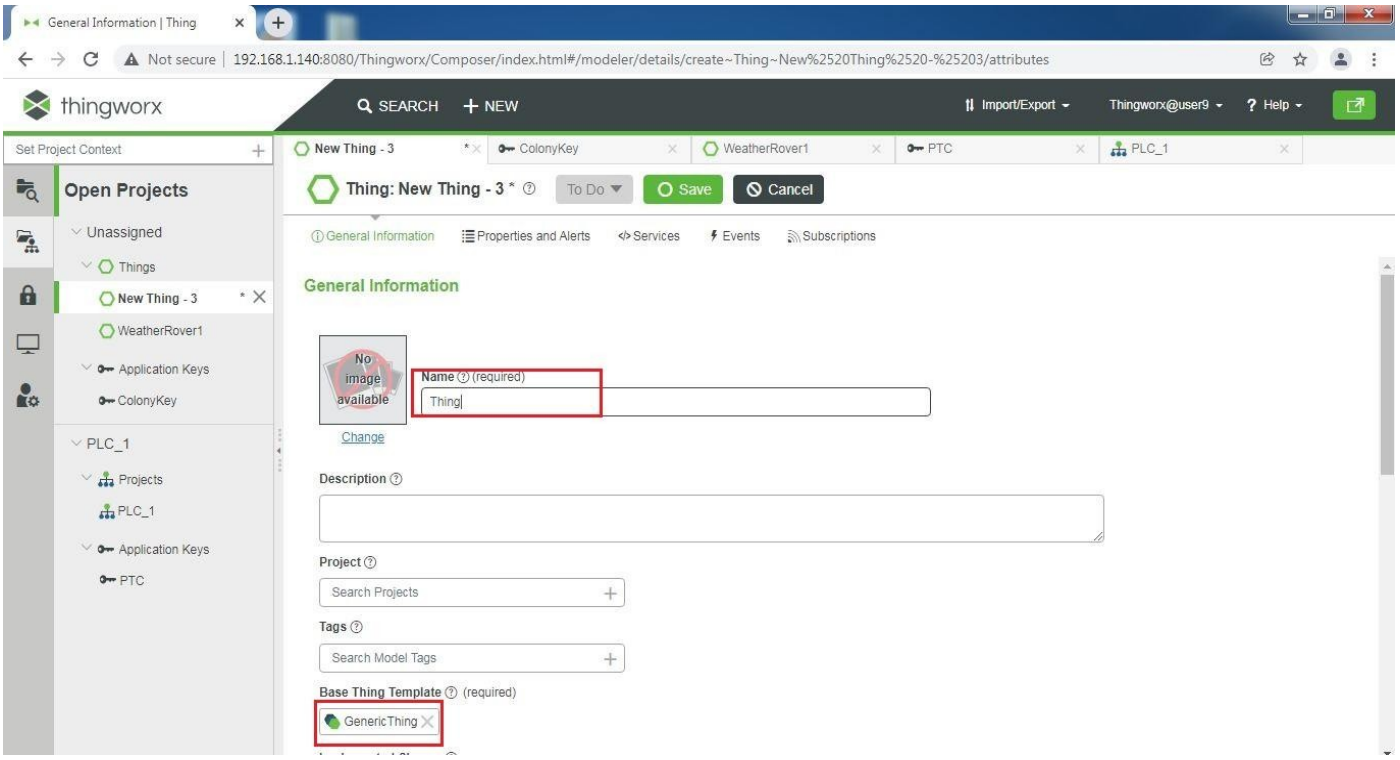
Step 2 In Modelling select Thing, The thing window will open



Step 3 Select 

option to create new Thing

Step 4 Assign name for the Thing and Select Base Thing



Step 5 : After Assign name for the Thing and Select Base Thing Click on save option

Exercise 4

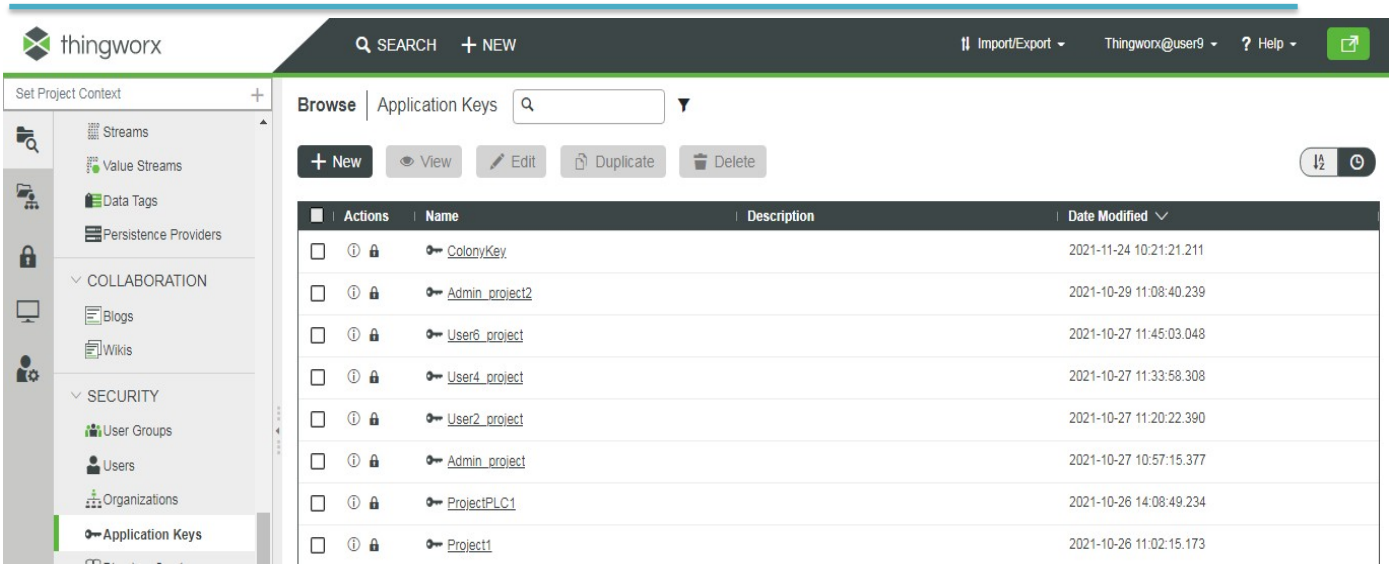
Application Key Creation

Step 1 In Browse Go to Security Option

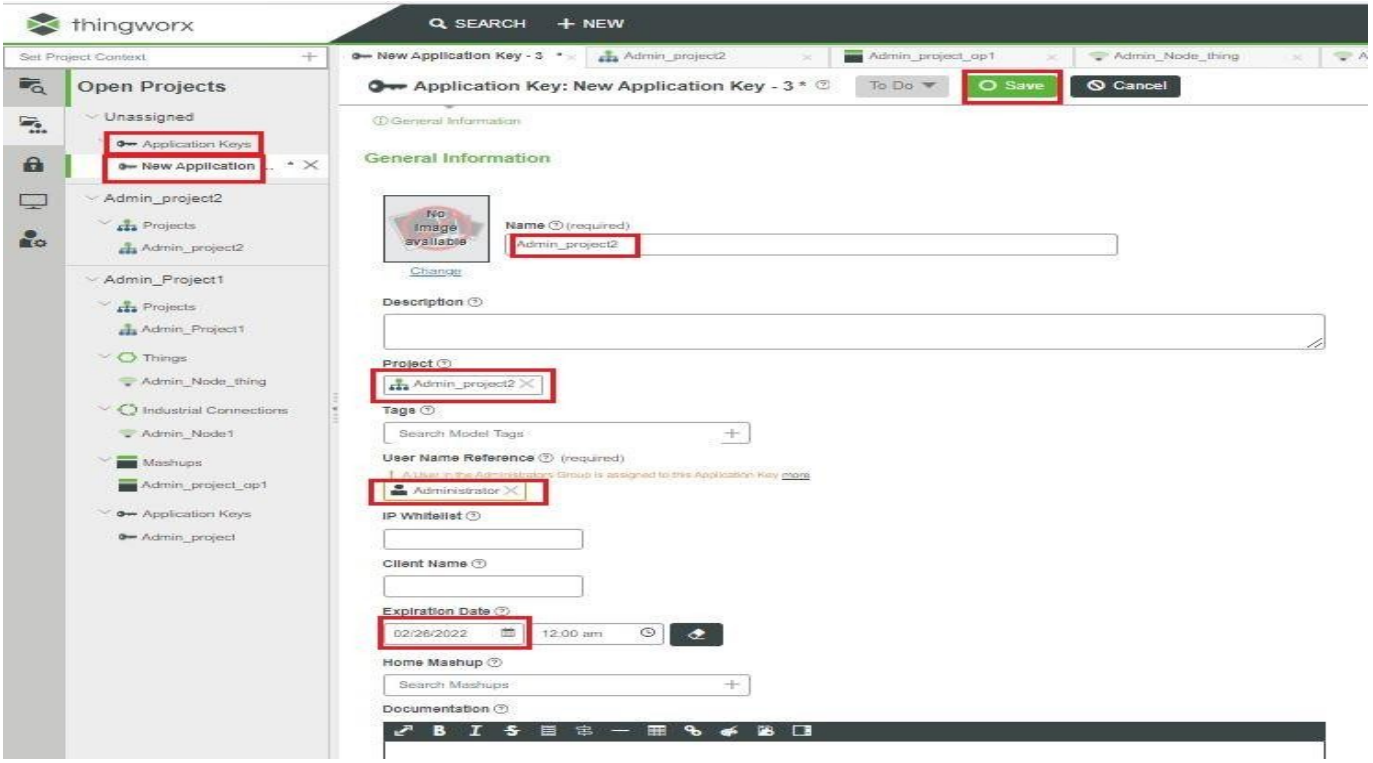
Step 2 In Security select Application key, The Application Key window will open

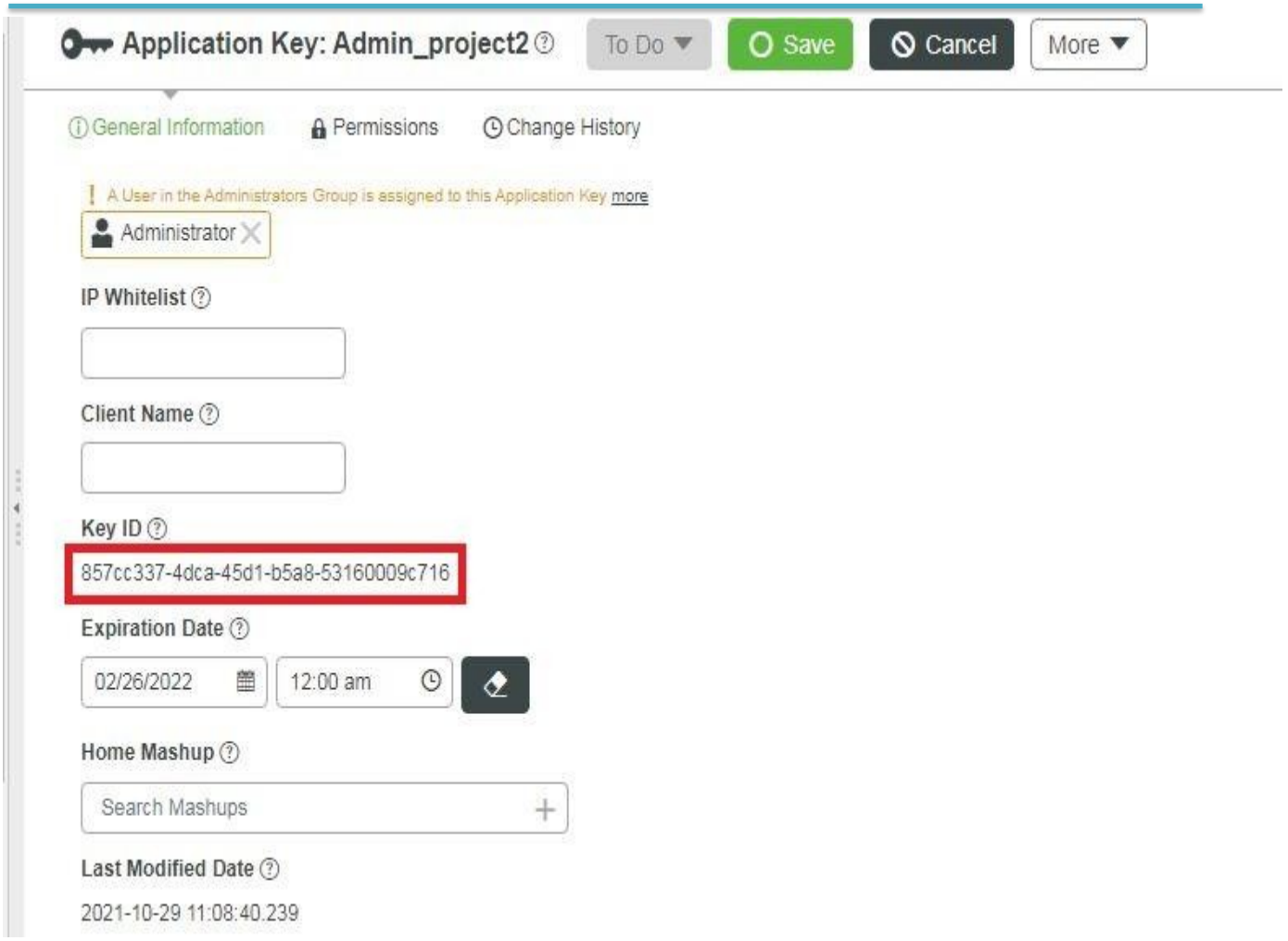
Step 3 Select 

option to create new Thing



Step 4 Assign name for the Application and select user



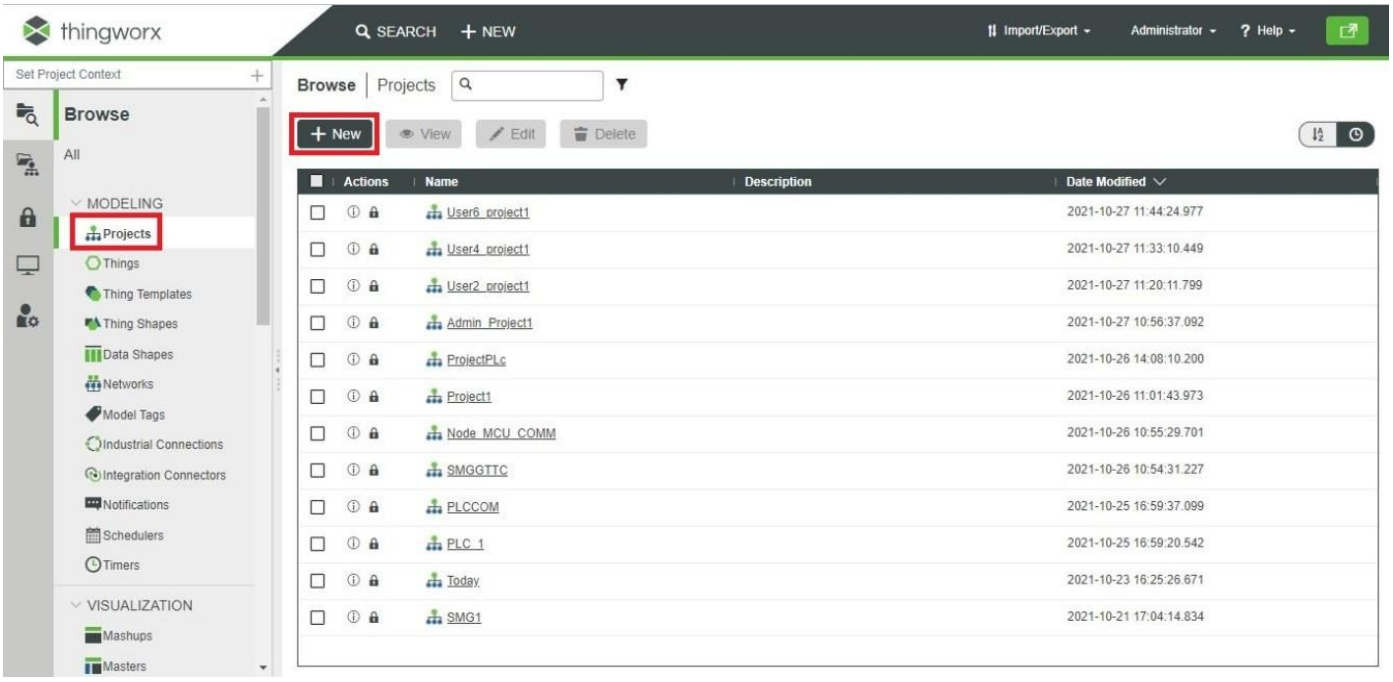

Application Key: Admin_project2
To Do
Save
Cancel
More
General Information
Permissions
Change History
A User in the Administrators Group is assigned to this Application Key more
Administrator
IP Whitelist
Client Name
Key ID
857cc337-4dca-45d1-b5a8-53160009c716
Expiration Date
02/26/2022
12:00 am
Home Mashup
Search Mashups
Last Modified Date
2021-10-29 11:08:40.239

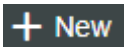
Exercise 5

Project Creation

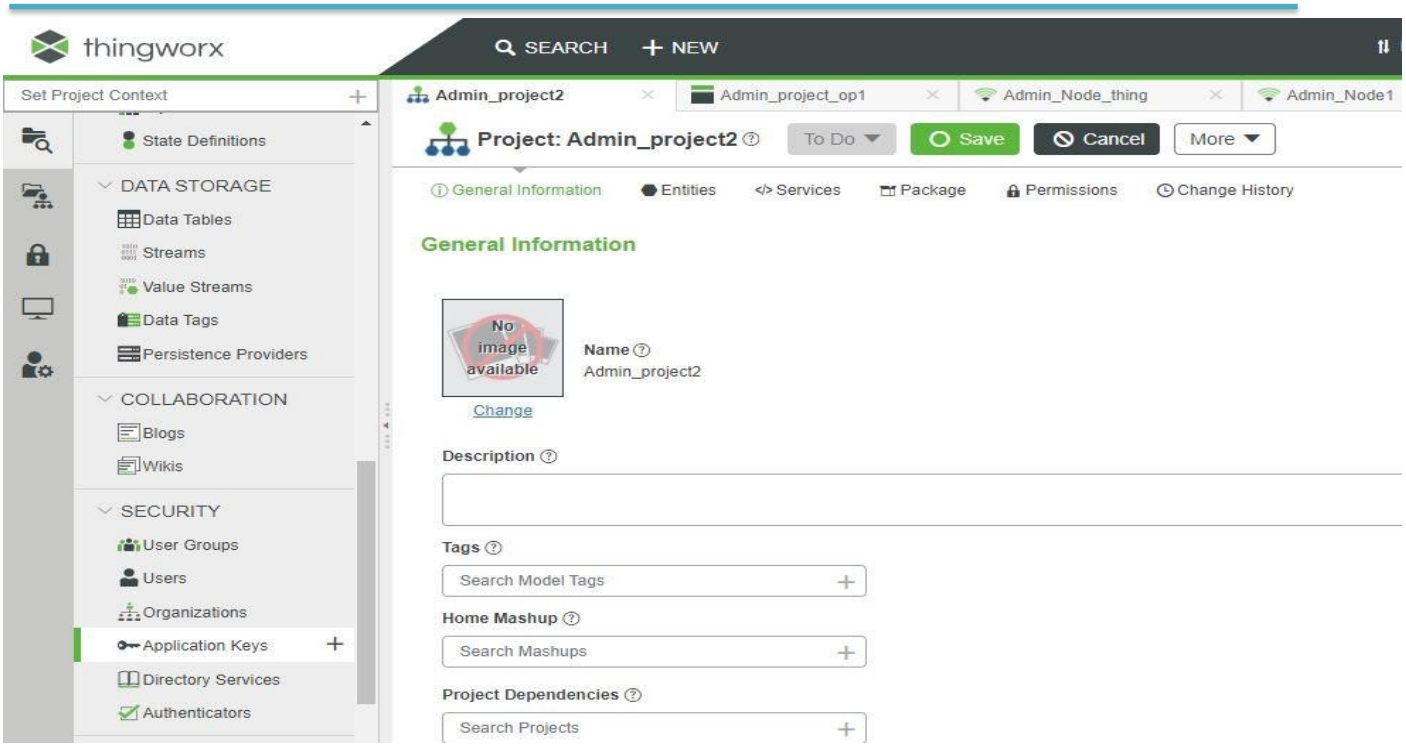
Step 1 In Browse Go to Modelling Option

Step 2 In Modelling select Project, The thing window will open



Step 3 Select  option to create new Project

Step 4 Assign name for the Project and save the project



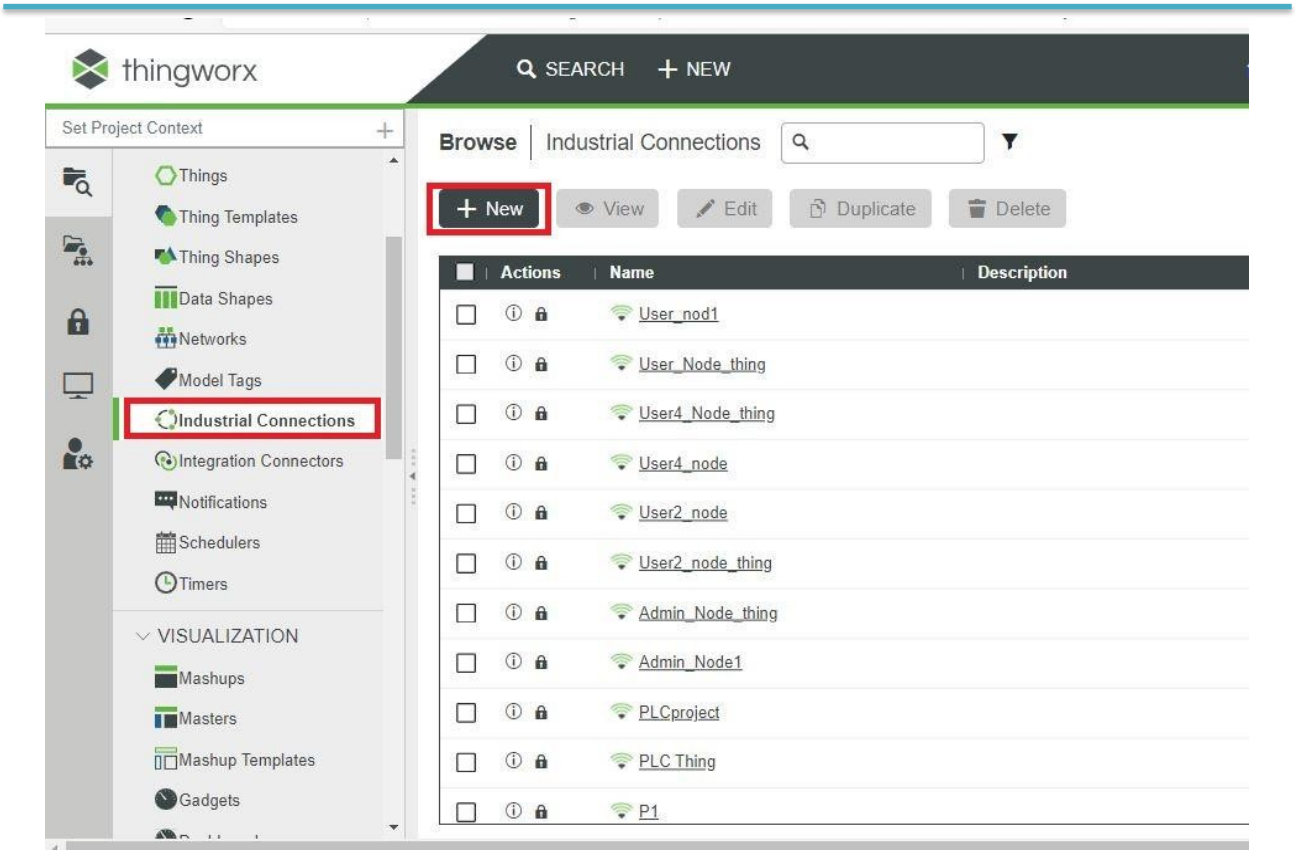
Exercise 6

Industrial connection

Step 1 In Browse Go to Modelling Option

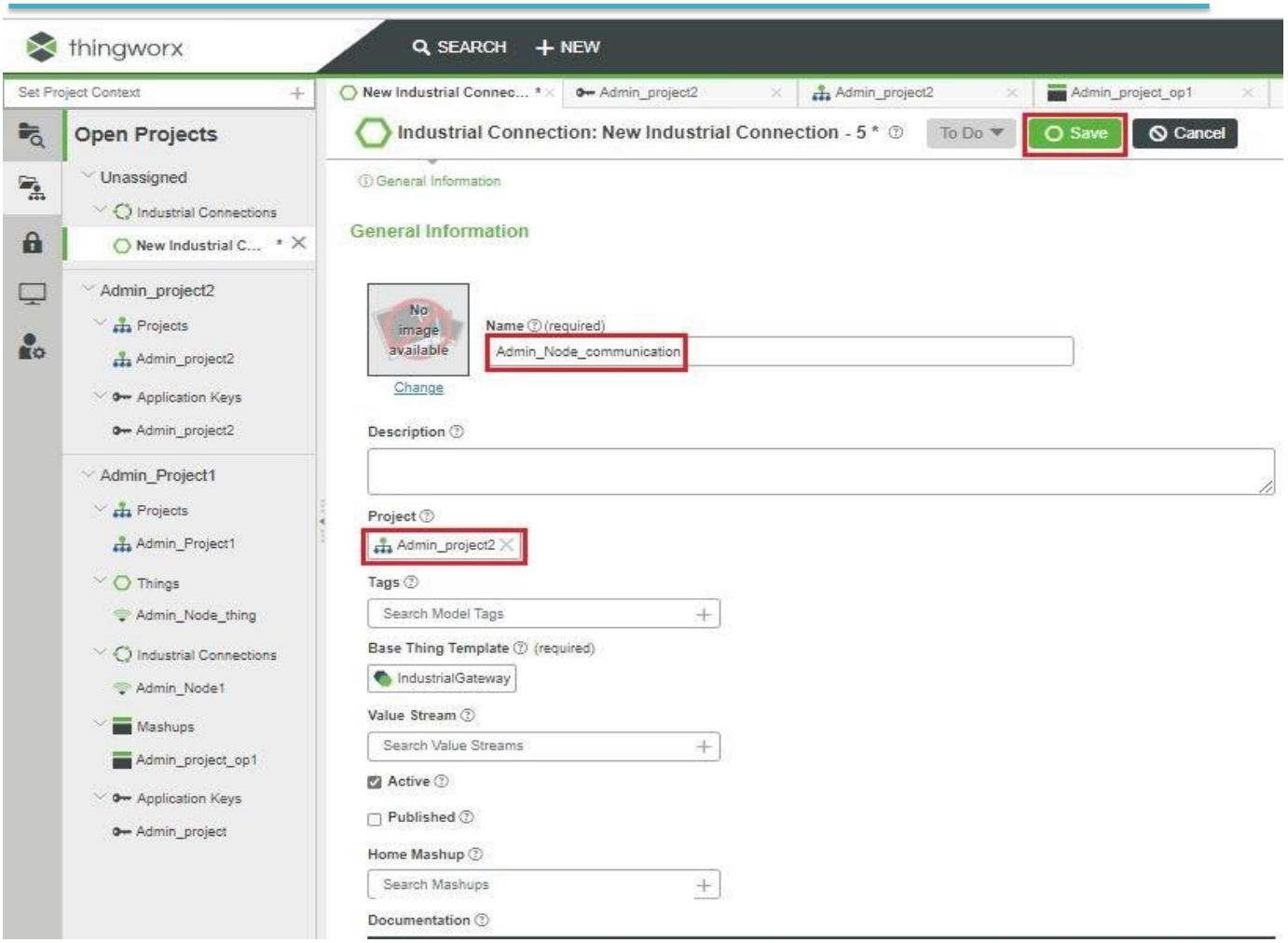
Step 2 In Modelling select Industrial connection, The Industrial connection window will open

Step 3 Select + New option to create new Industrial connection



Step 4 Choose Template Industrial Gateway

Step 5 Assign name for the Industrial connection and Select project



Step 6: After Assign name for the Industrial Connection and Select Project Click on save option 

Exercise 7

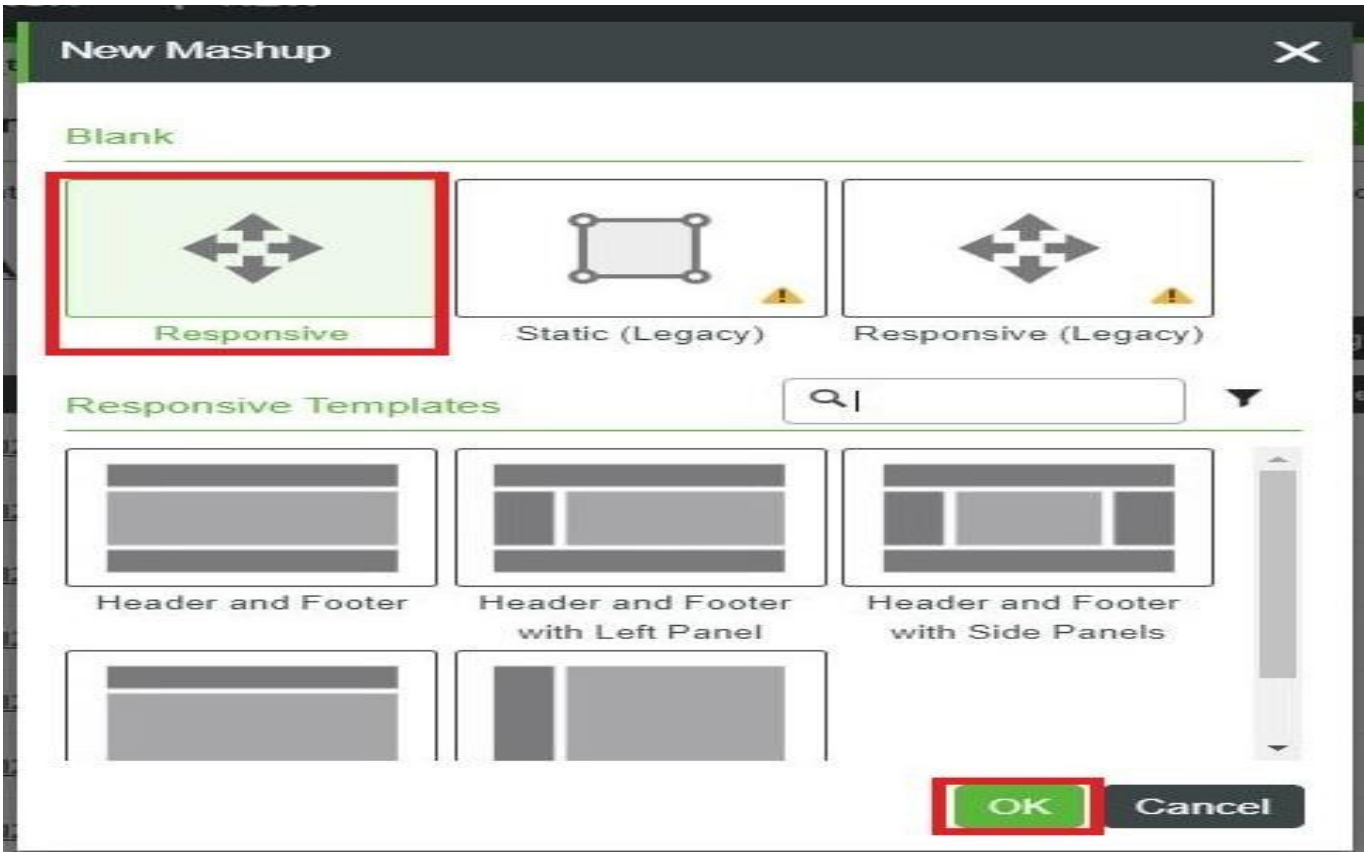
How to Create mashup?

Step 1 In Browse Go to Visualization Option

Step 2 In Visualization select Mashup, The Mashup window will open

Step 3 Select + New option to create new Mashup

Step 4 In new Mashup Select Blank as Responsive and click OK



Step 5 Assign name for the Mashup and Select project

Step 6: After Assign name for the Mash and Select Project Click on save option O Save

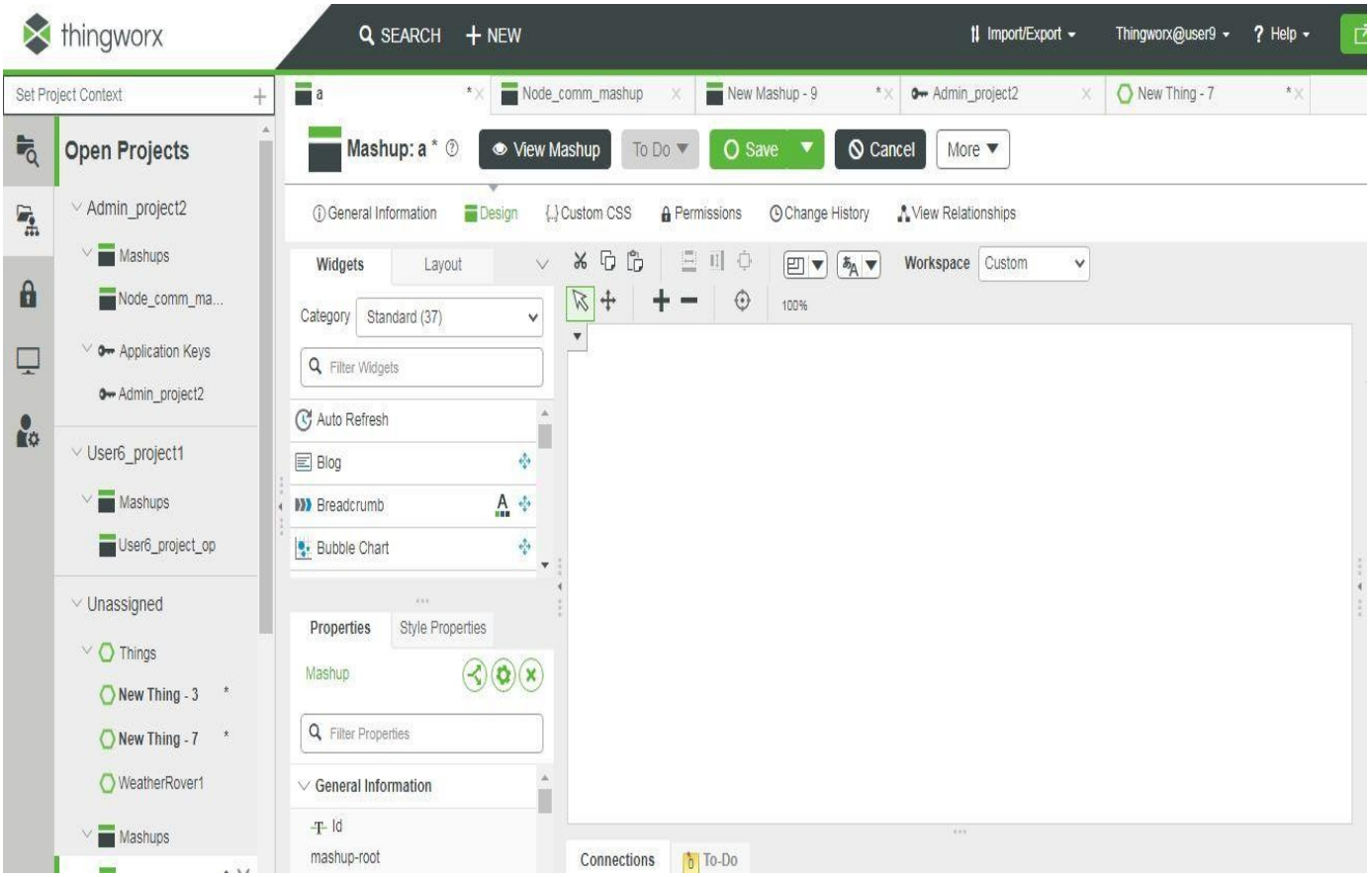
Exercise 8

How to Design the Mashup?

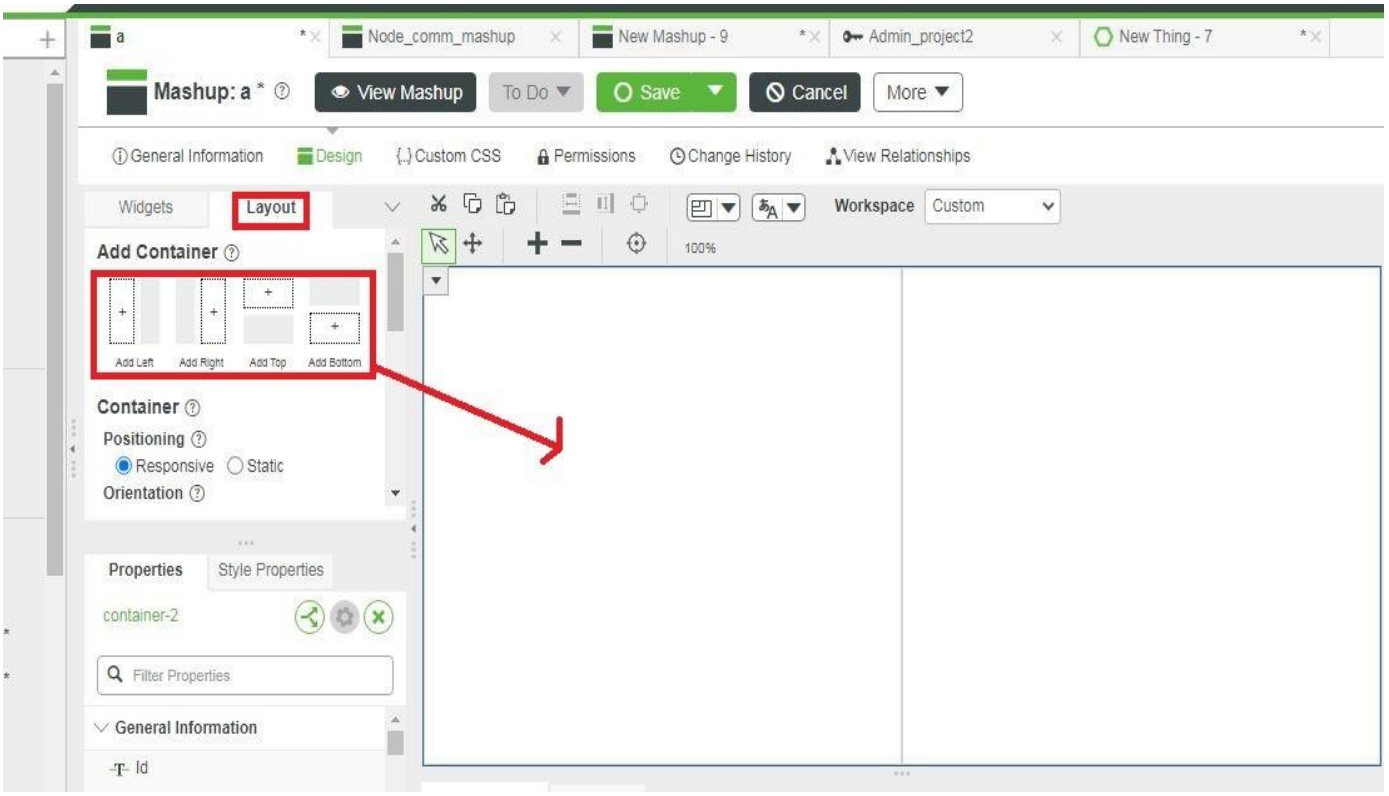
Step 1: Open the Mashup that was created

Step 2: At the top of the mashup window click on design option to enter design window

Step 3: After the click on the design option the design window will open

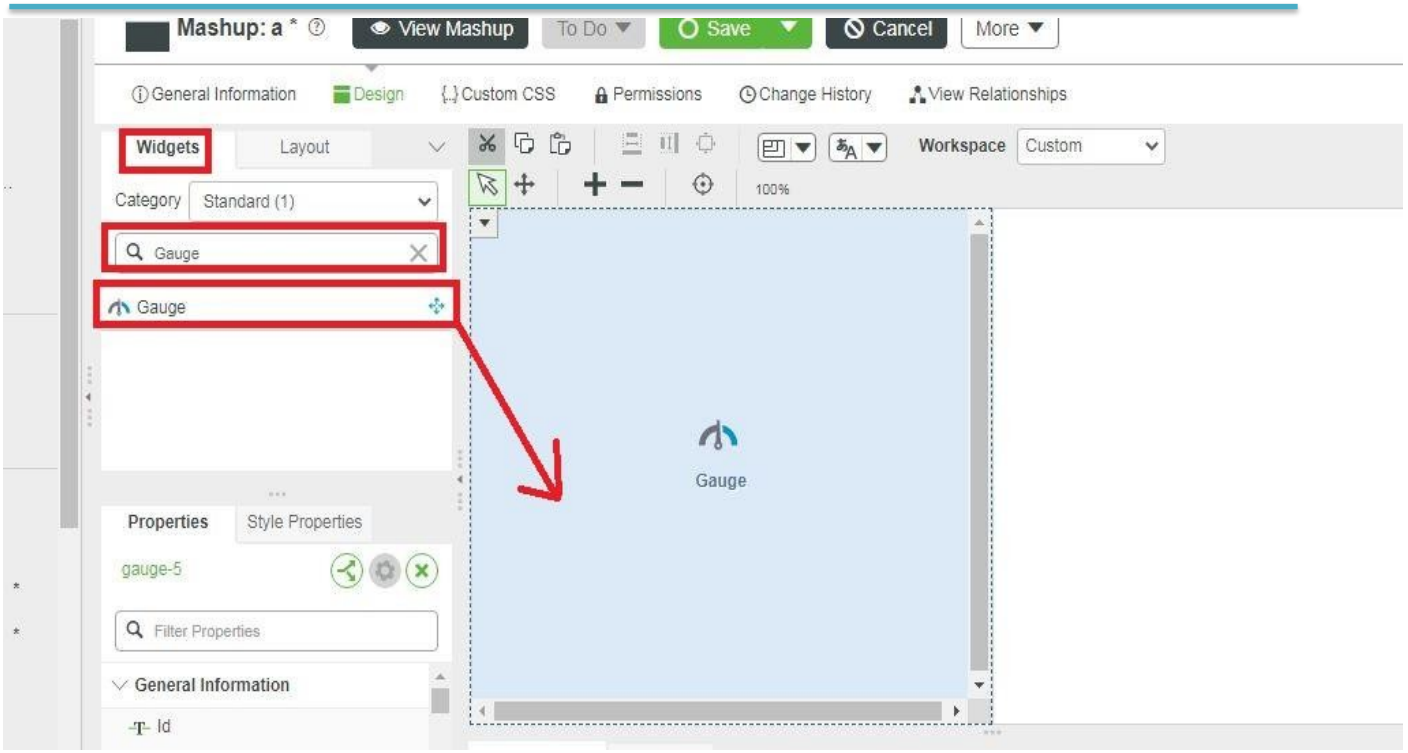


Step 4: go to layout and select number of containers required.



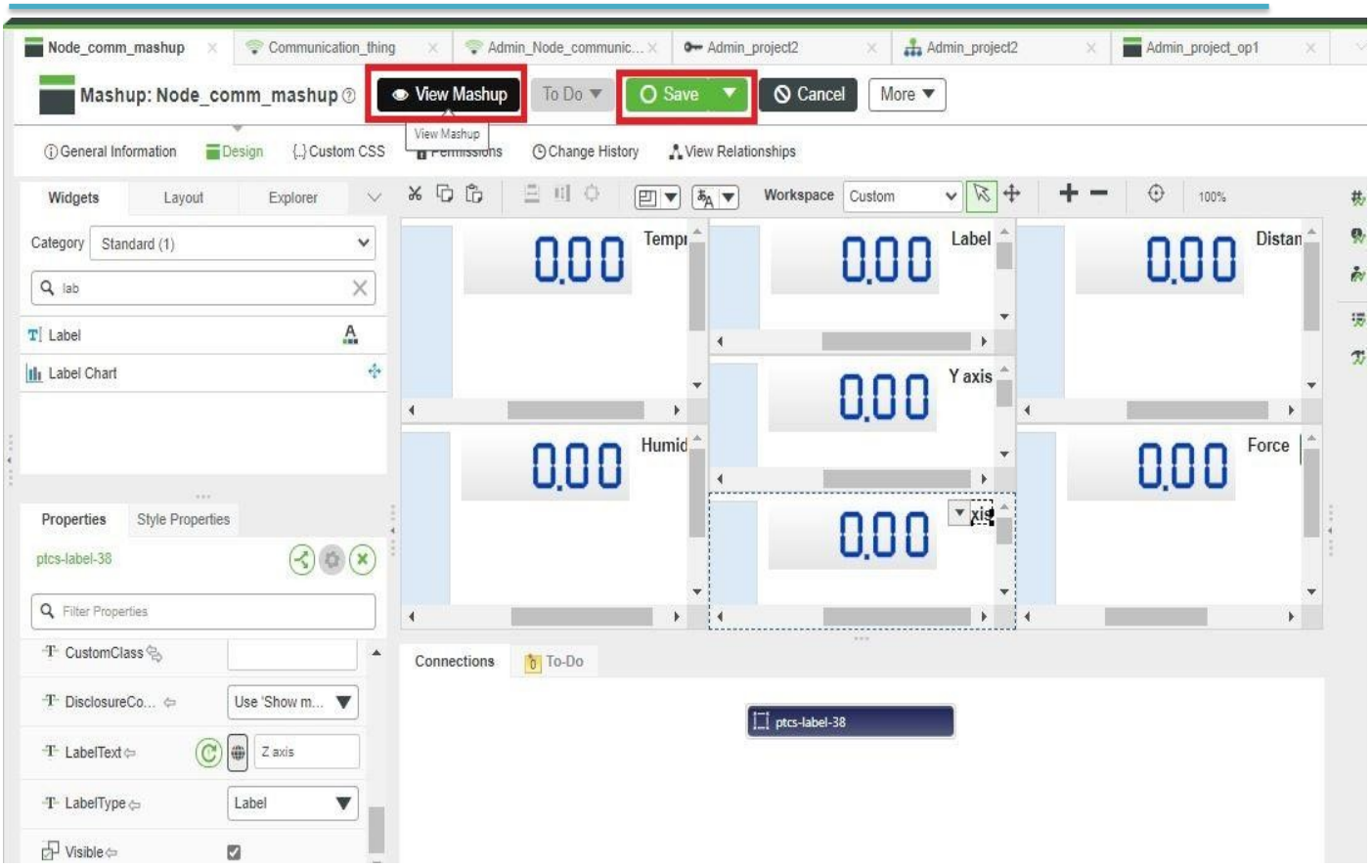
Step 5: go to the widgets to add widgets like gauge, Led, name tag etc.

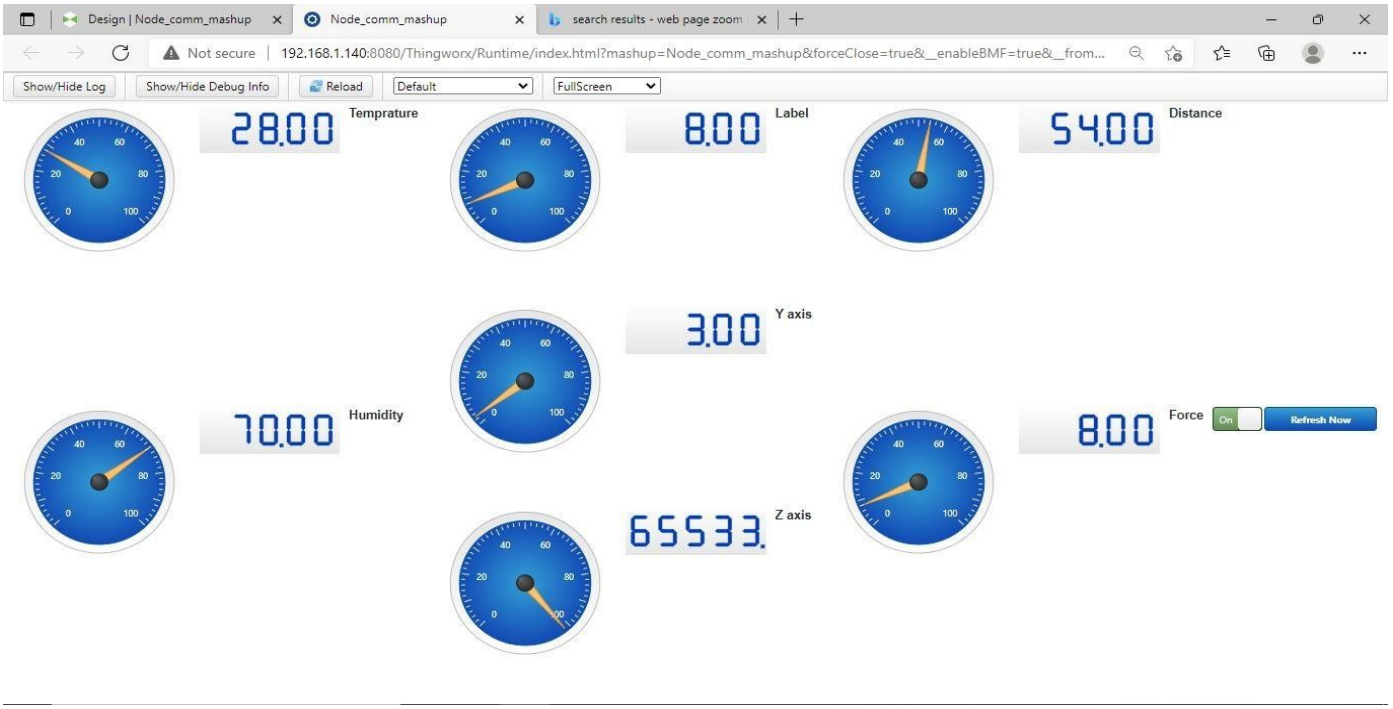
Step6: you can search the widgets in the search window



Step7: select the widget and drag and drop on design window

Step8: Save the mashup and click on view mashup to see the mashup


Node_comm_mashup
Communication_thing
Admin_Node_communic...
Admin_project2
Admin_project2
Admin_project_op1
Mashup: Node_comm_mashup
View Mashup
To Do
Save
Cancel
More
General Information
Design
Custom CSS
Change History
View Relationships
Widgets
Layout
Explorer
Workspace
Custom
Category
Standard (1)
lab
Label
Label Chart
Properties
Style Properties
ptcs-label-38
Filter Properties
CustomClass
DisclosureCo...
Use 'Show m...
LabelText
Z axis
LabelType
Label
Visible
Connections
To-Do
ptcs-label-38
Label
Y axis
Force


Design | Node_comm_mashup
Node_comm_mashup
search results - web page zoom
Not secure
192.168.1.140:8080/Thingworx/Runtime/index.html?mashup=Node_comm_mashup&forceClose=true&__enableBMF=true&__from...
Show/Hide Log
Show/Hide Debug Info
Reload
Default
FullScreen
28.00
Temprature
8.00
Label
54.00
Distance
3.00
Y axis
70.00
Humidity
8.00
Force
On
Refresh Now
65533.
Z axis

Introduction of Thingworx:

The **ThingWorx platform** is a complete, end-to-end technology platform designed for the industrial Internet of Things (IIoT).It delivers tools and technologies that empower businesses to rapidly develop and deploy powerful applications and augmented reality (AR) experiences.

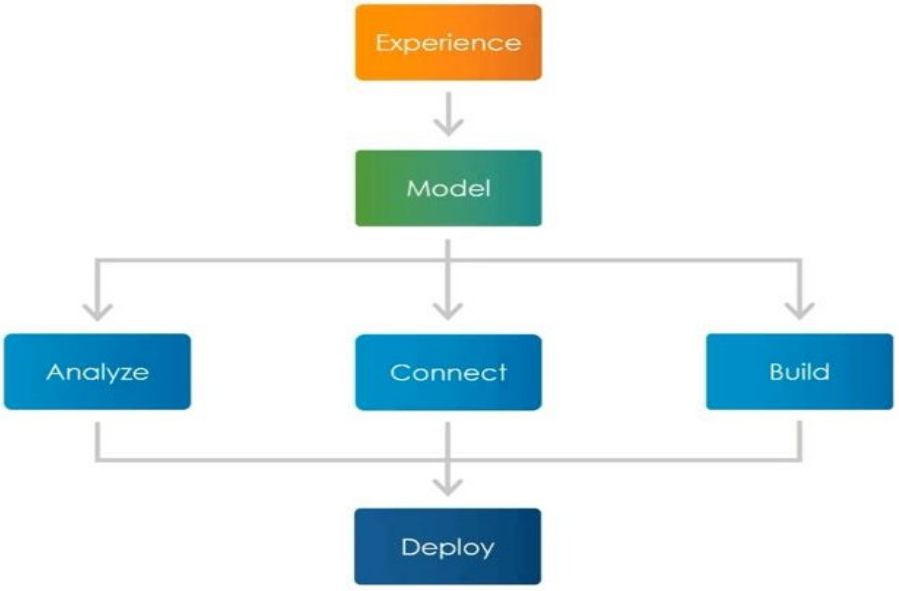
ThingWorx empowers enterprises to create smart, connected products, operations and software that:

- Improve customer experience
- Optimize business processes
- Differentiate product and service offerings
- Drive new revenue streams
- The ThingWorx IoT platform is a collection of modules that deliver the flexibility, capability, and agility establishment required to implement IoT applications.
- ThingWorx empowers businesses to develop and deploy powerful applications rapidly and augmented reality (AR) experiences.
- ThingWorx is the first platform that connects the people, systems, things, connection operations, connected products, connected applications, etc. ThingWorx reduces the time, cost, and risk which are required to build the IoT applications. It deploys the application 10-time faster with model-based development.

- ThingWorx allows you to deploy how you like by providing the complete application design, runtime, and intelligent environment. The ThingWorx IoT platform also has flexibility and scalability to adapt that application in future.
- Industrial companies face pressing challenges that require IIoT solutions. To address a wide range of manufacturing, service, and engineering use cases, PTC has spent years innovating the ThingWorx IIoT platform.
- From remote monitoring and service to workforce efficiency and asset optimization, ThingWorx solves common challenges across different industries.

Because building **Industrial IoT solutions** is often cited as a pain point, ThingWorx is designed to reduce these barriers. Cruise from pilots to enterprise-scale solutions, using pre-built applications and developer tools.

Thingworx development process:



- Experience - define the problem to solve and the value the application provides users.
- Model - create the schema to store the IoT data and functionality, which is the backbone thatlater stages rely upon.
- Experience - define the problem to solve and the value the application provides users.
- Analyse - aggregate and optimize the IoT data to make it easier to understand and morevaluable.
- Connect - establish the connection to IoT devices and enterprise systems.
- Build - create a user interface for your users.
- Deploy - move the IoT application into a reliable, scalable, and secure production system.

1. The Experience Stage:

Overview of the ThingWorx Development Process in which he asserted that the most important stage of the process is, without a doubt, the Experience Stage. Let's take a deeper dive into what this stage is and why it can be a make or break for your application's success.

The ThingWorx Development Process is comprised of stages users go through beginning with conceptualization all the way through deployment. But why did Adam emphasize the Experience Stage so heavily? It's because the Experience Stage is where you determine why you are building an IoT application in the first place and what's going to be included.

This is the critical point in the process where you define your scope – what are you building, why you're building it, and what value it should ultimately deliver. Every phase you subsequently proceed through refers back to the Experience Stage. It is the point where you will ask yourself which specific goals you are furthering by developing your project.

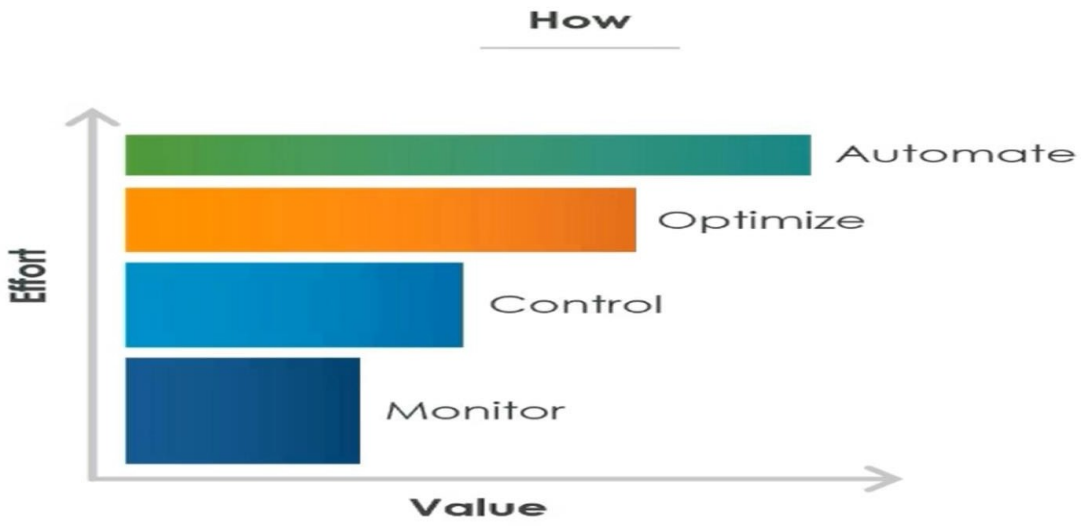
Within the Experience Stage, you'll ask yourself "why" questions. These "why questions" area series of use cases or problem statements that define why you're building the "Thing," who you are building it for, and what problems it solves. Below are some common examples of "why" questions you should be asking throughout the Experience Stage:

- What is the problem that the IoT application needs to solve?
- What business value does the IoT solution offer?
- Who will use the IoT application, and how does it help them?

Next, you'll ask yourself "how" questions. "How" questions cover what capabilities the applicationwill have to further those goals. Generally, in IoT development, there are four capabilities in escalating stages of complexity:

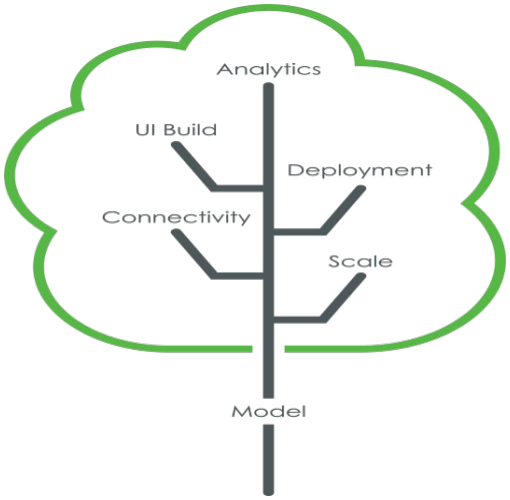
- Monitor: How do I monitor IoT devices and dashboards that retrieve information?

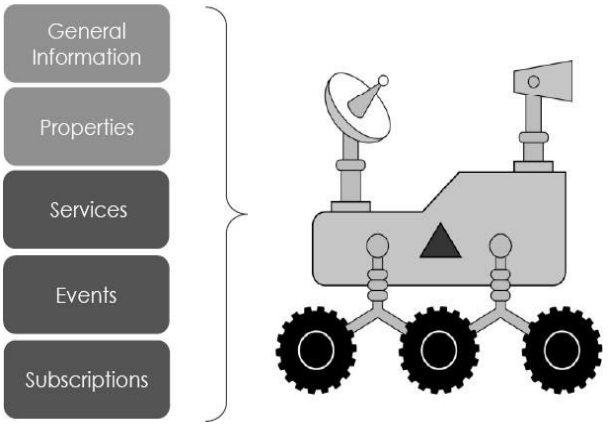
- Control: Once I can monitor a device, how can I control it? This involves sending instructions to the device that will be able to change its behavior.
- Optimize: Now that I can control it, how can I provide it with information to consistently generating a particular outcome? At this point you'll find it useful to leverage data analytics.
- Automate: How can I ensure that my desired outcomes happen automatically without userintervention?



Note that not all applications need all four capabilities. Some applications simply monitor whereasothers may monitor and control. However, you cannot achieve more complex capabilities withoutthe ones underneath it. Optimizing a device that you can't monitor is not possible.

2. The Model Stage;





The ThingWorx model is the schema of your IoT system, defining what IoT data it accesses andwhat it will do with that data.

A good analogy is to think of the model as the trunk and root of a tree, the central structure thatall other parts of the tree rely upon.

Devices connect to the model, providing the model with data. These devices may also receivecontrol requests from the model.

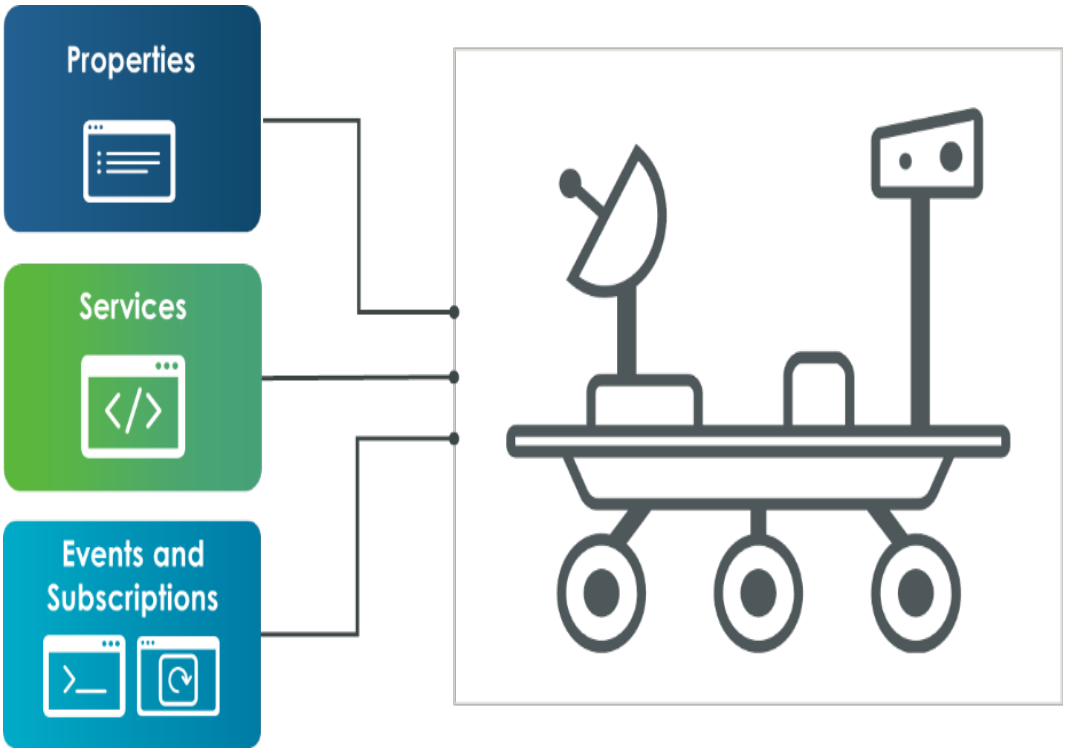
User interfaces are used to display data in the model and provide the user with the ability tosend data or commands to the model.

Analytics systems access the model to manipulate and optimize the data in numerous ways, such as predicting when a device will fail based on its properties and how similar devices have

behaved in the past. Lastly, a good model must be built with deployment and scalability in mind.You must consider the model's users and access control scheme to deploy it effectively, makingsure the devices and users have the access they need to accomplish their tasks but areprohibited from more restricted tasks and actions.

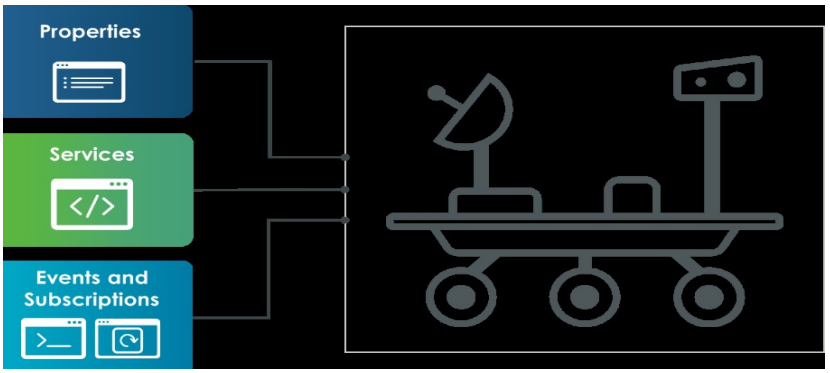
What is Thing in Modeling?

The ThingWorx model is based on a data structure called the ThingWorx Thing.



Thing:

The ThingWorx model is based on a data structure called the ThingWorx Thing.



A ThingWorx Thing represents a tangible set of information and data in the ThingWorx system.

Different things may represent different types of data. It may represent:

- A physical device, encapsulating its properties and the commands that can be done to it.
- An enterprise system, gathering data from a remote database or enterprise application, suchas Salesforce, SAP, or a mail server.
- A set of internal commands and functions that control the IoT application.

Each Thing may have attributes to support it. Four of these attributes are:

- Properties, which are status variables. They may be IoT data, such as oil pressure from a device, or internally computed data, such as the optimal speed of an assembly line as determined by advanced analytics.
- Services, which are commands that the user may issue to the Thing. For example, if a user can change the speed of an assembly line or add a user to the system, they would do so by executing a service.

- Events and Subscriptions are linked together. Subscriptions are similar to services. However, instead of a user executing them, they are executed automatically when an event occurs. For example, if ThingWorx sends an SMS message to the operator of a device when the oil pressure exceeds a certain level, that is a subscription executed automatically upon an oil pressure event.

Throughout the model stage, it is critically important to remember the experience stage. Consideryour users and consider access control. Each thing and attribute of the thing is subject to access control, permitting some users to perform the action while denying other users the same action.

Thing Properties:

Thing properties are used to describe the data points that are related to a Thing. For example, a customer may have a name property and address property. A truck may have the following properties: driver, capacity, and location.

Properties are a simple and convenient way to know the current conditions of a Thing. Properties can be static (for example, manufacturer and model number) or dynamic (for example, temperature). You set up properties based on your asset structure, business processes, and the solutions you want to deliver.

When you create a property, you can select one of the following kinds of bindings:

- My Property: A property for a local Thing. This is the default setting.
- Local Bound: Links the property to a property defined on a different Thing on the ThingWorx server. You can set this property to be read-only and it can still receive a valuefrom another

  property on the server. This is not used for remote devices, unless

you are creating a local binding to a remotely-bound property.

- Remote: When a remote device first connects to a ThingWorx server, the server binds the remote device to its corresponding Thing. Once the remote device is bound, you need to create a remote binding between each property defined on that remote device and the Remote Thing that represents the device in ThingWorx. ThingWorx uses this binding to send and receive updates for each property value from a remote device over an active Web Socket connection.

If a local Thing and a Remote Thing have a property with the same name, and another property is bound to the remote property setting, the remote property also sets the local property value.

Manage Property Bindings

Devices can range from a single sensor to a machine or vehicle, and even another system or application. When the values of the other Things change, the Thing properties that are bound to italso change.

There are three types of property bindings:

- Unbound (or My Property)
- Local Bound (bound to a Thing property on the server)
- Remote Bound (bound to a Remote Thing property on the edge)
- Binding is performed from the target's perspective. Individual property bindings can be setfor an individual property or from the Manage Bindings dialog box.
- Regular Things (non-remote) can only have Unbound or Local Bound properties.
- Remote Things can have Unbound, Local Bound, and Remote Bound properties.

Binding Remote Thing Properties

When binding Remote Things, the platform restricts only one bind per Remote Thing. If the RemoteThing is already set to bound, then subsequent attempts to bind to that Remote Thing will fail, as long as the original bind is

still active. This does not preclude multiple Things from pushing or pulling data to a given Remote Thing. However, it restricts the subscribed property handshake, so that it only happens to the actively-bound edge Thing.

The following process assumes that you have a Thing configured with a Remote Thing Thing template and the Web Sockets communication is set up to browse the Remote Things from the server side.

- From Composer, browse MODELING>Things, and then open a Thing based on the Remote Thing Thing Template.
- In the Properties and Alerts area, click Manage Bindings. The Remote tab displays the configured Remote Thing properties.
- Drag the property from the Remote list to the My Properties list to bind to that property, andthen click Done.
- Optionally, to view the bound and unbound properties, click the buttons at the top right- hand side of the screen. You can also click the Add all properties option to add all new properties.
- Click Save.
- Click the Refresh button to view the property values from your remote Thing.

Each property has a name, description, and a ThingWorx data type, known as a base type inThingWorx. Depending on the base type, additional fields may be enabled. A simple scalar type, like a number or string, adds basic fields like default value. More complex base types have more options. For example, a base type of INFOTABLE includes the ability to define a Data Shape to describe the data structure of the info table The defined base type provides context around the data stored in the property. The next table lists the base types are available for all Thing entities:

Base Type	Description
BASETYPENAME	A valid base type name.
BLOB	A binary large object.
BOOLEAN	A true or false value.
DASHBOARDNAME	The name of a dashboard.
DATASHAPENAME	A reference to a data shape in the model. Uses special handling.
DATETIME	A formatted date and time.

GUID	Globally unique identifier. When a GUID property is created, a GUID is automatically created if a default value or value is not setby the user. The following behaviours are followed for persistentand non-persistent GUID properties and are also applicable in a federation scenario: •If the property is set as persistent or not persistent and a default value or value is not set by the user, a new GUID value is generatedeach time the Thing is initialized. If a system or Thing restarts, a new value is generated. •If the property is persistent and a default value or value is set bythe user, a new value is not generated when a Thing or system restarts.
GROUPNAME	A user group name.
HTML	HTML content.
HYPERLINK	A standard URL (such as www.ptc.com).
IMAGE	Binary data that can be rendered as an image.
IMAGELINK	A URL link to an image.
INFOTABLE	A standard representation of data in ThingWorx that is similar to aSQL result set. There are a number of built-in services for building,consuming, and parsing an info table.
INTEGER	A number that can be written without a fractional component.
JSON	A JavaScript Object Notation (JSON) object.
LOCATION	Standard World Geodetic System (WGS) 84 coordinate, expressedas [longitude, latitude], elevation.

LONG	The LONG type should be used when a required range is longerthan what the INTEGER base type provides.
MASHUPNAME	A reference to a ThingWorx mashup. Uses special handling.
MENUNAME	The name of a menu.
NUMBER	A number. Exponential values are allowed.

PASSWORD	A masked password value. See Passwords for additional information.
QUERY	A JSON object that includes an array of filters. Each filter shouldinclude value, type, and fieldname.
SCHEDULE	A crone-based schedule configured using the Schedule Editor.
STRING	Any amount of alphanumeric characters.
TAGS	ThingWorx tag values.
TEXT	Any amount of alphanumeric characters. The difference withSTRING is that TEXT is indexed.
THINGCODE	A numerical representation of a Thing containing a Domain ID andInstance ID. For example, 2:1.
THINGNAME	A reference to a Thing in the model. Uses special handling.
THINGSHAPENAME	A reference to a Thing Shape in the model. Uses special handling.
THING TEMPLATE NAME	The name of a Thing Template.

USERNAME	A reference to a ThingWorx user.
VEC2	A collection of two numbers. For example, 2D coordinates x and y.
VEC3	A collection of three numbers. For example, 3D coordinates x, y,and z.
VEC4	A collection of four numbers. For example, 4D coordinates x, y, z,and w.
XML	An XML snippet or document

The values entered in the Min Value and Max Value fields are for information only. Actual values less than or greater than the specified values are accepted.

Aspects of Properties:

Properties can have the following aspect settings:

- **Persistent**

If selected or set to true, each value change is persisted to the database. Persistent property database writes occur asynchronously to avoid deadlocks. While the property value is set immediately, the database write occurs asynchronously at a later point in time.

The following validations occur before a persistent property value is written to the database:

- The Thing must still exist.
- The Thing must have an id.

- The Thing id and the pending write id must match.
- The Thing must still define the property with the same name as the pending write.
- The defined property must still be persistent.
- Updates and restarts do not affect queue processing.
- Read Only: If selected or set to true, the data is static and cannot be written at run time. Theonly way to change the value is by changing the default value. This is useful for static configuration data.
- Logged: If selected or set to true, the property value is automatically logged to a value streamwhen the data changes (based on the data change type).

If the data change event fails to fire under certain circumstances, the value stream entry may not be logged, but the property value set is retained. It is possible that properties can be set on an entity, but the corresponding value stream write may be dropped, due to the queue containing those writes becoming full and unable to empty to the database. This may occur if the volume of incoming writes is greater than how fast the queue is configured to empty. This can be tuned in platform settings.json per persistence provider. Loss of connection from ThingWorx to the databasemay also cause the queue to back up and not empty in a performant manner. Data Change Information Data Change Type

This setting specifies when a data change event is triggered by a property value change. You use itwhen other processes must be initiated based on the value of the property. Each subscriber is senta change notice with an info table that contains the old and new property values.

For example, you can set up a subscription for changes to the Delivery Schedule property. If the schedule changes,you can notify the driver via SMS.

Data Change Type options are as follows:

- Always— Fire the event to subscribers for any property value change.
- Never —Do not fire a change event.
- On— For most values, any change will trigger an event. For more complex base types, suchas info tables, event rules may be different.
- Off — Fire the event if the new value evaluates to a Boolean false.
- Value— For numeric values, if the new value has changed by more than the threshold value, fire the change event. The threshold value is specified with the data Change Threshold aspect(a NUMBER). For non-numeric values, this setting behaves the same as Always.

Remote Binding Information:

The next table lists the options that are available if the Binding option is set to Remotely bound

Option	Description
Remote Property Name	The name of the property, as it exists at the Edge. The property name and the bound Edge Thing property name need not be the same.

Cache Method	Cache method provides the following ways to read bound Edge property values: • **Read from server cache** prohibits server requests to the Edge property value. It only retrieves the value from the server. Any updates to the server-cached Edge property value depend on the Edge property's **Data Change Type** and the scan rate (Edge property value push definition). Without the proper settings at the Edge property, it is possible for the server to never have the Edge property value and to only return the server property default value. If the **Data Change Type** of the Edge property to which you are binding equals ALWAYS or VALUE, the cache type will default to this setting. • **Fetch from remote every read** retrieves the Edge property value from the Edge for every request. There is no caching with this option. If the **Data Change Type** of the Edge property to which you are binding equals NEVER, the cache type defaults to this setting. • **Cached for specific time** controls the frequency of requests to the Edge property. After the first request, the server accesses the Edge property for its value and does not make another request to the Edge property for the defined number of seconds. Note that the Edge property may update the server value (via push) during that time.
Cache Interval	The time period (in seconds) for which the server caches the Edge property value before a request for the property value retrieves it

	from the Edge. The value is always retrieved from the Edge at the first request.
Start Type	Specifies the value used to initialize a remotely-bound property when its Thing starts or restarts. This initialized value does not trigger a property change event. • **Use Default Value** — sets the initial value of the property to the specified default value, despite the edge-side value. If the property is persisted, then the initial value will be set to the last value persisted to the database. • **Read Edge Value** — queries the edge for its current value, so that the value on the server is always in sync with the value on the edge, even when the Thing restarts on the server.

Push Type	**Push Type** applies only to Edge Enhanced Thing properties. These components can push their value changes to the server. You can configure this ability using the server property binding. • **Pushed based on value change**: you can configure a value change threshold. When you use this setting, you can also set the **Push Threshold** value, a dead band that must be exceeded before a new value is pushed to the server from the edge. • **Never pushed** • **Always pushed**
Push Threshold	This option is available if **Push Type** is set to **Pushed based on value change**. It specifies the range (plus or minus) around the edge property value for the property value push to occur. The property value must change by more than the specified value.

When Disconnected	Specifies how the remote property value binding should be handled if the connection to the Remote Thing is temporarily lost. • **Ignore values that are changing** while disconnected. • **Fold all the changes into a single last changed value** — Send the last changed value when the connection is restored.
Timeout	The timeout used for calls to Remote Things during a property reador write. • **Use system default** — The default is 30 seconds. • **Custom timeout**. Add to **Timeout Interval (sec)**.

Naming Properties:

Name is a required field that uniquely identifies the property.Names are case-sensitive.
You cannot include the following in the name:

- Spaces
- Some special characters

You can include - (hyphens), _ (underscores), $ (dollar sign), and @ (at).
If a persistent property has an @ or $ symbol in the name, you cannot export the entity thatcontains this property to XML.
If you include hyphens in the name, they must be referenced using bracket notation. In theexamples below, temp-01 is the property name:
var result = me['temp-01'];

var result = Things['Thing

Name']['temp-01'];

Leading numbers
You can use numbers within and at the end of the name.

Services:

Thing services are functions that a Thing can perform. Each Thing can have one or more services.You can define services at the Thing Shape, Thing Template, or Thing level. A simple example of aservice is a query written for a database Thing.

There are several implementation methods, or handlers, for services depending on the template you use. Script, SQL Query, and SQL command are examples of handlers. Additional handlers maybe available depending on the specific functionality of a Thing, such as edge Things.

The specific implementation of a user-defined service is done via a server-side script (currently through SQL or JavaScript). The service can then be invoked through a URL, a RESTclient capable application, or by another service in ThingWorx.

When you create a new service, you can define input properties and an output. Inputs andoutputs can be any of the standard ThingWorx data types. Each service can also have individual
run time permissions defined in the service definition. A service is not required to have inputs oroutputs, but usually there is one or both. For example, if you want to send a delivery schedule toa truck, the truck Thing may have a service with an input named Delivery Schedule and a type ofXML. The service could take the incoming data and put a Thing property of the truck into a Data Table.

If you want to send the output directly to a mashup widget, you should choose an output of base type INFOTABLE. If you choose to output an info table, you will need to select a Data Shape. The Data Shape tells the application which columns and data types will be returned so it can renderthe data. You can choose any number of inputs based on your requirements. For example, your output could be a SQL query against a database that returns

data to a mashup. The service is automatically part of the REST API of the ThingWorx application server (the same as all definitions within your model). You can use the service through a REST callfrom another application or in a mashup.

When calling a service, if you define the output as an info table, you can ask for the result set as HTML, JSON, or XML using a URL call and the Accept URL parameter (for more information,see REST API). Because of this flexibility, and the ability for the mashup environment to be able toeasily consume an info table, it is recommended that you use this format as the default design pattern. Specific needs, such as an XML schema output, can be addressed as required.

Once you have defined your script function interface, you can implement the service by clickingthe Handler column for the service. This opens to the service implementation editor. In the implementation editor, you choose the handler (SQL Query or Script). SQL Query is only availablefor a database entity. The Script implementation is a server-side Java Script engine.

With SQL Query, write a query in the syntax you use for the source database. You can use service inputs as parameters in the query like you would with prepared statements. If your outputproperty is an info table, you don't need to manipulate the results. The query result appears in the info table and is available as output.

The script handler is a powerful way to use all the data, Things, and services on the server to meet the needs of your application. You can perform calculations and lookups, call services, oraccess properties from other Things in the model. Once you select script as the handler, you will be presented with a number of script helpers. You can see the inputs for the script, properties,

services, and events of the Thing you are currently editing, you can paste these inputs into the script window with a double click. You can also browse the properties, services, and events of anyother entity in your system. You can combine all the model capabilities within your service.

If the script is called from a web page or URL, it runs in the context of the logged-in user. Ifthe user does not have access to the run time services, properties, or events of any entity in the script, the script may fail.

The script implementation editor also has syntax helpers and code snippets to make thecreation of a script easier. By default, the script timeout setting on the Thing Worx platform is 30 seconds. If a script runs longer, the platform terminates the execution. A Thing Worx administrator can configure thescript timeout in the basic settings section of the platform-settings.json file.

Asynchronous Services

Asynchronous services create and execute in their own thread. Asynchronous services cannothave a return value because when you run them, the thread is created and runs independently on the platform. If called from within another service, the calling service will not wait for a sync service to complete. This can be very useful for long running services, especially the ones on timers that update the background data structure or perform system maintenance tasks.

If you choose a sync in the New Service editor, the Queue Calls option appears. This option is for remotely bound services that queues the service executions if the Remote Thing is not connected.Thing Worx queues each attempt at service execution, and then executes them in order when the Remote Thing is connected again.

Statistics for Services

The Utilization Subsystem collects metrics for JavaScript services that are terminated due to atimeout

Thing Subscriptions:

Subscriptions are services that receive and respond to events. A subscription has a source, usually a Thing. A Thing can have a subscription to an event that responds with an action. For example, if an entity fires a Motor is overheating event, it can subscribe to that event by triggeringa Turn motor off subscription. Things can inherit subscriptions from the Thing Templates and ThingShapes that they use.

A subscription is similar to a standard service, but it is explicitly linked to an event. This allowsyou to decouple the event from the code that responds to it. Like with a service, you can implement

custom business logic to react to the event. You can leverage the capabilities of the model by sending emails through a mail server Thing, writing to a database, or calling any services availablein the platform. A subscription does not have an explicit return output as a service does. However,a subscription can call any other service in the model to which the thread security context has access. The thread security context of a subscription is set to the same thread security context of the event that was fired. You can use the same JavaScript editing environment that is used to implement services.

Subscriptions have a defined input, which is the data packet issued by the event and referredto as event data. If the entity subscribes to a defined event, the event data is passed to the subscription function. The event data is described by the event Data Shape. Within the subscriptionimplementation, the data that is passed from the event acts as the input to the script function.

For example, if an entity is subscribed to a Thing property data change event, the subscriptionscript function is called. As a result, the Thing property value, along with other relevant data from the event, are passed to the function as part of the event data.

Many entities can subscribe to the same event. Each entity receives a call to the subscription with the passed event data. An entity can take any action from the subscription script to achieve the solution requirements.

Some advantages of using this technique versus using a service that is called from another serviceinclude:

- An event can be subscribed to by one or many subscriptions.
- Events are invoked based on system activity, and no user interaction is necessary.
- If more than one Thing is subscribed to an event, you can use a subscription instead ofchaining multiple services.

Multiple subscriptions to the same event defined on the same Thing, or duplicate subscriptions,are supported in ThingWorx version 8.4.0 and later.

Multiple Subscriptions

As of 8.4.0, subscriptions have a user-defined name as the unique identifier. Entities can have multiple subscriptions to an event on a Thing. For example, if an entity fires a Motor is overheating event, it can subscribe to that event with both a Turn motor off subscription anda Create Work Order subscription to have a maintenance check of the engine. Any number of othersubscriptions can also be created for that event.

If a Thing Template or a Thing Shape implements a subscription to an event, the Things that use that Thing Template or Thing Shape can also create subscriptions to the same event, without requiring a workaround to take additional actions when those events are fired.

Programmatically Enabling and Disabling Subscriptions

Subscriptions can be programmatically enabled and disabled

by using the Enable Subscription and Disable Subscription services.

Service Notes

- You cannot enable or disable a subscription on a Thing that was inherited from its Thing Template or Thing Shape. You can only enable and disable subscriptions at the level at whichit was defined.
- Dynamic subscriptions are in-memory only. If you restart ThingWorx or restart either of the Things involved in the subscription, the subscription is lost. For example, if you have a dynamic subscription to a timer event on a timer from another Thing, and you restart the timer Thing, the subscription will no longer be triggered.
- Non-administrator users can use these services.

Content crawler content crawler Thing is used to call a service on another entity. A content crawleris used to retrieve data and store the data in the Data Table of the content crawler Thing. On a separate entity from the content crawler Thing, you must define a service that fetches data and returns an info table of that data back to the content crawler. The content crawler then maps the incoming fields and the tags to the fields used in the Data Shape for the content crawler. Each rowis added as a new entry to the Data Table on the content crawler Thing. The index of the content crawler's Data Table works in the same way as a Data Table entity.

Creating a Content Crawler

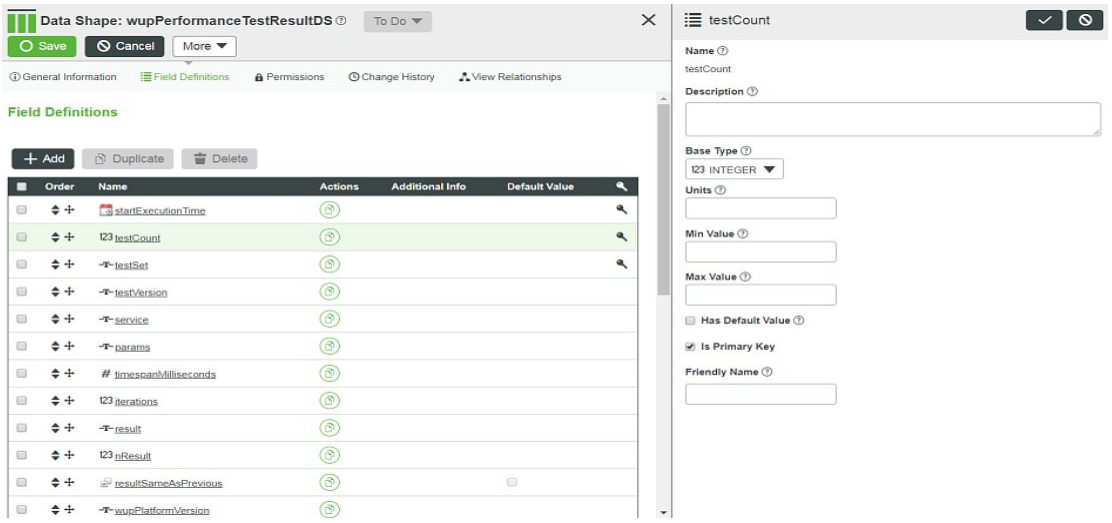
To retrieve data from the Data Table of an entity into the Data Table of the content crawler Thing,do the following:

1. Create a Data Shape and define fields to use in a Data Table. To create a Data Shape from Composer, browse MODELING>Data Shapes, and then click the new

button.

a. Enter a name and description.

b. In the Field Definitions area, click the Add button



c. In the new field definition pane, enter appropriate information, and then click.

2. Create a data table with the Data Shape created in the previous step. To create a Data Tablefrom Composer, browse DATA STORAGE > Data Tables, and then click the New button.

a. Select a Data Table template, and then click OK.

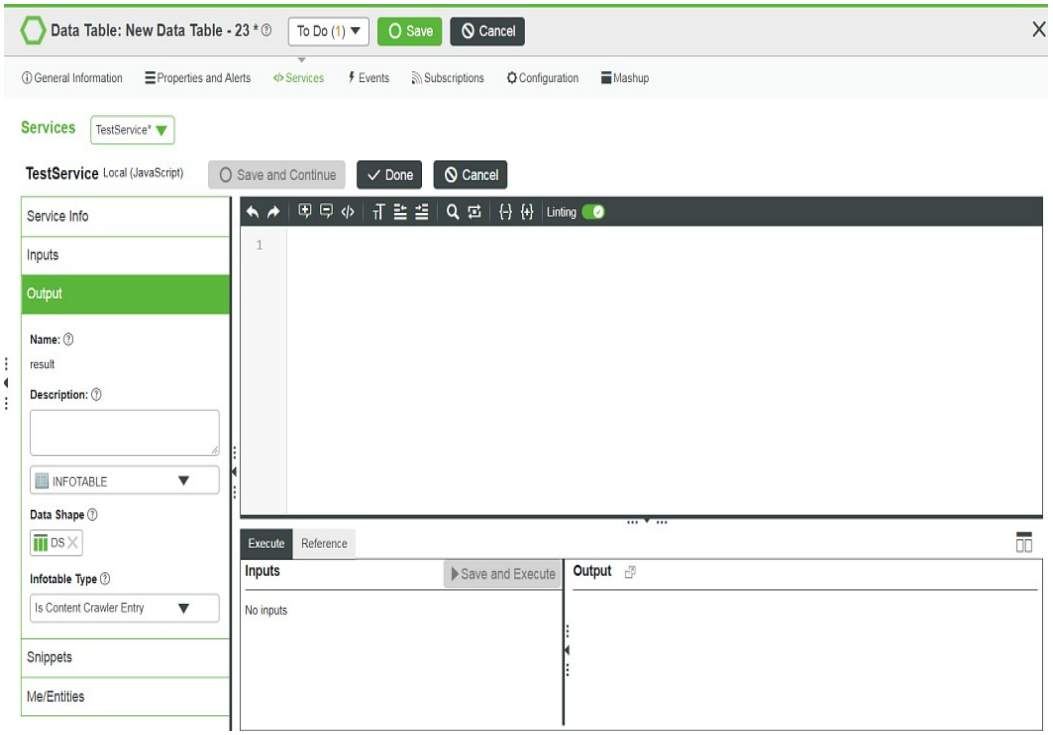
b. Enter the name, description, and select the Data Shape you created in the previous step.

c. In the Services area, create a custom service by clicking Add.

d. In the Output area, select INFOTABLE from the drop-down list.

e. Select the Data Shape created in the previous step.

f. Set the Info Table Type as Is Content Crawler Entry, and then click Done.



3. Create a new Data Shape for the content crawler thing.

You can create a new content crawler-specific Data Shape, or you can use the same Data Shape thatwas used in the Data Table created in step 1. Although this step is optional, we will use a new DataShape for the content crawler Thing in this example.

a. Create a new content crawler thing:

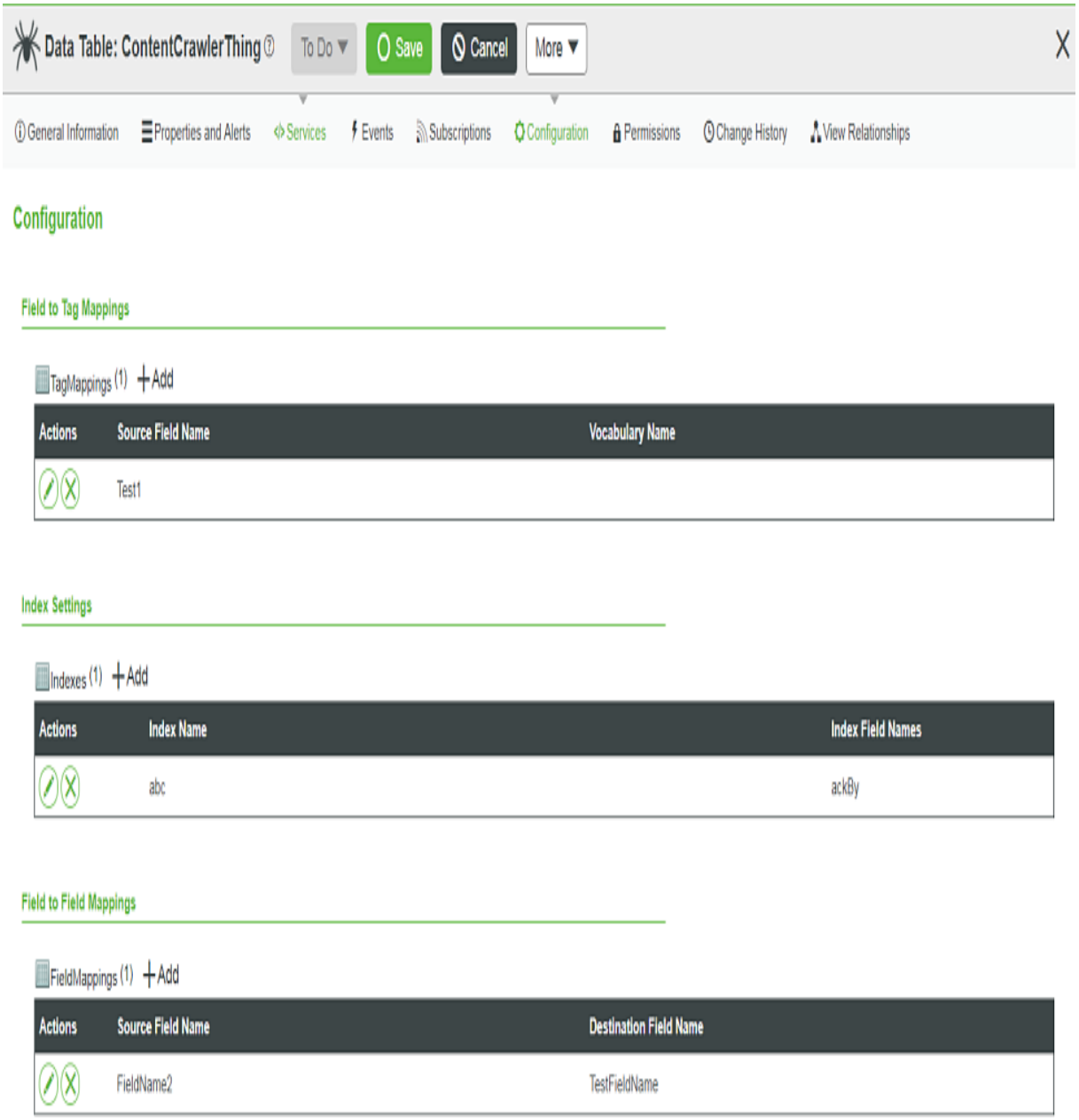
a. From Composer, browse MODELING > Things, and then click the New button.

b. Enter a name, and in the Base Thing Template field, select Content Crawler.

c. In the Data Shape field, select the Data Shape you created in the previous step, and thenclick Save.

Content Crawler Configuration

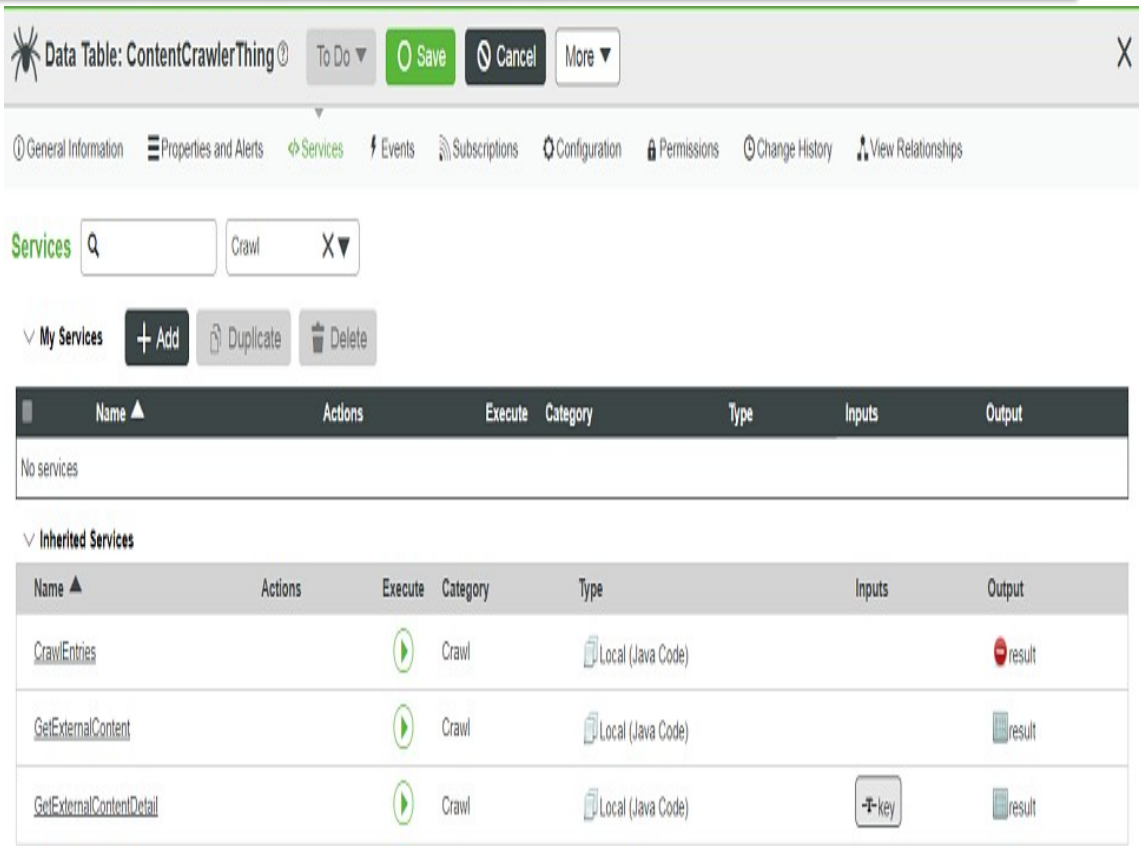
The Configuration area for the content crawler Thing contains configuration tables that allow youto map fields from the retrieved data.



The Field to Tag Mappings configuration table maps the values of a field to tags in a data tag vocabulary

- When the data tag vocabulary is dynamic, any value that is mapped from the data has a termautomatically entered into the vocabulary.
- When the data tag vocabulary is not dynamic, any value that is mapped from the data has apre-defined term representing the value to be mapped properly.
- For example: Testing Vocab: false; TestingVocab:iAmAString. The first part is the boolProp value, and the second part is the string Prop value.
- The Index Settings configuration for a Data Table allows you to define additional table indexes. This is similar to a relational database table, where in addition to the primary key (the primary key is defined in the Data Shape), you need to query the table based on other fields. You should create an index for each set of filter criteria commonly used. Doing so hasa significant impact on query performance.
- The Field to Field Mappings configuration table maps the fields from the retrieved data to the fields defined on the Data Shape of the content crawler Thing.

If the same Data Shape is used on the content crawler Thing and for the info table returned from the content crawler service, field mappings are handled automatically



The following services are unique to the content crawler Thing:

•Crawl Entries — Purges all of the Data Table entries for the content crawler, and then executes GetExternal Content.

•Get External Content — Executes the service defined on the General Information area of the content crawler Thing. An info table of retrieved values is returned from the service. No modifications to the Data Table for the content crawler are performed.

•Get External Content Detail — Retrieves a specific content item by key.

Thing Templates:

Thing Templates provide base functionality with properties, services, events, and subscriptions that Thing instances use in their execution. Every Thing is created from a Thing Template. A Thing Template can extend another Thing Template. When you release a new versionof a product, you simply add

the additional characteristics of the version without having to redefinethe entire model. This model configuration provides multiple levels of generalization of an asset. A Thing Template can derive one or more additional characteristics by implementing Thing Shapes.

When you make a change to the Thing Template, the change is propagated to the Things that implement that Thing Template, simplifying the model maintenance.

A Thing Template can be used to classify the kind of a Thing or asset class or as a specific product model with unique capabilities. If you have two product models and their interaction with the solution is the same (same properties, services, and events), you can model them as one Thing Template. You can classify Thing Templates to aggregate Things into collections that are useful in mashups. You can separate Thing Templates for indexing, searching, and future evolutions of the products.

System-Defined Thing Templates:

There are a number of system-defined Thing Templates that can be used to create Things forspecific tasks. Some of these Thing Templates can be useful as utilities for various services and capabilities when building applications.
The system-defined Thing Templates are the following:

- Blog — A blog Thing is used to implement a blog, comments, and/or discussion forum collaboration objects in your mashups.
- Content Crawler — A Thing designed to handle a specific interface to an external system or content area. You define a service to get a list of external content to be indexed and also a service to retrieve the details

for each content object, ThingWorx will then index the data and make it available via the ThingWorx search functions.

- Database — A JDBC connection to any third-party relational database system.
- Data Table — A Data Table is similar to a table in a relational database and can be used to store transactional rows of data in ThingWorx.
    - File Repository — A defined ThingWorx entity for external file content to be stored. When you transfer files to/from an Edge Thing, you do so to/from a specific repository. A file repository points to a folder in the server's Thingworx Storage/repository folder. The servicesof a file repository allow you to view and manipulate files in its folder.
    - Generic Thing — A base Thing with minimal inherited characteristics. It is a best practice to define a custom Thing Template. However, there may be cases where you have a one-off Thing definition and you want to use a generic Thing.
    - Mail Server — A mail server Thing can be created if you want to send email messages from your application.
    - Edge — An Edge Thing is a device or data source that is installed on another server, usually through a firewall to a different network. An Edge Thing communicates to the server througha locally installed EMS. An example of an Edge Thing is an OPC-DA server system-Defined Thing Templates;

1. Edge Database— An Edge Database Thing is for communicating to an OLE-DB or ADO.NET database or data source on a different server or workstation. Examples of an Edge Databaseis Microsoft Excel or Microsoft Access.
2. Edge Enhanced— A server model Thing corresponding to a remotely installed device or data store that needs to support remote desktop tunnelling or file transfer.
3. Scheduler — A scheduler Thing can be used to run jobs based on a crone pattern, for example, once a day or once an hour.
4. Source Control Repository — A source control repository can point to any folder on the server's file system, which can be the root of your local repository. It is usedby Import/Export > Source Control Entities.
5. Stream — Time series data storage.
6. Timer — A simple timer that fires an event on a defined interval.
7. Wiki — A collaboration object for sharing documents and related comments within your mashups.

When you create a specific instance of one of the system Thing Templates, you can configure it tomeet your business needs and your IoT environment.

System-Defined Remote Templates:

Several system-defined Thing Templates can be used to communicate over Web sockets to edge devices or data stores. Remote Thing is the naming convention for using Web sockets to communicate to another node, or Thing, in the network. The Thing Templates for the WSEMS andthe SDKs are as follows:

1. Remote Database — A remote OLE-DB data source.
2. Remote Thing — A Remote Thing with no file transfer or tunnelling needs. Also used for OPC-DA data source things. Supports properties, services, and events.

3. Remote Thing with File Transfer — A Remote Thing plus file transfer enablement.
4. Remote Thing with Tunnels — A Remote Thing plus tunnelling enablement.
5. Remote Thing with Tunnels and File Transfer — A Remote Thing with file transfer and tunnelling.
6. EMS Gateway — The EMS Gateway Thing Template is used when you wish to be able to address the WSEMS as a standalone Thing. This may be useful in situations where the WSEMSis running on a gateway computer and handling communication for one or more Remote Things, which may reside on different IP addresses within a local area network.
7. SDK Gateway — Similar to the EMS Gateway, but to be used when you are using an SDK implementation as a gateway.

In addition to the above Thing Templates, the following remote templates can be used in a federated storage scenario, where you want to offload the persistence objects to another server that is optimized for disk IO:

1. Remote Stream — Create a local proxy object to a stream Thing that is running and persistingdata on another ThingWorx server.
2. Remote Value Stream — Create a local proxy object to a value stream Thing that is runningand persisting data on another ThingWorx server.
3. Remote Data Table — Create a local proxy object to a Data Table Thing that is running andpersisting data on another ThingWorx server.
4. Remote Blog — Create a local proxy object to a blog Thing that is running and persisting dataon another ThingWorx server.
5. Remote Wiki — Create a local proxy object to a wiki Thing that is running and persisting dataon another ThingWorx server.

Creating Thing Templates with an Extension

Thing Templates created with an extension are basically the same as those created in ThingWorx Composer. They are base templates that are used to create Things with the same properties, configuration parameters, services. The difference between creating them in Composer and inside an extension framework is the language used for the services and visibility of those services.

Composer Template:

- Uses JavaScript for services
- Source code is visible
- Extension SDK Template:
- Uses Java for services
- Source code is not visible
- Can define configuration values

Thing Shapes:

Thing Shapes provide a set of characteristics represented as properties, services, events, and subscriptions that are shared across a group of physical assets. A Thing Shape is best used for composition to describe relationships between objects in your model.

Thing Shapes promote reuseof contained properties and business logic that can be inherited by one or more Thing Templates. In ThingWorx, the model allows a Thing Template to implement one or more Thing Shapes, similarto a class definition in C++ that has multiple inheritance. You can override business logic in the inherited services from the Thing Shape if you explicitly define the service to allow override in theparent object definition.

When you make a change to the Thing Shape, the change is propagated to the Thing Templates and Things that implement that Thing Shape, simplifying the model maintenance.

A use case for a Thing Shape is if you have multiple product lines using the same ERP system. Consider a company that has two business units: one makes residential lawn tractors, and the other makes commercial agricultural equipment. The lawn tractors and agricultural equipment do not have common data or behaviours. However, they are both ERP assets that can be tracked.

Both have information on the customer and on the service ticket system in the same CRM system. To implement these interfaces as a physical asset only once, you can insert business logic in a Thing Shape. For example, you can implement a way to get relevant data from an ERP system into an ERP

Connector Thing, represented as a Thing Shape. The ERP Connector Thing can have configuration data that knows how to reach the ERP system (for example, an IP address), how to authenticate against it (for example, using a technical user), and how to handle request responses.

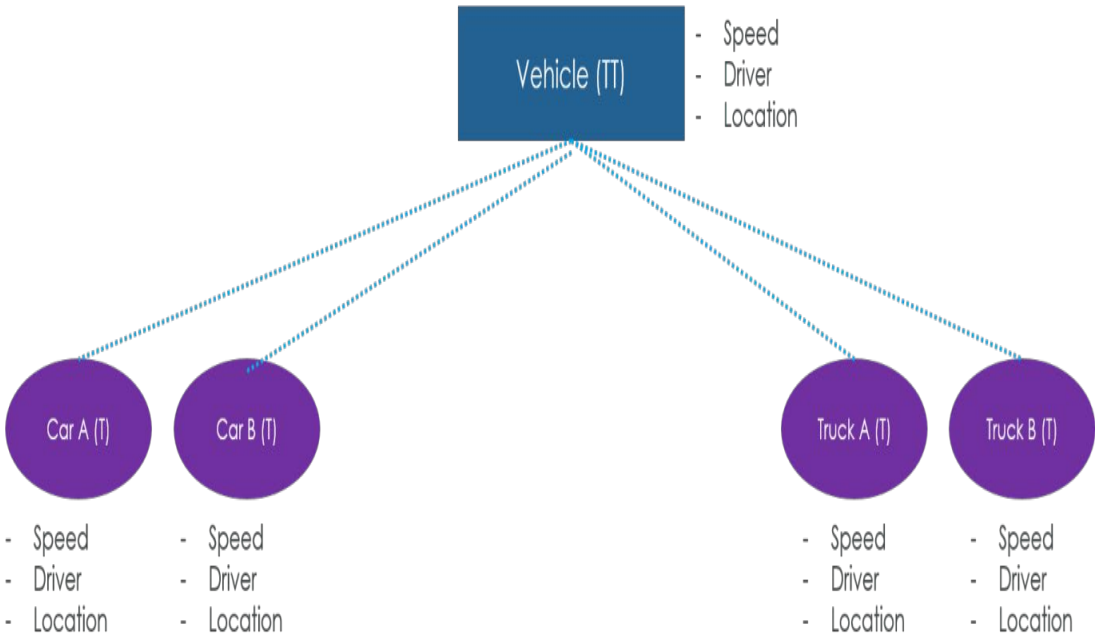
You should implement the request response functionality using services in the ERP Connector Thing. Then you can define specific functions to get request data from a Thing Shape forthe application. The Thing Shape should have basic data represented as properties (such as Locationand ERP Asset ID), services that get asset specific data (such as Get My Open Work Orders, Get MyWork Order History, and Get My Customer Entitlements). Then, Thing Templates for both lawn tractors and agricultural equipment can inherit capabilities from the Thing Shape and have access to ERP data through the encapsulated business logic in the Thing Shape. Creating Thing Shapes withan Extension

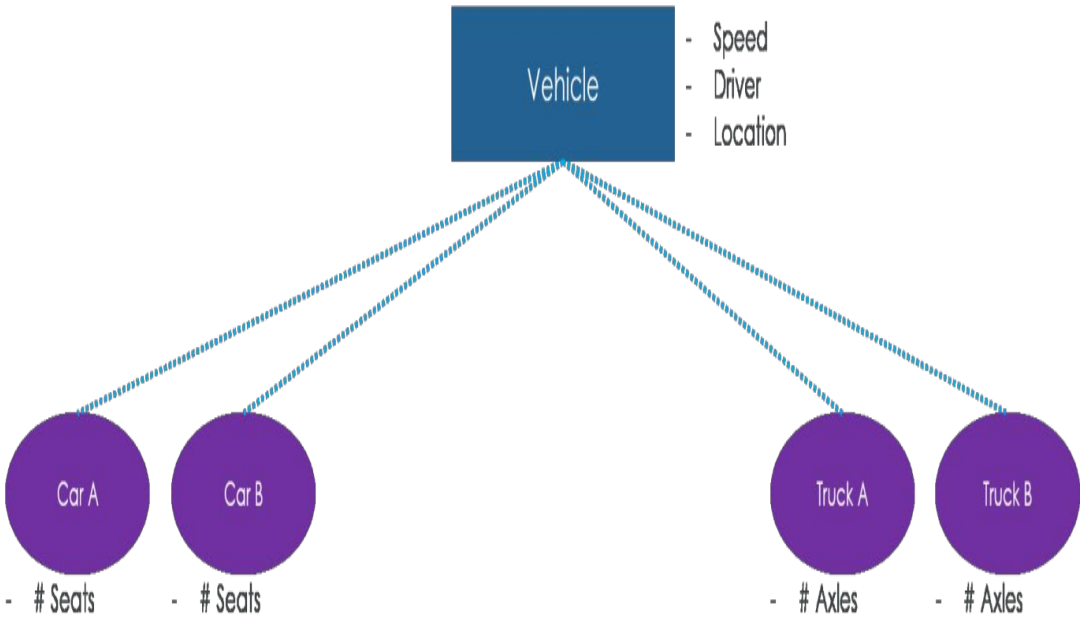
Thing Shapes created with an extension are similar to those created in ThingWorx Composer. Theyare base templates that are used to create Things with the same properties, configuration parameters, services, and so forth. The difference between creating them in Composer and inside an extension framework is the language used for the services and visibility of those services.

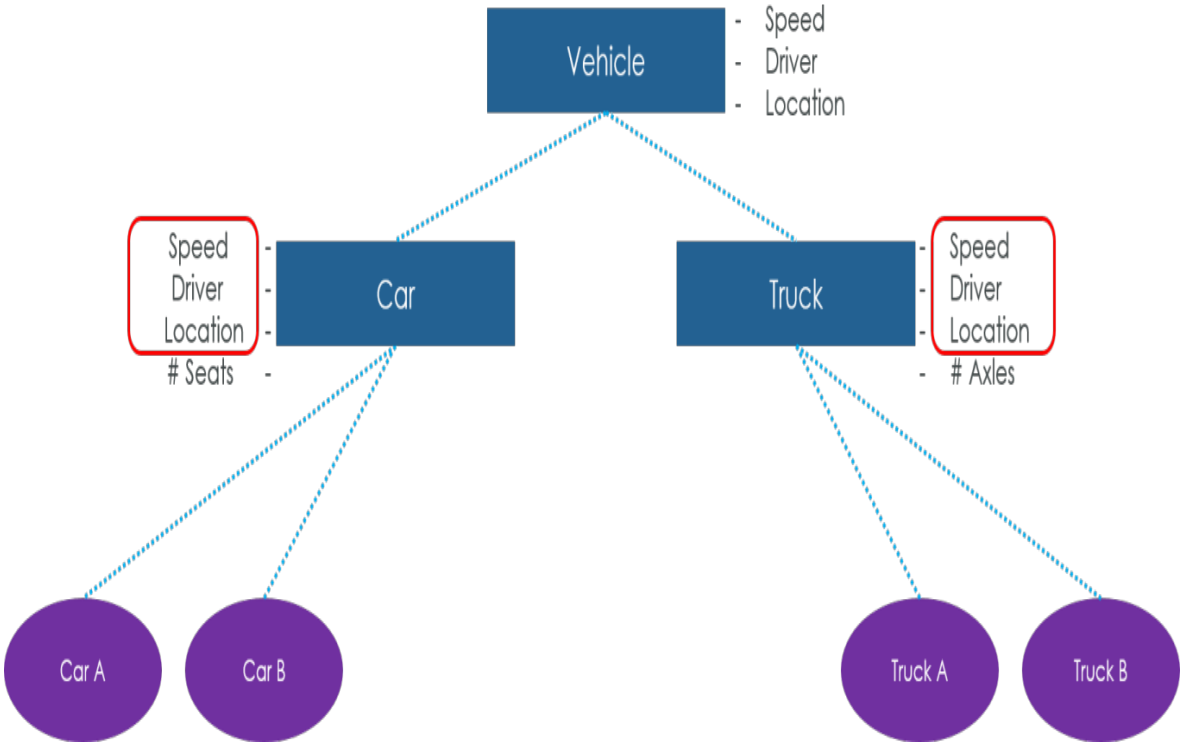
Composer Template:

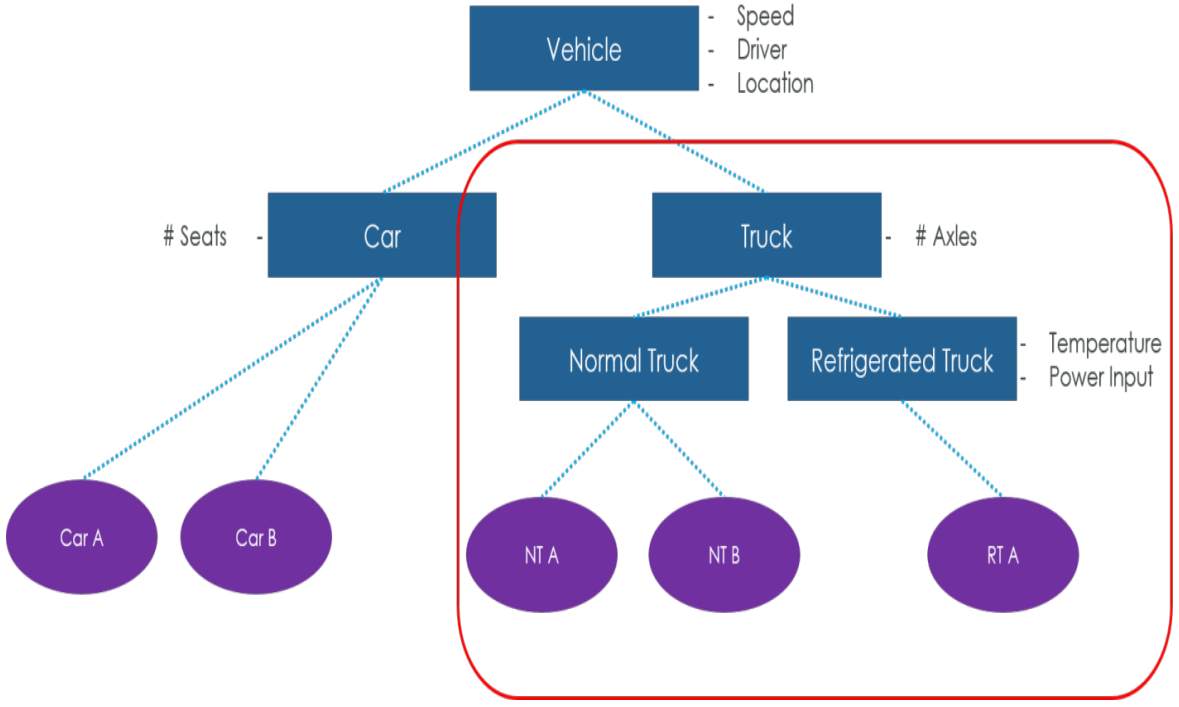
- Uses JavaScript for services
- Source code is visible
- Extension SDK Template:
- Uses Java for services
- Source code is not visible

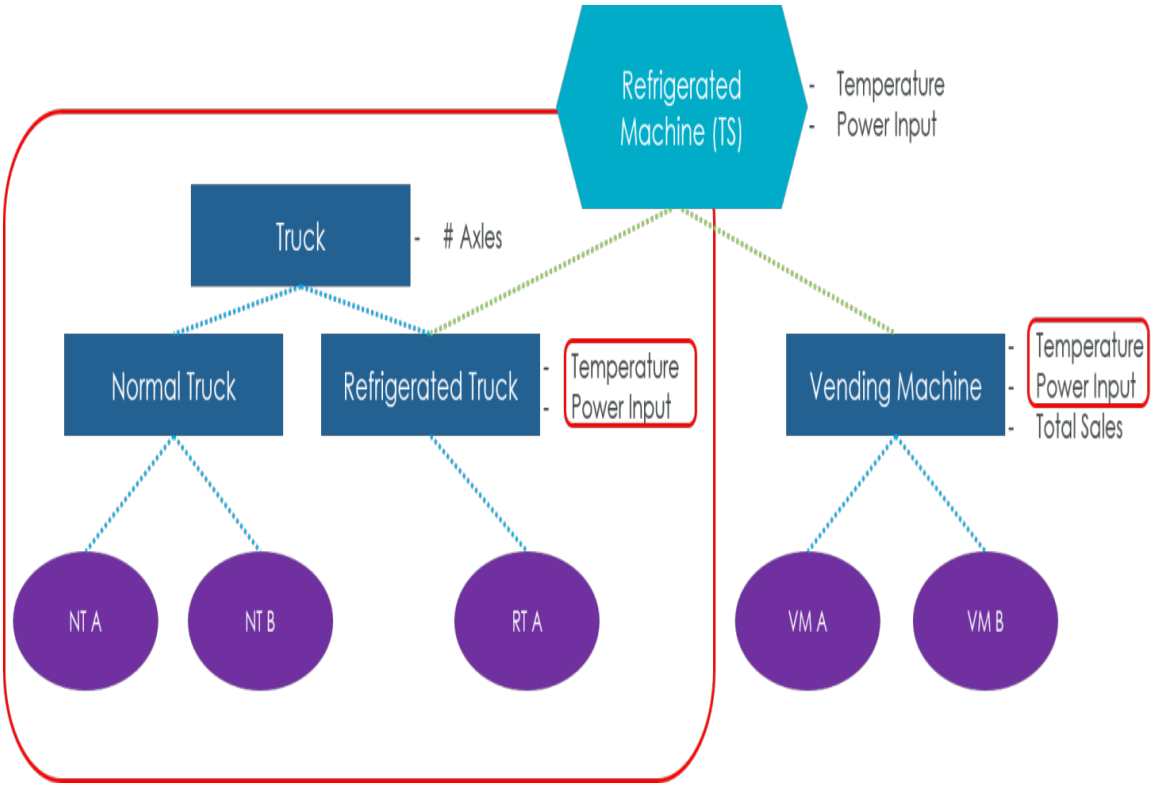
Thingworx hierarchy:

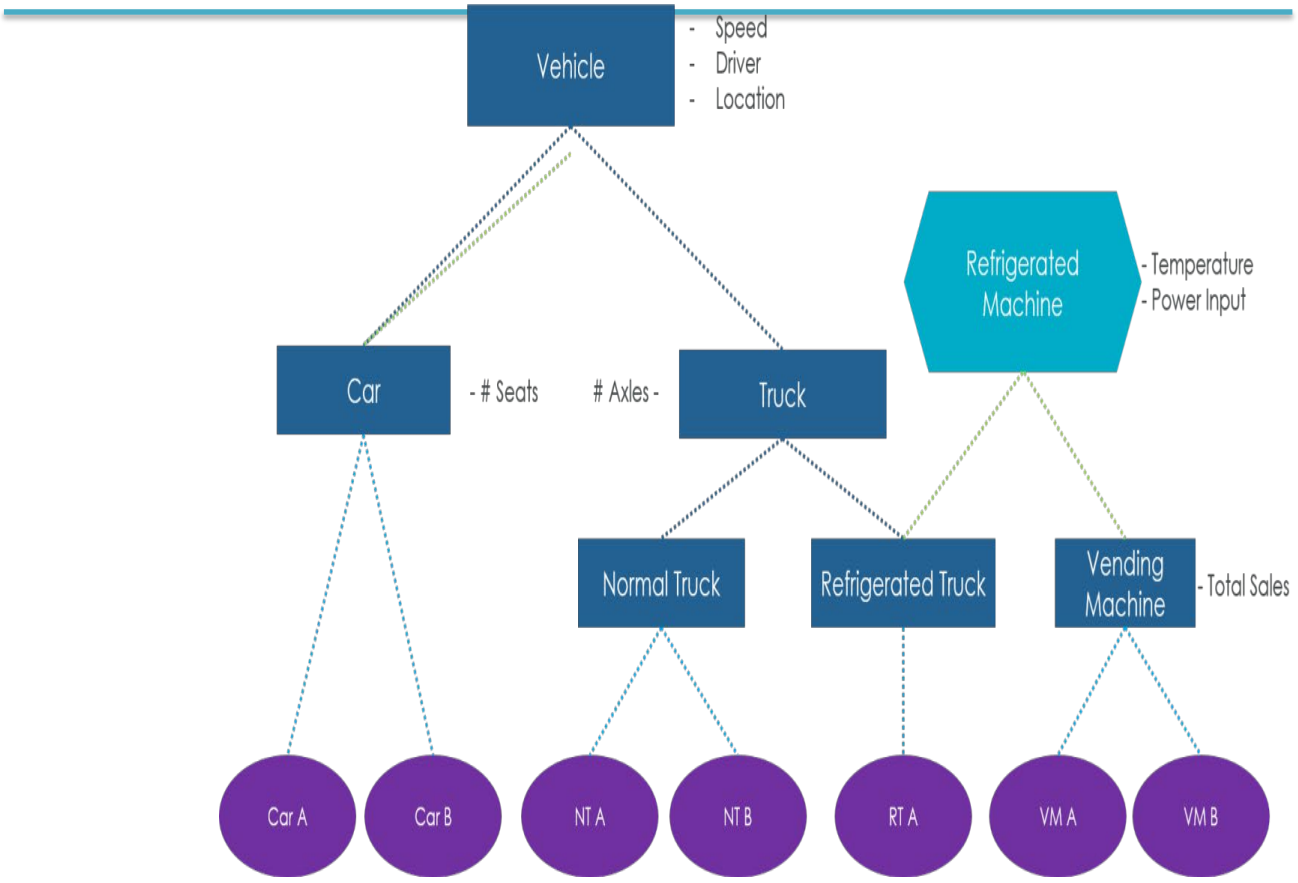





Vehicle
- Speed
- Driver
- Location
Speed
Driver
Location
Seats
Car
Truck
Speed
Driver
Location
Axles
Car A
Car B
Truck A
Truck B

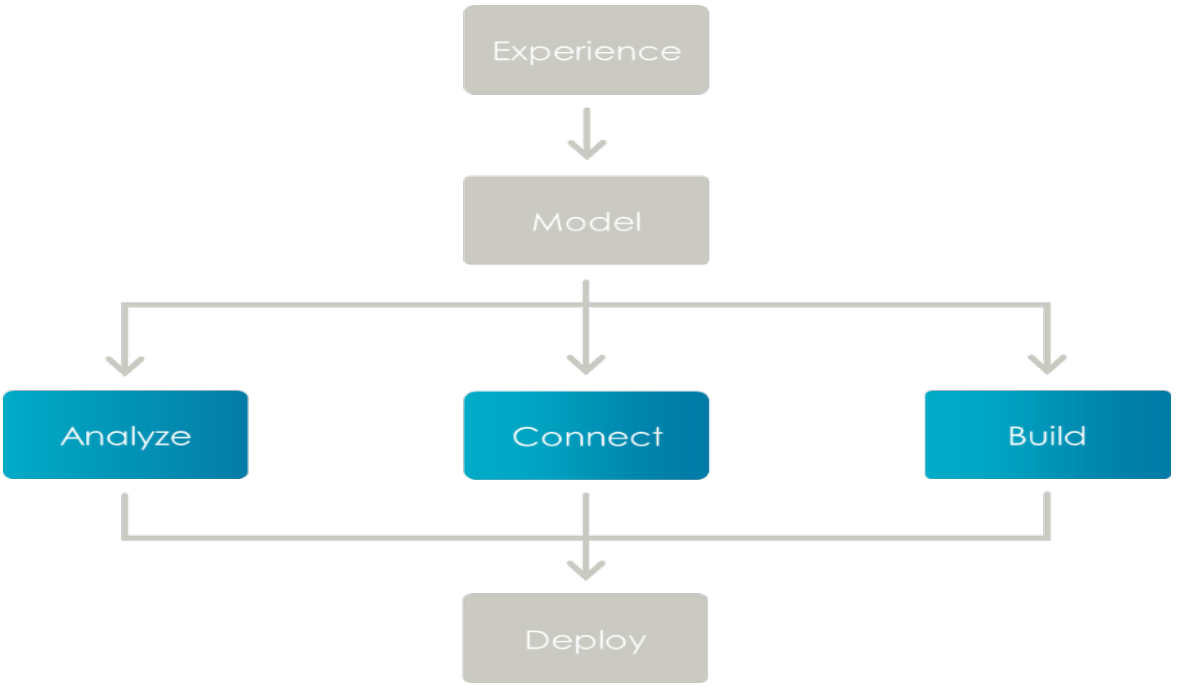

Vehicle
- Speed
- Driver
- Location
Seats
Car
Truck
Axles
Normal Truck
Refrigerated Truck
Temperature
Power Input
Car A
Car B
NT A
NT B
RT A


Refrigerated Machine (TS)
- Temperature
- Power Input
Truck
- # Axles
Normal Truck
Refrigerated Truck
- Temperature
- Power Input
Vending Machine
- Temperature
- Power Input
- Total Sales
NT A
NT B
RT A
VM A
VM B



Analyse, Connect, and Build Stages:

The next three stages, analyse, connect, and build, may be accomplished in any orderor in parallel. However, they cannot be accomplished without a model.



- You cannot analyse data that is not defined in the ThingWorx model.
- Devices require a modelled thing to connect to, such as a Database Thing or a Device Thing.
- The build stage creates a user interface informed by data in the model. This user interface is usually, but not always, a Web-based user interface. It may be an augmented reality UI based on Vuforia, or it may provide notifications using email or SMS.

Data Tables:

A data table is similar to a standard relational database table, but differs greatly in terms of performance. Generally, data tables should be used if you have less than 100,000 rows of data. For larger datasets, a relational database should be used and then connected via a Database Thing template. A Data Shape defines the columns or fields of the data table. See Model and Data Best Practices for additional information on data tables.

Possible use cases for data tables are maintenance work orders for a site or crew, or production orders for a manufacturing line. Storing this data in data tables makes it easy to create a custom mashup for the consumers of the data.

A data table has the following predefined fields:

- Timestamp: The time the entry was created. It is also possible to provide a timestamp when adding a data table entry.

1. Each data table Thing service has a predefined timestamp common property oftype DATETIME. If a data table is using a custom data shape that defines the timestamp field with a different type, an error will occur when executing a query service.
2. Tag: Each data table entry can be tagged. Data tags help to search for and consume specific run time data.
3. Source: The source of the data table entry. This is usually the name of the Thingwriting to the data table or an identifier of an external system.
4. Source Type: The entity type of the source (such as Thing or user).
5. Location: The location of the data table entry's source.

Data table Templates:

Name	Description
Content Crawler	A content crawler Thing is used to call a service on another entity that returns an info table of data that is then stored in the data table of the content crawler Thing. See Content Crawler for more information .data table Templates
Data Table	Stores non time-series data. For more information, reference:

	• Model and Data Best Practices • Data Table Services • Data Table Best Practices • Sizing Limits of Data Tables data table Templates
Remote Data Table	Creates a local proxy object to a data table Thing that is running and persisting data on another ThingWorx server. For moreinformation, see Remote Things. data table Templates

When you create the Data Shape and define the primary key, the system will automatically create a table index for the Sales Order ID. But, in reality you probably query the table by other columns than the primary key.

The compound index has two field names separated by a semicolon. This is a requiredformat, and should not contain spaces.

For example, two common queries could be:

1. Get table data where Customer Name = 'Some Customer Name'
2. Get table data where SalesRep = 'rep name' AND CustomerRegion = 'Northeast Region'

For this example, you would create two indexes, one for each common query. The indexname is up to you - it is a semantic name and not used in the query execution. So the indexes might look like the following:

indexes might look like the following: Index Name Index Field Names CustomerCustomer Name RepRegion SalesRep; CustomerRegion

Configuration of Persistence Provider Custom Settings

If you are Using DataStax Enterprise (DSE) as the persistence provider, you canconfigure the following:

Name	Default Value
data table Bucket Count	3

Analytics:

As part of the ThingWorx IoT technology platform, ThingWorx Analytics provides toolsto embed advanced analytics insights into your smart, connected solutions and to applysupervised machine learning to your data. For information about specific ThingWorx Analytics functionality, see the related links.

- Analytics Builder – Interact with your ThingWorx Analytics data via an intuitive user interface. Analytics Builder extends ThingWorx functionality with access to analytic capabilities, including predictive model building, predictive scoring, prescriptive scoring, and data analysis. The interface also supports loading dataand metadata, creating datasets, and filtering data.

- Analytics Manager – Deploy and execute computational models from external applications based on events and data from ThingWorx-connected devices. Analytics Manager extends ThingWorx functionality so that you can manage analysis providers, analysis models, analysis jobs, and analysis events.

- Descriptive Analytics– Use the on-demand services of the Statistical Calculation and Statistical Monitoring micro servers to perform common statistical calculations on raw data or monitor statistical trends in streaming data. Outputfrom these services can be added to an IoT solution to help visualize the data, create alerts based on logged property values, or generate transformed data for machine learning processes.

- Property Transform – Use the computational services of this microservers to derive value from streaming data entering ThingWorx. These services can be used to automate simple data transformations, prepare incoming data for use inpredictive analytics, and to generate alerts when a streaming signal behaves in aspecified way.
- Anomaly Detection – Observe the data signal from a device, learn what the expected data stream looks like, and monitor for data sequences that are unexpected and could indicate the device is in an anomalous state. Configure analert to send notifications when an anomaly occurs.

The Deploy Stage:

- The deploy stage covers tasks involved when your IoT application is built and needs to be placed into production. Topics, such as access control testing, scalability testing, useracceptance testing, selecting the correct hardware, scaling the production system appropriately, setting up redundancy, and high availability functionality, securing the system with encryption, setting up a backup scheme, and migration are all deploy stagetopics.

Thingworx for IOT

Course Objectives:

- Fundamentals of IoT Development with ThingWorx
- Review the ThingWorx development process
- Identify the different components of ThingWorx Composer required for data model development and user interface development
- Discuss the Thing model in ThingWorx
- Understand the function and structure of the ThingWorx REST API
- Know about the different ThingWorx Connectivity technologies
- Apply ThingWorx development process in an IoT Application Development exercise

Introduction of Thingworx Platform

This project will introduce you to the principles of ThingWorx Foundation by creating a working web application.

Following the steps in this guide, you will create the building blocks of your first application for the Internet of Things (IoT). You will use ThingWorx Composer to create Thing Templates, which are then used to create Things that model the application domain. A simulator is imported to generate time-series data that is saved to a Value Stream.

After modelling the application in ThingWorx Composer, you'll use Mashup Builder to create the web application Graphical User Interface (GUI). No coding is required in modelling the application, or in composing the web GUI that displays dynamically generated data.

Industrial companies face pressing challenges that require IIoT solutions. To address a wide range of manufacturing, service, and engineering use cases, PTC has spent years innovating the ThingWorx IIoT platform.

From remote monitoring and service to workforce efficiency and asset optimization, ThingWorx solves common challenges across different industries.

Because building **Industrial IoT solutions** is often cited as a pain point, ThingWorx is designed to reduce these barriers. Cruise from pilots to enterprise-scale solutions, using pre-built applications and developer tools.

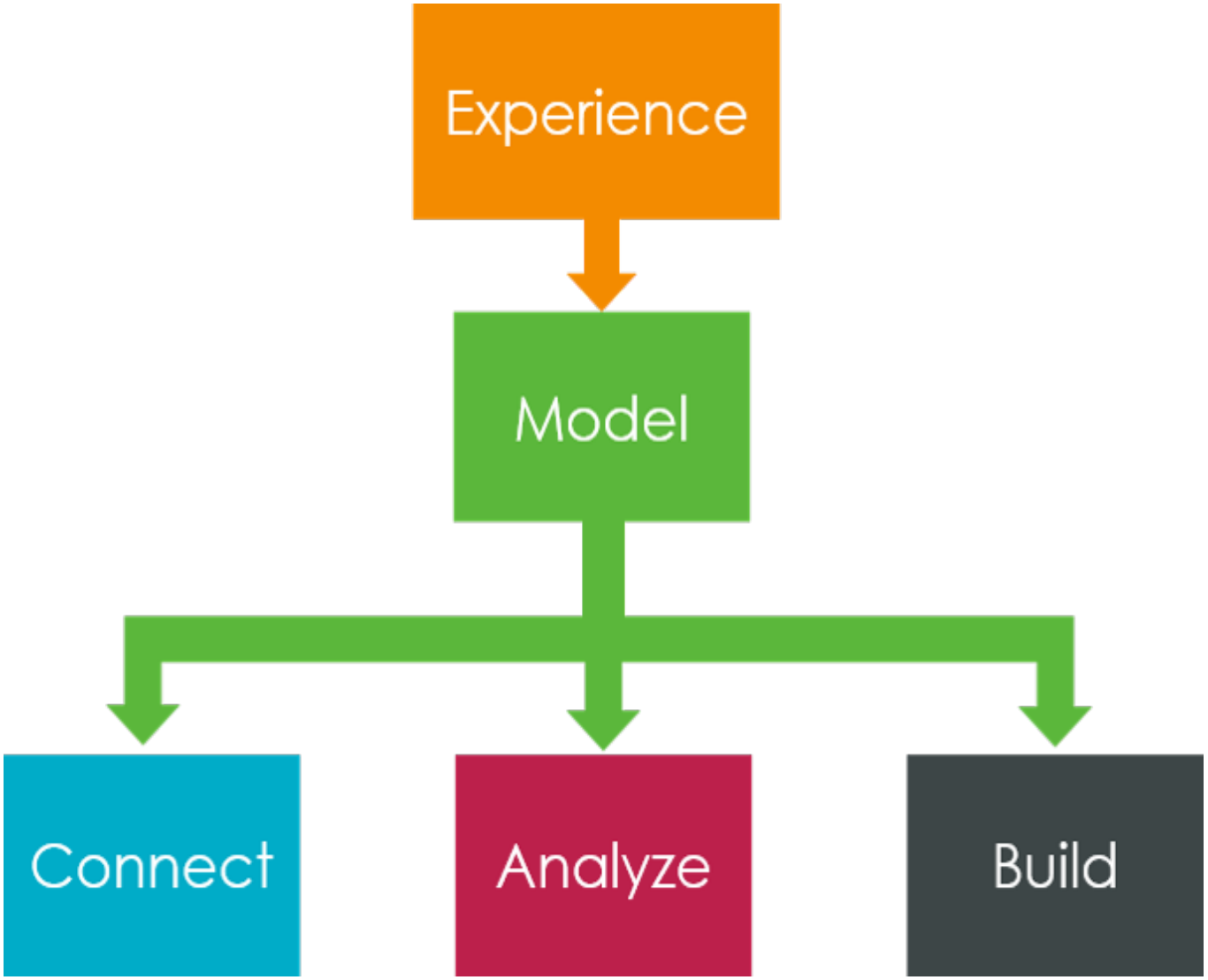
If you’re considering IIoT platforms, you should be evaluating their capabilities. Unlike other industrial IoT software, ThingWorx offers a complete IIoT platform. Industry leaders turn to ThingWorx for end-to-end capabilities—enabling them to address every facet of their digital transformation journey.

ThingWorx Development Process

ThingWorx offers a complete IIoT platform. Industry leaders turn to ThingWorx for end-to-end capabilities—enabling them to address every facet of their digital transformation journey.

Platform

Everything you need to make the Internet of Things work for business



1. CONNECT

Connect Your Data to ThingWorx

In the world of IoT application development, connectivity refers to the infrastructure and protocols which connect devices to the cloud or network.
Edge devices handle the interface between the physical world and the cloud.
ThingWorx
provides you with several different tools for connecting to the ThingWorx platform.
Your decision on which connectivity method to pick will be dependent on your individual use case.
Extend Standardized Industrial Connectivity across disparate devices and applications to enable Access to multiple data sources.

Experience Complete IoT Connectivity

Establish a standardized connectivity layer across your devices, applications, and enterprise systems—centralizing a pipeline of valuable industrial data.

The IIoT begins with industrial connectivity:

Connectivity is the first step to unlocking digital transformation—you can't implement IIoT solutions if you can't connect to your assets. But in complex manufacturing and service environments, connectivity can seem like an insurmountable challenge.

The ThingWorx IIoT solutions platform solves this challenge with powerful connectivity capabilities. ThingWorx makes it easy to establish standardized connectivity, so you can create a secure, single source for accessing industrial data across your IT and OT systems.

IoT connectivity isn't one size fits all

The diversity of industrial assets, products, and systems can make standardizing connectivity a serious challenge. Factory floors can contain countless protocols and PLCs; many vital pieces of workhorse equipment predate connectivity altogether. Introducing connected service can be similarly challenging for product makers with generations of customer equipment already in the field. If these challenges weren't enough, digital transformation requires integrating OT data with IT systems outside the shop floor or service truck.

Because there is no single solution for standardizing connectivity, ThingWorx offers a suite of integrated tools—each designed to address unique industrial IoT connectivity requirements

Connectivity in IoT is challenging—and critical to success

Connectivity plays a unique role in digital transformation. Explore further reading to understand why it's so important, and how you can simplify your connectivity challenges.

- REST API
- Edge SDKs
- Edge Microservers
- Kepware Server
- Device Cloud Connectors
- Protocol Adapters

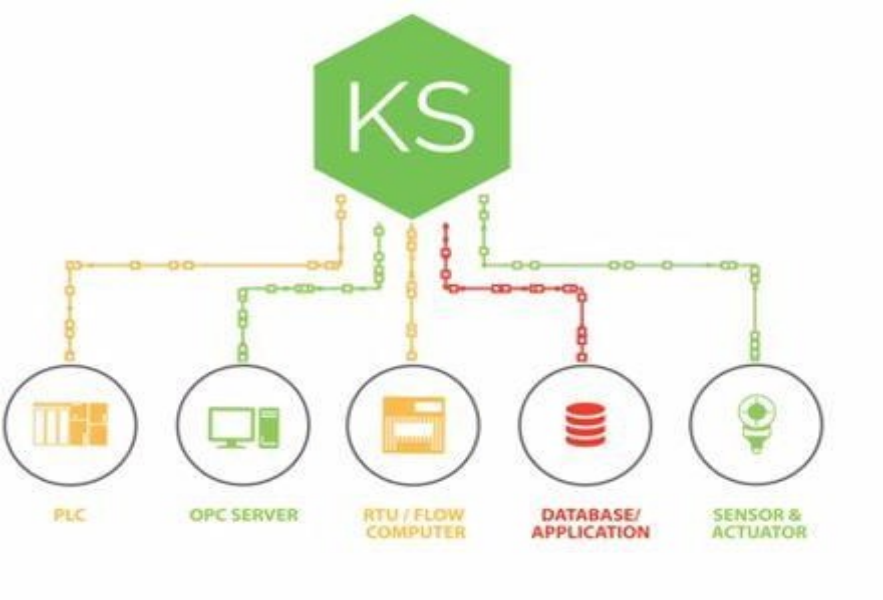
1. ThingWorx Kepware Server

ThingWorx Kepware Server unlocks IoT scalability for all of your sensors and equipment—including legacy assets—so you can connect with confidence.

Industry 4.0's benefits make it a critical pursuit for manufacturers who want to remain competitive. But for many companies, the prospect of connecting to their myriad of factory assets—particularly trusted older equipment—is simply too daunting. Industrial connectivity helps manufacturers break widely-implemented operational

technology systems like MES, SCADA, PLCs and CNCs out of niche-protocol silos. Doing so lets manufacturers get more out of valuable machine data, making it actionable, preventing maintenance and downtime, and ensuring equipment compliance.

Designed for IoT scalability, ThingWorx Kepware Server provides standardized, industrial connectivity that is cost-effective, easy to implement and supports a wide depth and breadth of disparate assets. Remove obstacles to connectivity so you can undertake meaningful Industry 4.0 projects that will transform your operations.



THINGWORX KEPWARE SERVER: PURPOSE-BUILT CAPABILITIES

ThingWorx covers industrial connectivity for your IIoT or Industry 4.0 project. ThingWorx Kepware Server provides the reliability, security, and scalability to support and accelerate your projects across the entire enterprise.

Reliability

The ThingWorx Kepware Server code base is trusted on more than 75,000 sites globally and was built by industrial automation connectivity experts. It seamlessly integrates with **ThingWorx** and other analytics platforms.

Security

ThingWorx Kepware Server uses the Always On protocol to encrypt your data and reduce IIoT attack surfaces.

Scalability

ThingWorx Kepware Server connects to almost any asset on the plant floor—new or old—so you can easily scale your IIoT projects within your factory and across multiple sites.

ThingWorx Kepware Edge allows the most valuable features of KEPServerEX to be deployed in Linux-based environments, enabling connectivity directly at the site of the machine, device, or sensor. The product utilizes secure, efficient, and IoT-ready protocols such as OPC UA or MQTT to connect to local, remote, and cloud-based applications or platforms. ThingWorx Kepware Edge offers flexible deployment of valuable Kepware functionality for improved connectivity to geographically distributed devices and machines.

ThingWorx Kepware Edge features native connectivity to **ThingWorx, the leading Industrial IoT platform**. By connecting to ThingWorx, users can quickly and programmatically add rich context to their industrial data and transform it into useful insights.

2. MODEL

Rapid, Model-based Application Development

Build your industrial IoT application using ThingWorx's drag-and-drop GUI development environment, model-based development platform. Using the ThingModel to describe assets, processes, and organizational elements and how they relate to each other. Define the functional behavior, add business logic, and extend your application with pre-built plugins. With a properly-constructed framework, your application will be scalable, flexible and more secure.

- Build the Data Model
- Leverage the Data Model
- Extend the Platform Capabilities

Build the Data Model

Define the properties, services, and events of Things you want to expose to your application developers. The ThingWorx Data Model is a logical representation of the physical devices, systems, and people that interact with your application.

Data Model Introduction

Learn how the ThingWorx Data Model makes your IoT application flexible and scalable.

consider data interactions based on user needs and requirements, as well as application modularity, reusability, and future updates.

LEARN HOW TO

- Utilize the function of a Data Model for your IoT application
- Conceptualize Data Model components, including Thing, Thing Shape, and Thing Template
- Design Foundation interaction with connected device.

Benefits

A **Data Model** creates a uniform representation of all items that interact with one another.

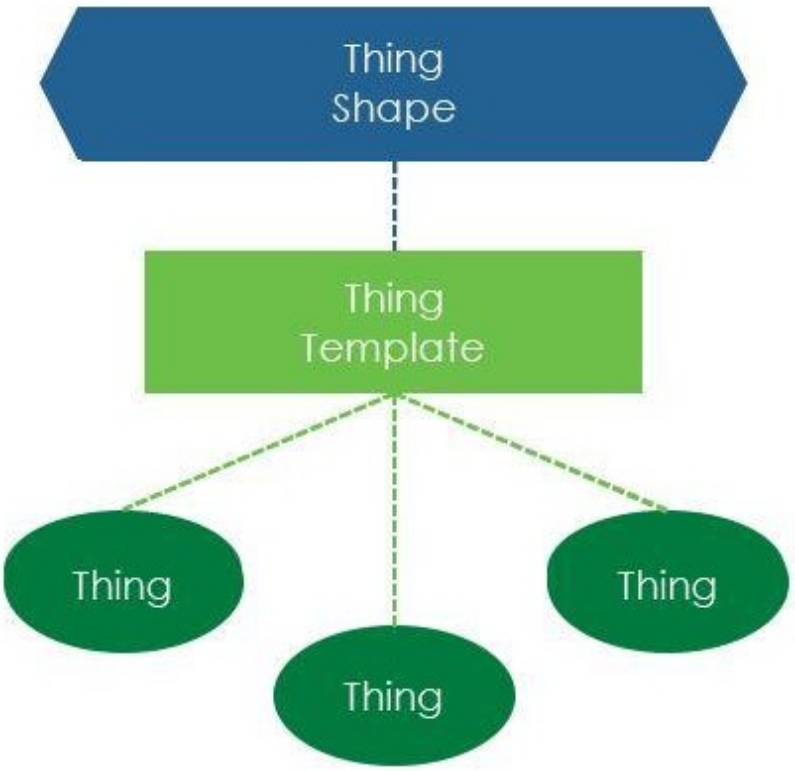
There are multiple benefits to such an approach, and the ability to break up items and reuse components is considered a best practice.

ThingWorx has adopted this model at a high level to represent individual components of an IoT solution

Entities:

Building an IoT solution in **Foundation** begins with defining your **Data Model**, the collection of **Entities** that represent your connected devices, business processes, and your application.

Entities are the highest-level objects created and maintained in Foundation, as explained below.

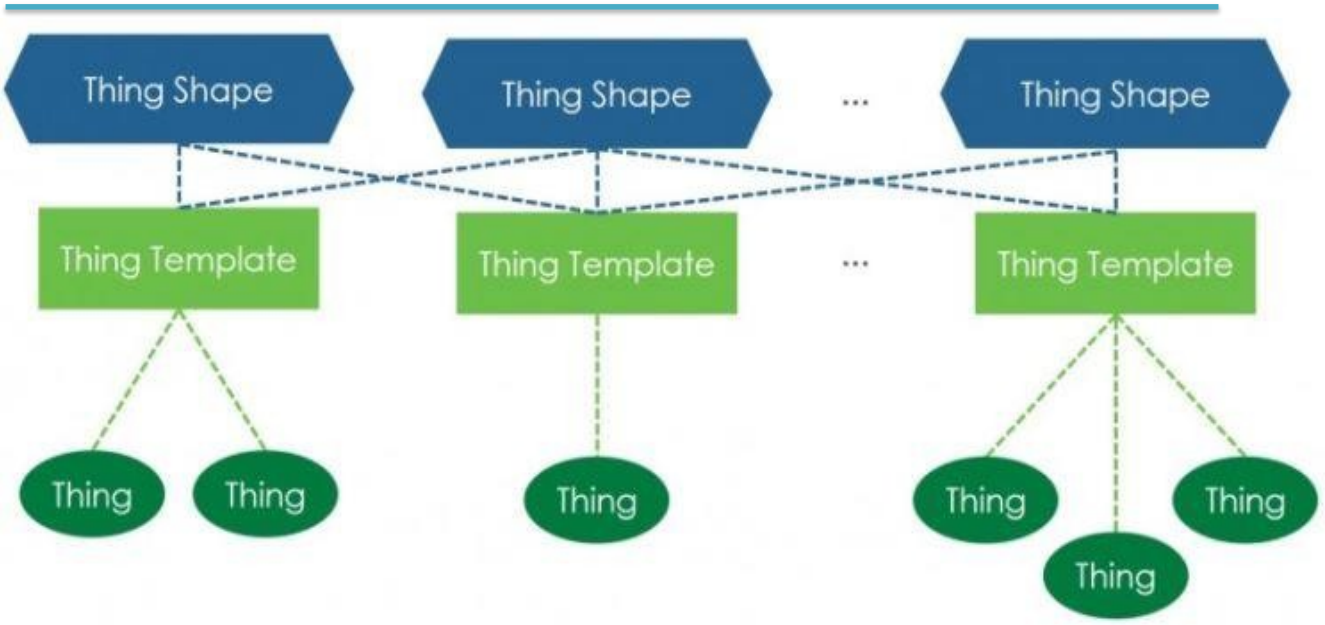


Thing Templates	Combine those components into fully-functional objects.
Thing	Unique representation of a set of identical components defined by the Thing Template.

Inheritance Model

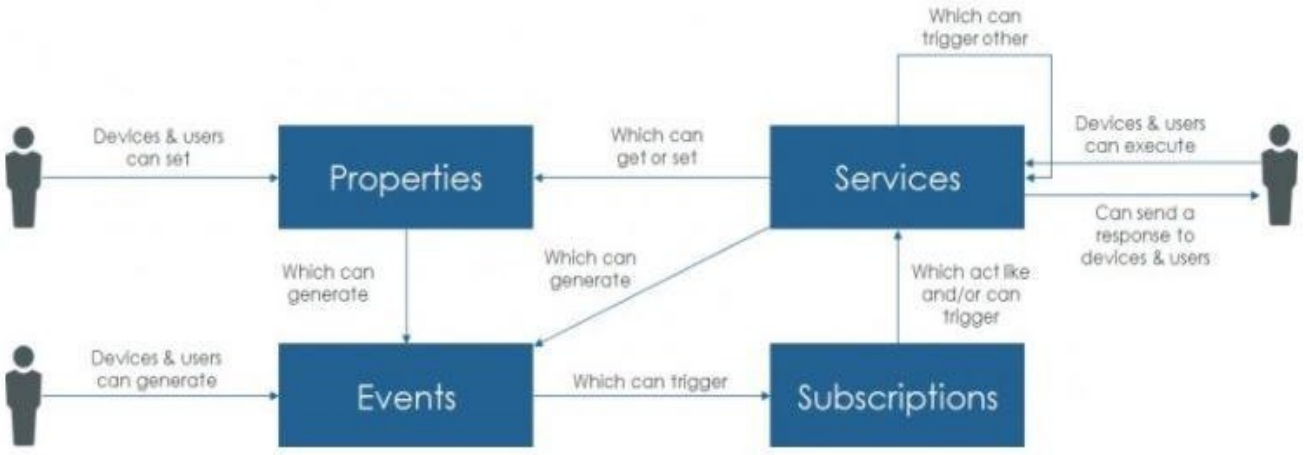
Defining **Things**, **Thing Templates**, and **Thing Shapes** in your **Data Model** allows your application to handle both simple and complex scenarios.

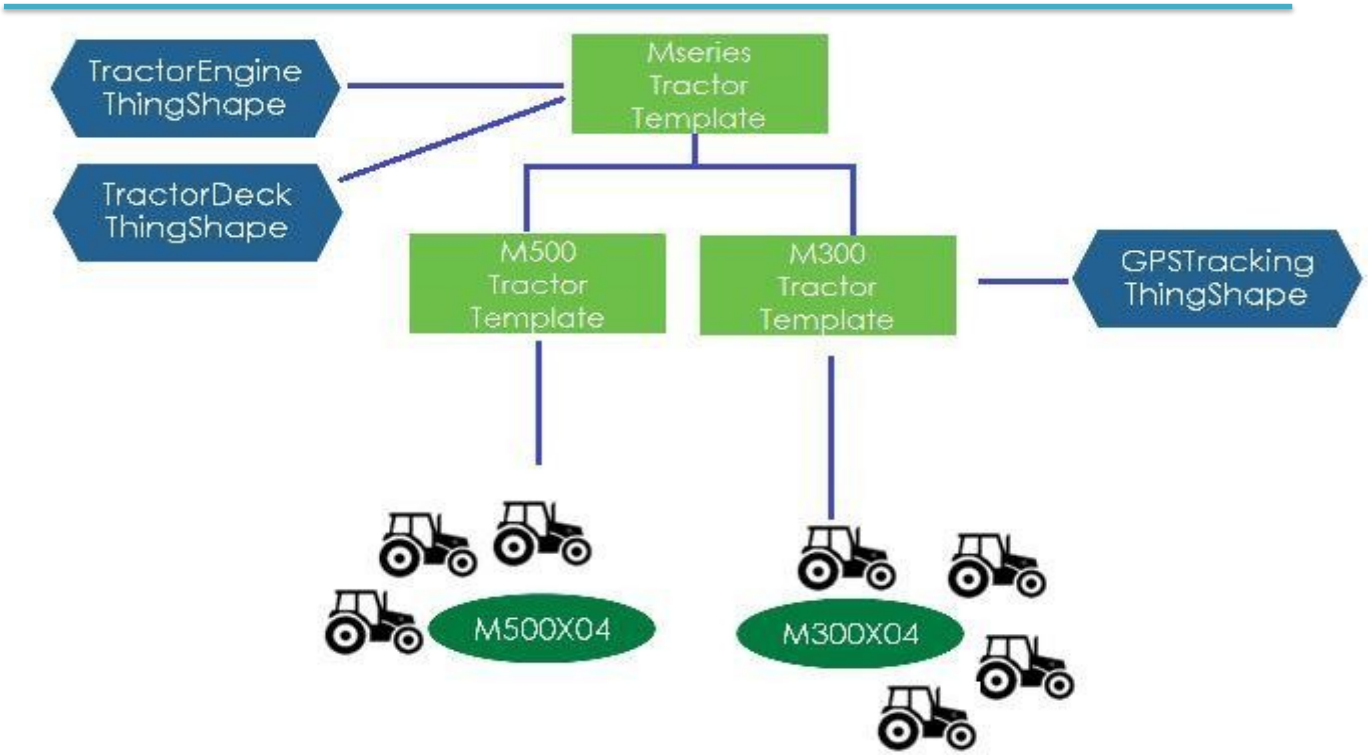
Entity	**Function**
Thing Shapes	Assemble individual components.



Scenario

The ThingWorx Data Model provides a way for you to describe your connected devices, and match the complexity of a real-world scenario. Things, Thing Templates, and Thing Shapes are building blocks that define your data model.





This diagram shows what a **specific Inheritance Model** might look like for a connected **Tractor**.

There is one master Template at the top. In this case, it's a collection of similar types of tractors.

The parent Template inherits a few Shapes - an Engine and a Deck that have been used in previous designs. Importing them as Shapes allows us to reuse previous design work and expedite the development process.

One of the child Templates incorporates another Shape, this time in the form of a GPS tracking device.

Then, at the bottom, there are the specific tractors with individual serial numbers that will report their connected data back to an IoT Application.

Implement Services, Events, and Subscriptions

Automate business processes with Services, Events and Subscriptions.
This will introduce Services, Events, and Subscriptions inside of the ThingWorx platform.
you will be able to expand your data model using Services, Events, and Subscriptions. We will teach you how to make a more robust and enjoyable experience for users simply by using the resources inside of the ThingWorx Composer.

YOU'LL LEARN HOW TO

- Create Events, Services and Subscriptions in Composer
- Utilize the ThingWorx Edge SDK platforms with Services, Subscriptions, and Events

- Functionality
- Create Events
- Services Interface
- Create Services
- Create Subscriptions

Create Events

Events are used to mark situations that can occur within an application. They can be used for scenarios ranging from a dangerous situation that is imminent and needs to be checked by personnel to just a system notification. Each Event is based on a DataShape that will hold the necessary information about the Event.

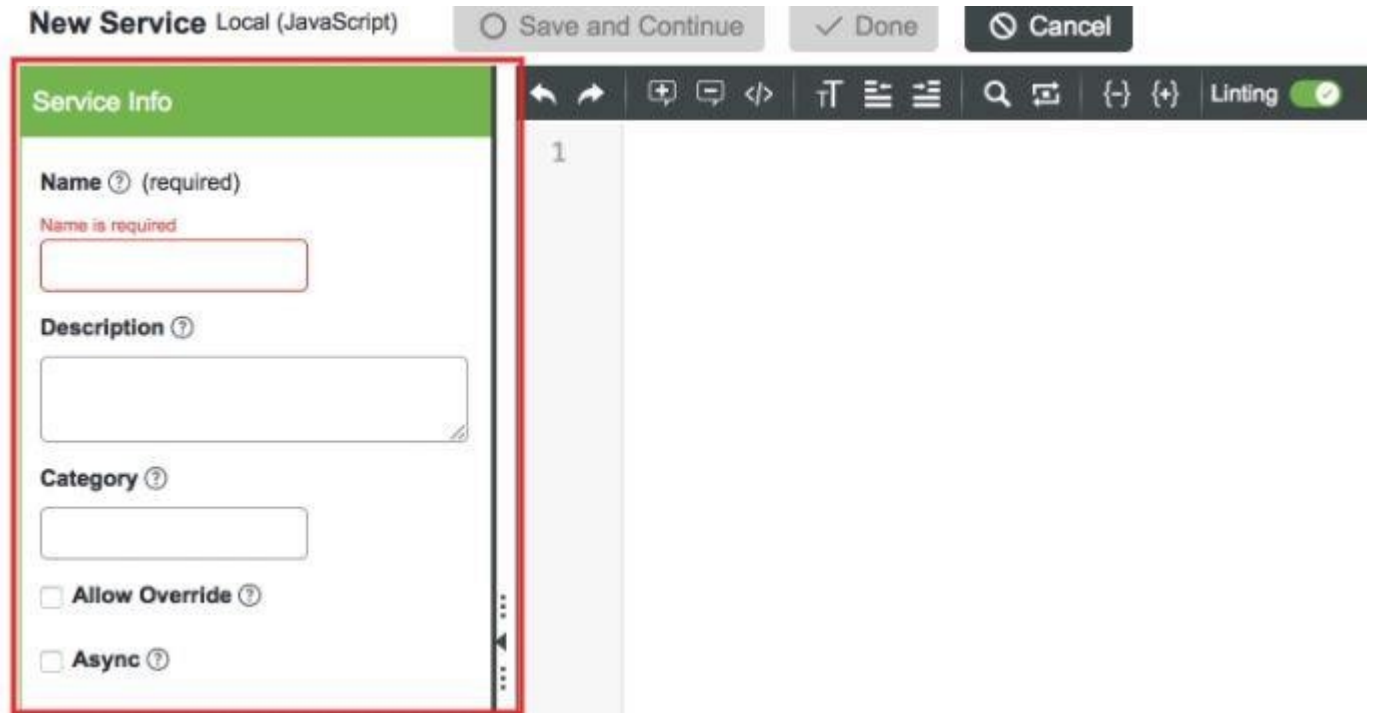
Service Interface

To create **Services** within ThingWorx, go to the Services tab of the Entity that will house the Service. Our **PTC Delivers Business Logic** Thing will contain several Services for us.

Within the Services tab under the Entity Information section of a Thing, you will see built in functionality for all the Thing Templates and Thing Shapes the Thing inherits from. You will also see the section that allows you to create new Services.

Service Info Tab:

This tab is for the general information of the Service. This is where you will add the Service name, description, category, whether the service is asynchronous, and whether the Service can be overridden.

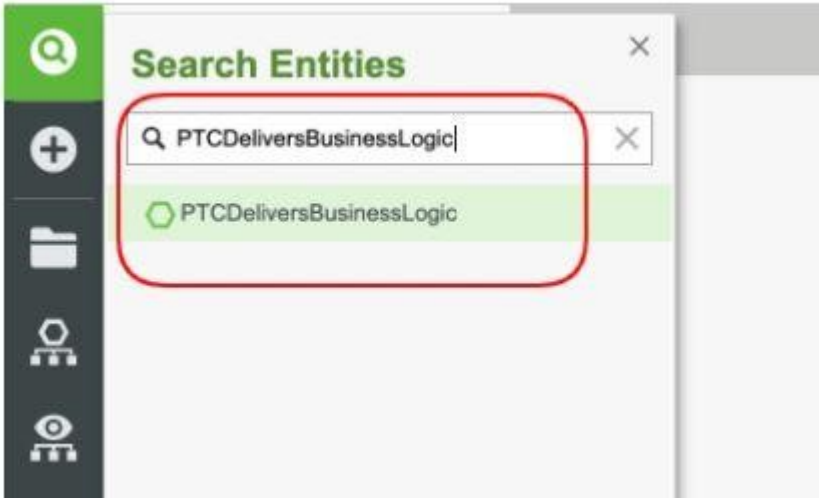


Create Service

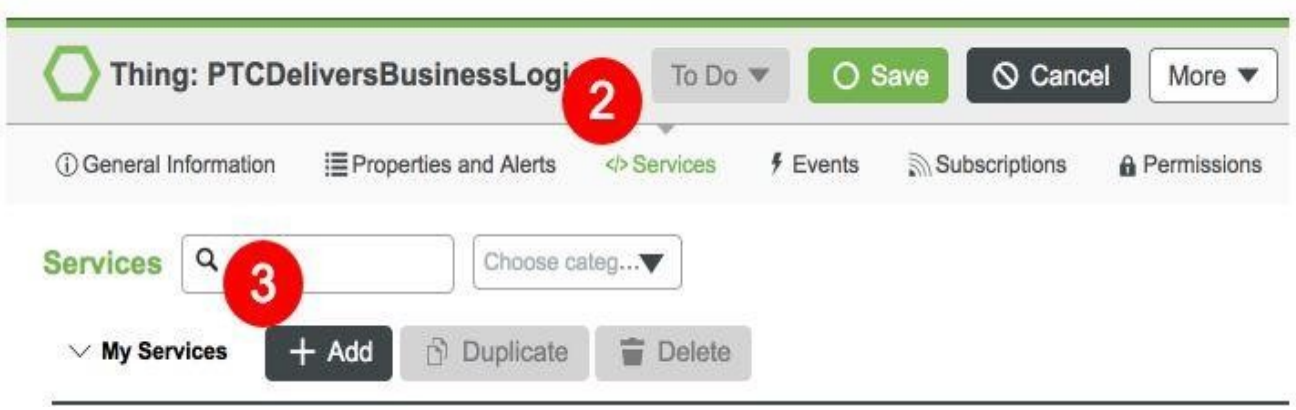
Below shows how we can create the **Get Customer Packages** Service for the **PTC Delivers Business Logic** Thing.

Create Service Definition

1. On the home page, filter and select the **PTC Delivers Business Logic** Thing



1. Switch to the **Services** tab.
2. Click **+ Add**.

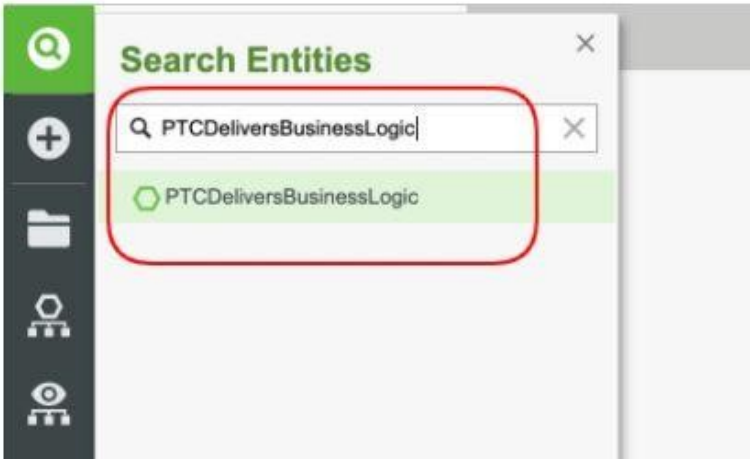


Create Subscriptions

Subscriptions are based on tracking the firing of an Event. When the Event is triggered or fired, all entities with a Subscription to the Event will perform the script as defined. The JavaScript interface and script tabs are the same as those utilized for the Services interface.

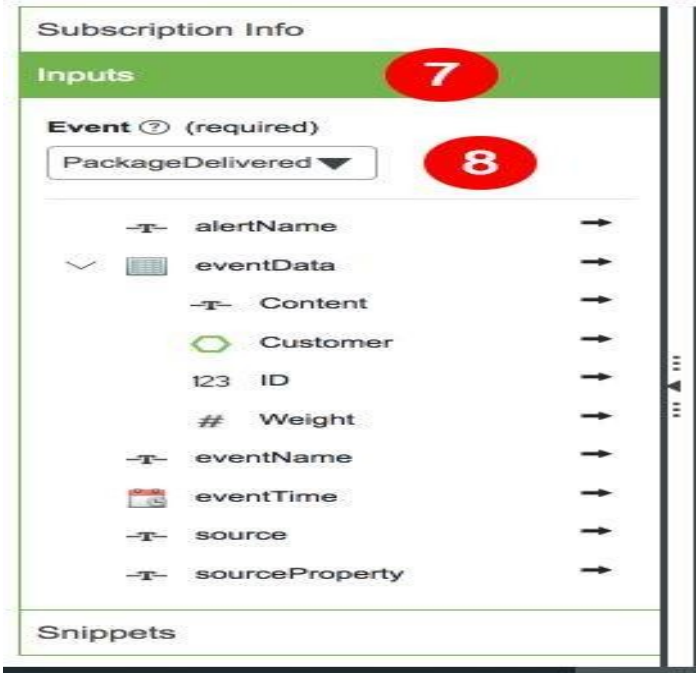
Subscriptions are a great resource for making updates in other Entities, Databases, or even just logging this information to your liking.

1. On the home page, filter and select the **PTC Delivers Business Logic** Thing.



Using a Stream to track events in the application allows for one source to watch for activity. The source for a Subscription can be other Entities if a single Entity is wanted. In order to capture all source data from the Package Stream, you will need to set it as the Stream for the Entity you would like to track.

Select the **Event** drop-down and pick the **Package Delivered** Event. This Package Delivered Event only exists in the completed download. If you are not using that download, create your own Event based on the **Package Data Shape**.



3.ANALYZE

Analyze and Visualize IoT Data

The AI and Machine Learning technologies used in ThingWorx Analytics automate much of the complex analytical processes involved in creating data-driven insights for your IIoT application. Simulate behavior of physical products in the digital world, use predictive analytic algorithms to find patterns in your business data and generate a prediction model, or build a real-time anomaly detection model by monitoring for data points that fall outside of an expected range.

1. Operationalize an Analytics Model

Use Analytics Manager and Thing Predictor to automatically perform analytical calculations.

This will introduce ThingWorx Analytics Manager.

you will learn how to deploy the model which you created in the earlier Builder guide.

how to utilize this deployed model to investigate whether live data indicates a potential engine failure?

LEARN HOW TO

- Deploy and execute computational models
- Define and trigger Analysis Events
- Map incoming data to the Model
- Map analytic outputs to applications

Analytics Architecture

You can leverage product-based analysis Models developed using PTC and third-party tools while building solutions on the ThingWorx platform.

- Use simulation as historical basis for predictive Models
- Create a virtual sensor from simulation
- Design-time versus operational-time intelligence

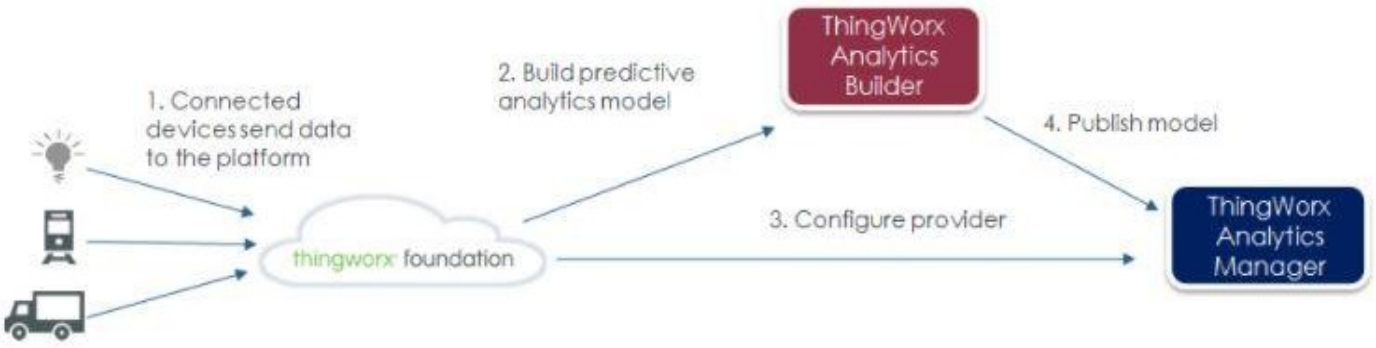
It is important to understand how Analytics Manager interacts with the ThingWorx platform.

Build Model

In an IoT implementation, multiple remote Edge devices feed information into the ThingWorx Foundation platform.

That information is stored, organized, and operated-upon in accordance with the application's Data Model.

Through Foundation, you will upload your dataset to **Analytics Builder**. Builder will then create an **Analytics Model**.

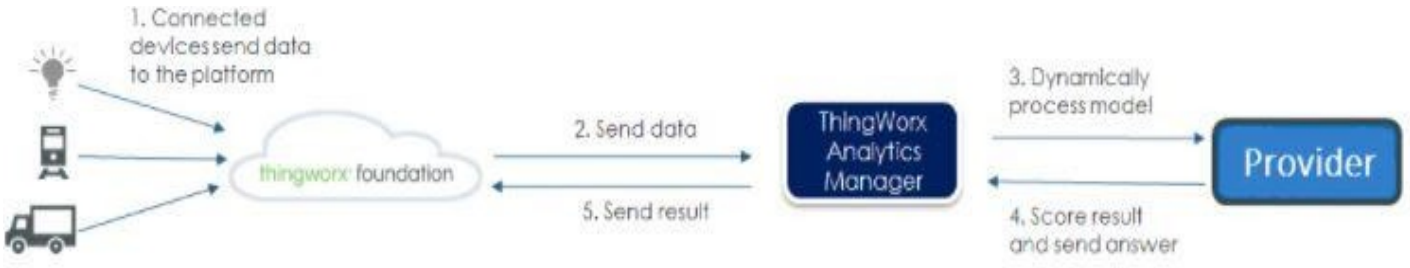


Operationalize Model

Analytics Manager tests new data through the use of a *Provider*, which applies the Model to the data to make a prediction.

The Provider generates a **predictive result**, which is passed back through Manager to ThingWorx Foundation.

Once Foundation has the result, you can perform a variety of actions based on the outcome. For instance, you could set up Events/Subscriptions to take immediate and automatic action, as well as alerting stakeholders of an important situation.



2. Build a Predictive Analytics Model

Generate predictions and gain insight into your data with machine learning.

This will introduce ThingWorx Analytics Builder. you will create an analytical model, and then refine it based on further information from the Analytics platform. how to determine whether or not a model is accurate and how you can optimize both your data inputs and the model itself.

LEARN HOW TO

- Load an IoT dataset
- Generate machine learning predictions
- Evaluate the analytics output to gain insight

Generate predictions and gain insight into your data with machine learning.

4. EXPERIENCE

Design Engaging Experiences

Use the industry's first purpose built IoT application development environment to design engaging experiences for web and mobile applications. Designed to reduce the time, cost, and risk required to build new innovative IoT applications, this layer has two distinct functions: build-time and run-time. Build-time encompasses the technology to create the things in your Industrial IoT solution while Run-time includes the operational permissions to execute and manage those things.

- Application Layout (UI)
- Charts & Graphs
- Reusable Components

1. Application Layout (UI)

Utilize the ThingWorx Mashup Builder tools to design and create engaging IoT applications.

Define Your UI Style

Convey information about IoT data effectively by customizing style definitions and implementing event-based logic

This project will help you identify how you would like to create an experience for Users.

Following the steps in this guide, you will use colour schemes to convey information quickly and effectively, for example to alert users of critical events. With ThingWorx Composer, you can implement **Styles and States** in your Mashups to *enhance your user experience*.

We will teach you how to create an affective IoT application experience that looks great and easy to navigate. How the UI is presented can influence users and their enjoyment of the application.

LEARN HOW TO

- Create a Style Definition
- Customize Style Definitions
- Create and implement State Definitions
- Implement event-based state changes

2. How to Display Data in Charts?

Graphically display multiple data points in a various chart.

- This will introduce various Chart Widgets.
- you will learn how to graphically represent multiple data points simultaneously.
- how to utilize the Pie, Label, Proportional, Bubble, Time Series, and Event Charts?

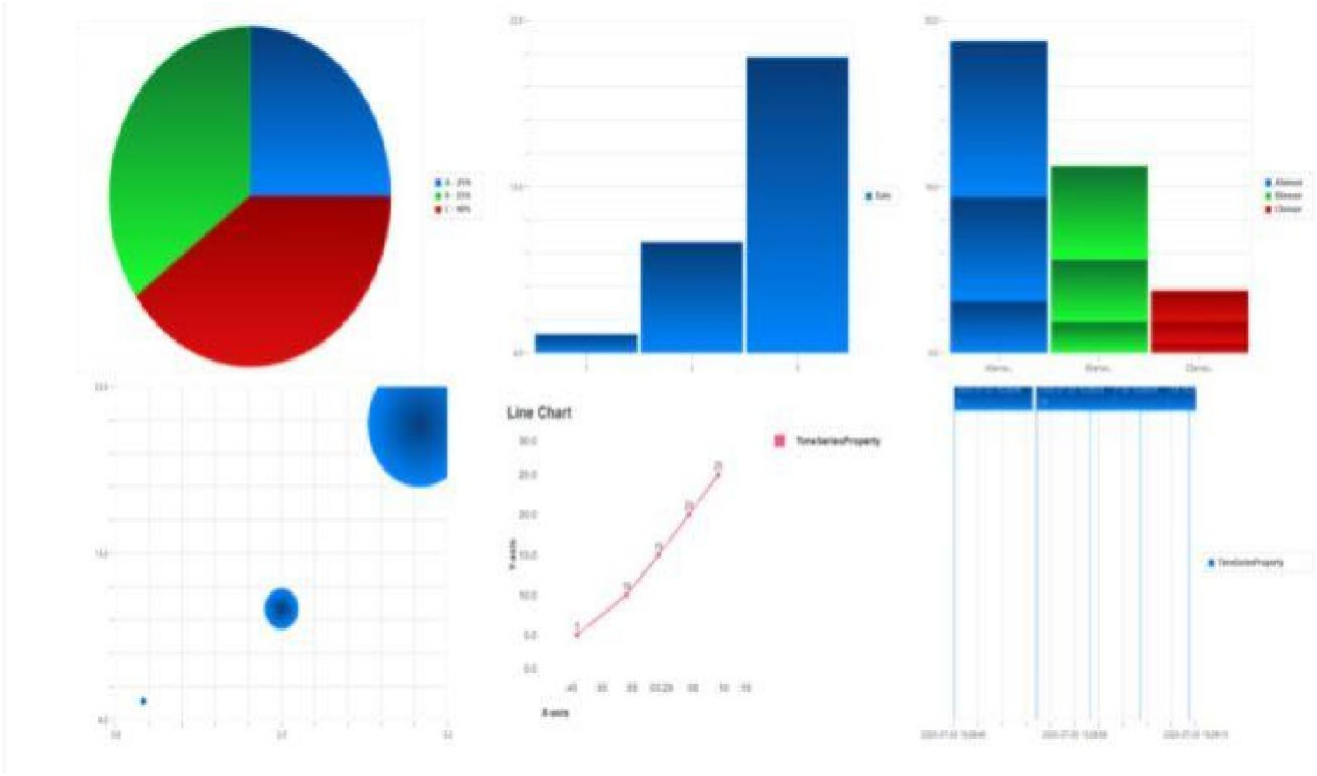
LEARN HOW TO

- Create a Data Shape to format an Info Table
- Create a Value Stream to record Property Value changes
- Create a Thing with an Info Table Property and a Time Series Property
- Create a Mashup with various Chart Widgets
- Link your Thing's Properties to the Chart Widgets

How to Display Data in Charts?
Outline

Step 1: Introduction

- Step 2: Create Data Shape
- Step 3: Create Value Stream
- Step 4: Create Thing
- Step 5: Create Mashup
- Step 6: Configure Charts
- Step 7: Next Steps



3. Reusable Components

Leverage the ThingWorx widget library to create a robust user experience and enhance your application capabilities.

Organize Your UI with the Collection Widget

Use the Collection Widget to organize visual elements of your application.

- Overview
- Step 1: Create a Data shape
- Step 2: Create a Thing
- Step 3: Create a Base Mashup
- **Step 4: Use Collection Widget**
- Step 5: Next Steps

Create a Data shape

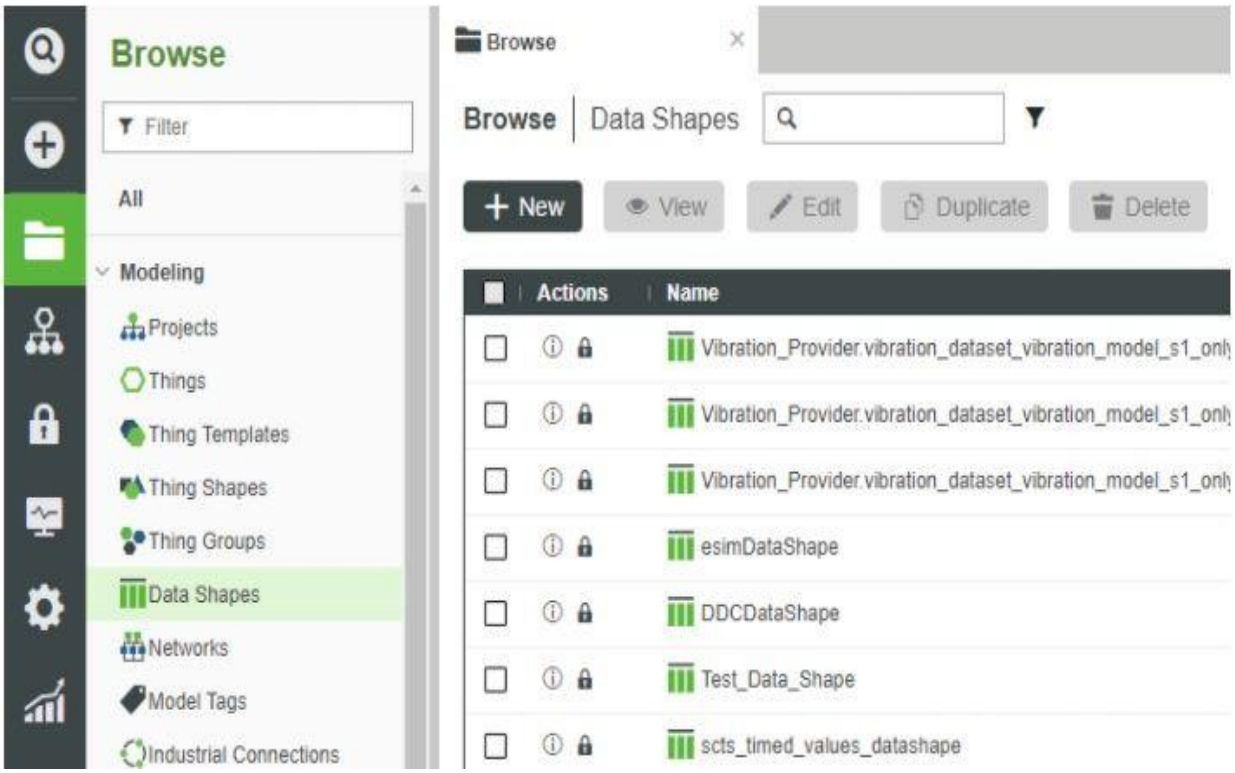
The **Collection Widget** uses a **Service** to dynamically define visual content.

The **data** must be in a **tabular format** (for example: Data Table, Info Table, or external Database-connection) in order for the Collection Widget to access it.

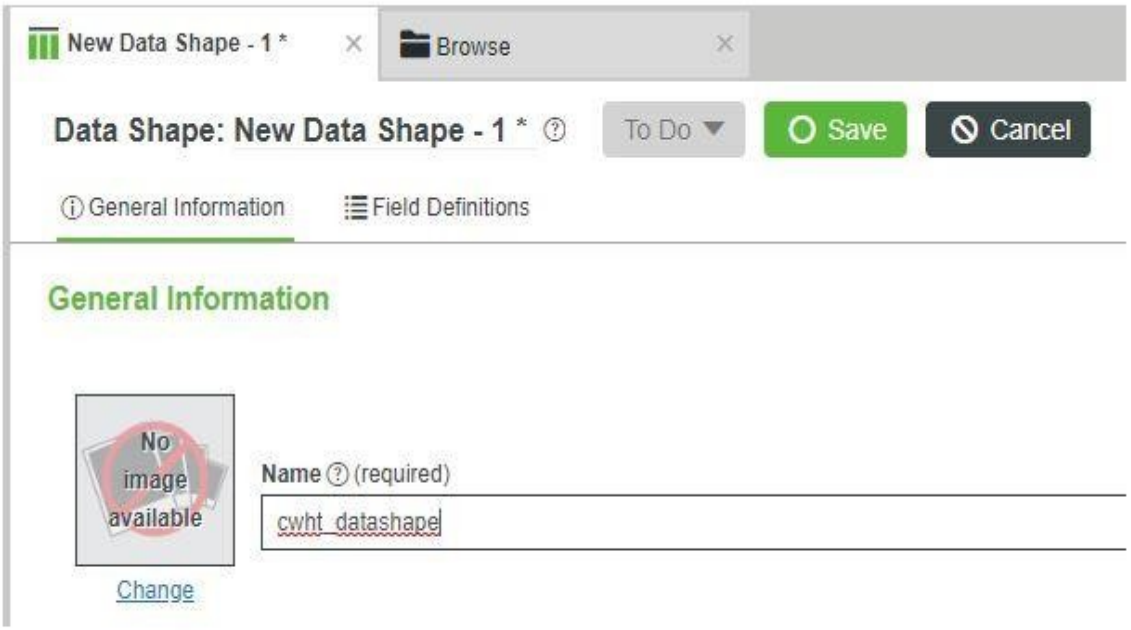
In this part of the lesson, we'll create a **Data Shape** to define the columns of the table.

In a subsequent step, we'll create an **Info Table Property** within a **Thing** in order to store the information.

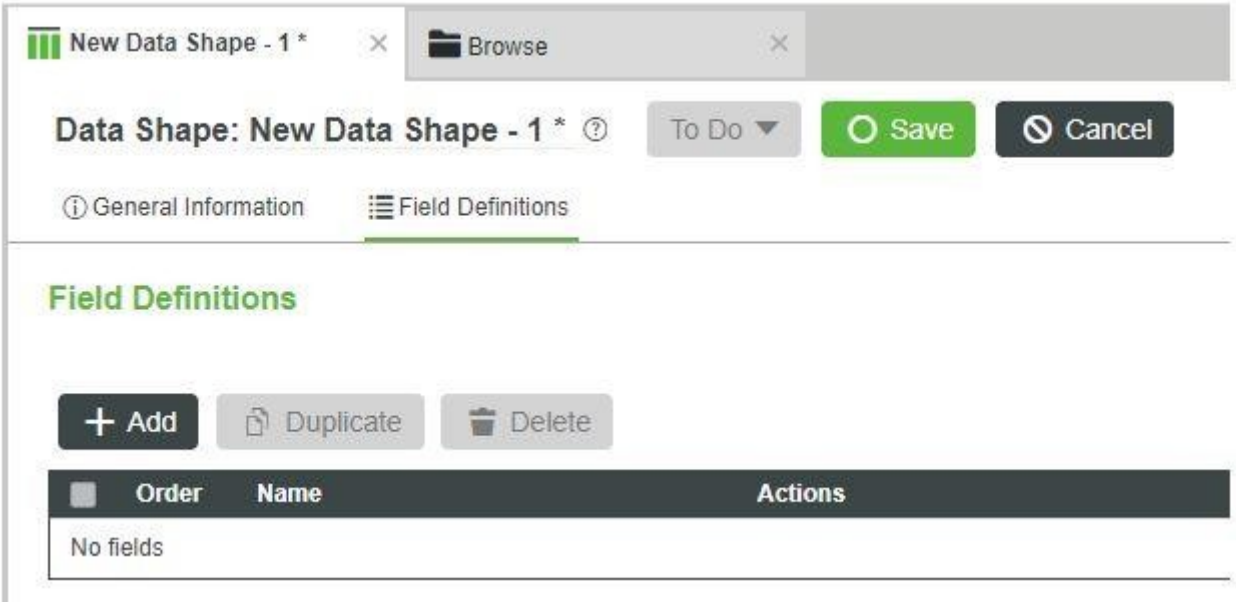
1. In the ThingWorx Composer **Browse** tab, click **Modeling > Data Shapes**, **+ New**.



2. In the **Name** field, enter `cwht_datashape`.



3. If **Project** is not already set, search for and select **PTCDefaultProject**.
4. At the top, click **Field Definitions**.



First Definition

1. Click **+ Add** to open the **New Field Definition** slide-out.
2. In the **Name** field, enter first number.
3. Change the **Base Type** to **NUMBER**.\

4.Check the **Is Primary Key** box.

- All Data shape must have a single Primary Key, and the first Field is as acceptable as any other for our purposes here.

. At the top-right, click the "Check with a +" button for **Done and Add**.

Second Definition

1. In the **Name** field, enter second number.
2. Change the **Base Type** to **NUMBER**.

: the top-right, click the "Check with a +" button for **Done and Add**.

Third Definition

1. In the **Name** field, enter third number.
2. Change the **Base Type** to **NUMBER**.

3. At the top-right, click the "Check" button for **Done**.

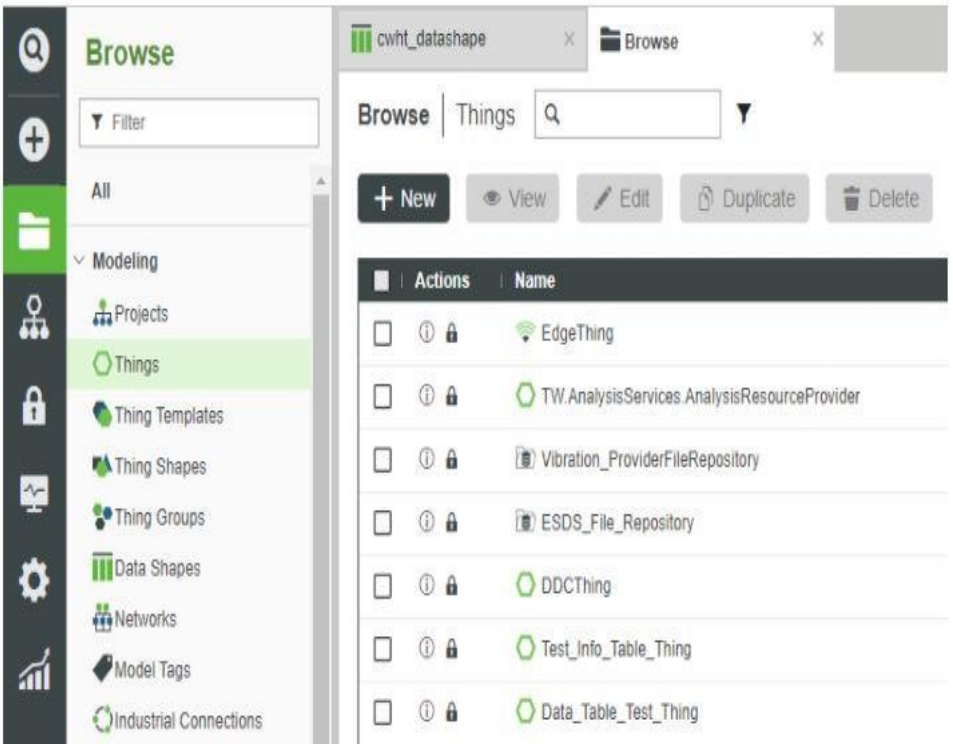
4. At the top, click **Save**.

Create a Thing

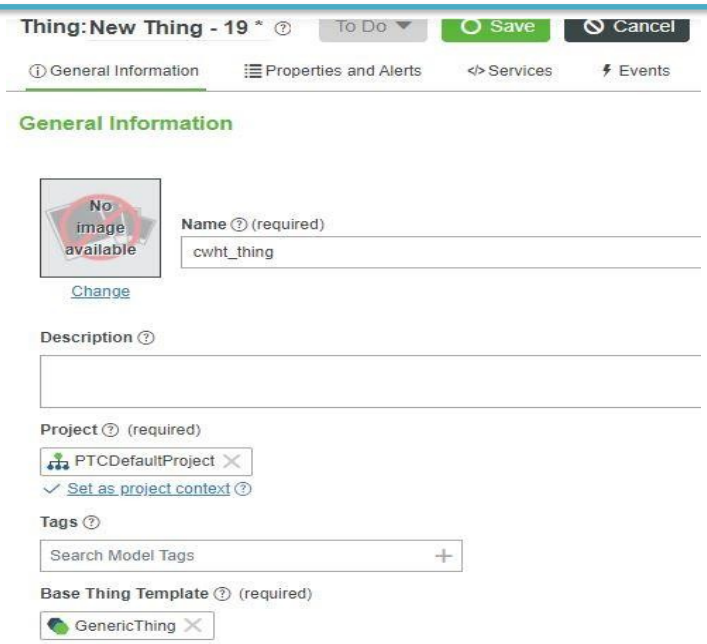
In the previous step, we created a **Data Shape** to define the **Info Table Property** columns.

Now, we will create a **Thing**, add an Info Table Property, format the Info Table Property with the Data Shape, and set some default values.

1. On the ThingWorx Composer **Browse** tab, click **Modeling > Things**, **+ New**.

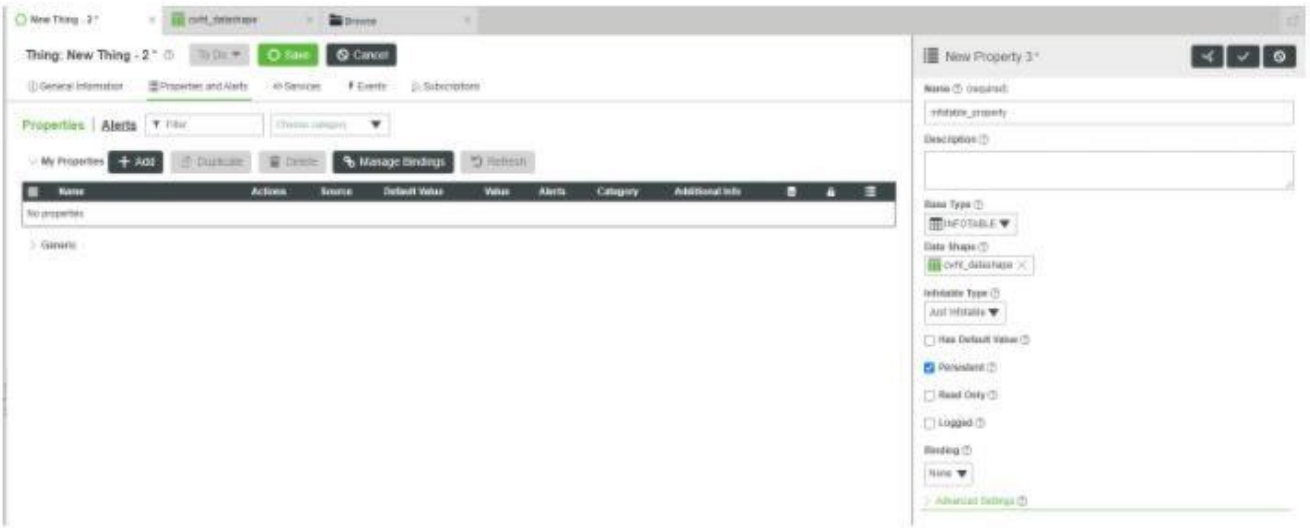


1. In the **Name** field, enter cwht_thing.
2. If **Project** is not already set, search for and select **PTC Default Project**.
3. Select **Generic Thing** in the **Base Thing Template** field.



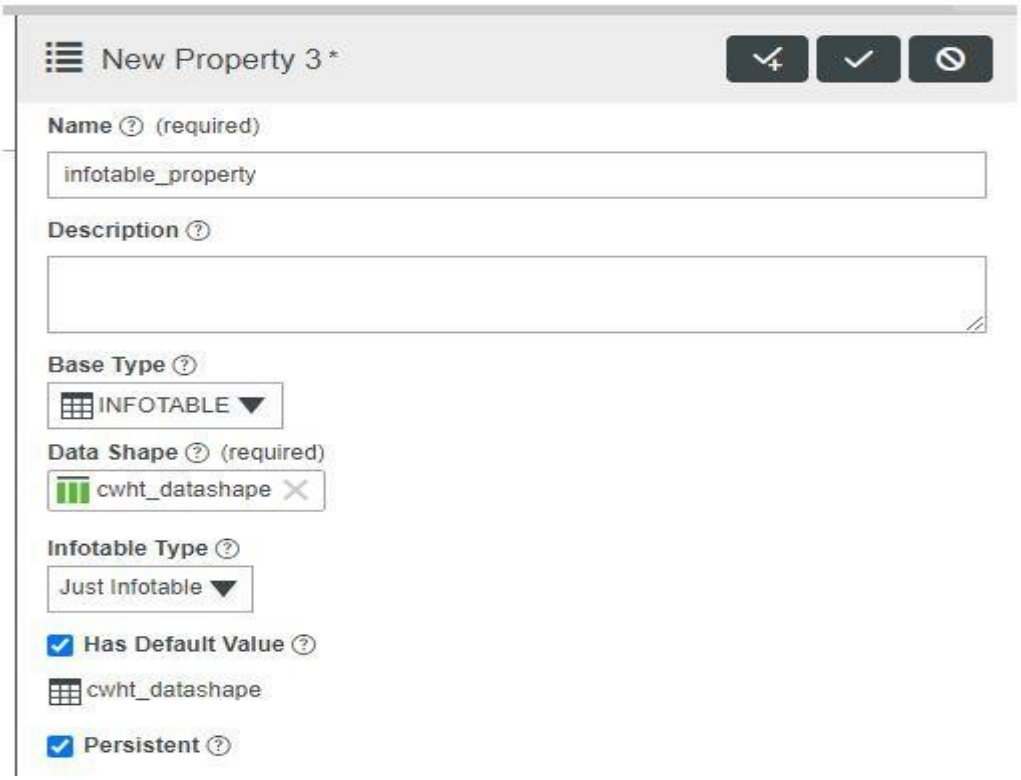
Add Info Table Property

1. At the top, click **Properties and Alerts**.
2. Click the **+ Add** button to open the **New Property** slide-out.
3. In the **Name** field, enter infotable_property.
4. Change the **Base Type** to **INFOTABLE**.
5. In the **Data Shape** field, select **cwht_datashape**.
6. Check the **Persistent** checkbox.

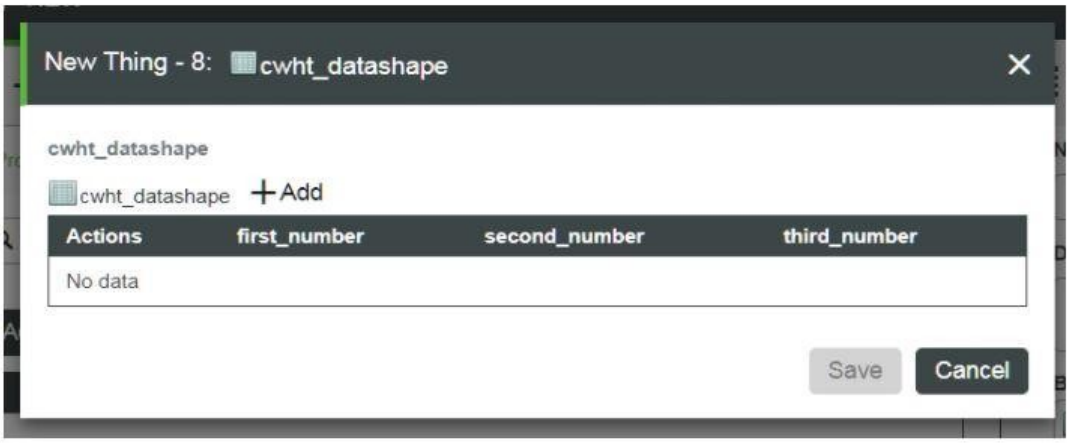


First Default

1. Check the **Has Default Value** checkbox.
    - A **cwht_datashape** icon will appear underneath.

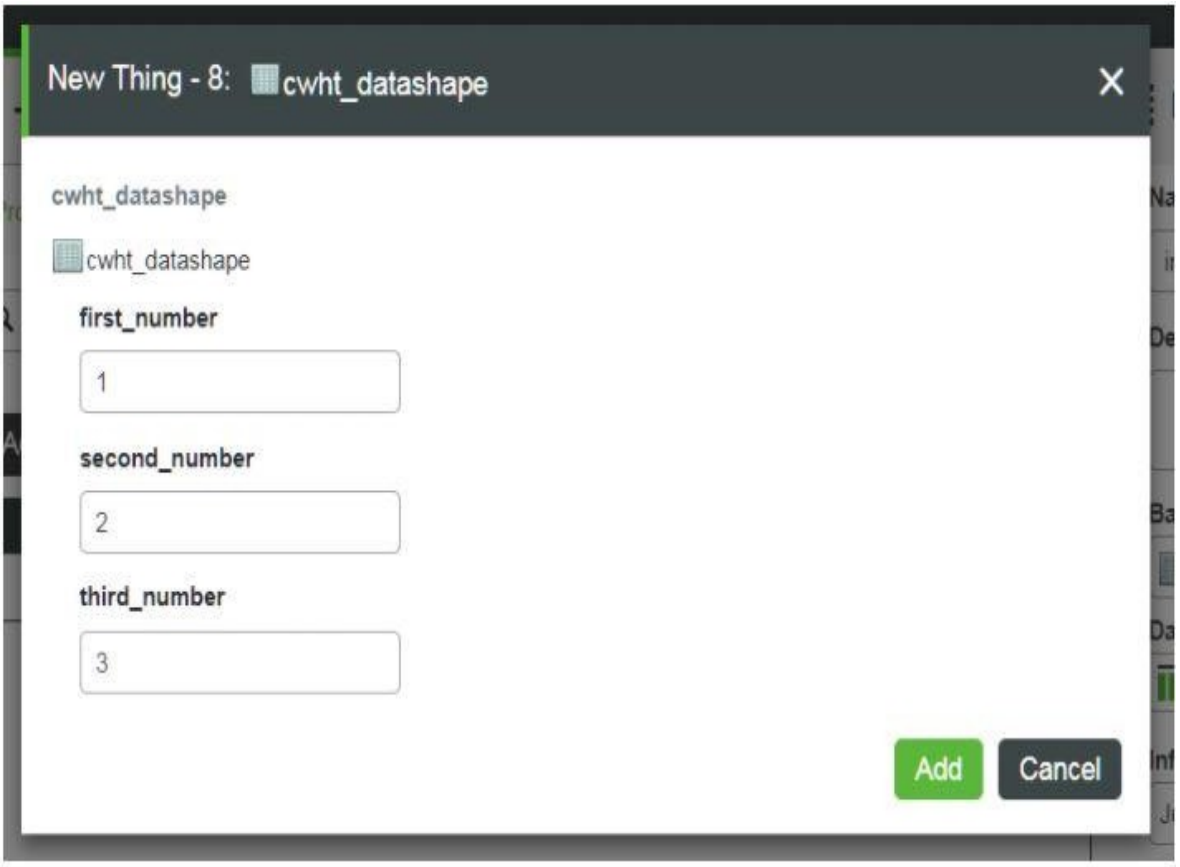


2. Under **Has Default Value**, click the **cwht_datashape** button to open the pop-up menu which sets the default values.

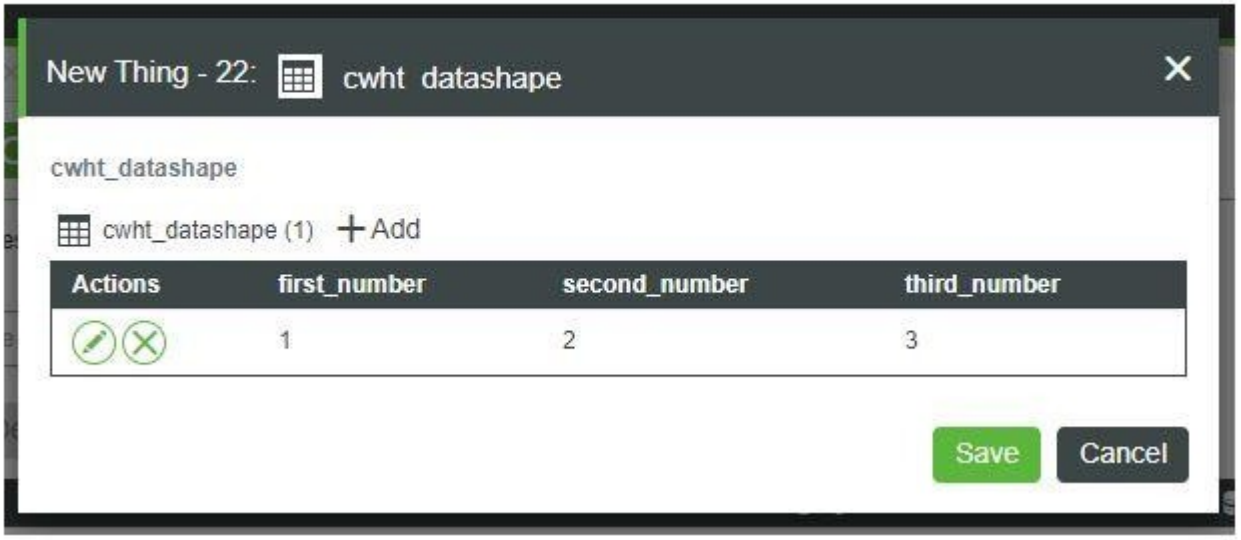


3. Click the **+ Add** icon.

4. Enter values in each number field, such as **1**, **2**, and **3**.

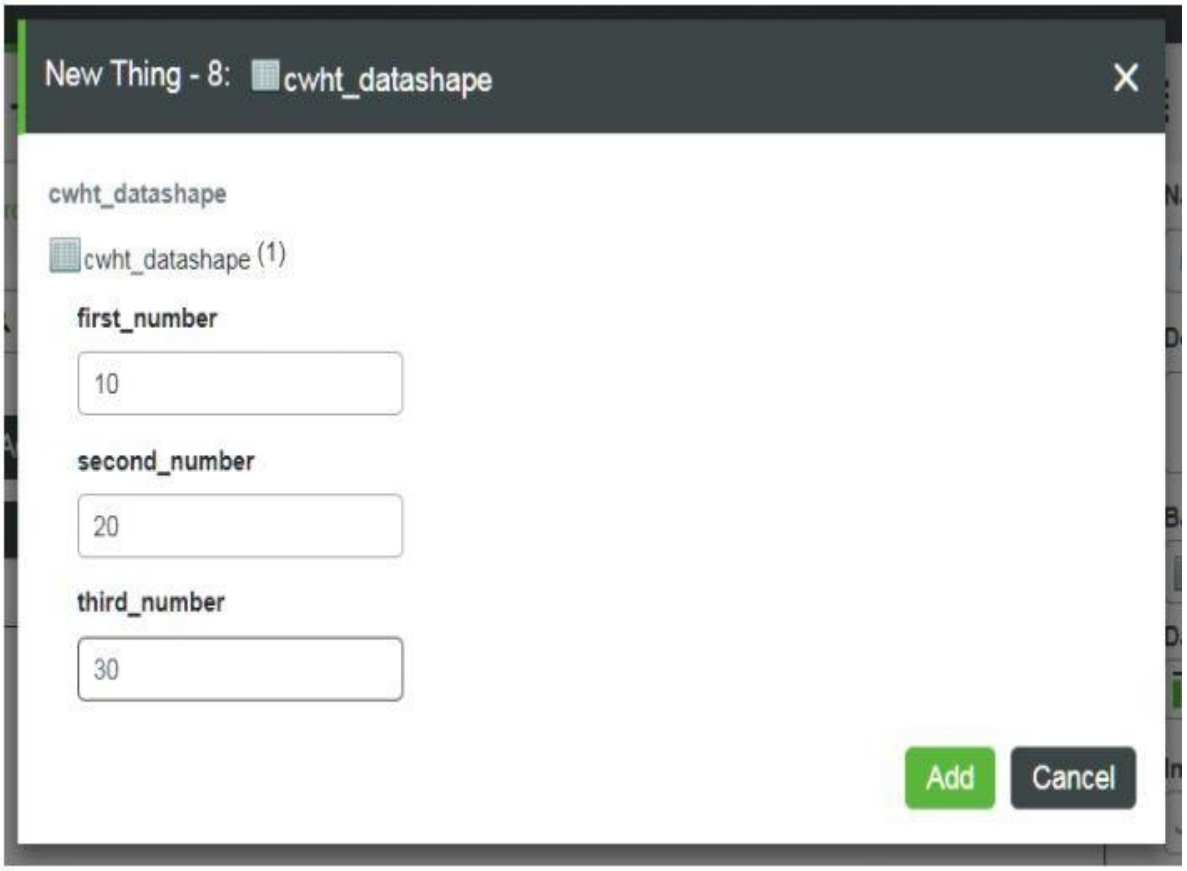


5. At the bottom-right, click the green **Add** button.

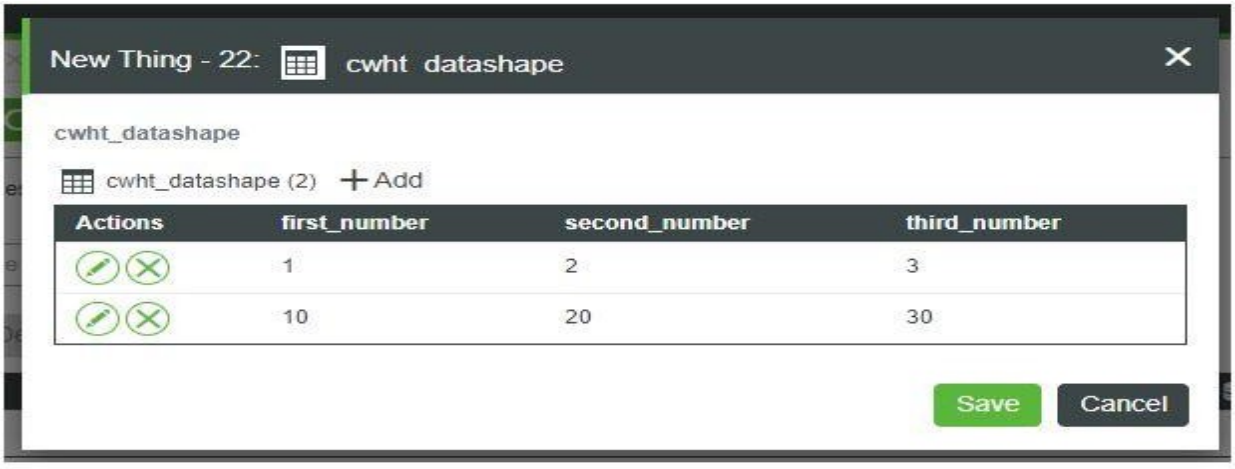


Second Default

1. Click the **+ Add** icon.
2. Enter values in each number field, such as **10**, **20**, and **30**.

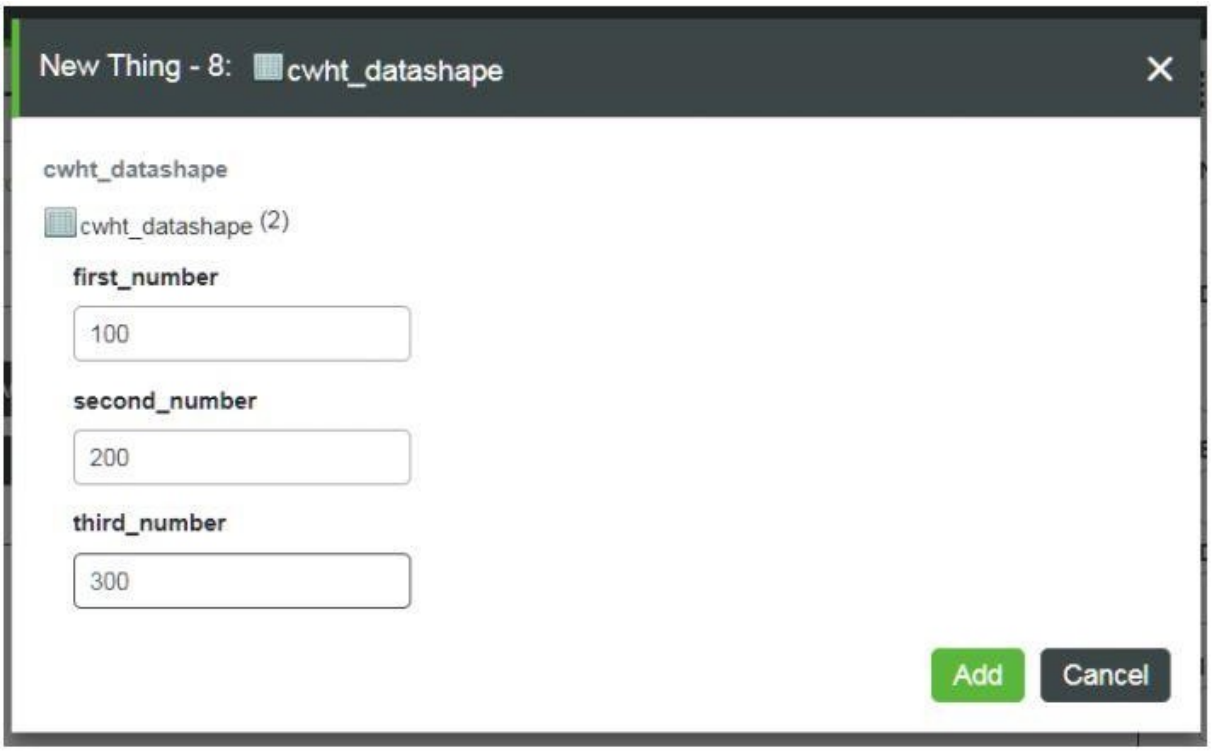


3. At the bottom-right, click the green **Add** button.

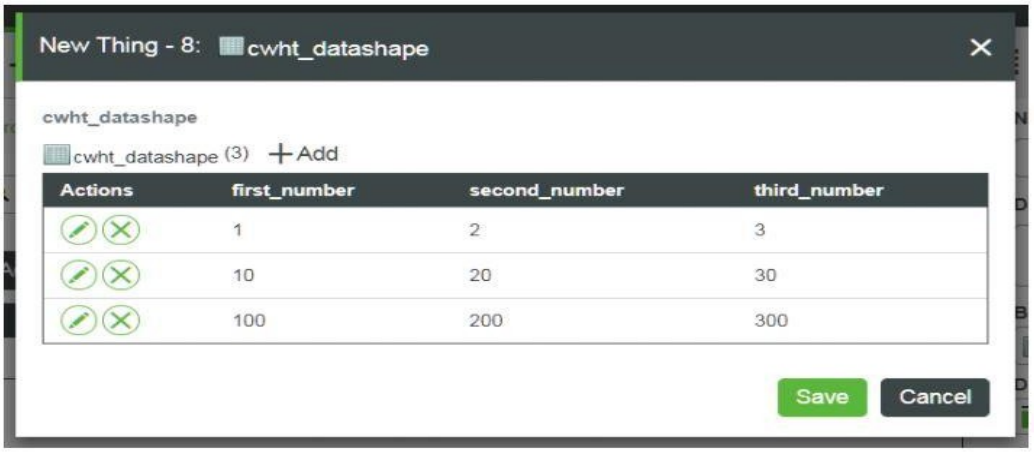


Third Default

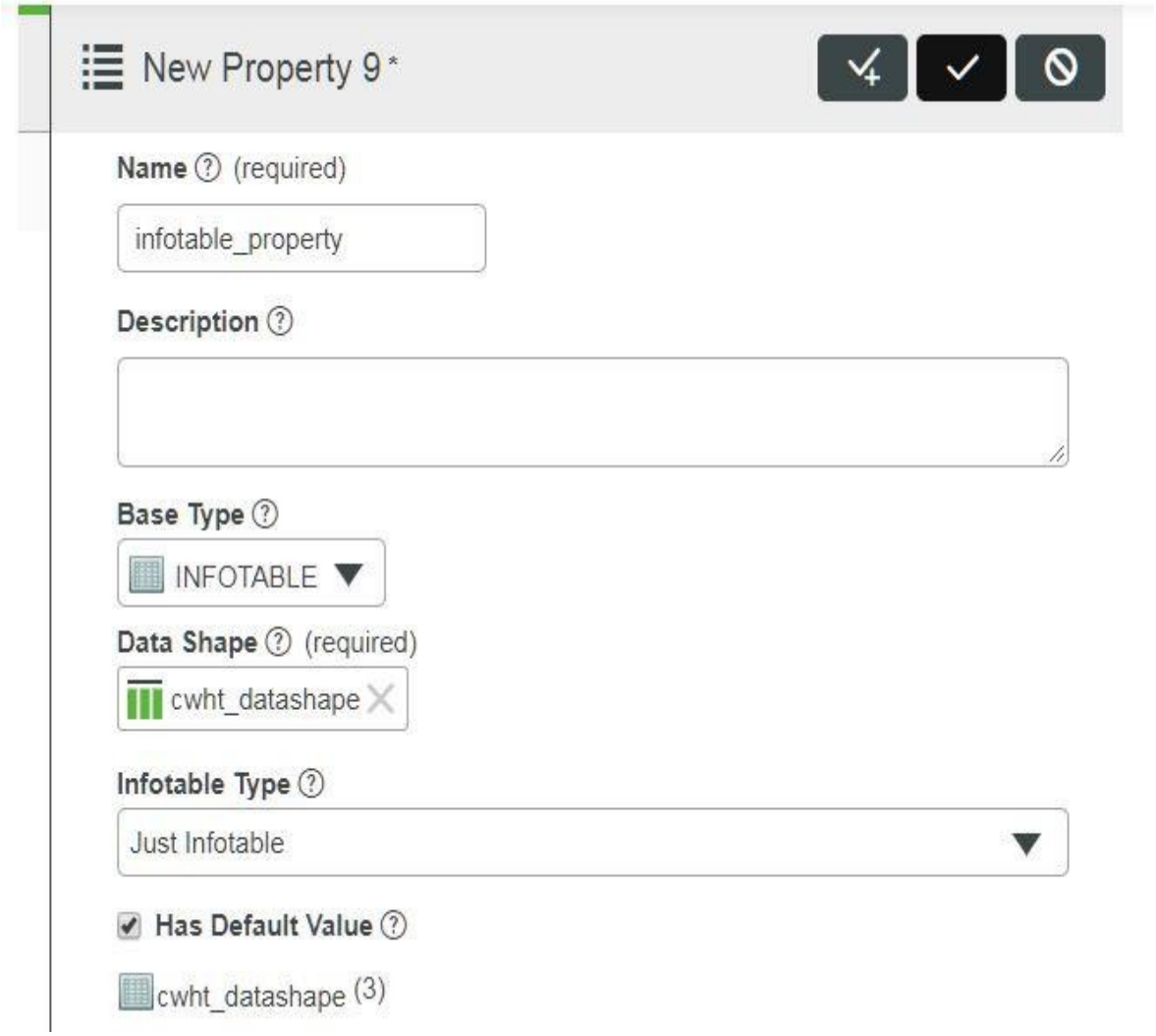
1. Click the **+ Add** icon.
2. Enter values in each number field, such as **100**, **200**, and **300**.



3. At the bottom-right, click the green **Add** button.



4. At the bottom-right, click **Save** to close the pop-up menu.



5. At the top-right, click the "Check" button for **Done**.

6. At the top, click **Save**

Basics Application Development with Thingworx

Basic Mashup Widgets

Use UI Widgets used by most Mashups

This will introduce how to use some basic Widgets in a Mashup. you will create a Mashup that reacts to user input using a Button, Toggle Button, and Slider Widget.

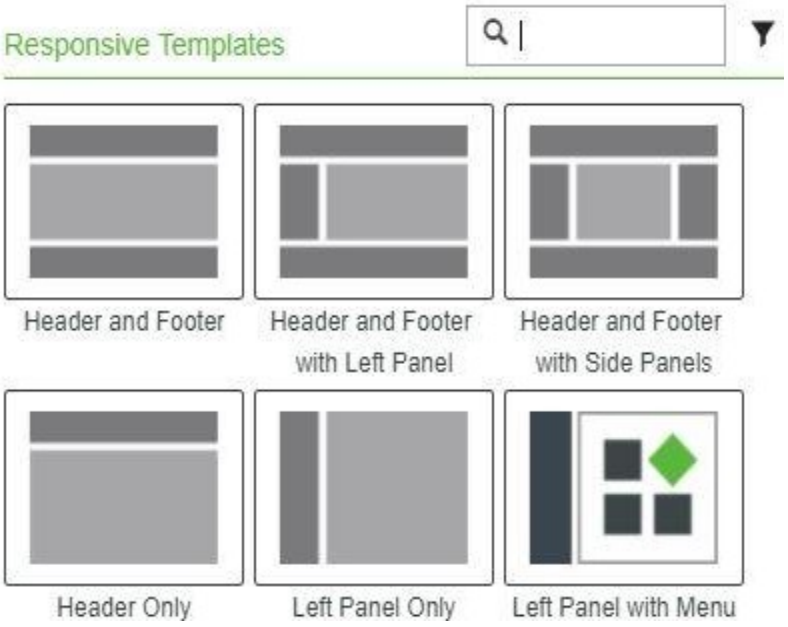
how to display data to users with the Grid Advanced, Gauge, and Property Display widgets.

Create Mashup

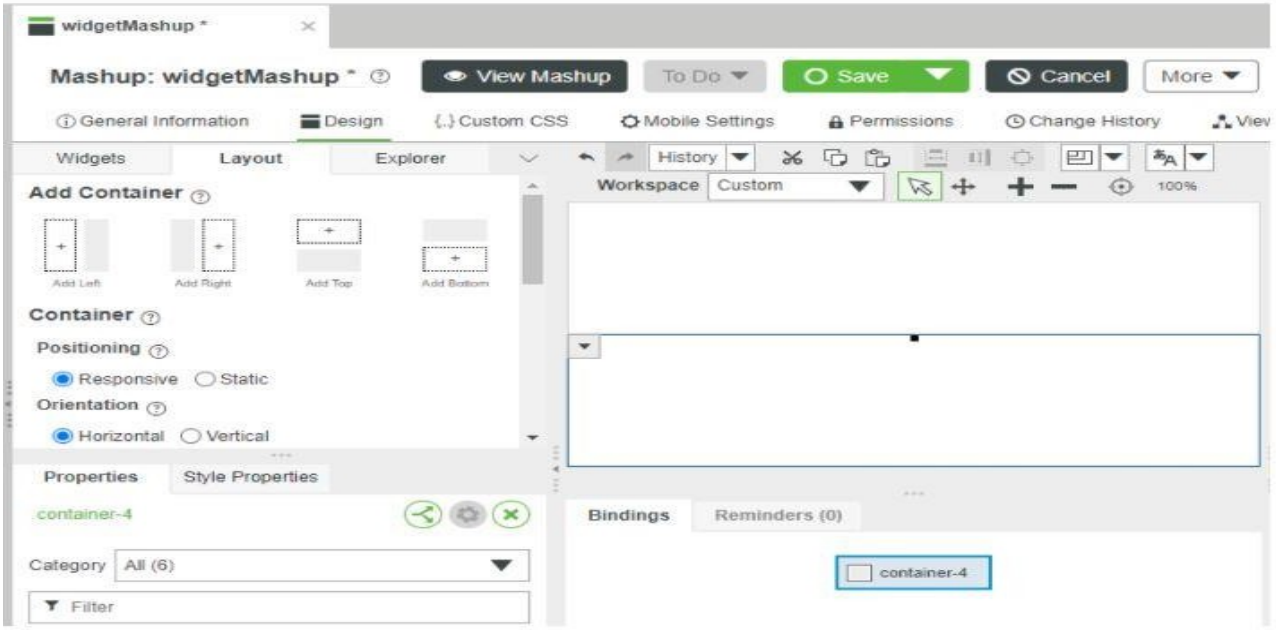
Build Mashup

1. Click the **Browse** folder icon on the top left of **ThingWorx Composer**.
2. Select **Mashups** in the left-hand navigation, then click **+ New** to create a new Mashup.

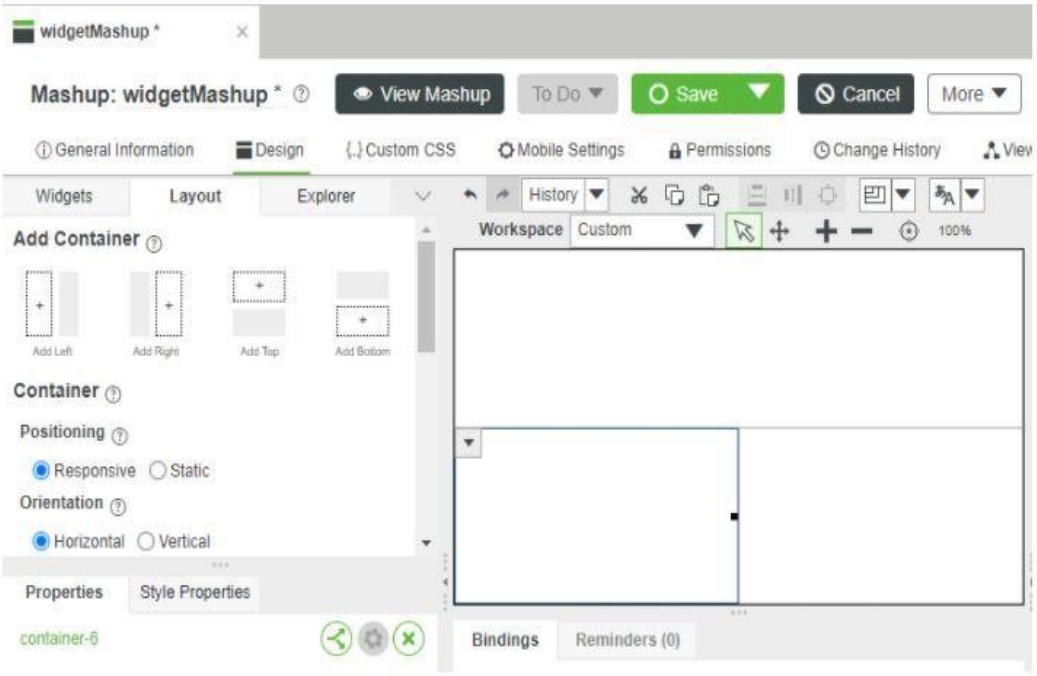
3. For **Mashup Type** select **Responsive**.



1. Click **OK**.
2. Enter a name for your Mashup.
3. If **Project** is not already set, click the **+** in the **Project** text box and select the **PTC Default Project**.
4. Click **Save**
5. Select the **Design** tab to display Mashup Builder
6. Select the **Layout** tab in the upper panel of the left dock.
7. Click **Add Bottom** to split the Mashup canvas into two halves.

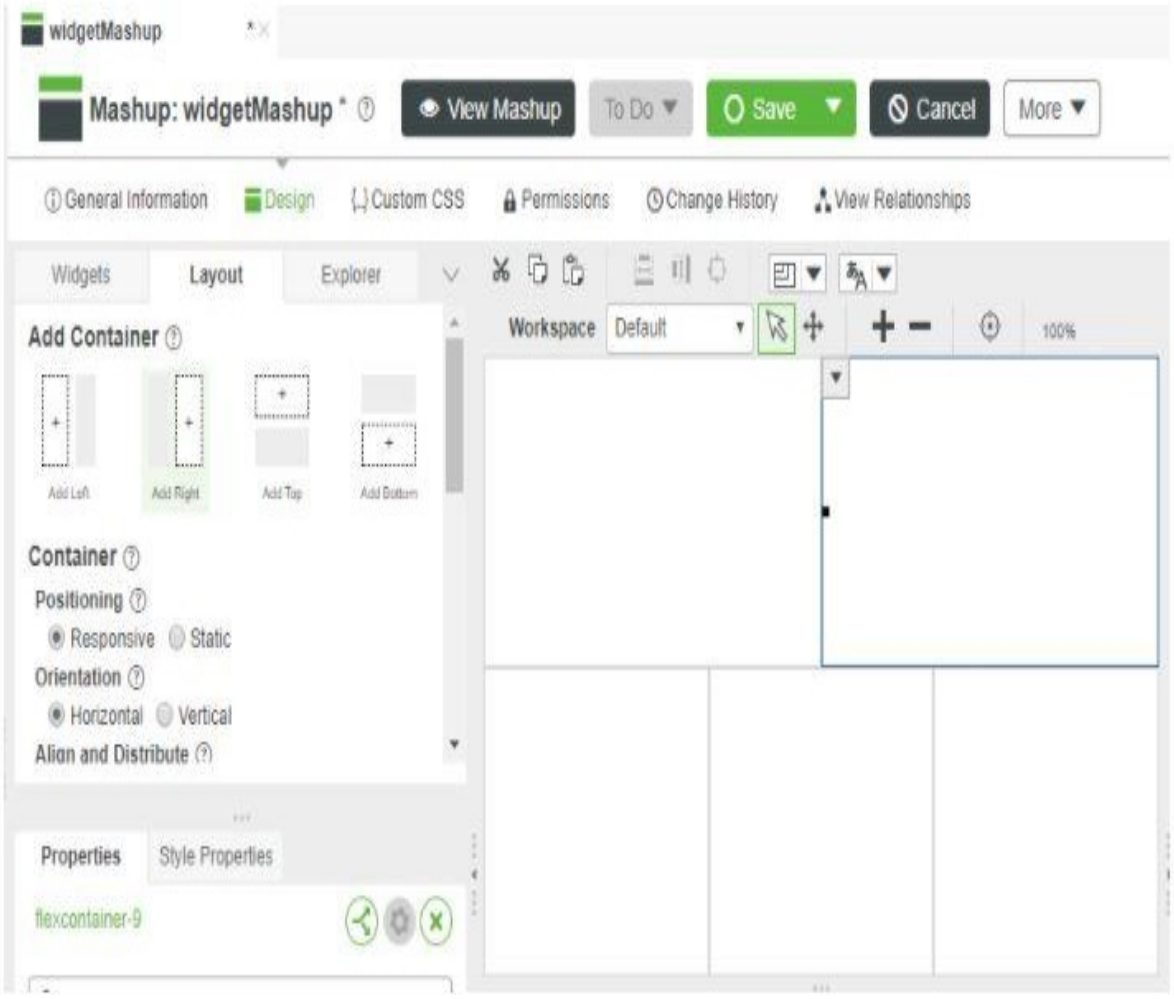


11. Click inside the bottom container to selected it, then click **Add Left**



12. Click inside the bottom-right container to select it, then click **Add Right**

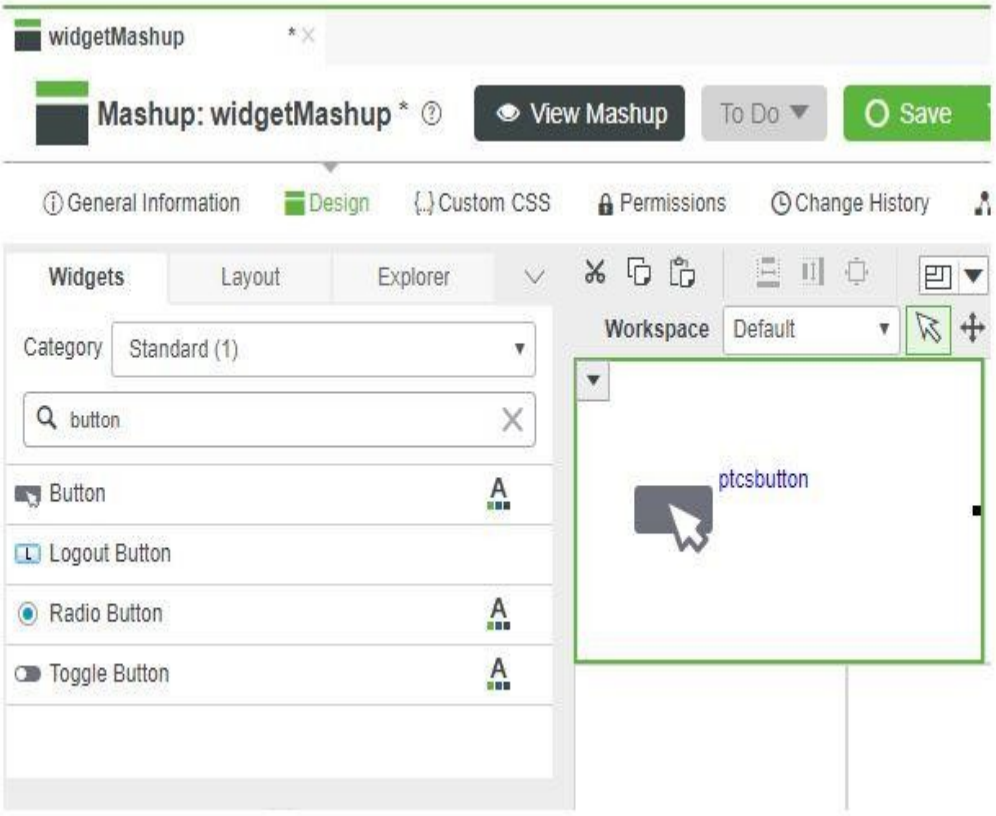
Click inside the top container to select it, then click **Add Right** again. You should now have 5 containers in two rows, ready to have widgets added.



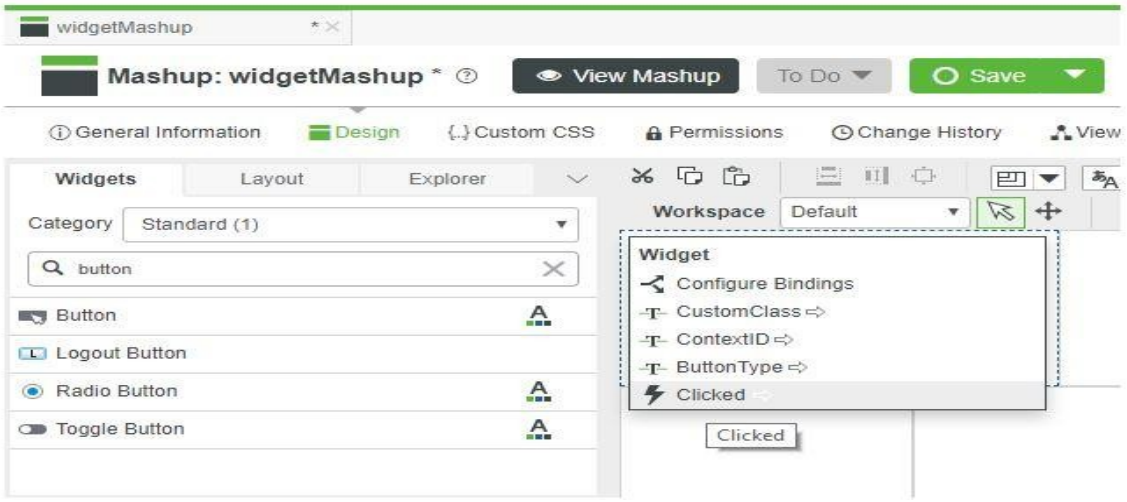
BUTTON

A button allows users to trigger an action, or stop and start long-running processes.

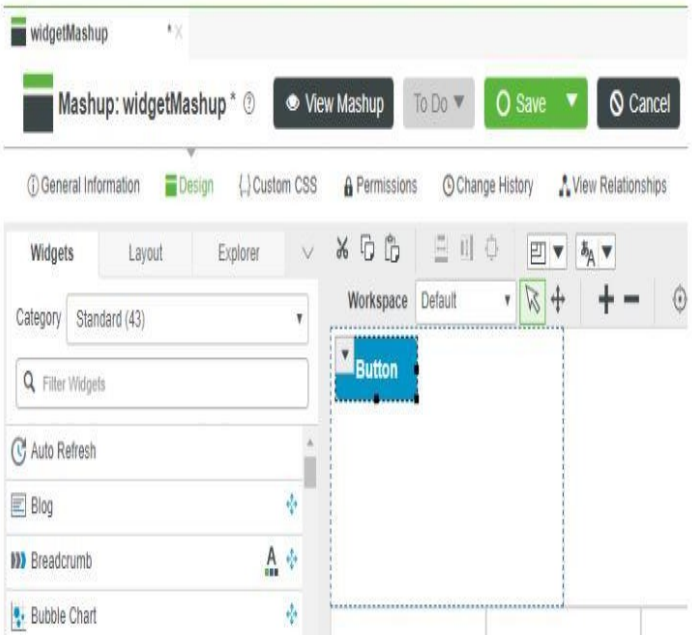
1. Select the **Widgets** tab in the upper panel of the left dock, then enter button inside the **Filter** field in the top-left.
2. Drag-and-drop the **Button** widget onto the upper left container.



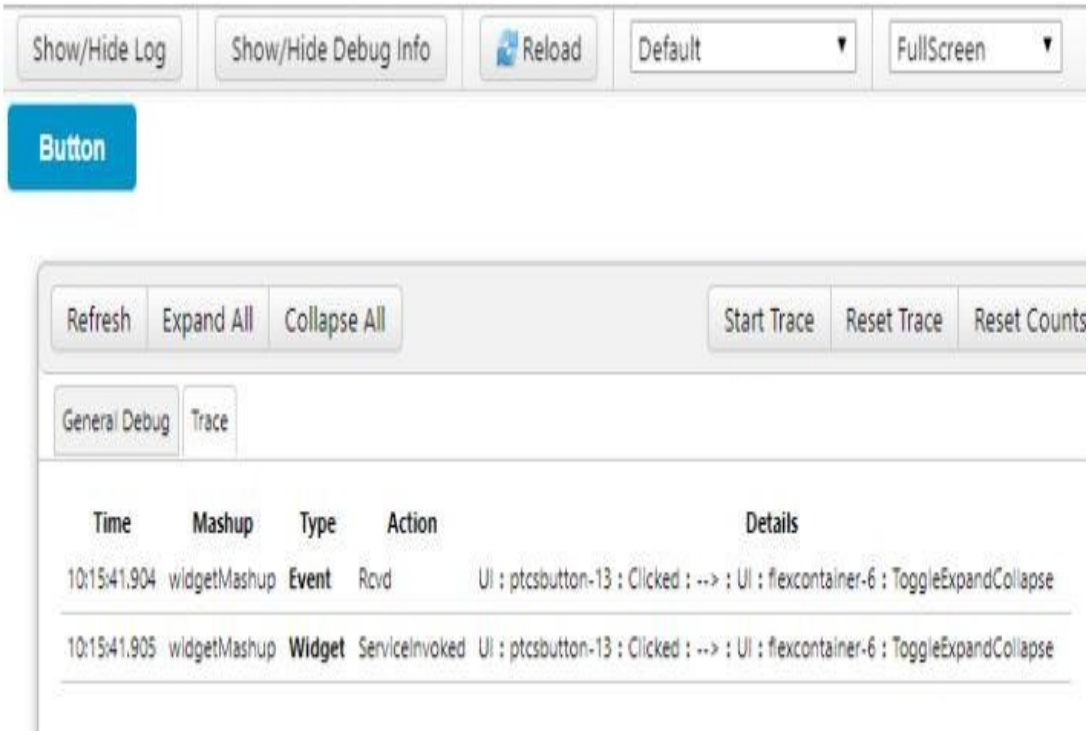
- Click the drop-down arrow on the left side of the Button widget
- Click and drag the **Clicked** service shown in the drop-down onto a free area of the Mashup canvas.



5. When the **Select Service** pop-up appears, Click the **ResetInputsToDefaultValues** service that is provided by the Container.



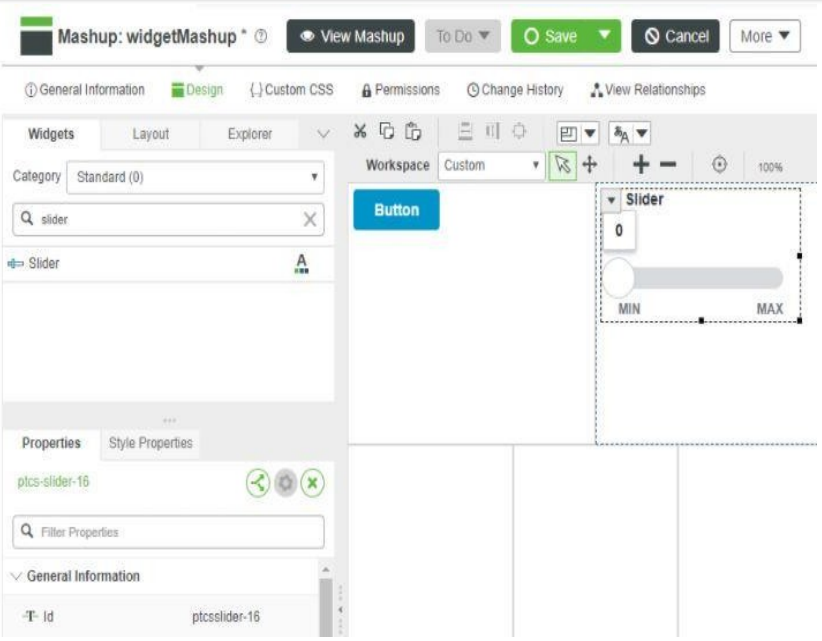
- Click **Save**
- Click **View Mashup** then click **Show/Hide Debug Info**
- Click the **Trace** tab and click **Start Trace**
- Click the button you created in your Mashup then click **Stop Trace** to see the log of the Clicked event triggering the Reset Inputs To Default Values service.



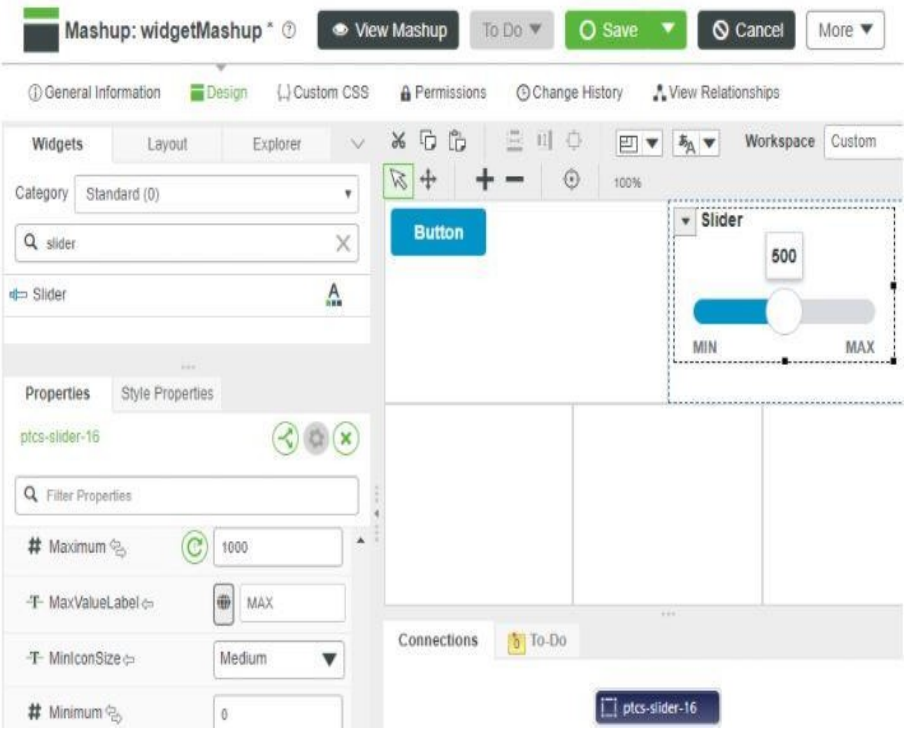
SLIDER

A Slider allows your application users to interactively select a numeric value. When the user changes the position of the slider bar, an event is fired and the value property is changed.

1. On the New Mashup tab, enter slider inside the **Filter** field in the top-left.
2. Drag-and-drop the **Slider** widget onto the top-right Container of your Mashup.



3. In the lower panel of the left dock, scroll to the **Maximum** property and change the value to 1000.



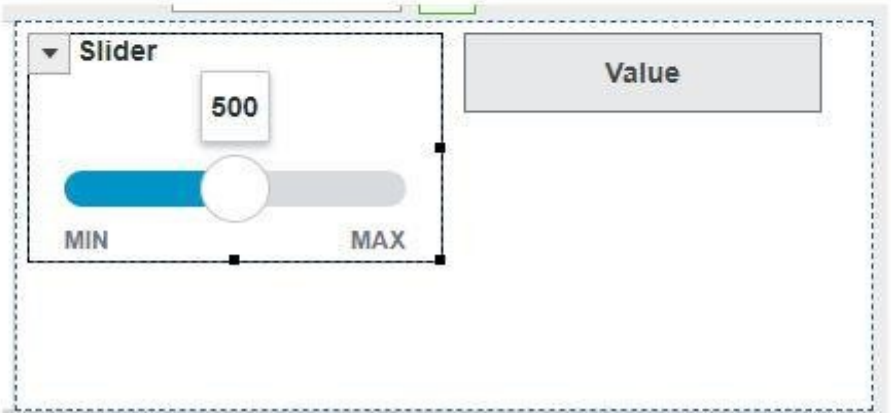
1. Change the **Value** property to 500 set the default position of the slider.

Add Value Display

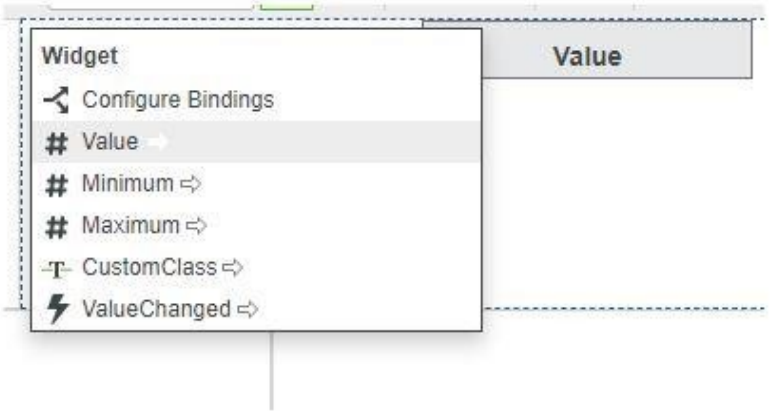
To demonstrate the Slider functionality, we will add a **Value Display** Widget within the same Container of your Mashup as the Slider and bind the slider widget output to the input of the Value Display.

1. In the **Filter** field where you entered **slider**, enter value.
2. Drag-and-drop a **Value Display** widget onto the same upper-right Container as the Slider widget

3. Click the drop-down arrow on the left side of the Slider widget.

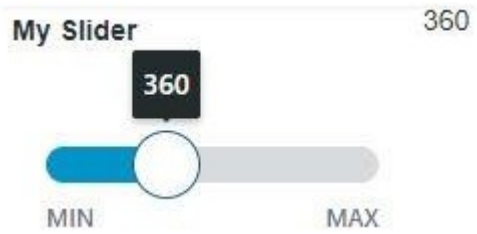


4. Click and drag the **Value** property listed in the Slider drop-down onto the Value Display widget



5. Click the **Data** property that is available from the **Select Binding Target** pop-up.

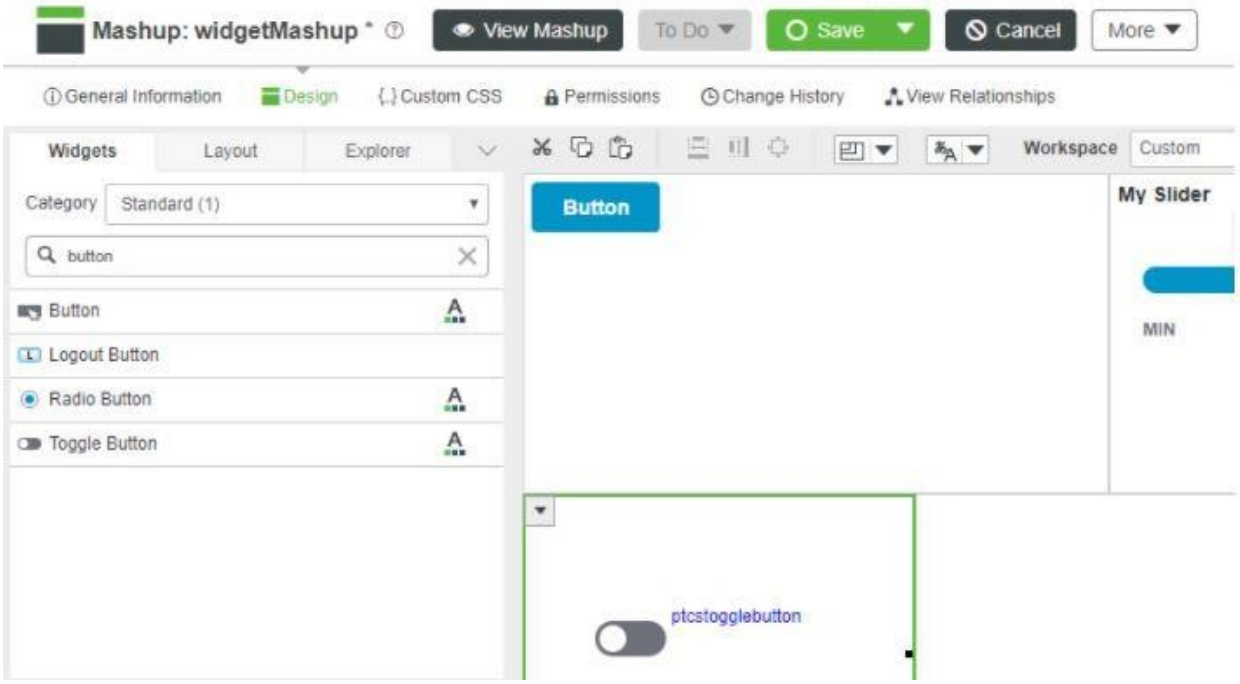
1. Click **Save** to save your Mashup.
2. Click **View Mashup** then adjust the slider to see the changing value shown in the Value Display widget.



Toggle Button

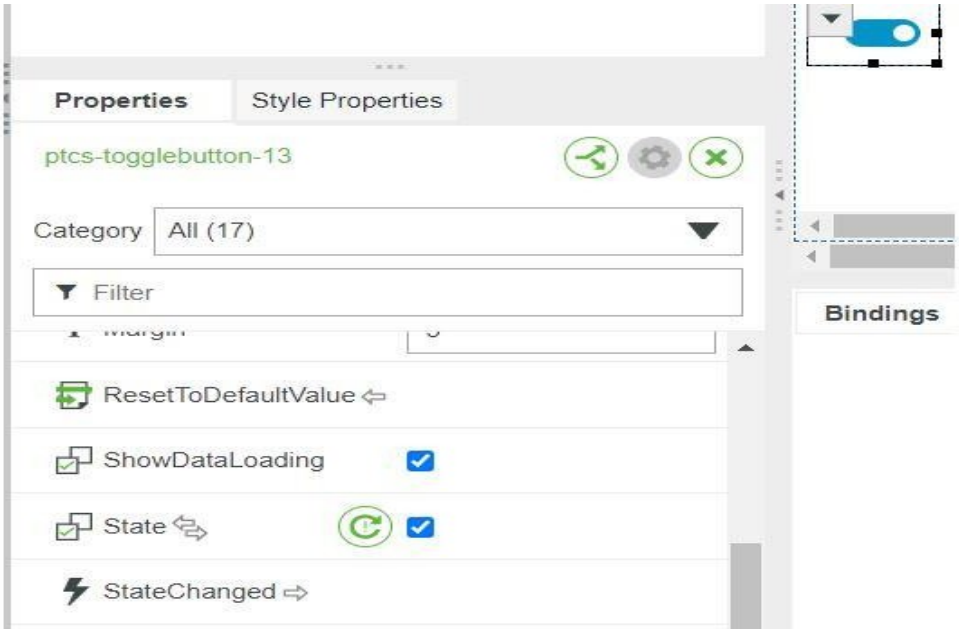
A Toggle Button widget is used when a Mashup user may select only ***one*** choice from two options.

1. On the **Widgets** tab in the top-left, enter button in the **Filter** field.
2. Drag-and-drop the **Toggle Button** widget onto the lower left container of the mashup.



Scroll to the **State** property in the lower panel of the left dock where Toggle Button widget properties are shown.

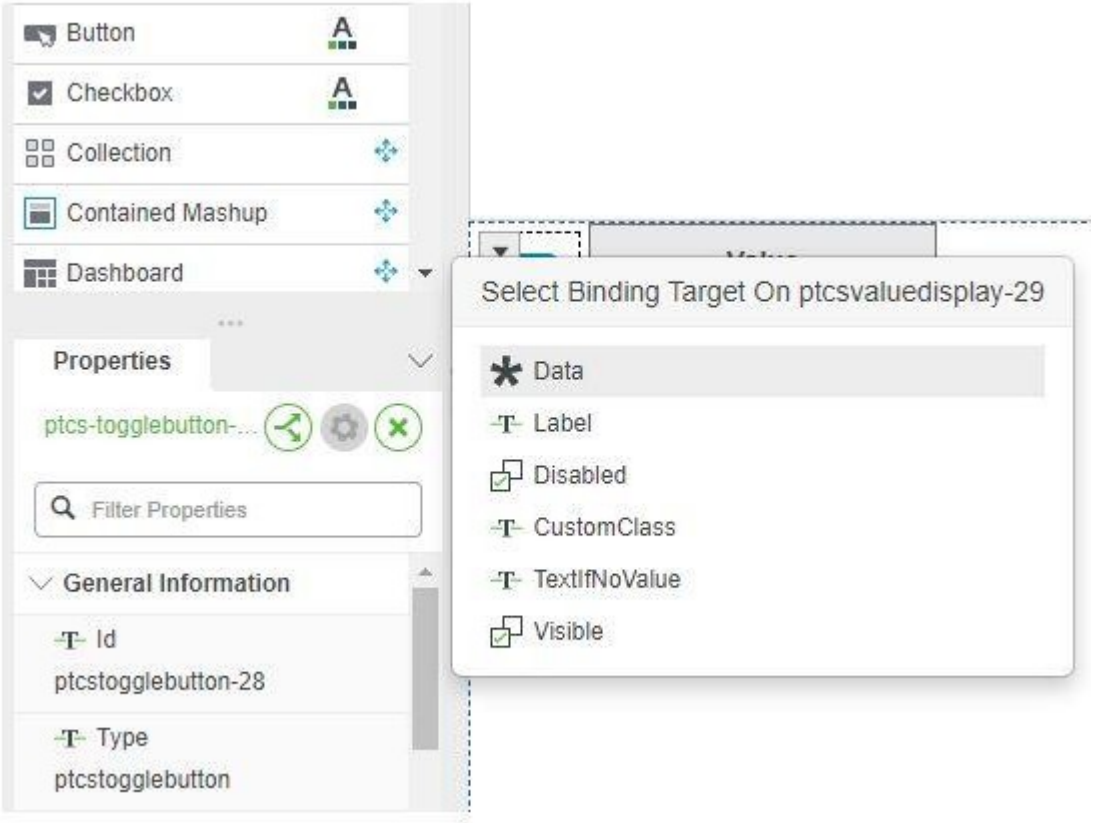
Click the check box to set the default state to **true**



Add Value Display:

To demonstrate the toggle Button, we will add a **Value Display** Widget to the same Mashup container and bind the toggle Button **State** property to be shown by the Value Display.

1. Drag-and-drop a **Value Display** Widget onto the same lower left container of the Mashup where the Toggle Button was placed.
2. Click the drop-down arrow on the left side of the Toggle Button Widget.
3. Click and drag the **State** property listed in the drop-down onto the Value Display Widget and a **Select Binding Target** pop-up will appear.

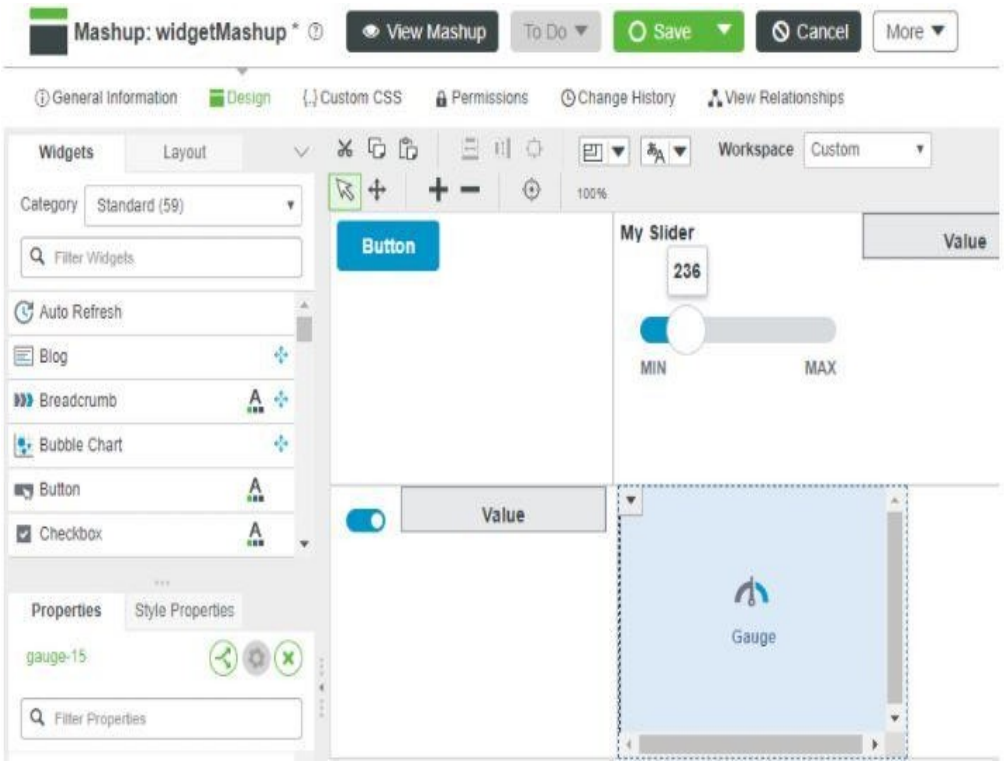


1. lick the **Data** property that is available from the Value Display Widget.
2. Click **Save** to save your Mashup so it can be tested.
3. Click **View Mashup** then click the Toggle Button to see the value of the button shown in the Value Display Widget.

GAUGE

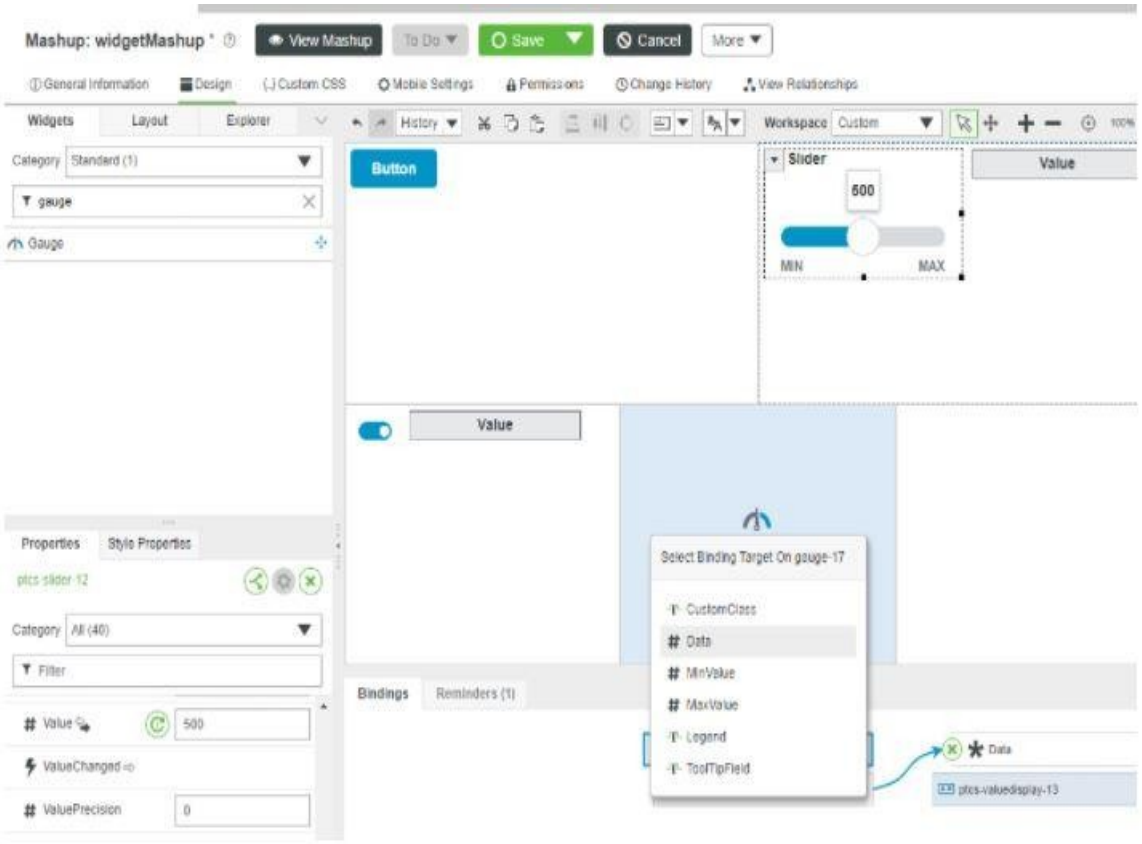
A Gauge is best for displaying a quickly readable approximate value.

1. Select the **Widgets** tab in the upper panel of the left dock, then enter gauge inside the **Filter** field in the top-left.
2. Drag-and-drop the **Gauge** widget onto the bottom-center container

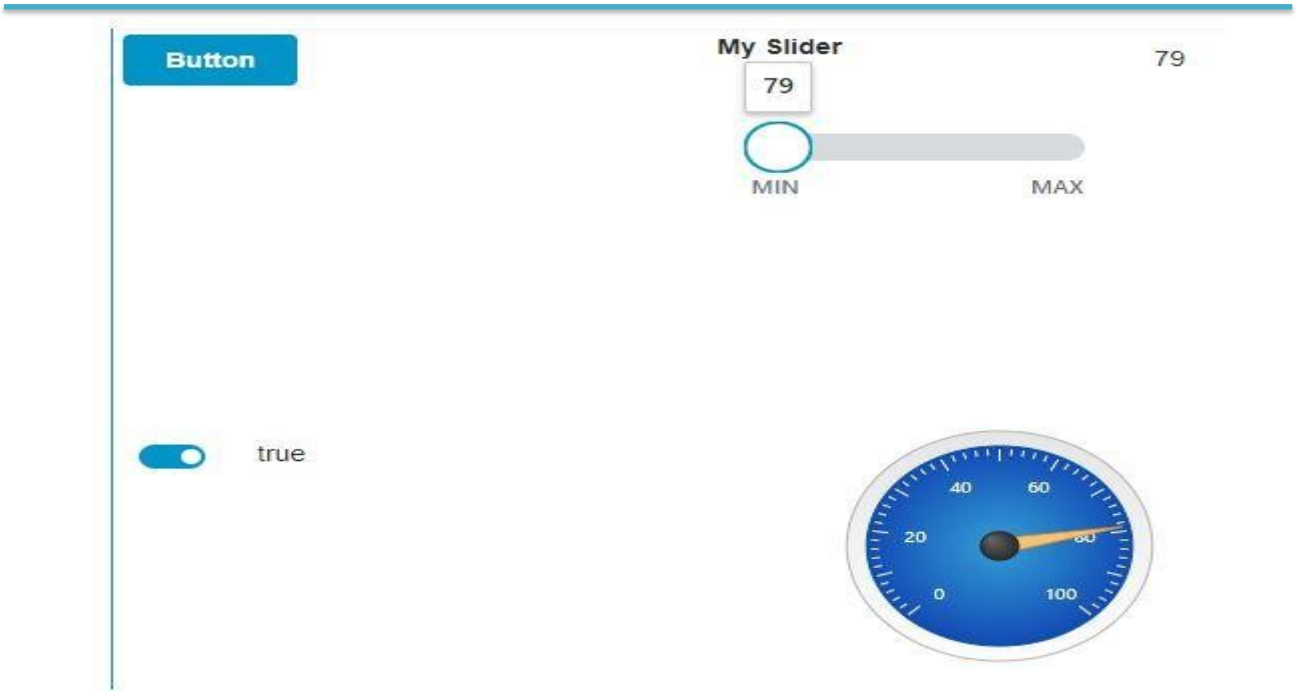


- We will now bind the Slider to the Gauge. Click the Slider Widget in the top-right Container.
- Click the drop-down arrow on the left side of the Slider widget.
- Click and drag the **Value** property listed in the Slider drop-down onto the Gauge widget

Click the **# Data** property that is available from the **Select Binding Target** pop-up



1. Click **Save** to save your Mashup.
2. Click **View Mashup** then adjust the slider to the left to see the changing value shown on the widget.



Configuration of Thingworx Kepware Server

Introduction of Kepware Server

Thingworx Kepware Server is the industry's leading connectivity platform providing a single source of industrial automation data to all of your applications. The platform design allows users to connect, manage, monitor, and control diverse automation devices and software applications through one intuitive user interface.

ThingWorx Kepware Server seamlessly integrates with and moves industrial control data to the Azure cloud platform by utilizing both OPC UA and MQTT via the Azure IoT Gateway SDK and Azure IoT Hub.

ThingWorx Kepware Server communicates with Microsoft® Azure IoT Hub using MQTT over TLS through Azure IoT Edge acting as

a transparent gateway. Once configured, modules in IoT Edge (such as Azure Stream Analytics, Azure ML, or custom code) can be used to "process" the data before sending it to the IoT Hub.

System Requirements

The server has minimum system requirements for both software and hardware. These requirements must
be met for the application to operate as designed.
This application supports the following Microsoft Windows operating systems:

- Windows 10 x64 (Pro and Enterprise Edition)3
- Windows 10 x86 (Pro and Enterprise Edition)
- Windows 8.1 x64 (Windows 8, Pro, and Enterprise Edition)3
- Windows 8.1 x86 (Windows 8, Pro, and Enterprise Edition)
- Windows 8 x64 (Windows 8, Pro, and Enterprise Edition)3
- Windows 8 x86 (Windows 8, Pro, and Enterprise Edition)
- Windows Server 2019 x643,4
- Windows Server 2016 x643,4
- Windows Server 2012 x64 R23
- Windows Server 2012 x643

Notes

1. When installed on a 64-bit operating system, the application runs in a subsystem of Windows called
WOW64 (Windows-On-Windows 64 bit). WOW64 is included on all 64-bit versions of Windows and is
designed to make differences between the operating systems transparent to the user. WOW64
requires the following minimums:

1 GHz Processor

1 GB installed RAM (defer to the suggestion for the OS)

530 MB available disk space

Ethernet Card

2. Verify the latest security updates are installed for the operating system.
3. Runs in the 32-bit compatibility mode.
4. Windows Server Core deployments are not supported

Navigating the User Interface

The Configuration provides the general means of interacting with the server Runtime. While various plug-ins
and drivers add buttons, menus, and icons; the standard interface elements are described below.

Title Bar

Displays the application name, when Configuration is connected to the Runtime, and the current Runtime project when applicable.

Menu Bar

File Includes the project-level commands; such as Save, Open, Import, and Export.

Edit Includes action commands; such as Copy, Paste, and New Channel.

View Includes the display commands; such as which elements of the user interface are visible or hidden and the type of tree organization to display.

Tools Includes the configuration commands; such as general options, connection settings, event log filters; and access to the License Utility And. Quick Client

Runtime Includes server connectivity commands; such as Connect..., Disconnect, and Reinitialize.

Help Includes commands to access the product documentation, by server, driver, or plug-in.

Button Bar

The standard buttons are described below. Plug-ins and drivers add, remove, enable, and disable buttons
based on available functionality for the active items and view.

New Project: Initiates creation of a new project file to replace the active project. The project file
defines the devices connected, their settings, and how they are grouped.

Open Project: Allows the user to browse for an existing project file to load, replacing the active project.

Save Project: Implements any recent changes and writes the active project file to disk.

Save As: Writes the active project with changes, such as to a new location or file name.

New Channel: Creates a new group or medium for data collection.

New Device: Defines a new hardware component or PLC for data collection.

New Tag Group: Defines a new collection of data points, or tags, that can be organized as a single unit.

New Tag: Defines a new data points for collection.

Bulk Tag Creation: Defines tags discovered in the target device or environment.

Duplicate Tag: Creates a copy of the selected tag.

Properties: Allows viewing and editing of parameters for the selected item.

Undo: Resets the value or item to its configuration prior to the most recent change.

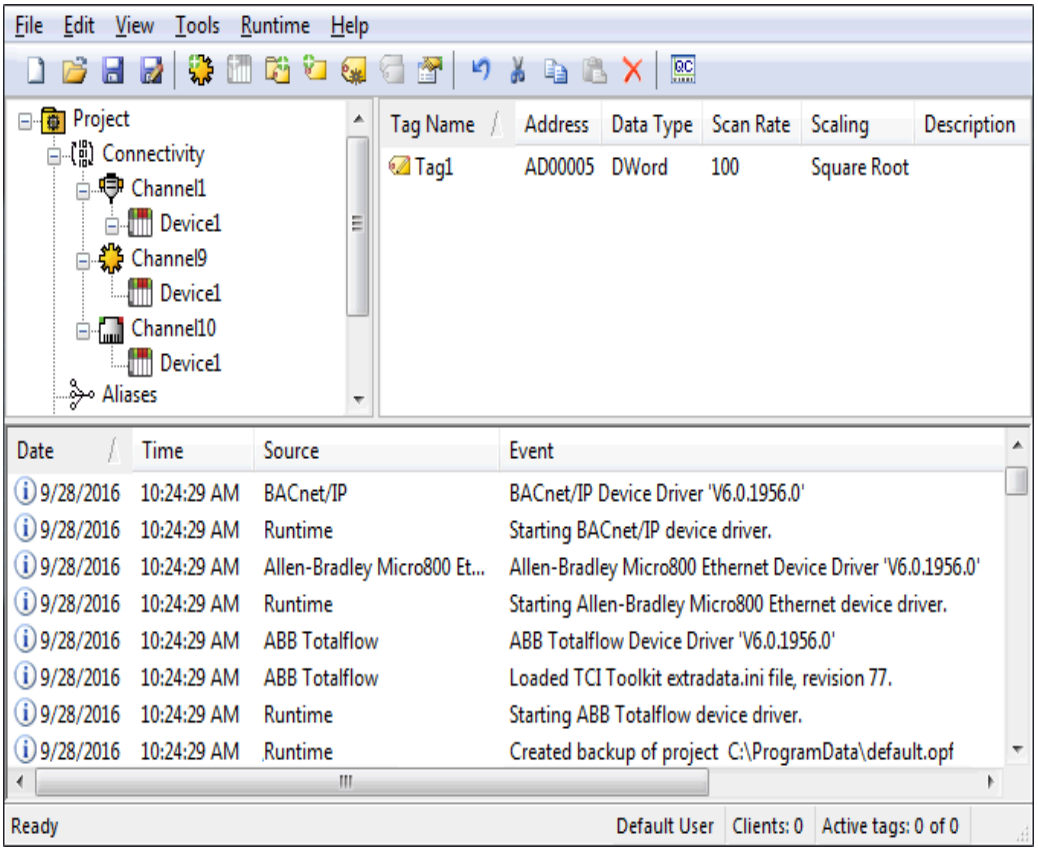
Cut: Removes the selected item and stores it on the clipboard.

Copy: Creates a duplicate of the selected item and stores it on the clipboard.

Paste: Inserts an item currently in the clipboard into the selected area.

Delete: Removes the selected item and / or its definition.

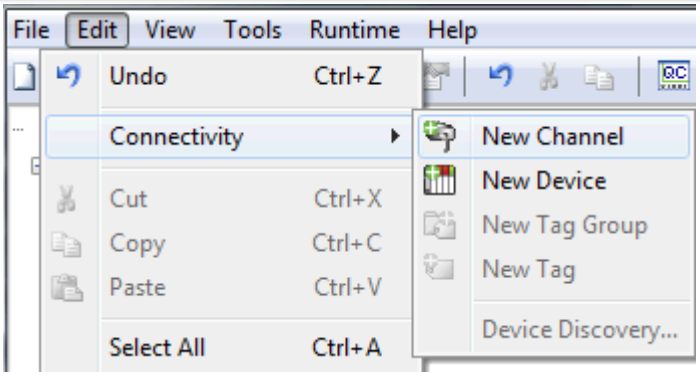
Quick Client: Runs the integrated client interface.



Above picture for your reference to see all the above mention option.

Adding and Configuring a Channel

1. To start, add a new channel to the project by: clicking Edit | Connectivity | New Channel - OR
clicking the New Channel icon on the toolbar
- OR
right-clicking on the Connectivity node in the tree and choosing New Channel



2. In the channel wizard, leave the channel name at its default setting "Channel1". Then, click **Next**.
3. In Device Driver, select the communications driver to be applied to the channel. Then, click **Next**. In
this example, the Simulator Driver is used.

4. For the Simulator Driver, the next page is Channel Summary. Other devices may have additional
channel wizard pages that allow the configuration of other properties (such as communications port,
baud rate, and parity). For more information, refer to Channel Properties — Serial Communication.

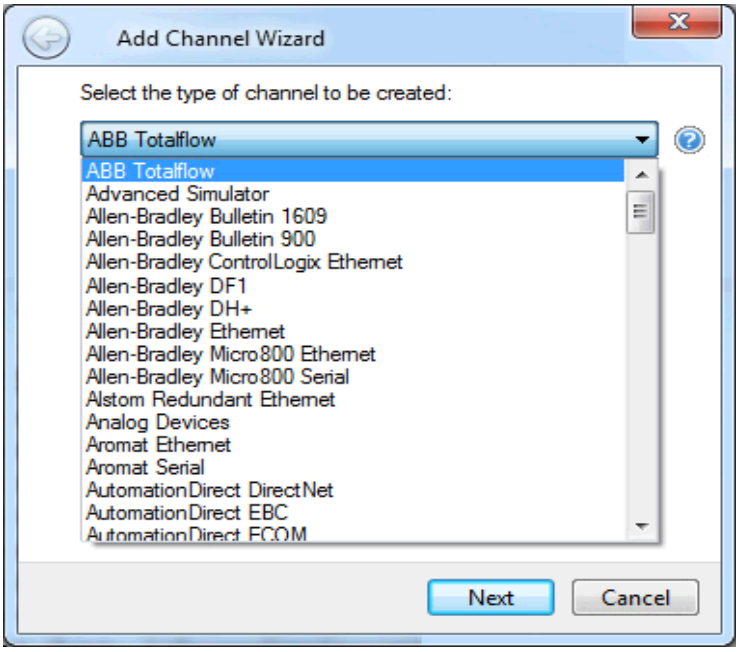
5. Once complete, click Finish.

Channel Creation Wizard

The Channel Creation Wizard steps through the process of configuring a channel (defined by the protocol being used). Once a channel is defined, its properties and settings are used by all devices assigned to that channel. The specific properties are dependent on the protocol or driver selected.

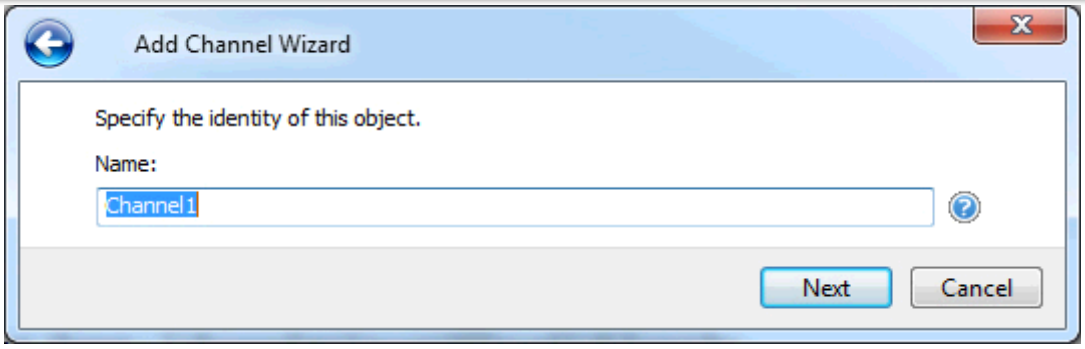
1. In the tree view, right-click on the Connectivity node and select New Channel (or choose Edit | Connectivity | New Channel).

2. Select type of channel to be created from the drop-down list of available drivers.



3. Click Next.

4. Enter a name for the channel to help identify it (used in tag paths, event log messages, and aliasing).



5. Click Next.

6. Configure the channel properties according to the options and environment.

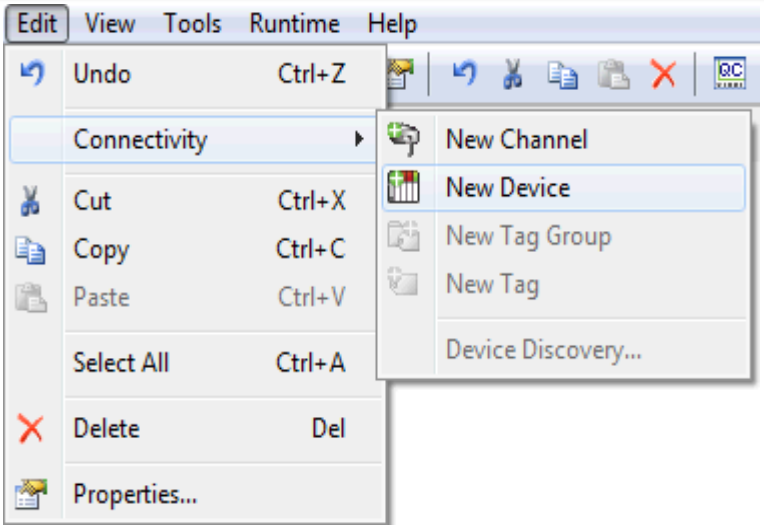
7. Review the summary for the new channel and choose Back to make changes or Finish to close.

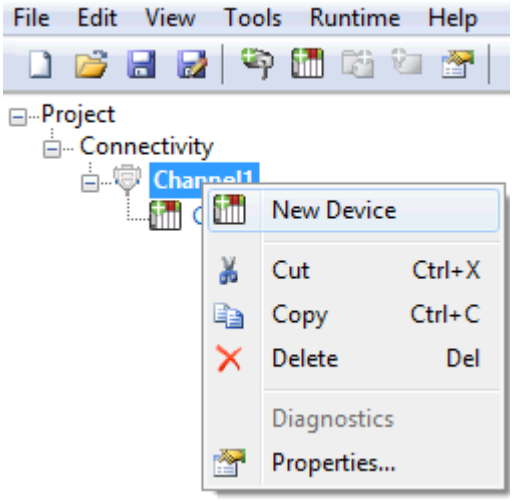
Adding and Configuring a Device

Once a channel has been defined, a device can be added. The device identifies a communication link's physical node or station, and can be thought of as a way to frame the connection's definition to a specific point of interest in the application. In this respect, a device is the correct term for describing the connection to a database object. As such, "device" refers to a specific device on a network, support multiple device nodes, and allows users to simulate networked devices.
Note: In this example, the Simulator Driver is used. The options in device wizard depend on the driver.

1. To start, select the channel to which the device will be added.

2. To start, add a new device to the project by: clicking Edit | Connectivity | New Device - ORclicking the New Device icon on the toolbar - ORright-clicking on the Connectivity node in the tree and choosing New Device





3. In the device wizard, leave the name at its default setting "Device1" and click Next.

4. In Model, select either an 8 or 16-bit register size for the device being simulated and click Next.
Note: Other device drivers may require users to select a device model instead. For this example,
the 16-bit register size is chosen.

5. In ID, select the device ID (which is the unique identifier required by the actual communications protocol). Then, click Next.
Note: The device ID format and style depend on the communications driver being used. For the Simulator Driver, the device ID is a

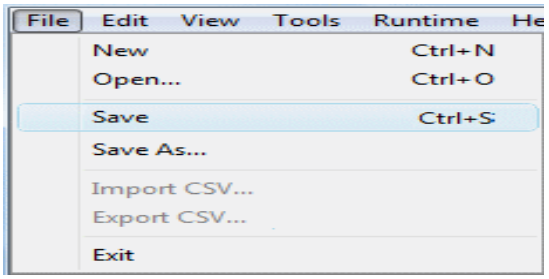
numeric value.

6.In Scan Mode, specify the device's scan rate. Then, click Next.

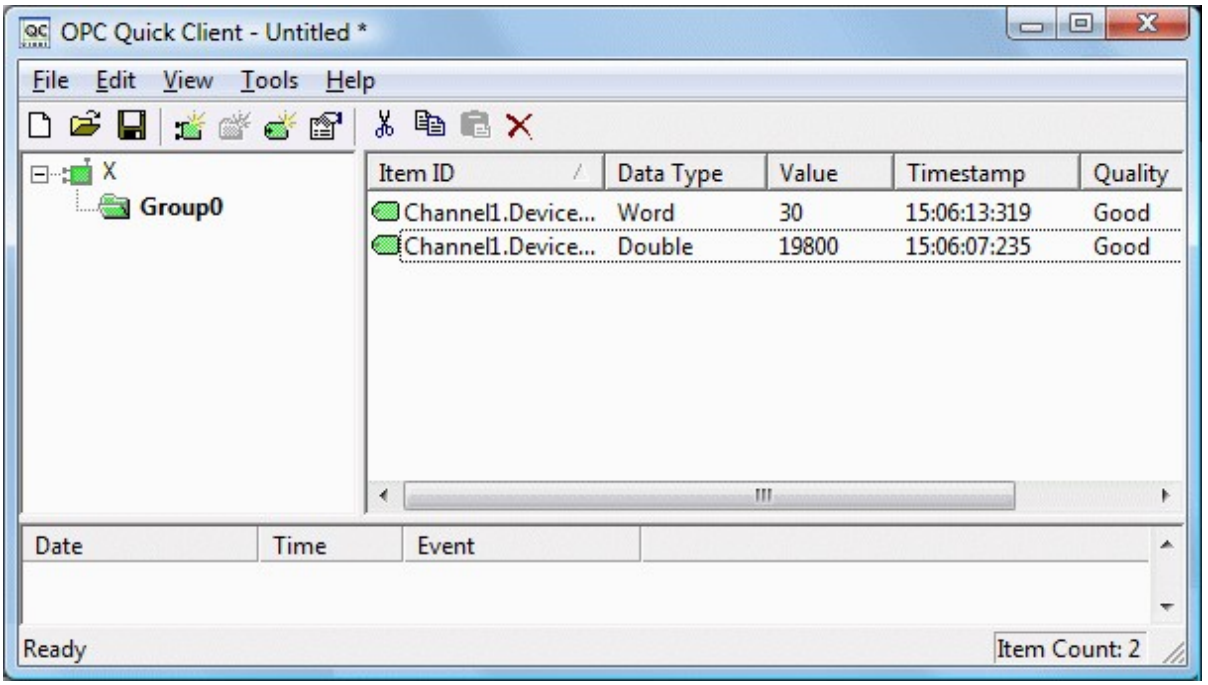
7. For the Simulator Driver, the next page is the Device Summary. Other drivers may have additional
device wizard pages that allow the configuration of other properties (such as Timing). For more
information, refer to Device Properties.

8. Once complete, click Finish

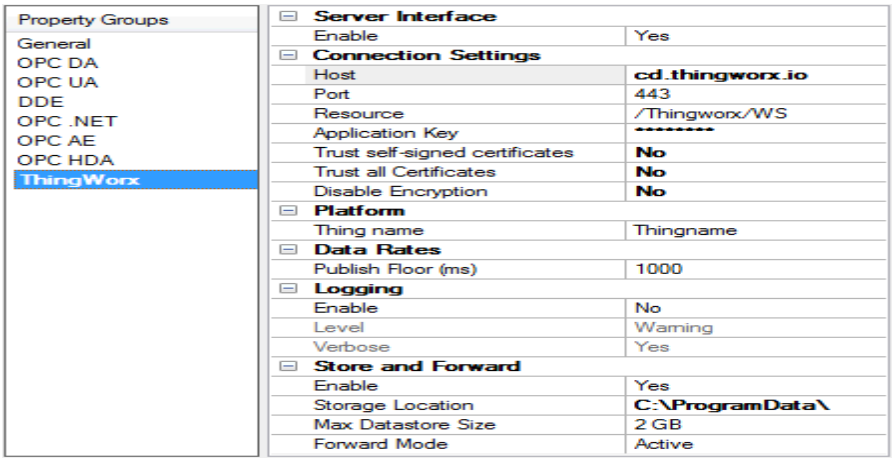
Saving the Project



OPC Quick Client



Thingworx Connection setting



Do the below setting for connection with Thingworx

Enable yes

Host - IP address of Thingworx Server

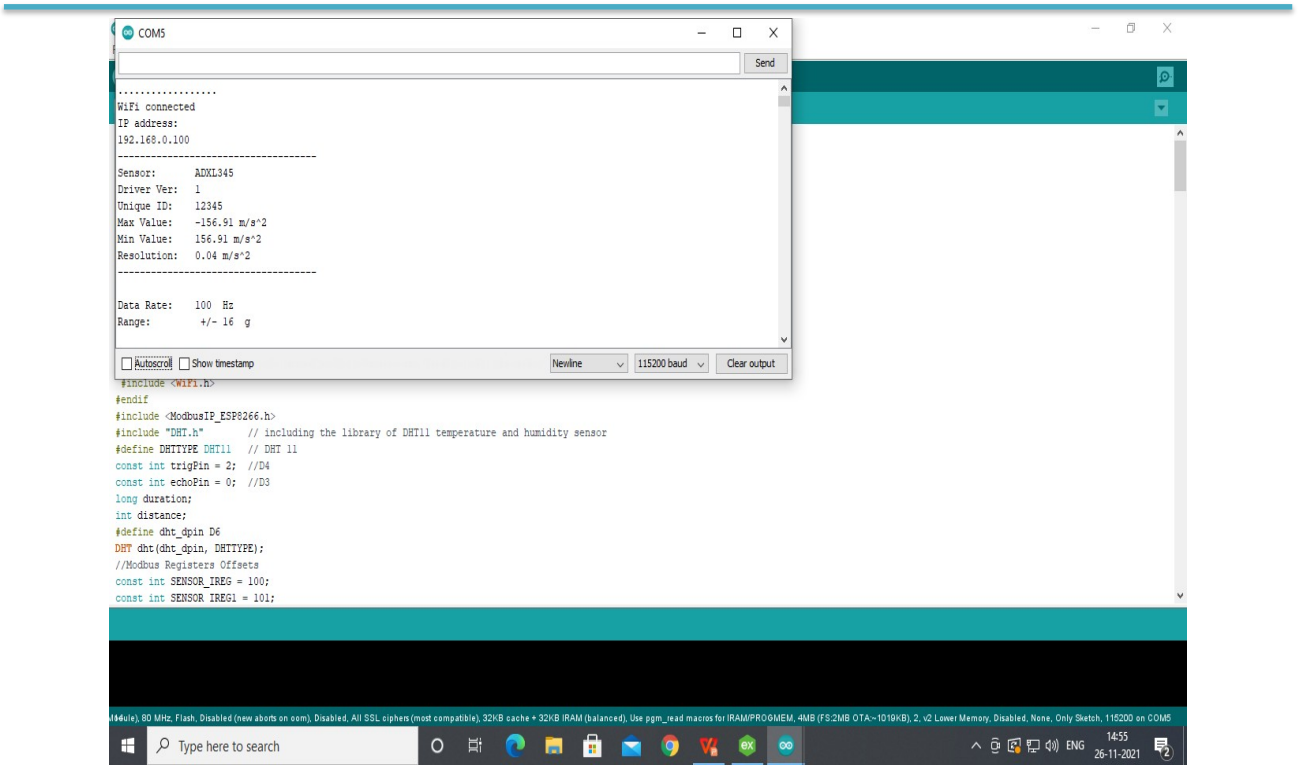
Assign the port number

Application Key

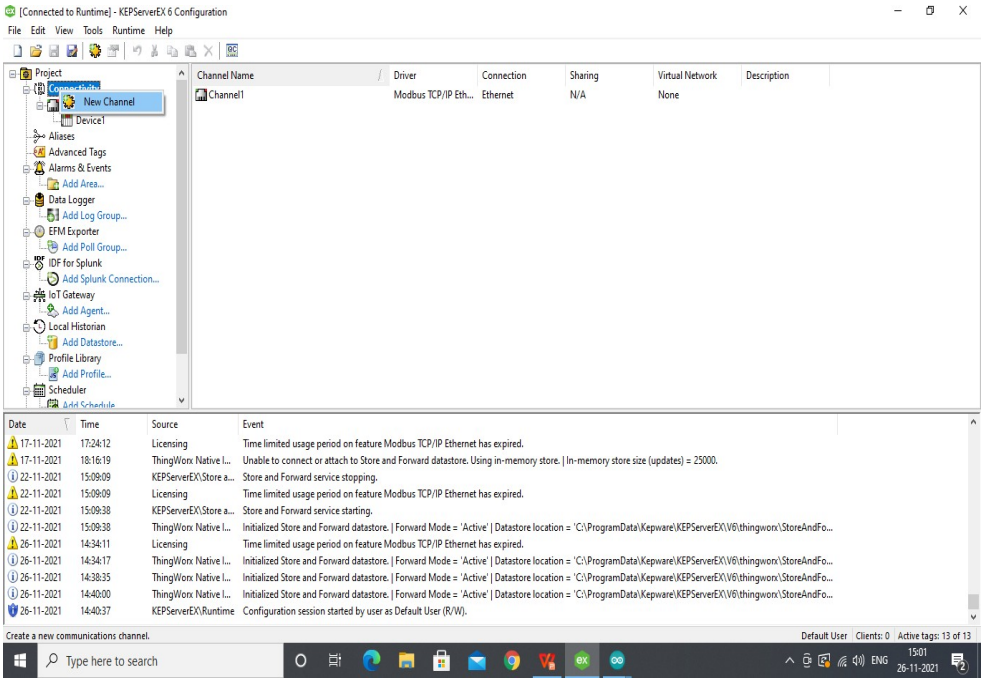
Thing Name

Configuration of NodeMCU with Kepware server Steps

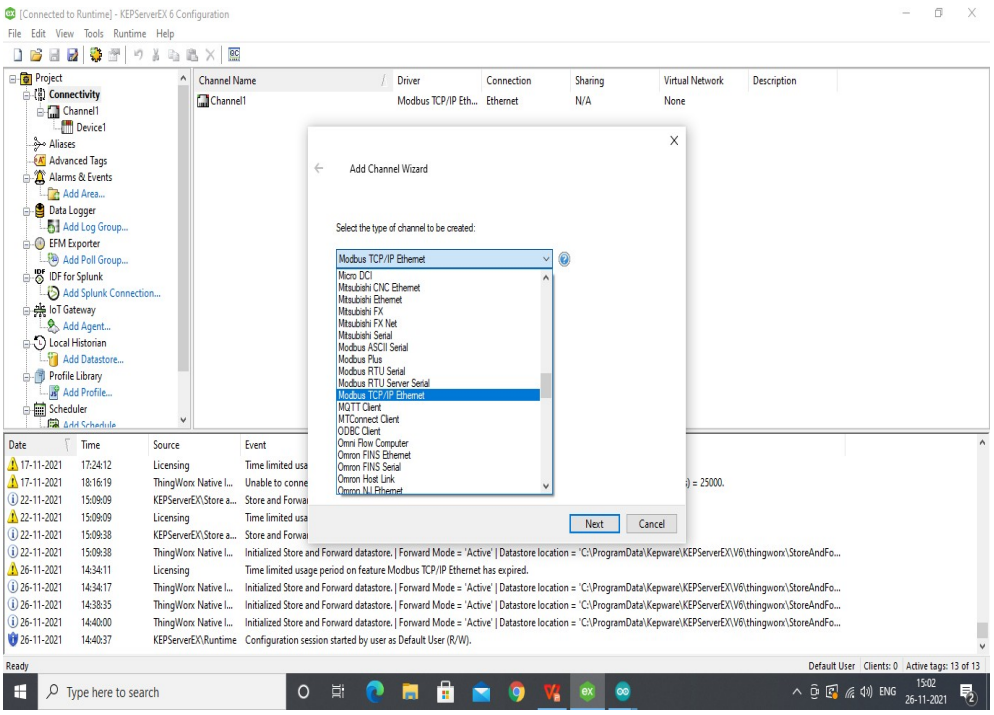
Check the IP of NodeMCU in Arduino software



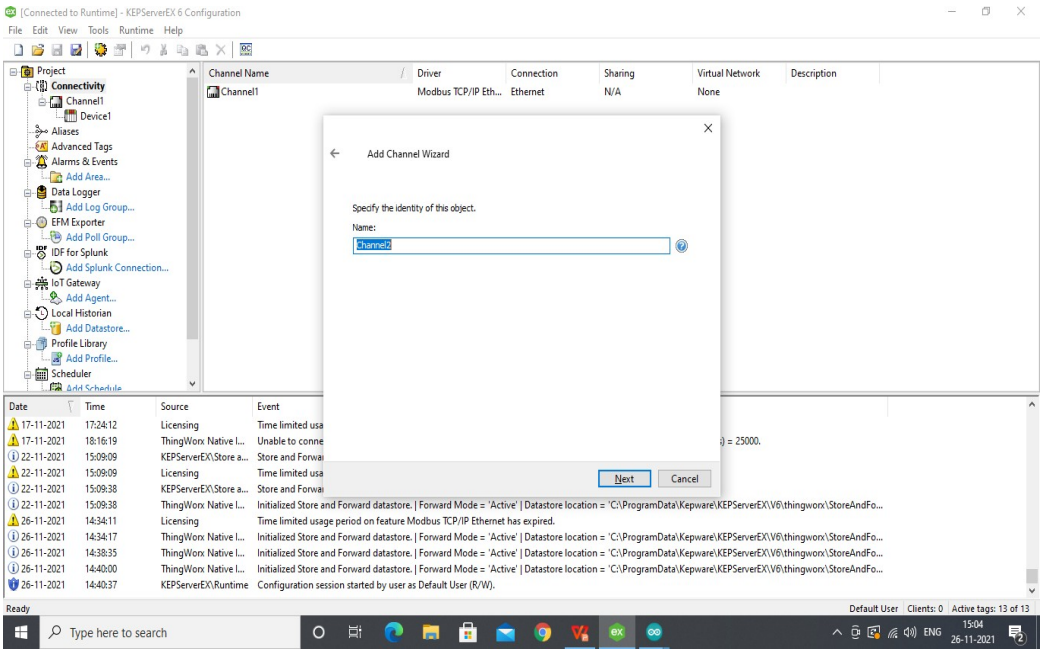
Open Kepware software. create new channel



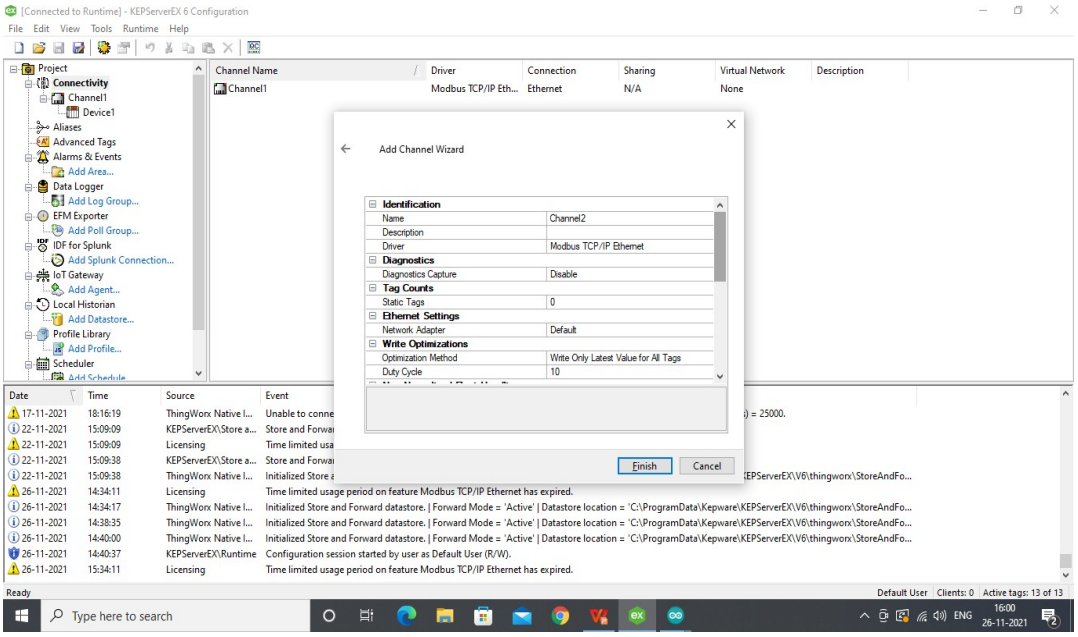
Select the type of the channel to be created



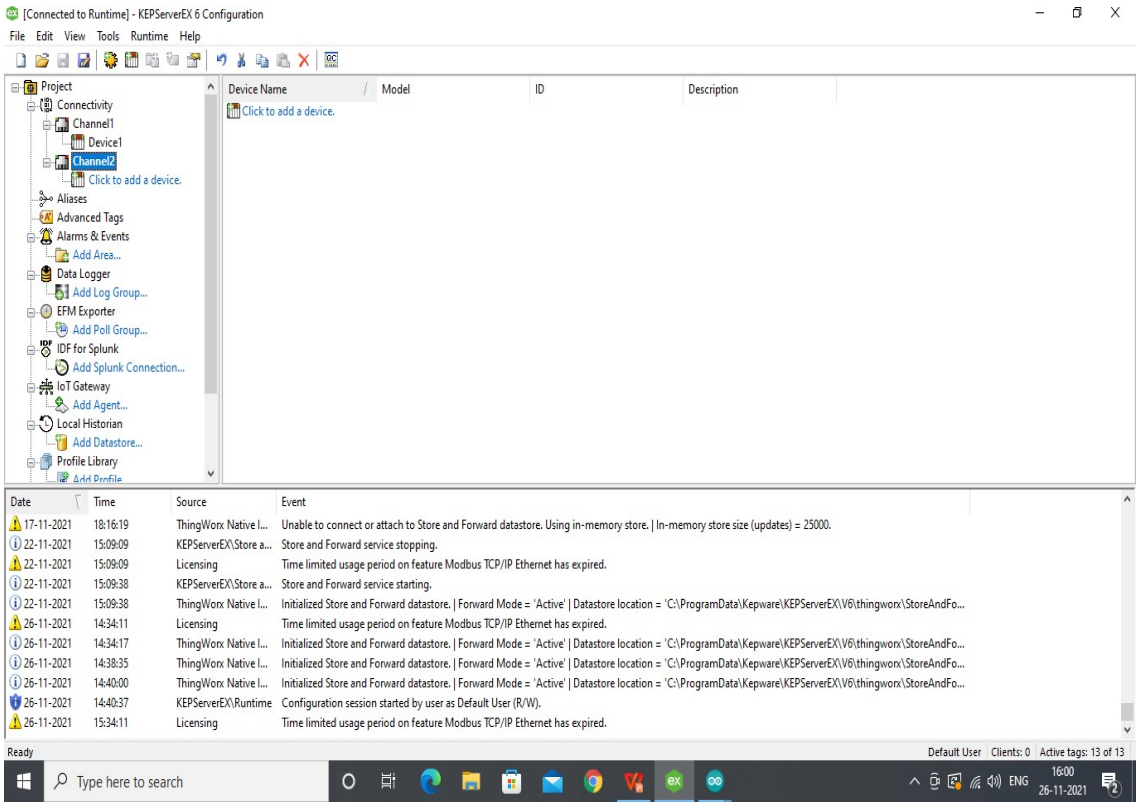
Give the channel Name like channel 2 and then click next till finish to set default setting.



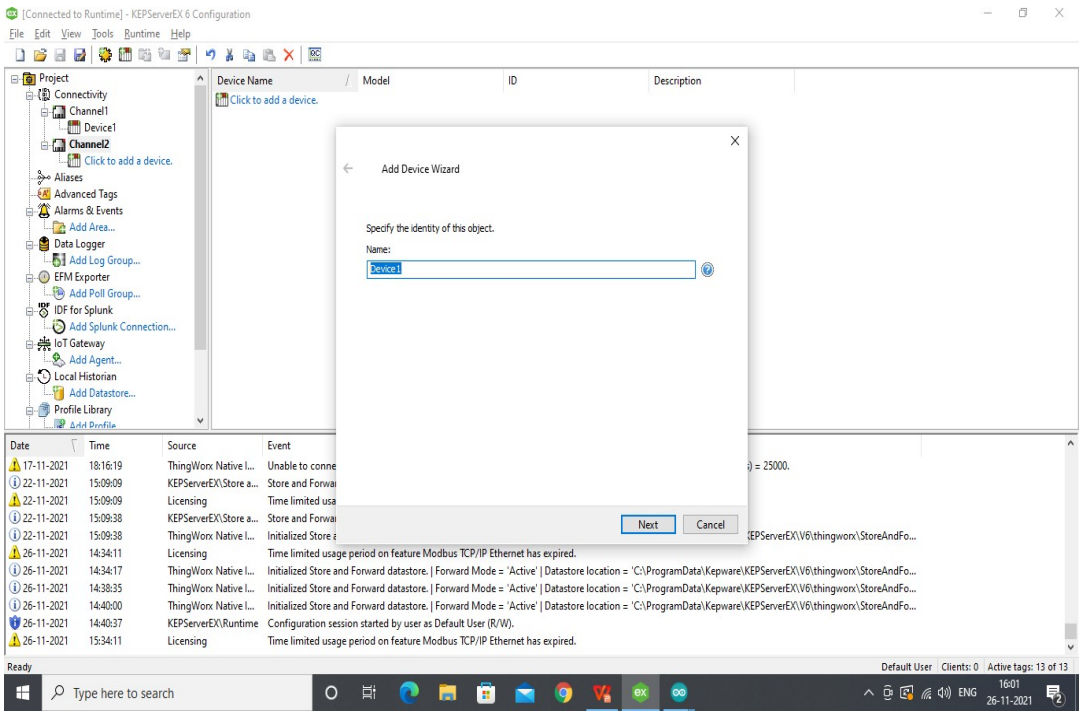
Click Finish to create the channel 2



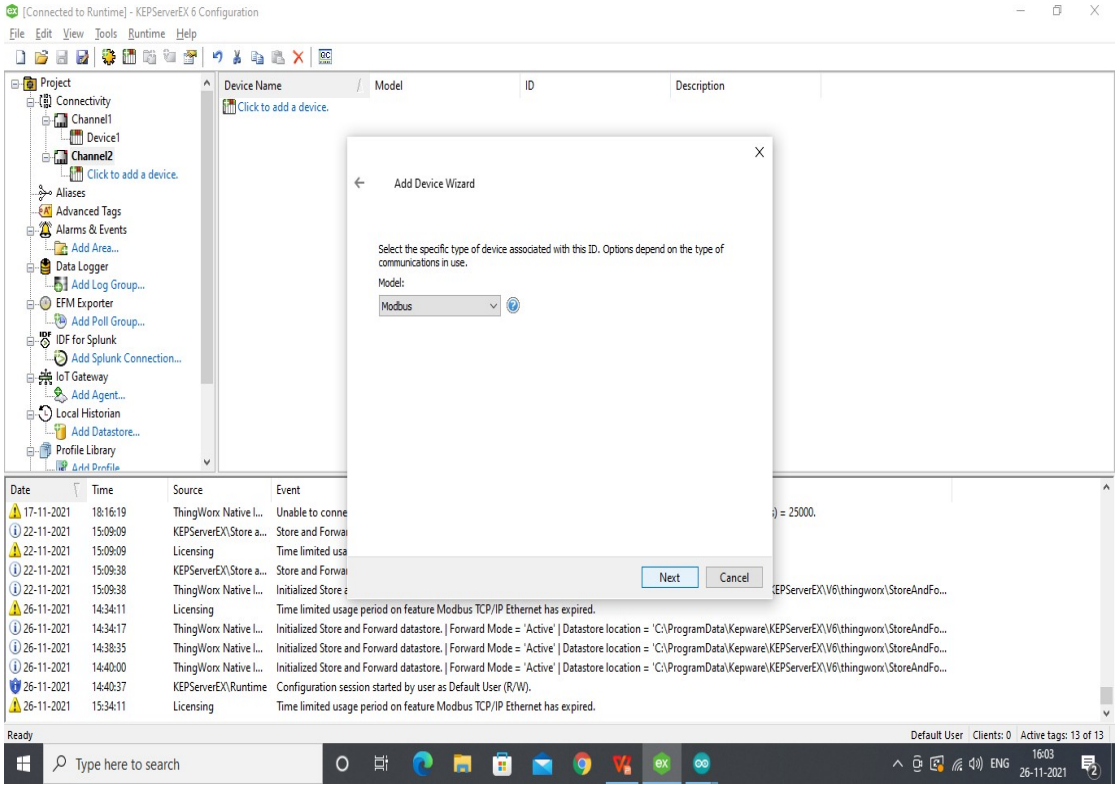
Click on add new device to create new device in the project



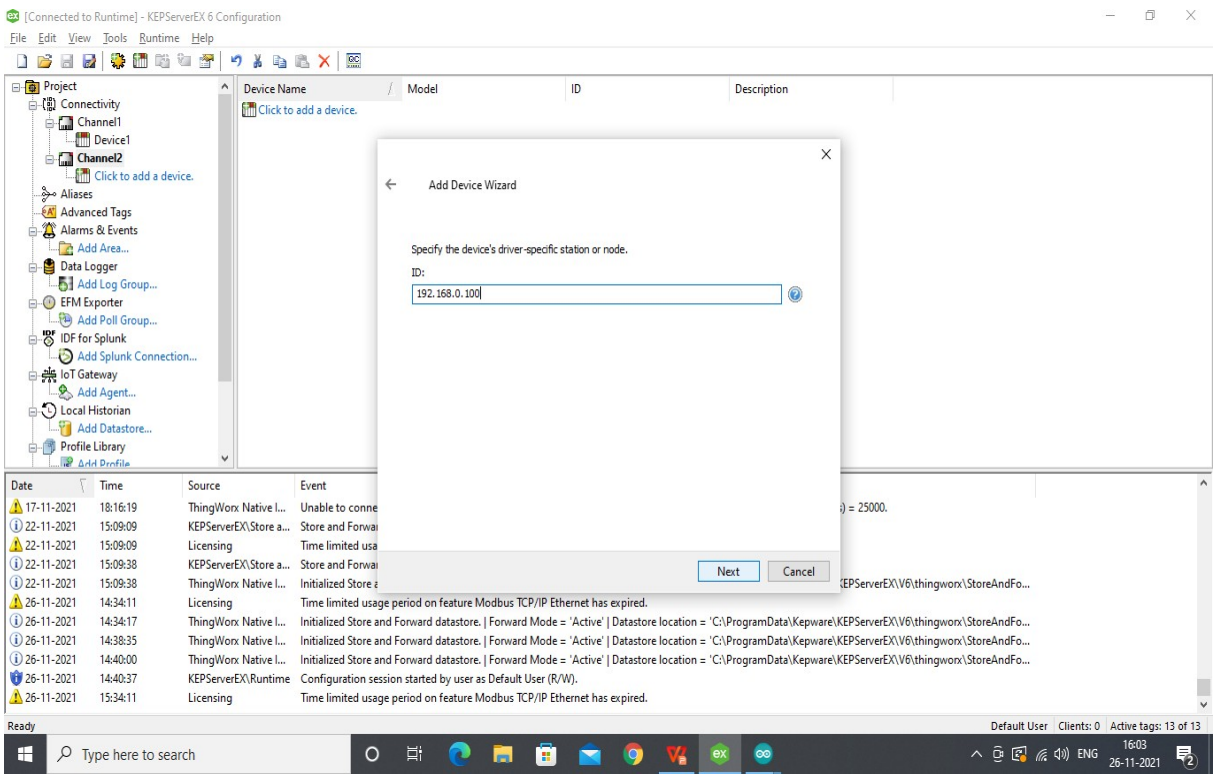
Give the device Name as Device 1



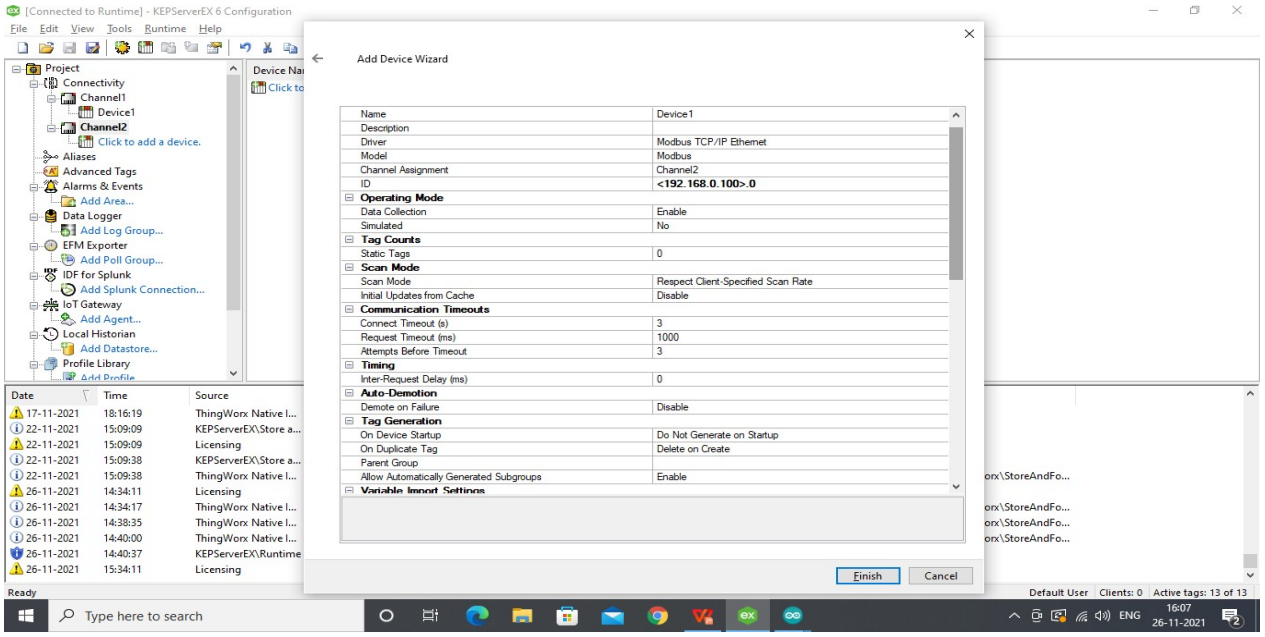
Select the type of the Mode is Modbus.



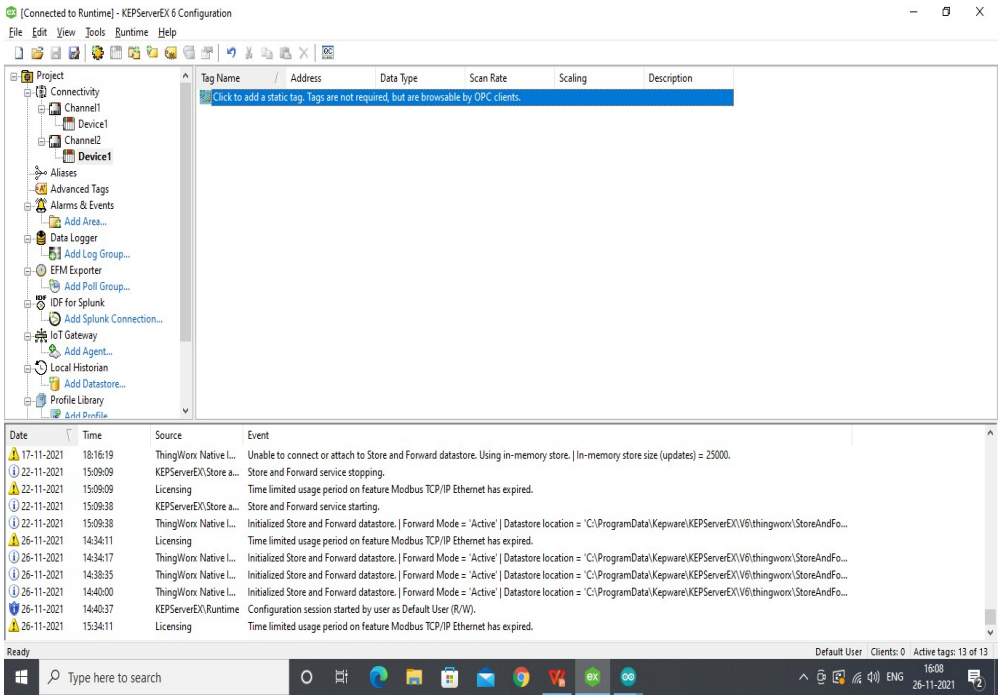
Write the IP address of NodeMCU then click next till Finished to set default setting.



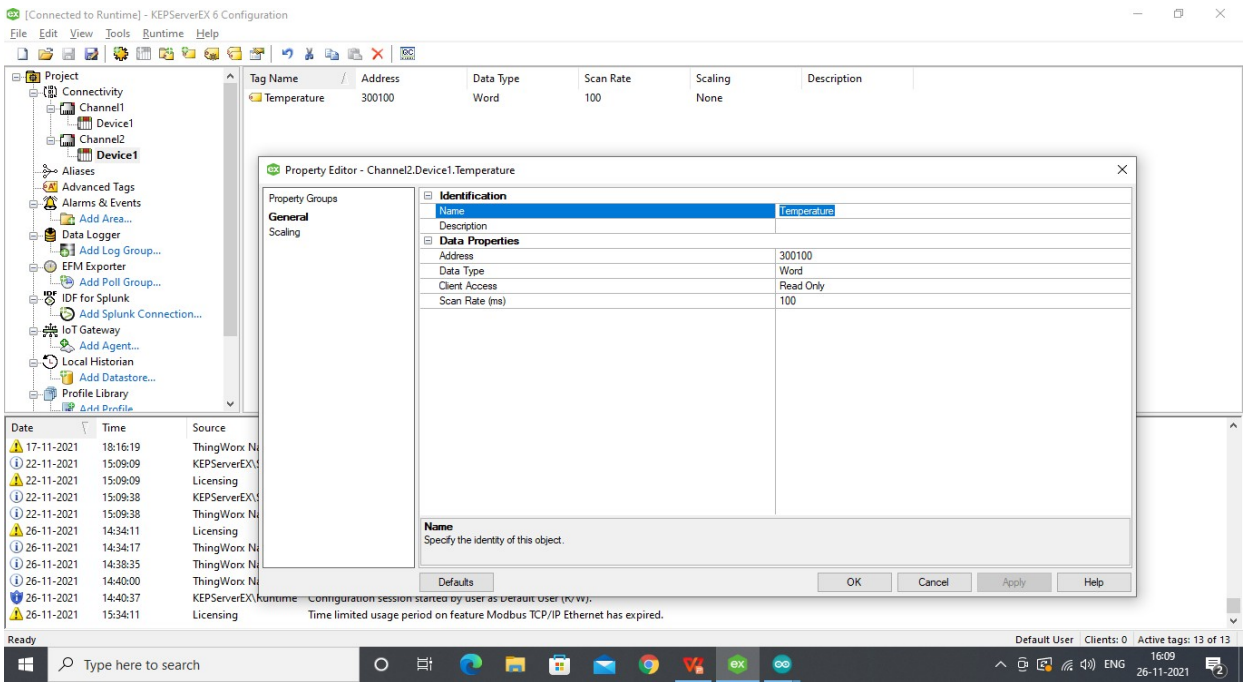
Check all the settings, check IP address of NodeMCU which is connected with CPU then click on finish.



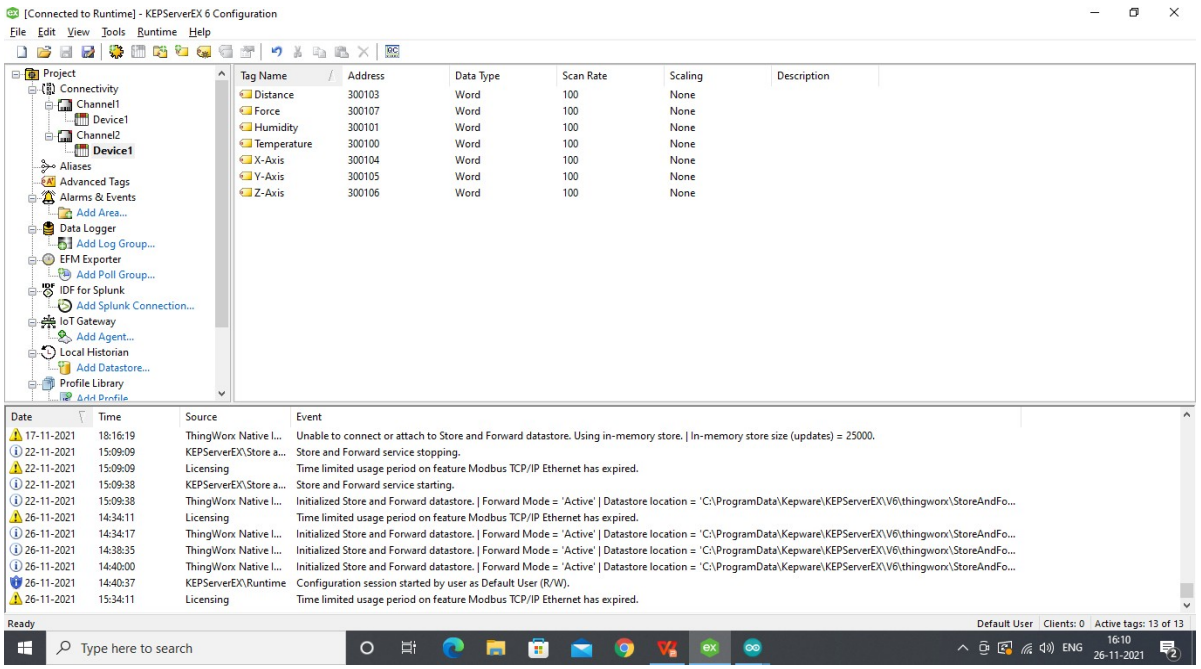
For creating new tag click on click to add static tag



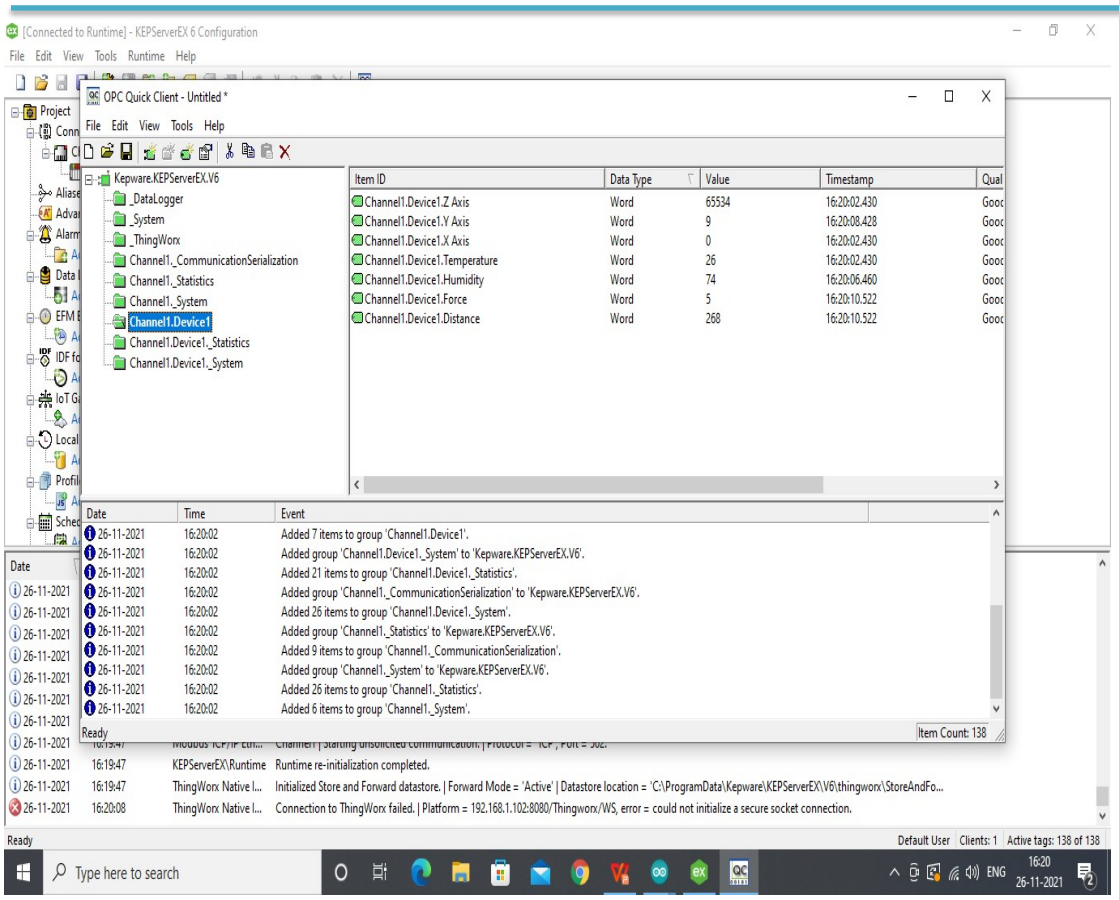
Give the name of the tag here I given name as temperature, assign address for temp. 300101



For creating New Tag Right Click on screen select New Tag then create all the tag as shown in screen shot for different sensor.



Click on OPC quick client for monitoring live values of all the sensor.


[Connected to Runtime] - KEPServerEX 6 Configuration
File Edit View Tools Runtime Help
OPC Quick Client - Untitled *
File Edit View Tools Help
Kepware.KEPServerEX.V6
_DataLogger
_System
_ThingWorx
Channel1._CommunicationSerialization
Channel1._Statistics
Channel1._System
Channel1.Device1
Channel1.Device1._Statistics
Channel1.Device1._System
Item ID
Data Type
Value
Timestamp
Channel1.Device1.Z Axis Word 65534 16:20:02.430
Channel1.Device1.Y Axis Word 9 16:20:08.428
Channel1.Device1.X Axis Word 0 16:20:02.430
Channel1.Device1.Temperature Word 26 16:20:02.430
Channel1.Device1.Humidity Word 74 16:20:06.460
Channel1.Device1.Force Word 5 16:20:10.522
Channel1.Device1.Distance Word 268 16:20:10.522
Date Time Event
26-11-2021 16:20:02 Added 7 items to group 'Channel1.Device1'.
26-11-2021 16:20:02 Added group 'Channel1.Device1._System' to 'Kepware.KEPServerEX.V6'.
26-11-2021 16:20:02 Added 21 items to group 'Channel1.Device1._Statistics'.
26-11-2021 16:20:02 Added group 'Channel1._CommunicationSerialization' to 'Kepware.KEPServerEX.V6'.
26-11-2021 16:20:02 Added 26 items to group 'Channel1.Device1._System'.
26-11-2021 16:20:02 Added group 'Channel1._Statistics' to 'Kepware.KEPServerEX.V6'.
26-11-2021 16:20:02 Added 9 items to group 'Channel1._CommunicationSerialization'.
26-11-2021 16:20:02 Added group 'Channel1._System' to 'Kepware.KEPServerEX.V6'.
26-11-2021 16:20:02 Added 26 items to group 'Channel1._Statistics'.
26-11-2021 16:20:02 Added 6 items to group 'Channel1._System'.
Ready
Item Count: 138
26-11-2021 16:19:47 KEPServerEX\Runtime Runtime re-initialization completed.
26-11-2021 16:19:47 ThingWorx Native I... Initialized Store and Forward datastore. | Forward Mode = 'Active' | Datastore location = 'C:\ProgramData\Kepware\KEPServerEX\V6\thingworx\StoreAndFo...
26-11-2021 16:20:08 ThingWorx Native I... Connection to ThingWorx failed. | Platform = 192.168.1.102:8080/Thingworx/WS, error = could not initialize a secure socket connection.
Ready
Default User Clients: 1 Active tags: 138 of 138
Type here to search
ENG
16:20
26-11-2021

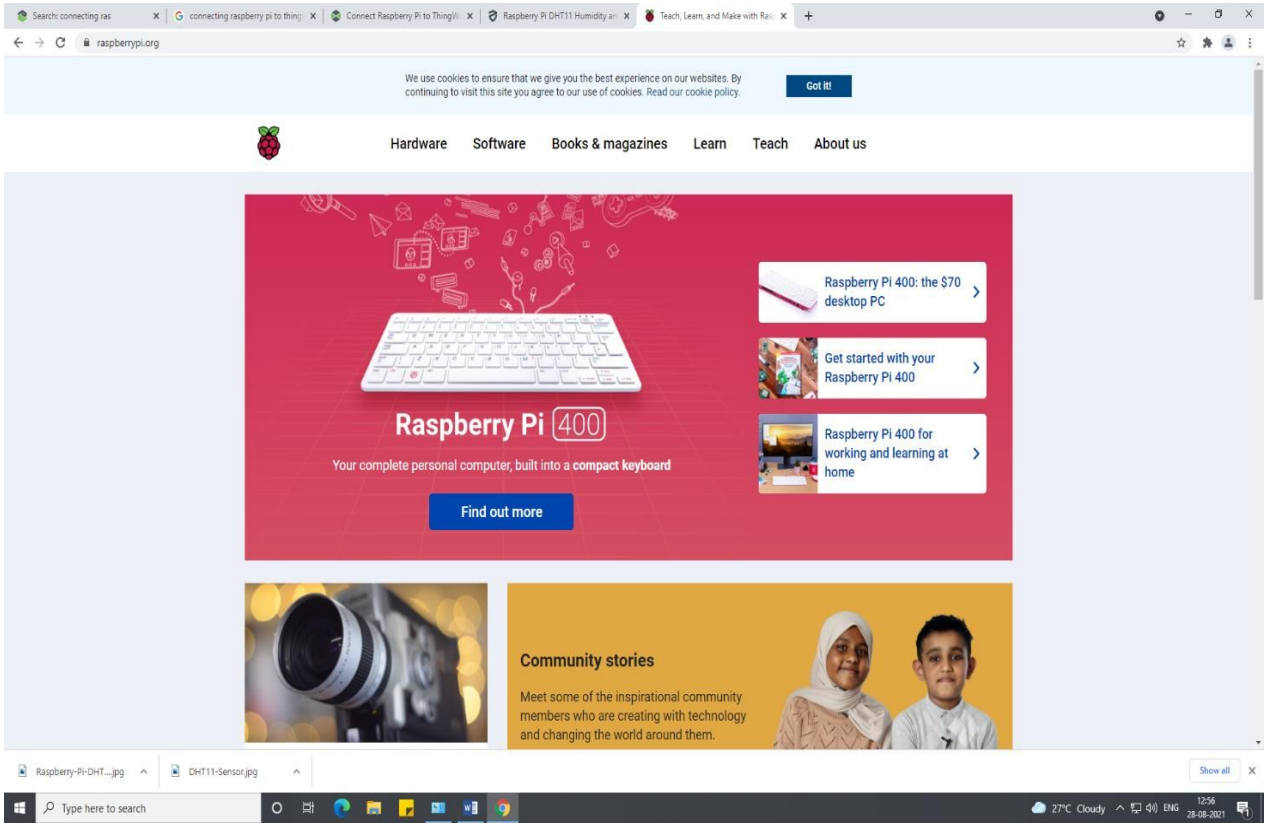
Installing Raspberry Pi OS

Components Required

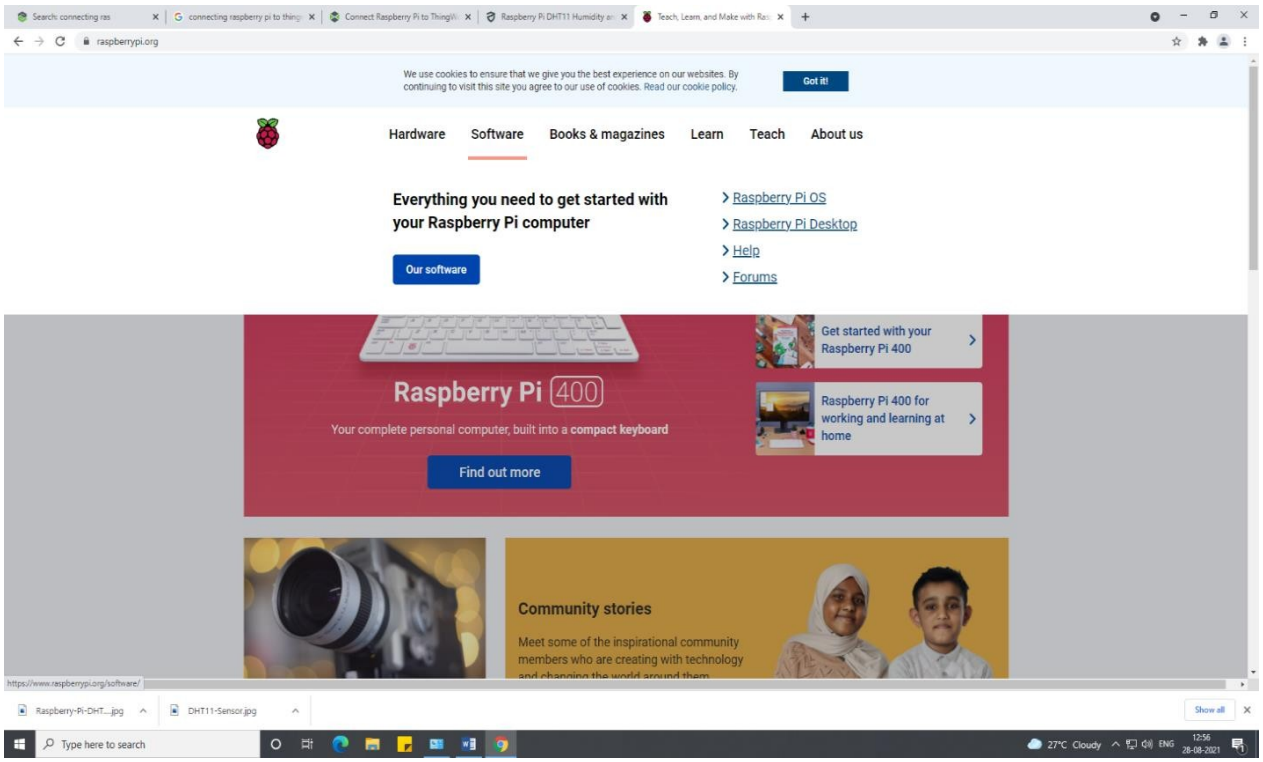
- Raspberry Pi
- Memory card
- Computer

Follow the following steps for OS installation

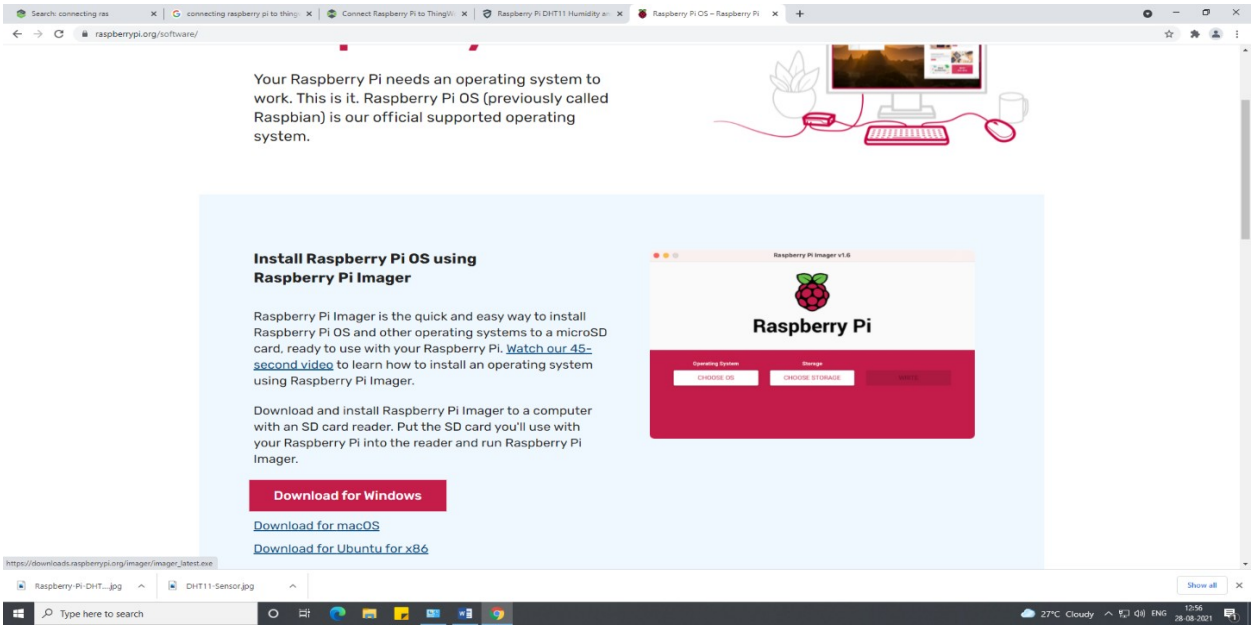
- Visit https://www.raspberrypi.org/



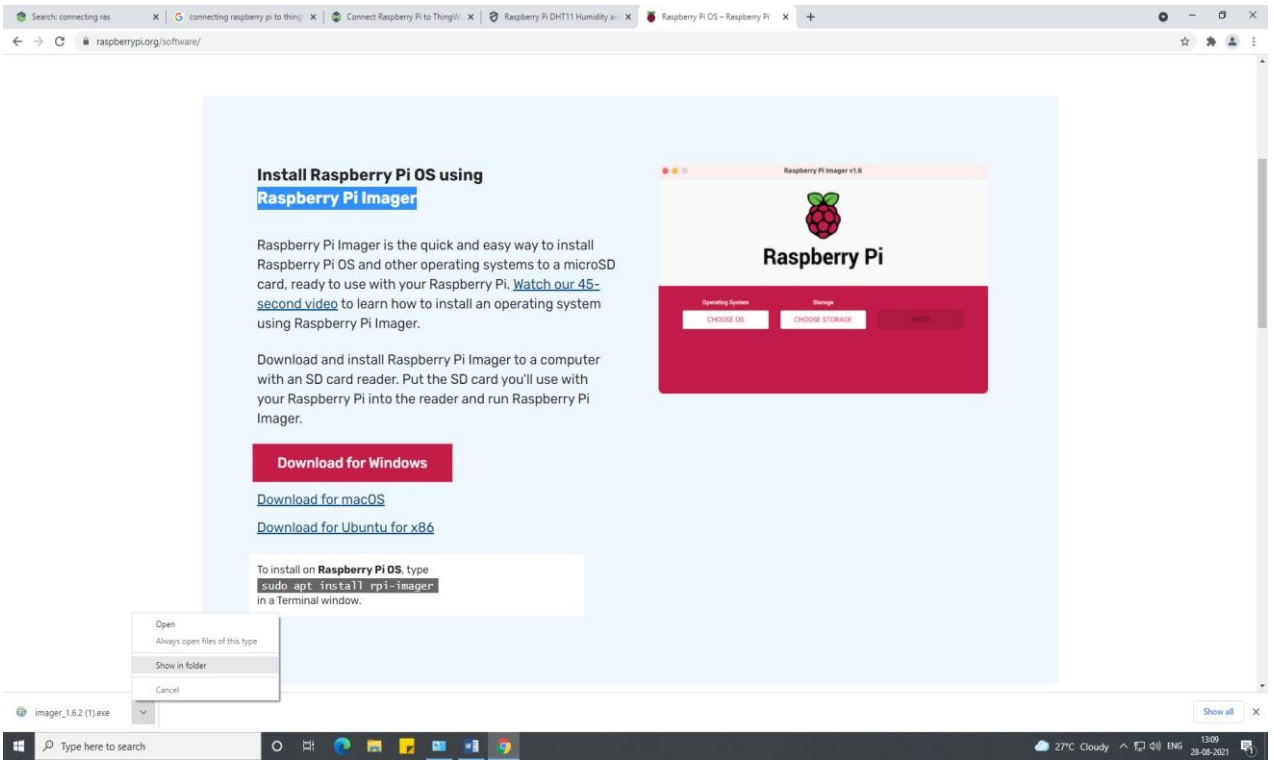
- Go to software



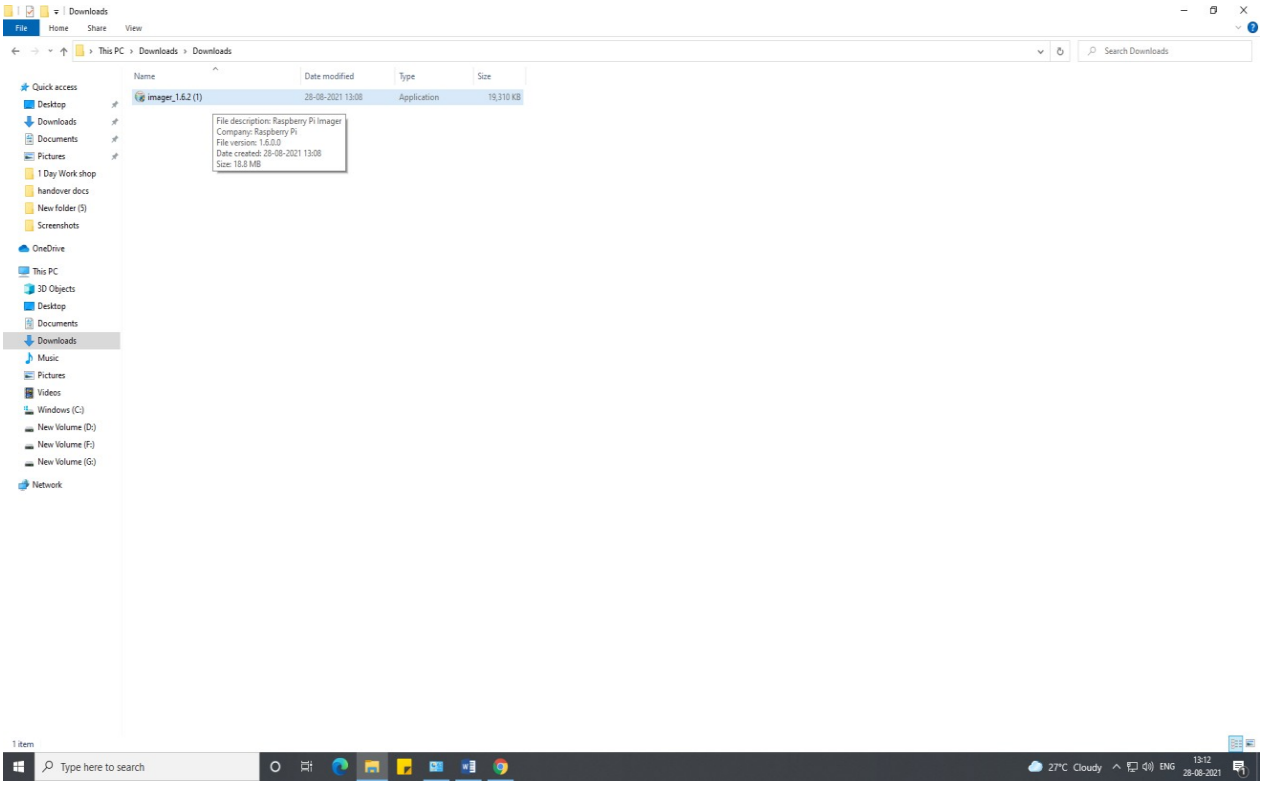
- Download Raspberry Pi Imager by selecting your operating system.



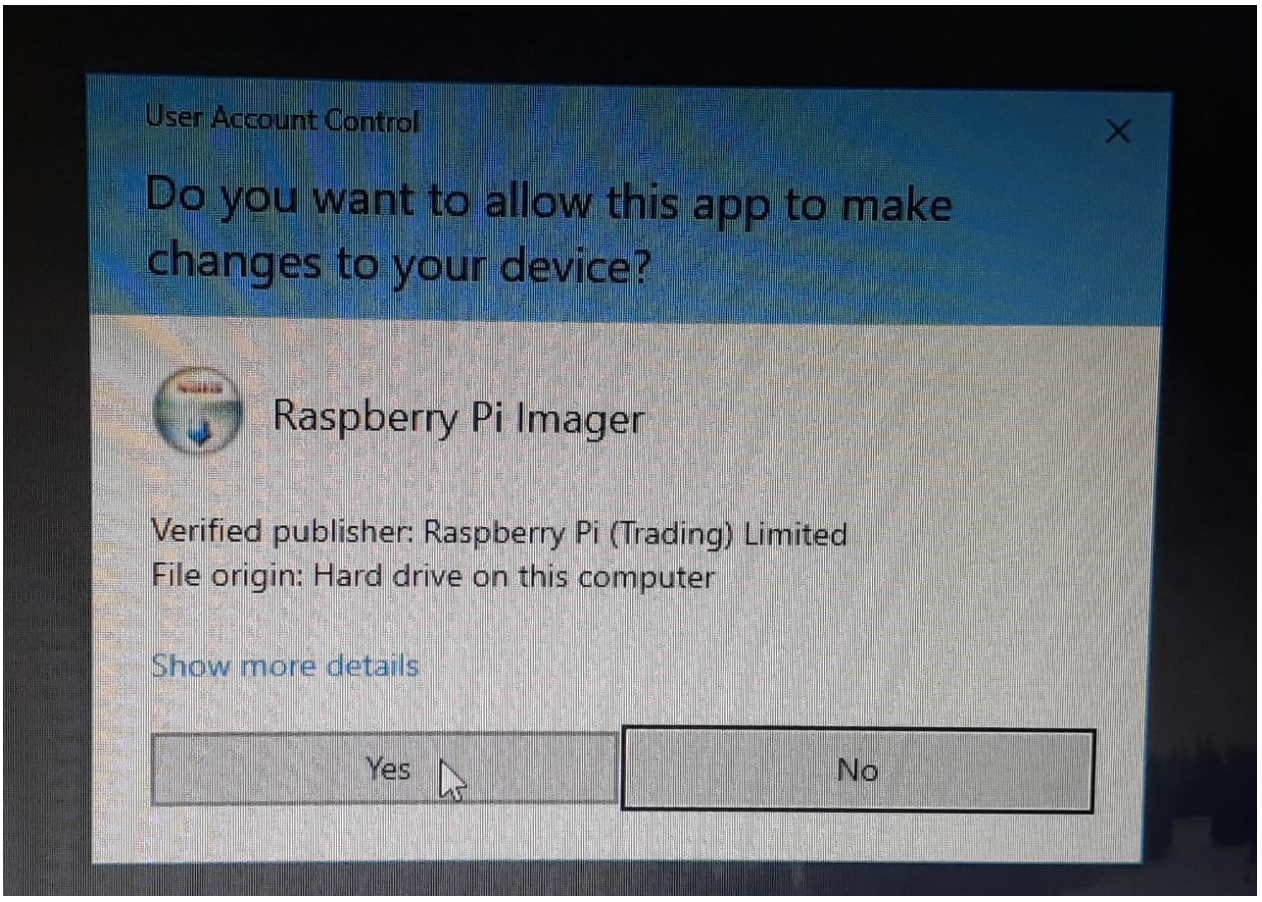
- Open the downloaded file in the folder.



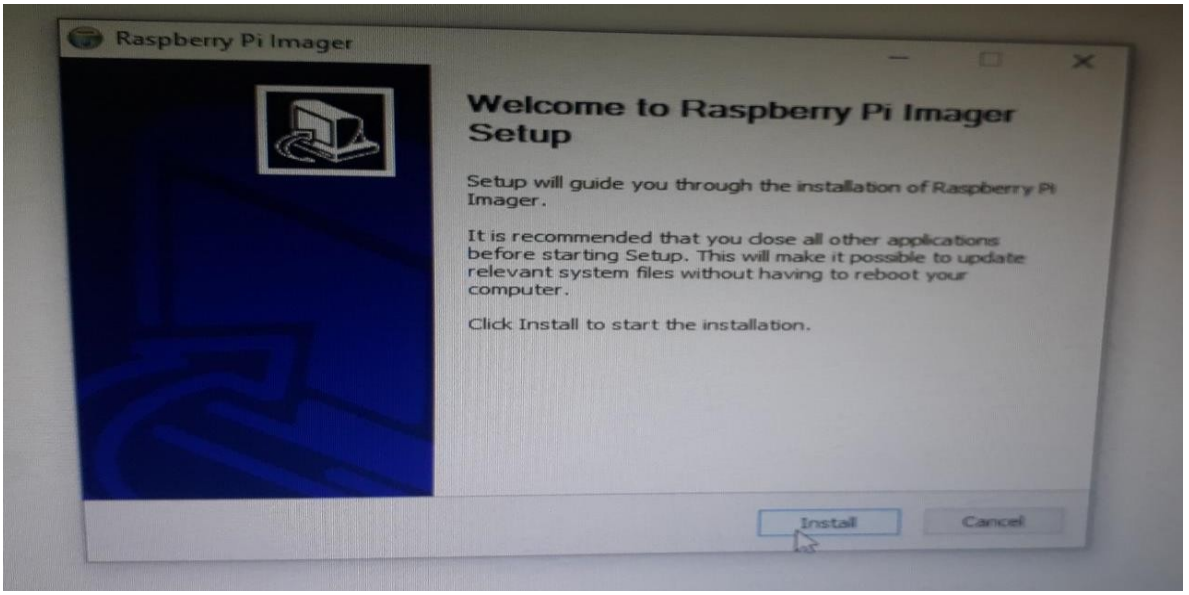
- Double click on the Downloaded file



- Then you will get a pop-up screen as shown below select yes

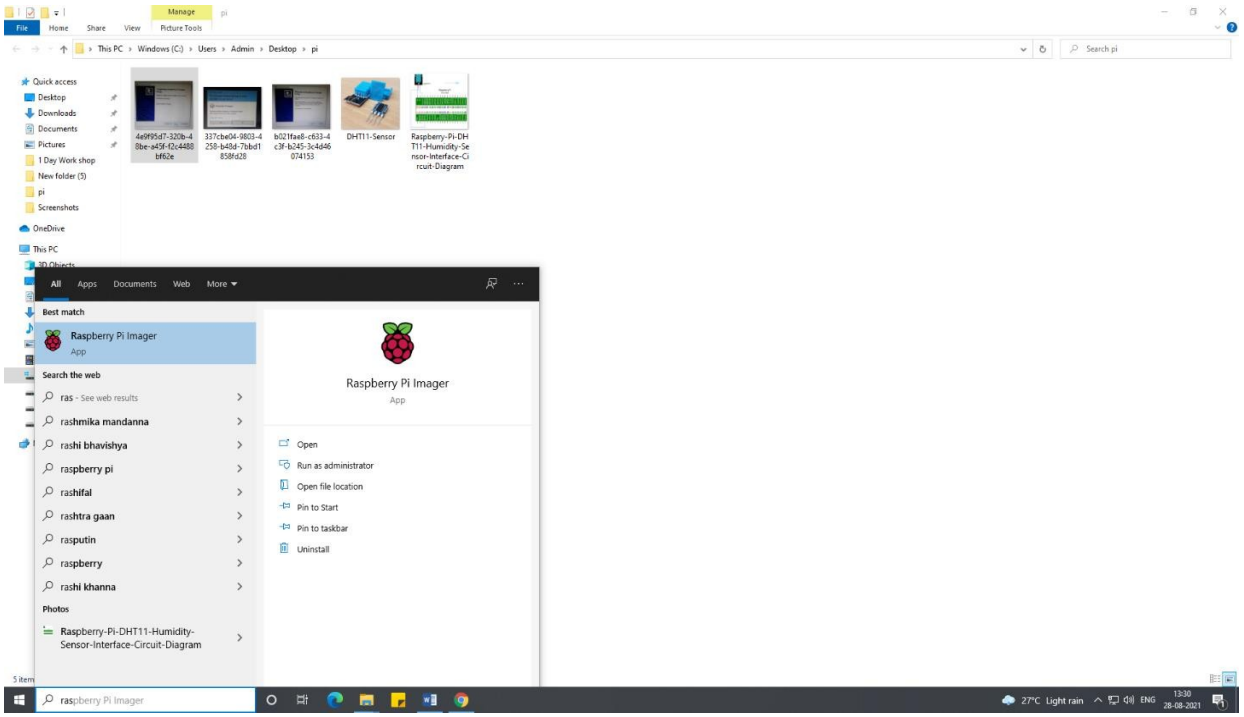


- Then u will redirect to set up as below select install

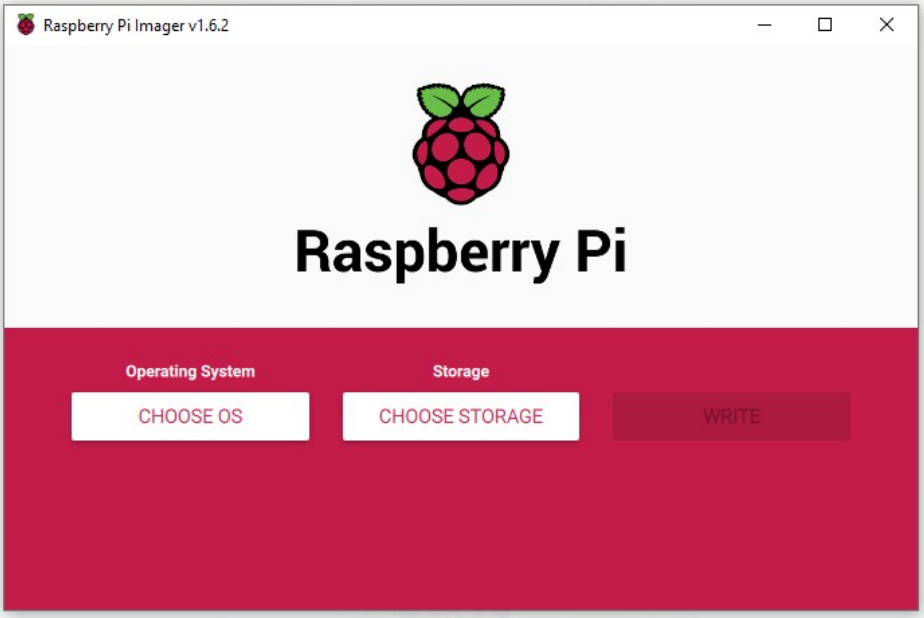


- After a few moments then you will get a pop screen as shown below select finish

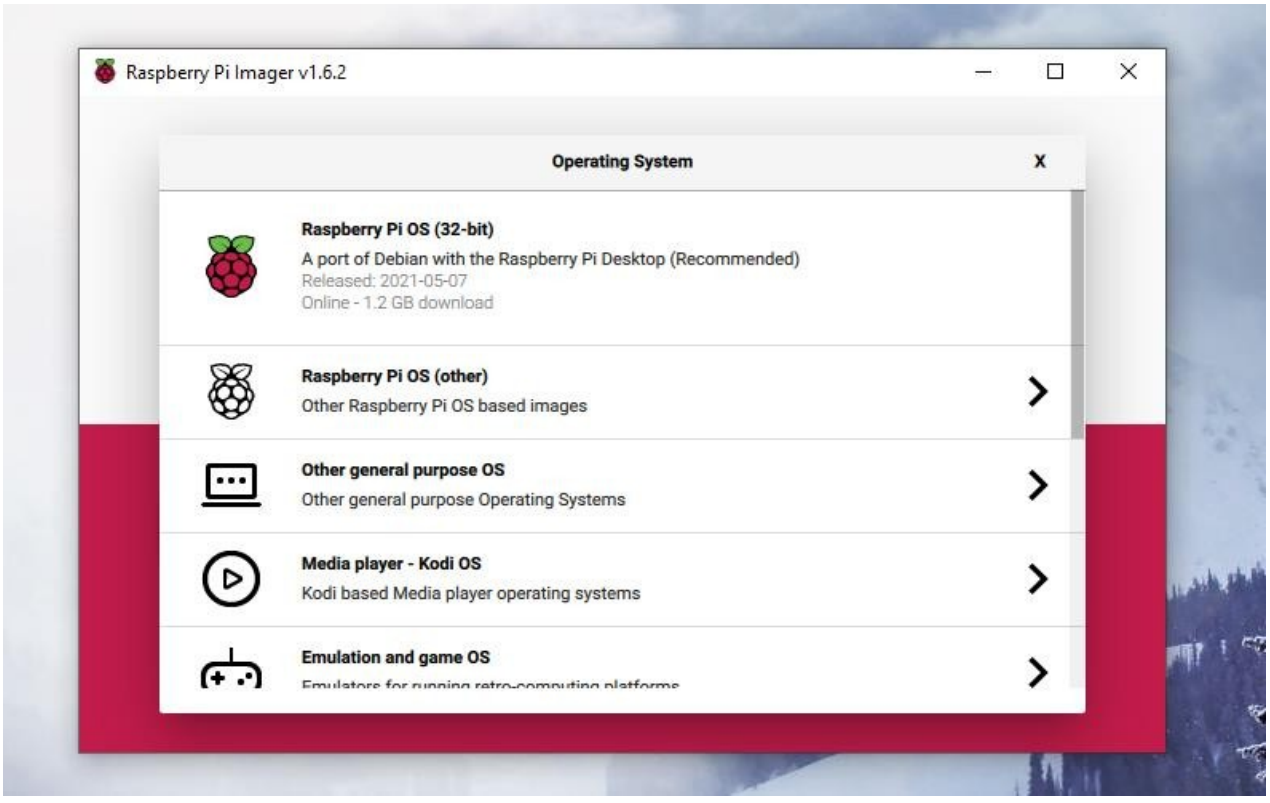
- Click on windows search raspberry pi imager you will get a screen as below select that one



- Raspberry pi imager App screen as below.

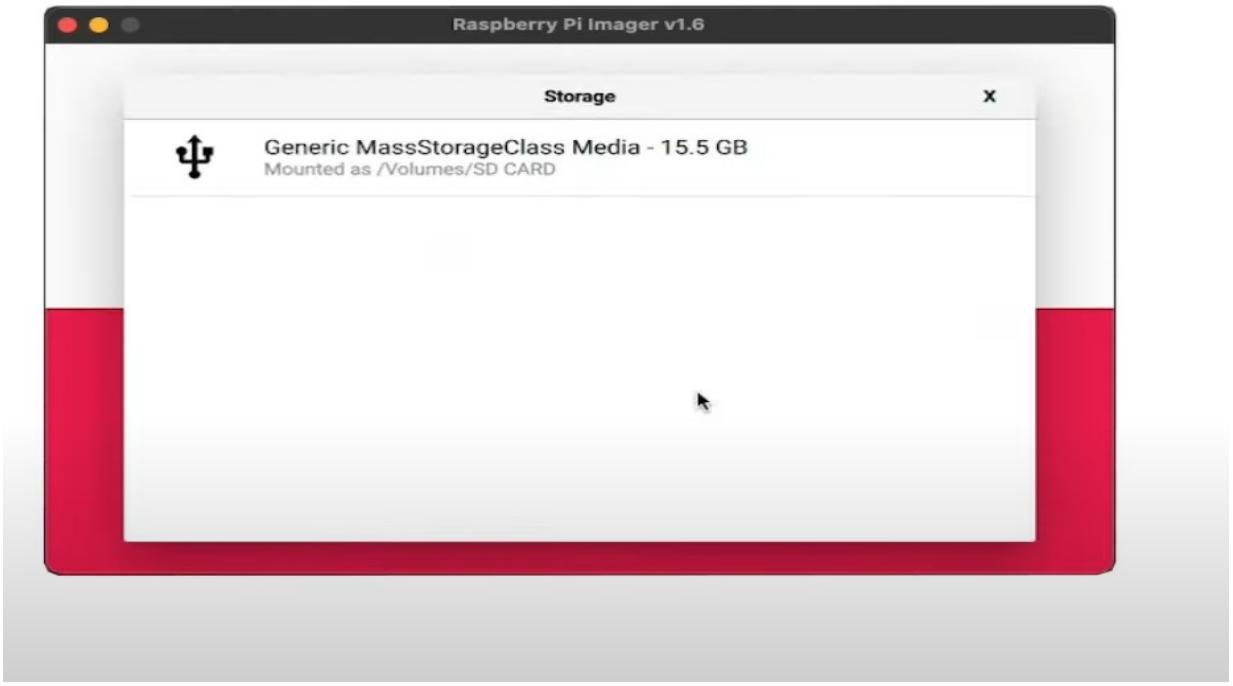


- Choose OS updated edition and concerning your hardware. (in particular, in our session as of now I am selecting raspberry pi 32 bits.

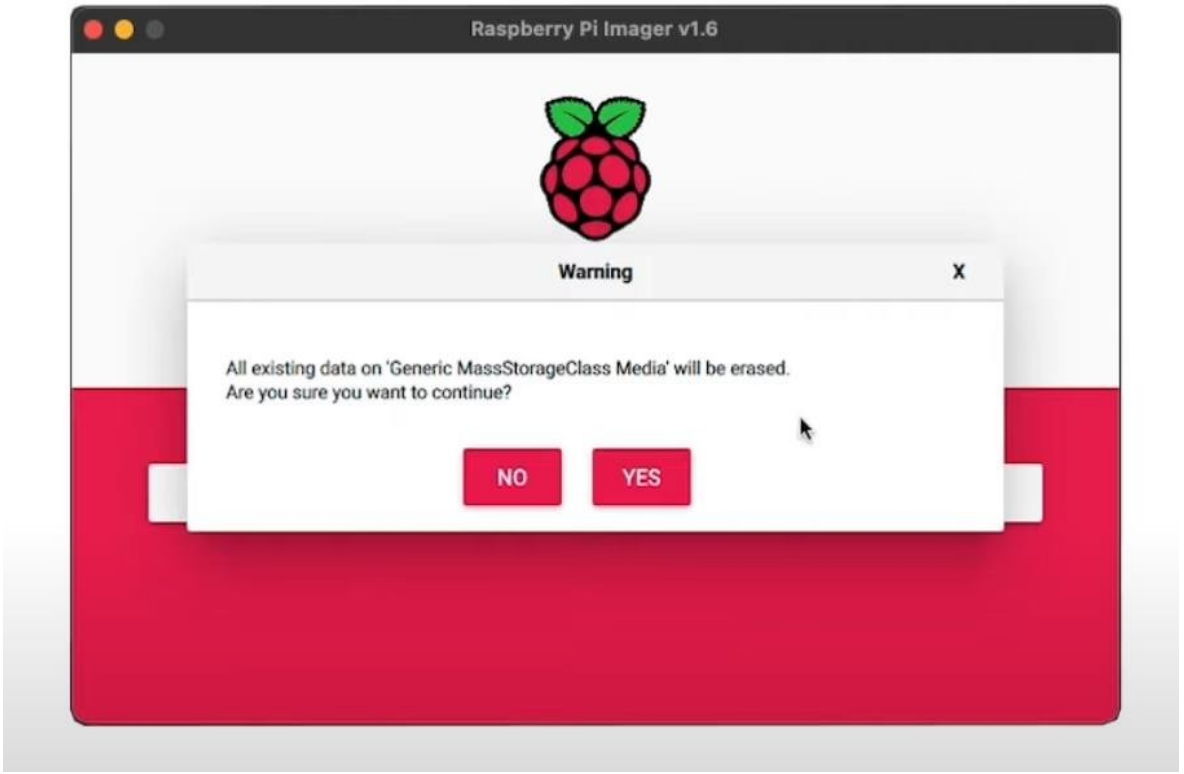


- Insert a Formatted memory card to the CPU of the system/Laptop using a standard card reader.

- Select your memory card in the raspberry pi imager app.

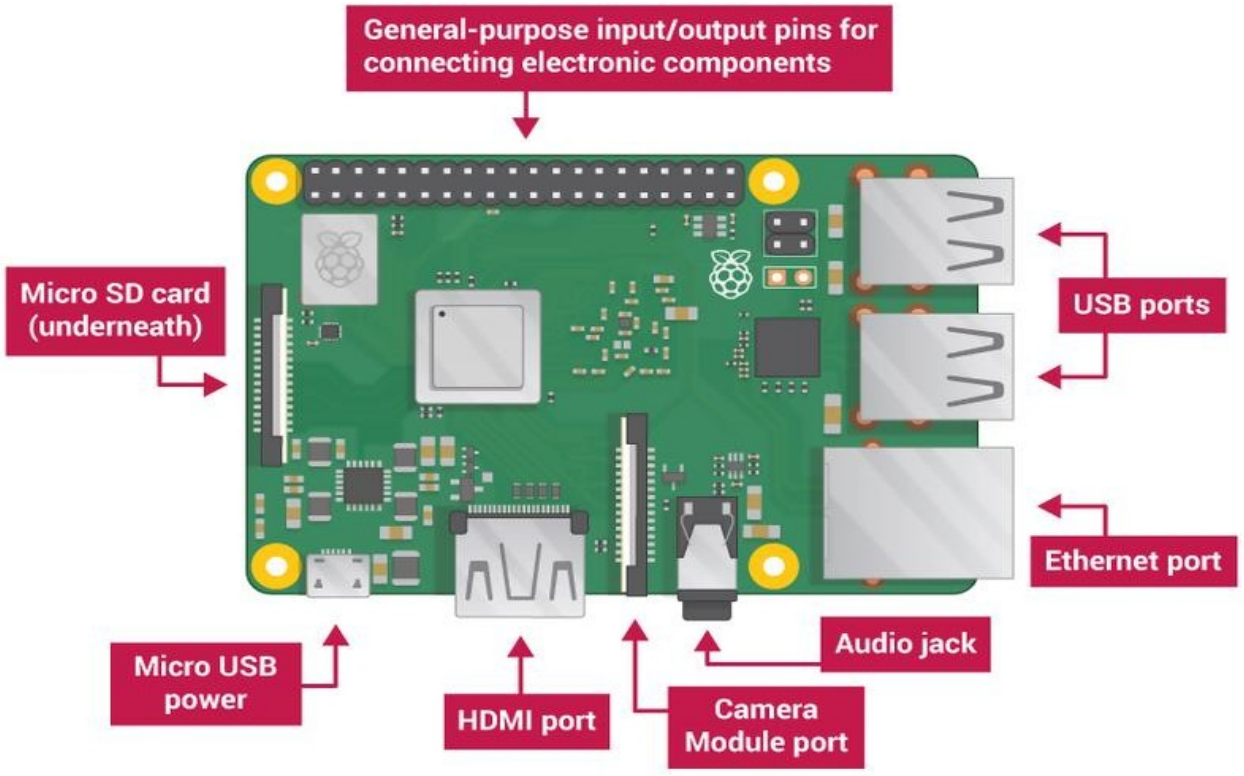


- After selecting a memory card Click on write you will get a pop-up screen as shown below click on yes.



- After OS installation is complete you will get instructions to remove the memory card as shown below click on continue and remove the SD card.

- Raspberry pi module ports information for setting up.

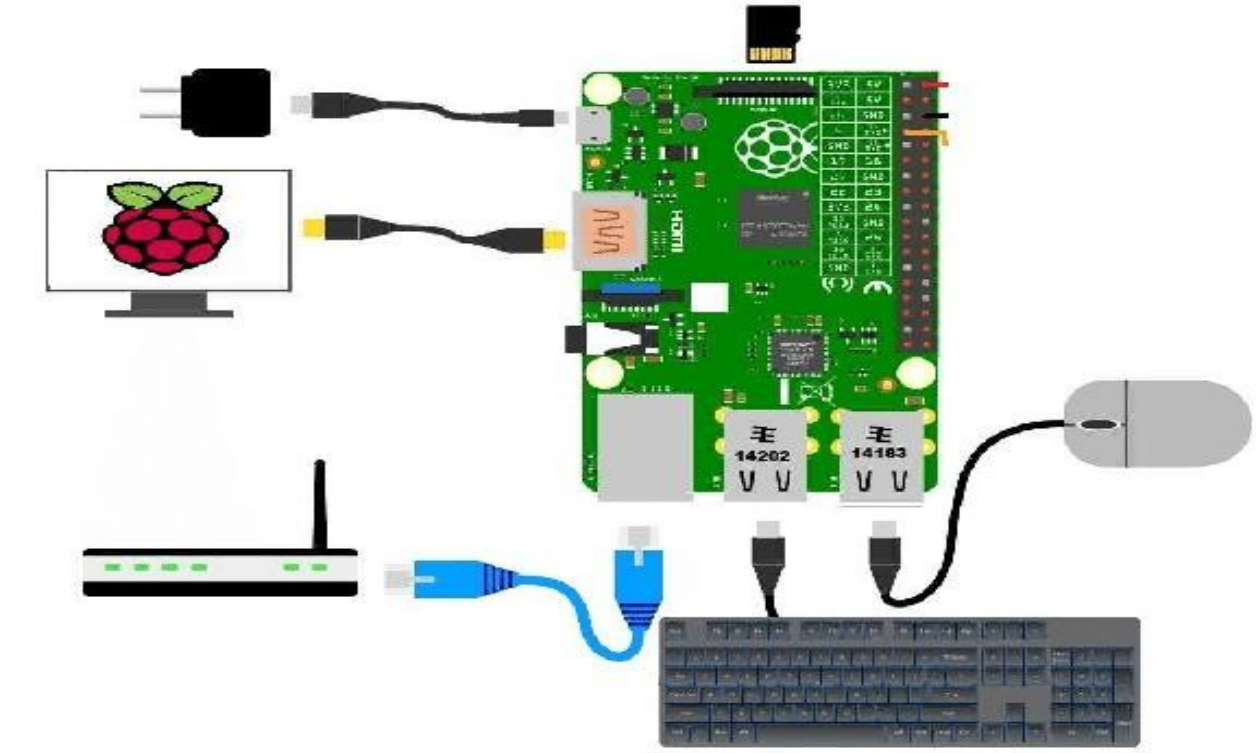


- Insert OS installed memory card into the raspberry pi module as shown below.

- Give Proper power supply to raspberry pi module (refer to the documentation provided with module)

Plug-in power supply connector to its rightful place in the raspberry pi as shown below.

- Make all other port connections as per below. Connect HDMI/micro-HDMI cable to Monitor as monitor shown below.

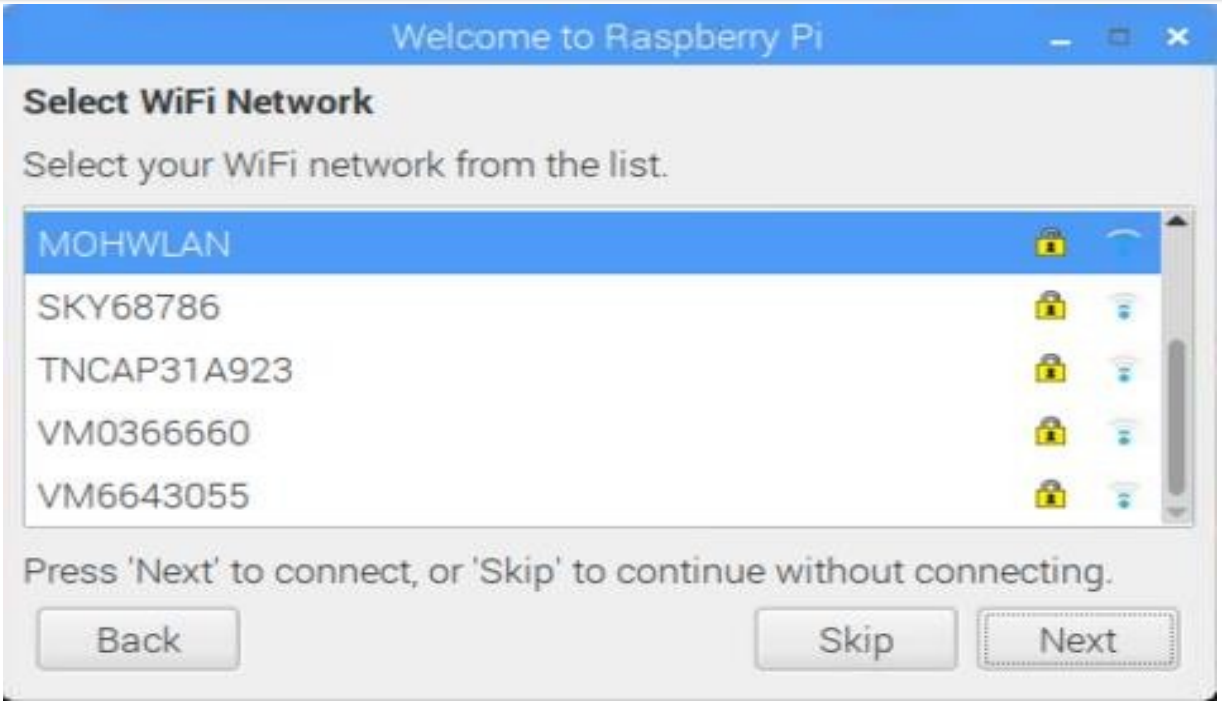


- Power up supply to raspberry you will see a welcome screen asking you to provide some basic settings. Choose your Country, Language, and Time zone. You can also set whether you are using a US keyboard layout.

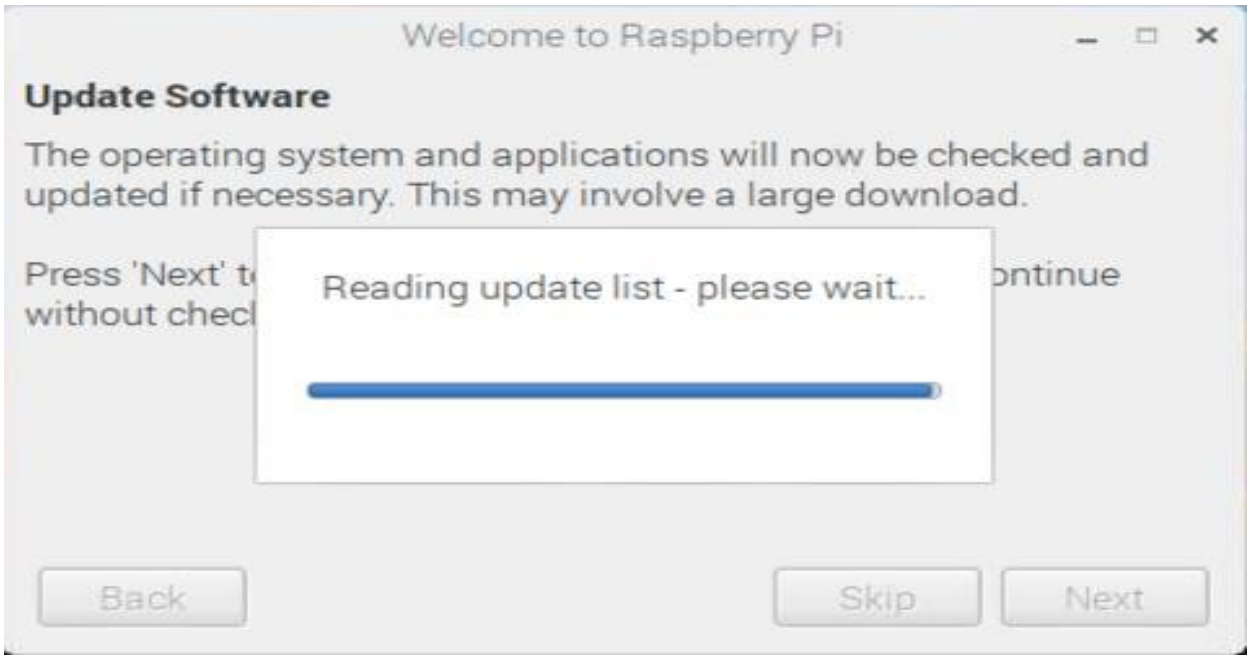
Click on next.

- For easy setup of multiple person usages, it's recommended to not provide username and password. just click on next

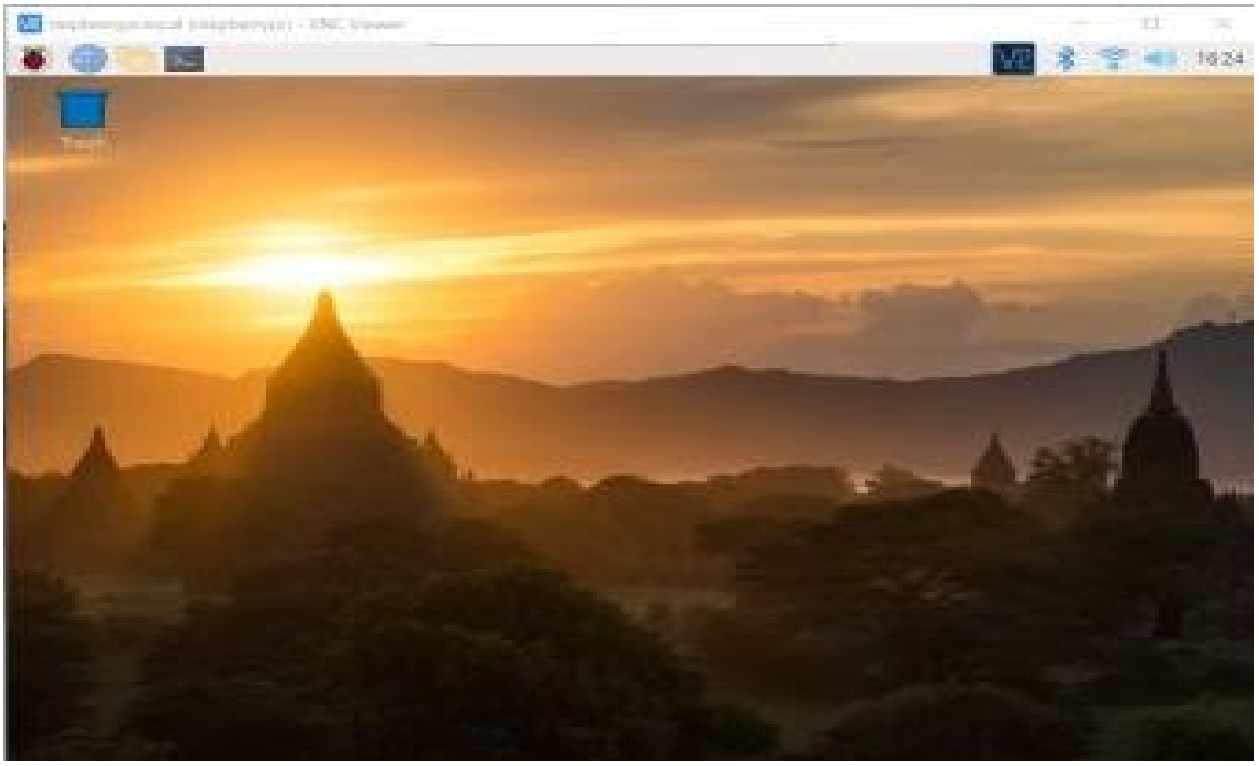
- For internet connectivity, if you connected LAN then skip this one, otherwise connect to Wi-Fi as usual.



- The Raspberry Pi will now check for any software updates, and download newer versions if available The Raspberry Pi might prompt you to reboot it after this stage, to complete an update, it installed. Skip the update session because we are installed the latest version of the software using pi imager.



- After settings, you will get the following screen.



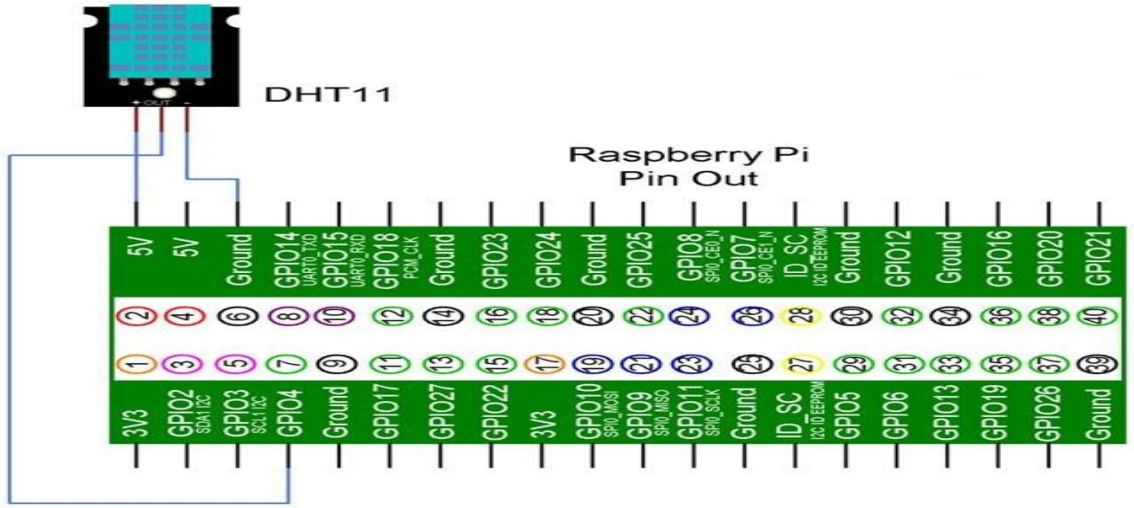
OS installation and Raspberry pi set up are completed.

Connecting DHT 11 sensor Raspberry Pi and Thingworx

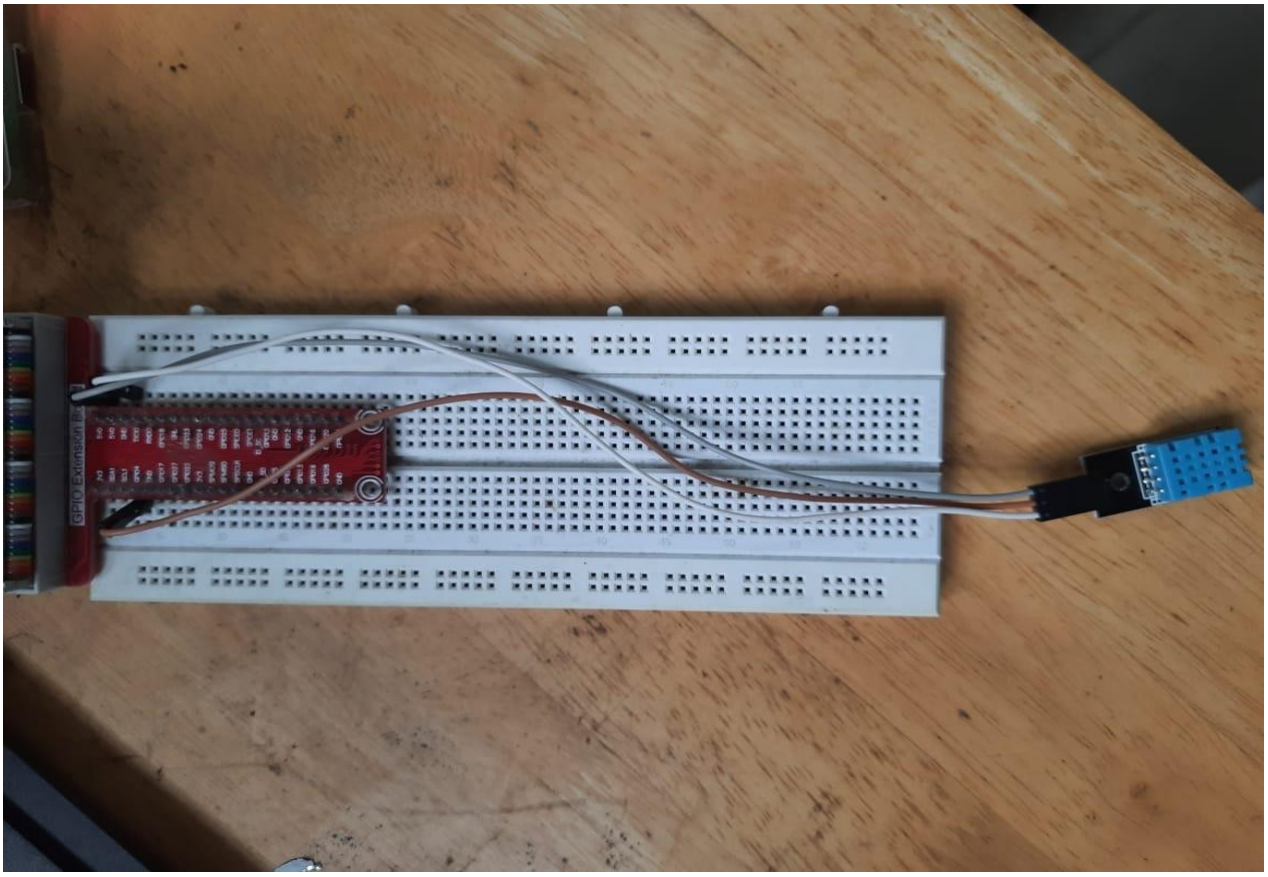
- DHT 11 sensor has 3 Connecting pins they are as follows

Ground
Positive
Output

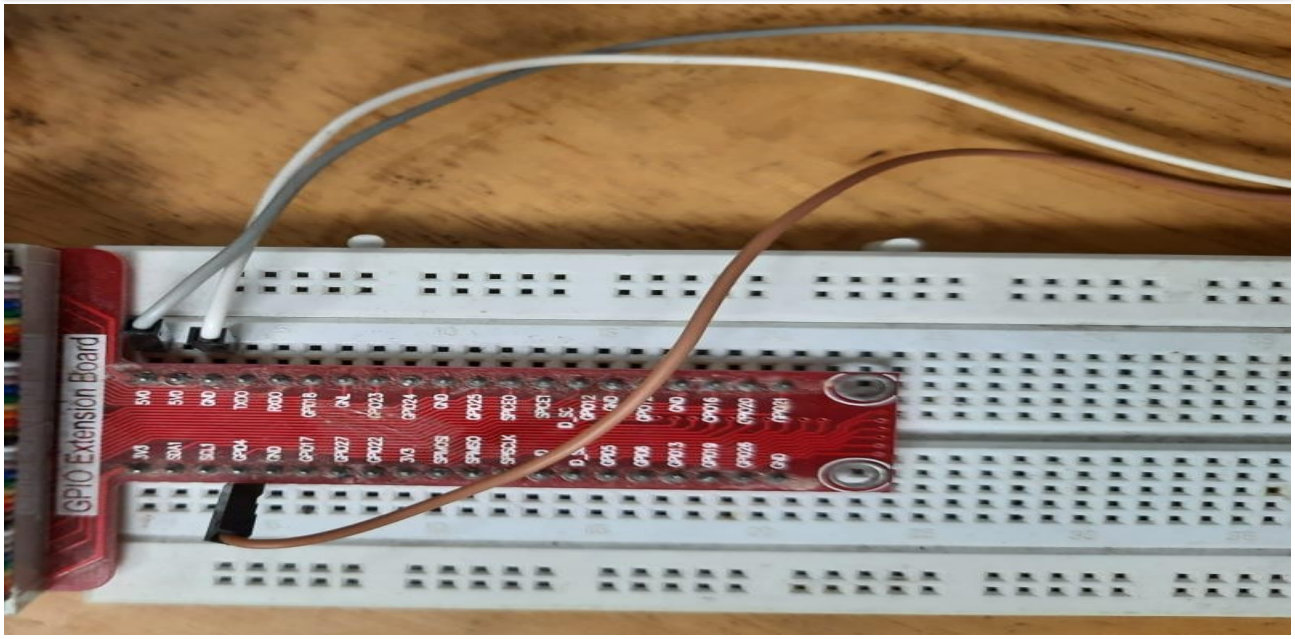
- Make connection as shown below



- Connection

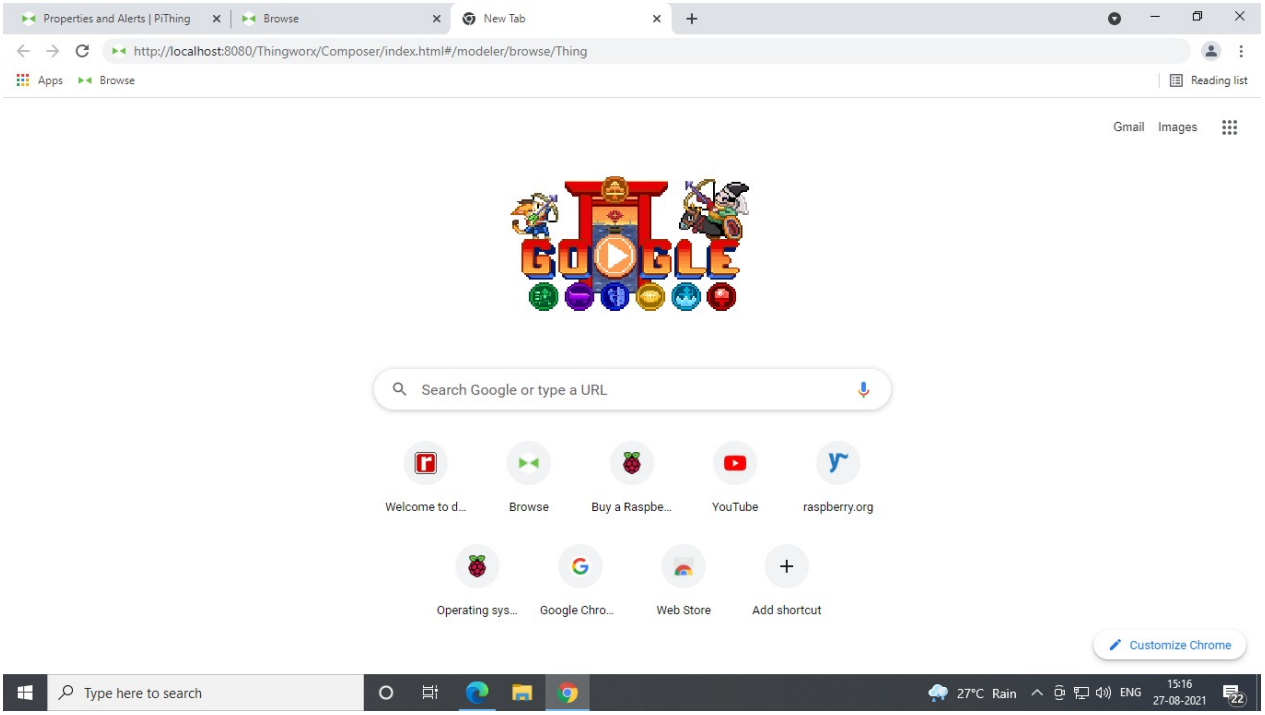


- GPIO Board

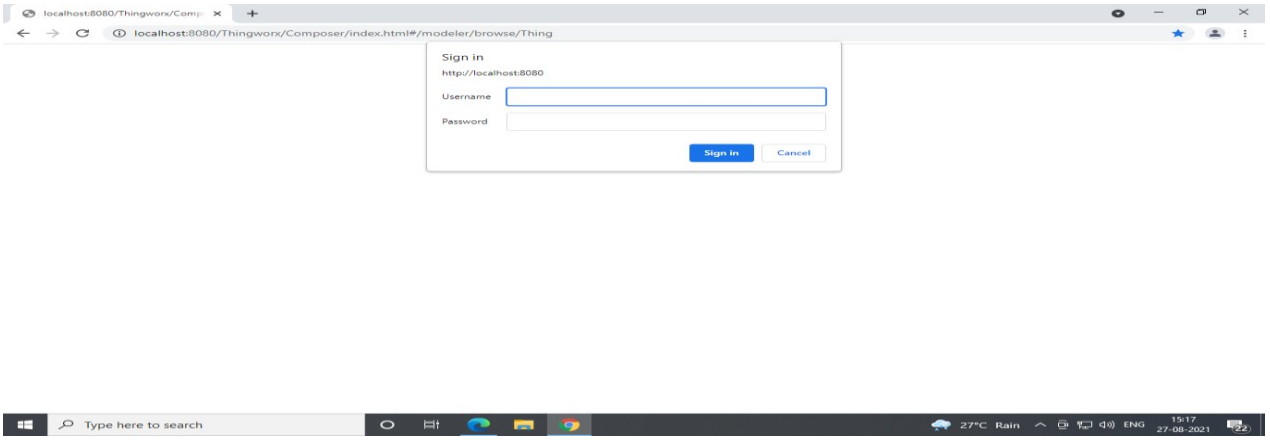


Connecting Raspberry Pi to ThingWorx

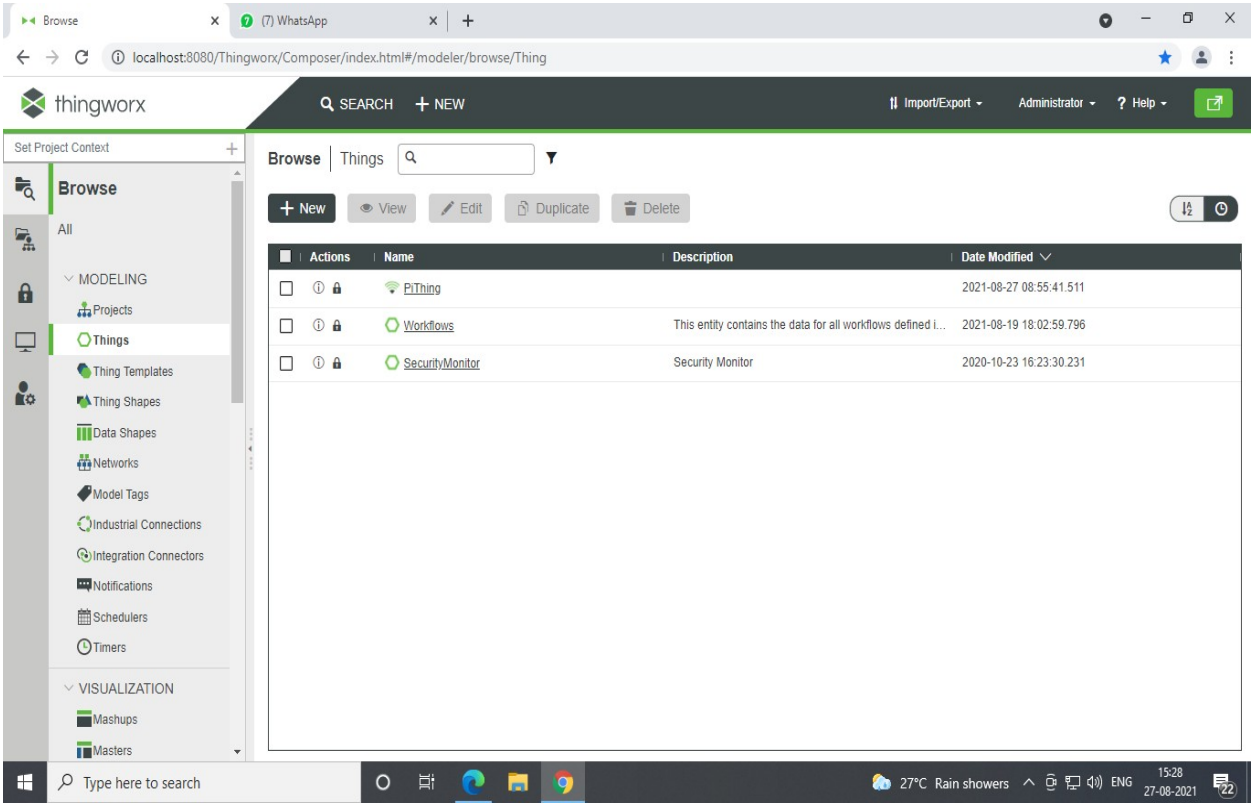
- Go to google and search http://localhost:8080/Thingsworx



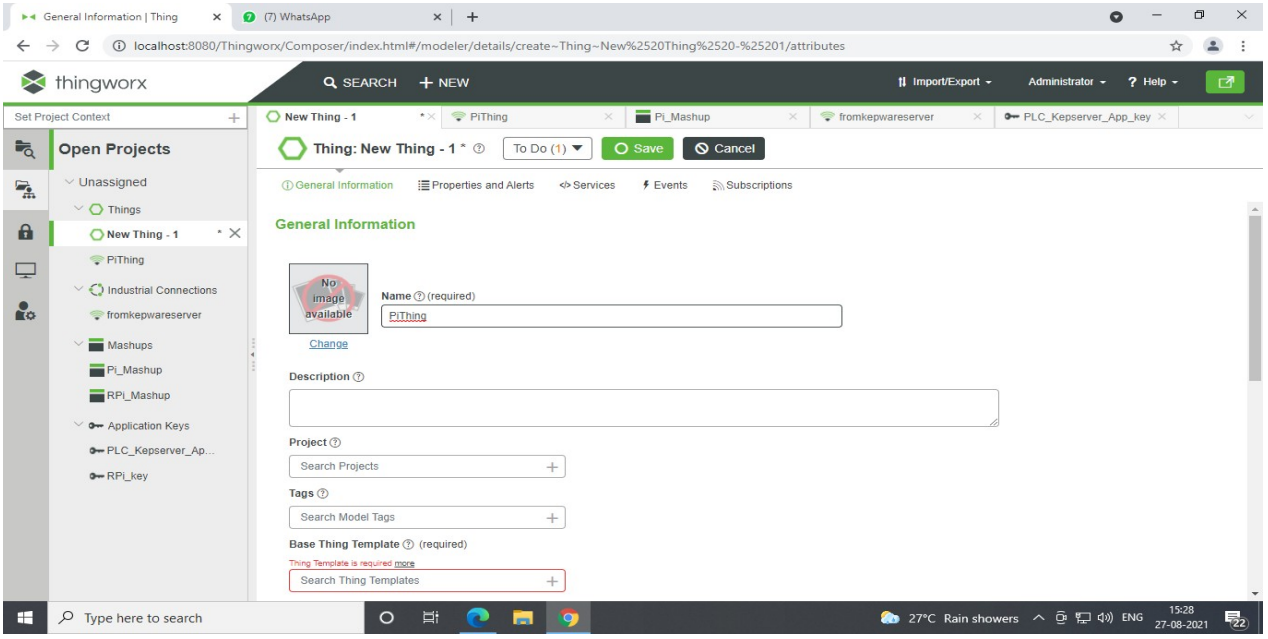
- Provide username and password and sign in

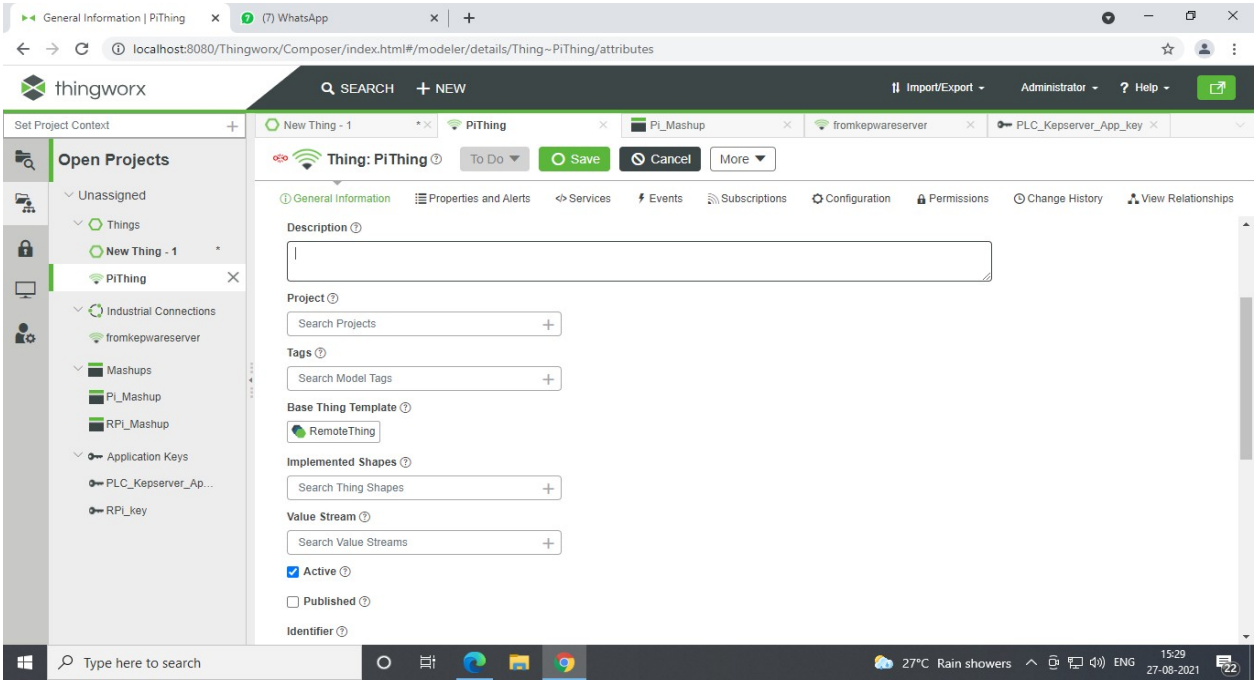


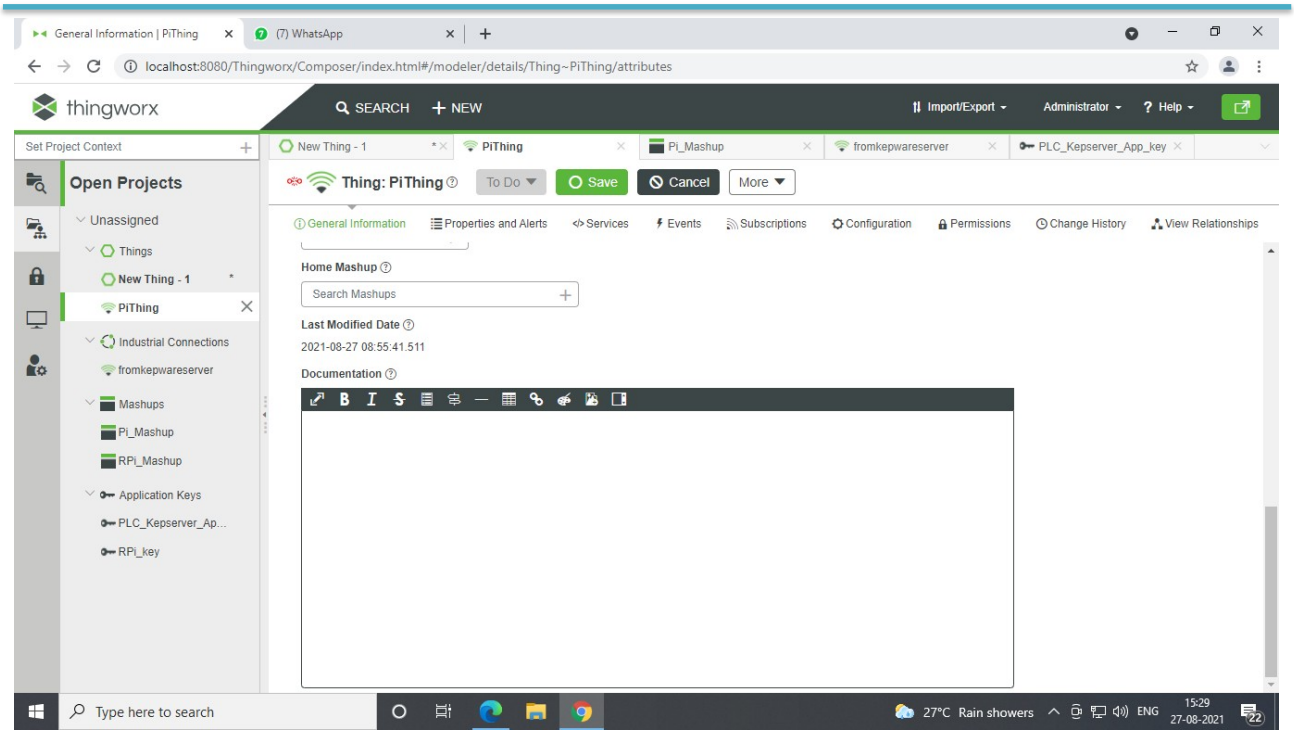
- Create things by following the steps
  Go to things
- Click on the new thing



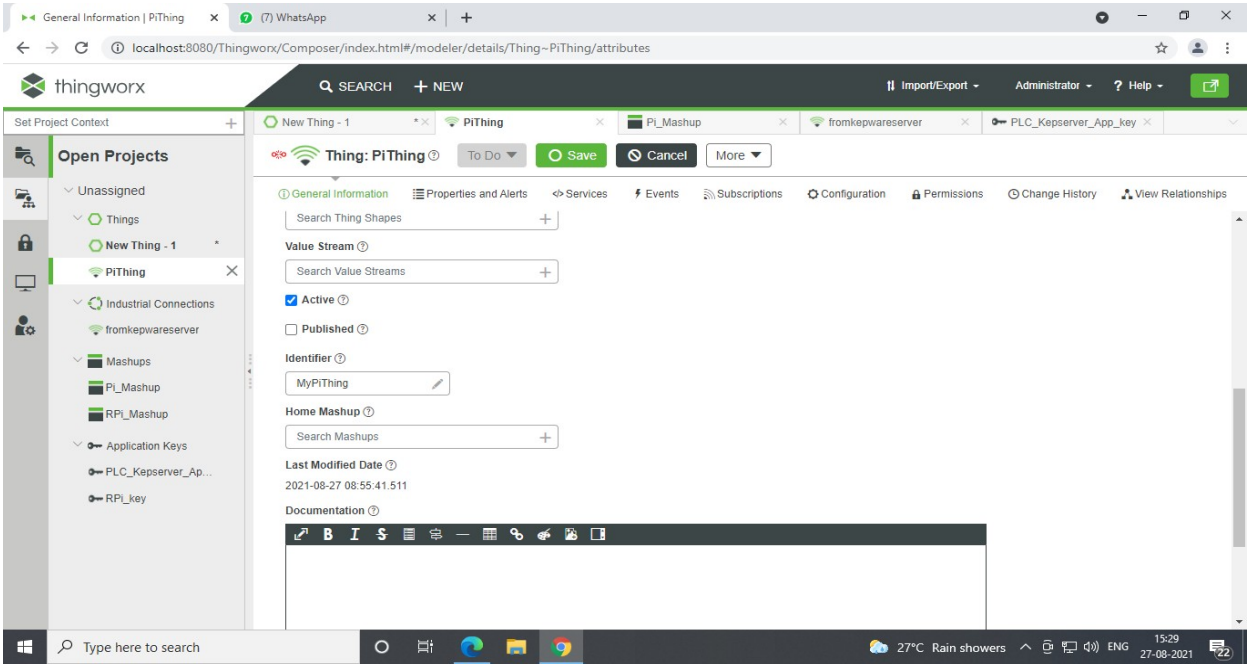
- Provide thing name and basic details as below and click on save.



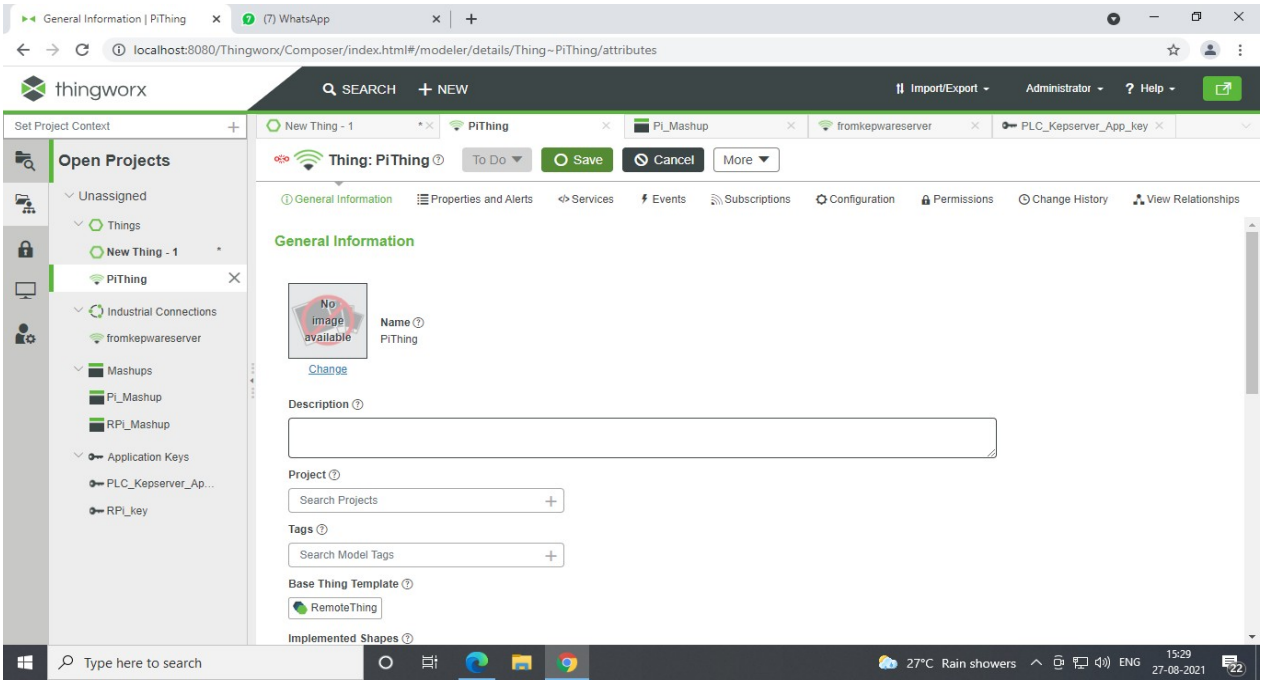




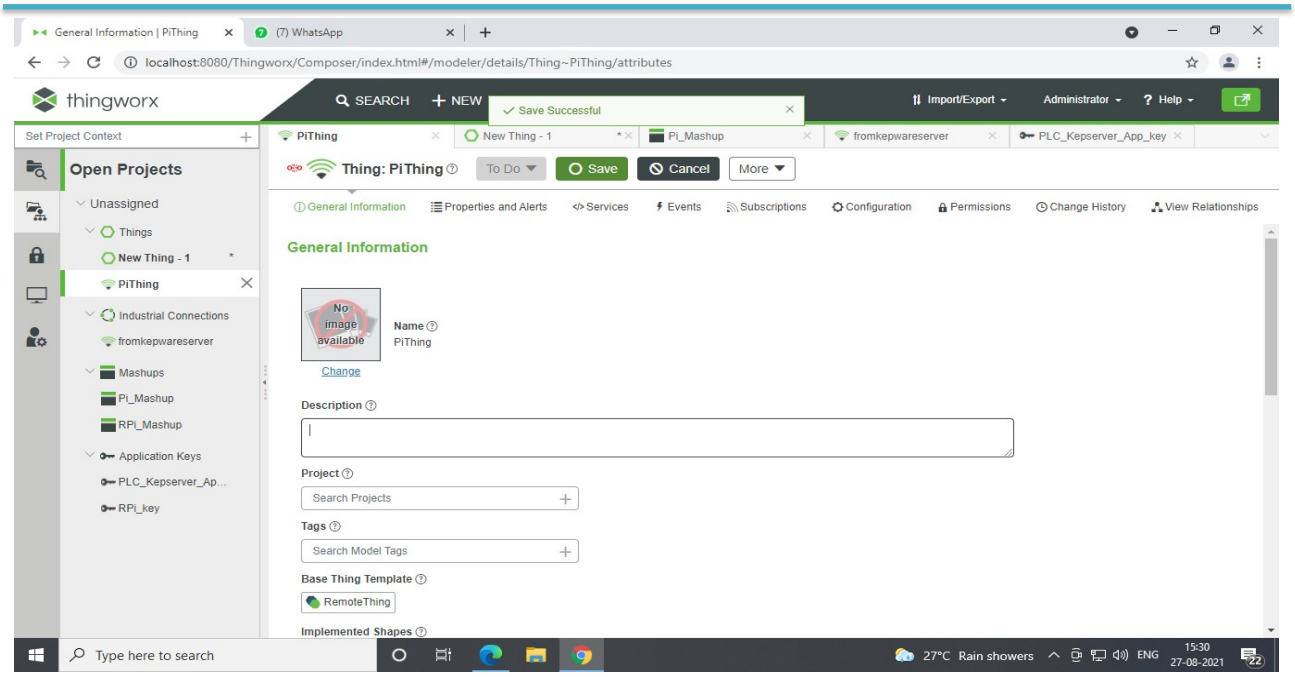
- Provide identifier name and note down the same we required this one in further



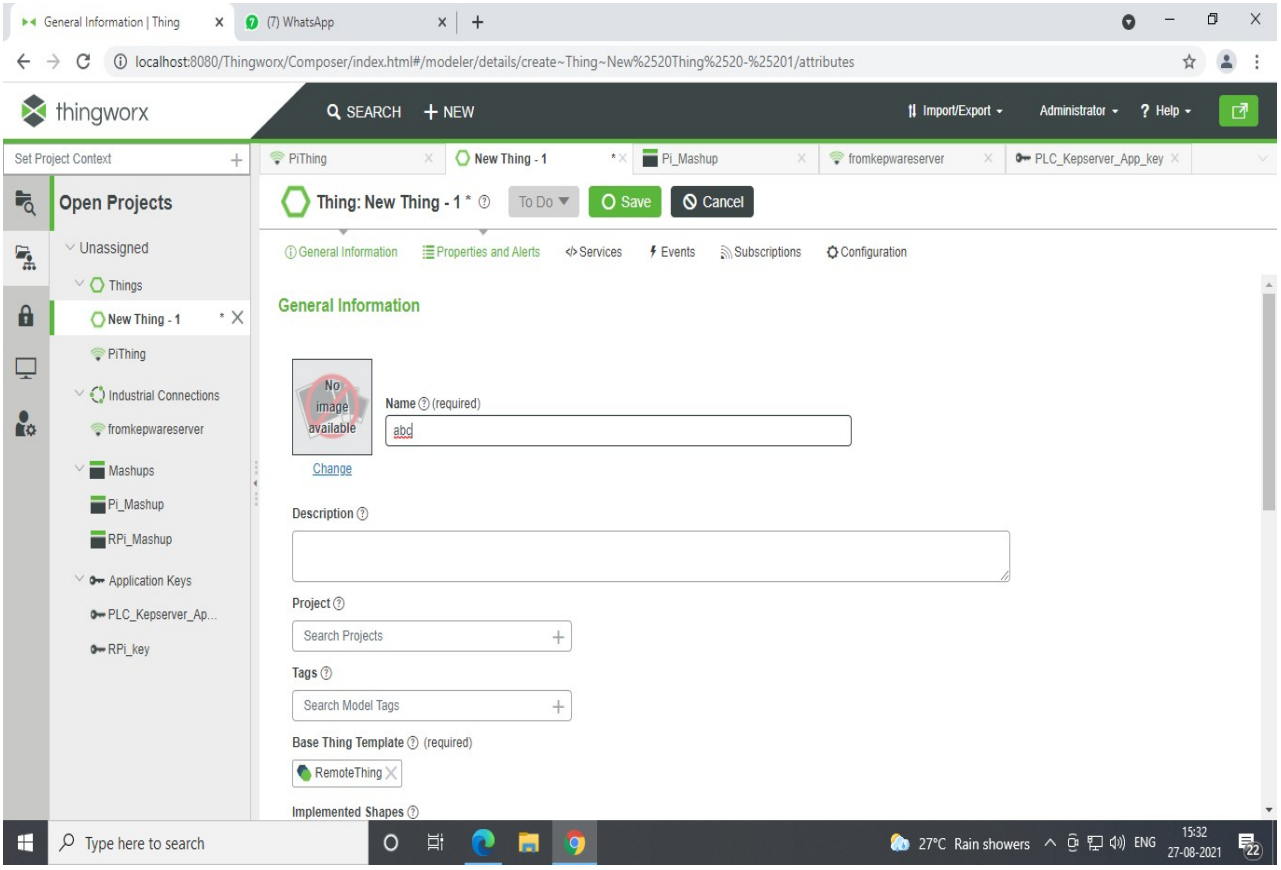
After filling in all details click on save



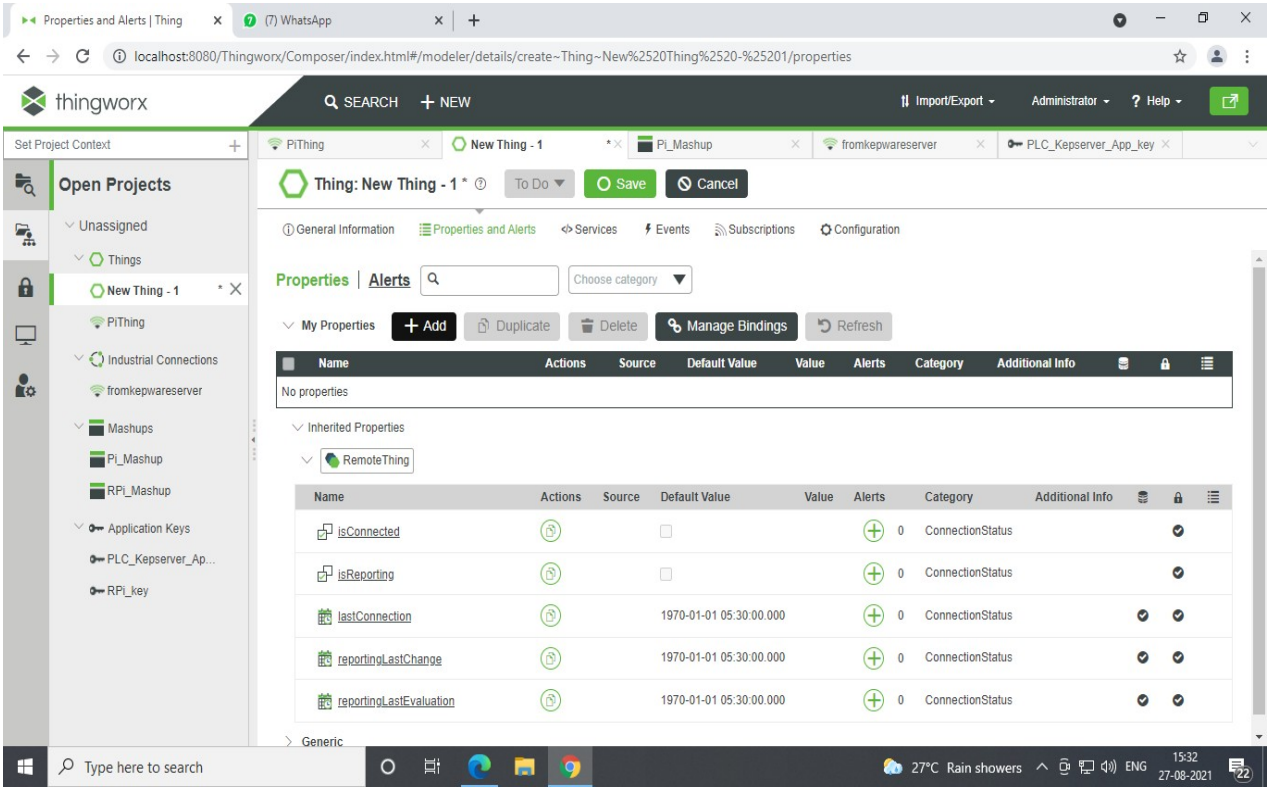
Then save successfully pop up screen will appear as shown below



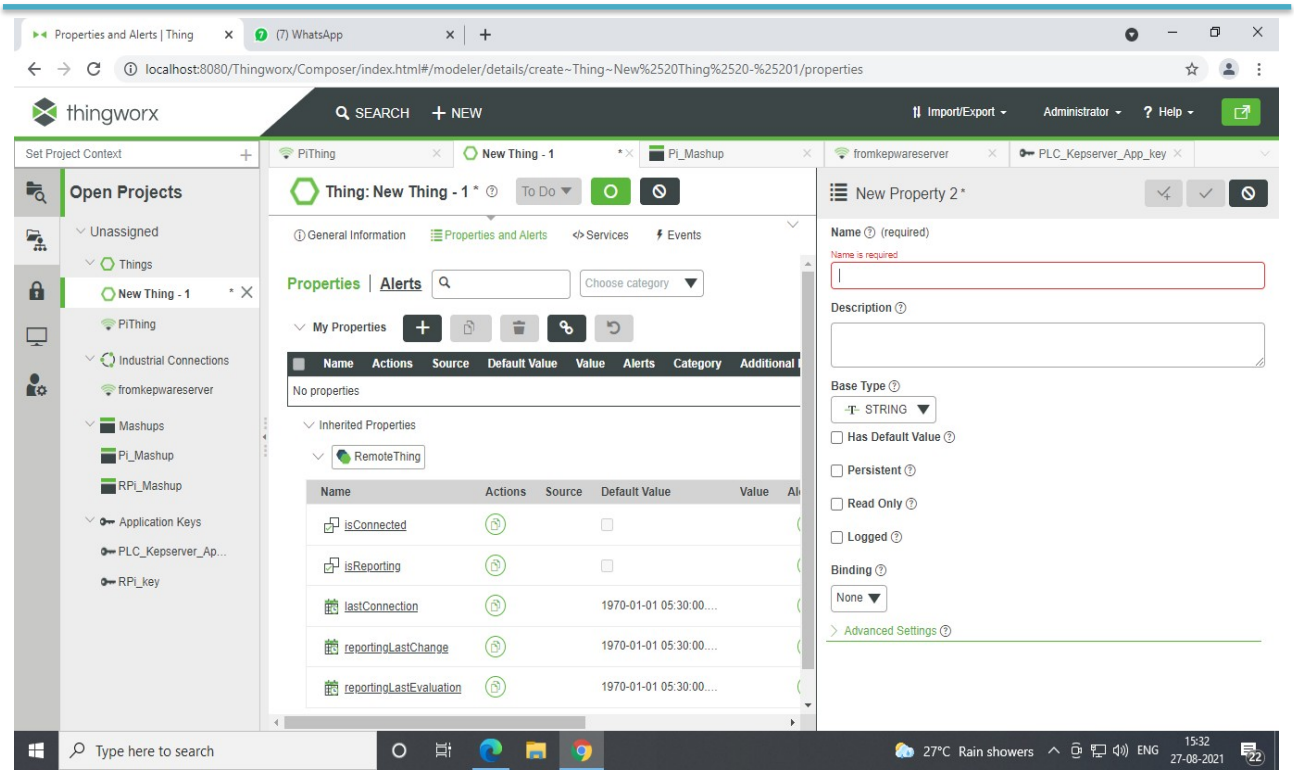
- Go to Things > Properties and alerts



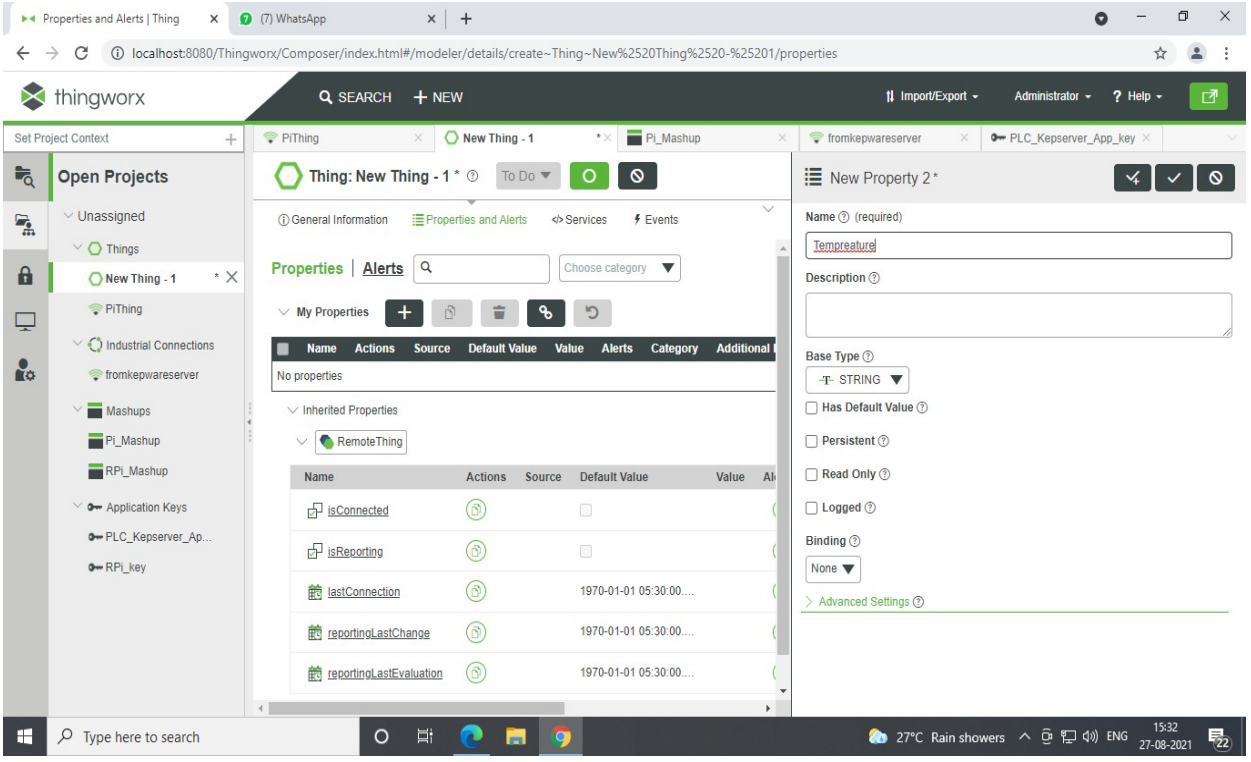
- Click on ADD to add new properties



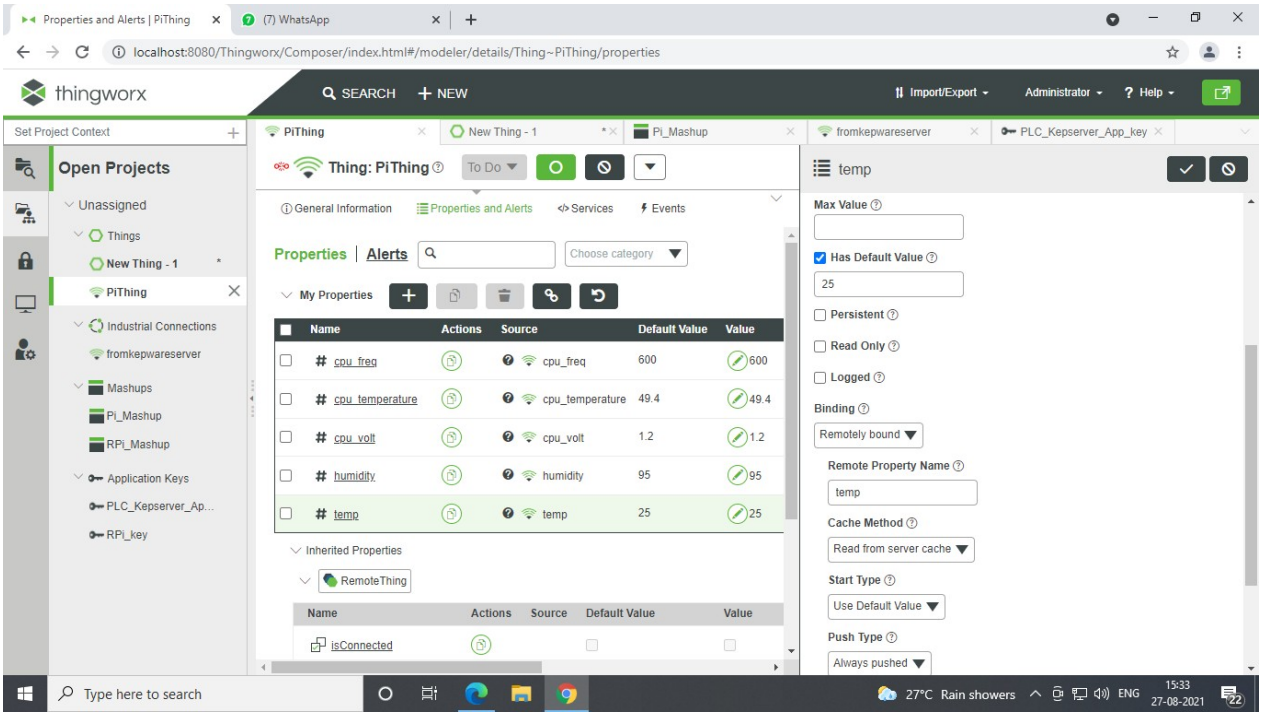
- Provide the name of the property for DHT 11 temperature
- For each value of a different parameter create one property



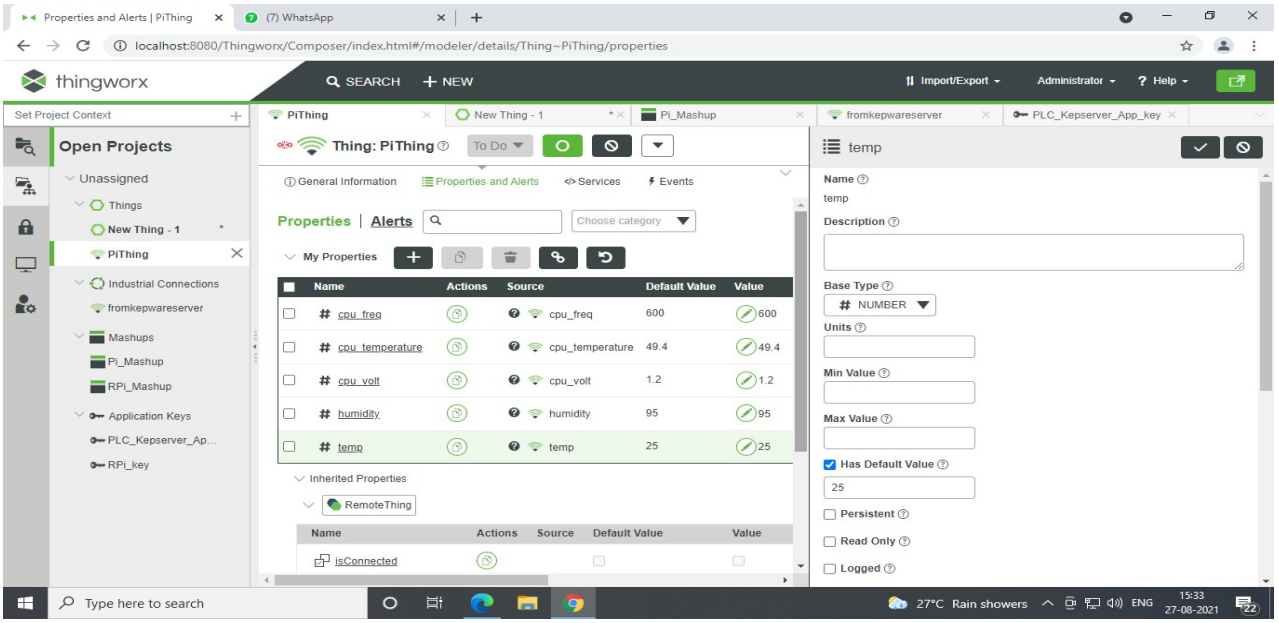
- Provide property name



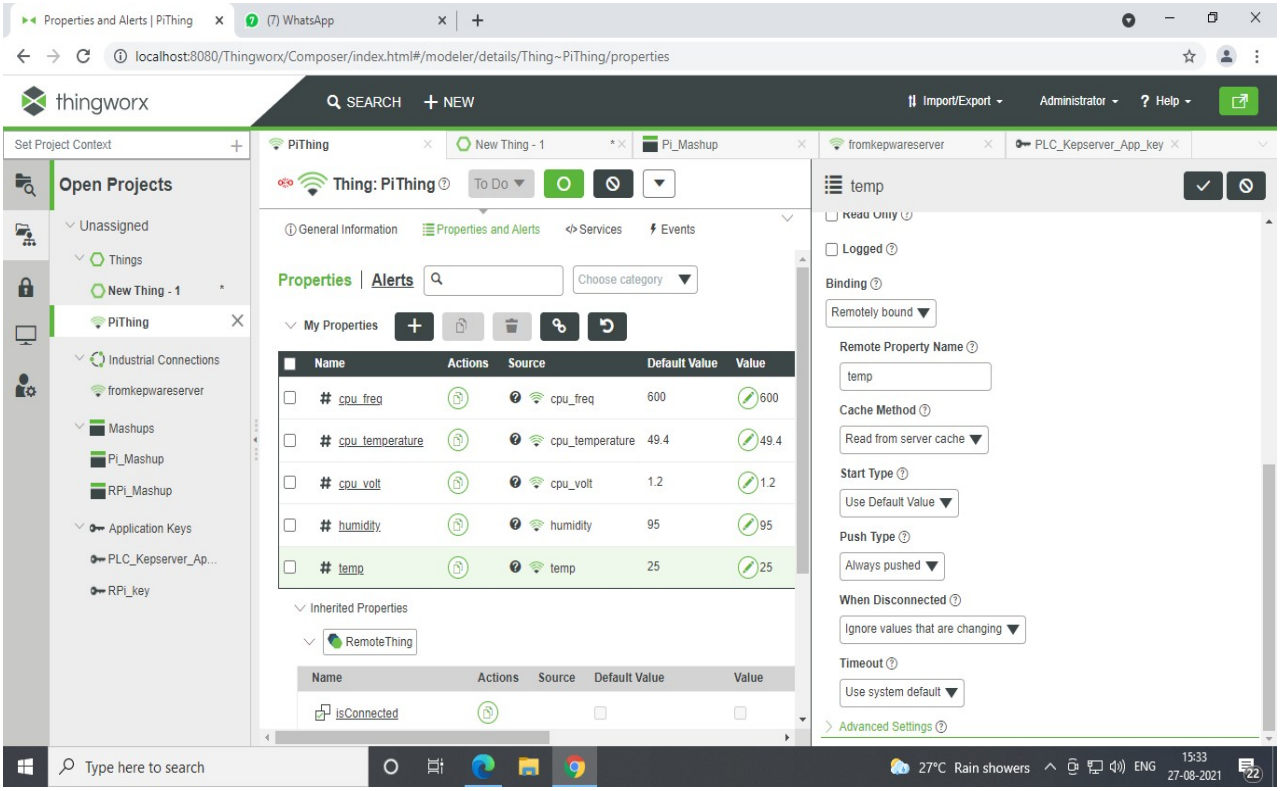
- Provide data type in Base type



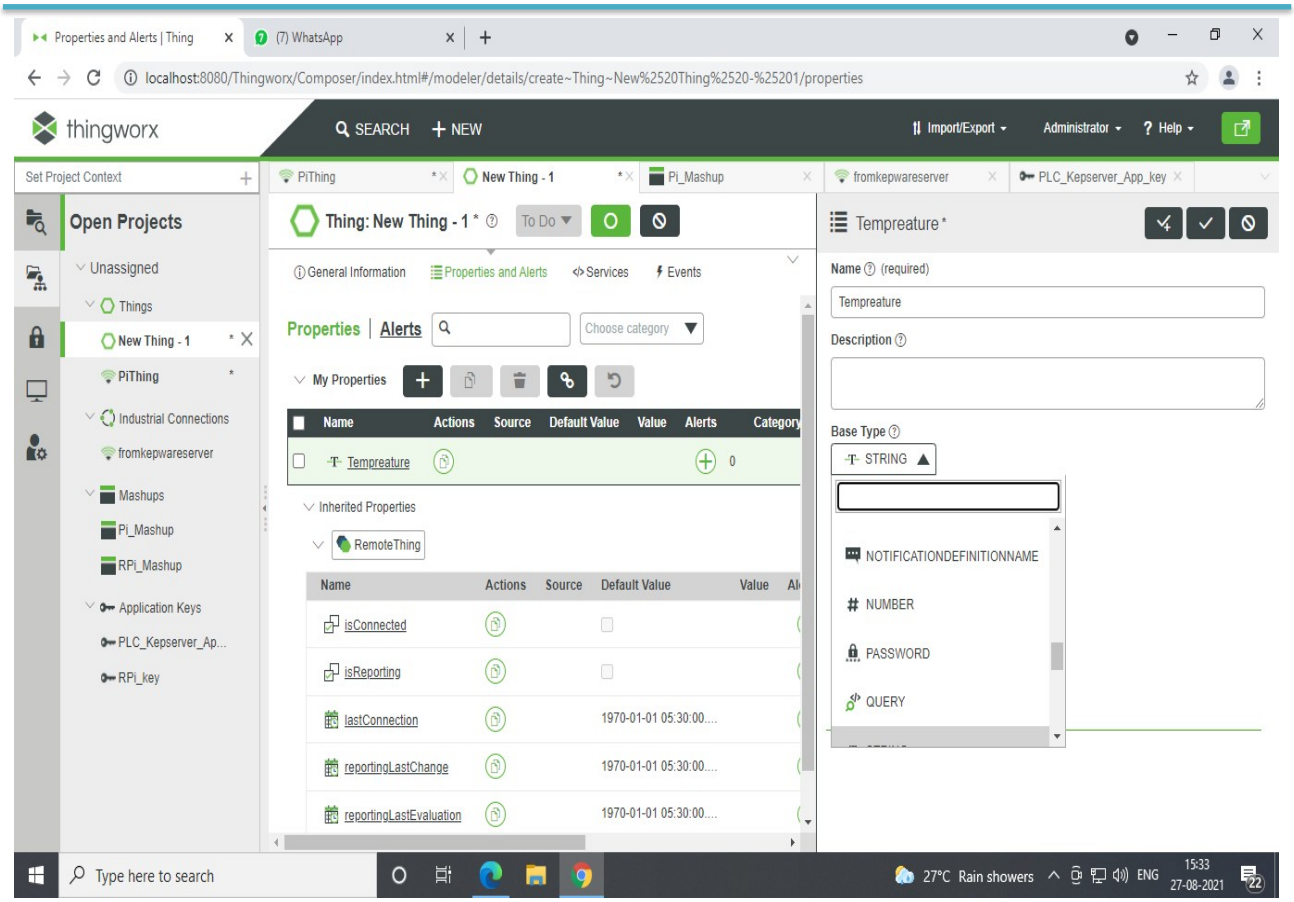
- Provide Default value

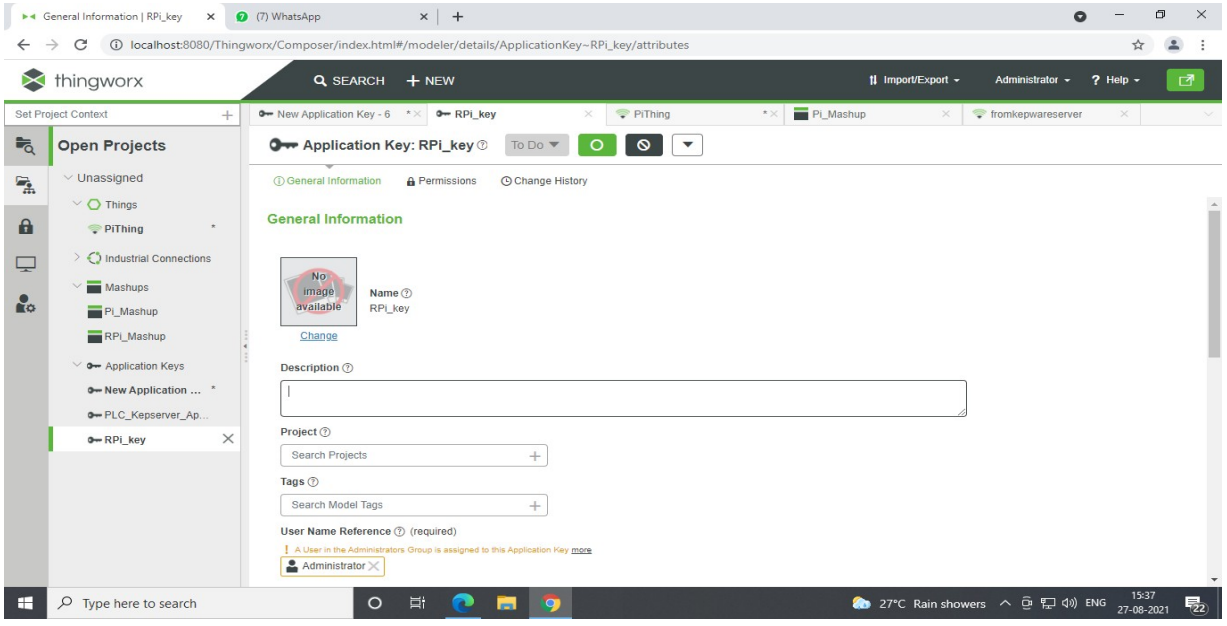


- Click on the right mark after setting all parameters as shown below

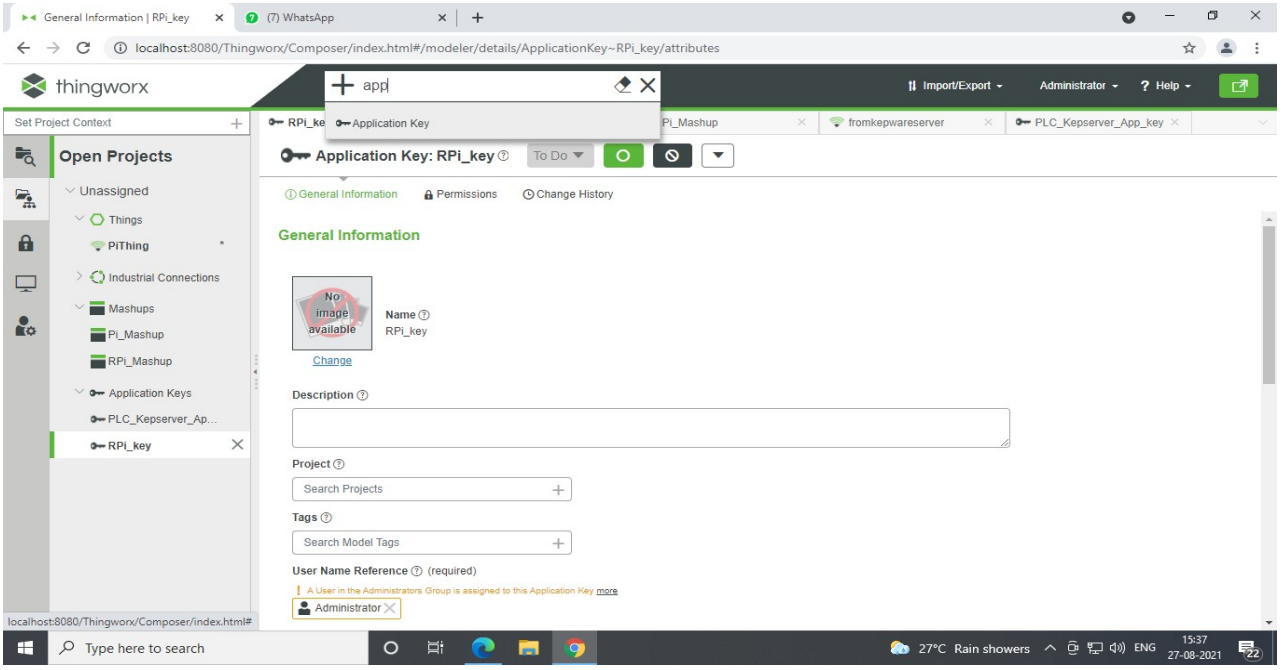


- As explained earlier created multiple properties

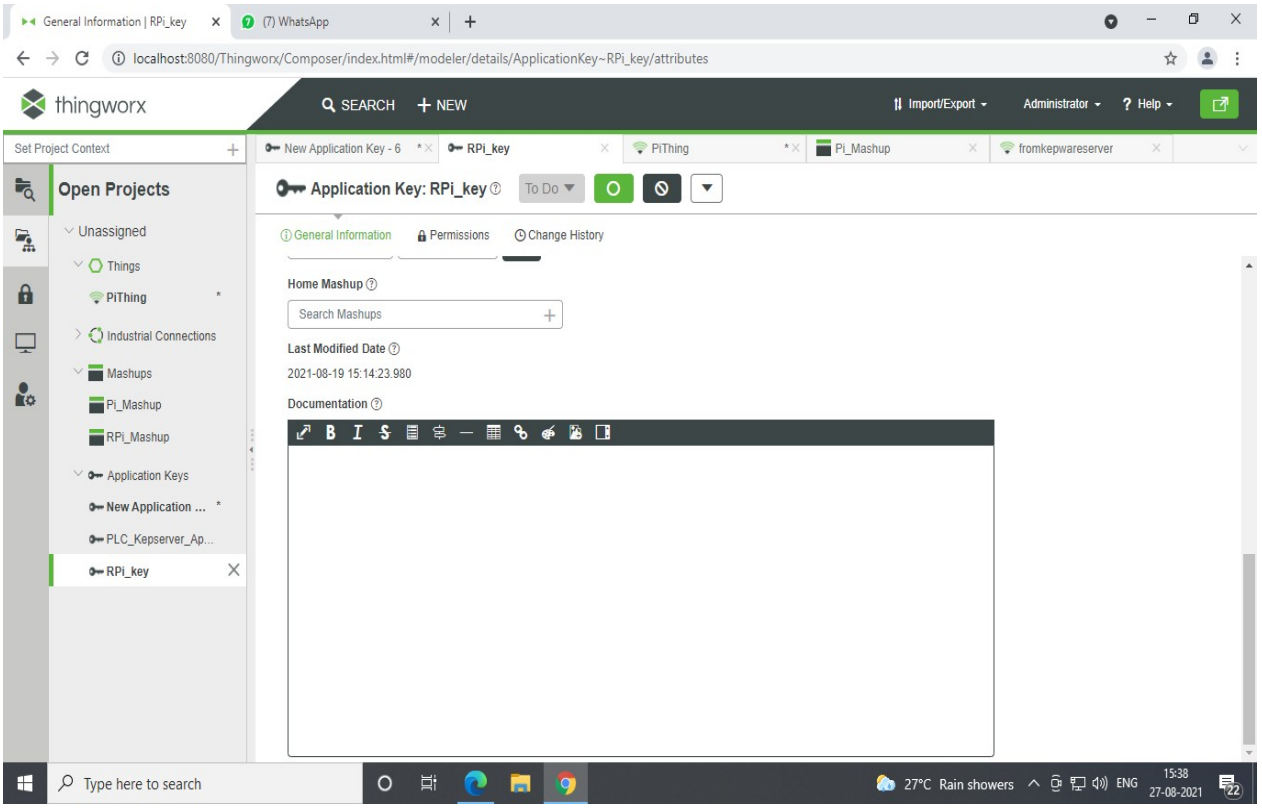

Properties and Alerts | Thing
(7) WhatsApp
localhost:8080/Thingworx/Composer/index.html#/modeler/details/create~Thing~New%2520Thing%2520-%25201/properties
thingworx
SEARCH
NEW
Import/Export
Administrator
Help
Set Project Context
Open Projects
Unassigned
Things
New Thing - 1
PiThing
Industrial Connections
fromkepwareserver
Mashups
Pi_Mashup
RPi_Mashup
Application Keys
PLC_Kepserver_Ap...
RPi_key
Thing: New Thing - 1 *
To Do
General Information
Properties and Alerts
Services
Events
Properties
Alerts
Choose category
My Properties
Name
Actions
Source
Default Value
Value
Alerts
Category
Tempreature
Inherited Properties
RemoteThing
isConnected
isReporting
lastConnection
1970-01-01 05:30:00....
reportingLastChange
reportingLastEvaluation
Tempreature *
Name (required)
Description
Base Type
STRING
NOTIFICATIONDEFINITIONNAME
NUMBER
PASSWORD
QUERY
Type here to search
27°C Rain showers
ENG
15:33
27-08-2021


General Information | RPi_key
(7) WhatsApp
localhost:8080/Thingworx/Composer/index.html#/modeler/details/ApplicationKey~RPi_key/attributes
thingworx
SEARCH
NEW
Import/Export
Administrator
Help
Set Project Context
Open Projects
Unassigned
Things
PiThing
Industrial Connections
Mashups
Pi_Mashup
RPi_Mashup
Application Keys
New Application ...
PLC_Kepserver_Ap...
RPi_key
Application Key: RPi_key
To Do
General Information
Permissions
Change History
General Information
No image available
Change
Name
RPi_key
Description
Project
Search Projects
Tags
Search Model Tags
User Name Reference (required)
A User in the Administrators Group is assigned to this Application Key more
Administrator
Type here to search
27°C Rain showers
ENG
15:37
27-08-2021

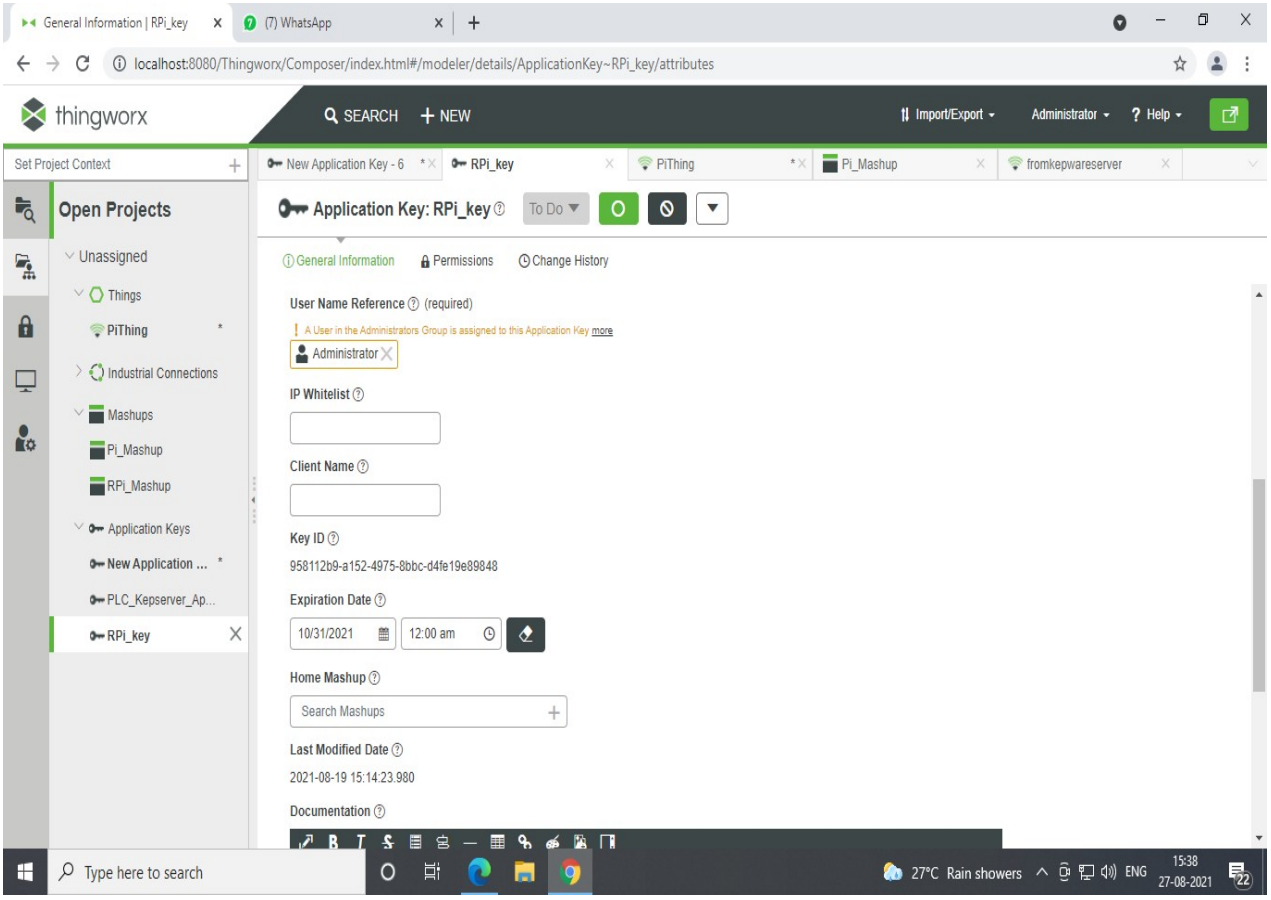
- Create APP key



- Provide user name reference Administrator and expiry date

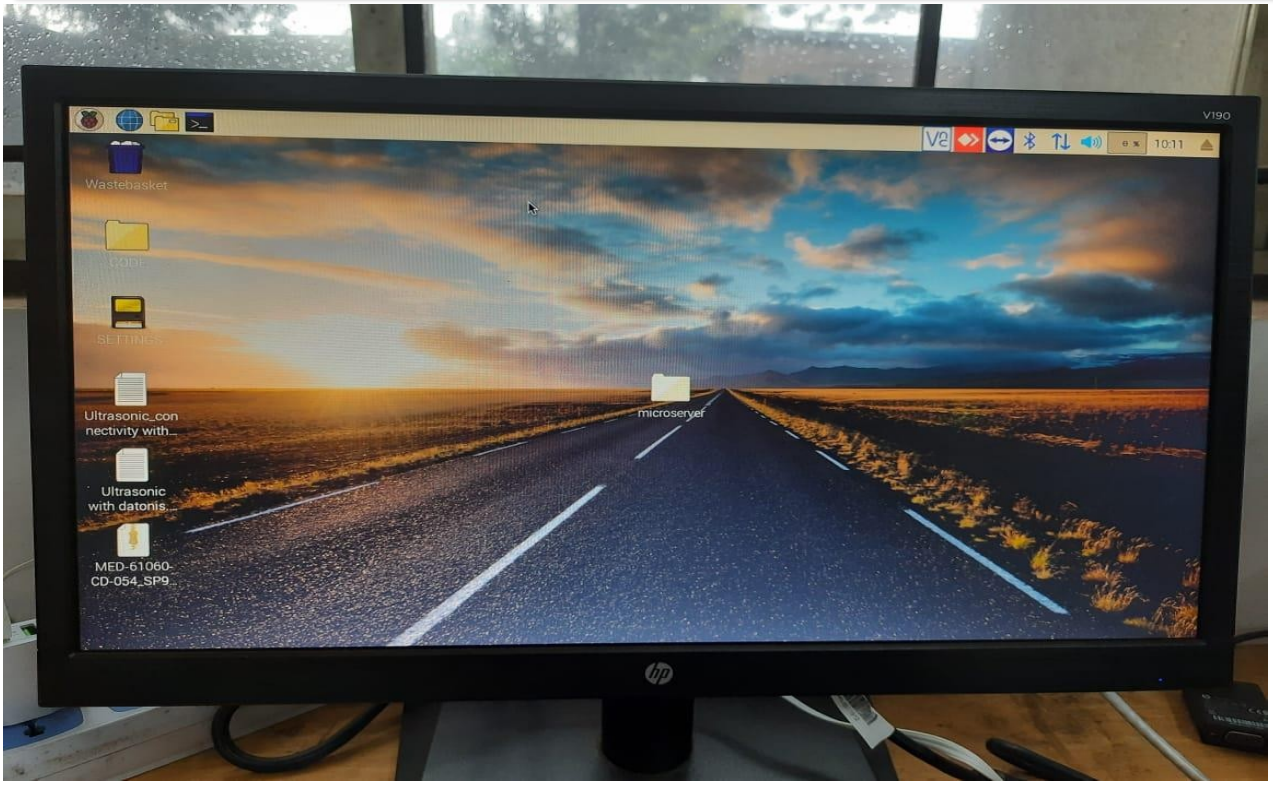


- Provide App key

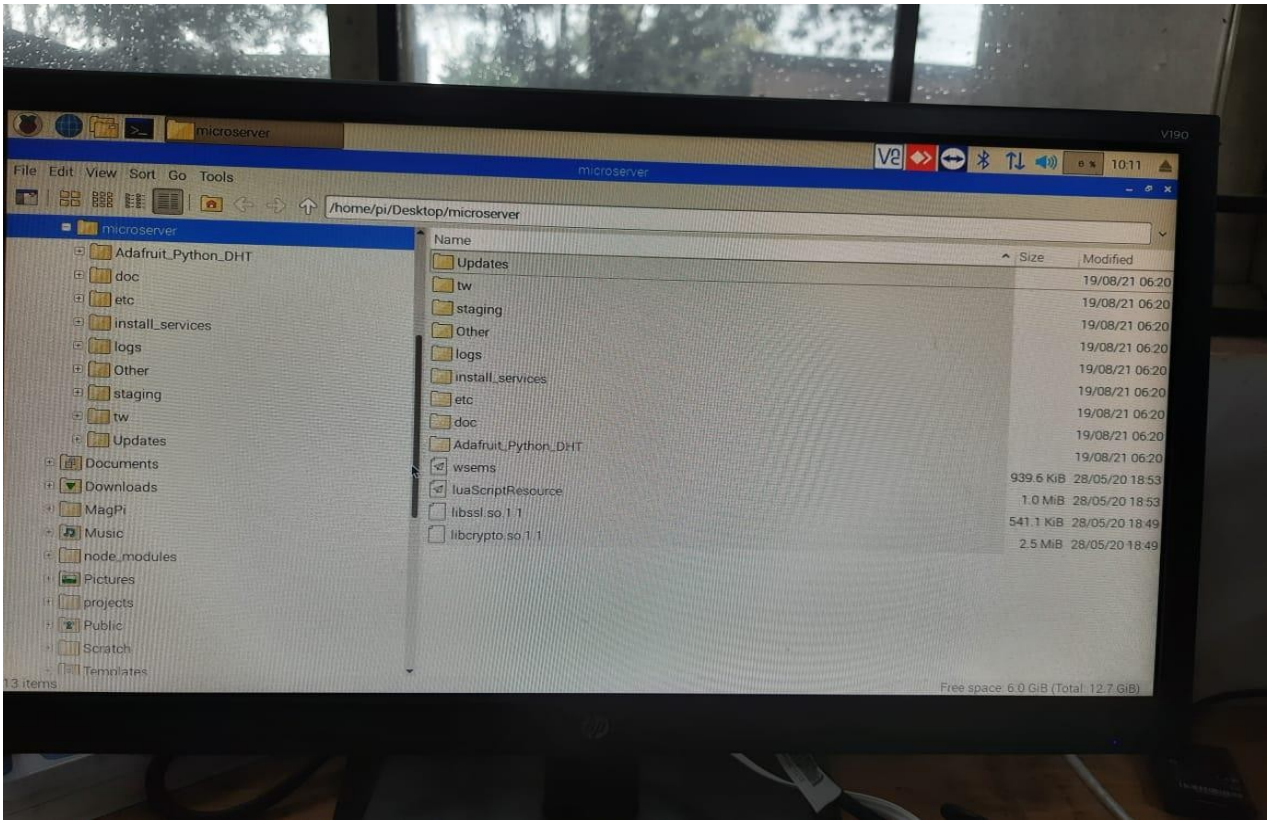


- Copy Key ID and save in note pad.

- Go to raspberry pi and Download the standard micro server file from the below link or PTC official website.
  https://drive.google.com/drive/folders/1FmeRp0fAVcj5pfT9Qrdpz2CfeYkTVXkB?usp=sharing



- Open folder Go to etc. folder



- Open config. json file

  In Ws server host: Provide IP address of PC

  Port no:8080

  Provide APP key in-app key portion which we have created and copied to notepad previously.

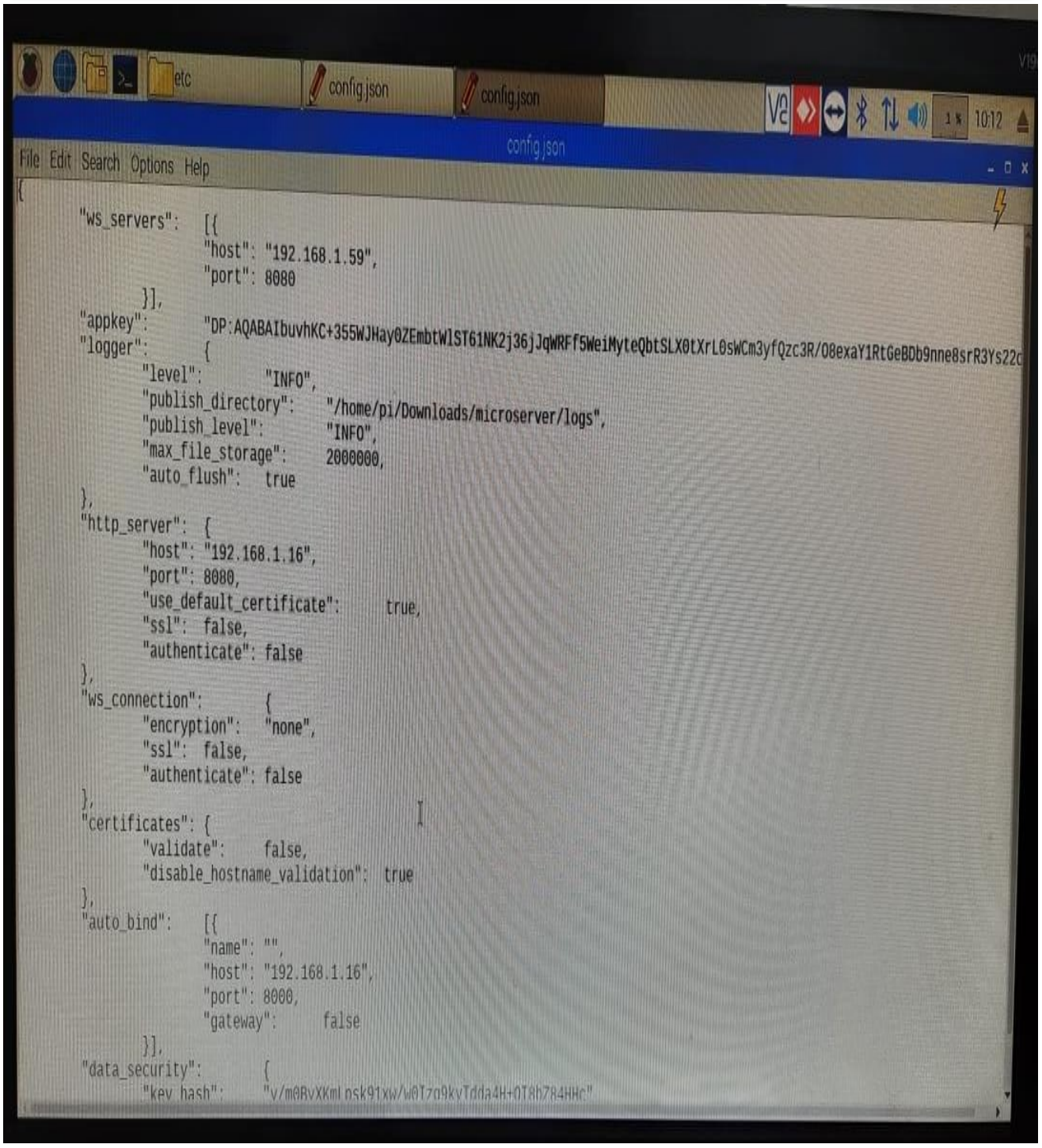
  In http sever host: provide the IP address of the raspberry pi

  Port:8080

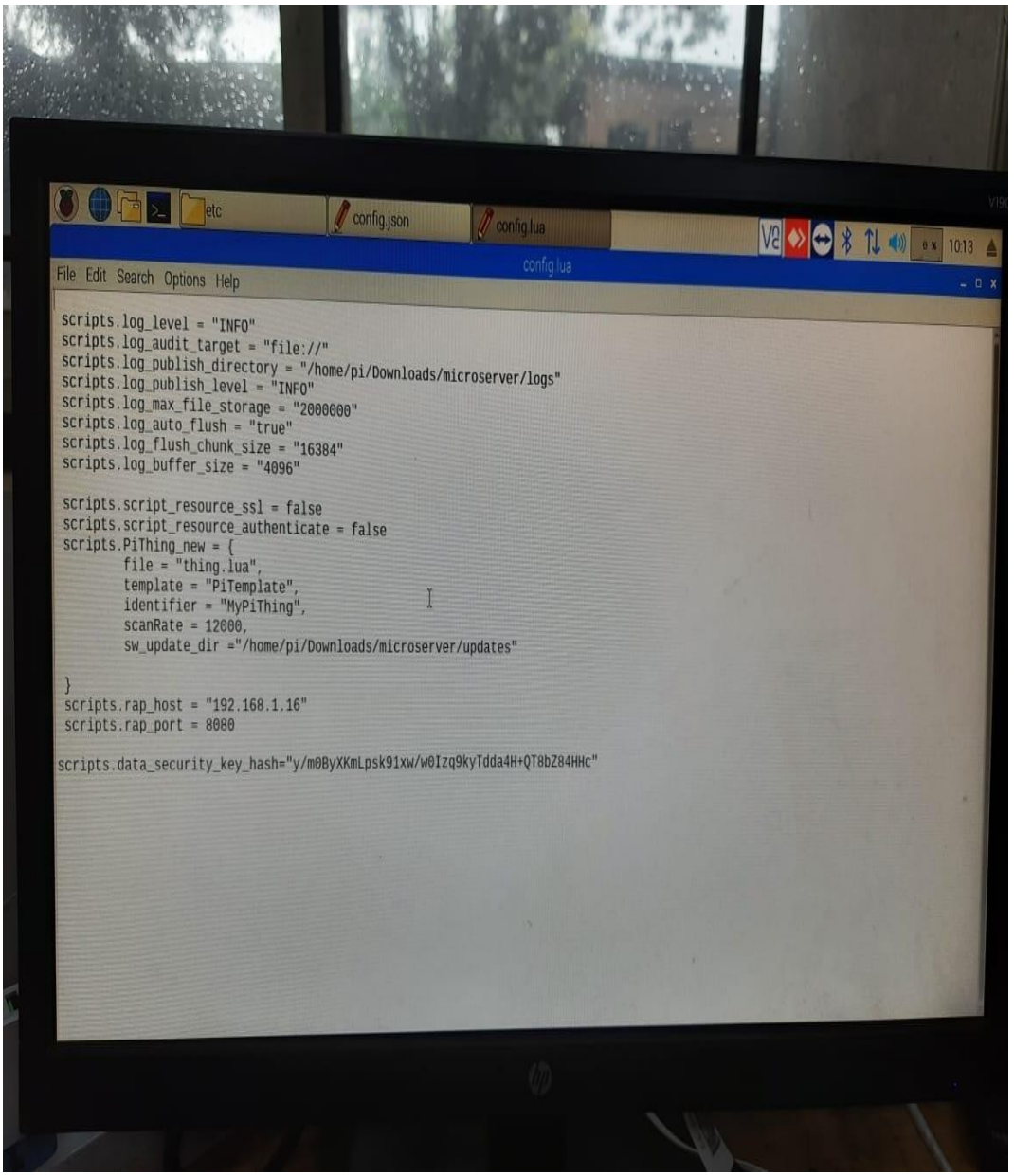
  In Auto bind host: provide the IP address of the raspberry pi

  Port:8000

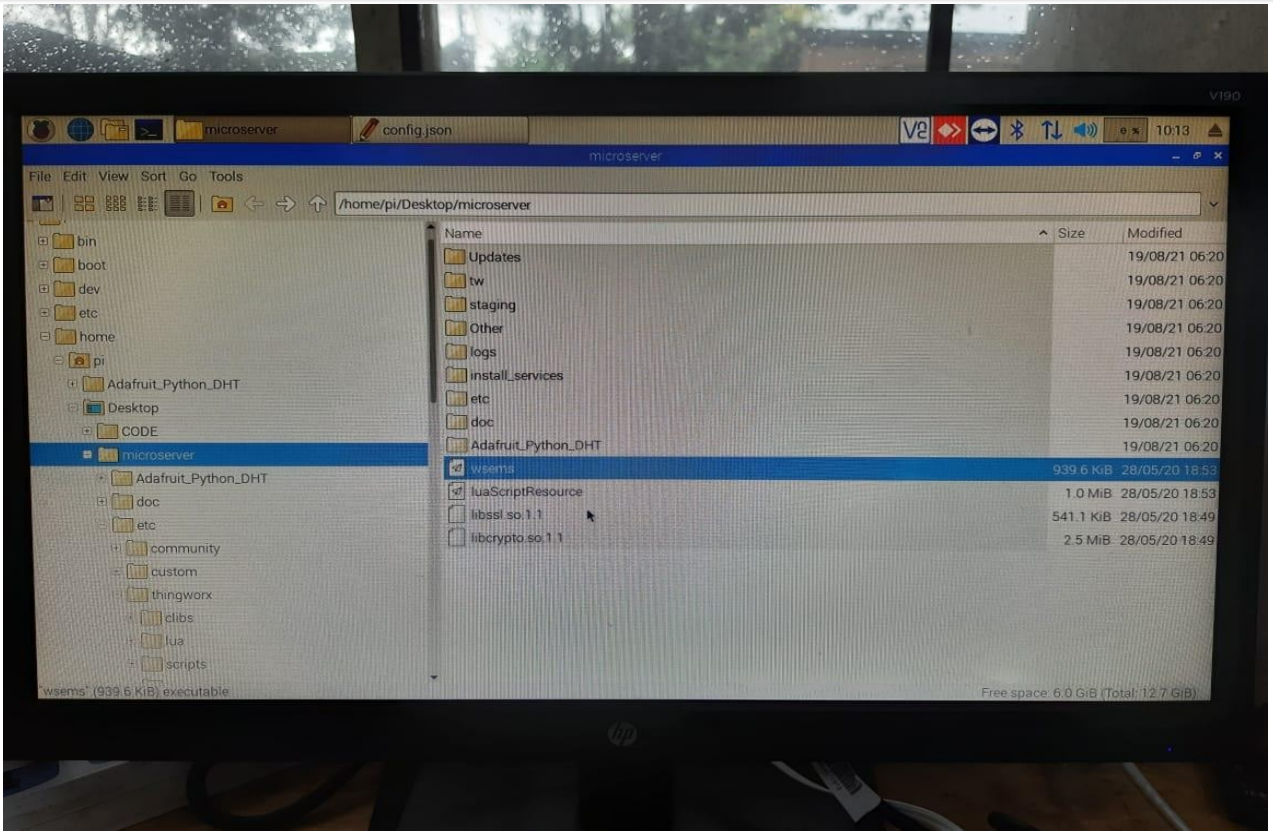
  Save and close.


etc
config.json
config.json
10:12
config.json
File Edit Search Options Help
{
"ws_servers": [{
"host": "192.168.1.59",
"port": 8080
}],
"appkey": "DP:AQABAIbuvhKC+355WJHay0ZEmbtWlST61NK2j36jJqWRFf5WeiMyteQbtSLX0tXrL0sWCm3yfQzc3R/O8exaY1RtGeBDb9nne8srR3Ys22c
"logger": {
"level": "INFO",
"publish_directory": "/home/pi/Downloads/microserver/logs",
"publish_level": "INFO",
"max_file_storage": 2000000,
"auto_flush": true
},
"http_server": {
"host": "192.168.1.16",
"port": 8080,
"use_default_certificate": true,
"ssl": false,
"authenticate": false
},
"ws_connection": {
"encryption": "none",
"ssl": false,
"authenticate": false
},
"certificates": {
"validate": false,
"disable_hostname_validation": true
},
"auto_bind": [{
"name": "",
"host": "192.168.1.16",
"port": 8000,
"gateway": false
}],
"data_security": {
"key_hash": "v/m0RvXKmlnsk91xw/w0Tzn9kvTdda4H+0T8h784HHc"

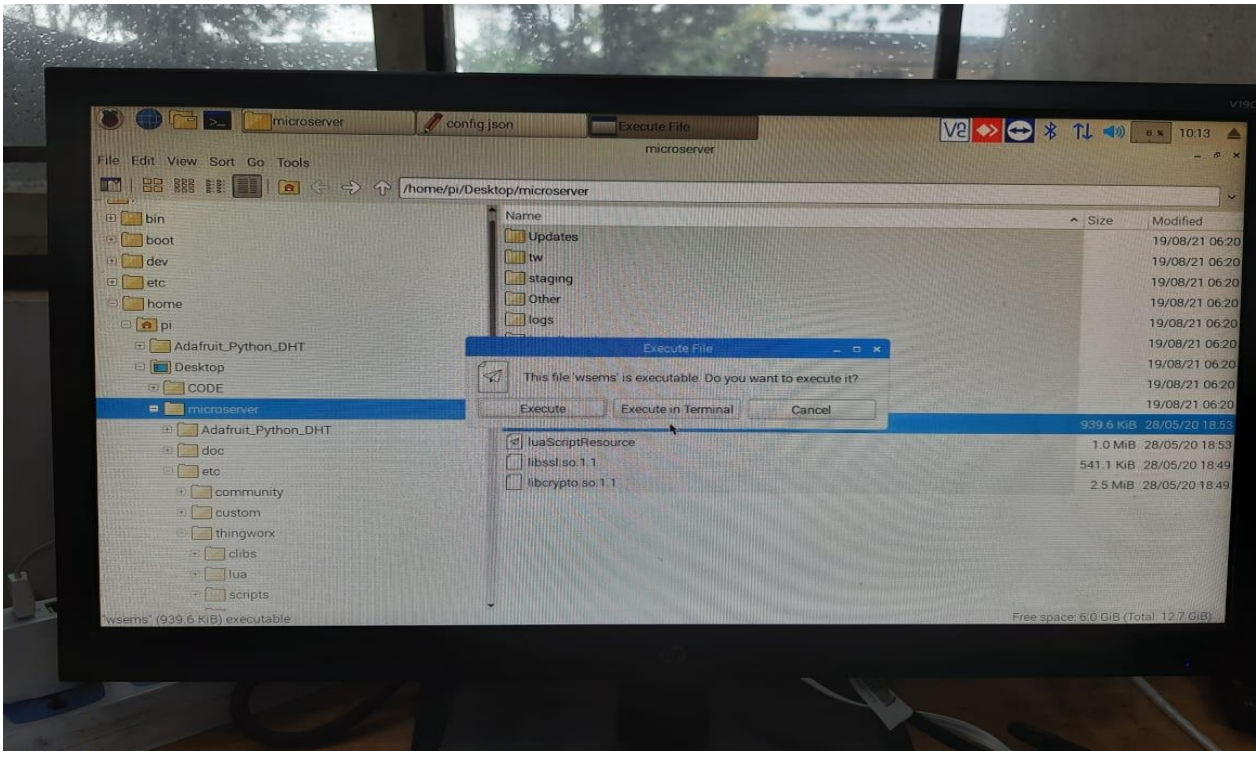
- Open config.lua
  In identifier provide identifier name which we have created earlier.
  In scripts.rap_post host: provide the IP address of the raspberry pi
  Port:8080
  Save and close.



- Go to the microservers folder and double click on wsems 939.6kb file

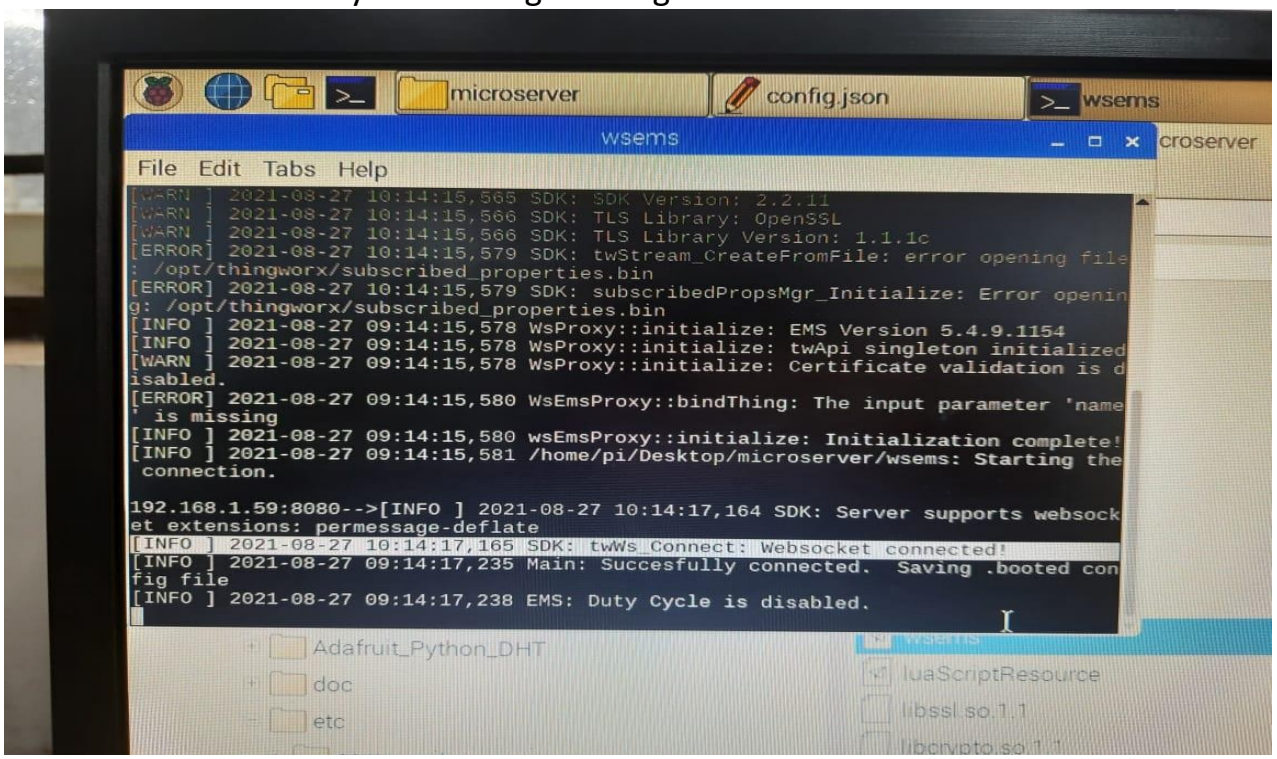


- Then popup screen as shown below will occur select execute in terminal

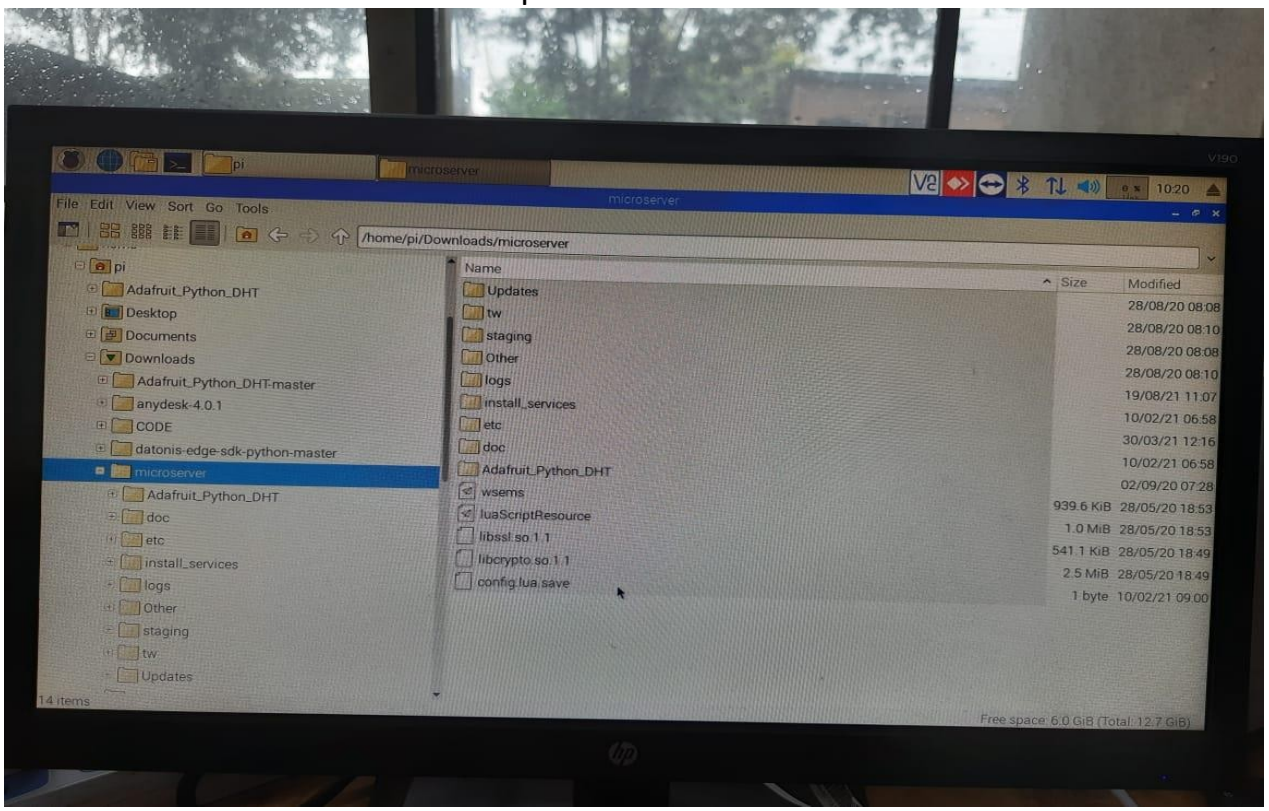


- Logs will open as shown below.

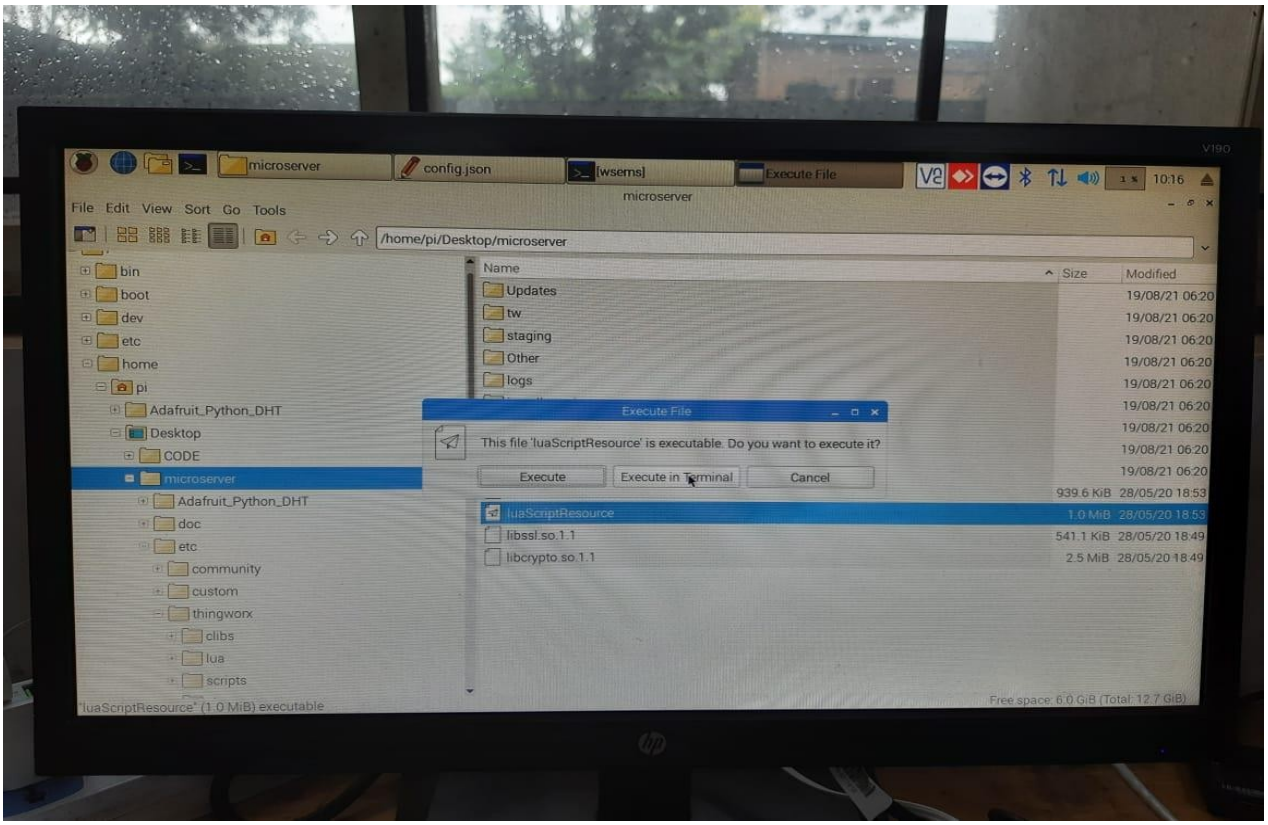
You will get logs like web socket successfully connected.
Minimize the logs window.
On the Thingworx server, u will get a pop-up of things connected.
And connection symbol changes into green.



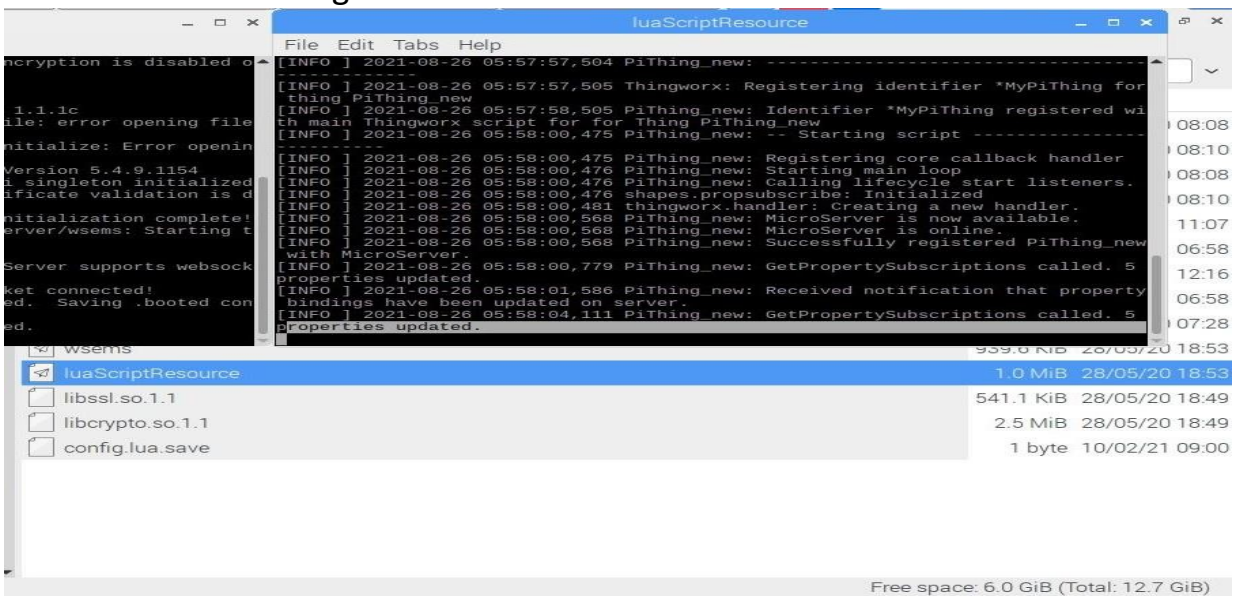
- Double click on the Lua-script resource file



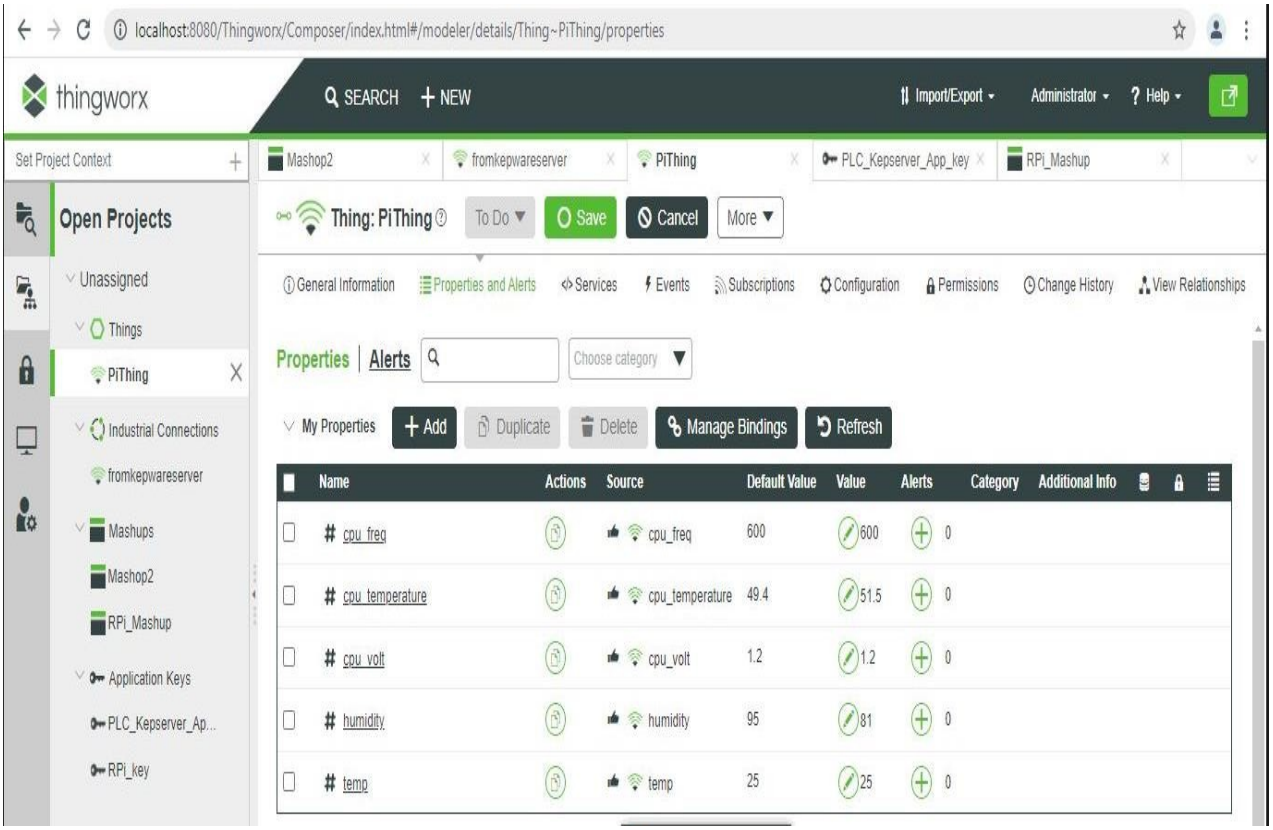
- Then popup screen as shown below will occur select execute in terminal



- Logs will open as shown below.
  You will get logs like property updated successfully.
  Minimize the logs window.

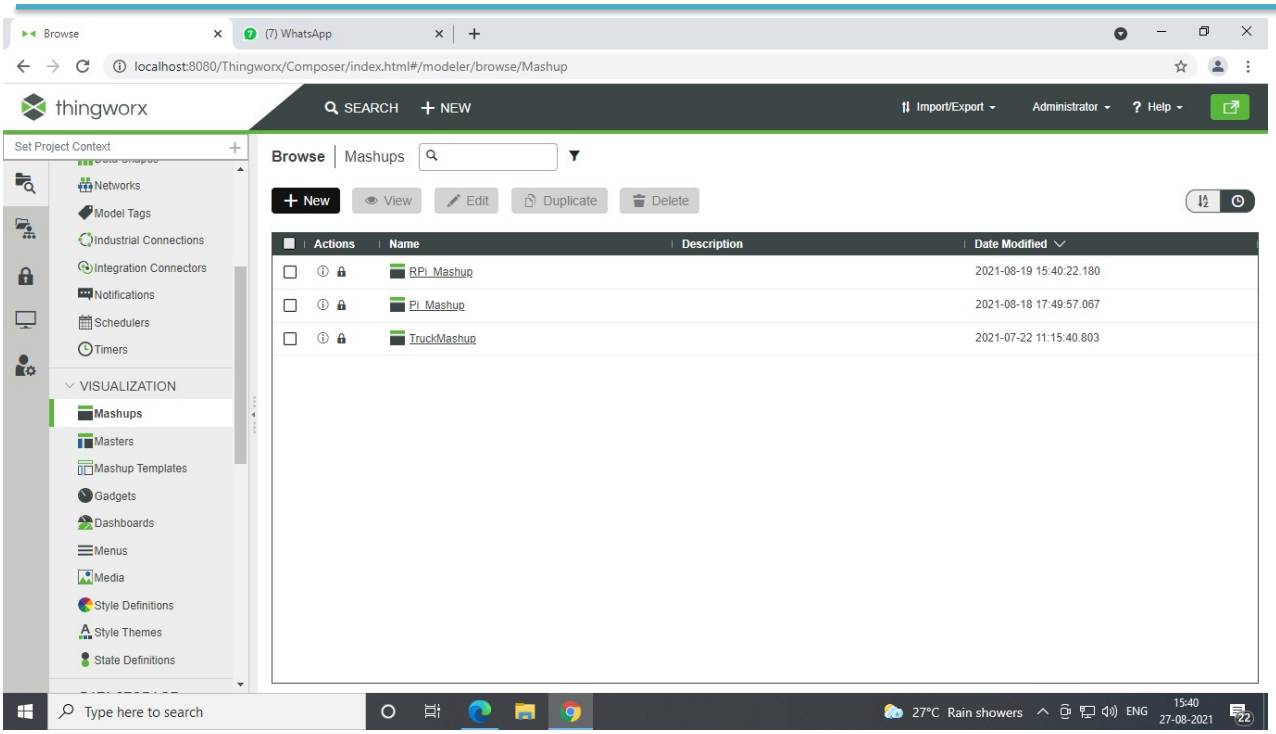


- In Thingworx we get the sensor value in things.

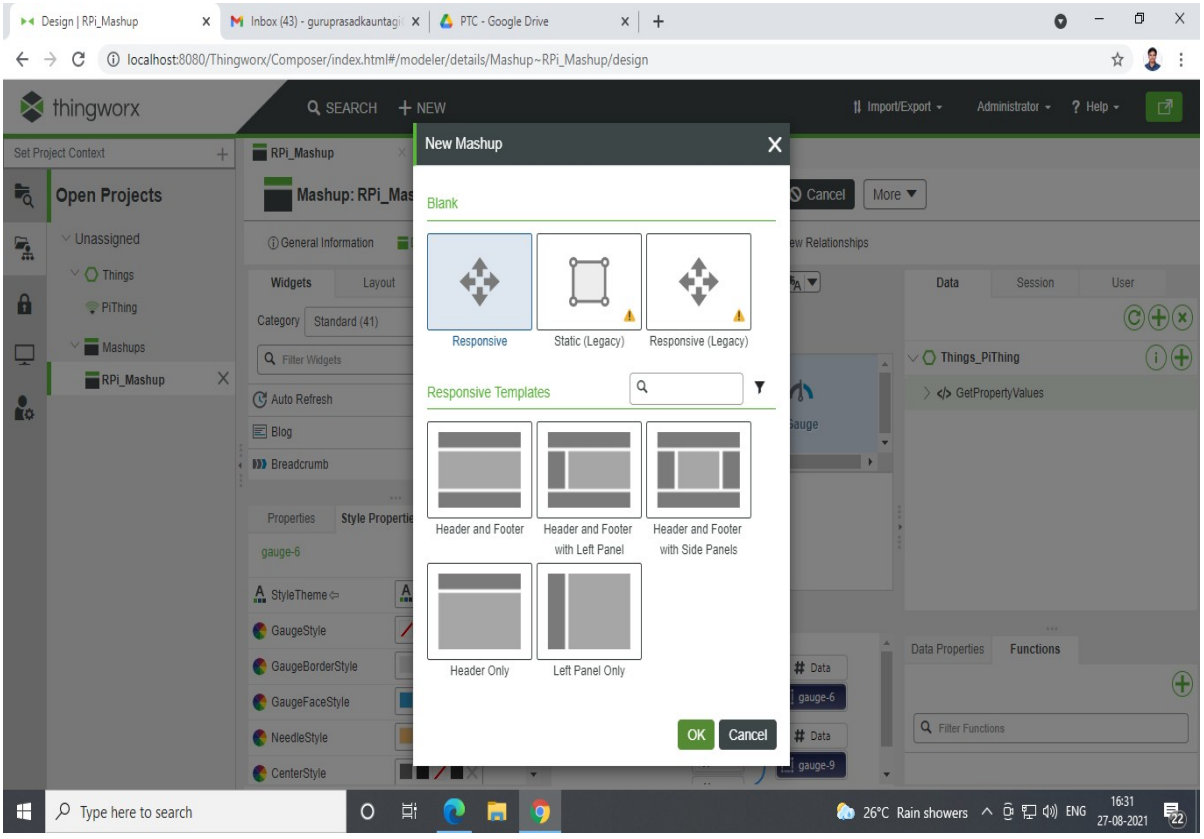


Thingworx Mashup creation

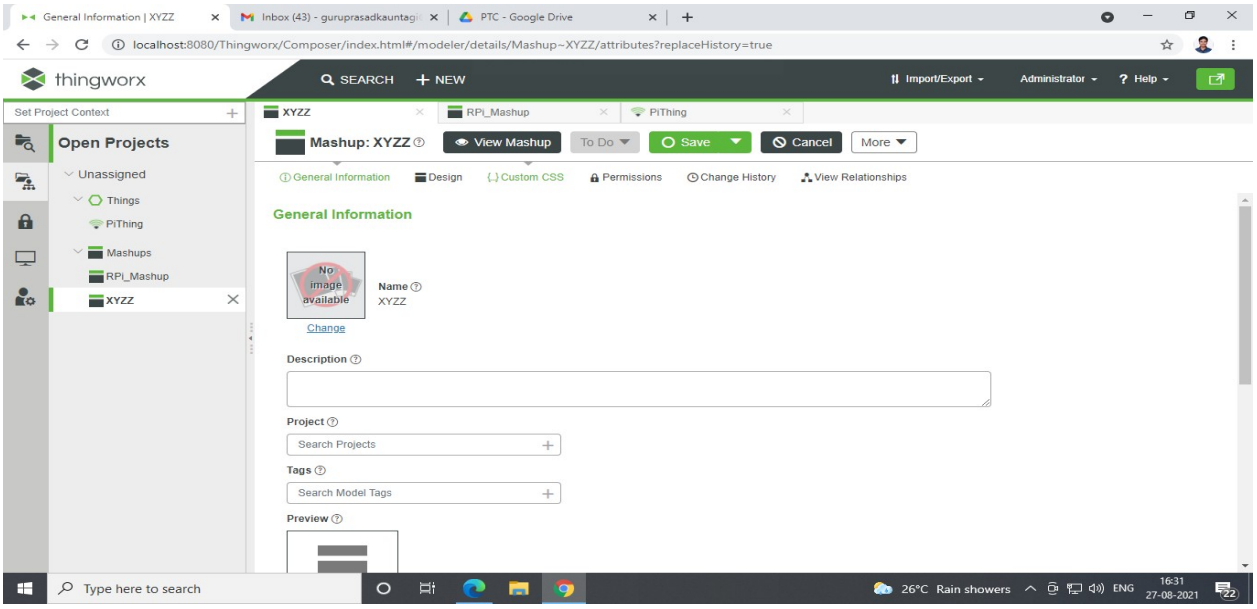
- Go to a mashup in Thingworx click on New



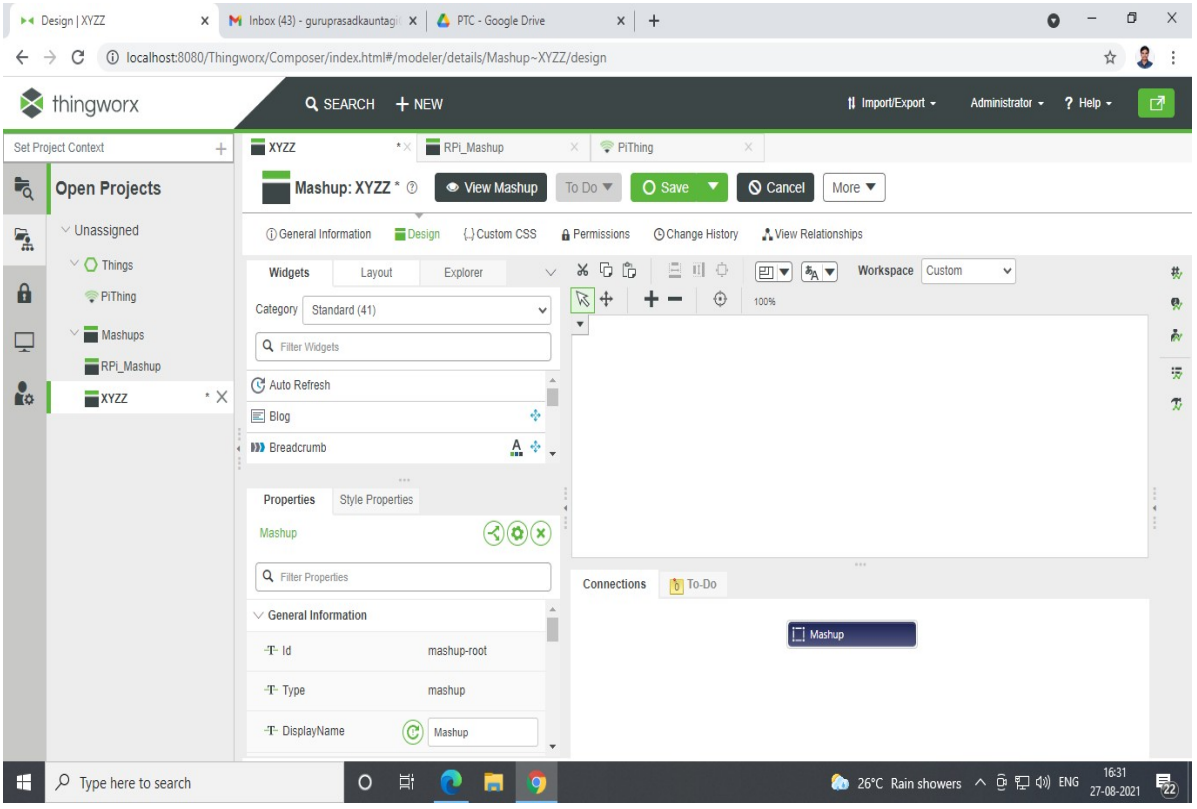
- Click on responsive



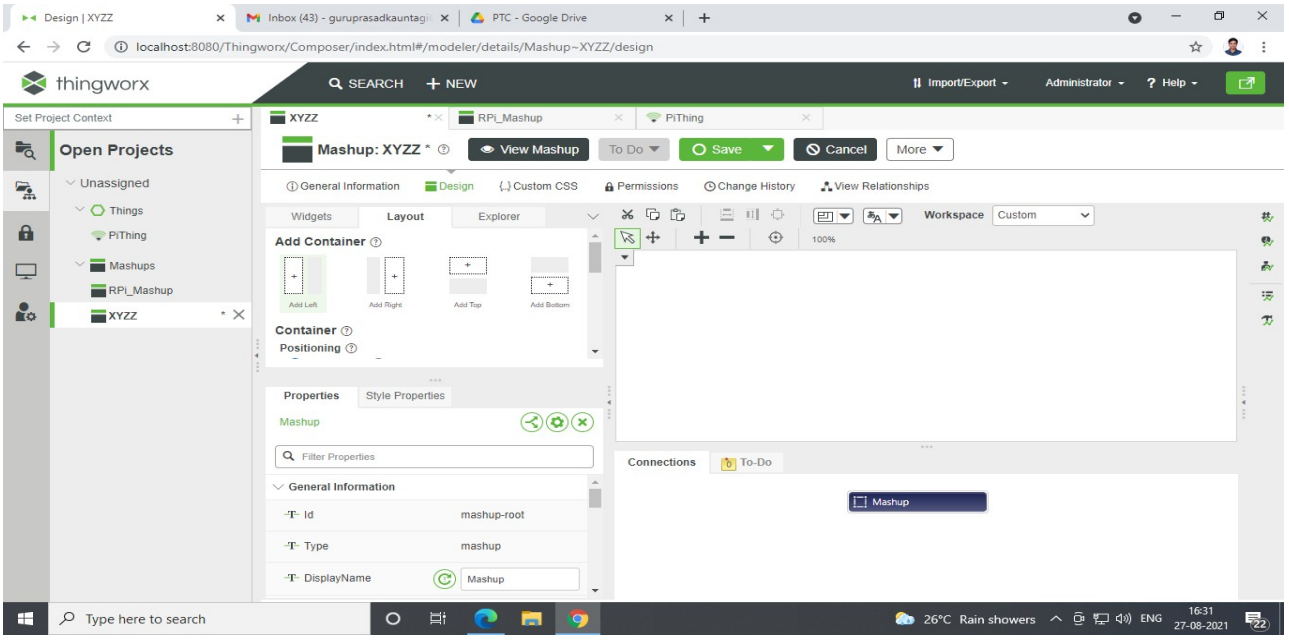
- Provide name Mashup Name and save it



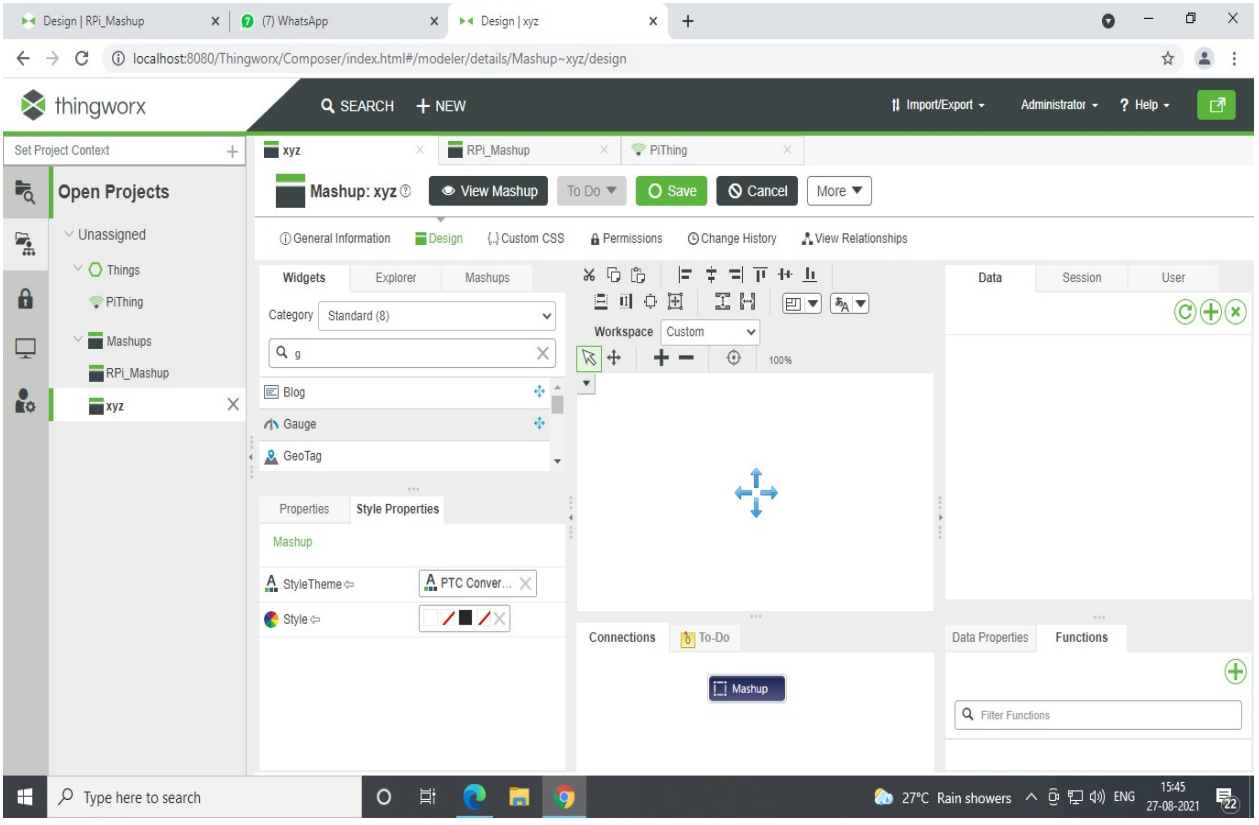
- Go to design and widgets



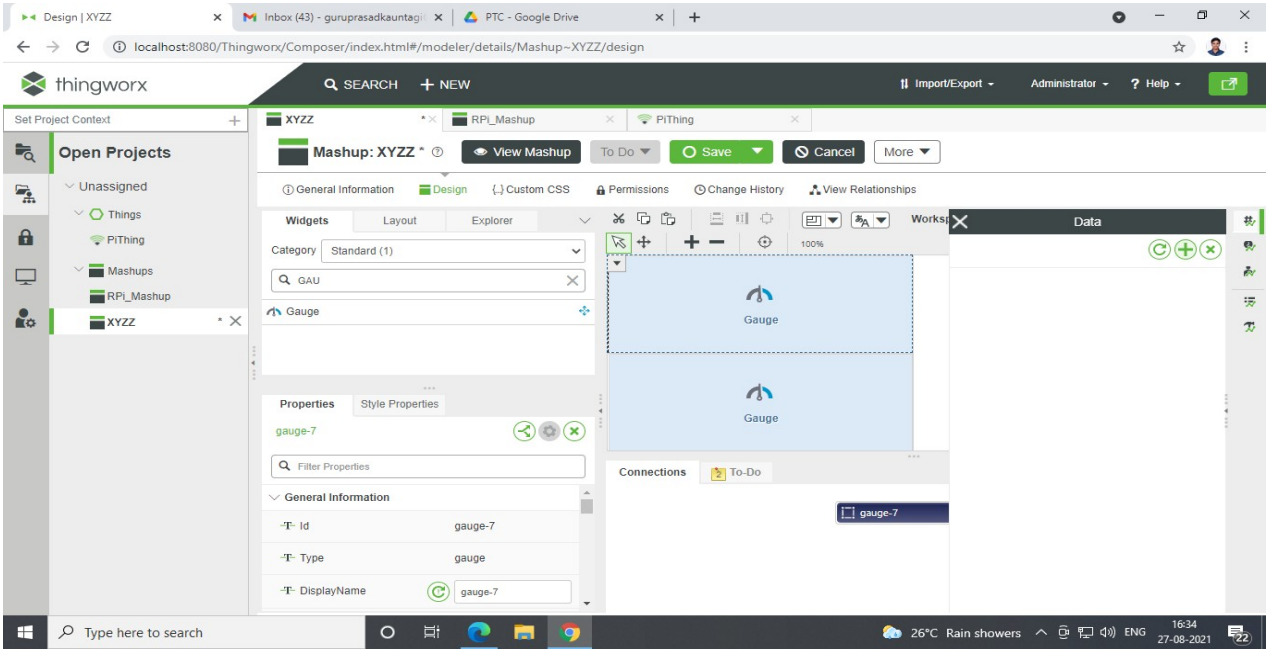
- Add Container



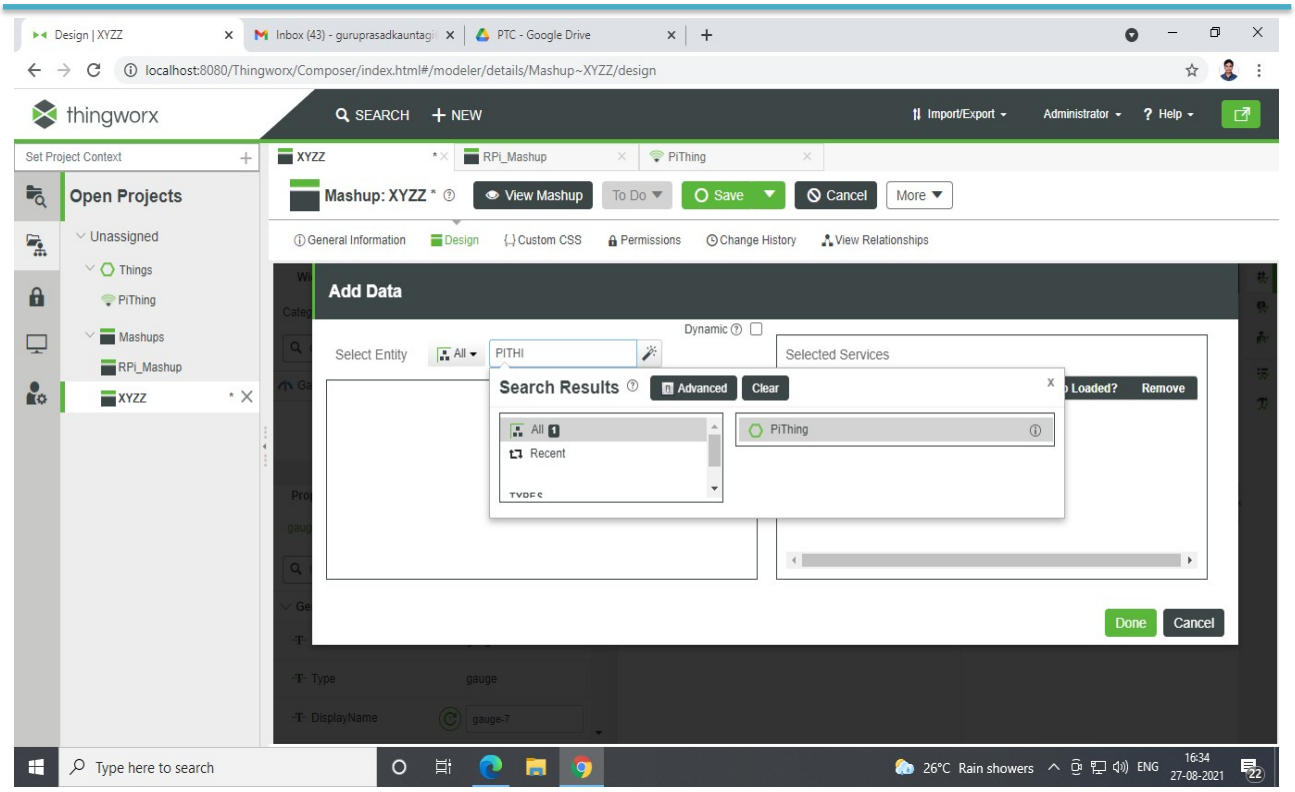
- Search for Appropriate mashup as per data as of now I am selecting the gauge



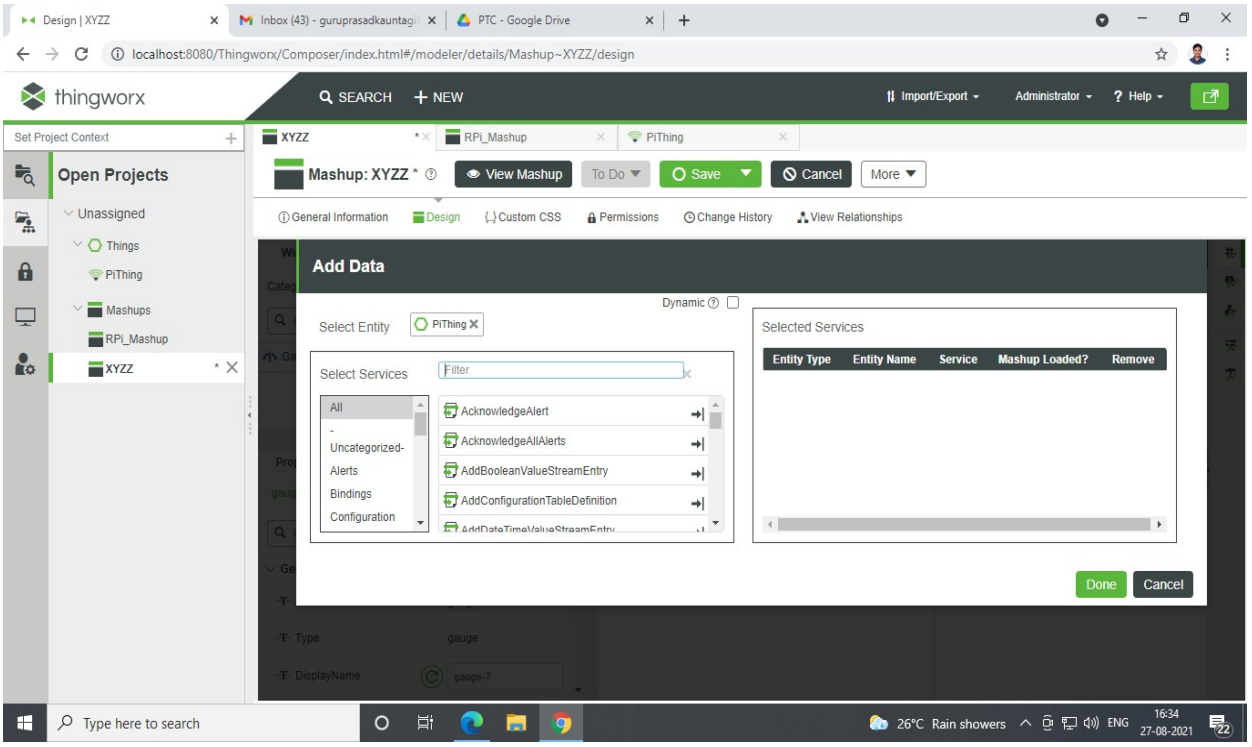
- Click on Plus sign on Data



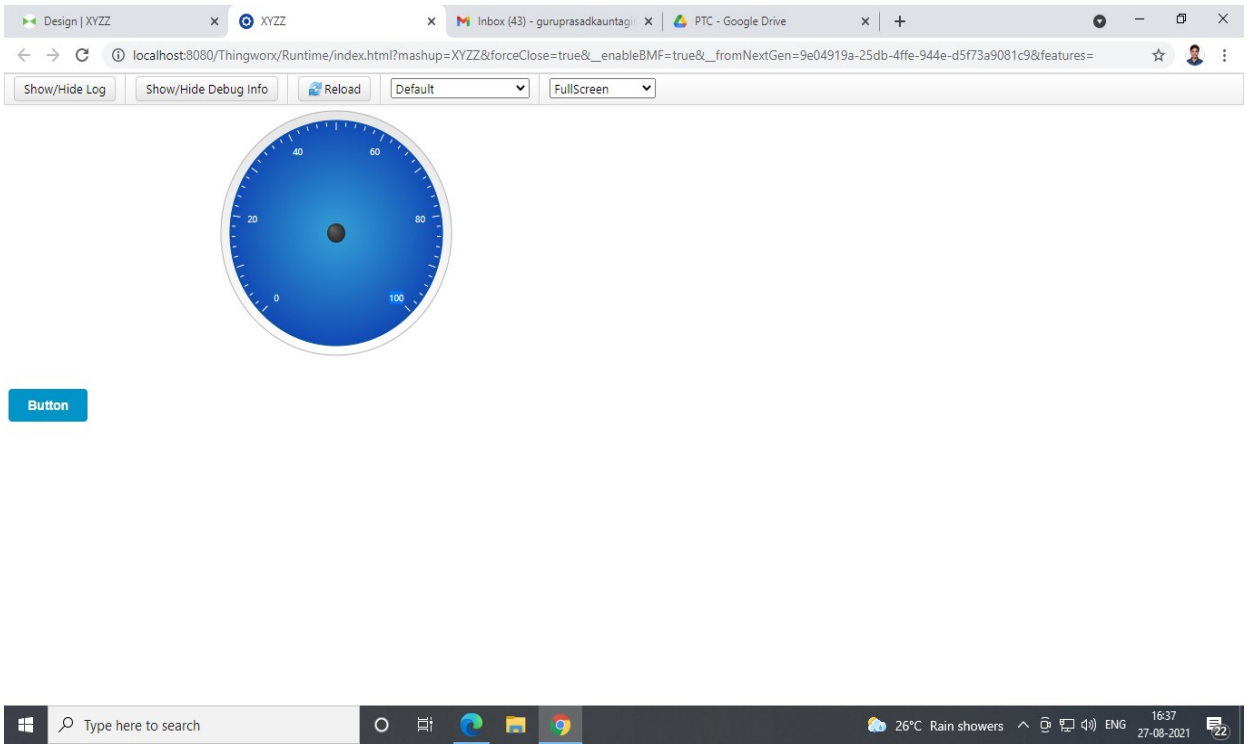
- Search for things and select your respective thing.



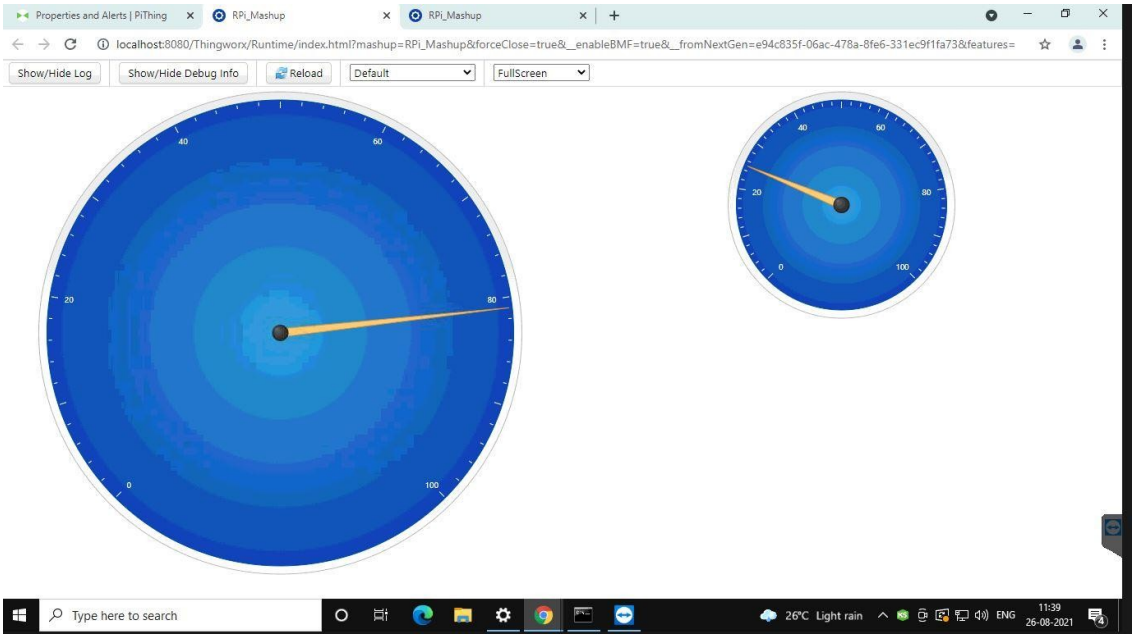
- Then select get property value services then click on done



- After clicking on done, you will redirect to the mashup screen.



- In the same way, u can create multiple mashups.

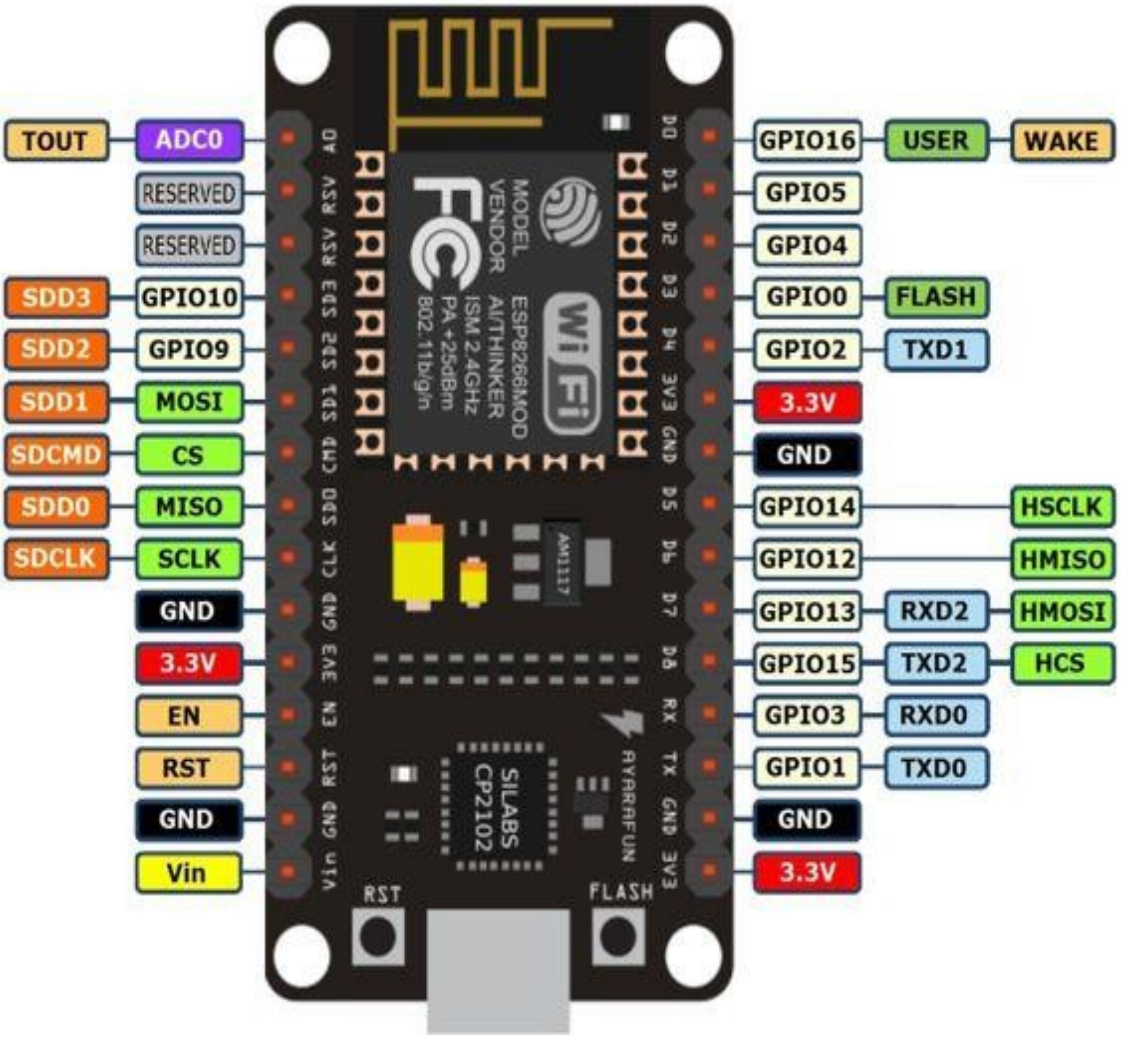


Node MCU with Thingworx

Details of the Node MCU

NODE MCU(ESP8266):

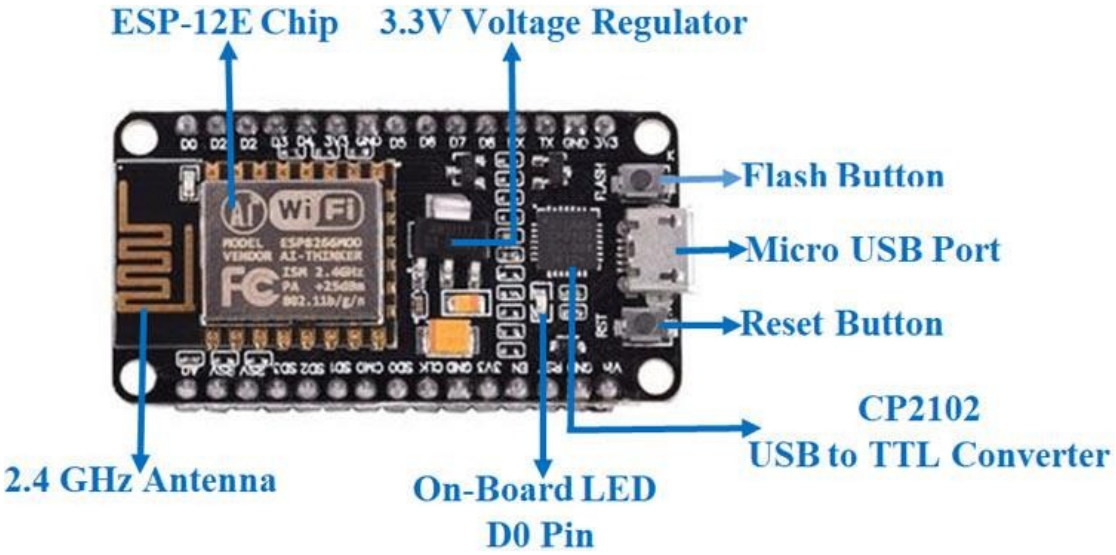
Node MCU is an open source firmware for which open source prototyping board designs are available. The name "Node MCU" combines "node" and "MCU" (micro-controller unit)



Node MCU Development Board Pinout Configuration

Pin Category	Name	Description
Power	Micro-USB, 3.3V, GND, Vin	**Micro-USB:** NodeMCU can be powered through the USB port **3.3V:** Regulated 3.3V can be supplied to this pin to power the board **GND:** Ground pins **Vin:** External Power Supply
Control Pins	**EN, RST**	The pin and the button resets the microcontroller
Analog Pin	A0	Used to measure analog voltage in the range of 0-3.3V
GPIO Pins	GPIO1 to GPIO16	NodeMCU has 16 general purpose input-output pins on its board
SPI Pins	SD1, CMD, SD0, CLK	NodeMCU has four pins available for SPI communication.
UART Pins	TXD0, RXD0,	NodeMCU has two UART interfaces, UART0 (RXD0 & TXD0) and UART1 (RXD1 & TXD1). UART1 is used to

	TXD2, RXD2	upload the firmware/program.
I2C Pins		NodeMCU has I2C functionality support but due to the internal functionality of these pins, you have to find which pin is I2C.



Applications

- Prototyping of IoT devices
- Low power battery operated applications
- Network projects

- Projects requiring multiple I/O interfaces with Wi-Fi and Bluetooth functionalities

Applications of ADXL345 Accelerometer

- Cost-sensitive, low power, motion- and tilt-sensing applications
- Mobile devices
- Gaming systems
- Disk drive protection Image stabilization
- Sports and health devices

Note: The Node MCU Development Board can be easily programmed with Arduino IDE since it is easy to use.

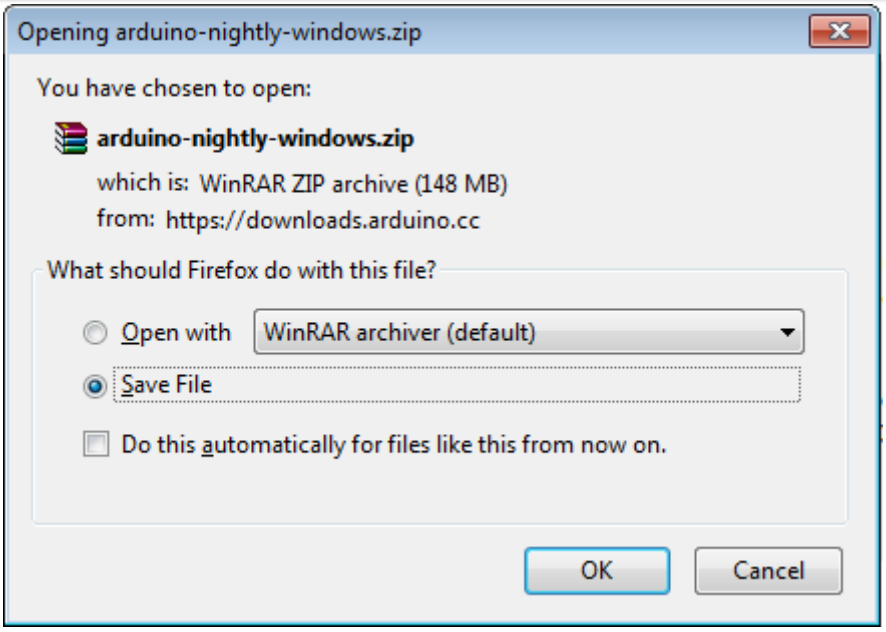
ARDUINO IDE installation

Step 1 – First you must have your Arduino board (you can choose your favourite board) and a USB cable. In case you use Arduino UNO, Arduino Duemilanove, Nano, Arduino Mega 2560, or Diecimila, you will need a standard USB cable (A plug to B plug), the kind you would connect to a USB printer as shown in the following image.

In case you use Arduino Nano, you will need an A to Mini-B cable instead as shown in the following image.

Step 2 – Download Arduino IDE Software.

You can get different versions of Arduino IDE from the Download page on the Arduino Official website. You must select your software, which is compatible with your operating system (Windows, IOS, or Linux). After your file download is complete, unzip the file.



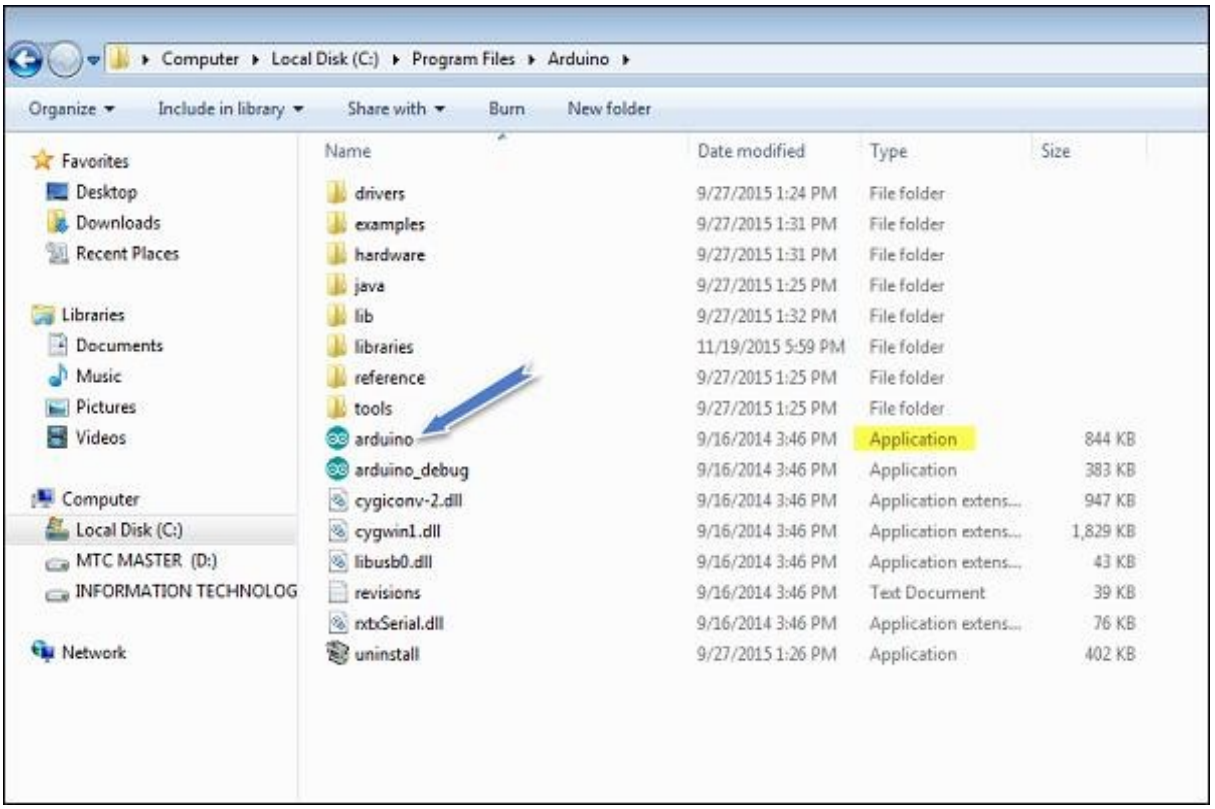
Step 3 – Power up your board.

The Arduino Uno, Mega, Duemilanove and Arduino Nano automatically draw power from either, the USB connection to the computer or an external power supply. If you are using an Arduino Diecimila, you have to make sure that the board is configured to draw power from the USB connection. The power source is selected with a jumper, a small piece of plastic that fits onto two of the three pins between the USB and power jacks. Check that it is on the two pins closest to the USB port.

Connect the Arduino board to your computer using the USB cable. The green power LED (labelled PWR) should glow.

Step 4 – Launch Arduino IDE.

After your Arduino IDE software is downloaded, you need to unzip the folder. Inside the folder, you can find the application icon with an infinity label (application.exe). Double-click the icon to start the IDE.

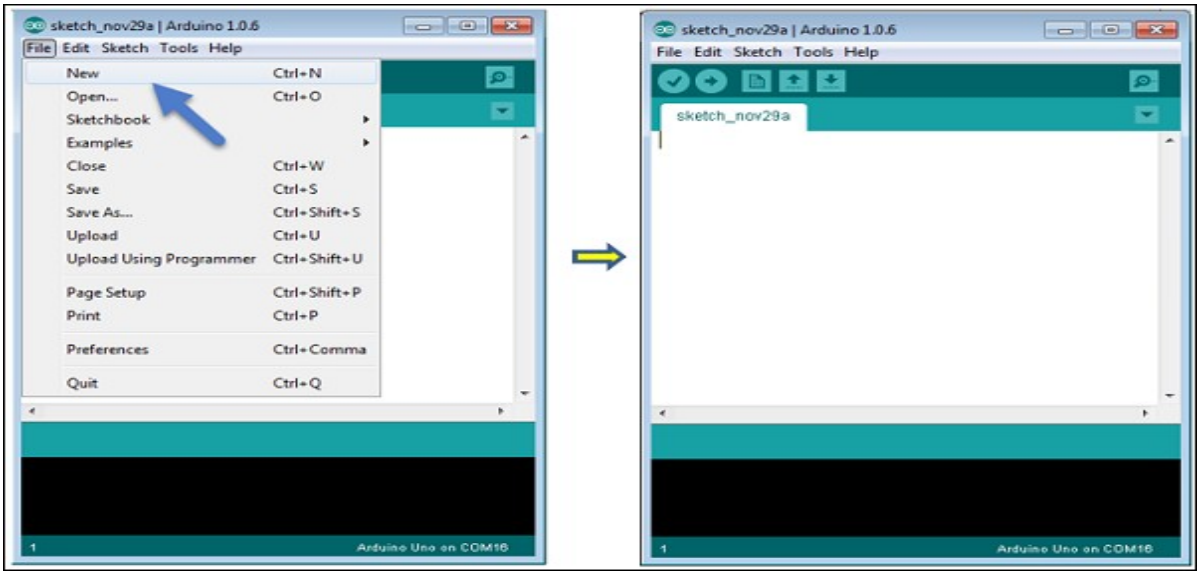


Step 5 – Open your first project.

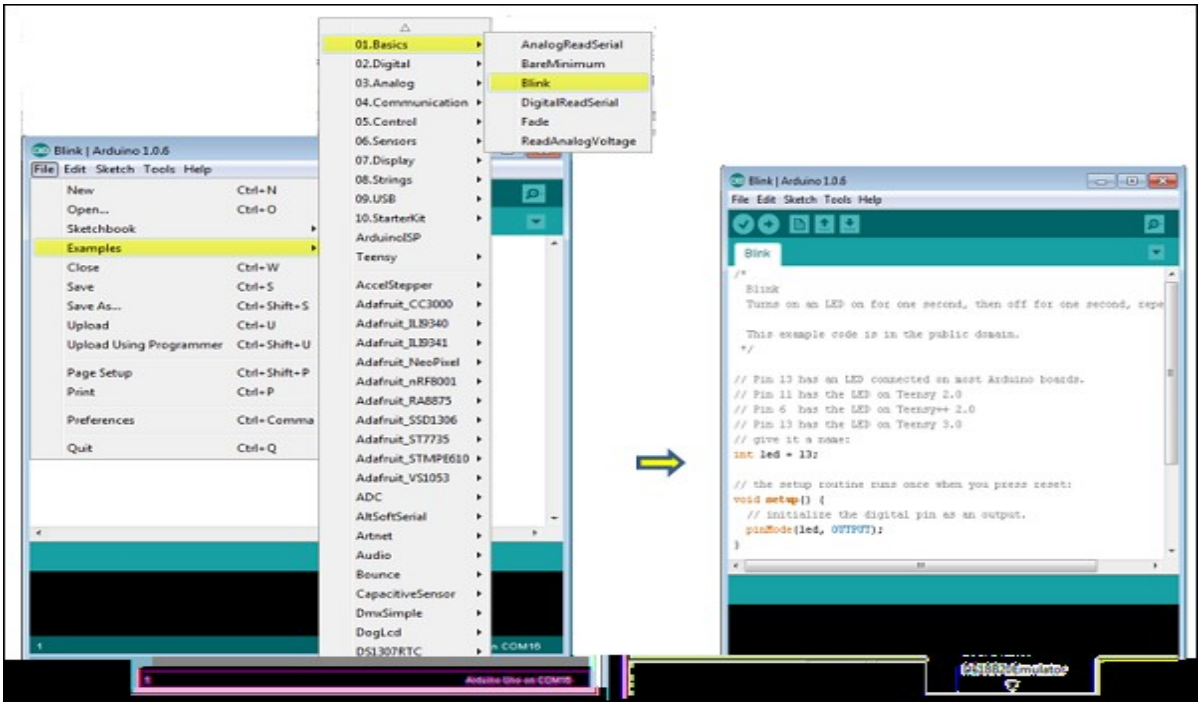
Once the software starts, you have two options –

- Create a new project.
- Open an existing project example.

To create a new project, select File → **New**.



To open an existing project example, select File → Example → Basics → Blink.

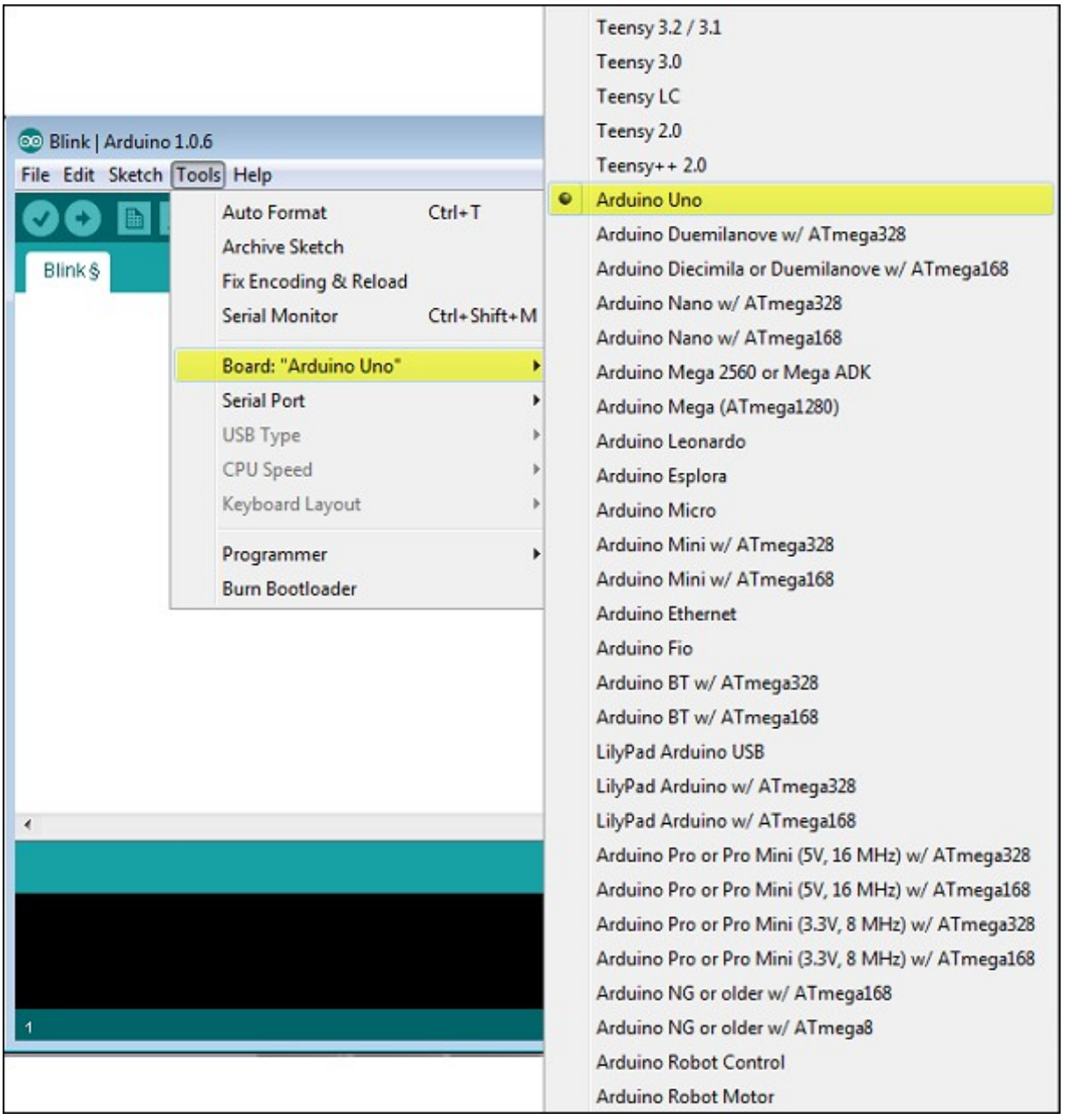


Here, we are selecting just one of the examples with the name **Blink**. It turns the LED on and off with some time delay. You can select any other example from the list.

Step 6 – Select your Arduino board.

To avoid any error while uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer.

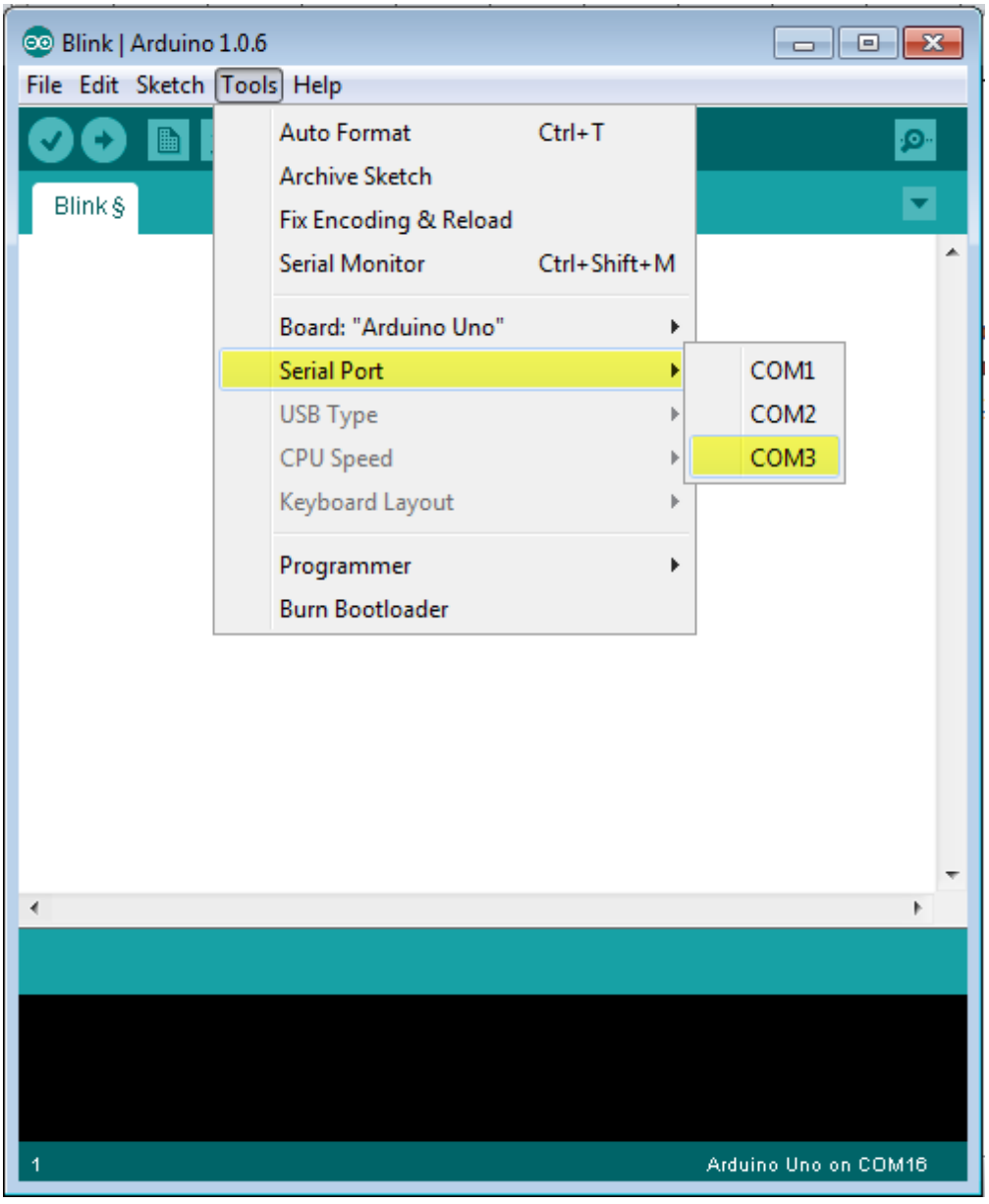
Go to Tools → Board and select your board.



Here, we have selected Arduino Uno board according to our tutorial, but you must select the name matching the board that you are using.

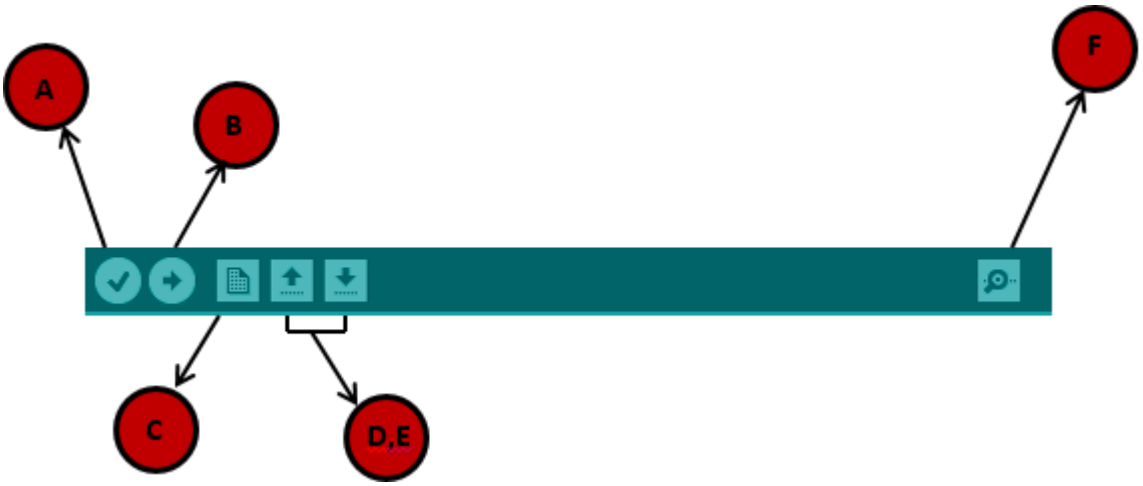
Step 7 – Select your serial port.

Select the serial device of the Arduino board. Go to **Tools → Serial Port** menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu, the entry that disappears should be of the Arduino board. Reconnect the board and select that serial port.



Step 8 – Upload the program to your board.

Before explaining how we can upload our program to the board, we must demonstrate the function of each symbol appearing in the Arduino IDE toolbar.



A – Used to check if there is any compilation error.

B – Used to upload a program to the Arduino board.

C – Shortcut used to create a new sketch.

D – Used to directly open one of the example sketch.

E – Used to save your sketch.

F – Serial monitor used to receive serial data from the board and send the serial data to the board.

Now, simply click the "Upload" button in the environment. Wait a few seconds; you will see the RX and TX LEDs on the board, flashing. If the upload is successful, the message "Done uploading" will appear in the status bar.

Detail study of DHT11, ADXL345, Ultrasonic (HSCR 04), Force sensors

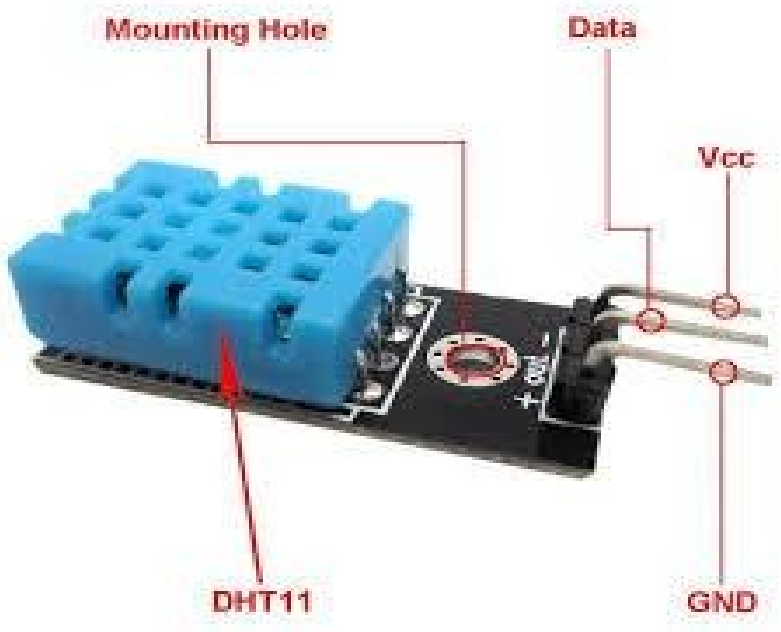
DHT11 Sensor:

DHT11 is a low-cost digital sensor for sensing temperature and humidity

Working Principle of DHT11 Sensor:

DHT11 sensor consists of a capacitive humidity sensing element and a thermistor for sensing temperature. The humidity sensing capacitor has two electrodes with a moisture holding substrate as a dielectric between them. Change in the capacitance value occurs with the change in humidity levels. The IC measure, process this changed resistance values and change them into digital form.

For measuring temperature this sensor uses a Negative Temperature coefficient thermistor, which causes a decrease in its resistance value with increase in temperature. To get larger resistance value even for the smallest change in temperature, this sensor is usually made up of semiconductor ceramics or polymers.



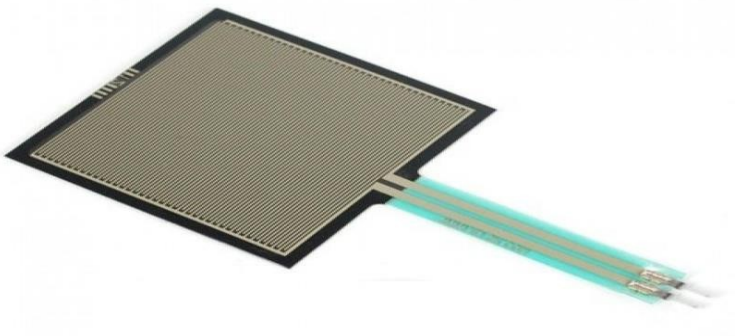
DHT11 sensor has four pins- VCC, GND, Data Pin and a not connected pin. A pull-up resistor of 5k to 10k ohms is provided for communication between sensor and micro-controller.

Force Sensor:

when subjected to force, can change their resistance values. These materials were known as Force-Sensing Resistors. These materials are used to produce a sensor that can measure the Force. A Force Sensor is a sensor that helps in measuring the amount of force applied to an object. By observing the amount of change in the resistance values of force-sensing resistors, the applied force can be calculated

Working Principle

The general working principle of Force Sensors is that they respond to the applied force and convert the value into a measurable quantity. There are various types of Force Sensors available in the market based on various sensing elements. Most of the Force Sensors are designed using Force-Sensing Resistors. These sensors consist of a sensing film and electrodes.



The working principle of a Force-sensing resistor is based on the property of 'Contact Resistance'. Force-sensing resistors contain a conductive polymer film that changes its resistance in a predictable manner when force is applied on its surface. This film consists of, sub-micrometres sized, electrically conducting and non-conducting particles arranged in a matrix. When force is applied to the surface of this film, the micro sized particle touches the sensor electrodes, changing the resistance of the film. The amount of change caused to the resistance values gives the measure of the amount of force applied.

To improve the performance of the Force-Sensing resistors various efforts are being made with multiple different approaches such as, to minimize the drift of polymer various electrode configurations are being tested, testing with sensor by replacing the polymer with new materials such as carbon nanotubes, etc.

ADXL345:

The **ADXL345** is a small, low power, complete 3-axis MEMS accelerometer modules with both I2C and SPI interfaces. The ADXL345 board feature on-board 3.3V voltage regulator and level shifter, which makes it simple to interface with 5V microcontrollers such as the Arduino.

ADXL345 Module Pin Configuration

Pin Name	*Pin Configuration*

GND	*Ground Pin*
VCC	*Power Supply pin (3V to 6V)*
CS	*Chip Select Pin*
INT1	*Interrupt 1 Output*
INT2	*Interrupt 2 Output*
SDO	*Serial Data Output*
SDA	*Serial Data Input & Output*
SDL	*Serial Communication Clock*

This ADXL345 Accelerometer module consists of an ADXL345 Accelerometer IC, Voltage Regulator IC, Level Shifter IC, resistors, and capacitors in an integrated circuit. Different manufacturers use a different voltage regulator IC.

ADXL345 IC from Analog Devices is the brain of this module. The ADXL345 is a small, thin, low power, complete 3-axis accelerometer with signal conditioned

voltage outputs. The product measures acceleration with a minimum full-scale range of ±16g.

How to Use the ADXL345 Accelerometer Module?

ADXL345 Accelerometer module consists of 8 pins. Using the ADXL345 module with a microcontroller is very easy. Connect VCC and GND pins to 5V and GND pins of Microcontroller. Also, connect SCL and SDA pins to the SCL and SDA pins of Arduino.

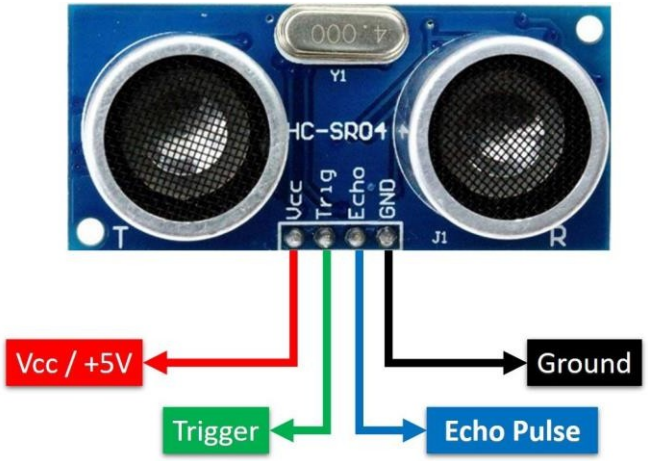
The basic structure of the accelerometer consists of fixed plates and moving plates. When the acceleration is applied on an axis capacitance between fixed plates and moving plates is changed. This results in a sensor output voltage amplitude, which is proportional to the acceleration.

Applications of ADXL345 Accelerometer

- Cost-sensitive, low power, motion- and tilt-sensing applications
- Mobile devices
- Gaming systems
- Disk drive protection
- Image stabilization
- Sports and health devices

Ultrasonic Sensor (HC-SR04):

An ultrasonic sensor is an electronic device that measures the distance of a target object by emitting ultrasonic sound waves, and converts the reflected sound into electrical signal.



Power Supply	5V DC
Working Current	15 mA
Working Frequency	40 kHz
Maximu	4 meters
Minimum Range	2 cm

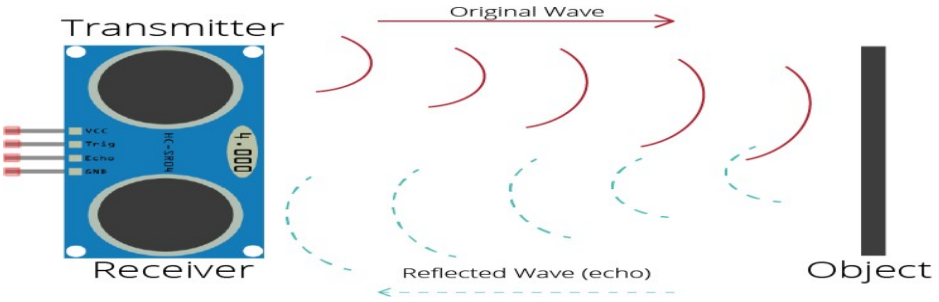
Measuring Angle	15º
Resolution	0.3 cm
Trigger Input Signal	10uS TTL pulse
Echo Output Signal	TTL pulse proportional to the distance range
Dimensions	45mm x 20mm x 15mm

The ultrasonic sensor uses sonar to determine the distance to an object. Here's how it works:

1. The ultrasound transmitter (trig pin) emits a high-frequency sound (40 kHz).

2. The sound travels through the air. If it finds an object, it bounces back to the module.

3. The ultrasound receiver (echo pin) receives the reflected sound (echo).



Taking into account the sound's velocity in the air and the travel time (time passed since the transmission and reception of the signal) we can calculate the distance to an object. Here's the formula

distance to an object = ((speed of sound in the air) *time)/2

speed of sound in the air at 20ºC (68ºF) = 343m/s

Connection of Node MCU to all sensors

Sensor Connections (Pin configuration) with Node MCU:

1. DHT 11

Sensor Pin	Positive	Negative	Output
Node MCU pin	3.3V	Gnd	D6

2. ADXL 345

Sensor Pin	VCC	GND	SCL	SDA
Node MCU pin	3.3V	Gnd	D1	D2

3. US HC-SR04

Sensor Pin	VCC	GND	Trig	Echo
Node MCU pin	Vin	Gnd	D4	D3

4. Force

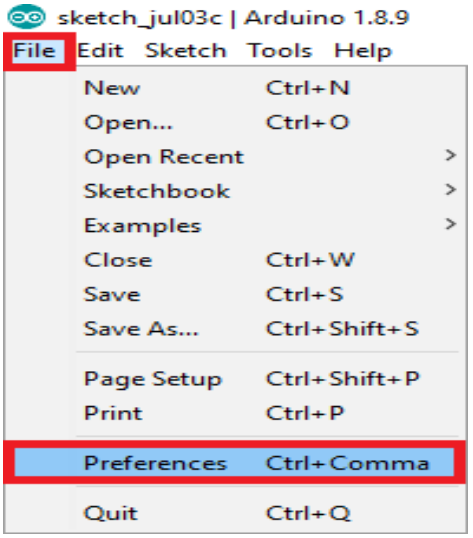
Sensor Pin	Terminal1	Terminal2
Node MCU pin	AO	gnd

Selection of ESP8266 & Port

Install ESP8266 Add-on in Arduino IDE

To install the ESP8266 board in your Arduino IDE, follow these next instructions:

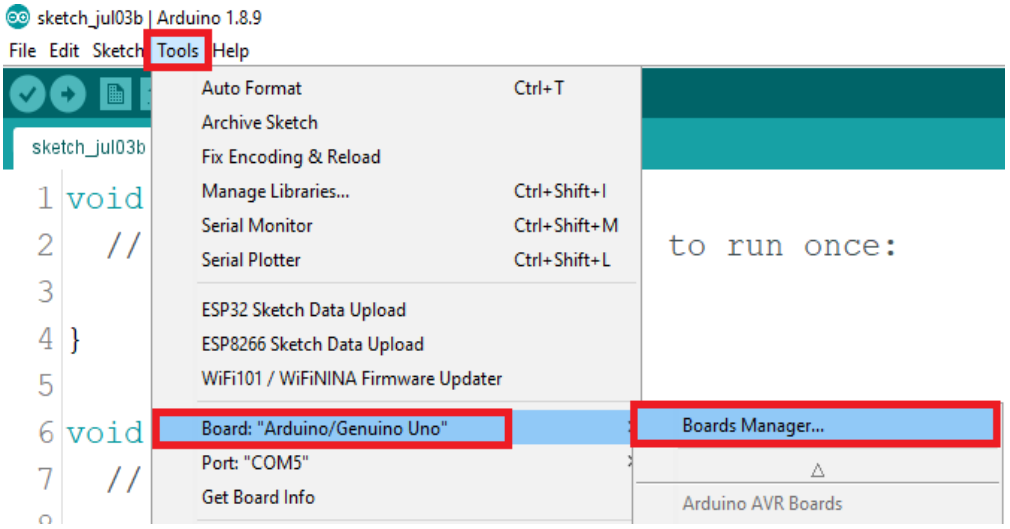
1. In your Arduino IDE, go to **File**> **Preferences**



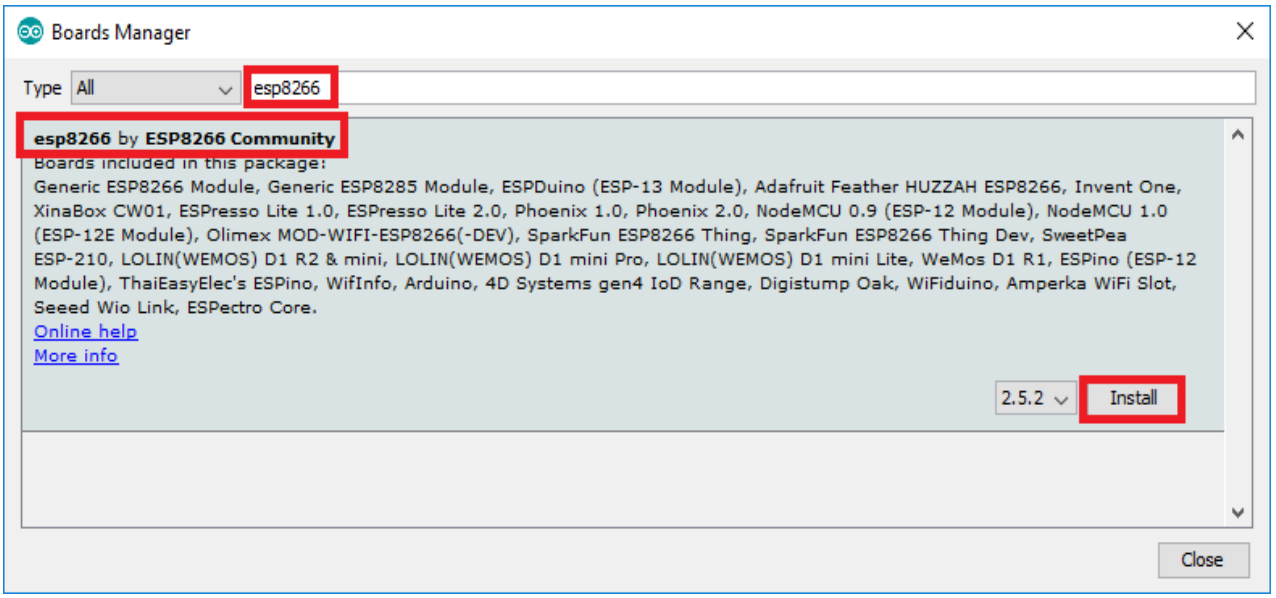
2. Enter **http://arduino.esp8266.com/stable/package_esp8266com_index.json**

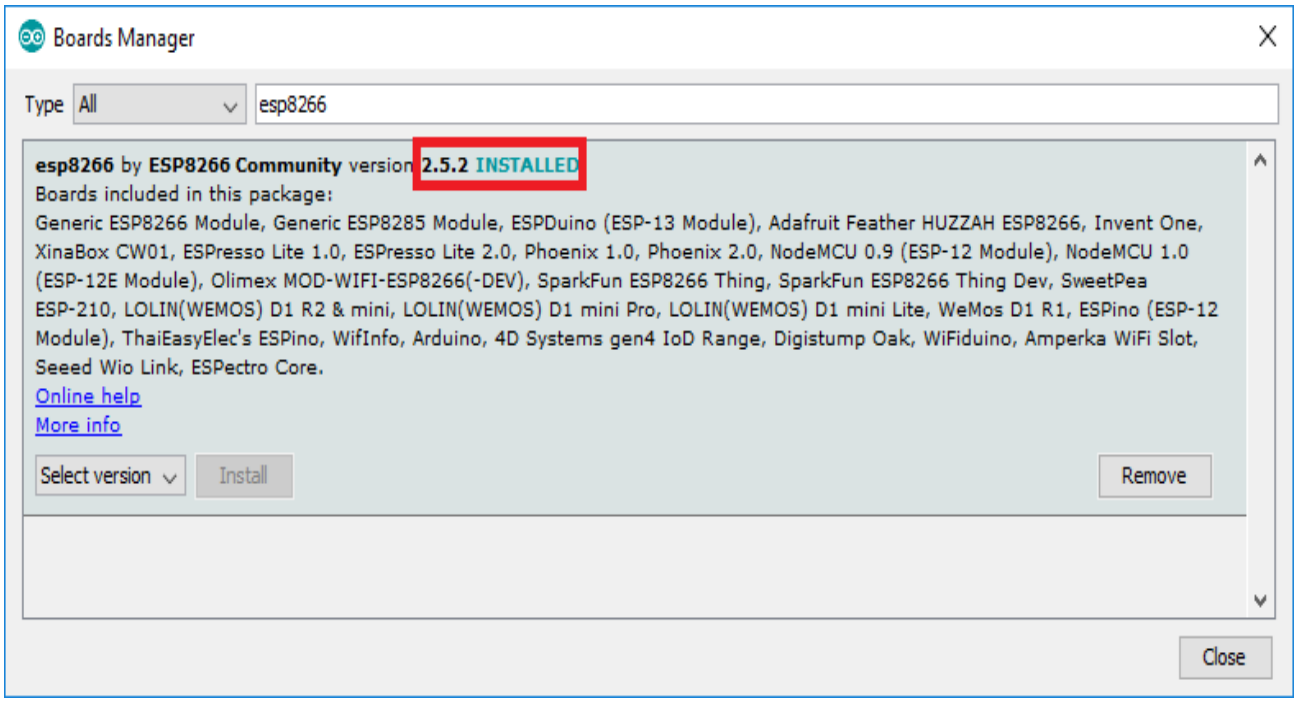
Into the “Additional Boards Manager URLs” field as shown in the figure below. Then, click the “OK” button:

3. Open the Boards Manager. Go to **Tools** > **Board** > **Boards Manager…**



4. Search for **ESP8266** and press install button for the "**ESP8266 by ESP8266 Community** ":



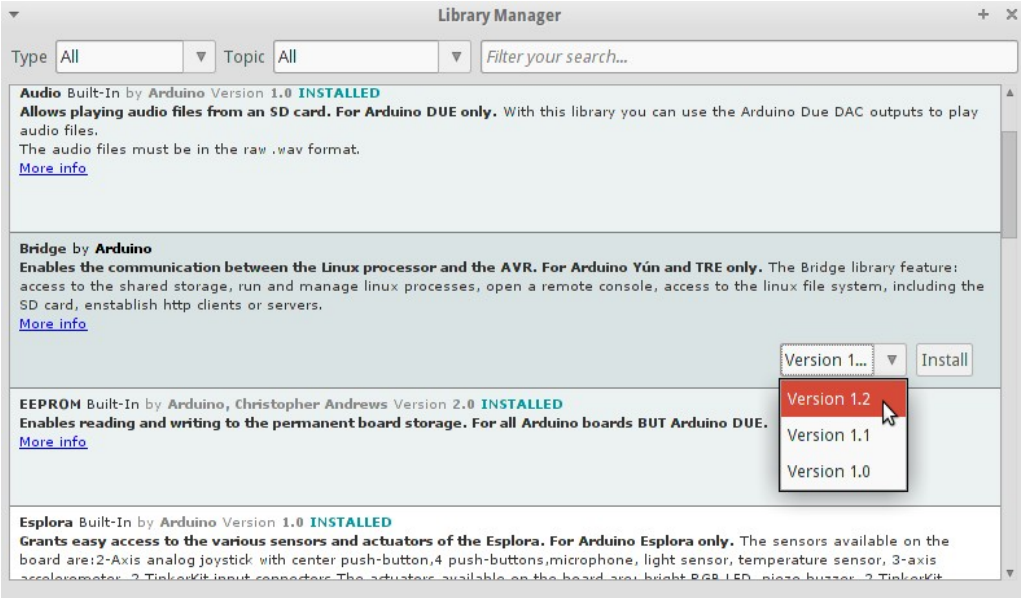


5. That's it. It should be installed after a few seconds.

Installation of all Sensors Libraries

How to add library files for different sensors in Arduino ide platform:

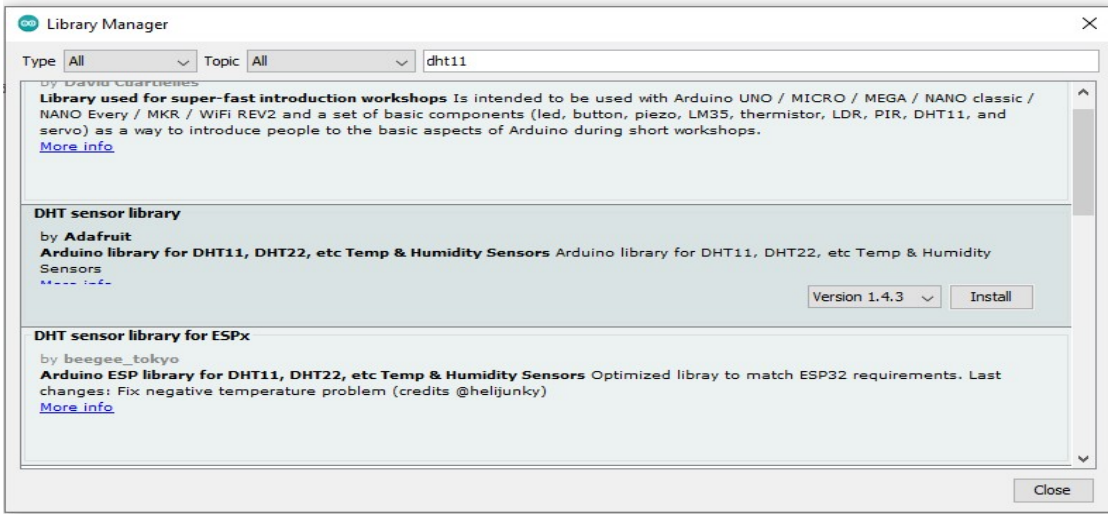
To install a new library into your Arduino IDE you can use the Library Manager (available from IDE version 1.6.2). Open the IDE and click to the "Sketch" menu and then Include Library > Manage Libraries.



DHT 11 sensor library files

⑩ Install DHT sensor file

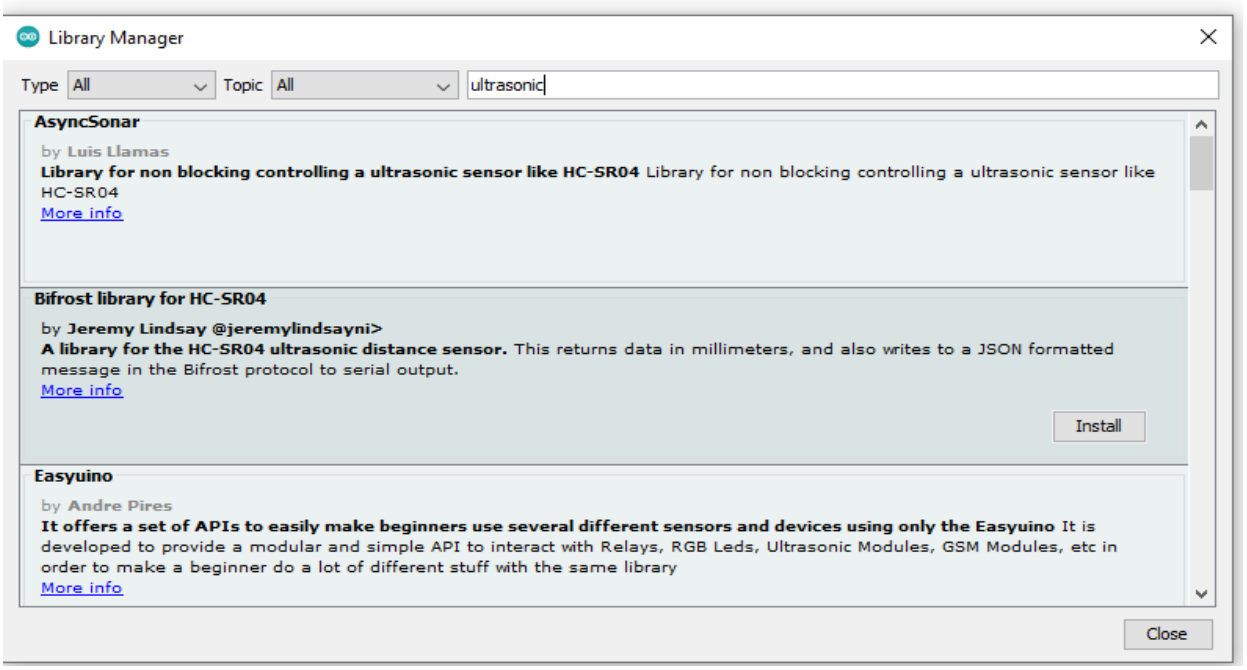
⑩ Install DHT sensor library for ESPx



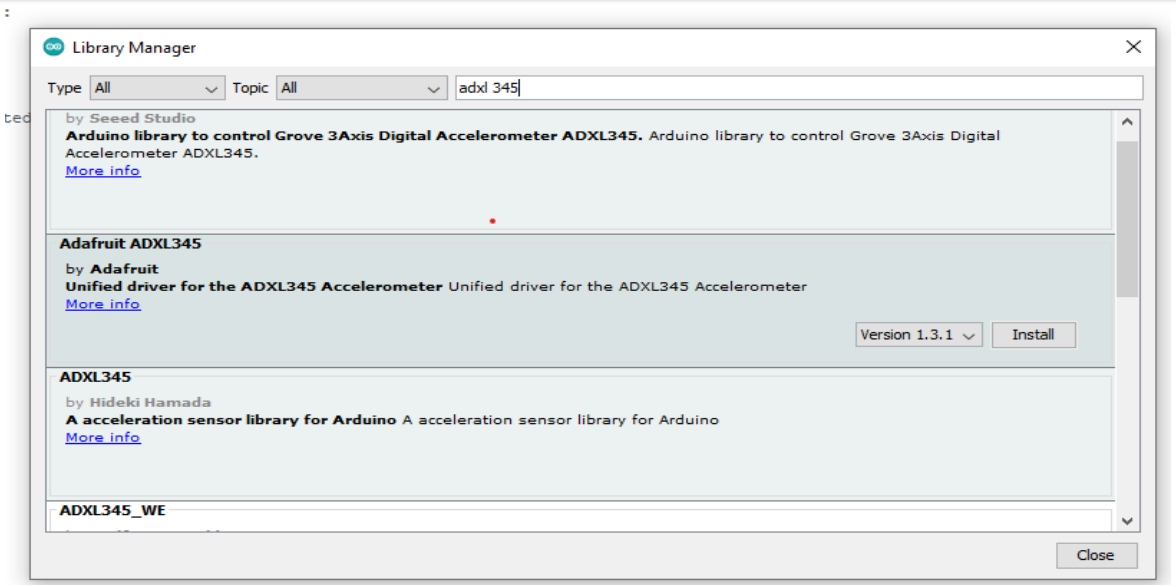
Ultrasonic sensor library files

⑩ Install Asyncsonar file

⑩ Install Bifrost for HC-SR04 file



ADXL 345 sensor library files



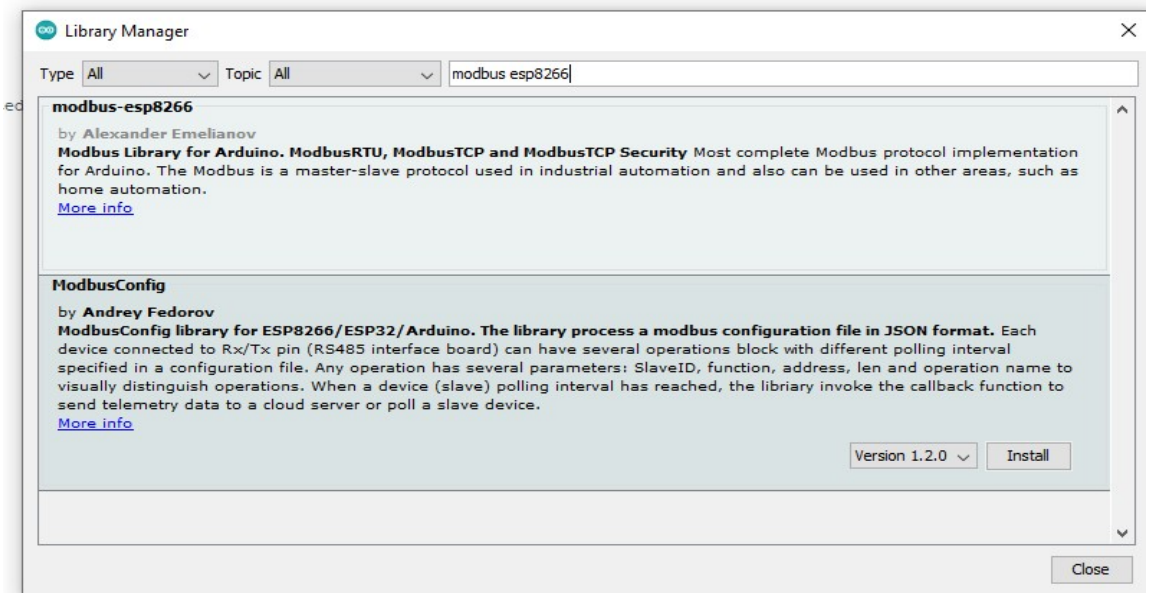
❿ Install Ad fruit ADXL345 file

❿ Install ADXL345 file

MODBUS protocol library files

❿ Install modbus-esp8266 file

❿ Install ModbusConfig file



Compilation and Downloading Code

Now Download Final code in Arduino IDE & Check your WIFI routers' ID/Password.

In Arduino IDE, first Compile & Download the Code.

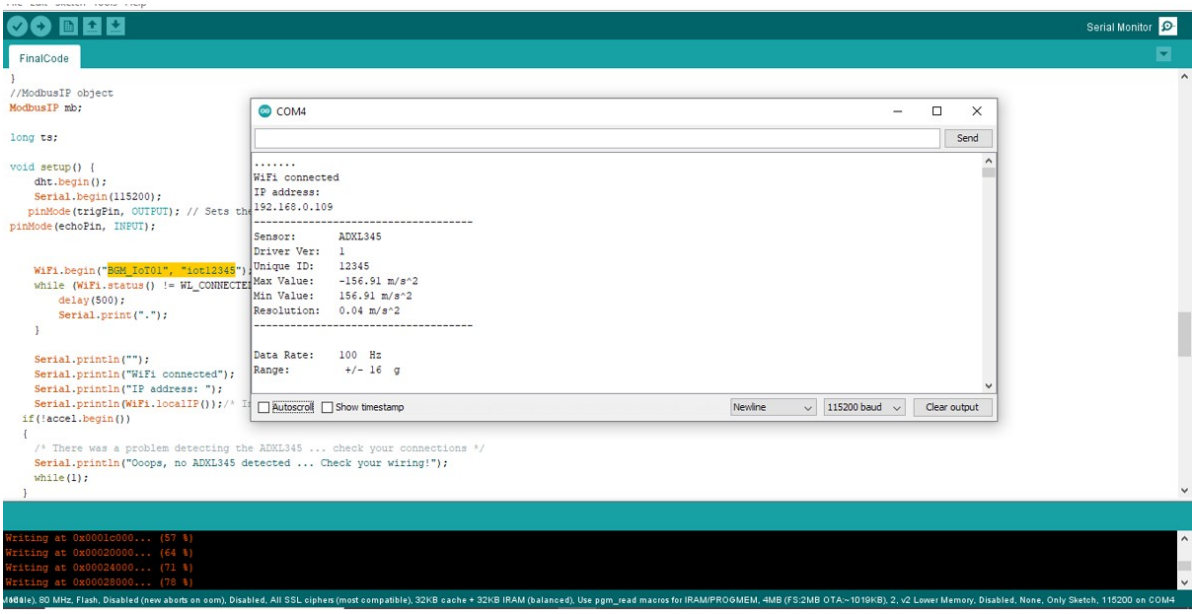
Serial monitoring of live values of all the sensors

After downloading, it will take few moments to upload in Node MCU.

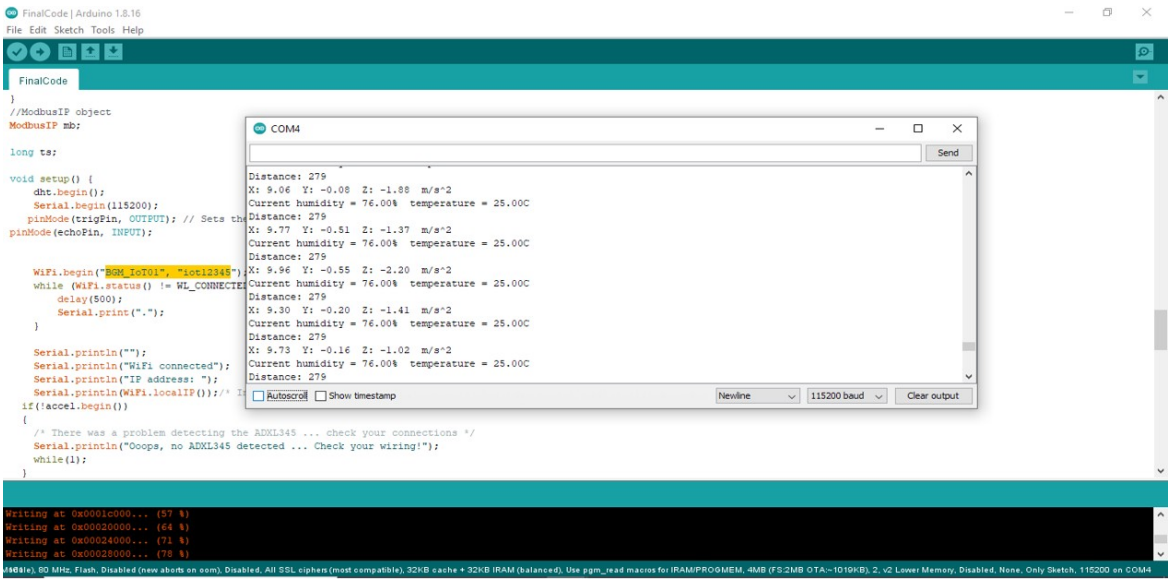
Wait a few seconds; you will see the RX and TX LEDs on the board, flashing. If the upload is successful, the message "Done uploading" will appear in the status bar.

Serial monitor used to receive serial data from the board and send the serial data to the board.
You will get IP address of Node MCU here, as well as all the live sensor values.

Node MCU IP address:



All Sensor values connected with Node MCU:



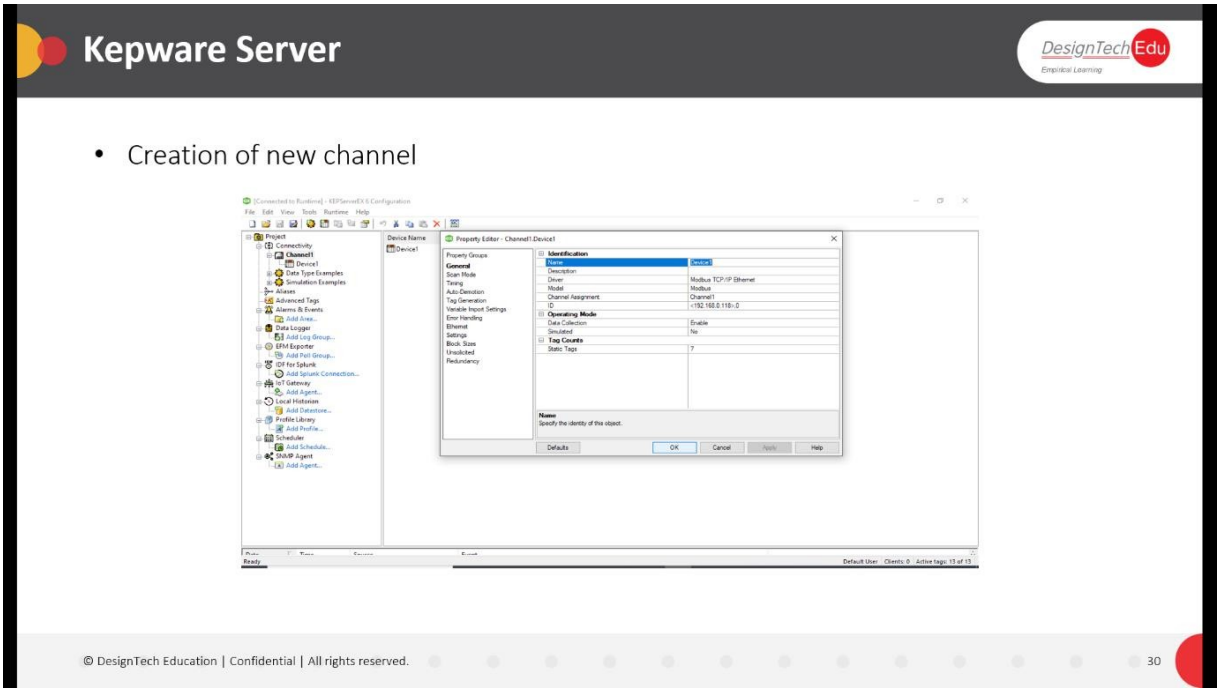
Connection of Kepware server

KEPServerEX is the industry's leading connectivity platform that provides a single source of industrial automation data to all of your applications. The platform design allows users to connect, manage, monitor, and control diverse automation devices and software applications through one intuitive user interface.

KEPServerEX provides critical technical features that enable accessibility, aggregation, optimization, connectivity, security, and diagnostics.

KEPServerEX supports the broadest range of drivers available, including current and legacy devices across various verticals, a variety of wired and wireless network mediums, and connectivity to databases, custom software applications, and other OPC servers.

Now add Node MCU to the Kepware server along with all available sensors.

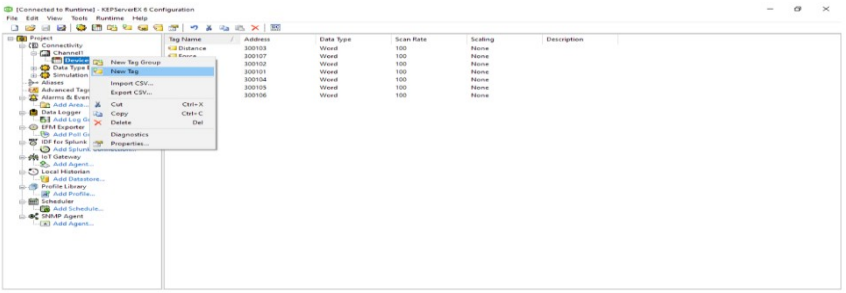


Create new channel in Kepware server Project.
Use Modbus TCP/IP Ethernet as a driver, & provide Node MCU IP address.
Add new device & add new tag based on sensors available.

In a new devices addition of tags

DesignTech Edu
Empirical Learning

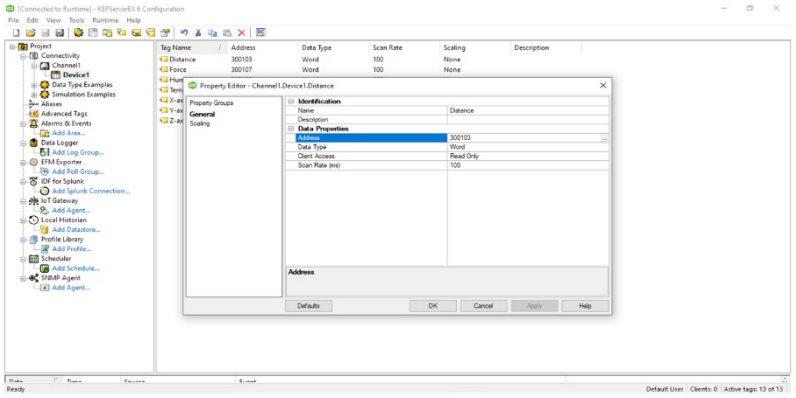
- Adding a new device and in device add new Tag



Address for a tags

DesignTech Edu
Empirical Learning

- Adding Address for a tags

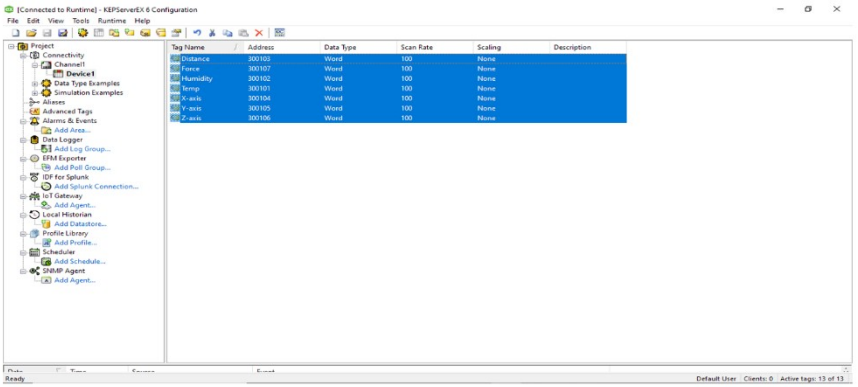


Every sensor properties has unique data type & address.

Adding tags in tag table

DesignTech Edu
Empirical Learning

- Add all the tags in a device



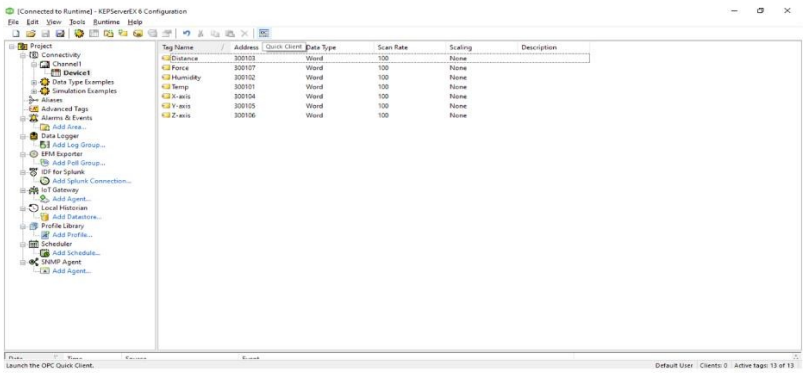
 33

Add all sensors properties & provide address & data type as per Final Code.

Quick client

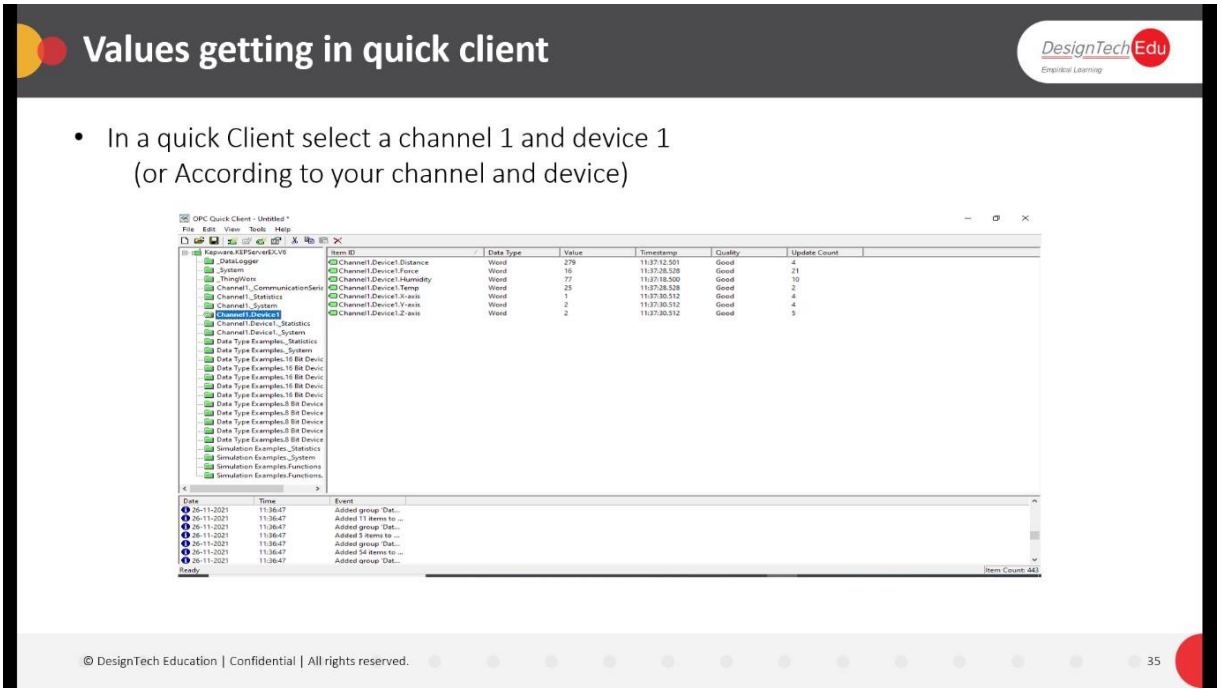
DesignTech Edu
Empirical Learning

- Go to Quick client

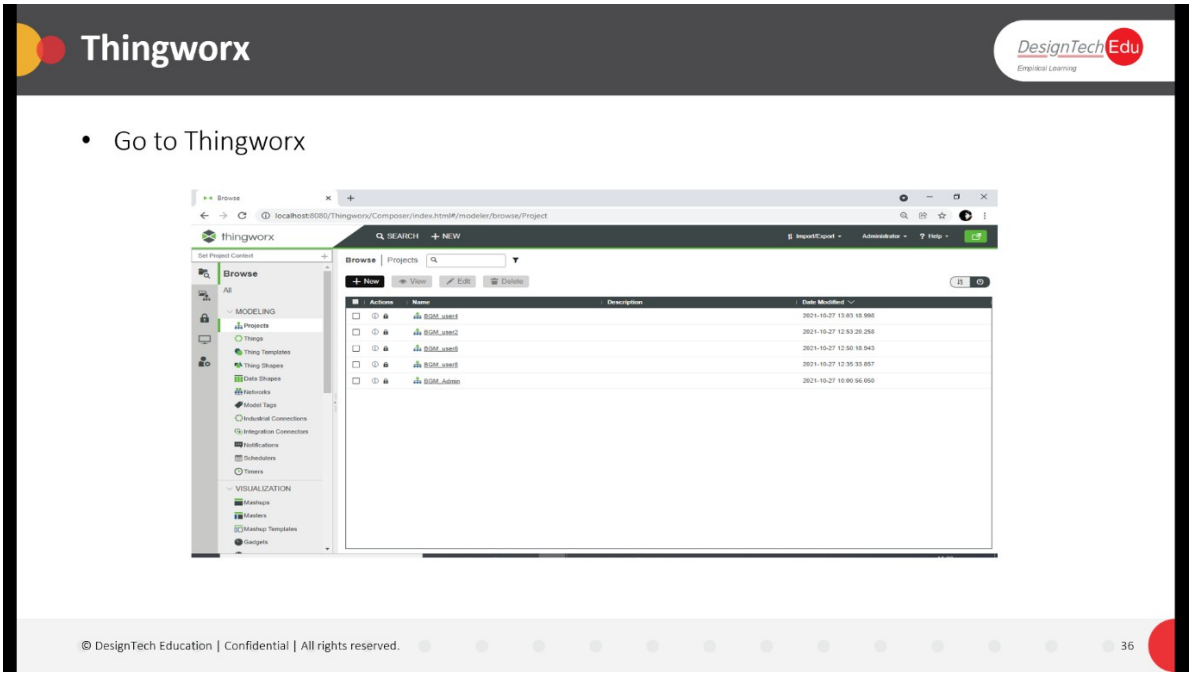


 34

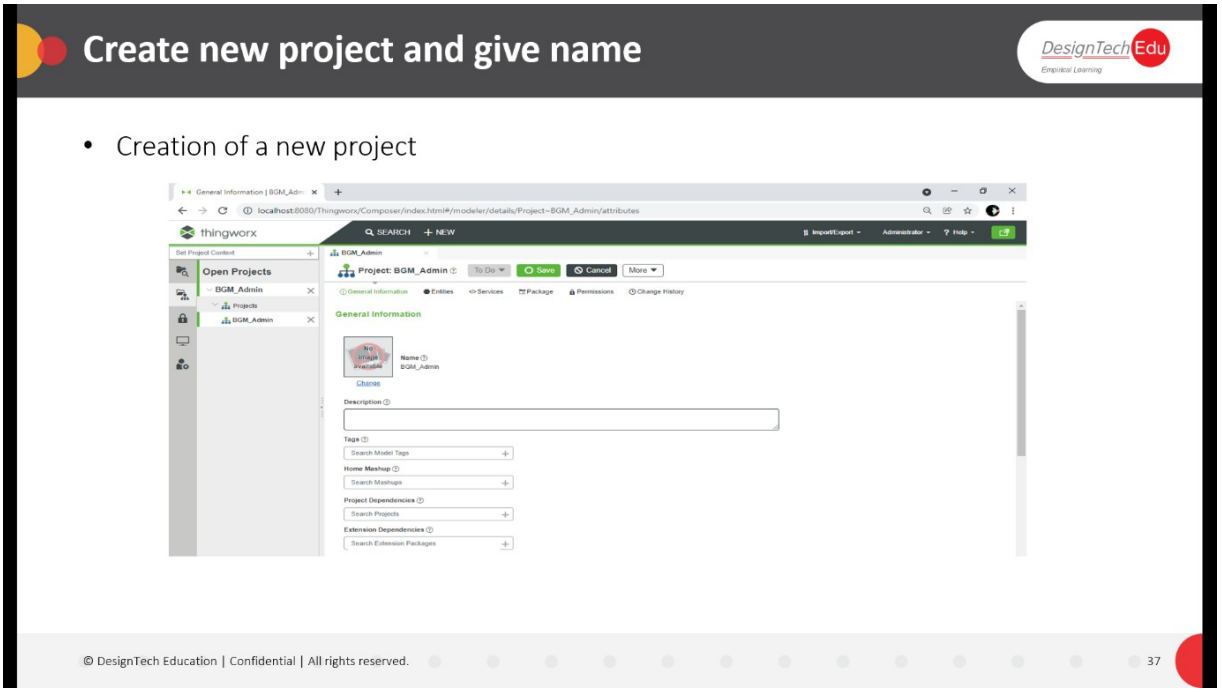
The OPC Quick Client assists in the testing and development of the OPC Data Access 1.0 and 2.0 servers. It supports both local and remote OPC server connections. Active items will automatically receive data change notifications. The current value and quality will
be displayed in the Item View.



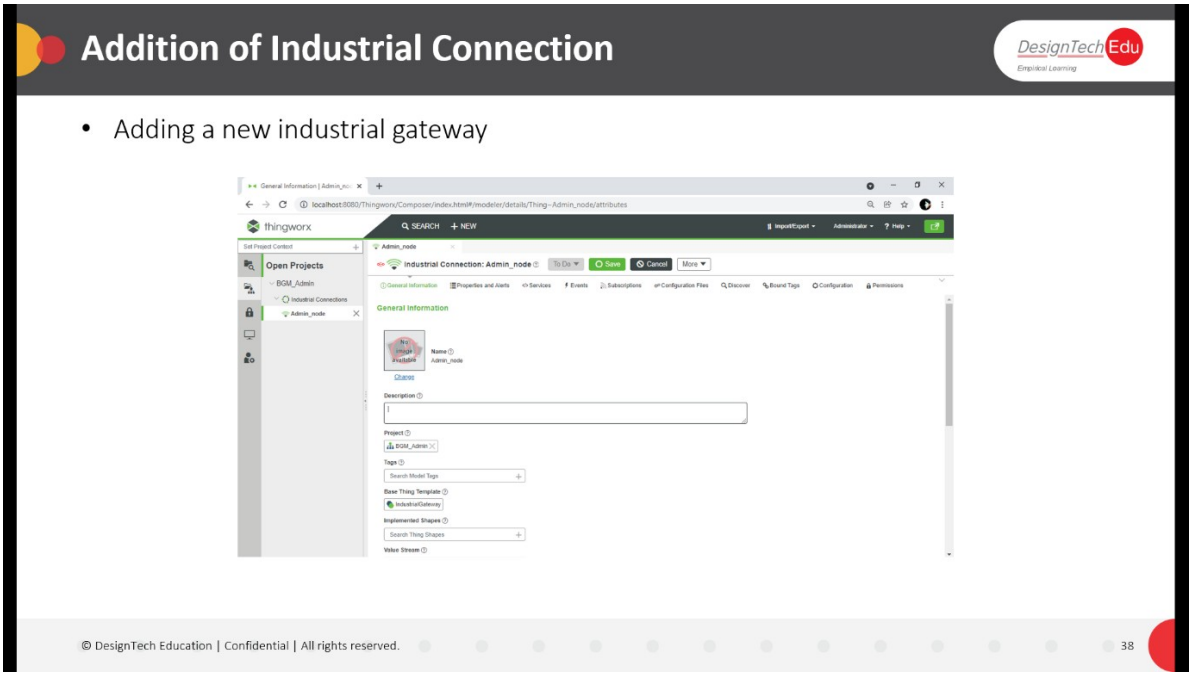
Select the same Channel & Device as shown in above slide to view all live values.



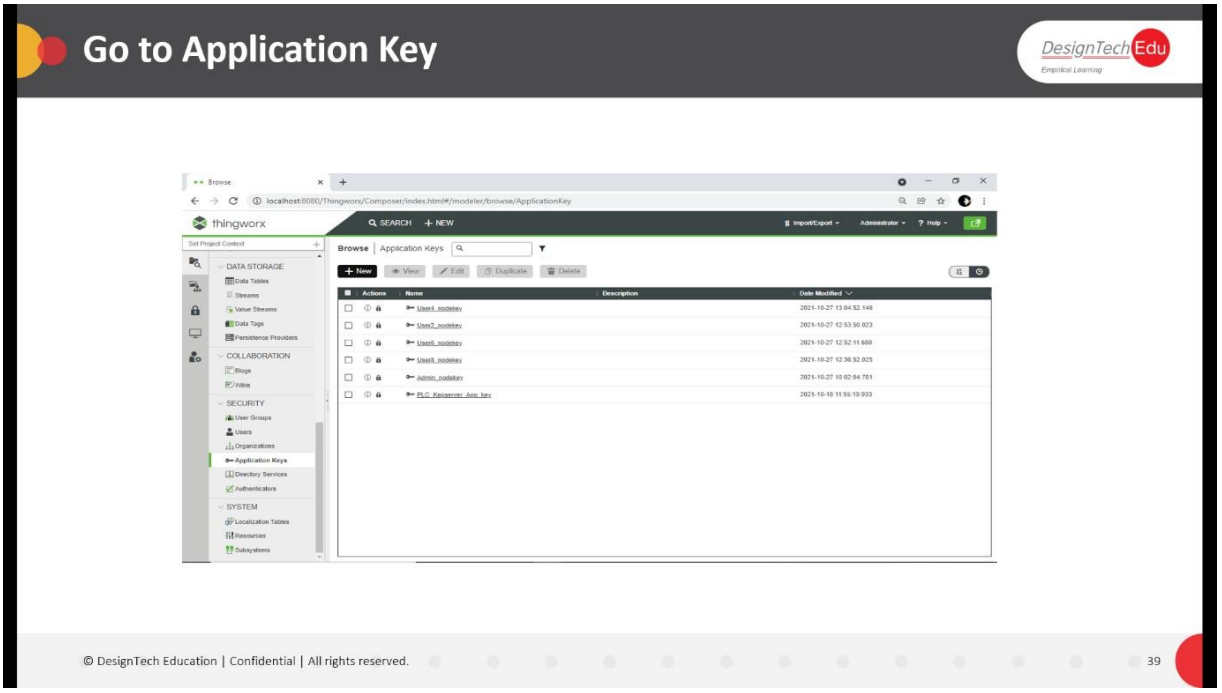
Now go to Thingworx to Create New project & connect with Kepware server to update live values of all sensors connected with Node MCU.



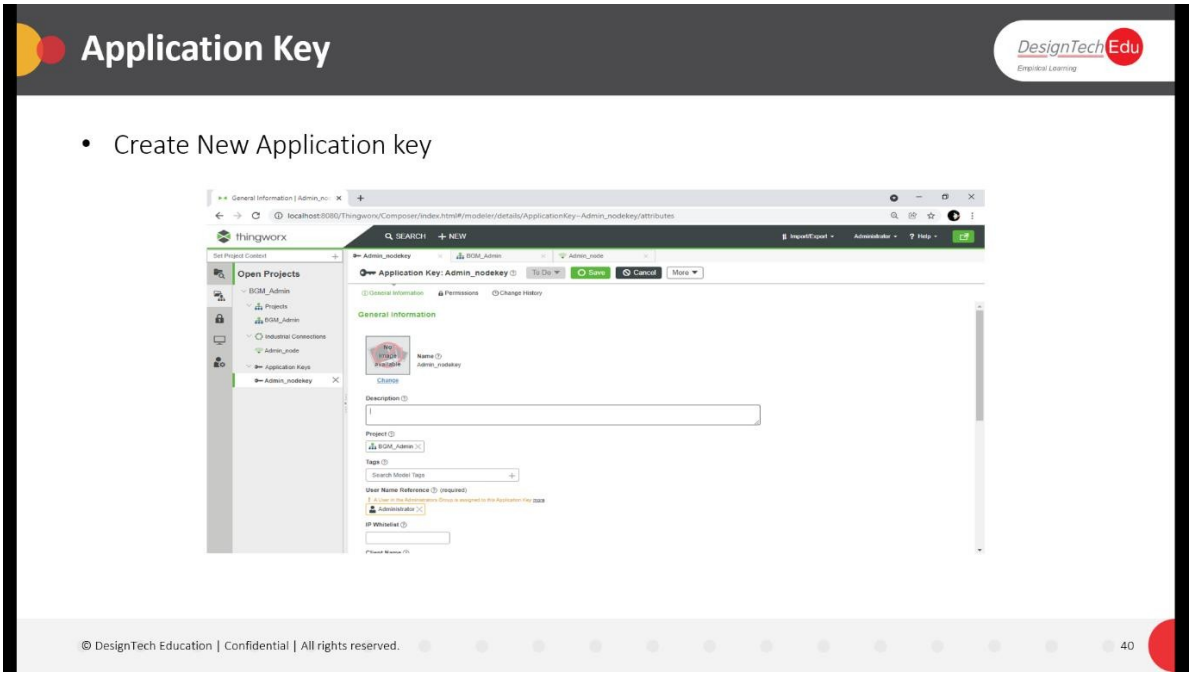
Go to Modeling>Project.
Click on New to create new Project.
Give name as BGM_Admin.



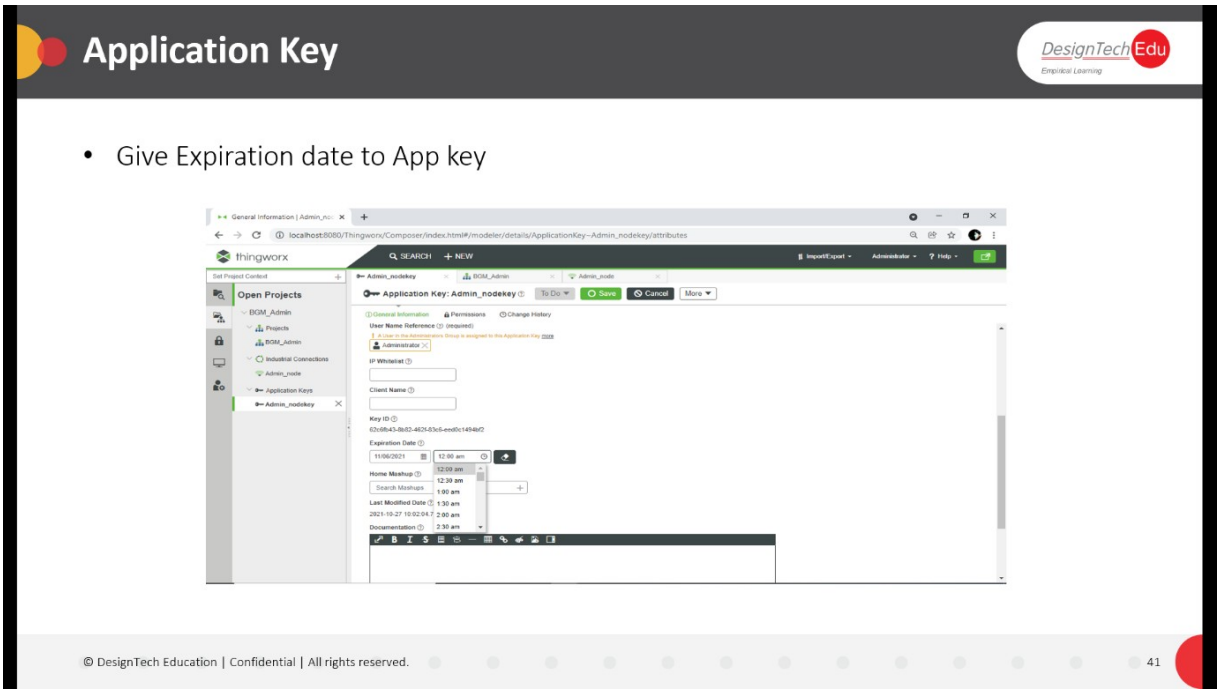
Add a new Industrial connection (Thing) & provide name as Admin_node. Assign Project created earlier & Assign Industrial Gateway as Template.



Now go to Application Key & Click “New” to add new application key.



Create New Application key & give name as Admin_nodekey.
Assign Project same project.
Assign User to the application key.

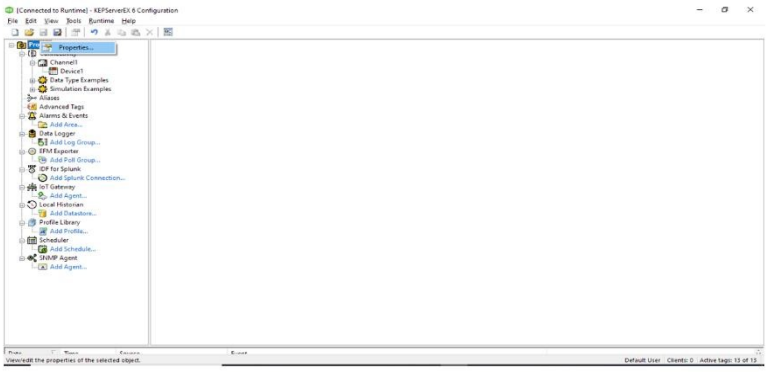


Provide Expiration date to the application key & Save the settings.

Kepware Server Project

DesignTech Edu
Empirical Learning

- In Kepware server go to project>right click to Properties



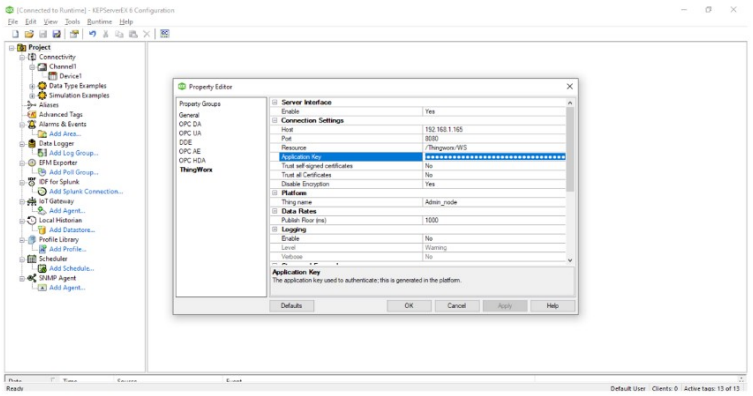
 42

Now, go to Kepware server's Project properties.

Thingworx properties

DesignTech Edu
Empirical Learning

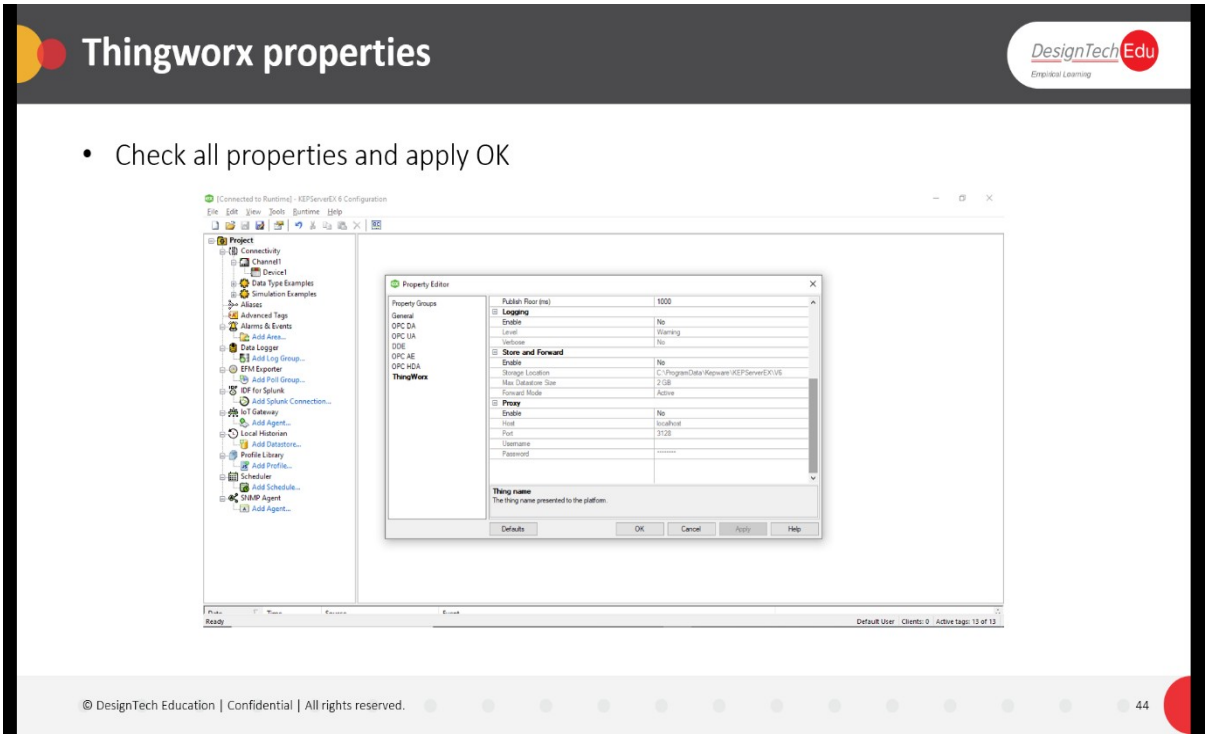
- In properties go to Thingworx
- Provide Host (IP address), Port, Application key and thing name



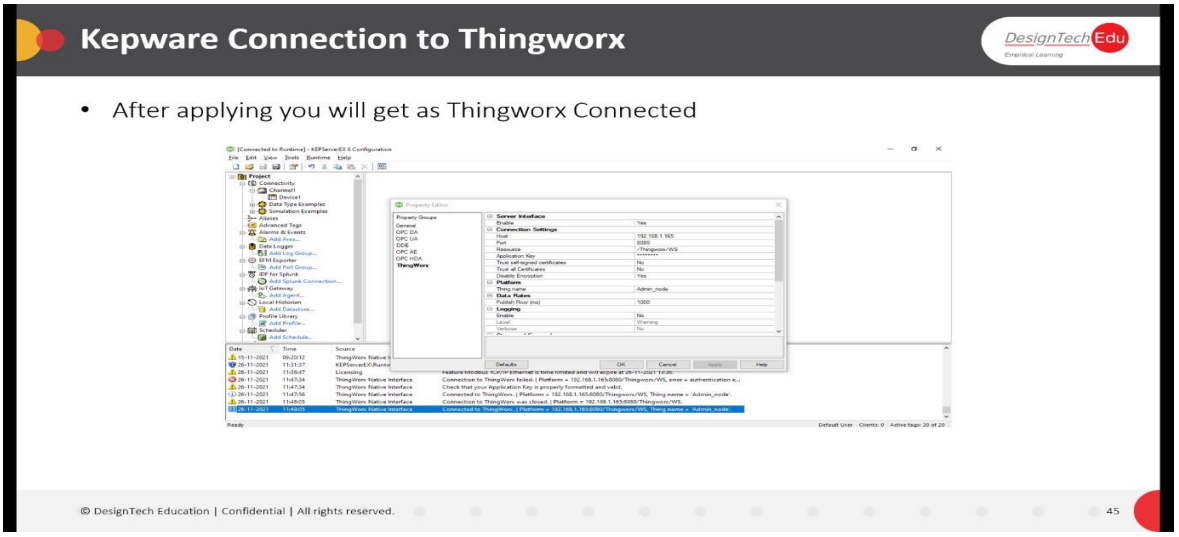
 43

Select Thingworx & provide details as shown above. i.e.

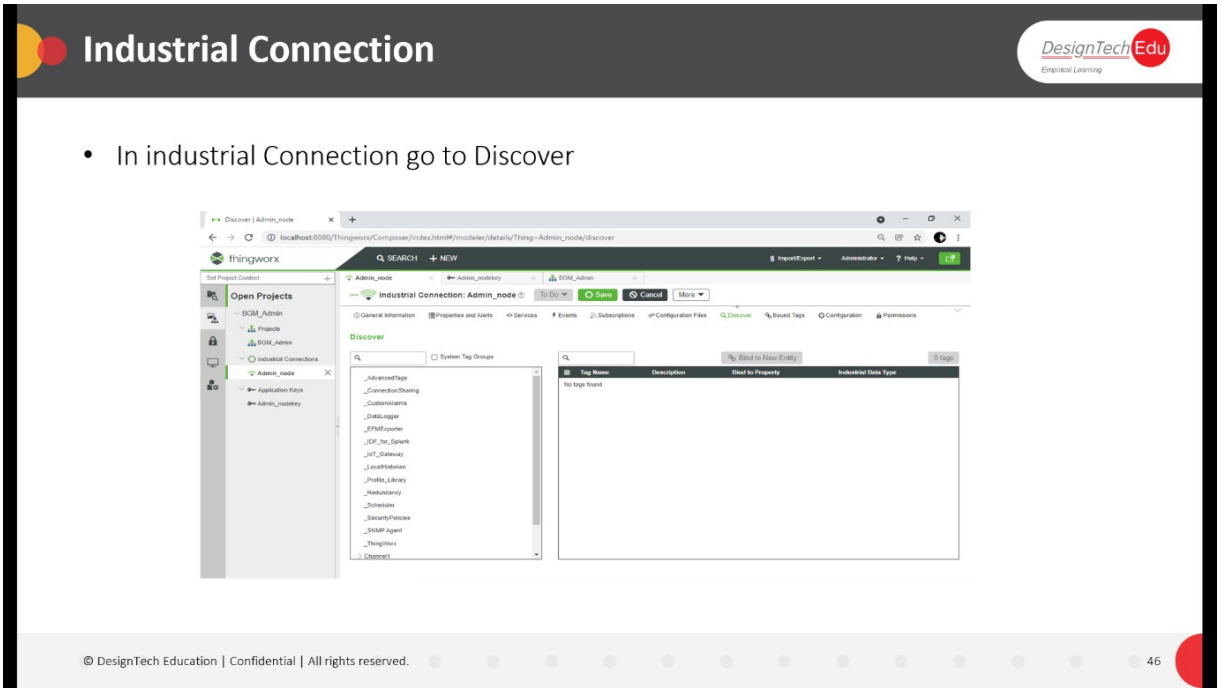
Host, Port, Application key, Thing name.



Check all setting as shown above & update it



After applying all settings “Connected to Thingworx” notification will come.

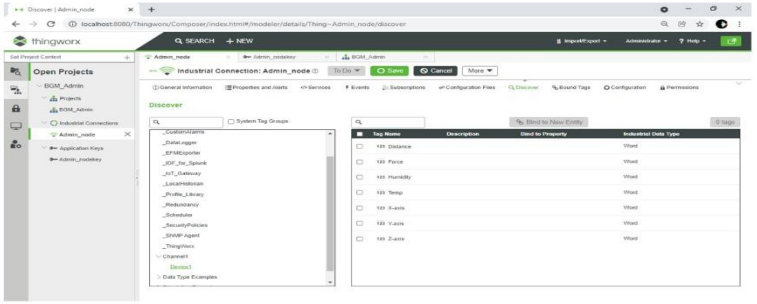


Go to Thingworx> Select Thing (Industrial connection) Admin_node to check all sensor values.
Click on Discover to select channel & device.

Selection of channel and device

DesignTech Edu
Empirical Learning

- Select channel 1 and device 1

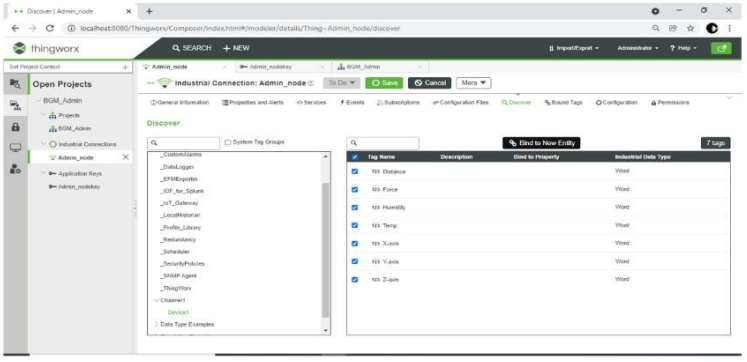


Now Select the same Channel & Device as per Kepware project.

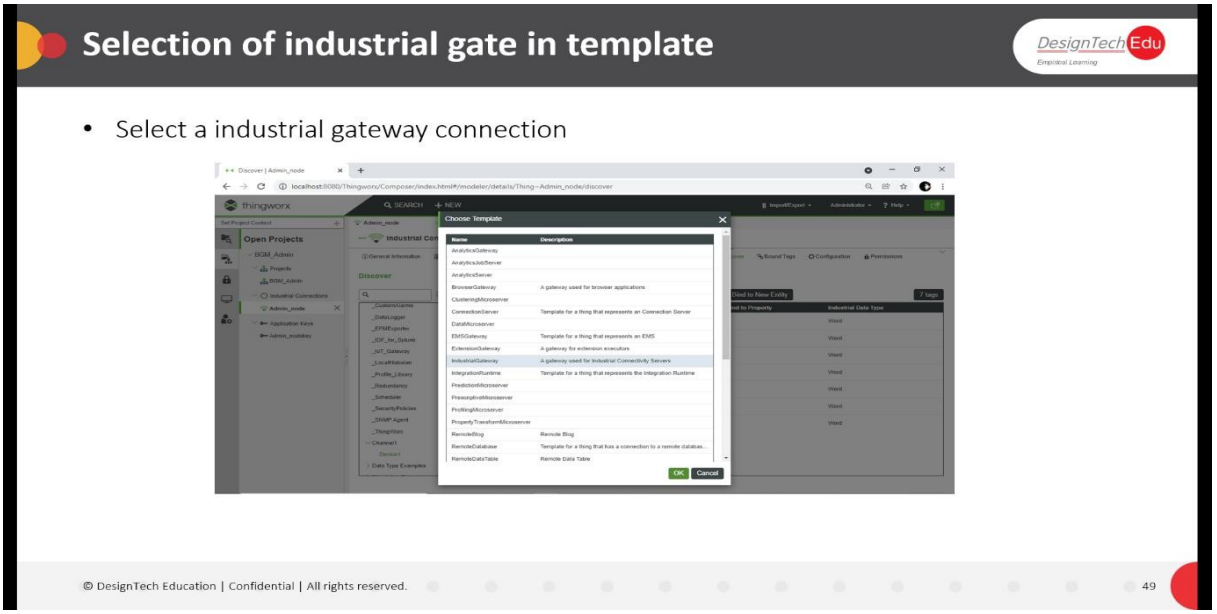
Binding all the entities

DesignTech Edu
Empirical Learning

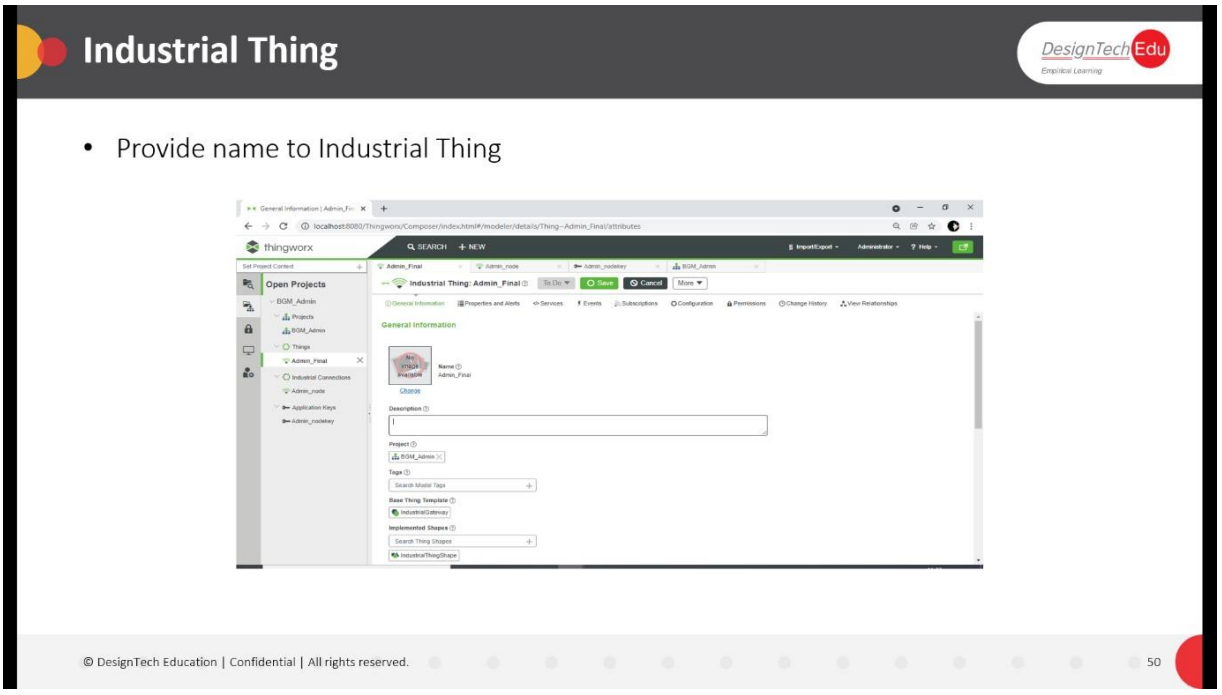
- Select all and bind to New Entity



Select all tags which belongs to Sensors data as per Kepware project.
And click to Bind to New Entity.



It will ask to select Template.
Select Industrial Gateway for Industrial connectivity servers which provide all Kepware server data to ThingWorx.

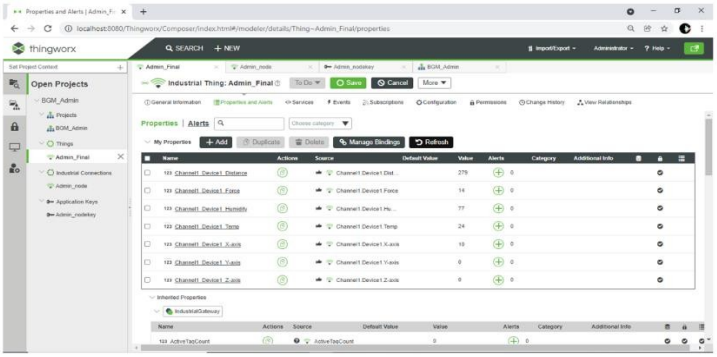


Here after providing Template New Thing will be created.
Provide name as Admin_Final to see sensor values.

Industrial Thing

DesignTech Edu
Empirical Learning

- Go to properties and Alerts

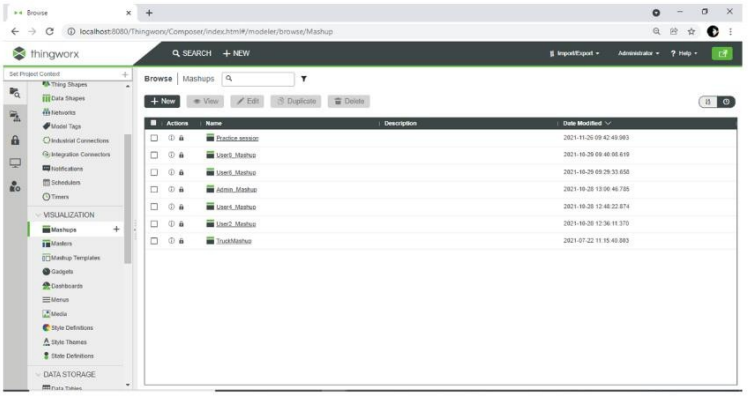


Go to Properties and alerts for Live Values or Alerts.

Creation of Mashup

DesignTech Edu
Empirical Learning

- Click on New to create Mashup

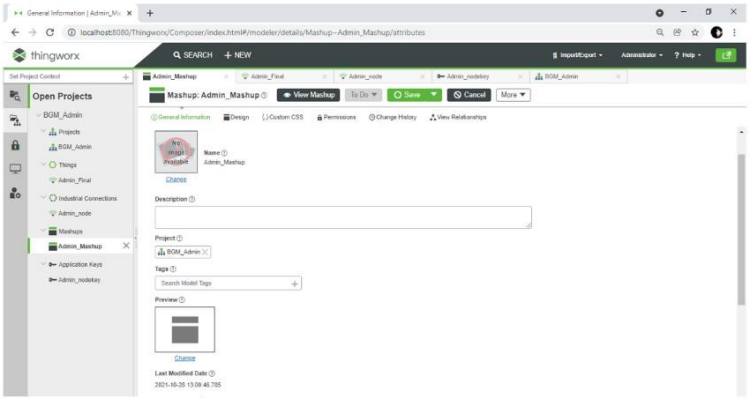


For Visualization, go to Mashups & click on New to create.

Mashup

DesignTech Edu
Empirical Learning

- Provide name to mashup

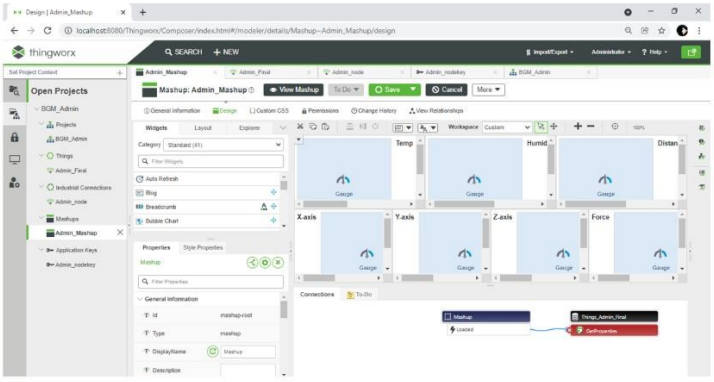


In General Information, Assign mashup name as a Admin_Mashup.
Provide Project name same as given in Industrial connection & Application Key.

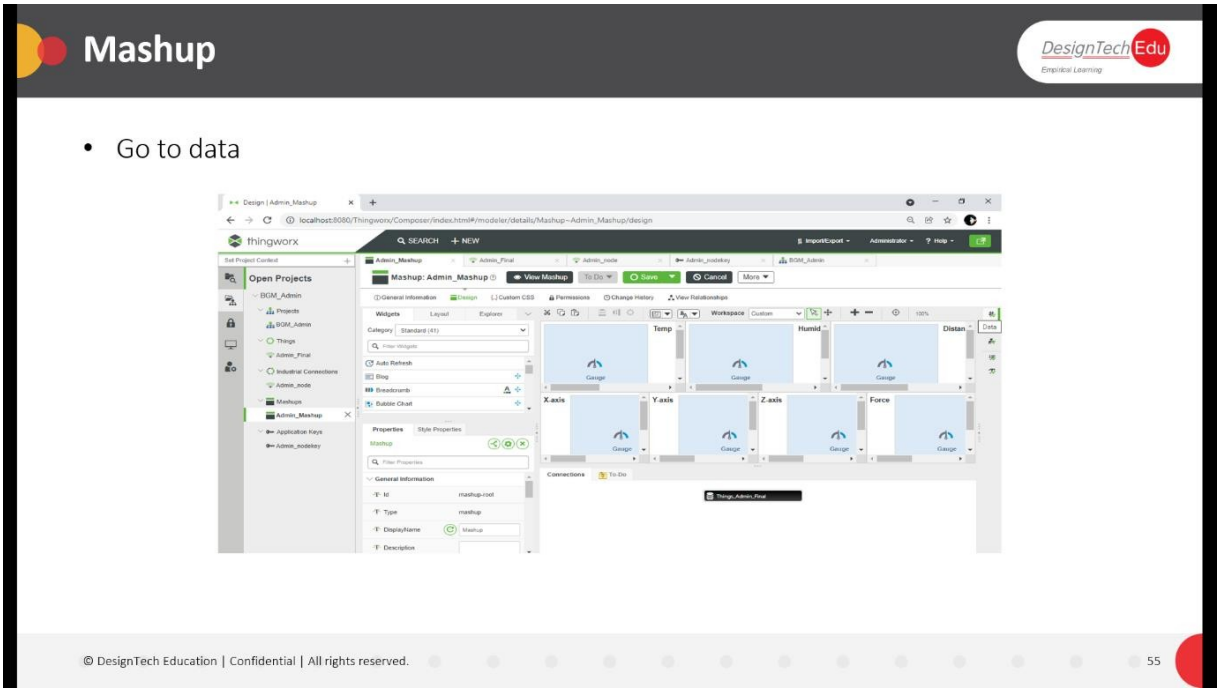
Mashup

DesignTech Edu
Empirical Learning

- Go to Design>Layout and Widgets



Now go to Design. Here first select Layout as per screen.
In Widget select different widgets as per requirements.

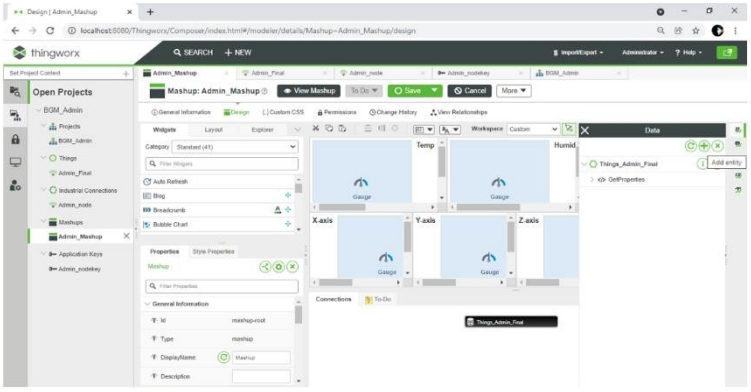


To add sensor properties to widgets, select Data as shown above.

Mashup

DesignTech Edu
Empirical Learning

- Add Entity

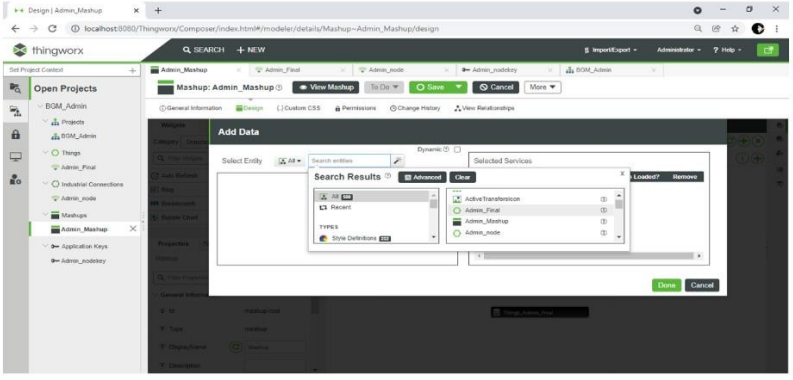


In Data click on + symbol to add entity.

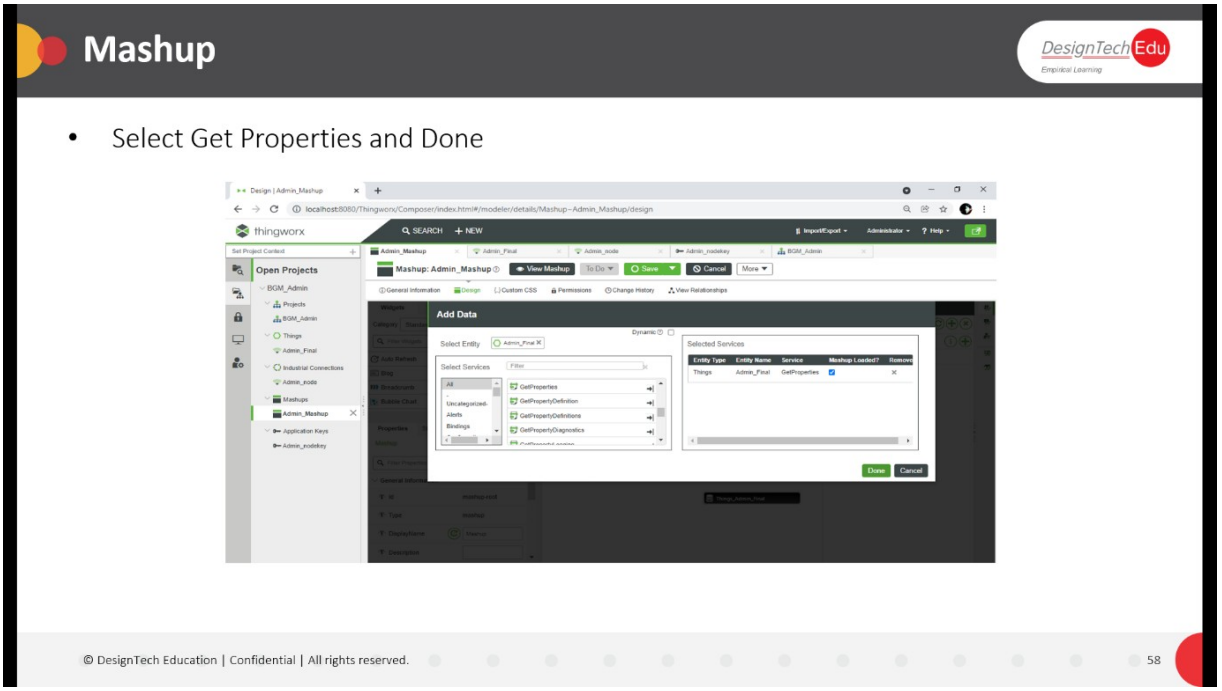
Mashup

DesignTech Edu
Empirical Learning

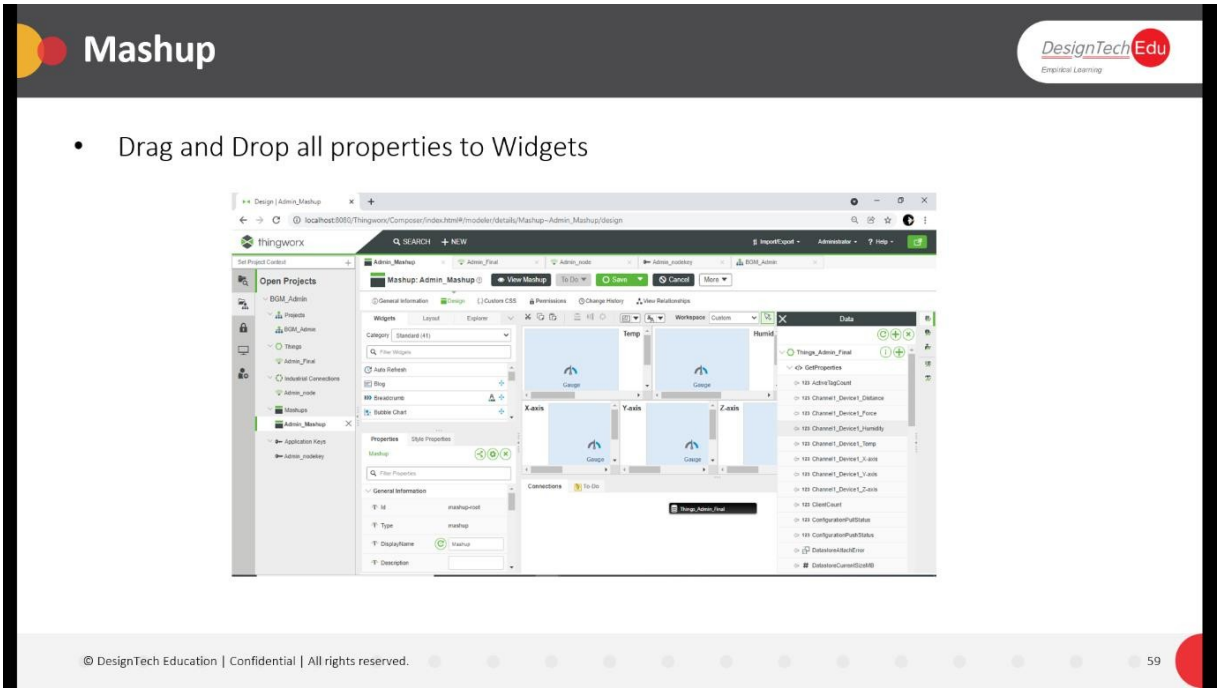
- Select Industrial Connection



Here select The Thing created to see all sensor values, Admin_Final.

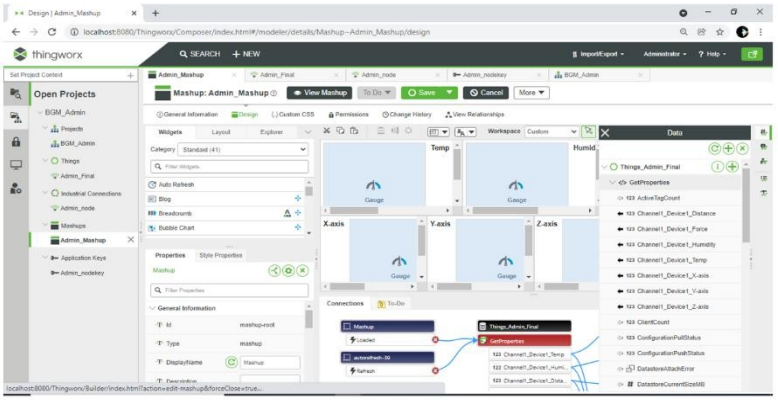


Select Get Properties Services of Selected Entity, Admin_Final.



All Sensor properties are available here, just Drag & Drop properties to widgets.

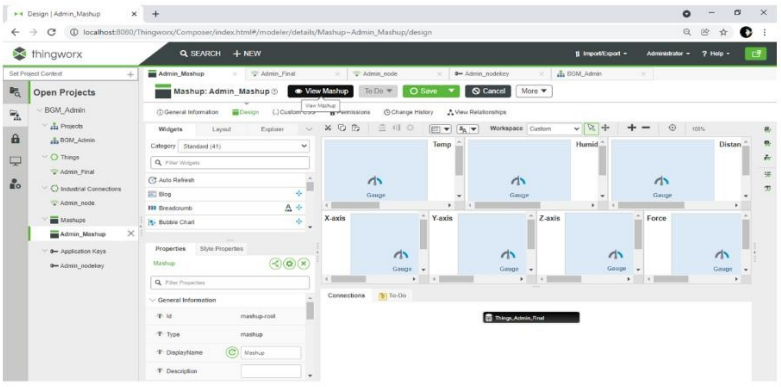
Mashup Connections



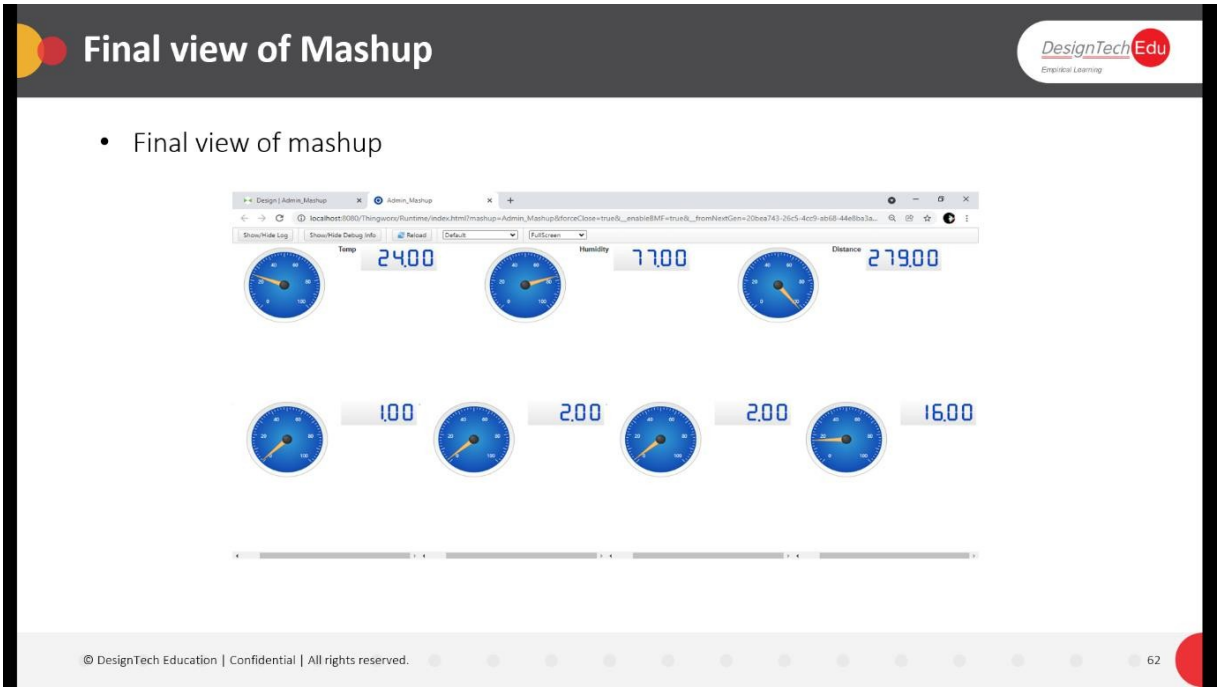
Add Auto Refresh to Get Properties to update all values automatically.

Mashup

- Save and View Mashup



Now Save the settings and click on View Mashup.



Check the values in mashup with Arduino IDE