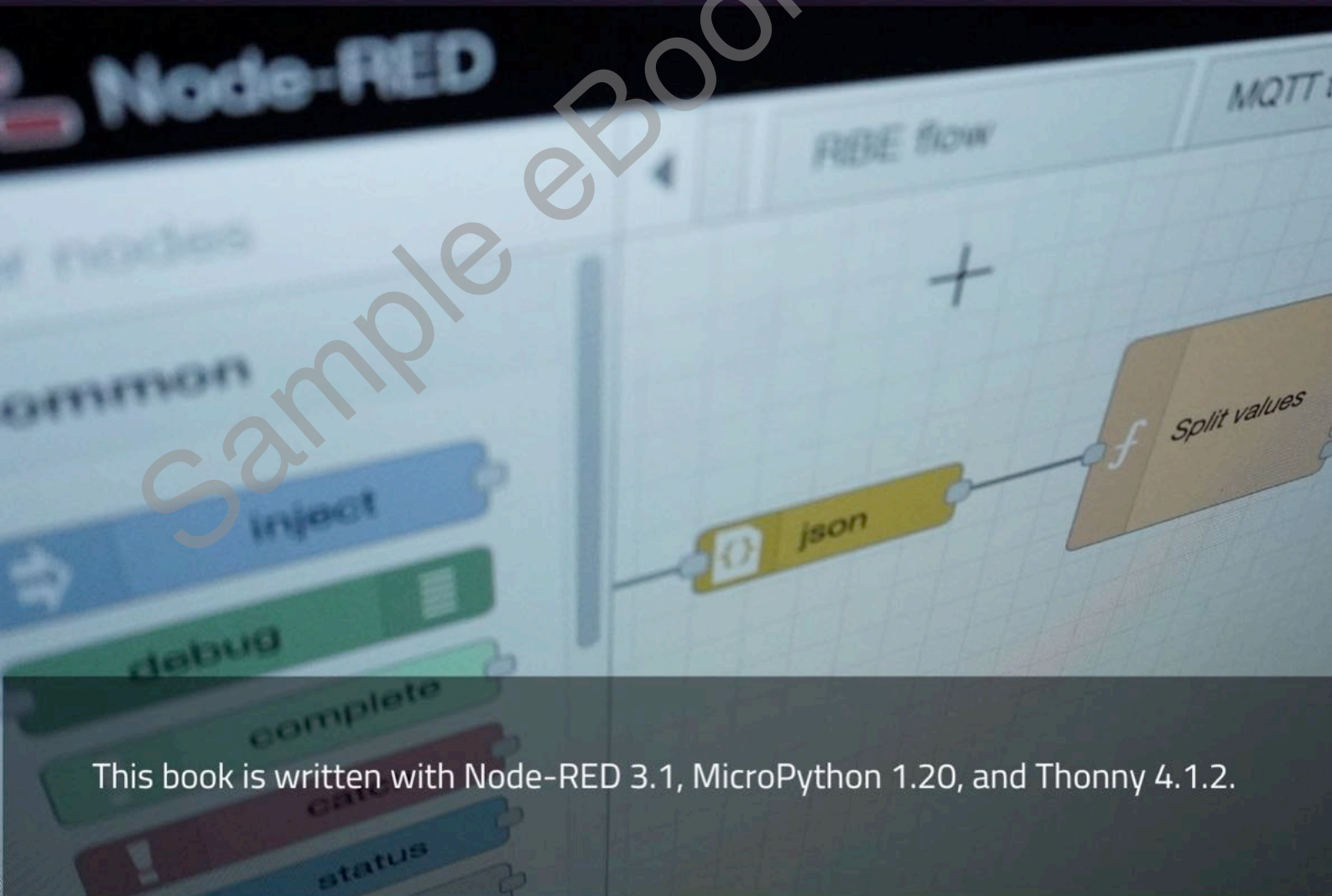
NODE-RED & RASPBERRY PI PICO W

COMBINE NODE-RED WITH THE RASPBERRY PI PICO W TO IMPLEMENT YOUR HOME AUTOMATION (OR INDUSTRIAL AUTOMATION) IDEAS. START WITH THE BASICS, AND CONQUER SENSORS, MOTORS, MQTT COMMUNICATIONS, AND CLOUD SERVICES ALL COORDINATED WITH NODE-RED FLOWS.

DR PETER DALMARIS

Sample eBook content



This book is written with Node-RED 3.1, MicroPython 1.20, and Thonny 4.1.2.

Tech Explorations
www.techexplorations.com

NODE-RED AND RASPBERRY PI PICO

Peter Dalmaris, PhD

Sample eBook content

Node-RED and Raspberry Pi Pico, 1st Edition

By Dr Peter Dalmaris

Copyright © 2023 by Tech Explorations™

All rights reserved. This book or any portion thereof may not be reproduced or used in any manner whatsoever without the express written permission of the publisher except for the use of brief quotations in a book review.

Printed in Australia

First Printing, 2023

ISBN (PDF) : TBA
ISBN (epub): TBA
ISBN (mobi): TBA

Tech Explorations Publishing
PO Box 22, Berowra 2081 NSW
Australia

www.techexplorations.com

Cover designer: Michelle Dalmaris

Disclaimer

The material in this publication is of the nature of general comment only, and does not represent professional advice. It is not intended to provide specific guidance for particular circumstances and it should not be relied on as the basis for any decision to take action or not take action on any matter which it covers. Readers should obtain professional advice where appropriate, before making any such decision. To the maximum extent permitted by law, the author and publisher disclaim all responsibility and liability to any person, arising directly or indirectly from any person taking or not taking action based on the information in this publication.

Version 0.1

Sample eBook content

Did you find an error?

Please let us know.
Go to **txplo.re/nodered**, and fill in the form.
We'll get it fixed right away.

Sample eBook content

About the author

Dr. Peter Dalmaris is an educator, an electrical engineer, electronics hobbyist, and Maker. Creator of online video courses on DIY electronics and author of several technical books. Peter has recently released his book 'Maker Education Revolution', a book about how Making is changing the way we learn and teach in the 21st century.

As a Chief Tech Explorer since 2013 at Tech Explorations, the company he founded in Sydney, Australia, Peter's mission is to explore technology and help educate the world.

Tech Explorations offers educational courses and Bootcamps for electronics hobbyists, STEM students, and STEM teachers.

A lifelong learner, Peter's core skill lies in explaining difficult concepts through video and text. With over 15 years of tertiary teaching experience, Peter has developed a simple yet comprehensive style in teaching that students from all around the world appreciate.

His passion for technology and the world of DIY open-source hardware, has been a dominant driver that has guided his personal development and his work through Tech Explorations.

Sample eBook content

About Tech Explorations

Tech Explorations creates educational products for students and hobbyists of electronics who rather utilize their time making awesome gadgets instead of searching endlessly through blog posts and Youtube videos.

We deliver high-quality instructional videos and books through our online learning platform, txplore.com.

Supporting our students through their learning journey is our priority, and we do this through our dedicated online community and course forums.

Founded in 2013 by Peter Dalmaris, Tech Explorations was created after Peter realised how difficult it was to find high-quality definitive guides for the Arduino, written or produced by creators who responded to their reader questions.

Peter was frustrated having to search for Youtube videos and blog articles that almost never seemed to be made for the purpose of conveying knowledge.

He decided to create Teach Explorations so that he could produce the educational content that he wished he could find back then.

Tech Explorations courses are designed to be comprehensive, definitive and practical. Whether it is through video, ebook, blog or email, our delivery is personal and conversational.

It is like having a friend showing you something neat... the "AHA" moments just flow!

Peter left his career in Academia after his passion for electronics and making was rekindled with the arrival of his first Arduino. Although he was an electronics hobbyist from a young age, something the led him to study electrical and electronics engineering in University, the Arduino signalled a revolution in the way that electronics is taught and learned.

Peter decided to be a part of this revolution and has never looked back.

We know that even today, with all the information of the world at your fingertips, thanks to Google, and all the components of the world one click away, thanks to eBay, the life of the electronics hobbyist is not easy.

Busy lifestyles leave little time for your hobby, and you want this time to count.

We want to help you to enjoy your hobby. We want you to enjoy learning amazing practical things that you can use to make your own awesome gadgets.

Sample eBook content

Electronics is a rewarding hobby. Science, engineering, mathematics, art, and curiosity all converge in a tiny circuit with a handful of components.

We want to help you take this journey without delays and frustrations.

Our courses have been used by over 70,000 people across the world.

From prototyping electronics with the Arduino to learning full-stack development with the Raspberry Pi or designing professional-looking printed circuit boards for their awesome gadgets, our students enjoyed taking our courses and improved their making skills dramatically.

Here's what some of them had to say:

"I'm about half way through this course and I am learning so much. Peter is an outstanding instructor. I recommend this course if you really want to learn about the versatility of the amazing Raspberry Pi" -- Scott

"The objectives of this course are uniquely defined and very useful. The instructor explains the material very clearly." -- Huan

"Logical for the beginner. Many things that I did not know so far about Arduino but easy to understand. Also the voice is easy to understand which is unlike many courses about microcontrollers that I have STARTED in the past. Thanks" -- Anthony

Please check out our courses at techexplorations.com and let us be part of your tech adventures.

Sample eBook content

From the back cover

TBA

Sample eBook content

Requirements

You will need a few things to make the most of this book. You probably already have most of all of them:

- A computer running Windows, Mac OS or Linux. If you have a spare Raspberry Pi Model B (any generation) or an old PC, you can use it as your Node-RED server.
- Access to the Internet.
- Hardware (all available in a kit from Sunfounder):
    - A Raspberry Pi Pico W
    - A breadboard and wires
    - Eight LEDs
    - Resistors (various Ohm ratings)
    - Buttons and slide switches
    - Sensors: DHT11, HCSR04, motion PIR sensor, water level sensor, thermistor, analog light sensor.
    - A joystick
    - A 5V relay
    - An RFID receiver and tag.
    - An infrared receiver and remote control.
    - A 2x16 LCD with the I2C backpack.
    - A 74HC595N THT integrated circuit.
    - A WS2812 RGB LED strip with eight RGB LEDs
    - A small servo motor.
    - A small DC motor

Sample eBook content

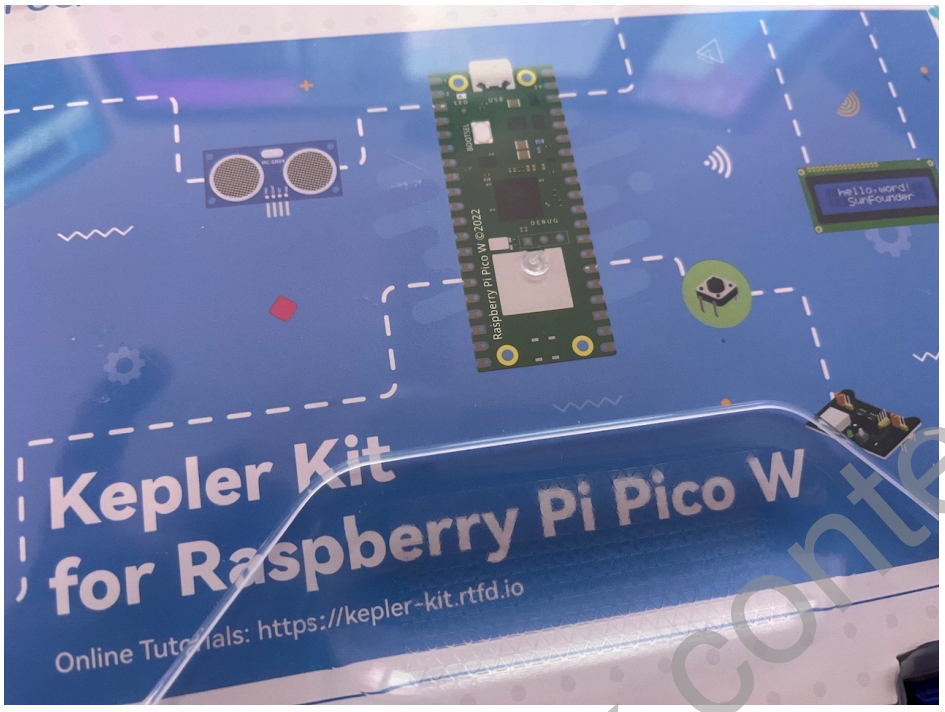


Figure not set.Not set.not set.1: The Kepler kit from Sunfounder contains all the parts you will need for the projects in this book.

Sample eBook content

The book errata reporting and resources web page

This book has a web page. This page contains an errata form to report bugs and access related resources as they become available.

Follow this URL to reach the book page: *https://txplo.re/nodered*.

Sample eBook content

This book is a learning guide and a reference.

You can use it to learn Node-RED, Raspberry Pi Pico W, and Micropython.

You can also use it as a source of information for these topics.

I have organised this book into four parts to fulfil these dual roles.

Part 1 is dedicated to Node-RED for the absolute beginner. In Part 1, you will learn about Node-RED and event-driven systems, how to install an instance using the Docker option, the basics of nodes and flows, the dashboard and MQTT. If you are new to Node-RED, read the chapters in Part 1 carefully and complete the various projects.

Part 2 brings the Raspberry Pi Pico W into the mix. In the chapters of Part 2, you will learn how to use the Pico W as a Node-RED peripheral. You will learn to use MQTT to enable communications between the Pico and the Node-RED instance. You will also learn how to connect different hardware components to the Pico to implement simple circuits and use Node-RED (and its Dashboard) to control these components or view the data they produce. In Part 2, you will encounter motion, distance and water level sensors, motors, displays, relays, and joysticks, to mention a few. In all projects in Part 2, you will create Node-RED flows and write Raspberry Pi Pico W Micropython scripts that typically implement an event-driven system.

Part 3 provides a primer to MicroPython. MicroPython is a language specifically designed for embedded systems using Python 3 syntax. Python 3 is one of the most successful programming languages ever. Python's syntax is straightforward, making it easy for beginners to learn. This simplicity encourages good programming practices and allows for a focus on problem-solving rather than syntax issues. MicroPython brings those attributes to Microcontroller programming. If you are not familiar with Python or MicroPython, the chapters in Part 3 will help you learn everything you need to be able to confidently write MicroPython programs for the Raspberry Pi Pico (and Pico W), as well as any other of the many Microcontroller boards that support MicroPython.

Sample eBook content

Finally, Part 4 provides additional Node-RED resources. These resources consist of content on important Nodes (all explained with the help of mini-projects), control structures, and ways to integrate your Node-RED flows with external services and APIs. In Part 4, you will learn how to create power flows that can be used in more advanced automation settings.

If you are new to Node-RED, Raspberry Pi Pico and Micropython, I recommend reading this book in a linear fashion. Don't skip anything!

If you are familiar with Node-RED, you can quickly read Part 1 and continue with the projects in Part 2.

If you are unfamiliar with MicroPython, start with Part 3 and continue with other parts of the book. The Part 3 primer on MicroPython is a mini-book in this book and can be used independently of the rest of the content.

In this book, I chose Docker as the infrastructure technology on which I installed my Node-RED instance. You may prefer a different method (and there are several). You are free to choose any installation method you prefer. As long as, in the end, you have an accessible instance of Node-RED and MQTT broker running, you will be able to complete all of the projects in this book.

Enjoy!

Sample eBook content

Table of Contents

Sample eBook content

Sample eBook content

Sample eBook content

Sample eBook content

Part 1: Node-RED novice to hero

Sample eBook content

1. What is Node-RED?

Node-RED is an open-source flow-based development tool that makes it easy to wire together devices, APIs, and online services. Imagine being able to drag and drop blocks on a screen to create a flowchart that does something—like turning on your lights at sunset or sending you an email when a sensor detects movement. That's what Node-RED lets you do, all without requiring you to write extensive code.

What is flow programming?

Flow programming in the context of Node-RED is a way to build applications by connecting different "nodes" in a flowchart-like manner. Each node performs a specific task, like reading from a sensor, performing a calculation, or sending an email. You create a "flow" by dragging and dropping these nodes onto a canvas and connecting them with "wires" to define the order of operations. The result visually represents your application logic, which you can deploy with a single click.

In traditional text-based programming, you write lines of code to define what your application should do. This often involves setting up loops, conditionals, and functions, and it requires a good understanding of the programming language you're using. In contrast, flow programming in Node-RED abstracts away much of this complexity. Instead of writing code, you're essentially drawing your program. This makes it easier to see the big picture, understand the data flow, and spot any potential issues.

Flow programming shines in scenarios where quick prototyping and iteration are essential. Because you can see the entire logic laid out visually, making changes or adding new functionalities is easier. You don't have to sift through lines of code to find the section you need to modify; you can rearrange or add nodes on the canvas.

It's also beneficial for people who may not have a strong background in programming. The drag-and-drop interface and pre-built nodes make it accessible for beginners, allowing them to focus on solving the problem rather than getting bogged down by syntax and language-specific rules.

Flow programming is compelling for IoT applications and automation tasks. When dealing with multiple devices, sensors, and APIs, the visual nature of flow programming makes it easier to manage the complexity. You

Sample eBook content

can quickly see how data moves from your sensors to your logic nodes and output actions, making the development process more intuitive.

Where Can You Use Node-RED?

Before continuing, looking at some areas of everyday life where Node-RED is being used with great results is helpful. This will help you understand the impact that Node-RED can have on your projects. To keep this segment short, I'll touch only on three areas where Node-RED makes a real difference: industry, education, and home automation.

In industrial settings, Node-RED is a game-changer. For instance, in manufacturing, it can be used to automate entire production lines. By connecting to PLCs (Programmable Logic Controllers) and sensors, Node-RED can control the sequence of machinery operations. It's also valuable for energy management. Companies use Node-RED to monitor and control how much energy is being used in real-time, allowing them to optimise consumption and reduce costs. Beyond that, it's used for predictive maintenance by collecting data from various machinery and using it to predict when a machine is likely to fail, thus scheduling timely maintenance.

Educational institutions find Node-RED to be an excellent tool for teaching and research. It's often used to introduce students to the concepts of IoT, networking, and automation in a hands-on manner. Because of its ease of use, students can quickly move from theory to practice. In research settings, Node-RED serves as a quick prototyping tool, enabling researchers to test their ideas without spending much time on initial setup and coding.

Node-RED offers endless possibilities for the DIY enthusiast or anyone interested in smart homes. You can set up intelligent lighting systems that adjust based on the time of day or occupancy. You can even create a home security system that sends you alerts based on sensor data. The possibilities are limited only by your imagination and the devices you have at hand.

Sample eBook content

The Tech Behind Node-RED

Node.js is an open-source, cross-platform JavaScript runtime environment that executes JavaScript code outside a web browser. Initially released in 2009, it has become a foundational technology for server-side development. Node.js allows you to build scalable network applications using JavaScript, a language traditionally used for client-side scripting in web browsers. It is incredibly lightweight and can run on various platforms, from a Raspberry Pi to a full-scale server.

It uses JSON to represent the flows, making it human-readable and machine-friendly. The logic is encapsulated in “nodes,” essentially pre-written JavaScript functions you can string together to create your application.

Node.js is commonly used to build web servers and RESTful APIs. Its non-blocking, event-driven architecture makes it well-suited for handling multiple simultaneous connections, making it a popular choice for real-time applications. Applications like chat rooms and collaborative editing tools often use Node.js to manage real-time, bidirectional communication between the server and clients.

Node.js excels in scenarios where data streaming is crucial. For example, it can process files while uploaded, reducing the overall processing time. Node.js is frequently used in microservices architectures due to its ability to handle multiple small tasks concurrently. Companies like Netflix and Uber employ Node.js to manage their complex, distributed systems.

If you plan to use Node.js solely for Node-RED, you don't need to be an expert in Node.js. A basic understanding of JavaScript and how to install Node.js packages would be sufficient. Node-RED abstracts most of the complexities, allowing you to focus on the logic of your flows rather than the underlying code.

If you can only remember four facts about Node.js, these should be:

1. **Non-blocking, Event-Driven Architecture**: Node.js uses a single-threaded, non-blocking event loop, allowing it to handle many connections simultaneously.
2. **NPM (Node Package Manager)**: Node.js has a rich ecosystem of libraries and frameworks available through NPM that can simplify the addition of new functionalities to your projects.
3. **Cross-Platform:** Node.js can run on various operating systems, including Windows, macOS, and Linux, making it highly versatile.
4. **Community Support:** Being open-source and popular means that Node.js has a large, active community contributing to its development and a wealth of tutorials and resources.

Node.js is a powerful runtime environment with many applications beyond Node-RED. Its non-blocking architecture and rich ecosystem make it a go-to choice for many projects, from web servers to real-time communication apps. Even if you're only interested in using it for Node-RED, a minimal

Sample eBook content

understanding will suffice, making it accessible for users with varying coding expertise.

And that’s all you need to know about this (unless you plan to build Node.js applications).

Node-RED and Microcontrollers

One of the most exciting aspects of Node-RED is its ability to interface with microcontrollers like Arduino and ESP8266. You can read sensor data from these devices and send commands back to control actuators, all through Node-RED's intuitive interface. This makes it an excellent tool for anyone looking to get into hardware hacking or create custom IoT devices.

In summary, Node-RED is a versatile and powerful tool with applications ranging from industrial automation to education and home DIY projects. Its drag-and-drop interface makes it accessible for people of all skill levels, while its underlying technology ensures it's robust enough for professional use. Whether you're a hobbyist looking to smarten up your home or an engineer aiming to optimise a production line, Node-RED has something to offer.

Sample eBook content

2. Node-RED in IoT and event-driven systems

The Internet of Things (IoT) has revolutionised how we interact with devices and systems daily. As IoT networks grow, the need for an efficient, user-friendly, and powerful tool to manage these connected devices becomes apparent. Enter Node-RED, an open-source, flow-based programming tool that simplifies the creation, deployment, and management of IoT applications and event-driven systems.

Node-RED's flexibility and ease of use make it a perfect fit for various IoT use cases, such as:

Device Management and Control

Node-RED simplifies the process of connecting and managing IoT devices. It supports many communication protocols, such as MQTT, HTTP, WebSocket, and more, making it easy to collect data from sensors, control actuators, and interact with devices in real time.

For example, a Node-RED flow can be created to monitor a sensor's temperature and humidity data, process the data, and control a smart thermostat based on predefined conditions.

Data Processing and Analytics

IoT applications often require processing and analysing data from multiple sources. Node-RED provides various built-in nodes for data processing, such as Function nodes (for custom JavaScript code), Switch nodes (for conditional routing), and Change nodes (for modifying message properties).

For instance, Node-RED can aggregate sensor data from multiple sources, filter out irrelevant information, and perform calculations to derive insights, such as detecting anomalies or predicting equipment failure.

Integration with Cloud Services and APIs

Node-RED enables seamless integration with popular cloud platforms, such as AWS, Azure, Google Cloud, and IBM Watson, as well as third-party APIs, like Twitter, Telegram, and Slack. This allows developers to build IoT applications that leverage cloud-based services for data storage, analytics, and machine learning.

Sample eBook content

An example use case involves using Node-RED to send sensor data to a cloud-based database, perform real-time analytics, and trigger notifications or alerts through third-party messaging services.

Node-RED in Event-Driven Systems

Event-driven systems are designed to respond to specific events or changes in state. Node-RED's flow-based programming model and support for various communication protocols make it an excellent choice for building event-driven applications.

Home Automation

Node-RED can create a custom home automation system that responds to events like motion detection, temperature changes, or voice commands. Integrating smart home devices and APIs, Node-RED can control lights, HVAC systems, security cameras, and more based on predefined rules and conditions.

Industrial Automation and Monitoring

In industrial settings, Node-RED can monitor and control equipment based on events or conditions, such as motors, valves, and conveyor belts. This can help optimise production processes, reduce downtime, and ensure equipment safety.

For example, a Node-RED flow can be created to monitor the status of a production line, detect when a machine malfunctions, and automatically shut down the affected equipment or send alerts to maintenance staff for immediate action.

Smart Cities and Infrastructure

Node-RED can be crucial in developing innovative city applications that respond to real-time events, such as traffic congestion, air quality, or energy consumption. By integrating with various sensors and services, Node-RED can be used to create intelligent systems that optimise urban infrastructure and enhance the quality of life for residents.

For instance, a Node-RED flow can be designed to analyse traffic data from multiple sources, detect congestion or accidents, and dynamically adjust traffic light timings or notify emergency services.

Sample eBook content

Conclusion

Node-RED's visual programming interface, extensive library of nodes, and support for a wide range of communication protocols make it a powerful tool for developing IoT and event-driven applications. Its flexibility and ease of use enable developers to create custom solutions for various use cases, from home automation and industrial monitoring to smart cities and infrastructure management.

As the IoT landscape continues to evolve, Node-RED is poised to play a significant role in developing and deploying connected systems that enhance our daily lives and drive innovation across industries.

Sample eBook content

3. Communication in Node-RED: Protocols and Methods

Node-RED has an impressive ability to communicate with other systems thanks to its compatibility with various communication protocols. Its flexibility in this domain is key to its versatility. Let's delve deeper into the protocols and communication methods used by Node-RED.

HTTP and HTTPS

HTTP (HyperText Transfer Protocol) and its secure variant, HTTPS, are fundamental to the web as we know it. They serve as the basis for data communication on the World Wide Web. In Node-RED, you can use the “http” node to make HTTP requests to APIs on the web. For example, you could use an HTTP GET request to fetch data from a weather API or an HTTP POST request to send data to a database.

When working with Node-RED, using HTTP and HTTPS can be really handy. HTTP is easy to set up, and you don't need special certificates. It's great for quick tests and prototypes. HTTPS, on the other hand, adds a layer of security by encrypting the data. This is crucial if you're dealing with sensitive information.

However, with HTTP (notice the lack of the “S” from the acronym), your data is not encrypted, so transmitting confidential information is unsafe. Anyone can intercept it. HTTPS solves this problem but is a bit more complicated to set up. You'll need to get a security certificate, and sometimes there might be compatibility issues with older devices or systems.

MQTT

MQTT (Message Queuing Telemetry Transport) is a lightweight messaging protocol often used in IoT applications where bandwidth and power consumption are concerns. With MQTT, devices publish messages to topics, and other devices subscribe to these topics to receive the messages. Node-RED has built-in MQTT nodes, making integrating with MQTT-based systems straightforwardly. For instance, you could have a Raspberry Pi publish temperature sensor data to an MQTT topic and then use Node-RED to subscribe to this topic and react accordingly.

Sample eBook content

MQTT is a popular choice for sending and receiving data when using Node-RED. As I mentioned earlier, MQTT is lightweight. It doesn't use much bandwidth, which is great if you work with devices with limited resources. It's also good at handling intermittent connections, so if your network is shaky, MQTT can still get the job done. Plus, it's designed for real-time communication, so you get updates as they happen.

On the flip side, MQTT has some limitations. It’s not the best choice for large data sets because it's designed for small, frequent messages. Also, while MQTT itself is pretty secure, adding extra layers of security can be a bit complex. You might need to integrate it with other security protocols, which could be a hassle if you're new to this.

MQTT is great for real-time, lightweight data communication, but it might not be the best fit for every scenario, especially those requiring high security or large data transfers.

WebSockets

WebSockets is a protocol that provides full-duplex communication between a client and server over a long-lived connection. This is especially useful for applications that require real-time data updates. Node-RED provides a “websocket” node for client and server-side communication using WebSockets. This allows you to create interactive, real-time flows.

WebSockets can be a great way to handle data communications using Node-RED. One of the coolest things about WebSockets is that they allow for two-way communication between the server and the client. This means you can send and receive data at the same time, making your applications more interactive and responsive. WebSockets are also pretty fast because data can flow freely once the connection is established without repeatedly opening and closing connections.

On the other hand, WebSockets come with their own set of challenges. One issue is that they can be more complex to set up than other protocols like HTTP. You must ensure the client and server are configured correctly to handle WebSocket connections. Not all network configurations also support WebSockets, so you might run into issues if you're behind certain types of firewalls or proxies.

TCP and UDP

TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) are two fundamental protocols of the Internet protocol suite. TCP is

Sample eBook content

connection-oriented and ensures the reliable delivery of a stream of bytes, while UDP is simpler, connectionless, and suitable for scenarios where speed is more important than reliability. Node-RED provides TCP and UDP nodes for handling raw TCP and UDP communications, giving you even more flow flexibility.

If you're working with Node-RED, TCP is a reliable way to send your data. It makes sure all your data packets arrive in the correct order. This is great when you can't afford to lose data, like sending a file or updating a database. TCP also handles congestion well, so if your network is busy, it will adjust to ensure your data gets through.

The downside of TCP is that it can be slower than other methods. It takes time to establish a connection, and the checks it performs to ensure data integrity can add delays. So, TCP might not be the best choice for real-time applications where speed is crucial.

UDP, on the other hand, is all about speed. It sends data without worrying too much about whether it arrives or not. This is good for things like streaming video or audio, where you'd rather lose a few packets than have a delay.

The problem with UDP is that it's not reliable. If ensuring that every piece of data arrives is important to you, then UDP is risky. It doesn't guarantee delivery or order so you might get incomplete or jumbled data.

Serial

Node-RED also supports the serial communication protocol, commonly used for interfacing with hardware such as microcontrollers. With the serial node, you can read data from a serial port or write data to it. This is handy when working with devices like Arduino, which often use serial communication.

Using serial communications with Node-RED can be a straightforward way to communicate with hardware devices. It's a tried-and-true method that's been around for a long time, so it's well-supported and reliable. You'll often find it used in industrial settings or for connecting to older equipment. It's also simple to set up; you usually just need to plug in a cable and configure a few settings.

Serial communications has its limitations. One of the main drawbacks is that it's not ideal for long-distance communications. The signal can degrade over distance, so it's mostly used for connections that are close by. Also, it's generally slower than other modern data transfer methods, so it might not be

Sample eBook content

the best choice for quickly transferring large amounts of data. Another thing to consider is that many modern computers don't come with a built-in Serial COM Port, so you might need an adapter if your Node-RED instance is running on more modern computer hardware.

In this book, as an example, I have set up my Node-RED instance to run inside a virtual machine, which, in turn, runs inside a modern personal computer without a serial port. Therefore, I cannot use this communication method without an adaptor.

On the other hand, if I had used a Raspberry Pi to host the Node-RED server, I would have been able to create flows that use the Pi's serial port to establish serial communication with microcontrollers such as the Raspberry Pi Pico and the Arduino UNO, both of which have support for serial communications.

Modbus

Modbus is a protocol often used in industrial applications for communication between electronic devices. If you're working in an industrial setting with Modbus equipment, you'll be pleased to know that Node-RED can handle this, too, thanks to community-contributed nodes.

Modbus can be a solid choice for data communications using Node-RED, especially for industrial applications. One of the big pluses is that Modbus is widely used in industrial settings, so it's well-tested and reliable. It's also straightforward to set up. You can use it over various connections, like serial or TCP, giving you some flexibility. Modbus is also good for systems that require high reliability, as it has built-in error checking.

On the downside, Modbus has some limitations. It's not the fastest protocol out there, so it might not be the best fit if you need to transfer large amounts of data quickly. Also, while it's great for simple tasks, it's not as well-suited for more complex data structures. You might find it limiting if you need to handle various data types or complex commands. Another thing to consider is that it doesn't have built-in security features because it's an older protocol. You'll need to add those yourself if security is a concern.

Modbus is reliable and well-suited for industrial applications but may not be the best choice for high-speed or complex data communications.

Node-RED's extensive protocol support makes it an excellent tool for integrating diverse systems. Whether pulling data from a web API,

Sample eBook content

communicating with an IoT device over MQTT, or reading sensor data from a microcontroller over a serial connection, Node-RED has you covered.

Sample eBook content

4. Node-RED installation options

When it comes to installing and setting up a Node-RED server, several options are available that cater to a range of technical requirements and resource availabilities. These options include installing Node-RED to your ”regular” work computer as you would with any other program, using a Raspberry Pi or other Linux computers, setting up a Virtual Machine (VM), or leveraging Docker. Each choice has unique features and advantages, but as we move forward, we will focus on the benefits of Docker as a preferred choice.

Installing Node-RED on Your Local Computer

Node-RED is very flexible when it comes to operating systems. You can install it on Windows, macOS, and various distributions of Linux.

Installing Node-RED locally has several advantages. For one, you have complete control over your environment. You can customise settings, add nodes, and test flows without worrying about external factors like network latency. It's also easier to integrate with other software and hardware on your computer, such as a webcam and microphone. Plus, you don't have to worry about monthly subscription fees or data limits that you might encounter with cloud-based solutions. Another advantage worth considering is that you do not need access to another computer with local installation. You already have a computer, and it’s the one that you are (probably) working on right now!

However, the local installation has several significant disadvantages. At the top of the list is that your work computer can be a busy and constantly changing environment. If you are like me, your work computer is where you do all your programming, where you install, remove, reconfigure and experiment with software. It has multiple network connections and lots of connected devices. Once in a while, it will run out of disk space, or other issues will pop up. All this can affect the smooth operation of server software like Node-RED, and you may spend too much time troubleshooting self-inflicted problems rather than learning and using Node-RED. Another issue to consider is that since your flows are stored locally, you won’t be able to access them from another computer unless you’ve set up remote access. And let's not forget, if your computer crashes, you risk losing all your work unless you've been diligent about backups.

Sample eBook content

For these reasons, I do not recommend installing Node-RED on your work computer unless you install it on a dedicated Virtual machine (see below) so that there is robust isolation between the host and the guest OS.

Raspberry Pi and Other Linux Computers

By deploying Node-RED on a dedicated Linux machine, such as a Raspberry Pi or an old re-purposed PC, you're positioning yourself for a highly efficient and customisable experience. Let's explore the nuances of this approach, from compatible Linux operating systems to the inherent pros and cons.

Node-RED offers compatibility with a range of Linux distributions. While Raspbian remains the go-to choice for Raspberry Pi users, other distributions like Ubuntu and Fedora are also viable options. Your selection will likely hinge on the specific requirements of your project or your familiarity with a particular Linux distribution.

Linux's open-source nature stands as one of its most compelling advantages, offering an unparalleled customisation level. Additionally, “headless” Linux distributions are generally lightweight, making them well-suited for hardware with limited computational resources, such as a Raspberry Pi. This results in a cost-effective yet powerful solution for running Node-RED. Furthermore, Linux's reputation for stability and security adds more reliability to your setup.

If you are unfamiliar with the term “headless”, here is what you need to know. ”headless Linux” refers to a Linux server or system run without a graphical user interface (GUI). In other words, it doesn't have the graphical desktop environment that you might be used to seeing on consumer-focused computers. Instead, all interactions with the system are done through the command line interface (CLI). Running Linux in headless mode is resource-efficient, as it eliminates the need for the system to allocate resources to the GUI. This is particularly beneficial for servers or systems designed to run specific tasks and don't require user interaction via a graphical interface. It's also a typical setup for Linux instances running on cloud platforms or in data centres.

Despite its merits, Linux has challenges, particularly for those less acquainted with its environment. The operating system often necessitates command-line interactions, which can be daunting for newcomers. Moreover, Linux's robustness and stability come at the cost of limited software compatibility, especially for applications designed with Windows or macOS in

Sample eBook content

mind. However, I recommend that you use your Linux machine as a Node-RED host only and not as a general-purpose computer; hence, the compatibility issue with applications that you find on Windows and macOS computers is a non-issue.

Raspberry Pi, a low-cost, credit-card-sized computer, is one of the most popular choices for running a Node-RED server. Its affordability and accessibility make it a compelling choice for hobbyists and enthusiasts just starting with the Internet of Things (IoT) and Node-RED. Other Linux computers, with their robust and open-source nature, also serve as great hosts for Node-RED.

Installing Node-RED on Raspberry Pi or any other Linux system is straightforward. With a few commands in the terminal, your server will be up and running. However, while these systems are great for smaller, personal projects, they might not scale well for larger, more resource-intensive applications.

Virtual Machine

Deploying Node-RED on a Linux Virtual Machine (VM) offers a unique blend of flexibility and isolation. This approach allows you to run Node-RED in a controlled environment, separate from your primary operating system. The virtual machine can run on your local ("work") computer or another computer in your local network or accessible via the Internet.

In terms of virtualisation options, there are several to choose from. Which one you pick does not matter much since all options can run and manage Linux guests. For example, options consider VMware, VirtualBox, Parallels, QEMU, Citrix, and Zen (and there are more). These are among the most popular and user-friendly options. Both provide a wide range of features and are well-documented, making it easier to find help if you run into issues. As I mentioned, all of these options support Linux guests; your choice will depend on your existing infrastructure and specific project requirements.

One of the most compelling benefits of using a VM is its isolation. Your Node-RED installation will be contained within the VM, separate from your primary operating system. This makes it easier to manage dependencies and avoid conflicts with other software. Additionally, VMs are highly portable. You can quickly move them between computers or upload them to cloud services. This portability simplifies backup procedures and makes it easier to share your Node-RED setup with others.

Sample eBook content

However, running Node-RED requires some planning and forward-thinking. Virtual machines consume additional system resources, including CPU, memory, and disk space, which could impact the performance of your primary operating system. You must be careful when you plan the deployment of VMs on a host so that the VM does not try to consume more resources (such as RAM, disk space and CPU cores) than are available on the host machine. Moreover, setting up a VM involves a series of configuration steps that might be daunting for those unfamiliar with virtualisation technology. Lastly, while VMs provide isolation, they can also add a layer of complexity to your setup, especially regarding networking and file sharing between the host and guest operating systems.

Installing Node-RED on a Linux Virtual Machine offers a flexible and isolated environment, ideal for testing and development. However, it does come with the trade-off of additional resource consumption and complexity. If you're comfortable navigating these challenges, a VM can be an excellent platform for your Node-RED projects.

My personal choice, and the one that I deployed for the production of this book, is to combine virtualisation with Docker (which I will discuss next). You will learn about the process I followed in the next chapter. The process involves setting up a Linux virtual machine on a small Windows personal computer (to act as a Docker container host) and installing Node-RED using the Docker method.

Docker: The Preferred Choice

While all of these options are valid and have unique advantages, Docker has emerged as a preferred choice for many when setting up a Node-RED server. Docker is a platform that allows you to package an application and its dependencies into a "container." This container can then be easily moved and run across different computing environments. Think of a Docker container as a lightweight, standalone, executable software package that includes everything needed to run a piece of software.

One of the primary benefits of using Docker for Node-RED is the ease of deployment. Docker containers encapsulate all the dependencies and configurations required, making the installation process remarkably straightforward. This eliminates the "it works on my machine" issue, where software behaves differently on different systems. Additionally, Docker containers are isolated from each other and the host system, enhancing security and reducing conflicts between software components. This isolation

Sample eBook content

also makes it easier to manage different Node-RED versions or run multiple instances simultaneously.

While Docker offers numerous advantages, it's not without its drawbacks. For starters, there's a learning curve associated with understanding Docker concepts like containers, images, and Dockerfiles. Getting comfortable might take some time if you're new to these. In this book, I used the Docker method to install my Node-RED instance, and I will show you how to install yours step-by-step.

Also, Docker containers do consume system resources, albeit less than traditional virtual machines. This could be a concern if you run on hardware with limited resources. Lastly, while Docker provides isolation, it can sometimes complicate networking configurations, especially for those not well-versed in networking principles.

Since Docker is the preferred method for Node-RED installation in this book, I take another moment to list the key takeaway advantages that convinced me to use this method:

1. **Portability:** Docker allows you to package an application with all its dependencies into a standardised unit called a Docker container. This container can run on any system that has Docker installed, irrespective of the underlying operating system. This high level of portability simplifies deployment and reduces potential issues caused by differences in system environments.
2. **Isolation:** Similar to VMs, Docker provides isolation. However, Docker containers are less resource-intensive than VMs, as they share the host system's kernel and do not require an entire operating system. This efficiency means you can run more containers on a given hardware combination than with VMs.
3. **Scalability:** Docker's design makes it easy to scale applications. You can quickly start, stop, replicate, or destroy containers with a single command or via Docker's APIs.
4. **Consistency:** With Docker, you can maintain consistency across multiple development, testing, and production environments. This consistency can significantly reduce developers' "it works on my machine" problem.
5. **Node-RED support**: Node-RED's Docker support and documentation are excellent. The Node-RED documentation includes a straightforward and detailed guide on how to install the software with Docker. This is

Sample eBook content

the guide that I based my process on. See more here: https://nodered.org/docs/getting-started/docker.

While Raspberry Pi, other Linux computers, and VMs are all viable options for installing a Node-RED server, Docker's portability, resource efficiency, scalability, and consistency make it an excellent choice for development and production environments. With Docker, you can build, test, and deploy Node-RED applications quickly and confidently on any hardware that supports it, be it an embedded computer, a laptop, a local server, or a cloud server. In the next chapter, you will learn how to install a Node-RED server using the Docker option.

Sample eBook content

7. Node-RED dashboard

The Node-RED Dashboard is a powerful tool that enables the creation of interactive graphical user interfaces for your Node-RED applications. It provides a set of nodes that allow you to create live graphs, charts, gauges, form inputs, and other UI components. This chapter provides an introduction to the Node-RED Dashboard, its main features, and its use.

In the screenshot below, you can see a dashboard that contains a variety of widgets from my course "Node-RED ESP32: Make a Terrarium Controller". In this example, you can see these types of widgets:

- Switch.
- Numeric.
- Gauge.
- Chart.
- Text.

Figure 1.7.not set.1: An example Node-RED dashboard

What is the Node-RED Dashboard?

The Node-RED Dashboard is an add-on module that provides a set of nodes in Node-RED to create live dashboards. It's a visual tool for wiring the

Sample eBook content

Internet of Things (IoT), but it can also be used for other types of applications to visualize data and interact with your Node-RED server.

The Node-RED Dashboard offers a range of UI components:

- **Buttons**: Buttons are simple UI elements that can send a message when clicked.
- **Sliders & Numeric Inputs**: These allow users to input numeric values in various ways.
- **Text Input & Dropdown**: Text input fields and dropdown selectors provide more ways for users to interact with your flows.
- **Charts & Gauges**: These nodes allow you to display data in various graphical formats.
- **UI Control**: This node can be used to dynamically control the layout and content of the dashboard.

Setting up the Node-RED Dashboard

To start using the Node-RED Dashboard, you need to first install it. This can be done directly from the Node-RED interface by navigating to 'Manage palette' from the menu, then searching for `node-red-dashboard` in the 'Install' tab.

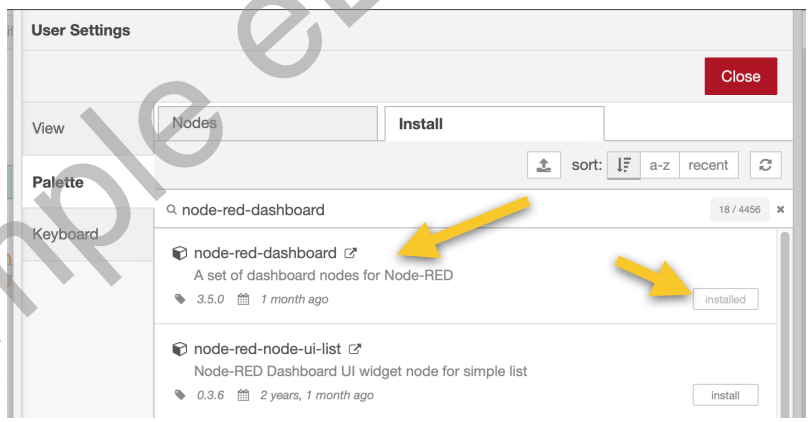


Figure 1.7.not set.2: Install the Dashboard nodes.

After installation, a new set of nodes labeled 'dashboard' will be available in the palette. You can start creating your dashboard by dragging and dropping these nodes onto your flow. You can see some of the installed Dashboard nodes in the screenshot below.

Sample eBook content

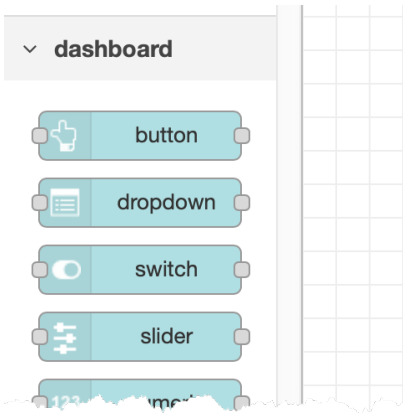


Figure 1.7.not set.3: Some of the installed Dashboard nodes.

Using the Node-RED Dashboard

Creating a dashboard involves creating a flow (or you can add a dashboard in an existing flow), and adding input and/or output nodes that interact with other dashboard nodes. Once you've created a flow with dashboard nodes and deployed it, you can view your dashboard by appending `/ui` to your Node-RED URL in your browser (e.g., `http://localhost:1880/ui`). You can also click on the Dashboard button that you can find in the Dashboard tab of the Information in the side bar (see below).

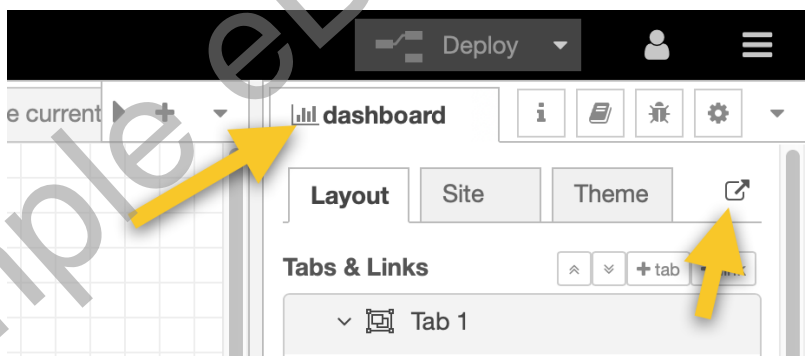


Figure 1.7.not set.4: You can open the. Dashboard from the Side bar.

Each dashboard node provides different configuration options, such as the group and tab it belongs to, size and labels. Experimenting with these options will help you create a unique and interactive dashboard. The dashboard is designed to create a single-page application, which means all interactions occur on a single web page.

The Node-RED Dashboard is a powerful tool for creating user interfaces for your Node-RED applications. It provides an easy and intuitive way to visualize data and interact with your flows. Whether you're building a home automation system, a data monitoring application, or an IoT project, the Node-RED Dashboard can be a valuable addition to your toolset.

Sample eBook content

7.1. Text input and output

One of the powerful features of Node-RED Dashboard is its ability to provide user interface elements, or widgets, that interact with your flows. Two of these widgets are the 'Text Input' and 'Text Output' nodes, which allow for the entry and display of text within your dashboard, respectively. This segment will give you an introduction to these nodes, detailing their configuration and use.

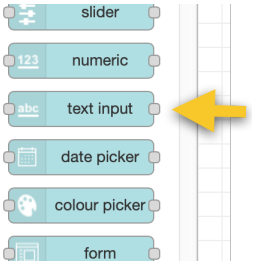


Figure 1.7.1.5: The Text Input node in the palette.

Text Input Node

The 'Text Input' node provides an interface for users to enter data that can be used within your flows. It can be used to receive simple text input, but also supports more complex types such as a date picker or color picker.

To configure a 'Text Input' node:

1. Drag and drop a 'Text Input' node from the dashboard category in the node palette to your workspace.
2. Double-click the node to open its configuration settings.

Sample eBook content

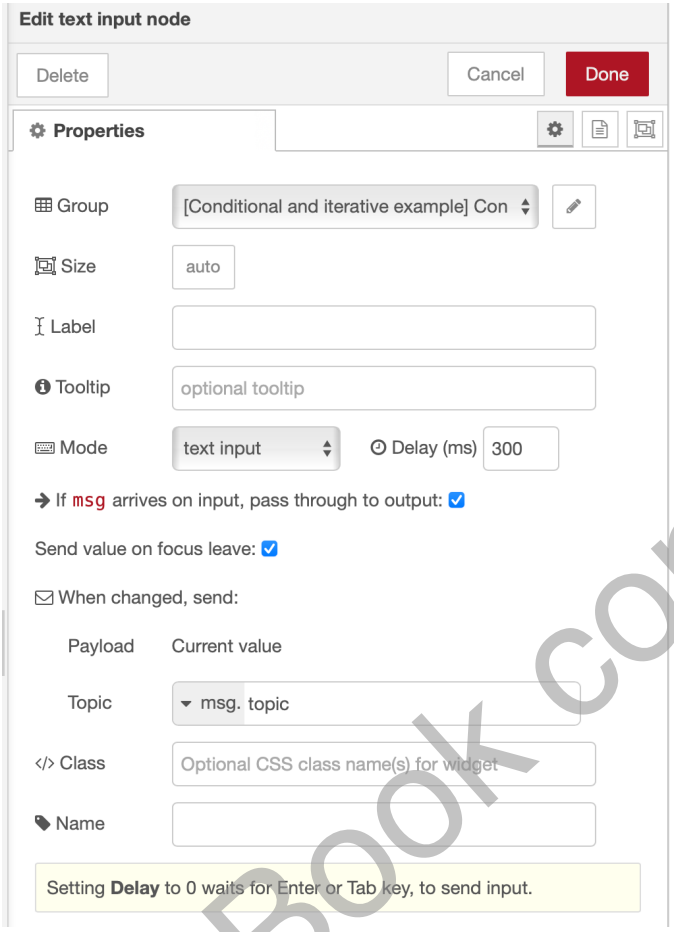


Figure 1.7.1.6: The Text Input node configuration options.

3. Set the 'Group' field to assign the node to a user interface group. You can either select an existing group from the drop-down menu, or click on the pen button to enter the Group dialogue box where you can create a new group and assign the group (and all its members) to a dashboard tab. The Dashboard may contain multiple tabs.

4. Under 'Mode', select the type of input you want (text, password, email address, number, date, time, color, and many types of input text).

5. Optional fields such as 'Label' and 'Placeholder' can be filled to provide additional information for the user interface.

6. Click 'Done' to save your settings.

The 'Text Input' node sends a message every time the user interacts with the input field in the dashboard. The message's payload contains the user's input.

Sample eBook content

Text Output Node

The 'Text Output' node provides a way to display text in the dashboard. This text can be static, but more commonly, it is dynamically updated based on the flow's operations.

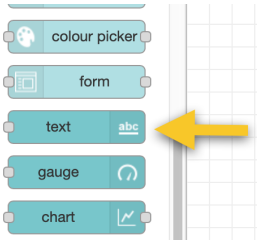


Figure 1.7.1.7: The Text Output node in the Palette.

To configure a 'Text Output' node:

1. Drag and drop a 'Text Output' node from the dashboard category in the node palette to your workspace.
2. Double-click the node to open its configuration settings.

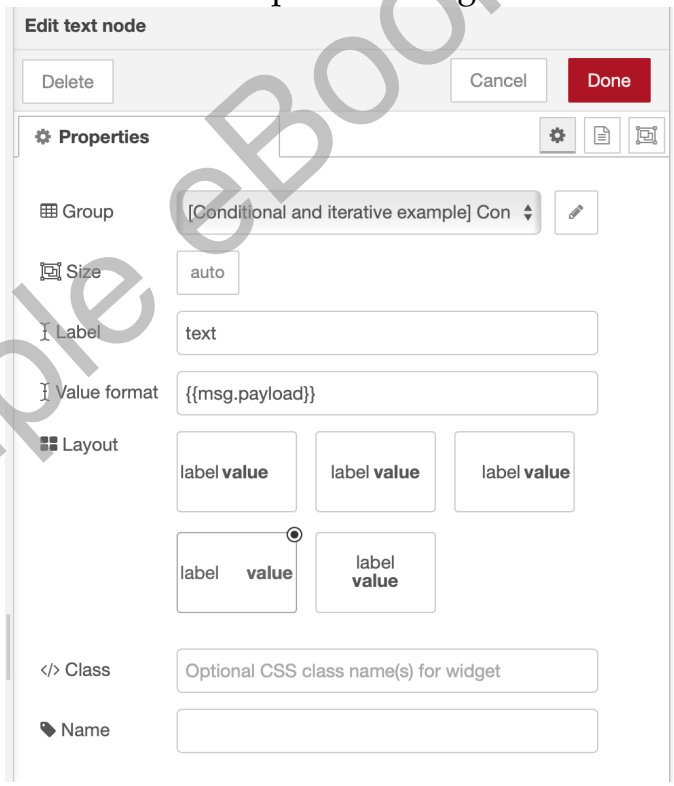


Figure 1.7.1.8: The Text Output configuration options.

3. Set the 'Group' field to assign the node to a user interface group.

Sample eBook content

4. In the 'Value Format' field, enter a Mustache template (more about Moustache in a moment) to format the displayed text. The default is `{{msg.payload}}`, which will display the payload of the input message.

5. Optional settings like 'Label' and 'Name' can be used to provide additional context in the dashboard.

6. Click 'Done' to save your settings.

The 'Text Output' node displays the input message's payload in the dashboard, formatted according to the 'Value Format' field.

About Mustache

"Mustache" is a simple web template system. It's often referred to as a "logic-less" system because it lacks the explicit control flow statements, like if and else conditionals or for loops, found in other templating engines. Instead, Mustache templates use tags, denoted by double curly braces (i.e., “{{“ and “}}”), which get replaced with actual values at runtime.

The name "Mustache" comes from the heavy use of curly braces, which resemble a mustache when seen from the side. Mustache can be used for HTML, config files, source code - basically anything. It works by expanding tags in a template using values provided in a hash or object.

In the context of Node-RED dashboards, a Mustache template is often used to dynamically generate the HTML content displayed on the dashboard.

Here's an example of how you might use a Mustache template in Node-RED. Let's say you have a data object like this:

```
javascript
{
  "name": "John",
  "age": 30
}
```

You could create a Mustache template in Node-RED that looks like this:

```
<p>Hello, {{name}}. You are {{age}} years old.</p>
```

When the data object is passed into the Mustache template, it will replace the {{name}} and {{age}} tags with the corresponding values from the data object, resulting in the following HTML:

Sample eBook content

```
<p>Hello, John. You are 30 years old.</p>
```

This resulting HTML can then be displayed on your Node-RED dashboard.

In a more complex scenario, you might have an array of objects, and you could use Mustache to iterate over them and generate HTML for each one. For instance, consider the following data:

```
{
  "people": [
    {"name": "John", "age": 30},
    {"name": "Jane", "age": 28}
  ]
}
```

You could use the following Mustache template to generate HTML for each person:

```
<ul>
  {{#people}}
    <li>{{name}} is {{age}} years old.</li>
  {{/people}}
</ul>
```

This would result in the following HTML:

```
<ul>
  <li>John is 30 years old.</li>
  <li>Jane is 28 years old.</li>
</ul>
```

Node-RED uses Mustache syntax in the template node to access the properties of the incoming message object (msg.payload, msg.topic, etc.) as well. It is a vital component of the Node-RED ecosystem. You can learn more about the Mustache templating system on its website: https://mustache.github.io.

Using Text Input and Output Nodes Together

Sample eBook content

A simple use case for the 'Text Input' and 'Text Output' nodes is to create a flow where the user enters text, some processing occurs, and the result is displayed.

1. Connect a 'Text Input' node to a function node that modifies the input in some way.
2. Connect the function node to a 'Text Output' node.
3. Deploy the flow.

Now, when a user enters text into the 'Text Input' field in the dashboard, the text will be processed by the function node and the result displayed in the 'Text Output' field.

In summary, the 'Text Input' and 'Text Output' nodes in the Node-RED Dashboard provide powerful and flexible ways to interact with your flows. Understanding and using these tools effectively can greatly enhance your Node-RED applications.

Example

Let's create a simple flow where we'll take text input from the user, process it, and display it in the dashboard as well as debug it.

1. **Setup the Dashboard.** Open the Dashboard properties tab from the right side of the editor. You want to create a new tab, and add a widget group to the tab. Click “+tab” to add a tab, and then “+group” to add a group.

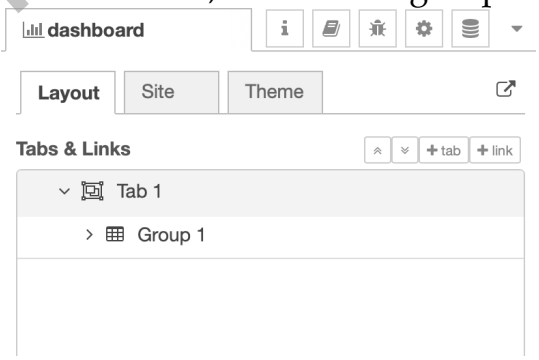


Figure 1.7.1.9: Create a new Dashboard tab.

2. **Text Input Node**: First, drag and drop a 'Text Input' node onto your workspace. Double-click on the node to open its configuration settings. Set the 'Group' field to assign the node to a user interface group. Under 'Mode', select 'text input'. Leave other settings as default and click 'Done'. Setup the properties like in this example, including the Group (set to Group 1):

Sample eBook content

Figure 1.7.1.10: Create a new Dashboard tab.

3. **Function Node**: Drag and drop a 'Function' node onto your workspace. Double-click on it to open its configuration settings. In the 'Function' field, write a simple function to process the input. For example, let's add a prefix to the input text:

```
msg.payload = "User said: " + msg.payload;
    return msg;
```

Click 'Done'.

4. **Text Output Node**: Next, drag and drop a 'Text Output' node onto your workspace. Double-click on it to open its configuration settings. Set the 'Group' field to the same group as your 'Text Input' node. Leave the 'Value Format' field as default (`{{msg.payload}}`) and click 'Done'. Set up this widget like this, including the Group:

Sample eBook content

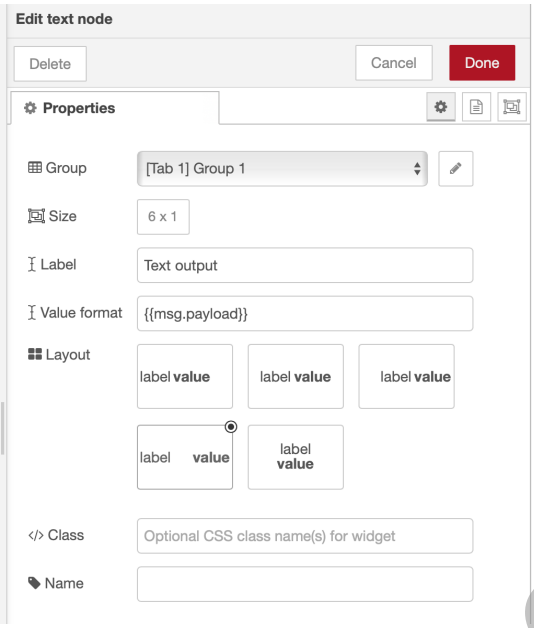


Figure 1.7.1.11: Setup the Text Output node.

5. Debug Node: Drag and drop a 'Debug' node onto your workspace. No configuration is needed for this node.

6. Now, connect the output port of the 'Text Input' node to the input port of the 'Function' node. Then connect the output port of the 'Function' node to the input ports of both the 'Text Output' and 'Debug' nodes.

7. In Dashboard layout, arrange the two dashboards like this:

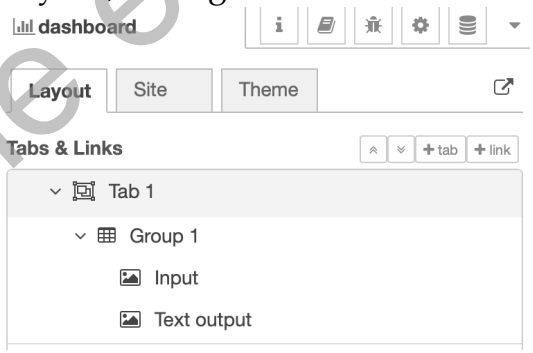


Figure 1.7.1.12: Open the dashboard.

8. Next, click 'Deploy' to deploy the flow.

10. Click on the Open Dashboard button to open the Dashboard in a new browser window.

With this flow, when a user enters text into the 'Text Input' field in the dashboard, the text will be processed by the function node and the result displayed in the 'Text Output' field. The processed text will also be output to the debug tab. Here is a screenshot of this dashboard in the browser:

Sample eBook content

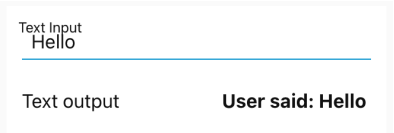


Figure 1.7.1.13: The dashboard we just created in the browser.

It is possible to define the exact location and size of each browser in the dashboard by using the Size attribute in the dashboard node's properties window. For example, I have set the text output widget to occupy and area of six cells width and one cell height by defining this area in the Size field:

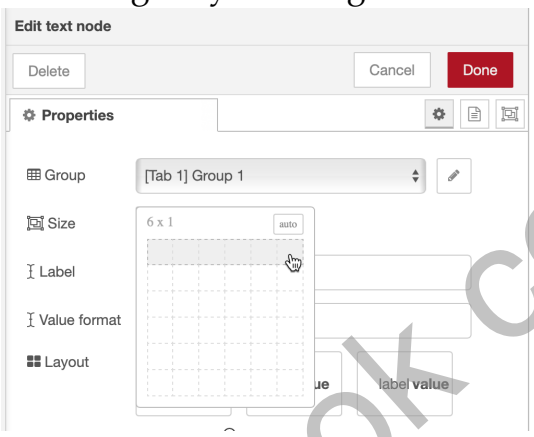


Figure 1.7.1.14: Define the size of the text widget.

You can further control the ordering of the widgets in the dashboard via the Tabs & Links box in the Layout tab of the Dashboard tab in the side bar. Use the handles in the left of each widgets to rearrange them:

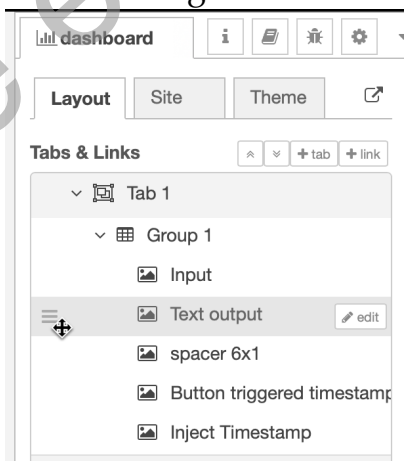


Figure 1.7.1.15: You can re-arrange the order of Dashboard widgets.

7.2. The button

Button widgets are one of the most straightforward yet powerful tools in the Node-RED Dashboard. They allow users to interact with a flow clearly and intuitively. This section will introduce the button widget, including its configuration and use.

Sample eBook content

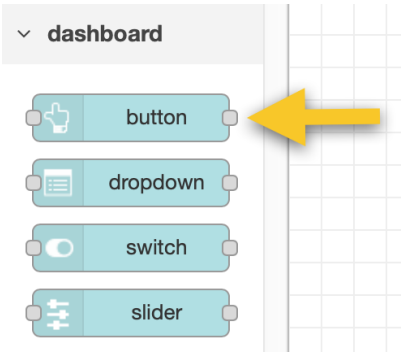


Figure 1.7.2.16: The Button node in the palette.

What is a Button Widget?

A button widget in Node-RED is a user interface element that sends a predefined message each time it is clicked or tapped. This makes it an excellent tool for triggering actions in your flow based on user interaction.

Setting Up a Button Widget

To use a button widget in Node-RED:

1. Drag and drop a "Button" node from the dashboard category in the node palette onto your workspace.
2. Double-click the node to open its configuration settings.

Sample eBook content

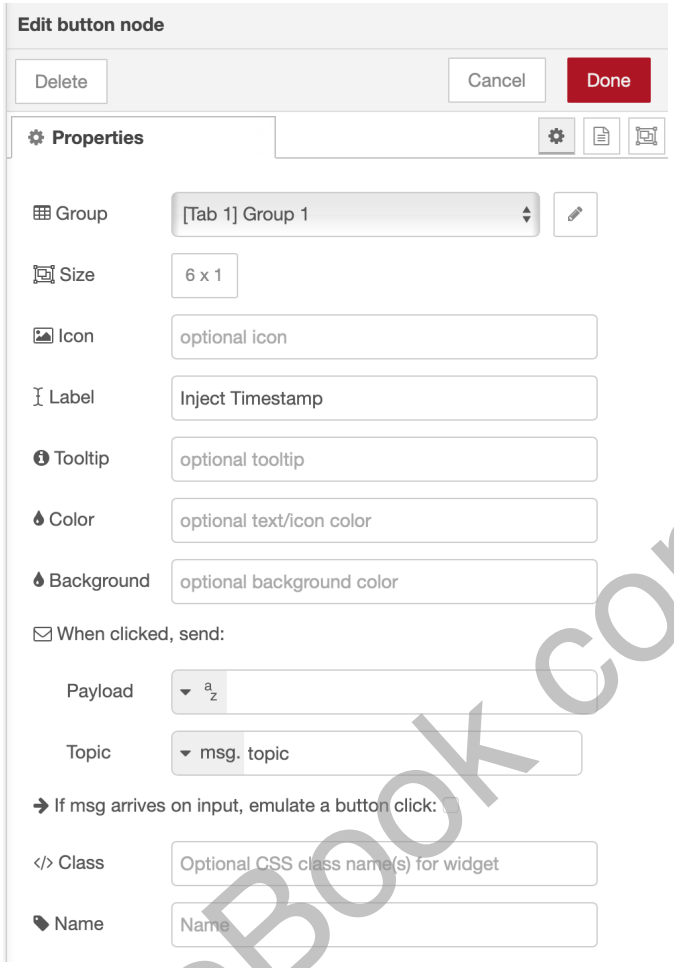


Figure 1.7.2.17: The Button node configuration options.

3. Set the "Group" field to assign the node to a user interface group.

4. In the "Label" field, provide a name for the button displayed on the dashboard.

5. In the "Payload" field, enter the message you want the button to send each time it is clicked. This can be a static value or a JavaScript expression that generates a dynamic value.

6. Set the "Payload Type" to the type of the payload value. This is usually "string" or "number" for static values. For JavaScript expressions, this should be set to "expression".

7. In the "Name" field, enter a name for the node. This name will be displayed in the workspace but not on the dashboard.

8. Click "Done" to save your settings.

Using a Button Widget

Once you've configured your button widget and deployed your flow, you can interact with it on your dashboard. Each time you click the button, it

Sample eBook content

will send a message with its configured payload. You can use the button widget to trigger any action in your flow. This could be turning on a light, starting a timer, sending a request to a web server, or any other action that can be triggered by a message.

For example, suppose you want to create a flow that turns on a lamp when a button is clicked. You could configure a button node with a payload of "on", connect it to a function node that translates "on" to the appropriate command for your light, and then connect that to an output node that sends the command to your lamp.

Button widget demonstration

Let’s create a simple flow using a button widget to trigger an action. In this case, we’ll trigger the injection of a timestamp.

1. Drag and drop a "Button" node onto your workspace from the dashboard section—Double-click on it to open its configuration settings. Set the "Group" field to assign the node to a user interface group. In the "Label" field, provide a name for the button, say "Inject Timestamp". In the "Payload" field, set it to "String" and leave the value empty. You can experiment with other payload types, including the timestamp (use a debug node to see the value). Click "Done".

Sample eBook content

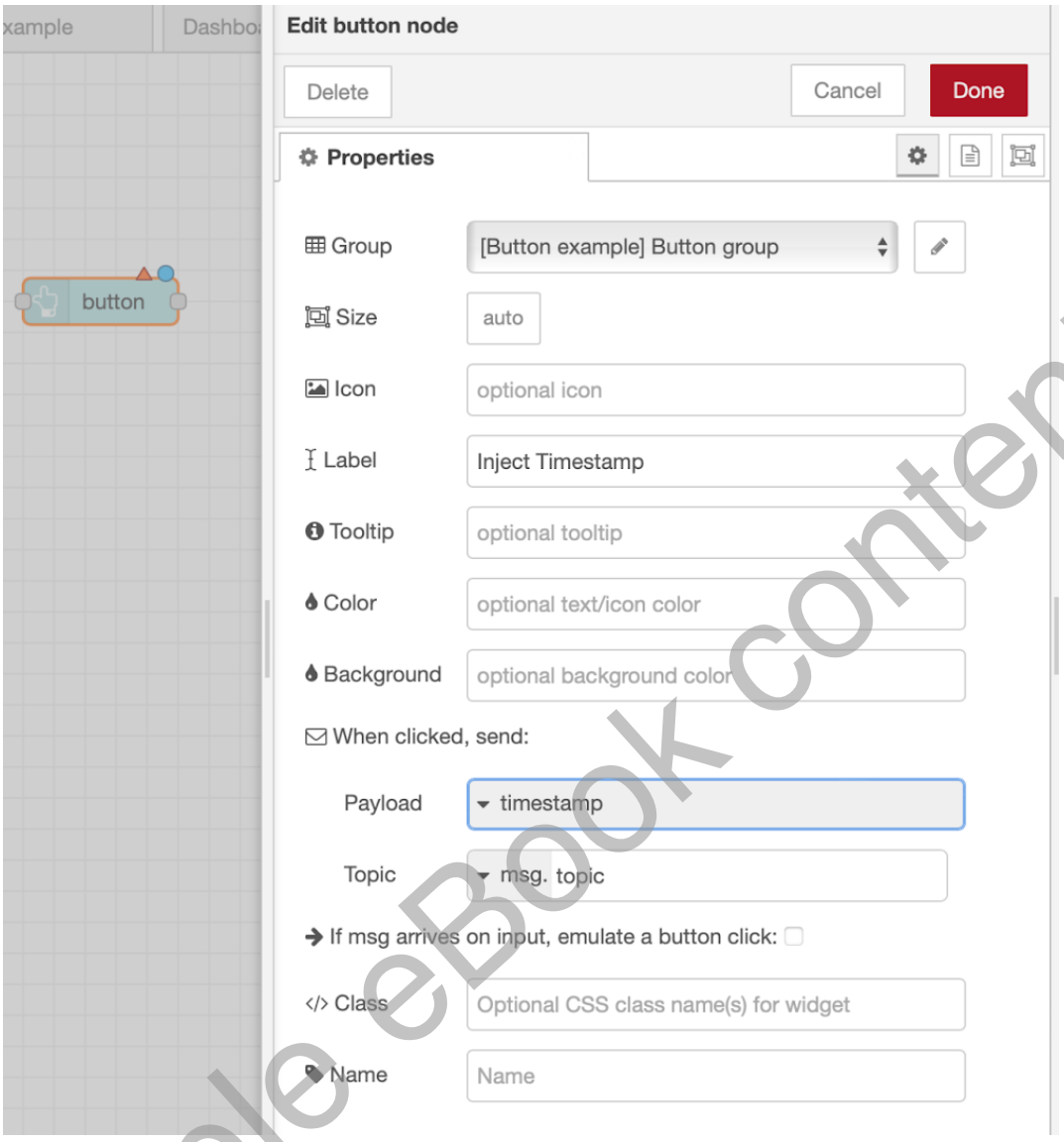


Figure 1.7.2.18: The example button node properties.

2. Drag and drop a "Function" node onto your workspace. Double-click on it to open its configuration settings. In the "Function" field, write a simple function to generate the current timestamp:

```
msg.payload = new Date().getTime();
return msg;
```

1. Click "Done".
2. Drag and drop a "Debug" node onto your workspace. No configuration is needed for this node.
3. Connect the output port of the "Button" node to the input port of the "Function" node. Then connect the output port of the "Function" node to the input port of the "Debug" node.

Sample eBook content

4. Drag a text output widget and give it a suitable name, like “Button triggered timestamp”. Add it to the button group.

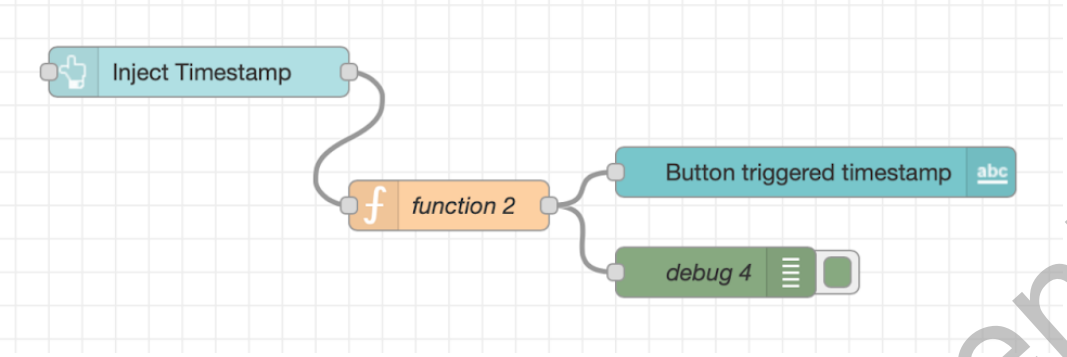


Figure 1.7.2.19: The button widget example flow.

5. Use the Dashboard layout editor to arrange the widgets. You can include a spacer to add a gap between the widgets.

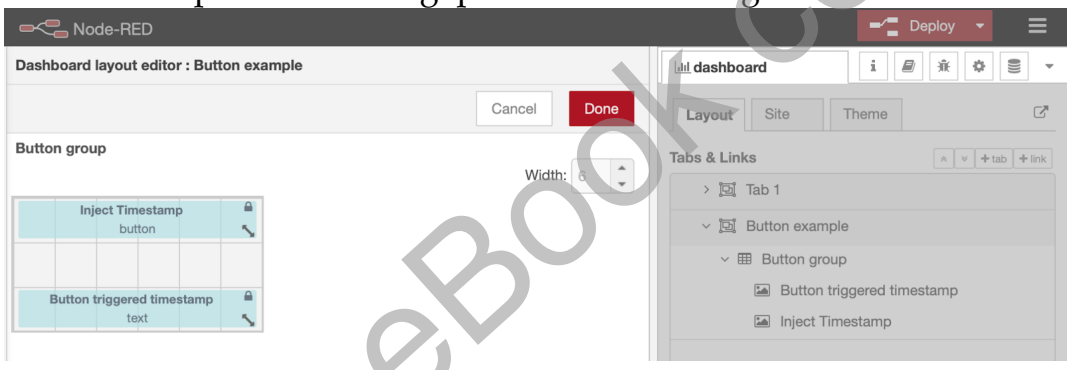


Figure 1.7.2.20: The button widget example dashboard layout.

6. Click "Deploy" to deploy the flow.

With this flow, when a user clicks the "Inject Timestamp" button in the dashboard, the current timestamp will be generated by the function node and output to the debug tab. You can see the dashboard for this flow below:

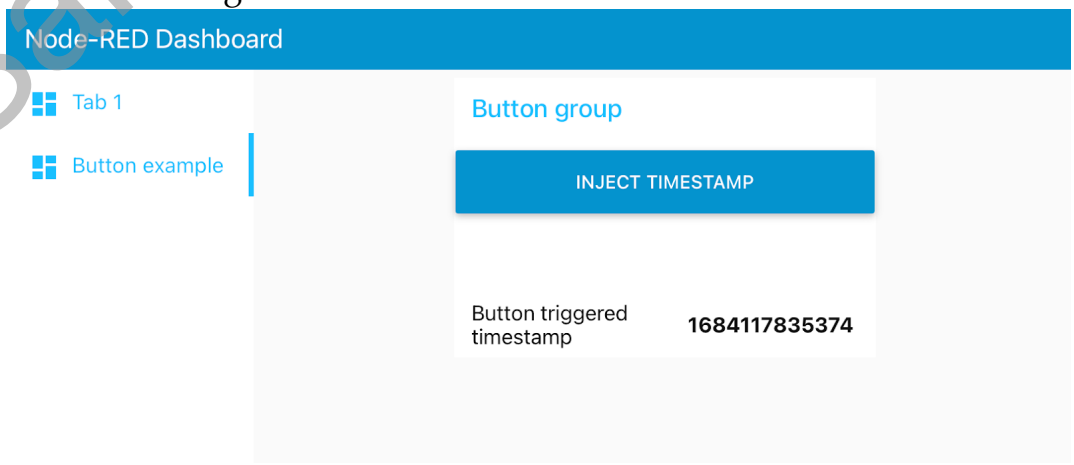


Figure 1.7.2.21: The button widget example dashboard in the browser.

Sample eBook content

7.3. The gauge and slider

The Node-RED Dashboard provides a set of nodes that make it easy to create interactive user interfaces. Among them, the Gauge and Slider widgets are versatile tools that display and control real-time data. This segment will introduce these widgets, including their configuration and usage.

The Gauge widget

The Gauge widget provides a graphical representation of data, perfect for real-time sensor data or system status monitoring.

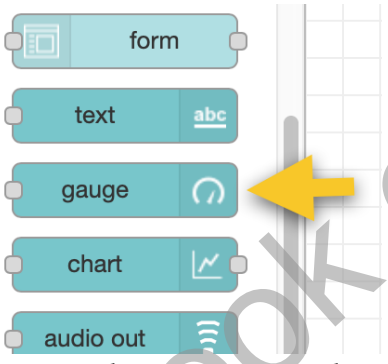


Figure 1.7.3.22: The Gauge node in the palette.

Setting Up a Gauge Widget

Here is how to set up a gauge widget in your Node-RED dashboard.

1. Drop a 'Gauge' node from the Dashboard category in the Palette onto your workspace.
2. Double-click the node to open its configuration settings.

Sample eBook content

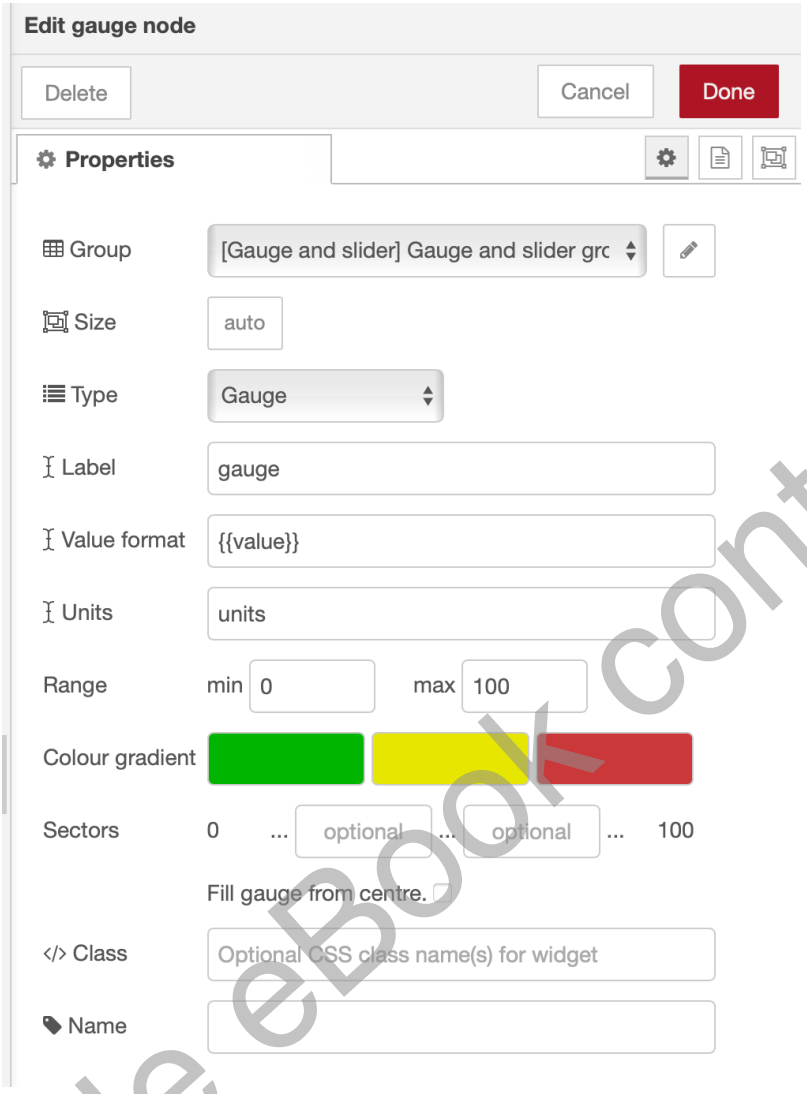


Figure 1.7.3.23: The Gauge widget configuration options.

3. Set the 'Group' field to assign the node to a user interface group.
4. In the 'Label' field, provide a name for the gauge displayed on the dashboard.
5. Set the minimum and maximum values that the gauge will represent.
6. Choose a type for the gauge (Gauge, Donut, Compass, or Wave).
7. Click 'Done' to save your settings.

Using a Gauge Widget

The Gauge widget will display the value of the payload property of any message it receives. For example, if you have a flow that reads the temperature from a sensor and sends it as the payload of a message, connecting this flow to a gauge will display the temperature on your dashboard.

Sample eBook content

The Slider Widget

The Slider widget allows users to input a numeric value by dragging a handle along a track. It's ideal for controlling devices that accept a range of numeric values, like dimming a lamp or setting a temperature.

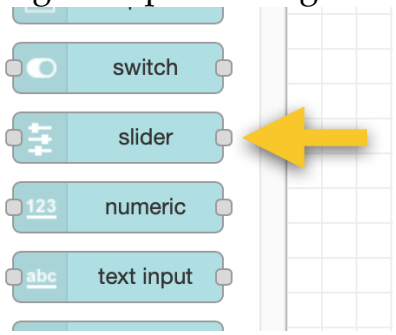


Figure 1.7.3.24: The Gauge node in the palette.

Setting Up a Slider Widget

Here is how to set up a gauge widget in your Node-RED dashboard.

1. Drag and drop a 'Slider' node from the Dashboard category onto your workspace.
2. Double-click the node to open its configuration settings.

Sample eBook content

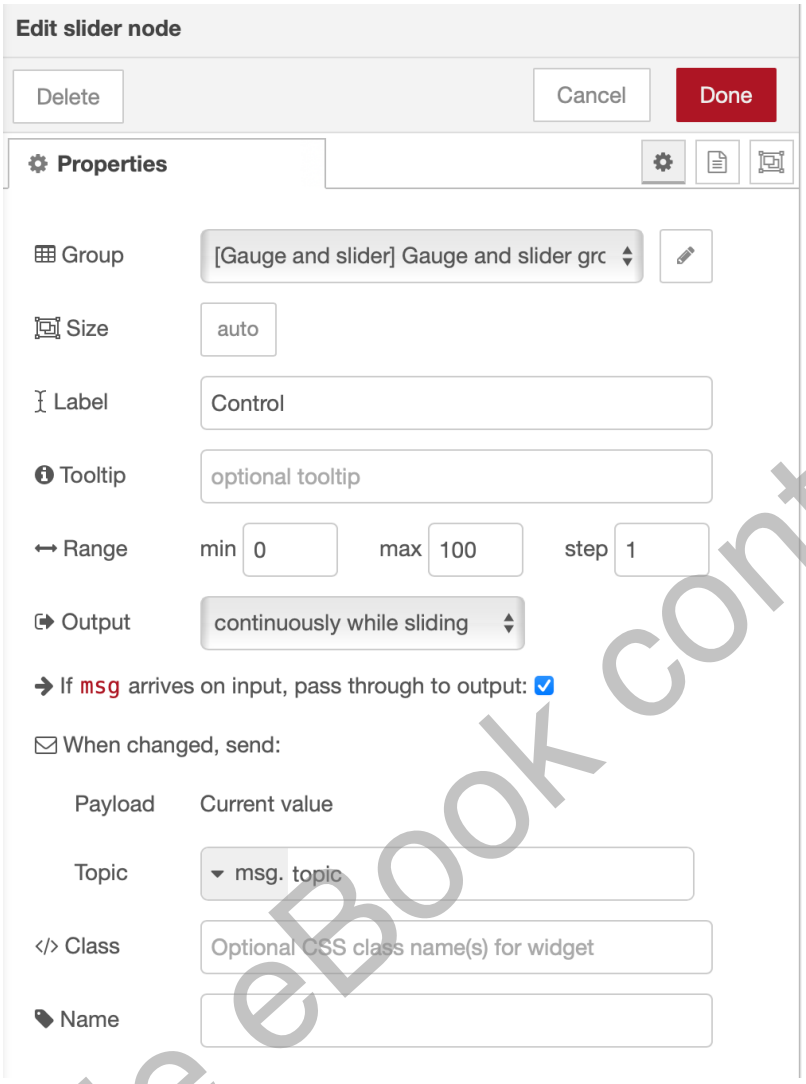


Figure 1.7.3.25: The Slider widget configuration options.

3. Set the 'Group' field to assign the node to a user interface group.
4. In the 'Label' field, provide a name for the slider displayed on the dashboard.
5. Set the minimum and maximum values that the slider will represent.
6. Click 'Done' to save your settings.

Using a Slider Widget

The Slider widget will send a message with the current slider value as the payload each time the slider handle is moved. This makes it easy to incorporate the slider into your flows. For example, you could connect a slider to a function node that translates the slider value into a command for a device, like setting the brightness of a light or the temperature of a thermostat.

The Gauge and Slider widgets in the Node-RED Dashboard provide an easy way to interact with your flows. The Gauge widget lets you visualise

Sample eBook content

data in real time, while the Slider widget allows for interactive control of devices. Together, they can enhance your IoT projects and provide users with a rich, interactive experience.

A demo of the gauge and slider widgets

Let's create a simple flow that utilises a slider to control a value and a gauge to display that value.

1. **Slider Node**: Drag and drop a 'Slider' node from the dashboard section onto your workspace. Double-click the node to open its configuration settings. Set the 'Group' field to assign the node to a user interface group. In the 'Label' field, provide a name for the slider, say "Control". Set the minimum and maximum values per your requirement, say 0 to 100. Click 'Done'.

2. **Gauge Node**: Drag and drop a 'Gauge' node onto your workspace— Double-click on it to open its configuration settings. Set the 'Group' field to assign the node to the same user interface group. In the 'Label' field, provide a name for the gauge, say "Display". Set the minimum and maximum values the same as the slider (0 to 100 in this case). Click 'Done'.

3. Connect the output port of the 'Slider' node to the input port of the 'Gauge' node. Click 'Deploy' to deploy the flow.

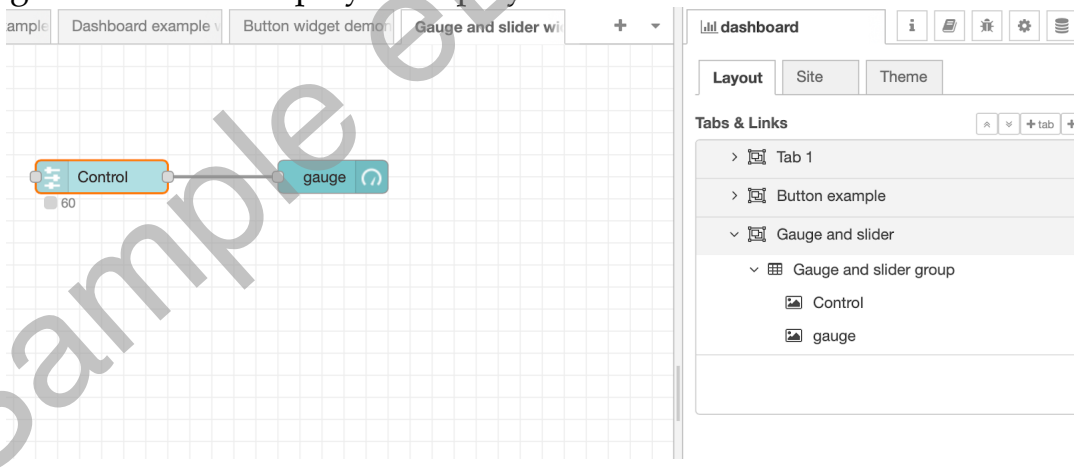


Figure 1.7.3.26: An example flow that combines Gauge and Slider widgets.

With this flow, when a user moves the slider in the dashboard, the gauge will display the slider’s current value. It looks like this:

Sample eBook content

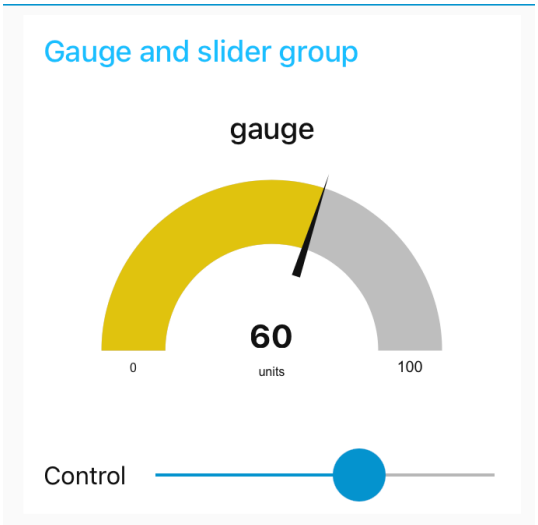


Figure 1.7.3.27: A dashboard that contains a gauge and a slider.

This simple example can be extended to control and monitor devices or systems. For instance, the slider could be used to manage the brightness of a light, with the gauge displaying the current brightness level.

7.4. The switch

The Node-RED Dashboard provides a set of nodes that make it easy to create interactive user interfaces. Among these, the Switch widget is useful for controlling binary states, acting much like a physical switch. This segment will introduce the Switch widget, including its configuration and usage.

The Switch widget in Node-RED is a user interface element that displays a switch on the dashboard. The user can toggle the switch on and off, enabling it to control binary states in your flows. For example, it could turn a light on and off or activate and deactivate a system or process.

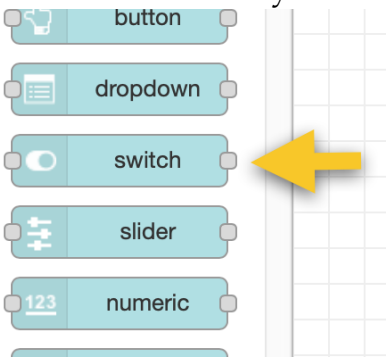


Figure 1.7.4.28: The Switch node in the palette.

Setting Up a Switch Widget

To use a Switch widget in Node-RED:

Sample eBook content

1. Drag and drop a 'Switch' node from the dashboard category in the node palette onto your workspace.

2. Double-click the node to open its configuration settings.

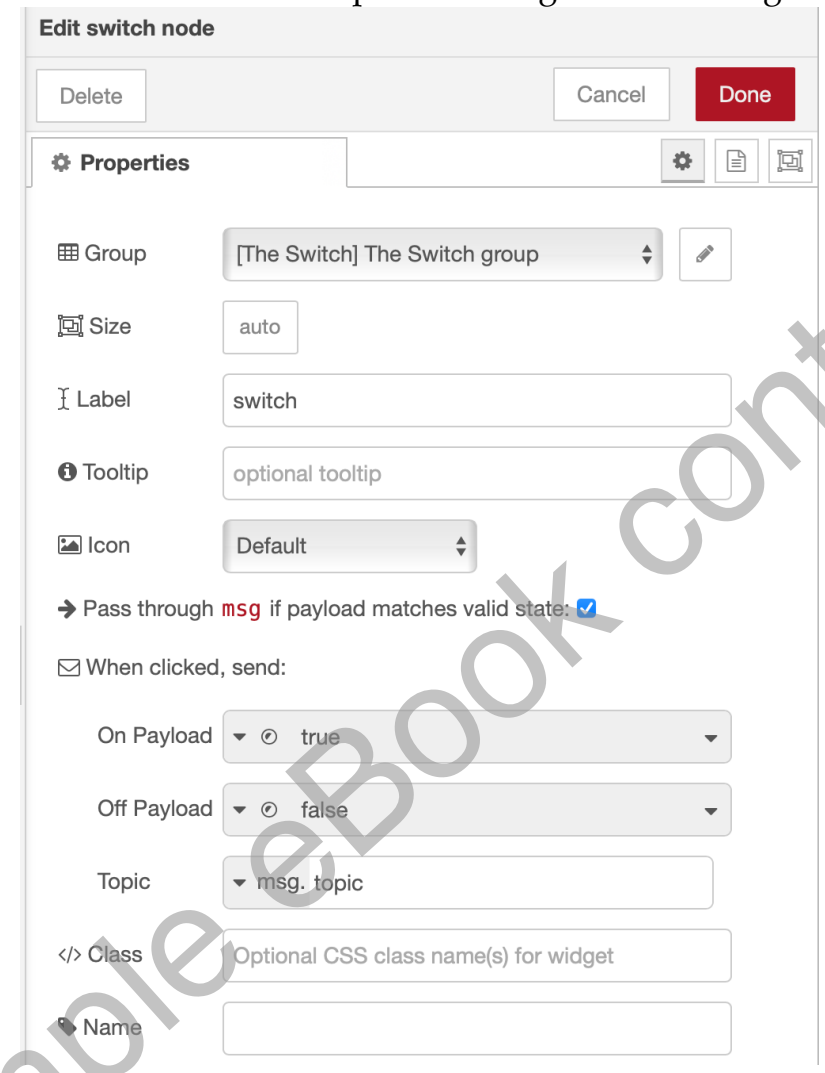


Figure 1.7.4.29: The Switch widget configuration options.

3. Set the 'Group' field to assign the node to a user interface group.

4. In the 'Label' field, provide a name for the switch displayed on the dashboard.

5. In the 'Payload' fields for 'On' and 'Off', enter the messages you want the switch to send when turned on and off, respectively. These messages can be static values or dynamic expressions.

6. Set the 'Payload Type' fields for 'On' and 'Off' to match the type of your payload values.

7. In the 'Name' field, enter a name for the node. This name will be displayed in the workspace but not on the dashboard.

8. Click 'Done' to save your settings.

Sample eBook content

Using a Switch Widget

Once you've configured your Switch widget and deployed your flow, you can interact with it on your dashboard. When you toggle the switch on, it will send its 'On' payload, and when you toggle it off, it will send its 'Off' payload.

You can use these payloads to control any binary state in your flow. For example, you could connect the switch to an output node that controls a lamp. When the switch sends its 'On' payload, the light turns on, and when it sends its 'Off' payload, the light turns off.

The Switch widget in the Node-RED Dashboard is a powerful tool for interacting with binary states in your flows. By configuring a switch with appropriate 'On' and 'Off' payloads, you can give users control over these states straightforwardly and intuitively. Whether you're creating a home automation system, a control panel for a machine, or any other project that requires binary control, the Switch widget can be an invaluable tool.

An example flow with the switch widget

Here's an example flow using the switch widget, inject, debug, and function nodes.

1. Switch Node: Drag and drop a 'Switch' node from the Dashboard category onto your workspace. Double-click the node to open its configuration settings. Set the 'Group' field to assign the node to a user interface group. In the 'Label' field, provide a name for the switch displayed on the dashboard, say "Light Control". Set the “On Payload” to "true" and “Off Payload” to "false". Click “Done”.

2. Function Node: Drag and drop a 'Function' node onto your workspace. In this function, we'll interpret the boolean payload from the switch node and turn it into a string representing the light's state. Double-click the node to open its configuration settings. You might write a function like this:

```
if (msg.payload === true) {
      msg.payload = "The light is ON";
} else if (msg.payload === false) {
      msg.payload = "The light is OFF";
}
```

Sample eBook content

```
return msg;
```

Click 'Done'.

3. Debug Node: Drag and drop a “Debug” node onto your workspace. This will allow you to view the output of the function node in the debug sidebar.

4. Add a text output widget to see the switch value in the dashboard.

4. Now connect the output of the “Switch” node to the input of the “Function” node and the output of the 'Function' node to the input of the 'Debug' node.

5. Click 'Deploy' to deploy the flow. You can see the flow for this example below.

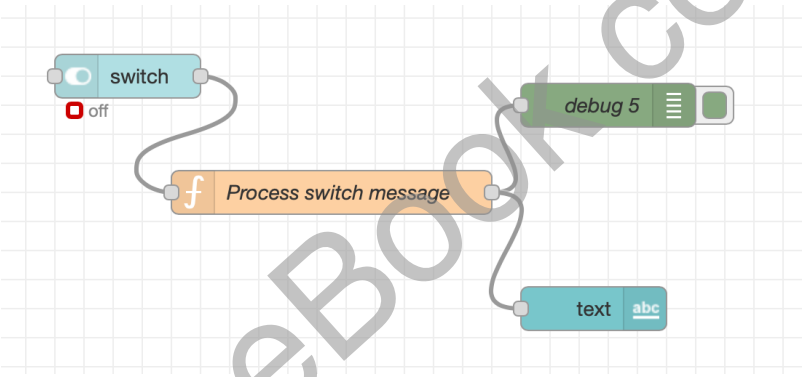


Figure 1.7.4.30: The Switch widget example flow.

You’ll see a switch labelled “Light Control” when you go to your Node-RED dashboard. You can see the dashboard below.

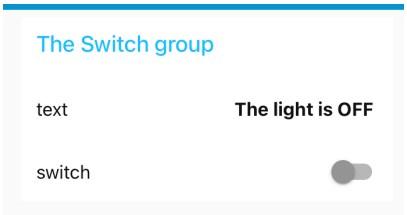


Figure 1.7.4.31: The Switch widget example dashboard.

Toggling this switch will send a message with a payload of "true" or "false" to the function node. The function node will translate this into "The light is ON" or "The light is OFF" and send this message to the debug node. The debug node will then print this message to the debug sidebar.

Sample eBook content