Hands-On Internet of Things with MQTT

Build connected IoT devices with Arduino and MQ Telemetry Transport (MQTT)

Packt
www.packt.com

Tim Pulver

Hands-On Internet of Things with MQTT

Build connected IoT devices with Arduino and MQ Telemetry Transport (MQTT)

Tim Pulver

Packt

BIRMINGHAM - MUMBAI

Hands-On Internet of Things with MQTT

Copyright © 2019 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

Commissioning Editor: Gebin George
Acquisition Editor: Rohit Rajkumar
Content Development Editor: Ronn Kurien
Senior Editor: Rahul Dsouza
Technical Editor: Mohd Riyan Khan
Copy Editor: Safis Editing
Project Coordinator: Vaidehi Sawant
Proofreader: Safis Editing
Indexer: Pratik Shirodkar
Production Designer: Jyoti Chauhan

First published: October 2019

Production reference: 1041019

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham
B3 2PB, UK.

ISBN 978-1-78934-178-2

`www.packt.com`

Packt.com

Subscribe to our online digital library for full access to over 7,000 books and videos, as well as industry leading tools to help you plan your personal development and advance your career. For more information, please visit our website.

Why subscribe?

- Spend less time learning and more time coding with practical eBooks and Videos from over 4,000 industry professionals
- Improve your learning with Skill Plans built especially for you
- Get a free eBook or video every month
- Fully searchable for easy access to vital information
- Copy and paste, print, and bookmark content

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.packt.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at customercare@packtpub.com for more details.

At www.packt.com, you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on Packt books and eBooks.

Contributors

About the author

Tim Pulver is a Berlin-based freelance interaction designer and developer. In his work, he combines his physical prototyping experience and knowledge of interface design with modern technologies such as 3D printing, laser cutting, web technologies, and machine learning to create unique interactive experiences.

In recent years, he has worked on interactive data visualizations, web-based audiovisual experiences, musical interfaces, and cables— an innovative browser-based visual programming language that enables the creation of interactive audiovisual prototypes without writing any code.

He holds a Bachelor of Arts degree in interface design from the University of Applied Sciences in Potsdam, Germany.

I wish to thank Naree. Without your support, this book would not have been possible.

About the reviewer

Federico Gonzalez is an Argentina-based cooperative developer and teacher. He studied information systems engineering at UTN with a focus on software development. He works at Devecoop (his cooperative). He has worked on many projects, with his current focus on developing software and teaching React.js. He has contributed to open source projects such as Lelylan (an IoT cloud platform with the microservices architecture) and eventoL (conference and install fest management software), and has made some minor contributions to projects with Docker environment configuration, Python, and JavaScript code. He has also presented various workshops at universities, conferences, and companies in Argentina featuring React.js, Python, Docker, open source, free software, and cooperatives.

Packt is searching for authors like you

If you're interested in becoming an author for Packt, please visit `authors.packtpub.com` and apply today. We have worked with thousands of developers and tech professionals, just like you, to help them share their insight with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Table of Contents

Preface

First and foremost, I would like to thank you for purchasing this book and going on a journey of **Internet of Things** (**IoT**) prototyping with me.

In my teenage years, before the first Arduino was invented, I had a friend who was knowledgeable about electronics. He was considered a nerd and really knew what he was doing. Looking at his soldered prototypes, which consisted of a lot of cables, chips, and other electronic components that I didn't know about at the time, I was fascinated. I did not have any idea where to start building electronic prototypes myself; it seemed like such a huge field that there was no way for me to find an entry point. At that time, it was very far from my imagination that I would later be able to create prototypes myself, and, even more so, that I would write a book about the topic one day.

A lot of things have changed since then. One of those things is the creation of Arduino and Processing, which made it possible to learn programming and hardware prototyping without any prior knowledge. The field opened up to designers, artists, and makers. It is now easier than ever to get started with hardware prototyping and bring your own ideas into reality.

While building physical projects alone creates a lot of possibilities, being able to use internet-connected devices and connect them to each other and the rest of the web opens up a whole new world of possibilities.

There are many ways to connect two internet-connected devices and exchange information between them. One of these possibilities, which in my opinion is the easiest and most open, is MQTT. Being *open* in this sense means that third-party developers can create apps and libraries for it. There is a vast ecosystem of tools, libraries, and apps that all speak the same language of MQTT.

Building internet-connected devices has never been easier, and using MQTT to build them makes it possible to prototype even faster, while also having access to all these third-party tools.

When I was initially looking into possibilities for how to let my prototypes talk to each other over the internet, MQTT stood out, but I could not find much information about it. It seemed like a niche topic and I could not understand why.

If you want to create your own inventions, experimental prototypes, and custom devices, getting to know MQTT is a great decision, and I hope that, by reading this book, you will benefit from a smooth entry into the world of IoT prototyping using MQTT.

Welcome on this journey!

Who this book is for

This book is ideal for readers who want to start creating internet-connected projects themselves, but only have a little bit of experience of programming and using the Arduino IDE. You do not need to have a computer science background to follow along. Simply curiosity and a basic understanding of programming in Arduino will be enough. If you understand basic programming concepts such as variables, loops, conditions, and functions, then you will have no problem following along.

Some of the chapters touch on a lot of different topics. I do not expect you to understand every single bit in detail. The goal here is to give you a general overview of related technologies and techniques, so you can dive into any of the topics after finishing the book. You can look at it as a starting point for IoT prototyping.

For an intermediate or professional programmer, the pace of the hands-on projects in this book might feel a bit slow, but there is plenty of relevant information for you in the theoretical chapters as well, including how to use MQTT.

What this book covers

`Chapter 1`, *The Internet of Things in a Nutshell*, gives you an overview of fields where IoT plays an important role. It introduces you to smart cars, the **Industrial Internet of Things** (**IIoT**), and smart homes.

`Chapter 2`, *Basic Architecture of an IoT Prototype*, introduces you to related technologies and concepts, as well as microcontroller recommendations for IoT prototyping. This chapter touches on a lot of topics and should be seen as a starting point for your own learning, after which you can dig deeper into the topics that interest you.

`Chapter 3`, *Getting Started with MQTT*, explains the concepts behind MQTT, the lightweight IoT protocol we will be using throughout the entire book. While we only use a subset of the features explained in this chapter during the hands-on projects, you can use it as a reference to go back to if you want to create more advanced MQTT projects later. Feel free to create a bookmark for this chapter, as you will hopefully re-read fragments from it.

`Chapter 4`, *Setting Up a Lab Environment*, explains how to install the software needed for the hands-on projects. It also includes a shopping list with all the necessary hardware for the projects.

`Chapter 5`, *Building Your Own Automatic Pet Food Dispenser*, is the first hands-on project in the book. It will show you, using a servo motor, how to control an automatic dispenser using MQTT, either from your computer or smartphone using one of the apps introduced in `Chapter 3`, *Getting Started with MQTT*. You can fill the dispenser with pet food, sweets, cereals, or whatever you like.

`Chapter 6`, *Building a Smart E-Ink To-Do List*, is the second hands-on project in the book. In this chapter, you will learn how to use an energy-efficient e-ink display and send text to it using MQTT. The display could, for example, be hung next to your front door to remind you to take out the trash or buy milk.

`Chapter 7`, *Building a Smart Productivity Cube, Part 1*, is the third and final hands-on project in the book. It explains how to create an orientation sensor from scratch using simple electrical components called tilt switches. The cube can then be used to record the time you spend on various activities. Because all three projects use MQTT, they can also be chained together, such that activating the smart productivity cube might show some text on the e-ink display and activate the food dispenser.

`Chapter 8`, *Building a Smart Productivity Cube, Part 2*, is the second part of *building a smart productivity cube*. Based on the prototype that we built together in the previous chapter, we will add MQTT connectivity and make use of third-party MQTT clients to receive and display the data coming from your prototype.

`Chapter 9`, *Presenting Your Own Prototype*, introduces technologies such as laser cutting and 3D printing to build great-looking and sturdy cases. It will give you an idea of how professional prototypes are made using **Printed Circuit Board** (**PCB**) services and software. While making a product out of your ideas probably seems far-fetched to you, I want you to explore different options and where continuing on this path might lead you.

To get the most out of this book

To get the most out of this book, you'll need to understand basic programming concepts such as variables, conditions, loops, and functions. You might have gained this knowledge by reading an Arduino book for complete beginners before that explained those concepts in detail. There are many books on getting started with Arduino for you to choose from.

If you run into any problems that you cannot solve yourself, feel free to ask for help on the book's repository on GitHub (`https://github.com/PacktPublishing/Hands-on-IOT-with-MQTT`) by opening an issue, and I will try to help you. To do so, you will need to create a GitHub account (if you do not already have one).

You can find all the required pieces of software, along with instructions on how to install them, in `Chapter 4`, *Setting up a Lab Environment*. If you skip the first hands-on project in `Chapter 5`, *Building Your Own Automatic Pet Food Dispenser*, starting directly with the second or third one (in `Chapters 6`, *Building a Smart E-Ink To-Do List*, and `Chapter 7`, *Building a Smart Productivity Cube, Part 1 and* `Chapter 8`, *Building a Smart Productivity Cube, Part 2*), please make sure to also follow the instructions on how to set up MQTT and the Wi-Fi library at the beginning of `Chapter 5`, *Building Your Own Automatic Pet Food Dispenser*.

You will get the most out of this book by reading `Chapter 1`, *The Internet of Things in a Nutshell*, to `Chapter 5`, *Building Your Own Automatic Pet Food Dispenser*, in a linear way. `Chapter 6`, *Building a Smart E-Ink To-Do List*, to `Chapter 9`, *Presenting Your Own Prototype*, can be read in any order you like.

Download the example code files

You can download the example code files for this book from your account at `www.packt.com`. If you purchased this book elsewhere, you can visit `www.packt.com/support` and register to have the files emailed directly to you.

You can download the code files by following these steps:

1. Log in or register at `www.packt.com`.
2. Select the **Support** tab.
3. Click on **Code Downloads**.
4. Enter the name of the book in the **Search** box and follow the onscreen instructions.

Once the file is downloaded, please make sure that you unzip or extract the folder using the latest version of:

- WinRAR/7-Zip for Windows
- Zipeg/iZip/UnRarX for macOS
- 7-Zip/PeaZip for Linux

The code bundle for the book is also hosted on GitHub at `https://github.com/PacktPublishing/Hands-On-Internet-of-Things-with-MQTT`. In case there is an update to the code, it will be updated on the existing GitHub repository. If you want to make sure to have the latest code you should download it from GitHub.

We also have other code bundles from our rich catalog of books and videos available at `https://github.com/PacktPublishing/`. Check them out!

Download the color images

We also provide a PDF file that has color images of the screenshots/diagrams used in this book. You can download it here: `https://static.packt-cdn.com/downloads/9781789341782_ColorImages.pdf`.

Code in Action

Visit the following link to check out videos of the code being run:
`http://bit.ly/2oSjufZ`

Conventions used

There are a number of text conventions used throughout this book.

`CodeInText`: Indicates code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles. Here is an example: "So, often, it is better to rewrite `delay()` calls to use a custom timer instead of using the `millis()` function."

A block of code is set as follows:

```
if temperature < 5°C {
   send me a reminder to use beanie and gloves
}
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
if (inputValue == 1) {
    lastTimeOpenend = millis();
    isOpen = true;
    myservo.write(90);
 }
```

Any command-line input or output is written as follows:

```
mosquitto_pub
```

Bold: Indicates a new term, an important word, or words that you see onscreen. For example, words in menus or dialog boxes appear in the text like this. Here is an example: "Click on **Tools** | **Serial Monitor** to open the serial monitor."

Warnings or important notes appear like this.

Tips and tricks appear like this.

Get in touch

Feedback from our readers is always welcome.

General feedback: If you have questions about any aspect of this book, mention the book title in the subject of your message and email us at `customercare@packtpub.com`.

Errata: Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book, we would be grateful if you would report this to us. Please visit `www.packt.com/submit-errata`, selecting your book, clicking on the Errata Submission Form link, and entering the details.

Piracy: If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at `copyright@packt.com` with a link to the material.

If you are interested in becoming an author: If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, please visit `authors.packtpub.com`.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions, we at Packt can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about Packt, please visit `packt.com`.

1
Section 1: An Introduction to IoT and MQTT

This section offers a quick introduction to what the **Internet of Things** (**IoT**) is, why readers should care about it, what MQTT is, and how MQTT can help readers to build their own prototypes.

The following chapters will be covered in this section:

1
The Internet of Things in a Nutshell

Kids who grew up in the 1990s, like myself, will have noticed a lot of changes to our everyday lives over the last 30 years because of the internet and its omnipresence. Having a wireless connection wherever we go, checking emails and Facebook on our smartphones: all of this didn't exist in the 1990s. Smartphones? We had mobile phones, yes, but with a monochrome display and an antenna, and they didn't have an internet connection. The multimedia highlight was playing the game Snake, a simple 2D game with a game logic that can be programmed in a day. There was no mobile internet and no wireless LAN. When we were visiting another city and got lost, we had to go inside a store and ask for help—no Google Maps.

Nowadays, the internet is ubiquitous and people are dependent on it for their day-to-day life. Everyone has a phone in their pocket, which they can use to find any information they want in seconds. According to a study by *Business Insider*, on average, there will be four internet-connected devices per human by 2020.

Smart devices come in various forms and shapes and are causing disruptions in every industry, just like the internet did in the early 2000s. Companies such as Google and Samsung have been developing wearable devices, such as smartwatches and health trackers, automated robots for industrial applications, self-driving cars, smart buildings, and smart home devices, as well as early-warning systems for tsunamis or earthquakes.

In this chapter, we will explore some of the devices and technologies that are driving the internet-connected device revolution. We will have a closer look at the following topics:

- Exploring smart homes
- Exploring smart cars
- Exploring industry 4.0 / the Industrial Internet of Things
- Prototyping for the greater good
- What is a prototype?
- Voice control
- Why should you invest in the IoT?

Exploring smart homes

The most prominent category of **Internet of Things** (**IoT**) devices for end consumers is **smart home**—connected electronic devices that are used in your flat or house. They often replace traditional electronic devices by enhancing them with an internet connection and a way to control them digitally. In most cases, they offer a smartphone app or integration with a smart home app, such as Google Home, which makes it possible, for example, to turn the device on and off from your sofa, change its settings, and check its status.

According to a study by *Zion Market Research* (`https://www.zionmarketresearch.com/news/smart-home-market`), the global smart home market is about to reach around USD 53.45 billion by 2022, with big tech companies such as Samsung and Google being at the forefront. Let's look at some examples of these smart home devices:

- Smart fridges
- Smart door locks
- Smart thermostats
- Smart scales
- Smart lights
- Smart pet food dispensers

These are only a few of the smart home devices available today. The potential of such smart devices to influence our lives is huge.

Let's look at an example of a smart device—a smart fridge—and see what benefits it brings over a non-smart device.

The main problems people tend to have with a regular fridge are the following:

- Food goes to waste because you have no clear overview of expiration dates for each item.
- You run out of a certain food because you did not keep track of your supply.

Smart fridges keep track of your groceries and their expiration dates and warn you when you might run out of your favorite food. When opening the fridge, it could also present you with recommendations for what to cook with the available ingredients.

To minimize food waste, it might detect that you have, for example, tomatoes, curry sauce, and onions that will go bad soon. It might then automatically run an online search and present you with recipes that include these ingredients for you to prepare.

If it is connected to third-party shopping providers, it might also take care of ordering food automatically for you—for example, when your milk is running low, it might order a new bottle for you, which will then either be delivered to your doorstep or be ready for pickup at your local grocery store, together with other items that you are running low on.

While you cannot expect that level of comfort from a smart device just yet, more and more devices in our households will have similar features that make our lives easier.

How smart devices connect to the internet

In most cases, smart home devices need internet access to function correctly, which means that you must tell your device how it can connect to your local network via a network name and password.

When you first set up a new smart device in your household, the process often looks like this:

1. You connect the smart device to a power source.
2. The smart device goes into setup mode.
3. The smart device opens an ad hoc network (see `https://en.wikipedia.org/wiki/Wireless_ad_hoc_network`).
4. You connect to the network provided by the smart device via your smartphone.
5. On your smartphone, you visit a special website that is served by the smart device, and which is only accessible when you are connected to its network.
6. You specify the network username and password on the website.

7. The information is transmitted to the smart device.
8. The smart device will close the temporary ad hoc network and connect to your normal network via the username and password you provided.
9. The smart device can go online via your regular wireless network.

The very same technique can be used for creating IoT prototypes with microcontrollers, such as Arduino, Raspberry Pi, or Particle Photon, to create a convenient setup routine. We will not cover this technique in this book, but this is something to keep in mind if you're thinking about giving a prototype to friends and family or with batch production.

For example, if you were to bring your prototype to your friend's house, it would not be able to connect to the internet there, because it does not know the network name and password to connect with.

Without a setup routine like the one described previously, the source code of the device has to be modified each time the device is placed into a different environment to successfully connect to the other network.

Useful and unnecessary use cases

If you have ever lived in a single apartment without too much knowledge about how to separate your clothes when doing your laundry, you might have ended up with a bunch of pink shirts because a red sock found its way into your collection of white shirts that were to be washed. If you don't know what I am talking about, it is one of the mistakes a lot of single people make when using a washing machine. Washing a red sock with your white shirts is not a good idea, and might result in your shirts taking on the color of the sock, thereby becoming pink.

Other things can go wrong. You might do something wrong when setting the temperature, accidentally washing your favorite Norwegian Christmas sweater at 90 degrees, causing it to shrink. Now you might wear it while exposing your belly on cold summer days.

Chances are high that you don't want these things to happen.

Imagine that your shirts had a voice and that, when you asked them about their color and washing preferences, they answered, "*Hi, my color is light-blue and I enjoy being washed at 30 °C. I will get sick when you tumble dry me, so please don't!*". Now your washing machine could ask each and every textile lying in the washing machine drum about their washing preferences and set the program accordingly, or warn you that the red sock might better be off with the other red socks.

This can be made possible by using smart tags, such as **RFID** (short for **Radio-Frequency Identification**).

Many such use cases are about to bring value to our lives and make our lives easier. However, many companies forget about their product's original use case. One example is a smart light bulb that cannot be used as a light source when it is having a firmware update (`https://twitter.com/BalrogGameRoom/status/1036644958973960192`), which lasts for up to one hour. A firmware update probably does not need to be done very often, but if the light bulb is going through a firmware update in the evening, when it is dark, you will question their update policy and want to go back to using a normal light bulb.

I think updating the firmware on devices is important to keep the device secure and safe from hackers, but, in this case, the company producing these smart light bulbs could have spent more time developing a better update routine to improve the user experience.

Often, smart devices only work under perfect conditions, in this case, with updated firmware. Compared to non-smart devices (in this case, a normal light bulb), this is a huge step backward. A normal light bulb has one purpose: provide light when it has power. Either it works or it does not. Adding internet capabilities to a device might not always be the best decision, as it produces problems such as the one mentioned previously. The device should be created in a way that means that using its core feature is not disturbed if there's no internet connection or outdated firmware.

Many companies that develop smart devices overemphasize the smartness of their products and forget about their original use case, reducing the usability of the device, which was its main purpose. Not all things need to be smart, and if they are made smart, then their main feature should work even without internet access.

In another example, two security researchers discovered a vulnerability in a smart vacuum cleaner model—a robot equipped with a camera cleaning your flat autonomously. According to their study, it is not too hard to get access to the vacuum cleaner with admin privileges, allowing an attacker to misuse the vacuum cleaner as a 360º spying device. Definitely not what you signed up for when buying a smart device to save yourself some time. In this case, the manufacturer should have spent more time securing the device properly to make sure it is safe from hacker attacks.

There definitely is a need for autonomous vacuum-cleaning robots, smart light bulbs, and probably smart coffee machines, but when building new smart devices, you should always critically decide whether adding internet access to the device really adds something to its utility. If it does, then you should bear in mind the following questions when thinking about these aforementioned edge cases:

- What happens when the device is not able to reach the internet? Is it still usable?
- Can the device be updated? If so, when will it be updated? Does an update block its main functionality?
- Is the device secure enough? Or will it be easy for hackers to access the device and use it for their purposes?

If you want a few hours of entertainment (and education), you should check out the Twitter account `https://twitter.com/internetofshit`, which collects IoT fails and available smart devices of little use. It will not only bring a smile to your face but also make you realize that things went too far for some companies when dealing with smart home devices. The first question to ask yourself should always be: *Is the thing I am about to build useful?* Just making something smart certainly does not make it more useful, but instead might make it vulnerable to hacker attacks or completely useless when the internet connection does not work.

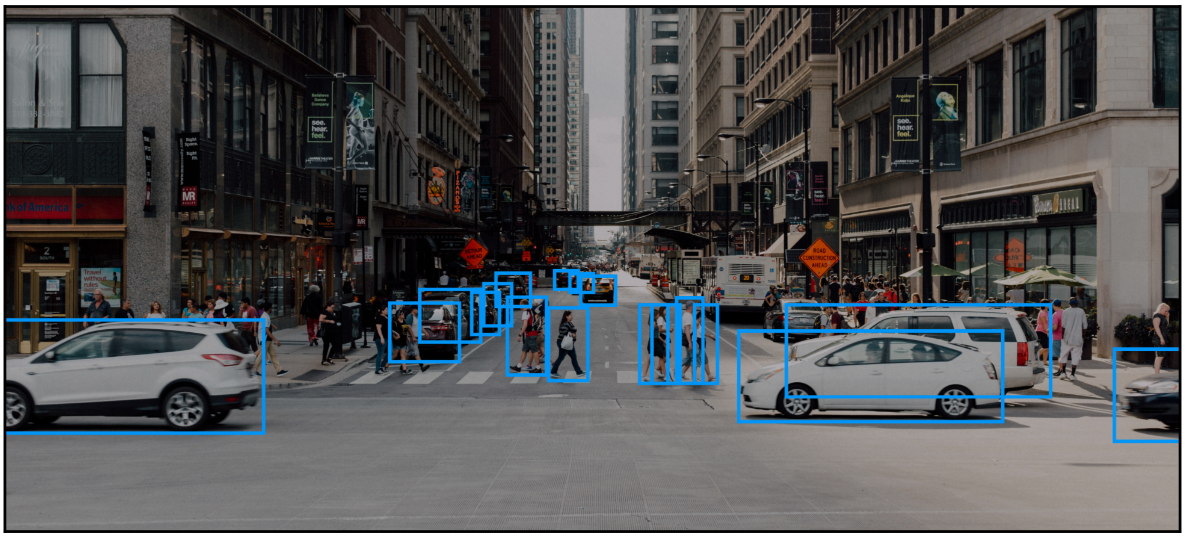
Exploring smart cars

Smart cars, another emerging field associated with IoT, is gaining momentum. Its progress is closely connected to the advancements made in machine learning in the last decade. If you have never heard about machine learning, you should put this book aside for a second and watch the TED talk *The Rise of Artificial Intelligence through Deep Learning* by Yoshua Bengio (`https://www.youtube.com/watch?v=uawLjkSI7Mo`). In a nutshell, machine learning makes it possible for computers to learn just like our brain does. It is another future technology that will be paired with IoT more and more to create smart, self-learning devices.

Machine learning is used in smart cars to develop many of its essential features:

- Detect the street, other cars, and people
- Understand signs and speed limits
- Identify dangerous situations and know how to solve them (for example, by applying the brakes)

The following screenshot shows a simplified view of object detection in a smart car:



A simplified view of object detection in a smart car (image based on photo by Josh Sorenson)
Source : (https://www.pexels.com/photo/architectural-design-asphalt-buildings-city-139303/)

If you compare the visual interpretation of a human's view of a street and the digital representation of the same theme, the two differ enormously—a computer seeing through a camera just sees raw data, the amount of red, green, and blue per pixel, and nothing more. Machine learning makes this data more usable by training the computer based on input footage—for example, by supplying a large number of images depicting street views.

After many learning iterations, the computer might be able to identify a street using fresh footage. Machine learning and IoT will be good friends in the future as internet-enabled microcontrollers become smaller and more powerful.

Currently, complex machine learning models require an expensive state-of-the-art computer, but there are already experiments using the Raspberry Pi, a tiny computer that runs Linux, for simple machine learning tasks. Google and NVIDIA introduced two new development boards (so-called edge devices) in 2019, which have a similar form factor to the Raspberry Pi and are intended for machine learning prototyping: Google Coral and NVIDIA Jetson TX2.

But so-called on-device training is not the only way hardware devices can use the power of machine learning. The most common way they use machine learning today is by sending the device's data to a cloud server where the heavy analysis is done. One example of this is Google Photos. It allows you to upload your photos, in most cases taken with a smartphone, to the Google servers. The servers will analyze each and every one of them using various machine learning models.

You can already use the results in two ways, as:

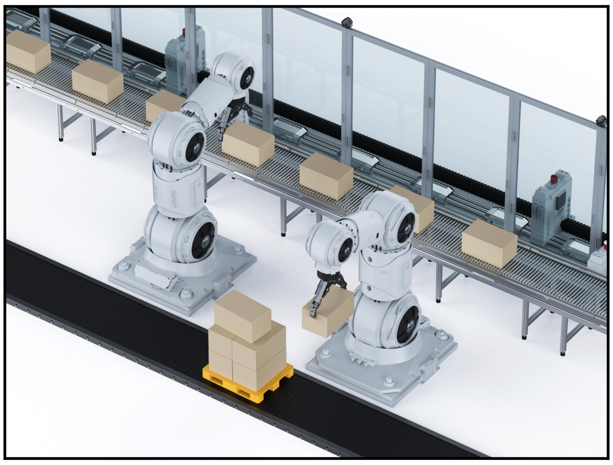
- The machine learning model detects all faces in a photo and groups them together into categories. This way you can easily find all photos that contain your face or any of your friends' faces.
- The machine learning model detects objects in photos. You can then, for example, filter all of your images that contain a red car.

Apart from machine learning, to understand their surroundings, smart cars can communicate with each other. Every now and then there are reports about a mass crash on the highway. Smart cars will be able to warn each other about dangerous situations: "*Attention cars behind me. There is an obstacle lying on the street. Better slow down!*".

When it comes to situations like this, the amount of time it takes for another car to receive this information can make the difference between life and death—one or even multiple seconds response time is just not good enough here, the response time needs to be in the milliseconds. If the cars were using the internet for communication, it might take too long. When sending data to a satellite and spreading it from there to all of the nearby cars, there are too many things that can go wrong and prevent the warning from being delivered in time. Fortunately, there is a solution for this: using a technology called **Vehicle to Vehicle** (**V2V**). With this technology, cars can talk directly to each other by opening a network themselves (like a router). Using this, they create a mesh of connected cars without needing internet access.

Exploring industry 4.0 / the Industrial Internet of Things

Industry 4.0 refers to the fourth industrial revolution. One of its driving technologies is IoT, connecting physical machines digitally with each other and the cloud.



Industrial robots (source: Depositphotos)

According to *IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things*, by *David Hanes* (*et al.*), (2017, Cisco Press), p 932-933, 1208-1209:

> *"There are estimates that there are 60 million machines in factories; the vast majority of them are more than 15 years old and 90% are not connected to the internet."*

In the airplane industry, preflight security checks involved a lot of manual work—each and every piece of essential equipment had to be checked off a list for the flight to be considered safe. On average, this took 6.5 hours per plane.

By integrating RFID tags into essential safety equipment, for example, the airplane industry made it possible for security staff to use an RFID scanner instead of a paper checklist to make sure that no important parts are missing.

And adding these chips was worth it. The 6.5 hours it took to manually check for the presence of each part could be reduced to 20 minutes this way.

There are many more industries less modern than the airplane industry that can profit from IoT as well.

By 2016, approximately 20 million smart meters had been installed globally, which can send their data automatically to the cloud. One of the areas where this is being used is power consumption in apartments.

Using smart meters and a web interface, tenants can check their monthly energy consumption. This way, it is easier to identify electronics or usage patterns that require a lot of energy, just by comparing the monthly costs.

Without smart meters, it is hard to tell whether the new electric grill you just bought is actually a power hog. You can imagine how much easier it is for the power companies as well: no need to send a technician from apartment to apartment, writing down numbers on paper that then have to be typed into a computer later.

Smart meters make everything much easier.

One of the areas where a lot of man-hours can be saved is semiautomatic maintenance of machines. Every machine part has a certain life expectancy: some fragile parts may last a few weeks, some several years. But sooner or later, physical parts need to be replaced. Most factories rely on their machines running in parallel—once one of them stops, the flow cannot go on.

Detecting machine parts that need to be replaced before they actually break can save the company a lot of analysis work and money. With self-monitoring machines, this is about to become more efficient and involve less human maintenance work. By equipping machines with various sensors to run self-tests and verify that all of their parts operate as planned, malfunctions or old parts can be identified early on by the machine. It can then call for a technician to replace part *x* or manually check part *y*. By pinpointing possible problems this way, machine downtime can be minimized.

The same self-analysis functions have found their way into the consumer market more and more as well. Commercial coffee machines have an internal counter that is incremented every time a coffee is made. After *x* coffees have been made, the machine might blink an LED to prompt you to run a manual maintenance program, for example, to get rid of unwanted deposits. While this isn't too smart, modern consumer 3D printers actually are. Being equipped with a multitude of sensors, they can detect malfunctions, identify broken parts, and fine-tune their own settings while printing.

Prototyping for the greater good

Building IoT devices can also contribute to the greater good in a non-commercial setting. In 2018, a non-governmental organization consisting of engineers and developers called **Rainforest Connection** (`https://rfcx.org/`) developed an IoT device to help to protect the Amazon rainforest from illegal deforestation. As it reduces the levels of greenhouse gases in the atmosphere, the rainforest plays an important role in our climate, and, according to Rainforest Connection's CEO Topher White, illegal deforestation accounts for nearly one-fifth of all the greenhouse gas emissions every year. Up to 90% of the deforestation of the Amazon rainforest is done illegally. Saving the rainforest could be the cheapest and fastest way to slow down climate change:

African jungle (source: Depositphotos)

In collaboration with the indigenous Tembé tribe, Rainforest Connection developed a device called **Guardian** to detect illegal deforestation and prevent it from happening. The devices are based on recycled smartphones, which have many of the ingredients of an IoT device on-board—a microprocessor, sensors, and a way to communicate via the cell phone's network over the internet.

The Guardians are hung inside the trees, their microphone transmitting the ambient sound via the cell phone network 24/7, forming a huge grid of microphones. In the cloud, where all of this data is assembled, a machine learning model based on Google's open source TensorFlow library comes into play. It was trained to detect the sound of chainsaws and trucks used in deforestation operations. Whenever one of the microphones detects a sound like this, disturbing the natural sound collage, the machine learning algorithms can identify it and the relevant information, such as GPS data, can be forwarded to the authorities.

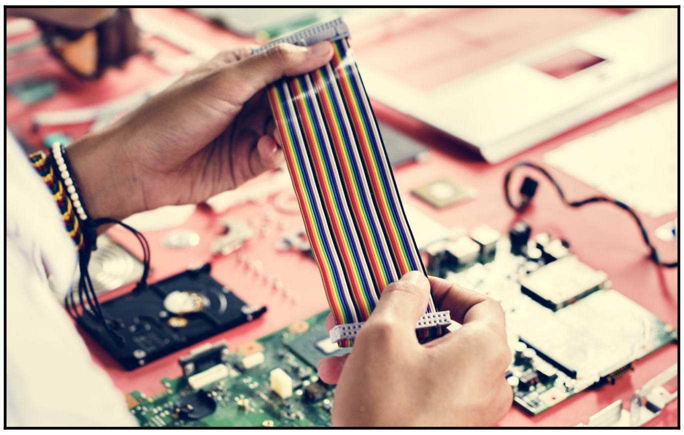
In March 2018, the Planet Guardians program was launched by Rainforest Connection. Students from Los Angeles helped to build new Guardian devices to be added to the grid, and it is expected that these will help to protect 100,000 acres of rainforest throughout the year 2020 (you can go to `https://www.blog.google/technology/ai/fight-against-illegal-deforestation-tensorflow/` and `https://www.prnewswire.com/news-releases/rainforest-connection-introduces-one-of-the-largest-programs-ever-launched-by-students-to-protect-the-worlds-rainforests-300617270.html` for more information).

Similarly, devices equipped with GPS and a sensor are used to detect earthquakes—once an earthquake is detected by the sensors, people can be warned accordingly.

The same principle can be used for other areas as well—as an early warning system for tsunamis or avalanches, for example.

What is a prototype?

Before starting to work on your prototype, it is important to understand what a prototype is and what it is not. From a product perspective, various things need to be evaluated before a device can be produced in batches, starting with the functionality. Is the device useful? Does it serve a purpose? You also might want to find out how it feels. What material is it made of? Does it feel good when you hold it in your hands? How does it look? Do the buttons have a nice degree of resistance to pressure? Is it fitting in to its environment? Is it easy to use? Does it need a manual? Where can you get all the materials for batch-production?


Hardware prototyping (source: Envato Elements)

If your device is supposed to be outside, you also need to find out which materials can be used for it to survive the wind and rain. For all of these questions, you might want to produce one or many prototypes to get closer to a possible product, step by step.

A lot of the areas mentioned previously can be tackled individually—to find a nice form factor, you might design various models in a 3D application, print them out, and iterate upon them. The material used can also have a huge impact. By using online services such as Shapeways (`https://www.shapeways.com`), you can have a 3D printing service at hand that can print in various materials, such as plastic, aluminum, and gold. That last one might be a bit expensive, but at least you have the option. Combined with other rapid prototyping techniques, such as CNC milling or laser cutting, the possibilities are endless.

While building the prototype, it is essential to find out quickly whether your idea works, both from a usability perspective and from a technical perspective.