Get More Productivity with Cloud Services

How to improve the development flow of embedded IoT and ML systems

By Reinhard Keil, Arm

May, 2021

White Paper

Today most embedded applications are still created on desktop computers. This has many reasons but is primarily caused by the validation process that relies heavily on target hardware. For other applications cloud computing is well established. This paper explores how cloud-based tools can help to improve the development flow for embedded projects. It starts during product evaluation, includes model- or simulation-based validation with continuous integration flows, model optimization for machine learning, up to device provisioning for deployment.

Introduction

Cloud computing is the fastest growing form of computing and has already changed the way we work. Just think of office applications that make documents accessible on various devices, customer relations management systems in sales and support divisions, or the many video conferences during the pandemic. Due to COVID-19, the trend towards home office has accelerated and we assume that even after the pandemic our everyday workplace will change permanently.

In software development, version management and the associated management tools such as issue tracking are increasingly hosted in the cloud. Today it is more economical for multinational corporations to use services like GitHub instead of own IT and server infrastructure. Such systems also enable decentralized development teams or collaboration with other companies that should not have direct access to a company proprietary IT infrastructure.

Overview of Technology and Audience

A. Cloud Computing – Advantages and Disadvantages

Cloud computing is well established and the fastest growing computing market as it offers great advantages to customers of all sizes: single users, small development teams, and enterprises. These advantages in a nutshell are:

- Cost savings as there is no need for physical hardware investments. For software licenses, cloud vendors offer frequently a pay-per-use business model.
- Up-to-date access to latest tools and software without spending time and money on installations.
- Resilience as big cloud providers take care of redundant setups and data recovery in case of emergency scenarios that can vary from natural disasters to power outages.
- High speed as it is easy to scale the compute resources that are required for getting a job done faster.
- Virtually unlimited storage as it is possible to quickly expand storage capacity with very nominal monthly fees.
- Mobility as working on premises or at remote locations is easy and effectively the same.
- Collaboration that helps employees who are in different geographies to work together.
- Advanced online security as cloud providers carefully monitor security, which is significantly better than most in-house systems.

Of course, there are also some drawbacks such as:

- Good, reliable internet connectivity is a must have.
- Potential technical issues of the internet or cloud infrastructure that may result in an outage.
- Potential security threats as sensitive information is shared with a third-party.
- Performance that may vary as underlying systems simultaneously provide resources to other businesses.

B. Arm technology for the whole range of computing

The Arm Neoverse Series Processors [1] are optimized for handling a wide range of cloud-native workloads at world-class levels of performance, efficiency, and compute density. These processors are designed for massive compute facilities that also care about redundancy. Cloud computers typically serve the needs of multiple customers simultaneously and provide enough power for compute intensive applications.

IoT gateway and 5G telecom connectivity is typically implemented with the Arm Cortex-A or Cortex-R processor series. Such gateways facilitate the network and telecom infrastructure but may also aggregate IoT downstream data. On internet gateways it is also possible to perform some latency sensitive compute that requires real-time response times which cannot be guaranteed on public internet structure.

IoT endpoint devices implement the interface to the physical world with I/O peripherals such as sensors and actuators. Many devices service only a single, real-time purpose. The software on such devices should be time deterministic and frequently needs to operate on very low power budgets. Endpoints connect to IoT gateways via various transport protocols such as WiFi, LoRa, Bluetooth or in case of wired interfaces Ethernet, CAN, or USB. Due to all these requirements, a typical IoT endpoint device uses a microcontroller that integrates a Cortex-M processor [2]. To support future application requirements the Cortex-M processor may be combined with an Ethos-U NPU processor [3] that is designed to accelerate machine learning workloads.

C. Developer Audience and Focus to IoT endpoint

This paper focuses primarily on the needs of IoT endpoint developers that today have a diverse range of tasks:

- Classic firmware and embedded development that focuses on low-level drivers, bare metal code, and middleware development.
- IoT and embedded application development where an RTOS and existing software stacks are integrated into the system.
- Machine learning (ML) and artificial intelligence (AI) development that focuses on data analysis, model optimization and deployment.

Usage of Cloud Computing for IoT Endpoint Application Development

Embedded software development traditionally uses local IT infrastructure and desktop computers that connect to test hardware. Cloud technologies have the potential to improve the development flows with the following technologies:

- Version control that is typically provided via a Git repository hosting service.
- Software development that could be based on a Software as a Service (SaaS) model where a ready-to-use complete tool environment is provided.
- Continuous testing where a cloud server provides a virtual machine or a container that includes tool environment with simulation models.

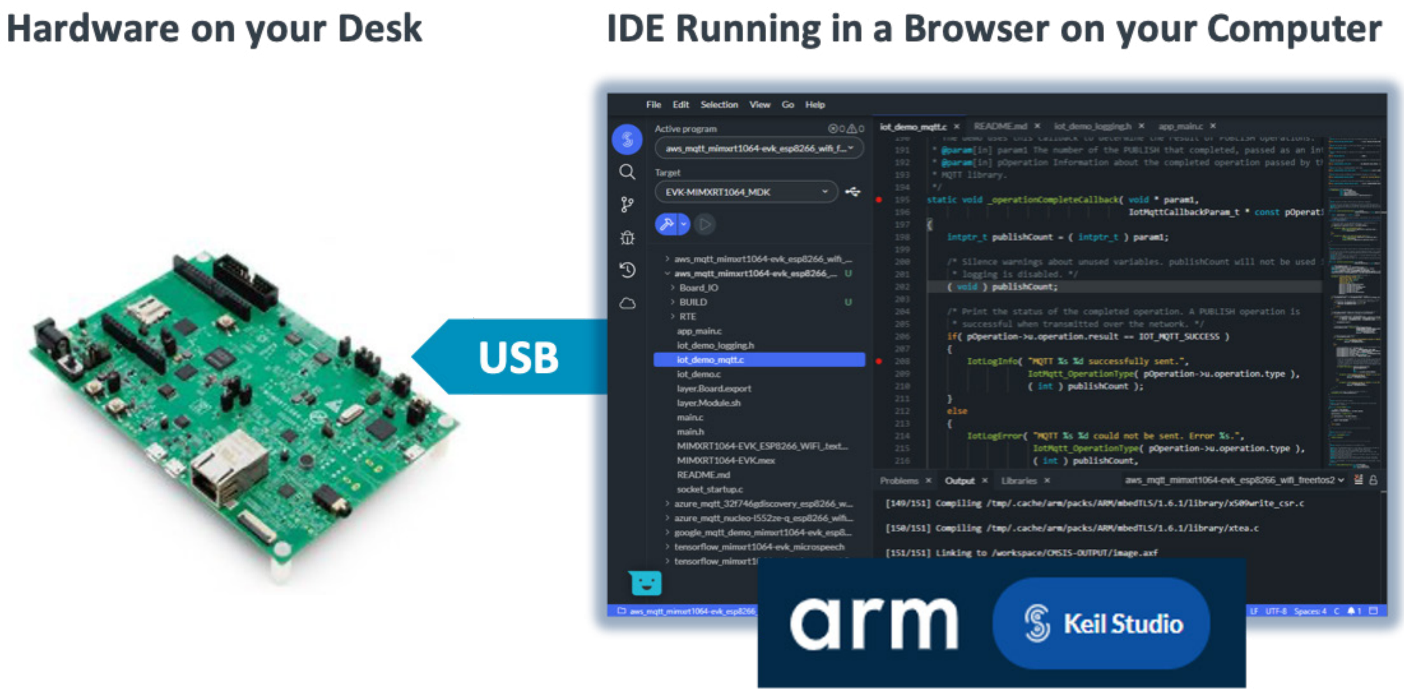
- Software deployment that supports a geographically distributed fleet of devices with Over-The-Air (OTA) programming.
- Machine learning that requires data analytics which could be also used to monitor devices for anomalies.

Version control services such as GitHub are already well established in the industry and typically include access control and collaboration features that also support a distributed development team. This paper does therefore not explore version control services but focuses on the remaining four topics.

A. Cloud-based Software Development

Server-based software development on a command-line interface is well understood and established. In such cases a complete toolchain that consists of compiler, linker, and related tools for project generation is available on a server. Combined with the CMSIS software pack [5] infrastructure this setup provides device support and essential software building blocks that supports a well-established software development workflow for embedded applications.

Embedded engineers also expect an IDE workflow and connectivity to a hardware target system. Using modern browser technology allows to create an IDE that integrates a debugger which can be connected via USB directly to target hardware as shown in Fig. 1. The browser connects this IDE to the previously described server-based software development infrastructure that is hosted in the cloud.


Hardware on your Desk
IDE Running in a Browser on your Computer
USB
File Edit Selection View Go Help
Active program
aws_mqtt_mimxrt1064-evk_esp8266_wifi_f...
Target
EVK-MIMXRT1064_MDK
iot_demo_mqtt.c
README.md
iot_demo_logging.h
app_main.c
Problems
Output
Libraries
aws_mqtt_mimxrt1064-evk_esp8266_wifi_freertos2
Keil Studio

```
 * @param[in] param1 The number of the PUBLISH that completed, passed as an in
 * @param[in] pOperation Information about the completed operation passed by th
 * MQTT library.
 */
static void _operationCompleteCallback( void * param1,
                                        IotMqttCallbackParam_t * const pOperati
{
    intptr_t publishCount = ( intptr_t ) param1;

    /* Silence warnings about unused variables. publishCount will not be used i
     * logging is disabled. */
    ( void ) publishCount;

    /* Print the status of the completed operation. A PUBLISH operation is
     * successful when transmitted over the network. */
    if( pOperation->u.operation.result == IOT_MQTT_SUCCESS )
    {
        IotLogInfo( "MQTT %s %d successfully sent.",
                    IotMqtt_OperationType( pOperation->u.operation.type ),
                    ( int ) publishCount );
    }
    else
    {
        IotLogError( "MQTT %s %d could not be sent. Error %s.",
                     IotMqtt_OperationType( pOperation->u.operation.type ),
                     ( int ) publishCount,
```

```
[149/151] Compiling /tmp/.cache/arm/packs/ARM/mbedTLS/1.6.1/library/x509write_csr.c

[150/151] Compiling /tmp/.cache/arm/packs/ARM/mbedTLS/1.6.1/library/xtea.c

[151/151] Linking to /workspace/CMSIS-OUTPUT/image.axf
```

Fig. 1: Arm Keil Studio IDE in Browser connects via WebUSB technology to hardware

Arm is working on a new generation of software development tools that will result in Keil Studio for microcontrollers. Keil Studio is based on the Eclipse Theia IDE platform that embraces the design concept of VS Code and will be first available as cloud version [4]. The new tooling will also support both command line and IDE workflows.

Instead of installing a software development environment locally on a computer, a complete, ready-to-use setup in the cloud is used. The IDE runs in the browser and the actual compilation takes place on a cloud server which is usually even faster. There is no complicated tool installation, and the example projects along with the related software packs are always up to date. This is the ideal setup to explore reference example code and compare microcontrollers from different manufacturers, as is saves the installation process of various software environments.

Initially we will therefore focus on evaluation workflows where a set of examples can be explored. It will be possible to continue software development in the cloud, but also export these reference example projects to classic desktop tools such as Keil MDK [6]. The platform is ideal for project exploration, but also for learning and training as it integrates an interface to Git-based version control services. In future we will open this platform also for professional software development where features for a distributed team or collaboration will provide significant benefits.

B. Cloud-based Continuous Testing

Safety standards [7] mandate for a staged test process that is composed of:

- Unit testing where code is tested primarily at function level using a test framework.
- Integration testing where multiple components are combined and interfaces between these components are verified.
- System testing where the complete system is tested against the requirements.

For embedded applications, these types of testing are frequently ignored even though the benefits (that apply also for none safety-related applications) are well understood [8]. There are some fundamental reasons as system testing of an embedded application typically requires a target hardware.

However, Arm Fixed Virtual Platforms (FVP) [9] offer a complete simulation of an Arm system, including processor, memory, and peripherals. These FVPs are configurable and provide a “programmer’s view”, which gives you a comprehensive model on which to debug and test your software. Running at speeds comparable to real hardware these FVPs are ideal to execute unit testing and when using test data interfaces also for integration testing. The Arm simulation models including FVPs are verified with the same process as the actual processor implementation and therefore match the logical behaviour of real devices.

However, there is reluctance by development teams to spend initial effort setting up continuous integration (CI) workflow even though the long-term benefits are undisputed as shown in Fig. 2.

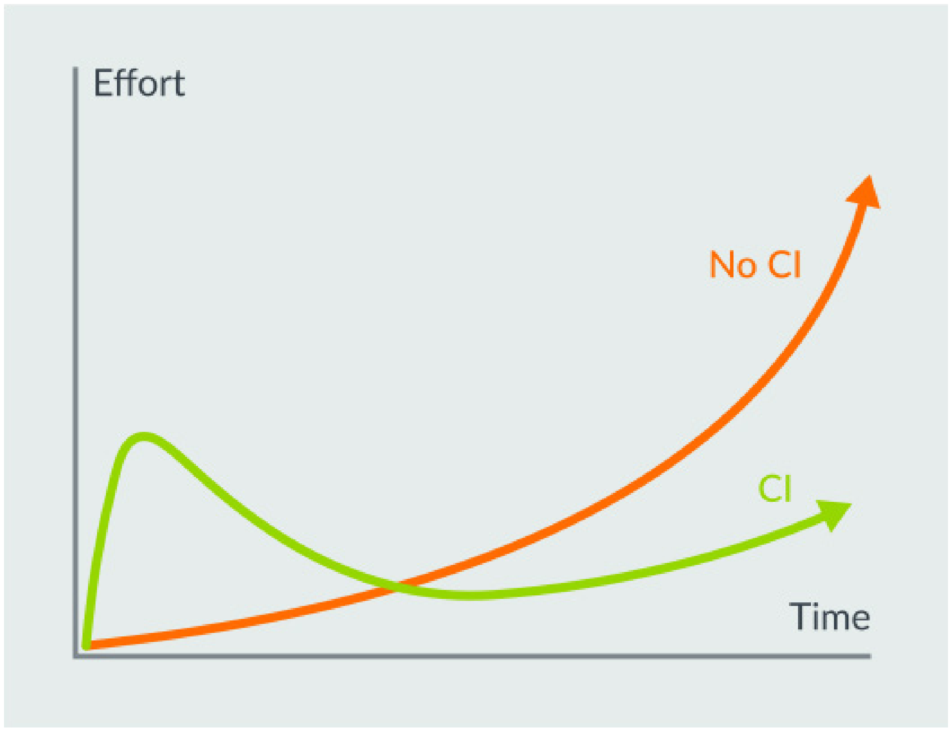


Fig. 2: Comparison of test efforts in CI and no CI workflows

Arm therefore released a ready to use platform [10] that supports Cortex-M55 software development and validation along with information on how to integrate it into the development process. The blog "Infrastructure for Continuous Integration Tests" explains how to run tests locally on a development computer, but also how to integrate it as CI flow into cloud-based version control systems such as GitHub.

Cloud-based validation is already popular in the automotive industry where safety development processes are established. The cloud server approach also allows to scale testing with multiple instances which gives a significant speed uplift during regression testing.

For unit tests or integration tests, that need to run quickly on small sets of code, virtual platforms offer advantages [11] over hardware, including:

- Speed - Virtual platforms have no overhead for flashing the application on physical hardware. This saves time on small and fast unit tests.
- Scale - Virtual platforms can scale to run many tests in parallel. This makes virtual platforms more cost-effective than a farm of physical hardware.

- Maintenance – Unlike physical hardware, virtual platforms do not overheat, wear out from overuse, break from misuse, or use physical space and resources.
- Upgrades – Virtual platforms can be adapted and re-configured to match corresponding changes to the underlying hardware platform that is under development. These types of changes can be costly or impossible with physical hardware.

The speed increase when using virtual platforms compared to hardware can be attributed to several factors, that include program download time and execution speed. Another substantial benefit of using virtual platforms for running tests is the scaling with more simulations in parallel to speed up test. Running tests in parallel is easier in a simulation-based cloud farm as compute power is charged in a pay-per-use model. It also avoids buying and managing more hardware boards. (TABLE I) compares the execution time on various platforms and lists the speed increase of simulation compared to target hardware.

Table 1: Test time on various platforms

Test Platform	Time (sec) *	% Speed increase
MPS2 Development Board (Target Hardware)	488	n/a
FVP simulation (on single instance)	259	88 %
FVP simulation (two in parallel)	160	205%

* Execution time measured with 124 test cases

C. Over-the-Air Programming

Over-the-Air (OTA) update is already available on many devices, but often requires user interaction. OTA is becoming increasingly automated and will be available on small microcontroller systems. This allows software development to be staged where initially a system with minimal functions is deployed. In later steps the device functionality can be extended via OTA and a deployed hardware product generation can be adapted to new market requirements to ensure that the product remains competitive. Such a staged software development approach reduces the initial time-to-market, avoids costly manual maintenance, but can also lead to new service business models.

Fig. 3 is a simplified view to the software components of an IoT endpoint system that incorporates the Trusted Firmware M (TF-M) [12] and a neuronal network with ML model data. TF-M implements a secure processing environment that utilizes TrustZone [13] on Armv8-M, Armv8.1-M architectures (that is implemented on the Cortex-M23, Cortex-M33, or Cortex-M55 processors). It is the reference implementation of the Platform Security Architecture (PSA) [14] and executes in the secure area of the processor. TF-M provides secure services to the non-secure area where the processor executes the actual user application along with internet and cloud connectivity.

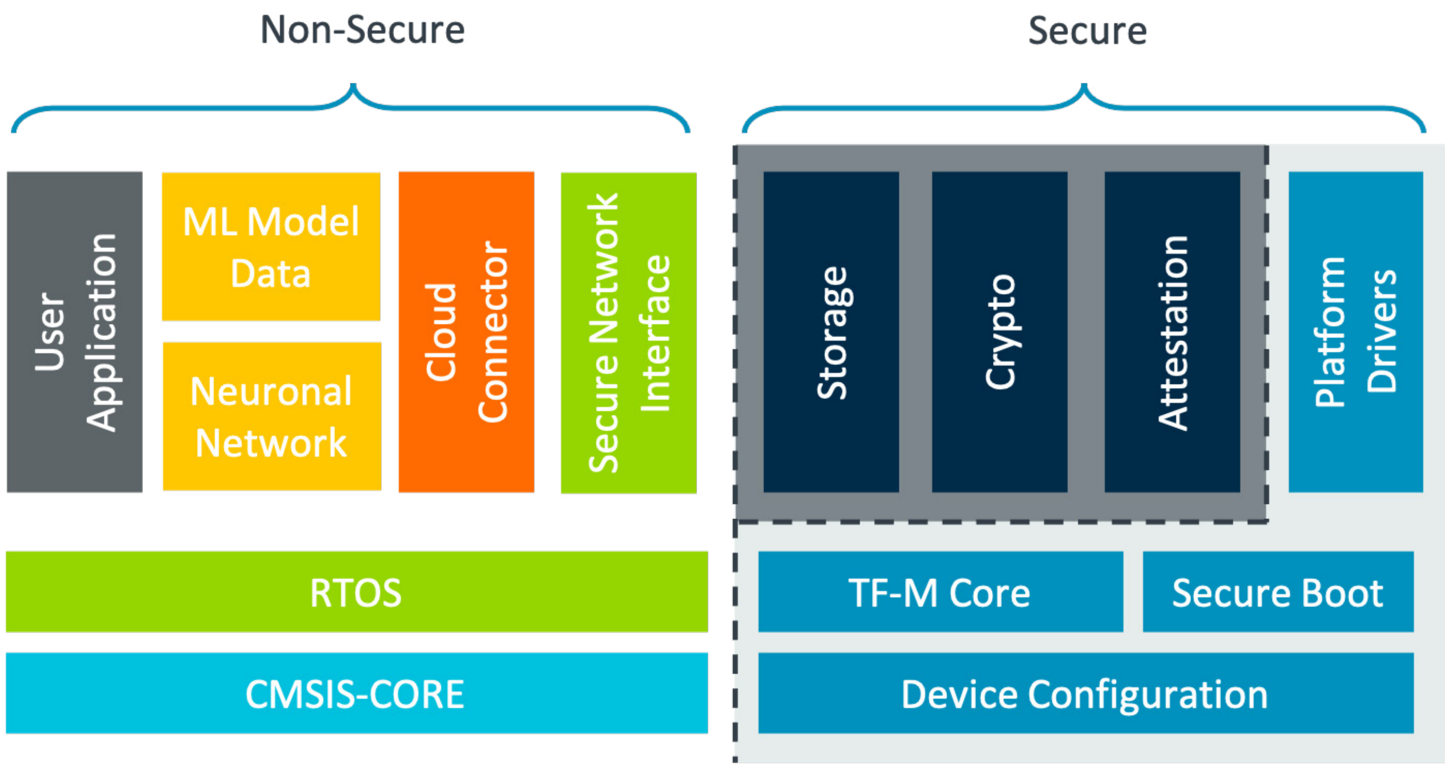


Fig. 3: Simplified view to the software components of a IoT endpoint system

To update the firmware, the new images are downloaded by the Cloud Connector and temporarily stored, for example using the TF-M Storage service. On the next system restart the Secure Boot verifies the digital signature of the image and updates the Flash memory of the device. A software stack as shown in Fig. 3 could have multiple images, for example:

- Secure firmware for the TF-M part.
- Non-secure firmware with the user application.
- ML model data for deployment of a new algorithm.

The firmware update process on such devices may be executed in the following steps:

1. The verified firmware has passed all tests as explain in the previous section and gets a digital signature.
2. The firmware image is stored on cloud infrastructure.
3. A cloud update service provides images to IoT devices that are deploy in the field.
4. The cloud update service protocols the IoT update process.
5. After validation, the IoT device installs the new firmware image in the IoT endpoint system.

OTA programming is an approach that requires a cloud service. Note that steps 2. – 4. are enabled and supported by cloud infrastructure.

D. Data Analytics and Artificial Intelligence

The IoT endpoint system can deliver valuable data to the manufacturer via the cloud. For example, if the device reports important parameters about the system status, potential errors can be identified at an early stage that could otherwise potentially lead to a system failure. This technology is called predictive maintenance and can reduce the overall service effort for the installed device base.

Today, AI and ML algorithms that operate on sensor data from IoT endpoint devices frequently execute on cloud servers. But to meet real-time requirements with AI the execution of the actual AI algorithm should be transferred to the edge device. This local edge approach also reduces the internet traffic that would otherwise drastically increase when billions of IoT endpoint systems depend on the cloud for AI algorithm execution.

AI and ML in low-cost IoT endpoint devices is today already possible and enables new innovations in future devices. For more demanding algorithms next generation devices will integrate the new Cortex-M55 [2] and Ethos-U55/65 [3] processors that offer a significant performance uplift on ML workloads.

For effective ML algorithm selection, training, and validation a large set of representative and qualified data is a pre-requisite. Having a deployed fleet of IoT endpoint devices that captures such data is therefore invaluable. The data could be grouped into various categories using data analytics that executes on cloud servers. For example:

- Test data that help to identify suitable algorithms for the neuronal network.
- Validation data that are used to verify a system.
- Training data that are required to generate the ML model data.

The training of the ML model will mostly take place in the cloud, because both an extensive data set, and high compute power is a prerequisite. The algorithm execution based on the ML model can then take place directly on the IoT endpoint device. Arm is working with partners [15] on this technology and the development flow for neuronal networks could have in future the following steps:

1. Collect and qualify real-world data with the installed fleet of IoT endpoint devices. The data are stored and analyzed on cloud servers.
2. The cloud may be also used to identify suitable neuronal network architectures and algorithms.
3. Tailor and re-training of the ML model requires typically a high-performance cloud computer.
4. For verification of the ML model, the regression testing on virtual simulation platforms may be used as described in section III.B of this paper.
5. After validation, a new firmware image or new ML model data set can be deployed to the IoT endpoint device as described in the previous section.

Correct decisions can only be made in areas where training data exist [16]. “Learning” means therefore that ML algorithms are retrained based on new data that delivers new “experience”. Fig. 4 illustrates that correct decisions cannot be done in areas where insufficient training data exist. For example, if a picture recognition application never “saw” the picture of a cat, it cannot be correctly qualified. It is therefore expected that IoT endpoint systems that incorporate AI and ML technology require periodic updates.

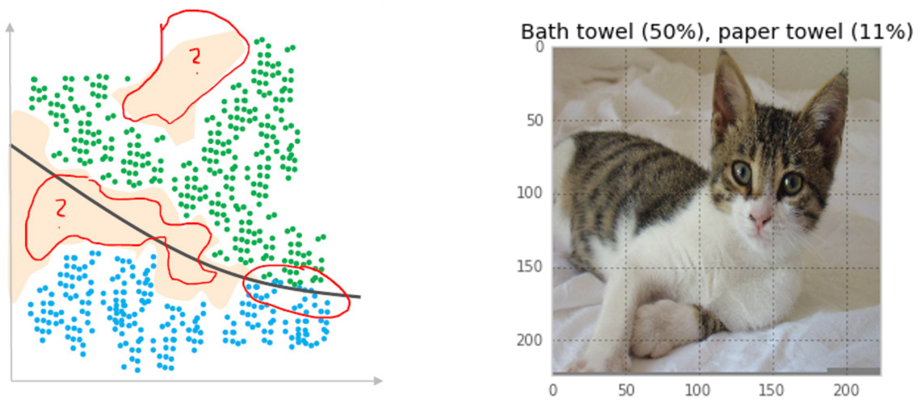


Fig. 4: ML requires real-world data for correct decisions [16]

Summary

Multinational corporations already extensively utilize cloud technologies, as advantages range from better collaboration in distributed development teams to the product life cycle management of deployed IoT endpoint devices.

The cloud is also attractive for smaller companies as the IT infrastructure can be simplified. With the raising trend towards home office, cost savings can be achieved by relocating software development tool environments at least partially to the cloud. Arm will therefore support a dual approach: cloud and desktop. The new Keil Studio – cloud can load projects created with the classic desktop version, Keil MDK, and vice versa. This gives development teams the flexibility to adopt cloud technology over time.

Many innovative products with the associated service, for example Alexa's voice recognition technology, are inconceivable without the cloud. It is expected that cloud technology will further accelerate innovations in the whole range of IoT, for example in medical fields where the COVID-19 pandemic created new challenges that require fast execution.

Today, the integration of cloud services in small embedded IoT endpoint devices is still a challenge. Arm therefore works with various industry partners on software stacks that will reduce this complexity. And with TF-M and PSA the foundation for security will be standardized.

References

[1] Arm Neoverse Series Processors

[2] Arm Cortex-M Series Processors

[3] Arm Ethos-U Series Processors

[4] Arm Keil Studio Cloud

[5] CMSIS: A success story

[6] MDK Microcontroller Development Kit

[7] Functional Safety

[8] What Is Continuous Integration and How to Benefit From It?

[9] Software Development Without a Hardware Target

[10] Infrastructure for Continuous Integration Tests

[11] Improve embedded software unit testing efficiency

[12] Trusted Firmware M (TF-M)

[13] Arm TrustZone Technology

[14] Platform Security Architecture

[15] Solutions for Artificial Intelligence

[16] How to trick a neural network