# 5 Tips for Fast Debugging of Deployed IoT Devices

How embedded software teams find and resolve bugs before customers report them

# 5 Tips for Fast Debugging of Deployed IoT Devices

No product is entirely bug-free and it’s up to embedded software development teams to minimize the impact of anomalous behaviors in the field.

For IoT devices in particular, with thousands of units in the hands of creative and unpredictable users, a fast diagnostics, resolution and update mechanism is key to eliminating bugs before they become a product recall.

How can you protect the reputation and revenue of your IoT product line? Consider these five best practices to reduce operational risks.

## 1. Adopt a “There will be bugs” methodology

Can embedded software teams deliver bug-free products? No.

Research shows that there are between 15 to 50 bugs introduced per thousand lines of code and about 5 percent make it through to deployment. With all possible combinations of application code, operating system, drivers, firmware, open source, network connections and more, these numbers should come as no surprise. Include the many ways that a user acts outside your test plans and a bug is almost guaranteed.

While some bugs are harmless, the same research shows that around 20 percent are major defects. Given a modestly sized project with 50,000 LOC, there could be as many as 15–50 user-facing anomalies. That poses a significant risk to customer satisfaction.

It’s not enough to recognize these problems. Steps must be taken and tools implemented to accelerate the debugging of deployed devices.

**Research shows that around 20 percent of bugs are major defects.**

Ensure your software collects and stores diagnostic information on runtime warnings and errors. The application can write periodic messages to a circular buffer in RAM and copy them to non-volatile memory upon encountering an issue. These messages should include specific information about the issue and the events leading up to it to assist developer investigations later.

## 2. Provide automatic alerts to the development team

Collecting diagnostic information is a good first step in reducing your debugging time. Typically, this information is only accessible by a support team member working on a ticket or when the user triggers a bug report. Both cases are too late as the customer is already in the loop.

To minimize the gap between problem occurrence and you knowing about it, devices should implement a real-time alerting mechanism. Second-hand data is also prone to gaps and errors in context – customers are rarely able to articulate useful information about the problem they had. Having direct access to the diagnostics data eliminates any confusion.

An automatic alerting mechanism, to send diagnostics data from the device to the development team, provides vital information about issues within seconds of the first occurrence. Rather than wait for a customer support ticket, such a mechanism gives you immediate access to real-world device diagnostics to speed up problem resolution, especially when the root cause may not be obvious from the user-facing symptoms.

Once in place, an automatic alerting mechanism also supports system testing, as developers get instant notifications about issues in their prototyping and pre-production code.

## 3. Monitor key performance metrics

It's not just about the bugs. With onboard diagnostics and automatic alerting comes the ability to make better development decisions. Designed correctly, these mechanisms provide valuable feedback about the real-world use of your IoT product.

Defining custom alert types to capture conditions of interest helps you monitor how code behaves in practice and discover use cases not considered before.

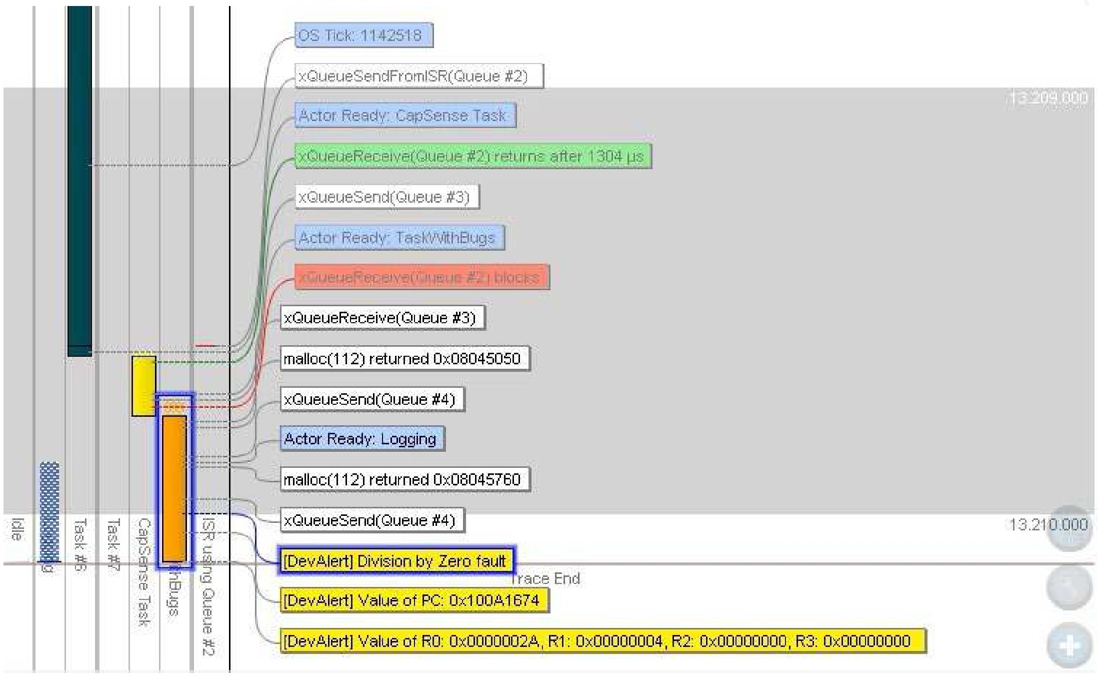
Maybe you want to measure product performance, such as start-up time latency and battery life, and send alerts from device to desktop when the application exceeds thresholds. You could also identify the features most frequently used by customers to make more informed decisions on future product improvements.

Using visual trace diagnostics reveals the context of what was happening on the device as the issue occurred and helps you understand and reproduce the problem much faster than traditional debugging approaches.

## 4. Use visual trace diagnostics to accelerate fixes

While diagnostic data may contain detailed software traces of threads, API calls, state changes and interrupts, it's difficult to scroll through multiple logs and isolate the root cause of a bug. Analysis of such information is much faster with visual trace diagnostics, providing both a visual timeline of events and multiple visual overviews from different perspectives of the software.

Using visual trace diagnostics reveals the context of what was happening on the device as the issue occurred and helps you understand and reproduce the problem much faster than traditional debugging approaches.



```
OS Tick: 1142518
xQueueSendFromISR(Queue #2)
Actor Ready: CapSense Task
xQueueReceive(Queue #2) returns after 1304 µs
xQueueSend(Queue #3)
Actor Ready: TaskWithBugs
xQueueReceive(Queue #2) blocks
xQueueReceive(Queue #3)
malloc(112) returned 0x08045050
xQueueSend(Queue #4)
Actor Ready: Logging
malloc(112) returned 0x08045760
xQueueSend(Queue #4)
[DevAlert] Division by Zero fault
[DevAlert] Value of PC: 0x100A1674
[DevAlert] Value of R0: 0x0000002A, R1: 0x00000004, R2: 0x00000000, R3: 0x00000000
```

## 5. Deploy over-the-air updates before customers see problems

Over-the-air (OTA) updates are the new normal. Already in wide use by most mobile applications, more complex systems are embracing the ability to improve their features and cybersecurity strength remotely. The automotive industry projects 120 million vehicles sold annually with OTA update capabilities and recent FDA guidance recommends "capabilities that enable device patching and updating in a timely way."

OTA updates are invaluable because the sooner a bug is fixed or a security patch released, the fewer customers are negatively affected.

Continuous IoT device monitoring provides immediate awareness and detailed diagnostics to enable more effective OTA updates. By using visual trace diagnostics to identify and solve problems faster, you can update devices in the field more quickly.

## Conclusion

Following these five steps helps you get the detailed information necessary to reduce deployment and maintenance risks for IoT devices. By closing the loop between an anomaly occurring in the field and a developer delivering a fix, embedded software teams achieve a more efficient continuous software improvement process.

**Percepio DevAlert provides IoT device monitoring for use in system testing, field testing and deployment. DevAlert notifies developers about abnormal conditions detected in the device software, with visual trace diagnostics to speed up resolution.**

## ABOUT THE AUTHOR//

Dr. Johan Kraft is CEO and founder of Percepio AB. Dr. Kraft is the original developer of Percepio Tracealyzer, a tool for visual trace diagnostics that provides insight into runtime systems to accelerate embedded software development. His applied academic research, in collaboration with industry, focused on embedded software timing analysis. Prior to founding Percepio in 2009, he worked in embedded software development at ABB Robotics. Dr. Kraft holds a PhD in computer science.

## Request an evaluation of Percepio DevAlert here.