



## Review of battery powered embedded systems design for mission-critical low-power applications

Matthew Malewski, David M. J. Cowell & Steven Freear

To cite this article: Matthew Malewski, David M. J. Cowell & Steven Freear (2018) Review of battery powered embedded systems design for mission-critical low-power applications, International Journal of Electronics, 105:6, 893-909, DOI: [10.1080/00207217.2017.1409813](https://doi.org/10.1080/00207217.2017.1409813)

To link to this article: <https://doi.org/10.1080/00207217.2017.1409813>



© 2017 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group.



Published online: 26 Dec 2017.



Submit your article to this journal [↗](#)



Article views: 440




View related articles [↗](#)



View Crossmark data [↗](#)

# Review of battery powered embedded systems design for mission-critical low-power applications

Matthew Malewski, David M. J. Cowell and Steven Freear 

School of Electronic and Electrical Engineering, University of Leeds, Leeds, UK

## ABSTRACT

The applications and uses of embedded systems is increasingly pervasive. Mission and safety critical systems relying on embedded systems pose specific challenges. Embedded systems is a multi-disciplinary domain, involving both hardware and software. Systems need to be designed in a holistic manner so that they are able to provide the desired reliability and minimise unnecessary complexity. The large problem landscape means that there is no one solution that fits all applications of embedded systems. With the primary focus of these mission and safety critical systems being functionality and reliability, there can be conflicts with business needs, and this can introduce pressures to reduce cost at the expense of reliability and functionality. This paper examines the challenges faced by battery powered systems, and then explores at more general problems, and several real-world embedded systems.

## ARTICLE HISTORY

Received 23 June 2017

Accepted 6 October 2017

## KEYWORDS

Embedded Systems;  
reliability; mission critical;  
low-power

## Introduction

An embedded system is a computer system that is built specifically to complete set tasks. Applications of embedded systems can vary from consumer, aviation, and space equipment. Systems can be almost, or completely isolated from human interaction, and may be expected to perform in such a state for many years. Because of the range of applications, there is no single strategy for designing an embedded system.

Mission critical systems are ones that are required for the successful completion or operation of a system. Safety-critical systems are ones that could potentially incur loss of life if a failure mode occurs (Fowler, 2009). Embedded systems for these applications are typically required to be operational with almost zero downtime. To minimise the downtime of these systems several techniques are used, examples include redundancy and watchdog timers (WDT). Further challenges are presented by the operational, environmental, and implementation constraints that these systems typically operate in.

Systems that are required in areas with unreliable power sources, or those which require uninterrupted power supply will often use a secondary power source, usually a battery, to allow continuous operation. Due to the energy limitations of batteries, and any other potential power source, it is important to ensure that embedded systems use as little energy as possible.

Faults are undesired or undefined operations. Typically in consumer products, embedded systems are not designed robustly and it is commonplace for users to cycle the power in order to restore operation. However, in mission and safety critical systems it is important to incorporate features to reduce the chances of faults occurring, and also create the appropriate mechanisms for

allowing the system to detect and correct any fault so that a system can attempt to return to a safe operating condition.

The problem landscape of embedded systems also extends to the software domain. The development of either hardware or software domain affects the complexity of the other. Software complexity and code length has been increasing over the years, and with that the probability of defects and errors increases. Typically, commercial software has 10–50 defects per thousand-lines of code (Ebert, 2009; Microsoft (S.McConnel)). Electronics and software development consists of 35–40% of a car cost (Rivett, 2005).

This paper discusses some of the methods of mitigating, detecting, and handling failure scenarios. It explores, and analyses some real-world examples of embedded systems that have experienced failures.

## Power considerations

Achieving low-power consumption in any embedded system is important. Embedded systems in space applications are an example where there is a defined, and typically low, power budget which must be met. Even in applications where at first it might appear that power consumption is a lower priority issue, such as in automotive applications, the growing use of embedded systems to replace mechanical systems has a large impact on quiescent power consumption (Little, 2015).

### Microcontroller operation

There are two extreme microcontrollers operating regimes: high-duty-cycle low-clock frequency systems; and low-duty-cycle high-clock frequency systems. Each regime presents unique challenges in power management and system design. A high-duty-cycle low-clock frequency system requires a larger time to complete a task. This regime tends to result in a peak current consumption that is closer to the average current consumption, which can be beneficial for some power sources. If low-power sleep states are not entered, this can also reduce software complexity. Figure 1 illustrates the two regimes, and how the peak current and average currents are related.

Operating with a low-duty cycle with a high-clock rate system, the completion of tasks is relatively quick, allowing the system to temporarily enter a low-power state between operations. This approach provides the greatest flexibility for applications where computational load can vary greatly (Segawa, Shirota, Fujisaki, Kimura, & Kanai, 2014).

In situations where battery capacity is low, the high peak current of a low-duty cycle high-clock frequency regime might cause a brown-out scenario due to a drop in rail voltage as the system draws more power. Figure 2 illustrates this issue: A system which enters a brown-out scenario due to high current draw of a high-performance MCU. The system is operating in a low-duty cycle regime, with a microcontroller periodically operating for short periods of time. The large current spikes cause voltage drops in the power rails close to brown-out levels (2.25V). In a high-duty-cycle regime the peak and average current draw are similar – hence the system voltage remains stable and does not approach brown-out.

The application of the embedded system will dictate the strategy. It is important to accurately estimate the computation load as in scenarios where the computational load is too high, reboot/reset scenarios might occur (National Highway Traffic Safety Administration, 2011).

### Rail voltage

To minimise the overall power usage of the system, a battery can be used to power the rail directly. This removes need for voltage regulation, and mitigates DC-DC conversion losses. By not regulating the power rails, an uncertainty is being created, in that the system voltage is not stringently defined. To ensure operation, the system should employ components which are rated for the

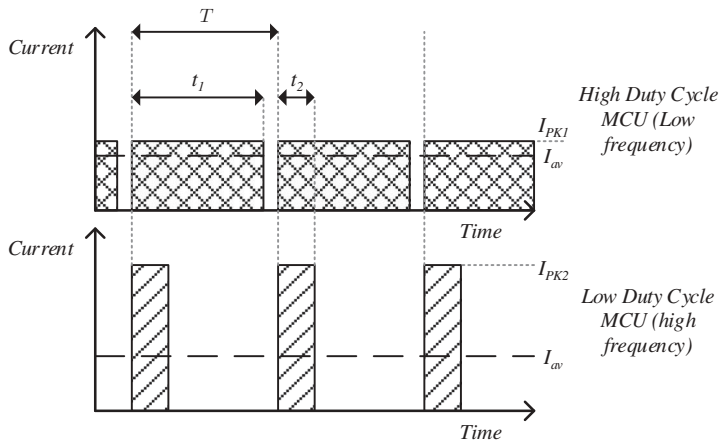


Figure 1. Comparing the current consumption of low and high duty-cycle systems.

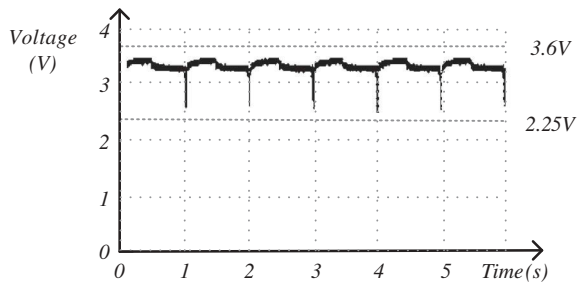


Figure 2. A system which enters a brown-out scenario due to high current draw of a high-performance MCU.

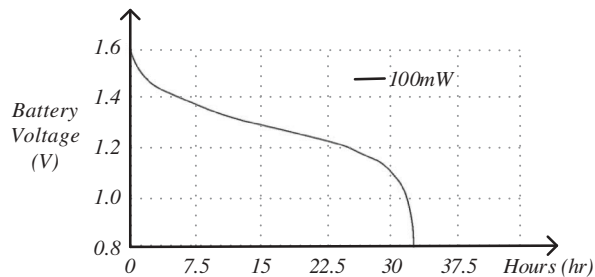


Figure 3. Terminal voltage with respects to time during at a constant 100mW power drain of a typical alkaline-manganese dioxide battery.

maximum and minimum allowable battery voltage. Brown-out detectors will be required to reset the microcontrollers when the voltage drops below rated operating levels (Huy-Binh, Xuan-Dien, Lee, & Ryu, 2011).

Alternatively, it might be more important for a system to employ a stable rail voltage by using a voltage regulator or converter. The use of a DC-DC converter can allow a system to utilise more capacity of a battery if the rail voltage has a tight region operating voltage range. Figure 3 shows the typical discharge characteristic of an alkaline-manganese dioxide battery when discharged at a constant power of 100 mW (Duracell Batteries [Online], 2016). If a system was able to be powered directly from the battery over the voltage range of 1.6–1.2V, the battery capacity available is

2.4 Wh. By implementing a DC-DC converter to allow the useable battery voltage range to increase to a range of 1.6–1.0V yields an extra 31.25% battery capacity, 3.1 Wh. Assuming 85% converter efficiency, then the total capacity has increased by 11.5% compared to a direct battery connection.

### Battery characteristics

Batteries use chemical reactions to generate electrical energy, and thus are affected by environmental conditions. Figure 4 shows the capacity of a D-size lithium thionyl chloride battery. The battery capacity is measured as the amount of current the battery can supply over a period of time from the initial terminal voltage of 3.6V to an end-of-life terminal voltage of 2V. The nominal current is specified as 6mA at 25°C, which relates to a peak capacity of 19Ah. Lithium Thionyl Chloride batteries are commonly used in low-power, long service interval equipment (Crompton, 2000). Figure 4 shows the effect of temperature and current-draw on battery capacity (Tadiran Batteries, 2016).

Temperature affects the rate of reaction, resulting in variation of output voltage. Changes in load have a greater effect on voltage swings at lower temperatures, as can be seen in Figure 5.

The systems operating regime needs to be carefully selected so that over its operating temperature the battery voltage is within limits. Lithium Thionyl Chloride batteries are a very high energy density battery which is well suited for long discharge durations with low peak and average currents, and are commonly used in the medical, military, commercial and space industries.

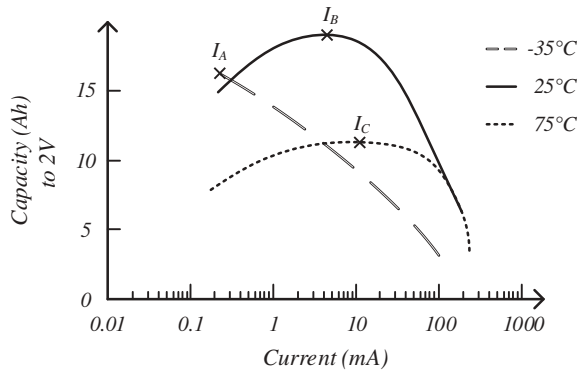


Figure 4. Lithium thionyl chloride battery capacity as a function of current draw and temperature.

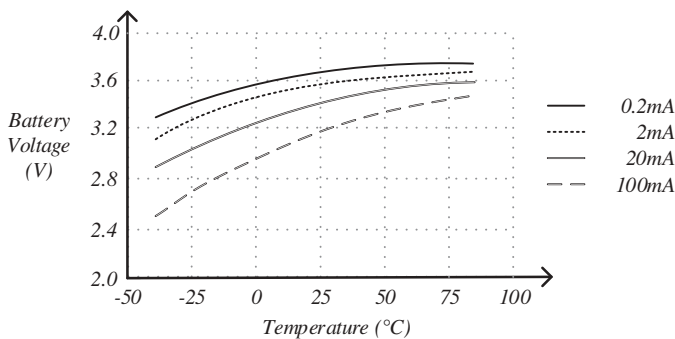


Figure 5. Battery terminal voltage vs. temperature under different current loads.

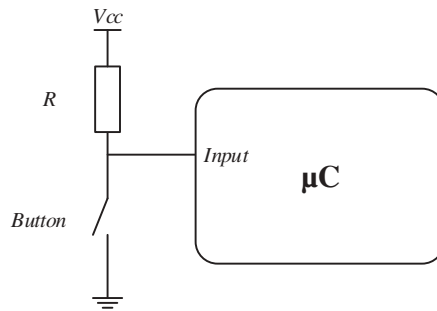


Figure 6. Simple button input to a microcontroller.

### Redundant power supply

Redundant power supply is the use of multiple power sources to ensure that system power is maintained (Phoenix Contact, 2017). Often, it is required that an embedded system is able to maintain a clock during times when the main system's power is cycled, usually to maintain a real-time clock. Computers being a notable example where a secondary power source is used to maintain the BIOS information and system clock.

## Software

### Data quality

Software issues can arise because of assumptions of the type and size of data. Software that interacts with users is especially susceptible, and should ensure that data is of an acceptable format, type and length. This is common practice in the IT industry, where SQL servers need to be able to detect and sanitise their inputs to protect against SQL injection (Shar, 2012). This is a form of defensive programming. Implementing data checks helps to catch errors, and prevent failures and/or damage to the rest of the system. The addition of error checks impacts computation time and resources.

Data corruption can be caused by various sources. These can be environmental impacts, or defects in hardware or software. The capability of detecting and correcting errors should be designed into an embedded system as it greatly increases the reliability of the system (NASA Lesson Info, 2016a). Error detection can be implemented using a simple parity bit. Upon detecting an error, the data can be re-requested. If correction is required, cyclic redundancy checks (CRCs) can be implemented. CRC is one of the most versatile error checking algorithms and is widely used throughout the electronics and IT industry (Microchip, 2008). Microcontrollers nowadays have dedicated CRC modules that allow hardware to generate and compute CRCs.

It is preferential that commands should be predefined rather than using general memory updates (NASA Lesson Info, 2016b). For example, an embedded system that employs user-defined configuration data sent from a computer should implement a separate update-command for each configuration variable, rather than having a generic command that allows memory alterations defined by the user. An additional benefit of restricting commands is that it can strengthen the security of a system.

### Compilers

Compilers translate code written in a programming language (such as C) into a language native to the hardware platform (such as Assembly). Different compilers interpret the source code in different ways which can result in varying system performance, size, and reliability (Park, Han,

Lee, & Kim, 2014). Software success criteria should indicate the key features of compiled code, so that the optimal compiler for the application can be chosen.

Higher compiler optimisation can result in a reduced computation time and more efficient processor utilisation, which can yield an increase of reliability (Demetzi, Annavaram, & Hall, 2011).

### **Tracing**

Tracing is the act of logging events and information regarding the programme execution and system variables. In contrast to event logging, tracing tends to be low-level data. Tracing provides information that is typically used for debugging purposes during development and during fault finding scenarios (Spear, Levy, & Desnoyers, 2012). Any reset event that occurs should be logged, with information such as the time, date, and programme counter. Event data should be stored in non-volatile memory so that it is not affected by brown-out scenarios. Real-time clocks should be employed with redundant power source so a constant clock is maintained. Any error displayed to the user should be clearly written, with the error meaning clearly stated. The Therac-25 is an example of where users ignored error messages as they did not explain the problem case, or the users could not easily find out their meaning (Leveson & Tumer, 1993).

### **Testing**

Software reliability is key in ensuring an embedded system is reliable.

Estimates of the number of defects in code made commercially available is approximately 10–50 defects per thousand lines of code (Ebert, 2009; Microsoft (S.McConnel)). The code coverage provided by tests is directly linked to defect discovery. It is therefore important that unit tests cover at least 80% of the code (Frate, Garg, Mathur, & Pasquini, 1995). Typically, an embedded system will exercise a small percentage of code during normal operation, making it difficult to be able to have large code coverage by simply operating in normal conditions. This is especially true with error-checking, where the majority of the time these segments are unused. It is important to inject incorrect data during testing to ensure the system handles and recognises errors correctly. This testing greatly increases the development time of software.

### **Static code analysis**

Static code analysis is a method of software debugging by analysing code without execution. Static code analysis can be performed on either source-code or compiled code (Louridas, 2006). This test method is limited in the amount of defects that can be detected. Studies have shown that this method can only capture 45% of test detectable defects (Lauesen & Younessy, 1998).

### **Programming practice**

Major mission and safety critical industries might have specific codes of practice or standards relating to software development. MISRA-C and DO-178C are examples of two commonly used standards in the automotive and aviation industries (Rivett, 2005).

### **Bit redundancy**

Care should be taken whenever data is interpreted. Electro-magnetic interference can alter the values of bit registers. The system should be designed such that if a single bit is affected, it does not compromise the whole system. For example, if a function can return one of two values, where 0b indicates normal operation, and 1b is a failure, then a single bit change can alter the operation. If however the values were 0b and 0101b, two bits would be required to change their state in order perform in error.

## Software re-use

Software re-use is the re-use of software from other or previous projects. This is often done by users as it saves time and there is an assumed reliability of the software as it has been used previously. It is important to note that software errors might not present themselves, or might not have been observed because of other system mechanics, such as safety or supervisor circuits. This was the case with the Therac-25, where errors were unknowingly hidden by mechanical fault-protection resulting in software being wrongly assessed as reliable, and eventually through re-use in a system without mechanical fault-protection, fatal errors occurred (Leveson & Tumer, 1993). It is important to ensure that re-used modules are also tested in the new system. Reliability should never be assumed. This problem is also present in COTS software.

## Fault prevention

### Debouncing

Any form of electro-mechanical equipment in their transient state can cause undesired responses. For example, relays or buttons can exhibit rapid mating and un-mating of a switch and contact, which is known as contact bouncing (Ciocirlan & Herrmann, 2009).

Contact bouncing can cause a variety of hardware and software issues, and can generate EMI which can disrupt neighbouring components (Zhai, Yang, & Zhou, 2008).

Take the following system shown in Figure 6. A button is used which is an input into a microcontroller. The button event triggers an interrupt on change.

Upon pressing the button, a metal strip is mechanically deformed to make a contact. It is likely for the metal to bounce initially before settling, causing the following voltage response. Figure 7 shows the voltage seen at the input of the microcontroller with respects to time. This is a potentially dangerous situation if the button drives an interrupt as several interrupt events might be generated.

One of the simplest methods of debouncing is by implementing a low-pass filter using an RC circuit, which if properly implemented can reduce most of the high-frequency components (Mancini, 2008).

Dedicated debouncer ICs are available, which count the duration of the pulse, and provided a steady state is maintained for a set length of time, the output changes accordingly.

Pure software implementations are possible, but the input voltage has no guarantee that it is within input specification for the device.

Software debouncers could be employed in addition to hardware debouncers as induced voltages caused by EMI can occur between microcontrollers and their protective circuits. EMI can cause similar effects as discussed here.

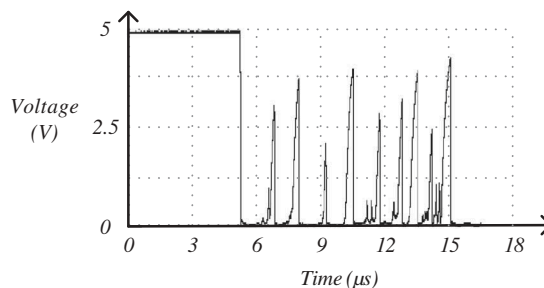


Figure 7. Voltage over time graph of the button being pressed whilst exhibiting contact bouncing.



## Unused components

Not all systems will utilise all of the populated components or microcontroller peripheral. Incorrect handling or configuration of unused components can cause system instability or performance issues.

Unused op-amps need to be properly terminated. Their high gain means that small changes induced on the inputs can result in large output charges. This can lead to increased power consumption, more heat, and noise (Mancini).

Incorrect termination of unused pins also affects digital components. Microcontroller pins set as inputs can introduce noise into the system, and any analogue component may have similar problems as with the op-amp. Typically, applications will set unused pins to outputs and in an open configuration to mitigate this (PIC MCU, 2007). However, in mission-critical applications this might not be sufficient. During power-up of the microcontroller the pins are configured as inputs, and if left unconnected, their state is unknown. This paper recommends that unused microcontroller pins should be tied to ground via a resistor (1–10k $\Omega$  approximately) and configured to output low. As microcontrollers typically initialise pins as inputs, if the pin is not tied to a defined potential, the state is ambiguous, the pull-down resistor will define this. The resistor acts as a current limit in situations where a pin is initialised incorrectly as an output driving a grounded pin high.

## Electromagnetic susceptibility

Electromagnetic interference (EMI) can cause hardware to behave in an unexpected manner. The likelihood and effects of EMI is minimised by employing good PCB design practices. This paper will not cover these design practices as they are well documented (Infineon, 2016a; NXP, 2013; Texas Instruments, 1999).

## Power decoupling

It is important to ensure that the system has a stable ground plane, and that there is sufficient decoupling (See, Manish, Liu, & Kyaw, 2004).

Every IC should have a decoupling capacitor placed near its pins. This helps to limit the impact of an IC on neighbouring or global components. Failure to properly maintain stable voltage rails can lead to problems, such as false write in EEPROMs, which are usually the result of noise in the ground and/or voltage rails (Greenliant, 2016).

## Fault detection and handling

### Software structure

The structure of the software will be dictated by the application. Systems with deterministic and strict periodic tasks can be written in a stricter manner than those that have aperiodic or sporadic tasks. State-machines are a common way of implementing an embedded system that is driven by external input, such as a USB driver stack.

Extra care should be taken whenever a task dependent on the actions of another system or user is invoked. Event handlers need to be in place to enforce deadlines for tasks and handle errors. The use of timeouts is required to stop the system from hanging (Ihrig, 2009).

A method of ensuring that the system calls the correct functions in the correct order is by using net zero counter, which is incremented during execution and reset at the end of the loop using a decrement. Functions compare the expected counter value against the actual counter value to determine whether it has been called correctly (see Figure 8).

Assuming that the counter,  $j$ , has a value of zero at point A, the function, Function 1, would only expect to see  $j$  with a value of zero. If this is the case, the function has been called correctly. The WDT can be reset at this point. If however, the value of  $j$  is not zero, an error has occurred, and the

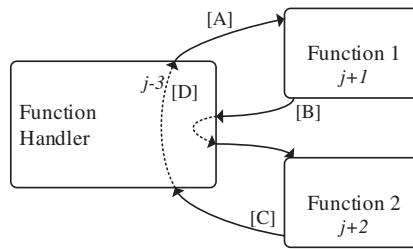


Figure 8. Simplified software to monitor the calling of functions.

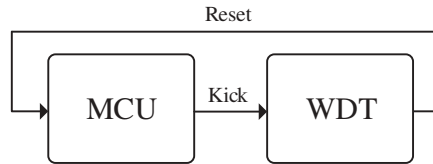


Figure 9. Single Stage WDT system.

system could either try to resolve the problem in software, or wait to be reset by the WDT. Once the function has completed its task, the counter  $j$  is incremented. The next function, Function 2, is called, and it performs the same checks against the counter, but this time it checks for a value of 1 instead of zero. Once the whole function loop is complete, the counter is decremented by 3. By not asserting a value to  $j$ , the system is reliant on correct operation for the counter to maintain a correct state. It should also be noted that the increments are not equal, so that the programme can distinguish incorrect function order calling.

### Watchdog timers

WDT provide a facility to attempt to reset a system from an unknown fault state to a safe known operating state.

In the most simplistic view, a WDT is essentially a count-down timer that causes a system reset once the timer has elapsed. These timers can be reset by asserting a register-bit high in the case of an internal WDT, or by asserting a signal-line high in the case of an external WDT. The act of resetting a WDT is commonly referred to as a 'kick'. Figure 9 illustrates a simplified functional diagram of how the MCU and WDT interact.

WDT have specific timing requirements for when they can be reset, which must be met. Missing a deadline will indicate an anomaly or fault has occurred, and should be reset. WDT timing intervals should be kept as small as possible to improve system response. Figure 10 shows a simplified timing diagram of the operation of a WDT. The kick must occur within the expire-time,  $T_{expire}$  for the counter to reset. If it does not, like with the case of  $T_2$  the system will reset.

Most production microcontrollers have a simple internal WDT. These internal WDT can provide basic protection, but depending on microcontroller architecture, they can be dependent on the system clock and thus should not be relied upon in mission critical applications. External WDT tend to provide better protection as physical component isolation is present. External WDTs are commonly packaged with additional circuitry, such as Brown-out Voltage Detection (BOV).

Windowed WDTs are a type of WDT which have a more stringent timing requirement to detect fast and slow failures. Fast failures occur when the kick occurs before it is expected – possibly indicating that the required tasks have not been completed. A slow failure, as illustrated in

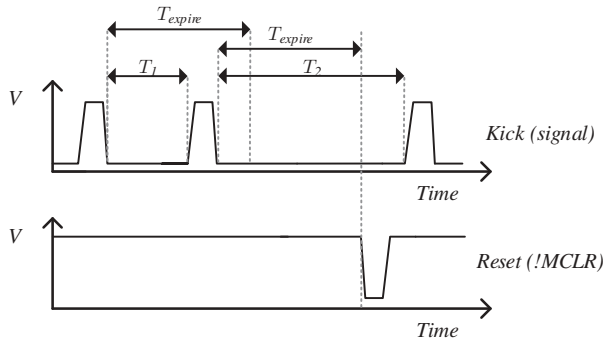


Figure 10. Timing diagram of a WDT system.

Figure 10, where a WDT failures to be kicked within the specified time, may indicate that the system has hung (El-Attar & Fahmy, 2007).

### Redundancy

Redundancy is the practice of implementing more than the minimum required number of parts with the aim of increase system reliability.

By duplicating or adding components or sub-systems in parallel, the overall system reliability increases. This can be represented by (1) (NASA Lesson Info, 2016c) where  $R_s$  denotes whole system reliability, and  $R_c$  denotes component or sub-system reliability,  $n$  being the number of components:

$$R_s = 1 - \prod_{c=1}^n (1 - R_c) \tag{1}$$

Figure 11 shows a resistor network that has two resistors,  $R_1$  and  $R_2$ . Assuming fixed film resistors, the failure rate ranges between 1.0 and 0.001% per 1,000 hours of use at a 60-confidence level (NASA, 1988). This yields a system level reliability of 99.99 and 99.99999999%, respectively.

Figure 12 illustrates the increasing reliability of a system when additional components/sub-systems are added in parallel.

Redundancy is not limited to hardware implementation. It should be implemented within software as well (see section III.G).

Conversely, implementing components in series results in a decrease of reliability. This can be represented by (2) where  $R_s$  denotes the whole system reliability, and  $R_c$  denotes the component or sub system reliability,  $n$  being the number of components:

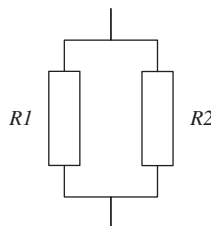


Figure 11. Parallel resistor network.

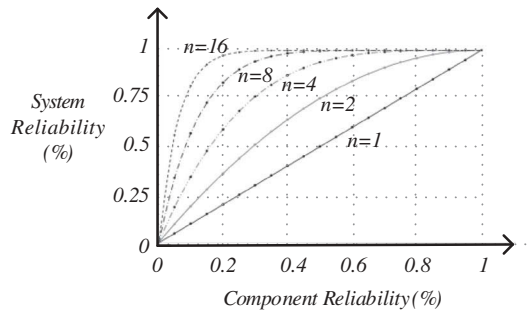


Figure 12. Component reliability vs. system reliability with n-number of parallel components.

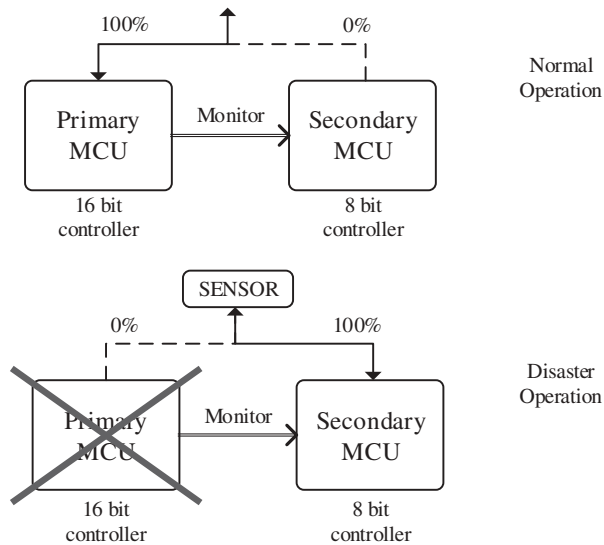


Figure 13. Sensor interaction in normal and disaster operation modes.

$$R_s = \prod_{c=1}^n R_c \quad (2)$$

### Failover system

Failover systems are a form of redundancy in which a secondary system will continue the operation of the primary system when the primary system experiences a failure. The failover system can either provide all, or just the core functionality of the primary system (Sun, Gong, Dong, Zhang, & Wang, 2014). The practice of implementing a failover systems is commonly used in other fields, such as web-servers in IT (Plankis, Horning, Ponto, & Brown, *in press*).

A failover device must monitor the operation of the primary system, and in the event of lack-of or incorrect operation, the secondary system needs to take over. Care should be taken as the increase in features and functions increases points of failure, possibly reducing system reliability. The failover system can mirror the operation of the primary system until stable operation is achieved. Once the primary system is operational, the failover system can be placed into a reduced power state. Figure 13

illustrates a system which uses a primary and secondary system to monitor a sensor. When the primary system is not operational, the secondary system can interact with the sensor.

It is generally recommended that the primary and secondary systems share as few characteristics as possible. Both microcontrollers should be of a different architecture, and possibly from of a different manufacturer. The software of both the primary and failover devices should be independently developed using different coding strategies or structures, and compiled by different compilers, which will reduce the chance of software bugs being inherent on two devices (Leveson & Tumer, 1993). The failover system has to have the same reliability or greater than the system it is protecting.

Alternatively, the failover system can provide functional assurance. Mirroring the decision-making process of the master, and allowing the master to operate. This means that both systems need to generate the same output for operation.

Regardless of the implementation, either system, primary or secondary, should be monitored with telemetry of the systems being recorded (NASA Lesson Info, 2016d).

## Mechanical considerations

### Shock and vibration

Shock and vibration are a major consideration when it comes to mechanical components. The US Air Force estimates that vibration and shock account for 20% of all mechanical failures in air-bounce electronics (Steinberg & Associates (Dave S. Steinberg), 2017).

In section IV.A, it was discussed that buttons can cause problems due to rapid un-mating and mating of contacts. This can also be experienced by mechanical components undergoing stress due to shock and vibration.

Heavier objects have more kinetic energy, and as a result they require greater support. Batteries can be an issue in high vibration environments, where standard sprung contacts might not have sufficient mechanical strength to hold a battery within a holder, which under certain shock and vibrations scenarios contact resistance and/or connection might change which may result in the system browning out (RCA Service Company (Robert E. Barire, Wayne Hall), 1958). Figure 14 shows an AA battery strapped in to a standard sprung-contact holder being subjected to shock in the direction of the major dimension. The shock has caused the battery to bounce disconnect for long periods of time.

Typically, solutions use either specially designed batteries which have high amounts of mechanical support, or by using a flying lead with a connector to provide mechanical isolation between the connector and battery. Both approaches can be seen in the mobile phone industry.

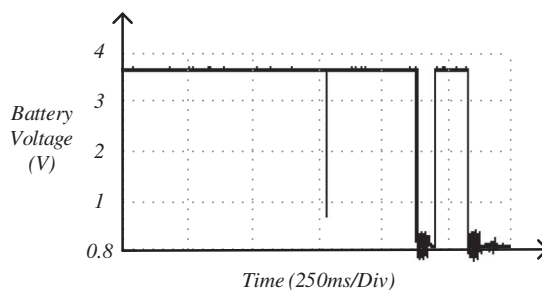


Figure 14. Voltage across a battery holder contacts during a shock scenario.

## Connectors

Connectors like any mechanical system are susceptible to environmental changes. The nature of connectors being removable introduces potential hazard areas in a circuit. Field data has shown that 30–60% of all electrical failures are connector related (Swingler & McBride, 1998; Wallinder & Eriksson, 1998).

It should also be noted that connectors in series have an inverse effect on reliability as discussed in section V.C.

## Environmental

The environment in which a system operates in needs to be taken into consideration as this can have a large impact on the system.

Temperature can impact the performance of components, such as batteries [Sec. II.C], and also their usable life. This is especially true with electrolytic components, which may dry out when exposed to high temperatures. Often it is quoted that capacitor lifetime halves for every decade temperature increases. This holds true for semi-conductors at lower temperatures (Texas Instruments, 2014).

The levels of humidity in the environment can impact the embedded system. High levels of humidity can lead to condensation of water onto circuits, which can result in corrosion of components (Zorn & Kaminski, 2014).

## System review

Many mission/safety critical systems implement characteristics as this paper discussed. This section shall briefly review and highlight some.

### Fire suppression control system

A fire suppression control system was examined. It was able to interface with several sensors, and utilised squib based actuators to actuate the suppression system. Squibs are a miniature explosive used to generate gas, and are commonly found in suppression systems and airbag systems in cars.

The fire suppression control system was able to fire multiple squibs in series. Squibs post discharge present themselves as an open-circuit, and in order to detect a fault or post-discharge scenario, squibs were arranged in series.

Interconnecting cables were a predefined set length, with connectors either end. If a cable with a length other than the predefined lengths was required, multiple cables were required. This cascading of connectors increases the failure rate.

The system did incorporate power redundancy so that during main power source outages the fire suppression system was still functional. It was also able to maintain a clock during primary and secondary power outages. Data logs were stored to non-volatile memory.

### Automotive ECU

Figure 15 shows the recommended architecture of an airbag controller by Infineon (Infineon, 2016b).

The system is reliant on two input sources, satellite (external) and on-board (internal) sensors. The master microcontroller does not interface directly with the satellite sensors, but instead the functionality is handled by an interface controller. The connection of the satellite sensors is not guaranteed, and therefore on-board sensors are used if a connection or communication issue arises.

The failsafe microcontroller provides functional assurance of the master microcontroller. It is able to interface with the same sensors, that is, the satellite sensors and on-board sensors. It provides an input to the Squib drivers, preventing or enabling deployment of the Squibs.

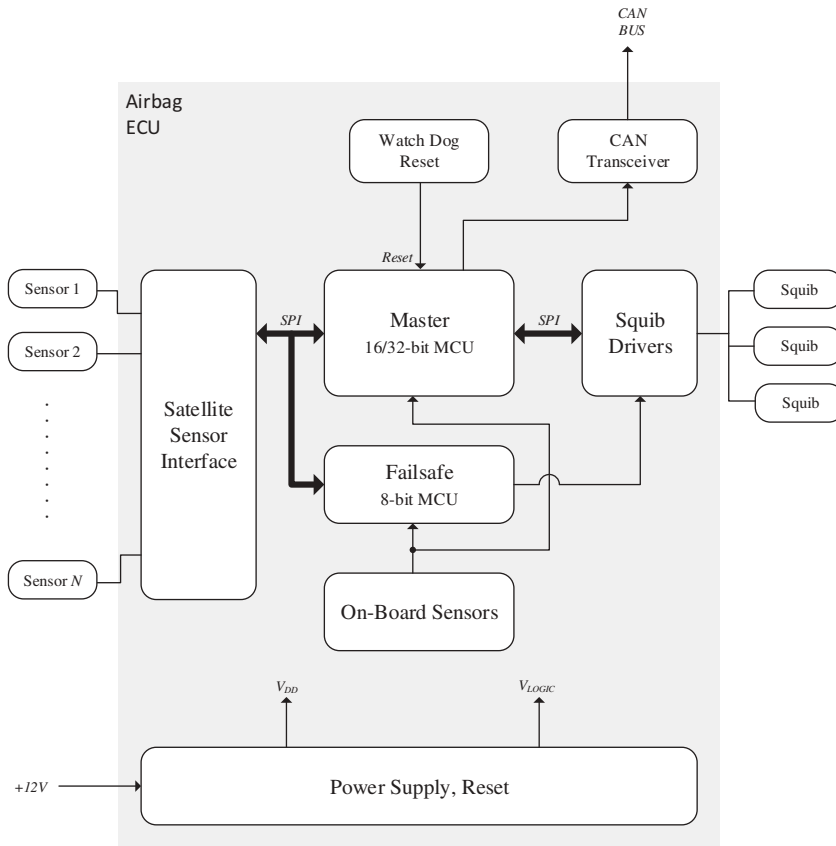


Figure 15. A recommended system for an airbag controller.

The master microcontroller is protected by a watch-dog timer. The WDT is external to the microcontroller it is protecting.

### Toyota pedal controller

Several accidents in the first decade of the 21st century were attributed to the design and implementation of an embedded system that was responsible for monitoring the throttle in certain Toyota vehicles. In March 2014, Toyota agreed to pay a fine of \$1.2bn (USD) due to poor system design of the embedded system that monitored the accelerator pedal (The Washington Post).

The system featured sensor redundancy, and a failover system, but poor software implementation meant that it did not conform to industry standards. Example of software weaknesses is that single bit changes could result in catastrophic failure of a system, and WDT were poorly implemented meaning that it was slow or not able to react to fault scenarios (Barr Group, 2005).

### Therac-25

The Therac-25 was a computer controlled radio therapy machine that exhibited several errors which resulted in the loss of life. These scenarios are well document, and this paper shall not go in great depth.

One of the major flaws was the use of software-reuse and removal of redundancy of protective circuits. Software-reuse is the act of reusing software already created, which has the risk of introducing bugs and errors. This was the case with the Therac-25, where it inherited faults from

**Table 1.** Faults observed in a safety critical system and their attributed cause.

Fault	Attributed cause	Total
Incorrect function call	Software	41
Incorrectly read inputs	Software	18
Incorrect MCU initialisation	Software	5
Poor connector contact	Hardware	5
Battery (general)	Hardware	17
Incorrect system output	Hardware	5
Total faults		91

the previous machines. However, unlike the previous machines the Therac-25 did not feature independent protective circuits and mechanical interlocks. Instead, it relied on software to monitor and ensure correct operation. It was due to these independent protective circuits and mechanical interlocks that the software bugs were not noticed in earlier models (Leveson & Tumer, 1993).

### *Two-tier firewalls*

In IT, two different firewalls from different vendors can be cascaded with in intention of reducing the possibility of security breaches. By utilising different vendors, commonality between the two firewalls will be minimised, and hopefully the two systems should not suffer from the same technical flaw. However, it has been noted that the majority (99%) of firewall security breaches are due to incorrect configuration (Gartner, 2008). Although this is not an embedded systems problem, it illustrates that adding redundancy and complexity without correct implementation and assessment of the problem landscape, the overall system performance can be reduced.

### *Proprietary safety-critical device*

Analysis of a proprietary safety-critical device over its 15-year service life attributed 70.3% of the faults (64 units of 91 sample size) to software errors, the remainder (27 units of 91 sample size) were attributed to hardware faults which can be seen in Table 1.

## **Conclusion**

This paper has brought forward some design principles and has discussed cases where they are applicable. It can be seen in certain scenarios that designers employing principles to mitigate a fault-scenario may degrade system reliability without correct implementation of fundamental design requirements. Likewise, we have seen scenarios where the designer failed to employ the appropriate principles, leading to catastrophic failure. It is therefore important that at the beginning of the design stage that the required level of reliability be assessed and that the design methodology and practices reflect this. The design needs to consider both hardware and software complexity, and how they interact. Any use of COTS products needs careful consideration, as this brings uncertainty in the form of software-bugs, support uncertainty, and product life time. There are also commercial and business factors that need to be considered throughout the design as unnecessarily large or stringent requirements may increase development time and costs.



## Disclosure statement

No potential conflict of interest was reported by the authors.

## Funding

This work was supported by the Innovate UK Reference.

## ORCID

Steven Freear  <http://orcid.org/0000-0001-7858-4155>

## Reference

- Barr Group [Online]. (2005). *Bookout V. Toyota*. Retrieved from [http://www.safetyresearch.net/Library/BarrSlides\\_FINAL\\_SCRUBBED.pdf](http://www.safetyresearch.net/Library/BarrSlides_FINAL_SCRUBBED.pdf)
- Ciocirlan, B. O., & Herrmann, H. (2009). *Switching contact bounce reduction*. Proceedings of the 55th IEEE Holm conference on electrical contacts, 56–64.
- Crompton, T. R. (2000). *Battery reference book* (pp. 38/3–38/4). Oxford: Newnes.
- Demetzi, M., Annavaram, M., & Hall, M. (2011, November). Analyzing the effects of compiler optimization on application reliability. In *IEEE conference* (pp. 184–194). doi:10.1109/ISWC.2011.6114178
- Duracell Batteries [Online]. (2016, March 28). *Alkaline-magnese dioxide battery MX1500*. Retrieved from <http://professional.duracell.com/en/product-datasheets>
- Ebert, C. (2009). Jones, embedded software: Facts, figures, and future. *IEEE Computer Society*, April 43. doi: 10.1109/MC.2009.118
- El-Attar, A. M., & Fahmy, G.. (2007). *A study of fault coverage of standard and windowed watchdog timers*. IEEE International Conference on Signal Processing and Communications, pp 325–328.
- Fowler, K. (2009). Best practices in mission-assured, mission-critical, and safety-critical systems. In *Mission critical and safety-critical systems handbook* (pp. 3). Boston, MA: Elsevier/Newnes.
- Frate, F. D., Garg, P., Mathur, A. P., & Pasquini, A. (1995). On the correlation between code coverage and software reliability. In *Proceedings of the sixth international symposium on software reliability engineering* (pp. 124–132). doi: 10.1109/ISSRE.1995.497650
- Gartner. (2008). *Q&A: Is it more secure to use firewalls from two different vendors?* (pp. 3–4). Retrieved from <https://www.gartner.com/doc/1462923/qa-it-secure-use-firewalls>
- Greenliant. (2016, March 28). *Protecting against unintentional writes when using single power supply flash memory*. Retrieved from [www.greenliant.com/dotAsset/40844.pdf](http://www.greenliant.com/dotAsset/40844.pdf)
- Huy-Binh, L., Xuan-Dien, D., Lee, S.-G., & Ryu, S.-T. (2011). A long reset-time power-on reset circuit with brown-out detection capability. *IEEE Transactions On Circuits And System*, 58(11), 778–782.
- Ihrig, C. J. (2009). *Pro node.js for developers* (pp. 158). New York, NY: Apress.
- Infineon. (2016a). *EMC and system-ESD design guidelines for board layout AP24026*. Retrieved November 28, 2017, from <https://www.nxp.com/docs/en/application-note/AN11267.pdf>
- Infineon. (2016b, July 25). *Airbag systems*. Retrieved from <http://www.infineon.com/cms/en/applications/automotive/safety/airbag-restraint/>
- Lauesen, S., & Younessy, H. (1998). Is software quality visible in the code? *IEEE Software Magazine*, 69–73. doi:10.1109/52.687949
- Leveson, N. G., & Tumer, C. S. (1993, July). An investigation of the therac-25 Accidents. *IEEE Computer*, 26(7), 18–41. doi:10.1109/MC.1993.274940
- Little, A. D. (2015) [Online]. *Market and technology study automotive power electronics 2015* (pp. 97). Retrieved November 28, 2017, from [http://www.adlittle.nl/uploads/tx\\_extthoughtleadership/ADL\\_Study\\_Power\\_Electronics\\_2015.pdf](http://www.adlittle.nl/uploads/tx_extthoughtleadership/ADL_Study_Power_Electronics_2015.pdf)
- Louridas, P. (2006). Static code analysis. *IEEE Software*, 58–61. doi:10.1109/MS.2006.114
- Ganssle, Jack G. (2008, June). An RC Debouncer. In *A guide to debouncing* (pp. 12). Retrieved from <http://www.eng.utah.edu/~cs5780/debouncing.pdf>
- Mancini, R. (2002, June). *Op amps for everyone – Design guide* (pp. 17–27). Texas Instruments.
- Microchip. (2007). *PIC MCU power managed tips 'n Tricks*. (pp. 14), Retrieved November 28, 2017, from <http://ww1.microchip.com/downloads/en/DeviceDoc/41200a.pdf>
- Microchip. (2008). *Cyclic redundancy check (DS01148A)*, Retrieved November 28, 2017, from <http://ww1.microchip.com/downloads/en/AppNotes/01148a.pdf>

- S.McConnel. (2004). Controlling loops. In *Code complete: A practical handbook of software construction*. Section 20.5. 381. Redmond, WA: Microsoft Press.
- NASA. (1988). Reistor, fixed, film (High Stability). In *MIL-HDBK-978-B (NASA) NASA parts application handbook* (pp 3–28). Retrieved November 28, 2017, from [https://npp.nasa.gov/files/21529/MIL-HDBK-978B\\_vol1.pdf](https://npp.nasa.gov/files/21529/MIL-HDBK-978B_vol1.pdf)
- NASA Lesson Info. (2016a, August 3). *Space Data System (SDS) hardware design practice*. Retrieved from <http://llis.nasa.gov/lesson/731>
- NASA Lesson Info. (2016b, August 3). *Mars Global Surveyor (MGS) spacecraft loss of contact*. Retrieved from <http://llis.nasa.gov/lesson/1805>
- NASA Lesson Info. (2016c, August 3). *Active redundancy*. Retrieved from <http://llis.nasa.gov/lesson/697>.
- NASA Lesson Info. (2016d, August 3). *MSL Sol-200 Anomaly*. Retrieved from <http://llis.nasa.gov/lesson/11201>
- National Highway Traffic Safety Administration. (2011, January). *National highway traffic safety administration toyota unintended acceleration Investigation - Appendix A* (pp. 18). Retrieved November 28, 2017, from [https://www.nhtsa.gov/staticfiles/nvs/pdf/NASA\\_FR\\_Appendix\\_A\\_Software.pdf](https://www.nhtsa.gov/staticfiles/nvs/pdf/NASA_FR_Appendix_A_Software.pdf) November 28, 2017, (pp. 18).
- NXP [Online]. (2013). *EMC and system level ESD design guidelines for LCD drives*. Retrieved November 28, 2017, from <https://www.nxp.com/docs/en/application-note/AN11267.pdf>
- Park, C., Han, M., Lee, H., & Kim, S. W. (2014). Performance comparison of GCC and LLVM on the EISC processor. In *Electronics, Information and Communications (ICEIC), 2014 International Conference*. doi:10.1109/ELINFOCOM.2014.6914394
- Phoenix Contact. (2017, August 22). *Redundant power supply concepts*. Retrieved from [https://www.phoenixcontact.com/assets/downloads/./Redundancy\\_modules\\_en.pdf](https://www.phoenixcontact.com/assets/downloads/./Redundancy_modules_en.pdf)
- Plankis, B. J., Horning, M., Ponto, N., & Brown, L. K. (in press). Designing a dependable and fault-tolerant semiautonomous distributed control data collection network with opportunistic hierarchy. *IEEE Journal of Oceanic Engineering*, 32, 400–407. doi:10.1109/JOE.2006.880390
- RCA Service Company (Robert E. Barire, Wayne Hall). (1958). Relays. In *Electronic design's shock and vibration guide for airborne applications*. 75.
- Rivett, R. (2005). Automotive safety assurance: The role of legislation and standards. *The IEE Seminar on Safety Assurance*. doi:10.1049/ic:20050410.
- See, K. Y., Manish, O., Liu, Z. H., & Kyaw, K. L. (2004). *Correlation between ground bounce and radiated emission*. Electronics Packaging Technology Conference, EPTC 2004. Proceedings of 6th, pp 640–642.
- Segawa, L., Shirota, Y., Fujisaki, K., Kimura, T., & Kanai, T. (2014). Aggressive use of deep sleep mode in low power embedded systems. *IEEE COOL Chips XVII*, 1–3. doi:10.1109/CoolChips.2014.6842956
- Shar, L. K. (2012). Hee Beng Kuan Tan, "Defeating SQL injection. *IEEE Computer Society*, 46(3), 69–77. doi:10.1109/MC.2012.283
- Spear, A., Levy, M., & Desnoyers, M. (2012). Using tracing to solve the multicore system debug problem. *IEEE Computer Society*, 45(12), 60–64. doi:10.1109/MC.2012.191
- Steinberg & Associates (Dave S. Steinberg). (2017, August 22). *Designing electronics for high vibration and shock*. Retrieved from <edge.rit.edu/edge/P10662/public/old/Mechanical/DesigningElectronics.doc>
- Sun, J., Gong, W., Dong, X., Zhang, X., & Wang, Y. (2014). *High availability analysis and evaluation of heterogeneous dual computer fault-tolerant system*. *IEEE conference* (pp. 460–464).
- Swingler, J., & McBride, J. W. (1998). *The synergistic relationship of stresses in the automotive connector*. Proc. 19th Int. Conf. Electric Contact Phenom., Nuremberg, Germany, pp. 141–145.
- Tadiran Batteries. (2016, March 28). *LTC Batteries, SL-2780 Size: D,iXtra*. Retrieved from <http://www.tadiranbatteries.de/pdf/lithium-thionyl-chloride-batteries/SL-2780.pdf>
- Texas Instruments. (1999). *PCB design guidelines for reduced EMI*. Retrieved November 28, 2017, from <http://www.ti.com/lit/an/szza009/szza009.pdf>
- Texas Instruments. (2014, November). *Calculating Useful Lifetimes of Embedded Processors*. Retrieved November 28, 2017, from <http://www.ti.com/lit/an/sprabx4a/sprabx4a.pdf> pp. 3.
- Wallinder, I. O., & Eriksson, P. (1998). *Characterization of the corrosivity of an automotive environment*. Proc. 19th Int. Conf. Electric Contact Phenom, Nuremberg, Germany, pp. 157–161.
- The Washington Post. (2014). *Toyota reaches \$1.2 billion settlement to end probe of accelerator problems*. Retrieved November 28, 2017, from [https://www.washingtonpost.com/business/economy/toyota-reaches-12-billion-settlement-to-end-criminal-probe/2014/03/19/5738a3c4-af69-11e3-9627-c65021d6d572\\_story.html](https://www.washingtonpost.com/business/economy/toyota-reaches-12-billion-settlement-to-end-criminal-probe/2014/03/19/5738a3c4-af69-11e3-9627-c65021d6d572_story.html)
- Zhai, G., Yang, W., & Zhou, X. (2008). Study on Conducted EMI from operation of electromagnetic relay contacts. *International conference on electrical machines and systems*, Vols. 543-547. New Jersey: IEEE.
- Zorn, C., & Kaminski, N. (2014, February). *Temperature humidity bias (THB) testing on IGBT modules at high bias levels*. Integrated Power Systems (CIPS), 2014 8th International Conference.