ECEN 5803

Mastering Embedded Systems Architecture

University of Colorado **Boulder**

What is the next great enabling technology?

1. ~~Virtual~~ Augmented Reality
2. Self-driving Cars
3. Nanobots/Robots
4. Quantum Computers
5. Fuel Cells

What is the next great enabling technology?

A Augmented Reality

B Self-driving Cars

C Nanobots/Robots

D Quantum Computers

E Fuel Cells

Automotive Engine Control Case Study

Electronics has been a boon to the Automotive industry, greatly improving safety and performance. The introduction of engine control modules (ECMs) improved the performance of automobile engines and decreased pollutant emissions. Engine tune-ups have virtually disappeared.

In the future, the use of electronics and **embedded systems** will be critical to achieve **the next great revolution** in transportation: **the autonomous car.** Self-driving cars are no longer a rare sight on Californian roads. Over 100 autonomous vehicles from a dozen manufacturers are now being tested in public, covering hundreds of thousands of kilometres each year.

In 2016, some car firms, including Nissan, Ford, Kia and Tesla, think self-driving technology will be ready by 2020. However, this hasn't quite happened. Now predictions are 2030 or later.

Kim Fowler, "Automobile Engine Controller", in *What Every Engineer Should Know About Real-Time Embedded Products*, Boca Raton, FL. CRC Press, 2012, pp. 53-94.

Automotive Engine Control Case Study

Process

Designing and manufacturing ECMs is a mission-critical process. While each company has its own process, most rely on a waterfall or V-model process. Most do not do spiral development because the ECM design generally must be correct when it goes to commercial production.

Kim Fowler, "Automobile Engine Controller", in *What Every Engineer Should Know About Real-Time Embedded Products*, Boca Raton, FL. CRC Press, 2012, pp. 53-94.

Automotive Engine Control Case Study

Architecture

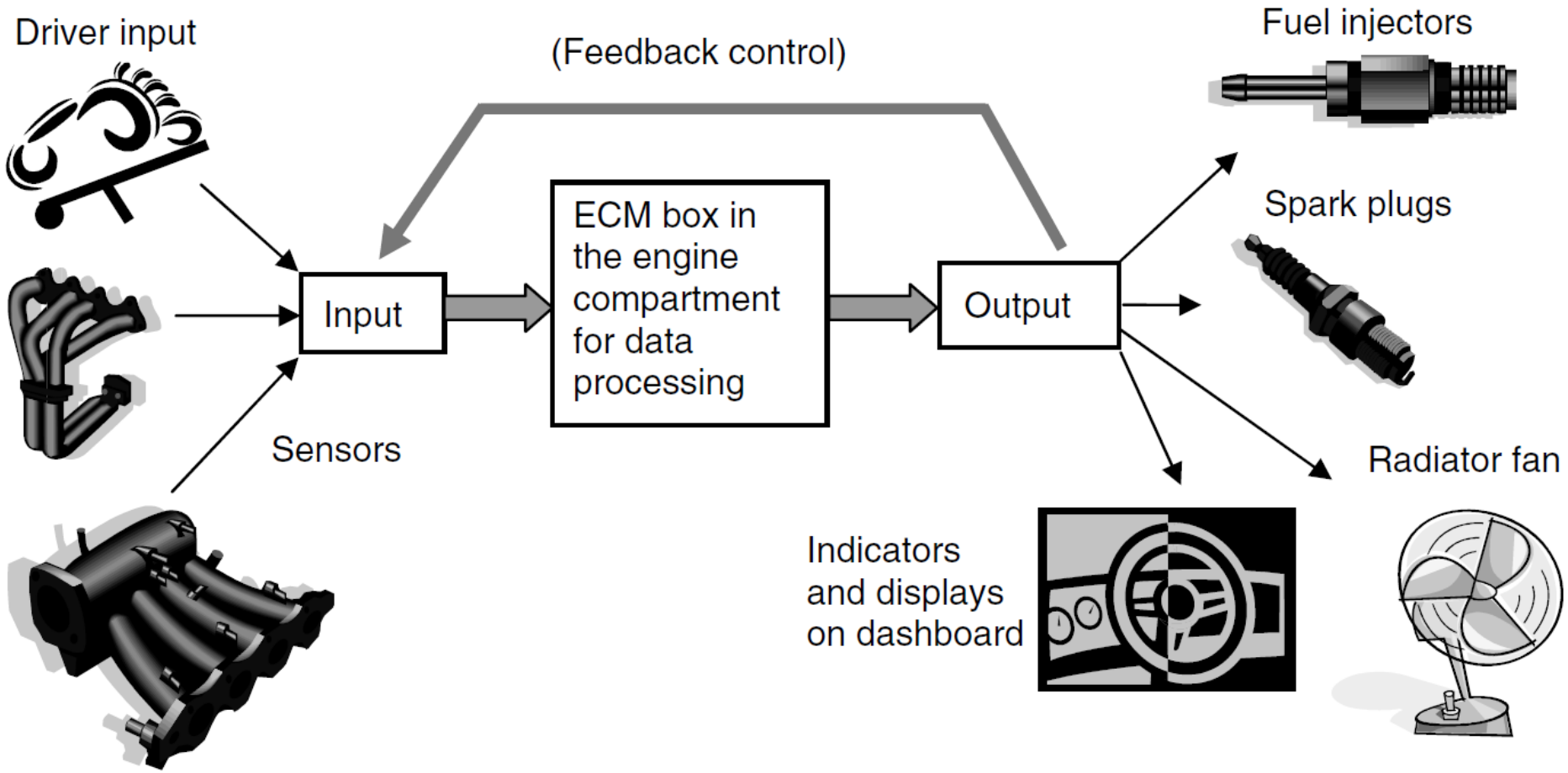
ECMs have circuits based on highly integrated microcontrollers that contain a number of peripherals, such as timing processors and analog converters.

Networking is done internally with CAN and LIN networks (automobiles typically have around 20 MCUs, sometimes as many as 60). Configurable hardware (e.g., FPGAs) is beginning to play a role in developing ECMs, while application-specific integrated-circuits (ASICs) have been used for nearly two decades.

Kim Fowler, "Automobile Engine Controller", in *What Every Engineer Should Know About Real-Time Embedded Products*, Boca Raton, FL. CRC Press, 2012, pp. 53-94.

Automotive Engine Control Case Study

Architecture



Kim Fowler, "Automobile Engine Controller", in *What Every Engineer Should Know About Real-Time Embedded Products*, Boca Raton, FL. CRC Press, 2012, pp. 53-94.

Automotive Engine Control Case Study

Architecture – Microcontroller Selection

An eight-cylinder automobile engine might allow a maximum of 8000 revolutions per minute (RPM). One spark plug out of four fires each revolution of the crankshaft. This means that the engine has two ignition sparks per revolution; at 8000 RPM this is 133 revolutions per second or 267 sparks per second. The maximum time the ECM has to process between sparks is the reciprocal of this or 3.75 msec.

The timing resolution on the sparks is even more stringent. The ECM must time and initiate the spark and control it to within 0.4° of the crankshaft angle, which resolves to 900 positions per revolution. At 133 revolutions per second, the maximum time is 7.50 msec. The time resolution = (minimum time to compute/revolution)/(positions resolved/revolution) = is (7.50 msec)/(900) = 8.33 usec/position. This type of high resolution in timing needs a dedicated peripheral (TPU) to do hardware timing on the microcontroller chip.

Kim Fowler, "Automobile Engine Controller", in *What Every Engineer Should Know About Real-Time Embedded Products*, Boca Raton, FL. CRC Press, 2012, pp. 53-94.

Automotive Engine Control Case Study

Architecture – Microcontroller Selection

Many ECMs use a 32-bit processor or microcontroller. A larger data path can provide for more sophisticated algorithms than smaller, simpler, and cheaper microcontrollers. As mentioned, the firmware in the ECM must perform many functions including high-speed calculations. One way to speed up operations and decrease the time of calculations is to use large look-up tables (LUTs); the trade-off is speed, gained by a large LUT for memory size, which increases cost and decreases reliability.

Within the ECM box, the hardware converts the electrical analog signal to a digital data value. Most inputs use the internal analog-to-digital converters (ADCs) built into the microcontroller. These signals have resolutions of 10 or 12 bits. Some signals, such as pressure, require fairly high resolution, which means that separate 16-bit or 18-bit or even 20-bit converters are sometimes used.

Kim Fowler, "Automobile Engine Controller", in *What Every Engineer Should Know About Real-Time Embedded Products*, Boca Raton, FL. CRC Press, 2012, pp. 53-94.

Automotive Engine Control Case Study

Architecture – Microcontroller Selection – Autonomous Car

But this car is different: its human driver keeps his hands firmly on the wheel. The vehicle, nicknamed “George” by HERE, a Berlin-based mapping company owned by BMW, Audi and Daimler, is not driving itself but collecting data that enable other cars to do so. For every second of its journey, a high-precision GPS receiver on George’s roof collects the car’s latitude, longitude and elevation ten times over; a motion-tracking inertial system records its yaw, pitch and roll 100 times; and the laser scanner calculates its distance from some 600,000 different points, such as trees, curbs and buildings. At the same time, four cameras also shoot a 96-megapixel, 360-degree panoramic image for every 6 metres the vehicle moves along the road.

A day’s driving can accumulate 100 gigabytes or more of data.

Automotive Engine Control Case Study

Design Cycle

The design cycle time for ECMs is 2 or 3 years, depending on the company, the team, and the particular ECM. After production release, only firmware changes to the ECM are allowed for the first 2 or 3 years of production. Three to six years following production, the design team might modify the hardware in the ECM, if needed.
Like all mission-critical project developments, full documentation, from contract and requirements to final test results and delivery transfer, is necessary to fulfil regulatory concerns. For instance, all source code must be archived and stored for 10 years after the last production unit rolls off the assembly line.

Kim Fowler, "Automobile Engine Controller", in *What Every Engineer Should Know About Real-Time Embedded Products*, Boca Raton, FL. CRC Press, 2012, pp. 53-94.

Automotive Engine Control Case Study

Design Tradeoffs

Most OEMs plan and design an automobile to last about 10 years. Automobile ECM designers are more worried about the number of ignition cycles than miles driven; a cycle bounds the period when a car turns on and to when it turns off. It is defined by the ignition switch, not by the engine turning. Many designers plan ECMs to operate for 100,000 to 200,000 cycles. The problem here is that the EEPROM inside the ECM is written on power-down and EEPROMs have a limited number of write cycles before they no longer function properly.

Kim Fowler, "Automobile Engine Controller", in *What Every Engineer Should Know About Real-Time Embedded Products*, Boca Raton, FL. CRC Press, 2012, pp. 53-94.

Automotive Engine Control Case Study

Design Tradeoffs

Most software is written in C with some assembly routines for critical timing concerns, such as the bootstrap code and the low-level drivers. The selection and application of a real-time operating system (RTOS) varies widely for the 32-bit controllers within the ECMs. These vary from simple schedulers and round-robin kernels to pre-emptive and co-operative RTOSes. Some ECMs have a custom RTOS, while others use a commercial-off-the-shelf (COTS) package. **QNX** has taken the market share lead for Automotive RTOS.

The trade-off between hardware and software balances performance, cost, reliability, and the ability to upgrade. More specifically this balance reduces to speed vs. ROM size vs. the cost of components. No surprise to a lot of you software developers, software sometimes makes up for the sins of the hardware design. Designers of ECMs often leave spare I/O pins for future use to fix things that inevitably had flawed specifications or design. While cost is still prohibitive, suppliers of ECMs might be moving toward using field-programmable gate arrays (FPGAs). An FPGA can blend together the advantages of both software and hardware—flexibility and speed.

Kim Fowler, "Automobile Engine Controller", in *What Every Engineer Should Know About Real-Time Embedded Products*, Boca Raton, FL. CRC Press, 2012, pp. 53-94.

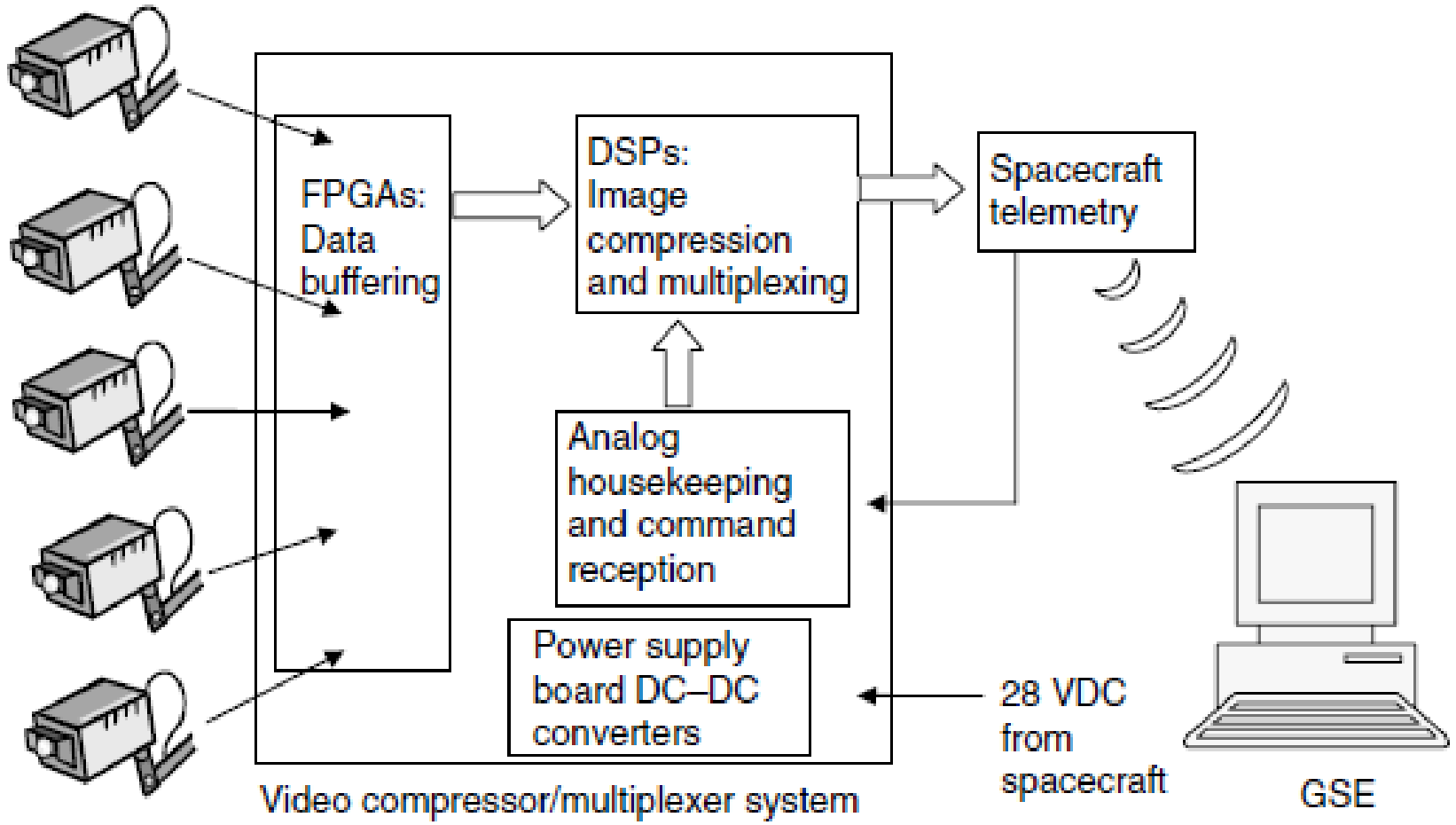
Automotive Engine Control Case Study

Buy vs. Build

Historically, buy vs. build has been a cyclical affair for ECMs. Car companies began developing ECMs in-house but found they did not always have enough resources to do the job properly. Then they contracted out the ECM design to vendors, but found that they were not always fast enough to meet their demand. Now that ECMs are so critical to the operation of automobile engines, more automobile companies are bringing or keeping ECM design in-house to maintain the design resources and capability. Companies have also worked to develop career paths for embedded engineers, making long-term employment more attractive to firmware engineers.

Kim Fowler, "Automobile Engine Controller", in *What Every Engineer Should Know About Real-Time Embedded Products*, Boca Raton, FL. CRC Press, 2012, pp. 53-94.

Aviation Video Acquisition Case Study



Kim Fowler, "Aerospace Video Processor", in *What Every Engineer Should Know About Real-Time Embedded Products*, Boca Raton, FL. CRC Press, 2012, pp. 263-277.

Aviation Video Acquisition Case Study

The flight hardware has four circuit boards: a video-compression board, a multiplexer board, a power supply board, and an analog housekeeping board. A field-programmable gate array (FPGA) on the multiplexer board buffers up the data streams from each sensor. Digital signal processor (DSP) chips on both the compression board and the multiplexer board multiplex data and to compress the images. The analog housekeeping collects temperature data and analog signals from various places on the sensors, converts it to digital format, and then multiplexes the housekeeping data into the data stream for telemetry.

Kim Fowler, "Aerospace Video Processor", in *What Every Engineer Should Know About Real-Time Embedded Products*, Boca Raton, FL. CRC Press, 2012, pp. 263-277.

Aviation Video Acquisition Case Study

Process

In the beginning, while trying to use the legacy code, they did not follow defined processes. Partway through, they had no development plans of any kind: design, schedule, or test; they had no code reviews to assure quality and functionality; a single code developer, with no accountability, was working with the legacy code.

The software was a mess! It was what some people call "spaghetti code." The style guide had not been followed; the software it had no consistent format. The software did not use a common set of event handlers; each sensor was handled differently, even though the sensors were very similar in data output. Orphan code segments floated throughout the software. The whole program was a set of nested interrupts—a major no-no!

Kim Fowler, "Aerospace Video Processor", in *What Every Engineer Should Know About Real-Time Embedded Products*, Boca Raton, FL. CRC Press, 2012, pp. 263-277.

Aviation Video Acquisition Case Study

Process

They recognized the problems and completely changed things around. They defined and instituted new software processes that required code reviews and metrics for anomalies and production. They held monthly reviews of their progress with the customer. They staged releases of the software (using a spiral development model). They also prepared a complete set of documents to cover the products design, development, and test.

The new processes included record keeping, metrics, and production guidelines. Records of production metrics are important to quantify aspects of quality; they include bug rates, such as lines of code per hour or per day (LOC/h or LOC/day), bugs found, bug severity, and status of fixes for bugs.

Kim Fowler, "Aerospace Video Processor", in *What Every Engineer Should Know About Real-Time Embedded Products*, Boca Raton, FL. CRC Press, 2012, pp. 263-277.

Aviation Video Acquisition Case Study

Architecture

The video compressor/multiplexer has a centralized architecture to save weight and power (Figure 11.1). The mechanical configuration has circuit boards that plug into a backplane. The sensors feed raw video data to the video compressor/multiplexer.

Kim Fowler, "Aerospace Video Processor", in *What Every Engineer Should Know About Real-Time Embedded Products*, Boca Raton, FL. CRC Press, 2012, pp. 263-277.

Aviation Video Acquisition Case Study

Design Tradeoffs

The main set of decisions for the electronic hardware centered on the number of FPGAs vs. DSPs and on the number of circuit boards in the enclosure. To keep weight and size down, the chassis could hold no more than four circuit boards. This drove the design to use a combination of FPGAs and DSPs for data manipulation, compression, and multiplexing.

They also upgraded a small DSP to a more recent model to handle the housekeeping and receiving the commands from the spacecraft. Ecliptic relied on legacy software and a former board design—in hope of reducing the effort through reuse—which forced some of the data to circulate between several different boards.

What DSP did they use?

ADSP-2191, a 16-bit fixed point 160 MIPS DSP

Kim Fowler, "Aerospace Video Processor", in *What Every Engineer Should Know About Real-Time Embedded Products*, Boca Raton, FL. CRC Press, 2012, pp. 263-277.

Aviation Video Acquisition Case Study

Architecture – Processor Selection

What motivated the use of the DSP selected?

Desire to reuse Legacy code.

Why use a DSP instead of a general purpose processor?

More power efficient and higher performance when implementing signal processing algorithms at the same cost as a GP MCU.

Kim Fowler, "Aerospace Video Processor", in *What Every Engineer Should Know About Real-Time Embedded Products*, Boca Raton, FL. CRC Press, 2012, pp. 53-94.

Aviation Video Acquisition Case Study

Design Tradeoffs - Software

The software development was the most difficult part of this project. It involved between 15,000 and 18,000 lines of code (LOC). With the 70- and 80-hour weeks put in over 6 months, the production rate averaged 100 LOC/day or about 10 LOC/hour. This rate of production for the software is unusually high when compared with most teams developing code for mission-critical instruments. (The principal developer was extremely skilled and highly motivated.) Typical production rates for most companies range from 5 to 25 LOC/day or between 1 and 3 LOC/hour.

Kim Fowler, "Aerospace Video Processor", in *What Every Engineer Should Know About Real-Time Embedded Products*, Boca Raton, FL. CRC Press, 2012, pp. 263-277.

Is LOC/day a good Software metric?

A. Yes
B. No

Aviation Video Acquisition Case Study

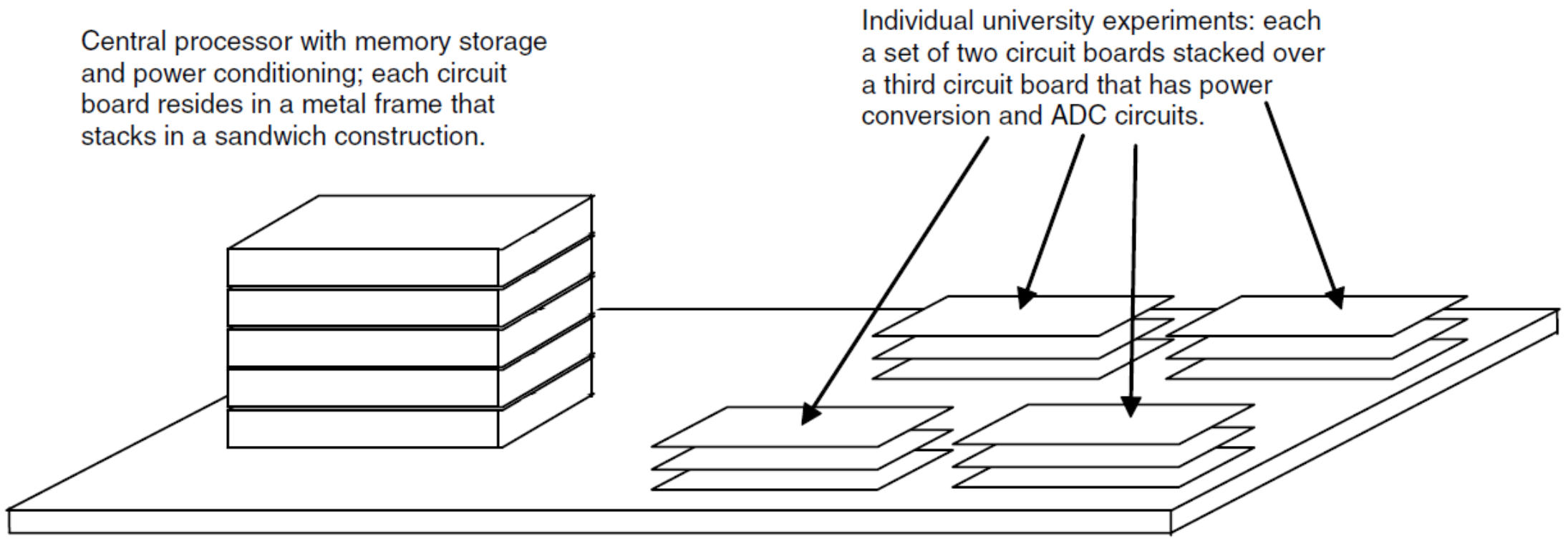
Buy vs. Build

Originally, Ecliptic had planned to write custom software to run the ground support equipment (GSE). Because of devoting all their efforts to the software in the video compressor/multiplexer, they ran out of time to write a custom software package. Instead, they contracted with a developer in a small company to modify his product to run on their GSE.

They made extensive use of COTS power supplies in the power supply board design.

Kim Fowler, "Aerospace Video Processor", in *What Every Engineer Should Know About Real-Time Embedded Products*, Boca Raton, FL. CRC Press, 2012, pp. 263-277.

Satellite Experiment Platform Case Study



Kim Fowler, "Satellite Subsystem", in *What Every Engineer Should Know About Developing Real-Time Embedded Products*, Boca Raton, FL. CRC Press, 2012, pp. 279-301.

Satellite Experiment Platform Case Study

Process

A V-model project plan was used with these phases:

Concept—complete all specifications for features and development timeline, define the architecture, and select basic components (duration *10 months)

Preliminary design—finish breadboarding all prototype circuits and modules, define all the fabrication processes, and have all design processes running with initial drafts of all documents (duration *10 months)

Critical design—prepare engineering models, begin system tests, begin fabrication of flight components and circuit boards, and have an initial GSE system working (duration *12 months)

Fabrication and integration—complete the fabrication, assembly, inspection of all modules and the SET carrier platform, perform all environmental tests, and integrate the SET carrier on the launch vehicle (duration *14 months)

Launch and mission (duration *6 months)

Kim Fowler, "Satellite Subsystem", in *What Every Engineer Should Know About Developing Real-Time Embedded Products*, Boca Raton, FL. CRC Press, 2012, pp. 279-301.

Satellite Experiment Platform Case Study

Architecture

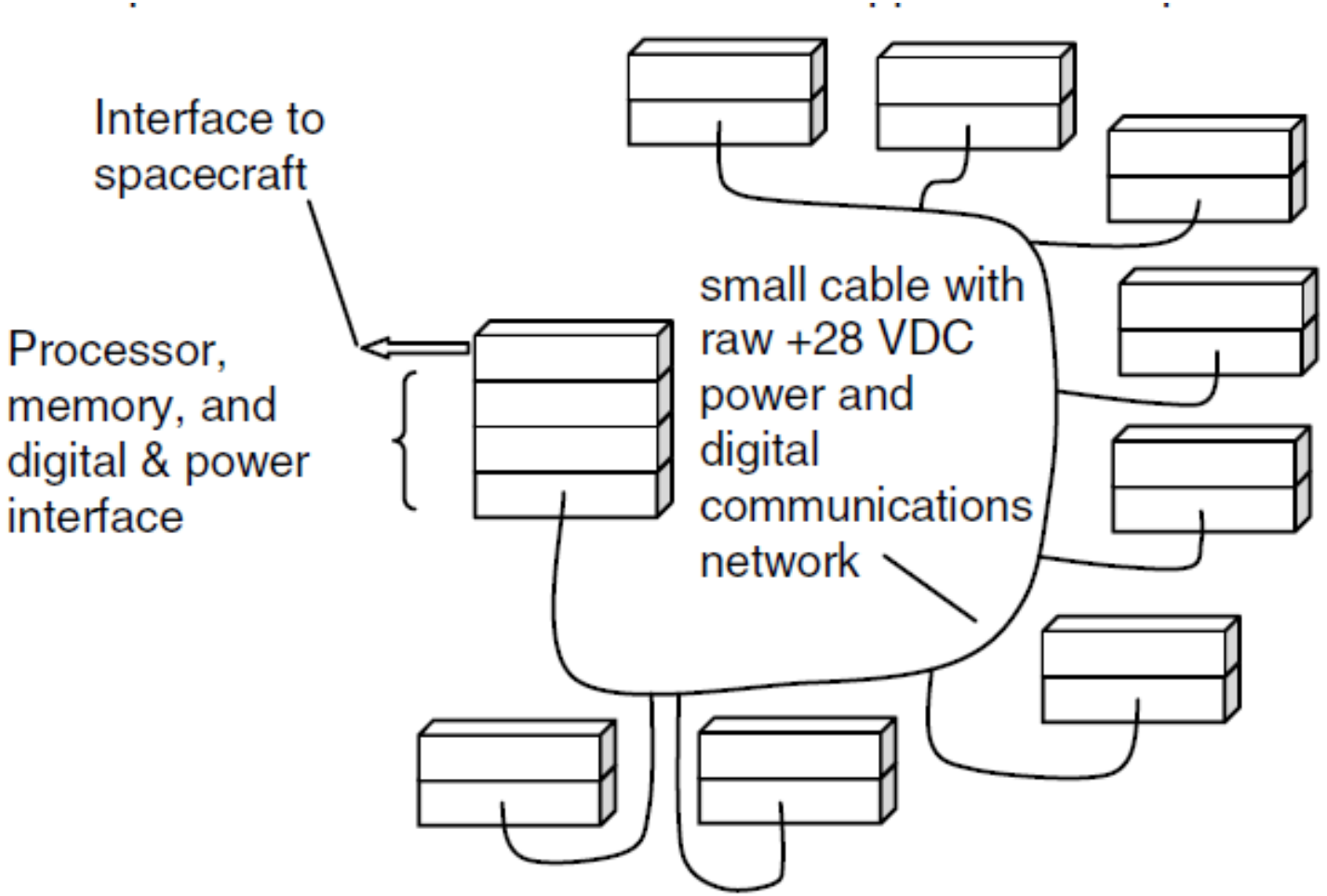
We examined and compared the following concerns between the different approaches:

- Design complexity
- Integration risks
- Fault isolation
- Experiment flexibility
- Mission adaptability

The distributed approaches (one is illustrated in Figure 12.1) were better than the centralized "star" configuration.

n Fowler, "Satellite Subsystem", in *What Every Engineer Should Know About Developing Real-Time Embedded Products*, Boca Raton, FL. CRC Press, 2012, pp. 279-301.

Satellite Experiment Platform Case Study



Kim Fowler, "Satellite Subsystem", in *What Every Engineer Should Know About Developing Real-Time Embedded Products*, Boca Raton, FL. CRC Press, 2012, pp. 279-301.

Satellite Experiment Platform Case Study

Architecture – Processor Selection

They studied a number of different processors for the SET carrier. Besides radiation hardness and low power consumption— which were required of any IC considered—we focused on ease of use and corporate experience with their development tools. We also considered computational power. See Table 12.3 for the details of our trade-offs.

They selected the UT80CRH196KDS from United Technologies, a radiationhard version of the Intel 80196 microprocessor, because it had the necessary qualities we desired. JHU/APL had corporate knowledge working with its software development tools and its price and availability were acceptable.

Cost of a RadHard processor?

$300,000

Kim Fowler, "Automobile Engine Controller", in *What Every Engineer Should Know About Real-Time Embedded Products*, Boca Raton, FL. CRC Press, 2012, pp. 53-94.

Satellite Experiment Platform Case Study

Design Tradeoffs

We planned to write the source code in C, hold regular code reviews, and have defined, rigorous tests to verify functionality. The software design needed to be modular, and we planned to reuse code to support multiple different missions.

We believed a real-time operating system (RTOS) was necessary to aid development of the software and make the system more adaptable for different missions. A commercial RTOS seemed to make the most sense—shorter time to install and run, vendor technical support, and ease of software development. The choices quickly narrowed down because most commercial RTOSs had either too many features or architectures that required non-radiation-hard components in the interface; memory also constrained the size of the RTOS.

Kim Fowler, "Automobile Engine Controller", in *What Every Engineer Should Know About Real-Time Embedded Products*, Boca Raton, FL. CRC Press, 2012, pp. 53-94.

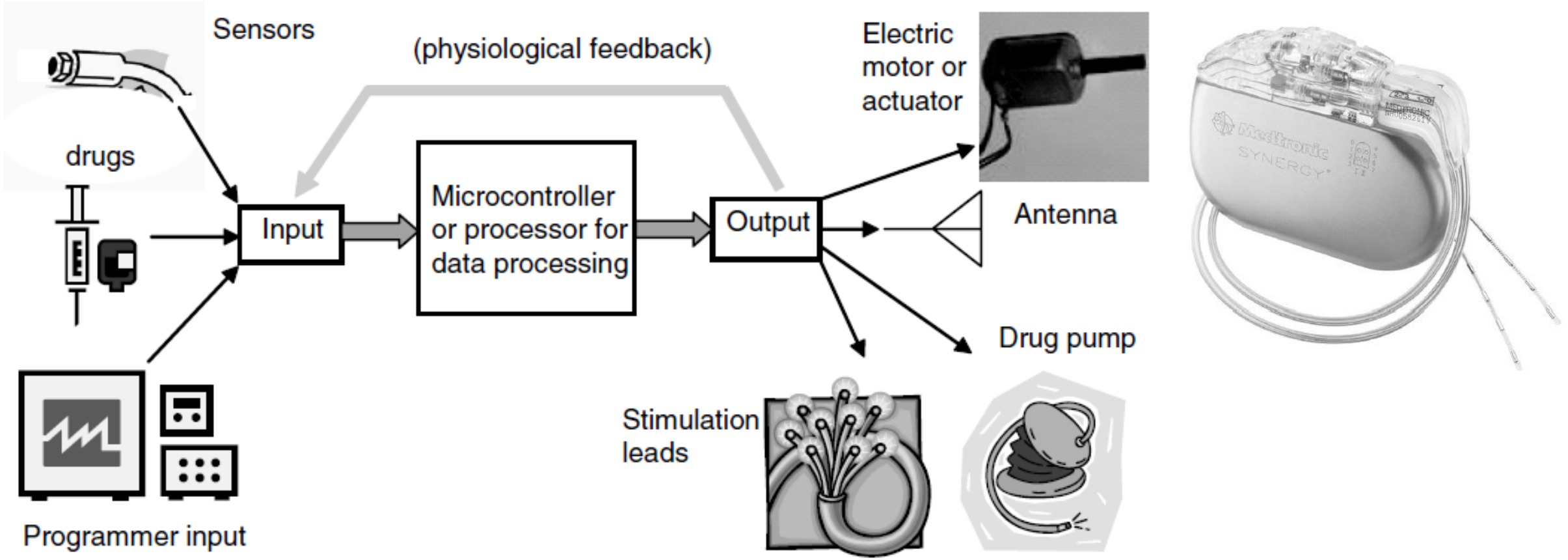
Satellite Experiment Platform Case Study

Buy vs. Build

As is true for most spacecraft, most of the SET carrier was going to be custom; the requirements for radiation hardness, low weight, and low power consumption all drive a custom design. We had planned for COTS RTOS software, GSE, and GSE software.

Kim Fowler, "Automobile Engine Controller", in *What Every Engineer Should Know About Real-Time Embedded Products*, Boca Raton, FL. CRC Press, 2012, pp. 53-94.

Implanted Medical Devices



Kim Fowler, "Implanted Medical Devices", in *What Every Engineer Should Know About Real-Time Embedded Products*, Boca Raton, FL. CRC Press, 2012, pp. 333-375.

Implanted Medical Devices

Process

A medical device is safety-critical; its development must reflect that reality. The V-model process is most typical in developing an implanted medical device.

Most medical devices cannot be reprogrammed once released; such would be contrary to FDA approval, so spiral development is ruled out beyond the design phase.

Software is a particular concern. It needs to be right, and it helps to be fault-tolerant too. This forces you to perform regular, in-depth code reviews in the early phases before clinical testing.

All requirements must be tested and the results approved as satisfactory before the product goes into clinical testing.

Kim Fowler, "Implanted Medical Devices", in *What Every Engineer Should Know About Real-Time Embedded Products*, Boca Raton, FL. CRC Press, 2012, pp. 333-375.

Implanted Medical Devices

Process – Phases

Phase 1: Concept
Vision
Feasibility and tradeoff analyses
Risk plan
Risk assessment: ETA, FTA, FMEA
Business risk

Phase 2: Planning and Scheduling
Product description
Feasibility and tradeoff analyses
Risk plan
Risk assessment: ETA, FTA, FMEA
Business risk

Phase 3: Design and development
Product description
Feasibility and tradeoff anal
Design transfer plan
Clinical plan
Code standards
Test results
Recorded errors
Test metrics
Traceability
Design documents
Software design document
Source listings
Hardware design document
System design document
Final parts list

Phase 4: Controlled Releas
Product description
Clinical results
FDA submission
Submissions for UL, CE, IEC certifications
Training plan

Phase 5: Commercial relea
Product description
Design history file
Device master record
Version description document
Publications
Brochures
Training materials
FDA approval
UL, CE, IEC certifications
Training plan

Kim Fowler, "Implanted Medical Devices", in *What Every Engineer Should Know About Real-Time Embedded Products*, Boca Raton, FL. CRC Press, 2012, pp. 333-375.

Implanted Medical Devices

Architecture – Processor Selection

Probably the biggest concern for implanted electronics is power consumption. I know of at least one manufacturer who designs and fabricates their own custom ICs, particularly small microcontrollers for implanted medical devices. They want sufficiently low power consumption so that the implanted device will function for years before draining the battery.

Another major concern is fault-tolerant operation. Should anomalous behavior occur, whether externally generated or due to internal failure, the device should operate or shut down in a safe way. One example of this need is the communications link—it should be robust. I know of another situation where a particular model of device behaved erratically when patients walked between the security monitors in a particular department store.

Kim Fowler, "Implanted Medical Devices", in *What Every Engineer Should Know About Real-Time Embedded Products*, Boca Raton, FL. CRC Press, 2012, pp. 333-375.

Implanted Medical Devices

Architecture – Processor Selection

Software in an implantable medical device is safety-critical development if there is any! Development processes need to include thorough studies that support detailed and complete requirements, careful design, properly conducted code reviews, tests, and field tests.

Some companies build their own custom, time-slice real-time operating systems (RTOSs) that use a cyclic executive; these are straightforward to analyze [11]. A time-slice RTOS can guarantee that no tasks are starved or left undone; the trade-off for that guarantee is efficiency. A certified compiler is a worthwhile investment for medical-software development. If your company fabricates its own processors, then your team or someone in your company will have to write and test a compiler and linker. If you decide to purchase the RTOS, there are several companies that provide certified software products for safety-critical markets.

Implanted Medical Devices Case Study

Design Tradeoffs

As complexity of medical devices increases more functions will implement in software. Hardware accelerates specialty functions. The balance is somewhat dictated by how thoroughly the software modules and hardware subsystems can be verified.

One aspect of fault tolerance is use of a hardware or mechanical function as a safety check or limit for the software operation. Conversely, software can check and verify hardware function. An example might be a restriction in the delivery tube that will not allow fast delivery of drug. Another might be a hardware counter that independently prevents an overdose of operations when it times out. The watchdog timer is a simple example of a hardware check on software operation. Another example might be a thermal sense switch on the battery that opens should temperature exceed a maximum limit and shuts down the device to prevent a failure from overheating.

Implanted Medical Devices Case Study

Buy vs. Build

Most subsystems within an implantable medical device are custom designed and built. The basic components may be purchased, although in some cases, even the processors can be custom application-specific integrated circuits (ASICs). The concerns for specificity, reliability, longevity, obsolescence, and inventory in commercial off-the-shelf (COTS) products all disappear with custom designs. The trade-off is that cost, particularly due to non-recurring engineering (NRE), increases greatly.

Kim Fowler, "Implanted Medical Devices", in *What Every Engineer Should Know About Real-Time Embedded Products*, Boca Raton, FL. CRC Press, 2012, pp. 333-375.