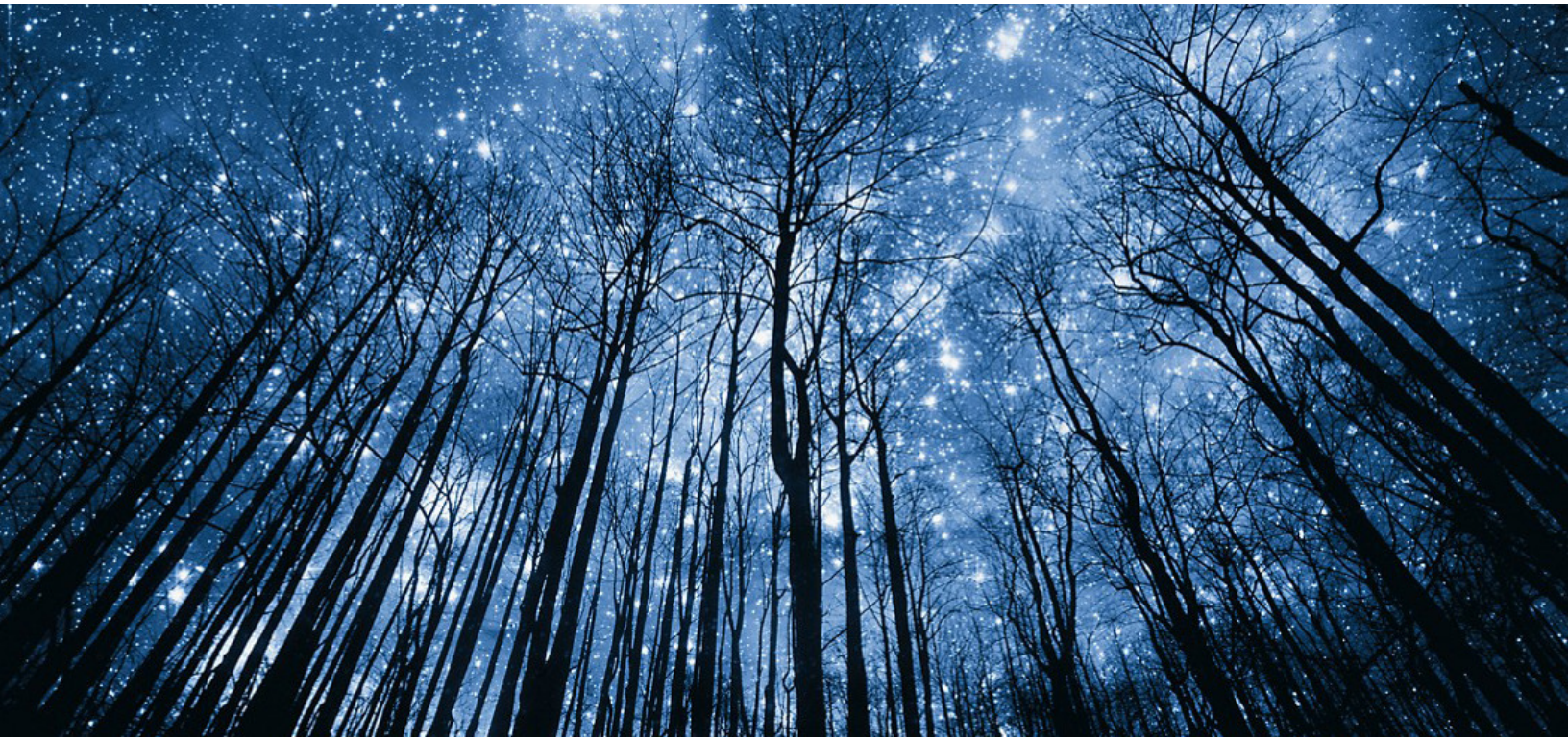
DELL Technologies

THE HIDDEN WORLD OF EMBEDDED SYSTEMS



Bruce Yellin
Retired Advisory Engineer
Bruceyellin@yahoo.com

Proven Professional

Knowledge Sharing Article

The Dell Technologies Proven Professional Certification program validates a wide range of skills and competencies across multiple technologies and products.

From Associate, entry-level courses to Expert-level, experience-based exams, all professionals in or looking to begin a career in IT benefit from industry-leading training and certification paths from one of the world's most trusted technology partners.

Proven Professional certifications include:

- Cloud
- Converged and Hyperconverged Infrastructure
- Data Protection
- Data Science
- Networking
- Security
- Servers
- Storage
- Enterprise Architect

Courses are offered to meet different learning styles and schedules, including self-paced On Demand, remote-based Virtual Instructor-Led and in-person Classrooms.

Whether you are an experienced IT professional or just getting started, Dell Technologies Proven Professional certifications are designed to clearly signal proficiency to colleagues and employers.

Learn more at www.dell.com/certification

Contents

.

We have far more embedded systems in our lives than general-purpose systems, and it is not even close. Even if we count smartphones and tablets as computers, and we should, our lives are surrounded by brainy devices that revolutionize the way we prepare food, clean our houses, and treat human conditions. Our world is increasingly digital with new connected devices marketed every year. Whether at work or home, embedded systems are an extension of us, yet they are often overlooked.

Many Computer Science (CS) curriculum and the media tend to focus on general-purpose computing and information processing and pay less attention to embedded systems. A PC was designed to use hardware and software and let the user decide what it should do. Embedded systems also have hardware and software, but they were designed for a specific function by the manufacturer, operate 24/7, last over a decade, and function without a keyboard or mouse.

Manufacturers introduce new features and revise existing ones when designing new vehicles, appliances, and other products, all to improve the user experience. Embedded systems are in many of the goods that we take for granted and even depend upon, and without them, our lives would be vastly different. They appear in a multitude of battery and plug-in devices in our homes, offices, transportation, medical facilities, stores, and more.

Working with sensors, actuators, intelligent software, and micro-miniature circuitry, inexpensive and low-power devices perform necessary specific tasks our PCs are ill-suited for, such as an ingenious home cleaning robot or an amazing pod coffee maker. Even though embedded and general-purpose computing has a lot in common, and some general-purpose systems contain embedded systems (for example, the lithium-ion battery management system in your PC), embedded software is a radical departure from an accounting or office productivity program.

Embedded systems account for nearly 99% of all the microprocessors in use. Conversely, only 1% of them are found in PCs. If you add up all the processors you ever used in a server, PC, or tablet, it would be a fraction of the 50 to 150 in your house today. Think about the number of electronic control units (ECUs) in your car, one of which senses an approaching object and applies the brakes or prevents you from veering into oncoming traffic.

In 2019, the global embedded processor market was valued at $19B and should reach $44B by 2030.[1,2] The automotive market is the largest segment through 2028, followed by consumer electronics, healthcare, IT, and telecommunications. Meanwhile, the automotive embedded software market alone reached $22B in 2022 and is forecasted to exceed $40B by 2027.[3]

This article introduces you to the exciting world of embedded systems through interesting comparisons with general-purpose systems and examples of their use in our everyday lives. It illustrates both their simplicity and complexity, how they function without our intervention and shows their broad appeal to just about every industry. It also gives you insight into becoming more involved with them and even how to start an embedded systems career.

Introduction

Ask a friend a computer question, and the answer is usually in terms of a general-purpose Windows PC or Mac, or something of that ilk. Embedded systems, which contain embedded processors, are “cousins” of general-purpose machines. Like the iceberg analogy where the bulk of it is underwater, the same is true of embedded systems. Compared to a PC, the number of chips in use in our embedded devices is staggering and reveals an almost stealth industry.

If you look at worldwide units shipped each year, you get a different view of embedded systems. The industry sold 349 million PCs in 2021 and **31 billion microprocessors**, meaning 98.9% went into non-PC devices.[4,5] With a decade-plus lifespan, the cumulative number of embedded processors worldwide is staggering. With 122 million households in just the US, if we assume that each household had 75 embedded devices that equals 9 billion microprocessors in just one country.[6] The number is higher when you add in the number of businesses, vehicles, and other devices with multiple embedded processors.

Embedded systems are one of the miracles of the 20th century.[7] They are found in farming, patient medical gear, industrial equipment, and more. Here is just a small list:

answering machine	ATM	Calculator	CD player	coffee maker
computer display	cordless phone	credit card	digital camera	digital clock
digital watch	dishwasher	dryer	DVD player	DVD remote control
fax machine	fitness device	game console	garage door opener	GPS device
hard drive controller	keyboard	medical device	microwave oven	network card
PDA	pod coffee maker	printer	refrigerator	robot vacuum
security system	SSD controller	stereo receiver	stereo remote control	TV
thermostat	TV remote control	TV set-top box	VCR	vehicle cruise control
vehicle front detection	vehicle infotainment	vehicle motor control	vehicle wireless comm	washing machine

Many embedded systems interact with processes or the environment, so their ability to quickly take an action is important. Some of these are real-time devices, so they start and finish a task in a defined time. Others are stand-alone, networked, or mobile, and they do not rely on any other system to complete a task.

Some embedded devices lack an operating system. Many are low-cost battery-operated units that handle tasks that are ill-suited for PCs. Others may never need a software upgrade or reboot and require little to no maintenance, making them useful non-serviceable devices. A majority are not upgradable or repairable, and many are labeled “disposable electronics.”

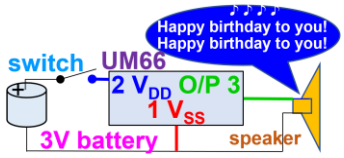
Embedded systems and microprocessors began around the same time. In the 1960s, they went to the moon, automated fuel injection in 1970s cars, and became a fixture of battery-powered calculators. Even with a common heritage, many of us still focus on general-purpose information processing disciplines instead of the constrained challenges that are posed by limited computing power, low cost, small power supplies, and tiny memory modules often found in embedded systems.

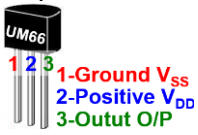
What is An Embedded System?

The embedded system definition is sometimes vague. For instance, is an iPad an embedded system? [This author calls it a general-purpose system since it performs functions that are not fully defined at the factory]. Embedded systems are found in larger devices that use hardware and software processing to perform dedicated functions. They can be real-time if a guaranteed timely response to a set of inputs is needed, such as with an autonomous driving system.

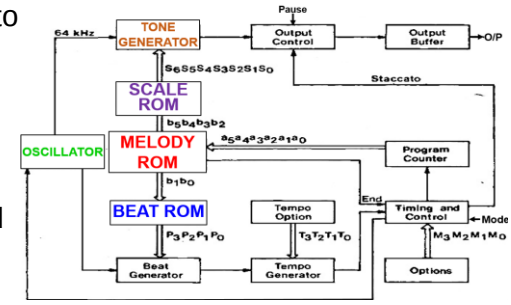
Make music

Let’s start simple. Did you ever get a musical greeting card? Open it and it started playing “Happy Birthday to You!” The card had four components - a **UM66** embedded monophonic sound microprocessor, a **battery**, a tiny **piezo speaker**, and a **switch**.[8] The **switch** is a metal contact that closes as the card opens. The low-voltage **speaker** and **3V power** are based on datasheet circuit requirements. The **UM66** looks like a 3-pin transistor. Its black body has leads numbered **1-2-3** with special meanings. In the diagram above, this IC is wired like a paint-by-number painting – you don’t have to be an engineer to attach **lead #3** to a speaker and create this musical embedded system.

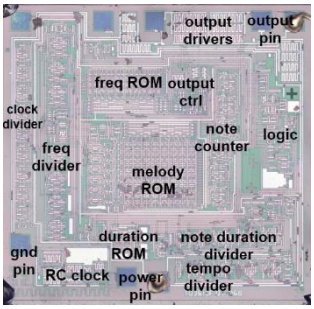




The magic is in the **UM66**. With an integrated **oscillator** to supply a **beat** from 15 defined levels, **tones** are created from 62 musical notes stored in the **melody Read-Only Memory (ROM)**. Each IC’s **ROM** dictates what song plays, so the UM66T08 has a “Happy Birthday” **ROM** and the UM66T33 has “Mary Had a Little Lamb.” An engineer



labeled this 1980s **UM66T** melody generator block diagram that converts electrical signals to audio signals.[9] The chip uses 4 bits to control the **scale** and 2 bits for the rhythm.[10] To the left is a comparison of the 40-year-old chip with a penny, and it measures just 1/16” by 1/16” or about the size of **two raised digits of the coin’s year**.

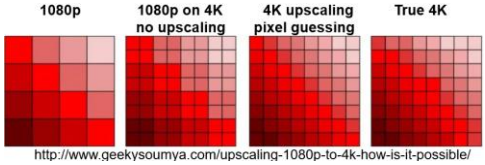


The sound can be amplified by adding a **BC547** transistor, or with additional circuitry to record a greeting, welcome someone who presses your doorbell, or be the basis for a police car siren.

An embedded system contains one or more processors, often called microcontrollers. Intel and AMD are market leaders in the general-purpose world, while STMicroelectronics, Infineon, Analog Devices, and others are popular in embedded systems. Microcontrollers are ICs, like the ones in your PC, but are designed to be less powerful, lower cost, and carry out specific tasks.

Your TV

Inside a 4K resolution TV, algorithms run on an embedded processor to upscale lower-resolution video, such as 1080p with 2M pixels to 4K and 7.5M pixels. A nearest-neighbor interpolating algorithm takes each pixel and replaces it with four similar pixels.[11]



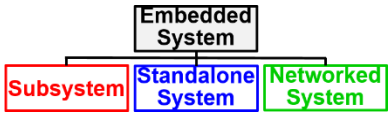
http://www.geekysoumya.com/upscaling-1080p-to-4k-how-is-it-possible/

There are many approaches, each with different outcomes. The latest TVs use a deep learning algorithm with 16 neural networks to analyze images and enhance them.[12]

Some TV brands use embedded processors and controller boards from MediaTek, Novatek, Sigma, or other embedded suppliers.[13] These processors run applications to handle Bluetooth, HDMI input, TOSLINK optical fiber audio, and more. It is a “Smart TV” when hooked to the Internet. TVs became intelligent when they incorporated embedded systems and processors. For example, MediaTek’s **MT9950** processor is coded for 8K TVs with AI algorithms for enhanced motion.[14] Its 64-bit Arm Cortex-A73 quad-core 4-thread processor and integrated TV graphics GPU operate at 1.8 GHz. It handles 3.6M floating point operations a second.

Embedded systems became popular because their software gave devices more capabilities than they could affordably achieve with just hardware. Sometimes called firmware, it is not found on a hard drive or SSD PC storage, but in ROM, Programmable ROM (PROM), or flash memory chips. The code may have been written in the microcontroller’s machine code (generally not X86 instructions), or in a high-level C or C++ complied language. From there, it is fed into the embedded system PROMs or flash memory.

Embedded systems fall into these categories:



- **Subsystem** – As part of a bigger system, like a car's  instrument cluster which has a fuel gauge, speedometer, odometer, tachometer, oil pressure gauge, and alerts for things like an unlatched seat belt.
- **Standalone** – Embedded systems that perform their tasks autonomously, such as a calculator or an old iPod. These devices are not part of another device. Input from analog or digital ports is processed and output is sent to output ports.
- **Network** – Connected embedded systems, perhaps through Wi-Fi. For example, an XBOX game played with a world of friends. This is a form of Internet of Things (IoT) or Industry 4.0 where a device with sensors, software, and processors exchange data with other Internet-embedded or high-level devices such as general-purpose computers.

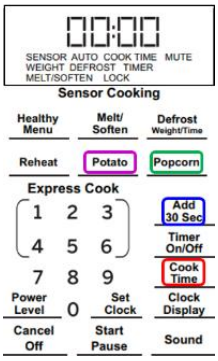
Embedded System Example – In The Kitchen

In my kitchen, I have embedded systems in my toaster, air fryer, refrigerator, microwave oven, dishwasher, coffee machine, LED lighting, calculator, Google mini, and more. There could be a million lines of programming code in just one room! Two of my favorite embedded systems are my microwave oven and my pod coffee maker. They are appliances that make my life better, especially when I am in a hurry. They are both affordable and hide the technology from me with their simple user interfaces.

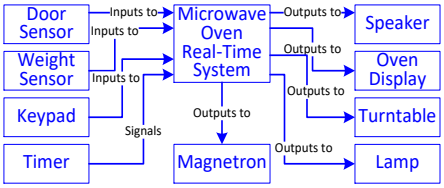
Reheat With The Touch Of A Microwave Button

The original 1947 "Radarange" began with WWII radar.[15] Its underlying microwave technology is truly amazing and often taken for granted. Designed to be reliable, it has a radar-inspired 2.45 GHz magnetron, motor, display, touch panel, and other components. Mass-produced, safe, and low-cost, its embedded system logic board and microcontroller run a multilingual program.

Most microwave ovens lack an operating system because the tasks, while complicated, are straightforward. Microcontroller software interfaces with the keypad to store data and update the display. It processes simple user logic, monitors electrical connection sensors including the safety latch, checks temperature, issues magnetron commands, and alerts to an item left inside.



The technology is purposefully hidden from the user. All that they see is a simple touch-panel user interface (UI) of labeled buttons for **cook time**, **add 30 seconds**, **popcorn** or **baked potato** menus, and more. The microcontroller queries the user when its logic recipe requires information about the weight of the food, whether it is frozen or defrosted, and the number of servings. The code was designed by the manufacturer based on the device's price points, component selection, marketing features, Commercial Off-The-Shelf (COTS)



"Real-Time Software Design for Embedded Systems", ISBN 978-1-10704109-7, P 860

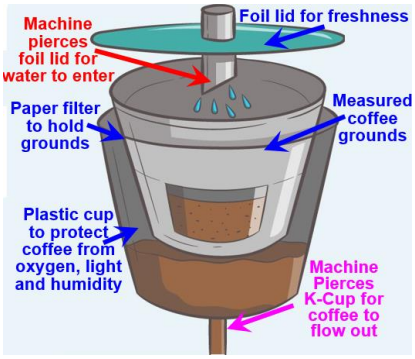
part availability, and more. New models use Wi-Fi to interface with a smartphone app.[16] Here is a system context diagram for the embedded logic. Different embedded systems have different diagrams. While this looks complicated, it does not approach the complexity of your PC.

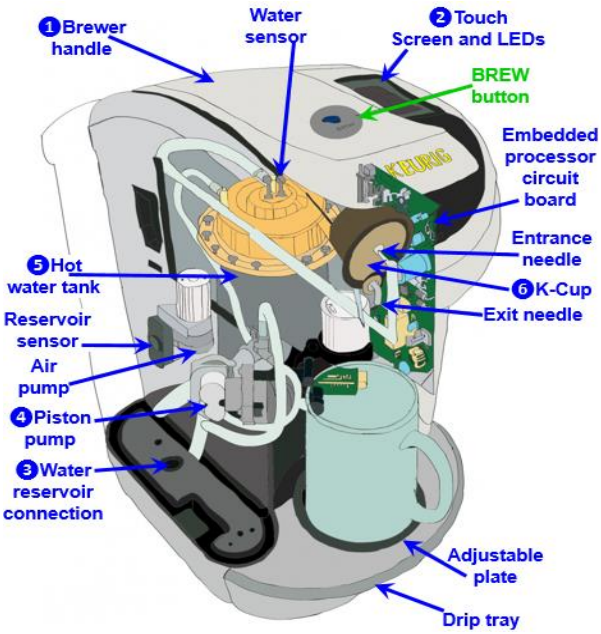
Have a Cup of Coffee with an Embedded System

When the Keurig coffee pod machine was introduced in 1998, I had no interest.[17] I either made a pot of coffee or had instant coffee in the pantry. I became hooked on it a few years later.

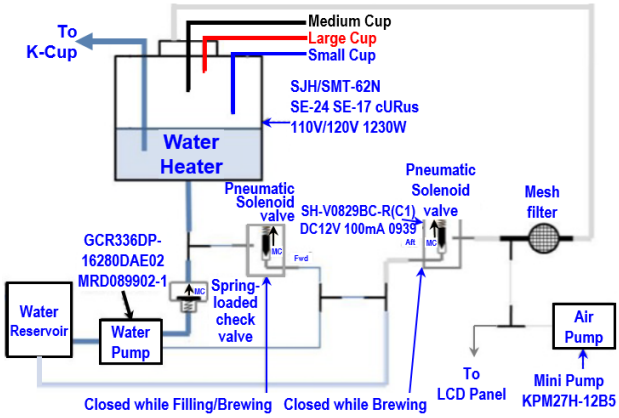
To many of us, waking up in the morning can be a challenge, especially if we are late for work. It is nice to know our first cup of fresh-brewed coffee is just an embedded system coffee pod away. Keurig is a good example of an embedded system in our kitchen. The latest version detects the brand and roast of the pod to adjust the machine's settings, and through the app on your smartphone or tablet, you control the brewing strength, temperature, and cup size.[18]

What happens when you want to brew a cup of coffee?[19] The pod was sealed at the factory to protect the coffee from oxygen, light, and humidity.[20] When you put it into the machine, pulling the handle down❶ causes a sharp needle to pierce the **foil lid** and the **bottom** of the K-Cup. The microcontroller takes over, and you pick cup size and brew strength from the user panel❷. Press the flashing BREW button, and a command is sent to the reservoir pump❸. ❹Water flows into the ❺hot water tank, and when it is hot enough, it is air-pumped through the ❻pod. Coffee pours into your cup. The used coffee grounds are held by a paper filter that is bonded to the cup.





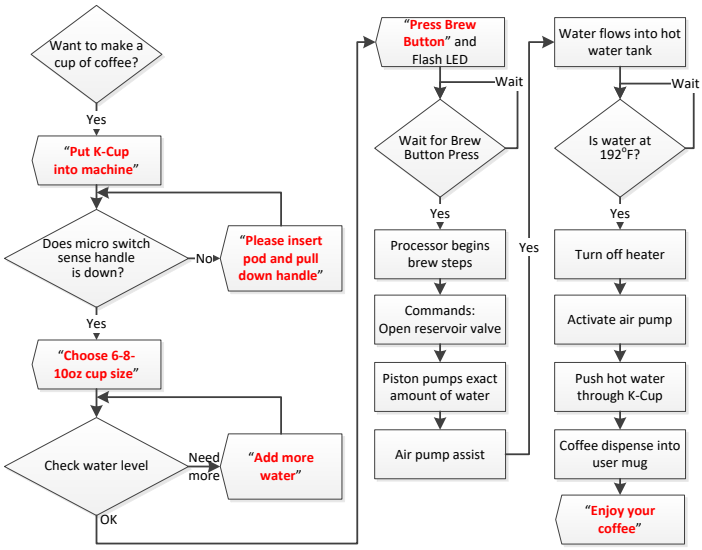
Inside the friendly Keurig machine is a scary set of silicone tubes, 12 V pumps, buttons, LEDs, a magnetic reed switch that senses the tank's water level, a step-down 120 V to 14.5 V .5A transformer, microswitch, check valve, solenoid valves, sharp coffee pod piercing needles, a



water heating unit, pressure switch, plenty of screws, and an embedded system to orchestrate activities.

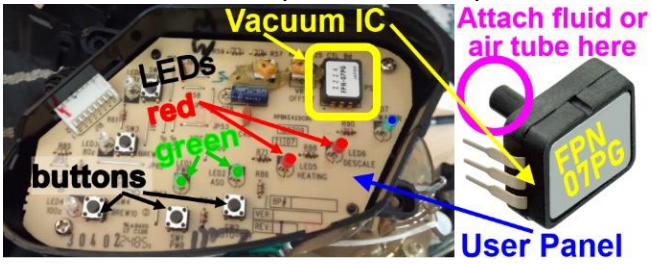
To the left is the Keurig model B60 mechanical schematic.[21] Although every model is different and can incorporate different COTS parts, the diagram is helpful to Keurig software engineers when they code and debug the microcontroller.

This simple algorithm flowchart describes the activities.[22] A welcome message says to insert a K-Cup, and a microswitch detects it is ready and signals that to the embedded controller. The user picks cup size and brewing strength choices on the control panel. The Keurig's magnetic reed switch lets the microcontroller know if there is enough reservoir water to brew one cup of coffee. Otherwise, an alarm may sound and a message that is displayed on the display screen. Once the "brew" button is pressed, status messages appear, and LED lights flash or change color to let the user know the brewing process has begun.



Valves that are controlled by the processor open and a calculated amount of water is pumped from the reservoir into the heating chamber. The heater is activated, and a sensor lets the controller know when it reaches 192°F.[23] Rather than rely on gravity to circulate the hot water through the K-cup and to avoid drips, an air pump pushes the hot water through the pierced pod. The process takes about 3 minutes, and the user display provides the ongoing status.

In a Keurig K45, pumps and other components attach to the control panel.[24] The top of the **user panel** has **red** and **green** status LEDs and **user push buttons**. There is also an uncommon 6-pin **FPN-07PG** vacuum pressure sensing IC that goes through the board. On the back of the control panel, a

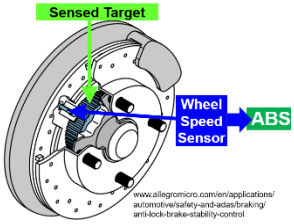


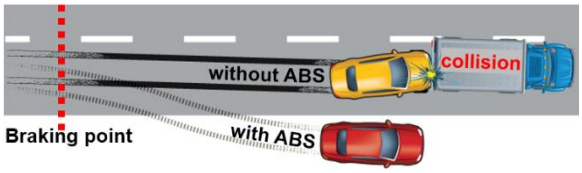
plastic tube attaches to the **vacuum sensor**.[25] The chip's pressure ratings, fluid, and air specifications are useful to the Keurig engineer designing the plumbing.

The intelligence is in this circuit board. **Microchip**'s low-cost 44-pin **PIC16F917** is a high-performance RISC CPU supporting just 35 instructions.[26] With an 8-level hardware stack, it incorporates direct, indirect, and relative addressing modes, and a power-saving sleep mode. This embedded processor supports an LCD panel and has 14 KB memory and 256-byte EEPROM with an endurance exceeding 40 years.

Embedded System Example – The Automotive World

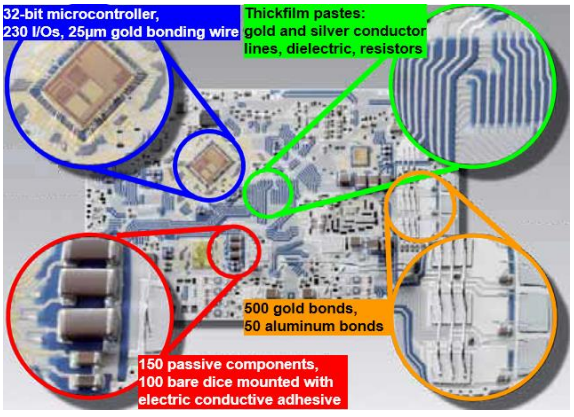
The automotive world is an industry revolutionized by embedded systems. Vehicles have Electronic Control Units (ECU) that monitor and guide their function. Their real-time embedded applications deliver ride features that might be impossible with mechanical systems. This technology originated with the 1978 Antilock Braking System (ABS).[27] When embedded ABS software and sensors detect a wheel is locked up, it keeps the car going straight and prevents skidding by rapidly pumping the brakes while commanding the valves to reduce brake pressure on all four wheels.[28] If you feel the brake pedal pulsing, it is the ABS stopping your car from skidding and automatically helping you stay in control.

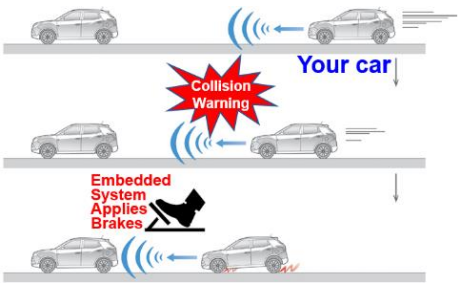




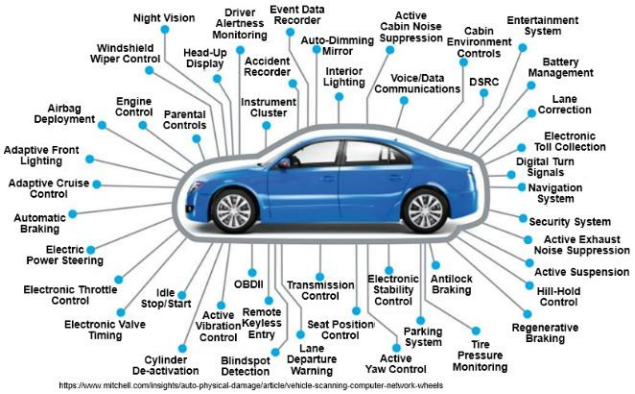
You are driving your 3,000-pound 2011 Volkswagen Golf 1.2 L TSI when it dawns on you its

controlled turbocharged engine is communicating with a Transmission Control Module (TCM) over its Controller Area Network (CAN Bus). The automatically shifting DQ200 dual-clutch 7-speed Direct Shift Gearbox (Doppelkupplungsgetriebe in German) is given instructions by a surface-mounted **32-bit microcontroller** that was made in 2008.[29] The DQ200 TCM to the right has **hundreds of tiny connections** and **tiny components** that are connected with **gold and silver lines**.[30] It is making decisions in real-time based on the Golf's speed, the RPM of the wheels, the load on your 4-cylinder engine according to the Throttle Body Sensor, Turbine Speed Sensor, and even your transmission fluid temperature.





Another incredible embedded system detects front collisions. Using radar, laser, and camera sensors, your vehicle warns you or is automatically commanded to apply its brakes to keep you from crashing into a car that you are rapidly approaching. Just the software in this market sector has experienced a compound growth rate of over 13%.[31]



Every year manufacturers add more embedded systems to their designs to improve safety, economy, automation, or creature comforts. As shown, your vehicle has 50 to 150 specialized ECUs, each with one or more embedded processors that control what you take for granted. As its sensors send reports over the CAN Bus, they are handled by 8, 16, and 32-bit technology using fractions of the memory size compared to your PC. They make your ride safer, fuel-efficient, reliable, and secure since serious injuries can occur if your car's ECU "blue screens" or is forced to reboot at 65 MPH.

The trend for vehicle-embedded systems is to reduce hardware costs by combining several individual units and their software functionality into fewer, more powerful multitasking ECUs.[32] Expect this to occur as the popularity of electric vehicles with self-driving features increases. At the same time, the cost to develop this new software keeps increasing with each model year.

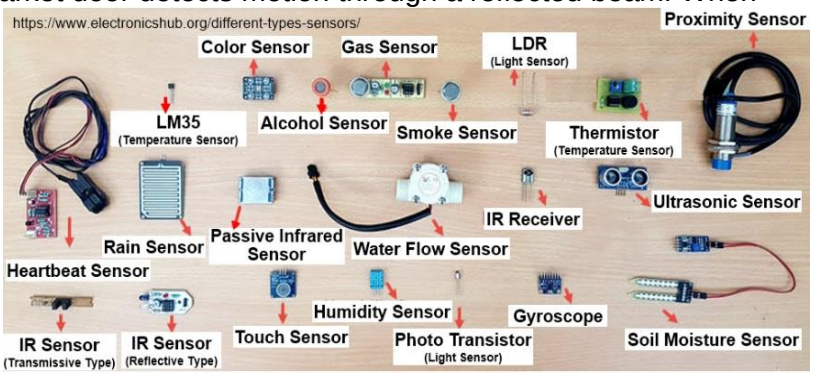
What is a Sensor?

Sensors are a critical component of an embedded system, especially when they are part of the IoT. They work within the physical environment to detect "something," such as changes in light, sound, temperature, pressure, motion, moisture, or other environmental occurrences. A useful example is how a traffic signal is aware your car is waiting for a green arrow to turn left. Signals can use timers or detect a magnetic field from a coiled induction sensor wire in the road beneath your car. Sensors can be binary in function, such as detecting the presence of a wall or not, or analog in the case of a pressure sensor. Sensors often fall into this partial list of categories:

acoustic	**ambient light/optical**	**electric/magnetic**	**force/pressure**	**chemical/gas/radiation**
humidity	**leakage/level/flow**	**locked/unlocked**	**motion/acceleration**	**temperature**

They allow an embedded system to mimic our senses of smell, sight, taste, hearing, and touch.

One type of sensor at the supermarket door detects motion through a reflected beam. When your body breaks the wave, the sensor stops receiving the signal causing the embedded system to send a command to the door motor to turn. This shows some of the sensors that are used by embedded systems.



An active sensor does not require external power while a passive sensor does. Thermocouples are active sensors that report the temperature as an analog voltage using two different metals joined at a junction. The voltage level directly maps to a temperature.[33] Your digital camera uses a passive sensor that receives reflected light. In both cases, the sensor data is processed by an embedded processor.

Sensors can be processed through analog-to-digital conversion or vice-versa, and be external to the microcontroller or part of it. When a robot's bumper senses a wall, the event is processed by its embedded processor, resulting in its program issuing commands for motors to reverse and change direction. In a factory setting, a digital sensor can detect a jar filled with the correct weight of peanut butter. The embedded system receives the sensor's value, commands the flow to stop, and commands the conveyor belt to move the jar to the lid-fitting station.

Programming Languages and Operating Systems

Embedded systems often have small memories and little processing power in comparison to the enormous memory and high-clock multi-core CPUs of general-purpose computers. One of the things they have in common is their support for C and C++ to build applications. While programming in Python or Java is easier and is supported by embedded systems, neither is generally used because they do not generate machine code. This could cause them to run slower. In addition, they typically do not manipulate hardware features or manage memory.

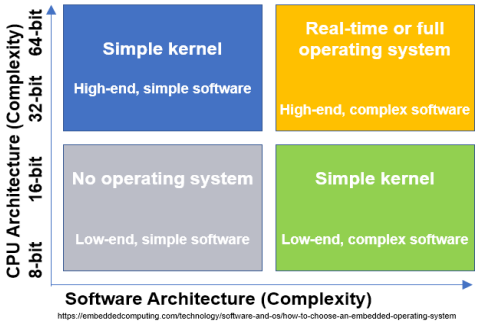
C is a general-purpose compiled programming language that was created in 1972 by UNIX operating system pioneer Dennis Ritchie.[34] It generates efficient compiled machine code and is an essential portable piece of technology that is found in perhaps 80% of all embedded systems.[35] C++ is another compiled language that adds object-oriented programming to C and is found in embedded systems. It is widely used for systems-level programming and building applications on Windows and various UNIX-derived operating systems such as Linux. With C++,

care is needed to not create “bloated” code with extra features not needed by a device. The embedded processor’s native assembly language produces the most efficient and fastest code since it interfaces directly with the hardware, however, assembly language is harder to read and maintain.[36]

Embedded coders may need to follow standards, such as when the device is used in vehicles. The Motor Industry Software Reliability Association’s coding guidelines ensure that code is safe, secure, and reliable.[37] For example, one of its rules requires initializer lists with expressions to be evaluated only at run-time. Unfortunately, **lines 6** and **7** modify the value of a variable that is used in another element, and the evaluation order can lead to unexpected values.

```
1  int x=0;
2  int y=1;
3  volatile int v;
4  void func()  {
5       int arr[2] = {x+y, x-y};
6       int arr2[2] = {x++, x+y};
7       int arr3[2] = {v, v};
8  }
```

Modest devices with low-end processors and simple software often have enough embedded program functionality to not need an operating system.[38] At the other end of the spectrum, complicated multitasking software is easier to write and support when embedded systems use one. This chart shows part of the decision process.



Designers often choose between embedded Linux such as Android and Ubuntu, Windows for IoT, or others that are freely found on GitHub.[39] As with Windows, most embedded operating systems can be upgraded or patched to fix bugs. For example, your Smart TV may have an operating system and its manufacturer can release a new version of it or a new program to add features or close security loopholes. In general, you will not see general-purpose operating systems like Windows 11, UNIX, Solaris, or HP-UX running on an embedded system.

Some devices, such as those used for telecommunication or avionics must complete complex tasks in a specified time and need a Real-Time Operating System (RTOS). It does not need to equal high performance, but it must enforce output time constraints and is often deterministic (the same results are generated from a given initial state.) High-priority tasks can be given more resources than low-priority tasks. Wind River VxWorks and BlackBerry’s QNX are RTOS examples.[40,41] They run on top of the embedded processor and support real-time embedded applications. RTOS embedded systems are complex since they handle unpredictable interrupt-driven passive sensors and actuators and generate pre-determined time constraint outputs.

With an RTOS, the coder must use a compatible language such as C. Other languages have real-time versions, but in general, C++ uses dynamic memory which makes it incompatible, Python is non-deterministic, and Java performs dynamic garbage collection.[42] Comparing a general Linux distribution with QNX, we see the following differences:[43]

General-purpose Linux	Embedded QNX
Designed for embedded systems, mobile devices, personal computers, servers, mainframe computers and supercomputers	Designed for automotive, medical, smartphones, consumer, industrial, embedded systems and safety
Supports IA-32, x86-64, ARM, PowerPC, SPARC	Supports x86, SH-4, PowerPC, ARM, MIPS
Kernel: Monolithic	Kernel: Microkernel
Native APIs: LINUX/POSIX	Native APIs: POSIX and Java
License: GNU GPLv2 (kernel)	License: Proprietary
Non-native APIs: supported through its subsystems are Mono, Java, Win16/32	Non-native APIs: are not supported through its subsystems
Supported file systems: ext2, ext3, ext4, btrfs, ReiserFS, FAT, ISO 9660, UDF and NFS	Supported file systems: QNX4FS, QNX6, ext2, FAT, ISO 9660, Joliet, NFS, CIFS, ETFS, UDF, HFS, HFS+ and NTFS

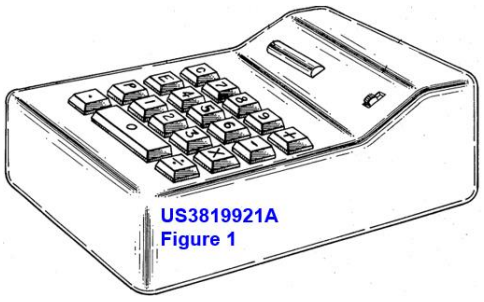
Specific Purpose vs General Purpose Computing

Windows or Linux computers have the same basic display, keyboard, trackpad, USB ports, Wi-Fi and Bluetooth, and so forth since they run the same basic software (but may use different drivers.) They are also programmed to perform many tasks such as office software, web surfing, databases, tax preparation, and so on. They are often upgradable with more memory, different storage devices, new programs, and perhaps a replaceable battery. Their versatility makes them end-user-friendly and popular, but because they are designed to carry out many types of user processes, they are intentionally not built for any particular task.

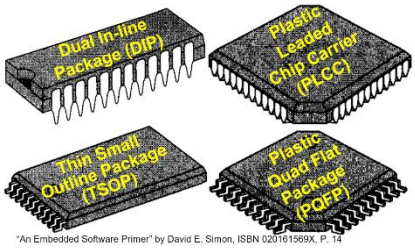
When cost, size, or simplicity is paramount, embedded systems often provide better solutions than general-purpose computer systems. The embedded design process focuses on selecting components based on the dedicated tasks. This is in contrast to a general-purpose Windows or Linux system which starts with a compatible processor, perhaps a reference motherboard, an SSD, hard drive, removable Dual In-line Memory Modules (DIMM), and so forth. Embedded systems are also capable of running real-time processes, something that is harder to accomplish with a general-purpose system.

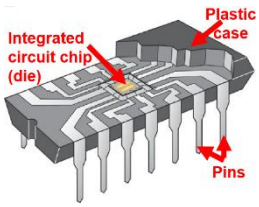
Where Did Embedded Systems Begin?

The building blocks of the microprocessor date back to the Nobel Prize recipient Jack Kilby who invented the IC in 1958. He co-invented (**US3819921**) the handheld calculator in 1967, marking one of the first embedded systems in the market. It contained an “integrated circuit array” of four ICs, three other chips, and a thermal printer instead of a display.[44]

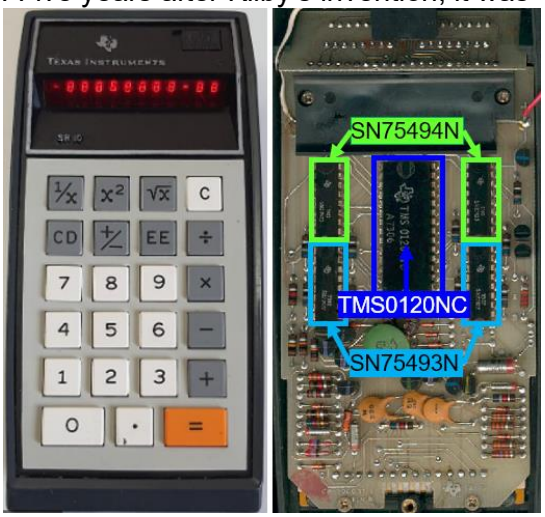


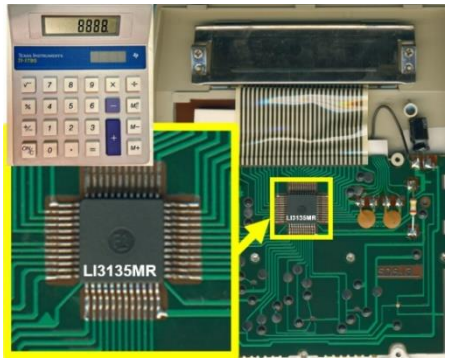
Intel introduced the first microprocessor in 1971 when they shrunk the logic of discrete components into one silicon IC. A chip is wired to larger pins and put in a housing such as this Dual In-line Package (DIP) to the right. There are dozens of packaging options, each with its own cost, robotic machinability, and other factors.





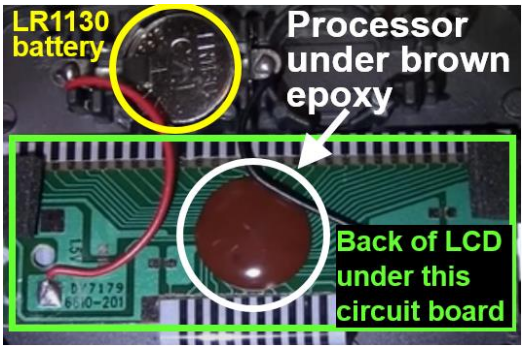
My first calculator was a Texas Instruments (TI) SR-10. Five years after Kilby's invention, it was marketed as an electronic slide rule for just $150 (~$900 today.) It had four functions plus reciprocals, squares, and square roots, but no user memory nor all the functions of a real slide rule. Its 3 rechargeable NiCd batteries powered a circuit board packed with dozens of resistors, a handful of capacitors, components, and 5 ICs. Its TI **TMS0120NC** calculator chip used a 320-word ROM and 182-bit memory, with a pair of **SN75493N**s and **SN75494N**s ICs to operate the 12-digit **red LED** display.[45]

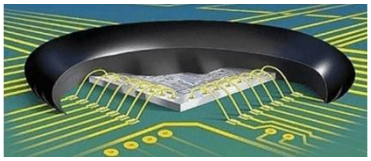




Jump ahead about a dozen years and this Texas Instruments TI-1795 rebranded sub-$20 solar-power calculator from Compal Electronics has a memory. The circuit board has just one resistor and two capacitors, and instead of a TI chip, it has one tiny Sharp Electronics **LI3135MR** chip.[46] By then, calculator mass production is in full swing as prices get even lower and functionality increases – Moore's Law.

These days, "dollar store" scientific calculators are plentiful. Outwardly they do not look all that different from Kilby's design, but inside it is a different story.[47] Disassemble it and you will not see any discrete components, just a keypad, **battery**, **red** and **black** wires, and an **LCD backed by a tiny circuit board** with a **blob of brown epoxy** covering the processor.[48] If you could remove the epoxy,

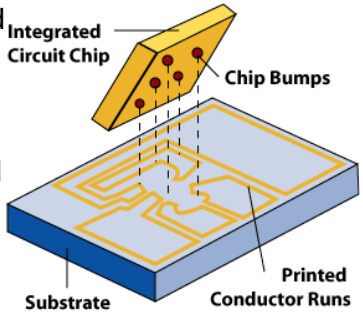




you would not find chip markings. The circuit uses low-cost non-repairable Chip On Board (COB) wire bonding to attach it to the board and no publicly available schematics.

The calculator's program that displays the answer to your key presses may have been written in C or C++, but most likely in assembler language to get the byte size as small as possible. Unlike an EPROM where binary code is stored in the chip, the binary code is translated into a photo "masked ROM" where the code is printed through photolithography and can never be changed.

The $1 calculator is profitable even with pennies for royalties, printed packaging, housing, screws, keys, LCD, solar panel, IC, and other items. Cost pressure helps push technology to find ways to mount chips such as this clever use of solder bumps on the silicon chip and their placement on the board.[49] After 50 years, you are still seeing incredible miniaturization with cost being the overriding issue, yet all these calculators are still able to handle the basic math functions.



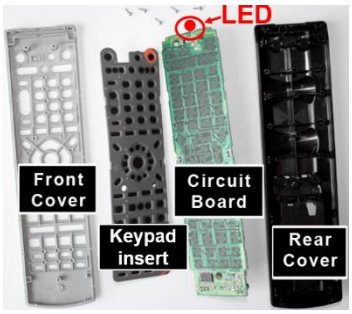
More Examples of Embedded Systems

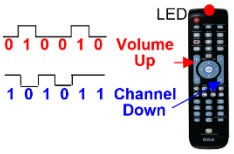
Remote Control

Many of us have over a dozen remote controls that operate the TV, open the garage door, control the stereo, and more. They originally came wired to a VCR or used sound waves for on/off, channel up/down, and mute by vibrating an aluminum rod to create an ultrasonic tone.[50] This 1955 embedded device had no electrical components – it was purely mechanical.

Today, there are learning or code library-embedded system remotes that store codes in a ROM. They are bar-shaped, have rubberized buttons that you press, and an invisible-to-the-eye infrared **LED** on the front that transmits rapidly flashing signals. When your TV's infrared receiver picks up those signals, it converts them to actions such as volume up.

Many remote control buttons have a rubber-feeling Santoprene (thermoplastic elastomer polymer) membrane that rests on top of a flat circuit board.[51,52] Push a button and a contact touches conductive circuit board traces that signal the microprocessor to generate a specific binary code. The code signal is sent by rapidly flashing the **LED** on the front of the control.





Volume up's code could be **010010** while **channel down** uses a different code. Signals use a 6-bit "Manchester" encoding with "0" for low voltage and "1" for high voltage. Each key is mapped to a code, which is modulated and sent at 36 kHz to avoid interference from the sunlight's infrared light.

Many remotes are "universal" and operate DVD players, stereos, cable boxes, sound bars, and other devices. This is done by using a **5-bit Address** code in an infrared packet to identify the device it is controlling.[53] With 14 total bits, a "Start" bit indicates the start of the frame, a "Field" code toggles the commands bits to 127 devices, and a "Control" bit alternates between 0 and 1 every time a button is pressed. This coding method was created by Philips in their RC 5 protocol with some of the commands shown to the left.

Command	Function
16	Increase volume
17	Decrease volume
18	Increase brightness
19	Decrease brightness
22	Increase bass
23	Decrease bass

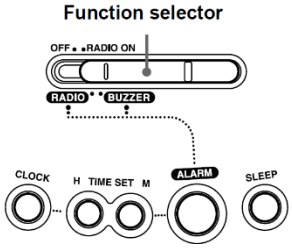
https://en.wikipedia.org/wiki/RC-5

S	F	C	Address bits	Command bits
3 bits			5 bits	6 bits

Some remotes use an **ADAM27P16G** 4-bit processor that was designed for transmitter applications such as a TV, fan, stereo, air-conditioner, and more.[54] It has 2,048 bytes of program memory, 32 x 4-bit data memory to address the ROM, and a nested subroutine architecture. It is programmed using 43 OP code/operand 8-bit instructions.

	Ground	GND	1	16	VDD	Positive power supply
	2-bit I/O	CS1	2	15	ROUT	High current pulse output
	4-bit I/O	K0	3	14	CS0	2-bit I/O
Serial clock,	4-bit I/O	K1	4 ADAM27P16	13	P3	4-bit I/O
	4-bit I/O	K2	5	12	P2	4-bit I/O, Serial Data I/O
	4-bit I/O	K3	6 (16-sop)	11	P1	4-bit I/O
Program/Verify Power,	4-bit I/O	R0	7	10	P0	4-bit I/O
	4-bit I/O	R1	8	9	R2	4-bit I/O

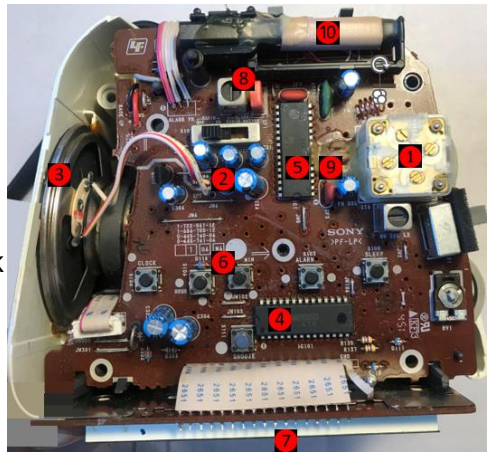
Digital Clock



My 2003 Sony "Dream Machine" digital alarm clock always woke me with an FM song. It never blue-screened, had a code upgrade, nor a license agreement, and I'm still using it. Engineers had a simple UI to set the time and gave me a buzzer or radio alarm option. This embedded system even included a 9V battery backup.[55]

Sony used COTS components to design the logic board:[56]

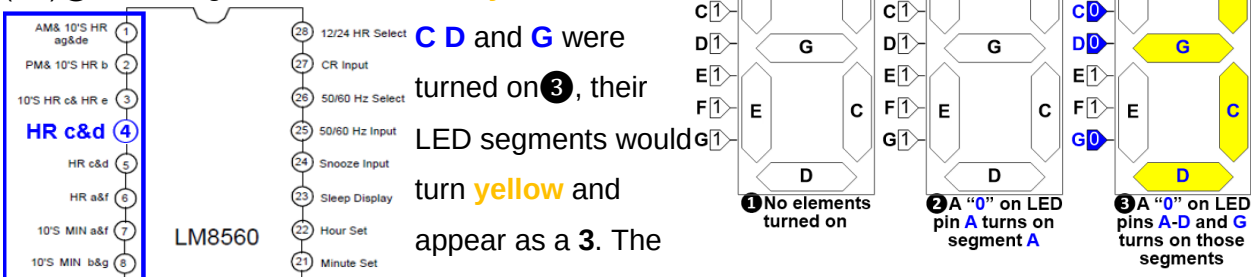
1 radio	2 capacitors	3 speaker	4 **LM8560 IC**
5 **CXA1019S IC**	6 user button	7 LED screen	8 IF transformer
9 ceramic filter	10 AM antenna		



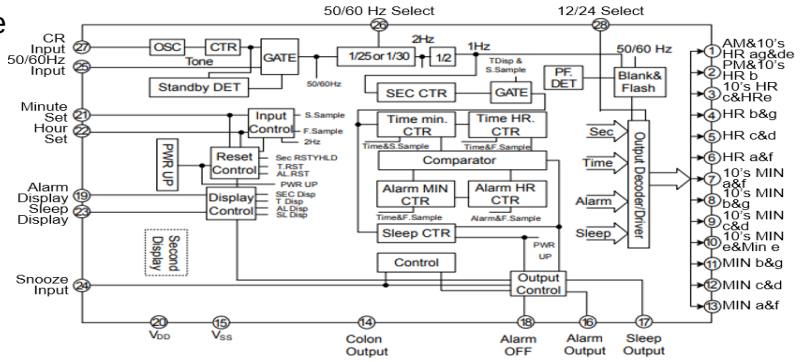
Sanyo's **LM8560 IC**4 special-purpose embedded processor was designed to be a 24-hour alarm digital clock IC that displayed the 12-hour AM/PM or 24-hour time on an LED7. Through code, it can fast-forward the time and provide a variable-time snooze function.

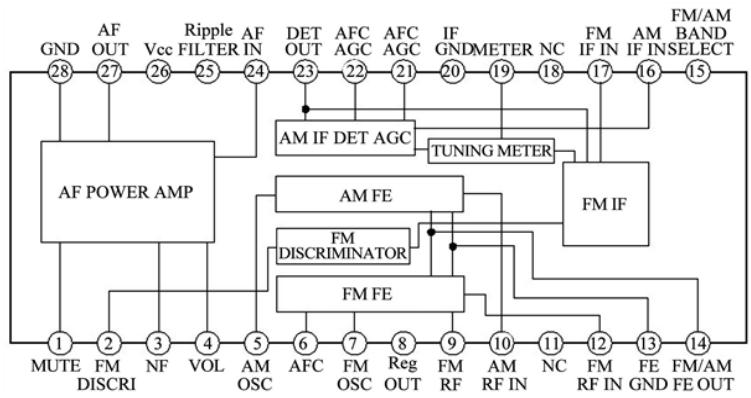
The time is displayed using four 7-segment LEDs arranged as shown to the right. Each LED can display the numbers 0-9 when their binary codes are sent by the **LM8560**.[57] When all 7 pins **A**-**G** are turned off ("1")❶, all LED segments are off. If **A** was on ("**0**")❷, LED segment "**A**" would turn **yellow**. If **A B C D** and **G** were turned on❸, their LED segments would turn **yellow** and appear as a **3**. The **LM8560** and pre-programmed code control the LED segments. For example, using the **blue pinouts** on the left, **pinout** ④ controls **digit #2** in the table above for single-digit **hours (HR)** with **LED segments c & d**.

digit #1	digit #2	digit #3	digit #4
10's place for hour	**hour**	10's place for minute	minute



This **LM8650** logic diagram has the factory ability to set an alarm, snooze, and set the current time, It turns on and off a buzzer or radio, blinks the colon yellow dots on the display between the hours and minutes, and other functions.[58]



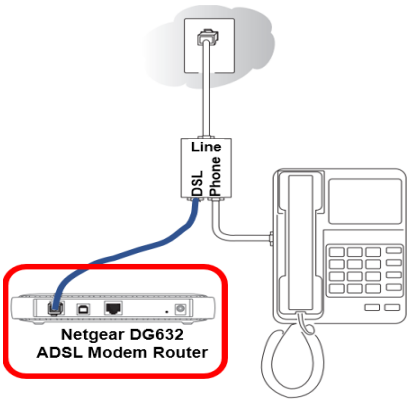


The **CXA1019S IC**❺ in the teardown on the previous page is a socketed SONY FM/AM radio receiver chip. Programmed once, it was used in many products, some dating back to 1985, or 18 years earlier than this clock radio. It was also used in radio-cassette units requiring an amplifier.[59]

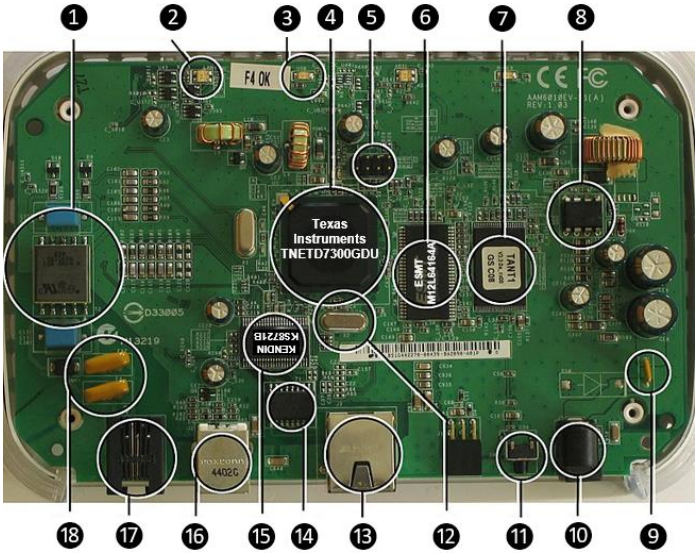
Affordable Netgear Modem/Router

In 2004, Netgear released a DG632 modem/router that replaced two separate Ethernet-cabled devices.[60] It was easier to use, needed one

power supply, and cost less than buying two individual pieces of equipment. It had all the hardware DSL users needed to get their PC on the Internet, including a security firewall (an important 2004 feature), and RJ-11 WAN, Ethernet, and USB ports.[61] You could attach your Xbox Live for network play, and Netgear provided code updates. A step-by-step "wizard" made it easy to install, an "expert mode," and menus to change security passwords and other settings.

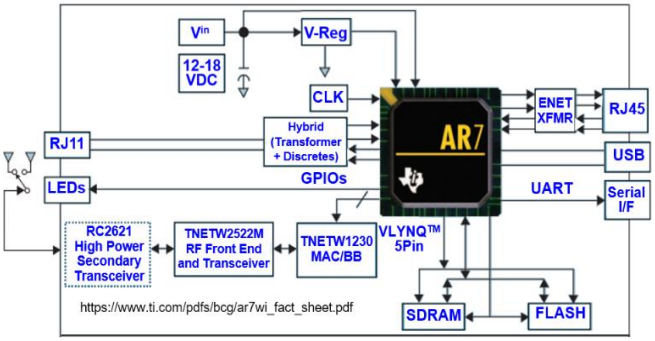


Netgear's creative engineers designed affordable and reliable equipment. This single logic board included: ADSL telephone connector❶, multicolor LED for network status❷, and a single color LED for USB status❸. A TI **AR7Wi** wireless ADSL router chip **TNETD7300GDU** that uses a MIPS 32-bit 160 MHz RISC processor❹, a test and programming port❺, and the 8MB synchronous DRAM ESMT **M12L64164A** memory chip is on the left❻. The other components are the flash memory to store the device's software and settings❼, power supply regulator❽, fuse❾, and a power connector❿. The reset button⓫, quartz crystal to keep time⓬, Ethernet port⓭, 10/100Base-Tx Ethernet transformer Delta LF8505⓮, **KS8721B** 10BaseT/100BaseTX/FX Ethernet transceiver⓯, USB port⓰, an RJ11 telephone line port⓱, and a telephone line fuse⓲. It was an amazing embedded system design.



Netgear allows for code upgrades in its embedded systems to give developers the ability to customize the firmware and applications with open-source code or patch code bugs after the product ships. This device eventually needed a security patch. **CVE-2009-2257** is the cybersecurity nomenclature used in the National Vulnerability Database to track **Common Vulnerabilities and Exposures**. In **2009**, item **2257** discusses the DG632 and its firmware 3.4.0_ap web interface that allows an attacker to bypass authentication.[62] Embedded devices will have cybersecurity issues, and while the product engineer's goal is to provide a secure environment, complex embedded systems need a mechanism to upgrade code as necessary.

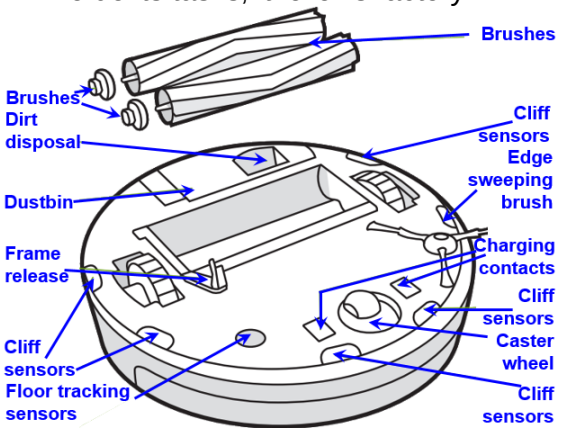
This highly-integrated **AR7Wi** chipset was the brain of the Netgear DG632 device. It allows wireless LAN components to be affordably placed on the same logic board as all the necessary cable attachments, status lights, and power hookups, all while supporting the necessary 2004 specifications.



Roomba

We all have household chores, and I use an embedded system to help clean mine - a Roomba robotic vacuum. Press the big **CLEAN** button or use a smartphone app and this intelligent cleaning machine rides around with spinning brushes vacuuming up dirt. My unit goes to its power station when it needs charging. Fancier models automatically empty the dirt container.

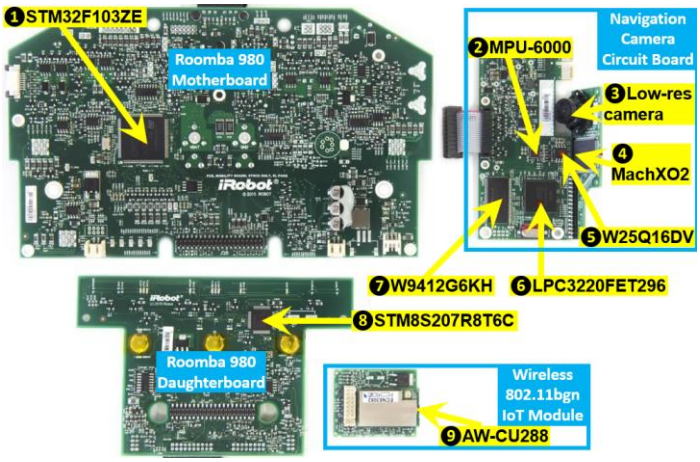
Turn the Roomba over, and it looks like this illustration.[63] To do its tasks, it follows factory-programmed instructions such as "wall following" and "random bounce" as it learns your cleaning floorplan. An embedded processor uses onboard infrared and other sensors to detect obstacles and staircases as it collects dirt in its dustbin. It also senses if the brushes are getting tangled in carpet fringes or wires on the floor. If its piezoelectric dirt sensor finds a dusty area, the location is noted as requiring extra cleaning. Some models chart the base location and obstacles with a camera and navigation algorithms to learn your cleaning area and minimize the cleaning time. Your smartphone app works with Wi-Fi models and can instruct it to clean when you are away.

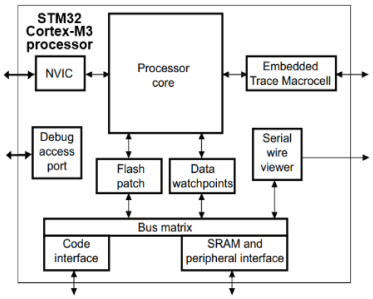


Inside the Roomba is an embedded controller that commands a suction pump, instructs brush motors to turn, automatically recharges built-in batteries, and collects feedback from 15 sensors. If you took apart a model 980 from 2015, you would find a diagnostic/software upgrade USB port.[64] The unit's front bumper has a sensor switch that is wired to the control board that detects contact with a solid object. Other sensors include the dustbin full sensor, a floor-tracking optical sensor like a computer mouse that helps navigate in the dark, and a downward infrared cliff sensor. If the infrared photocell detector does not receive the reflected signal in the correct amount of time, the control board determines there is a down staircase in front of the Roomba.[65] In all, it has the intelligence to detect 80 common objects.[66]

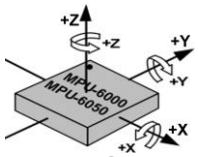
The processor runs proprietary navigation software that issues rule-based commands from smart mapping services that are called Visual Simultaneous Localization and Mapping (VSLAM.) The sensors send VSLAM mapping distance and direction, enabling the robot to identify objects like a couch corner. This concept is called odometry, and it allows a robot to know its location by motion sensing a position change relative to a previous location.[67] For example, with its starting position (X, Y), the wheel diameter, and traveling in a straight line, it can compute its new location (X', Y') with a simple math formula as wheel encoders report a revolution. Sensors detecting the locations of chairs, staircases, and other objects allow VSLAM to map the area.

Roomba's magic comes from its four logic boards. The 980 motherboard holds the STMicroelectronics **STM32F103ZE** 32-bit ARM Cortex Microcontroller that runs at 72 MHz❶, has 64 Kb of SRAM, and a 512 Kb embedded Flash to store data and code.[68] It controls the LCD, operates 11 timer circuits, and its 13 communication interfaces including a CAN Bus, USB, and Wi-Fi. The CAN Bus, which is a standard car interface, supports Roomba motors and actuators.



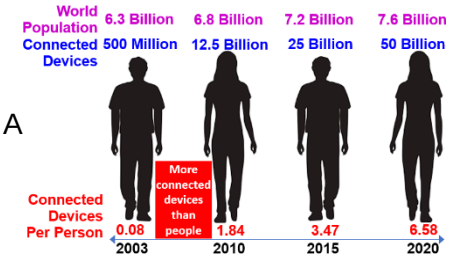


The robot navigates by relying on an **MPU-6000** with a built-in 3-axis X-Y-Z gyroscope and a 3-axis accelerometer❷, a Digital Motion Processor, and algorithms to determine where it has cleaned. As power runs low, it uses its front infrared receiver and the base infrared transmitter to return to its charger. The MachXO2❹ Lattice Semiconductor Field Programmable Gate Array uses sensor and camera❸ data to store map coordinates and landmarks that aid in Roomba navigation. Winbond's **W25Q16DV** 16Mb Serial Flash memory has 8,192 pages to store data with 8 pins❺. NXP's **LPC3220FET296** is a 32-bit ARM floating-point coprocessor, USB, and memory interface operating at 266 MHz❻.[69] The microcontroller has 32 kB of RAM, uses a 5-stage Harvard architecture with separate instruction and data buses, and 32 kB instruction and data cache. **W9412G6KH** is a Double Data Rate Synchronous Dynamic Random Access Memory with 2M words x 4 banks x 16 bits for 128Mb❼.



The T-shaped UI board handles the **CLEAN** button, buzzers, speaker, status lights, and other user tasks through the STMicroelectronics **STM8S207R8T6C**❽. This powerful 8-bit processor provides 20 MIPS at a 24 MHz CPU clock frequency.[70] With 64 pins and 6KB of RAM, it has a 32-bit wide program memory bus, an 80-instruction Harvard architecture, and a 3-stage pipeline. An **AW-CU288** 300Mbps Wi-Fi add-in module provides wireless Ethernet 802.11 b/g/n 2.4GHz allowing the Roomba to be remotely programmed❾.[71] All this processing power, algorithms, and sensors allow Roomba to move autonomously on its cleaning journey.

Internet of Things and Embedded Systems



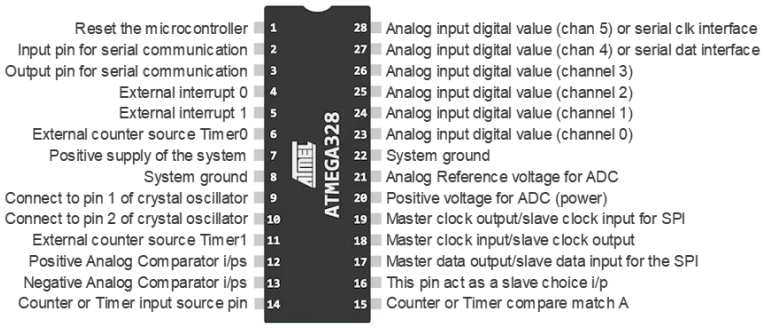
There have been many IoT discussions over the last two decades.[72] In 2003, there were **500 million** mostly PC or business Internet-connected devices, and **6.3 billion people**. A few years later, it reached a tipping point and networked systems outnumbered people, **12.5 billion** to **6.8 billion**. These days, the set of embedded networked devices has far eclipsed that number.

In contrast to general-purpose computers, IoT devices are web-enabled smart embedded systems. With Wi-Fi, Ethernet, or Bluetooth, the device could be a benign part of an environment such as a Samsung “Smart” refrigerator.[73] It keeps beer cold, and ice cream frozen, and is a member of a Google Hub, so it displays a calendar and photos, shows who is at the door and maintains a virtual grocery list. And as you expect from an IoT device, the refrigerator's warranty does not mention software, upgrades, rebooting, or anything associated with a general-purpose computer.

Soil Moisture

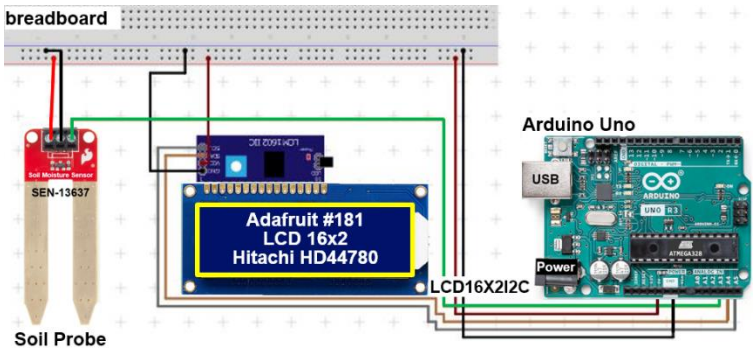
Here is an IoT device for a tomato farmer. Droughts have necessitated the use of automated drip irrigation methods. Water content is key to the ratio of sugar in fruit, and is measured by the Balling Relative Intensity Index (BRIX).[74] Instead of watering the tomatoes daily, it is best to water them when they need it. The farmer’s local agricultural college’s CS department might create an embedded system prototype to monitor soil moisture.

The systems analyst decides to use COTS parts and picks the Arduino Uno because it supports 14 digital pins, 6 analog inputs, a USB connection, and external power. The Arduino is programmed using an Integrated Development Environment (IDE) that runs on a PC. The IDE is a text editor, debugger, C++ compiler, linker, and USB uploader into the Arduino's EEPROM as binary.



The Arduino's **ATmega328P** 8-bit microcontroller with 32 x 8 general-purpose registers, a built-in 1kb EEPROM programmable flash memory, and analog inputs can be programmed with 131 different instructions.[75]

Developing solutions with this embedded system is accomplished using a wiring breadboard. The completed embedded circuit looks like this image. You can see the AdaFruit LCD and Soil Moisture Sensor probe that is attached to the Arduino Uno.



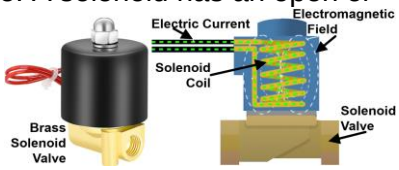
The 16-character, 2-row Hitachi **HD44780** LCD #181 supports English and Japanese text from a character ROM as driven by an **SPLC780D1** dot-matrix display IC controller on the reverse side of the LCD.[76] Put the sensor's two soil probes in the soil and they act as a variable resistor that conducts electricity based on the detected moisture level.[77] Higher water concentration lowers the resistance and increases the output signal (SIG) value. The sensor's Common Collector Voltage (VCC) and ground (GND) connect to the Arduino through the breadboard.

The following code sample reads the analog value from the sensor and compares it to preset cutoffs for **dry**, **damp**, and **wet** soil using the maximum return value of 1,023 divided into three categories. It begins by setting the library address for the pre-written `LiquidCrystal_I2C` display at `0x27` and initializing it for `16` columns and `2` rows. The sensor is on analog pin#3.

The variable **sensorValue** will contain the new moisture reading and it is compared against cutoff values of **<=341**, **>341 and <682**, or **>=682**. In this case, the LCD receives the appropriate moisture messages before a delay allowing the display to be readable.

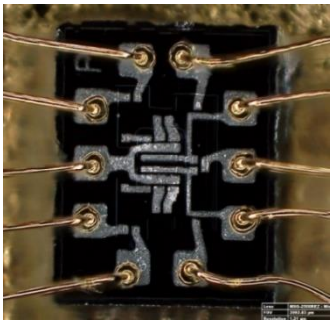
```
#include <LiquidCrystal_I2C.h>
// Initialize the library, analog moisture sensor on input pin A3
LiquidCrystal_I2C lcd(0x27,16,2);
const int analogInPin = A3;
int sensorValue = 0;

void setup() {
lcd.init();
lcd.backlight();
lcd.setCursor(0,0); lcd.print("Soil Moisture");
}
void loop() {
// Read value 0-1023
sensorValue = analogRead(analogInPin);

lcd.setCursor(14,0); lcd.print(sensorValue);
lcd.setCursor(0,1);

if(sensorValue <= 341) lcd.print("Very Dry - Must Water");
if(sensorValue > 341 and sensorValue < 682) lcd.print("Damp, Add Some Water ");
if(sensorValue >= 682) lcd.print("Very Wet, No Water   ");
// wait 300ms so A-D converter can settle
delay(300);
}
```

In addition to displaying messages, the crop could be programmatically watered. Driven by sensors and logic, the tomatoes are watered when the Arduino sends a signal to an analog relay that triggers the solenoid on the top of a standard brass valve. A solenoid has an open or close operation while an actuator uses a motor and gears for intermediate open valve settings. A measured amount of water could be dispensed with a 30-second timer or 3 minutes if dry. To make this an IoT device, it needs to connect through the Internet to a device such as a farmer's PC and provide status on the tomato crop watering. Newer Arduino Uno versions include Wi-Fi, so some extra code would be needed to provide status or act upon the farmer's commands.



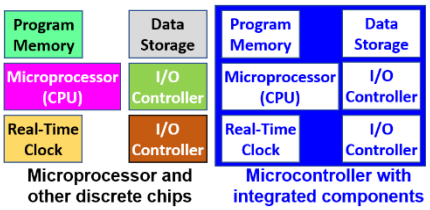
Embedded Processors

Since its inception, we have seen the microprocessor come in various shapes and with different capabilities, instruction sets, costs, and efficiency. Designers mostly use pre-engineered chips to build embedded systems, however, specialty embedded systems such as the Apollo Guidance Computer (AGC) used for the Moon landings were built from scratch and looked like this.[78] The AGC's 2,048 words of magnetic-core memory and 36,864 words of read-only core rope memory had a clock speed of 1 MHz or abo56% of a Radio Shack TRS-80 computer with its 1.78 MHz Zilog **Z-80** processor.



Designers want a firm understanding of the requirements before choosing a processor. They need to know the performance goals, expected cost, voice or video interfaces, support lifetime, MIL-SPEC if necessary, real-time demands, development tools, and more. For instance, if the embedded system is to perform a single simple rigidly defined task, a 4-bit processor might be used instead of a fast multicore chip.

Whether the embedded system needs a **microcontroller** or a **microprocessor** requires analysis. The **microcontroller** on the far right includes a CPU, non-upgradable ROM, RAM, and integrated support for additional peripherals and devices. On the left, **separate support chips** are combined with the **microprocessor**.[79] With a single chip, the **microcontroller** could cost less, require less circuit board space, be easier to source and build, run faster, and so forth. The following covers some of the differences in more detail:[80]



- **Cost** - Engineers design their microcontroller with simpler specific tasks in mind, so it generally costs less than a microprocessor. The **UM66** music chip engineer from our earlier discussion did not have a video interface. Microprocessor designers have general-purpose multitasking requirements where the tasks may not be known. It results in extra circuitry to support all options, leading to more complex and expensive devices.
- **Power** – Handheld battery-operated embedded devices with low clock speeds and low price points need a low-power microcontroller. General-purpose applications can have more robust power supplies, and the power needs of the entire computer can be higher.
- **Speed** – There is often a big difference between microcontroller and microprocessor clock speed. In some embedded devices such as a washing machine, a low-clocked 4 to 8 bit CPU is fine. In a PC, the processor might run Zoom videos or multiple complex tasks on different cores, making speed important, and faster speed is expensive technology.

Engineers choose from this partial list of chips:[81,82]

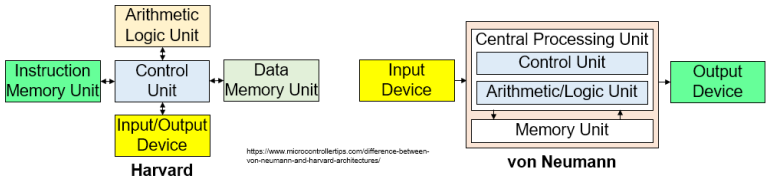
- **ARM** - Advanced RISC Machines are a family of low-cost, low-power 32/64-bit RISC multicore processors that are designed and licensed by ARM Ltd. Billions are licensed from ARM from manufacturers such as Infineon and Microchip.
- **ASIC** - Application-Specific Integrated Circuit is a custom-designed chip for a specific purpose. Popular in computer network switches and barcode readers.
- **ASIP** - Application-Specific Instruction set Processor is an SoC that uses custom instructions to handle a function such as audio signal processing and vision sensors.
- **ASSP** - Application-Specific Standard Parts is a custom pre-defined feature set built for a specific application mass market such as DVD players and digital cameras.
- **CISC** - Complex Instruction Set Computer is a processor with a large set of instructions that can be simple or complex. The CPU typically runs a slower clock speed than RISC, but the instructions tend to do more work.
- **DSP** - Digital Signal Processors are built for fast shift-and-add and multiply-and-add instructions of Fast Fourier Transform devices such as Bluetooth speakers, AirPods, and image and audio processing. Used in the 1978 TI Speak and Spell game.
- **FPGA** - Field-Programmable Gate Array contains supplier-programmed logic blocks.
- **RISC** - Reduced Instruction Set Computer is a CPU with fewer and simpler instructions allowing it to run at a higher clock speed than CISC. There are royalty-free open-source standards such as 32/64-bit RISC-V which is supported by 3,100 members.[83]
- **SoC** - System-on-Chip encapsulates many cores, microcontrollers, graphics, DSPs, and other hardware on one chip. Useful for multiple task-embedded systems or where a complex circuit board design could be handled by a single chip.

The **RISC** versus **CISC** architecture debate is decades old and is based on a lengthy list of pros and cons. **RISC** has simple instructions that minimize the number of cycles to increase performance while **CISC** achieves the same goal with fewer but more complex instructions. In this assembly code multiplication example, **CISC**'s single complex instruction **MULT** multiplies the contents of array **row 2 column 3** by **row 5 column 2**.[84] **RISC** stores values into registers to perform a multiplication. In this code, **LOAD A** with **row 2 column 3**, **LOAD B** with **row 5 column 2**, then multiply with **PROD A** x **B**. Store the result from the **A** register back into array location **row 2 column 3**. **CISC** and **RISC** produce the same results, but **RISC** is usually faster.

```
CISC                RISC
MULT 2:3, 5:2       LOAD A, 2:3
                    LOAD B, 5:2
                    PROD A, B
                    STORE 2:3, A
```

The embedded **RISC** CPU tends to favor a Harvard architecture and general-purpose **CISC** processors such as Intel's X86 use Von Neumann's architecture. It can come down to the efficiency of I/O devices storing data in the CPU's memory address space alongside instructions and other data or segregating the I/O address space from traditional memory locations.[85]

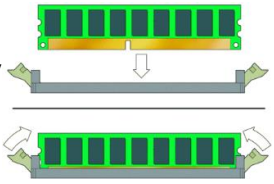
In the soil moisture discussion, **ATmega328**'s Harvard design separates the program from the data. The Von Neumann architecture from CS pioneer John von Neumann in 1945 combines the Control Unit, Arithmetic and Logic Unit (ALU), and a single Memory Unit.[86] This table summarizes other differences. Both have evolved over the years and the lines between them have blurred. **RISC** still has fixed-length instructions to simplify timing, and separates load and store instructions and register-to-register operations, while the **CISC** decoder is optimized to spot, for example, independent instructions that can be run in parallel. Usually, **RISC** vs **CISC** comes down to the target cost followed by complexity.



Harvard Architecture	von Neumann Architecture
Separate code and data memory	Shared code and data memory
Single clock cycle to retrieve both code and data	Processor fetches code in a clock cycle and data in another cycle
Slower speed, more time-consuming	Higher speed, less time consuming
Complex design, more expensive	Simple design, cheaper

https://www.tutorialspoint.com/embedded_systems/es_architectures.htm

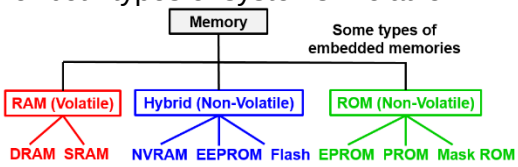
Embedded System Memory



In a PC, the primary memory is directly addressable by the processor and it usually resides on upgradable socketed DIMM strips. Secondary memory is external to the processor and needs an I/O controller to send commands to the SSD to read or write data. In an embedded system, memory is generally not upgradable and is either contained in the microcontroller or often soldered to the

main circuit board, although some embedded devices like computer printers allow for a user addon DIMM. Some embedded devices also have external storage as in the case of MP3 or disk players which can have small hard or SSD drives, or CD/DVD drives.

The volatile versus non-volatile memory categories exist for both types of systems. Volatile memory requires constant power and non-volatile (secondary memory) retains its data when the power is removed. Primary memory is generally volatile.[87]



DRAM Primary Dynamic Random-Access Memory does not retain data without power and must be refreshed by a controller every 64 milliseconds.[88]

SRAM Primary Static Random-Access Memory is faster and more expensive than **DRAM**, it retains data as long as there is power.

NVRAM Like **SRAM**, Non-Volatile Random Access Memory does not lose data when power is lost. It is expensive but fast. Engineers use this when startup time is crucial.

EEPROM Electrically Erasable Programmable Read-Only Memory that is electrically programmed and erased.

Flash A type of EEPROM that is electrically programmed and erased (fewer erase cycles than EEPROM) and is common in embedded systems.

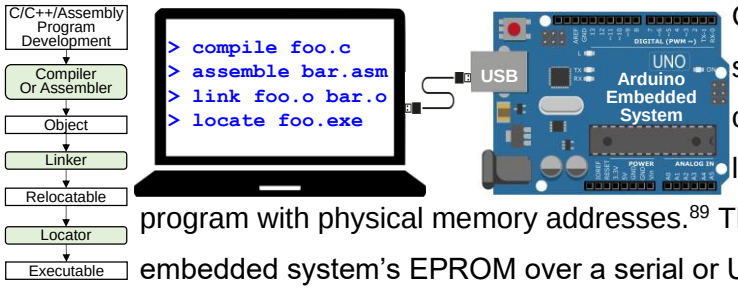
EPROM Erasable Programmable Read-Only Memory can be electrically programmed but is erased only by ultraviolet light.

PROM Programmable Read-Only Memory can be written once with a special device.

Mask ROM Read-Only non-volatile Memory is low-cost but cannot be changed once written at the factory.

Programmable Embedded Systems

At a high level, embedded processors, like other CPUs, only run machine code. Programming languages like C are for programmer convenience. A program that is written in a programming language must be translated into CPU executable binary code. Code translation can be performed by the IDE toolset accompanying a processor. The process is iterative, performed on a host PC, and debugged so the final program is logical and error-free.



Code flows through these various steps. The program is compiled to create an object file. All object files are linked to create a single relocatable program with physical memory addresses.[89] The executable binary is loaded into the embedded system's EPROM over a serial or USB cable, or Ethernet/Wi-Fi communications. The device is then reset or started to see if the code works.

Cybersecurity and Embedded Systems

One of the great things about embedded systems is they quietly go about their job and we hardly know they are there. That is how the story started in Iran in 2010. They used centrifuges based on a stolen Dutch design at their Natanz plant to enrich uranium for nuclear weapons by spinning it to separate the unwanted isotope U-238 from the desired and lighter U-235.

Stuxnet allegedly began in 2005 as a US-Israel offensive computer worm for Operation Olympic Games to disable a part of Iran's nuclear capability.[90] Centrifuges are delicate and designed to spin at a certain rate. If Stuxnet was running on a Siemens embedded system, the worm rapidly spun the centrifuge up and down to damage or destroy it.[91] The display also falsely showed it was running fine. A Dutch spy posing as a worker installed Stuxnet on Iranian systems resulting in 1,000 of the 5,000 centrifuges being damaged or destroyed.[92]

Cybersecurity is important for embedded systems. Manipulated code could cause a malfunction. Embedded systems in a car, ATM, municipal water plant industrial control, or other devices must have safeguards. Many embedded systems are designed around low cost and function, limiting memory and processor performance to the task at hand. A redesign could be required if security was not part of the original design.

Hackers do not target your Keurig. They look for high-value or safety-critical upgradable targets, whether it is the firmware, application, or device entry points such as the Wi-Fi in your baby's security camera. No one will use a network printer that sends copies of their documents to a ransomware gang, nor would they own a car knowing a thief can disable its security.

A byproduct of more robust embedded systems is that more code and circuitry can create new attack vectors, so sensors, actuators, firmware, Bluetooth, and so forth, need to be checked. User data and passwords must be protected by making sure it adheres to security standards.

Developers need firmware upgrade safeguards, support third-party monitoring tools, encrypt data, use a secure boot to verify the correct software is running, and restrict update access. Devices should have user validation, test that sensor data is within specification, and include a ROM security certificate check. Secure coding should be used to stop criminals from accessing the code, network, or device. They may need "beefier" hardware to run extra layers of security.

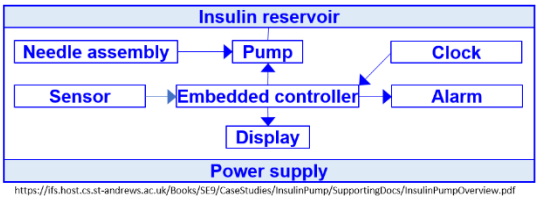
Sometimes it is as simple as automatically checking that the development tools are at the latest revision and patches are in place. That is especially true of IDEs, which run on a PC – if the IDE is vulnerable, how safe is the code it creates?

Medical and Safety Embedded Systems

Insulin pumps

Diabetes is a relatively common condition where the human pancreas is unable to produce sufficient quantities of a hormone called insulin. The conventional treatment of diabetes involves regular injections of genetically engineered insulin. Diabetes affects 10% of the US population and 537 million adults worldwide.[93] There are two types of diabetes. Type 1 is when the body cannot make insulin and must be given insulin every day to survive. Type 2 means the body does not use insulin well but diabetes can be controlled through other means.

Someone with type 1 had to manually monitor blood glucose and inject themselves with insulin during the day. With a battery-powered embedded system insulin pump, glucose levels are monitored continuously through a sensor and a dose from an insulin cartridge is automatically injected into the patient to maintain levels.[94] This device has greatly improved their quality of life.

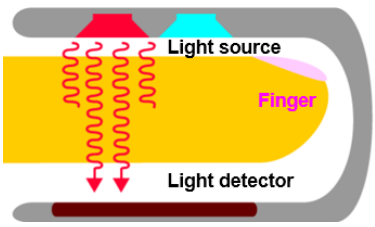


https://ifs.host.cs.st-andrews.ac.uk/Books/SE9/CaseStudies/InsulinPump/SupportingDocs/InsulinPumpOverview.pdf

There are many types of embedded system insulin pumps on the market at reasonable prices. Inside the device, controller circuitry interfaces with a mechanical pump to manage the medication using algorithms. The embedded hardware needed to sense the blood's conductivity is part of this Unified Modeling Language diagram that shows how the device functions. An older OmniPod insulin pump uses multiple embedded processors to do the work, such as a powerful Freescale **MC9328MX21S** 266 MHz ARM processor and other supporting chips to operate the control software. The pump also includes Bluetooth communications to generate reports and notifications.

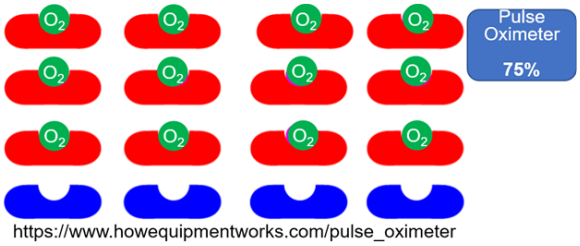
Pulse Oximeter

During COVID, many with symptoms used a non-invasive pulse oximeter to estimate the level of oxygen in their blood. A level below 95% indicated they should seek medical attention. It is a small inexpensive embedded system that clips onto a finger.[95]



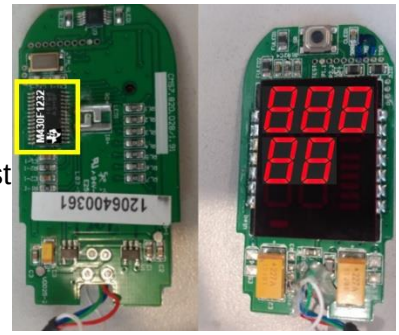
Medical science established that oxygenated hemoglobin allows **red light** to pass through while it absorbs **infrared light**, and deoxygenated hemoglobin absorbs **red light** and allows **infrared light** to pass through it. An oximeter's **660 nm red LED** and **940 nm infrared LED** transmit light through the **finger** and a sensor on the other side of that finger receives the wavelengths. An embedded processor analyzes the reflected or absorbed wavelengths to estimate if the blood has sufficient oxygen. [96]

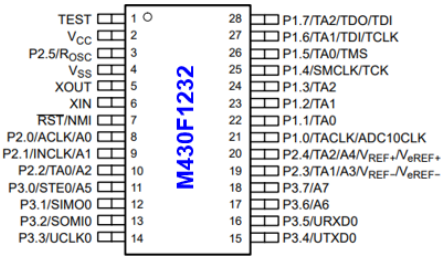
Essentially, if the hemoglobin has oxygen, it is reported against the count of detected hemoglobin. This shows the patient with just 75% oxygen saturation, meaning they should immediately seek medical aid.



https://www.howequipmentworks.com/pulse_oximeter

Inside a typical battery-operated oximeter is a circuit board such as this one.[97] To the right is an LED display similar to the large one the digital clock used earlier. On the left is its low-cost 16-bit TI **M430F1232** RISC processor.[98] The chip's functionality includes an Analog/Digital converter, a power-saving battery mode, memory, and a Universal Asynchronous Receiver-Transmitter (UART) to communicate with other systems.



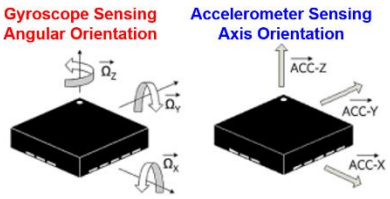


The Elderly and Falling

"Don't get old" is a word of advice you may hear from senior citizens. The truth is, getting old has its set of concerns. As a toddler, falling is part of learning to walk. For a senior who falls, it can be difficult or impossible to get up or pose a severe risk of injury requiring immediate medical attention. In 2021, 37 million seniors fell worldwide and required help.[99]

Fall detection through a monitoring service can alert a designated contact and call emergency services. Apple's sophisticated watch is part of a general-purpose system that can detect a fall, use haptic vibration to check on consciousness, inform emergency contacts, and provide an emergency response. A simple dedicated medical alert-embedded device can alternatively be worn on a lanyard and triggered by pressing a big (**red**) button or by the device sensing the fall. With a low price point and simple UI, it plays a crucial role in the elderly receiving assistance.

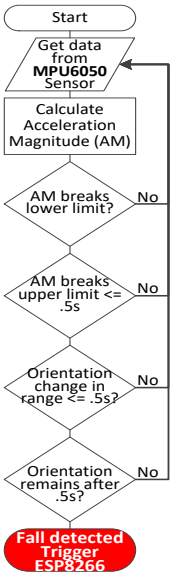
The device can have **gyroscope** and **accelerometer** sensors that supply data to a fall determination algorithm. An **accelerometer** detects motion in any direction, and a **gyroscope** measures rotation on these axes. These sensors are used for other location functions in a Roomba and smartphone. The XYZ values reflect the position and acceleration of the wearer. The software checks for sudden changes which could be indicative of a fall.



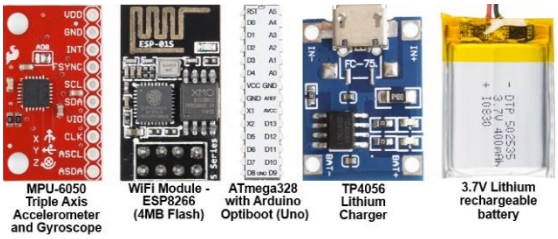
If the wearer presses the red button or the algorithm finds gyroscope and accelerometer values in an "alert range," the logic to call for help takes over. Devices can have the following attributes:

- **Precision** – Determining a fallen wearer is tricky. The algorithm must separate non-events such as a plunging roller coaster ride, walking down the stairs, or riding an elevator from serious events such as falling out of bed or slipping down the stairs. The wearer can always press an alert button.
- **Urgent response** – With a detected fall, a prerecorded call is made with a retry feature until responders acknowledge the situation. A responder may call the wearer to check on them. Greater complexity raises the cost by needing more circuitry, code, and testing.
- **Battery power** – The device must use a battery since a fall can happen anywhere.
- **Ease of use** – The UI must be simple since its user community can be challenged by complexity, or an injury can cause them to be incapacitated. That is why you do not rely on them to dial an emergency phone number.
- **Waterproof** – Emergencies can happen in a slippery bathtub or swimming pool. Making a device waterproof adds extra cost, weight, and complexity.
- **User reset** - If the wearer accidentally presses the alert, they could de-escalate the issue and perhaps speak with a responder to assure them that they are fine.

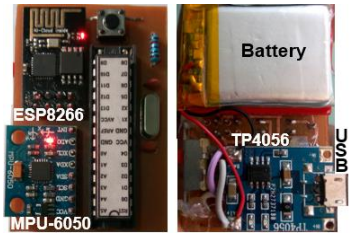
An alerting device was built by an enterprising hobbyist who constructs embedded circuits. Following this flowchart, they made a small unit and crafted code that would sample accelerometer and gyroscope values from a 6-function **MPU-6050** (the source code can be found in this footnote.)[100] The premise is that during a fall, there is an instant period of freefall or deacceleration, followed by an acceleration burst, and finally an orientation change.



The algorithm eliminates some of the false positive and negative readings to obtain higher levels of accuracy. When conditions are in the alert range, the code issues the "**ALERT**" SMS and email command through the **ESP8266** Wi-Fi module that is sent to a predetermined cell phone. The device is packaged for wearing around the neck and has a single-press alert button.



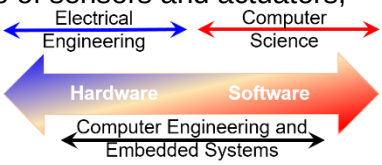
It is built from these five components. Adding more functionality means adding circuitry, coding, and cost.



Careers

When I was younger, I dreamed about getting a CS degree and finding a job doing application programming. These days, the scope of many college CS departments has broadened to

include embedded hardware and software system skills, knowledge of sensors and actuators, and electronics.[101] My undergraduate university now has a combined Electrical and Computer Engineering Department and offers degrees with classes that are taken from both disciplines.[102]



An embedded systems career is ideal for the curious engineer who likes advanced hardware, software, and electrical engineering, and wants to design and develop technology that helps society. As AI, virtual reality, deep learning, and more are added to embedded systems, this field should grow even faster. Embedded systems mean different things to different parts of the industry, and the skills that are needed are also varied. This table shows some of the applications, their expertise, and some of the challenges with that segment.

Embedded System Area	Expertise	Description
Mining and marine	Marine environment, material engineering	The marine environment is harsh with saltwater corrosion and big temperature swings
Transportation controllers	Roadway/railway plans, traffic light standards	Transportation systems must follow rules and regulations of each systems
Handheld devices	Design, ergonomics	User intefaces, fonts, and handheld packaaging are part of industrial design contraints
Automotive devices	SAE standards, mechanical design, thermal mgmt.	Devices undergo vibrations and temperature, and industry standards must be followed
IoT and Drones	Device security, signal conditioning	Secure systems are important to fend off hackers
Robotics	Math, control systems	Math, mapping, path planning, and control system skills are necessary
AI	Computer vision	Computer vision is very popular in vehicle, robotics, and more
Monitoring systems	Networks, security, control systems	From security to agriculture, sensors play a big role

From a CS perspective, embedded engineers have rewarding careers. Systems remain in our daily life as technology costs drop and we invent new devices. For instance, an enormous amount of hardware and software goes into a modern vehicle, and an engineer must design a hardware and software solution that withstands harsh environments while following regulations.

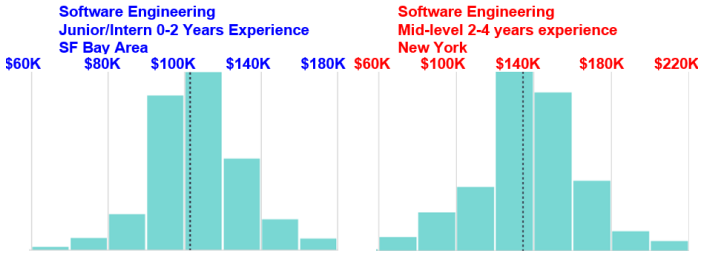
Every powered device seems to have an embedded system. These are not just simple devices like calculators, but sophisticated equipment in home robotics, office, medical science, military, and more. Whether in school, through self-study, or other forms of instruction, the field has a minimal level of individual and team skills you will need from these broad categories:[103]

- Design and implement embedded C and C++
- Interfacing peripherals, compilers, vision control
- Assemblers to convert code and libraries
- Embedded hardware and electronics schematics
- Verbal communication and problem-solving
- Device drivers, firmware coding
- Documentation, schematic, and circuit design
- Instructor training preparation
- Linux and other RTOS design
- Specification design systems engineering
- Computer architecture
- Battery and power management

Careers could require skills that you already have and others you must learn. For instance, robotics requires geometry and advanced engineering skills such as control systems. Some jobs can immerse you in high-speed circuits, advanced algorithms, and high-performance computing, so you will likely need additional instruction as your career and technology progress. There are also non-technical skills that you picked up along the way, such as verbal communication, writing, research, time management, problem-solving, and more.

Many CS majors write their code for X86 platforms, so it can be both fun and frightening when they work on their first embedded system. Their first task requires research to determine which of the hundreds of microprocessors, microcontrollers, and sensor components to use. Some students get an early jump on the design effort through inexpensive embedded system Arduino kits. The more familiar you are with them, the better prepared you are for an embedded systems career. Try to do some fun projects around the house such as home automation or security.

Hired.com has a salary report for embedded software engineers based on location, organization size, industry, and more.[104] You should explore colleges with internship programs. A Google search for "embedded systems courses" will show some are free and others offer college credits. Get experience and certifications in subjects such as computer networks, security, and Linux.[105] Take courses such as these:



C, C++ for Embedded	Microwave Engineering	Embedded Systems	Electrical Codes and Regulations	Communications Systems	Malware Reverse Engineering	Autonomous Systems
Power Electronics	Fields & Waves	Antennas	Basics RF	Computer Design	Neural Networks	Microprocessors
Control Systems	Circuit Analysis	Logic Design	Power Systems	Linear Systems	Cybersecurity	Digital Forensics

You want to learn about different embedded system architectures and features/brands of microprocessors/microcontrollers. In addition to a mastery of interrupts and memory systems, it helps to know how to interface peripherals, use version control, IDEs for writing code, compilers/assemblers, debuggers, libraries, and simulators.

You might get on-the-job training. For example, a new robotic system could require new math skills and knowledge of its control systems. You might get to work on a new greenhouse controller which takes into account many disciplines. Computer vision, lidar, radar, sonar, and AI are critical for applications like autonomous driving.

Some of the many exciting jobs in the embedded system profession include:

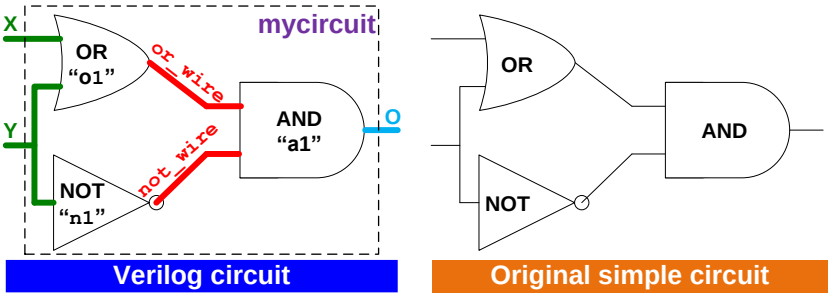
- **Microcontroller Firmware Engineer** who codes microcontrollers,
- **Embedded Linux Engineer** who handles real-time tasks of complex systems,
- **Embedded Applications Engineer** since embedded systems need IoT applications,
- **Embedded Network Engineer** who interfaces devices with higher-level systems,
- **Cybersecurity Embedded Developer** since security is a must-have in every device.

Writing an embedded system program is different than writing a typical user application. The environment is often resource-constrained, which is both a blessing and a curse. In application programming, running out of memory is likely an error while in embedded applications, limited

memory is a planning task. Embedded system code bugs are less forgiving than in application programming, especially after a program is burned into a ROM installed in thousands of devices. Essentially, toaster ovens and thermostats should not “blue screen” since they lack upgradable code, or worse, rebooting a car at 65 mph to clear a coding error could be deadly.

Careers in embedded design tend to offer more job security than general-purpose application programming because of the focused knowledge you acquire and how it puts your skills in demand. For example, many use an abstracted simulation Hardware Description Language such as Verilog in the embedded system design phase.[106] Created in 1985, Verilog uses C-like constructs such as **`if/else`**, **`for`**, **`while`**, **`case`**, and so forth, to describe a circuit. It is an international standard for creating modules that communicate with other modules using defined inputs and outputs. Once the design is finalized and debugged, it is compiled, not into a program, but into a description language to build the IC at a “silicon foundry.”[107] The language describes how modules interconnect allowing the foundry to design silicon masks to build the chip on a silicon wafer.

For example, assume we needed a chip that could perform a specialized design. You would start with a small piece of it – **a simple logic circuit as shown to the far right**.[108]



Using a tool such as Verilog, a complex circuit is broken into **`module`**s, and an engineer uses the tool to textually describe how the circuit works. The first step is to thoroughly **label the diagram to the left** beginning with the name of the **`module`**, such as **`mycircuit`**. The name is repeated in the descriptor language for the **`endmodule // mycircuit`**. The **`module`** needs defined outputs and inputs such as **`(o, x, y)`**. Next, use Verilog primitives for **`and`**, **`or`** and **`not`**. We then have three statements to describe the primitives. **`OR`** labeled **`o1`** uses the **`or_wire`** for output and its two inputs labeled **`x`** and **`y`**. **`not n1`** uses the **`not_wire`** for output and its sole input is **`y`**. **`and a1`** whose output is **`o`** and uses **`or_wire`** and **`not_wire`** for inputs.

```
module mycircuit (o, x, y);
  output o;
  input x, y;
  wire or_wire, not_wire;

  or o1 (or_wire, x, y);
  not n1 (not_wire, y);
  and a1 (o, or_wire, not_wire);

endmodule // mycircuit
```

Conclusion

In our digital world, we are surrounded by embedded systems. Their contribution to society has been enormous. They greet us in the morning, help us during the day, and even keep us secure at night. We rely on them to alert us to our COVID symptoms, grow our food, keep our car going straight in a rainstorm, assist people with disabilities, and augment our children's education. These systems run nonstop with minimal human intervention, and it is all due to the miracle of the microchip.

The saying goes, "You ain't seen nothing yet." The advances of low-cost high-performance processors have been amazing as devices remain affordable yet get smarter as they tackle more advanced algorithms. The promise of future intelligent transportation, remote patient healthcare, and smart homes is great for society. Some of this is attributable to technology such as extreme ultraviolet lithography which helps make the low-cost chips even smaller and faster. New architectures are more complex, with deeper hardware pipelines, more levels of cache, multiple cores, and faster clock speeds - Gordon Moore must be smiling at his analysis.

The future may bring us a time when every device has an embedded processor and sensor of some type, all wirelessly communicating on our behalf and helping usher in an era of ubiquitous computing. If it is done right, these intelligent devices will improve our quality of life.

I have barely scratched the surface of embedded systems, but I hope it has encouraged you to do some further exploration of this hidden world.

Footnotes

[1] https://www.alliedmarketresearch.com/embedded-processor-market
[2] https://finance.yahoo.com/news/embedded-processor-market-size-projected-192000075.html
[3] https://www.tirebusiness.com/aligning-adas/automotive-software-market-hit-401b-2027
[4] https://www.pcmag.com/news/despite-chip-shortage-2021-was-a-surprisingly-good-year-for-pc-shipments
[5] https://www.design-reuse.com/news/51679/mcu-market-history-and-forecast.html
[6] https://www.census.gov/quickfacts/fact/table/US/HSD410220
[7] https://www.techspot.com/news/94208-integrated-circuit-shipments-forecast-reach-nearly-430-billion.html
[8] https://enviremental b.com/melody-generator-circuit/
[9] https://www.datasheetq.com/datasheet-download/472693/1/UMC/M66T11
[10] http://www.righto.com/2021/12/reverse-engineering-tiny-1980s-chip.html
[11] https://en.wikipedia.org/wiki/Image_scaling
[12] https://blog.bestbuy.ca/tv-audio/everything-you-need-to-know-about-tv-upscaling
[13] https://en.tab-tv.com/?p=22193
[14] https://www.mediatek.com/products/digital-tv/s900-mt9950
[15] http://www.historyofmicrowave.com/
[16] https://www.lifewire.com/smart-microwave-4159823
[17] https://en.wikipedia.org/wiki/Keurig
[18] https://www.keurig.com/content/k-supreme-plus-smart-coffee-maker
[19] https://youtu.be/32RR9PAToao
[20] https://www.homegrounds.co/how-to-make-iced-coffee-with-keurig/
[21] https://manualzz.com/doc/22577465/how-a-keurig-coffee-maker-works-keurig-mechanical-schematic
[22] https://keurig.2018.cctp506.georgetown.domains/wp-content/uploads/2018/05/Algorithm-Design-2.pdf
[23] https://coffee.germar.net/?p=74
[24] https://hackaday.io/project/7577-keurig-hack/log/24792-teardown
[25] https://m.dzsc.com/product/infomation/44122/20125118303604.html
[26] http://www.datasheet-pdf.com/PDF/PIC16F913-Datasheet-MicrochipTechnology-520695
[27] https://www.drivespark.com/four-wheelers/2018/abs-anti-lock-braking-system-history-facts-details-first-car-mercedes-026266.html
[28] “Computers as Components”, by Wayne Wolf, ISBN 978-0-12-374397-8, P. 4
[29] https://en.wikipedia.org/wiki/Direct-shift_gearbox
[30] https://silo.tips/download/innovative-technologies-for-transmission-control-units
[31] https://www.tirebusiness.com/aligning-adas/automotive-software-market-hit-401b-2027
[32] https://www.alliedmarketresearch.com/automotive-integrated-circuit-ics-market
[33] https://www.electricaltechnology.org/2021/12/difference-between-thermistor-thermocouple.html
[34] https://en.wikipedia.org/wiki/C_(programming_language)
[35] https://blog.felgo.com/embedded/embedded-systems-programming
[36] https://www.qt.io/embedded-development-talk/embedded-software-programming-languages-pros-cons-and-comparisons-of-popular-languages
[37] https://www.mathworks.com/discovery/misra-c.html
[38] https://embeddedcomputing.com/technology/software-and-os/how-to-choose-an-embedded-operating-system
[39] https://github.com/topics/embedded-linux
[40] https://www.windriver.com/
[41] https://blackberry.qnx.com/en
[42] https://skill-lync.com/blogs/most-popular-programming-languages-for-embedded-systems
[43] https://www.geeksforgeeks.org/difference-between-linux-and-qnx/
[44] https://americanhistory.si.edu/collections/search/object/nmah_1329686
[45] https://smartasset.com/investing/inflation-calculator
[46] http://www.datamath.org/Desktop/ti-1795.htm
[47] https://www.youtube.com/watch?v=1SILRTtJ24Q
[48] https://www.quora.com/What-is-the-name-of-the-processor-used-in-an-electronic-calculator
[49] https://www.quora.com/What-is-the-cpu-of-a-calculator-enclosed-in-Bit-which-resembles-a-lump
[50] https://www.metv.com/stories/a-history-of-the-television-remote-control-as-told-through-its-advertising
[51] https://www.khanacademy.org/science/electrical-engineering/reverse-engin/dvd-player/v/what-is-inside-of-a-universal-remote-control-1
[52] https://en.wikipedia.org/wiki/Santoprene
[53] https://www.youtube.com/watch?v=LAz9YC7reWs
[54] http://www.datasheet-pdf.com/PDF/ADAM27P08-Datasheet-ABOV-935749

[55] Sony ICF-C112 FM/AM Clock Radio Instructions
[56] https://makingstudio.blog/2019/09/17/sony-clock-teardown/
[57] Note: sending the LED a 0 or 1 to turn on the LED element will depend on if it has a common anode or cathode.
[58] https://onelec.ru/uploads/product_pdf/8397/LM8560.pdf
[59] https://www.radiomuseum.org/tubes/tube_cxa1019s.html
[60] https://kids.kiddle.co/Embedded_system
[61] https://www.downloads.netgear.com/files/dg632_install_guide.pdf
[62] https://nvd.nist.gov/vuln/detail/CVE-2009-2257#
[63] https://www.explainthatstuff.com/how-roomba-works.html
[64] https://www.techrepublic.com/article/cracking-open-the-roomba-980-robot-vacuum/
[65] https://www.cnet.com/home/kitchen-and-household/appliance-science-how-robotic-vacuums-navigate/
[66] https://www.irobot.com/
[67] https://groups.csail.mit.edu/drl/courses/cs54-2001s/odometry.html
[68] https://community.element14.com/products/devtools/technicallibrary/m/files/7303
[69] https://www.nxp.com/docs/en/data-sheet/LPC3220_30_40_50.pdf
[70] https://www.st.com/resource/en/datasheet/stm8s207r8.pdf
[71] https://www.router-reset.com/info/AzureWave/AW-CU288
[72] https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf
[73] https://www.samsung.com/us/home-appliances/refrigerators/3-door-french-door/25-1-cu--ft--3-door-french-door-refrigerator-with-family-hub--in-stainless-steel-rf262beaesr-aa
[74] https://covingtonnaturals.com/blogs/news/how-to-raise-brix-levels-in-plants
[75] https://pdf1.alldatasheet.com/datasheet-pdf/view/1132281/ATMEL/ATMEGA328P.html
[76] https://cdn-shop.adafruit.com/product-files/181/p181.pdf
[77] https://www.sparkfun.com/products/13637
[78] https://en.wikipedia.org/wiki/Apollo_Guidance_Computer
[79] Embedded Systems Design: An Introduction to Processes, Tools, and Techniques, by Arnold S. Berger, ISBN 1578200733, P. 34
[80] https://www.eletimes.com/microprocessor-vs-microcontroller-what-is-the-difference
[81] https://www.gartner.com/en/information-technology/glossary
[82] https://www.pcmag.com/encyclopedia
[83] https://riscv.org/about/
[84] https://iq.opengenus.org/risc-vs-cisc-architecture
[85] https://www.sciencedirect.com/topics/engineering/harvard-architecture
[86] https://www.microcontrollertips.com/difference-between-von-neumann-and-harvard-architectures/
[87] https://barrgroup.com/embedded-systems/how-to/memory-types-ram-rom-flash
[88] https://www.technipages.com/what-is-a-refresh-cycle
[89] “Programming Embedded Systems, Second Edition with C and GNU Development Tools” by Michael Barr and Anthony Massa, ISBN 978-0-596-00983-0, P. 58
[90] https://en.wikipedia.org/wiki/Operation_Olympic_Games
[91] https://etd.auburn.edu/bitstream/handle/10415/4889/Secure Design Considerations for Embedded Systems_ETD_fixes.pdf
[92] https://www.timesofisrael.com/dutch-mole-planted-infamous-stuxnet-virus-in-iran-nuclear-site-report/
[93] https://idf.org/aboutdiabetes/what-is-diabetes/facts-figures.html
[94] https://arxiv.org/ftp/arxiv/papers/1709/1709.06026.pdf
[95] https://www.amazon.com/Facelake-FL400-Oximeter-Carrying-Batteries/dp/B0117V8Q2O
[96] https://www.howequipmentworks.com/pulse_oximeter
[97] https://giddi.net/posts/commercial-pulse-oximeter-teardown/
[98] https://www.ti.com/lit/ds/symlink/msp430f1232.pdf?ts=1670415714633
[99] https://www.who.int/news-room/fact-sheets/detail/falls
[100] https://www.instructables.com/Emergency-Fall-Notifier-Cum-Panic-Button/
[101] https://www.electronicsforu.com/career/guide-to-career-in-embedded-systems
[102] https://ece.fiu.edu/academics/undergraduate/bs-computer-electrical-engineering/index.html
[103] https://hired.com/job-roles/embedded-engineer
[104] https://hired.com/salary-calculator/software-engineering/new-york?yoe=2-4
[105] https://www.comptia.org/certifications
[106] https://www.verilog.com/
[107] Embedded Systems Design: An Introduction to Processes, Tools, and Techniques, by Arnold S. Berger, ISBN 1578200733, P. 55
[108] https://youtu.be/q1QwC3YlHG0