

Distributed Computer Systems Engineering

CIS 508: Lecture 7
Embedded Systems

Lecturer: Sid C.K. Chau
Email: ckchau@masdar.ac.ae



What is Embedded System

Embedded system is a computer inside a product

- Computer system as part of some other equipment
- Typically dedicated software (may be user-customizable)
- Often replacing previously electromechanical components
- Often no limited input interface or display device
- In many products: household appliances, vehicles, machines

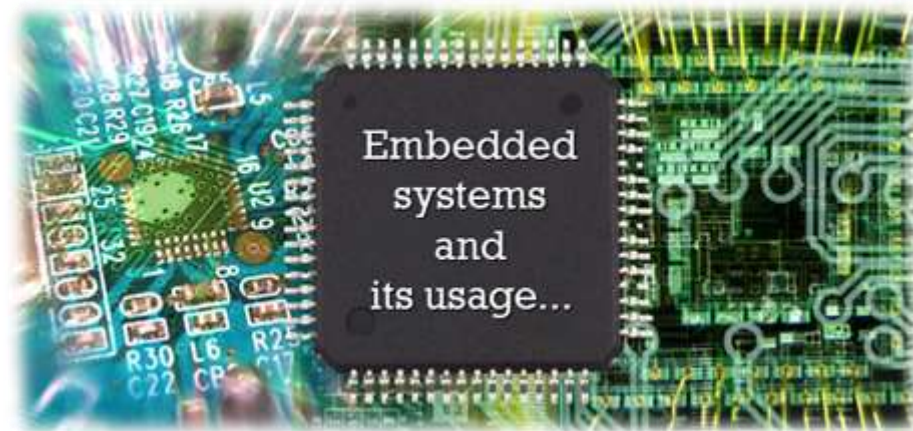


Need for Embedded Systems

- Providing pervasive computational intelligence
- Appliance, equipment, facilities are getting ever more intelligent
 - Predicting your need
 - Optimizing consumption, performance
 - Coordinate with each other
- ***Internet-of-things vision***
 - Every physical object has a virtual counter-part in Internet
 - Every object needs a computational system built-in
 - Interconnecting simple machine intelligence to achieve complex social computational behavior

Examples of Embedded Systems

- Consumer electronics, such as MP3 audio, digital camera, home appliance
 - I/O, product functioning, infotainment
- Vehicles, aircraft, machinery
 - Control, automation, optimization
- Medical equipment
 - Sensing, analysis, monitoring
- Toys, robots, vacuum cleaners
 - Computational intelligence



Evolution of Embedded Systems

- Early era: purposely built chips and boards
 - Integrating mostly analog components
 - Hardwired computation logic
- Recent: rise of cheap general purpose microprocessors
 - Programmable computation logic
 - On-board memory
 - Networking and communication capabilities
- Two major types:
 - FPGA (Field-programmable Gate Array)
 - Programmable at logic gate level
 - Microprocessors
 - Programmable by programming languages (Assembly, C, Java)

Properties of Embedded Systems

Embedded systems integrating with other systems, and must be dependable:

- **Reliability**: probability of system working correctly provided that it was working at the beginning
- **Maintainability**: probability of system working correctly after error occurred
- **Availability**: probability of system working at current time
- **Safety**: no harm to be caused
- **Security**: confidential and authentic communication

Properties of Embedded Systems

- Embedded systems have small footprint and must be efficient:
 - Energy efficient
 - Code-size efficient (to be implementable in a chip)
 - Run-time efficient
 - Weight efficient
 - Cost efficient
- Specific to a certain application:
 - Knowledge about behaviour at design time can be used to minimize resources and to maximize robustness
- Specific user interface (no mouse, keyboard and screen)

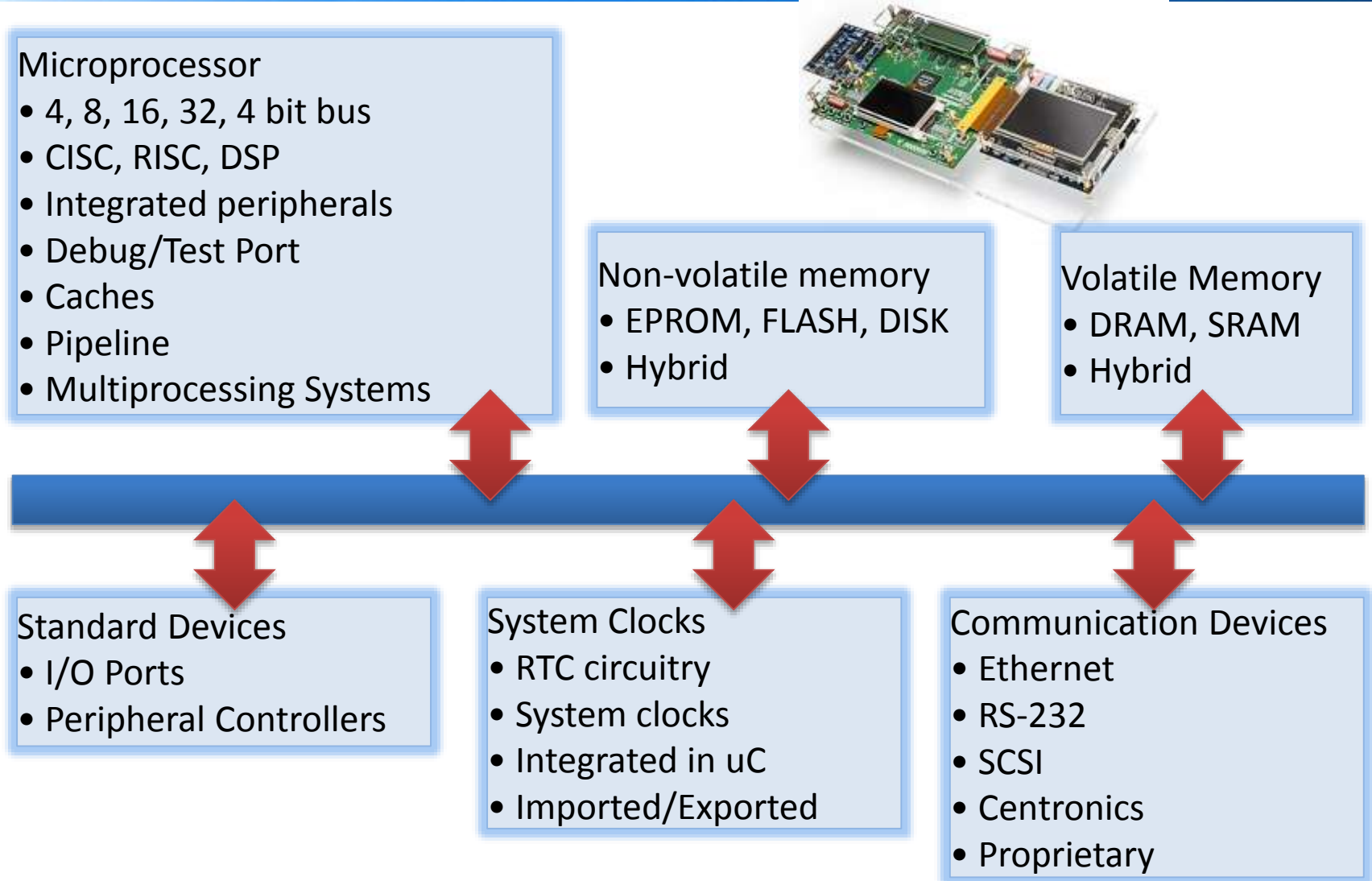
Properties of Embedded Systems

- Embedded systems are meant to be front-end, and must meet real-time constraints:
 - A real-time system must react to stimuli from the controlled object (or the operator) within the time interval dictated by the environment
 - For real-time systems, correct answers arriving too late are wrong
- Frequently connected to physical environment through sensors and actuators,
 - Hybrid systems (analog + digital parts)
 - Embedded systems are reactive systems, in continual interaction with environment and executes at a pace determined by environment

Comparisons

- *Embedded Systems*
 - Limited applications that are implemented at design-time.
 - Not programmable by end user
 - Fixed run-time requirements (additional computing power not useful)
 - Criteria: cost, power consumption, predictability
- *General Purpose Computing Systems*
 - Broad class of applications
 - Programmable by end user
 - Faster is better
 - Criteria: cost, average speed

Typical Architecture



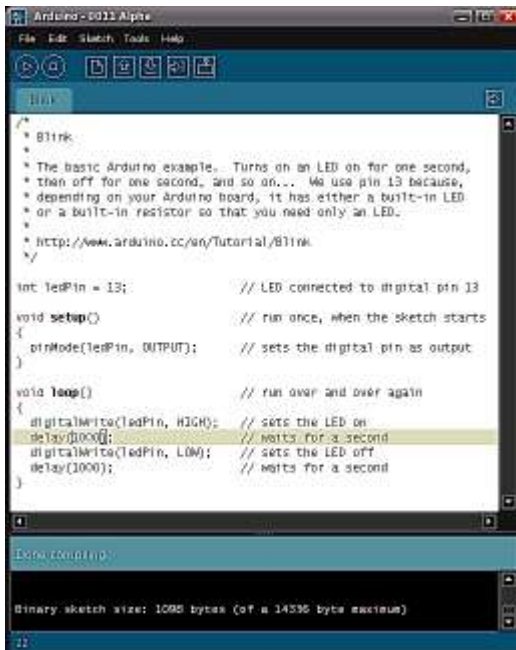
Future of Embedded Systems

- Embedded systems overtook market of PCs
- Ubiquitous and pervasive computing:
 - Information anytime, anywhere; building ambient intelligence into our environment; internet of things:
 - Wearable computers
 - “Smart Labels” on consumer products
 - Intelligent buildings
 - Environmental Monitoring
 - Traffic control and communicating automobiles
- Embedded systems provide the basic technology

Trends

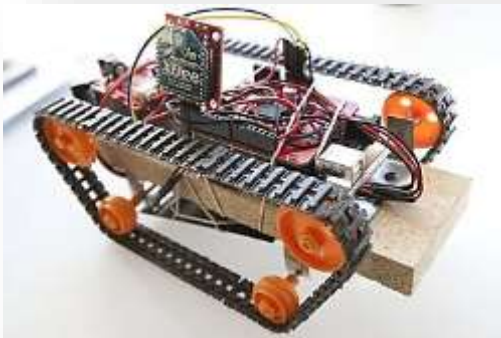
- Higher degree of integration on a single chip:
 - Memory + processor + I/O-units + (wireless) communication
 - Network on chip for communication between units
 - Multiprocessor Systems on a Chip (MPSoC)
- Software increasing (amount and complexity)
- Hardware/software co-design gets increasing importance
- Low power constraints (portable or unattended devices)
- Communicating embedded systems, very often wireless

Arduino



- Highly extensible open embedded systems platform
- Supported by open source, open design community
- Hardware consists of a simple open source hardware board designed around an 8-bit Atmel AVR microcontroller, up to a 32-bit Atmel ARM
- Software consists of a standard programming language compiler and a boot loader that executes on the microcontroller

Arduino



- Arduino boards can be purchased pre-assembled or do-it-yourself kits
- Hardware design information is available for those who would like to assemble an Arduino by hand
- Many potential applications
 - Smart plug
 - Sensors
 - Home automation
 - Robotics
 - Smart phone accessories
 - Automobiles
 - Arts and designs

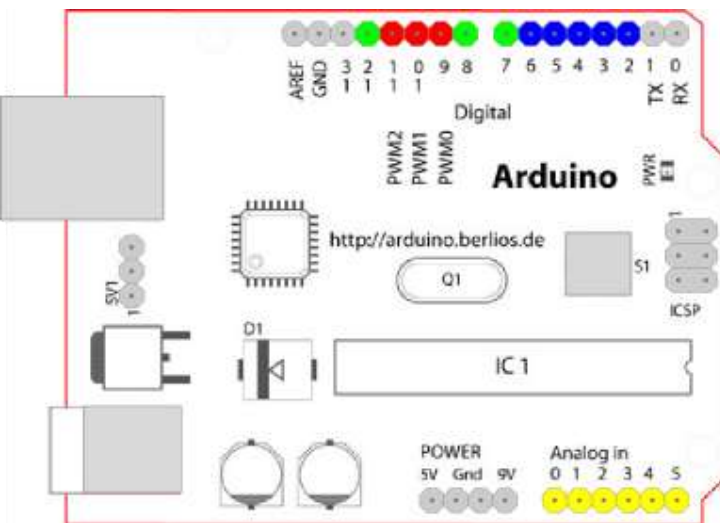


Arduino Capabilities

- 16 kBytes of Flash program memory
- 1 kByte of RAM
- 16 MHz processor
 - Apple II: 1 MHz / Intel 286: 12.5 MHz / Intel Core 2: 3 GHz
- Inputs and outputs
 - 13 digital input/output pins
 - 5 analog input pins
 - Range (0-255 typically)
 - 6 analog output pins
 - 1 (HIGH) or 0 (LOW) value (i.e. on/off)
- Price: \$26 (UNO) - \$85 (Mega ADK)



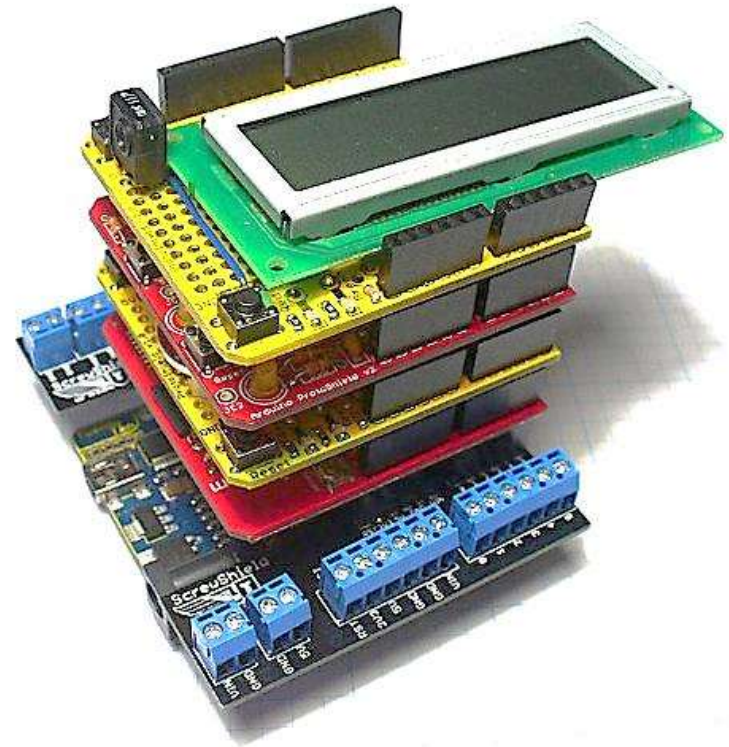
Arduino Capabilities



- Analog Reference pin
- Digital Pins 2-13, Digital Pins 0-1/Serial In/Out - TX/RX
 - These pins cannot be used for digital i/o (digitalRead and digitalWrite) if you are also using serial communication (e.g. Serial.begin).
- Reset Button - S1
- In-circuit Serial Programmer
- Analog In Pins 0-5
- Power and Ground Pins
- External Power Supply In (9-12VDC) - X1
- Toggles External Power and USB Power (place jumper on two pins closest to desired supply) - SV1
 - USB (used for uploading sketches to the board and for serial communication between the board and the computer; can be used to power board)

Arduino Shield Boards

- Expansion boards that plug into the normally supplied Arduino pin-headers
- Shields can provide a variety extensions and additional functions:
 - Motor controls
 - GPS
 - Ethernet
 - Sensors
 - Solar panel
 - Wireless communications
 - LCD display



Programming Arduino

- Global Variables
 - Declare variables at top
 - **int ledPin = 13;** – LED connected to control pin 13
- **setup()**
 - Initialize - run once at beginning, set pins
 - **pinMode(ledPin, Output);** – set the pin 'ledPin' as an output
- **loop()**
 - Running - run repeatedly, after setup()
 - **digitalWrite()** – set a digital pin high/low
 - **digitalRead()** – read a digital pin's state
 - **analogRead()** – read an analog pin
 - **analogWrite()** – write an “analog” PWM value
 - **delay()** – wait an amount of time



```
Arduino - 0011 Alpha
File Edit Sketch Tools Help

Blink

/*
 * Blink
 *
 * The basic Arduino example. Turns on an LED on for one second,
 * then off for one second, and so on... We use pin 13 because,
 * depending on your Arduino board, it has either a built-in LED
 * or a built-in resistor so that you need only an LED.
 *
 * http://www.arduino.cc/en/Tutorial/Blink
 */

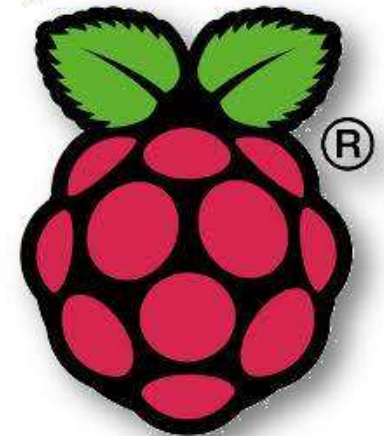
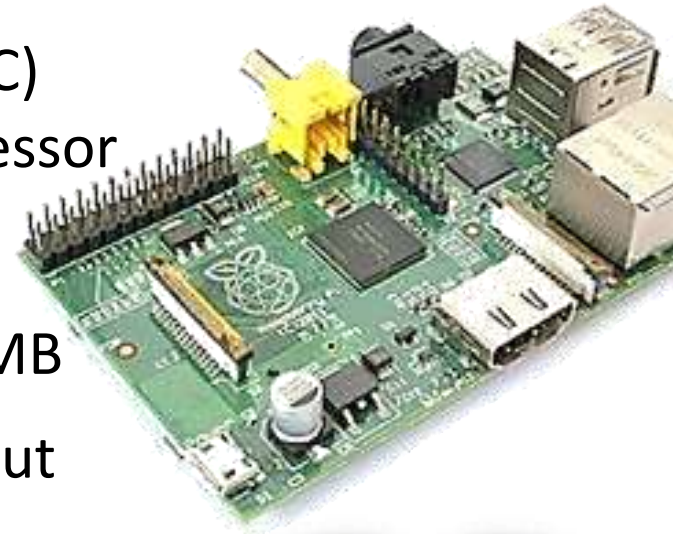
int ledPin = 13;           // LED connected to digital pin 13

void setup()               // run once, when the sketch starts
{
  pinMode(ledPin, OUTPUT); // sets the digital pin as output
}

void loop()                // run over and over again
{
  digitalWrite(ledPin, HIGH); // sets the LED on
  delay(1000);                // waits for a second
  digitalWrite(ledPin, LOW);  // sets the LED off
  delay(1000);                // waits for a second
}
```

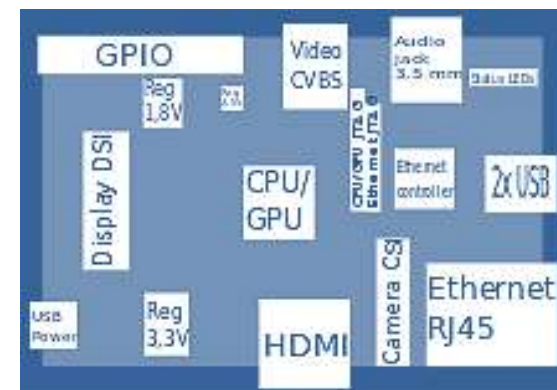
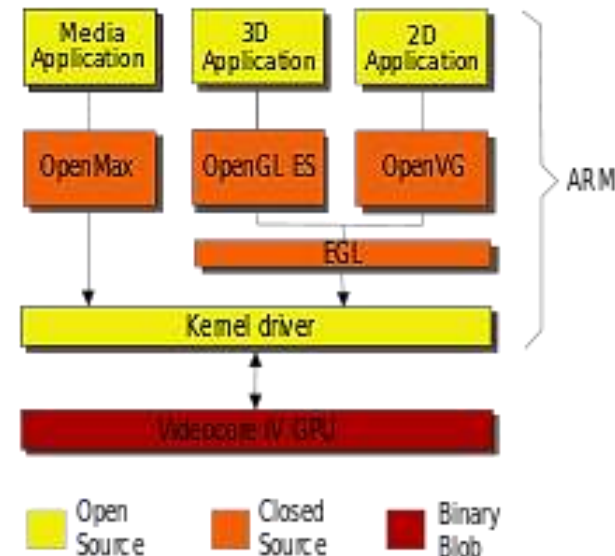

Raspberry Pi

- Full-blown single-board computer
- Broadcom BCM2835 system on a chip (SoC) includes an ARM1176JZF-S 700 MHz processor
- VideoCore IV GPU
- 256 megabytes of RAM, upgraded to 512MB
- No built-in hard disk or solid-state drive, but uses an SD card for booting/storage
- Linux/Android/Chrome OS/Firefox OS (being ported)
- Programming language: Python, C , Perl
- Price: \$25 - \$35



Raspberry Pi

- Model A has 1 USB port and no Ethernet controller, and cost less
- Model B has 2 USB ports and a 10/100 Ethernet controller
- Though the Model A doesn't have an Ethernet port, it can connect to a network by using a user-supplied USB Ethernet or Wi-Fi adapter
- Raspberry Pi uses Linux kernel-based operating systems. Raspbian, optimized for the Raspberry Pi hardware



Embedded Systems in Cars

- 1970s, emerging environmental regulations for clean emission
- Car manufactures put under pressure to improve fuel mileage and reduce pollution in the US
- Solution:
 - Fuel injection based systems requiring computers to automatically control fuel system
- Electronic controller unit (**ECU**) is an embedded system controls/monitors systems in car
- A collection of in-vehicle ECUs is referred as the cars computer
- The cars “computer” is a collection of embedded systems connected by an in-vehicle network

Electronic Controller Units

Electronic controller unit (ECU)

- Embedded system controlling the electrical systems in vehicle

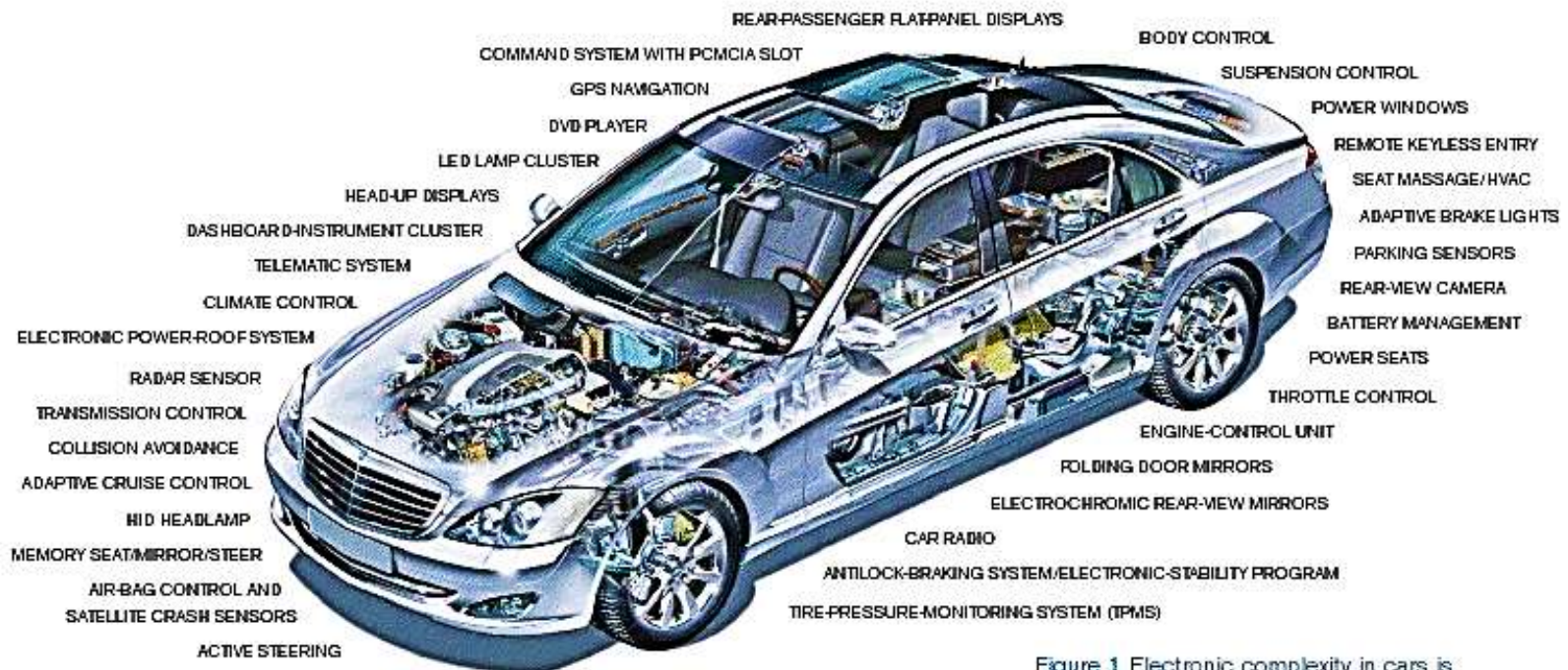


Figure 1 Electronic complexity in cars is increasing. New Mercedes S-Class cars employ at least 70 networked ECUs (electronic control units); 10 years ago, most cars had three ECUs (photo courtesy of DaimlerChrysler; source: Gartner Research, November 2005).

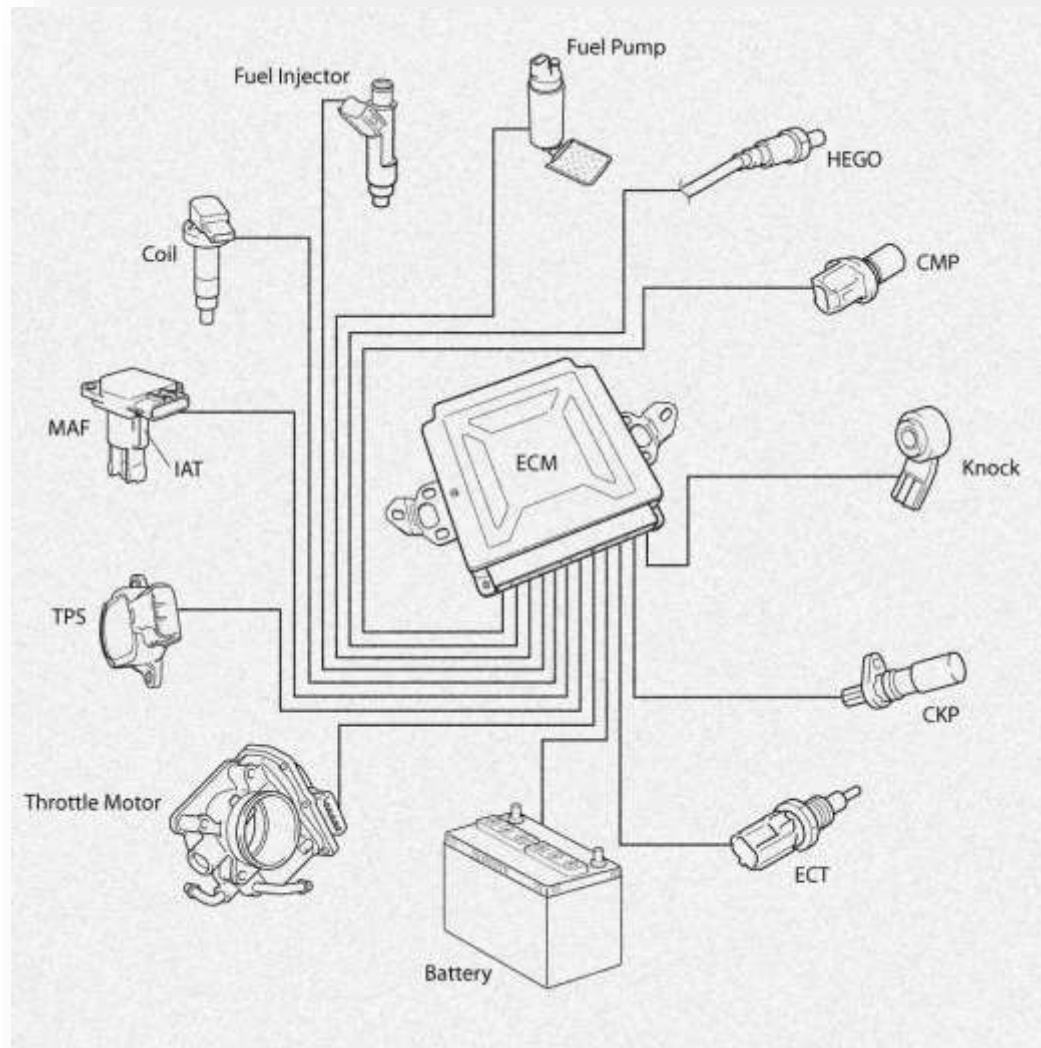
Embedded Systems in Cars

- Up to 1 million lines of code in some cars (80+ CPUs in a luxury car)
- Engine controller
 - Hard real time (ignition cycle), Fail safe,
 - 32-bit CPU (resource adequate)
- Transmission controller
 - Soft real time (shift points), Fail safe,
 - 8-bit to 32-bit CPU (resource marginal)
- Anti-lock Braking System (ABS)
 - Firm real time (pulses brake pedal)
 - Fail operational (for brakes), Fail safe (for electronics)
 - 8-bit CPU (resource constrained)

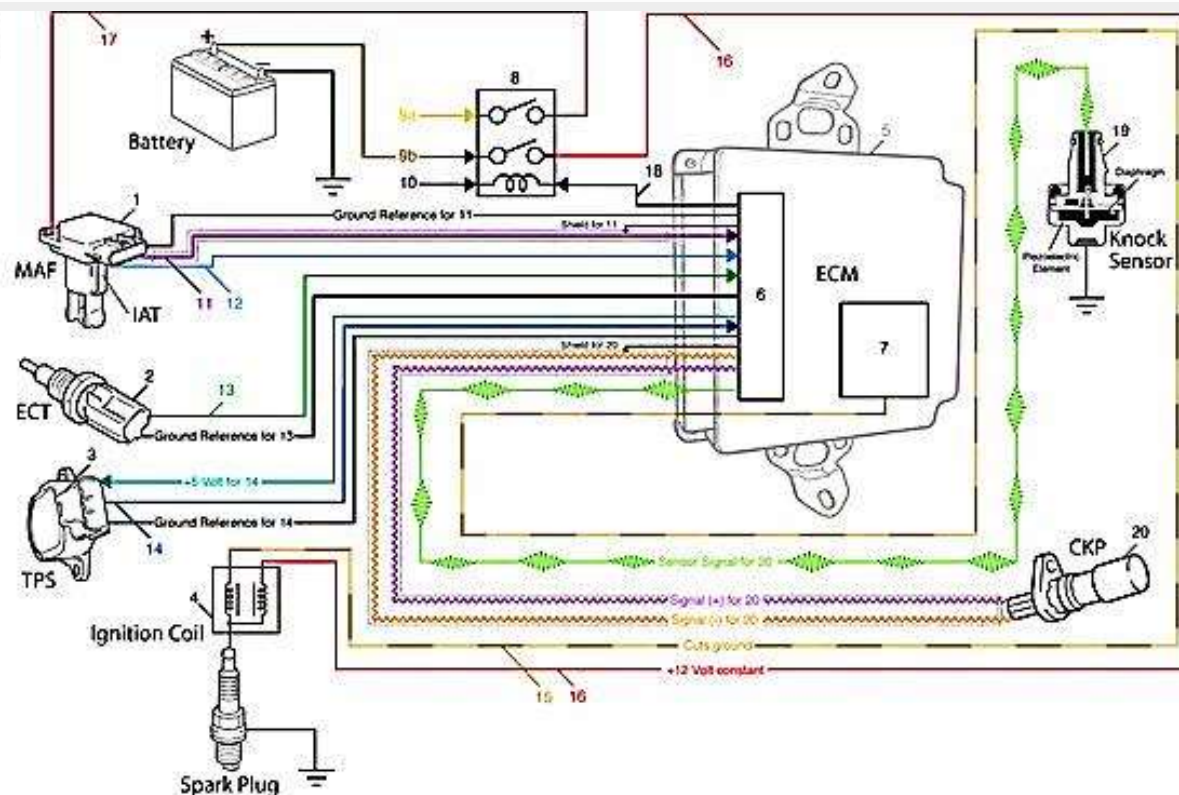
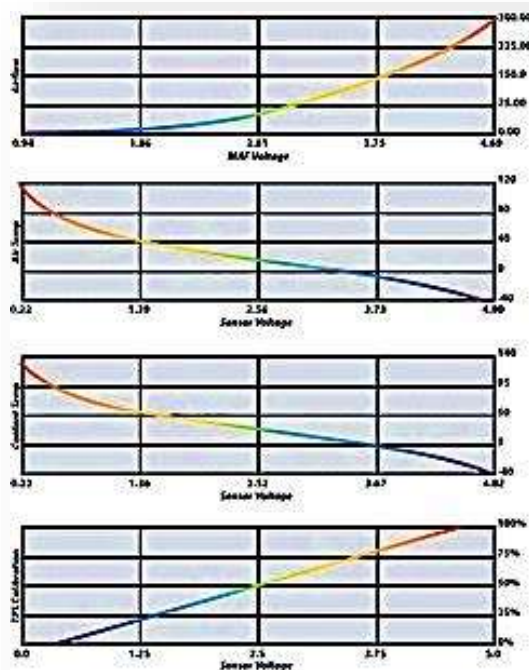
Developments of ECU

- First ECU (early 1980)
 - Manufactures moved towards Fuel injection based systems which require computers to control the system
- Engine Control Module (ECM)
 - Determine parameters for an Internal combustion engine
- Roll Stability Control (ESC)
 - Preventing loss of control (change steering angle)
- Anti-Lock Braking System (ABS)
 - Preventing the brakes from locking (change braking)
- Lane Assist System, Active Cruise Control, Automated Parking
 - Applying brake without users' intention

Engine Control Module (ECM)



Engine Control Module (ECM)

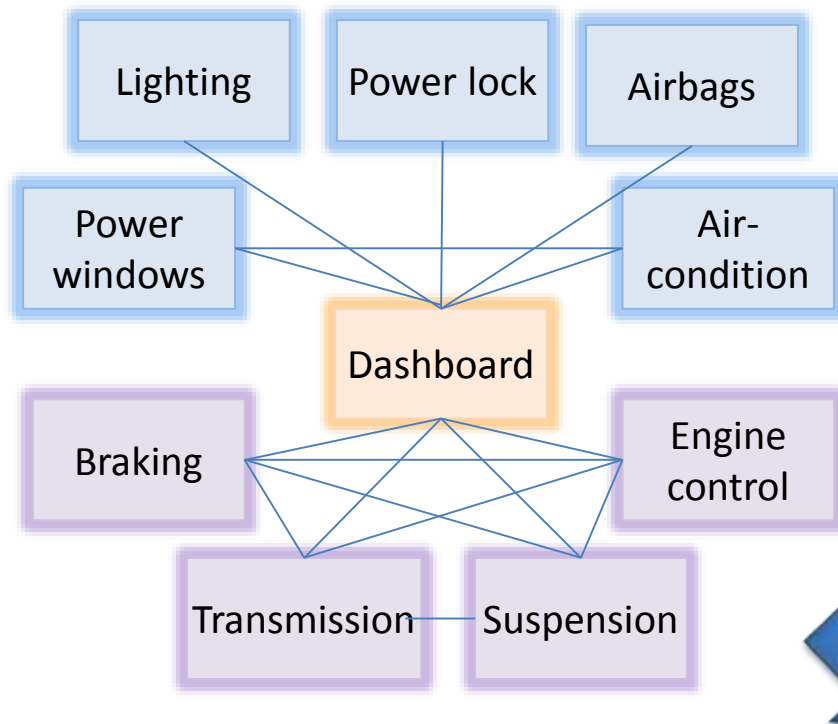


| | | |
|--------------------------------------|---|---|
| 1. Mass Air Flow w/ IAT Sensor (MAF) | 8. Main or EFI relay | 14. Throttle Position Sensor (TPS) sensor signal |
| 2. Engine Coolant Temp (ECT) Sensor | 9a. From 'EFI or Main' fuse +5v | 15. Drive Signal for Ignition Coils. ECU cuts ground to coil grounds through spark plug |
| 3. Throttle Position Sensor(s) (TPS) | 9b. From 'EFI or Main' SBFI fuse +12v | 18. Power Supply for fuel injector(s) and CMP sensor, +12v switched |
| 4. Ignition Coil(s) | 10. From Main relay | 17. Power Supply MAF and IAT, +5v switched |
| 5. Engine Control Module (ECM) | 11. Mass Air Flow (MAF) sensor signal | 18. Control signal for Main or EFI relay |
| 6. Central Processing Unit (CPU) | 12. Intake Air Temp (IAT) sensor signal | 19. Camshaft Position (CMP) sensor |
| 7. Drive circuit for Fuel Injectors | 13. Engine Coolant Temp (ECT) sensor signal | 20. Crankshaft Position (CKP) sensor |

How ECUs Communicate? CAN Bus

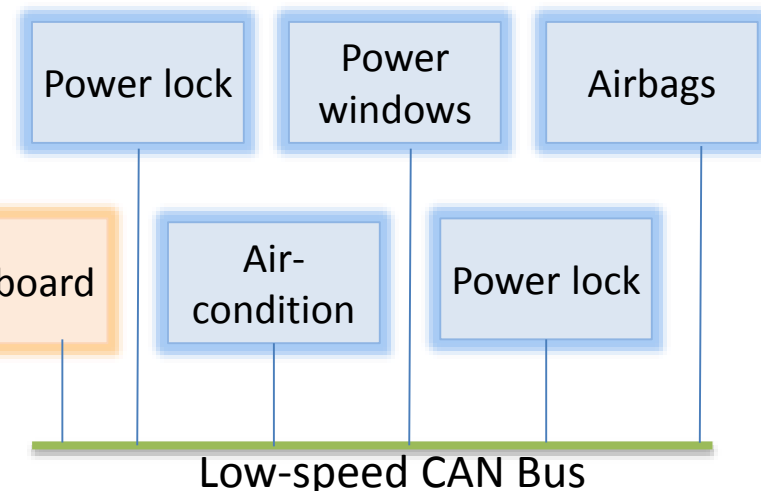
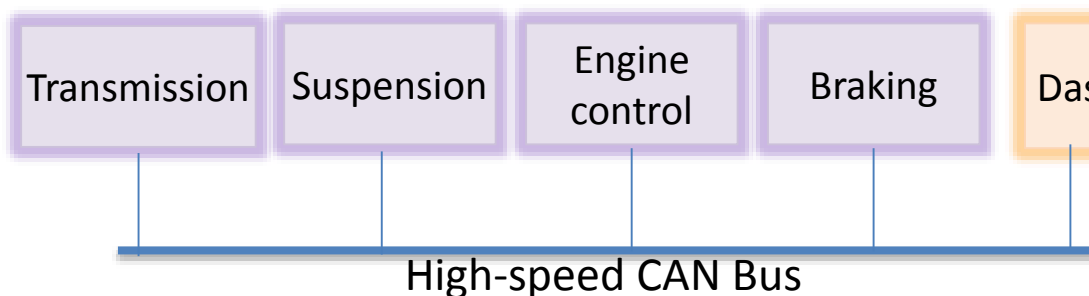
- The CAN bus (Controller Area Network)
 - Defined in the late 1980 by Bosch
 - Initially designed for in-vehicle networks
 - From 2008 , almost all passenger cars and light trucks in the US were CAN-equipped
- Other applications:
 - Medical equipment, UPS (uninterruptible power supply), Industrial systems, Appliances

Why CAN Bus

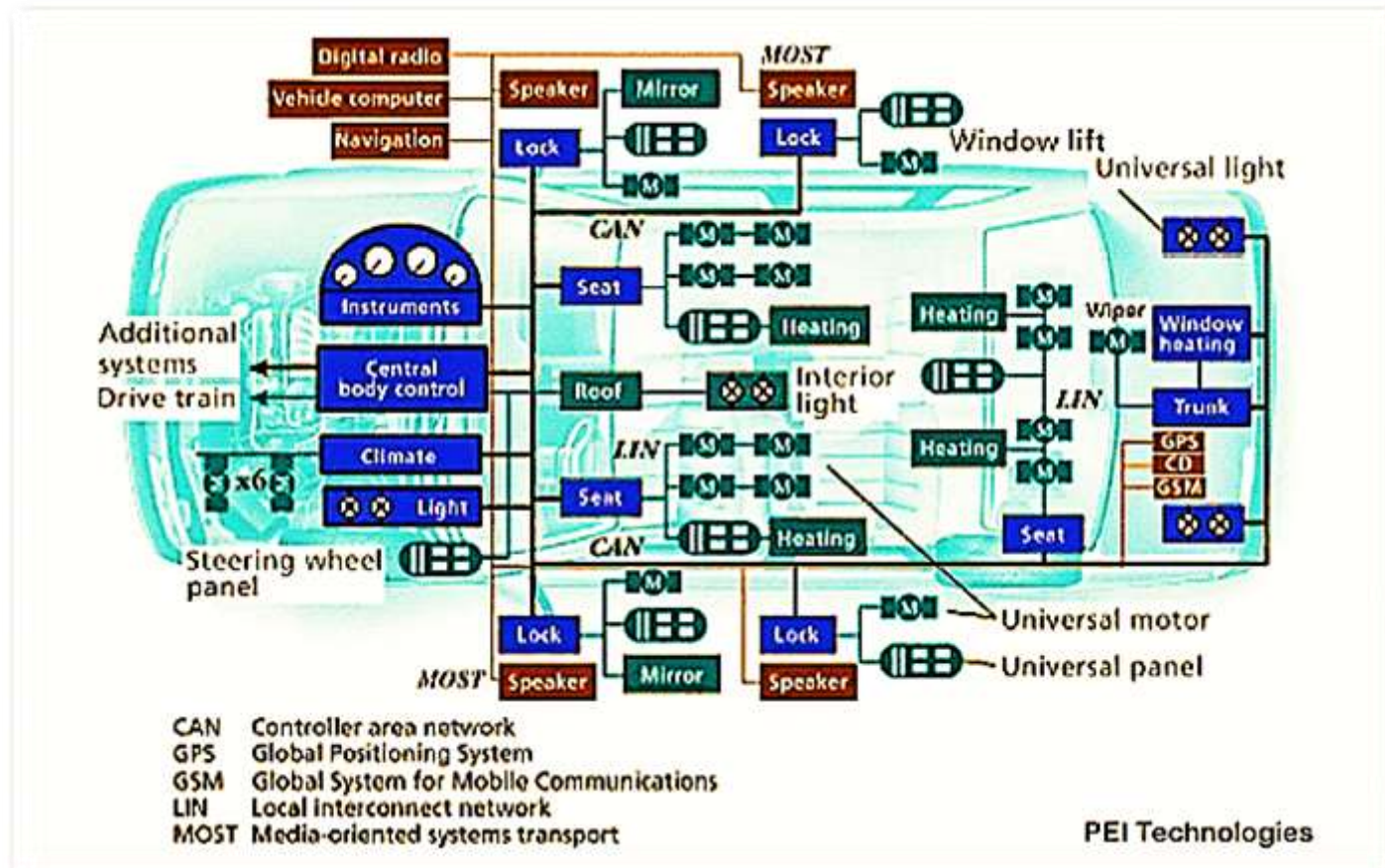


- Benefits

- Reduce one-to-one cabling
- The complexity of the wiring makes fault diagnosis difficult
- Wiring is not scalable, making minor modifications. hard



In-Vehicle Data Networks

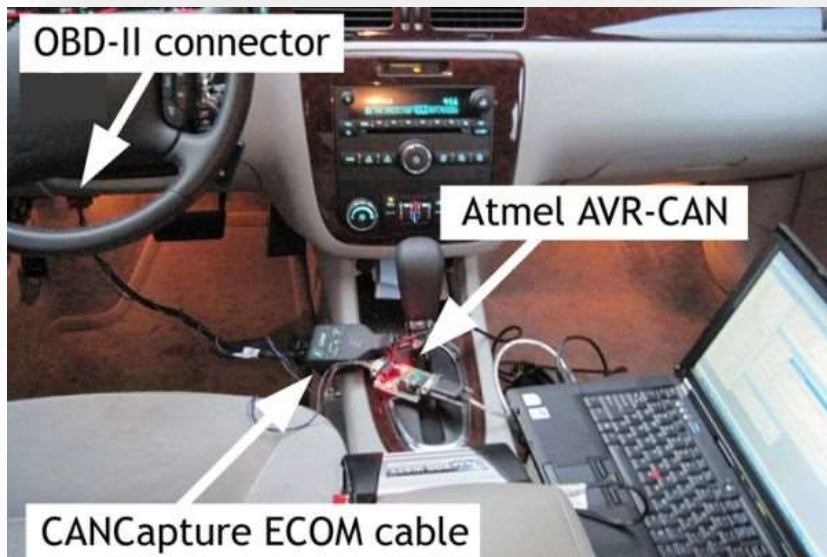


Hacking into In-Vehicle Networks

- CAN bus is exposed to external connection by OBDII
- OBD-II port is intended for read-only operations
- Present for enabling car-testing and repairing
- Special control allows overwriting operations
- But often disallowed, unless can be enabled



Arduino CAN Shield



In-Vehicle Security Vulnerability

- Broadcast – All packets are physically and logically sent to all ECU's
- Denial of Service – Priority based protocol allows malicious packets to dominate the network
- No Authentication – Some control packets are not authenticated, no source information is stored in CAN packets
- Ease of Access – Variety of tools available to access components and change settings or even re-flash the component
- Poor Network Segregation – Car critical components need to be isolated but bridging the networks is easily accomplished

Hacking Your Car

PCWorld How Hackers Attack Cars | 1

www.pcworld.com/article/196320/how_hackers_attack_cars.html

PCWorld News Reviews How-To's Downloads Shop & Compare Apps Business Center

TRENDING: Phones Tablets Security Gadgets Laptops Games Web Desktops Search Engines Windows MORE

Sign in with [f](#) [t](#) [v](#) [g](#) [in](#) [PCW](#) or Create a New Account.

PCWorld » Security


Recommend: [Like](#) [3](#) [+1](#) [0](#) [35](#) [Email](#) 7 Comments

How Hackers Attack Cars

Researchers demonstrate how car hackers can kill brakes, engine, and control vehicles on the road.

By Robert McMillan, IDG News May 14, 2010 9:02 pm

[ALL IMAGES](#) [RELATED ARTICLES](#) 1 of 8



Car Hacking

Researchers at the University of Washington and the University of California, San Diego, have taken the computer systems used to run today's cars and discovered new ways to hack into them: sometimes with frightening results. In a paper set to be presented at a security conference in Oakland, California, next week, the researchers say that by connecting to a standard diagnostic computer port included in late-model cars, they were able to do some nasty things, such as turning off the brakes, changing the speedometer reading, blasting hot air or music on the radio, and locking passengers in the car. For much of their testing, they simply put the test-car on blocks, pictured here.

Related Stories

- [Car Whisperer Puts Hackers in the Driver's Seat](#)
- [Security Experts Warn of Satellite Navigation Hacks](#)



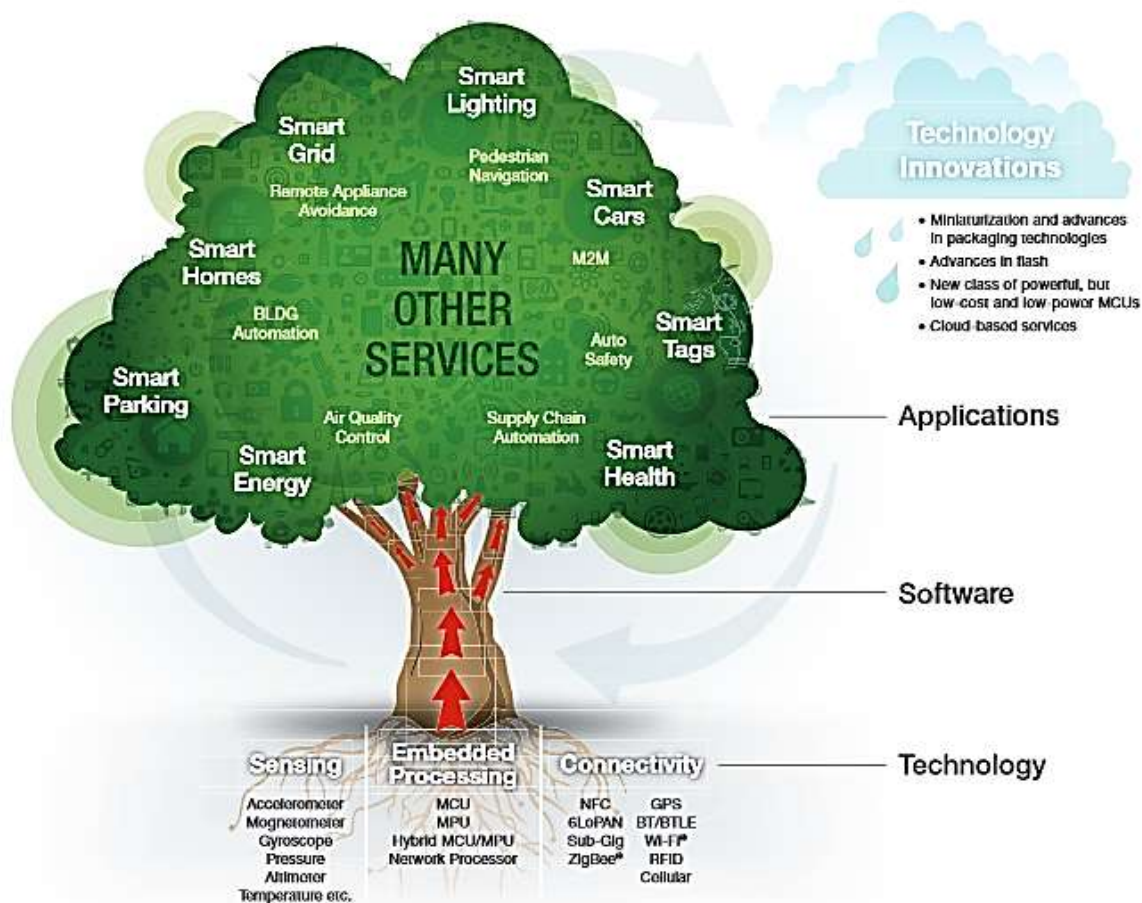
Autonomous Driving Car

- Cars are becoming highly computerized and network-connected
- Semi-autonomous cars are in the market
 - Lane departure warning
 - Blind spot detection
 - Obstacle detection
 - Pedestrian and cyclist recognition
- Fully autonomous cars are well-developed
 - Legalized in certain regions
 - Becoming a reality soon
- More embedded systems are equipped in new vehicles



Internet-of-things

The IoT: Different Services, Technologies, Meanings for Everyone



Internet-of-things

- Extending the current Internet and providing connection, communication, and inter-networking between devices and physical objects, or "Things"
- "The technologies and solutions that enable integration of real world data and services into the current information networking technologies are often described under the umbrella term of the Internet of Things (IoT)"
- Vision: IoT enable "Smart X"
 - where X can be everything such as phone, watch, TV, fridge, car, home, city, power grid, etc

Internet-of-things

- Smart device is a “Thing”
 - with sensors/actuators/tag
 - with some processing power
 - with communication capability



Watch out: Intel Edison

- Tiny computer offered by Intel as a development system for wearable devices
- Debut in Jan 2014 by Intel
- Same size and shape as an SD card
- Contains a dual-core Intel Quark CPU 400Mhz
- Can communicate via Bluetooth and Wi-Fi
- Will be able to run Linux
- Applications: wearable technology





References

- Course: Distributed Embedded Systems (CMU)
 - <http://wwwece.cmu.edu/~ece649/>
- Course: Embedded Systems (ETH)
 - <http://www.tik.ee.ethz.ch/education/lectures/ES>
- <http://arduino.cc/>
- Arduino Cookbook (M. Margolis) O'Reilly

