

ECE 471 – Embedded Systems

Lecture 27

Vince Weaver

`http://web.eece.maine.edu/~vweaver`

`vincent.weaver@maine.edu`

9 November 2020

Announcements

- HW#9 was assigned, due a week from Friday
- No class on Wednesday
- Power was out
- Responded to all project topics I received, please send if you haven't



HW#9 Info

- Re-use i2c display code from earlier homework
- Re-use temp code (either TMP36 or the 1-wire)
- Display the temperature on display
- Code coverage, Writing good testable code
- Converting a double to characters to print
 - Use `sprintf()`

```
char string[128];  
double temperature;  
sprintf(string, "%.1lf", temperature);  
/* Now string[0] has first digit, string[1] second.
```



- Use division/modulus

```
double temperature=23.4;  
int hundreds ,tens , ones ,remainder;  
  
hundreds=temperature/100;  
remainder=temperature%100;  
tens=remainder/10;  
ones=remainder%10;
```



Examples – CANbus

- 2010 IEEE Symposium on Security and Privacy.
Experimental Security Analysis of a Modern Automobile
U of Washington and UCSD.
- Fuzzing/ARM/CANbus
- can control brakes (on / off suddenly)
- heating, cooling, lights, instrument panel
- windows/locks Why? fewer wires if on a bus then direct-wired
- electronic stability control, antilock, need info from each



wheel

- roll stability control (affect braking, turning to avoid rollover)
- cruise control
- pre-crash detection (tighten seatbelts, charge brakes)
- while it might be nice to have separate busses for important and unimportant, in practice they are bridged
- Locks– monitor buttons, also remote keyfob... but also disengage if airbag deploys
- OnStar – remotely monitor car, even remotely stop it (in case of theft) over wireless modem



- Access? OBD-II port, also wireless
- 2009 car
- cars after 2008 required to have canbus?
- Problems with CAN
 - Broadcast... any device can send packets to any other
 - Priority.. devices set own priority, can monopolize bus
 - No authentication... any device can control any other
 - Challenge-response. Cars are supposed to block attempts to re-flash or enter debug mode without auth. But, mostly 16-bits, and required to allow a try every 10s, so can brute force in a week.



- If you can re-flash firmware you can control even w/o ongoing access
- Not supposed to disable CAN or reflash firmware while car moving, but on the cars tested they could.
- Probing – packet sniffing, fuzzing (easier as packet sizes small)
- experiments – on jackstands or closed course
- controlled radio – display, sounds, chimes
- Instrument panel – set arbitrary speed, rpm, fuel, odometer, etc
- Body control – could lock/unlock (jam by holding down



- lock), pop trunk, blow horn, wipers on, lights off
- Engine... mess with timing. forge "airbag deployed" to stop engine
 - Brakes.. managed to lock brakes so bad even reboot and battery removal not fix, had to fuzz to find antidote
 - can over-ride started switch. wired-or
 - test on airport. cord to yank laptop out of ODB-II
 - fancy attacks. Have speedometer read too high. Disable lights. "self-destruct" w countdown on dash, horn beeping as got closer, then engine disable.



Stuxnet

- SCADA – supervisory control and data acquisition
- industrial control system
- STUXNET.. targets windows machines, but only activates if Siemens SCADA software installed. four zero-day vulnerabilities
USB flash drives
signed with stolen certificates



- Interesting as this was a professional job. Possibly by US/Israel targeting very specific range of centrifuges reportedly used by Iran nuclear program. While reporting "everything OK" the software then spun fast then slow enough to ruin equipment.



Examples – JTag/hard-disk

- JTAG/Hard-disk takeover
- <http://spritesmods.com/?art=hddhack&page=8>
- Find JTAG
- 3 cores on hard-disk board, all ARM. One unused.
- Install custom Linux on third core. Then have it do things like intercept reads and change data that is read.



Places for More Info

- Embedded projects: <http://hackaday.com>
They had a recent series on CAN-bus
- Computer Risks and Security Issues: The RISKS digest
from comp.risks
<http://www.risks.org>



Software Bugs

- Not all bugs are security issues
- Coding bugs can have disastrous effects



Automotive

- Until recently no standard
- Bugs, Toyota firmware
- <http://www.edn.com/design/automotive/4423428/2/Toyota-s-killer-firmware--Bad-design-and-its-conse>



Airplanes

- DO-178B / DO-178C
- Software Considerations in Airborne Systems and Equipment Certification
 - Catastrophic: fatalities, loss of plane
 - Hazardous: negative safety, serious/fatal injuries
 - Major: reduce safety, inconvenience or minor injuries
 - Minor: slightly reduce safety, mild inconvenience
 - No Effect: no safety or workload impact



- AA Flight 965. Autopilot to waypoint R. Re-entered it, two starting with R, so it helpfully picked one with highest frequency, did a semi-circle turn to east right into a mountain.
- Air France Flight 447, reliance on autopilot
- Boeing 737MAX issues with MCAS system



Military

- Patriot missile – clock drift slightly, but when on for hundreds of hours enough to affect missile tracking
- Yorktown smart ship – 1997 – Running Windows NT. Someone entered 0 in a field, divide by 0 error, crashed the ship. Database crash, crashed propulsion system. Rumors that it needed to be towed in, but no, only down for 2.75 hours.
- F-22s computers crashed when crossing 180 degrees longitude? Lost navigation and communication, had to



follow tankers back to Hawaii.



Software Bugs

- Not all bugs are security issues
- Coding bugs can have disastrous effects



Spacecraft

- Mariner 1 (1962) – rocket off course due to mis-transcribed specification into FORTRAN, missing overbar
- Apollo 11 (1969) – landing on moon.
 - 36k ROM (rope), 2k RAM, 70lbs, 55W, 5600 3-input NOR
 - Processor normally loaded with 85% load. DELTAH program run which take 10%. But buggy radar device was stealing 13% even though in standby mode.



- Multiple 1202 overload alarms
- Mini real-time OS with priority killed low-priority tasks so things still worked.
- Ariane 5 Flight 501 (1996) – famous. \$370 million.
 - Old code copied from Ariane 4. Horizontal acceleration
 - Could not trigger on Ariane 4 (accel never that large)
 - Could trigger on more powerful Ariane 5
 - Conversion from 64-bit float to 16-bit signed int overflowed. Trap
 - Primary guidance computer crashed
 - Secondary computer, but ran same code, crashed



- Sent debug messages after crash, autopilot read those as velocity data
- Destructed 37s after launch
- Written in ADA
- NASA Mars Polar Lander (1999)
 - likely mistook turbulence vibrations for landing and shut off engine 40m above surface
- NASA Mars Climate Orbiter
 - ground software using lbf (pound/foot) units, craft expecting Newtons
- NASA Mars Spirit rover (2004)



- temporarily disabled due to too many files on flash drive
- Constantly rebooting
- Radio could understand some commands directly, could reboot with flash disabled.
- Fixed when deleted some unneeded files.
- Eventually reformat.
- Issue is 90 day design period, lasted years (until 2010)
- Phobos-Grunt (2012)
 - Bit flip in memory caused it to crash before firing rockets to Mars



- Entered safe mode waiting for command
- Antennas not deployed until after rocket firing
- Could not receive command to leave safe mode.
- ExoMars Schiaparelli Lander (2016)
 - Bad data to inertial measurement unit for 1 second
 - thought this meant it was below ground level, released parachute when still 3.7km up.
 - Had valid data from radar
- Boeing Starliner OTF-1 flight issues, lack of proper testing (2019)



Medical Example

- Therac-25 radiation treatment machine, 1985-1987
- 6 accidents, patients given 100x dose. Three died
High power beam activated w/o spreader too.
Older machines had hardware interlock, this one in software. Race condition. If 8-bit counter overflow just as entering manual over-ride, it would happen.
- Triggering the bug
 - To trigger, had to press X (mistake), up (to correct), E (to set proper) then "Enter" all within 8 seconds.



This was considered an improbable series of keypresses.

- This missed during testing as it took a while for operators to get used to using machines enough to type that fast.
- Used increment rather than move to set flag, this meant sometimes it wrapped from 255 to 0, disabling safety checks
- Written in Assembly Language

Things that went wrong with design

- Software not independently reviewed
- No reliability modeling or risk management



- Something wrong: Printed “MALFUNCTION” and error number 1 to 64 which was not documented in manual. Press P to clear.
- Operators not believe complaints from patients.
- The setup was not tested until after it was installed at hospital.
- cut-and-pasted software from earlier model that had hardware interlocks
- Concurrent (parallel) operation with race conditions



Financial

- Knight Capital. Upgrade 7 of 8 machines, missed last. Re-used a flag definition with new software. Caused massive selloff, \$440 million



Power

- 2003 Blackout
 - Power plant fail. Cause more current down transmission lines in Ohio. Heat, expand, touch tree, short out.
 - Race condition in Unix XA/21 management system, so alarms not go off
 - Eventually primary system fail as too many alarms queue up
 - Backup server also fail



- During failure, screens take 59s (instead of 1s) to update
- Blackout of most of NY and a lot of north east.



Space Shuttle Design

- https://www.nasa.gov/mission_pages/shuttle/flyo/flyfeature_shuttlecomputers.html
- Issues normal embedded systems don't have: Vibration at liftoff, Radiation in Space
- If computer stopped for more than 120ms, shuttle could crash
- “Modern” update in 1991: 1MB Ram, 1.4MIPS. Earlier was 416k and 1/3 as fast and twice as big
- Change to code, 9 months testing in simulator, 6 months



more extensive testing

- 24 years w/o in-orbit SW problem needing patches
- 12 year stretch only 3 SW bugs found
- 400k lines of code
- HAL/S high-order assembly language (high-level language similar to PL/I)
- PASS software – runs tasks. Too big to fit in memory at once
- BFS – backup flight software. Bare minimum to takeoff, stay in orbit, safely land, fits in memory, monitors pASS during takeoff/landing Written by completely different



team.

- 28 months to develop new version
- IBM
- Extensive verification. One internal pass, one external
- 4 computers running PASS, one running BFS
- Single failure mission can continue; still land with two failures
- 4 computers in lock-step, vote, defective one kicked out

