Real-Time Systems

Lecture Topic–Real time Service Implementation

Dr. Sam Siewert

Electrical, Computer and Energy Engineering

Embedded Systems Engineering Program

Quick Review

- Service Utility, Qualitative Utility over Time
- RM Policy, Feasibility, Safety
- Timing Diagrams over LCM (Examples Here)
- Use Cheddar to Check Work - Overview
- Assignment #1 - My Solution (not ideal, but basically works) - Lab 1 Sequencer
- Better solution for sequencing (one delay) - Generic Sequencer

Most Safe Policy for HRT Systems

- RMA with Necessary and Sufficient Feasibility Test
- Maintain Safety Margin (10%+) Given the 0% Margin Harmonic Services are Feasible
- RM LUB is High Margin - Safe, but Pessimistic

RM LUB - inexact, but safe with margin - estimate

$$U = \sum_{i=1}^{m} (Ci / Ti) \leq m(2^{\frac{1}{m}} - 1)$$

Lehoczky, Sha, Ding Theorem

Lehoczky, John, Lui Sha, and Yuqin Ding. "The rate monotonic scheduling algorithm: Exact characterization and average case behavior." *RTSS*. Vol. 89. 1989.

Exact characterization with margin

Test	Safety Margin?	Feasibility Test?
LUB	Yes, f(m)	Sufficient=some false failures
N&S (Completion Test or Scheduling Point)	Some cases are zero margin (harmonic) and therefore edgey	Exact
Safe Criteria (CT or SP feasible with 10%+ margin)	Yes, fixed	Fails only if margin is too high or analysis is incorrect

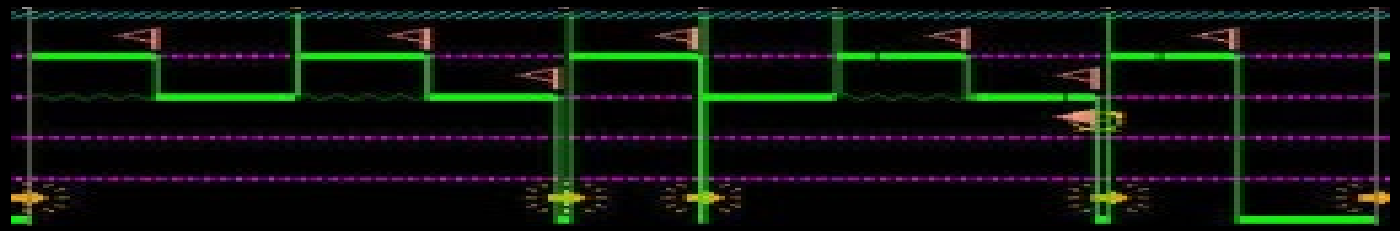
RTOS to Linux Comparison to Implement Theoretical Schedule

- RM Theory with 10 msec units (S_1=fib10, C_1=1, T_1=D_1=2; S_2=fib20, C_2=2, T_2=D_2=5)

Lab 1	T1	2	C1	1	U1	0.5	LCM =	10		
	T2	5	C2	2	U2	0.4	Utot =	0.9		
RM Schedule										
S1										
S2										FREE

- VxWorks RTOS equivalent with synthetic workload

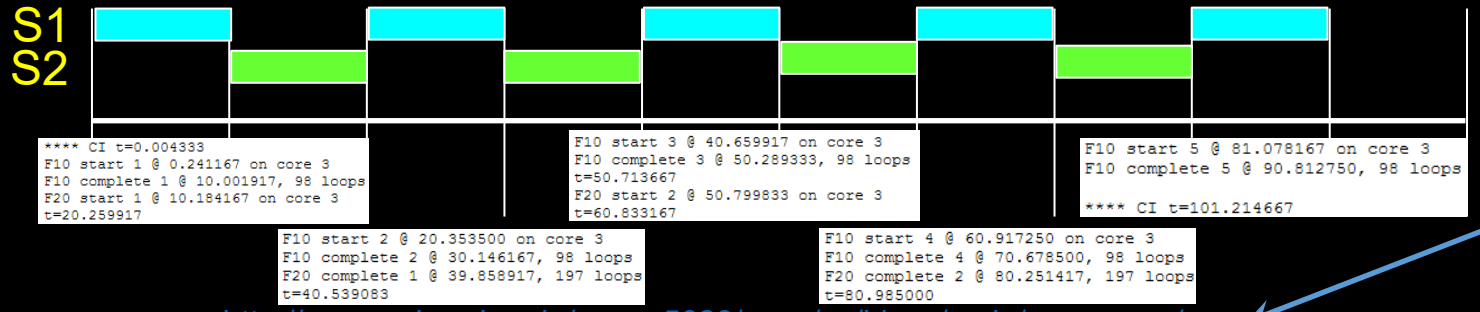
S1
S2



http://ecee.colorado.edu/~ecen5623/ecen/ex/Linux/code/VxWorks-sequencers/lab1.c

- Linux equivalent with synthetic workload (float msec)

S1
S2



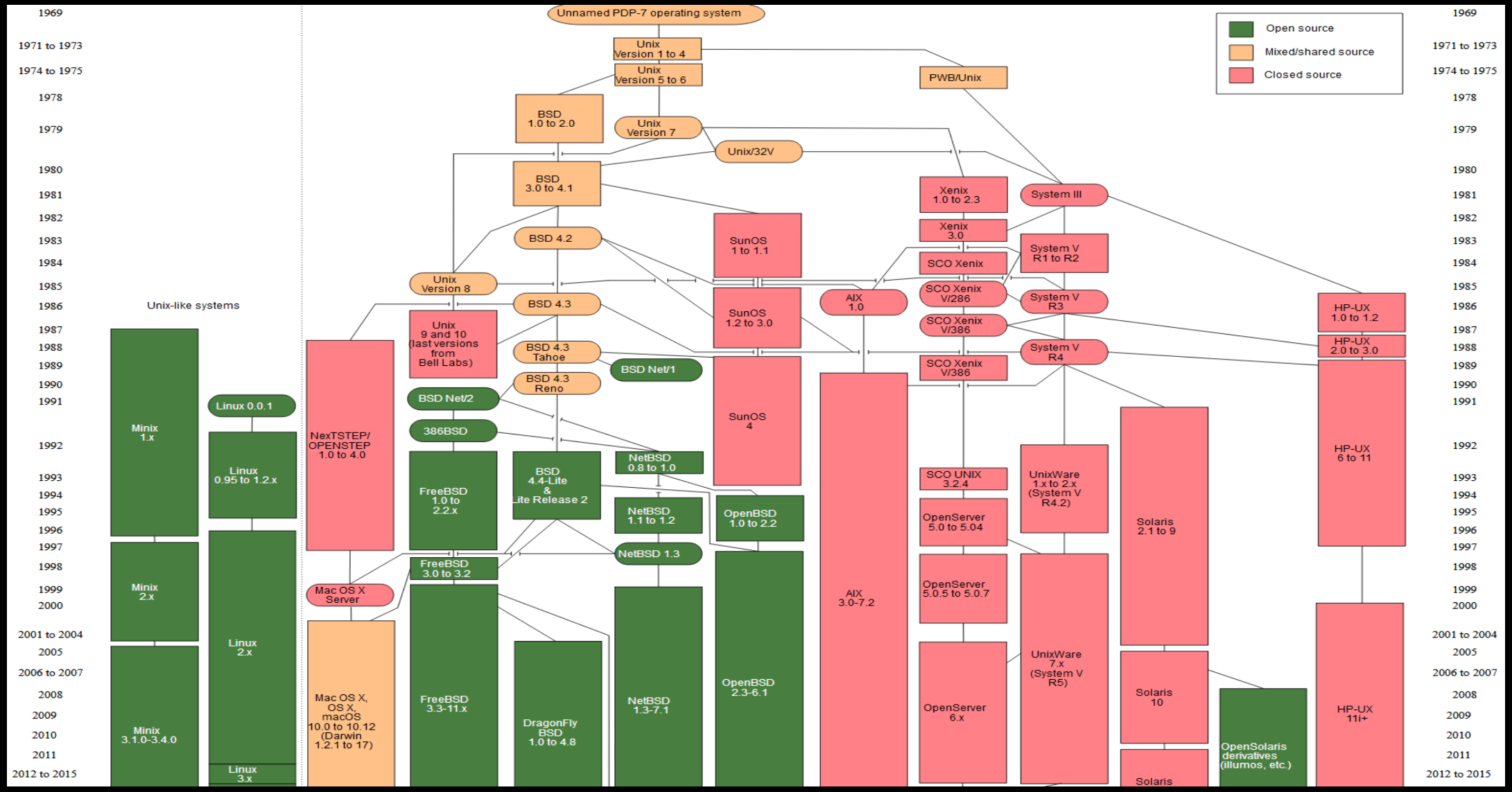
http://ecee.colorado.edu/~ecen5623/ecen/ex/Linux/code/sequencer/

Review code examples

3 Approaches to RT Systems

Cyclic Exec	RTOS	OS + RT POSIX
E.g. Shuttle Flight Software, Network elements, process control, Ada83/95 CE	VxWorks, QNX, ThreadX, TI RTOS, FreeRTOS, Zephyr, Nucleus, RTEMS, etc.	RT Linux, Solaris, LynxOS, FSM Labs, Concurrent, RTAI, Linux Foundation RT, etc.
Custom, Deeply Embedded Systems	Embedded Systems, Scalable and Portable	RT Services for Scalable Apps and Systems
Low over-head, purpose-built	Medium overhead, quick to market	Highest overhead, mixed RT + SRT + BE
Costly to develop	License costs	Maintenance costs

Brief History of Unix, Linux, OS-X vs RTOS



1951 - TI

Intel RAM

Zilog Z80

Intel 8051

Wind River

C/ASM & CE, Fortran, Ada

RTEMS

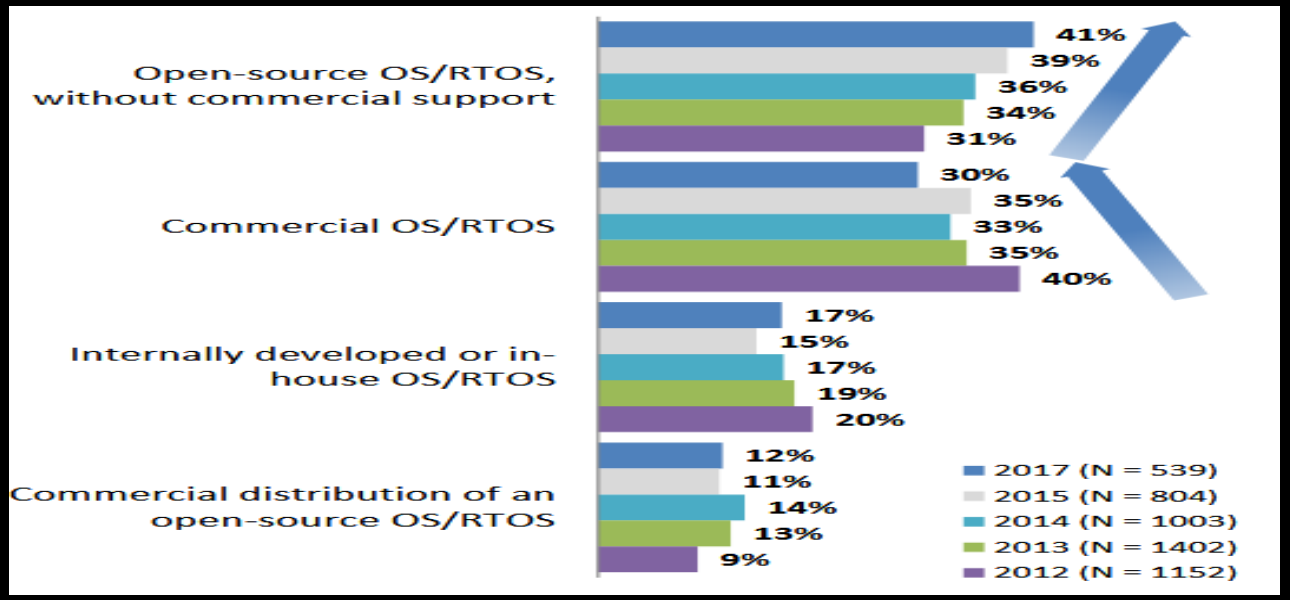
VxWorks

OS-less

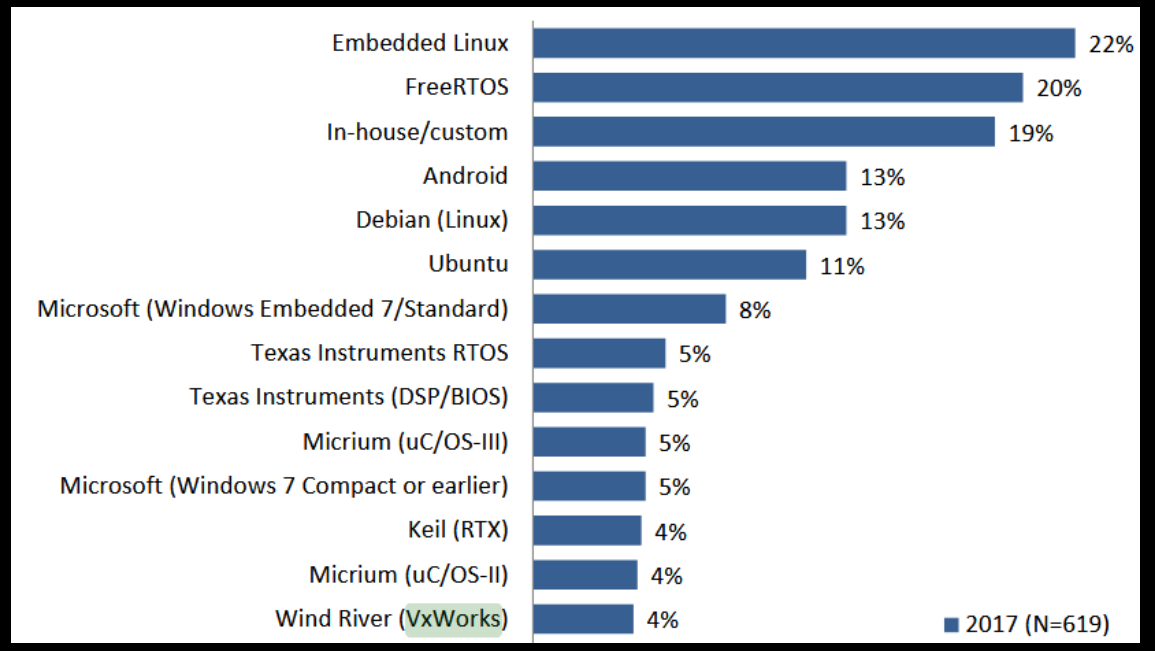
Linux, Zephyr, FreeRTOS

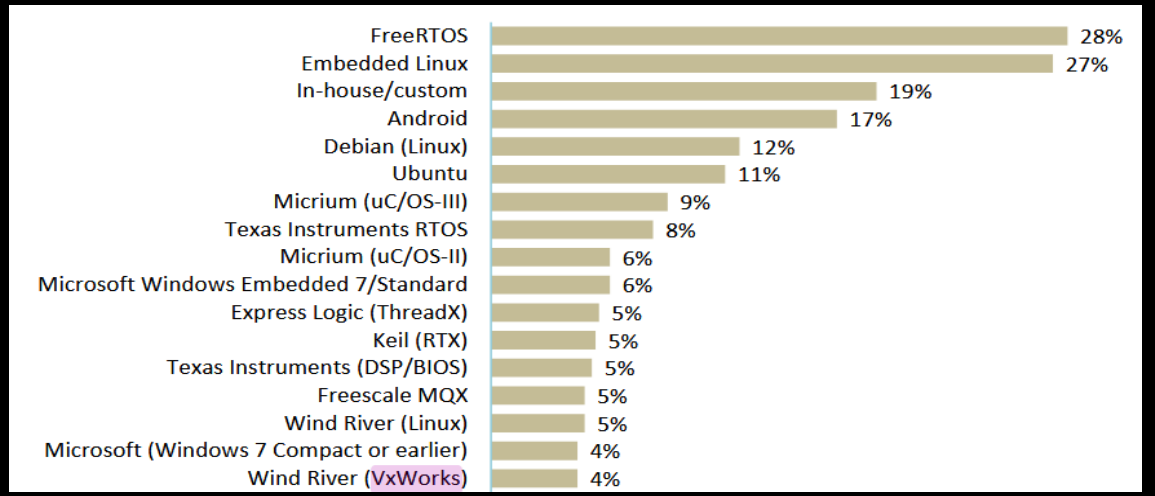
RTOS - Proprietary vs Open Source

- New Projects - Open Source
- Embedded Linux and FreeRTOS are future
- VxWorks, etc. are Legacy Embedded Systems



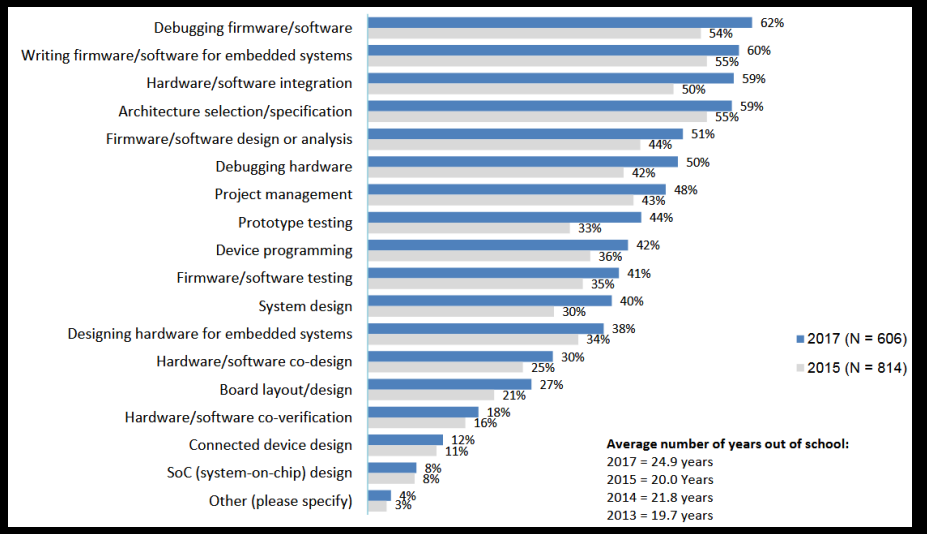
https://m.eet.com/media/1246048/2017-embedded-market-study.pdf





Embedded Systems

- Software is primary job function along with Firmware & System Integration
- HW System, PCB, and least-most Chip Design
- Industrial control and IoT (sensor networks)
- Consumer electronics and communications



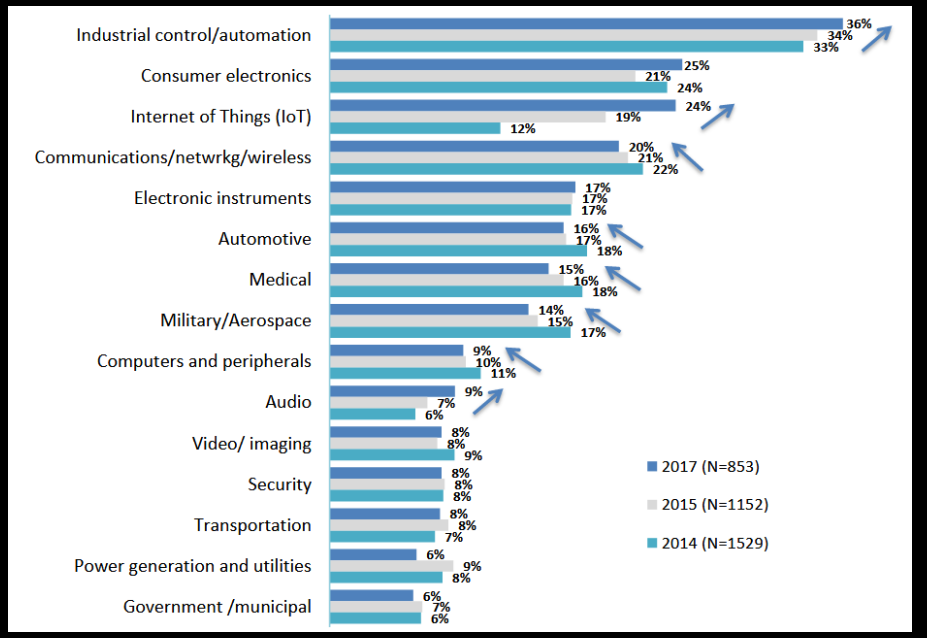


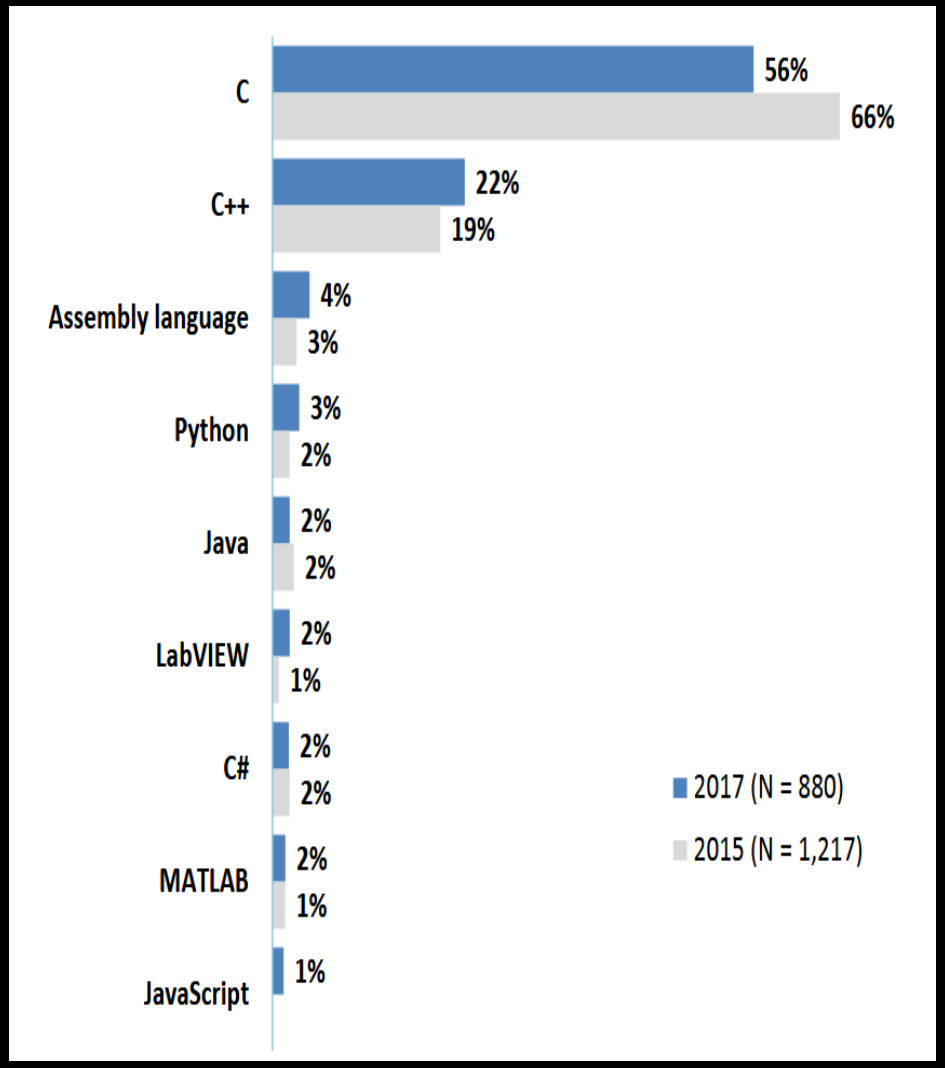
Image Source: https://m.eet.com/media/1246048/2017-embedded-market-study.pdf

Embedded Programming

- C/C++
- ASM
- Python - prototyping and verification

- Real-time is most common capability with DSP and Networking

- Analog signal processing for Sensor AFE (Analog Front End)



https://m.eet.com/media/1246048/2017-embedded-market-study.pdf