Debugging

MINOS 2010

The tenth annual MINOS conference on micromouse design, with wall-following and full maze-solving competitions

Rob Probin

10th April 2010

Overview

- Some ideas for debugging, measuring and analysing embedded systems and robots
- Some entertainment as well!

"Bloody instructions, which, being taught, return
To plague the inventor …"

— William Shakespeare
Macbeth, act 1, scene 7, Macbeth

Define: debugging

"is the process of locating and fixing or bypassing bugs (errors) in computer program code or the engineering of a hardware device."

www.sunnet.us/terms_definitions.html

Define: debugging

“Debugging is a methodical process of finding and reducing the number of bugs, or defects, in a computer program or a piece of electronic hardware, thus making it behave as expected."

http://en.wikipedia.org/wiki/Debugging

Define: debugging

Often: converge on problem

Why talk about debugging?

"As soon as we started programming, we found to our surprise that it wasn't as easy to get programs right as we had thought. Debugging had to be discovered. I can remember the exact instant when I realized that a large part of my life from then on was going to be spent in finding mistakes in my own programs. "

— Maurice Wilkes

Why talk about debugging?

- Too little discussion
- Mostly self taught

"Everyone knows that debugging is twice as hard as writing a program in the first place. So if you're as clever as you can be when you write it, how will you ever debug it?"

— Brian Kernighan, "The Elements of Programming Style", 2nd edition, chapter 2

Why talk about debugging?

- Difficulty underestimated
- Not done methodically
- Can be a creative activity

"With good program architecture debugging is a breeze, because bugs will be where they should be."
— David May

Why talk about debugging?

- Understanding problem and system
- Empirical evidence of software development

Traditional Debuggers

Debuggers are software tools which enable the programmer to:

- monitor the execution of a program,
- stop it,
- re-start it,
- set breakpoints,
- change values in memory and
- even, in some cases, go back in time.

http://en.wikipedia.org/wiki/Debugging

On embedded targets via ...

- Target debugging stub
- JTAG / OCD / BDM

But!

- Not always available
- Not good for dynamic work

Usual but not talked about

- Print statements
- Simple logging (simple trace)
- Debug Macros (e.g. step, monitor)
- Code reading
- Debug on PC
- Conditional actions

Create new techniques to find problem

Less ‘traditional’

- Interactive command lines:
  - Forth/Lisp/Python
  - C/C++ command Line (Demo!)
- Offer ‘incremental bottom-up development’

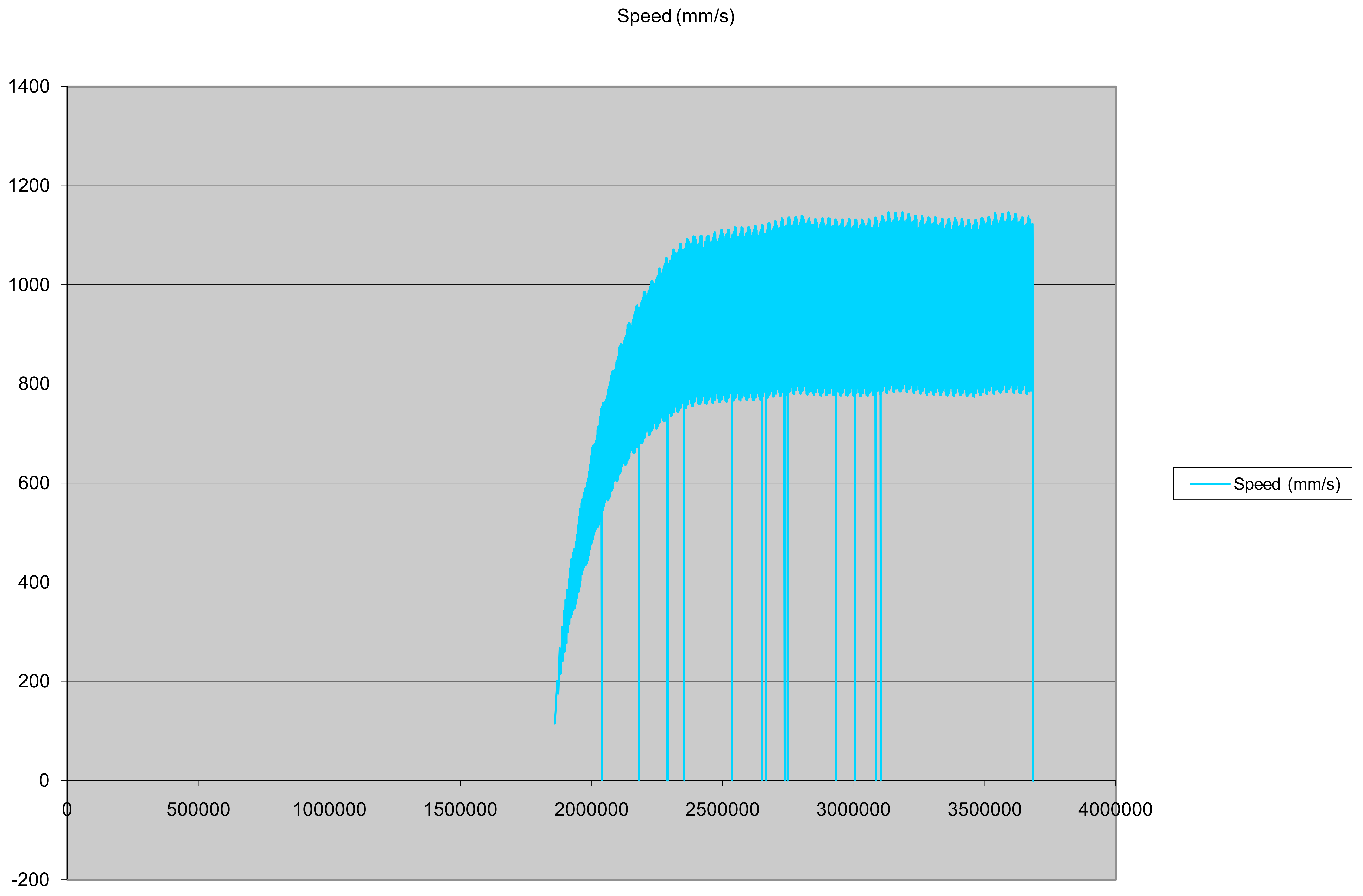
"Count what is countable, measure what is measurable, and what is not measurable, make measurable."

— Attributed to Galileo Galilei (1579)
(Quoted in Wilfred J. Kaydos, Operational Performance Measurement, 20)

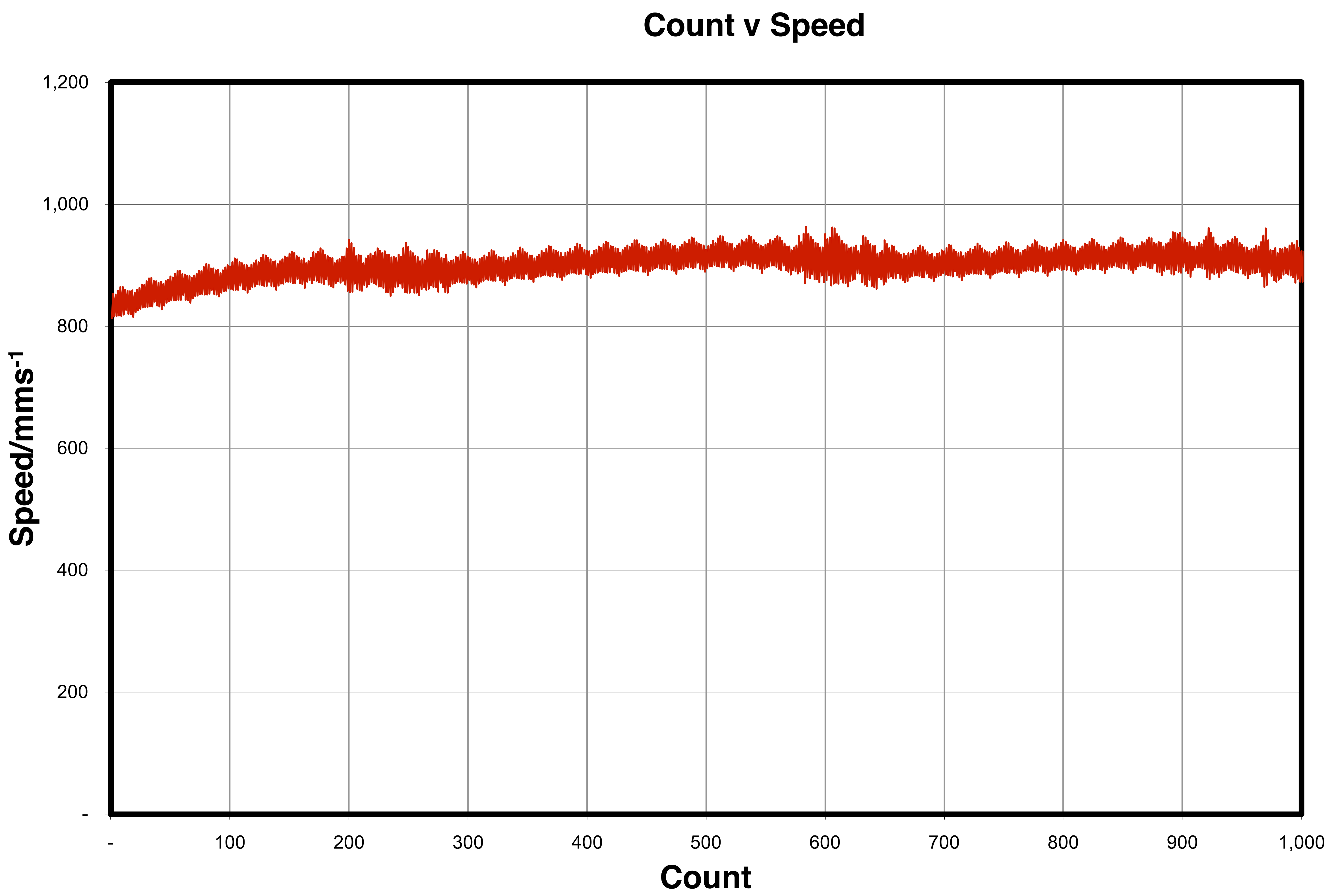
Data Analysis

- Log data - to serial or memory
- Analyse data off-line
- Spreadsheets, graphs, averages
- Monitoring applications

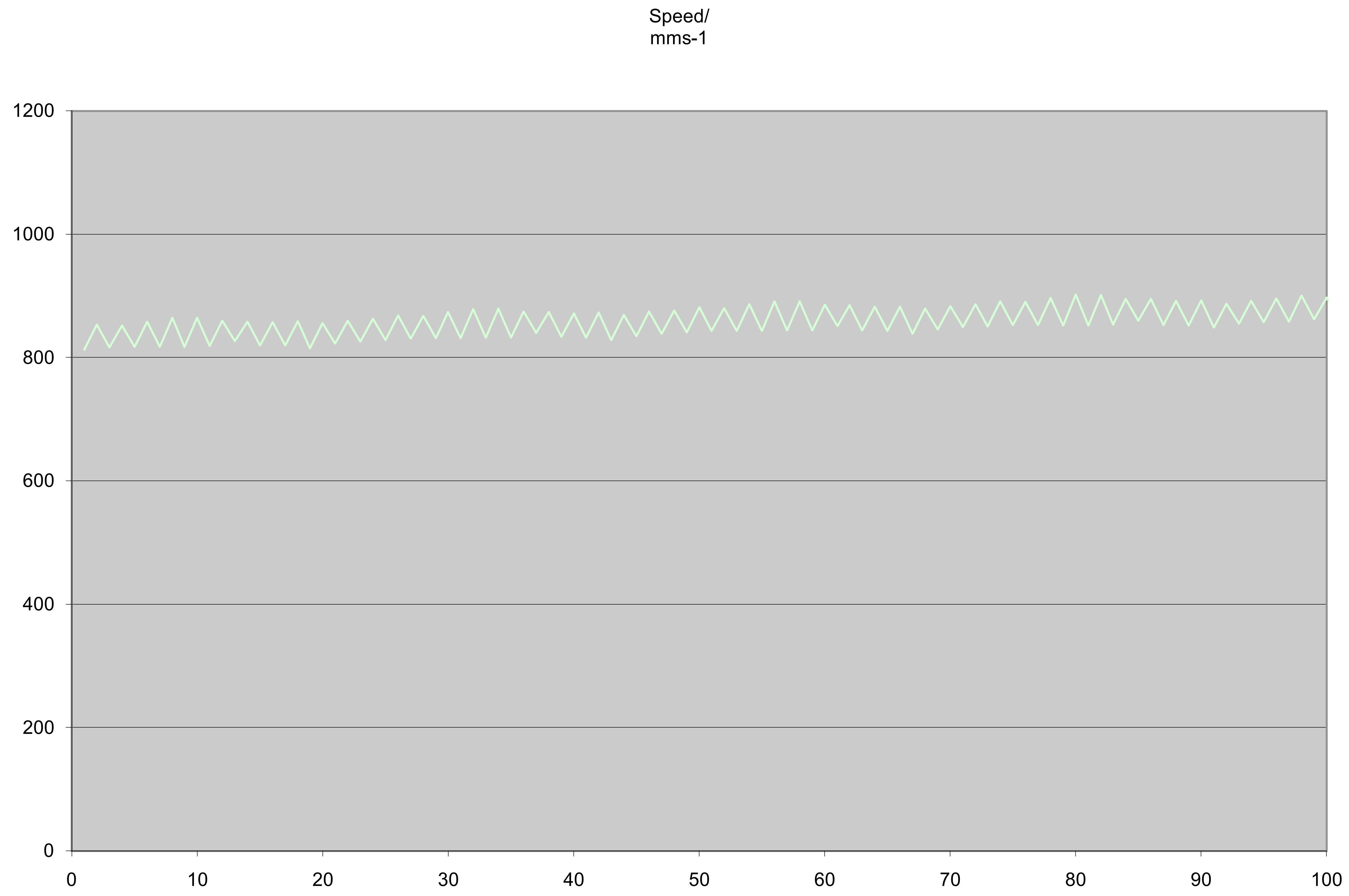
Example Graph



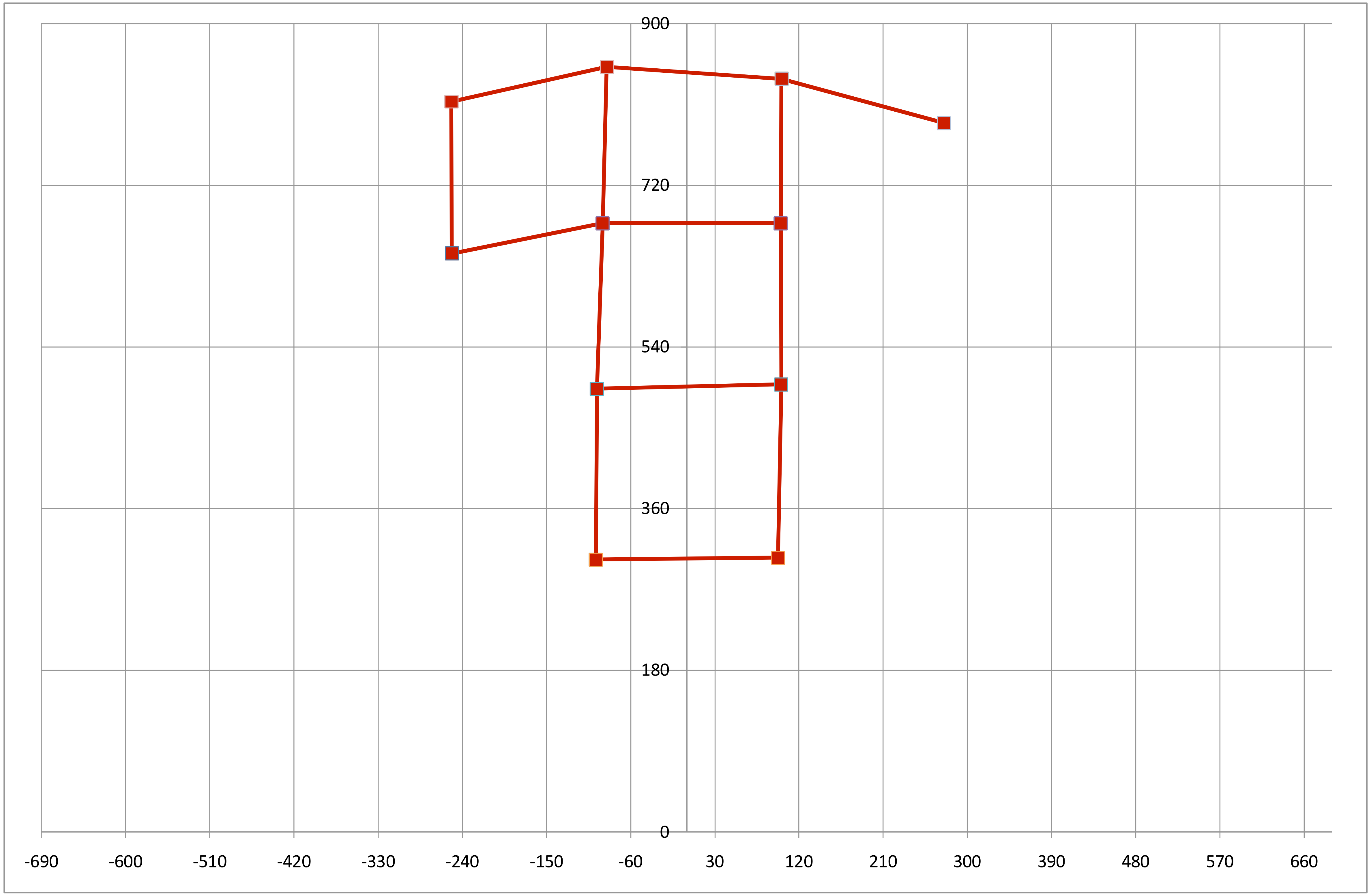
Example Graph



Example Graph



Example Graph



Other Thoughts

- Embedded File System, plenty of flash

  Or Forth-like blocks

Other Thoughts

- More...

interpreting stack frames, static & dynamic code analysis, design testing, modelling (continous/ discrete), profiling, advanced breakpoints, bounds checking, defensive programming, programming by contract, ...

Finally...

Beware of bugs in the above code; I have only proved it correct, not tried it.

— Donald Knuth, March 22, 1977 [5]

Testing can only prove the presence of bugs, not their absence.

— Edsger W. Dijkstra

bug, n: An elusive creature living in a program that makes it incorrect. The activity of "debugging", or removing bugs from a program, ends when people get tired of doing it, not when the bugs are removed.

— Datamation, January 15, 1984