Building Embedded Systems with Embedded DSLs

Pat Hickey
with
Lee Pike, Trevor Elliott, James Bielman, John Launchbury

| galois |

Thanks, Kathleen

- This work supported under DARPA's HACMS program
- SMACCM Partners: Rockwell Collins, University of Minnesota, NICTA, Boeing
- Open source: github.com/GaloisInc

Goal:

Build a High Assurance Helicopter Controller

Goal:

Build a High Assurance Quadcopter Controller

- Run on a small embedded system (microcontroller)
- Hard real time
- Safe
- Secure

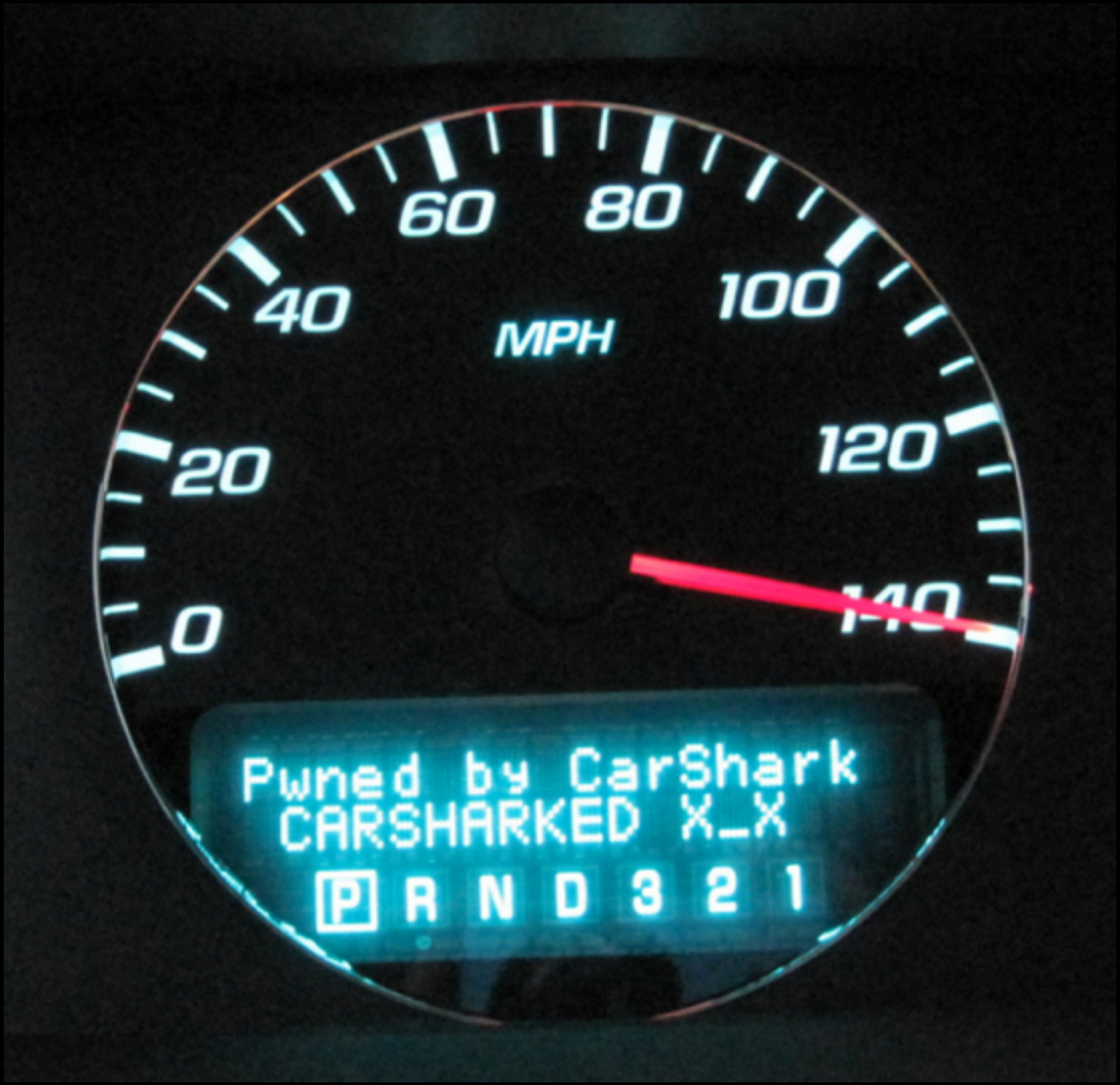
- 3 Engineers
- 18 Months
- ~50kloc C/C++

Embedded Systems

- They’re everywhere: hundreds in your home, your car. Billions sold per year.
- They’re basically just computers from the 80s.
    - “Now with 192k RAM!”
    - Shrunk down to be very small and very cheap.
- Development tools are right out of the 80s, too.

Embedded Systems

- All the security flaws you’d expect
- Can’t push a patch
- More attack surfaces than ever



Kohno, Savage et al,
USENIX Security 2011

Approach

Haskell, OCaml?

Very resource limited system
GC incompatible with hard real-time

C, C++?

Safety ⟷ Productivity

NASA Jet Propulsion Lab writes high assurance C.
Its very, very costly.

Build your own tools.

Starting with a clean slate
Language approach
Correct by construction

We built an embedded DSL
in just a few months.

Ivory

Embedded in Haskell
Compiles to C

Ivory DSL

- Safe subset of C
    - Memory Safety
    - No undefined or implementation defined behavior
- No heap, only global and stack allocation

Ivory DSL

- Embedded in Haskell
- Haskell type system guarantees properties of Ivory language programs
- Haskell is Ivory’s macro language

Ivory Syntax

- Expressions are pure Haskell values
- Statements are effectful, embedded in Monad
- Untyped AST is simple to pretty-print to C

Ivory Syntax

```
add_array :: Def (‘[ Ref Global (Array 10 (Stored Sint16))
                   , Sint16] :-> ())

add_array = proc “add_array" $
   \ref val -> body (prgm ref val)



prgm :: Ref s (Array 10 (Stored Sint16)) -> Sint16
     -> Ivory eff ()

prgm ref val = arrayMap $ \ix ->
    store (ref ! ix) val
```

Ivory Compositions

- Ivory programs (modules) are a collection of top level C procedures and variables
- So, just like in C, composing programs is not like composing functions, its based on local (argument) or global names.

Tower DSL

- Tower composes Ivory procedures into applications
- Initial problem: multithreading with message passing
    - Code generation of low level primitives
- Tower is a Haskell macro that generates Ivory code and system descriptions

Tower DSL

- Tower composes Ivory procedures into applications:
  - Code and state was easily collected into Tasks
  - Connections between task ports using Channels
  - User doesn’t have to maintain as many modules
- Tower became the DSL describing software components, which happened to be implemented in terms of event loops
- Currently working to decouple threading & scheduling from software modularity

SMACCMPilot

SMACCMPilot Application

Drivers	10 kloc
Application	3 kloc
Message marshaling	10 kloc

48 kloc C

SMACCMPilot Evaluation

- Red team results:
    - Delivery at 16 months of development
    - Found no buffer overflows, no undefined behaviors, no denial of service
    - There were one or two subtle architecture level bugs

Conclusion

- We were able to build a large, complex application, quickly and cheaply, with relatively few bugs
- We caught lots of errors early by Haskell type checking, were able to focus our efforts on application design rather than low level bugs
- Using Haskell as a macro system allowed us to build compositional programs

Thank You

SMACCM

Source code & more info:

http://ivorylang.org http://smaccmpilot.org

@path1ckey @galoisinc