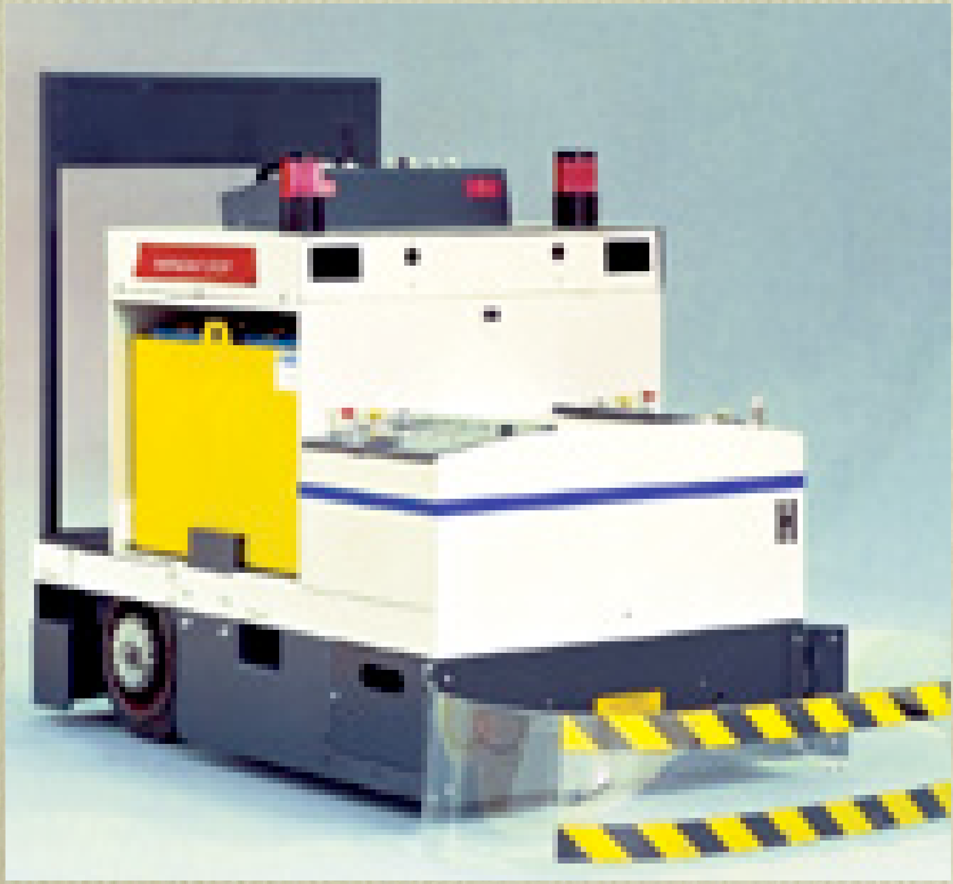
Evolving into Embedded Development

Matt Fletcher
Atomic Object

Agile 2007

About us

- Atomic Object develops custom software

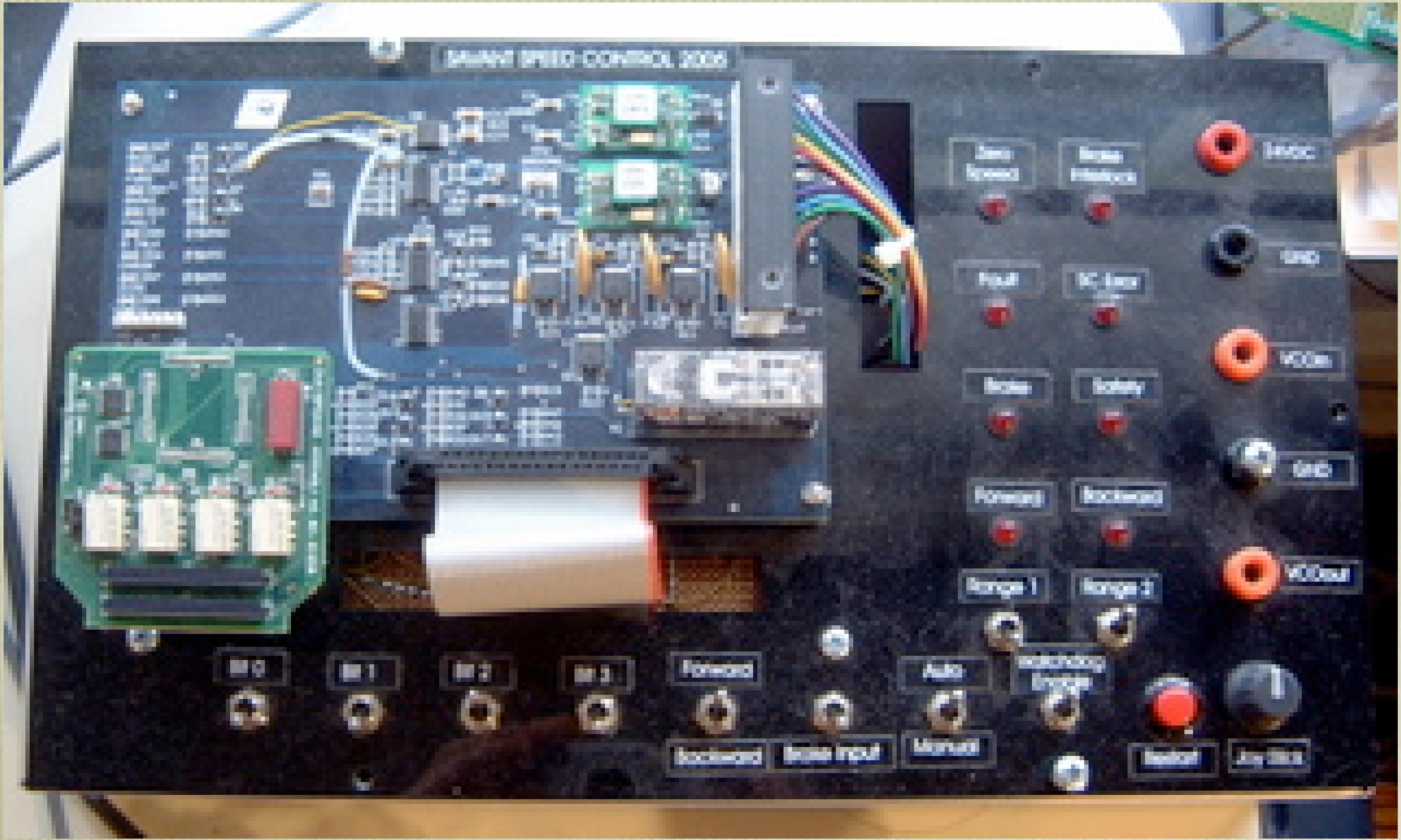


- Savant Automation produces automated-guided vehicles

Speed control

- Accepts speed requests from main computer
- Translates speed into voltages the drives understand
- Input and output:
  - Digital and analog
  - CAN bus
- Microchip `PIC18F4480`
  - 768 bytes RAM
  - 16K bytes flash

Speed control



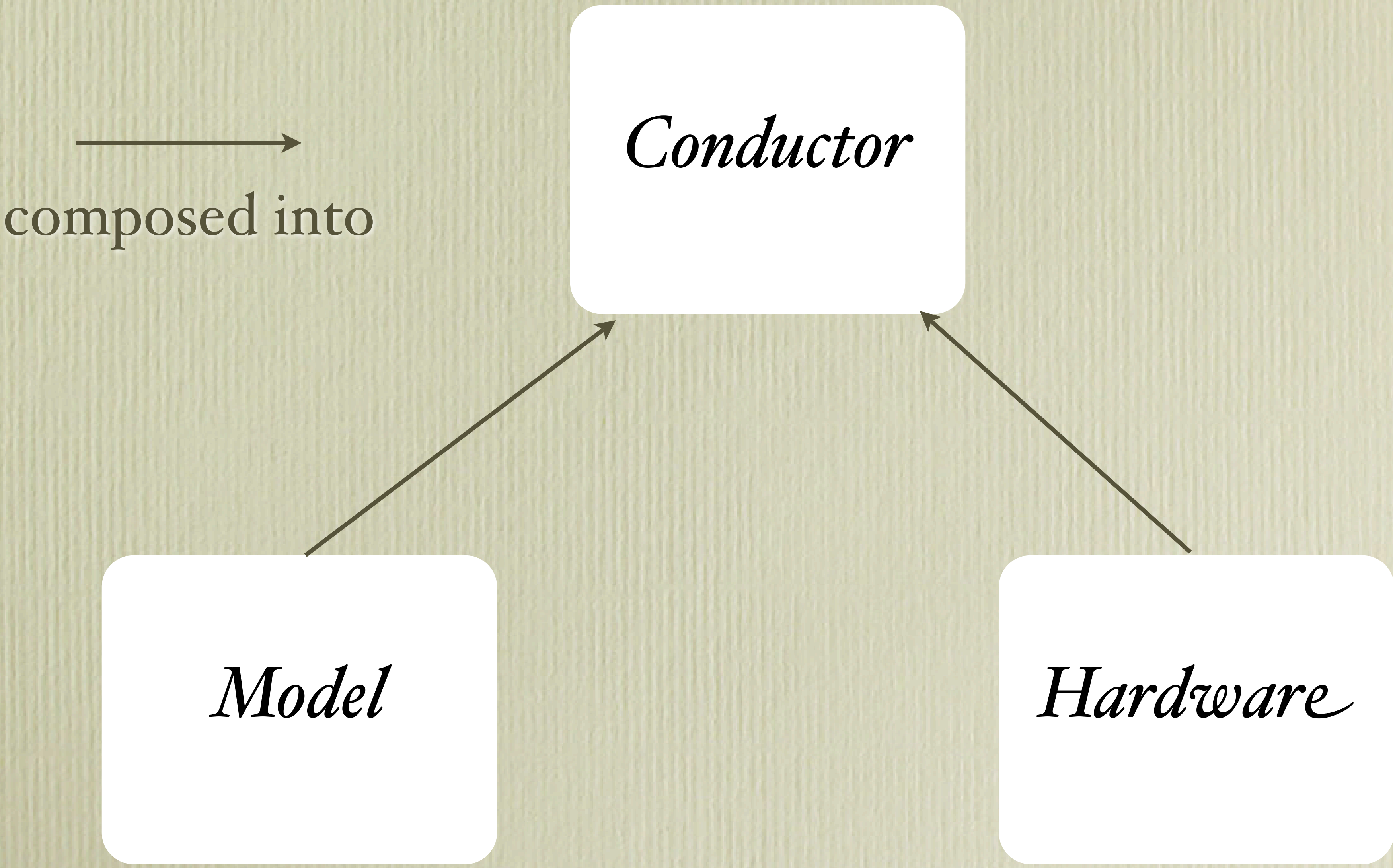
Tools

- Microchip C compiler and simulator
- Ruby
- Rake - Ruby make
  - Main build script
  - Easy injection of Ruby scripts into the build process
- Other Ruby scripts

Unit testing

- Created our own unit testing macros and runner
- Each test executable ran through the GUI simulator
- Blend of state-based and interaction-based testing
- Model-Conductor-Hardware made testing easier
- All of the code was developed test-first

Model-Conductor-Hardware



composed into

Conductor

Model

Hardware

Argent

- Inline code generation
- Ruby script
- Fast; easy to use

```
void testVoltageCalculatorWithPositiveVoltage()
{
...
}
void testVoltageCalculatorWithNegativeVoltage()
{
...
}
```

```
//[[$argent require “tests.rb”; generate_tests;$]]
int main(int argc, char* argv[]) {
  RUN_TEST(testVoltageCalculatorWithPositiveVoltage);
  RUN_TEST(testVoltageCalculatorWithNegativeVoltage);
}
//[[$end$]]
```

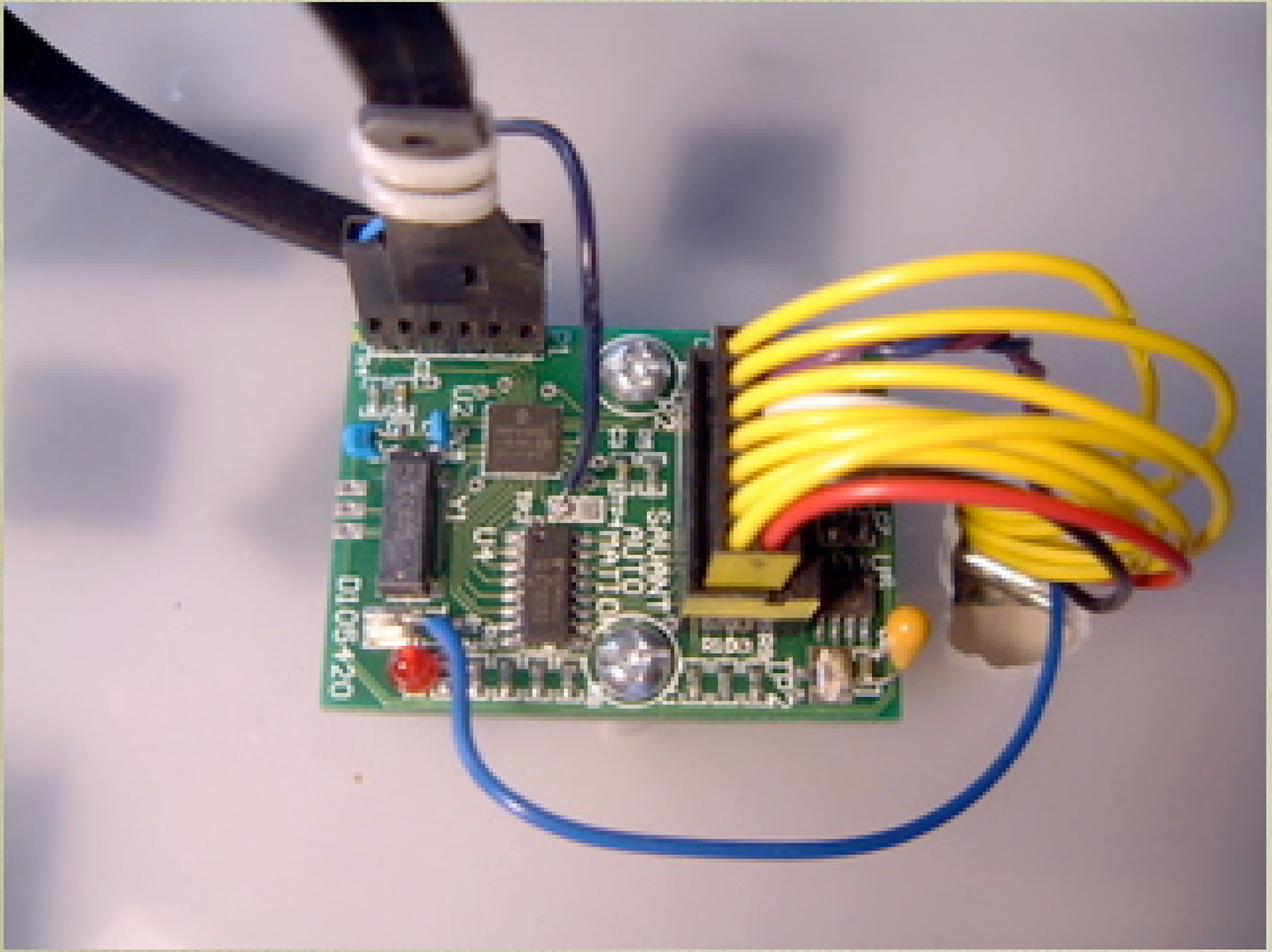
Positives

- Test-driven development resulted in good code
- Rake made for simple, straightforward builds
- Argent automatically detected and executed tests
- We learned a lot about programming this PIC

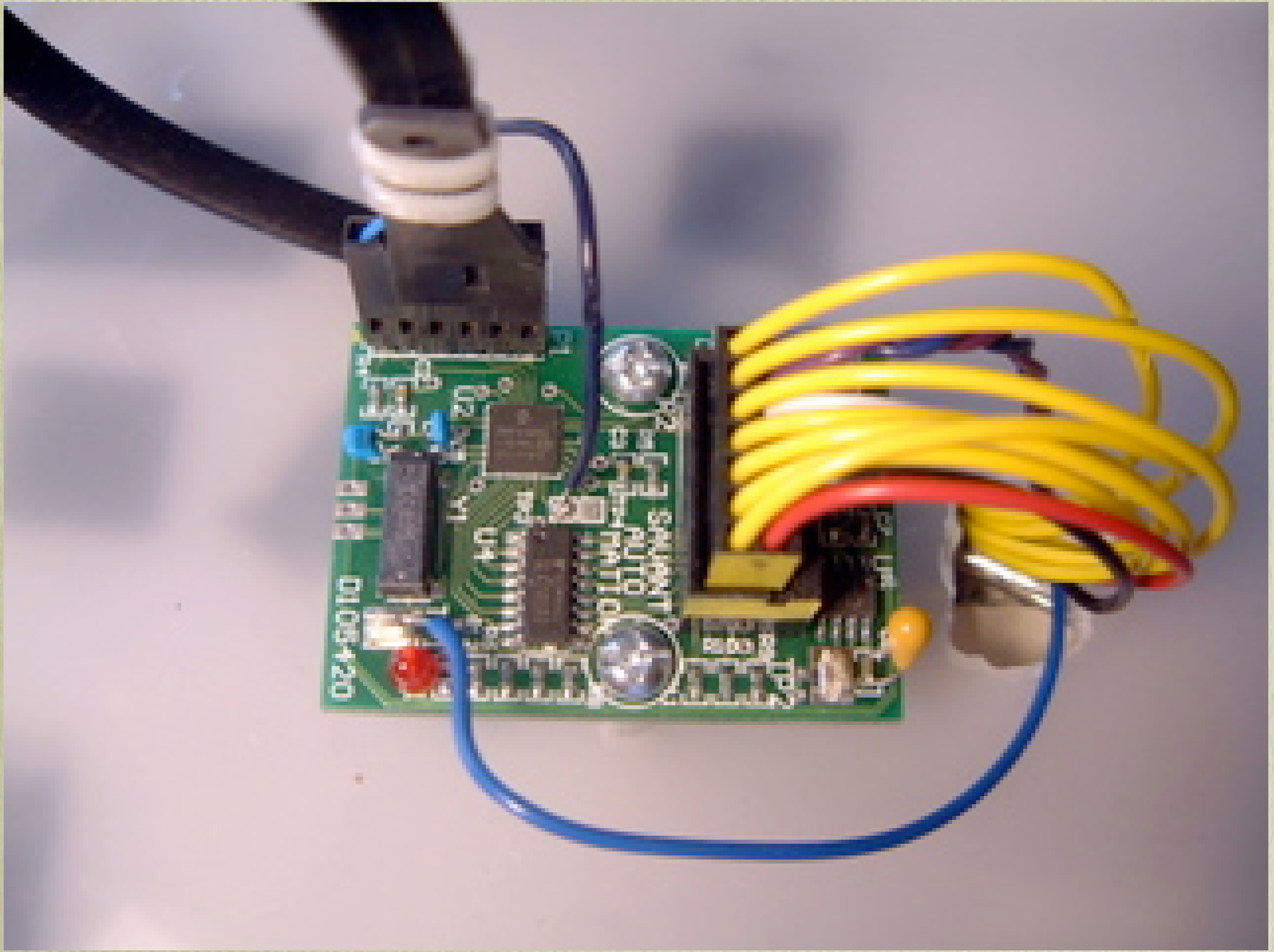
Negatives

- State-based tests were hard to maintain
- Manual system testing
- GUI simulator
  - slow
  - brittle

Battery monitor

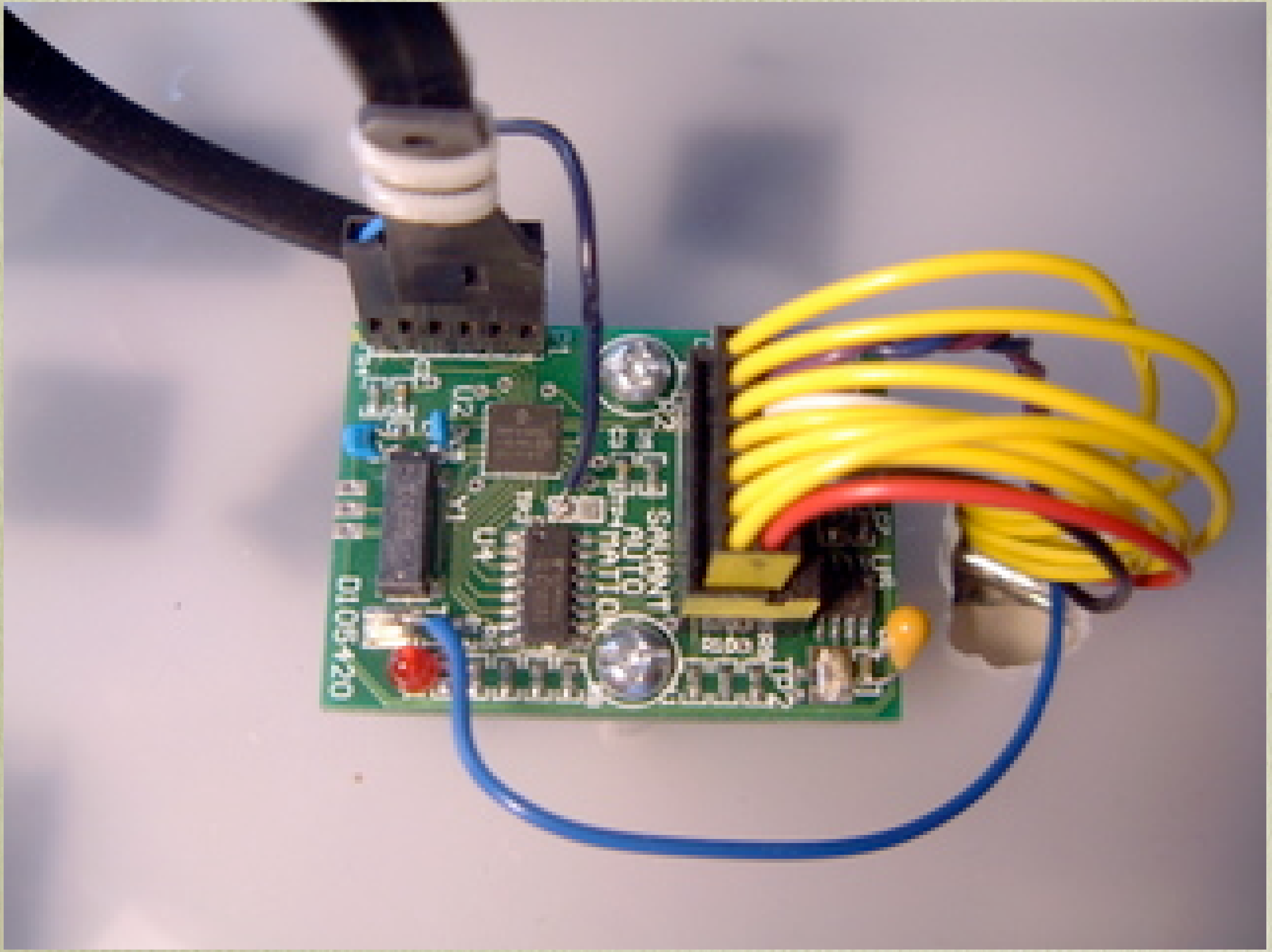


Battery monitor



Analog input

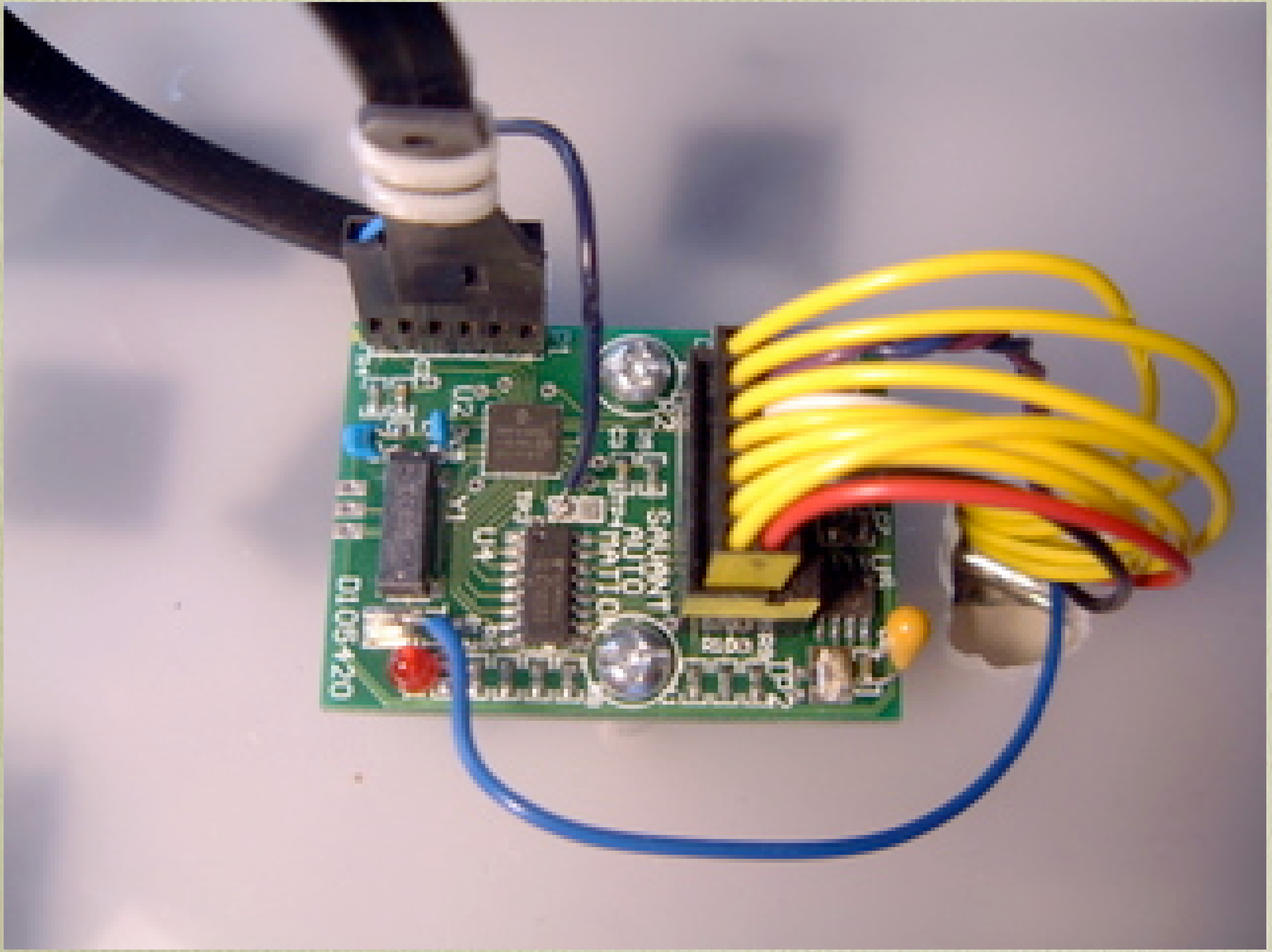
Battery monitor



Analog input

Digital output

Battery monitor

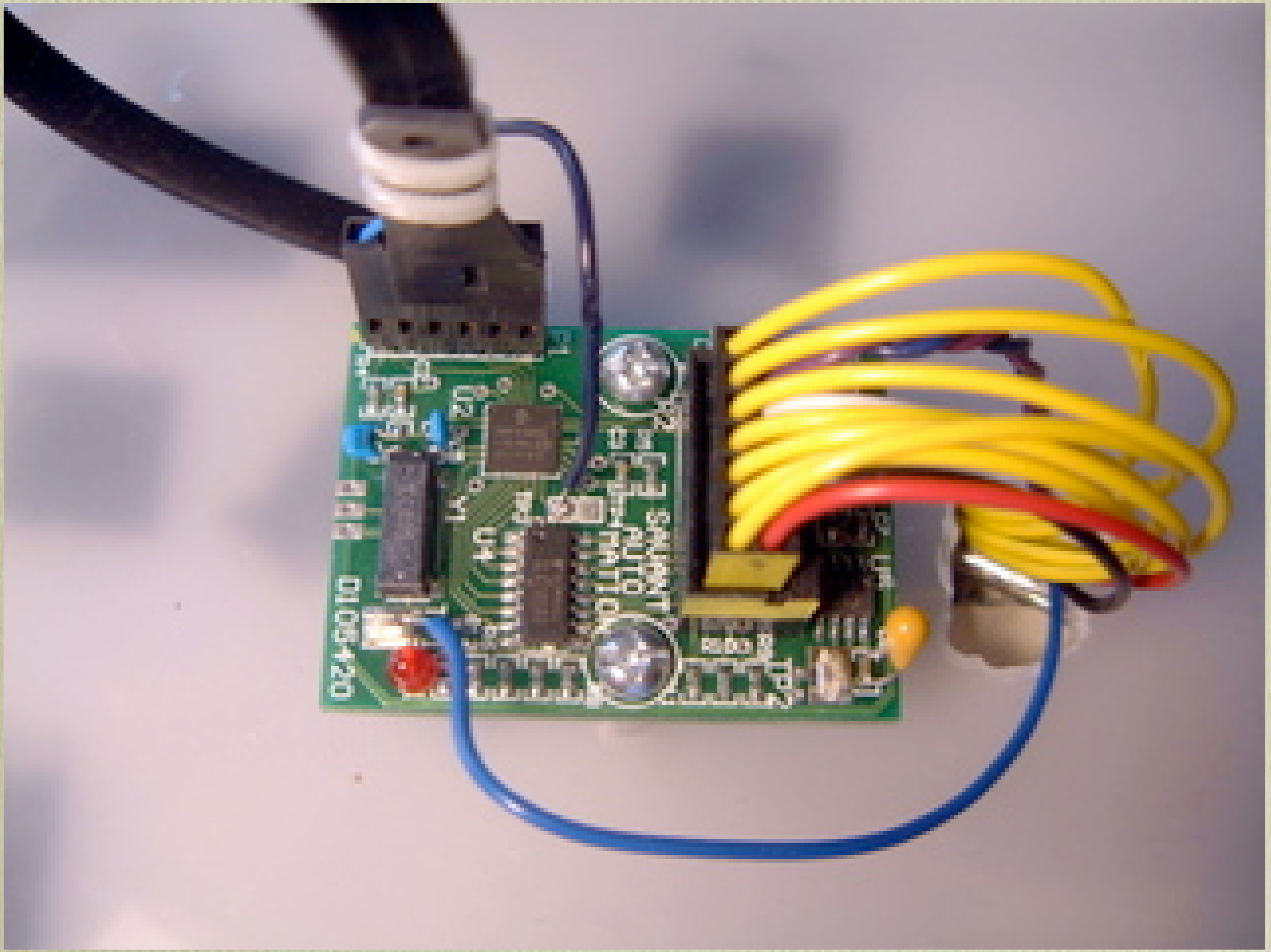


Analog input

Digital output

CAN output

Battery monitor



Analog input

Digital output

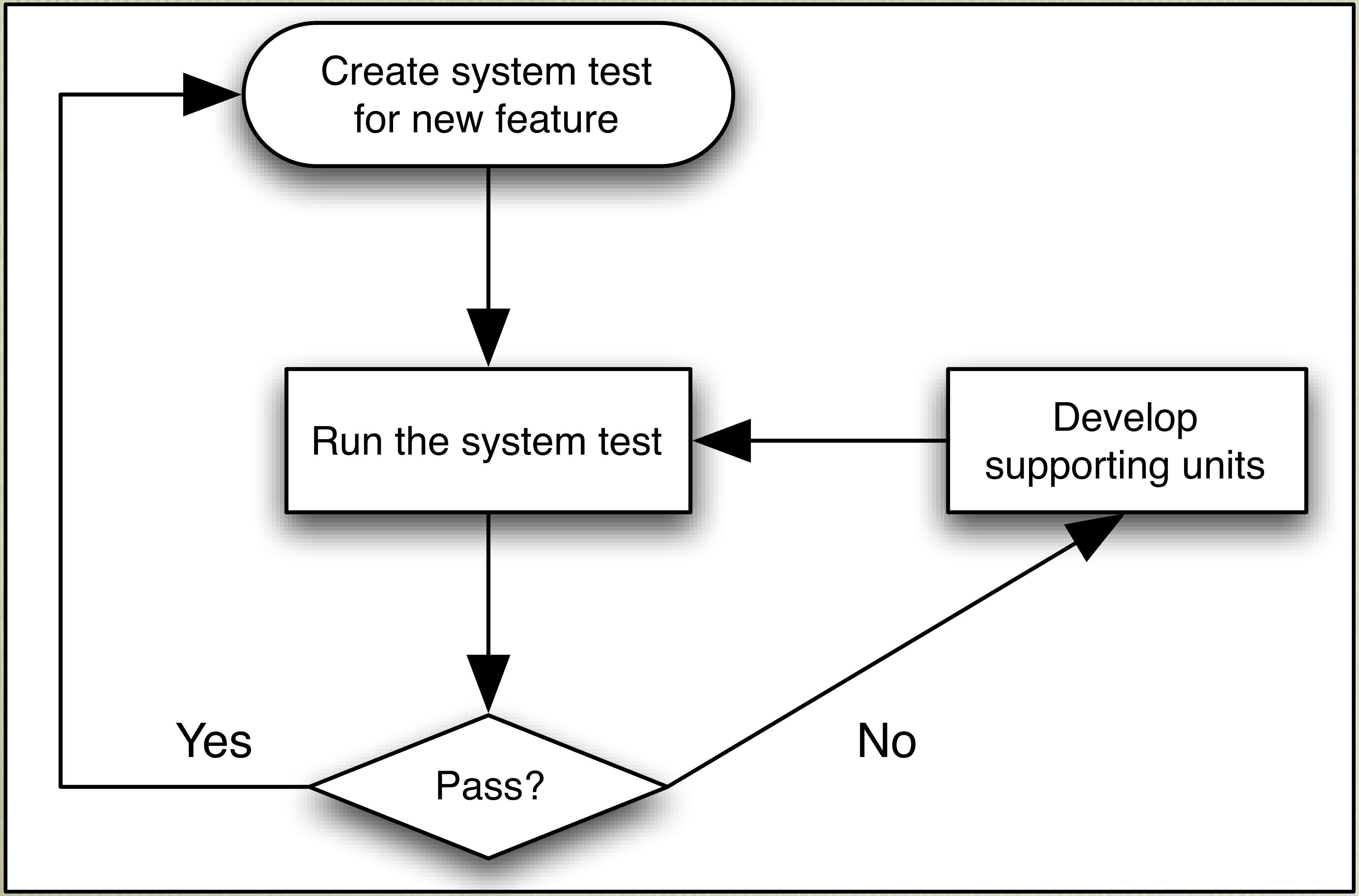
CAN output

Microchip
PIC18F4480

New tools

- IAR Systems workbench for PIC
  - better compiler
  - command-line simulator
- New simulator made the tests run much faster

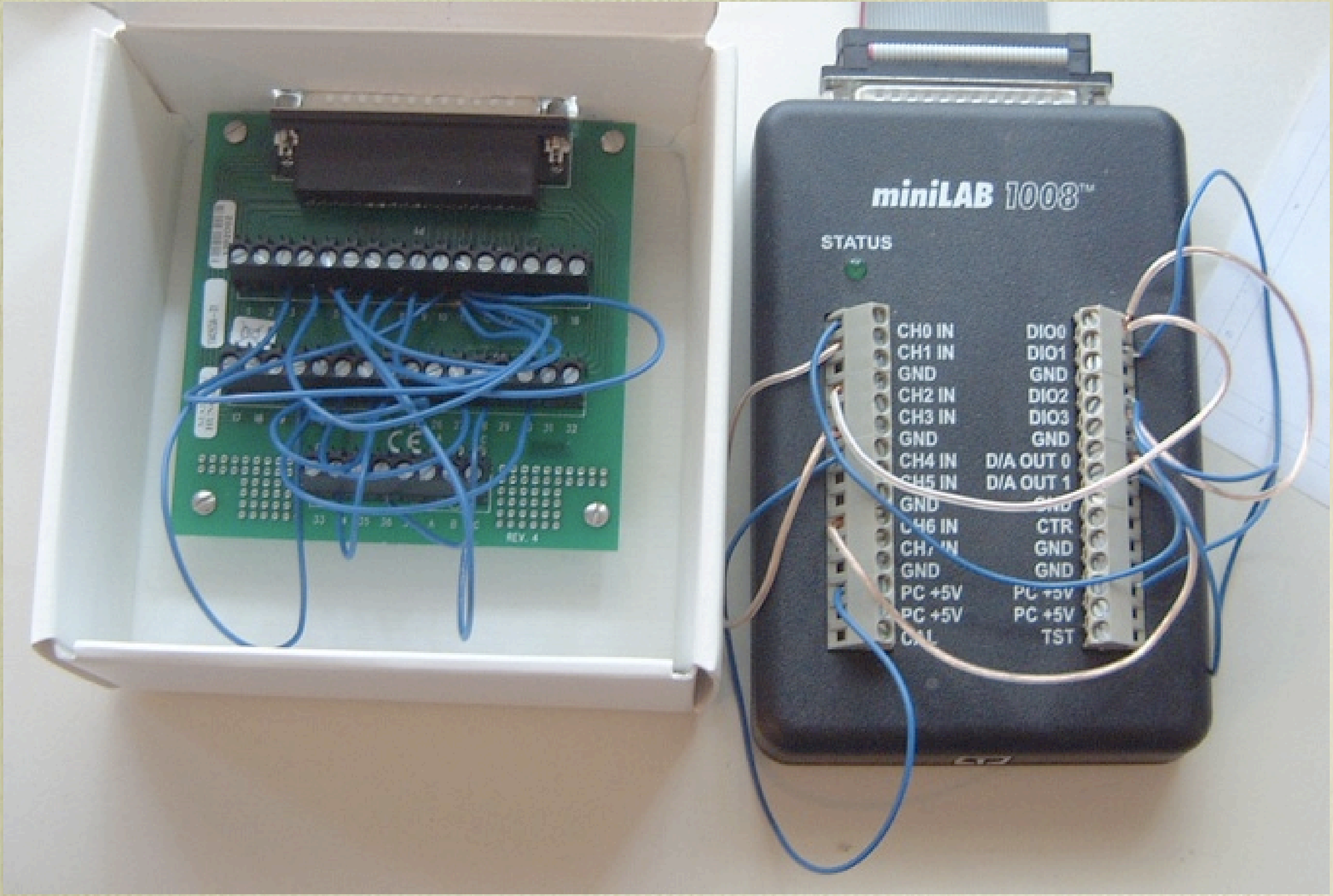
System testing



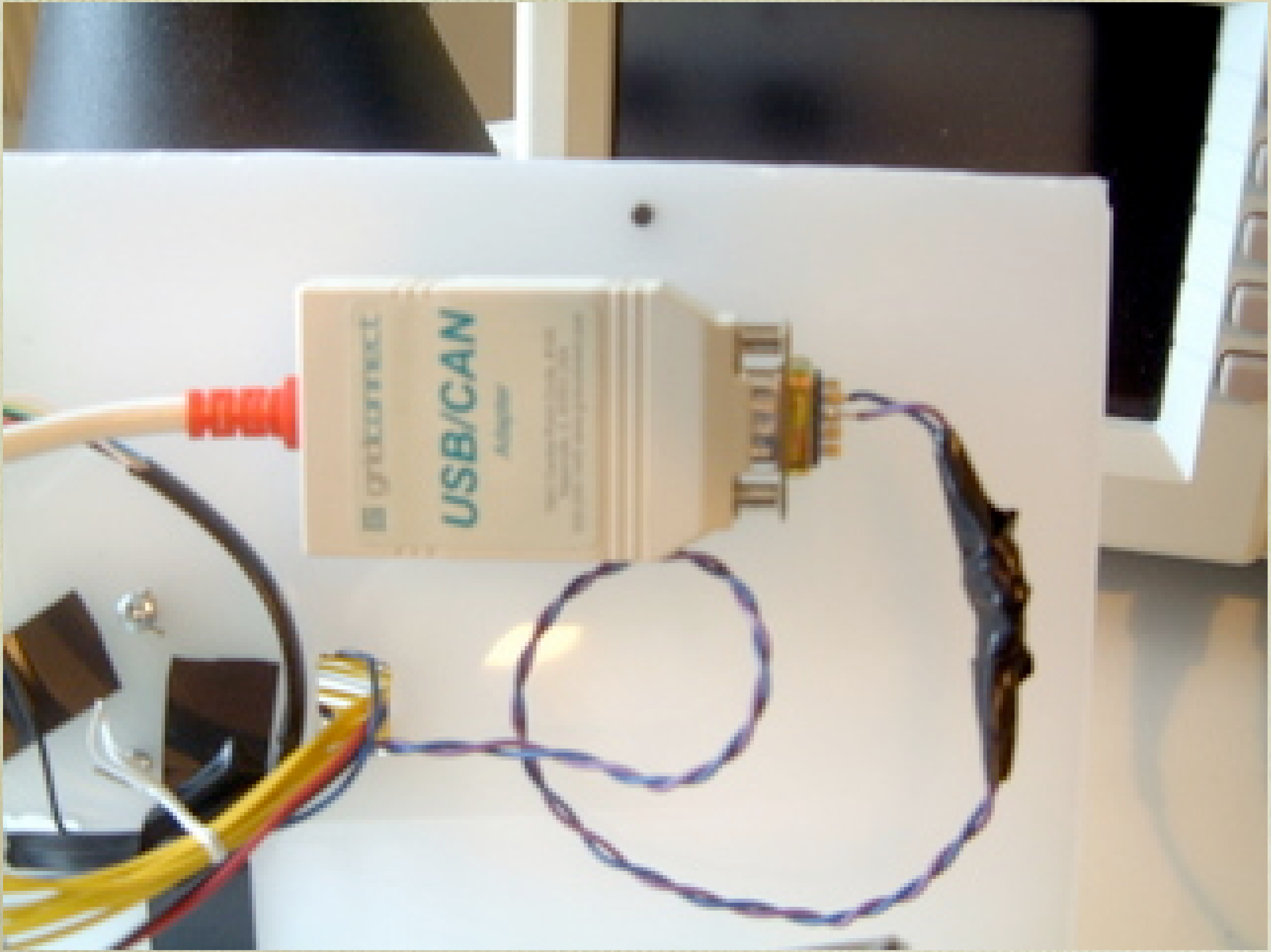
Supporting system tests

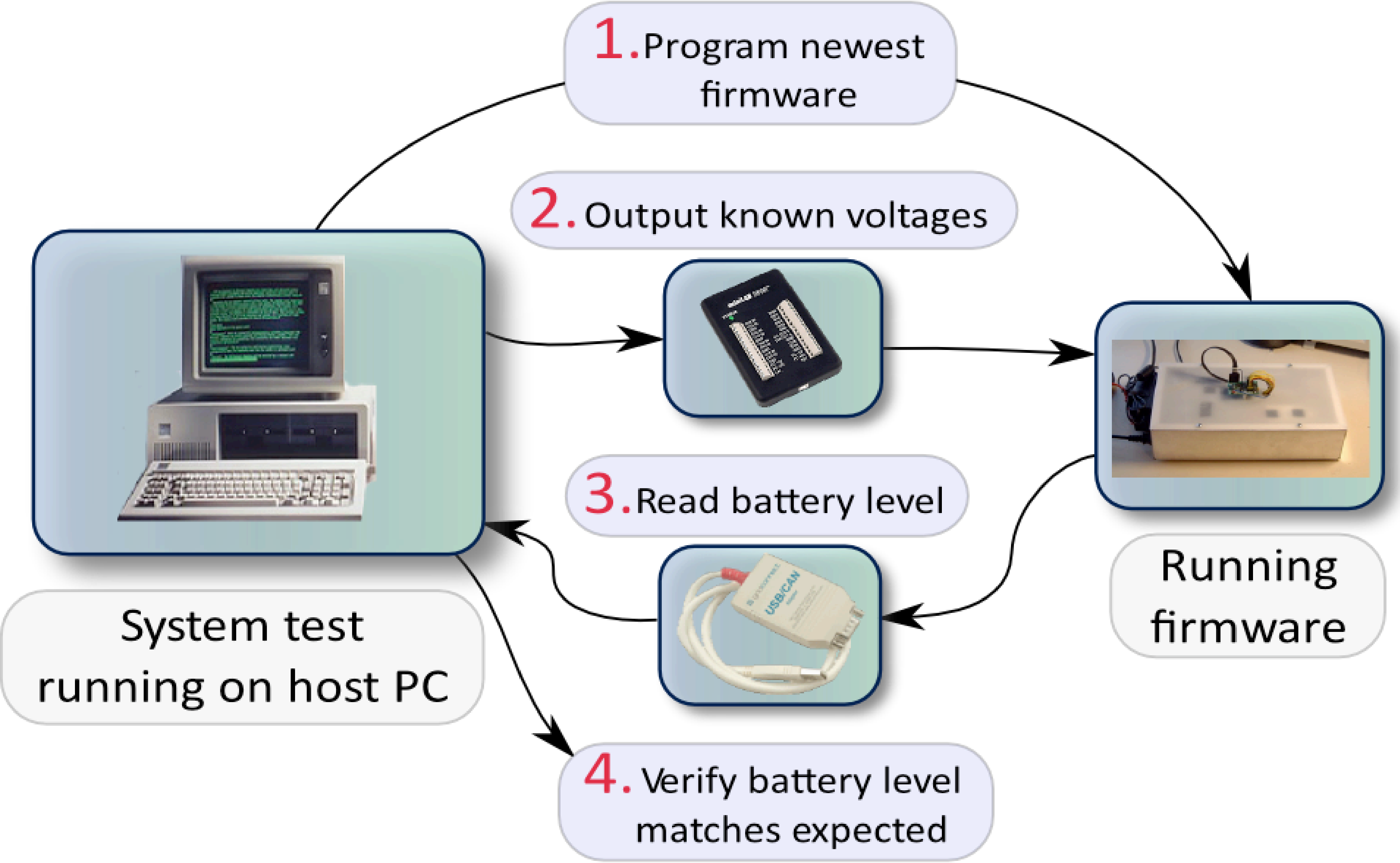
- Software:
  - Domain-specific language written using Ruby
- Hardware
  - Digital and analog input and output
  - CAN input and output
  - PIC programmer

miniLAB 1008



pCAN-USB





1. Program newest firmware

2. Output known voltages

3. Read battery level

4. Verify battery level matches expected

System test running on host PC

Running firmware

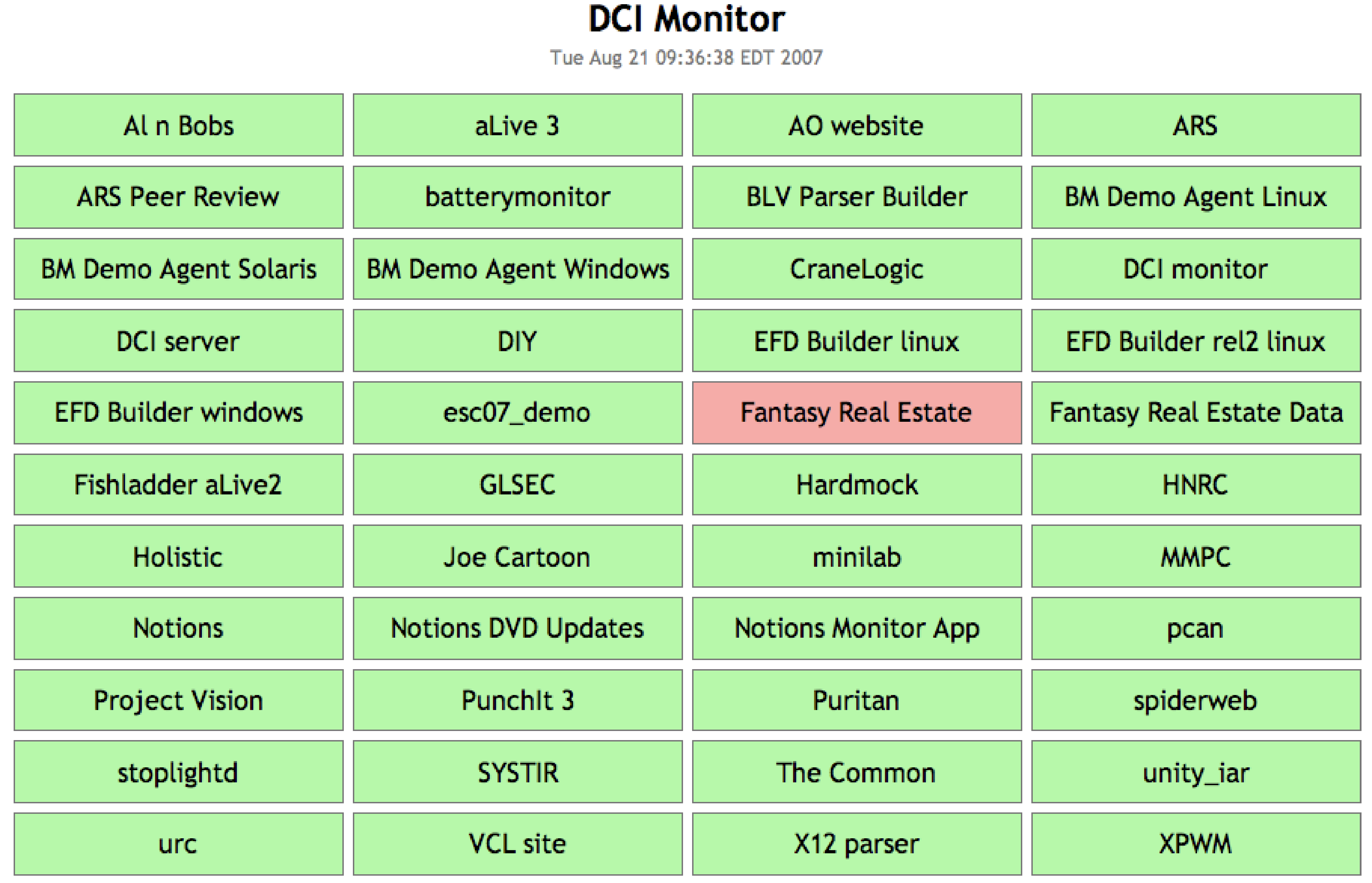
System test-driven development

- Test automation
- Choosing support hardware with command line and programmatic interfaces
- Diligence. System-test first!

Unit testing

- Much more interaction-based testing than speed control
- Mocks created automatically
  - Ruby script scans header file
  - Creates mock header and source
- Naming conventions dictate how to declare functions and how to call mock functions
- Argent generated code to automatically verify mocks

Continuous integration



DCI Monitor

Tue Aug 21 09:36:38 EDT 2007

Al n Bobs	aLive 3	AO website	ARS
ARS Peer Review	batterymonitor	BLV Parser Builder	BM Demo Agent Linux
BM Demo Agent Solaris	BM Demo Agent Windows	CraneLogic	DCI monitor
DCI server	DIY	EFD Builder linux	EFD Builder rel2 linux
EFD Builder windows	esc07_demo	Fantasy Real Estate	Fantasy Real Estate Data
Fishladder aLive2	GLSEC	Hardmock	HNRC
Holistic	Joe Cartoon	minilab	MMPC
Notions	Notions DVD Updates	Notions Monitor App	pcan
Project Vision	PunchIt 3	Puritan	spiderweb
stoplightd	SYSTIR	The Common	unity_iar
urc	VCL site	X12 parser	XPWM

Positives

- System tests provided automated regression testing of complete features
- Interaction-based testing made tests easier to write and less brittle
- Command line simulator made tests run faster
- Cool Ruby extensions

Negatives

- Expensive compiler and simulator (but worth every penny)
- Small project
  - did not last very long
  - did not stress our system testing strategy

vs.

Build your own tools

- Solve specific problems with custom tools
- Automate tedious processes
- Generate repetitive code

Resources

- Rake
  - rake.rubyforge.org
- Argent
  - rubyforge.org/project/argent
- Minilab Ruby gem
  - minilab.rubyforge.org

Resources

- Atomic Object embedded
  - www.atomicobject.com/pages/Embedded+Software
  - papers
  - Embedded Systems Conference 2007 demo project
- Agile Embedded Yahoo! group:
  - tech.groups.yahoo.com/group/AgileEmbedded

Thanks

- Bill Bereza, Mike Karlesky, and Greg Williams
- Matt Werner and Andrew Black
- Chad Fowler and Carl Erickson

Questions?