FOSDEM 2015
What’s new in GNAT GPL 2014 ?

Presented by

Tristan Gingold

Agenda

- GNAT GPL 2014
- Bareboard runtimes
- Ravenscar profile (technical)
- Boards
- Using and porting guide (technical)
- Demos

What is GNAT GPL ?

- GNAT compiler, based on gcc sources + AdaCore patches
- + IDE (gps), builder, ASIS tools…
- Released every year (June-ish)
- Many add-ons available: AWS, PolyORB, ASIS, GNATbench, AJIS, Aunit, GNATcoll, GtkAda, XML/Ada, Florist, SPARK

Targeted audience

- Academics: members of the GAP program.
- Students
- Free Software / Open Source developers

The license of the GNAT GPL runtime is GPL. Software built with GNAT GPL and linked with its runtime must follow the GPL.

GNAT GPL 2014

- New: includes SPARK 2014
- New: GNAT GPL for Bare Board ARM

SPARK 2014

Complete redesign:

- Provable subset of Ada 2012
- Use the GNAT front-end
- WhyML as intermediate language (instead of FDL)
- Use SMT solvers as automatic proof tools
- Support for Isabelle, Coq, …

SPARK 2014

- Uses Ada 2012 aspects for contracts (instead of special comments)
- Sound IEEE-754 floating point support
- Support of combination of test and proof
- See http://spark-2014.org

GNAT Bare Board for ARM

- Targets ARM Cortex M and ARM Cortex R
- Cortex A is not supported (often used with an OS)
- Comes with IDE (gps), builder (gprbuild), debugger (gdb)…
- … like other GNAT GPL ports

GNAT Bare Board for ARM

Runtimes:

- ZFP – Zero FootPrint
- Ravenscar-sfp
  - First GNAT GPL release with a ravenscar-sfp runtime

Bare Board: restricted runtimes

No full-runtimes:

- No obvious storage for files
- Reduced memory size
- Reduced power

ZFP

- Almost the smallest possible runtime
  - System, Unchecked_Conversion, Machine_Code, Interfaces, …
- Can build an application without code from the runtime.
- Still include software engineering features: packages, generics, child units …

ZFP

Also includes (require code from the runtime):

- Secondary stack
    - To return unconstrained types
- Last chance handler
    - No exception propagation (but local handlers supported)
- Library-level tagged types

Ravenscar

Ravenscar is a profile (subset) of the tasking

For hard real-time applications

For safety-critical applications

Part of the Ada standard

Efficient implementation, with small footprint

Ravenscar: tasking model

Not enforced, but 2 common patterns:

- Cyclic / periodic tasks
    - Eg: compute position by reading sensors (speed, gyroscopes)
- Reactive tasks
    - Run on events, generated by an interrupt or by another tasks

Ravenscar: tasking model

Inter-tasks communication only by protected objects

- No rendez-vous

No easy way to multiplex inputs

- Eg: serial output driver for logs from multiple tasks

Ravenscar-sfp

Ravenscar small foot print

- Runtime with ravenscar tasking
- Based on ZFP for the sequential part
- No underlying OS – designed for bareboards

Ravenscar-sfp

2 parts:

- The tasking kernel (in system.bb)
- The ‘usual’ runtime
  - Ada units defined in the ARM
  - Units to implement high-level Ada constructs

Ravenscar kernel

- Scheduler
  - Follow Ravenscar semantic: FIFO within Priorities
- Clock and Timer
  - For Ada.Real_Time.Clock and delay until
- Interrupts
  - For clock, and Attach_Handler pragma (aspect)

Scheduler

Real-time scheduler

- A task can be preempted by an higher priority task
- Woken up by an interrupt or by the end of a delay
- FIFO within priority: order is deterministic
  - Simplify (and make possible) schedule analysis
  - But prevent multiplexing

Scheduler

- The highest priority task runs until it is blocked:
  - Either by a delay statement
  - Or by calling an entry (of a protected object) whose barrier is false

Protected types

- No locks: not needed by Ravenscar
- Exclusion achieved by Priority.
  - For multiprocessors: need a spin-lock
- At most one entry per protected object
- Entry queue length is 1
- => Task to wake-up is known.

Exclusion In Protected types

- Ceiling Locking policy:
  - Within a protected object, the priority is raised to the priority of the object
  - Can only raise the priority (not decrease it)
  - Avoid priority inversion and deadlocks.
- No blocking actions (delay, entries, …) allowed within a protected object

Exclusion In Protected types

Consequence: while a task is executing a protected object

- its priority is >= than priority of all other potential callers
- It cannot be blocked
- Can only be pre-empted by tasks with higher priorities
- These tasks cannot call the protected object

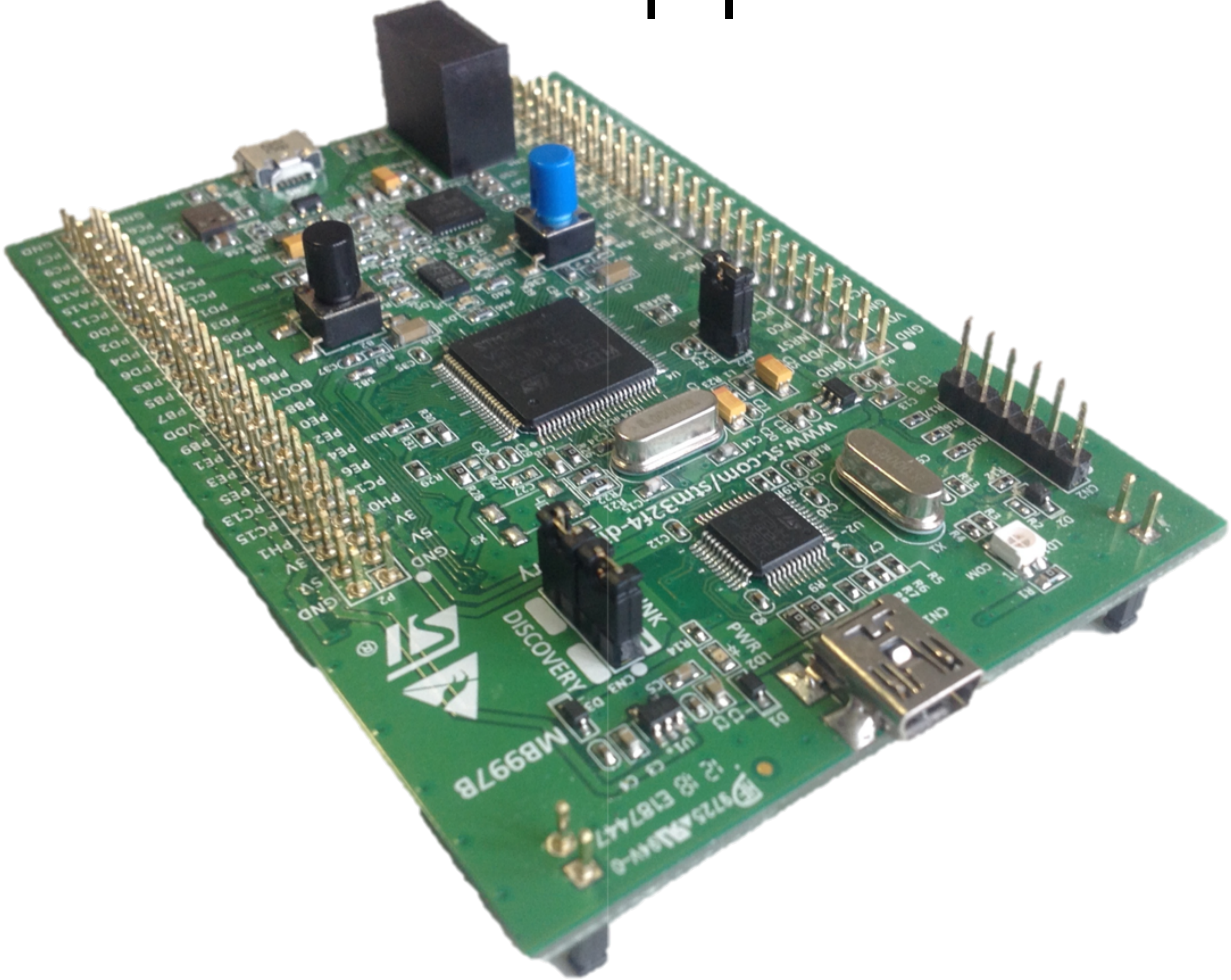
=> Mutual exclusion

Interrupts

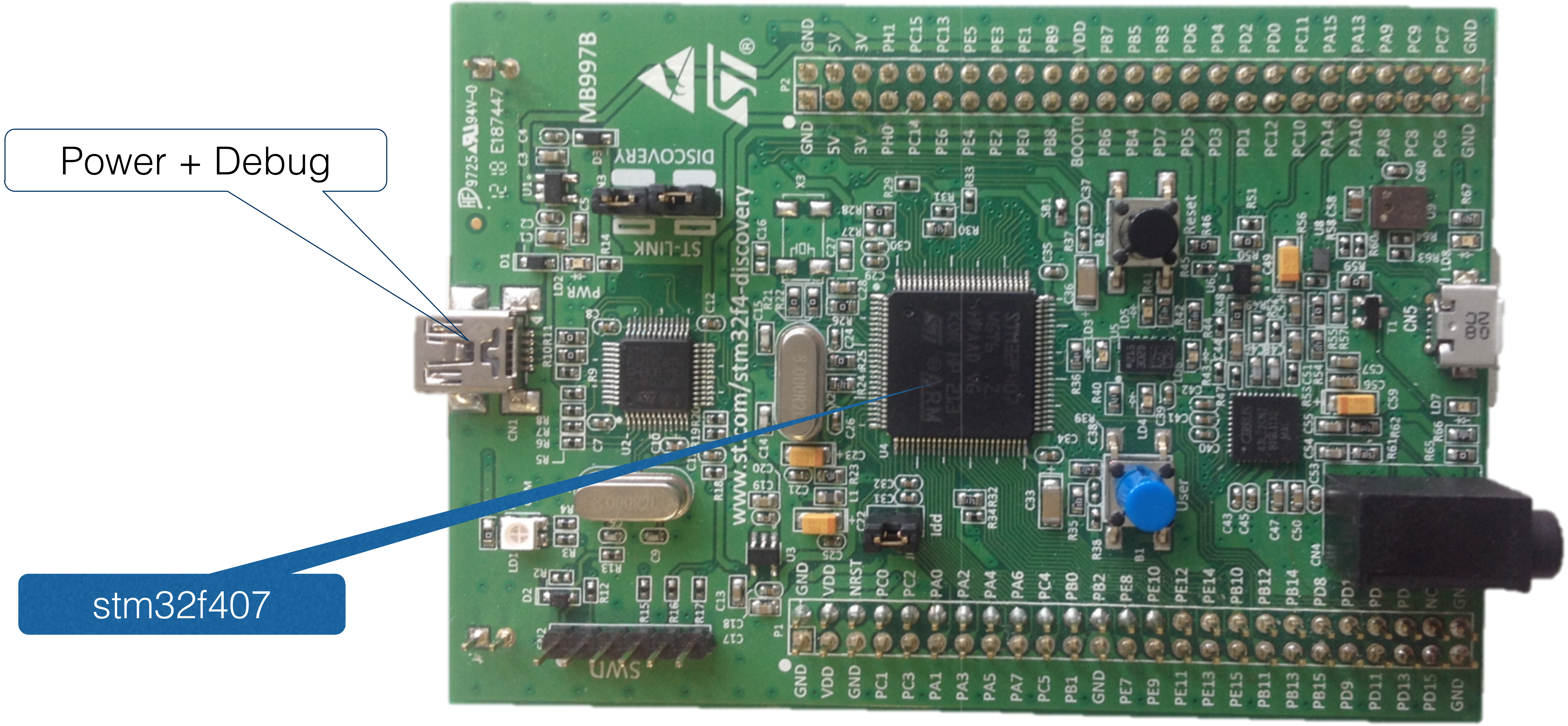
A protected procedure can be attached to an interrupt

- Support of interrupts within the language ☺
- Easy way to connect to interrupts
- Ceiling priority must be an interrupt priority
- Interrupts at lower priority are masked within the protected object
  - Provide mutual exclusion

Board supported



Stm32f4-discovery

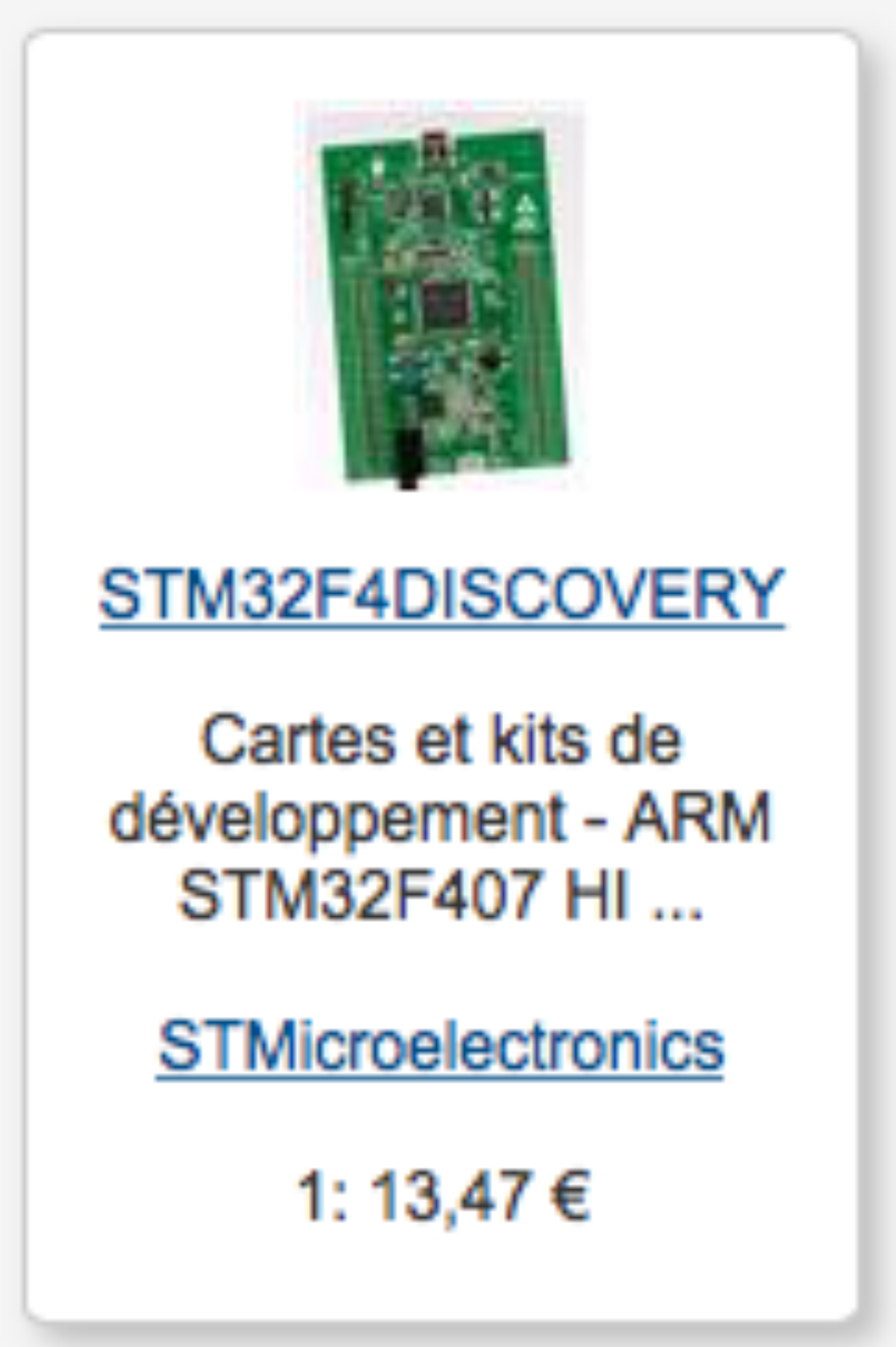


Power + Debug

stm32f407

Why stm32f4 ?

- Cheap and easily available:

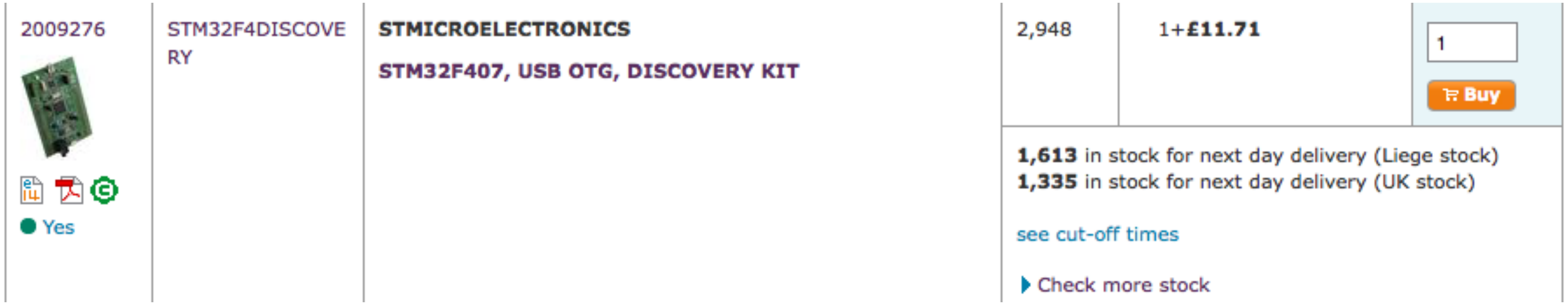


STM32F4DISCOVERY

Cartes et kits de développement - ARM STM32F407 HI ...

STMicroelectronics

1: 13,47 €

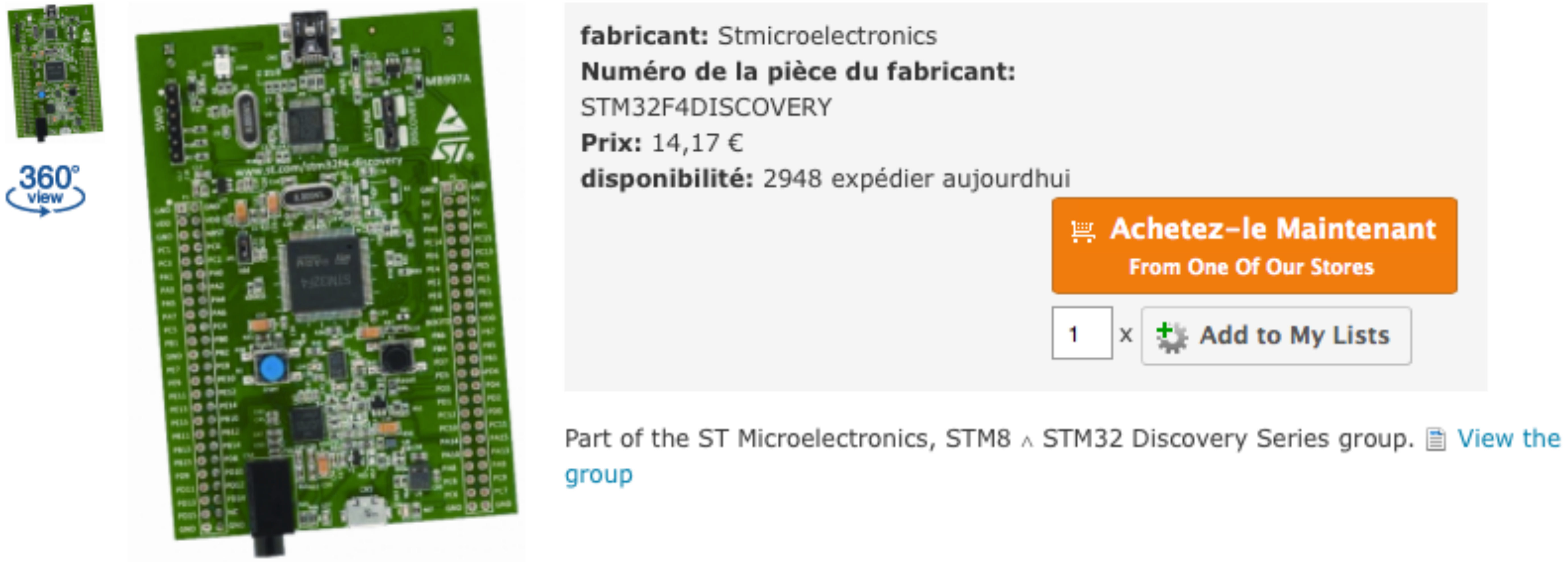


2009276 ● Yes	STM32F4DISCOVERY	**STMICROELECTRONICS** **STM32F407, USB OTG, DISCOVERY KIT**	2,948	1+**£11.71**	1 Buy

1,613 in stock for next day delivery (Liege stock)
1,335 in stock for next day delivery (UK stock)

see cut-off times

Check more stock



fabricant: Stmicroelectronics
Numéro de la pièce du fabricant:
STM32F4DISCOVERY
Prix: 14,17 €
disponibilité: 2948 expédier aujourdhui

Part of the ST Microelectronics, STM8 ∧ STM32 Discovery Series group. View the group

Why stm32f4

Easy to use

- Include a probe
- Open tools to flash and debug the board:
  - st-util (https://github.com/texane/stlink)
  - Openocd (http://openocd.sourceforge.net)
- Works with gdb!

Why stm32f4

- Very common
- Cpu (cortex m4f) is a nice microcontroller
- May devices included in the chip
    - USB, serial, gpio, timers, …
- Lots of I/O on the discovery board

Building a program

Must use gprbuild:

```
$ gprbuild --RTS=arm-eabi/ravenscar-sfp-stm32f4 --target=arm-eabi -Pleds.gpr


arm-eabi-gcc -c -fcallgraph-info=su,da -g leds.adb

gprbind leds.bexch

arm-eabi-gnatbind leds.ali

arm-eabi-gcc -c b__leds.adb

arm-eabi-gcc leds.o -o leds
```

A little bit heavy, but will be improved.

Building a program

User project file (required)

Target (also required)

```
gprbuild --RTS=arm-eabi/ravenscar-sfp-stm32f4 --target=arm-eabi -Pleds.gpr
```

Runtime path (not a name)
Either absolute or search in install dir

Build sub-configuration

Build for RAM:

```
gprbuild --RTS=arm-eabi/ravenscar-sfp-stm32f4 --target=arm-eabi
-Pleds.gpr —XLOADER=RAM
```

Application will be loaded in RAM.

Build for Flash:

```
gprbuild --RTS=arm-eabi/ravenscar-sfp-stm32f4 --target=arm-eabi
-Pleds.gpr —XLOADER=FLASH
```

Loading to the board

1. Start debug agent

```
$ st-util
```

(On windows: from a CMD window)
Could use openocd.

2. Load with gdb

```
$ arm-eabi-gdb leds
(gdb) target remote :4242
(gdb) load
(gdb) c
```

Loading to the board

Notes:

- Reset the board before downloading
- If program is loaded in FLASH, will stay after power-off

Other boards ???

There are many many many Cortex-M boards

- We cannot provide runtimes for each board
- We needed to start with one board
- We tried to make porting easier

Runtime location

The runtime can be anywhere

- You need to give its path to GPRbuild
- Implicit search in the install directory

Start by copying and renaming an existing runtime.

Runtime compilation

The runtime can be easily recompiled.

```
$ gprbuild –P path/ravenscar-sfp-stm32f4
```

The runtime comes with a project file

You can recompile it with debug info, optimization off…

GCC flags

The runtime contains a configuration file: runtime.xml

- Read by gprconfig
- Contains compiler, binder and linker switches
- Can specify switches like –mcpu=xxx, -msoft-float, …
- No need to modify gcc spec files

Linker scripts

The runtime contains the linker scripts

- Referenced by gprconfig
- Describe memory map
- May differ according to –XLOADER=

Starting code

Code executed from the reset vector

- Copy initialized data from FLASH to RAM (if starting from FLASH)
- Clear .bss
- Enable the FPU
- Setup PLL

Starting code

.data copy, enable FPU, clear .bss:

- Code already written. May require some adjustments if ported

PLL setup:

- Code highly device and board dependent
- Usually very similar within a family.

At this point, non-tasking program should work !

Cortex-M

Cortex-M is the arm v7 variant for micro-controllers

Other variants:

- Cortex-R: for real-time (not very common)
- Cortex-A: for application (very common in smart phones)

Porting ravenscar runtime

- FPU or no FPU (eg: M4 vs M4F)
- Speed
- Number of interrupts

Constants in System.BB.Parameters:

```
Clock_Frequency : constant := 168_000_000;

Has_FPU : constant Boolean := True;
--  Set to true if core has a FPU

Number_Of_Interrupt_ID : constant := 85;
```

Ada.Interrupts.Names

Declare names of the interrupts

- Not required (not used by the runtime)
- Useful for users
- Very device specific

Priority is defined by the user (must be an interrupt priority)

Tasks may not be at interrupt priority.

Publishing new Runtimes

AdaCore has already ported the ravenscar runtime

- For Atmel SAM4SD32C (SAM4S Xplained Pro board)
- For STM32F429I-DISCO

Public repository

AdaCore plan to create a Github repo

- Not a commitment (currently only a plan)
- Ravenscar runtime from GNAT GPL
- Examples
- Device drivers
- Wiki

Public repository

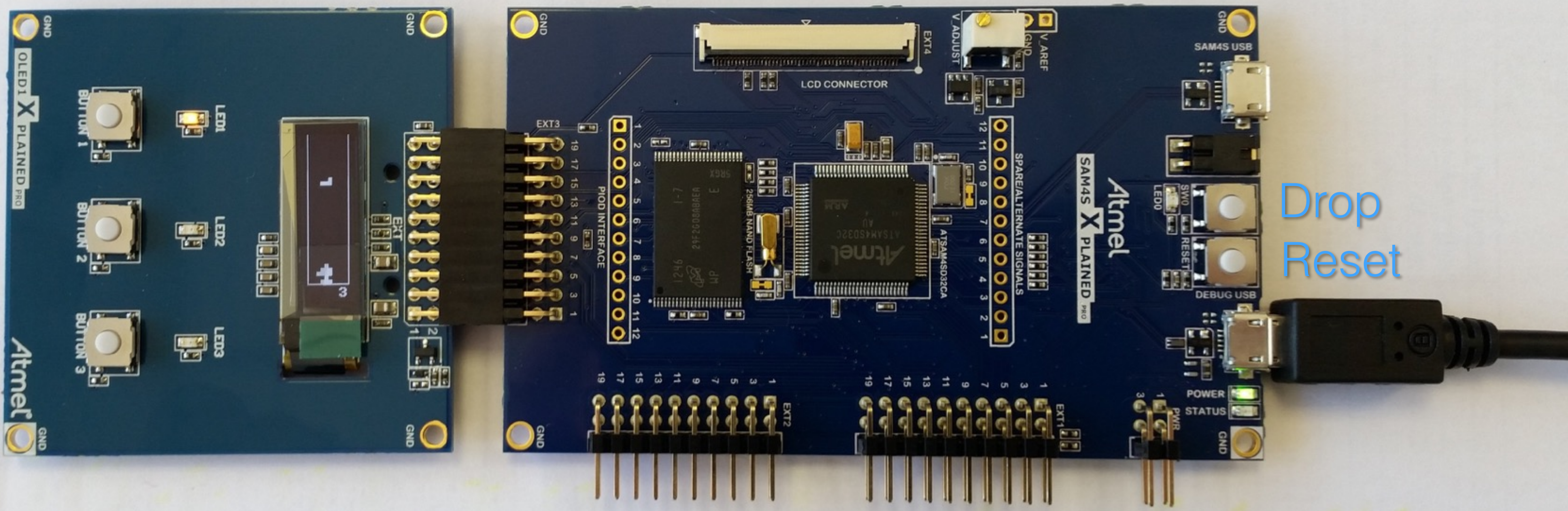
Hobbyist can:

- Fork the repo
- Port the runtime to a new board or new cpu
- Be referenced on the wiki
- (No commitments yet)

Demo 1: Tetris

- Board: Atmel sam4s Xplained Pro
- Application: a Tetris game
- Runtime: ravenscar-sfp (for sam4sd32c)
- Core written in Spark2014 and proven
- See http://blog.adacore.com/tetris-in-spark-on-arm-cortex-m4

Atmel sam4s Xplained pro demo



Move Right

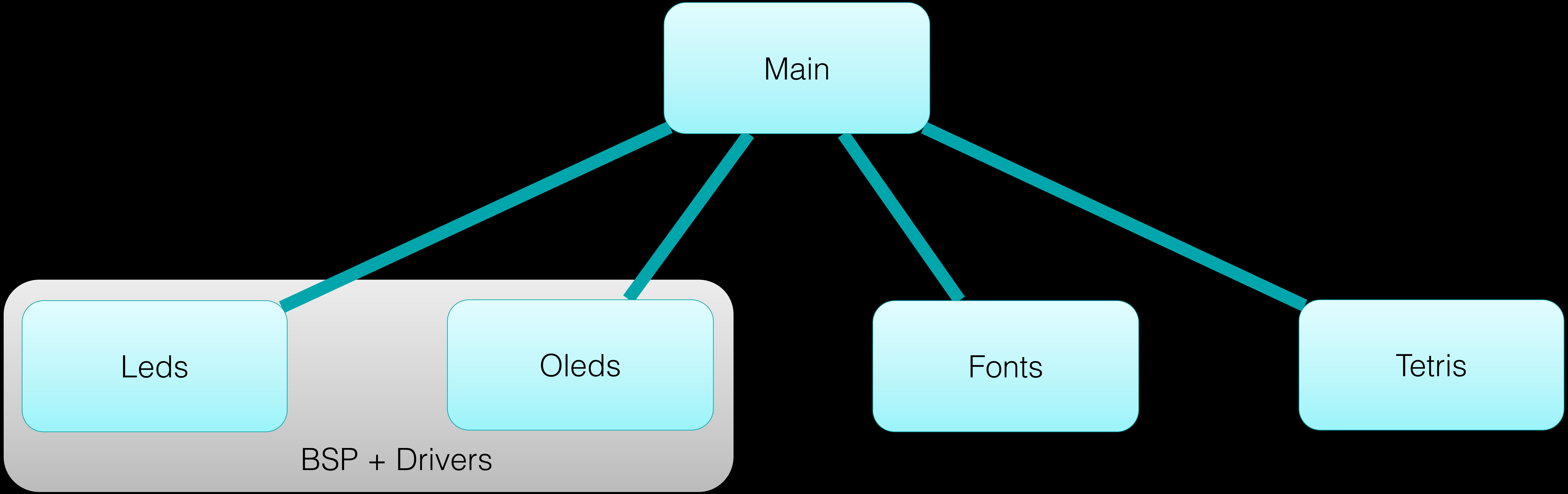
Rotate

Move Left

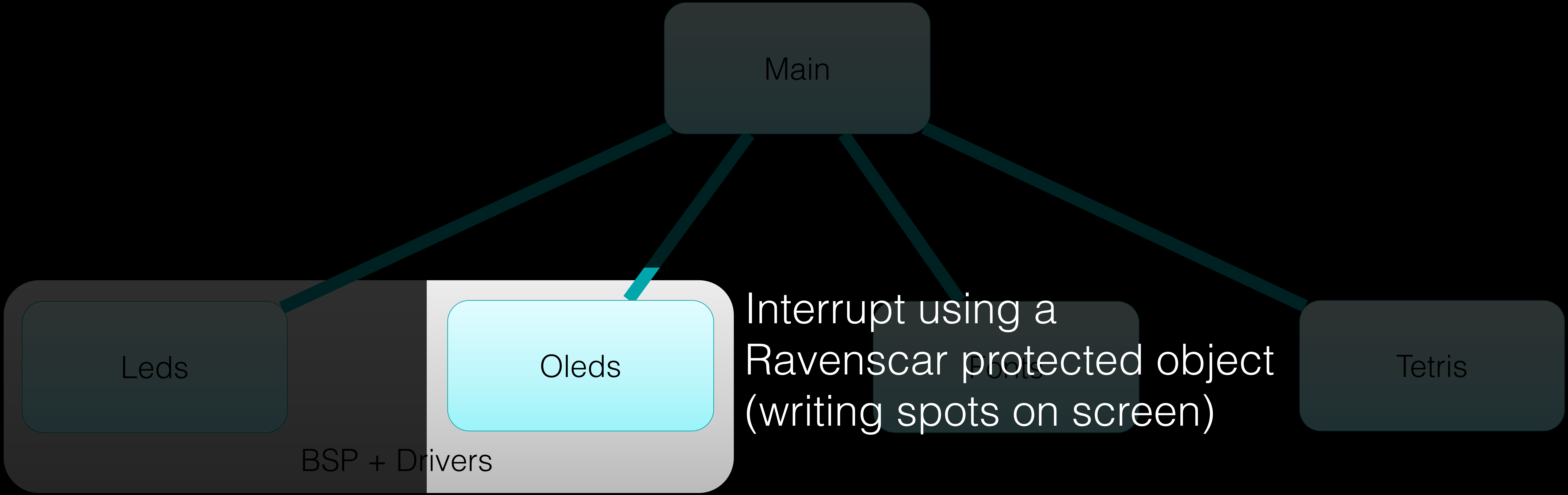
Drop

Reset

Tetris SW Architecture

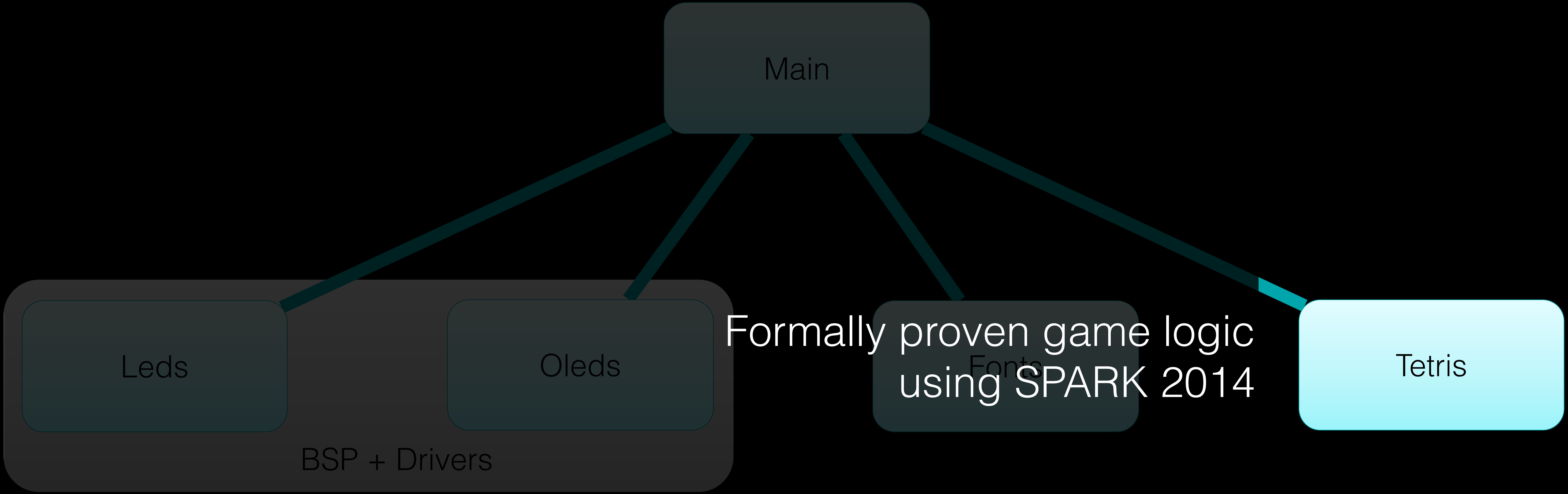


Tetris SW Architecture



Interrupt using a Ravenscar protected object (writing spots on screen)

Tetris SW Architecture

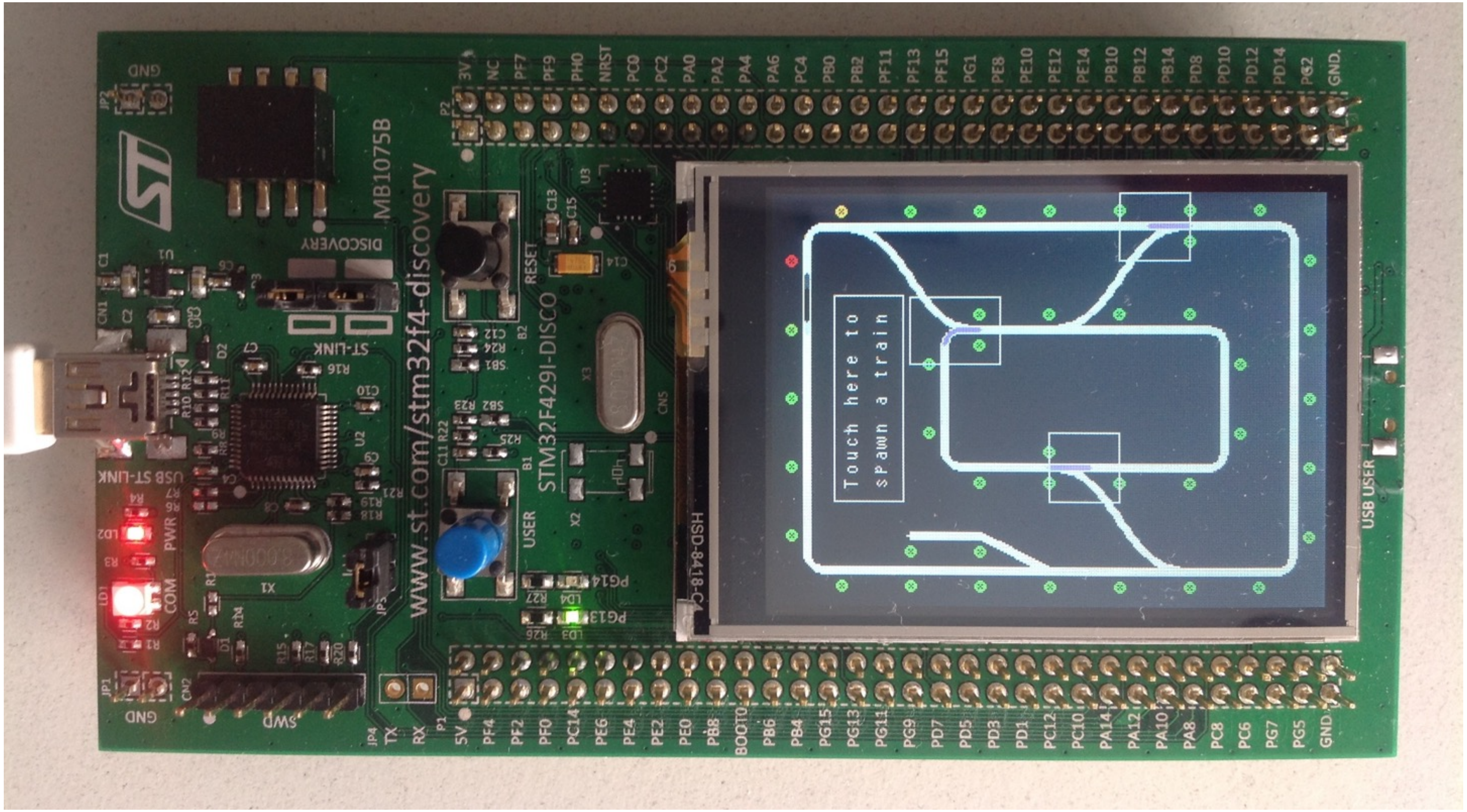


Formally proven game logic
using SPARK 2014

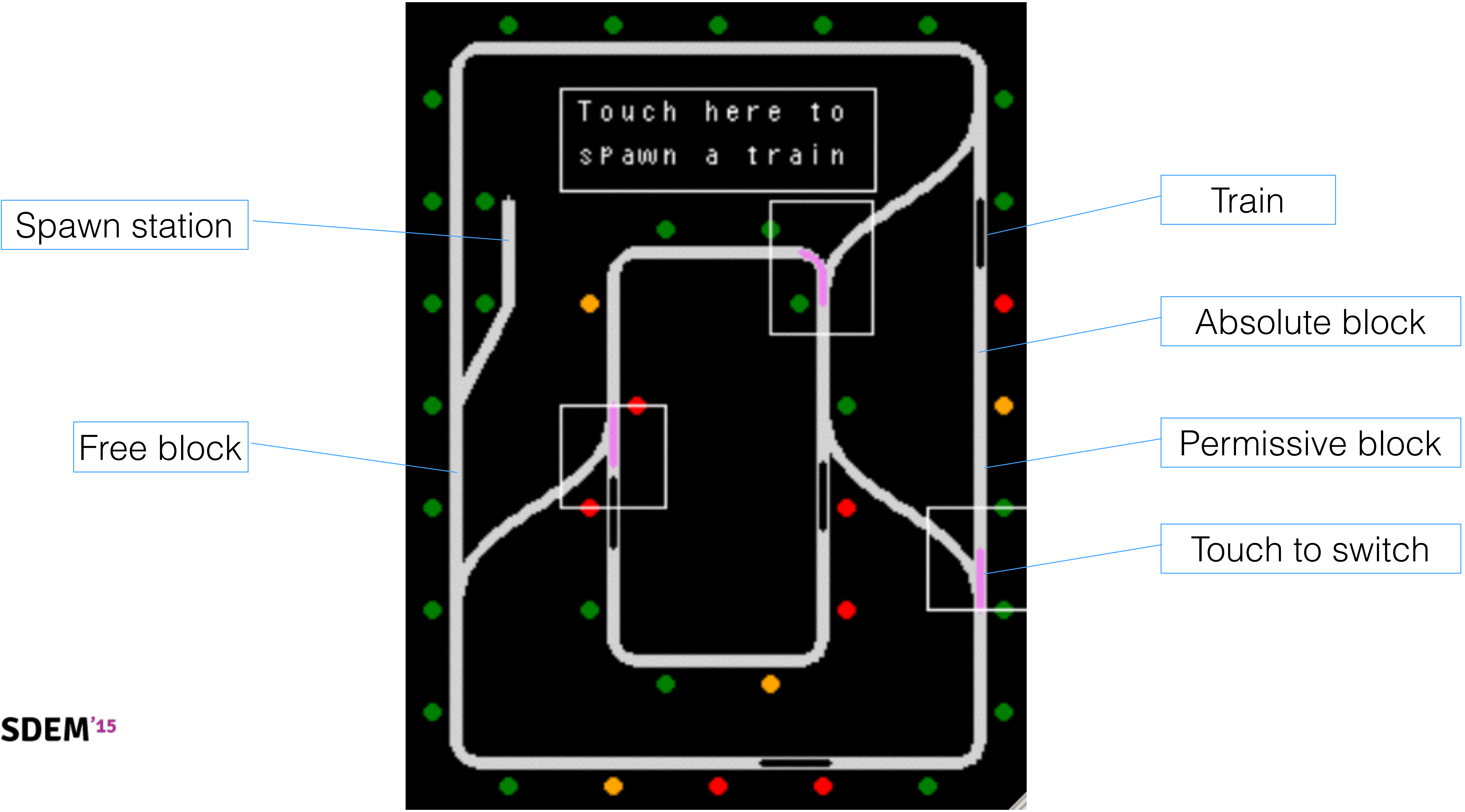
Demo 2: Railway Signaling

- Board: STM32F429I-DISCO
- Application: Railway signaling simulation
- Runtime: ravenscar-sfp (for stm32f429) + drivers
- Signaling written in Spark2014 and proven

STM32F429 board demo

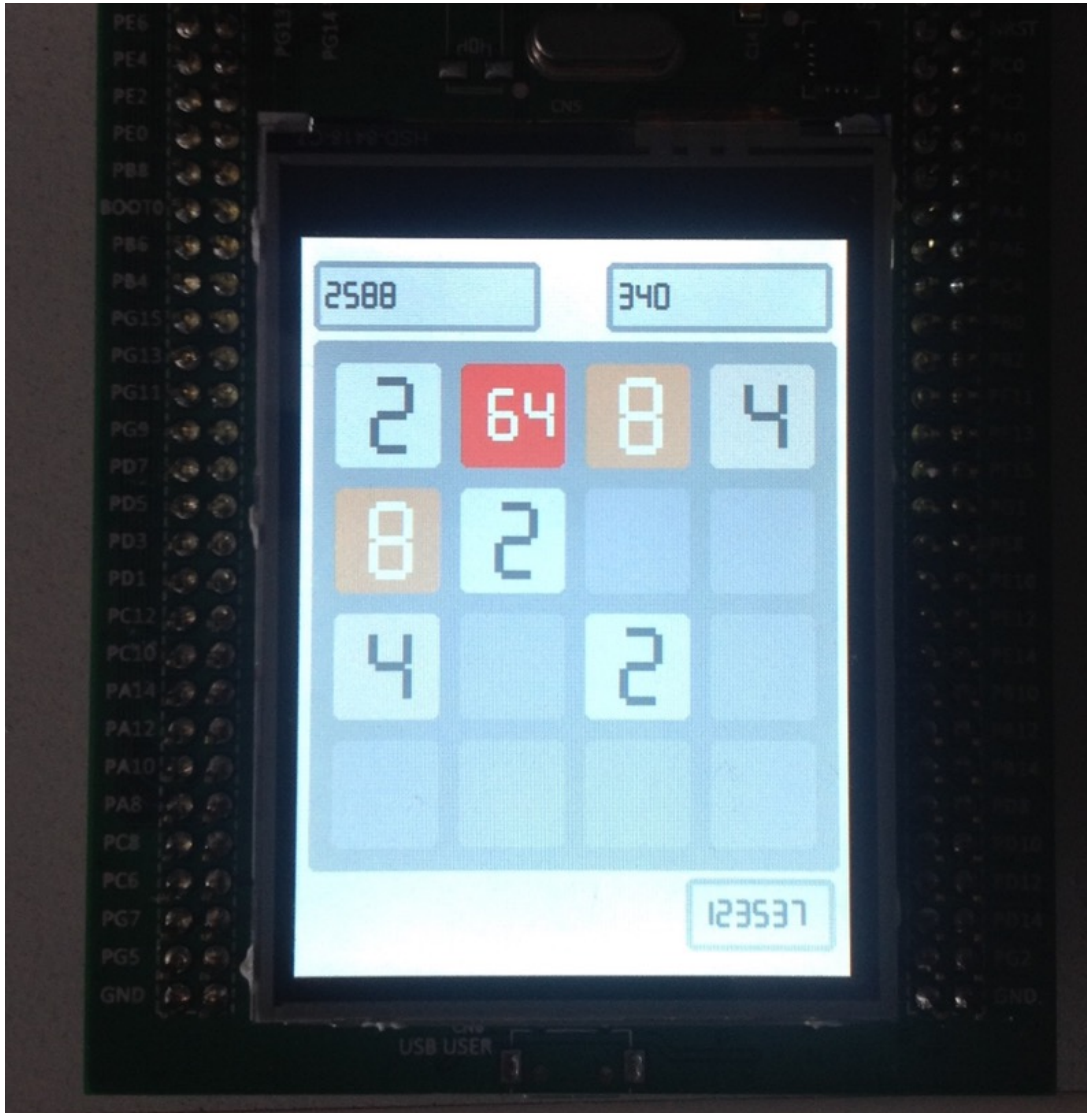


Railway Signaling



FOSDEM'15

2048 (by students)



Gravity simulation