Embedded Programming with Ada

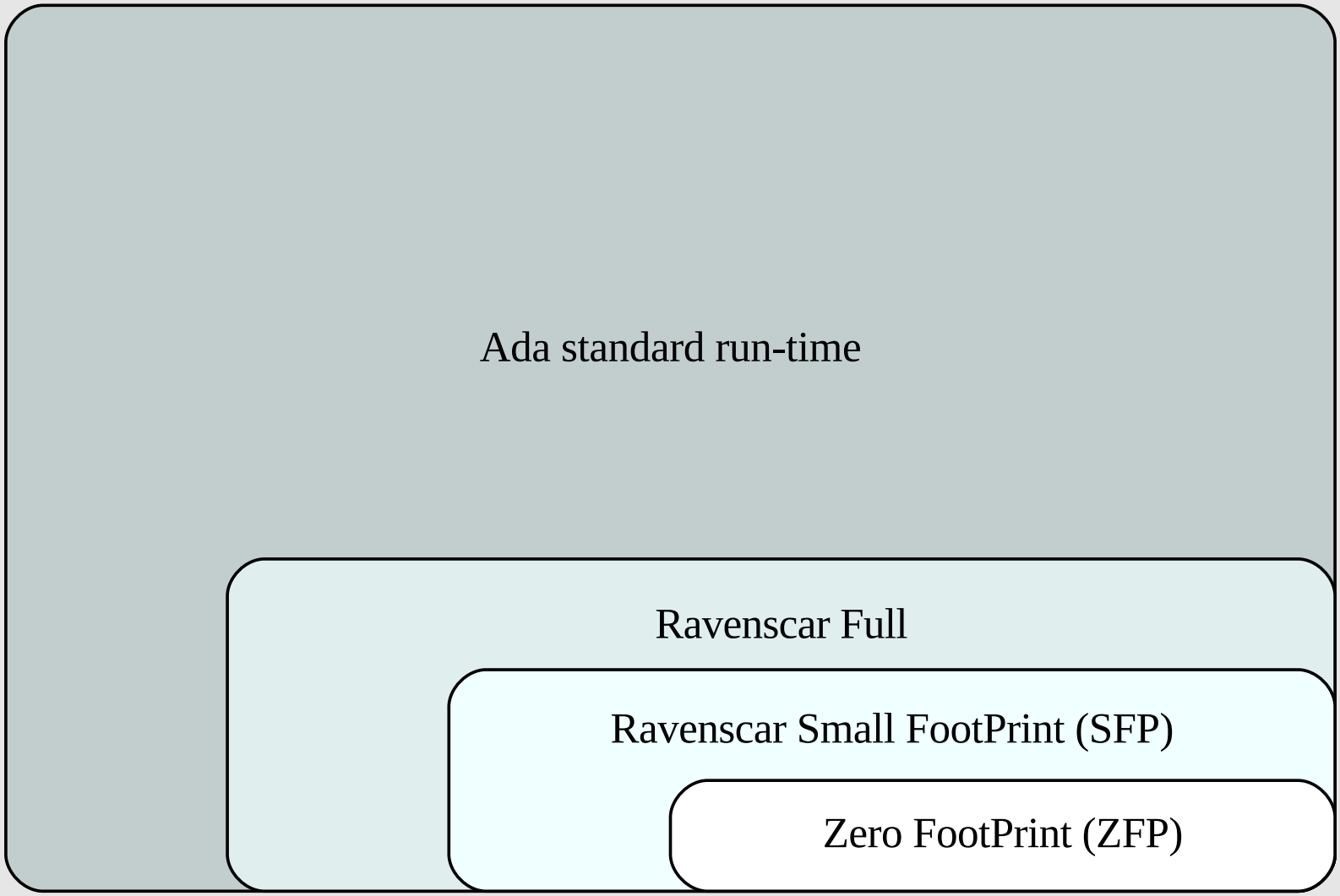
Fabien Chouteau

Embedded Software Engineer at AdaCore

Twitter : @DesChips
GitHub : Fabien-Chouteau
Hackaday.io: Fabien.C

Ravenscar Tasking

Runtimes profiles



Tasks

```
--  Task type declaration
task type My_Task (Prio : System.Priority)
  with Priority => Prio;
```

```
T1 : My_Task (Prio => 1);
T2 : My_Task (Prio => 2);
```

Time

```
task body My_Task is
   Period       : constant Time_Span := Milliseconds (100);

   Next_Release : Time := Clock + Period;
   --  Set Initial release time
begin
   loop
      --  Suspend My_Task until the Clock is greater
      --  than Next_Release.
      delay until Next_Release;
      --  Compute the next release time
      Next_Release := Next_Release + Period;

      --  Do something really cool at 10Hz...
   end loop;
end My_Task;
```

Mutual exclusion and shared resources

```
protected My_Protected_Object
  with Priority => 3
is

   procedure Set_Data (Data : Integer);
   --  Protected procedues can read and/or modifiy the
   --  protected data.

   function Data return Integer;
   --  Protected functions can only read the protected data

private
   --  Protected data are declared in the private part
   PO_Data : Integer := 0;
end;
```

Synchronization 1/2

```
protected My_Protected_Object is
   entry Wait_For_Signal;
   procedure Send_Signal;
private
   We_Have_A_Signal : Boolean := False;
end My_Protected_Object;
```

Synchronization 2/2

```
protected body My_Protected_Object is

   entry Wait_For_Signal when We_Have_A_Signal is
   begin
       We_Have_A_Signal := False;
   end Wait_For_Signal;

   procedure Send_Signal is
   begin
       We_Have_A_Signal := True;
   end Send_Signal;

end My_Protected_Object;
```

Interrupt handling 1/2

```
protected My_Protected_Object
  with Interrupt_Priority => 255
is
   entry Get_Next_Character (C : out Character);

private
   procedure UART_Interrupt_Handler
           with Attach_Handler => UART_Interrupt;

   Received_Character  : Character := ASCII.NUL;
   We_Have_A_Character : Boolean := False;
end;
```

Interrupt handling 2/2

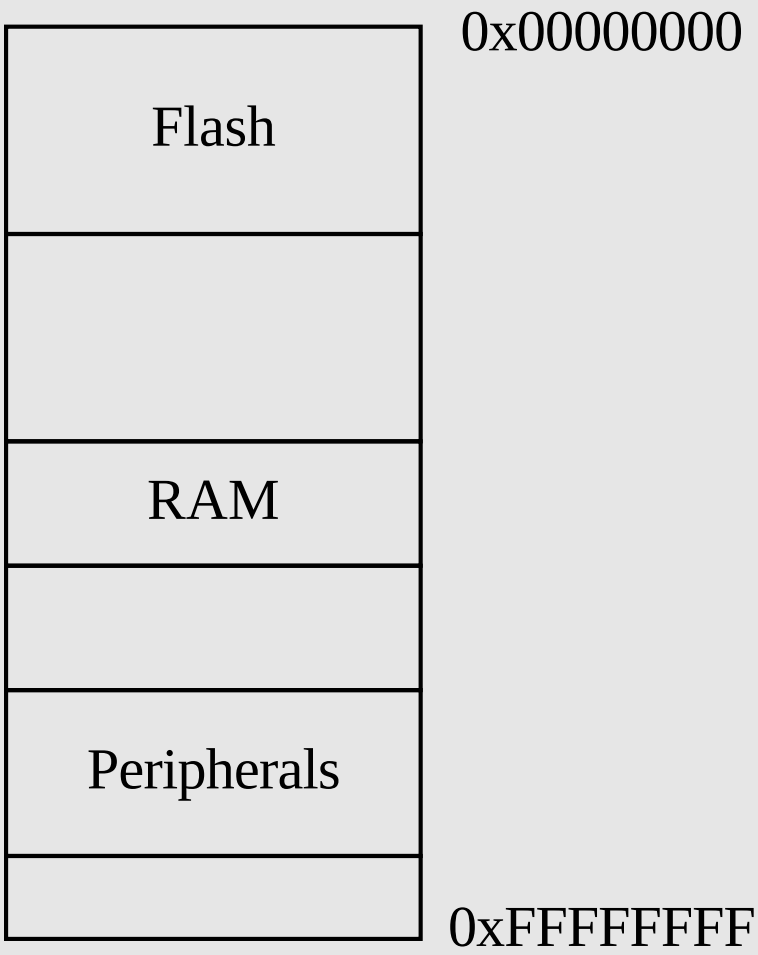
```
protected body My_Protected_Object is

   entry Get_Next_Character (C : out Character)
     when We_Have_A_Character
   is
   begin
       C := Received_Character;
       We_Have_A_Char := False;
   end Get_Next_Character;

   procedure UART_Interrupt_Handler is
   begin
       Received_Character  := A_Character_From_UART_Device;
       We_Have_A_Character := True;
   end UART_Interrupt_Handler;
end;
```

Hardware mapping

Hardware mapping

```
--  High level view of the type
type Servo_Angle is range -90 .. 90

--  Hardware representation of the type
  with Size      => 8,
       Alignment => 16;
```

Memory mapped registers



Hardware mapping

7	6	5	4	3	2	1	0
Reserved		Sense		Reserved			

Sense: Pin sensing mechanism

- 0: Disabled
- 2: Sense for high level
- 3: Sense for low level

Hardware mapping

```
#define SENSE_MASK     (0x30)
#define SENSE_POS      (4)

#define SENSE_DISABLED (0)
#define SENSE_HIGH     (2)
#define SENSE_LOW      (3)

uint8_t *register = 0x80000100;

// Clear Sense field
*register &= ~SENSE_MASK;
// Set sense value
*register |= SENSE_DISABLED << SENSE_POS;
```

Hardware mapping

```
--  High level view of the Sense field
type Pin_Sense is
  (Disabled,
   High,
   Low)
  with Size => 2;

--  Hardware representation of the Sense field
for Pin_Sense use
  (Disabled => 0,
   High     => 2,
   Low      => 3);
```

Hardware mapping

```
--  High level view of the register
type IO_Register is record
   Reserved_A : UInt4;
   SENSE      : Pin_Sense;
   Reserved_B : UInt2;
end record;

--  Hardware representation of the register
for IO_Register use record
   Reserved_A at 0 range 0 .. 3;
   SENSE      at 0 range 4 .. 5;
   Reserved_B at 0 range 6 .. 7;
end record;
```

Hardware mapping

```
Register : IO_Register
  with Address => 16#8000_0100#;
```

```
Register.SENSE := Disabled;
```

SVD -> Ada

```
<field>
  <name>SENSE</name>
  <description>Pin sensing mechanism.</description>
  <lsb>16</lsb> <msb>17</msb>
  <enumeratedValues>
    <enumeratedValue>
      <name>Disabled</name>
      <description>Disabled.</description>
      <value>0x00</value>
    </enumeratedValue>
 [...]
```

github.com/AdaCore/svd2ada

Interfacing with C / Assembly

Interfacing with C / Assembly

```
with Interfaces.C; use Interfaces.C;

[...]

function My_C_Function (A : int) return int;
pragma Import (C, My_C_Function, "my_c_function");

function My_Ada_Function (A : int) return int;
pragma Export (C, My_Ada_Function, "my_ada_function");
```

Interfacing with C / Assembly

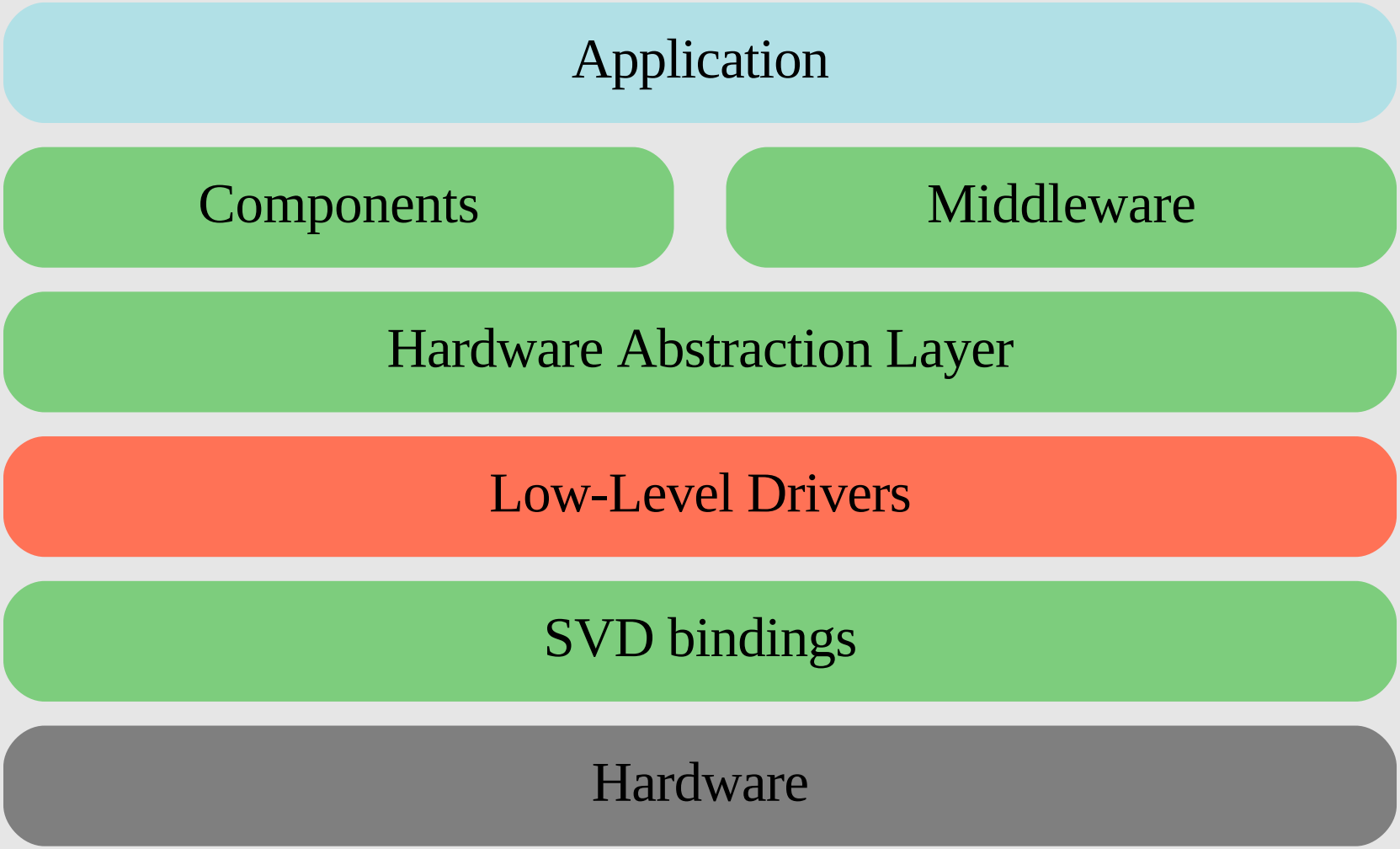
```c
int my_ada_function(int a);

void adainit(void);

void adafinal(void);


void main(void) {
  adainit();

  my_ada_function(42);

  adafinal();
}
```

The Ada Drivers Library

Ada Drivers Library

- Firmware library
- Hardware and vendor independent
- 100% Ada
- Hosted on GitHub:
  `github.com/AdaCore/Ada_Drivers_Library`

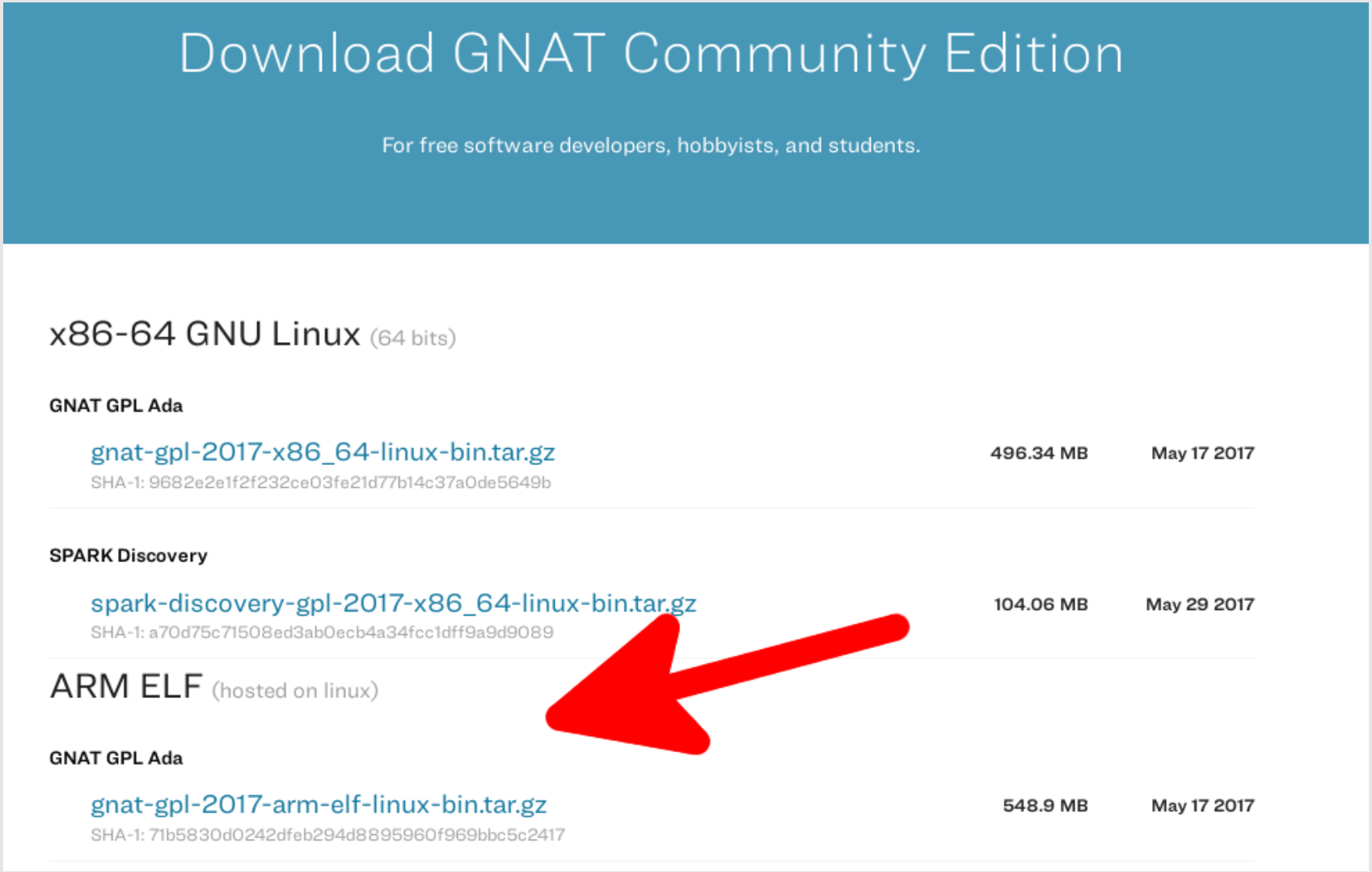
Architecture



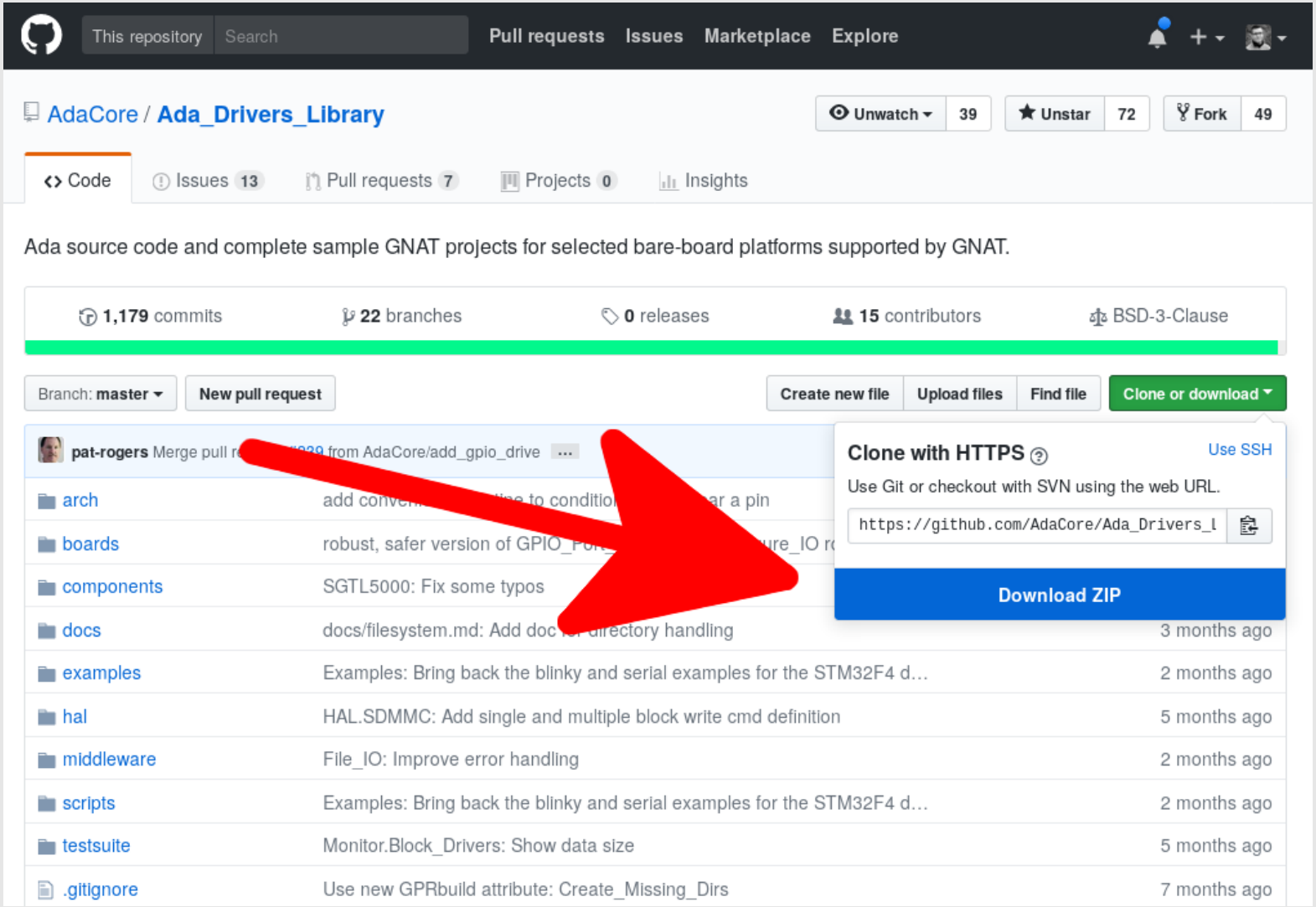
Supported platforms

Getting started demo

Download and install the tools: adacore.com/community



Download Ada Drivers Library

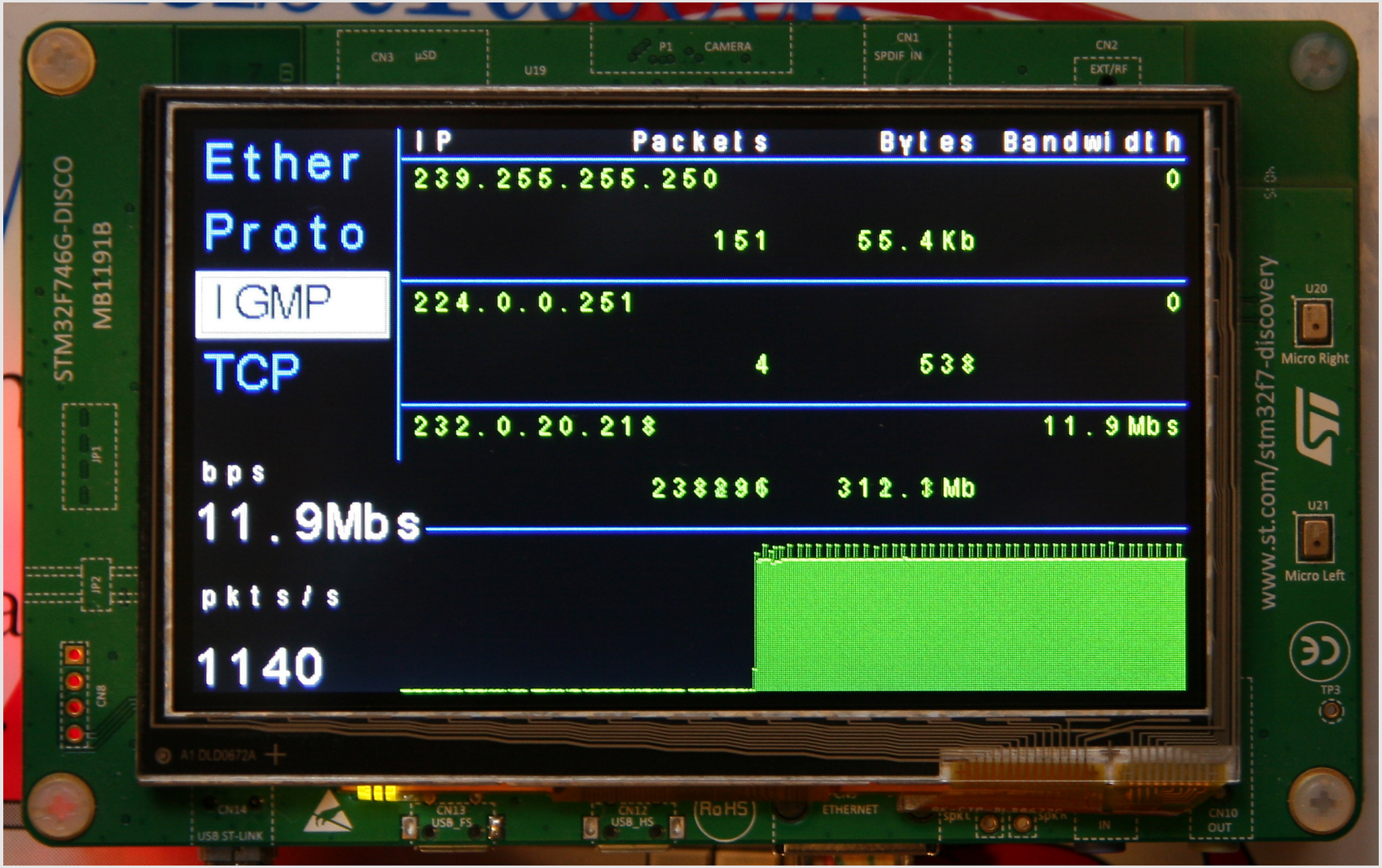


The Make with Ada Competition

- Embedded software project competition
- Open to everyone
- ~8000 euros in prize
- `makewithada.org` (Twitter @adaprogrammers)

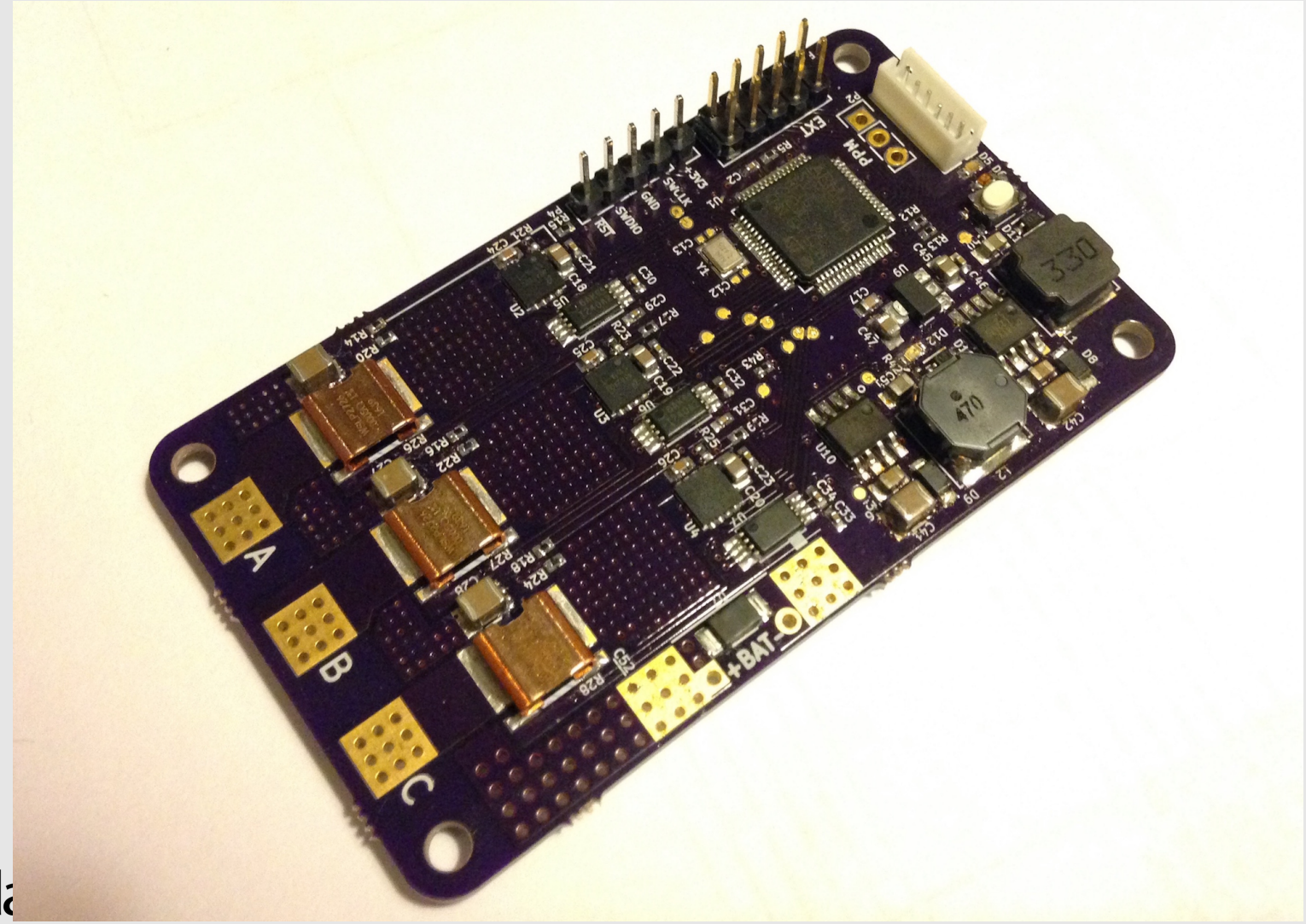
2016 Winner project (Stephane Carrez)

github.com/stcarrez/etherscope



2017 Winner project (Jonas Attertun)

blog.adacore.com/make-with-ada-2017-brushless-dc-motor-controller



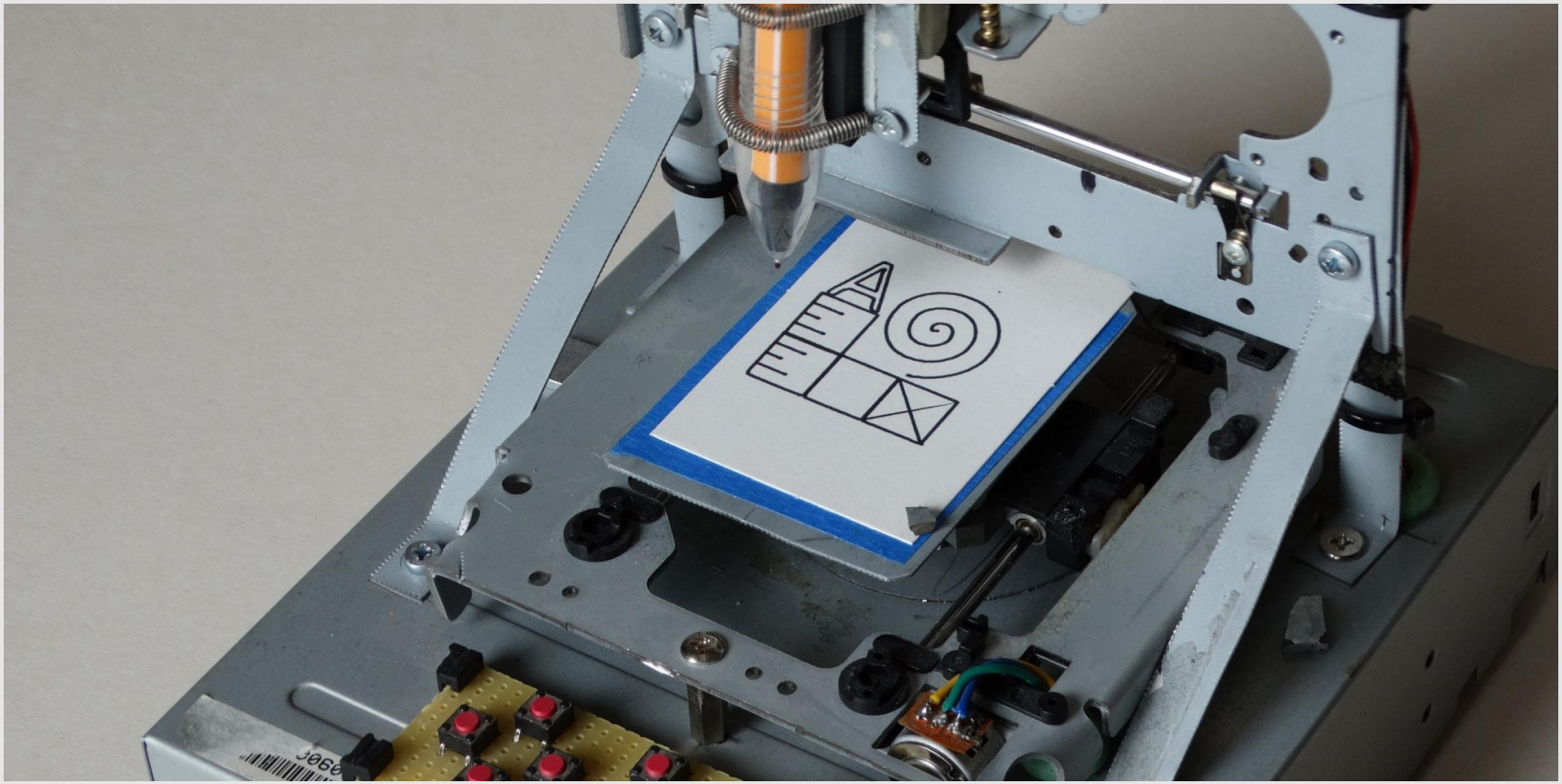
Some projects base on the Ada Drivers Library

Crazyflie 2.0 Flight controller

blog.adacore.com/how-to-prevent-drone-crashes-using-spark

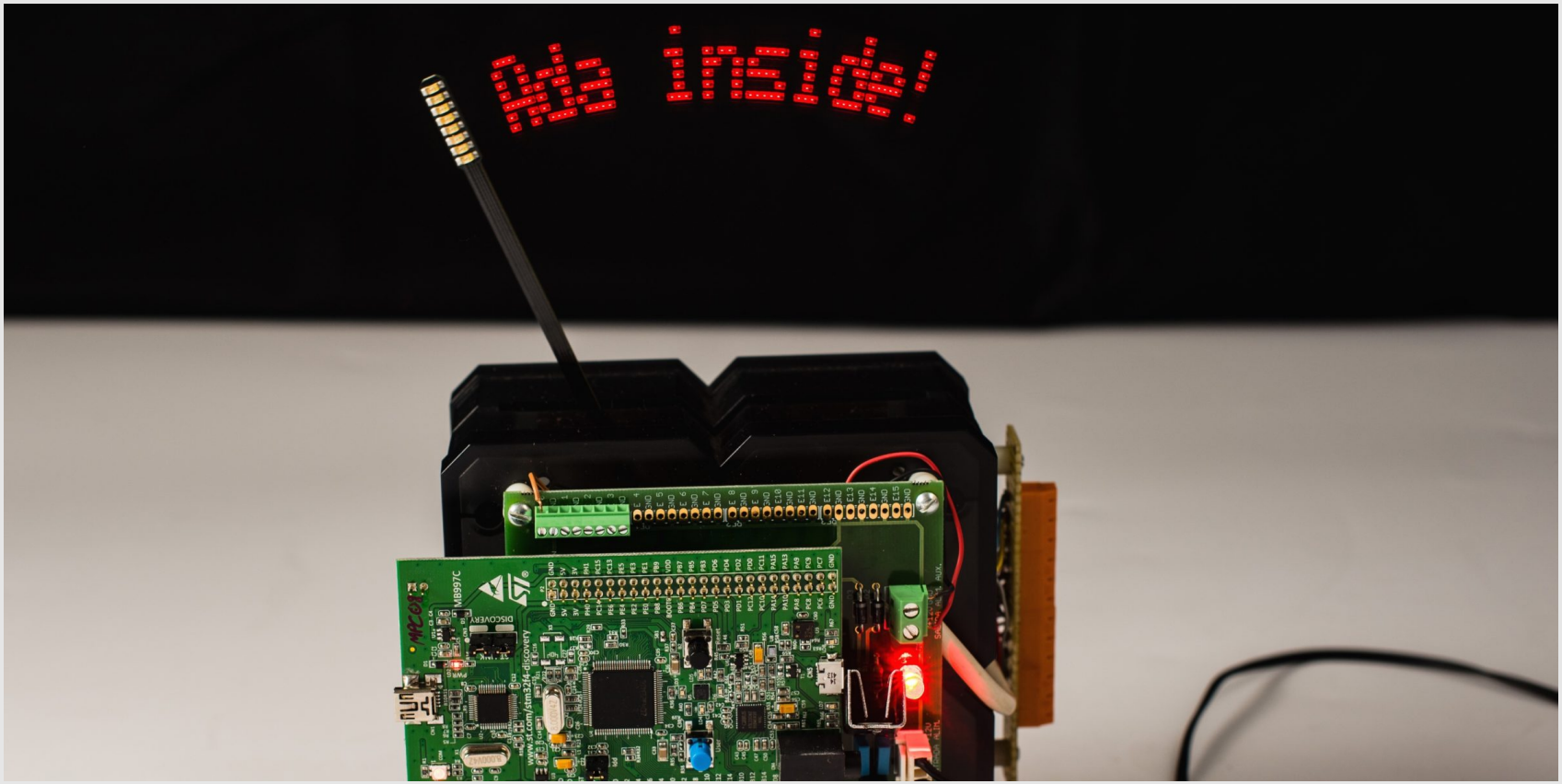
CNC Controller

blog.adacore.com/make-with-ada-arm-cortex-m-cnc-controller



Pendulum clock LED

blog.adacore.com/writing-on-air



DIY instant camera

blog.adacore.com/make-with-ada-diy-instant-camera

Wolf

github.com/lambourg/Ada_Bare_Metal_Demos

Wee Noise Maker

github.com/Fabien-Chouteau/Wee-Noise-Maker