Hobby Electronics With Erlang on the Raspberry Pi

Angela Johansson
EUC 9-10 June 2014
*Video - MB Led*
* MB Led
* Cool, so this is written in Erlang then?
  * Well, no.
* So what’s your point?
  * It might as well have been written in Erlang!
    * **Communicating** entities, message passing
    * Choosing leaders, autonomous parts, scaling…
    * State machines, recursion

Let me know if you implement an Erlang version before me!

The Sky is the Limit!
*MB Led
http://mbled.wordpress.com/

inspired by
*GLiP - (a) Great LED Interactive Puzzle
http://www.glip.fr/

Check it out
* So how and where did you start?
  * I built Adafruit (and other) projects
  * DIY-kits ”IKEA style”

* You get source code written in Python - very similar to Erlang!

Start small, go from there
LED matrix fun
It’s not always as difficult as it seems!
*There is so much to choose from!
* Change something, combine, experiment

*Let your imagination roam free -
What do you want to build and what does it do?

Have fun!
*So why use Erlang?

1. Simply because it’s possible.
2. Your code will design and write itself.
3. Erlang is ideal for talking to HW: communication, state machines, fault tolerance, value crunching…
4. You want to extend your code later on. Add an Erlang touch!

Why not use Erlang? … when it’s a Raspberry Pi
*Where does Erlang fit into this?*

*First, understand the existing code*

```
Erlang application
Erlang driver code
Erlang libs
```

```
C API
Memory mapping
Linux kernel
Files
```

```
C
```

```
Adafruit repo
"Prerequisites"
```

```
OS
Command line tools
```

```
HW
```

It’s a piece of cake!
*So, what did you do then?*

*googled “I2C Linux”*

*took some C-code, wrote a NIF*

Use SMBus API exposed by libi2c-dev

open address 0x.. read register 0x.. write 0x.. to register 0x..

Data sheet

Test HW

Yes, it’s really that easy.
Once written, the libs can of course be reused

1. My code:
   git://github.com/drimtajm/erlang-rpi-hw-drivers
   * Upcoming feature: SPI support

2. From the author of Mockgyver: WPI
   git://github.com/klajo/wpi
   * Uses the Wiring Pi library (C code)

3. ALE - Erlang Actor Library for Embedded
   git://github.com/esl/erlang_ale
   * From Erlang Solutions

* Or write your own…
* Why not work test driven?
  * Unit testing with Mockgyver and Proper
  * System testing with common test
  * Simulators and messages

Testing is good for you
setup() ->
    [...]
    %% Mock I2C interface methods
    ?WHEN(i2c_interface:open_i2c_bus(_Address) -> {ok, ?HANDLE}),
    ?WHEN(i2c_interface:close_i2c_bus(_Address) -> ok),
    [...]
    {ok, Pid} = ads1015_driver:start_link(),
    Pid.

    [...]

    init_should_open_i2c_bus_test(_) ->
        ?WAS_CALLED(i2c_interface:open_i2c_bus(?I2C_ADDRESS)).

    terminate_should_close_i2c_bus_test(Pid) ->
        ads1015_driver:stop(),
        wait_for_exit(Pid),
        ?WAS_CALLED(i2c_interface:close_i2c_bus(?HANDLE)).
prop_set_status_bit_always_sets_status_bit() ->
  ?FORALL(BitPattern, word_value(),
    begin
      NewBitPattern =
        ads1015_driver_lib:set_status_bit(BitPattern),
      is_integer(NewBitPattern)
        and ((NewBitPattern band ?STATUS_BIT) > 0)
    end).

prop_decodes_encoded_data_rate() ->
  ?FORALL(DataRate, data_rate_value(),
    DataRate ==
      ads1015_driver_lib:decode_data_rate(
        ads1015_driver_lib:encode_data_rate(DataRate))).

data_rate_value() ->
  oneof([128, 250, 490, 920, 1600, 2400, 3300]).
Back to the Future

Build something new
* How hard can it be?!
  * Actually, it turned out to be as easy as I imagined
  * But: Routing was time-consuming in Eagle and I left the surface mounting part to an expert

Data sheet

Let the hardware do most of the job for you
"KISS"

"Simple" is relative...
* Putting it together
* Display test mode
* I only needed my I2C primitives
Tell us about the software!

”Thrown together” to make it work

At least some thoughts behind the design

change destination toggle_leds

check_time_and_toggle_leds

check_time

It works!
Demo
* So, what about communication?
* ”Connected by Cybercom”
* Make the system distributed, just ”for fun”

```
Simply run this on a remote node using rpc:call/4
```

In a connected world, Erlang rules!
* Is there bluetooth support for Erlang?
* Strangely, I found nothing when I googled
* I would like to send binaries ”the Erlang way”
* **Bluez** provides a bluetooth stack in Linux
* RFCOMM (”serial port emulation”) can be used to transfer data, you only need to create sockets
* So I wrote a NIF against Bluez
  * Cards must be put in ”scan mode”
  * Packets are ”concatenated” when they arrive

**If it doesn’t exist, write it yourself**
go() ->
    [...]
    {ok, Socket} = bluetooth_interface:create_rfcomm_socket(),
    ok = bluetooth_interface:bind_bt_socket(Socket, ?PORT, LocalMac),
    ok = bluetooth_interface:bt_socket_listen(Socket),
    Pid = spawn_link(?MODULE, socket_acceptor, [self(), Socket]),
    receive
        {Pid, done} -> ok
    after 60000 ->
        error(timeout)
    end,
    bluetooth_interface:close_bt_socket(Socket).

socket_acceptor(Caller, Socket) ->
    {ok, Socket2, RemoteAddress} =
        bluetooth_interface:bt_socket_accept(Socket),
    receive_loop(Socket2),
    Caller ! {self(), done},
    ok.
go() ->
    {ok, Socket} = bluetooth_interface:create_rfcomm_socket(),
Pid = spawn(?MODULE, socket_connector, [self(), Socket, RemoteMac]),
    receive
        {Pid, done} -> ok
    end,
    bluetooth_interface:close_bt_socket(Socket).

socket_connector(Caller, Socket, RemoteMac) ->
    ok = bluetooth_interface:bt_socket_connect(Socket, ?PORT, RemoteMac),
    Data = erlang:term_to_binary({self(), greetings}),
    ok = bluetooth_interface:bt_socket_send(Socket, Data),
    [...]
    Data2 = term_to_binary("Bye!")
    ok = bluetooth_interface:bt_socket_send(Socket, Data2),
    timer:sleep(10000),
    Caller ! {self(), done},
    ok.
So do you plan on developing this further?

* Absolutely!
* But I would like some help from you...
* Ideally, one would like to have the same support in Erlang as for TCP sockets/inet - bnet!?
* Make use of bluetooth services - ebpmd?
* Facilitate automatic card setup/configuration
* Rewrite it as an Erlang port
* Support for Windows (Widcomm?)
* Other suggestions?

Let me know if you’re interested!
Angela Johansson
angela.johanssson@gmail.com

Twitter & Github:
@drimtajm