Physical interfaces with Arduino and DBUS
(and ZeroMQ)

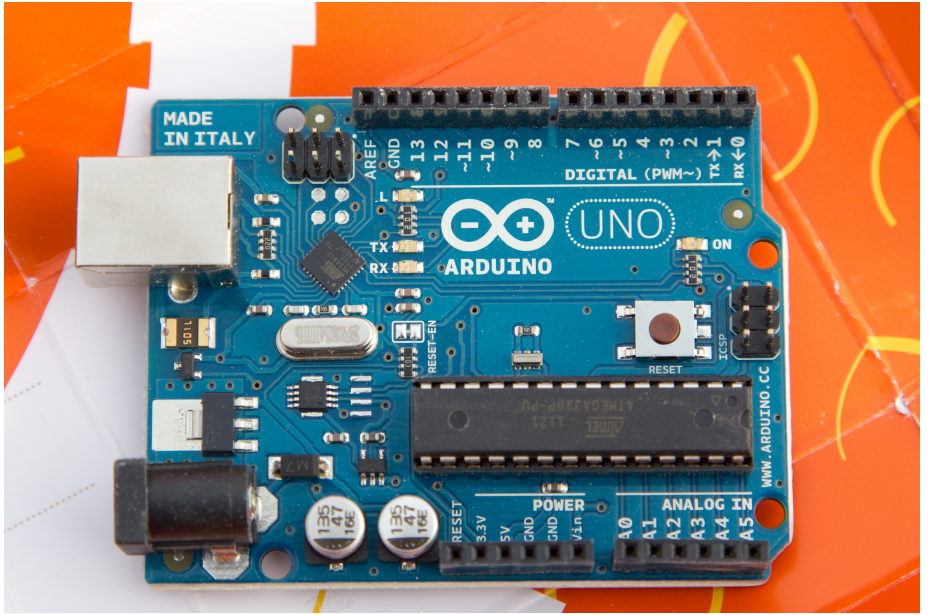
D-Bus

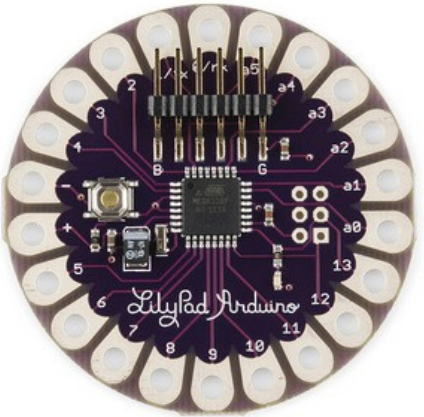
About me

- Electronics hobbyist since childhood
- Active in Finlands hackerspace scene
- Believer: Open Source & Open Hardware
- https://github.com/rambo
- din.com/in/

Arduino

- ”Electronics for artists”
- Open Source (HW & SW)
- Easy to learn
- Large ecosystem
- Multiple form factors
- http://arduino.cc





D-Bus

- Emit signals
- Service RPC methods
- Super easy-to-use Python decorators included
- Introspection & service discovery
- Painfull to use over network
- http://dbus.freedesktop.org

```
106 @dbus.service.signal('fi.hacklab.reactorsimulator.engine')
107 def emit_movement_stop(self, x, y, sender):
108     #print "movement stop emitted %d_%d" % (y,x)
109     pass
110
111 @dbus.service.signal('fi.hacklab.reactorsimulator.engine')
112 def emit_movement_start(self, x, y, sender):
113     #print "movement start emitted %d_%d" % (y,x)
114     pass
115
116 @dbus.service.method('fi.hacklab.reactorsimulator.engine')
117 def start_move(self, direction):
118     if self.scram_active:
119         return
120     if (   bool(direction) #True is up
121         and not (self.depth <= reactor_module.rod_min_depth)):
122         self.current_velocity = self.current_max_speed * -1
123         self.emit_movement_start(self.x, self.y, self.object_path)
124         return
125     # We return above so here we can just check if we can move
126     if (not (self.depth >= reactor_module.rod_max_depth)):
127         self.current_velocity = self.current_max_speed * 1
128         self.emit_movement_start(self.x, self.y, self.object_path)
```

Z(ero)MQ

- Fast, **very** fast
- Low-level
- No service discovery or other niceties like that
- Multiple transports
  - Inproc
  - Unix sockets
  - Network
- Lots of language bindings
- http://zeromq.org

arDuBuS

- Almost direct mapping to Arduino concepts
- Configured with YML
- Automatic Arduino code generation
  - The ”hard” stuff is done in the libraries anyway

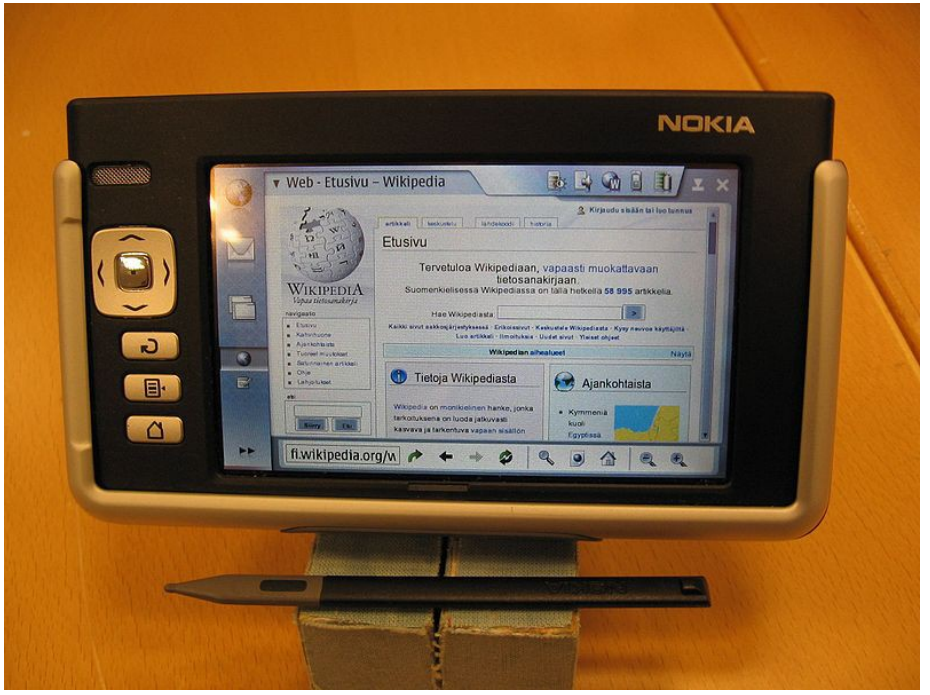
https://git)o/ardubus

rod_control_panel §

```
#include <Bounce.h>
#define ARDUBUS_DIGITAL_INPUTS { 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 15, 16, 17, 18,
#include <I2C.h>
#include <i2c_device.h>
#define ARDUBUS_PCA9535_BOARDS { 0 }
#define PCA9535_ENABLE_BOUNCE
#define PCA9535_BOUNCE_OPTIMIZEDREADS
#define ARDUBUS_PCA9535_INPUTS { 0, 1, 2, 3, 4, 5, 6, 7 }
#define ARDUBUS_PCA9535_OUTPUTS { 8, 9, 10, 11, 12, 13, 14, 15 }
#include <pca9535.h>
#include <pca9635.h>
#include <pca9635RGB.h>
#include <pca9635RGBJBOL.h>
#define ARDUBUS_PCA9635RGBJBOL_BOARDS { 0, 1 }
#define ARDUBUS_AIRCORE_BOARDS { 4, 5, 6, 7, 8 }
#define ARDUBUS_I2CASCII_BOARDS { 9 }
#define ARDUBUS_I2CASCII_BUFFER_SIZE 6

#include <ardubus.h>
void setup()
{
    Serial.begin(115200);
    Serial.println(F(""));
    Serial.println(F("Board: rod_control_panel initializing"));

    I2c.timeOut(500); // 500ms timeout to avoid lockups
    I2c.pullup(false); //Disable internal pull-ups
    I2c.setSpeed(false); // Fast-mode support
    ardubus_setup();
    PCA9635.set_driver_mode(0x0);
    PCA9635.set_sleep(0x0);
    Serial.println(F("Board: rod_control_panel ready"));
}

void loop()
{
    ardubus_update();
}
```

Background

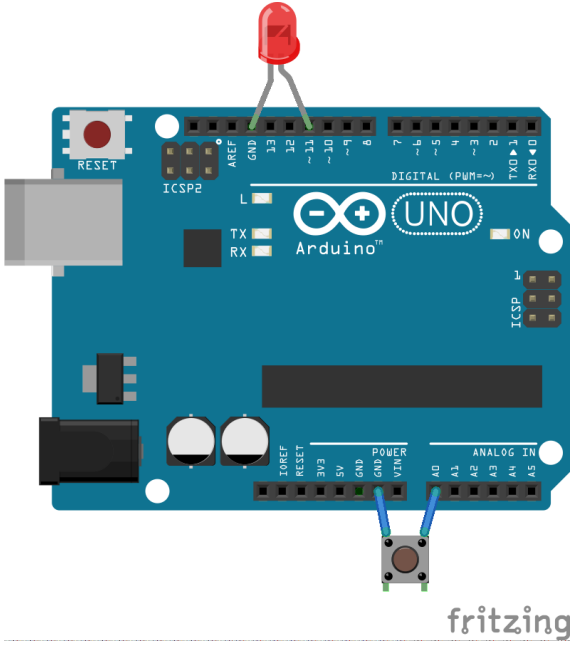
- Maemo introduced me to D-Bus back in N770 days
- One of those cool ideas I never had the time for
- ”The reactor project” as catalyst
  - And thus dictated the features to be implemented first



”Practical” example

- It's demo time!

```
gtk_example_board:
    digital_in_pins: # pins passed to ARDUBUS_DIGITAL_INPUTS
        - pin: A0
          alias: example_button
    digital_pwmout_pins:
        - pin: 11
          alias: dimmable_led

```

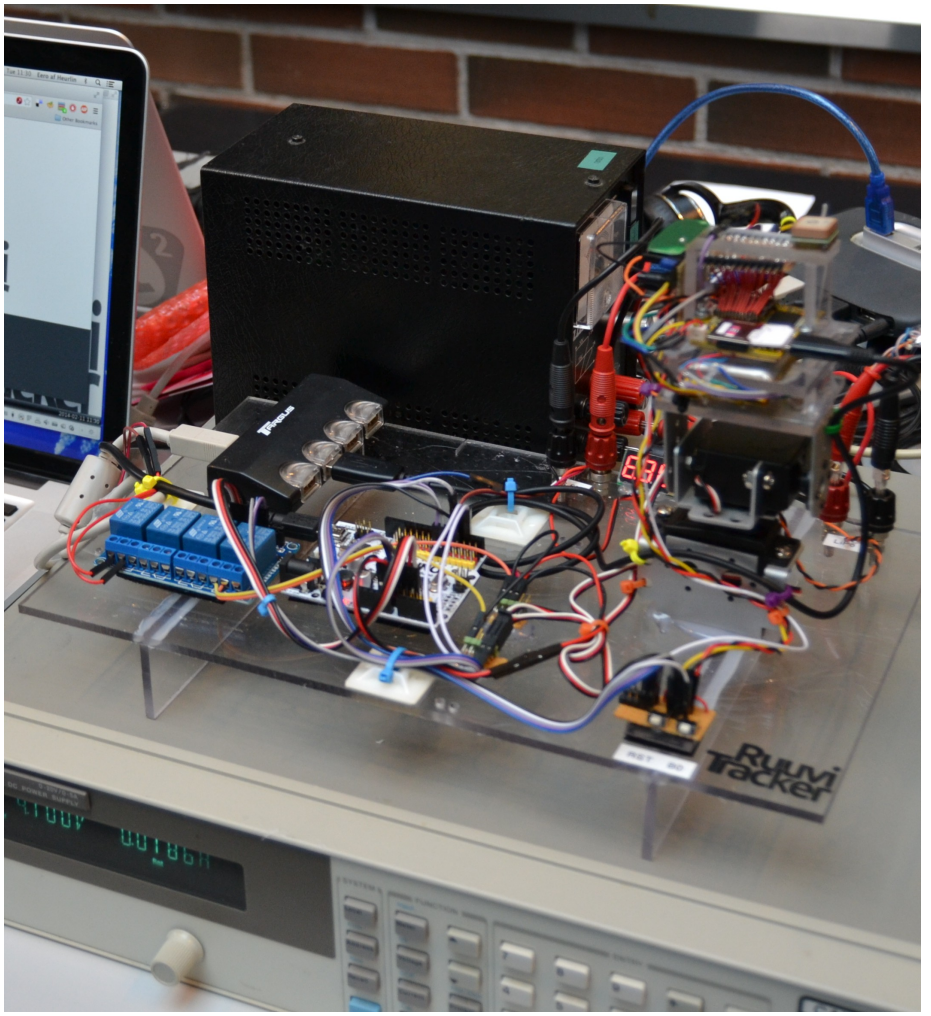


http://youtu.be/t78oJ-nSe-k

Real world example

- Test rig for automatic integration testing
- Sync signals input
- Servos, relays, reset and boot0 contol lines as outputs

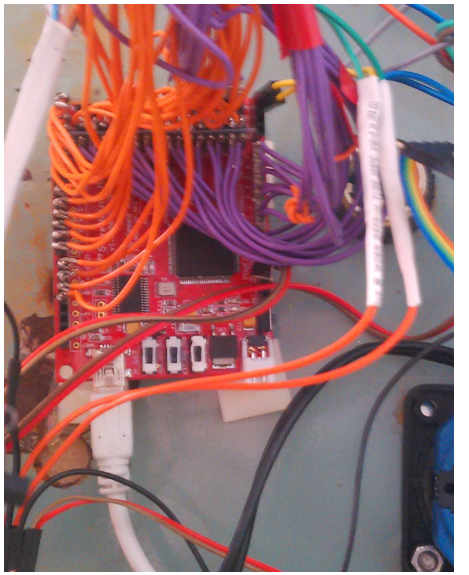
https://github.com/rambo/ruuvitracker_testplatform

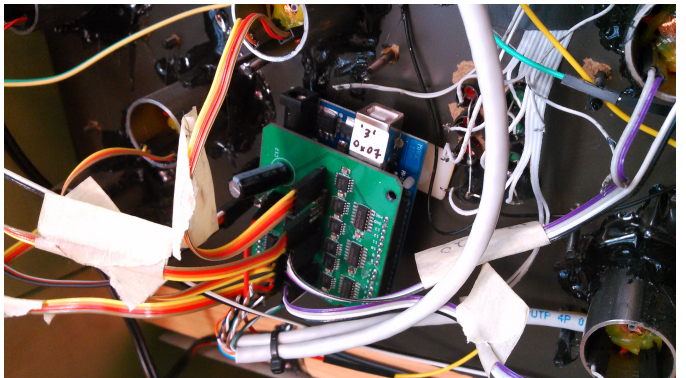


The Reactor project

- RBMK ”simulator”
- First for ALT2011
- Because: Art!
- Lots of moving parts all either controlled by or controlling the simulation.

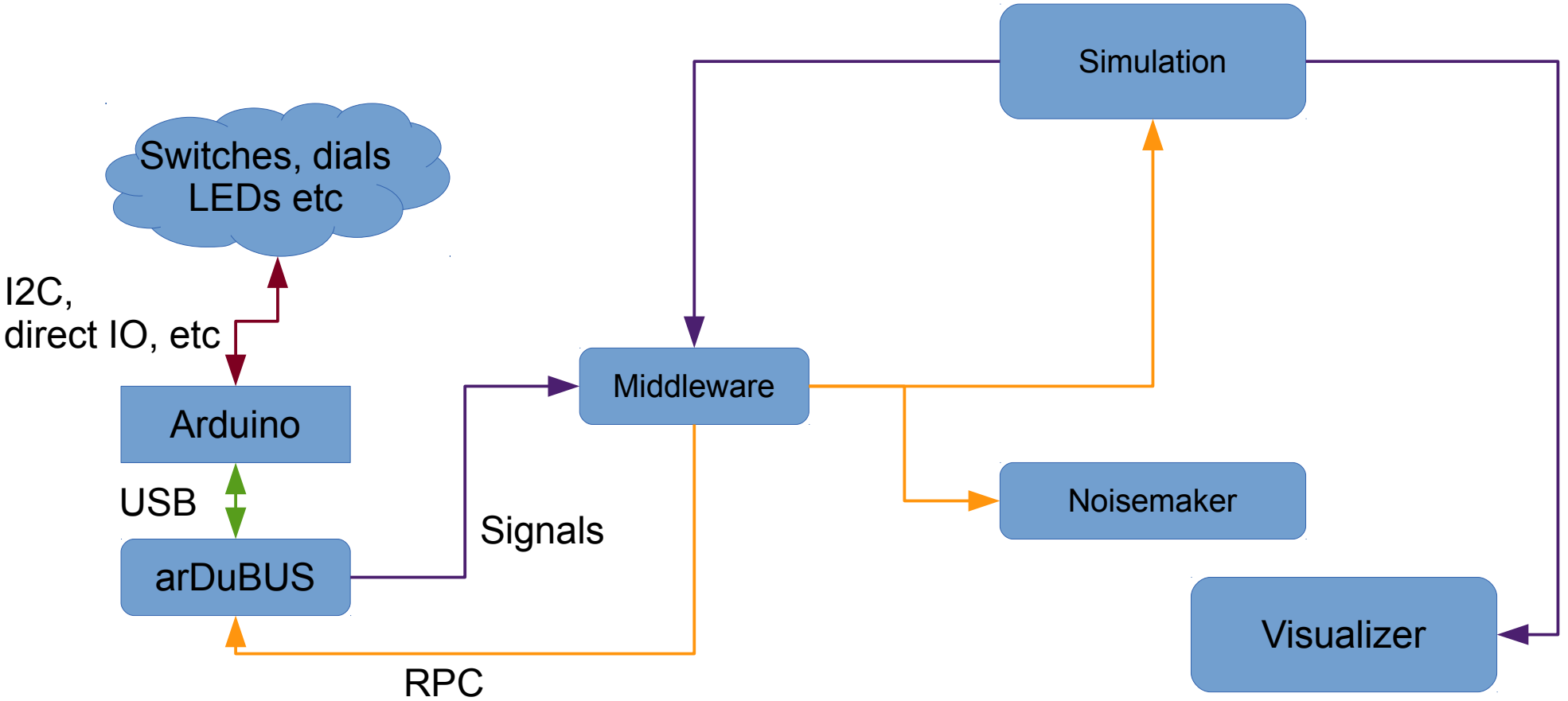
https://github.c klab/reactor





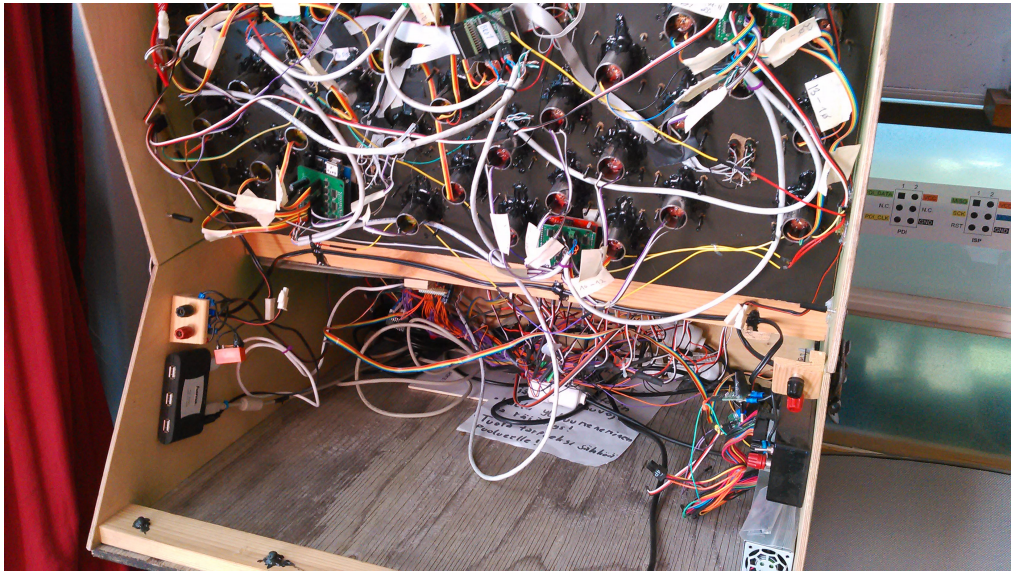

4539
MW
Helsinki
Hacklab

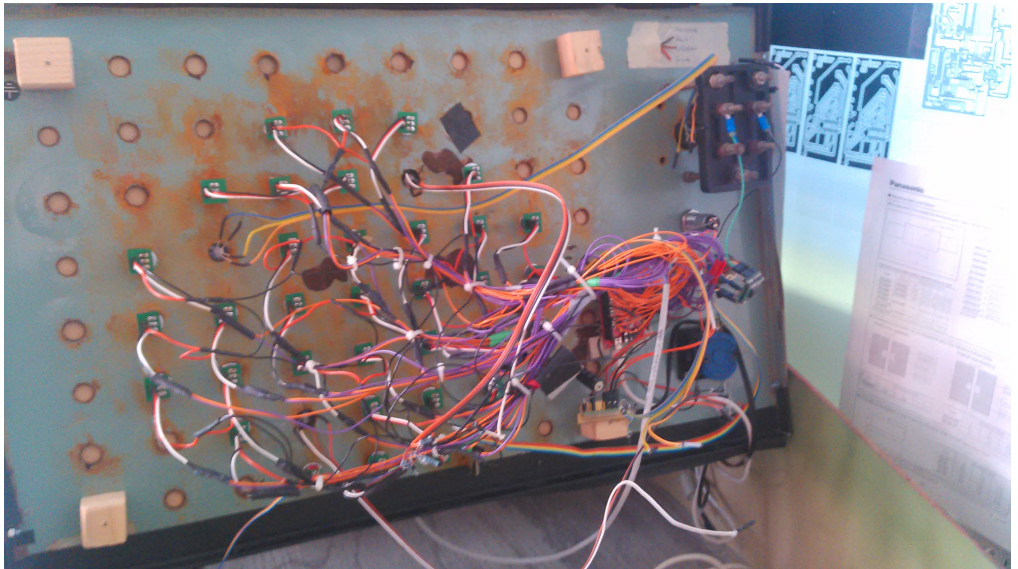
Reactor architecture



Architecture (contd)

- Message passing
- Separation of duties
- Flexibility for using multiple languages
- D-Bus chosen initially due to familiarity (and those sweet decorators)





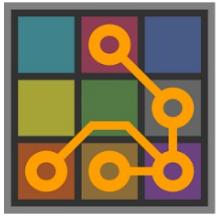
You mentioned ZMQ?

- We've had some issues with D-Bus
- ZMQ ought to help with those
- But first it must be made as easy to use as D-Bus

https://github.com/rambo/python-zmqdecorators

That's it for now

- Help with Arduinos: Helsinki Hacklab http://helsinki.hacklab.fi/
  - Open Tuesdays 5pm onwards
- More about arDuBUS: Talk to me
  - I'm usually at Helsinki Hacklab on tuesdays
- More about RuuviTracker: #ruuvitracker @IRCNet
- 2 minute Q & A



Helsinki Hacklab