10th Central and Eastern European Software Engineering Conference in Russia - CEE-SECR 2013

October 23 - 25, Moscow

2014 CEE-SEC(R) Software Engineering Conference in Russia

Reactive Robotics Programming with F# and Mono

Alexander Kirsanov SPbSU, JetBrains lab

Reactive robotics

- Robots are indeed a reactive systems
- Resilient
  - Special requirements for correctness
- Scalable
  - A lot of peripheral devices
  - Integration with servers and clouds
- Responsive

Evolution of controllers

- Ability to use higher-level development tools
  - New platforms
  - Reusing of popular technologies
- Rise of personal robotics
  - Next step after smartphones and the internet
  - A lot of enthusiasts

Development for embedded systems

- Different architectures
  - Host and target machines
  - Cross-compilers
- Controller specific Libraries
  - Slightly extensible
- Poor toolset

Reusing of popular technologies

- Development unification
- Using all tools and profits from the industry
- Involving of SE professional in field of robotics

F# language

- Type inference
- Functional first
- First class event
- Asynchronous computations
- Fully compatible with CLR
- Well supported by many .NET tools

.NET/Mono

- Cross-platform
- Rapidly evolving
- Xamarin
  - New performance team
  - Optimization for ARM and mono itself
  - Using the same versions of mono

Enough even for robotics

- A lot of tools and libraries
- SDK for clouds and NUI
- Applications in field of robotics

KINECT
for XBOX 360.

nuget

Kinect example

```
let speedDisposable =
    kinect.SkeletonFrameReady
    |> Observable.map ExtractTrackedSkeletons
    |> Observable.choose getPoints
    |> takeDerivative                    //
    |> Observable.buffer 10 1            //   Making flapping routine
    |> Observable.map tupleAverage       //
    |> Observable.map flappingScale      //
    |> Observable.DistinctUntilChanged
    |> Observable.subscribe(sendSpeed robot)
```

Reactive Extensions

- Implemented for 8+ languages
- Ease the using of threads
- Well documented

Development notes

- All devices can be divided
  - Sensors
  - Actuators

Library overview

- All sensors can emit events
- All other devices can handle them
- User can manipulate robot in different levels
  - From i2c module
  - To Model

Library overview

- Different kind of sensors
  - Gyroscope, Accelerometer
  - Sensors of lines and objects
  - IR and other analog sensors
- Motors, led bulbs, lightning stripes

Example

```
let powerSetterDisposable =
        model.ObjectSensor.ToObservable()
        |> Observable.choose (fun o -> o.TryGetLocation)
        |> Observable.filter (fun l -> l.Mass > 5)
        |> Observable.scan (fun acc l -> updatePosition l.X acc) 0
        |> Observable.subscribe (fun x -> model.Servo.["E1"].SetPower -x)
```

Results

- Linux TRIK
- Mono
- All available devices are supported
- Not just F#-faced library. C# friendly API

Limitations

- JIT-compilation has impact on startup time
  - AOT
  - Library is written without third-party assemblies
  - GC and using structs

What do we get?

- The ability to program robots freely
  - No fear of low-level gpio manipulation
  - No Linux-only C and ARM assembler development
- Standard programmer toolset
  - No cross-compilers
- High-level declarative robots programming

Educational Robotics

- Well awarded in STEM (Science, Technology, Engineering, and Mathematics)
- The same Influence as becoming mandatory computer classes in schools 30 years ago

Modern Software Engineering

- Difficult
- Need to know a lot beyond the language itself
- Many difficult subjects
- Foreign abstractions
- All knowledge needs good practise

Learn SE via robotics

- A lot of interesting and challenging tasks
- Interactive environment
  - Very interactive
    - Physically
  - Debugging with your own hands and eyes
- Using the same technologies as you need for job

Results in educational scope

- AYcamp
- All examples are written by first and second year students
- Happiness of high-level declarative robotics

Thanks!

- Questions?