



Sense - An alternative visualisation

Tim Mackinnon
<http://morethan.technology>



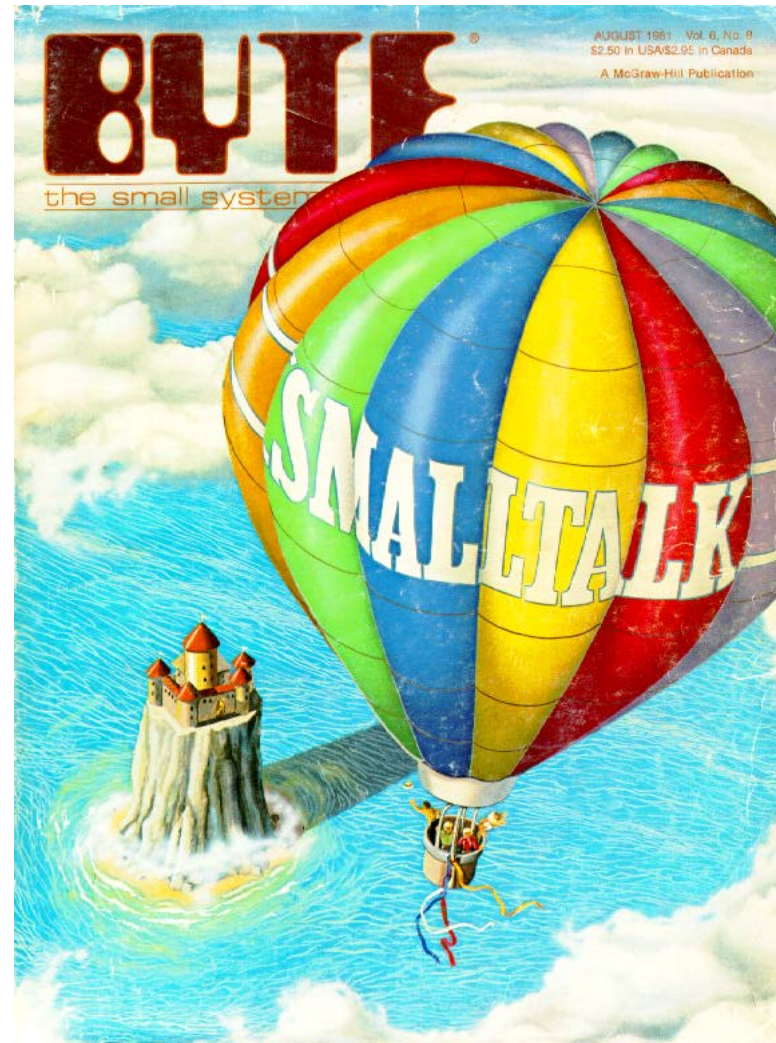
More Than Technology Ltd.

- I am a small company focused on:
 - Solving problems using extreme Agility
 - Delivering business value in partnership with clients
 - Working in an honest, iterative and collaborative manner
 - **Helping teams communicate without killing each other**

- Tim Mackinnon
 - Learned Smalltalk at Carleton University late 80's
 - Originally didn't like it, cute but meh...
 - Now it's the language I keep coming back to for its elegance and simplicity & community

Thank You Smalltalk!

- Vibrant Helpful Community
- So much beautiful code
- Amazing, Inspiring ideas
- FUN!



The OU TU100 Course



The Open University

The Open University **Study at the OU** Research

Study at the OU Undergraduate Postgraduate Research degrees Short courses Study explained

An undergraduate course in Computing and IT.

My digital life

On this page

[What you will study](#) ▾ [Entry](#) ▾ [Regulations](#) ▾ [If you have a disability](#) ▾ [Study materials](#) ▾ [Teaching and assessment](#) ▾ [Future availability](#) ▾ [Students also studied](#) ▾ [How to register](#) ▾ [Student reviews](#) ▾ [Distance learning](#)

While you're learning about tomorrow's technology why not help create it? *My digital life* takes you on a journey from the origins of information technology through to the familiar computers of today, and on to tomorrow's radical technologies. You'll get hands-on experience of the ubiquitous computing approaches that will become increasingly common over the next decade. You'll also learn about the profound social and technological changes associated with information technology – changes that will affect every one of us. This key introductory Level 1 course will help you prepare for these changes – think of it as an online survival kit for the twenty-first century.

Open University's "Sense"

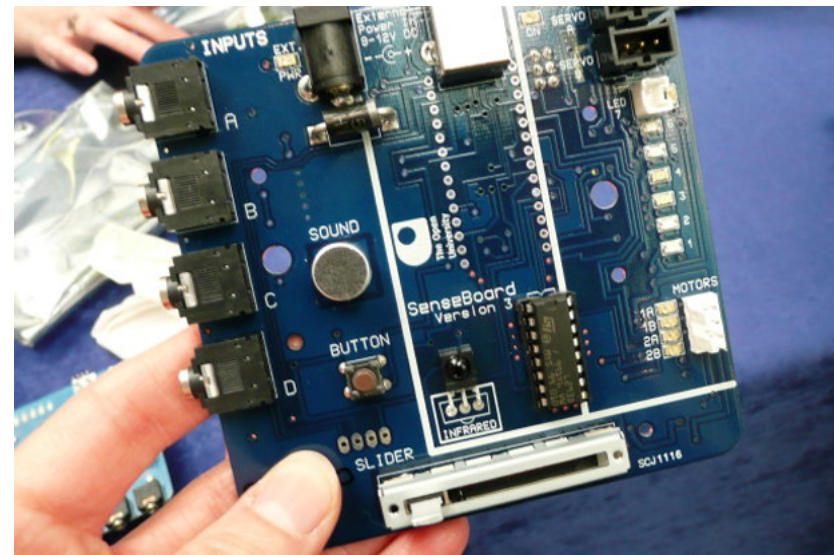
- www.sense.open.ac.uk



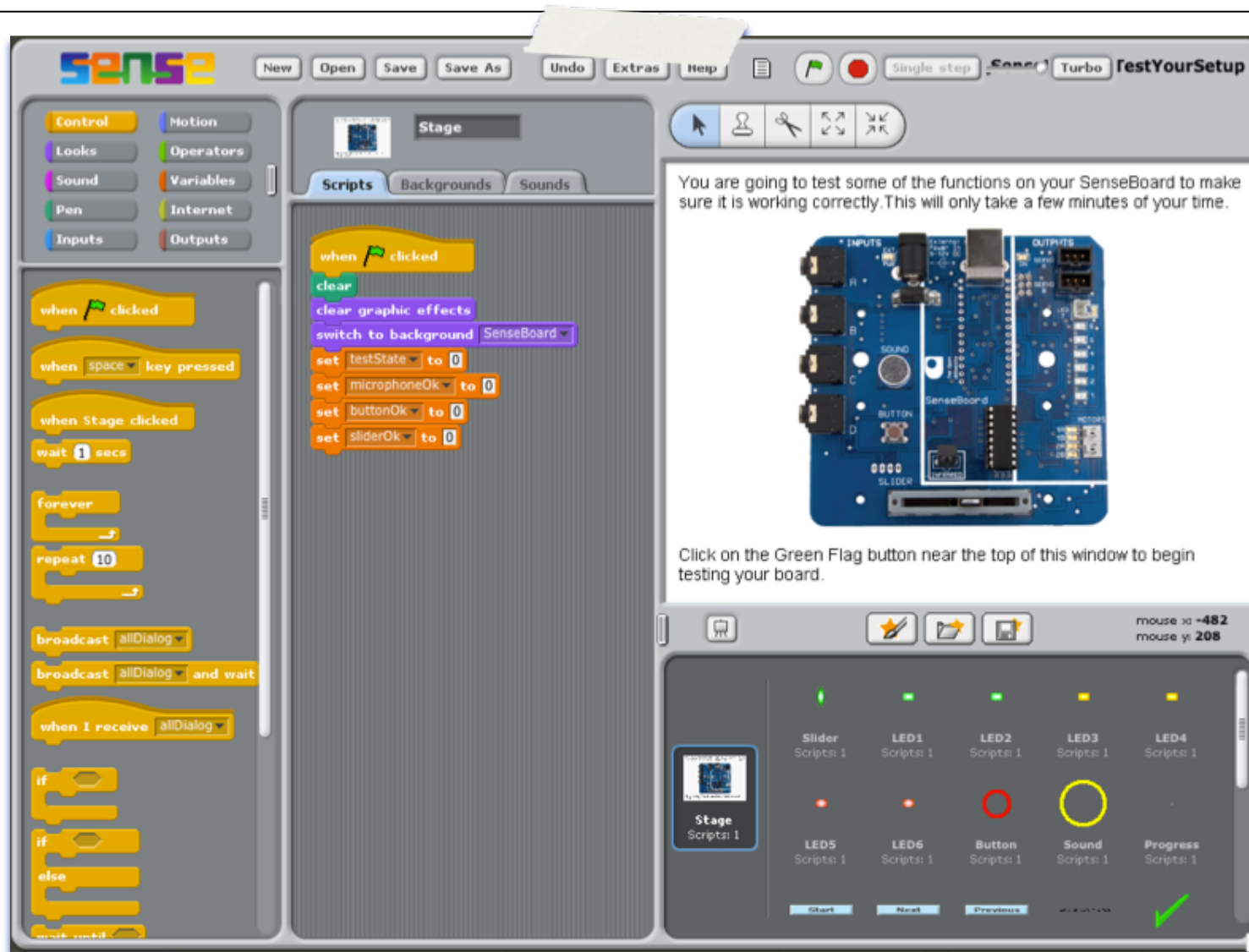
- SenseBoard (usb)

- Freeduino based
- Input sensors – button, slider, mic, ir, accelerometer, thermometer
- Outputs – led's, ir, servo & stepper motors

- Sense IDE – modified Scratch environment



The Sense IDE



sense New Open Save Save As Undo Extras Help Single step Sense Turbo TestYourSetup

Control Motion Looks Operators Sound Variables Pen Internet Inputs Outputs

Stage Scripts Backgrounds Sounds

when green flag clicked
clear
clear graphic effects
switch to background SenseBoard
set testState to 0
set microphoneOk to 0
set buttonOk to 0
set sliderOk to 0

You are going to test some of the functions on your SenseBoard to make sure it is working correctly. This will only take a few minutes of your time.

Click on the Green Flag button near the top of this window to begin testing your board.

mouse x: -482
mouse y: 208

Slider Scripts: 1 LED1 Scripts: 1 LED2 Scripts: 1 LED3 Scripts: 1 LED4 Scripts: 1
LED5 Scripts: 1 LED6 Scripts: 1 Button Scripts: 1 Sound Scripts: 1 Progress Scripts: 1

Start Next Previous

TU100 Course is Well Received

- “I already had some knowledge of programming but was amazed by ***sense***. THIS IS BRILLIANT!

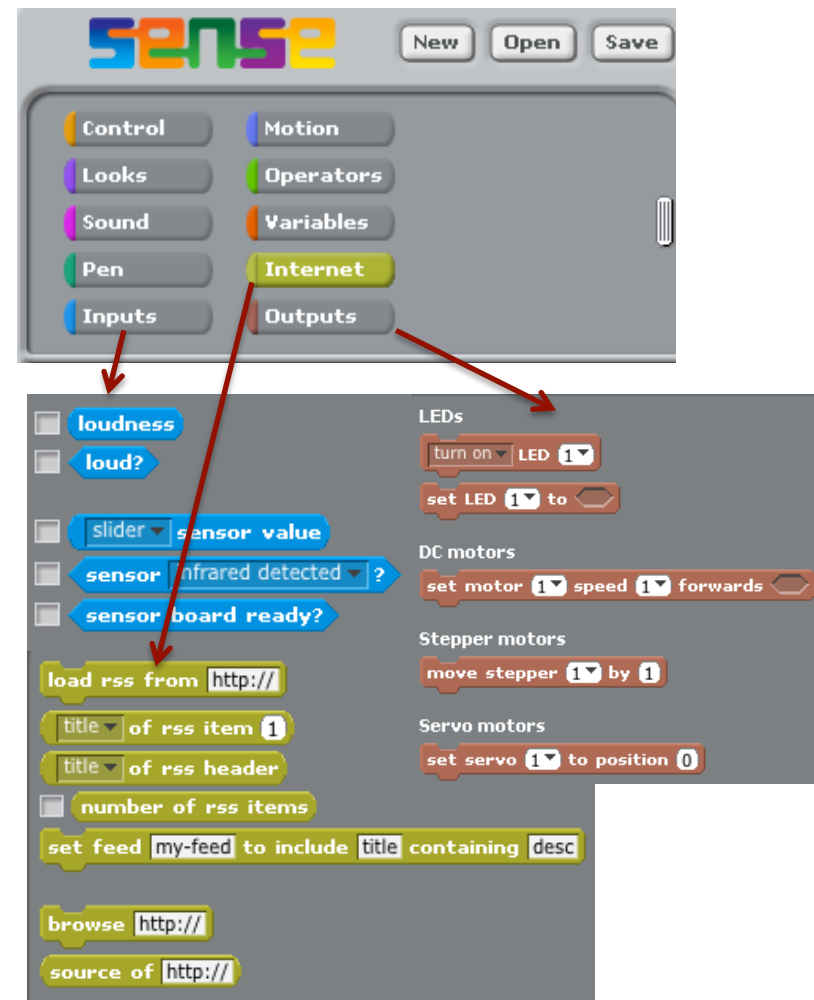
Whoever thought of this way to teach programming needs promoting.”



<http://www.youtube.com/watch?v=g-m3O5vhMss>

V1.0

- Original work completed by:
 - Open Univeristy +
 - John Daniels
 - Dave Cleal
- USB interface to SenseBoard
- Additional blocks
 - Internet
 - Inputs
 - Outputs



V2.0 - Specification

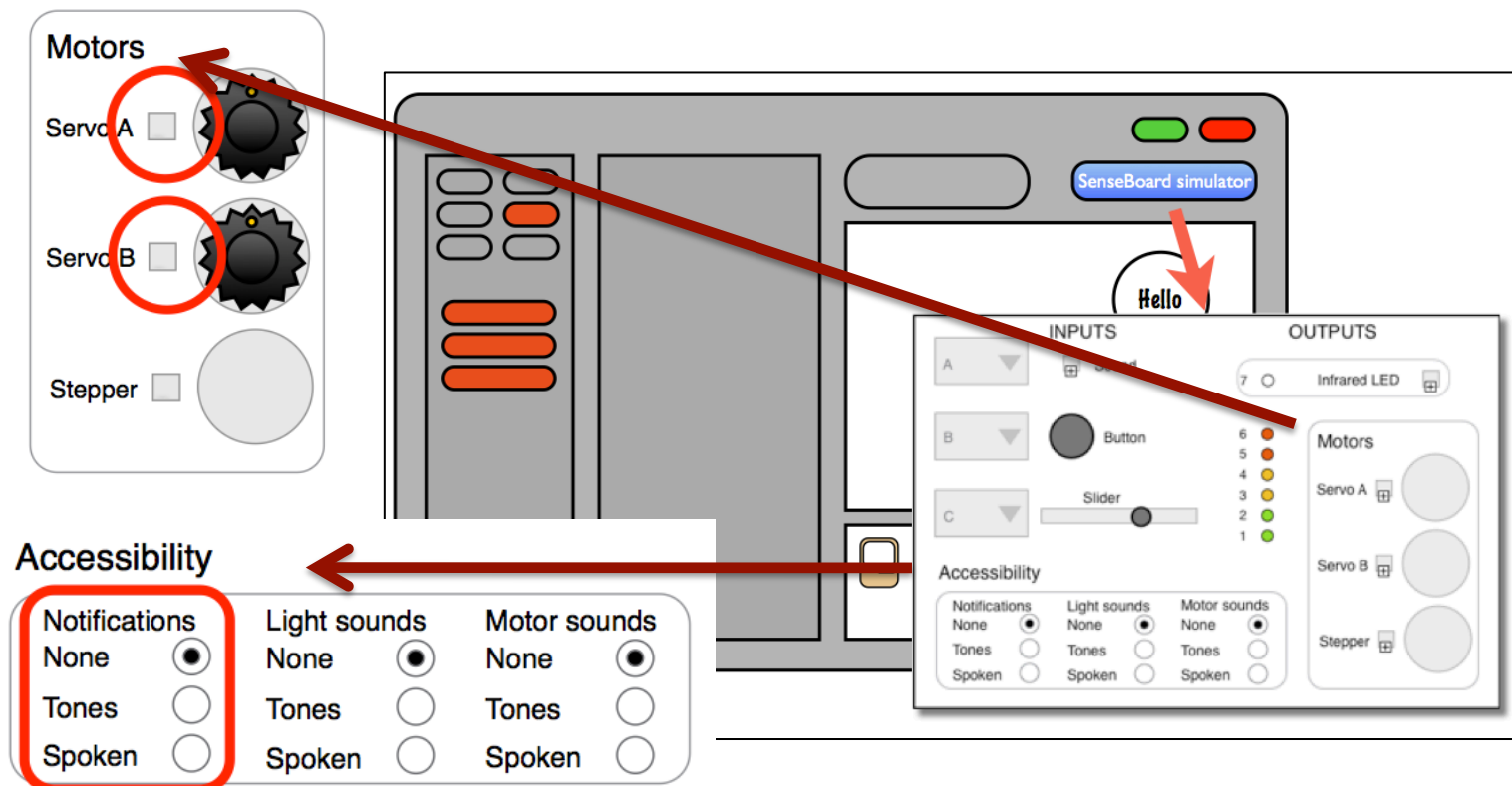
■ Deliverables

- To provide the Simulator and maintenance documentation. The simulator should be delivered either as part of a revised version of the Sense programming environment, *or as a plugin module that can be installed alongside Sense.*

- The simulator is intended to act as a partial replacement for this physical item for those students who are unable to use the board because:
 - They have a physical disability which precludes them from handling the board or sensors
 - They are an inmate, or tutor visiting an inmate, in a prison where access to the SenseBoard and cables are not allowed.

V2.0 Mockups

- Launch a simulator without a SenseBoard



Which technology?

- Everyone seemed afraid to use Squeak ?
- PRO's
 - It's working well
 - already multiplatform
 - can already safely deploy
- CON's
 - Could Squeak do cross platform sound output, mic input, keyboard control?
 - Can anyone still figure this stuff out?
- Worried about the additional Java footprint and installation complications
- Help?
(where's Tim Rowledge)



The Shock of early Squeak...

Workspace

Scratch Source Code 1.3.1

Copyright (c) 2008 Massachusetts Institute of Technology.
All rights reserved.

Please see the file "License.txt" included in this package.

To open a Scratch window, select "open..." from the Squeak menu, then select "Scratch".

In the Scratch window, hold down the shift key while clicking the "Extras" button to get the Scratch developers menu. This menu contains commands to enter or exit user mode and to save this Squeak image file in user mode. In user mode, the Scratch user interface fills the window and Squeak developer commands and tools are disabled.

Dolt's

The following are useful Dolt's for starting with a fresh image:

Halt encountered.

```

UndefinedObject(Object)>>halt:
UndefinedObject(Object)>>halt
UndefinedObject>>Dolt
Compiler>>evaluate:in:to:notifying:ifFail:
TextMorphEditor(ParagraphEditor)>>evaluateSelection
TextMorphEditor(ParagraphEditor)>>dolt
[] in TextMorphEditor(ParagraphEditor)>>dolt:
TextMorphEditor(Controller)>>terminateAndInitializeAround:
TextMorphEditor(ParagraphEditor)>>dolt:
| methodNode method value |
class ← (aContext == nil ifTrue: [receiver] ifFalse: [aContext receiver]) class.
self from: textOrStream class: class context: aContext notifying: aRequestor.
methodNode ← self translate: sourceStream noPattern: true ifFail:
[+failBlock value].
method ← methodNode generate: #(0 0 0 0).
self interactive ifTrue:
[method ← method copyWithTempNames: methodNode tempNames].
context == nil
ifTrue: [class addSelector: #Dolt withMethod: method.
value ← receiver Dolt.
self
all inst vars
sourceStream
requestor
class
context
thisContext
all temp vars
textOrStream
aContext
receiver
aRequestor
failBlock
methodNode
method
value

```

System Browser

Scratch-Objects	CameraMedia	-- all -- initialize accessing scratch op compression copying object i/o
Scratch-Blocks	FilterPack	
Scratch-Execution Engine	ImageMedia	
Scratch-Object ID	MovieMedia	
Scratch-UI-Dialogs	ScratchMedia	
Scratch-UI-Panes	ScratchSpriteMorph	
Scratch-UI-Variations	ScratchStageMorph	
Scratch-UI-Watchers	ScriptableScratchMorph	
Scratch-UI-Support	SoundMedia	
Scratch-Paint		
Scratch-Sound		
Scratch-Translation		
Scratch-Networking		
Kernel-Objects	instance ? class	

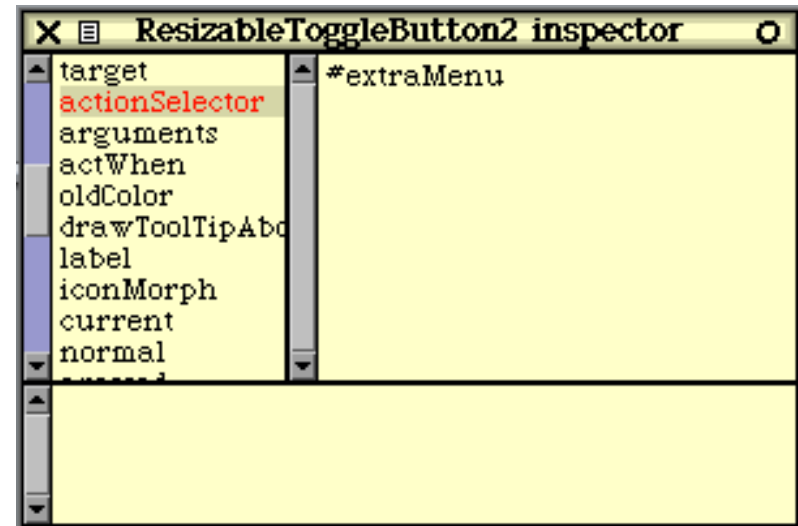
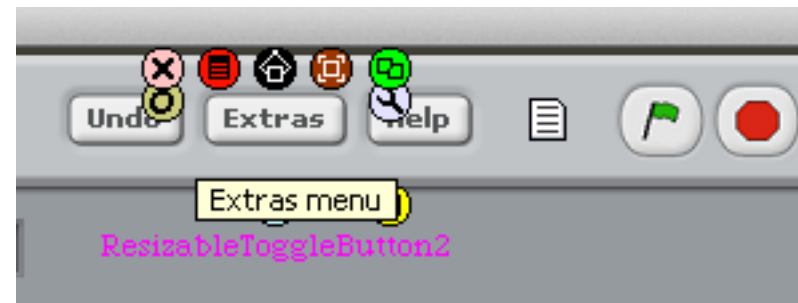
infoString

"Answer a string for this media, typically something about its size."

+ self totalSeconds hhmms

Smalltalk Spike

- Could I even understand how things worked?
- Worried about the “accessibility requirements” of the project
 - Keyboard only navigation
 - Sound + Microphone (cross platform)
- Did an initial spike, and found that I could get something running quickly



Refactoring to a “handler” hierarchy

The image shows two overlapping Smalltalk IDE windows. The left window, titled "SenseProtocolHandler hierarchy", displays the class hierarchy for "Scratch-Sensor Board". It shows "Object" as the superclass, with "SenseProtocolHandler" as a subclass. Below "SenseProtocolHandler", there are "SenseBoardProtocolHandler" and "SenseSimulatorProtocolHandl". A list of methods is shown: "-- all --", "internal", "hooks", "accessing", "configuration", "stepping", "pwm ops", "stepper ops", and "serial port". At the bottom, there are buttons for "senders", "implementors", "versions", "inheritance", and "hierarchy".

The right window, titled "SenseProtocolHandler hierarchy", shows the "Scratch-Sensor Simulator" hierarchy. It lists "Object", "SenseProtocolHandl", "GoGoProtocolHand", "SenseBoardProtoc", and "SenseSimulatorPr". The "led ops" category is selected, showing methods: "accessing", "stepping", "pwm ops", "stepper ops", "serial port", "motor ops", "sensor ops", and "led ops". On the right side, there are instance variables: "isLedOn:", "ledAllOff", "ledAllOn", "ledOff:", "ledOn:", and "ledValueOf:". Below this, there is a timestamp "tam 12/17/2013 02:21", a status "led ops • 6 implementors • only in change", and buttons for "senders", "implementors", "versions", "inheritance", "hierarchy", and "inst vars".

The bottom part of the right window shows the implementation of the "ledOn: ledNum" message:

```
ledOn: ledNum
    (super ledOn: ledNum) ifFalse: [ +false ].

    ledValues at: ledNum put: (TimedValue value: true).
    +true
```

Smalltalk is so malleable...

- Fixed some of the bugs/missing features
 - Package saving problems
 - “Create it” in the debugger
- Re-incarnated Sunit 2.7, Modernized approach with CI

```
echo "Creating ou-latest.image..."
cp ou-release.image ou-latest.image
cp ou-release.changes ou-latest.changes
rm $FNAME.log 2> /dev/null

echo "Launching image for processing..."
./Squeak\ 4.2.5beta1U-VM.app/Contents/MacOS/Squeak\ VM\ Opt ou-latest.image -filein $FSCRIPT $FNAME.log

if [ ! -f "$FNAME.log" ] || ! (grep -i -e "clean below:" --quiet "$FNAME.log"); then
    echo
    echo "ERROR: Problems detected in release, didn't finish building. Check $FNAME.log !!!!!"
    exit 2
fi

if [ ! -f "$FNAME.log" ] || (sed '/IGNORE/,/clean below:/d' "$FNAME.log" | grep -i -f errors.regex --color --quiet); then
    echo
    echo "ERROR: Problems detected in release, errors in log. Check $FNAME.log !!!!!"
    exit 2
else
    echo "Changesets successfully applied to image..."
    echo

    ant touch build
```

The Hanging Image

```
Debug console
To close: F2 -> 'debug options' -> 'show output console'
To disable: F2 -> 'debug options' -> 'show console on errors'
Printing all processes:
Process
;3736908 Semaphore>initSignals
;3736772 Semaphore class>new
;3736680 Delay class>forSeconds:
;3736588 SenseEmulatorProtocolHandler>processIncomingData
;3736496 SensorBoardMorph>processIncomingData
;3736128 SensorBoardMorph>waitForDataWithTimeout
;3736036 ScriptableScratchMorph>checkForSensorBoard
;3735944 ScriptableScratchMorph>sensorPressed:
;3736220 [] in UpdatingStringMorph>valueFromTargetOrNil
;3735788 BlockContext>ifError:
;3735696 UpdatingStringMorph>valueFromTargetOrNil
;3735604 UpdatingStringMorph>readFromTarget
;3735512 UpdatingStringMorph>step
;3735420 Morph>stepAt:
```


Learning Morphic

System Browser [SenseLedMorph]

- SUnitTests
- SUnitUI
- TestMock-Arguments
- TestMock-Test
- TestMock-Context
- TestMock-Error
- TestMock-Expectation
- TestMock-Return
- Scratch-Sensor Simulat
- Scratch-Sensor Tests

SenseLedMorph

- SenseMicrophoneMorph
- SenseProtocolHandlerPr
- SenseServoMorph
- SenseSimulatorProtocol
- SenseSliderMorph
- SenseStepperMorph
- SenseStringFieldMorph
- SenseSwitchMorph
- SenseThreePhaseButton
- SpinButtonMorph

instance ? class

-- all --

- initialization
- notifying
- accessing
- stepping

step

stepTime

tam 11/12/2013 15:52 • stepping • 65 implementors • only in change set ou-simulator •

senders implementors versions inheritance hierarchy inst vars class vars diffs

step

```

| timedValue |

(timedValue := stateBlock value) > lastValue ifTrue: [
    timedValue value
    ifTrue: [
        super color: onColor.
        self notifyState: true]
    ifFalse: [
        super color: (onColor mixed: 0.5 with: Color black).
        self notifyState: false].
    lastValue := timedValue]

```

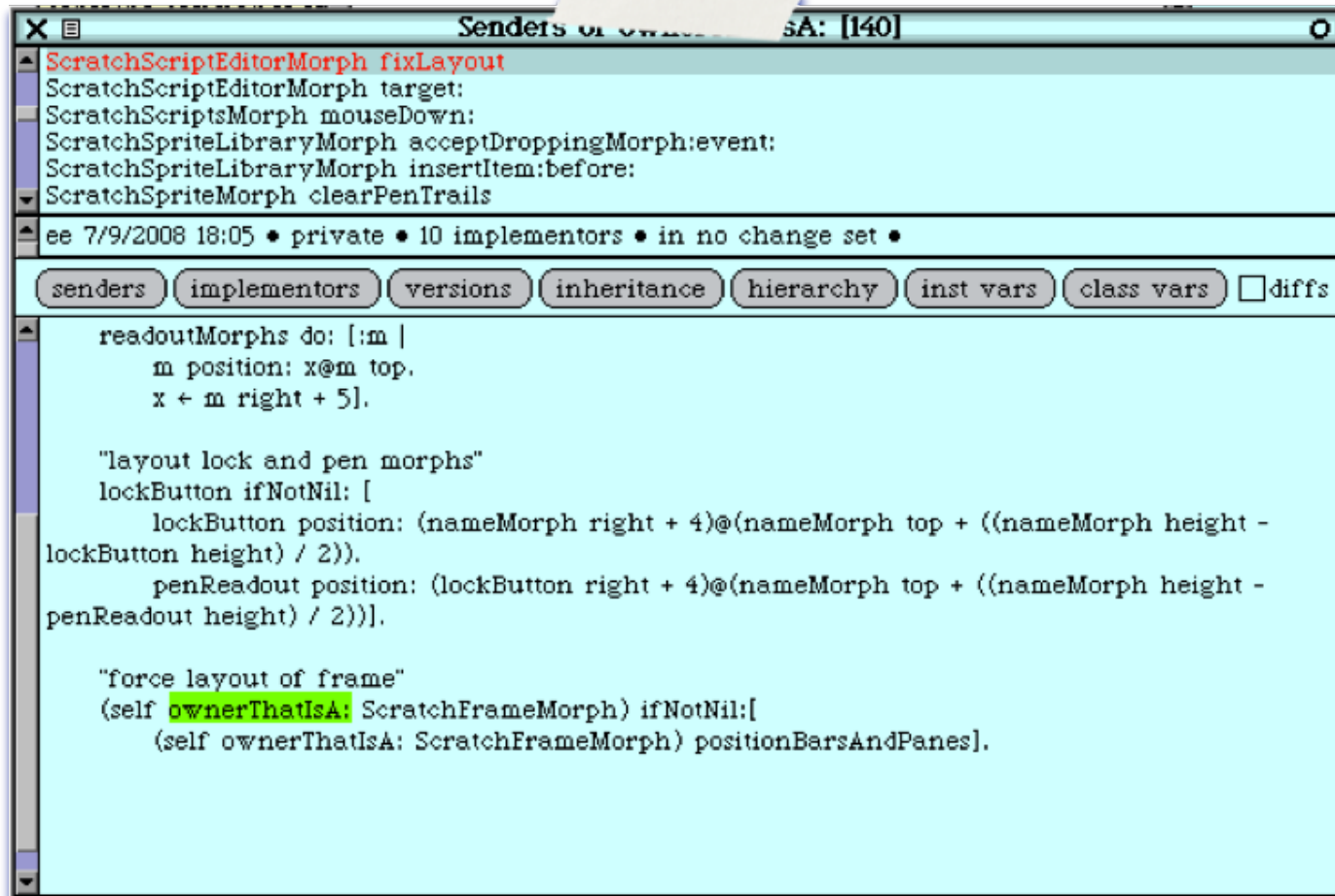
```

step

self whenChangedValue: [:ledOn |
    ledOn
        ifTrue: [
            super color: onColor.
            self notifyState: true]
        ifFalse: [
            super color: (onColor mixed: 0.5 with: Color black).
            self notifyState: false].
]

```

Scratch is not a good example of Smalltalk...



```
ScratchScriptEditorMorph fixLayout
ScratchScriptEditorMorph target:
ScratchScriptsMorph mouseDown:
ScratchSpriteLibraryMorph acceptDroppingMorph:event:
ScratchSpriteLibraryMorph insertItem:before:
ScratchSpriteMorph clearPenTrails

ee 7/9/2008 18:05 • private • 10 implementors • in no change set •

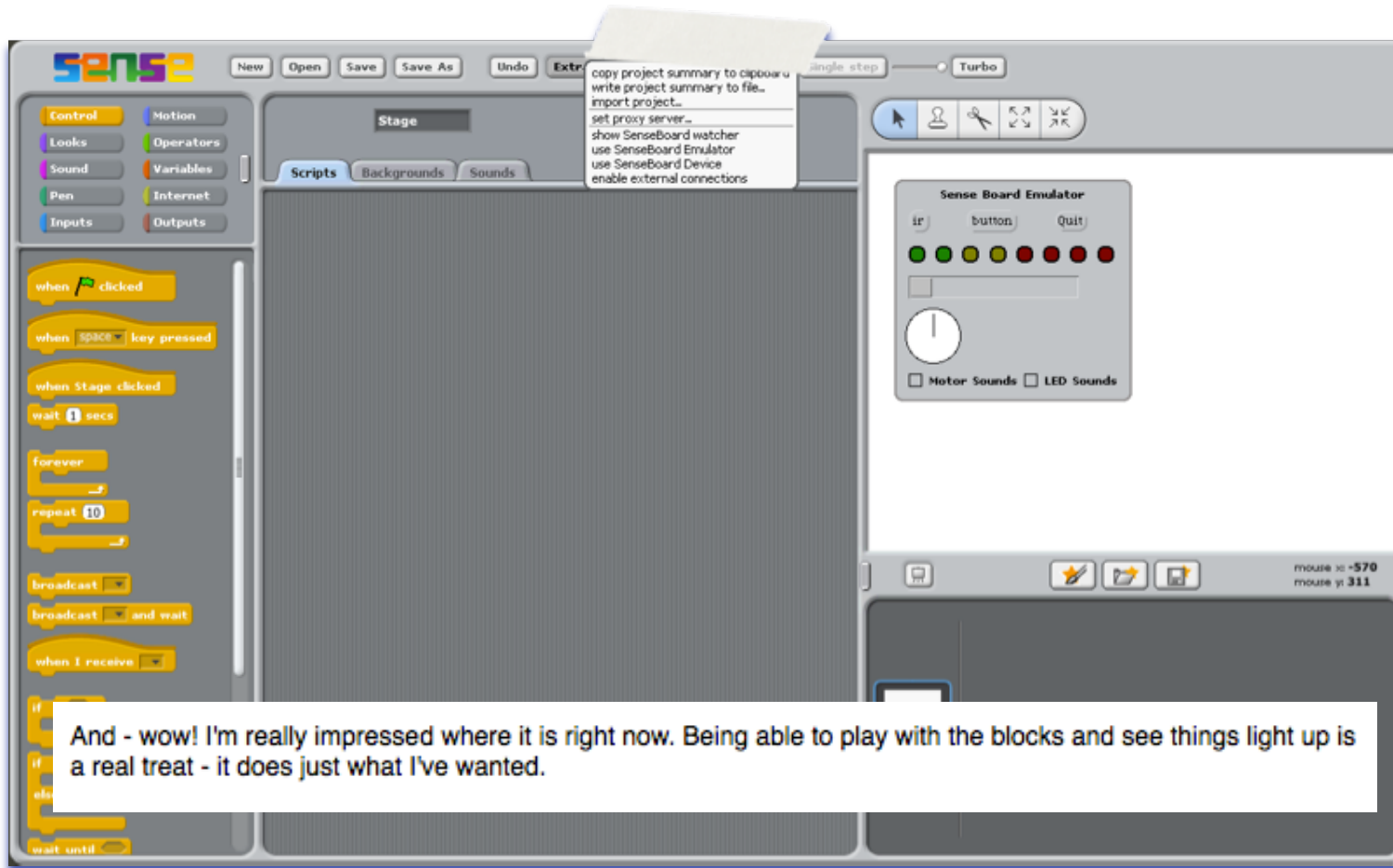
senders  implementors  versions  inheritance  hierarchy  inst vars  class vars  diffs

readoutMorpha do: [:m |
    m position: x@m top.
    x + m right + 5].

"layout lock and pen morphs"
lockButton ifNotNil: [
    lockButton position: (nameMorph right + 4)@(nameMorph top + ((nameMorph height -
lockButton height) / 2)).
    penReadout position: (lockButton right + 4)@(nameMorph top + ((nameMorph height -
penReadout height) / 2))].

"force layout of frame"
(self ownerThatIsA: ScratchFrameMorph) ifNotNil:[
    (self ownerThatIsA: ScratchFrameMorph) positionBarsAndPanels].
```

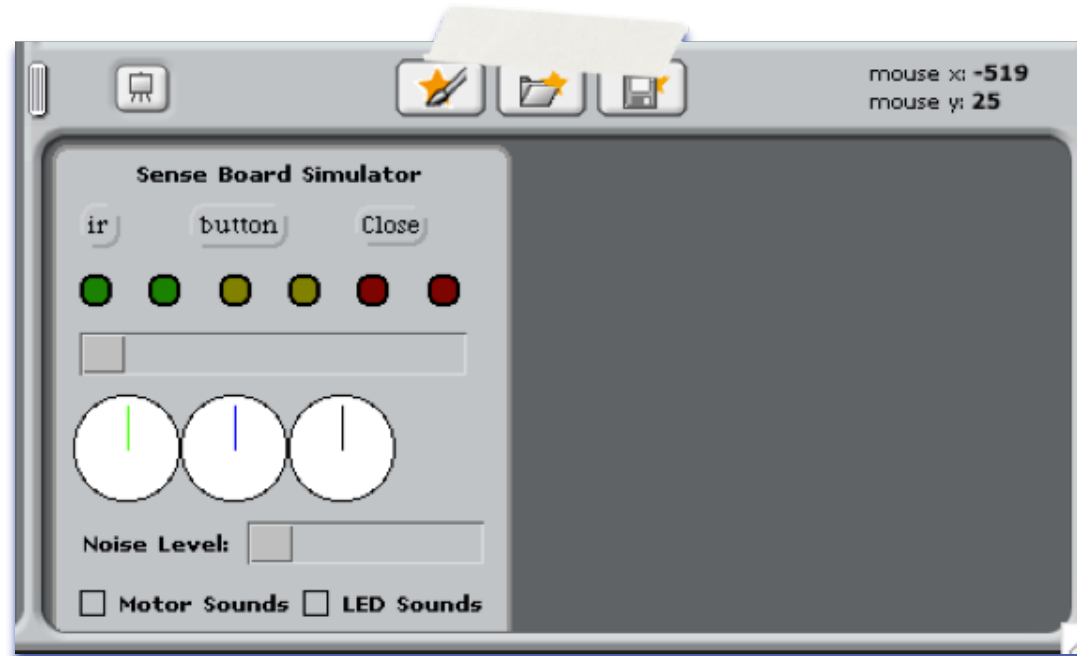
Iterative Development: Initial working version



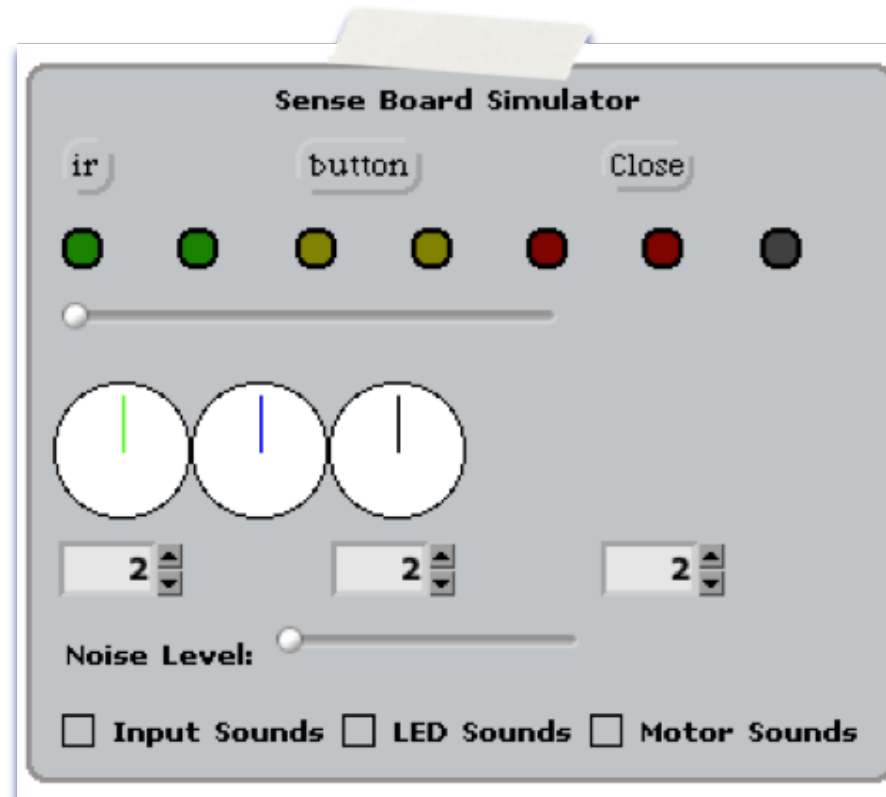
The screenshot shows the SENSE programming environment. On the left is a palette with categories: Control, Looks, Sound, Pen, Inputs, Motion, Operators, Variables, Internet, and Outputs. The main workspace is titled 'Stage' and contains a script of blocks: 'when clicked', 'when space key pressed', 'when Stage clicked', 'wait 1 secs', a 'forever' loop containing a 'repeat 10' block, 'broadcast' blocks, and 'when I receive' blocks. A 'Sense Board Emulator' window is open on the right, featuring a 'button' and 'Quit' control, a row of seven colored LEDs (green, yellow, red, red, red, red, red), a progress bar, a clock, and checkboxes for 'Motor Sounds' and 'LED Sounds'. A tooltip is visible over the 'Extra' menu, listing options like 'copy project summary to clipboard', 'write project summary to file', 'import project...', 'set proxy server...', 'show SenseBoard watcher', 'use SenseBoard Emulator', 'use SenseBoard Device', and 'enable external connections'. A text box at the bottom of the screenshot contains the following text:

And - wow! I'm really impressed where it is right now. Being able to play with the blocks and see things light up is a real treat - it does just what I've wanted.

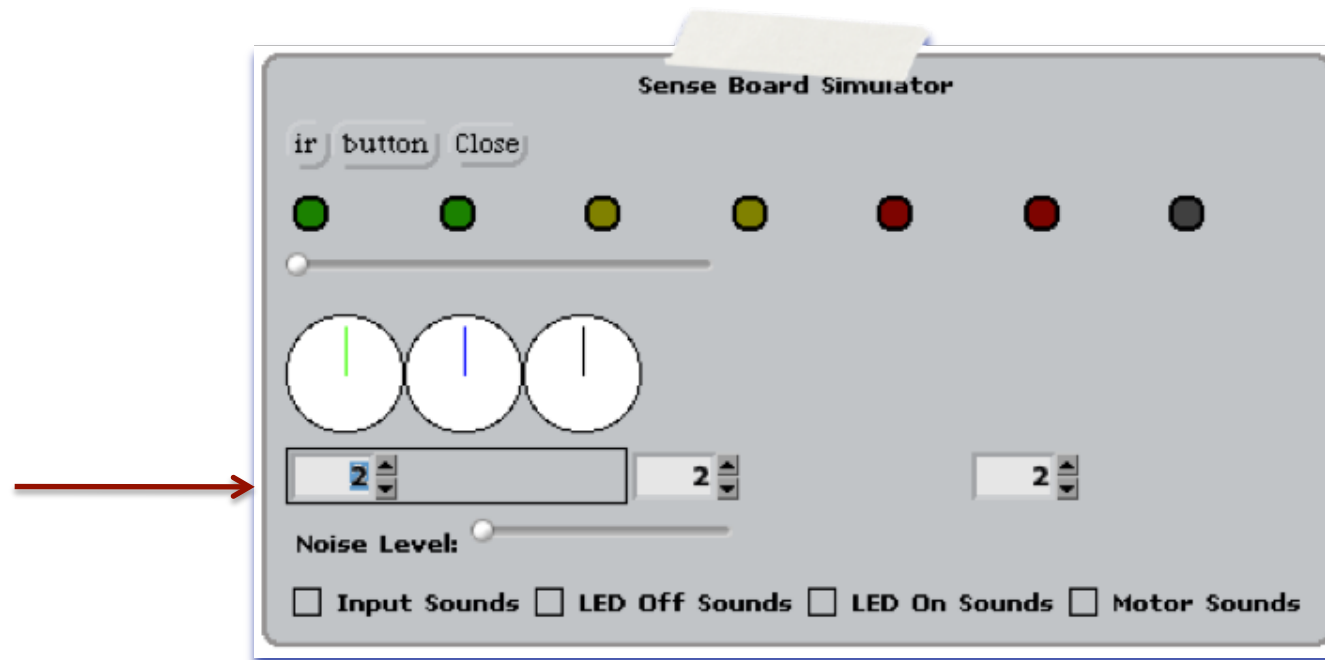
Iteration 2 – multiple motors



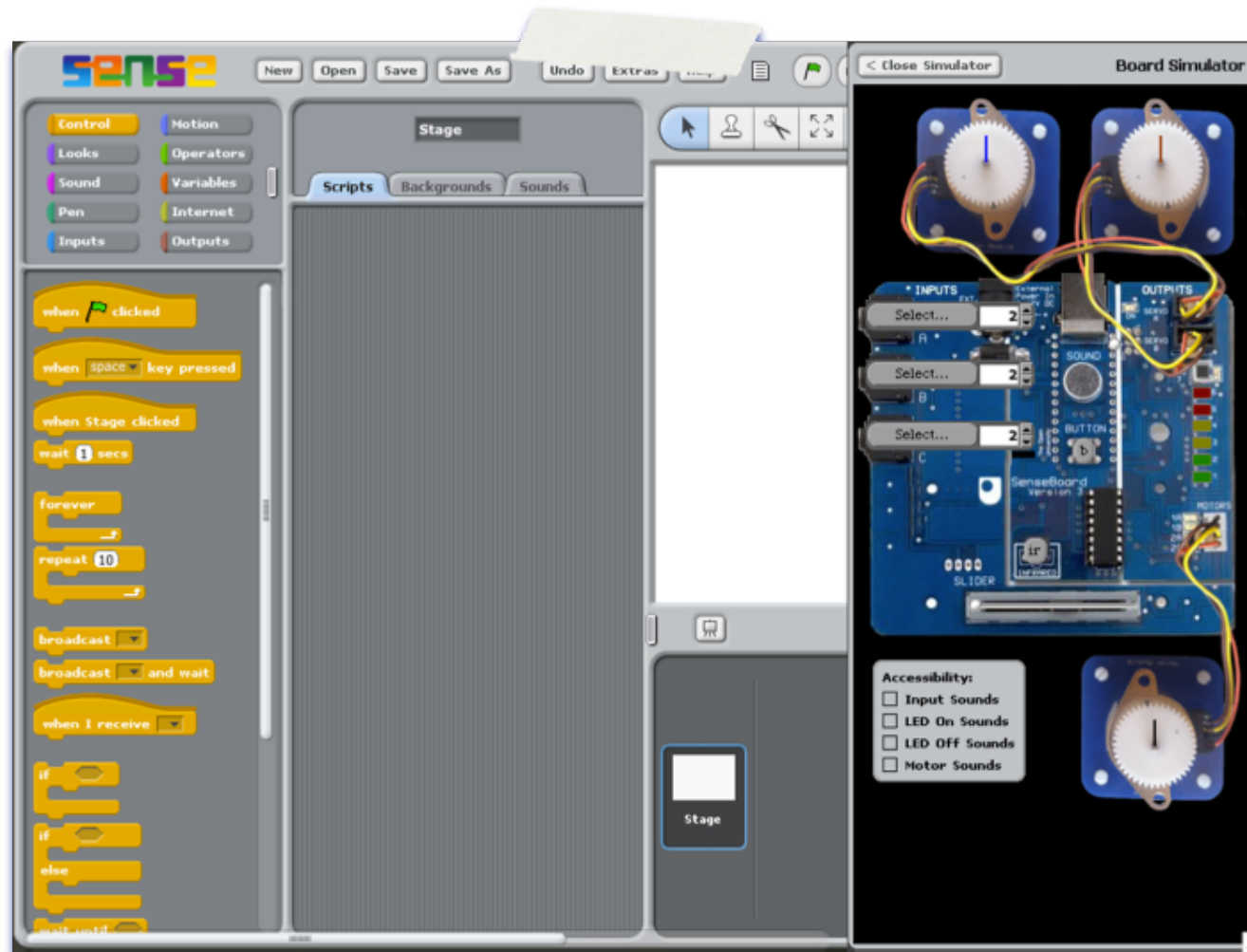
Iteration 3 – input ports



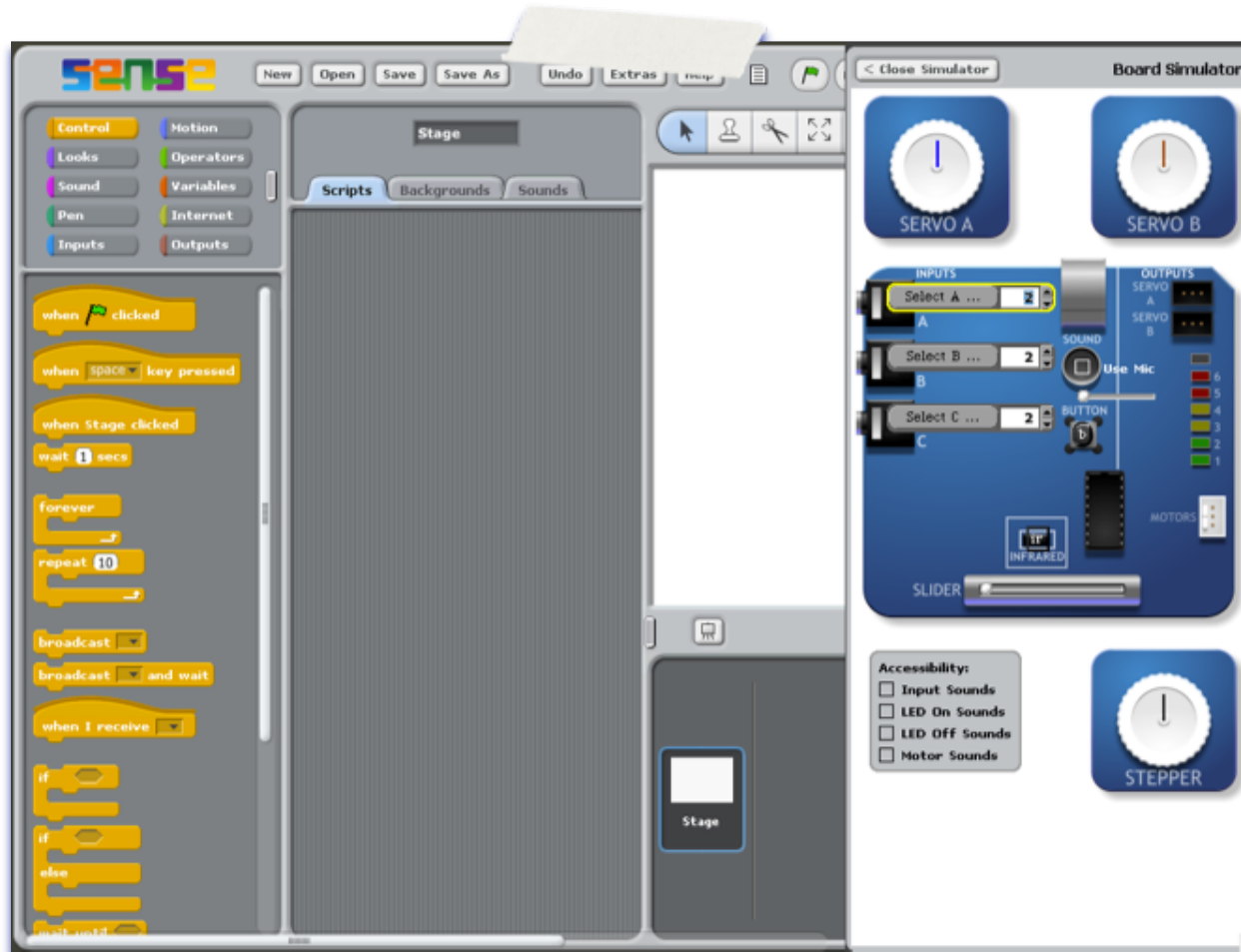
Iteration 4 – accessibility tab selection



Iteration 5 – dressed up



Iteration 6 – OU cognitive makeover



Coping with Accessibility

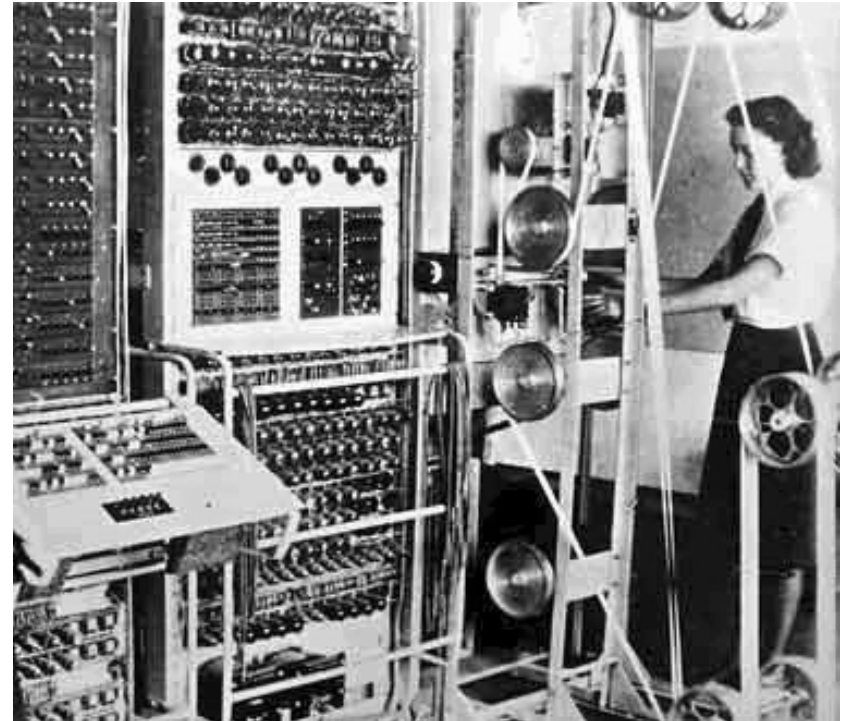
- Squeak was not made with keyboard access in mind
- Teaching widgets how to tab, show focus
- Menu's continue to be a problem

Inspiration from Accessibility

- When I hooked up sound accessibility and tried it out - it sounded awful!
- However I found it interesting that I could reason about code I knew nothing about...

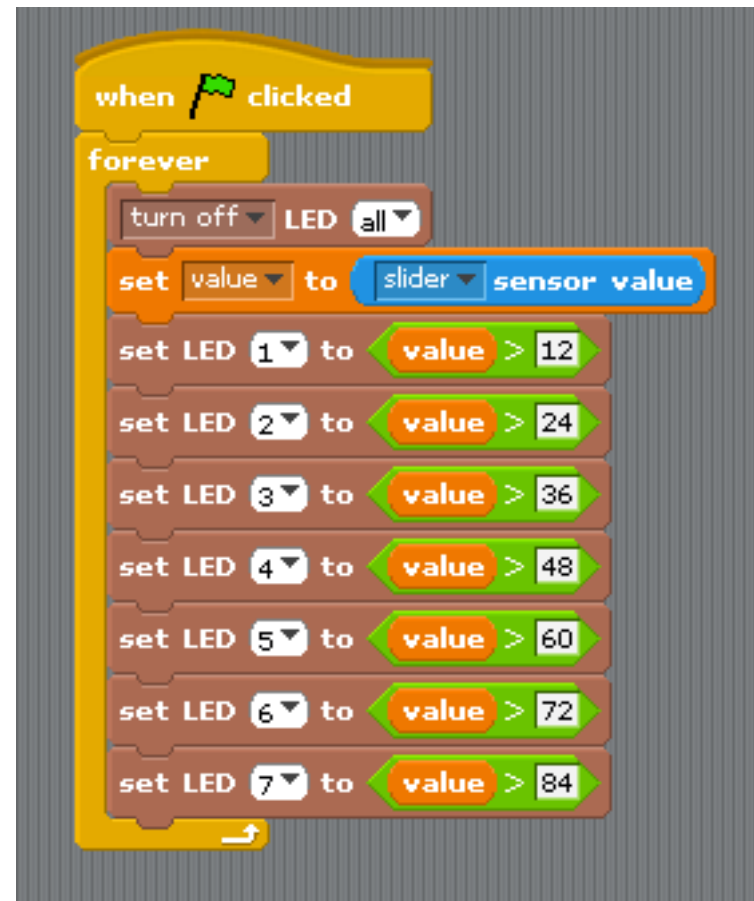
The sound of computing

- When I mentioned listening to your program at UKST...
- It turns out old mainframe programmers could hear infinite loops from the hardware...



Demo of Sound in action...

- Constant event loop
- What does this sound like?



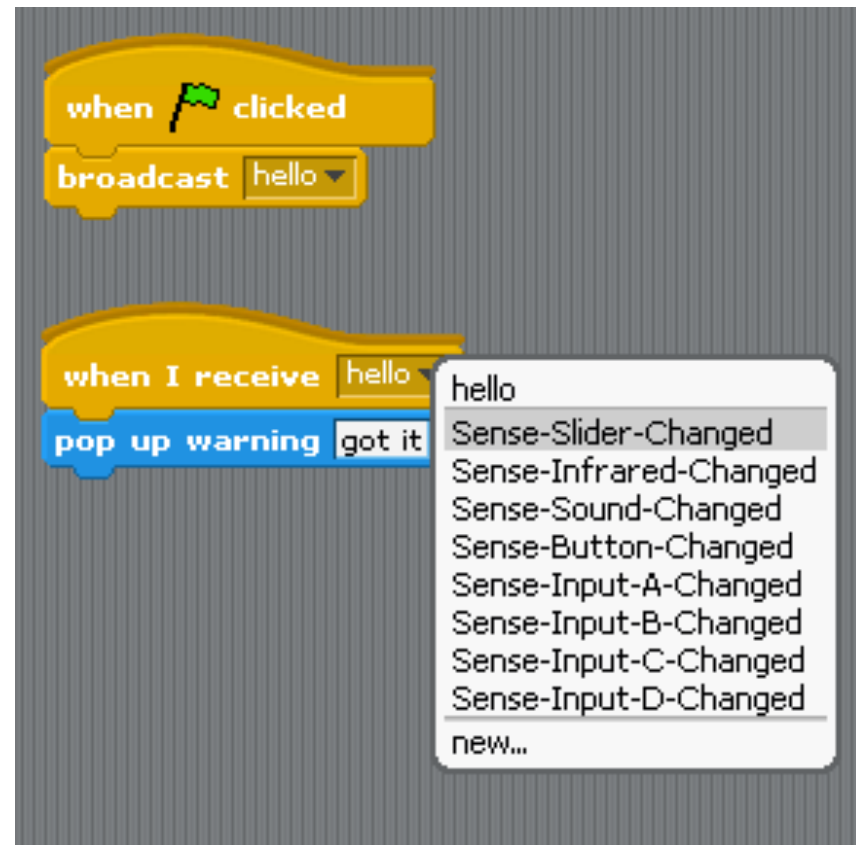
But polling is bad... Can we do better?

- Scratch already has a broadcast mechanism
- Couldn't we hook into that somehow?



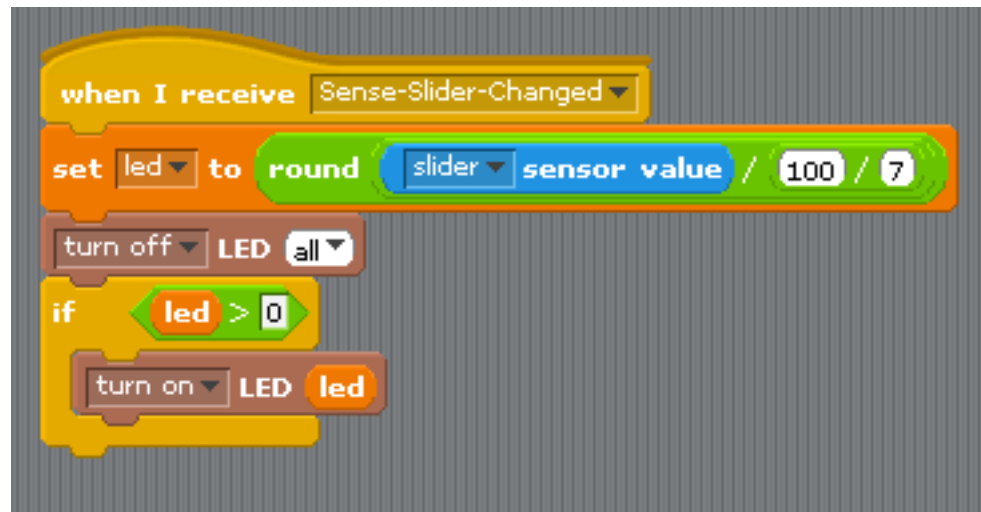
If you know the right lines of code...

- Convince scratch there are some predefined events
- Wire in some support for them



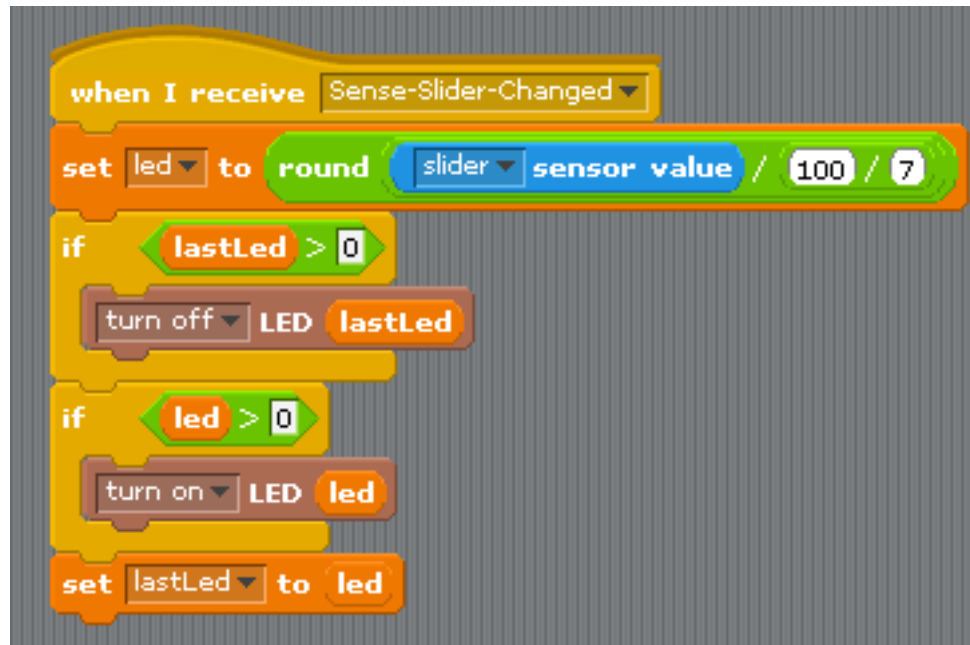
Now rewrite the LED Slider with events

- But it still doesn't quite sound right...



This sounds better... but ...

- There is still a slight sound overlap... we can be more efficient...



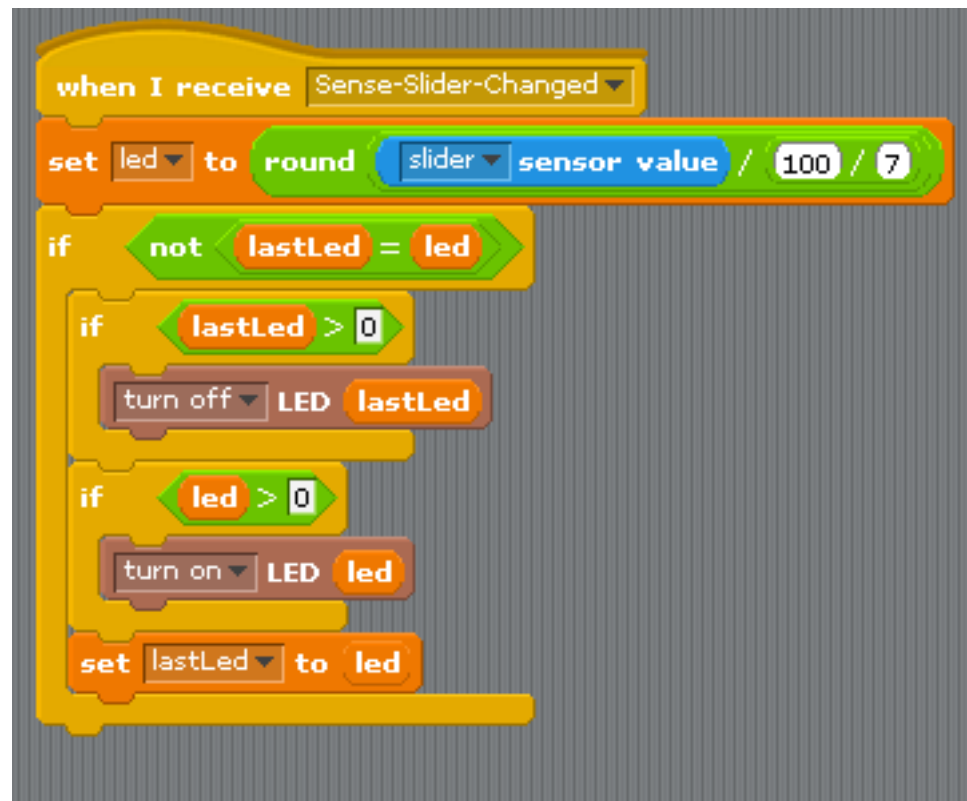
```
when I receive Sense-Slider-Changed
set led to round (slider sensor value / 100 / 7)
if lastLed > 0
  turn off LED lastLed
if led > 0
  turn on LED led
set lastLed to led
```

The image shows a Scratch script with the following blocks:

- when I receive** Sense-Slider-Changed
- set** led to round (slider sensor value / 100 / 7)
- if** lastLed > 0
 - turn off** LED lastLed
- if** led > 0
 - turn on** LED led
- set** lastLed to led

This sounds much better

- Although it is a bit more complicated, there is less processing...



```
when I receive Sense-Slider-Changed
set led to round (slider sensor value / 100 / 7)
if (not (lastLed = led))
  if (lastLed > 0)
    turn off LED lastLed
  if (led > 0)
    turn on LED led
  set lastLed to led
```

The image shows a Scratch script for controlling an LED. It starts with a 'when I receive' block set to 'Sense-Slider-Changed'. This is followed by a 'set' block that calculates the LED state: 'led' is set to the rounded value of 'slider sensor value' divided by 100 and then by 7. A large 'if' block follows, with the condition 'not (lastLed = led)'. Inside this block, there are two nested 'if' blocks. The first checks 'if (lastLed > 0)' and contains a 'turn off LED lastLed' block. The second checks 'if (led > 0)' and contains a 'turn on LED led' block. Finally, a 'set lastLed to led' block updates the 'lastLed' variable.

Related Work

- Craig Latta – VM Sonification

virtual machine sonification

A while back I instrumented the Smalltalk virtual machine to generate **visualizations of object memories**. Now I'm designing instrumentations for sonification, so that the virtual machine will generate sound as it runs. This could be especially interesting for insights into the virtual machine's operation, since I think it will run usefully in real time. It might be a useful profiling tool, for example. If you have any ideas for how to map messages and executed instructions to sound, please let me know!

Share this:

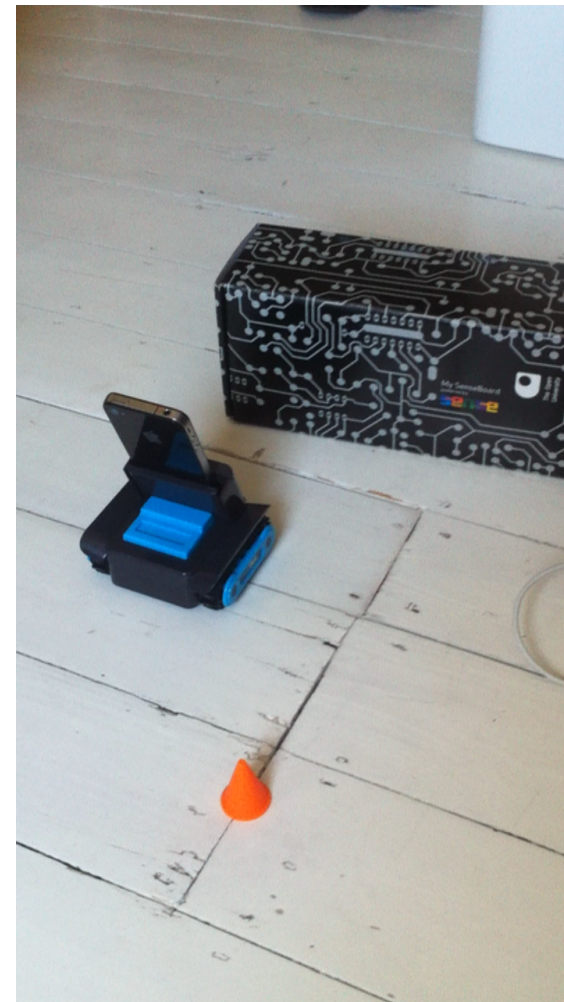


Google+



Events & Remote Sensor Protocol

- The simulator and remote protocol also provide possibilities to interface with devices
- iPhone app sending remote data as if from the SenseBoard
- Controlling a Robot like the YoBot!



Thankyou

- Tim Mackinnon
tim.mackinnon@morethan.technology