Bezvadu Sensoru Tīkli

Programmēšanas abstrakcijas
(jeb īsumā par BST OS)

Reinholds Zviedris
Datorikas fakultāte
Latvijas Universitāte
04.11.2015.

KĀ SENSORU TĪKLU PROGRAMMĒŠANU PADARĪT VIENKĀRŠĀKU

Sensoru tīklu galvenais uzdevums:

Savākt informāciju!

(Un darīt to ilgi!)

Informācijas savākšana:

- Jēlo datu interpretāciju
- Informācijas apstrāde, agregācija
- Rezultātu nogādāšana līdz apstrādes vietai

Kāpēc nesūtīt visu uz bāzes staciju?

- Komunikācija tērē enerģiju
- Radio kapacitāte ir ierobežota

Galvenās pieejas

- Programmēt katru mezglu veidu atsevišķi
- Makro programmēšana
- Aģentu bāzētas pieejas
- Vaicājumu (query) bāzētas pieejas

Makro programmēšana

- Ļauj programmēt visu tīklu kopumā
- Kompilators globālo programmu nokompilē individuāliem mezgliem

Makro programmēšanas priekšrocības un trūkumi

+ Ļauj domāt par visu tīklu kopumā
- Šāda paradigmas maiņa nav vienkārša
- Sarežģīta kompilatoru būve

Makro programmēšanas piemērs: Pleaides

- Specifiska valoda, kas kompilējas uz TinyOS programmām

```
1: #include "pleiades.h"
2: boolean nodelocal isfree=TRUE;
3: nodeset nodelocal neighbors;
4: node nodelocal neighborIter;

5: void reserve(pos dst) {
6:  boolean reserved=FALSE;
7:  node nodeIter,reservedNode=NULL;
8:  node n=closest_node(dst);
9:  nodeset loose nToExamine=add_node(n, empty_nodeset());
10: nodeset loose nExamined=empty_nodeset();


11: if(isfree@n) {
12:  reserved=TRUE; reservedNode=n;
13:  isfree@n=FALSE;
14:  return;
15: }

16: while(!reserved && !empty(nToExamine)){
17:  cfor(nodeIter=get_first(nToExamine);nodeIter!=NULL;
          nodeIter = get_next(nToExamine)){
18:   neighbors@nodeIter=get_neighbors(nodeIter);
19:   for(neighborIter@nodeIter=get_first(neighbors@nodeIter);
          neighborIter@nodeIter!=NULL;
          neighborIter@nodeIter=get_next(neighbors@nodeIter)){
20:    if(!member(neighborIter@nodeIter,nExamined))
21:     add_node(neighborIter@nodeIter,nToExamine);
22:   }
23:   if(isfree@nodeIter){
24:    if(!reserved){
25:     reserved=TRUE; reservedNode=nodeIter;
26:     isfree@nodeIter=FALSE;
27:     break;
28:    }
29:   }
30:   remove_node(nodeIter,nToExamine);
31:   add_node(nodeIter,nExamined);
32:  }
33: }
34:}
```

Ko dara šī programma?

```
1: #include "pleiades.h"
2: boolean nodelocal isfree=TRUE;
3: nodeset nodelocal neighbors;
4: node nodelocal neighborIter;

5: void reserve(pos dst) {
6:  boolean reserved=FALSE;
7:  node nodeIter,reservedNode=NULL;
8:  node n=closest_node(dst);
9:  nodeset loose nToExamine=add_node(n, empty_nodeset());
10: nodeset loose nExamined=empty_nodeset();


11: if(isfree@n) {
12:  reserved=TRUE; reservedNode=n;
13:  isfree@n=FALSE;
14:  return;
15: }

16: while(!reserved && !empty(nToExamine)){
17:  cfor(nodeIter=get_first(nToExamine);nodeIter!=NULL;
          nodeIter = get_next(nToExamine)){
18:   neighbors@nodeIter=get_neighbors(nodeIter);
19:   for(neighborIter@nodeIter=get_first(neighbors@nodeIter);
          neighborIter@nodeIter!=NULL;
          neighborIter@nodeIter=get_next(neighbors@nodeIter)){
20:    if(!member(neighborIter@nodeIter,nExamined))
21:     add_node(neighborIter@nodeIter,nToExamine);
22:   }
23:   if(isfree@nodeIter){
24:    if(!reserved){
25:     reserved=TRUE; reservedNode=nodeIter;
26:     isfree@nodeIter=FALSE;
27:     break;
28:    }
29:   }
30:   remove_node(nodeIter,nToExamine);
31:   add_node(nodeIter,nExamined);
32:  }
33: }
34:}
```

Atbilde: autostāvvietas lietotne, kas realizēta Pleiades

Aģentu bāzētas programmas

- Kas ir aģents?

Aģentu bāzētas programmas

- Aģents ir programma, kas ceļo starp mezgliem
- Nevis dati nāk pie programmas, bet programma iet pie datiem
- Aģenti autonomi pārvietojas
- Vienlaikus tīklā var darboties vairāki aģenti
- Kaut kas ļoti eksotisks

Aģentu priekšrocības

- Ļoti plašas iespējas
- Iespējami mazs datu pārraides apjoms
- Spēcīga decentralizācija, apstrāde seko datiem
- Spēja dinamiski reaģēt uz izmaiņām tīklā

Aģentu trūkumi

- Sarežģīti modelēt tīkla komponentes un to sadarbību
- Grūti paredzēt izmaiņas, uz kurām jāreaģē
- Sarežģīta testēšana un atkļūdošana

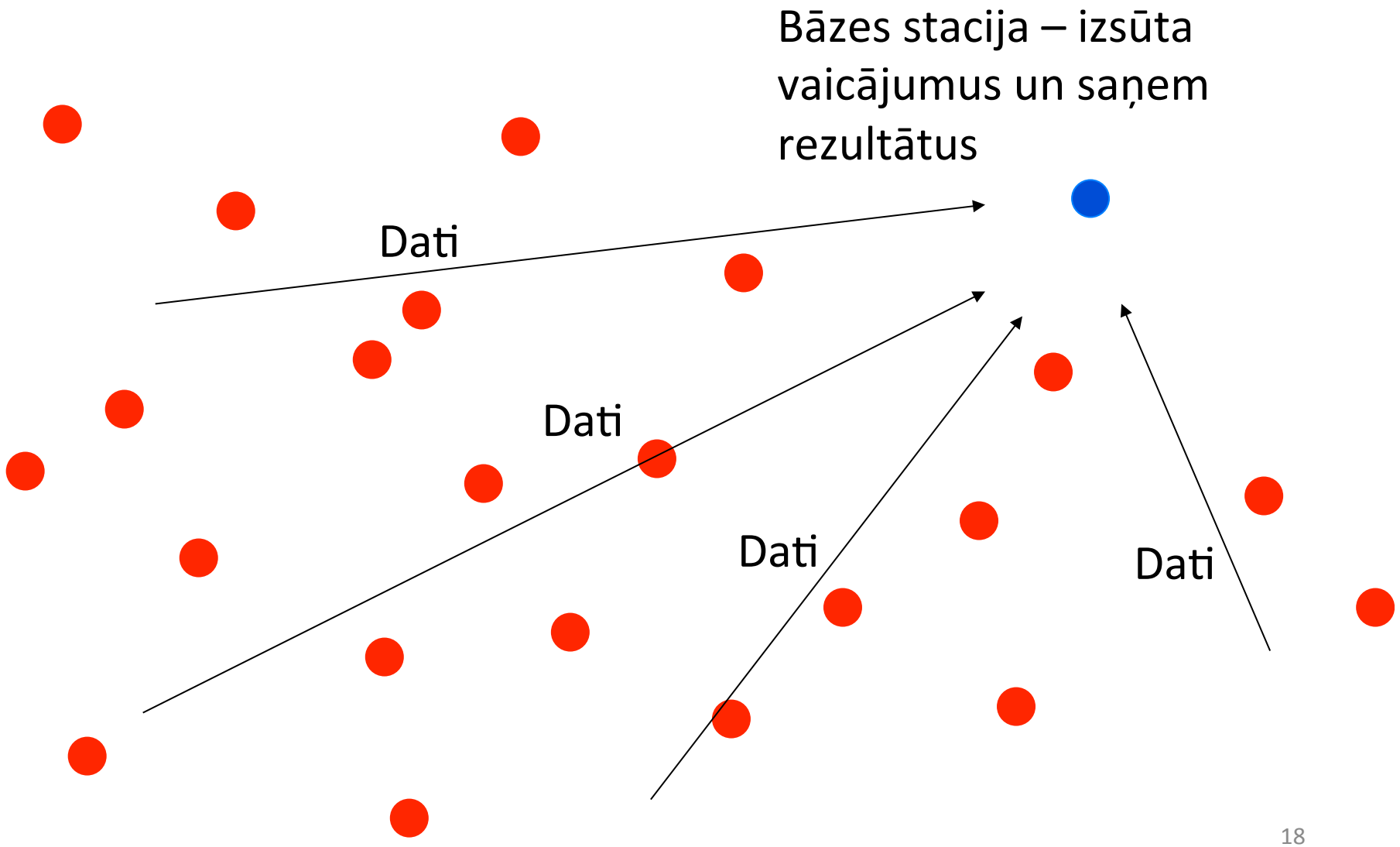
Vaicājumu bāzēta BST programmēšana

Vaicājumu bāzētas pieejas

- Tīklā iesūta vaicājumu, atpakaļ nāk atbilde
- Parasti centralizēti risinājumi: bāzes stacija vaicā, tīkls atbild
- Vispopulārākā pieeja – vienkārša realizācija

Piemēri: TinyDB, SwissQM, ...

Tipiska vaicājumu pieejas arhitektūra



Pieejas pamatmērķis

- Sensoru tīkls kā dalīta (distributed) datu bāze
- Problēmas:
  - Ierobežota atmiņa
  - Nepieciešams kopīgs *duty cycle*
  - Mezgli parasti nav uzticami
  - Dati var būt trokšņaini
  - Parasti datiem svarīgs arī laiks un vieta

Vaicājumu pieejas priekšrocības

- Vienkārša realizācija
- Abstrahēšanās no tīkla topoloģijas

Vaicājumu pieejas trūkumi

- Pārsvarā triviāla datu apstrāde
- Vaicājums parasti vai nu konkrētam mezglam, vai visam tīklam
- (Sarežģīti apstrādāt telpas un laika atribūtus)

Vaicājumu pieejas piemērs: TinyDB

- SQL-tipa dalītā datu bāze sensoru tīkliem
- Darbojas kā slānis virs TinyOS
- Visi sensoru mezgli kā daļa no tabulas
- Katrs sensoru lasījums kā viens ieraksts
- Vaicājumus izsūta un atbildes savāc bāzes stacija

- Sīkāk rakstā: S. Madden, M. Franklin, J. Hellerstein, and W. Hong, “TinyDB: an acquisitional query processing system for sensor networks,” ACM Transactions on Database Systems (TODS), vol. 30, no. 1, pp. 122–173, 2005.

TinyDB vaicājuma piemērs

```
SELECT nodeid, light, temp
  FROM sensors
  SAMPLE PERIOD 1s FOR 10s
```

Lasa gaismu un temperatūru ik sekundi, 10 sekunžu garumā

TinyDB agregācija

- Datu agregāciju veic pēc iespējas tuvu datu avotam, notiek automātiski

SELECT AVG(volume), room FROM sensors
WHERE floor = 6
GROUP BY room
HAVING AVG(volume) > *threshold*
SAMPLE PERIOD 30s

TinyDB kopsavilkums

- Dalītā datu bāze sensoru tīkliem
- Slānis virs TinyOS [1.x]
- Piedāvā vienkāršu saskarni, efektīvu agregāciju

Trūkumi:

- Paplašināmība
- Iebūvēta vaicājumu valoda ierobežo
  - nevar veidot patvaļīgas programmas

SwissQM virtuālā mašīna

- ETH Zurich, 2007.g

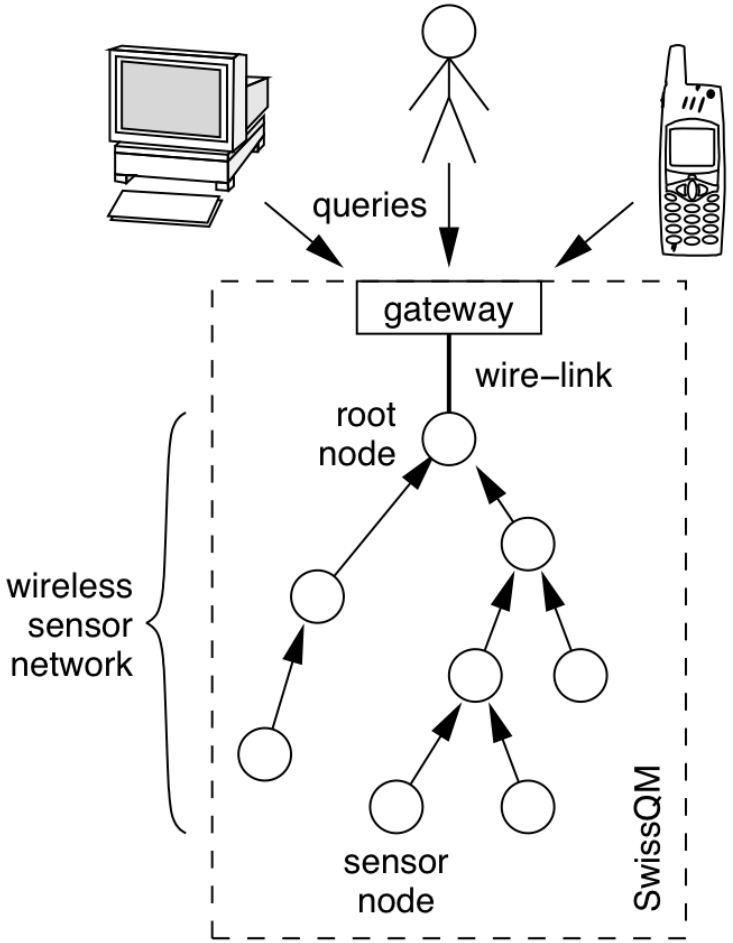
- Izteiksmīgāka par TinyDB
- Kompaktāka par Mate VM

- QM = Query Machine
  - Query [Engine] + [Virtual] Machine

R. Muller, G. Alonso, and D. Kossmann, “A virtual machine for sensor networks,” in Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007, pp. 145–158, ACM, 2007.

SwissQM pieeja

- Gateway kā ”gudrais mezgls”
  - vaicājumu pārveidošana mezglu programmās
  - vaicājumu merging
  - multi-query optimization
- Paaugstina efektivitāti
  - motes izpilda tikai to, kas vajadzīgs (sense, aggregate, transmit)



Gateway

- Daudz plašāka funkcionalitāte kā pierasts
  - multi-query optimization and merging
- Var "klausīt" dažādām vaicājumu valodām
  - SQL, XQuery, web services (t.sk. vienlaikus)
- Paplašināms
  - var realizēt dažādu papildus funkcionalitāti

- Ģenerē *byte-code,* ko izpildīt meglu VM

XQuery piemērs

Atgriezt NodeID motēm, kam temp > 60 :

```
for $n in xt:sample($sensors, 10s)//node
    where $n/temp gt 60
    return $n/nodeid
```

vaicājums XML “dokumentam” ar <node/> elementiem, kas satur <nodeid> un <temp>.

Agregācijas piemērs (SQL)

Atrast vidējo temperatūras vērtību nodēm, kam ir līdzīgi gaismas sensora rādījumi:

```
SELECT (light-512) / 10, AVG(temp)
    FROM sensors GROUP BY (light-512) / 10
    SAMPLE PERIOD 30s
```

vaicājums satur uz motēm izpildāmus aprēķinus

- TinyDB ir ļoti minimālas iespējas to veikt

SwissQM kopsavilkums

- Vienkāršāk programmējams kā Mate VM
- Plašākas iespējas par TinyDB
  - Dažāda veida interfeisi (XQuery, SQL, ...)
  - Multi-user, multi-programming
- Adaptējama / pielāgojama sistēma

MansOS

- Izstrādāta LU un EDI
- Paredzēta bezvadu sensoru tīkliem un resursierobežotām iegultajām iekārtām
- Mērķauditorija ir programmētāji ar C un UNIX programmēšanas pieredzi

MansOS pamatprincipi

- Viegli lietojama
- Portējama

Viegli lietojama BST OS

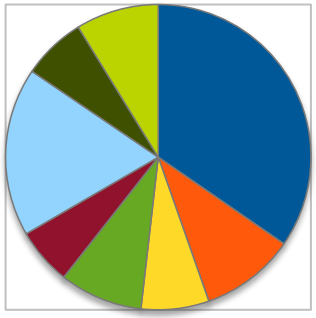
- Sistēmprogrammētājiem ar UNIX un C pieredzi
- Piemēram, TinyOS prasa specifiskas valodas (nesC) un programmēšanas paradigmas (event-based execution) apguvi
- Contiki prasa vismaz *protothreads* abstrakcijas apguvi
- MansOS pārsvarā var programmēt līdzīgi kā "klasiskas" UNIX sistēmas
- Uzinstalēt TinyOS ir netriviāli
- MansOS piedāvā *all included* binārās distribūcijas

Portējama BST OS

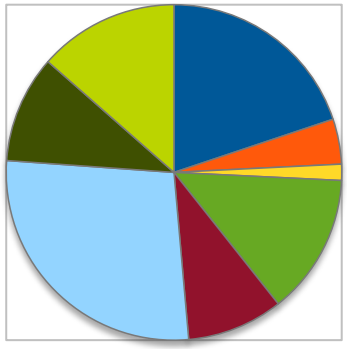
- Aparatūra bieži ir specifiska (vienam) lietojumam
- Daudz eksistējošo platformu, maz standartu
- MansOS darbojas gan uz MSP430, gan uz AVR arhitektūrām
  - t.sk. uz *Tmote Sky*, *Atmega (Arduino)*, *Zolertia Z1, AdvanticSYS XM1000, TI MSP430 Launchpad* u.c. platformām

MansOS pirmkoda sadalījums

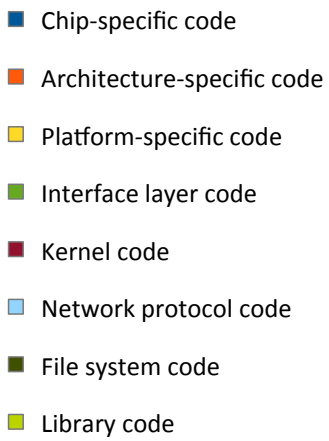
- ***Komponentu proporcijas:***



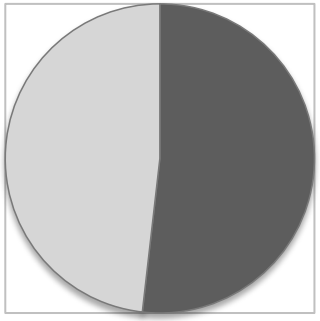
- *a) all*



- *b) TelosB*

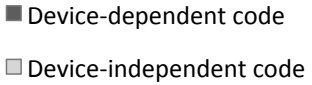


- ***Aparatūrneatkarīgā koda proporcijas:***



- *a) all*

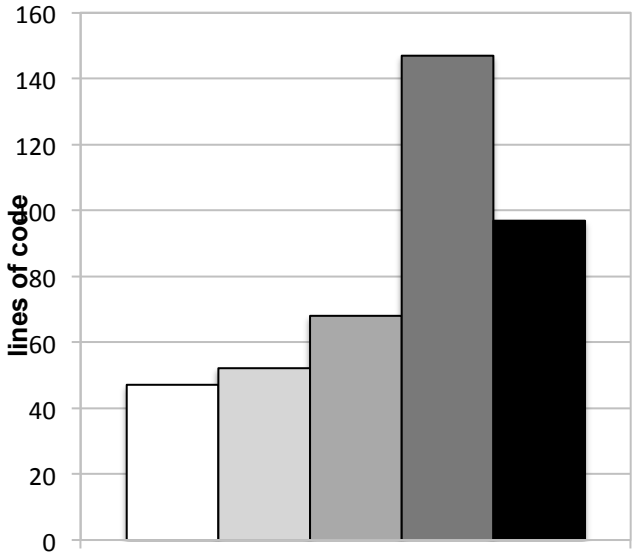
- *b) TelosB*



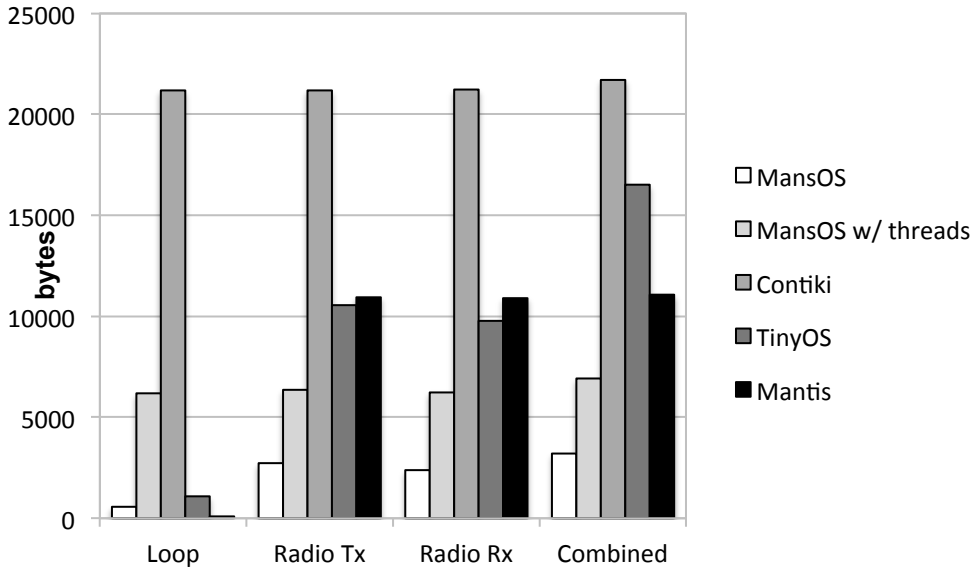
MansOS priekšrocības

- Īsāks lietotņu pirmkods
- Mazāks lietotņu binārais kods
- Vienkārši, bet robusti pavedieni (*threads*)

•Lietotnes koda rindiņu skaits



•Četru dažādu lietotņu binārā koda izmērs



MansOS funkcionalitāte

- Analogo un digitālo I/O portu apstrāde
- Digitālie piekļuves protokoli (SPI, I^2C, u.c.)
- Zema enerģijas patēriņa režīmi
- Radio komunikācija un tīkla steks
- Iespējams IPv6 atbalsts
- Vairākas aparatūras platformas
- *Run-time* pārvaldība
- *Run-time* pārprogrammēšana

MansOS pavedieni

- Vieglāk programmēt!
- Koda izkārtojums labāk atspoguļo programmas izpildīšanos
- Var izveidot bezgalīgos ciklus, bet sistēma turpinās strādāt…

Pavedienu pamatidejas

- Pavedienu ir maz (bieži pietiek ar diviem)
- Pavedieni ir savstarpēji “draudzīgi”
- Tiek darbināti “pa riņķi” (*round robin scheduling*)
- Kodola pavedienam vienmēr priekšroka
- Nianse - ***sleep()*** ieliek sistēmu zema enerģijas patēriņa režīmā ***tikai*** tad, ja citi pavedieni neaktīvi!

Papildus informācija par MansOS

- A. Elsts, G. Strazdins, A. Vihrov, and L. Selavo, "Design and Implementation of MansOS: a Wireless Sensor Network Operating System," Scientific Papers, University of Latvia, Volume 787, pp. 79-105, 2012.

- http://mansos.edi.lv
- http://selavo.lv/wiki/index.php/MansOS (novecojusi informācija)

SEAL valoda

- SEAL – **Se**nsor **A**pplication Development **L**anguage
- Domēnspecifiska valoda BST lietojumu izstrādei
- Balstīta uz MansOS
- Komplektēta ar vizuālo programmēšanas vidi (IDE)

- NesC kods (fragments):

```
#include "Timer.h"
#include "RadioSenseToLeds.h"

module RadioSenseToLedsC @safe(){
  uses {
    interface Leds;
    interface Boot;
    interface Receive;
    interface AMSend;
    interface Timer<TMilli> as MilliTimer0;
    interface Timer<TMilli> as MilliTimer1;
    interface Packet;
    interface Read<uint16_t>;
    interface SplitControl as RadioControl;
  }
}
implementation {

  message_t packet;
  bool locked = FALSE;

  event void Boot.booted() {
    call RadioControl.start();
  }

  event void RadioControl.startDone(error_t err) {
    if (err == SUCCESS) {
      call MilliTimer0.startPeriodic(2000);
      call MilliTimer1.startPeriodic(6000);
    }
  }
  event void RadioControl.stopDone(error_t err) {}

  event void MilliTimer0.fired() {
  }

  event void MilliTimer1.fired() {
  }

  event void Read.readDone(error_t result, uint16_t data) {
    if (locked) {
      return;
    }
    else {
      radio_sense_msg_t* rsm;
```

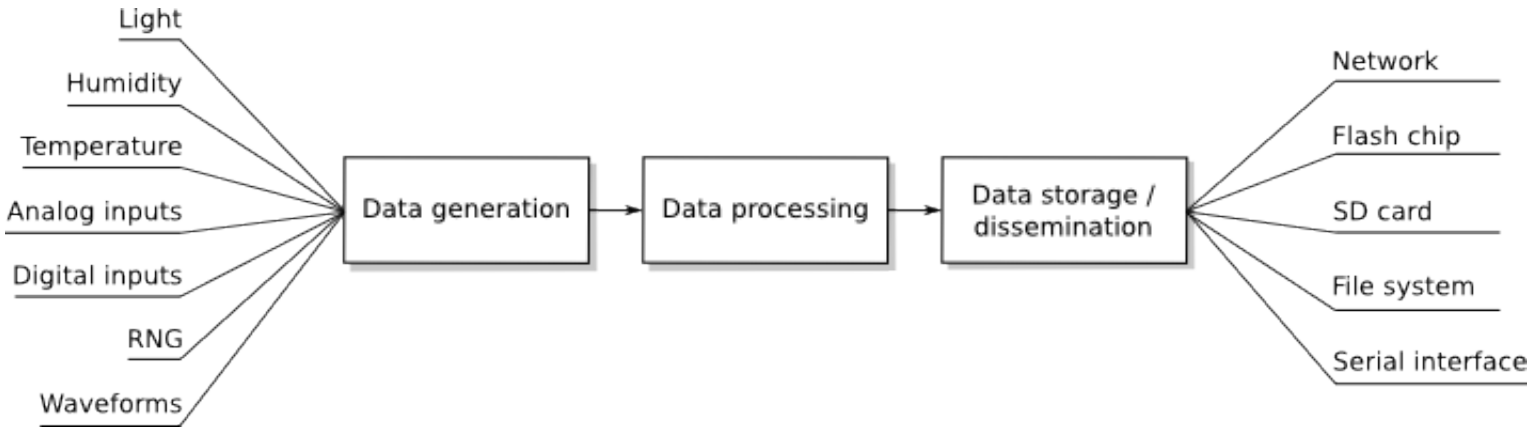
- C kods:

```
emacs23@desktop <2>
File  Edit  Options  Buffers  Tools  C++  Help
#include "stdmansos.h"
#include <hil/alarm.h>
#include <string.h>

#define SENSOR_READ_INTERVAL 3000 // milliseconds

// our timer
Alarm_t timer;

//-----------------------------------------
//      Timer callback
//-----------------------------------------
void onTimer(void *param)
{
    // read light sensor value
    uint16_t light = readLight();
    PRINTF("light = %u\n", light);

    // send the value read to radio
    radioSend(&light, sizeof(light));

    // blink LED
    toggleRedLed();
}

//-----------------------------------------
//      Entry point for the application
//-----------------------------------------
void appMain(void)
{
    // initialize and schedule the alarm
    alarmInitAndRegister(&timer, onTimer, SENSOR_READ_INTERVAL, tru
    // set radio packet receive callback
    radioSetReceiveHandle(onRadioInterrupt);
    // turn radio listening on
    radioOn();
    // our job here is complete
    return;
}
```

- SEAL kods:

```
emacs23@desktop <2>
File  Edit  Options  Buffers  Tools  C++  Help
read Light, period 2s;
read Humidity, period 2s;
output Radio;
```

Datu plūsma BST motē

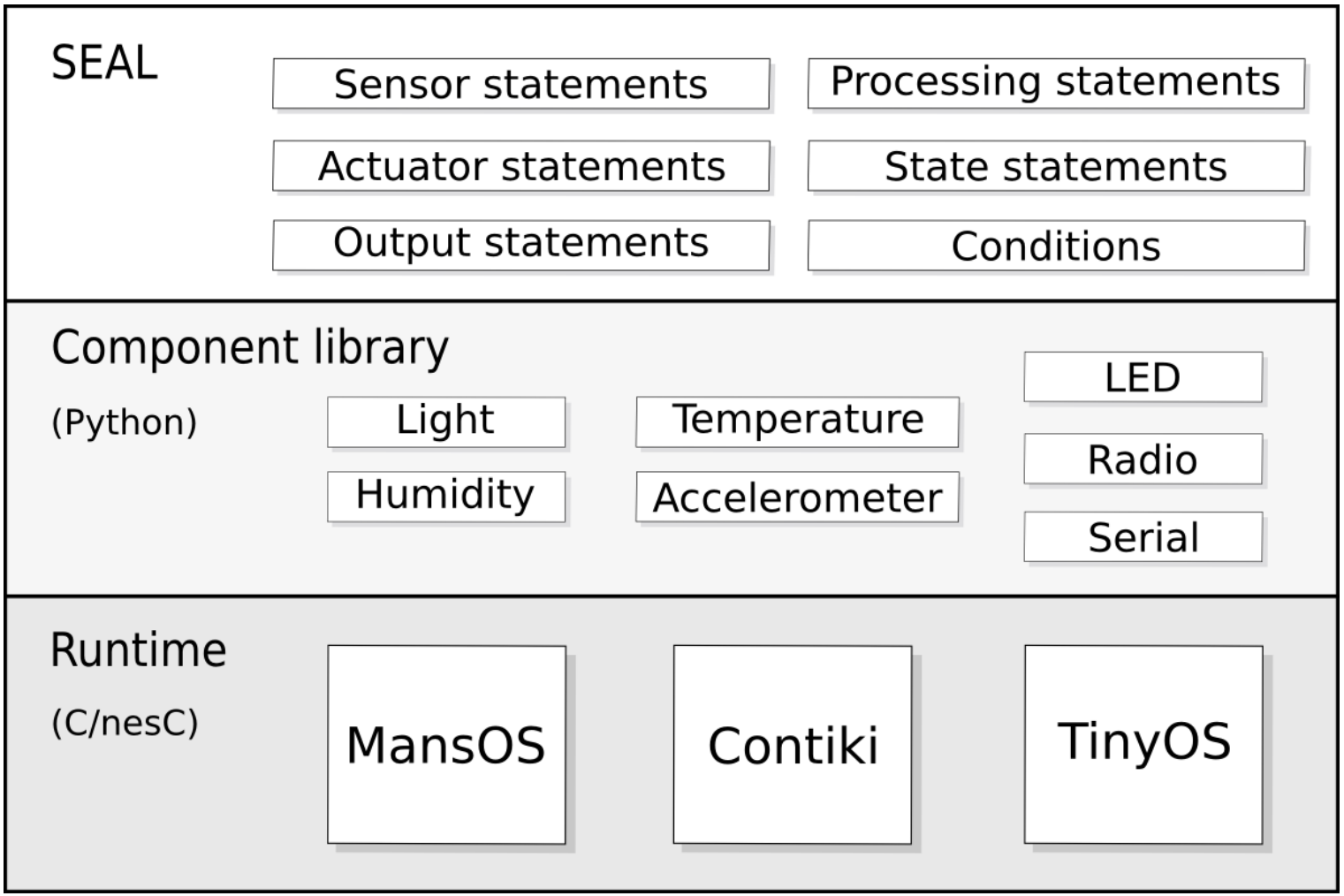


- Datus **ievāc** ar sensoriem un *pseidosensoriem*
- Datus **apstrādā** programmas loģika
- Datus **izvada** tīklā, atmiņā, seriālajā saskarnē u.c.

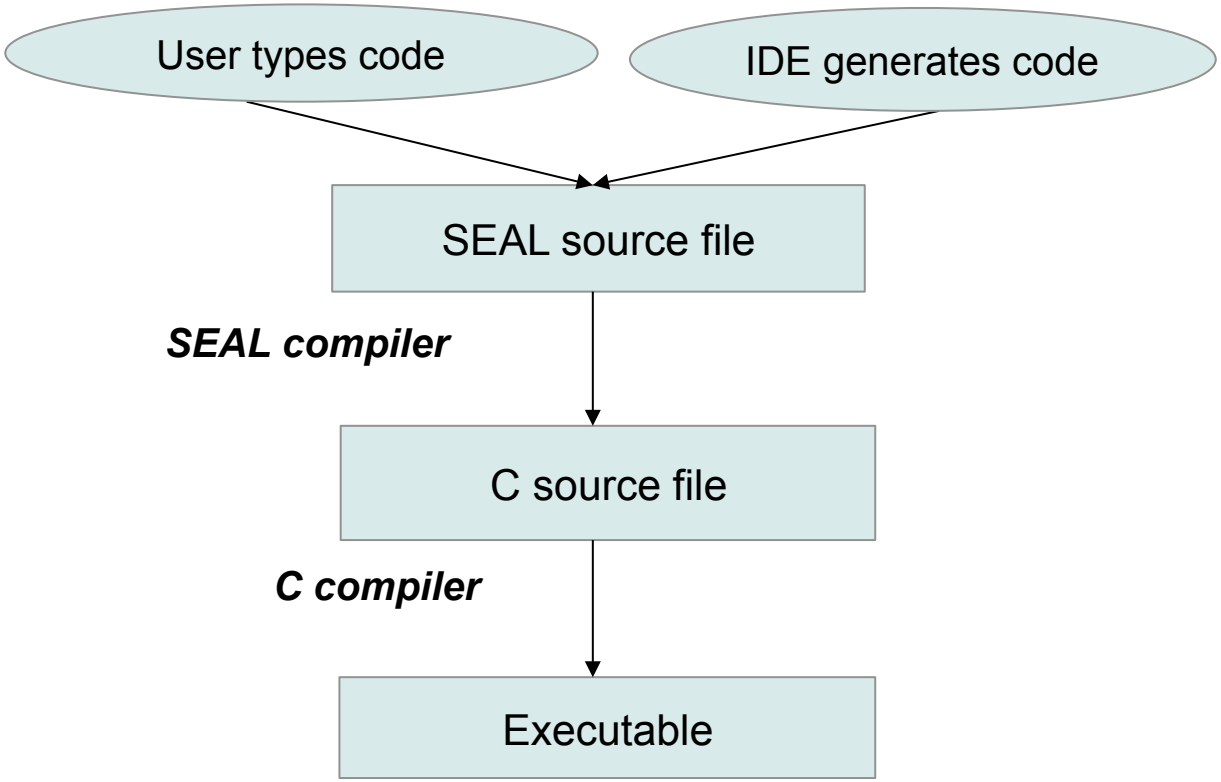
SEAL arhitektūra

- Deklaratīva ar dažiem imperatīviem elementiem
- Kompakta un dabīgi lasāma sintakse
- Divi līmeņi: sintakse & komponentu bibliotēka, kā arī “runtime support” no OS

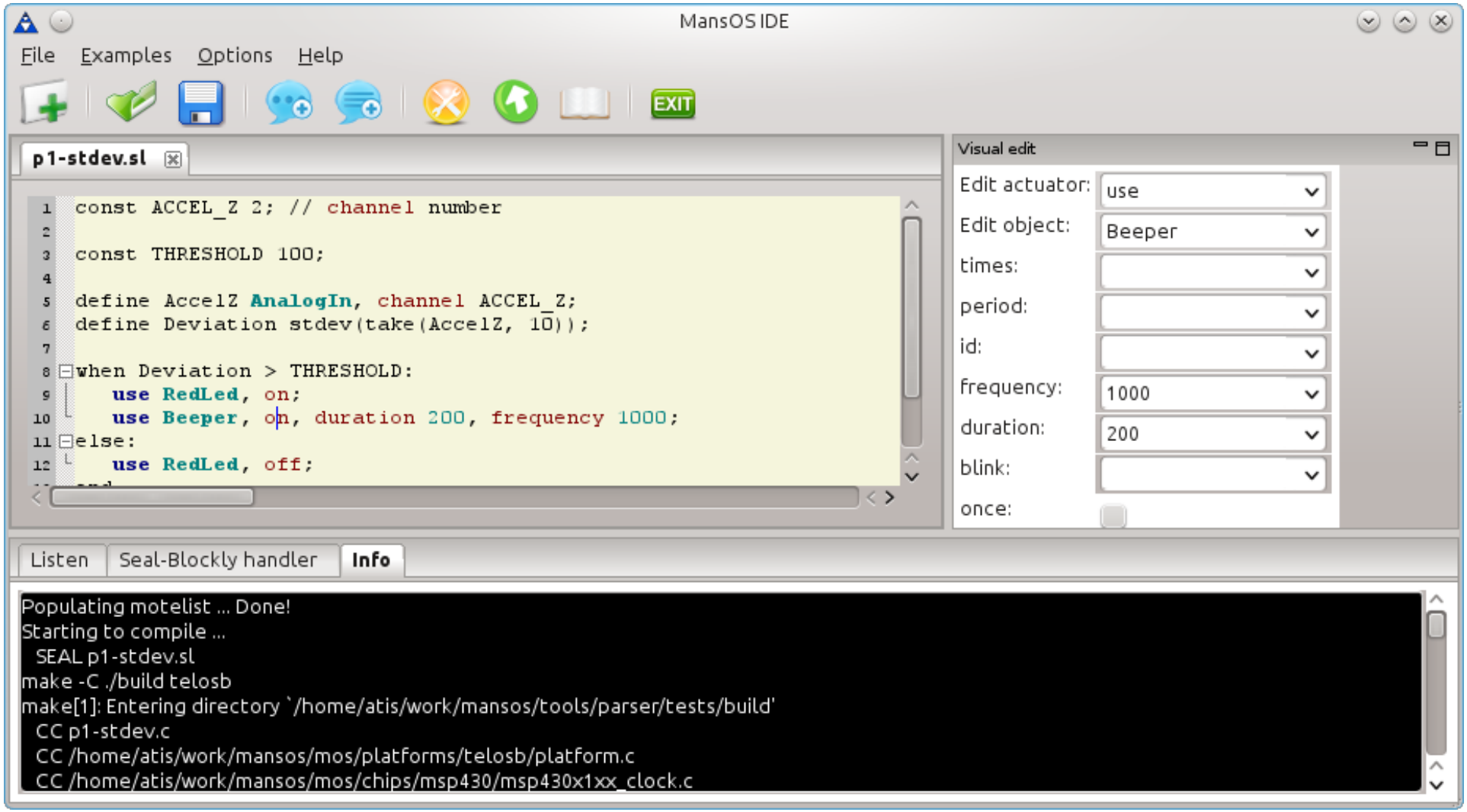
SEAL konceptuālā arhitektūra



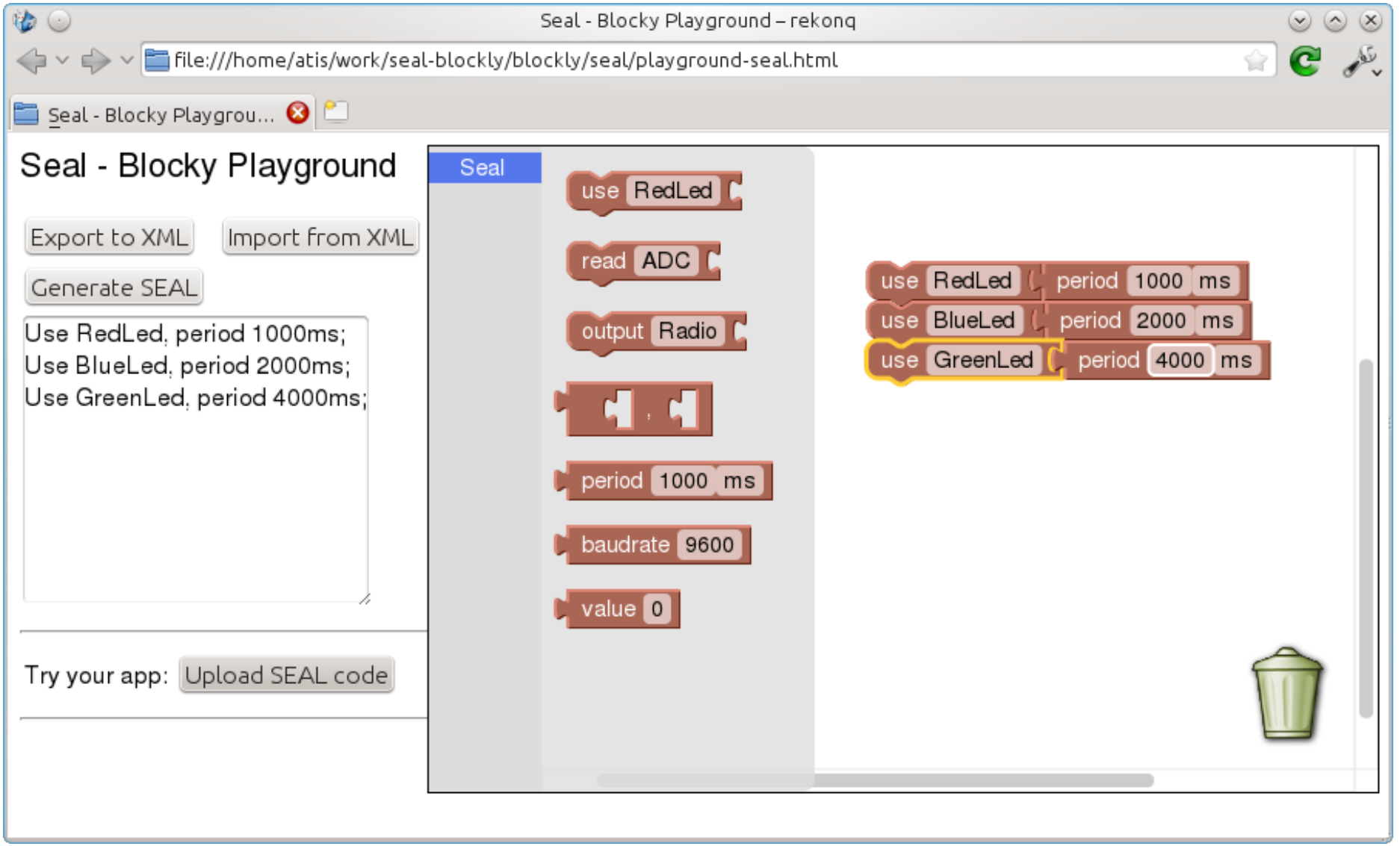
SEAL izstrādes process



SEAL grafiskā vide



SEAL programmēšana no tīmekļa



SEAL programmas piemērs

- Ko dara šī programma?

```
read Temperature, period 10s;
output Network;
when Temperature > 40C:
    use RedLed, on;
end
```

8. eseja

Tēma: atrodiet un aprakstiet vismaz divas BST operētājsistēmas un to atšķirības. Aprakstiet, kādiem lietojumiem jūs tās lietotu.

Termiņš: 11.11.2015. 10:00