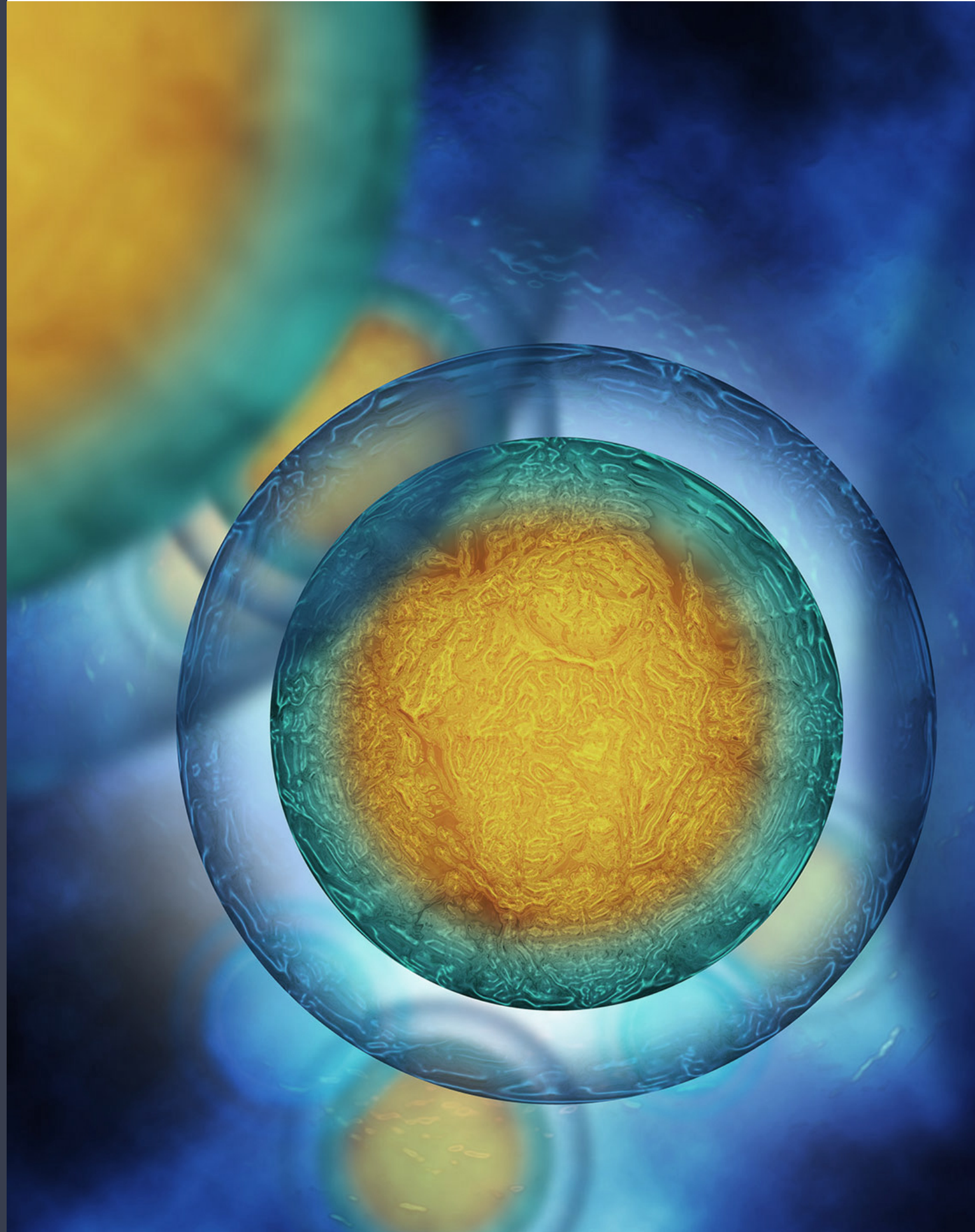
FE3CWS

MATERIÁL KU ŠKOLENIU V NIJMEGEN-E

Intelektuálny výstup č.3
ERASMUS+ projektu číslo 2017-1-SK01-KA203-035402



Niekoľko slov

O OBSAHU

- **Unikátne významná téma o kompozícii, zrozumitelnosti a korektnosti softvéru**
- **Dostupný v 7 jazykoch: anglický, maďarský, slovenský, chorvátsky, rumunský, bulharský a portugalský**

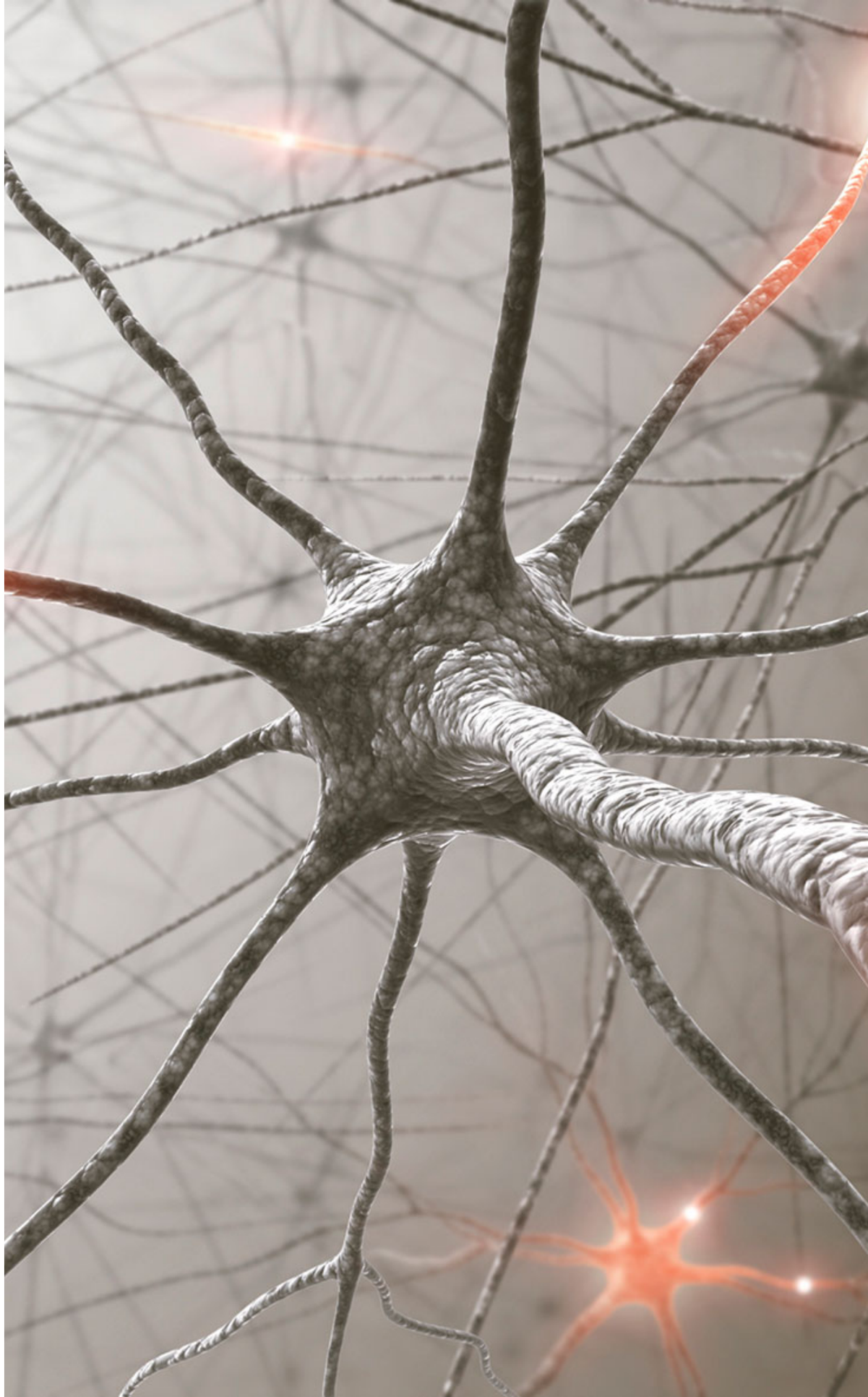
Co-funded by the Erasmus+ Programme of the European Union

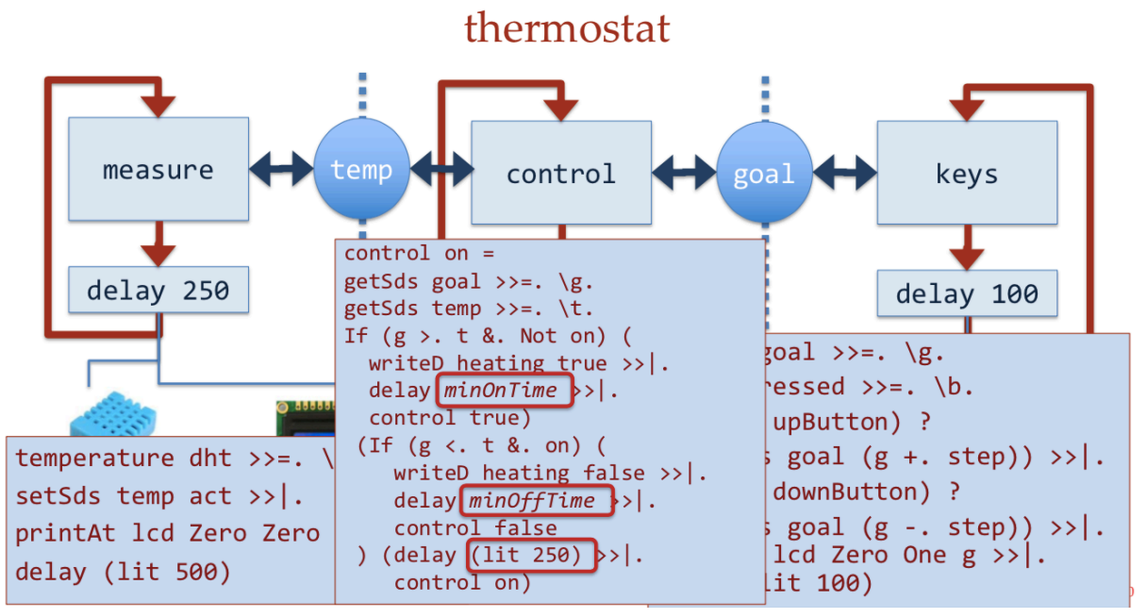
Second Teacher Training Material - "Functional Programming in the New Devices Lab": Funkcionálne programovanie v laboratóriu nových zariadení (interaktívne úlohy). Táto brožúra je tlačenou obdobou intelektuálneho výstupu.

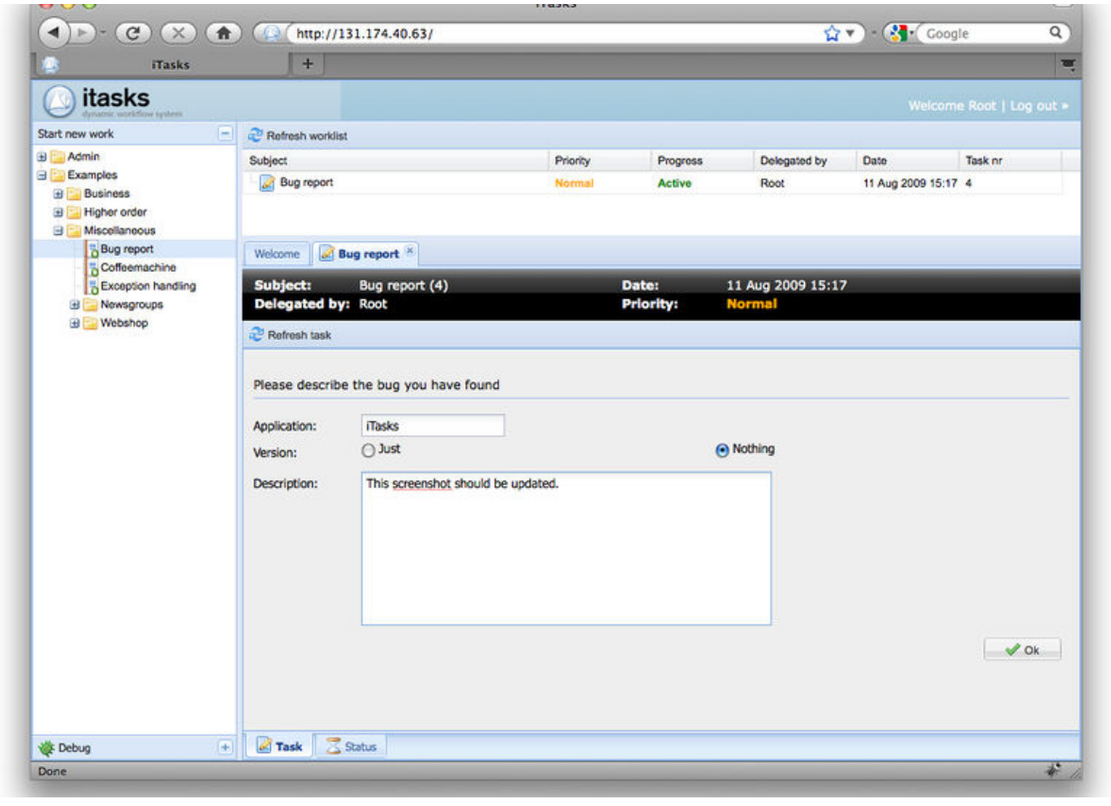
ZHRNUTIE

Vzdelávanie učiteľov v holandskom Nijmegene pokrýva päť dní intenzívneho vzdelávania a spolupráce na Radboud University Nijmegen. Cieľom tohto školenia učiteľov v rámci projektu FE3CWS bolo priniesť účastníkom úroveň súčasného stavu funkčného programovania a programovania podľa úloh, TOP. Témy, ktoré sa vyskytli, pokrývajú:

- Zhrnutie čisto funkcionálneho programovania v programe Clean. Pozrite si https://clean.cs.ru.nl/Clean.
- Generické programovanie: ako definovať manipulácie, ktoré fungujú pre akýkoľvek typ údajov prvého rádu, dokonca aj pre typy, ktoré ešte nie sú definované.
- TOP s cieľom špecifikovať spoluprácu ľudí a automatizovaných úloh na dosiahnutie cieľov.
- Systém iTask podporuje a monitoruje vykonávanie úloh. Pozrite si https://clean.cs.ru.nl/ITasks. Interakcia s používateľmi sa uskutočňuje hlavne prostredníctvom webových editorov, ktoré sú generované z typov používaných v definíciách úloh na vysokej úrovni.







Formalizmus TOP obsahuje malé množstvo flexibilných základných úloh (ako sú webové editory a kontrola periférnych zariadení internetu vecí) a sadu kombinátorov pre paralelné a sekvenčné zloženie úloh.

TOP na ovládanie internetu vecí, internet vecí.

TOP je veľmi vhodný na ovládanie internetu vecí a oslobodzuje vývojárov od záťaže mnohých programovacích jazykov, protokolov, rozhraní a ich interoperability. Na zvládnutie obmedzeného výpočtového výkonu a energetických obmedzení zariadení IoT sa na programovanie zariadení IoT vyrába špeciálny doménovo špecifický jazyk DSL, nazývaný mTasks. Systémy iTask a mTask hladko spolupracujú na prepojení sveta webových úloh s malými úlohami bežiacimi na IoT.

Výučba týchto tém sa uskutočnila kombináciou interaktívnych tutoriálov a praktického programovania s účastníkmi. Témy tohto vzdelávania učiteľov sú jadrom projektu. Stručné programy na vysokej úrovni abstrakcie priamo prispievajú k vlastnej zrozumiteľnosti.

Kombinátory úloh sú veľmi užitočným príkladom skladateľnosti v softvéri. Statický typ systému čisto funkčného hostiteľského jazyka Clean zaisťuje, že sa nemôžu vyskytnúť chyby typu runtime. Spolu so stručným zápisom v týchto DSL to prispieva k správnosti programov TOP.

ODKAZ NA STIAHNUTIE

https://fe3cws.kpi.fei.tuke.sk/O3SVK.html

BLIKAJÚCE LEDKY = AHOJ SVET

Arduino kód

```
void setup (){
        pinMode(D4, OUTPUT);
}
void loop(){
        digitalWrite(D4, HIGH);
        delay(500);
        digitalWrite(D4, LOW);
        delay(500);
}
```

mTask (Clean) kód

```
blink :: Main (MTask v ()) | mtask v
blink = { main=rpeat (
        writeD d4 (lit True)
>>|.  delay (lit 500)
>>|.  writeD d4 (lit False)
>>|.  delay (lit 500)
)}
```

DVA PRÍKLADY NA FAKTORIAL

```
factorial :: Int → Main (MTask v Int) | mtask v
factorial x =
   fun λfac=(λi →
      If (i ==. lit zero)
         (lit one)
         (i *. fac (i -. lit one))) In
   {main=rtrn (fac (lit i))}

//Tail call optimized factorial
factorial‘ :: Int → Main (MTask v Int) | mtask v
factorial‘ x =
   fun λfacacc=(λ(n,a) →
      If (n ==. lit zero)
         a
         (facacc (n -. lit one, n*.a))) In
   fun λfac=(λi →
      facacc (i, lit one)) In
   {main=rtrn (fac (lit i))}
```

BLIKAJÚCE LEDKY FUNKCIONALNYM SPÔSOBOM V MTASK

```
1  module blink
2
3  import StdEnv, iTasks
4
5  import Interpret
6  import Interpret.Device.TCP
7
8  Start :: *World → *World
9  Start w = doTasks main w
10
11 main :: Task Bool
12 main = enterDevice >>= λspec→withDevice spec
13    λdev→liftmTask blink dev -|| viewDevice dev
14 where
15    blink :: Main (MTask v Bool) | mtask v
16    blink
17       = fun λblink=(λx→
18                delay (lit 500)
19             >>|. writeD d4 x
20             >>=. blink o Not)
21       In {main=blink (lit True)}
```

INTERAKTÍVNE BLIKANIE

```
1  main :: Task Bool
2  main = enterDevice >>= λspec → withDevice spec
3     λdev → withShared 500 λdelayShare →
4             liftmTask (blink delayShare) dev
5        -|| updateSharedInformation "Interval" [updater] delayShare
6  where
7     updater :: UpdateOption Int Int
8     updater = UpdateUsing (λx → (x, x)) (const fst)
9        (panel2
10          (slider <<@ minAttr 5 <<@ maxAttr 10000)
11          (integerField <<@ enabledAttr False))
12
13    blink :: (Shared s Int) → Main (MTask v Bool) | mtask, liftsds v & RWShared s
14    blink delayShare = liftsds λdelaysh=delayShare
15       In fun λblink=(λx →
16                writeD d4 x
17          >>|. getSds delaysh
18          >>∼. delay
19          >>=. λ_ → blink (Not x))
20       In {main=blink (lit True)}
```

INTERAKTÍVNY PROGRAM MTASK NA INTERAKCIU S MATICOU LEDIEK

```
1  :: Ledstatus = {x :: Int, y :: Int, status :: Bool}
2  derive class iTask Ledstatus
3
4  main = enterDevice >>= λspec → withDevice spec
5     λdev→ viewDevice dev >^*
6        [OnAction (Action "Toggle") (always (
7           enterInformation () [] >>= λs → liftmTask (toggle s) dev
8              >>~ viewInformation "done" []))
9        ,OnAction (Action "Clear") (always (
10          liftmTask clear dev
11          >>~ viewInformation "done" []))
12       ] @! ()
13 where
14    dot lm s = LMDot lm (lit s.x) (lit s.y) (lit s.status)
15
16    toggle :: Ledstatus → Main (MTask v ()) | mtask, LEDMatrix v
17    toggle s = ledmatrix D5 D7 λlm→{main=dot lm s >>|. LMDisplay lm}
18
19    clear :: Main (MTask v ()) | mtask, LEDMatrix v
20    clear = ledmatrix D5 D7 λlm→{main=LMClear lm >>|. LMDisplay lm}
```

MERANIE TEPLOTY A VLHKOSTI

```
1 ||main = enterDevice >>= λspec→withDevice spec
2 ||   λdev→liftmTask temp dev >&> viewSharedInformation () [ViewAs templens]
3 ||where
4 ||   templens = maybe (0.0, 0.0) λ(t, h)→(toReal t / 10.0, toReal h / 10.0)
5 ||
6 ||   temp :: Main (MTask v (Int, Int)) | mtask, dht v
7 ||   temp = DHT D4 DHT22 λdht→{main=temperature dht .&&. humidity dht}
```

Literatúra

Peter Achten. Clean for Haskell98 Programmers. 13th July 2007.

Peter Achten, Pieter Koopman and Rinus Plasmeijer. 'An Introduction to Task Oriented Programming'. In: Central European Functional Programming School. Springer, 2015, pp. 187– 245.

Douglas Adams. The Hitchhiker's Guide to the Galaxy Omnibus: A Trilogy in Four Parts. Vol. 6. Pan Macmillan, 2017.

Matheus Amazonas Cabral De Andrade. 'Developing Real Life, Task Oriented Applications for the Internet of Things'. Master's Thesis. Nijmegen: Radboud University, 2018. 60 pp.

Tom Brus et al. 'Clean – a language for functional graph rewriting'. In: Conference on Functional Programming Languages and Computer Architecture. Springer, 1987, pp. 364– 384.

Jacques Carette, Oleg Kiselyov and Chung-Chieh Shan. 'Finally tagless, partially evaluated: Tagless staged interpreters for simpler typed languages'. In: Journal of Functional Programming 19.5 (Sept. 2009), p. 509. issn: 0956-7968, 1469-7653. doi: 10.1017/S0956796809007205.

James Cheney and Ralf Hinze. First-class phantom types. Cornell University, 2003.

Li Da Xu, Wu He and Shancang Li. 'Internet of things in industries: a survey'. In: Industrial Informatics, IEEE Transactions on 10.4 (2014), pp. 2233–2243.

L. M. G. Feijs. 'Multi-tasking and Arduino : why and how?' In: Design and semantics of form and movement. 8th International Conference on Design and Semantics of Form and Movement (DeSForM 2013). Ed. by L. L. Chen et al. Wuxi, China, 2013, pp. 119–127. isbn: 978-90-386-3462-3.

Patrick C. Hickey et al. 'Building embedded systems with embedded DSLs'. In: ACM Press, 2014, pp. 3–9. isbn: 978-1-4503-2873-9. doi: 10.1145/2628136.2628146.

Pieter Koopman, Mart Lubbers and Rinus Plasmeijer. 'A Task-Based DSL for Microcomputers'. In: Proceedings of the Real World Domain Specific Languages Workshop 2018 on - RWDSL2018. Vienna, Austria: ACM Press, 2018, pp. 1–11. isbn: 978-1-4503-6355-6. doi: 10.1145/3183895.3183902.

Mart Lubbers. 'Task Oriented Programming and the Internet of Things'. Master's Thesis. Nijmegen: Radboud University, 2017. 69 pp.

Mart Lubbers, Pieter Koopman and Rinus Plasmeijer. 'Multitasking on Microcontrollers using Task Oriented Programming'. In: 2019 42st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). COnference on COmposability, COmprehensibility and COrrectness of Working Software. Opatija, Croatia: IEEE, 2019, pp. 1842–1846.

Mart Lubbers, Pieter Koopman and Rinus Plasmeijer. 'Task Oriented Programming and the Internet of Things'. In: Proceedings of the 30th Symposium on the Implementation and Application of Functional Programming Languages. International Symposium on Implementation and Application of Functional Languages. Lowell, MA: ACM, 2018, p. 12. isbn: 978-1-4503-7143-8. doi: 10.1145/3310232.3310239.

Rinus Plasmeijer, Peter Achten and Pieter Koopman. 'iTasks: executable specifications of interactive work flow systems for the web'. In: ACM SIGPLAN Notices 42.9 (2007), pp. 141–152.

Rinus Plasmeijer and Pieter Koopman. 'A Shallow Embedded Type Safe Extendable DSL for the Arduino'. In: Trends in Functional Programming. Vol. 9547. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016. isbn: 978-3-319-39109-0 978-3-319-39110-6. doi: 10.1007/978-3-319-39110-6.

Camil Staps and Mart Lubbers. The Clean language search engine. 2017. url: https: //cloogle.org.

Jurrien Stutterheim, Peter Achten and Rinus Plasmeijer. 'Maintaining Separation of Concerns Through Task Oriented Software Development'. In: Trends in Functional Programming. Ed. by Meng Wang and Scott Owens. Vol. 10788. Cham: Springer International Publishing, 2018, pp. 19–38. isbn: 978-3-319-89718-9 978-3-319-89719-6. doi: 10.1007/978- 3-319-89719-6_2.