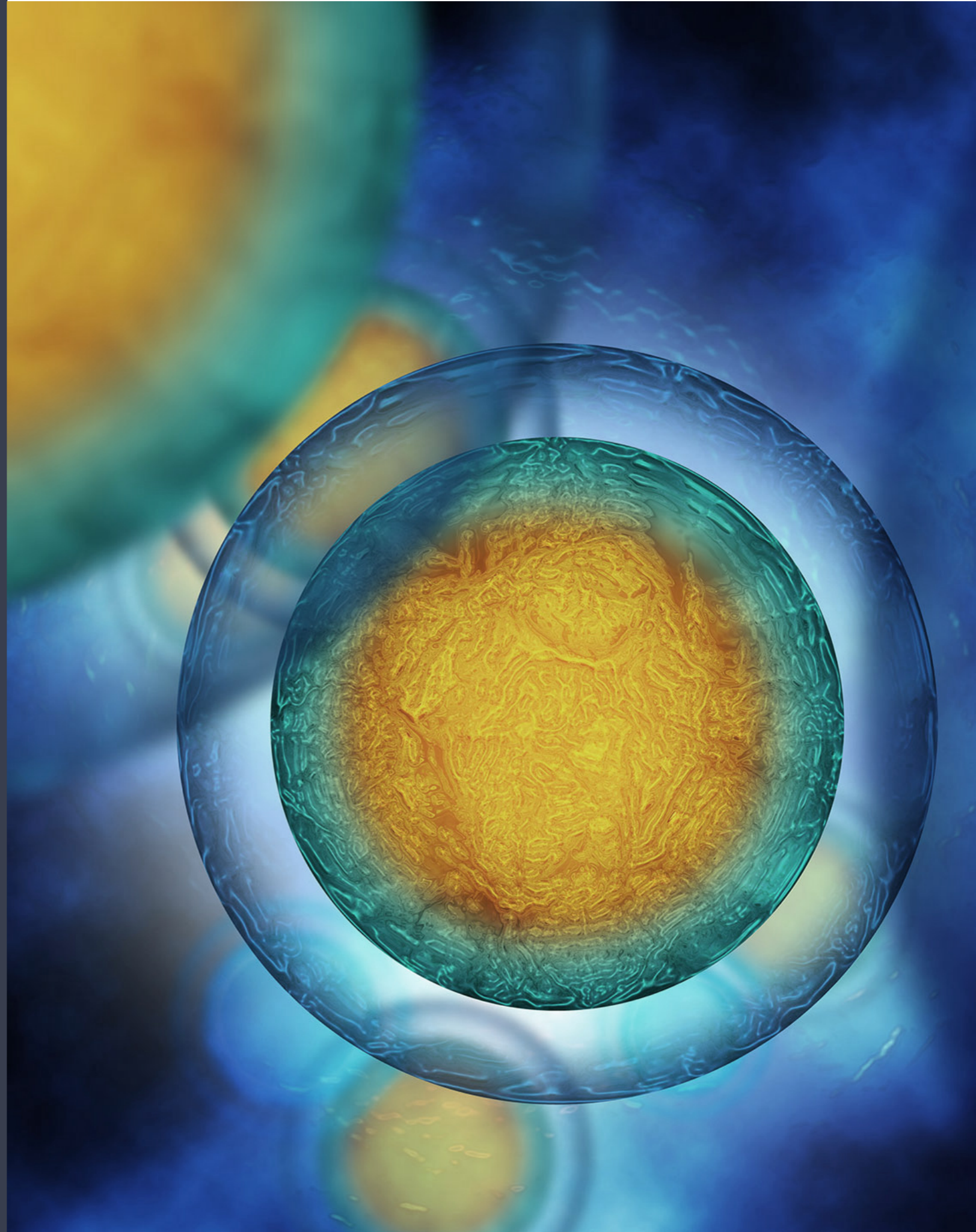


FE3CWS

ОБУЧЕНИЕ
НА
УЧИТЕЛИ
FE3CWS В
НАЙМЕГЕН

Интеллектуален резултат 3 на
проекта ERASMUS + 2017-1-
SK01- KA203-035402



Някои думи по

съдържанието

- Една единствена, но мощна тема, свързана със софтуерен състав, разбиране и коректност
- Предлага се на 7 езика: английски, унгарски, словашки, хърватски, румънски, български и португалски

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



Second Teacher Training Material - "Functional Programming in the New Devices Lab".

Втори материал за обучение на учители - „Функционално програмиране в лабораторията за нови устройства“.

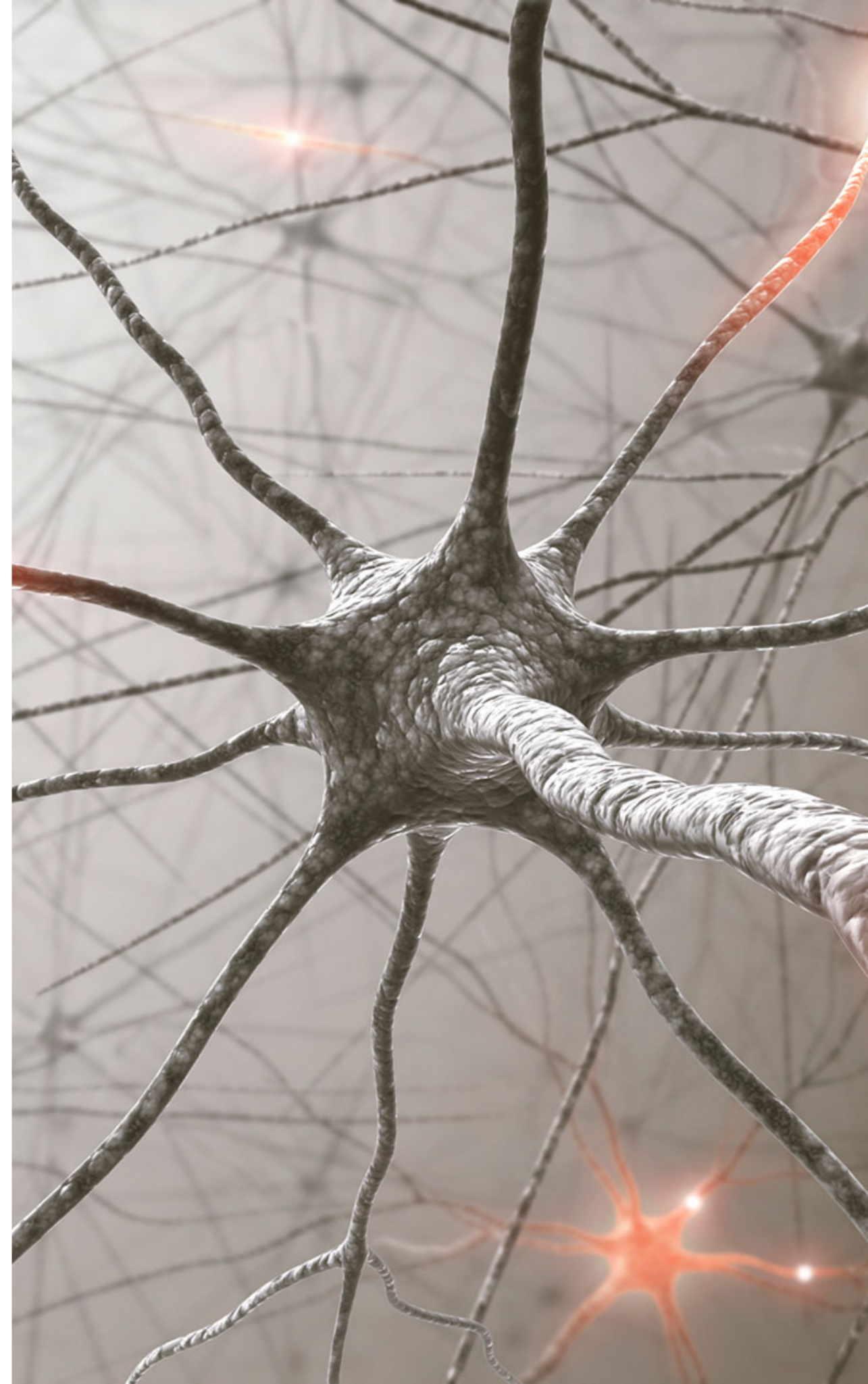
© Европейски съюз, 2017-2019

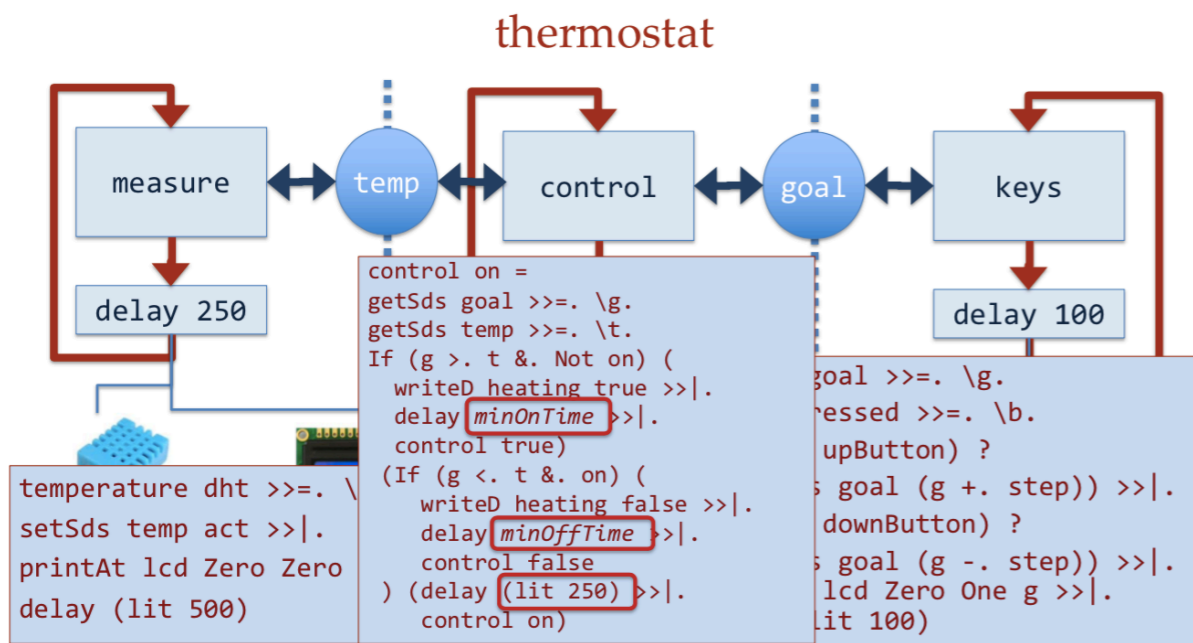
Информацията и възгледите, изложени в тази публикация, са тези на автора (ите) и не отразяват непременно официалното становище на Европейския съюз, институциите и органите на Европейския съюз, както и което и да е лице, действащо от тяхно име, не може да бъде отговорно за използването на съдържащата се тук информация.

ОТНОСНО

Обучението на учители FE3CWS обхваща интензивно петдневно обучение и сътрудна работа в Университета Радбуд в Неймеген, Холандия. Целите на обучението, в рамките на проекта FE3CWS, беше да обучи участниците, така че те да са конкурентно способни и наясно с новостите на функционалното програмиране и програмирането ориентирано към задачи (Task oriented programming (TOP)). Темите, които бяха обхванати са:

- Резюме на функционалното програмиране в Clean. Вижте <https://clean.cs.ru.nl/Clean>.
- Обобщено програмиране (Generic programming): как да се определят манипулациите, които работят за всеки тип данни, дори и за типове, които още не са дефинирани.
- TOP за конкретизиране на сътрудничеството между хората и автоматизирани задачи за постигане на целите. Системата iTask поддържа и следи за изпълнението на задачите. Вижте <https://clean.cs.ru.nl/ITasks>.

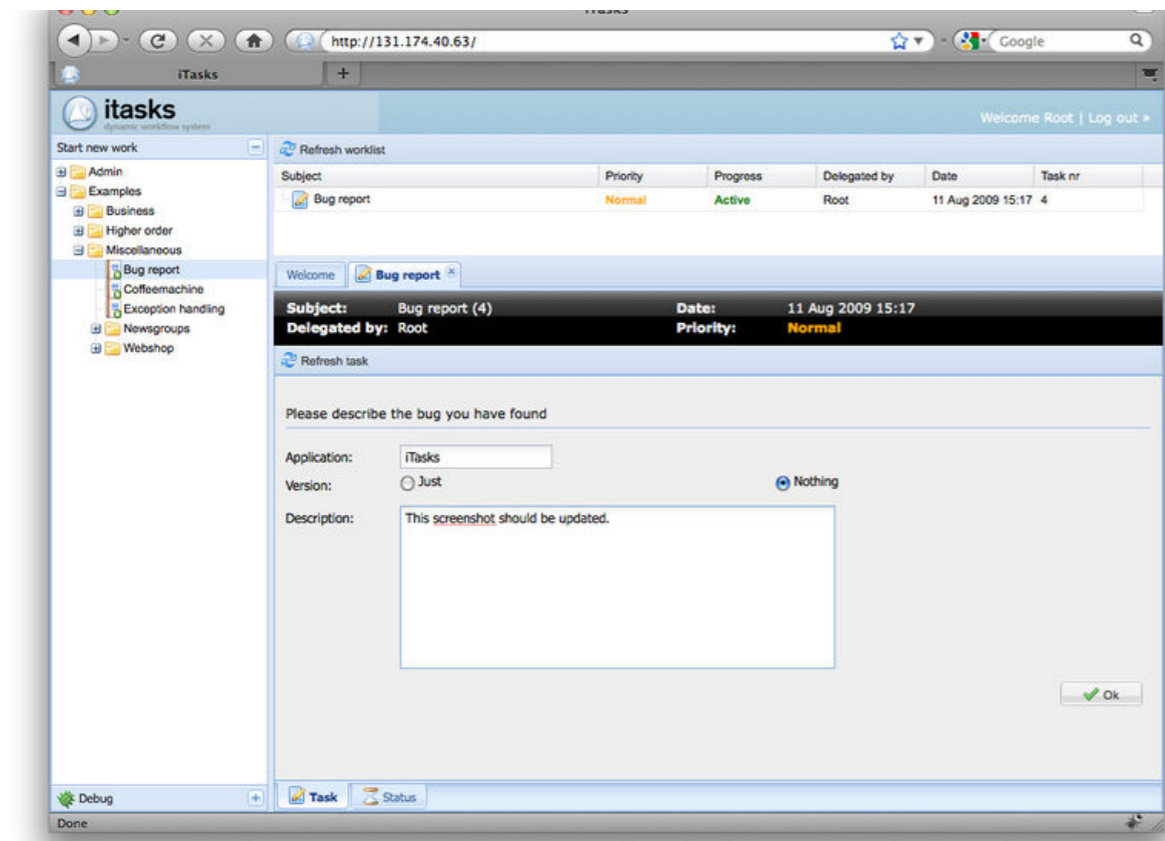




Взаимодействието с потребителите се осъществява главно чрез веб-базирани редактори, които се генерират от типовете, използвани в дефинициите на задачите от високо ниво. TOP формализмът съдържа малък брой гъвкави основни задачи (като веб-редактори и контрол на периферни устройства на IoT) и набор от комбинатори за паралелен и последователен състав на задачите.

TOP за контрол на IoT

TOP е много подходящ за управление на IoT и освобождава програмистите от използването на много програмни езици, протоколи, интерфейси и потенциални проблеми при оперативна съвместимост. За да се справи с ограничените изчислителни мощности и енергийни ограничения на IoT устройствата, се прави специален Domain Specific Language, DSL, наречен mTasks за програмиране на IoT устройствата. Системите iTask и mTask работят безпроблемно заедно, за да свържат света на веб-базирани задачи с малките задачи, изпълнявани на IoT.



Обучението на тези теми е комбинация от интерактивни уроци и практическо програмиране с участниците.

Темите на обучението на учители са в основата на проекта FE3CWS. Малките програми на високо ниво на абстракция пряко допринасят за възприемането им от обучаемите. Комбинаторите на задачи са много полезна илюстрация за начина, по който е построен софтуерът. Системата за статичен тип на чистия функционален език на хоста, Clean гарантира, че грешки от типа време за изпълнение не могат да възникнат. Кратката нотация в DSL езиците допринася за достоверността при TOP програмите.

ЛИНК ЗА ИЗТЕГЛЯНЕ

<https://fe3cws.kpi.fei.tuke.sk/O3BG.html>

Мигащи светодиоди = здравей свят

Arduino код

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

mTask (Clean) код

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

Два примера за FACTORIAL ИЗЧИСЛЕНИЕ

```
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))}
```

Мигащи светодиоди по функционалния начин в 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)}
```


Интерактивно мигане

```
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)}
```

Интерактивна програма MTASK за взаимодействие с светодиоди матрицата

```
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}
```

Измерване на температура и влажност

```
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}
```

Литература

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). COnterence 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.