Teaching Functional Patterns through Robotic Applications

J. Boender, E. Currie, M. Loomes, G. Primiero, F. Raimondi

School of Science and Technology
Middlesex University London

June 2, 2015

Overview

- The curriculum at Middlesex
- Racket
- MIRTO
- Using MIRTO for functional programming

Middlesex University

- Post-1992 university
- Very diverse student community
- Students tend not to do well in exams

CS at Middlesex

- Completely integrated first year
- Student-driven learning
- No formal assessment
- Assessment by SOBs
- 24 weeks, 120 credits

CS at Middlesex

Blocks

- Block 1: introduction
- Block 2: data structures and design
- Block 3: robots

Each block has an associated project

CS at Middlesex

Workshops

- Lecture: overview of the week
- Physical
- Programming
- Design
- Synoptic

Racket

Used to keep everything together:

- Teaching programming
- Implementing theoretical concepts
- Design
- Controlling physical devices

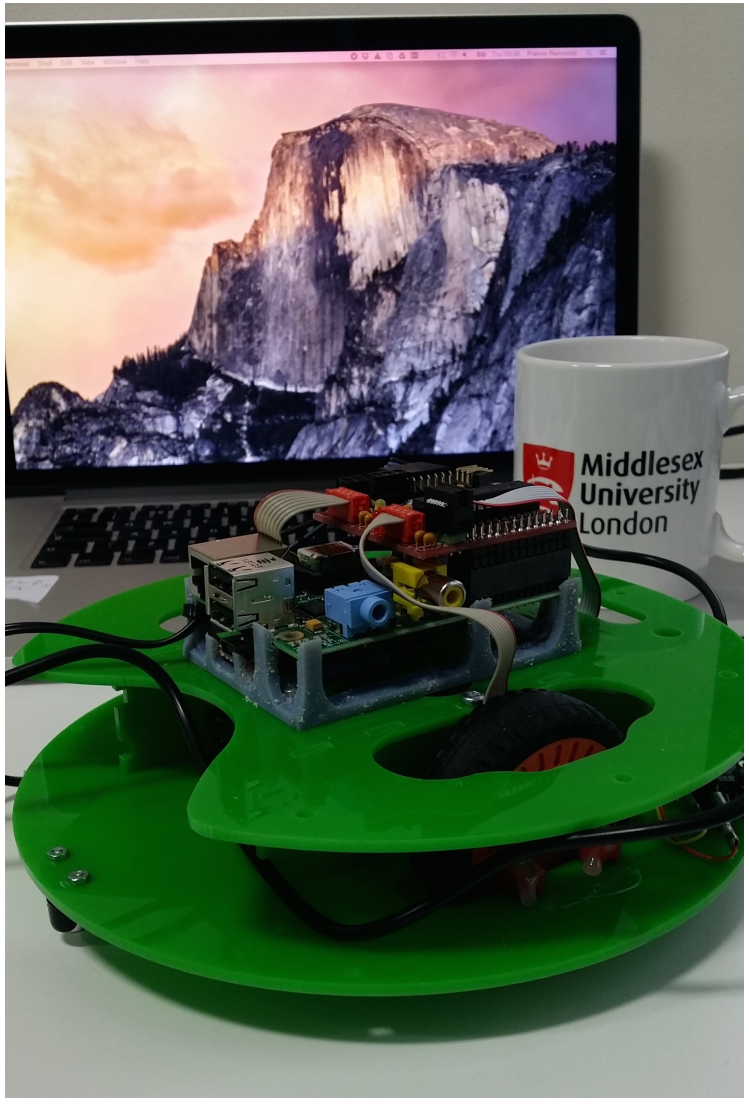
Why Racket?

- Functional language
- Flexibility
- Good available GUI
- Libraries aplenty
- *None of our students know it*

MIRTO

- MIddlesex Robotic plaTfOrm
- Arduino
- Raspberry Pi

MIRTO



MIRTO

How does it work?

- Raspberry Pi runs Linux+Racket
- Arduino runs custom-made firmware
- Communication through ASIP protocol

Functional programming

- Block 1: introduction
  - Lists
  - Functions
  - Recursion
- Block 2: data structures
  - Vectors, structs, ...
  - State
  - Design
- Block 3: MIRTO

Example 1

Random movement

```
(define moveLeft
 (lambda ()
 ;; code here to move left, using the
 ;; racket-asip library
 )
 )

(define moveRight ...)

(list-ref (list (moveLeft) (moveRight)) (random 2))
```

Example 1

Random movement

```
(define moveLeft
 (lambda ()
 ;; code here to move left, using the
 ;; racket-asip library
 )
 )

(define moveRight ...)

(list-ref (list moveLeft moveRight) (random 2))
```

Example 2

Higher order functions

```
(define (process-analog-values input))

(define analogValues (string-split (substring input
                          (+ (str-index-of input "{") 1)
                          (str-index-of input "}") ) ",") )

(map (lambda (x) (vector-set! ANALOG-IO-PINS
       (string->number (first (string-split x ":")))
       (string->number (second (string-split x ":")))
     ) ) ;; end of lambda
   analogValues) ;; end of map
 (printf "analog␣pins␣value:␣~a␣\n" ANALOG-IO-PINS)
  ) ;; end process-analog-values
```

Example 3

Higher order functions

```
(cond ( (> (- currentTime previousTime) interval)
  ;; We use map to print the value of each sensor
  (map (lambda (i) (printf "IR␣~a␣->␣~a;␣" i (getIR i)))
       irSensors)
  (printf "\n")
   (set! previousTime (current-inexact-milliseconds))
  )
)) ;; end of print IR
```

Example 4

Contracts

```
(provide (contract-out ;; Begin of contract
  [speed (and/c number? exact-nonnegative-integer?)]
  [added_speed (-> checkSpeed any)]
  [current_speed (-> number?)]
 ) ;; End of contract
)
```

Final projects

- Domino racing
- Eurobot competition
- Line following

Conclusion

- FP works for teaching
- Student engagement is up
- Allows excellent students to shine