A systems re-engineering case study

Programming robots with occam and Handel-C

Dan Slipper

Supervisors:

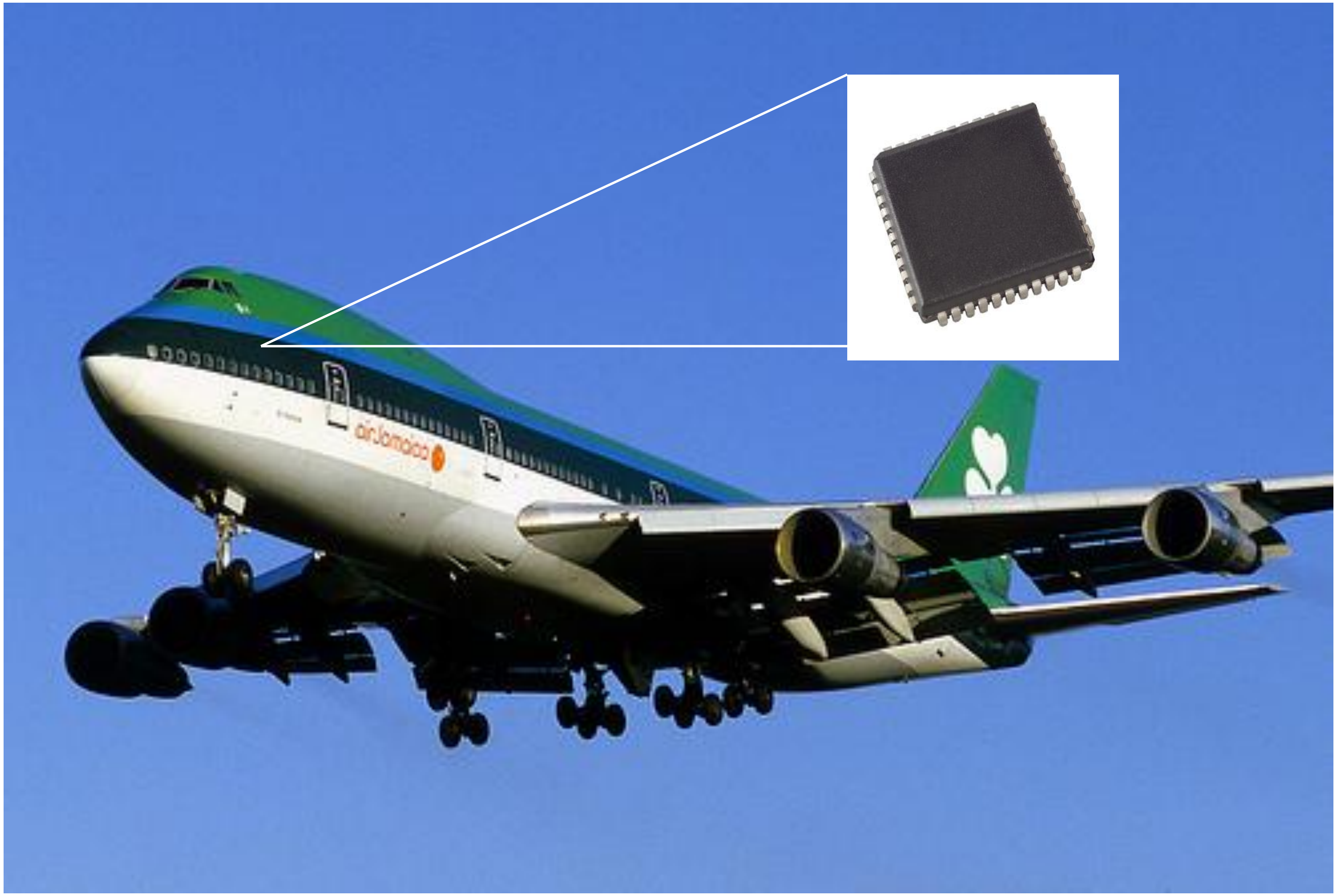
Alistair A. McEwan

University of Leicester

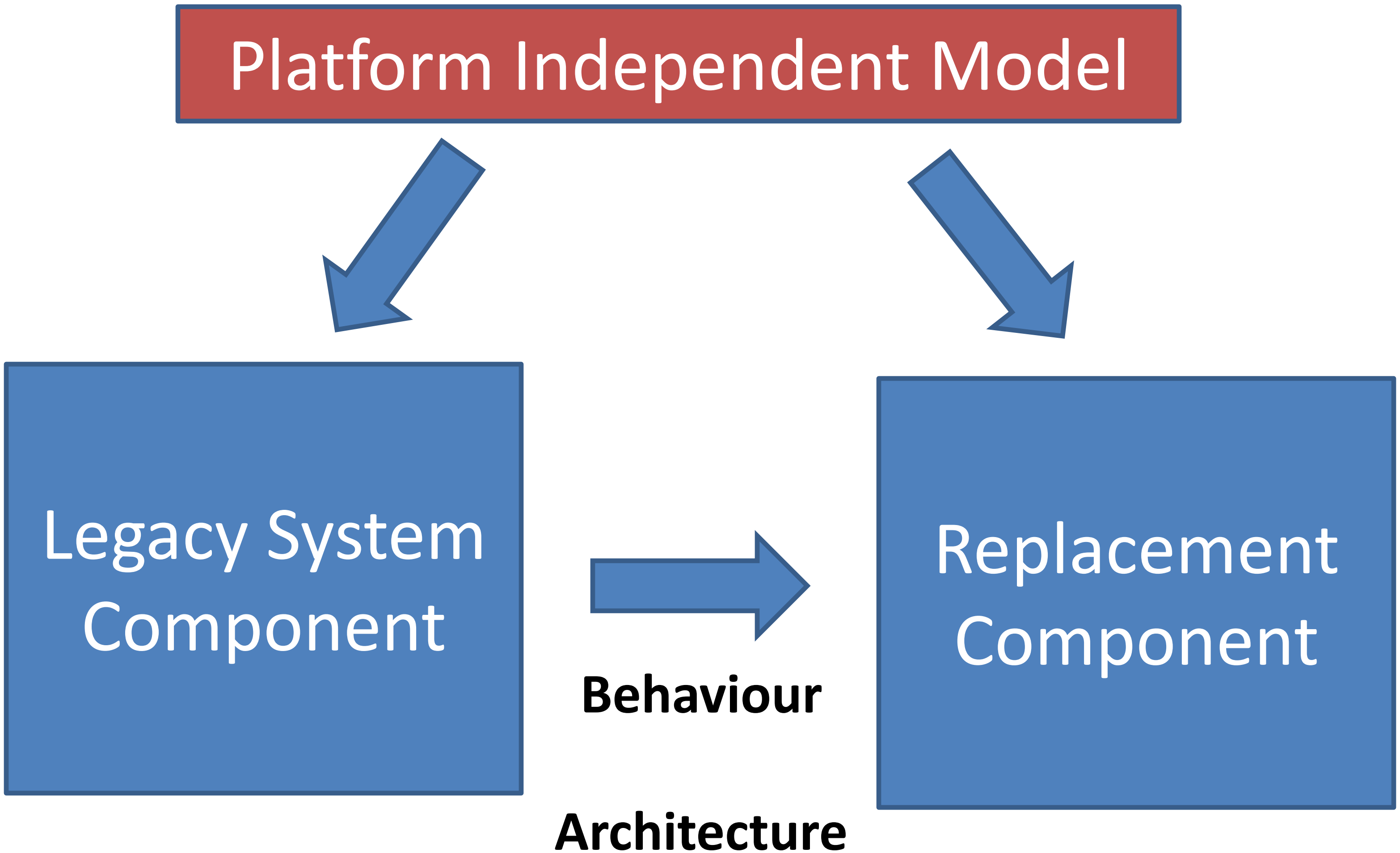
Wilson Ifill

AWE

The Problem



Re-engineering



Re-engineering

LEGO
MINDSTORMS
NXT

LEGO Mindstorms NXT

- 32-bit ARM7 microprocessor
- 4 input ports
- 3 output ports
- 100x64 pixel LCD
- Bluetooth
- Power, 6 AA Batteries

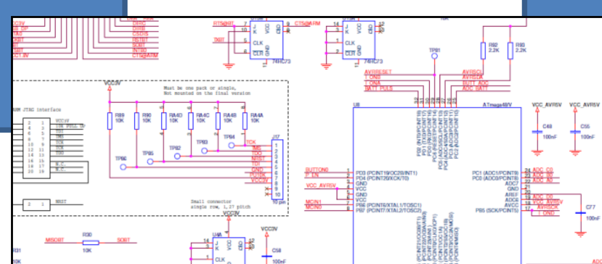
Case Study

CSP Program Structure

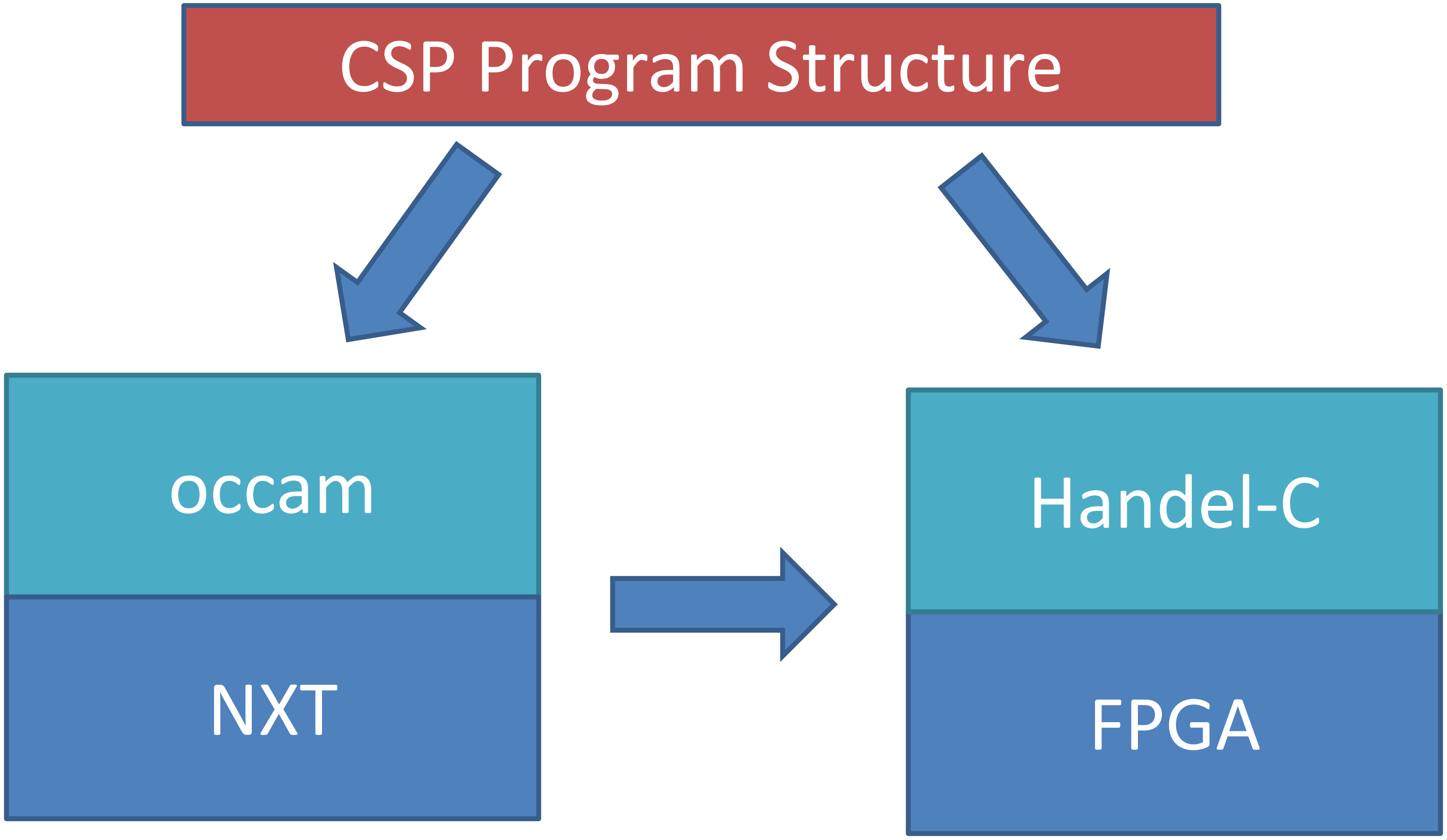
```
PAR
  motorASpeed ! 70
  motorBSpeed ! 70
reqLight ! 0
fromLight ? light
WHILE light > 800
  SEQ
    reqLight ! 0
    fromLight ? light
PAR
  motorASpeed ! 0
  motorBSpeed ! 0
do.delay(2000)
```

LEGO MINDSTORMS NXT

Replacement Hardware System



Case Study



Overview

- Problem space
- Project aims and background
- NXT/Transterpreter implementation
- FPGA implementation
- Case study
- Conclusions and future work

Aims

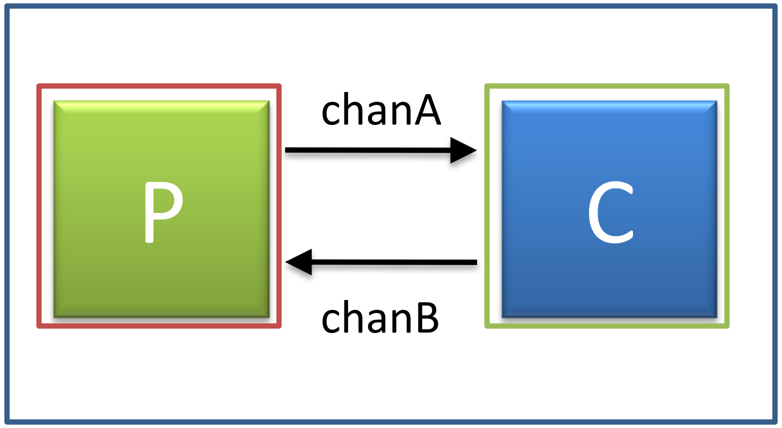
- Investigate the impact of modelling at a platform independent level
- Introduce two platforms using a common model of concurrency
  - Running a simple common task
- Demonstrate behavioural differences and integration issues between implementations

FPGAs and Handel-C

- FPGAs are "reconfigurable hardware"
- Can be reprogrammed any number of times unlike ASIC
- Handel-C is a HDL making FPGA hardware programming look like software
- C like language augmented with a CSP model of concurrency

Handel-C Example

```
static macro proc P();
static macro proc C();
void main (void)
{
     chan int 1 chanA;
     chan int 1 chanB;
    par{
          P(chanA, chanB);
          C(chanB, chanA);
    }
}
```

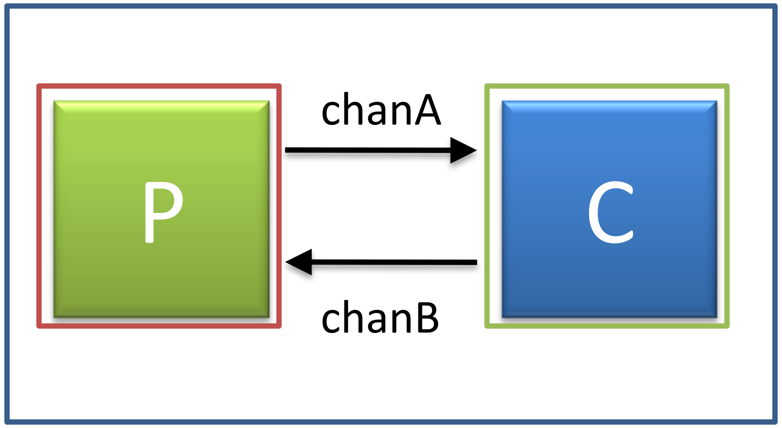


occam and the Transterpreter

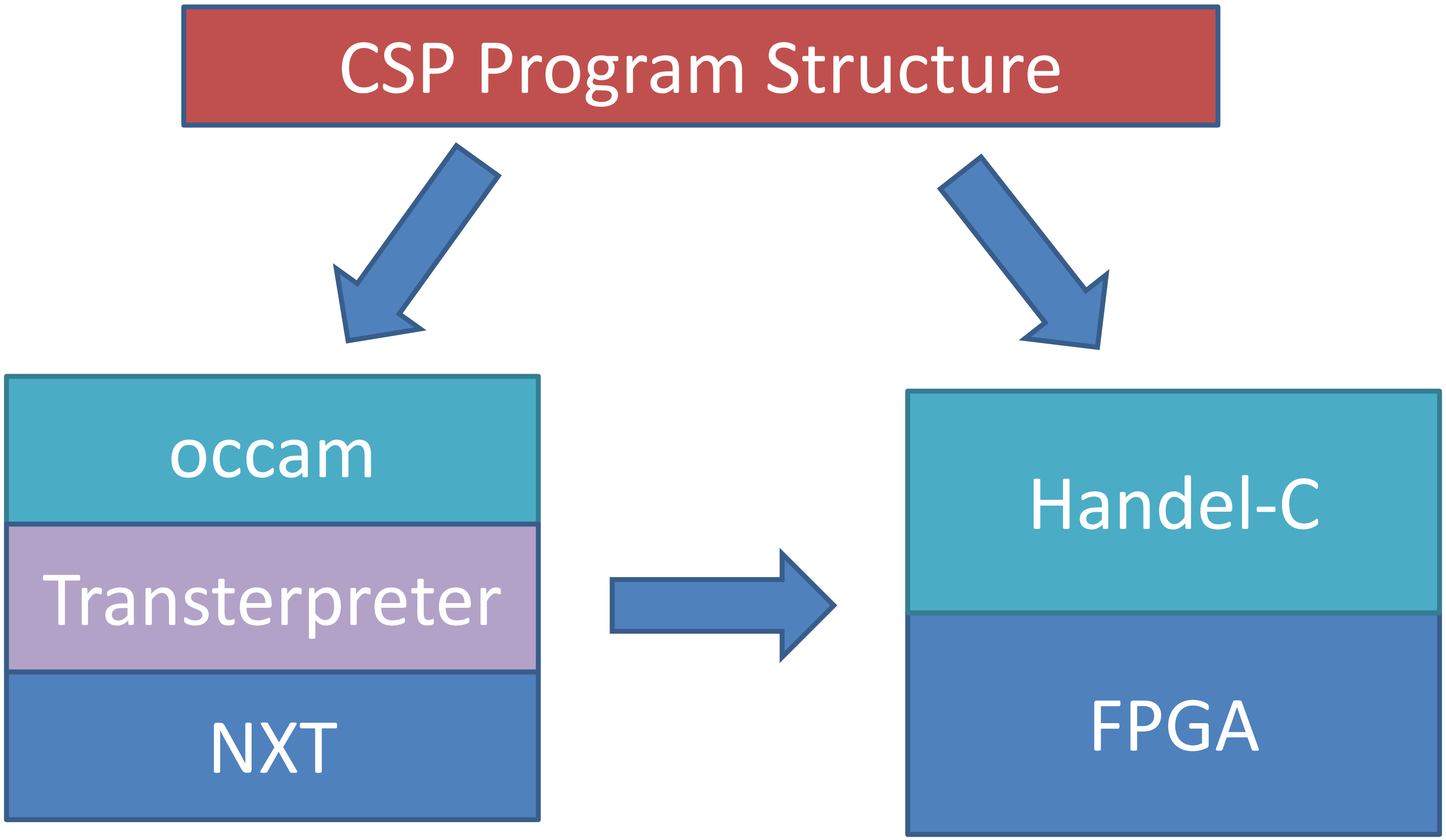
- The Transterpreter is a modern virtual machine for a variety of commodity platforms
- Interprets bytecode as on the Transputer
- Written as a C library

occam Example

```
PROC main ()

  CHAN INT chanA, chanB:

  PAR
    P(CHAN INT chanA, chanB)
    C(CHAN INT chanB, chanA)

:
```



Case Study

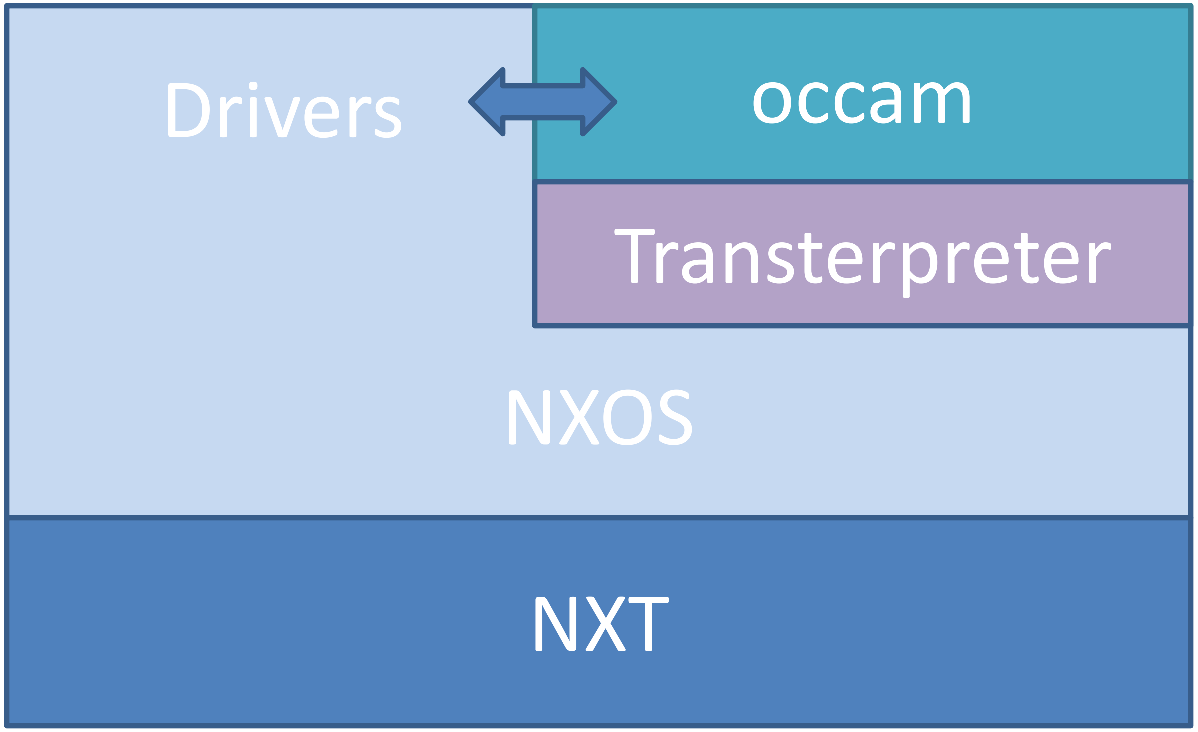


Legacy System Component

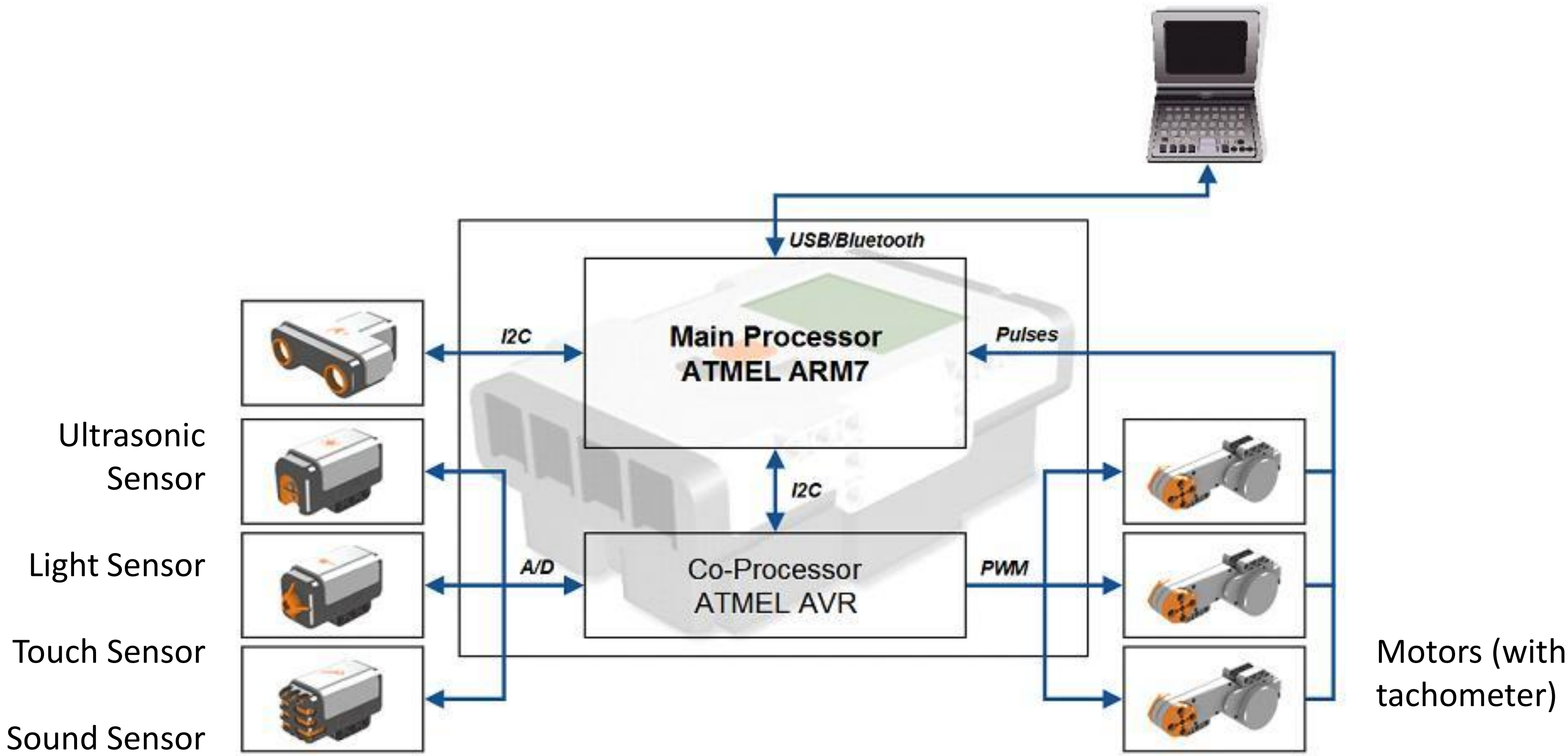
Replacement Component

NXOS + Transterpreter

- Built on top of existing OS
  - NXOS – set of C drivers and boot code



NXT Architecture

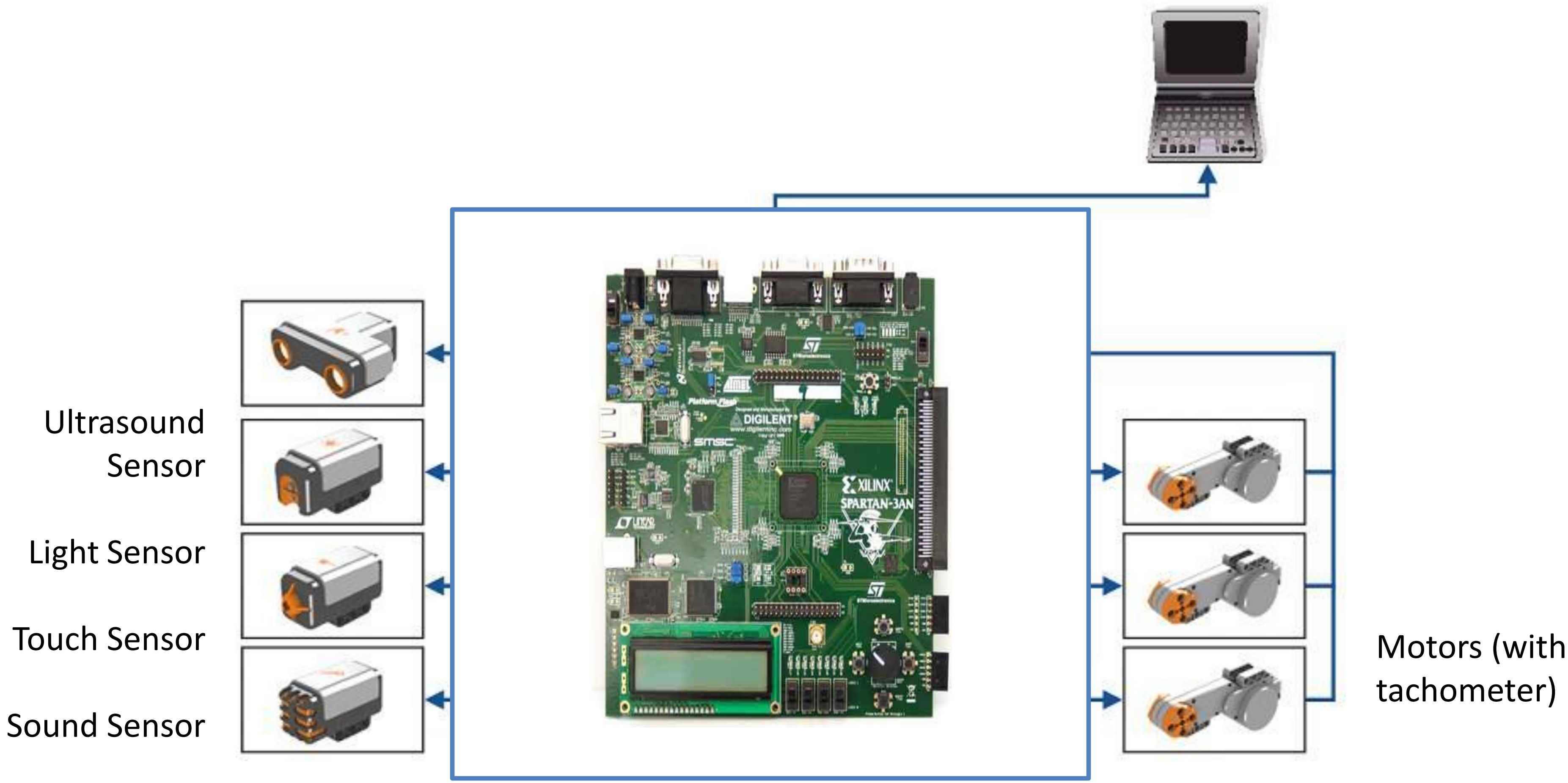


Peripheral Support in occam

- Standard peripherals have support
  - Analogue sensors
  - Motors
  - Ultrasonic sensor
  - Speaker
  - LCD
  - Bluetooth

- NXT 2.0
  - Colour sensor currently unsupported
  - 3rd party sensors partially supported – Using I^2C

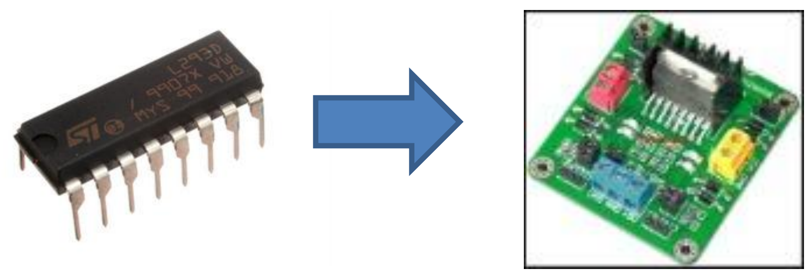
FPGA Architecture



FPGA features

- Light/Sound/Touch
  - Implemented through ADC

- Motors
  - PWM
  - Tachometer
  - H-Bridge driver board



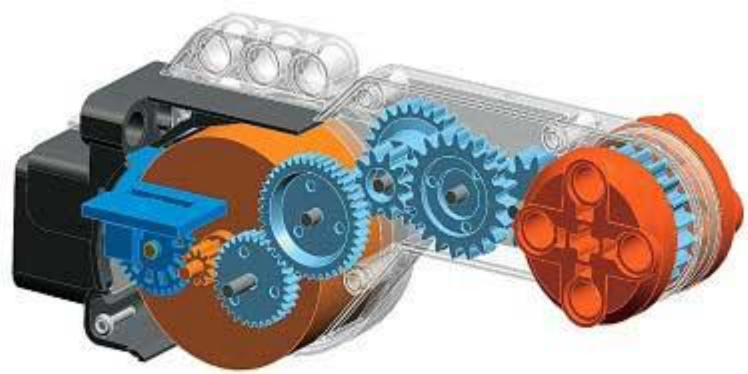
Case study

- Use implementations on FPGA and NXT
- Common program architecture
- Utilising a number of peripherals
- Simple case study
  - Follow a set path around the desk, changing direction with different speeds, (therefore angles)
  - Recognise the edge of the desk

Path to follow

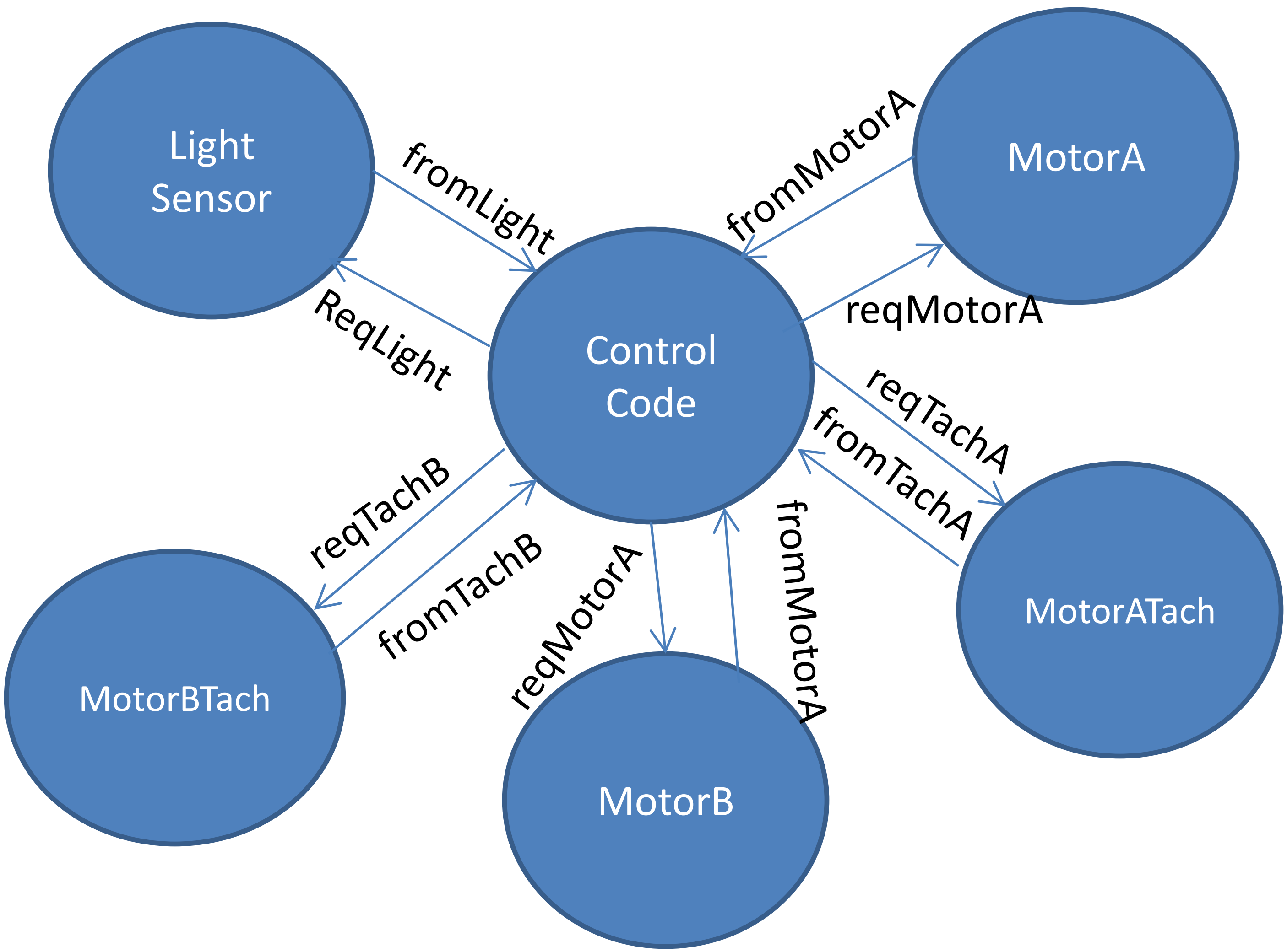
- The path the robots should follow...

Design of Experiment



Robot Structure

Process / Communication Structure



Main Implementation

- occam

```
PROC main ()
 CHAN INT motorASpeed,fromLight,
    reqLight,  fromTachA, reqTachA,
    reqTachB:
...
PAR
    LightSensor(fromLight, reqLight)
    Motor1(motorASpeed)
    Motor1Tach(fromTachA, reqTachA)
    ControlCode(motorASpeed,
    fromLight, reqLight, fromTachA,
    reqTachA)
:
```

- Handel-C

```
void main (void)
{
   chan int 32 toMotorA;
   chan int 32 motorATach;
   chan int 1 reqTachA;
  par{
     ADC_Read();
     LightSensor(fromLight, reqLight);
     Motor1(toMotorA);
     Motor1Tach(motorATach, reqTachA);
     ControlCode(fromLight, toMotorA,
     reqLight, reqTachA, motorATach);
   }
}
```

Control Implementation

occam

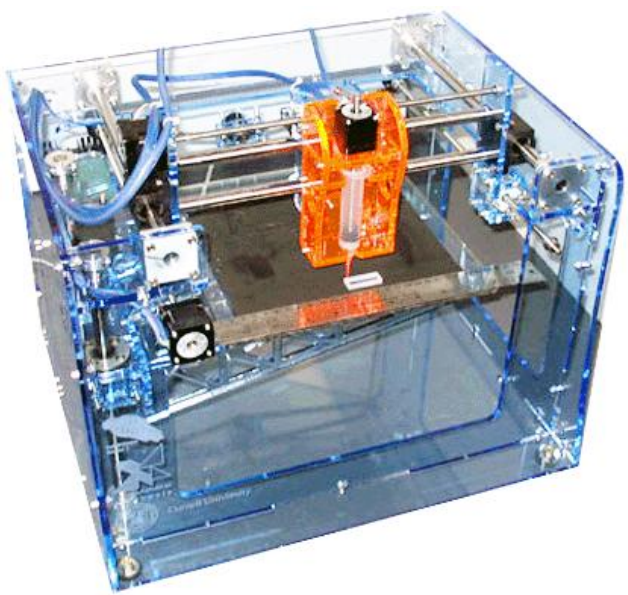
```
PAR
 motorASpeed ! 70
 motorBSpeed ! 70
reqLight ! 0
fromLight ? light
WHILE light > 800
 SEQ
  reqLight ! 0
  fromLight ? light
PAR
 motorASpeed ! 0
 motorBSpeed ! 0
```

Handel-C

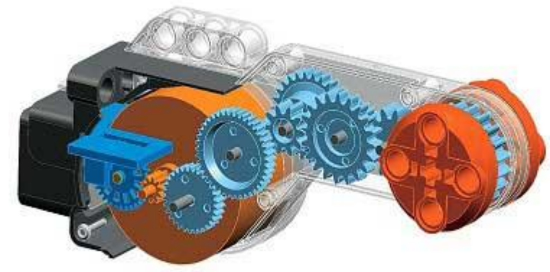
```
par{
    toMotorA ! 70;
    toMotorB ! 70;
}
reqLight ! 0;
fromLight ? light;
while(light == 0)
{
    reqLight ! 0;
    fromLight ? light;
}
par{
    toMotorA ! 0;
    toMotorB ! 0;
}
```

Test Results

- Path from occam implementation was a reference
- Handel-C path was wrong
  - Turning angles were different
  - Tachometer readings were therefore different
  - Overall system behaviour was incorrect



Further Tests



- Pulses after travelling 1000
  - Speed 100%

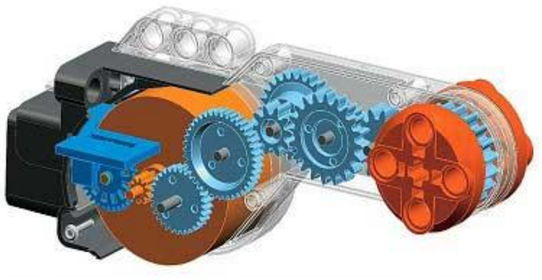
Robot	Run 1	Run 2	Run 3	Run 4	Run 5	Mean	Range	Standard Deviation
Handel-C	1247	1240	1239	1233	1232	1238	15	6.058052
occam	1773	1783	1793	1814	1787	1790	41	15.26434

  - Speed 80%

Robot	Run 1	Run 2	Run 3	Run 4	Run 5	Mean	Range	Standard Deviation
Handel-C	1213	1208	1209	1203	1206	1208	10	3.701351
occam	1397	1417	1431	1415	1413	1415	34	12.1161

Further Tests (2)

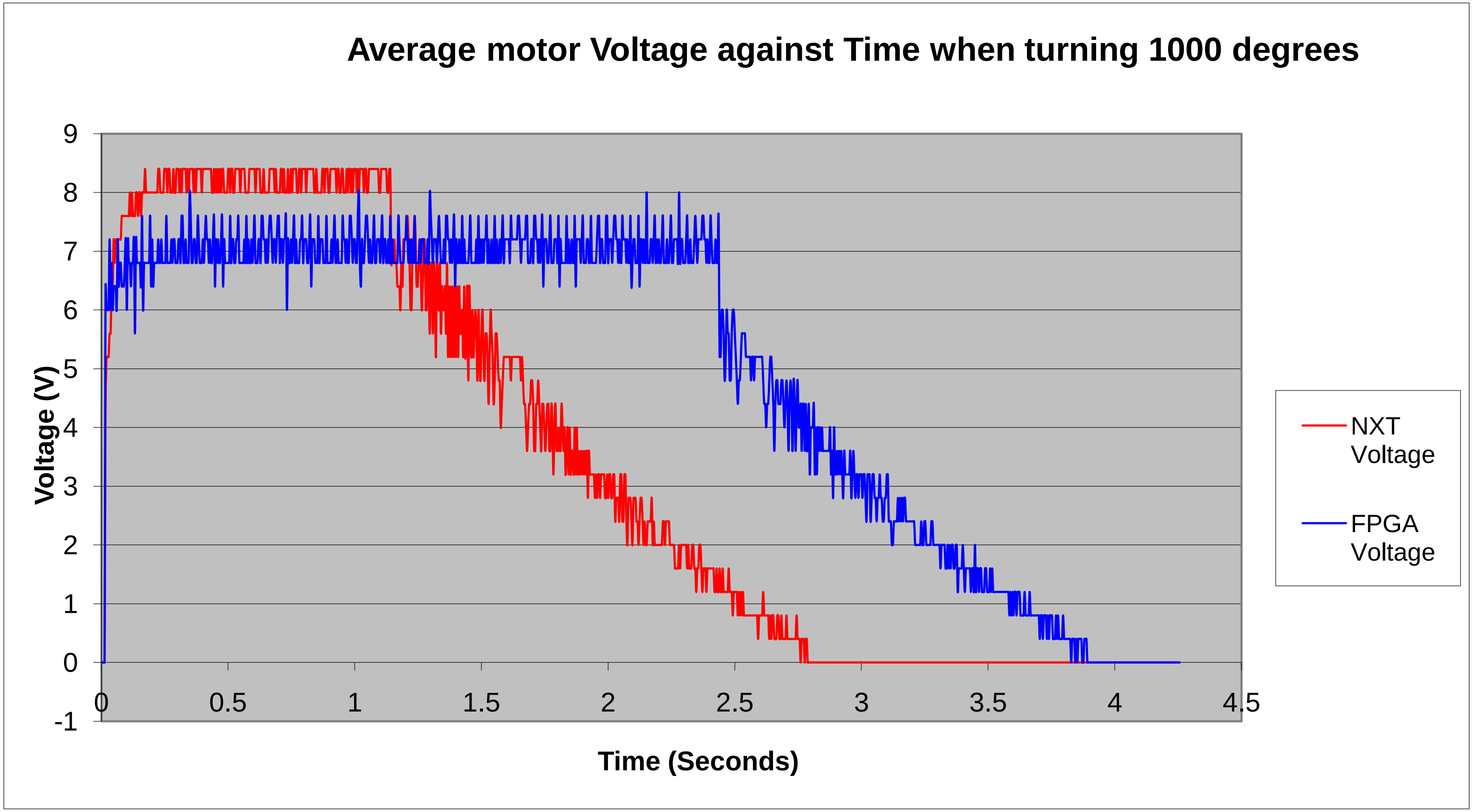
- Distance travelled (cm)
  - Speed 100%



Robot	Run 1	Run 2	Run 3	Run 4	Run 5	Mean	Range	Standard Deviation
Handel-C	73.8	71.3	68.6	69.6	68.6	70.38	5.2	2.207261
occam	70.4	70.4	71.0	70.6	71.2	70.72	0.80	0.363318

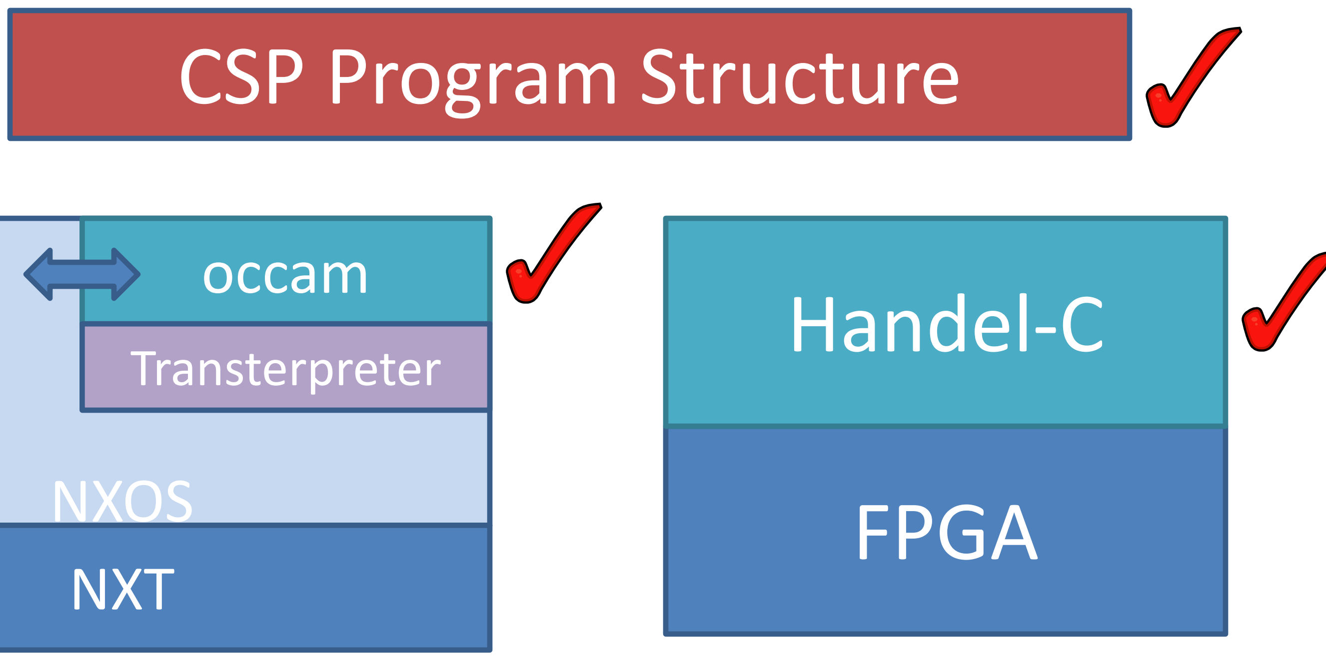
- Distances average the same over a vast range
- Investigation is required into circuit and differences in hardware

Further Tests (3)



Results

- Voltage difference between hardware implementations
  - Larger H-Bridge circuit
- Braking only tested with ‘float’method
  - void nx_motors_stop(U8 motor, bool brake)



Conclusions and Future Work

Conclusions

- Experiments demonstrate that programming same high-level missions leads to different behaviours.
- Therefore just modelling the high level behaviours is not reliable enough for a systems re-engineering project
- Modelling and verification methods are required for the whole system

Future Work

- CSP model of system behaviour
  - Translation to both implementation languages
- Improve motor drivers to braking methods
- More complex case study