Daisy Chain Protocol Development in Force-Guiding Particle Chains for Shape-Shifting Displays

Developmet Board Prototype

Raoul Rubien
rubienr@sbox.tugraz.at

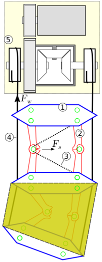
Institute for Technical Informatics
Graz University of Technology

6th February 2016

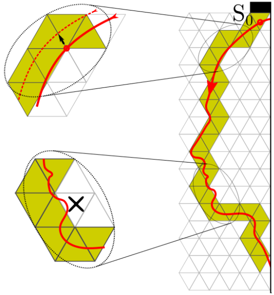
Underlying work

Force-Guiding particle Chains for Shape-Shifting Displays[1]
chain is locked in natural state

particle chain



folding a shape



shape Memory Alloy used as actuator (3)
joints unlocked by actuators (2)
force F_s contracts/folds chain

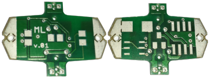
Limitation & approach

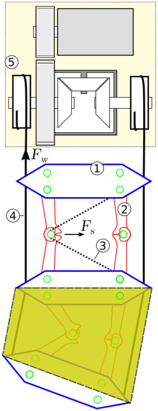
1-Wire current limits: **TODO: ask for I-details.**

too many actuations make 1-Wire communication infeasible

$\Rightarrow$ decouple communication from power supply

opted for daisy-chain protocol





Project extent

hardware development particle board
easy accessible test points and transmissions
flexible network assembly

Project constraints

reasonable low level MCU
minimize number of components on particle PCB
comm.: exploit SMA wires, decouple from power supply
small MCU package of final productive particle
all chain communication driven by one device

Netowork approach

linear network

daisy chained participants

Advantages

simple to implement

no media access

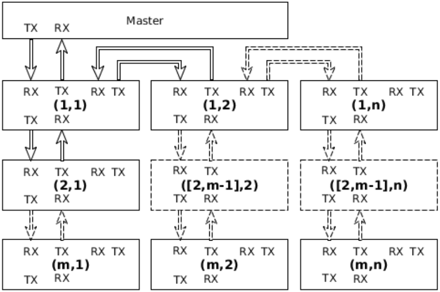
no loops

no dynamic routes

Disadvantages

faulty particle: detaches network part

faulty partice: no recovery

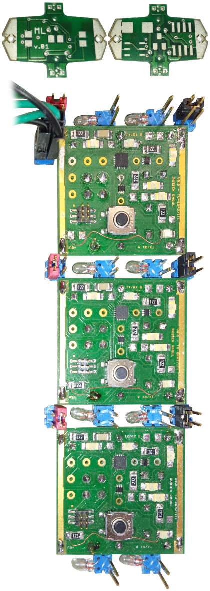


The prototype

- not compatible to the new design
- too small
- unhandy for measuring

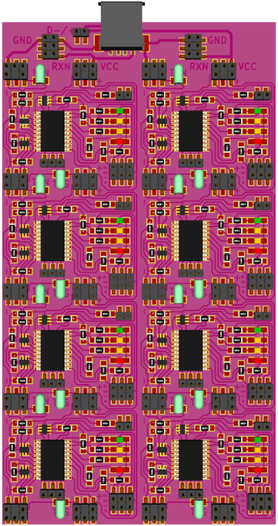
Version 1.0

- linear chain of development particles
- no mounting
- time consuming assembly
- $\Rightarrow$ new design



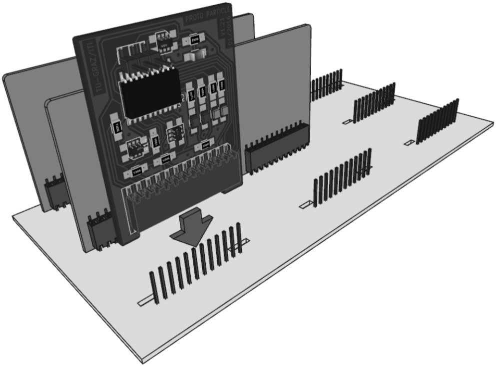
Version 1.1

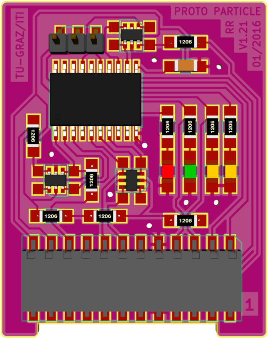
repetitive design
configurable network shape
costly soldering
expensive connectors
faulty particle must be repaired
$\Rightarrow$ new design



Version 1.21

configurable network dimension
easy extensible network
faulty particles are replaced
cheapest design
a more compressed design





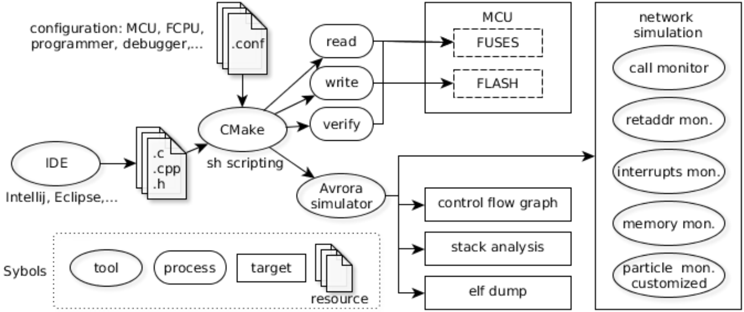
MCU selection

three separate ext. interrupt necessary
self programmable flash: firmware replication
small package for productive particle
optional bigger package for development board

	ATTiny20	**ATTiny1634**
# ext. INT	1	1
# PCI	12, PA[0:7], PB[0:3]	20, PA[0:7], PB[0:3], PC[0:5]
self. prog. flash	no	yes
flash	$2kB$	$16kB$
SRAM	$128B$	$1kB$
small package	yes ($3mm \times 3mm$)	no
alt. big package	yes($4mm \times 4mm$)	yes (SOIC)

Tool chain

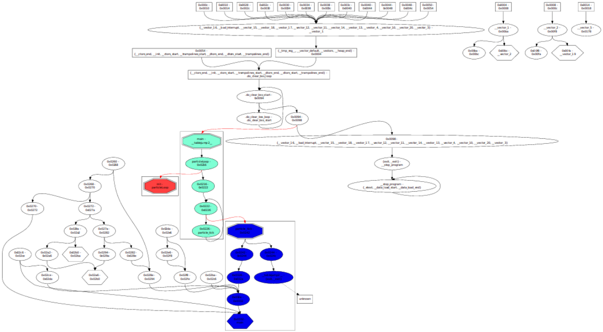
IDE independent
CMake
Simulation targets: CFG, stack analysis, network monitoring



Avrora[1] control flow graph

square: entered from interrupt
double octagon: procedures entry points
hexagonal: blocks end with a return

edges: jumps, branches, fall-throughs
red edges: calls
dotted: indirect calls or jumps



[1]http://compilers.cs.ucla.edu/avrora

Avrora simulation trace

very informative for proving

can be used to trigger other visualization/simulation frameworks

customize-able

```
0  0:00:00.00193878320  enable interrupts
1  0:00:00.00198403484  SRAM[SREG.(I | T | H | S | V | N | Z | C)] <- (0b00100001)
1  0:00:00.00198403484  SRAM[5f] <- 21 @ 14e
1  0:00:00.00198403484  disable interrupts
1  0:00:00.00198465904  main:particle_tick: @ 0x0154 <-(RETI)-- #20 0x004C
1  0:00:00.00198465904  enable interrupts
2  0:00:00.00194290985  SRAM[64] <- 0 @ 2a6
2  0:00:00.00194315919  SRAM[globalState.type] <- NODE_TYPE_TAIL (3)
2  0:00:00.00194315919  SRAM[63] <- 3 @ 2aa
2  0:00:00.00194390981  main: @ 0x02AE <-(RET )-- particle_tick
2  0:00:00.00194415915  main: @ 0x0222 --(CALL)-> particle_tick
2  0:00:00.00194415915  SRAM[45d] <- 13 @ 222
2  0:00:00.00194415915  SRAM[45c] <- 1 @ 222
2  0:00:00.00194503459  SRAM[particle_tick.loopCount] <- (-101)
2  0:00:00.00194503459  SRAM[60] <- 9b @ 22e
0  0:00:00.00194178328  SRAM[64] <- 0 @ 2a6
0  0:00:00.00194203342  SRAM[globalState.type] <- NODE_TYPE_HEAD (1)
```

Avrora profiling

customized memory profiling
interrupt profiling

```
=={ Particle state profiling results for node 1 }=============================
   Address       Writes       Changes
------------------------------------------------------------------------------
  particle_tick.loopCount  219/218
  globalState.state  5/4
  globalState.type  2/1
  globalState.nodeId  1/0
  globalState.northRxEvents  14/13
  globalState.southRxEvents  14/13
  globalState.flags( - | - | - | - | - | RECORD_RX_SOUTH | RECORD_RX_NORTH |)  1/0
  globalState.rxNorthByte1  1/0
  globalState.rxNorthByte2  1/0
  globalState.rxSouthByte1  1/0
  globalState.rxSouthByte2  1/0
  globalState.rxBitCounter  1/0
  dirD.(D7 | D6 | STH_RX | D4 | D3 | NRTH_RX | D1 | D0)  0/0
  portD.(D7 | D6 | STH_RX | D4 | D3 | NRTH_RX | D1 | D0)  1/1
  MCUCR.(SM2 | SE | SM1 | SM0 | ISC11 | ISC10 | ISC01 | ISC00)  1/1
  dirA.(TP | STH_SW | A5 | STH_TX | LED | A2 | NRTH_TX | NRTH_SW)  1/1
  portA.(TP | STH_SW | A5 | STH_TX | LED | A2 | NRTH_TX | NRTH_SW)  175/175
  GCIR.(INT1 | INT0 | INT2 | - | - | - | IVSEL | IVCE)  2/2
  SREG.(I | T | H | S | V | N | Z | C)  115/1
```

```
=={ Interrupt monitor results for node 0 }====================================
Num  Name         Invocations  Separation  Latency     Wakeup
-------------------------------------------------------------------------------
  1  RESET                  0
  2  INT0                   0
  3  INT1                  43  353.5476    10.232558   0.0
```

Future work

daisy chain communication protocol

- Phy., Data and Network Layer (OSI)

addressing

- self enumeration

time synchronization

- task scheduling (TDM)

coding

- exploit Manchester coding for network synchronization
- runtime calibration of internal RC-oscillator firmware

replication

- customize boot loader

This project was supported by the Institute for Technical Informatics of Graz University of Technology and Matteo Lasagni who has always been sincere and helpful and assisted this project.

M. Lasagni and K. Römer, "Force-guiding particle chains for shape-shifting displays," *CoRR*, vol. abs/1402.2507, 2014.