Daisy Chain Protocol Communication for Shape-Shifting Displays

Developmet Board Implementation

Raoul Rubien
rubienr@sbox.tugraz.at

Institute for Technical Informatics
Graz University of Technology

5th August 2016

Outline

Shape Shifting Display principle

- display consists of multiple fold-able chains
- chains can be programmed to make curvatures
- with multiple chains 3D shapes can be approximated

Goal: Scalable system with tiny chain links.

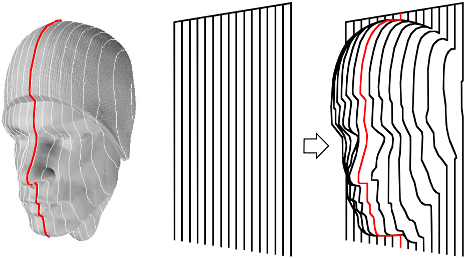


Figure 1: *display principle* [1]

[1] DOI 10.1145/2695664.2695932

Alignment

- nodes are aligned to a square lattice
- columns/chains are connected at the top
- each node has its own MCU
- communication is routed from node to node
- the layout requires nodes with three connections

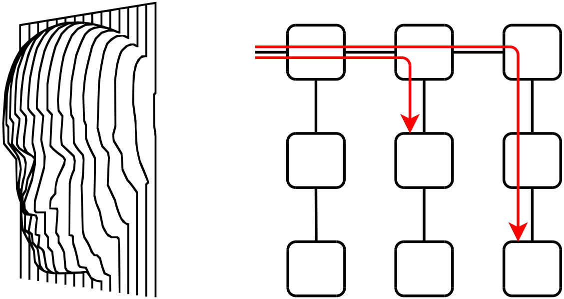


Figure 2: *node lattice*

Current implementation

- communication via power supply wires
- 1-Wire protocol
- due to lattice arrangement folding must be synchronized

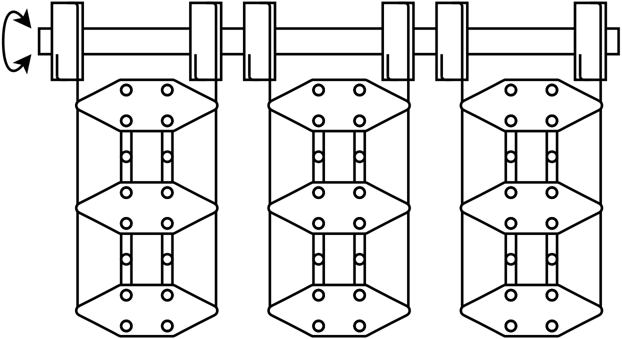


Figure 3: *particle chains*[2]

[2] http://arxiv.org/abs/1402.2507

Limitation & motivation

- power must be switched among levels of $0/+5/+12V$
- particle localization is costly (non scalable, brute force)
- 1-Wire communication is limited by power consumption
- communication is limited to one particle at once
- remote programming of MCUs not possible (i.e. firmware upgrade)

Idea

- decouple communication from power supply
- without extra wires (less error prone)

Conclusion

We need a daisy chain protocol that can execute synchronous folding commands using actuator wires.

Outline

Methodology

- choose suitable line coding
- select reasonable MCU
- particle board design
  - easy accessible test points and transmission wires
  - flexible and fast network assembly

Project constraints

- cheap and small MCU
- tiny particle
- communication:
  - exploit same actuator wires between consecutive particles
  - enable simultaneous communication and actuation
- single communication entry point to the network
- obtain a lightweight structure

Line coding

- opted for Manchester coding

Advantages

+ easy to implement

+ no extra clock wire for communication synchronization

+ time synchronization from communication clock

Disadvantages

- needs large decoding buffer

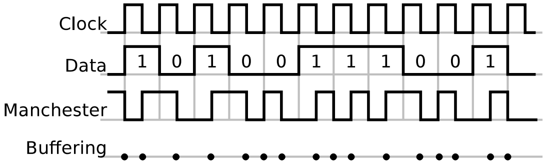


Figure 4: *Manchester coding: buffering events*

MCU memory requirements - upper bound estimation

Flash

- expected max. firmware size: $\sim 4k$ SLOC
- estimated object code bytes per SLOC[3] $\sim 4.0B$
- flash usage estimation:
  $4k * 4B =$ **$\sim$16kB**

SRAM

- tx/rx buffers: 3 ports, 8byte
  $3 * 8B * 2 =$ $16B$
- Manchester code decoding buffer
  with 2 flank time stamps per bit
  $3 * 8 * 2 * sizeof(uint16\_t)B * 0.75 =$ $576B$
- other global variables $200B$
- stack: max. 50 nested void function
  calls with $\sim (1 * uint8\_t)$ argument
  $50 * (1 + 2)B =$ $150B$
- SRAM estimation: **$\sim$950B**

[3] ISBN 0750686251

MCU requirements - capabilities

- three separate external interrupts
- self programmable EEPROM
  - remote programming (firmware replication)
- small MCU package

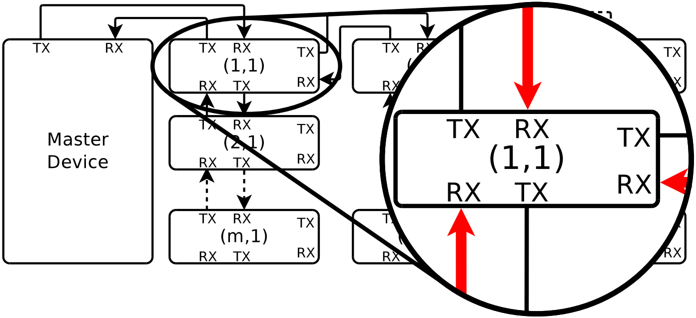


Figure 5: *external interrupt inputs*

Candidates

- candidates are ATTiny family MCUs having
  - $\geq 16kB$ flash and
  - $\geq 1kB$ SRAM

Comparison of used MCUs

	ATTiny20 (proof of concept)	**ATTiny1634**
# pin change int.	sufficient	sufficient
EEPROM	no	yes
flash	$2kB$	$16kB$
SRAM	$128B$	$1kB$
small package	$3mm \times 3mm$	$4mm \times 4mm$
alternative pkg.	no	yes, SOIC

Network approach - I

- using actuator wires for communication

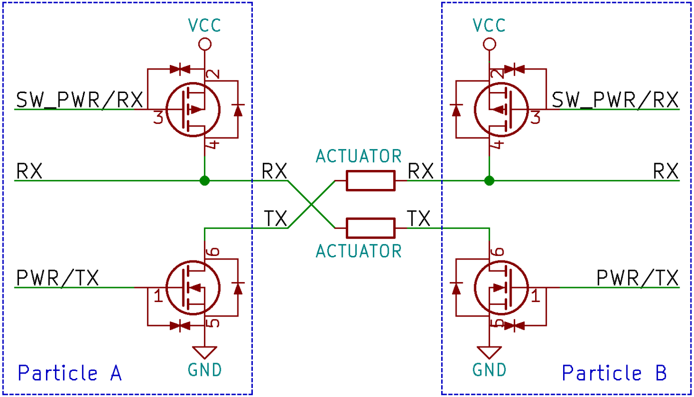


Figure 6: *transmission and reception wiring*

Network approach - II

- linear network
- daisy chained participants

Advantages

+ simple to implement
+ no media access control
+ no loops
+ no dynamic routes

Disadvantages

- error detaches segment
- no recovery for segment

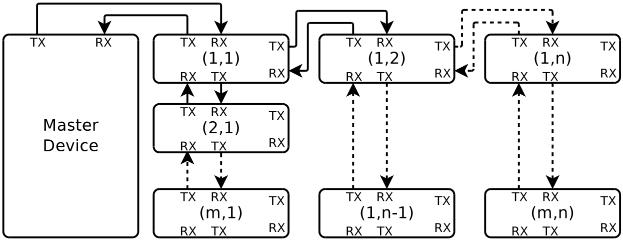


Figure 7: *network topology*

Particle version 1.0

- chain of development particles
- light bulbs replaced actuators

Advantages

+ not mounted in chain mechanics
+ adjustable length via jumpers

Disadvantages

- time consuming assembly
- ATiny20: too less SRAM and FLASH

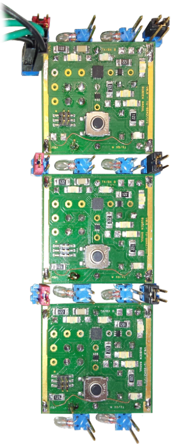


Figure 8: *Version 1.0*

Particle version 1.21

Advantages

+ simple to extend network
+ configurable network dimension
+ faulty particles can be replaced
+ higher particle density

Disadvantages

- bound to grid board size

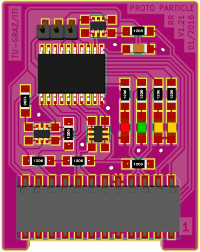


Figure 9: *pluggable particle*

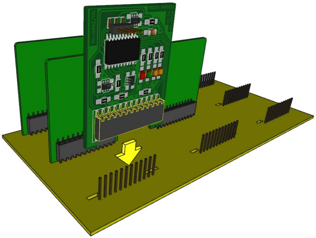


Figure 10: *grid board*

Outline

Results

We have designed
- a particle hardware that applies actuators also for communication and
- a network structure for daisy chain communication in a square lattice.

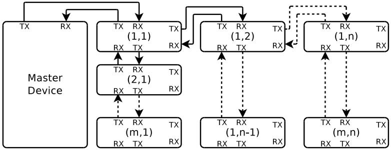


Figure 11: *network topology*

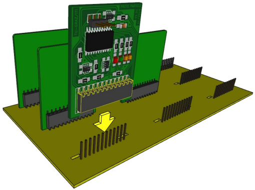
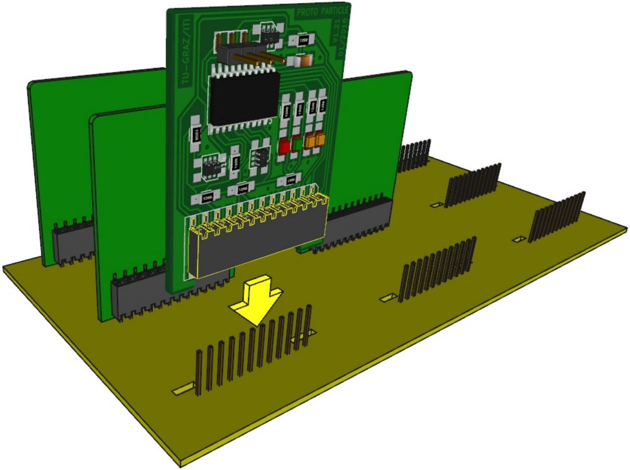


Figure 12: *development hardware*

Outline

Future work

- communication protocol
  - Physical Layer: implement Manchester coding
  - Data Layer: fault detection
  - Network Layer:
    network initialization
    time synchronization and compensation
    actuator task scheduling
- runtime compensation of RC-oscillator drift
- customize boot loader for remote programming
- hardware
  - simplify development board
  - enhance grid board


PROTO PARTICLE
TU-GRAZ/ITI