Daisy Chain Communication for long Chains of Robotic Particles

Raoul Rubien, BSc
rubienr@sbox.tugraz.at

***Abstract*—Chains of robotic particles are able to change their shape in a programmatically. By applying multiple chains surfaces capable of shifting their form may be realized. The sum of such chains can be viewed as programmable matter. Programmable matter is a collection of small scaled units integrating computing, sensing, actuation, and locomotion mechanisms [1] that is able to change physical properties on command [2] and thus a universal material. It usually consists of a high volume of units.**

We apply particle chains as presented by Lasagni et al. in [3] and present a method to exploit the actuators in the system as communication channel. This method helps minimizing the particle size and thus the weight which extends the maximum chain length. It overcomes also other limitations of that work.

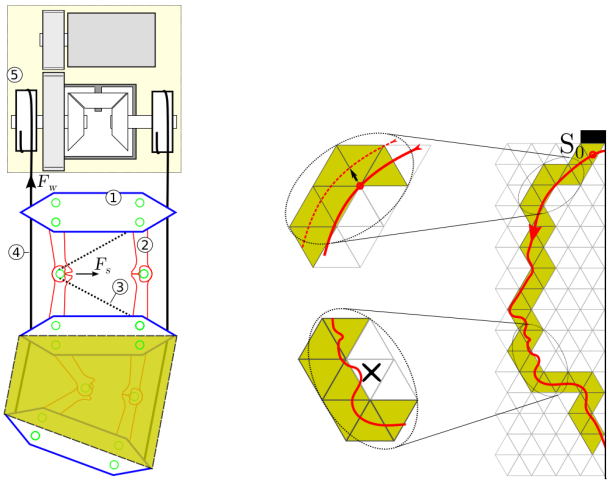
This work elaborates the design and assembly of a particle prototype and also the necessary programming tool chain. It focuses on the hardware implementation. Software details as communication protocol, addressing or network discovery will be part of the upcoming work based on the outcome of this project.

I. Introduction

Unlike several in literature proposed folding methods [1, 2, 4] the force guided chain nodes do not utilize motors, magnetic adherence fields or origami folding to achieve folding. Force guided chains are made of tied particles that are fold-able in two directions of one geometric dimension similarly to a snake's shape when moving on a plane surface. With multiple parallel mounted chains, shape shifting surfaces may be realized. Such surfaces are able to approximate 3-dimensional models at least partly in a 2.5-dimension.

A. Functionality [3]

All nodes of a free-hanging particle chain are naturally pulled down by gravity. Thus the chain's natural state is unfolded and the particles are ordered in a line. Between neighboring particles two connections, one at the left and a second at the right-side, are realized with monostable hinged edges, as illustrated in fig. 1a. When the chain is straight, the edges reside in a locked state. In this state an utilization of the external force (4) and (5) in fig. 1a would lead to a contraction of the chain but will not shape it. Only if hinges are unlocked in advance the force would lead to shape forming. As an example let us see what happens if the chain is contracted but one left hinge in-between two particles is unlocked and all other are locked. The applied force contracts the chain and thus compresses the particle's left-side (shorten the distance) at the unlatched hinge position whereas the right-side remains at the same length. This ends in folding of two particles to the direction where the hinge was previously unlocked as shown in the highlighted zone of fig. 1a.



(a) Mechanical design of chained particles with hinges (2) that are unlocked actuators (3). When applying external force (4) unlocked hinges are folded.

(b) Shaped chain example. Chain is aligned to a grid following a shape.

Figure 1: Mechanical design and folding example [3] of a chain.

B. Limitations

The applied particles do not operate autonomously. They have to be coordinated at a higher level. Thus they must at least communicate with a bus master. In Lasagni et al. as communication protocol the Dallas 1-Wire®[1] bus is utilized. Among others, for this application the bus brings significant disadvantages. 1) The protocol's maximum current limitation: If the current consumption of the attached devices exceeds the limitation communication must be decoupled from the power supply. To overcome this issue in Matteo et al. the communication is decoupled through time division. The system switches in between power supply and communication mode accordingly. As a consequence of that the bus master looses synchronization with the slaves which introduces a delay after each operation to restore the bus communication. During communication the particle power supply must be

[1] https://en.wikipedia.org/wiki/1-Wire (01/2016)

buffered beforehand with a capacitor which introduces a more electronic parts per particle. 2) Large addressing overhead of 64 Bits: The addressing is rather huge according to the needed payload for this application thus the addressing field produces a lot of overhead. 3) Lack of advanced features: The bus provides no advanced addressing features which were desirable in a particle network such sending a datum to a range of nodes. 4) Also the network discovery comes with some limitations. Network addresses can be easily retrieved with the Dallas 1-Wire® bus but not the placement of nodes. Thus the network positions must be probed in a brute-force way. Beside the bus limitations a chain's maximum length is physically limited by its weight.

II. Motivation

The optimal particle design would be very small. Combining a optimal hardware design, a customized communication protocol and the chained arrangement we want to present a daisy chain communication method which in contrast to Matteo et al. [3] exploits the actuators embedded in the system.

Our primary motivation is to minimize the size and weight of particles. Therefore we attempt to lessen the number of electronic components per particle since this physical parameters help chains to be miniaturized and extend the physically limited maximum length. Although we decouple communication from power supply we cannot eliminate the need of time division multiplexing as transmissions and actuating must never overlap.

Our secondary motivation is to enhance the communication overhead, the duration and also the network discovery. Thus we set up a daisy chain protocol which allows to use the underlying physical infrastructure as bus or peer to peer network. For the upcoming work this ensures enough freedom to choose one or both option/s for data transmission according to the use case scenario. With two actuator wires per particle pair the the communication protocol can be developed to support full duplex operation.

III. Goals

As the new protocol's physical layer differs from the current one there is no chance to re-use the circuity. This circumstance forces us to build a new prototype that is able to sustain the upcoming work. The outcome we are interested in are a combination of hardware and software infrastructure that sustains the protocol development. 1) Hardware related: a) a fully functional PCB project (schema and routed PCB) that can be chained, that allows b) modifying rapidly the number of network nodes and c) a simple debugging method (for example test point pinout). 2) Software related: a) a Unix-based toolchain, b) a test software that can be used to check newly assembled boards for errors, c) a simple debug possibility and d) a convenient method to invoke test cases on a sensor network simulator.

IV. Requirements

With respect to the principal requirements illustrated in fig. 2 the project requirements can be listed as: 1) Building a development particle prototype that substitutes the design of fig. 3 during development. In contrast to the currently existent particle, the new prototype must be capable of transmitting data to adjacent nodes using the actuator wires. 2) A new prototype must be able to control the actuators and provide the support for serial full-duplex communication via a N/P-Channel MOSFET. 3) Prototypes have to be handy, offer access to test points such as rx, tx before and after the MOSFET transistors and 4) several spare test points directly connected to the MCU. Despite of the productive particle the development PCB size is not required to be at minimum. 5) CLKO [5] pin must be connected to one test point for potential internal RC-oscillator calibration. 6) Particles have to signal their internal state such as heartbeat, status, error by means of LEDs. 7) The ISP programming interface must be easily accessible. 8) Self programming: Particles need to be capable of enrolling their firmware on their next neighbors. 9) Actuators should be replaced by 5V light bulbs with similar electrical characteristics. 10) The network topology is a combination of tree [6] [7] and linear daisy chain [7] network. Additionally added chains must be connect-able to the network as illustrated in fig. 4a.

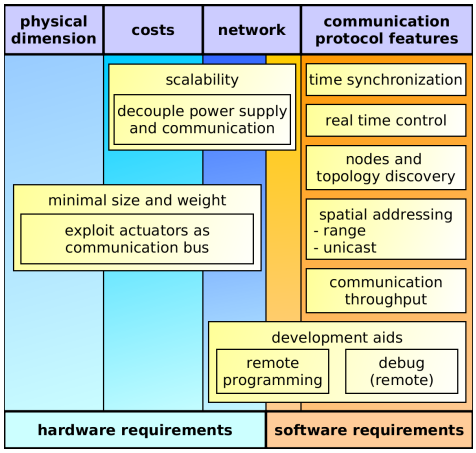


Figure 2: Principal Requirements

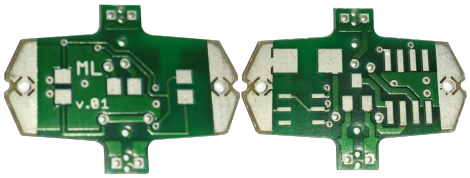


Figure 3: Front and back-side of the current unequipped particle PCB layout [3]. The dimensions are approximately $2cm \times 1cm$.

V. Materials and methods

This section elaborates the hardware development and software simulation approach with regard to the listed requirements in section IV. Decisions made are explained in detail.

A. MCU selection

With respect to the requirements in section IV we started searching a low level economically priced MCU. When comparing MCUs of different manufacturer we chose Atmel® because of several reasons. The most significant are the availability of 1) a free of charge usable open-source compiler, 2) a variety of inexpensive programmer hardware, 3) good documentation and many examples and 4) the increasingly used MCU family. With the Tiny (ATTiny) MCU category Atmel® provides small sized 8-Bit low level micro controllers which perfectly meets our needs.

1) First prototype: The firstly created development particle board applied an ATTiny20 MCU. This MCU provides 2 Kilobytes of flash memory and 128 Bytes of static random access memory (SRAM). To proof the protocol concept with the chosen MCU a quick survey demonstrated that 2 Kilobytes of flash will not be sufficient. Just the implementation of a simple neighbor discovery exhausted up to 50% of the flash memory. Regarding SRAM size we did no extra survey but the experience we made showed up that this resource may be critically low.

2) Modified requirements: During investigating the communication protocol requirements have been refined slightly. A chain communication port was introduced to connect whole chains to the network without the need of additional master device per chain. For that reason three independent pin change interrupts, one per reception wire, are necessary. Hence the ATTiny20 is not be applicable any more. Also self programming is desirable at a later moment when the firmware of a whole network of particles has to be exchanged. This can be solved using a customized boot loader that receives, writes and forwards a firmware. Further we desire a big MCU package on the development board since it is very handy to mount and access for later measurements.

3) Result: Taking in account that the MCU package on a productive particle should be as small as possible we chose the ATTiny1634. This MCU comes with 16 Kilobyte of flash memory, 2 Kilobyte of SRAM, enough pin change interrupts and also 2 UART ports.

4) Side benefit: The shift away from the ATTiny20 also eases the firmware flashing. In case of ATTiny20 flashing a firmware is very costly since it supports no Serial Peripheral Interface (SPI) but only a Tiny Protocol Interface (TPI). A modified RS232 breakout board from SparkFun[2] with a customized avrdude[3] configuration using BitBang[4] protocol had to be applied. Fortunately this is not necessary for the SPI supported by the ATTiny1634.

B. Network topology

For this project many network topologies may be applicable but as mentioned our motivation is to exploit the already available actuator wires. They can be safely used as communication channel. Since each particle is connected via two actuators to it's neighbors we can use them to build a dual cable network system [8]. The network system uses one wire as up-link channel and the second as down-link channel.

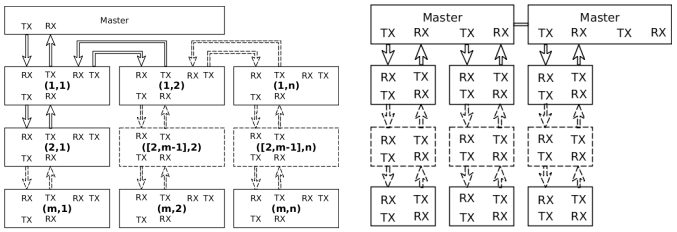
1) Network topology: We also opted for building a daisy-chained network where nodes are connected as peer to peer nodes. With that decision particles can be connected as linear network to achieve a particle chain. Due to the fact that a high number of chains is to be expected within an application the communication with chains needs to be bundled. In case of chains being connected directly to a master device (no bundled communication) each chain occupies two I/O pins. That also implies that multiple master devices need to bee employed if the number of available pins is exceeded as illustrated in fig. 4b. It also complicates the protocol by adding the necessity of master to master communication. A bundled method lowers the amount of occupied I/O pins at the network master regardless of the network size (fig. 4a). Thus we embed three identical communication channels per particle: 1) north - the communication port to the upper particle, 2) south - the communication port to the lower particle and 3) chain - the communication port to the next chain. The chosen network topology is a tree structure with chained nodes. This involves some risks. If a particle malfunctions the network is split into two parts. The interrupted segment is then not able to communicate with the root any more. Furthermore the daisy chained nodes' nature is to work as repeater. Each received data must be intercepted and forwarded. This adds a specific delay per node during data forwarding. Nevertheless the delay can be minimized by forwarding each received signal immediately to the next communication ports. Preliminary experiments showed the time shift between incoming signal and forwarded signal is about $2.2\mu s$ per MCU. The test was set up with an ATMega2560 using an external crystal oscillator at $16MHz$. This delay is expected to be longer in the real application since the MCU frequency is lower and the main routine is not remaining empty as in our test. Hence a slightly increased interrupt latency is caused by the jitter of multi cycle operations being executed when the inputing signal arrives.

2) Linking the network: To realize the network, particles chains are connected at the first chain's particle (later addressed as head particle or head), to the next chain. As an example two chains a and b are connected by linking the head particle's chain port of a to the head's north port of chain b. With this arrangements we can construct a network as illustrated in fig. 4a. The only communication entry point to the network is the north communication port of first chain's head particle which we term as the origin node.

C. Hardware layout

To sustain the upcoming development of a new communication protocol which uses actuators as communication lines a custom PCB that is capable of switching the actuator working mode (communication or actuating) is necessary. Also effortless access to hardware is desirable to be less time consuming when analyzing the physical communication. For that reasons a new prototype PCB needs to be developed and assembled. It should permit the developer to have fast access to several important test points and provide some visual signals as well.

[2]www.sparkfun.com (01/2016)
[3]http://www.nongnu.org/avrdude/ (01/2016)
[4]https://en.wikipedia.org/wiki/Bit_banging (01/2016)

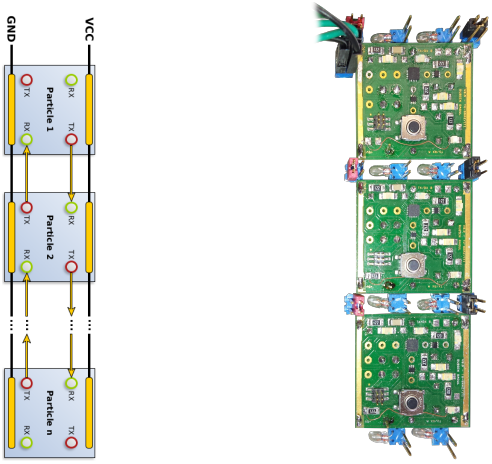


(a) Particle network with cascaded particle chains. They are connected their first particle.

(b) Particle network without cascaded chains. Each chain communicates directly to one master device.

Figure 4: Cascaded chains versus direct chain communication. One dashed rectangle represents a set of nodes.

1) Preparatory work: The current particle development board (V1.21) has past several versions. The first idea was to chain particles without using an underlying frame. Development nodes were conceptually designed to be connected at their power supply pads by using strong inflexible wires as depicted in fig. 5a. This should give enough stability to handle short chains and protect the light bulb terminals from breaking. Therefore the pads were realized stable and placed along the whole adjacent PCB sides. As a consequence of that, once a chain is assembled segments cannot be detached any more. Though detaching chain parts is desirable. To deal with that we mounted particles on a matrix board as shown in fig. 5b. All particle to particle connections were passed via jumpers. The result was stable and handy but unfortunately the fixing consumed too much time.

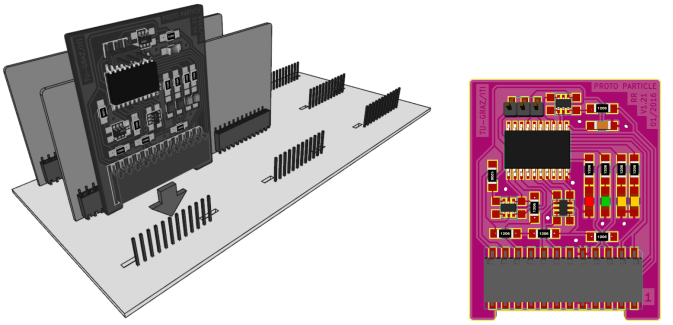


(a) Proposed model of a particle chain. Power supply can be soldered on a stable rail. Actuators may be replaced by 5V light bulbs.

(b) First working version (1.0) of a development particle chain. Actuators are substituted by light bulbs.

Figure 5: First implementation of a development particle chain.

2) Current result: For the reasons mentioned before we decided to build a grid board (see fig. 6a and fig. 6b) that provides the network connections for each particle. The idea is to make the network configurable by plugging/unplugging particles to/from the grid board. The In-System Programmable port is outsourced to the board which makes the particle design more uncomplicated. The upcoming protocol development will for sure need debugging capabilities. For this tasks the board provides three arbitrarily usable test pints and several LEDs. It is most likely that the protocol needs to synchronize the particles. How this is solved in detail is not part of this work. Never the less if the internal oscillator has to be calibrated at runtime the CLKO pin is needed to reflect the internal oscillator frequency. Therefore the CLKO pin is connected to one test point. If at a later moment an additionally serial communication is desired it can be derived from the SPI port since the MOSI/MISO pins also provide UART.



(a) An examplary 3×3 grid board. It provides the network infrastructure.

(b) Pluggable particle board. Two leads provide additional stability when plugged into the grid board.

Figure 6: A grid board exemplar and the particle board in detail.

D. Software simulation

To speed up the upcoming software development a software simulation is desired. Fortunately a particle network does not need synthesized input samples if any network node can be depicted in a network simulation. Hence we just need a simulation framework that is capable of simulating whole networks. Anyway if a network can be simulated there are still issues to investigate. For example how are particle's clock synchronized within the framework? Is the framework capable of scaling the clock or introduce clock drift per particular particles? Since one network does not only consist of MCUs but also some periphery components per particle's PCB we want to simulate a particle as a whole. With this desires we investigated available simulations and opted for the Avrora [9] sensor network simulation. Among others the framework is capable of simulating an ATTiny16 MCU and it is possible to simulate particles as a whole. The specific implementation of a particle including actuators, test points and LEDs is achieved by implementing Avrora's Platform interface. A proof of concept has been done with particle prototype hardware version 1.0. A neighbor discovery has been implemented by use of the simulation and then successfully tested on hardware.

E. Tool chain

The current state of the project also covers software implementations for concept proving. We organized the source with

CMake and a couple of Unix tools. With that we constructed a build chain to easily launch builds, flash particles, start sensor network simulations, retrieving simulation statistics or debug particles via UART and much more.

VI. Future work

In our future work we plan to develop a communication protocol that provides a way to communicate to each chain's particle. The protocol will span the first three layers of the OSI model: 1) Physical Layer, 2) Data Link Layer and 3) Network Layer and will address the network coding [10], self enumeration and addressing, scheduling of actuator tasks and time synchronization.

For the time synchronization it is to be determined if exploiting the synchronization of a Manchester coding (layer 2) is accurate enough or if it has to be solved in layer 3. Since particles use their internal oscillator it is of high interest if calibrating the internal oscillator at runtime [11] is feasible.

Also enrolling of particles' firmware we plan to achieve by using a customized boot loader to speed up the deployment in networks. The idea is to replicate on particles firmware to its next neighbor et cetera.

VII. Acknowledgements

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.

References

[1] Seth Copen Goldstein; Jason D. Campbell; Todd C. M. Programmable Matter.

[2] Ara N. Knaian, Kenneth C. Cheung, Maxim B. Lobovsky, Asa J. Oines, Peter Schmidt-Neilsen, and Neil a. Gershenfeld. The Milli-Motein: A self-folding chain of programmable matter with a one centimeter module pitch. *IEEE International Conference on Intelligent Robots and Systems*, pages 1447–1453, 2012.

[3] Matteo Lasagni and Kay Römer. Force-guiding particle chains for shape-shifting displays. *CoRR*, abs/1402.2507, 2014.

[4] E Hawkes, B An, N M Benbernou, H Tanaka, S Kim, E D Demaine, D Rus, and R J Wood. Programmable matter by folding. *Proceedings of the National Academy of Sciences of the United States of America*, 107(28):12441–12445, 2010.

[5] Atmel. *8-bit Atmel tinyAVR Microcontroller with 16K Bytes In-System Programmable Flash*, 2 2014. Rev. 8303H.

[6] Joseph Kizza. *Guide to computer network security*. Springer, London, 2015.

[7] Barrie Sosinsky. *Networking bible*. Wiley, Indianapolis, IN, 2009.

[8] Andrew Tanenbaum. *Computer networks*. Prentice-Hall, Englewood Cliffs, N.J, 1988.

[9] Ben L. Titzer, Daniel K. Lee, and Jens Palsberg. Avrora: Scalable sensor network simulation with precise timing. In *2005 4th International Symposium on Information Processing in Sensor Networks, IPSN 2005*, volume 2005, pages 477–482, 2005.

[10] Andrew Tanenbaum. *Computernetzwerke*. Pearson, Munchen, 2012.

[11] Atmel. *AVR054: Run-time calibration of the internal RC oscillator*, 4 2008. Rev. 2563C-AVR-04/08.

List of Figures

List of Tables

Contents