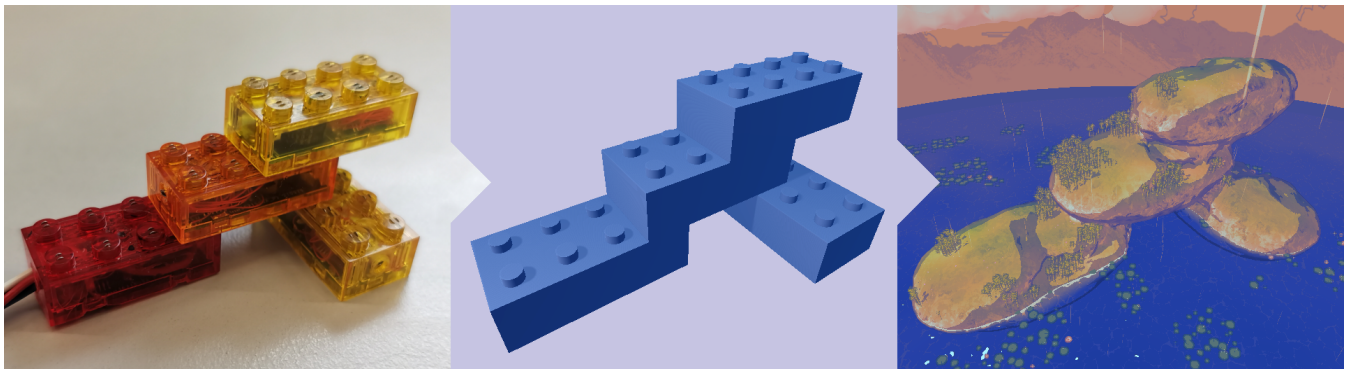
Vision, Modeling, and Visualization (2023)
T. Grosch and M. Guthe (Eds.)

# Digitizing Interlocking Building Blocks

Sebastian Lieb, Thorsten Thormählen and Felix Rieger

Philipps-University Marburg, Germany



**Abstract**

*Interlocking building blocks (such as LEGO®) are well-known toys and allow the creation of physical models of real objects or the design of imaginative 3D structures. In this paper, we propose a novel approach for digitizing building blocks with the original LEGO® form factor. We add a microprocessor to each 4 x 2 block, and the blocks communicate with each other via a two-wire connection provided in every nub. This poses the additional challenge that communication and power supply must use the same two-wire connection, which is addressed by alternating between the two modes over time. We introduce a protocol that checks for connections and propagates all connection information through the block network. We can then pass this information to a connected computer, which reconstructs the structure of the block network. We present several successfully digitized example configurations and discuss failure cases. Furthermore, two end-user scenarios are demonstrated, which show the potential of our approach as an intuitive human-computer interface.*

## 1. Introduction

Every child knows how to use and build with building blocks without much explanation. Today's computer interfaces as well as 3D design and modeling applications rely primarily on vision and provide feedback to their users through a monitor screen. However, learning processes benefit from multisensory stimulation, meaning that tackling and analyzing any subject or task with multiple senses enhances engagement and therefore improves the outcome, or as [Cha17] states: "combining information from multiple senses creates robust perceptions, speeds up responses, enhances learning, and improves detection, discrimination, and recognition". Interlocking building blocks, such as LEGO®, are an intuitive and accessible visual-haptic interface to create physical models in a natural and playful way.

[Gau15] describes the LEGO® system as "a tool for supporting creative thinking, developing creative cultures, and contributing to processes which might make a difference in how the world works". Also, [BLCCL20] shows that when LEGO® blocks are applied to a creative design process, divergent thinking indicator scores significantly increase for both professional and non-professional users, compared to a pen and paper design process. A study carried out by the LEGO® foundation [PT19] emphasized the importance of play in education and additional studies show how "children can foster cognitive, social, emotional, creative and physical skills through active engagement in learning that is experienced as joyful, meaningful, socially interactive, actively engaging and iterative" [PTB22]. Furthermore, LEGO® Mindstorms® and [RMSS96] have successfully opened up a new way of teaching computer science and robotics to students and LEGO® can be considered "More than Playing a Toy" [CGKC20]. A study carried out by [MI18] indicates that "tangibles have a greater positive impact on learn-



DOI: 10.2312/vmv.20231221

ing, situational interest, enjoyment, and programming self-beliefs" when applied to a collaborative programming task. Consequently, interlocking building blocks have the potential to act as an intuitive interface for human-computer interaction if the physical creation can be digitized reliably.

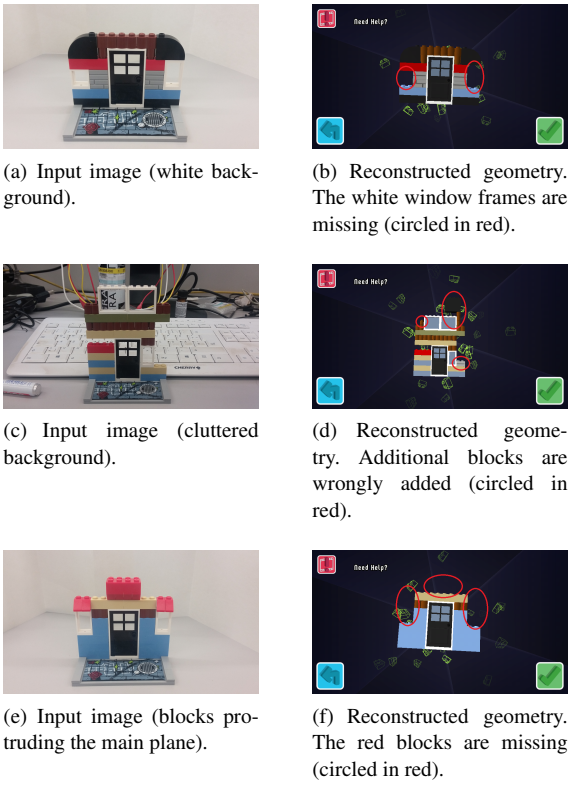
In this paper, we propose a system that digitizes the structure of interlocking building blocks and reconstructs their configuration as a 3D model. We present special hardware in the form of interlocking building blocks with the original LEGO® form factor that contains a micro-controller, a custom PCB, and electronic components to power the micro-controller and connect its IO pins to the building block's nubs. Additionally, the software for both the micro-controller and a connected PC to run our digitizing algorithm is described. Existing approaches with LEGO®-sized building blocks use single-view image-based techniques to reconstruct block configurations, which is limited because occlusion and ambiguities make robust computer vision challenging. Using additional hardware that can be put inside interlocking building blocks is potentially more reliable for the reconstruction of complex structures and small, low-power micro-controllers are available at a low cost. In the next chapter, we will first take a look at similar work on the topic of the digitization of interlocking building blocks and related topics. We will then continue to present our approach by introducing both the hardware and software aspects of constructing a smart interlocking building block, and our proposed digitizing algorithm. At the end of this paper, we present our results, and further evaluate our system by demonstrating two prototypical end-user scenarios. The paper concludes by discussing the limitations of our approach and future work.

## 2. Related Work

Representing and interacting with virtual three-dimensional objects by using tangible small "claytronic atoms" (or short "catoms") is a concept envisioned in [GCM05] and further built upon in [ILBL12]. Both papers focus on the possibilities and implications of programmable matter: self-assembling miniature robots that can take any shape to represent virtual objects and make them tangibly interactive. Implementation-wise, there have been several projects on the topic of digitization of interlocking building blocks, that can be put into two main categories: image-based and non-images-based methods.

**Image-based block digitizing:** [LEG14] and [Osm20] propose an image-based approach to reconstruct physical block structures inside the digital realm, where [LEG14] reconstructs a given configuration of blocks and [Osm20] is using the reconstructed block configuration to generate simple programs to teach programming to beginners in a tactile way. Both approaches are only capable of digitizing a planar configuration of blocks and can not handle more complex topologies. [MWC*12] and [GFCC12] both propose a system that reconstructs the configuration of LEGO® DUPLO® sized building blocks with the use of a depth-sensor. [GFCC12] reconstructs the configuration block by block over time and requires additional user input and camera angles during each step of the building process. A similar approach by [JB11] uses a depth-sensor camera to successively scan and add physical building blocks to a digital structure to build larger three-dimensional models with a small number of building blocks. In contrast, we propose a non-image-based approach to reconstruct block configurations that is independent of camera visibility.

To highlight some of the issues related with camera visibility, we stress-tested the commercially available *LEGO® Fusion Town Master* application [LEG14]. Fig. 1 shows several examples.



(a) Input image (white background).

(b) Reconstructed geometry. The white window frames are missing (circled in red).

(c) Input image (cluttered background).

(d) Reconstructed geometry. Additional blocks are wrongly added (circled in red).

(e) Input image (blocks protruding the main plane).

(f) Reconstructed geometry. The red blocks are missing (circled in red).

**Figure 1:** *Failure cases of LEGO® Fusion's image-based reconstruction method [LEG14].*

As we can see, the *LEGO® Fusion*'s solution has some anticipated disadvantages, due to its image-based approach. Image quality and background setup can drastically influence the performance of the method and their approach either misses blocks (Fig. 1(b)) or interprets background object as additional unwanted blocks (Fig. 1(d)). Also, the LEGO® Fusion algorithm can not deal with blocks that protrude the one-nub wide main plane on the construction plate (Fig. 1(f)). LEGO® Fusion requires a base plate and the main door to be present as they are used as calibration markers. However, the quantity and variety of blocks that can be reconstructed by LEGO® Fusion is substantial and the manufacturing costs of individual blocks are relatively low.

**Non-image-based block digitizing:** [Mer09] uses computerized tiles that connect and form networks on a two-dimensional plane. These tiles are equipped with LED screens to run and display multiple applications that can perform math, play music, and communicate with their connected neighbors. Similarly, [KIK01,

SIW*02, WIA*04] propose connectable cubes that form digitizable, actuator-enhanced, three-dimensional networks that are able to run interactive applications. [SG06] propose connectable cubes to build programmable networks that form small configurable robots. These hardware approaches are more complex and larger in size than our blocks, creating an experience similar to a rearrangeable tablet or smart device. Our method focuses solely on reconstructing the shape of the block network, so higher-level applications can run on a host PC and our block hardware can be kept simple, inexpensive, and small. [AFM*00] shares this very focus and proposes a system that reconstructs the shape of micro-controller-enhanced interlocking building blocks with the dimension of 100 x 50 x 25 mm. However, our goal is to digitize building blocks with the dimensions of a regular 4 x 2 LEGO® block, which is 31.8 x 15.8 x 15.8 mm. [LRL17] and [GOI98] both implement an interface that digitizes three-dimensional topology, where [LRL17] uses struts and [GOI98] uses triangles as their atomic building block in contrast to our LEGO®-shaped blocks. [AIH*14] propose a system that allows users to construct 3D shapes by stacking blocks at arbitrary positions and angles. This system uses infrared communication between the blocks to digitize the shape of the block arrangement with blocks of the size 100 x 50 x 25 mm, which is also much larger than our form factor. [CMRB12] and [LCT*14] use differently-shaped magnetic building blocks in combination with a touch screen tablet to build tactile interfaces and applications. Their approach is not designed to build larger connected 3D structures, but to directly interact with applications shown on the touch screen. [IS18] analyze the magnetic field of magnetized LEGO® DUPLO® blocks to digitize their block structure. However, their system consists of larger blocks and is limited by a building height of three blocks, whereby the accuracy degrades with the distance to the ground plane.

**Other related work:** Other interesting projects, such as [Bri21] are very successful at using image recognition to identify loose interlocking building blocks to find possible block configurations and suggest building instructions, but have the same visibility problem as other image-based methods and aim at a different use case. [Wes15] propose an approach to find block configurations that can be built with physical blocks, given a digital 3D model. [McS13], [Mic15], and [Yta15] follow the idea of augmenting interlocking building blocks with micro-controllers and additional hardware to be able to put more functionality into the building block world. They show that modern hardware is small and powerful enough to be able to augment very small toys with computing capabilities, but their goal is not to reconstruct the block connections themselves, but rather to augment the block behavior, which differs from our goal. Also, they do not provide a solution for a power supply mechanism that is suited for our use case. The idea of using augmented interlocking building blocks to create complex modular structures is presented in [OH18], [Zha19], and [Gon18], who are using precisely printed interlocking building blocks in the area of microfluids to build quick prototypes for complex microfluid units. In the educational field, [MD21] use traditional interlocking building blocks as modules for general chemistry teaching in the classroom and, similarly, [MSGD16] use interlocking building blocks to depict periodic properties of elements such as electron configuration to blind and visually impaired students in a playful and tactile way. Also, [Ent17] utilizes interlocking building blocks to explain the life cycle of plastics from their production from monomers through the different treatments of end-of-life plastics. Although these approaches do not use interlocking building blocks digitally, their educational applications demonstrate that interlocking building blocks can be, and are already successfully used as an expressive and tactile tool to model complex processes and entities.

In general, our proposed approach differs from the related work mentioned above by either being much smaller (31.8 x 15.8 x 15.8 mm) or not using computer vision instead of additional hardware to digitize the true three-dimensional topology and not only a planar configuration.

## 3. Design Goals

Our aim is to design, implement, and produce a working prototype for digitizing interlocking building blocks that incorporates the following goals:

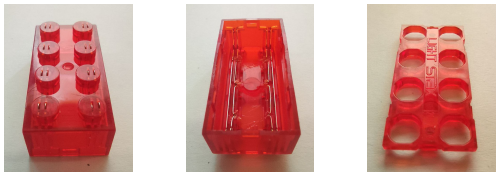
- The electronic components should fit inside a regular 4 x 2 LEGO block with the dimensions of (31.8 x 15.8 x 15.8 mm).
- The block design should eliminate the need for a battery, as we want to prevent the individual charging of blocks.
- Hardware and block components should be low-cost and should consist of readily available components.
- The digitizing algorithm and protocol should detect block connections at interactive speed.
- The system should allow the digitization of truly three-dimensional topology.
- The system should be stateless, meaning any number of blocks can be removed and added at any time.
- The proposed system should digitize the shape of the network robustly and accurately, so applications can be built on top.

## 4. Implementation

### 4.1. Hardware

One of the biggest problems to solve during the development of our hardware prototype was the supply of power to each connected block. We are using the plastic frame from the 4 x 2 blocks from LIGHT STAX® (Fig. 2, [STA14]) as the housing for our block hardware. The LIGHT STAX® hardware is intended to power a LED that illuminates the transparent block, which is why every nub is equipped with two contacts ($V_{CC}$ and ground). In the worst-case scenario, this results in only two connections between two blocks (Fig. 3). Because the plastic molding of submillimeter structures for a custom housing is very expensive during the design stage and off-the-shelf 3D printing does not have the required accuracy, our prototype had to rely on the existing LIGHT STAX® frame and two-wire connection per nub.

Our approach does not use batteries inside the building blocks, because constantly replacing or recharging batteries in many blocks might be a dissatisfying user experience. Therefore, our approach uses one contact for ground and the other one is used for both data-transfer and power supply, alternating in time. Modulating the data



(a) top (b) bottom (c) bottom lid

**Figure 2:** *Plastic frame of the LIGHT STAX® block. There are only two wire contacts per nub.*

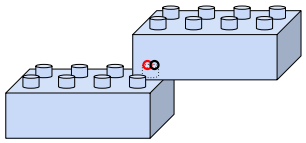


**Figure 3:** *Two blocks connected by a single nub, connecting both ground (black) and power supply/communication wires (red).*

on top of the voltage supply would require additional hardware, which could not be produced in a form factor that fits into a 4 x 2 block with the technology readily available for us. Therefore, our approach relies on the micro-controller software to alternate one contact per nub between data transfer and power supply, which will be outlined in more detail in chapter 4.2.
The proposed hardware consists of:
1. A micro-controller driving a communication protocol and logic. The ATMega328P is chosen, because of its low cost, low power consumption, small size (TQFP), and a sufficient number of IO pins.
2. A custom PCB, connecting the IO pins and the ground to the outer nubs of the interlocking building block and providing a mount to connect a small capacitor (see Appendix Fig. 14) to the chip.
3. A 0.047F capacitor to provide enough power for the ATMega328P to run for approx. 10 seconds without an external power supply.
4. A plastic housing of the components in the form factor of interlocking building blocks (by LIGHT STAX®).
5. Insulated copper wires.
The schematics of our building block hardware are shown in the Appendix Fig. 13. Furthermore, one block is required that we call "root block", which only differs from the other blocks by having a direct power supply and a connection of one of the IO pins to a serial port to communicate with a PC. This root block acts as our main interface with the PC reconstruction software and all other blocks need to have a direct or indirect connection to the root block to be recognized by the block network.

### 4.2. Software / Protocol

There are protocols, such as "1-Wire", that share our goal of combining communication and power supply over a 2-wire connection, but we realized that those were not satisfying all of our requirements. For example, "1-Wire" devices are all connected to a shared bus system, making it possible to communicate with each other, but impossible to retrieve their physical connection topology. Furthermore, the way "1-Wire" discovers connected device addresses (instead of dynamically assigning them). This led us to the conclusion that a more fitted and custom-tailored solution is needed, which is described in the following.
Our software running on the micro-controller in each block consists of two main phases, that can be outlined as:
**1. Power supply phase:** The root block provides power to all connected blocks over their IO pins. All directly or indirectly connected blocks output power to all pins that are not currently receiving power. Consequently, power is distributed through the block network. During this phase, each block's capacitor is charged to provide enough power for the upcoming communication phase.
**2. Communication phase:** Starting from the root block, communication requests are sent to all connected blocks in a depth-first search manner and connection information is propagated back to the root block. At the end of the communication phase, all connection information is received by the root block and can be sent via its serial port to the connected PC. During communication, information about which pins are directed towards the root block and which are directed away from it in the connectivity graph are gathered and kept for the next power supply phase to prevent cyclic connections in the graph.

These two phases alternate and a complete connectivity graph can be reconstructed after each communication phase. In some cases, connections, which happen ill-timed to the alternating protocol, can prevent the recognition of blocks in the current cycle. These blocks are recognized in the next cycle. We can visualize each block's behavior as a flowchart, shown in Fig. 4.

Fig. 5 illustrates the behavior of the block network. We simplify the example by assuming only one connection between two blocks exists, but the principle works the same for multiple connections. Fig. 5(a) shows the transfer of "connection request" packages (blue numbers) and "full connection" packages (red numbers) in the logical order in which they are sent. Keep in mind that in actuality, the packages do not have to be sequentially sent in the displayed order. For example, when sending the request package 6, the block does not have to wait until an answering connection package is completely propagated back to the root block via 7, 8, and 9 before package 10 is sent. It only has to wait for a little longer than the transmission time of one package. This makes the process a lot faster because communication is happening in parallel, but the block network is still traversed in the same deterministic depth-first manner. Fig. 5(b) shows the broadcast of a "reset" package, communicating to every block that the communication phase has ended and the power supply phase is starting. When broadcasting a package, race conditions can occur, as in Fig. 5(b) two "reset" packages are sent to the topmost block almost simultaneously (package 3). This is not a problem, as every "reset" package is identical and it does not matter which one reaches the block first.

We can also express the communication of our block network in a sequence diagram. Fig. 6 shows the cascading package exchange inside the block network starting at the root block. "Connection request" packages are sent to connected blocks, which will answer with a "full connection" package containing the IDs of the

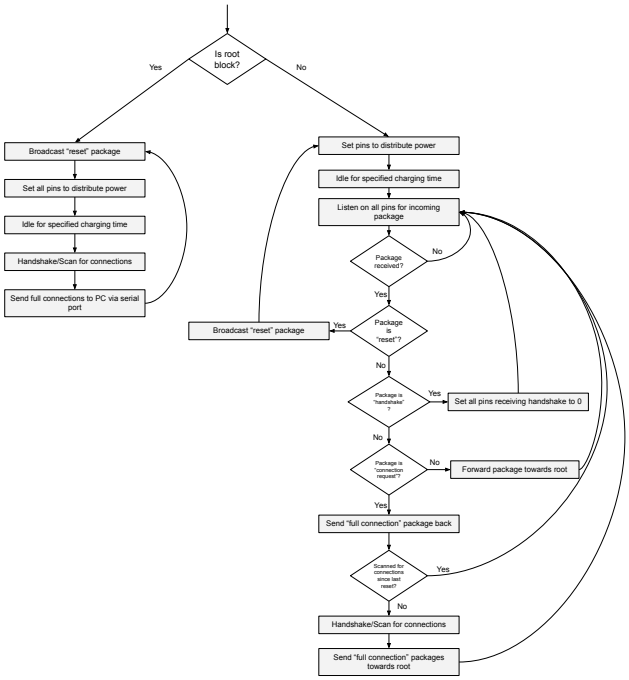
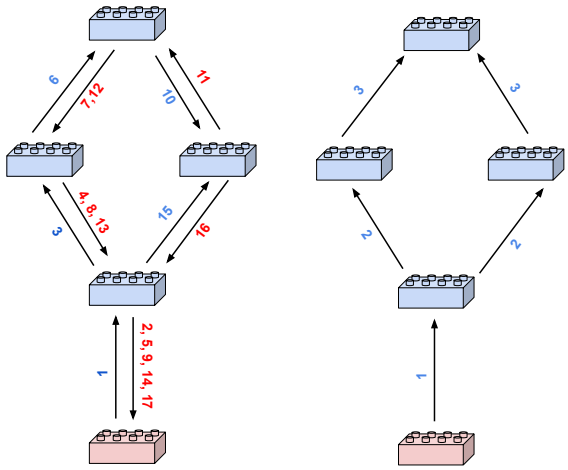



**Figure 4:** *Behavior of a single block sending, receiving and reacting to packages.*




(a) "connection request" (blue) and "full connection" packages (red)

(b) broadcast of "reset" packages

**Figure 5:** *Block network package propagation over time. The blue and red numbers denote the order in which packages are send. The root block is depicted in red.*

involved blocks and pin numbers of the IO pins that are connected. Afterward, the requested block starts sending "connection request" packages to its connected neighbor blocks and forwards all received connections to the root block. This will cascade until all connections are traversed and the full network can be reconstructed (similar to a depth-first search). During one communication cycle, every block keeps track of which pins go towards the root block and which go away from it, by tagging the pins of its first received connection request as "towards root" and every other pin that answers to a connection request as "away from root". This implicitly forms an acyclic-directed tree graph with the root block being the root node and all paths pointing away from it. Since the whole communication process is sequential, we can dynamically assign block IDs by encoding the next free block ID as an 8-bit integer inside the "connection request" packages and increasing its value by 1 every time a new connection is communicated.

Each arrow displayed in Fig. 6 represents an information exchange via a binary encoded package. Each package consists of 32 bits, transferred via asynchronous two-wire serial communication. The first 4 bits being the package header used for synchronization and the next 3 bits notating the package type. The rest of the 32 bits are used depending on the package type, which - in case of a "full communication" package - are used to store the IDs of both connected blocks and the pin numbers of the pins the connection is established with. For most other package types, the package is just filled with trailing zeros. Our full package definition is shown in Fig. 8.

The "reset" packages are always broadcasted to all blocks, meaning the root block is sending a "reset" package to all pins simultaneously and every block receiving a "reset" package sends a "reset" to all pins going away from the root block, communicating the start of the charging phase, without waiting for a response. After receiving and broadcasting a "reset" package, every block goes into the power supply phase for a defined amount of time, after which it wakes up and transitions into the communication phase. The "handshake" is also sent on all pins simultaneously and immediately checked for a response to quickly receive information about which of the block's pins are connected to other blocks. This is helpful because the block has to send "connection request" packages to only those pins in the next step. When a handshake is completed or an "end of scan" package is received, blocks send and forward an "alive" package towards the root to keep the communication chain from timing out. Whenever a "connection request" is received, a "full connection" package is generated and sent back. The "full connection" packages are always forwarded towards the root block. Whenever a block has sent a "connection request" to each of its connected pins (end received connection packages from each of them), it sends an "end of scan" package towards the root. If the root block has received the "end of scan" from its last connected block, all blocks of the block network have been traversed, and the communication phase ends. All received connections are communicated via the serial port to the connected PC and the next "reset" packages are broadcasted to initiate the next power supply phase.

### 4.3. Reconstruction

After each communication phase, all communication packages are communicated to a connected PC via a serial port, where a host application fetches the information for digital reconstruction. Given a digital representation of a single block that fits its physical counter-

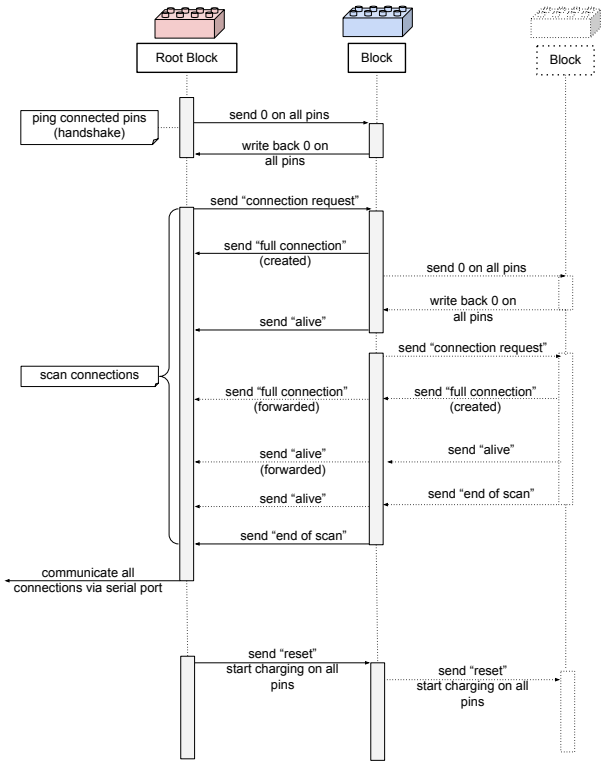
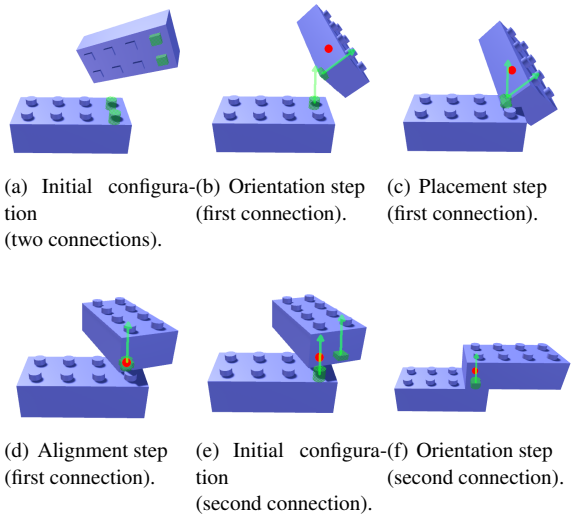



**Figure 6:** *Communication cascade of connected blocks, starting at the root block.*



(a) Initial configuration (two connections).
(b) Orientation step (first connection).
(c) Placement step (first connection).
(d) Alignment step (first connection).
(e) Initial configuration (second connection).
(f) Orientation step (second connection).

**Figure 7:** *Block placement algorithm. The two nubs involved in the connection and their up-orientations are displayed in green. The red dot indicates the current pivot at each step. The placement and alignment step for the second connection is not displayed as it does not change the block's 3D configuration anymore.*

*Bit 0-4:* always set to 0001 (synchronization bits)
*Bit 5-7:* package type, where
- *000:* connection request
- *001:* full connection
- *010:* end of scan
- *011:* alive
- *100:* reset
- *111:* handshake

For a "connection request", the package is defined as:
**0001000AAAAAAAAPPPPBBBBBBBB**
where
- *A:* ID of the block the package is send from
- *P:* pin the packages is send from
- *B:* next free block ID

And a "full connection" package:
**0001001AAAAAAAAPPPPBBBBBBBBKKKK**
where
- *A:* block A ID (closer to root)
- *P:* block A pin
- *B:* block B ID (farther from root)
- *K:* block B pin

**Figure 8:** *Binary package definition.*

part and by the use of the received connection list, we can digitize all involved blocks and calculate their placement and orientation in 3D. To reconstruct the full block structure, we first take a look at all connections and create a virtual block for each unique block ID found in the connection list. Iterating over the connection list again, we can then calculate every block's placement and rotation, relative to the root block, given that the connection list is in depth-first order, starting with root block connections and continuing with connections going away from the root block. Knowing that every connection consists of two blocks, where the first one is nearer to the root block, it is sufficient to always calculate the placement of the second block stated in each connection. This will also make sure that we do not accidentally re-position a block that is already placed correctly. To find the right placement of one block, we can then separate its placement into three steps, visualized in Fig. 7. Our reconstruction algorithm (Alg. 1) consists of three steps that are being executed for each unaligned block of each connection:

**1. Orientation** Rotate the unaligned block such that its connected nub faces towards the nub of the aligned block. The axis that goes through the pivot point and results in the shortest rotation angle is used as a rotational axis. Initially, each block starts with its center as its pivot point.

**2. Placement** Translate the unaligned block such that the connected nub's position is equivalent to the position of the nub of the aligned block. Set the new pivot point to this position.

**3. Alignment** Rotate the unaligned block using its new pivot point and a rotation axis perpendicular to both up-vectors such that the up-vector of both blocks coincide.

It is important that these steps are executed for each connection in the order they occur in the retrieved connection list, as we consider the order of the connection list to successively go away from the root block.

**ALGORITHM 1:** Block structure reconstruction algorithm

```
/* Create a block for each block id */
Set blocks = ∅
foreach Connection c in connectionList do
    if blocks.NotContains(c.blockId0) then
        Block b = CreateBlock(c.blockId0)
        blocks.Add(b)
    end
    if blocks.NotContains(c.blockId1) then
        Block b = CreateBlock(c.blockId1)
        blocks.Add(b)
    end
end

/* Block rotation and positioning */
foreach Connection c in connectionList do
    Block b0 = blocks.Get(c.blockId0)
    Block b1 = blocks.Get(c.blockId1)
    Nub n0 = b0.nubs[c.pin0]
    Nub n1 = b1.nubs[c.pin1]

    /* orientation */
    Vector3 a = n0.position - b1.pivot
    Vector3 b = n1.position - b1.pivot
    Vector3 axis = a.Cross(b)
    float angle = SignedAngle(a, b, axis)
    b1.Rotate(b1.pivot, axis, -angle)

    /* placement */
    Vector3 d = n0.position - n1.position
    b1.Translate(d)

    /* alignment */
    b1.pivot = n1.position
    axis = n0.up.Cross(n1.up)
    angle = SignedAngle(n0.up, n1.up, axis)
    b1.Rotate(b1.pivot, axis, -angle)
end
```

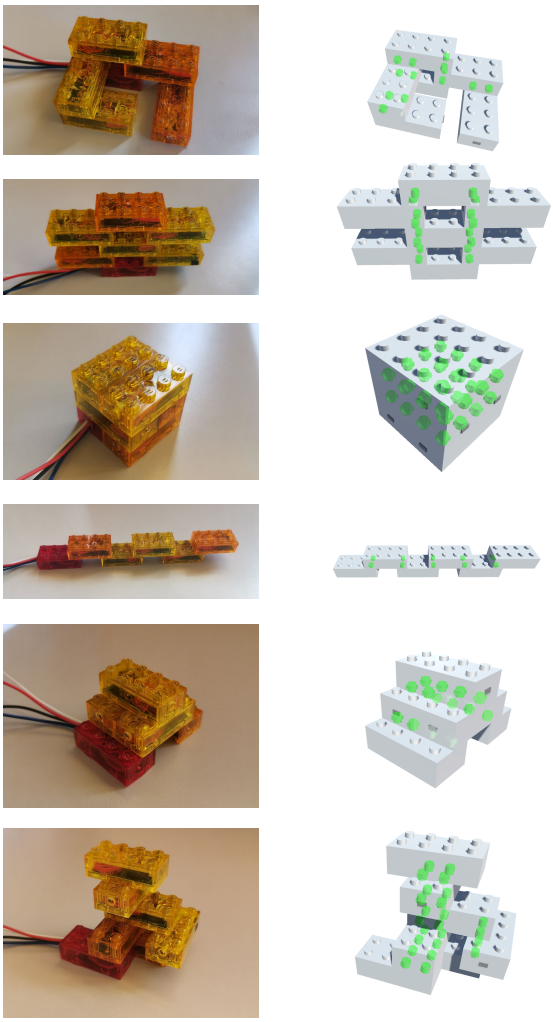


**Figure 9:** *Examples of successful digitization of our approach. Input configuration on the left and reconstruction on the right.*

## 5. Results and Discussion

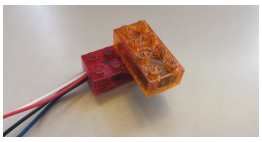
Fig. 9 and Fig. 10 show the results of our prototype. They show various digitized block configurations, whereby Fig. 9 shows successful digitizations and Fig. 10 shows failure cases. Our implemented system meets the design goals stated in Section 3, as it digitizes true three-dimensional topology with low-cost, battery-free hardware. As shown in our supplemental video, our implementation runs at interactive speed and is stateless, allowing for the removal or attachment of multiple blocks at once.

As we can see in Fig. 9, various 3D block configurations can be reconstructed without problems. We can see an issue with individual connections not being closed at 10(b), indicated by missing green nubs where physical connections do take place. This issue rarely appears and does not matter for the proper reconstruction in most cases as not all connections are needed to reconstruct the shape of the block network without ambiguity. We can also see in 10(d) that having two blocks connected via a single nub can introduce ambiguities, which our algorithm is not able to solve properly. However, this is an unusual building technique as it is also physically fragile. In contrast to image-based methods, our method is not prone to image-based errors and is therefore not dependent on camera visibility, clear backgrounds, or good lighting.

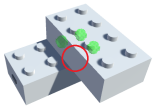
### 5.1. End-user scenarios

To illustrate the purpose and possibilities of our system, we implemented two proof-of-concept applications. Both applications are also presented in the supplemental video.

**Digital assisted building instructions:** The goal of this end-user scenario is to guide the user through a sequence of assembly steps by giving a visual representation of the current state of the block configuration in addition to a visual hint on where to place the next block. The next block in the building instructions is displayed in green and animated with a pulsing animation. Whenever the user places a correct block, auditive feedback in form of an assuring

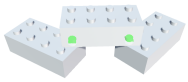


(a) Input block configuration.



(b) Reconstructed geometry. One connection (circled in red) is not closed properly, but our reconstruction succeeds nevertheless.

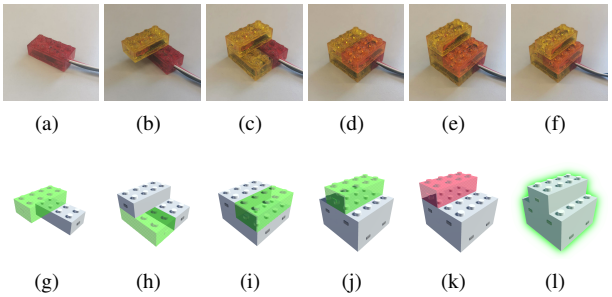
(c) Input block configuration.



(d) Reconstructed geometry. Only one connected nub per block-pair is not sufficient to reconstruct the shape of the block network accurately.

**Figure 10:** *Failure cases of our digitization approach.*

sound is played and the newly placed block is visually animated right before the next step is displayed. Misplaced blocks, as displayed in Fig. 11(k) are highlighted in red to indicate that the user should remove the misplaced block. In addition to the visual feedback, an error sound is played whenever a block is misplaced. If the whole assembly is completed, another assuring sound is played and the digitized block structure is visually highlighted (Fig. 11(l)).

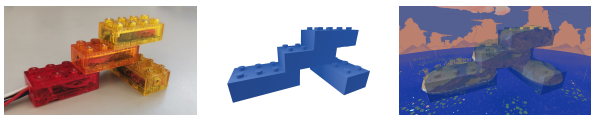


(a) (b) (c) (d) (e) (f)

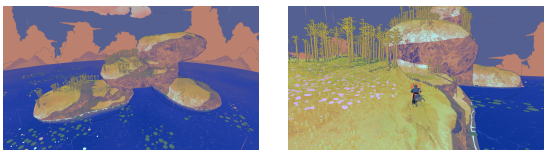
(g) (h) (i) (j) (k) (l)

**Figure 11:** *Digital-assisted building instructions. The Subfigures (a)-(f) show the block configuration in each building step and the Subfigures (g)-(l) the visual feedback for the next step during the assembly. Green blocks always indicate the next block to be placed and red blocks indicate misplaced blocks.*

**Tangible world building for video games:** In another prototypical application, we apply our interlocking building block system to build interactive game worlds. The digitized block configuration (Fig. 12(b)) is used to generate the terrain (Fig. 12(d)) of the game world. The terrain generation step produces a smoothed, procedurally enhanced mesh with natural terrain features that fit our game's aesthetic. The different terrain 'blobs' produced by each building block are smoothly connected similar to the metaballs algorithm described by [Bou97]. In addition, the produced terrain mesh is automatically populated with flowers, grass, and plants to produce a richer game world. During world-building, the user can freely move the camera through the generated game world. We call this state of the application "build mode". By pressing the "F5" key, the user can take control of the player character and start walking, climbing, and jumping on the generated terrain (Fig. 12(e)). This state of the application is called "play mode". Another press on the "F5" key brings the user back to the "build mode". While in "build mode", the underlying interlocking building blocks are displayed as a semi-transparent overlay on top of the generated terrain and faded out when switching to "play mode". The main purpose of the toggling of the two modes is to enable the user to quickly iterate on their design by tangibly working on their world layout and testing it in-game.



(a) Physical block configuration. (b) Digitized block configuration. (c) Generated game world (with overlay).



(d) Generated game world (without overlay). (e) Playable game world.

**Figure 12:** *Different steps of the tangible world-building process from the physical block configuration (a) to the playable game world (e).*

## 6. Limitations and Future Work

One potential problem of our approach is that the wire connections between two interlocked blocks are not closing robustly. However, we found that this happens rarely and in most cases did not matter, because the remaining closed connections were sufficient to completely reconstruct the building block structure without ambiguities. In most practical use cases, blocks are connected over several nubs directly and indirectly, which allowed for an unambiguous reconstruction even with some of the connections getting lost.
Due to a small voltage drop over an internal diode of the ATMega328P, the number of building blocks that can be chained together is not limitless. For very large networks, additional hardware for voltage management would be required.
The proposed system aims to be intuitive and easy to use and, due to its toy-like nature, could be employed for a variety of applications. Besides the two end-user scenarios shown in Section 5.1, our system could be used for educational applications, or even as a modeling tool for the blind, because it does not necessarily require vision and provides haptic feedback. In the future, we would like

to develop such applications and software solutions to be used with our interlocking building block system.

## 7. Conclusion

In this paper, we propose a system for digitizing interlocking building blocks to create a reliable, easy-to-use, three-dimensional, tactile, human-computer interface. The proposed system does not use image-based techniques and constructs the true 3D topology of its block network. Our low-cost hardware fits into a 4 x 2 interlocking LEGO®-sized building block and consists of only a microcontroller and a single capacitor. Our proposed host application and micro-controller code allow to obtain the digitized shape of the connected interlocking block network for complex topologies (and not only planar configurations as provided by LEGO® Fusion). As we rely on the existing LIGHT STAX® housing and its two-wire connection, power supply and communication must alternate over time.

## References


[AFM*00] ANDERSON D., FRANKEL J., MARKS J., AGARWALA A., BEARDSLEY P., HODGINS J., LEIGH D., RYALL K., SULLIVAN E., YEDIDIA J.: Tangible interaction graphical interpretation: A new approach to 3d modeling. *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics* (05 2000). doi:10.1145/344779.344960.

[AIH*14] ANDO M., ITOH Y., HOSOI T., TAKASHIMA K., NAKAJIMA K., KITAMURA Y.: Stackblock: Block-shaped interface for flexible stacking. *UIST 2014 - Adjunct Publication of the 27th Annual ACM Symposium on User Interface Software and Technology* (10 2014), 41–42. doi:10.1145/2658779.2659104.

[BLCCL20] BOURDEAU S., LESAGE A., COUTURIER CARON B., LÉGER P.-M.: *When Design Novices and LEGO® Meet: Stimulating Creative Thinking for Interface Design*. Association for Computing Machinery, New York, NY, USA, 2020, p. 1–14. URL: https://doi.org/10.1145/3313831.3376495.

[Bou97] BOURKE P.: Implicit surfaces - also known as "metaballs", "blobbies", "soft objects".

[Bri21] BRICKIT: Brick it app. App, July 2021. URL: https://brickit.app/.

[CGKC20] CAVAS B., GUENEY L. O., KARAGOEZ E., CAVAS P.: More than playing a toy: The effects of lego mindstorms on the students' perceptions about scientists. URL: http://www.icaseonline.net/journal/index.php/sei/article/view/202, doi:https://doi.org/10.33828/sei.v31.i1.10.

[Cha17] CHANDRASEKARAN C.: Computational principles and models of multisensory integration. *Current Opinion in Neurobiology 43* (2017), 25–34. Neurobiology of Learning and Plasticity. URL: https://www.sciencedirect.com/science/article/pii/S0959438816302227, doi:https://doi.org/10.1016/j.conb.2016.11.002.

[CMRB12] CHAN L., MÜLLER S., ROUDAUT A., BAUDISCH P.: Capstones and zebrawidgets. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems-CHI'12* (2012), vol. 2189.

[Ent17] ENTHALER S.: Illustrating plastic production and end-of-life plastic treatment with interlocking building blocks. *Journal of Chemical Education 94*, 11 (2017), 1746–1751. URL: https://doi.org/10.1021/acs.jchemed.6b00888, arXiv:https://doi.org/10.1021/acs.jchemed.6b00888, doi:10.1021/acs.jchemed.6b00888.

[Gau15] GAUNTLETT D.: *The LEGO System as a tool for thinking, creativity, and changing the world*. Making Media Studies: The Creativity Turn in Media and Communications Studies. Peter Lang, New York, 2015. URL: https://davidgauntlett.com/complete-publications/.

[GCM05] GOLDSTEIN S., CAMPBELL J., MOWRY T.: Programmable matter. *IEEE Computer 38* (01 2005), 99–101. doi:10.1038/35036656.

[GFCC12] GUPTA A., FOX D., CURLESS B., COHEN M.: Duplotrack: a real-time system for authoring and guiding duplo block assembly. In *Proceedings of the 25th annual ACM symposium on User interface software and technology* (2012), pp. 389–402.

[GOI98] GORBET M. G., ORTH M., ISHII H.: Triangles: Tangible interface for manipulation and exploration of digital information topography. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (USA, 1998), CHI '98, ACM Press/Addison-Wesley Publishing Co., p. 49–56. URL: https://doi.org/10.1145/274644.274652, doi:10.1145/274644.274652.

[Gon18] GONICK M.: Lab on a lego. MIT Project, Jan. 2018. URL: https://news.mit.edu/2018/microfluidics-lego-bricks-0131.

[ILBL12] ISHII H., LAKATOS D., BONANNI L., LABRUNE J.-B.: Radical atoms: Beyond tangible bits, toward transformable materials. *Interactions 19*, 1 (jan 2012), 38–51. URL: https://doi.org/10.1145/2065327.2065337, doi:10.1145/2065327.2065337.

[IS18] IKEGAWA K., SHIZUKI B.: Tesla blocks: Magnetism-based tangible 3d modeling system using block-shaped objects. In *Proceedings of the 30th Australian Conference on Computer-Human Interaction* (New York, NY, USA, 2018), OzCHI '18, Association for Computing Machinery, p. 411–415. URL: https://doi.org/10.1145/3292147.3292221, doi:10.1145/3292147.3292221.

[JB11] JOTA R., BENKO H.: Constructing virtual 3d models with physical building blocks. In *CHI'11 extended abstracts on human factors in computing systems*. 2011, pp. 2173–2178.

[KIK01] KITAMURA Y., ITOH Y., KISHINO F.: Real-time 3d interaction with activecube. In *CHI'01 extended abstracts on Human factors in computing systems* (2001), pp. 355–356.

[LCT*14] LIANG R.-H., CHAN L., TSENG H.-Y., KUO H.-C., HUANG D.-Y., YANG D.-N., CHEN B.-Y.: Gaussbricks: magnetic building blocks for constructive tangible interactions on portable displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2014), pp. 3153–3162.

[LEG14] LEGO: Lego fusion town master. App, June 2014. URL: https://www.apppicker.com/reviews/17887/lego-fusion-town-master-app-review-next-generation-lego.

[LRL17] LEEN D., RAMAKERS R., LUYTEN K.: Strutmodeling: A low-fidelity construction kit to iteratively model, test, and adapt 3d objects. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, 2017), UIST '17, Association for Computing Machinery, p. 471–479. URL: https://doi.org/10.1145/3126594.3126643, doi:10.1145/3126594.3126643.

[McS13] MCSHAN D.: Bryx: Lego + arduino + bluetooth smart. Kickstarter Project, Oct. 2013. URL: https://www.kickstarter.com/projects/380641742/bryx-iphone-controllable-arduino-programmable-bric.

[MD21] MELAKU S., DABKE R. B.: Interlocking toy building blocks as modules for undergraduate introductory and general chemistry classroom teaching. *Journal of Chemical Education 98*, 7 (2021), 2465–2470. URL: https://doi.org/10.1021/acs.jchemed.1c00001, arXiv:https://doi.org/10.1021/acs.jchemed.1c00001, doi:10.1021/acs.jchemed.1c00001.

[Mer09] MERRILL D.: Stiftables, toy tiles that talk to each other.

Product, Feb. 2009. URL: https://www.ted.com/talks/david_merrill_toy_tiles_that_talk_to_each_other.

[MI18] MELCER E. F., ISBISTER K.: Bots and (main)frames: Exploring the impact of tangible blocks and collaborative play in an educational programming game. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2018), CHI '18, Association for Computing Machinery, p. 1–14. URL: https://doi.org/10.1145/3173574.3173840, doi:10.1145/3173574.3173840.

[Mic15] MICRODUINOSTUDIO: Microduino mcookie: The smallest electronic modules on lego®. Kickstarter Project, Sept. 2015. URL: https://www.kickstarter.com/projects/microduino/microduino-mcookie-the-smallest-electronic-modules.

[MSGD16] MELAKU S., SCHRECK J. O., GRIFFIN K., DABKE R. B.: Interlocking toy building blocks as hands-on learning modules for blind and visually impaired chemistry students. *Journal of Chemical Education 93*, 6 (2016), 1049–1055. URL: https://doi.org/10.1021/acs.jchemed.5b00252, arXiv:https://doi.org/10.1021/acs.jchemed.5b00252, doi:10.1021/acs.jchemed.5b00252.

[MWC*12] MILLER A., WHITE B., CHARBONNEAU E., KANZLER Z., LAVIOLA JR. J. J.: Interactive 3d model acquisition and tracking of building block structures. *IEEE Transactions on Visualization and Computer Graphics 18*, 4 (2012), 651–659. doi:10.1109/TVCG.2012.48.

[OH18] OWENS C. E., HART A. J.: High-precision modular microfluidics by micromilling of interlocking injection-molded blocks. *Lab Chip 18* (2018), 890–901. URL: http://dx.doi.org/10.1039/C7LC00951H, doi:10.1039/C7LC00951H.

[Osm20] OSMO: Osmo coding-bricks. App, Aug. 2020. URL: https://www.playosmo.com/en/shopping/bundles/coding-family/.

[PT19] PARKER R., THOMSEN B. S.: Learning through play at school: A study of playful integrated pedagogies that foster children's holistic skills development in the primary school classroom. URL: https://research.acer.edu.au/learning_processes/22/.

[PTB22] PARKER R., THOMSEN B. S., BERRY A.: Learning through play at school – a framework for policy and practice. *Frontiers in Education 7* (2022). URL: https://www.frontiersin.org/article/10.3389/feduc.2022.751801, doi:10.3389/feduc.2022.751801.

[RMSS96] RESNICK M., MARTIN F., SARGENT R., SILVERMAN B.: Programmable bricks: Toys to think with. *IBM Systems Journal 35* (02 1996), 443 – 452. doi:10.1147/sj.353.0443.

[SG06] SCHWEIKARDT E., GROSS M. D.: roblocks: a robotic construction kit for mathematics and science education. In *Proceedings of the 8th international conference on Multimodal interfaces* (2006), pp. 72–75.

[SIW*02] SHARLIN E., ITOH Y., WATSON B., KITAMURA Y., SUTPHEN S., LIU L.: Cognitive cubes: a tangible user interface for cognitive assessment. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (2002), pp. 347–354.

[STA14] STAX: Light stax led. Product, 2014. URL: https://www.light-stax.de/.

[Wes15] WESTERDIEP A.: Legolizer; custom lego objects. Application, June 2015. URL: https://drububu.com/miscellaneous/legolizer/index.html.

[WIA*04] WATANABE R., ITOH Y., ASAI M., KITAMURA Y., KISHINO F., KIKUCHI H.: The soul of activecube: implementing a flexible, multimodal, three-dimensional spatial tangible interface. *Computers in Entertainment (CIE) 2*, 4 (2004), 15–15.

[Yta15] YTAI: Electro-legos. Home Project, May 2015. URL: http://ytai-mer.blogspot.com/2015/05/electro-legos.html.

[Zha19] ZHANG Y.: Three-dimensional-printing for microfluidics or the other way around? *International Journal of Bioprinting 5* (07 2019). doi:10.18063/ijb.v5i2.192.

## 8. Appendix

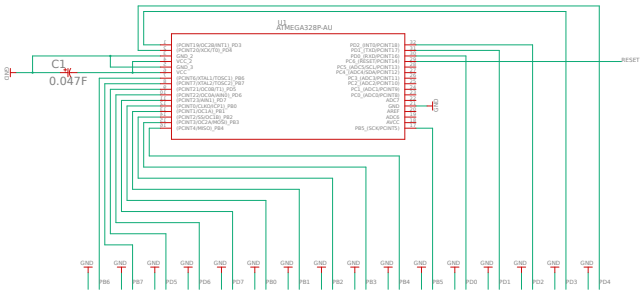
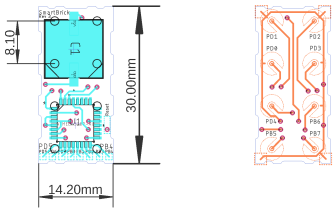


**Figure 13:** *Schematics of our building block hardware. The hardware only consists of the ATMega328P micro-controller and a 0.047F capacitor. The lower part of the figure the connections from the IO pins to the nubs' two-wire connections are shown.*




(a) top (b) bottom

**Figure 14:** *Our PCB design. The top (a) shows a mount for the micro-controller and capacitor with all required wire traces. The bottom (b) shows the bottom pads and all connecting wire traces.*