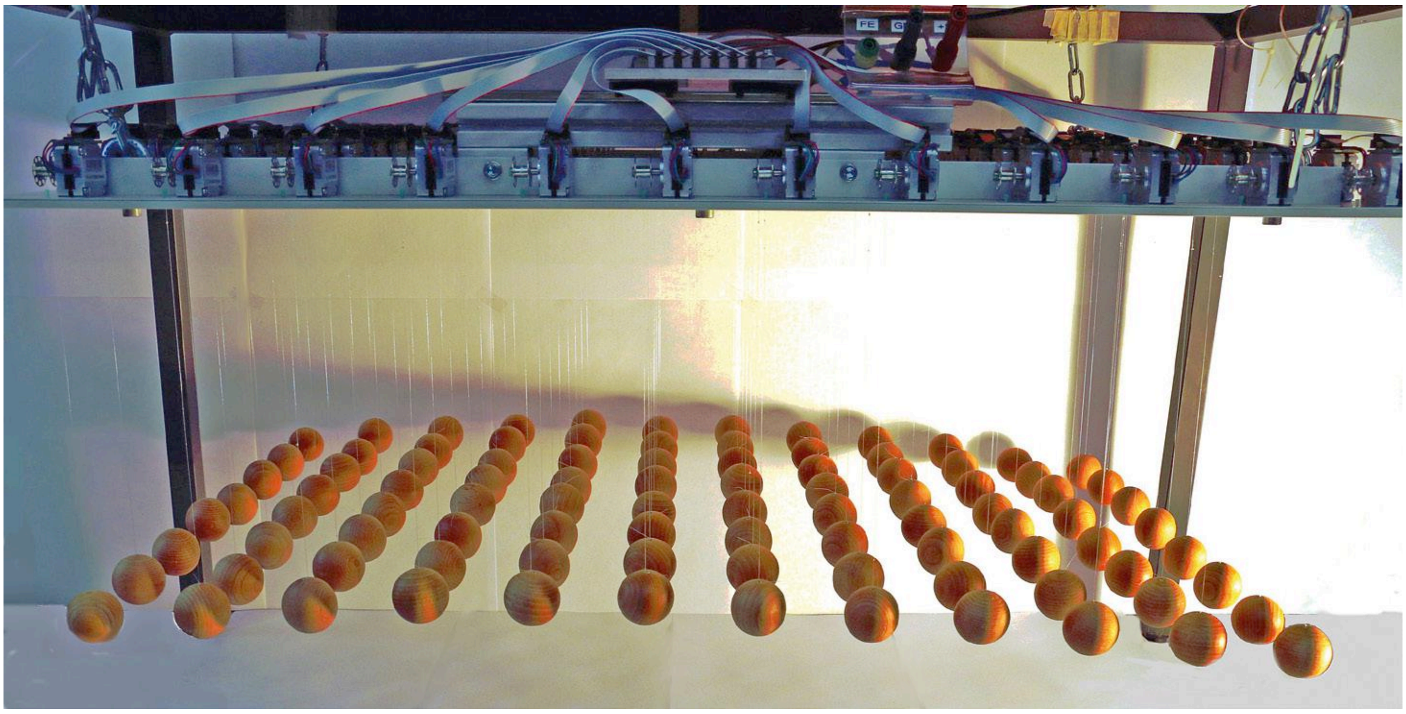
Kinetic Sculpture

Where Art meets Science...



By Andreas Lambers and Prof. Michael Bragard (Germany)

When the world of ones and zeroes teams up with the boundless imagination of artistic expression something magical happens. Take for example a sculpture that uses motors to change its shape. At Aachen University of Applied Sciences such a project was realized: A matrix of wooden spheres seemingly hovering in space can be programmed to display pictograms or visualize mathematical functions. In this article we describe how you can make the system.

A meeting between artist and engineer has taken place right behind the cash desk area of the BMW Museum in Munich where a large-scale "kinetic sculpture" is on display. This sculpture consists of 714 silver balls arranged in a matrix and suspended by thin threads connected to hidden motors in the ceiling which raise and lower the balls in a coordinated way to form 3D outlines [1]. At the museum the balls move seamlessly to form one shape after another, conveying a narrative representing the processes involved in building a car.

The effect is quite beautiful and here we show you how you can create a similar installation in your own living room. The project is the result of a Bachelor thesis at the Aachen University of Applied Sciences (Department of Electrical and Information Technology).

Hardware overview

Our kinetic sculpture consists of a matrix of 12 x 8 wooden balls. Compared to the shiny BMW version ours looks a little dull but the balls are at least from a sustainable source. The system is nevertheless very flexible and a great platform to try out your own designs and extensions. Each ball is attached to a barely visible nylon thread, wound on sewing machine bobbins, attached to stepper motor shafts. The individual drive unit modules are mounted on a plate in a grid pattern with constant pitch. As you can probably make out from **Figure 1** there are 12 rows of motors, each row has one ribbon cable that plugs into the central board called the hub unit. For the first test run, as the title picture shows, the plate with the stepper motors is suspended from a structure made from a table frame so that the balls appear to float freely in the room.

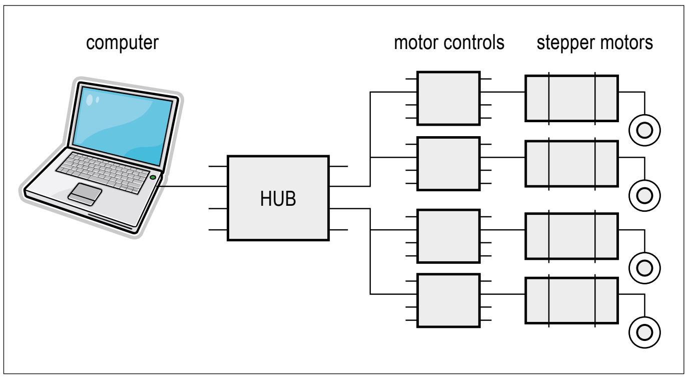


Figure 1. A hybrid 'mixed-star' network topology.

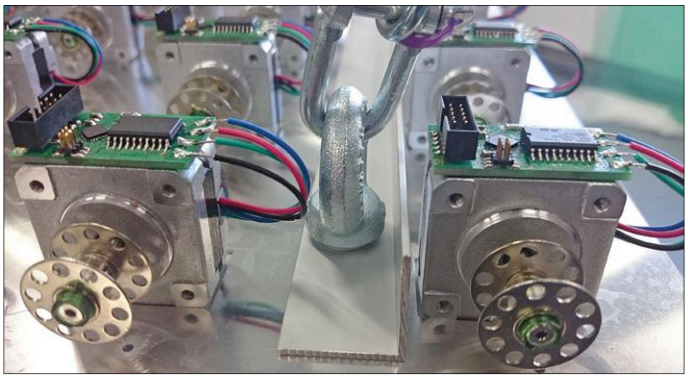


Figure 2. Stepper motor and control board.

Stepper motor selection

Stepper motors are the ideal drive units for this project because they are comparatively easy to control and they have good availability at relatively low costs. The choice of motor is an important factor, the available torque; power consumption and motor mounting options are important considerations. The NEMA 14 design is ideal and makes mounting relatively easy. After some research, we went for the stepper motor type PSM-35BYG104 (**Figure 2**), it's a common motor type and you can often find them offered at a good price. Preliminary tests showed that the available detent or no-current torque of 0.5 Ncm will be sufficiently large. This parameter is particularly important because if it's too low, without power to the motors all the balls would unwind to the bottom of their travel. In combination with a bobbin diameter of less than one centimeter, the weight of the ball is about 100 g. The selected motor has 200 physical steps, that is, it achieves a resolution of 1.8° in full-step operation. This corresponds to a positioning accuracy of approximately 0.1 mm with the selected mechanics!

Decentralized motor control

As shown in Figure 2, a small controller board is mounted on each stepper motor. This receives control information signals from the central hub and incorporates a power output stage to drive the motor.

The control board circuit (**Figure 3**) consists of an ATmega88-20MU microcontroller [2] to handle communications and also to provide drive signals to the popular push-pull four channel full-bridge driver chip L293DD [3] via PC0 to PC3 (IN1 to IN4) pins. This driver IC contains two pairs of drivers which connect to (i.e. in the middle of the bridge) the bipolar stepper motor windings. The microcontroller generates a switching sequence that moves the rotor forward or backward step by step. The microcontroller can also switch the bridge drivers on and off independently via PD6 and PD7 (EN1, EN2).

The L293DD is operated here with 5 V supply voltage for the logic and output driver stage. The output can be supplied from an external 12 V and intermediate connectors would need to be fitted to K2 pins 1 and 2 to make use of this facility. This higher supply is not possible with the motor used here however. The driver stage operates at up to 1 kHz clock rate for the desired step sequence. At 5 V, a maximum current of 600 mA is possible, which can be permanently maintained. This is more than sufficient for the motor used here. The shunt resis-

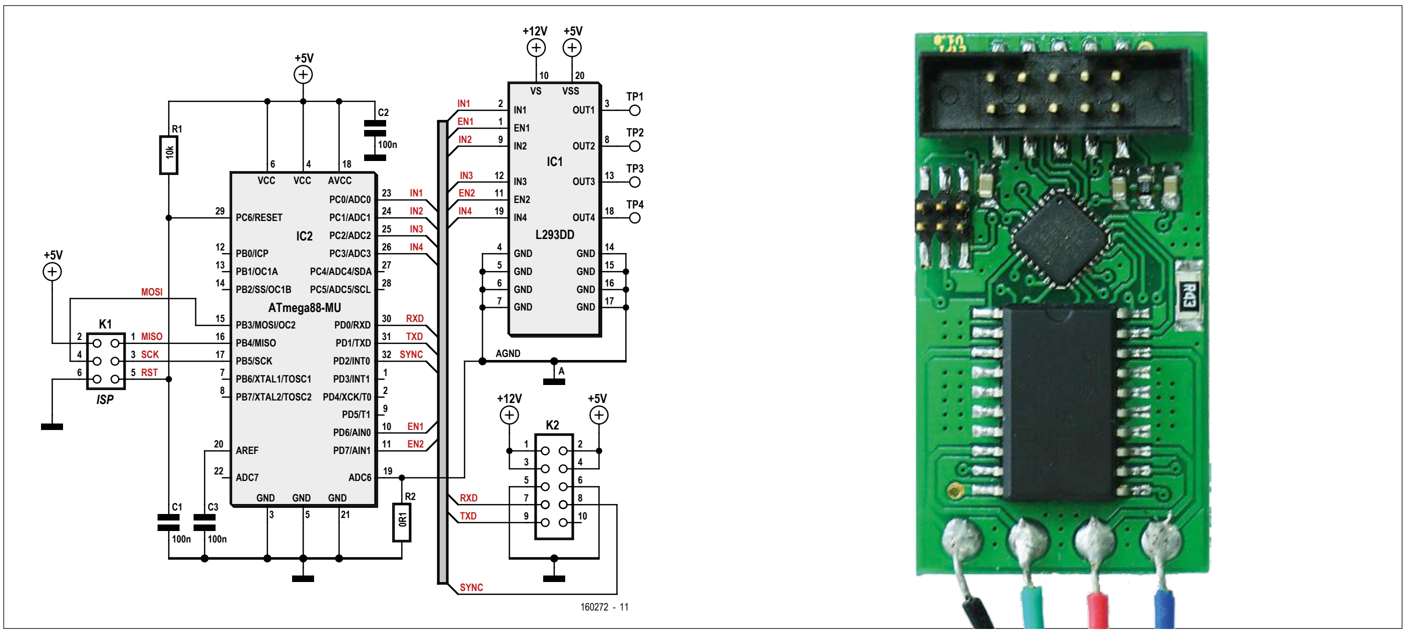
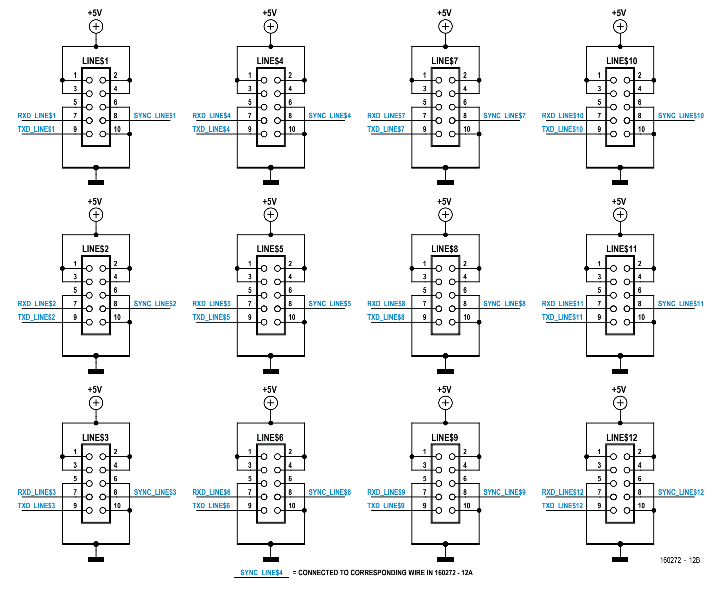


Figure 3. Circuit and layout of the stepper motor board.



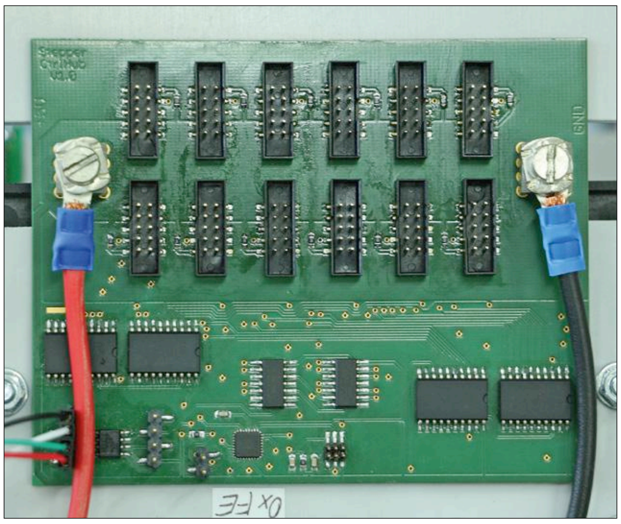


Figure 4. The hub unit PCB has connections for the rows of motor in the upper half.

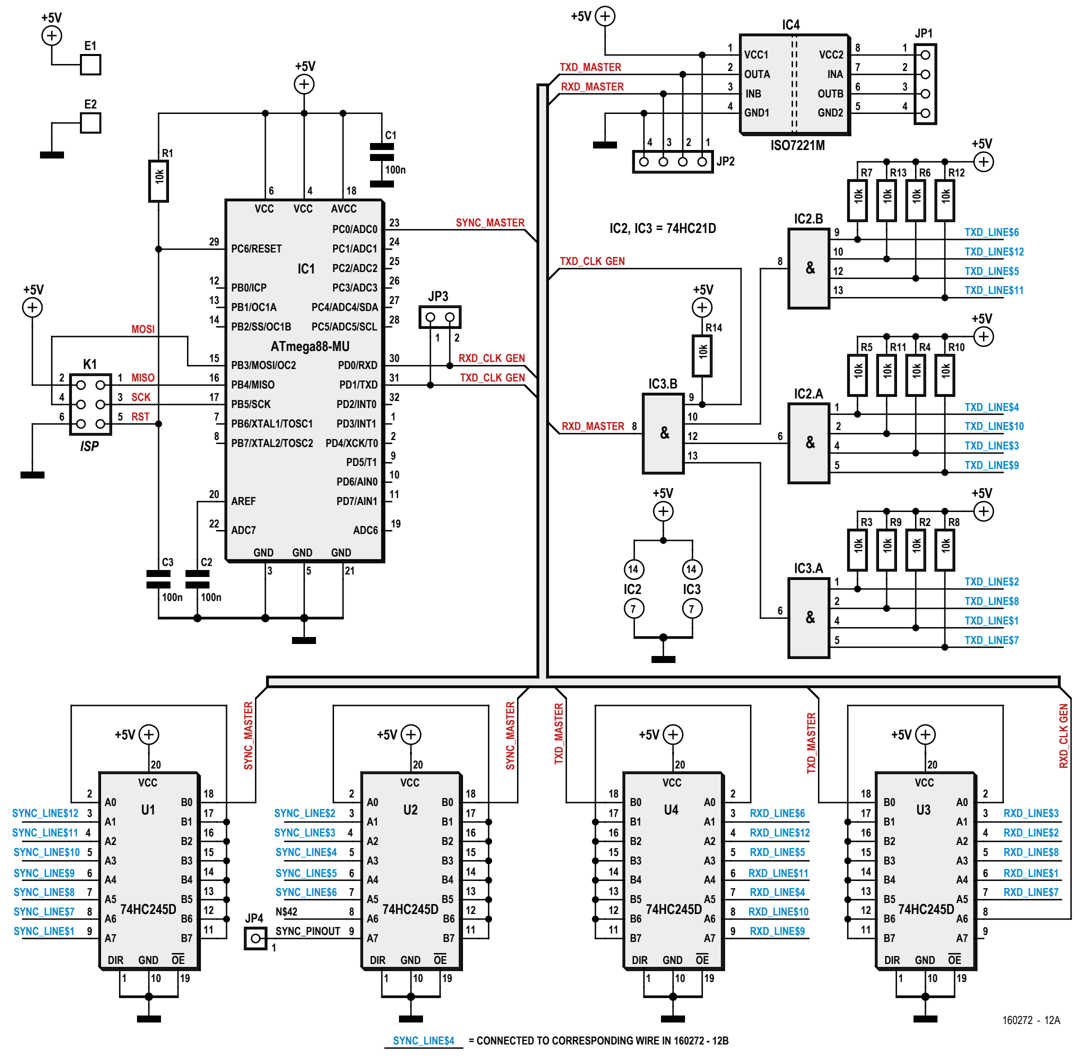


Figure 5. Hub communications unit schematic.

tor R2, connected at the ADC inputs of the controller, is used to measure the current flowing to the driver chip. The comparison of the ADC input voltage with the controller's internal reference gives a good approximation of the current flowing, when it exceeds a limit value the motor is turned off

Three signals are required for communications with the central hub. The SYNC signal on pin 32 (PD2) is generated by the hub unit and ensures all the motor movements are synchronized. The communication paths RxD and TxD, individual for each motor controller, connects to the serial data pins of the controllers UART (pin 30 and pin 31) for communication with the central hub unit.

The remaining components on board are for supply decoupling and smoothing the analog reference (AREF) voltage. A power-on reset (PoR) is also provided by R1 / C1. The small 6-pin DIP pin header is the controller programming interface and has the usual AVR-ISP pin assignment.

The hub, the central communications unit

At the heart of the communication system is the hub unit which acts as the bus master and guarantees synchronicity of all the individual motor movements. The hub unit also distributes power to the individual motors and their controlling electronics. Because this particular network topology has a hybrid mixed-star structure [4], the hub represents a system bottleneck in terms of reliability and operational stability.

Figure 4 shows that the power and communication paths are clearly separated from each other on the board. This spatial separation ensures that any possible interference is kept to a minimum. In order not to drown in a mass of cables, eight motor control units in each row are connected in parallel to one another with a single flat band cable. For each of the 12 rows of motors there is one plug (2 x 5-pin header plug) for the ribbon cable connector. Connections for an external power supply are located to the right and left of the 12 pin-header connector area. The total maximum current of all motors is in the range between 15 A and 20 A. For this reason we have used special high-current terminals from Würth [5] and on the PCB layout copper polygons are used to maximize the cross-sectional area. The total current for the motor modules is also always divided over several pins of the connectors.

Along the lower section of the hub unit PCB is the controller and communication section, the circuit is given in **Figure 5**. It is the link between the external controlling PC, which connects via JP1 and relays the signals TxD_MASTER and RxD_MASTER to the individual stepping motors controllers. Here, too, an ATmega88 in the MLF32 package has the same standard circuitry as on the motor controller modules, the power available from its I/O pins is not enough to supply all the motor control modules with the SYNC signal. Therefore, two separate transceiver chips (74HC245) are used to drive the SYNC_MASTER signal to all of the twelve rows of motor controllers. The outputs of these transceivers on their own provide enough current to distribute the signal without the need for additional drivers. Not all of the transceiver outputs are used, one spare output is sent to a test pad (it could come in useful).

A similar signal distribution system is used for the TxD_MASTER signal which comes in at JP1. In this case IC4 can only drive a few milliamps, nowhere new enough for the 96 motor control units. Here too, two octal transceiver chips distribute the signal to twelve outputs for the motor controllers (plus the RXD_CLK GEN signal routed to the ATmega in the hub).

The TxD signals from the stepper motor controller module rows are inputs and need to be handled differently in the hub. The twelve lines are pulled high to 5 V using pull-up resistors. If a control module sends data to the hub, it pulls the line to a low voltage which signals a logical 1 (i.e. we are dealing with a negative logic). In order to use only one input line of the ATmega in the hub, logic gating routes the signal from the currently transmitting module onto the one RX line of the ATmega. The controller in the hub functions as a controllable clock and a message sent by any control module is channeled through the hardware logic. This allows for a large number of subscribers to communicate using just one input and one output line for communication between the PC and the hub.

The signals TxD_Clock_Gen and RxD_Clock_Gen available at JP3 are useful for initial controller setup so that the logic can be bridged and measurements of the transmission signals before and after the logic can be more easily made. We haven't touched on IC4 yet; the ISO7221 is a fast two-channel bus isolator in the communication path between the PC port and the hub electronics. This provides good electrical isolation between the communication and supply lines. Alternatively the PC can be connected at header JP2, bypassing the isolator.

The idea for control

The PC and Hub are connected using a USB-to-TTL converter cable [6]. Communication can then take place via the UART using a protocol with eight data bits, even parity and a stop bit. For test purposes, the baud rate is initially limited to 600 baud (600 8E1). A message to a motor control board uses the following format:

```
Address  Data0  Data1  Data2  Data3  Data4  Data5  ID
```

Instructions for movement and control (example).

	Data0	Data1	Data2	Data3	Data4	Data5
Move x steps	0xA0	Speed	x Steps High-Byte	x Steps Low-Byte	Direction	-
Move to position x	0xA1	Speed	Position x High-Byte	Position x Low-Byte	-	-
Move using binary data	0xA4	Binary step instruction 1	2	3	4	Direction
Stop movement	0xB0	-	-	-	-	-
Stop movement and switch off motor current	0xB1	-	-	-	-	-
Continue on to stop	0xB2	-	-	-	-	-

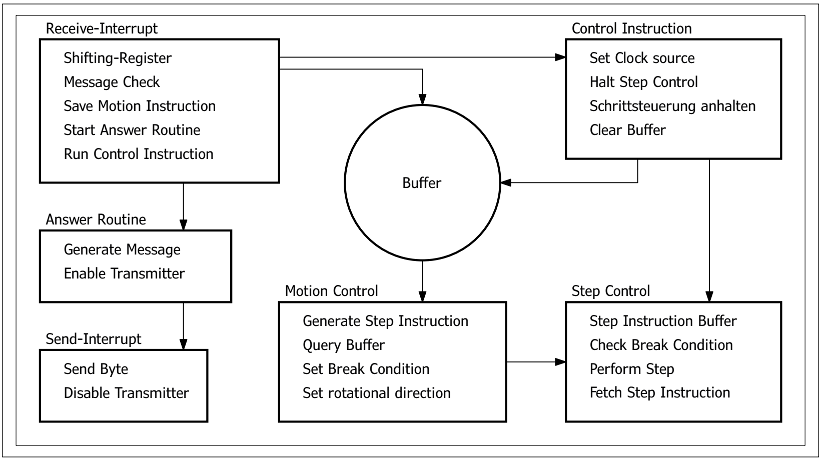


Figure 6. Hub-Firmware structure.

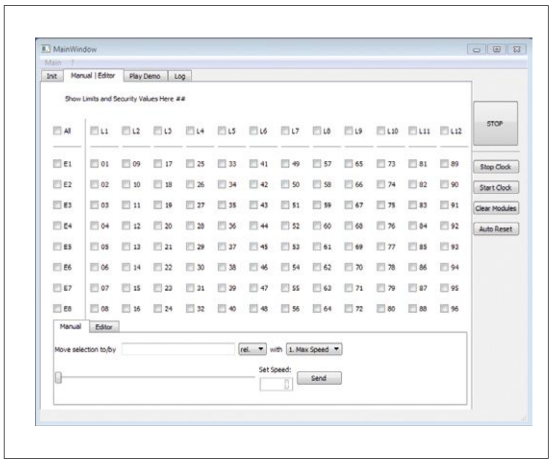


Figure 7. Simple to use, the PC software GUI.

```
   CRC  STOP
```

With an 8-bit address field we can uniquely assign 254 addresses, the broadcast addresses 0x00 and 0xFF are not available for use to send individual commands. Each message is bytewise checked for parity errors and a possible overflow. These functions are already implemented in the microcontroller hardware so it's only necessary to read the appropriate status register to detect possible errors.
After the message has been successfully checked, it is decided whether it is a movement instruction or a control instruction. Control statements are executed directly, movement instructions are first written into the memory. This method ensures that all motors can be preloaded with the movement information and then triggered to run at the same time when the software sends the command.
After the master has sent a message, the status of the individual motor controller is queried. If a slave does not answer, the message is sent again. The slave response is a message of nine bytes:

```
Address  Receiver ID  Memoryspace  Pos.H  Pos.L
   Clock H  Clock L  CRC  STOP
```

In addition to its own address, the last received message ID and the currently free memory space, the high and low bytes of position information, current clock are transmitted. Error detection is carried out by calculating the CRC value from all the previous bytes according to the CCITT/ITU-T standard. The stop byte ends the message.
The control and movement information referred to above contain all necessary data to produce a coordinated flowing motion of the sculpture. Some of the commands are explained in **Table 1**. Additional commands can be easily implemented in the firmware to meet specific application requirements. A simple test routine, in the form of a sine wave chart using pre-calculated instructions is stored and contains all the calculated values for each motor controller, to represent a sinusoidal motion on the sculpture. Without any additional programming, you can, for example just run this movement profile to check current consumption, synchronicity and rotation direction of each individual motor.

The hub firmware

The firmware running on the central hub microcontroller is interrupt driven. A receive interrupt triggers a routine which processes all incoming values and checks them. When the check is successful, the firmware determines whether a response to the message is required. If so, it is generated and transmitted by the Sender. When this is complete the Sender switches off again. The motion controller reads the commands stored in its buffer and converts them into step instructions for the stepper motor control. In addition, it sets the rotation direction and limit conditions.
The step control is triggered by the clock and takes care of the actual step; afterwards the limit conditions are checked. The sequence instructions are stored as binary data in the step memory so that the motion controller is less heavily loaded. The sequences of the firmware are shown in **Figure 6**.

The PC software

The PC software provides a convenient control and programming environment which is a useful addition to the system. The software allows you to send movement and control instructions to the hub unit as well as creating movement profiles. The message is made up of the respective instruction combined with the address, an identification number, a checksum and a stop bit. When only one message is sent, a response is expected immediately, this is then checked immediately upon arrival. If no response has been received within a specified waiting time, the original message is sent again until after a fixed number of attempts the operation is terminated. When sending several messages, a memory area is reserved for each message in order to buffer the individual responses for evaluation later on. When sending a sequence of instructions an ordered sorting of the messages is arranged according to address.
The software interface in **Figure 7** has been kept a little Spartan for the purposes of system testing but it can be easily adapted and extended as necessary. From the menu tabs along the upper edge of the display you can select manual control, use the editor demo application, select demo mode and also view a Logging function. The editor allows you to program a movement profile in the matrix for the entire sculpture. With this it's necessary to enter the height to which the balls are to travel and the speed at which this is to occur.

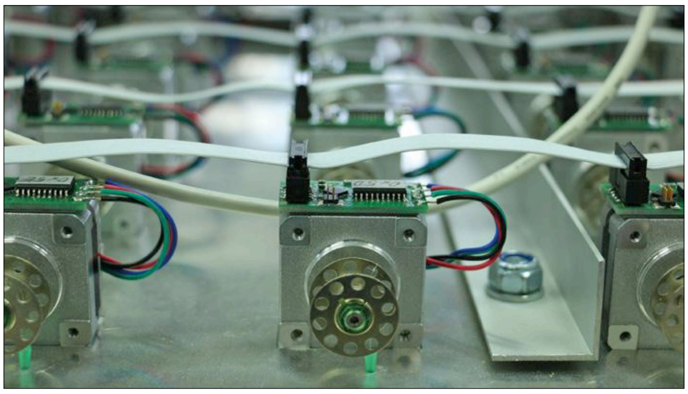
In Play Demo a pre-programmed pattern for test purposes is available; you can use it to quickly check the function of the sculpture. By monitoring the Log window, it's possible to see errors that have been detected directly via the software, without resorting to an oscilloscope to track them down.

Finally: switch on

At the beginning, a synchronization process is carried out, so that all the balls start out at the same height. All the motors drive the balls up to the stop. The stop position is detected in each motor by the current sensing shunt resistor which detects the increased current flow. Now the ball height can be defined using the step position accuracy of the motor.
In the Laboratory for Electric Drive Systems at Aachen University of Applied Sciences, the first trial functioned according to plan. The individual motors can all be made to move in synchronization with each other. In the meantime, initial tests to produce a programmed pattern have also been successfully completed.
Observing the entire system in operation with a thermal imaging camera showed there were no unexpected hotspots. The electrical design can therefore be described as stable. It remains to be seen how long the nylon threads will last in the long run. The free suspension (in the frame of an old table) is very quiet in operation.

Summary

The visit to the BMW Museum certainly left a lasting impression and was the inspiration for this project. The Aachen University of Applied Sciences has now gained an interesting work of art in the Electric Drive Systems laboratory, which will no doubt provide future potential for further dissertation work and is already a talking point for students and visitors.
Maybe it has inspired you also, everything you need to build your own version is given - circuit diagrams and layouts in Eagle format, software for all parts and even technical drawings of the carrier plate (it will only be suitable for the motor type specified) all available on the Elektor Project page [7]. As you dare to imagine a replica hovering above the coffee table in your living room, be warned! When it actually comes to assembling the same circuit board almost 100 times it sounds like you may need to call on the good will of some friends to help with the work and relieve the tedium. It may cost you a few beers...



(160272)

Web Links

[1] https://artcom.de/en/project/kinetic-sculpture/

[2] http://bit.ly/2jimvRu

[3] http://bit.ly/2jR13nv

[4] www.lawerence.de/map/struktur.php

[4] https://en.wikipedia.org/wiki/Network_topology (engels, hydrid topology)

[5] http://katalog.we-online.de/en/em/WP-BUTR_TWO-ROWS/7460307

[6] www.elektor.com/usb-ttl-interface-cable-5v

[7] www.elektormagazine.com/160272

Author profiles

Michael Bragard studied electrical engineering at the RWTH Aachen University in 2000. After completing his degree as a graduate engineer in 2006, he worked as a research associate at the Institute for Power Converters and Electric Drives. From 2012, he worked for Delta Energy Systems in the development of wind turbines. In January 2015, he was appointed professor in the 'Electric Drive Systems' department at Aachen University of Applied Sciences.

Andreas Lambers studied electrical engineering at the RWTH Aachen University and finished his studies of electrical engineering at the Aachen University of Applied Sciences as a Bachelor of Engineering. The topic of his thesis was the 'Development and commissioning of a scalable visualization platform using stepper motors with serial control'.