T. Fischer, K. De Biswas, J.J. Ham, R. Naka, W.X. Huang, *Beyond Codes and Pixels: Proceedings of the 17th International Conference on Computer-Aided Architectural Design Research in Asia,* 317–326.

DIGITAL AND PHYSICAL COMPUTING FOR INDUSTRIAL ROBOTS IN ARCHITECTURE

Interfacing Arduino with industrial robots

JOHANNES BRAUMANN AND SIGRID BRELL-COKCAN
Association for Robots in Architecture, TU Vienna, Vienna, Austria
johannes@robotsinarchitecture.org

Abstract. Customisation is one of the most important topics in architecture, as architects generally work on individual prototypes instead of mass-produced designs. By using customised design and fabrication tools, architects are able to individually respond to challenges, instead of relying on universal software tools. This paper proposes new software components for interfacing industrial robots with physical computing microcontrollers, thereby allowing the customisation of physical tools for industrial robots. By pairing physical computing with rapid prototyping, architects are able to design and prototype individual fabrication processes for industrial robots.

Keywords. Industrial robots; physical computing; interfaces; rapid prototyping; computer aided manufacturing.

1. Introduction

In recent years, architects have become interested in the topic of physical computing. Due to powerful and accessible integrated development environments (IDEs) such as Arduino, architects and designers are now able to prototype and program complex electrical circuits without the help of electrical engineers. A similar trend can be observed in the area of industrial robotics, where software written by architects is enabling a wider range of users to control robots out of affordable, architecture-centric Computer Aided Design (CAD) software. This development is not only driven by the desire to streamline the whole workflow or to reduce costs, but aims to free users from the constraints of current Computer Aided Manufacturing (CAM) software, by letting them

create their own parametric design and fabrication environments instead of having to rely on preset fabrication strategies.

However, being inherently multifunctional machines, industrial robots are very much dependent on their end-effectors, which often have to be custom-built by engineers for the robot's particular task. By using a physical computing platform for prototyping robotic end-effectors, architects and designers can create their own hardware tools for robotic fabrication and directly control the data-flow between the robot's control unit and any kind of external devices. Furthermore, this data flow can be reversed by attaching sensors to a microcontroller and sending the collected data to the robot, which then interprets it and reacts according to its programming. The following chapters will provide a brief introduction to industrial robots and physical computing and discuss existing work where non-standard robotic end-effectors have been used.

We will present a range of custom components for the parametric design software Grasshopper, capable of communicating with industrial robots as well as writing robot code for interfacing with microcontrollers. Using this interface, a KUKA KR5 robot can control custom end-effectors such as a 3D printed gripper and a modified airbrush spray gun.

2. Industrial robots

In the scope of this paper, the term "industrial robot" refers to robotic arms, or serial articulated robots, which are most commonly seen in the automotive industry where they are used for a wide variety of tasks such as spot-welding or spray-painting. This multi-functionality, which allows a robot to change from a milling machine to a 3D-scanner, just by switching its end-effector, along with its large working space and – compared with multi-axis Computer Numeric Control (CNC) machines – the affordable price, has sparked an interest of both the construction and creative industry in robotic arms.

In the past few years, architectural faculties at institutions such as the ETH Zürich, TU Vienna, University of Stuttgart and the University of Michigan have acquired industrial robots and are actively researching the use of robots in architecture. However, this research is not limited to academia, with architectural offices like Snøhetta in Norway using robots in-house.

The main challenge when dealing with industrial robots is to efficiently program their movements. Moving from CAD to a physical output involves a unidirectional workflow consisting of up to three software packages, in addition to the handling of the robot itself. (see Brell-Cokcan and Braumann 2010, Bechthold 2010). These issues encouraged the development of plugins such as KUKA|prc (Braumann and Brell-Cokcan 2011) and SuperKUKA (Pigram and McGee 2011), which integrate into software used for architectural design and

give users the possibility to plan and simulate the robot's movement within the CAD environment and to ultimately generate robot control data files that can be directly executed at the robot. By using these plugins, the user is no longer forced to work with predefined fabrication strategies, but can program individual toolpath scripts which are immediately transformed into robot control data by the robot-plugin. This approach has several advantages: A direct feedback loop when changes are performed to the initial geometry, allowing a more intuitive grasp of the action-reaction relationship, the possibility to automate the generation of robot control data for mass-customisation of elements and ultimately a much more affordable price compared to commercial CAM software.

However, the main goal of these plugins is to generate robot control data files that contain a sequence of positions – by default they do not create code that can interact with external sensors or actuators.

3. Devices and interfaces

It is necessary to find a common interface that can be accessed by the parametric design tool, the industrial robot and the microcontroller. In the scope of this paper, Grasshopper was chosen as the parametric design and robot control environment, Arduino as the platform for physical computing, and a KUKA KR5 as an exemplary industrial robot. Transmitting data between such different devices requires an interface to be as simple as possible, in order to reduce the margin for errors.

3.1. ARDUINO

The Arduino board is part of the Arduino environment, an "open source physical computing platform based on a simple input/output (I/O) board and a development environment that implements the Processing language" (Banzi 2008). Users write code in the well-documented Processing language and submit it via serial or USB port to the Arduino unit. Other interfaces like Ethernet are only available as hardware addons - the native and easiest way of communicating with an Arduino board is therefore via the RS232 serial port.

3.2. GRASSHOPPER

Grasshopper is built as a .NET application with groups of components contained in separate class libraries. The .NET framework includes several classes for interfacing with external devices, such as the SerialPort and UdpClient classes for serial and Ethernet communication.

3.3. INDUSTRIAL ROBOTS

All current control units for KUKA robots are based on common PC hardware. Therefore, they offer the regular range of connectivity, such as USB, Ethernet, serial, and parallel ports. However, while the underlying Windows operating system can access all these ports, not all of them are by default available to the robot application. Commonly used are only the Ethernet (Chinello et al. 2010) and the serial port (Maletzki et al. 2006). While the Ethernet port allows a more direct real-time control with less lag, it requires modifications to the control unit or additional technologies such as the RSI-XML Ethernet interface. On the other hand, the serial port can be configured within a single *.ini file and accessed via regular, well documented commands of the KUKA Robot Language (KRL), without requiring any expenses besides the RS232 cable.

3.4. SERIAL INTERFACE FOR INDUSTRIAL ROBOTS

We therefore conclude that a common interface for connecting parametric design software, Arduino microcontrollers, and an industrial robot in a low-tech/low-cost context has to be based on either the RS232 serial port or on Ethernet technology. According to Axelson (2007), serial communication adds only little extra information to each transferred byte, while Ethernet uses sophisticated protocols that define the format of transferred data. Aiming to keep both complexity and costs down, communication via the RS232 serial port (Figure 1) seems to be the best choice of interface.

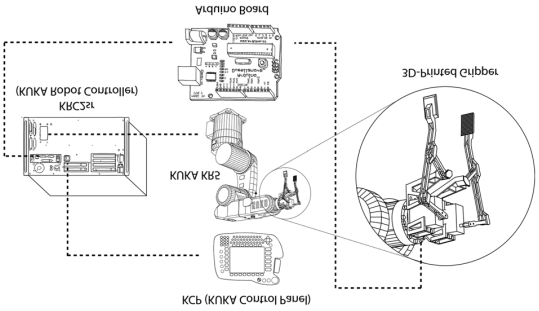

Figure 1. KUKA control unit sending data to the control panel, the robot, and an Arduino board, which controls a 3D-printed gripper.

4. Realised interfaces and non-standard end-effectors

The interfacing of external devices with PCs and industrial robots is mostly researched in electrical engineering and computer sciences, where these inter-

faces are used for robotic vision, pick-and-place systems etc. However, these systems are complex, specialised, and usually running on top of software such as MATLAB (Chinello et al. 2010), instead of design tools that are commonly used in architecture. The following overview is therefore limited to projects within an architectural context.

4.1. INTERFACES

In previous research, we developed KUKA|prc, a plugin for Grasshopper that is capable of creating robot control data files for KUKA robots (Braumann and Brell-Cokcan 2011). While a full kinematic simulation of the robot´s toolpaths is possible, this plugin does not directly communicate with the robot, as the control data files have to be sent to the robot before they can be executed.

Previous work on the topic of linking parametric design and physical computing has been performed by Andrew Payne and Jason Johnson who developed Firefly, a plugin that allows direct communication between the parametric environment and Arduino boards. Payne (2011) also presented a robotic motion controller for designers – basically a 5-axis digitiser with the proportions of an industrial robot, which sends its posture in real-time to Grasshopper, where it is turned into robot movement commands.

4.2. END-EFFECTORS

Standard end-effectors for industrial robots are e.g. gripper, milling spindles, and welding equipment. While these end-effectors have been involved in innovative projects in architecture, they are not innovative by themselves, as they have been used in similar form by the automation industry for many years. In the past years, the University of Michigan published several projects involving customised robot-hardware (Figure 2).

Figure 2. Robotic rod-bending (left), hotwire-cutting (middle), foam-depositing (right).

One such example is the wavePavilion by MacDowell and Tomova (2011), which was constructed out of three-dimensionally bent rods. Using a custom-

ised gripper, the robot placed the steel rods with the right position and rotation inside a custom CNC rod-bending device.

Several other end-effectors were also prototyped, such as a 2-dimensional roller cutter, or a foam depositing device, which was used to perform large-scale additive manufacturing. For the Periscope Tower, a hotwire cutter was mounted onto the robot, creating a multi-axis CNC hotwire-cutter capable of processing 4 m long EPS blocks (Pigram and McGee 2011).

None of the projects listed above mention the interface responsible for communication between the industrial robot and the end-effector. It therefore has to be assumed that industry-standard technology from automation such as a fieldbus protocol was used, requiring proprietary controllers.

5. Bidirectional robot control

State of the art IDEs allow architects to program software and control applications without knowledge from automation. Instead of taking fieldbus courses aimed at robot technicians, designers can duplicate the relevant part of the functionality in the Processing language and send it to the Arduino board, which then manages the communication between the device attached to its serial port (i.e. the industrial robot or a PC) and its actuators and sensors. Similarly, using environments such as Microsoft Visual Studio, architects can build their own interactive design-tools.

The software developed for this paper is a range of four custom components for Grasshopper, integrated into the KUKA|prc plugin. They are written in C# using Visual Studio, the Grasshopper SDK and the RhinoCommon framework. These components can be divided into two groups: Real-time components that allow the user to *send* position and other information via the serial port in (near) real time from the parametric environment to the industrial robot, as well as *receive* data in real-time from the robot. The second group is movement planning components which create robot toolpath similar to KUKA|prc, but insert additional information that operates external devices by sending data from the robot's serial port to the Arduino board (see Section 6.2).

5.1. REAL TIME COMPONENTS

Three components belong to the group of real-time components (Figure 3c): First, there is the *Serial Setup* component. After setting the correct COM port and baud-rate, the component opens a connection to the serial port and locks it for all other applications. Furthermore, a graphical user interface (GUI) is available, where the user has to set the robot's local coordinate system, the mounted tool and the robot's serial port. Using this information, the component can generate three types of robot control data files: One that prepares the

robot for receiving Cartesian position data in XYZABC format, one for receiving axis rotation values in A1–A6 format, and finally one control data file that continually transmits the robot's current axis rotation values to Grasshopper.

If the program running in the robot's control unit is transmitting its current axis values, the *Serial Receiver* component in Grasshopper can access the data from the serial port and send it to a forward-kinematic solver which instantly shows the robot's current posture inside the CAD environment. Therefore, the parametric environment is always aware of the current position of the industrial robot and can e.g. disable/enable devices whenever the robot enters a defined area of its workspace. The *Serial Send* component is used whenever the robot is set to receive position data via the serial port. This component accepts either the coordinate frame of the end-effector or the rotation values of all six axes and sends that data in either XZYABC or A1A2A3A4A5A6 format via a serial connection to the robot. Once the robot has arrived at the transmitted position, it stops until it receives the next set of position values. Simulation inside the parametric environment is possible by connecting the position data to the kinematics solver.

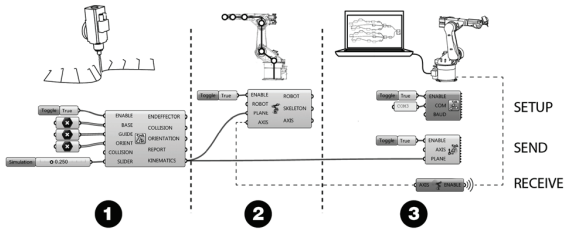

Figure 3. Parametric real time control workflow: Toolpath generation (a), forward/inverse kinematic solver (b), Serial Send/Receive components (c).

5.2. OFFLINE COMPONENT

The fourth component works differently, as it does not send data in real-time, but generates robot control data files similar to KUKA|prc, but containing *both* movement and serial commands (see Figure 5). As an input, it accepts lists of curves, representing tool movements, which get divided into three sets of points. Each triplet of points can then be expressed in XYZABC and written as a movement command into the KRL file. In the GUI the user chooses a start command and an end command which is sent from the robot to the Arduino at the beginning and end of each movement. This process can be used for a wide variety of tasks, such as welding, additive manufacturing or spray painting, where a device has to be continually enabled and disabled in between tool paths.

5.3. SCALABILITY OF CODE

It is important to note, that the components outlined above are not unique to the KUKA KR5, but generate code that is compatible with nearly all KUKA robots. This "scalability" of code is one of the greatest advantages of robots compared to conventional CNC machines, as it ensures that code written for a compact KUKA robot can also be executed on the massive heavy-lifting KUKA Titan. This enables the concept of "fabrication prototyping", where fabrication processes can be tested on small scale robots, before moving on to full scale production.

6. Exemplary Applications

6.1. REALTIME CONTROL IN TEACHING

The realtime components described in Chapter 5 were used to realise a near real-time control of the robot from a touchscreen device. This was done to provide students with a more direct interface to the robot's kinematics, as they feel much more comfortable with a tablet device than with the intimidating KUKA control panel. TouchOSC, a commercial iPhone-app for DJs was adapted to provide six numeric sliders, one for each axis, which provides the input for joint coordinate programming. These values are sent over the wireless network and captured by a Grasshopper component. Another small custom component then divides the data back into six axis values and transmits them to the Serial Send component, which passes them with a checksum to the robot. While the lag is considerable and only one axis can be moved per transmission, students were able to much more quickly get a grasp of the robot's kinematics.

6.2. ARDUINO-CONTROLLED END-EFFECTORS

Our Arduino-control component was tested with custom-built end-effectors at a two-day robot workshop with both professional architects and students.

For a pick-and-place application, we developed our own, kinematic gripping device in the CAD software Rhinoceros and had it prototyped for 80USD via selective laser sintering (Figure 4, left). The physical model was then equipped with two servo motors, mounted on the flange, connected to an Arduino board and put into action. As programmed in the Grasshopper definition, the robot moved to the first point, sent a serial command containing the desired position of the servo motors for closing the gripper (e.g. "40") and then moved via a safety distance to the end point, where the gripper released ("0") the object again. A second iteration of the gripper, constructed out of

laser-cut acrylic class and again equipped with two servo motors, proved to be both cheaper and more robust than the SLS version (Figure 4, right).

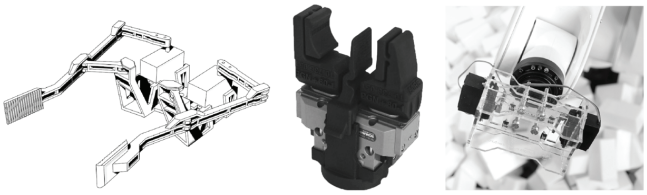

Figure 4. 3D gripper model for rapid prototyping (left), commercial gripper with 3D-printed jaws by Schunk (middle), acrylic-glass gripper mounted on KR5 (right).

Robotic graffiti was performed with the tool path generation component. By hacking an airbrush compressor it was possible to enable and disable the compressed air using a relay and an Arduino board. At the start of each spray painting tool path, the Grasshopper-generated KRL file sent a serial command to the Arduino board, enabling the compressed air, and at the end shut it down via another serial command (Figure 5).

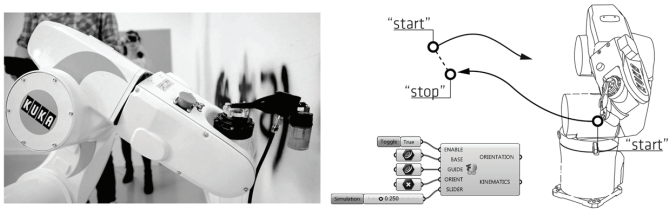

Figure 5. Robotic spray painting (left), spray painting process scheme (right).

7. Conclusion and outlook

This paper represents another small step towards a more open and adaptable system for the design and fabrication with industrial robots. Projects such as the KUKA|prc tools prove that it is possible for architects to develop their own digital design tools to control and optimise robotic fabrication. The approaches presented in this paper show that this control does not have to be limited to the software side, but can also be extended to the hardware side, thanks to accessible IDEs such as Arduino.

We imagine, that in the near future architects will not only program industrial robots, but will soon learn from the automation industry to further develop their own physical tools. Custom digital and physical tools brought together will allow architects to control fabrication of mass customised parts, assembly and mounting alike.

In the scope of this paper, customised end-effectors were used for researching possible low-cost approaches towards linking physical computing with industrial robots. The topic of rapid prototyping for highly customised end-effectors is already relevant for industry: Just recently, the automation supplier Schunk introduced a gripper system that uses 3D-printed, custom designed jaws that are modularly mounted on a standardised gripper module (Figure 4, middle). In the near future we therefore see physical computing paired with rapid prototyping as an accessible and powerful tool allowing architectural practices and universities to gain an active role in researching full scale industrial robotic processes. The scalability of code will ensure that the concepts developed in this paper can be used for the full range of industrial robots, from KUKA KR5 to KUKA Titan.

Acknowledgements

We would like to thank Vienna University of Technology. This research has been supported by the Austrian Research Promotion Agency (FFG) via the project *KUKA|prc for spline TEX industry.* KUKA|prc can be downloaded from *www.robotsinarchitecture.org*.

References

Axelson, J.: 2007, *Serial Port Complete: COM Ports, USB Virtual COM Ports, and Ports for Embedded Systems*, Lakeview Research, Madison.

Banzi, M.: 2008, *Getting Started with Arduino*, O'Reilly, Sebastopol.

Biggs, G. and MacDonald, B.: 2003, A survey of robot programming systems, *in* Roberts, J. and Wyeth, G. (eds.), *Proc. Australasian Conference on Robotics and Automation*, Brisbane.

Bechthold, M.: 2010, The return of the future: a second go at robotic construction, *in* Oxman R. and Oxman R. (eds.), *The New Structuralism: Design, Engineering and Architectural Technologies*, Wiley, Hoboken, 116–121.

Braumann, J. and Brell-Cokcan, S.: 2011, Parametric robot control: integrated CAD/CAM for architectural design, *in* Cheon, J. H., Hardy, S. and Hemsath, T. (eds.), *Proc. 31st ACADIA Conference,* Banff, 242–251.

Brell-Cokcan, S. and Braumann, J.: 2010, A new parametric design tool for robot milling, *in* Sprecher A., Yeshayahu, S. and Lorenzo-Eiroa, P. (eds.): *Proc. 30th ACADIA*, Printinghouse, New York, 357–363.

Chinello, F., Scheggi, S., Morbidi, F. and Prattichizzo, D.: 2010, KCT: a MATLAB toolbox for motion control of KUKA robot manipulators, *in* Kumar, V. (ed.) *Proc. IEEE International Conference on Robotics and Automation*, Anchorage, 4603–4608.

MacDowell, P. and Tomova, P.: 2011, Robotic rod-bending: digital drawing in physical space, *in* Cheon, J. H., Hardy, S. and Hemsath, T. (eds.), *Proc. 31st ACADIA Conference,* Banff, 132-137.

Maletzki, G., Pawletta, T., Pawletta, S. and Lampe, B.: 2006, A model-based robot programming approach in the MATLAB/Simulink environment, *in* Morgan, M. N. (ed.), *Proc. 4th International Conference on Manufacturing Research*, Liverpool.

Payne, A.: 2011, A five-axis robotic motion controller for designers, *in* Cheon, J. H., Hardy, S. and Hemsath, T. (eds.), *Proc. ACADIA Conference*, Banff, 162–169.

Pigram, D. and McGee, W.: 2011, Formation embedded design: a methodology for the integration of fabrication constraints into architecture design, *in* Cheon, J. H., Hardy, S. and Hemsath, T. (eds.), *Proc. 31st ACADIA Conference*, Banff, 122–131.