Proceedings of the 2015 International Conference on
Advanced Mechatronic Systems, Beijing, China, August, 22-24, 2015

Extended Development of LinuxCNC for Control of a Delta Robot

Feng Huo
Department of Mechanical Engineering
National University of Singapore
Email: huofeng118@gmail.com

GeokSoon Hong
Department of Mechanical Engineering
National University of Singapore
Email: mpehgs@nus.edu.sg

AunNeow Poo
Department of Mechanical Engineering
National University of Singapore
Email: mpepooan@nus.edu.sg

***Abstract*—In recent years, open architecture motion controllers, including those for CNC machines and robots, have received much interest and support among the global control and automation community. This paper presents work done in extending a well-known and supported open-source control software called LinuxCNC for the control of a Delta robot, a translational parallel mechanism. Key features in the development process are covered and discussed and the final customized system based on LinuxCNC described.**

***Index Terms*—LinuxCNC, Delta robot, Real-time Linux.**

I. Introduction

With the increasing demand for greater productivity and higher accuracy in manufacturing and in robotic manipulators, the open architecture controller (OAC) has received much attention and support because of its flexibility, economy and efficiency [1]. Much research efforts have been directed to this research area in the last few years. This has greatly promoted the development and application of OAC. Various software platforms for the development of OAC are currently available, one of which is the popular LinuxCNC.

LinuxCNC is open-source software system running on real-time Linux developed for computer control of the motions of machines such as CNC machines, plasma cutters, robotic manipulators, etc. It is the descendant of the public-domain Enhanced Machine Controller (EMC) software developed by the National Institute of Standards and Technology, an agency of the US Department of Commerce [2]. Since the software and the OS platform are free and freely available for public use with extensive application information available on the internet, many people have been attracted to it and many have contributed vast amount of effort and time to develop and advance its capabilities and applications.

LinuxCNC is designed to be very general and versatile and can be adapted for various applications which require the simultaneous control of the motion of up to 9 axes of machines or robots. Provisions are available for users of LinuxCNC to add their own user-defined functions into LinuxCNC and to customize it according to their requirements [3], [4], [5], [6], [7], [8]. Many of precision motion control algorithms can be realized through LinuxCNC [9], [10], [11], [12]. In this paper, the application and adaptation of LinuxCNC for the control of a translational parallel manipulator, in the form of a delta robot, is presented together with details and discussion of the processes involved.

A brief introduction to LinuxCNC is provided in Section II. Section III then presents the general structure of the system and a brief discussion on the software design. Development details are covered in Section IV with Section V containing the conclusions.

II. Brief introduction of LinuxCNC

The source codes of LinuxCNC was written mainly in the language of C/C++. There are four main components [13]. These are a motion controller (EMCMOT), a discrete I/O controller (EMCIO), a coordinating task executor (EMCTASK) and several graphical user interfaces (GUI) templates. HAL (Hardware Abstraction Layer) is a component of LinuxCNC which provides a convenient way of allowing a number of building blocks, many of which are low-level drivers for hardware devices, to be interconnected, via named ”pins”, to build a complex system. The structure of LinuxCNC is illustrated in Fig. 1.

A. EMCMOT

EMCMOT is the only real-time component among the four main components. On one hand it uses either shared memory buffers or real-time Linux FIFOs mechanism to exchange commands, status and errors with EMCTASK. On the other hand it communicates with actual hardware module drivers through HAL pins. Since the motion controller is a real-time module, it executes cyclically at a user-defined sampling frequency. Typically, during each control/sampling cycle in the control of actual machine drives, EMCMOT will sample the positions of axes or joints to be controlled, compute the reference positions for each axis/joint in the desired trajectory for the next sampling instance, compute the position errors and the control output for the motors, and send these out to the motor servo-drivers. Within each sampling period, it also updates any status parameter, such as actual axis/joint position, in the shared memory buffers for display as required.

B. EMCIO

The EMCIO module handles all the I/O functions. It consists of a hierarchy of subordinate controllers for functions such as the control of coolant, spindle, auxiliary functions and

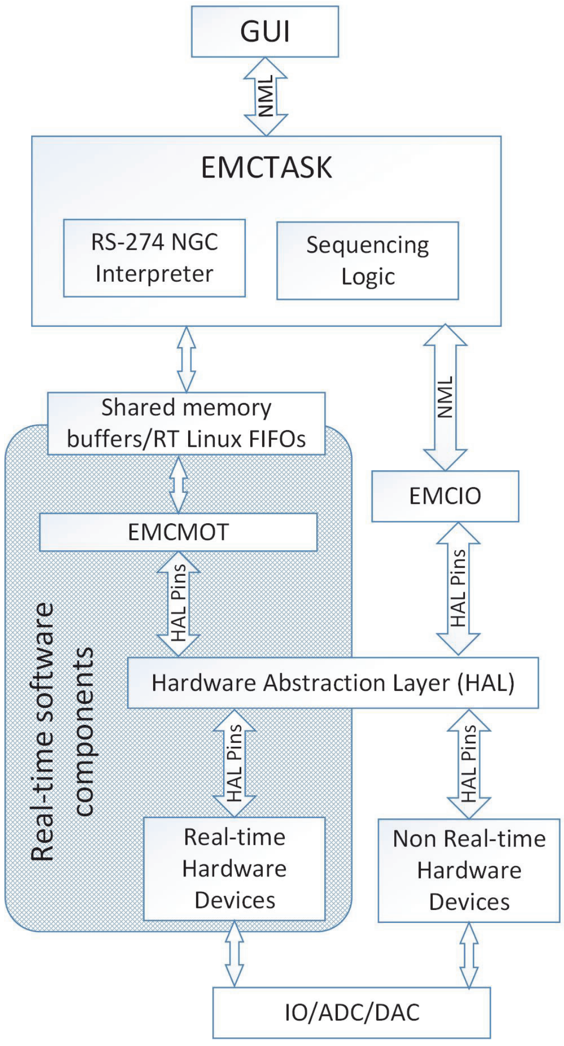
GUI
NML
EMCTASK
RS-274 NGC Interpreter
Sequencing Logic
NML
Shared memory buffers/RT Linux FIFOs
EMCIO
EMCMOT
HAL Pins
HAL Pins
Real-time software components
Hardware Abstraction Layer (HAL)
HAL Pins
HAL Pins
Real-time Hardware Devices
Non Real-time Hardware Devices
IO/ADC/DAC

Fig. 1. The structure of LinuxCNC

user-defined subsystems. EMCIO is highly machine-dependent and, as such, programmers need to develop specific application programming interface (API) functions for the different hardware devices in order to integrate these into LinuxCNC without modifying the core control codes.

C. EMCTASK

EMCTASK perches at a higher level with respect to EMCMOT and EMCIO. This module is written by using NIST RST Library and NML-Module-based class, which is similar with EMCIO. However, it is less machine-dependent. Its main function is to interpret the G and the M codes in the standard CNC part programs. Here the Neutral Message Language (NML) is a common API to communicate functions which are implemented within different LinuxCNC modules.

D. GUI & HAL

LinuxCNC provides several sample graphical user interface templates including AXIS, mini, and tklinuxcnc. The GUI communicates using NML and can not only display the information received from other modules but also send messages including power on/off, run a program with G/M codes. The GUI can also be run natively in a remote PC and exchange NML messages with the main PC through the network.

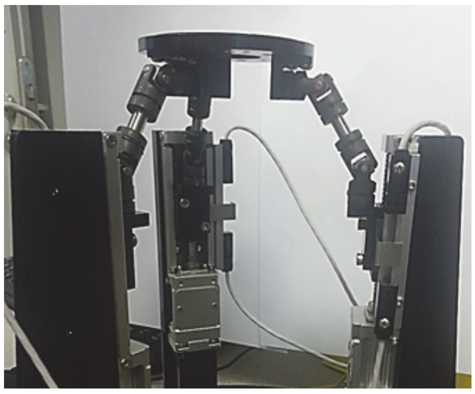

Fig. 2. Structure of Delta robot

For creating or editing a GUI, two LinuxCNC components are available, the Python Virtual Control Panel (PyVCP) and the Glade Virtual Control Panel (GladeVCP). The former is easier to use but the latter is more flexible and has more features available to the user. PyVCP is limited to use its TkInter widget set and has no support for embedding user-defined event handling. GladeVCP has more choices in GTK+ widgets and provides support for a Glade user interface editor. Arbitrary command execution is also supported by GladeVCP but its complex configuration and programming may deter first-time or less-experienced users.

HAL provides the facility to specify the interconnections between hardware devices and software modules by using the concept of software pins and functions for hardware interfaces and drivers. It allows for the interconnections between hardware devices and the software modules of LinuxCNC. Signals defined by HAL can be easily exchanged and transmitted. The HAL user interface (halui) connects HAL pins to NML commands. Most of the functionalities in the GUIs mentioned above are provided by HAL pins in halui.

III. System Structure for Delta Robot Control

A delta robot is a type of parallel manipulator. Its key feature is the use of a parallelogram structure for some of its links to maintain the orientation of the end-effector platform. A 3-DOF delta robot will only have the translational motions in its three axes with the actuating motors connected directly to the links. First developed in the 1980s, delta robots have been widely used for pick and place operations in the sorting and packing industry, their advantage being their capability for very fast motions.

In the work presented here, LinuxCNC is used to control a mini delta robot, shown in Fig. 2, for force/torque control together with pure position control. To achieve this, a 6-DOF force/torque sensor is mounted on the end-effector platform.

A simple schematic diagram showing the main components of the system is shown in Fig. 3. LinuxCNC runs in the PC under the RT Linux operating environment. It communicates with the external hardware devices through an installed LinuxCNC-supported interface card, a Motenc-Lite motion control and data acquisition card. This card provides four differential quadrature encoder inputs, eight channels each of analog inputs and outputs, and 48 digital I/O. These are sufficient to control the drivers for the three servo-motors of the robot, to read the digital encoders mounted on the three motors, and to acquire the six data signals from the force/torque sensor. The hardware wiring details are relatively straightforward and will not be discussed in this paper in which the focus will be on software development for adding to, and adaptation of, LinuxCNC.

IV. Adaptation of LinuxCNC

In order to use LinuxCNC to implement a CNC controller for the delta robot for both spatial trajectory control and for force control, changes need to be made to LinuxCNC together with the development and incorporation of a few user-defined, application-specific modules. These include changes to some initialization and configuration files, the creation or adaptation of a GUI panel, and the development of two user-defined modules related to the kinematics of the robot and another for computing the forces and torques acting on the robot from signals read from the 6-DOF force/torque sensor. Interconnections among the affected software modules and with the hardware device drivers are specified using HAL ”pins”.

The three user-defined modules are real-time modules in the sense that they are included in the interrupt loop and execute in real time at a frequency determined by the frequency specified for the interrupt thread. As such, their executions times are critical and they must execute their tasks fast enough so that all the modules included in the interrupt loop can be executed within the specified sampling period. The other modules are not executed in real time but are executed during the start-up phase of LinuxCNC to define the configuration of the system before the interrupt thread or threads are started. They are thus not time-critical.

Details of these changes and the user-defined modules developed are described in more detail in the following sections.

A. *Initial Configuration*

There are several key files that need to be introduced before a meaningful discussion of the user-defined modules can be made. These files and their functions are described in Table I. LinuxCNC is configured using text files, one of which is the INI file which overrides defaults compiled into LinuxCNC. In the motenc.ini file, the parameters required to configure the robot system are defined. These include specifying the number of axes as three corresponding to the three actuating motors for the Delta robot. For each of these axes, suitable dynamic parameters such as the the PID control gains, maximum allowable velocity and maximum allowable acceleration are

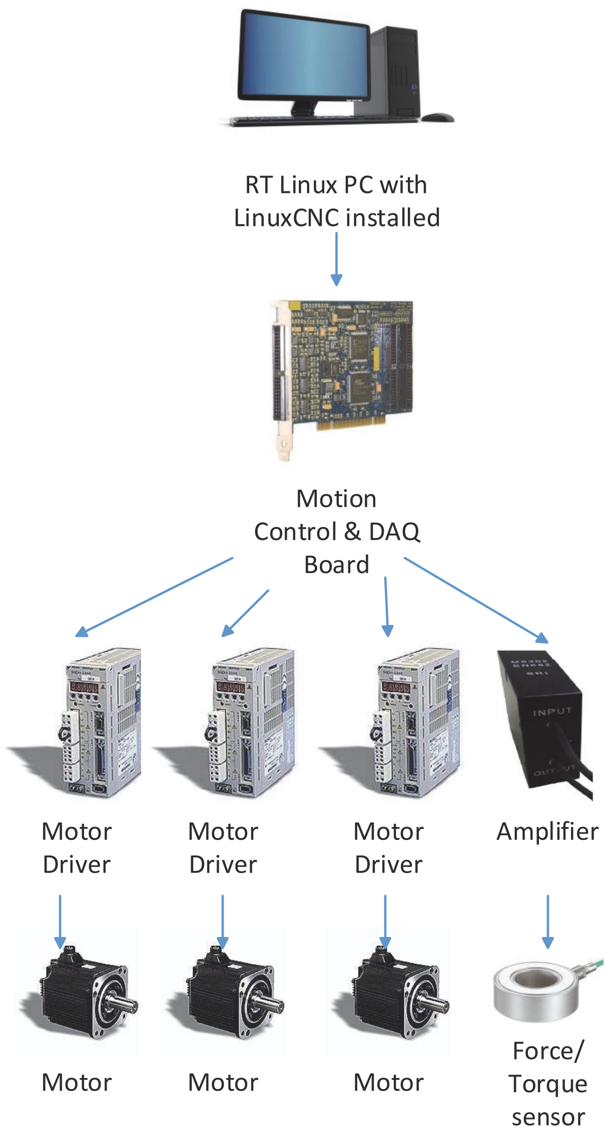

Fig. 3. LinuxCNC controlled delta robot

TABLE I
CORE CONFIGURATION FILES

motenc.ini	Initialize all the parameters for EMCTASK, EMCMOT, EMCIO and some general sections.
core_servo.hal	Core real-time modules are loaded in this file and all the basic PID pin connections are defined here too.
motenc_motion.hal	Hardware driver and other RT modules are loaded in this file and all the HAL pins configuration related to EMCMOT are defined here.
motenc_io.hal	The HAL pins configuration related to EMCIO are defined in this file such like GPIO, limit switch signals and watchdog.
custom_postgui.hal	Custom HAL commands and signals are defined in this file. It will be loaded after GUI starts up.

specified according to the characteristics of the motors and the load they drive. The names of HAL files which are to be loaded are also specified here.

B. HAL Pins

Provided in LinuxCNC are several HAL commands for loading modules, specifying their interconnections and for assigning values to certain parameters. The ”loadrt” command is used to load the user-defined modules while the ”net” command is used to connect two HAL pins as a signal. The ”setp” command is used to assign a value to a HAL parameter. An example of the use of HAL commands using HAL pins to connect input signals from the hardware device to a software module in LinuxCNC is shown in List 1, In this example, the commands are for connecting the six signals from the force/torque sensor connected to the analog inputs of the Motenc interface card to a user-defined module for processing these signals.

Listing 1.

```
# connect ADCs to force/torque Module
net ch1 motenc.0.adc-00-value => huoef.sriraw.0
net ch2 motenc.0.adc-01-value => huoef.sriraw.1
net ch3 motenc.0.adc-02-value => huoef.sriraw.2
net ch4 motenc.0.adc-03-value => huoef.sriraw.3
net ch5 motenc.0.adc-04-value => huoef.sriraw.4
net ch6 motenc.0.adc-05-value => huoef.sriraw.5
```

C. User-defined Modules

For the Delta robot, a set of non-linear kinematics equations define the relationships between the positions of the actuating motors and the position of the end-effector. For this project, a user-defined Forward Kinetics Module was developed to compute the coordinates of the position of the end-effector based upon the positions of the actuating motors. Development of this module for the Delta robot is quite involved requiring the solution of a set of polynomial constraints to determine the possible end-effector positions. Another user-defined that was developed is the Inverse Kinematics Module. This module does the reverse and computes the required positions of the actuating motors for a given desired position of the end-effector. For the Delta robot, the inverse kinetics is relatively straightforward.

LinuxCNC has some pre-defined variables in a ”trivkins” module which can be used as reference. The relationships that need to be redefined for this project are shown in Table II. In the table, the joints are the positions of the three motors while the tran.x, tran.y, and tran.z are the coordinates of the end-effector. For inverse kinematic, tran.x,y,z to express joints and vice versa for forward kinematic. If we let joints be equal to tran.x,y,z, then reference position in G codes will be just the position of each actuator.

The third user-defined Force/Torque Sensor module reads the raw signals from the force/torque sensor and process these using a decoupling matrix to obtain the three forces and three torques acting on the sensor. the decoupling matrix is obtained from a calibration procedure performed on the sensor. The processed force/torque values are sent to the GUI through the *rtapi_snprintf* function. The target values of the force and torque can be set so that this module can compensate the reference command to the end-effector.

TABLE II
KINEMATIC RELATIONSHIP

Inverse kinematic	Forward kinematic
joints[0]	pos->tran.x
joints[1]	pos->tran.y
joints[2]	pos->tran.z

D. GUI

In the project discussed here, the sample AXIS GUI panel provided in LinuxCNC was used and modified by the addition of a user-designed panel. This panel is for the display of the readings of the six components of the force/torque sensor. The pin connections are shown by the codes in List 2. In the list, obj_flag is used as a flag to provide users with a choice of two modes of specifying the required motion of the Delta robot, an end-effector mode and an actuator mode. In the former, user inputs are in end-effector coordinates while in the latter, these inputs are in actuator motor coordinates which directly correspond to motor positions. When the desired motion is specified in end-effector coordinates, the interpolations for the required motion, e.g. linear interpolation, are performed by LinuxCNC and the resulting motion trajectory in end-effector coordinates at each sampling period are converted by the Inverse Kinematic Module into command reference positions for the control of the actuator motors. When the required motion are specified directly in actuator motor positions, any interpolation required are performed directly in motor coordinates and these are used as the command reference positions of the motors for the feedback loops without the need of processing by the Inverse Kinetic Module.

During motion, the actual positions of the three actuator motors, provided by the digital encoders attached to their shafts, are read through the Motenc interface card. These are processed by the Forward Kinetics Module to generate the corresponding position of the end-effector in end-effector coordinates. These are then used for display in the GUI. For the human eye, the update frequency of this display need not be as high as the sampling frequency of the position control loop for the motors. As such, this process can be performed at a lower sampling frequency. LinuxCNC allows more than one interrupt thread to be created, each with its own sampling frequency.

Another example of the use of HAL command codes is shown in List 2. Here the codes are used to specify how the force/torque values computed by the Force/Torque Sensor module are to be connected for display in the GUI panel. Shown in List 3 are the command codes used for the design of the bar display in the GUI for Fx, the force on the sensor along the X-axis. The result can be seen in the display panel shown in Fig. 4.

Listing 2.

```
net   Fx  huoef.Fx  =>  pyvcp.my_fx
net   Fy  huoef.Fy  =>  pyvcp.my_fy
net   Fz  huoef.Fz  =>  pyvcp.my_fz
net   Mx  huoef.Mx  =>  pyvcp.my_mx
net   My  huoef.My  =>  pyvcp.my_my
net   Mz  huoef.Mz  =>  pyvcp.my_mz
net  obj_flag  eefkins.obj  <=  pyvcp.Ctr_Obj.Tool
```

Listing 3.

```
<label>
                <text>"Fx"</text>
        </label>
        <bar>
                <halpin>"my_fx"</halpin>
                <min_>0</min_>
                <max_>200</max_>
                <bgcolor>"grey"</bgcolor>
                <fillcolor>"red"</fillcolor>
        </bar>
```

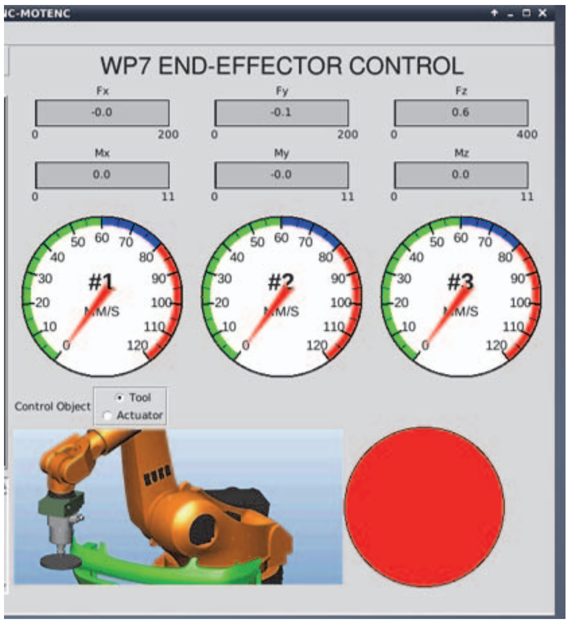

Fig. 4. GUI Panel

V. Conclusions

This paper presented the adaptation and changes required in LinuxCNC, an OAC designed for real-time motion control, to control a delta robot for force/position control. Details of the key points in the implementation process are presented and discussed including the user-defined modules, interconnections among hardware devices and software modules and changes to configuration files. The LinuxCNC platform is a convenient and easy-to-use software platform for implementation of a control system for any specific motion control application/ It provides good facilities for adaptation an the convenient incorporation of user-developed and tailored modules to meet specific user requirements. The work described here should help as a practical guide for engineers developers of control systems for CNC machines and robotic manipulators.

Acknowledgment

The authors acknowledge the support from the Collaborative Research Project under the SIMTech-NUS Joint Laboratory (Industrial Robotics), Ref: U12-R-051SU and support from the SERC Industrial Robotics Program Work Package 7, project number: 1225100007.

References

[1] Y. Koren, "Open-architecture controllers for manufacturing systems," *Open Architecture Control Systems–Summary of Global Acitvity*, pp. 15–37, 1998.
[2] "Enhanced machine controller," http://www.linuxcnc.org.
[3] L. Romero Munoz, M. Garcia Villanueva, and M. Santana Gomez, "Development of a computer numerically controlled router machine with 4 degrees of freedom using an open architecture," in *Power, Electronics and Computing (ROPEC), 2013 IEEE International Autumn Meeting on*, Nov 2013, pp. 1–6.
[4] E. Wings, M. Mller, and M. Rochler, "Integration of real-time ethernet in linuxcnc," *The International Journal of Advanced Manufacturing Technology*, pp. 1–10, 2015. [Online]. Available: http://dx.doi.org/10.1007/s00170-015-6786-y
[5] J. Ros, R. Yoldi, A. Plaza, and X. Iriarte, "Real-time hardware-in-the-loop simulation of a hexaglide type parallel manipulator on a real machine controller," in *New Trends in Mechanism and Machine Science*, ser. Mechanisms and Machine Science, F. Viadero and M. Ceccarelli, Eds. Springer Netherlands, 2013, vol. 7, pp. 587–597. [Online]. Available: http://dx.doi.org/10.1007/978-94-007-4902-3_62
[6] L. F. Cerecero-Natale, J. C. Fernandez, L. E. Ramos-Velasco, and V. E. Pedraza Vera, "Fuzzy gain scheduling pi controller for a mechatronic system," in *World Automation Congress (WAC), 2012*, June 2012, pp. 1–5.
[7] L. Pusman and K. Kosturik, "Integration of digimatic measuring tool into linuxcnc controlled milling machine by using modbus," in *Telecommunications Forum Telfor (TELFOR), 2014 22nd*, Nov 2014, pp. 691–693.
[8] F. Tajti, G. Szayer, B. Kovacs, and P. Korondi, "Universal rt-middleware robot controller," in *Industrial Electronics Society, IECON 2013 - 39th Annual Conference of the IEEE*, Nov 2013, pp. 7862–7867.
[9] F. Huo and A.-N. Poo, "Precision contouring control of machine tools," *International Journal of Advanced Manufacturing Technology*, doi: 10.1007/s00170-012-4015-5.
[10] F. Huo, X.-C. Xi, and A.-N. Poo, "Generalized Taylor series expansion for free-form two-dimensional contour error compensation," *International Journal of Machine Tools and Manufacture*, vol. 53, no. 1, pp. 91–99, 2012, doi: 10.1016/j.ijmachtools.2011.10.001.
[11] F. Huo and A.-N. Poo, "Improving contouring accuracy by using generalized cross-coupled control," *International Journal of Machine Tools and Manufacture*, vol. 63, no. 0, pp. 49 – 57, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0890695512001423
[12] ——, "Nonlinear autoregressive network with exogenous inputs based contour error reduction in {CNC} machines," *International Journal of Machine Tools and Manufacture*, vol. 67, no. 0, pp. 45 – 52, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0890695513000023
[13] T. Staroveski, D. Brezak, T. Udiljak, and D. Majetic, "Implementation of a linux-based cnc open control system," in *12th INTERNATIONAL SCIENTIFIC CONFERENCE ON PRODUCTION ENGINEERING*, Jun 2009, pp. 209–216.