Low Cost Robots as Target System for Students Training Using Java

doi:10.3991/ijoe.v5i4.1120

D. A. H. Samuelsen and O. H. Graven
Buskerud University College, Kongsberg, Norway

***Abstract*—The topic of this paper is a cost efficient programming environment designed with a focus on enhancing the students' engagement and effort to complete the assignments. The environment is designed to give the students an approach that is similar to a professional work situation with respect to hardware-software interaction, documentation and complexity. The system presented is based on low-cost, off-the-shelf equipment for easy implementation on mass in an educational institution. As part of the system a low-cost positioning system for mobile robots in an indoor application is included, giving further opportunities for engaging task for the students.**

***Index Terms*—e-learning, high level programming, distributed systems, Java, interfacing.**

I. INTRODUCTION

One of the goals in engineering education is to prepare large number of students for "real life" work. This preparation needs to be an integral part of the various modules in their bachelor degree. Added to this is the ever increasing need for a cost effective way of achieving this goal.

In addition to preparing the students for real life in a cost efficient manner the change in student population and attitude over the last few years makes this an ever increasing challenge. The number of students expecting to complete a higher education degree is growing [1], giving us as educators an increased challenge to encourage and motivated the students to complete assignments and keep them engaged through lab exercises. In the method used for the tutorials described in this paper, we focus our effort in trying to keep the students challenged trough the whole assignment by gradually increasing the difficulty of the tasks the students are given. Our aim is to entice a state of flow in the students. Flow can occur when an activity is challenging enough to encourage exploratory behaviour without the activity being too difficult for the student to handle [2]. The task at hand becomes its own driver and the feeling of accomplishment will drive the students to even greater effort.

The scenario set for the lab is that programmers are often set in a situation where they are given a piece of off-the-shelf of custom made hardware controlled by a processor[3], and then given the task to design and implement the software for this system. In the system used for these lab exercises, the scenario is that the system is custom made by other engineers, the students filling the role of software engineers, and given a set of pre made protocols for interfacing with the hardware.

II. FLOW

The concept of flow was introduced in 1975, through a study of people involved in activities such as rock climbing, chess and dance[4]. Flow is described as

"*A state of complete absorption or engagement in an activity and refer to the optimal experience. During optimal experience, a person is in a psychological state where he or she is so involved with the goal driven activity that nothing else seems to matter.*"

According to flow theory, flow can occur when an activity challenges an individual enough to encourage playful, exploratory behaviours, without the activity being beyond the individual's reach. For example, if the activity is too demanding it may produce anxiety rather than flow. Or, if it is not challenging enough, boredom, not flow, may be the result. Past research[5, 6] has shown that the flow state has positive impact on learning.

In a situation where we desire to give the students ever increasing difficult tasks to keep them challenged, there is a danger of pushing it too far. Some educators will argue that this is not a bad thing and even argue for the role of frustration in the learning experience, or more precisely the resolution of the frustration. We as educators must be careful with the use of or inflicting frustration on students during the learning process. The final resolution of the frustration is a powerful stimulator and source of great joy, the target state of flow. The major problem with this approach is frustration can and will turn many students off, if not applied with great care and tailored to the individual student. There will often be some initial frustration or resistance when moving into a new area of learning, the use of scaffolding and increased help to struggling students as described in the design for our system is targeted at reducing the frustration and ensure that the students are not turned off the whole experience. On the other hand the system must not make the challenges to easy and turning the students off in that way. To avoid this, fading techniques are used to gradually remove the scaffolding as the skills of the students evolve, and the tasks given throughout the assignment become more complex and difficult to solve.

Therefore, one main aim for us as designers of learning material is to design content in such a way as to allow different students to obtain the state of flow, irrespective of their different knowledge and abilities.

III. THE STUDENT ASSIGNMENT: MAKE A ROBOT THAT CAN COMPETE IN A GAME OF TAG WITH OTHER STUDENTS' ROBOTS

For engineers, the initial task when faced with custom made systems as presented in the introduction is to familiarize with a large amount of documentation with more or less relevant information on the hardware system. This is a challenge for us as educators when designing the exercise for the students, due to the type of documentation. The documentation for this type of custom made systems normally do not consist of a single document created with a coherent narrative, but rather a collection of separate documents of different types, with a large variance in relevance to the task to be solved. We want to balance the aid given to the students to get them trough the challenges with the necessity of letting them stuggle and feel that there is a real challenge they are working with. We do not want the students to give up in despair, but equally we do want them to struggle a bit, and in this way understand the necessity of learning how to handle these types of tasks

The assignment is to let the students program the robots to play the well known children's game, "tag". The rules of the game are as follows: One player always has the tag. The player having the tag runs after another player and touches this player; the tag is then transferred to the other player, and the player originally having the tag turn back into a normal player. Now, the player having the tag has to pursue the other players in order to get rid of the tag.

Rules of the tag game; robot style:

- At the start of the game the robot that has the tag has to broadcast its position. Every time the tag is passed on to another player, the new robot has to broadcast its position.
- A robot that has the tag is allowed to run at 100% speed
- A robot without the tag is limited to 90% of maximum speed
- The tag is passed when a robot that has the tag touches another robot.
- When a robot has passed the tag to a new robot it has a 30 second "free" period where it cannot be tagged back.

The robots are equipped with several actuators, among them a set of motors to move the robot around on the floor. There are also several sensor systems to detect objects and obstacles in the vicinity of the robot. The robots are capable of detecting special signals from beacons placed at specific locations in the room. These beacons are used to derive an absolute position for each robot on the floor. The robots have the ability to communicate with each other through wireless communication. This communication is done either through a central unit or direct.

The actuators and sensors can be controlled by the high level software, using software library functions when programming the processor running the Java code. The library functions for both the actuators and the sensors are given to the students, ready to use, in a basic form. The processor running the Java code is embedded in a sunspot[7]. This gives the students the opportunity of building the library functions on their own, improving the performance of the software implementation, when the student's skills allows for this. At the same time it allows for a quick first implementation of a moving robot, as well as something to fall back on if they are unsuccessful in later stages.

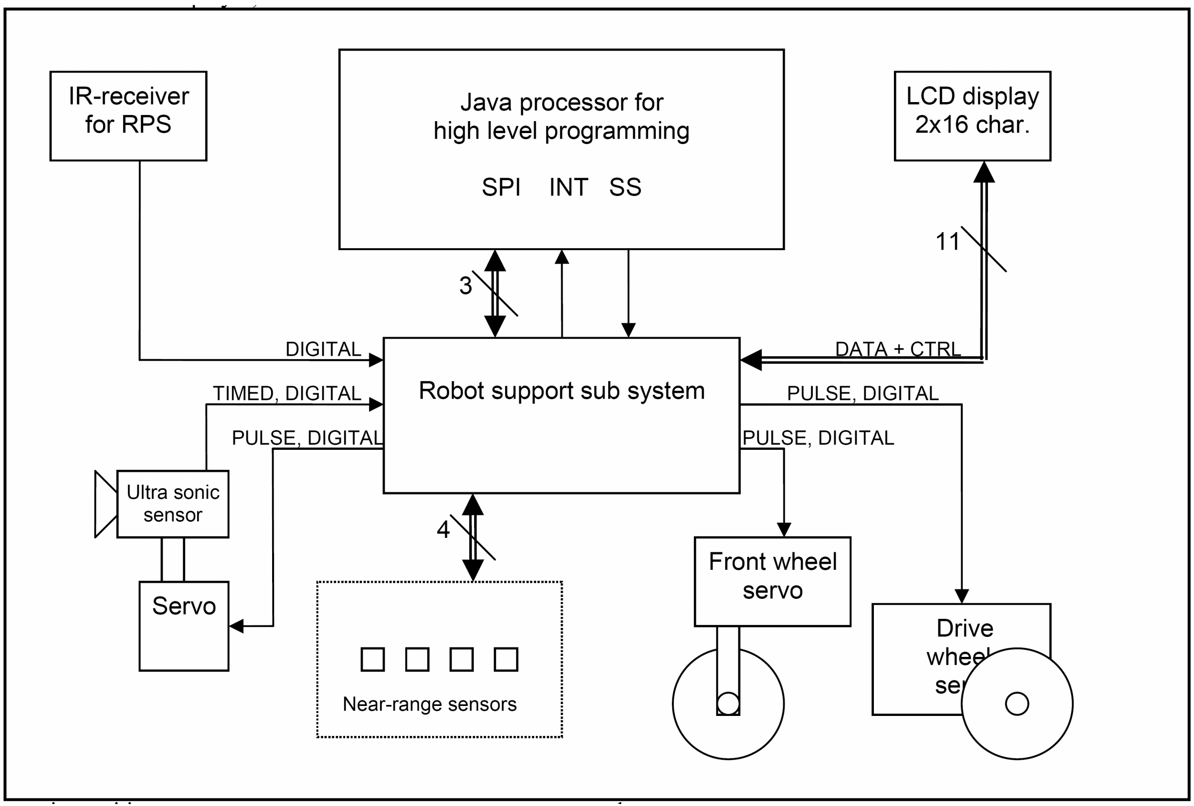

Figure 1. Overview of the system

A. *Actuators available in the game*

The actuators present on the robot that can be controlled are shown in figure 1:

- Main drive motor[8]. This motor makes the robot move forward and backwards. The motor can be set at different speeds, the number of levels being dependent on the hardware/low level software implementation, the current implementation of the robot support subsystem giving 256 different levels in total, which gives 127 levels forward and 127 levels backwards.
- One servo[9] controls the front wheel in a 180 degree arc with 90 degrees being straight forward. The ability to steer the front wheel is known as a “rack-and-pinion” steering, as opposed to the system most robots have which make use of two motors both for driving and turning of the robot, by running the motors on either side of the robot at different speeds This “rack-and-pinion” steering of this robot is more complex in use, and gives the students an extra challenge in controlling the robot. This is done to emphasize that the performance of the robots behaviour in the environment is dependent on the quality of the programming, which again is reflected in how good a robot of a certain group performs relative to the other robots.
- One servo[9] is used to control the direction of the ultrasonic sensor, mounted in the front of the robot. The sensor is described in the next section.

B. *Sensors available for the game*

There are a set of sensor that feed the robot information about the environment it moves around in, shown in figure 1:

- A set of near-range infra-red sensors are placed around the robot, making it possible to avoid having the robots physically colliding with objects. The actual avoidance of objects is done via the program the students put into the robots. An area that is heavily influenced by the performance of this part of the programming by the students is the robots ability to successfully back out of corners.
- An ultrasonic sensor[10] is used to detect obstacles at a relative far distance from the robot, in order to allow the use of more long-term strategies for moving around, avoiding running the robot into corners, etc. This is a directional sensor, and its use is described in detail in a later section.
- The positioning system gives each robot’s position relative to a fixed point in the room the robots are placed in. This sensor is not necessary to play the game of “tag”, but rather one of several other possible solutions to allow the robots to identify their own and the position of the other robots on the floor. One possible advanced use of this positioning system is the possibility for the robots to construct a map of the area and the position of obstacles on this area. Compared to the use of the ultrasonic sensor, this will allow for even higher level strategies for movement of the robot on the floor, as different optimizations can be implemented, increasing the performance of the robot further.
- A set of contact bands surrounds each robot. These bands are used to identify contact between robots. The tag game this is used to transfer the tag from one robot to another. The transfer of the tag is also broadcasted via the wireless network to all the other robots, along with the position of where the exchange has taken place.

It is important to note that, although the robots have a number of sensors that makes it possible to implement a complex control of the robots, it is not given that all students will be able to utilise the full potential of the system. This gives us and the students the opportunity of adapting the complexity of the tasks to the experience levels of the individual students, in order to fulfil the state of flow in students, even if the level of skills differ among the students.

IV. HARDWARE IMPLEMENTATION – ROBOT DETAILS’

The hardware in the robot is described in this section, starting with the robot support system, which is the first line of communication between the processor running the java code and the actuators and sensors, described later in this section.

A. *Robot support subsystem*

The utilisation of the robot depends much on the interface between the Java processor and the actual sensor or actuator. This interface is part of the robot support system, and is not relevant for the students to deal with in this context. The robot support system is also responsible for normal housekeeping task, including charging of the batteries and power supply for the electric installations in the robot.

This makes it possible to use a single power source (the battery pack) for all devices installed in the robot. This makes the robot simpler, cheaper to purchase, easier to maintain and charging simpler as there is only one battery pack which needs charging. For the robot to be able to move around for extended periods of time during a lab exercise, a quite large and heavy (but low cost) battery pack is mounted on the robot. This battery pack is quite heavy, and we found it not recommended placing this battery pack on the robot itself. This is also troublesome due to limited space on the initially small robot. This problem was solved by adding a trailer to the robot. The trailer is

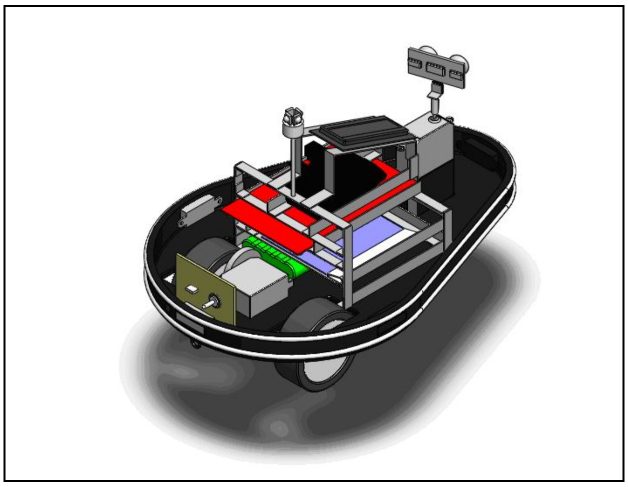

Figure 2. Interior of the robot

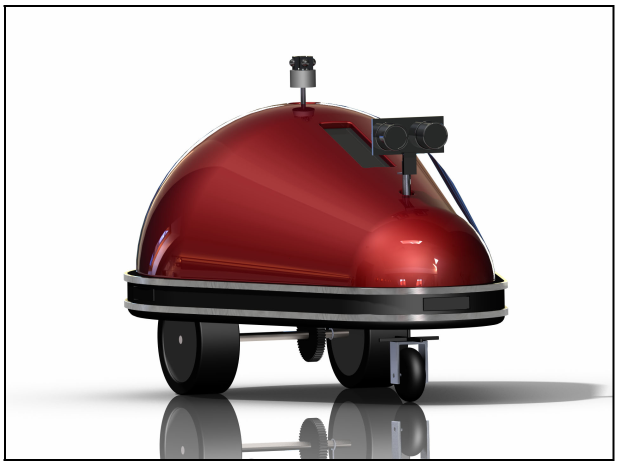

Figure 3. CAD-drawing of the robot with sensors

only shown in the early model. This solution also gives rise to a range of new challenges for the students. The robot can also function without the trailer, at the cost of reduced operational time.

The interface is controlled by a PIC18-series microcontroller, which is pre-programmed with the necessary software for interfacing with the Java processor as well as the actuators and sensors of the robot. A normal 3-wire SPI-interface is used for interfacing with the Java processor, and the robot support system acts as a slave on the SPI-bus. There is also a separate "slave select"-line, to allow for connection of more devices to the same bus. In addition, there is an interrupt line in order to issue an interrupt to the Java processor. This is necessary as the slave of an SPI-bus cannot initiate communication when needed.

An example of a situation where this is needed is the occurrence of physical contact between two robots, through the contact bands. Now the robot support system needs to inform the Java processor of the situation. The support system issue an interrupt to the Java processor via the interrupt line, and the Java processor responds to this interrupt by asking the support system, via the SPI bus, what caused the interrupt.

B. *Actuator system*

The actuators are simply two different types of standard, low cost servos often found in toy models: a standard position servo, and the continuous rotation servo, both built by Parallax[10]. These servos are controlled by supplying a control signal consisting of a pulse of varying length. This pulse is repeated every 4ms, or the servo will go into standby mode. A pulse of 1.5ms will cause the continuous rotation servo to go to standstill, or the position servo to set the middle position. Reducing the pulse length gradually down to 1.25ms, causes the continuous rotation servo to gradually increase speed to maximum backwards speed, while the position servo will gradually move to the leftmost angle. Increasing the pulse width gradually to 1.75ms, do the opposite, i.e. maximum forward speed or rightmost position.

C. *Sensor system*

The sensors are low cost equipment with a generally simple interface.

1) *Contact bands*

The contact bands are used for detection of physical contact between two robots. These bands are simply two bands of non-isolated, current conducting material, with a voltage difference between the two bands. The voltage difference between the bands is set by weak pull-up or pull-down resistors, causing the voltage of a certain robots to change when a direct contact between two robots with different voltage potential between the bands occurs. This system seldom fails, and imposes relatively low challenges to the software designer.

2) *Near-range sensors*

The near-range sensors consist of an infra-red emitter and receiver, and gives out a voltage proportional to the distance between the sensor and any surface capable of reflecting infra-red light. Large surfaces like walls are generally good reflectors, while smaller objects like table legs are easily missed by these sensors. The uncertainty of the object detection is one challenge in designing robust higher order movement strategies.

3) *Ultrasonic sensor*

In order to detect objects further away from the robot than the near-range sensor is capable of, an ultra sonic sensor is placed on the top of the robot. This sensor emits a relatively narrow beam of ultrasonic sound (air waves) in a certain pattern, and the sensor "listen" for this particular pattern to be reflected of some surface back to the sensor. The time difference from emission to reception is proportional to the distance of the object. As the beam is relatively narrow, the sensor will only detect objects or surfaces in the direction on the sensor. In order to detect objects in other directions, the sensor must be directed by the servo which it is mounted on.

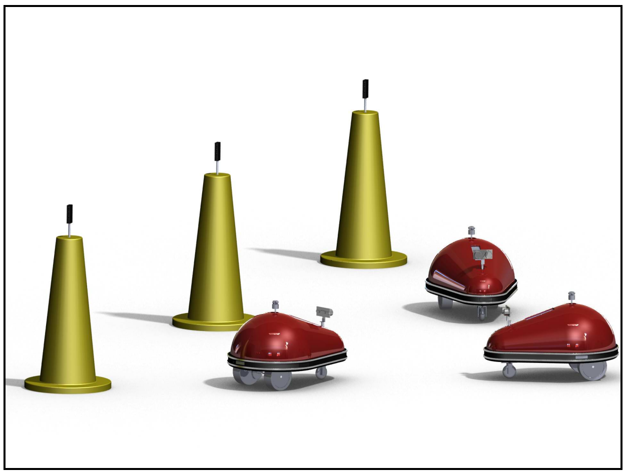

Figure 4. Positioning system setup with towers.

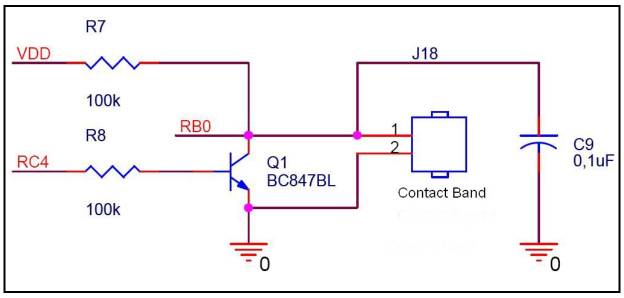

Figure 5. Schematic for the contact bands.

As the sensor depends on reflection of sound waves, the uncertainty lies in the surfaces ability to reflect the sound back to the sensor. Soft surfaces like fabric usually damp the energy of the sound, making it difficult for the sensor to detect the reflected pattern, while smooth surfaces that are placed normal to the sensor gives a good reflection. As with the near-range sensors, the possibility of missing objects that should be detected, impose an extra challenge to the programmer.

4) *Positioning system*

The robots position can be revealed using a system of three beacons mounted on one of the walls in the lab/room, and infra-red receivers mounted on top of the robots. The beacons emit a very narrow beam of infra-red light in an arc around each beacon. By synchronising the movement of the beams through the arc, the time difference between each "hit" of the beam as seen from the robot, can be used to calculate the position of the robot, with a relative high degree of accuracy.

V. THE TASKS FOR THE STUDENTS

One property of a good student project is the ability to adapt the level of complexity to the student, both when the student start to work on the project, and adapt the challenge to the skills the student acquire during the project.

The project presented in this article is designed to be adaptable in this fashion. The game assignment is applicable for students with just an introductory course in programming, as well as more experienced students. This is achieved by splitting the assignment into parallel sub-assignments with differing levels of support and constraints made available for the students, thus creating challenges for students with different backgrounds and capabilities This spilt is achieved in two ways; the robots equipment gives the students a number of possibilities as well as constraints, while we as educators also impose different limitations and offer support throughout the assignment, and the documentation will be presented at different levels and in different forms.

In order to have a stepwise approach the final task, i.e. playing the game of 'tag', the students are given a set of sub-assignments, they need to complete in order:

- The first task is to get the robot moving on the floor from point to point, without hitting anything or anyone. This task involves the setting of commands to the actuators, and reading of the sensors. Which sensors to use are left to the students to decide, as a means of letting the students themselves find the correct level of complexity versus performance of the robot.
- Next, the robots should find and communicate their position on the common network, and thereby be able to find other robots in the vicinity of their own robot, and moving either away from them, or closer to them. This test is run without obstacles on the floor.
- Now, the game of 'tag' is employed, and obstacles are placed on the floor for making the navigation of the robots more complex. This last step is set to give the students a sense of competition and thought to encourage the students to put more effort into the programming of the robots, as to enhance its performance.
- For the very skilled students, the positioning system can as a last step be used to build a map of the playground and its obstacles, and more sophisticated strategies can be employed for moving the robots around on the floor. This last step is considered a bit complex for undergraduate students, but suitable for graduate students.

The students are offered different levels of assistance during the tasks. Ranging from just making sure the students had access to the basic documentation through to prewritten stubs of code both for the java and the FPGA parts.

In order to maintain a state of flow in the students, the construction of the robot system allows for two strategies to be used. One of the strategies deals with the adaptation to the average skill level of the group of students present. This adaptation is done by the tasks given throughout the assignment, giving the student enough to work on at the start of the assignment, and then increasing the complexity of the tasks towards the end of the assignment. The other strategy is also used for the complete duration of the assignment, but is more pronounced at the end of the assignment. This strategy relies on how the students utilise the different sensors available in the robot. Less skilled students might use the more simple sensors, such as the contact bands or the near-range sensors, still fulfilling the tasks specified in the assignment, while the more skilled students might utilise the more complex sensors, resulting in a higher degree of performance of the robot, introducing a sense of competition based on how well the sensory system is utilized, giving the students a goal to strive for. This is the realization of the scaffolding and fading techniques as described in the introduction of this paper.

VI. CONCLUSION

A system that is based on low-cost, off-the-shelf equipment for easy implementation on mass in an educational institution has been successfully implemented and presented. As part of the system a low-cost positioning system for mobile robots in an indoor application is included, giving further opportunities for engaging task for the students. A trial run has been performed with a small group of students with high levels of skills. The trial students have shown an observable increased level of engagement compared to traditional lab work. While utilising the while range of actuators and sensors available. We are now moving on the trials with larger and more diverse groups of students.

REFERENCES

[1] Statistisk sentralbyrå, "Statistics Norway," http://www.ssb.no/.

[2] T. Govindasamy, "Successful implementation of e-Learning Pedagogical considerations," The Internet and Higher education, vol. 4, pp. 287-299, 2001. (doi:10.1016/S1096-7516(01)00071-9)

[3] Microcontroller PIC18F4620: http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en010304

[4] M. Csikszentmihalyi and I. Csikszentmihalyi, *Beyond boredom and anxiety*. San Francisco: Jossey-Bass, 1975.

[5] J. Webster, L. K. Trevino, and L. Ryan, "The Dimensionality and Correlates of Flow in Human-Computer Interactions," *Computers in Human Behavior*, vol. 9, pp. 411-426--, 1993.

[6] M. Csikszentmihalyi, *Flow : the psychology of optimal experience*. New York: HarperPerennial, 1990.

[7] Sun Microsystems, "Sunspot world," http://www.sunspotworld.com/. Last visited 30.September.2009

[8] Parallax, "Parallax (Futaba) Continuous Rotation Servo," http://www.parallax.com/Store/Accessories/MotorServos/tabid/163/CategoryID/57/List/0/Level/a/ProductID/102/Default.aspx?SortField=ProductName%2cProductName

[9] Parallax, "Parallax (Futaba) Standard Servo," http://www.parallax.com/Store/Accessories/MotorServos/tabid/163/CategoryID/57/List/0/Level/a/ProductID/101/Default.aspx?SortField=ProductName%2cProductName

[10] Parallax Inc., "Ping documentation v1.5," http://www.parallax.com/Portals/0/Downloads/docs/prod/acc/28015-PING-v1.5.pdf

AUTHORS

Dr. Dag A. H. Samuelsen (Dag.Samuelsen@hibu.no) and **Olaf Hallan Graven** (Olaf.Hallan.Graven@hibu.no) are both with Department of Technology, Buskerud University College, Postbox 235, 3601 Kongsberg, Norway.

Submitted 23 October 2009. Published as resubmitted by the authors on 8 November 2009.