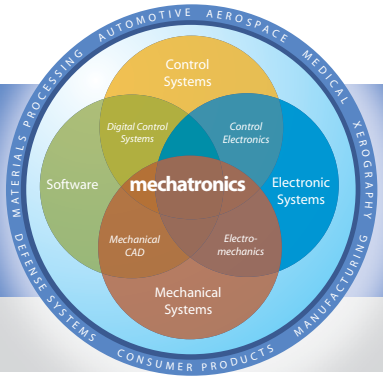
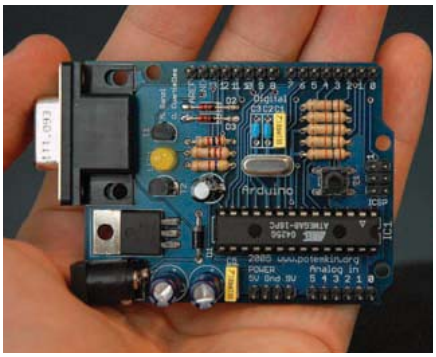
MECHATRONICS IN DESIGN



Automatic Code Generation & the Arduino

An unbeatable combination for students and practicing engineers.

Innovation in engineering practice and education has been reinvigorated by two complementary concepts: automatic code generation and the inexpensive, open-source, single-board microcontroller. Automatic code generation from system block diagrams has been around for decades, and now is an entrenched way of developing embedded control systems and performing hardware-in-the-loop testing for many aerospace and automotive



companies; it is quickly moving into other industries. The Arduino microcontroller has been around since 2005, but quickly has become a favorite for inventors and students.

There is something about computer programming and implementation that seems so out of place in engineering problem solving that very often an engineer relegates it to a specialist. The phrase "engineer programmer" is an oxymoron. I have said that the human, the computer tools (software and hardware), and the problem should all be in perfect harmony throughout the entire problem-solving process. The combination of graphical programming using block diagrams, automatic code generation from the block diagrams, and then implementation on an easily understandable, yet powerful, microcontroller comes close to that ideal.

Honeywell, in 2005, observed that the typical software process injects 100 defects, due to both design and coding errors, per 1,000 lines of source code using manual processes. Manufacturing processes have been automated to achieve six-sigma quality levels, i.e., not more than 3.4 defects per million opportunities. Honeywell has automated the manufacturing of software through the use of automatic code generation and demonstrated the achievement of six-sigma quality.

Northrop-Grumman has fine-tuned the process of going from the desktop directly to flight code on flight hardware. Rapid prototyping and hardware-in-the-loop testing are now the rule, rather than the exception.

What could possibly excite an engineer or engineering student more than solving a real-world problem? Seeing that solution implemented in hardware does that! While teaching model-based design and controls over the past 20 years, I have not seen a more exciting, effective, and accessible problem-solving combination than graphical block-diagram programming, as is done in the MatLab/Simulink environment, along with automatic generation of C code for a microcontroller, as is done using the Arduino microcontroller with the Simulink Coder.

Kevin C. Craig, Ph.D., Robert C. Greenheck Chair in Engineering Design & Professor of Mechanical Engineering, College of Engineering, Marquette University.

Today, all varieties of robots and self-balancing transporters are conceived, modeled, simulated, controlled, and virtually prototyped before construction in a way that all engineers embrace.

This is an age of diminishing meaningful human interaction. A university engineering program's value then lies in demonstrating the importance of this interaction through faculty-student mentoring and education in the context of real-world, human-centered, team-based problem solving. The automatic code generation and Arduino microcontroller should be a part of that strategy from the beginning of an engineering student's career. Innovative concepts, expressed graphically for all to see and understand, automatically become real-time instructions for a computer, and then become real-world actions to make a difference. **DN**