Graphical Microcontroller Programming

John R. Rogers and R. Clayton McVay
Department of Civil and Mechanical Engineering
United States Military Academy
West Point, NY, USA
john.rogers@usma.edu

Abstract—**Graphical programming techniques developed in an academic setting enable engineers to quickly iterate their robotic algorithms. In a constrained undergraduate environment mechanical engineering students at West Point are not allotted sufficient time in their curriculum to master text-based programming languages. In studying mechatronics, it is desirable for them to program robots to demonstrate useful behavior, and a method that simplifies the programming is necessary. Graphical programming in lieu of text-based programming was used at the Academy and was shown to reduce the time for the students to learn to program without limiting the functional capability of the programming language.**

The method uses Simulink with a third-party chip-specific Simulink blockset that allow programmers to automatically generate executable code for inputs, outputs, and internal functions of the microcontroller chip. PIC32 microcontrollers were used.

It is shown that it is easier to convey the algorithm in the Simulink implementation than it is to convey the traditional C-language implementation of the same algorithm. It is quicker to develop algorithms using the Simulink-based method. It is found that these benefits outweigh the disadvantages associate with the higher level of programming abstraction.

The method is relevant as a software development tool in that it allows an engineer to move quickly from theory to proof-of-concept and into prototyping. The method is scalable to military and industrial applications outside of academia although it is not yet widely used there.

Keywords-programming; algorithm; high-level language; graphical programming; microcontroller; software development

I. Introduction

The design of mechatronic systems requires knowledge in three domains: mechanical engineering, electrical engineering, and computer science. Mechatronics educators face a challenge: provide students with enough knowledge in the three domains so that they can execute meaningful mechatronic exercises. This paper describes an approach taken at West Point to confront this problem.

At West Point an introductory course in mechatronics is offered to mechanical and electrical engineering majors. The course covers microcontroller basics, broad programming concepts, some analog circuits, sensors, actuators, and some system modeling. Traditionally the mechanical engineers who enroll in the course are weak in programming and sensor integration. The electrical engineering students are normally weak in the actuation and kinematic and dynamic analysis parts of the course.

Lectures serve a supporting role to the hands-on activities in the labs and in the course project. An electric-powered radio-controlled vehicle is used in the course project and labs. See Fig. 1. By the end of the course, the student has designed and implemented an autonomous ground vehicle. The prospect of incorporating sensors and embedding closed-loop control on a microcontroller in the unmanned vehicle generates interest and motivates students to learn.

One of the objectives of the course is to acquaint the students with the capabilities of microcontrollers and their benefits: small, low-cost, and re-programmable. Typical mechanical engineering students enter the course with little computer programming instruction: one half-semester of introductory programming concepts in their freshman year, plus one half-semester of computer programming using MATLAB and Simulink in their sophomore year. The electrical engineering students are better prepared in this regard, but do not take a course in dynamics or machine design. The mechatronics course is introductory level so instructors deliberately design course activities at a level incoming students can handle. What is needed is a way for students to develop autonomous vehicle behavior without getting bogged down in programming.

Beginning in the spring semesters of 2011, the students in the mechatronics course at West Point programmed their vehicles using a graphical method. Their results demonstrated that not only can students with limited traditional programming experience develop robotic applications, but that the approach

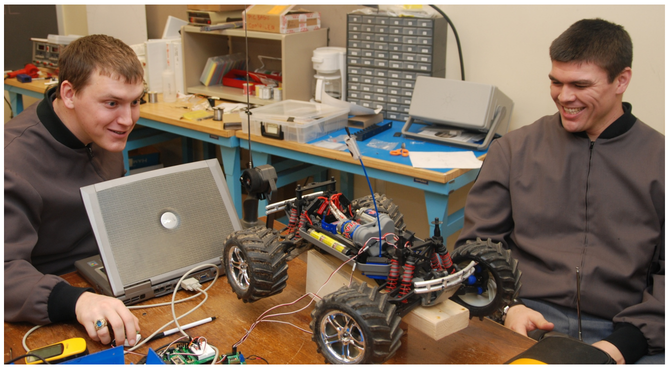
Figure 1. Students develop autonomy for unmanned vehicles.

used could also facilitate design by decreasing the time it takes from product concept to hardware implementation.

II. Prior Work

Prior work has shown that graphical approaches to programming have benefits. Bucks and Oakes [1] used LabVIEW to introduce concepts of programming; their students designed exercises to teach 7th graders math or science concepts. They report that students involved in these graphical programming activities were initially apprehensive about the learning content but became excited when they saw their own progress. End-of-course student surveys noted positive responses to graphical programming with LabVIEW in spite of concerns at the beginning of the semester. The work of Bucks and Oakes did not involve microcontrollers. Burns and Sugar [2] describe the benefits of using MATLAB and Simulink to design, simulate, and implement controls in hardware. The authors show how the use of automatic code generation can mitigate the burden of coding for real-time systems. They cite the bypass of coding as an "obvious benefit." Their implementation of hardware-in-the-loop (HIL) uses the MathWorks' xPC Target in which a stand-alone executable is run in a target personal computer or single-board computer. This method allowed the students to focus on the control design learning points. Their system allows parameter tuning and data collection. The user interface of xPC Target is essentially the same as the method proposed by the authors of this paper—both use Simulink, but xPC Target is not used to program microcontrollers. Giurgiutiu, Lyons, Rocheleau, and Liu [3] cite the challenge of learning a script language such as assembly or C++ when students enter their microcontroller course having been exposed only to visual languages (MathCad and LabVIEW). These authors teach simulation of hand-written assembly code as an aid to learning text-based programming. Other authors [4-6] use Simulink as a simulation tool to teach engineering concepts.

III. Simulink-Based Graphical Programming

The graphical programming interface used in the mechatronics course at West Point is Simulink produced by The MathWorks. Simulink is primarily intended for simulating dynamic systems and is normally introduced to the mechanical engineering students at the Academy as sophomores or juniors. It is a component of MATLAB wherein blocks are mouse-dragged from a palette and connected with lines to create a model. The lines represent variables and are commonly referred to as signals. The blocks symbolize mathematical operations, functions, signal sources, and the like. Subsystem blocks are used to organize models. A large library of general and application-specific blocks allows fast development of complex functions and algorithms. The critical component that enables the method discussed in this paper is the set of blocks, or blockset, that accesses the microcontroller functions such as the digital inputs and outputs. The blockset is an add-on that is obtained in addition to Simulink; it is produced by L. Kerhuel. Programs developed in Simulink are equivalent to C programs in terms of functionality, ability to document, ability to control revisions, and ability to divide labor on teams of students.

The structure of complex algorithms is communicated visually with graphical programming and therefore more efficiently than text-based code. Subsystem blocks may be opened to examine details, and subsystems within subsystems are allowed, analogous to nested functions in text-based code. Simulink is able to maintain version numbers and can be integrated with source control software. These features of Simulink make the projects scalable; in fact Simulink is widely used in industry.

A student-written sample algorithm is presented in Fig. 2. This Simulink model generates code that can be loaded on a microcontroller to perform an "arm and fire" function: press a first button to "arm" a system, and press a second button to "fire" the system. In the classroom implementation, an LED on board the microcontroller development board is illuminated to signify the "armed" state, and an additional two LEDs are rapidly flashed alternately to signify "fire." This algorithm can be used for weapon safety that requires "arming" before firing, and also for home alarm systems where "arm" means to set the system up to detect intrusion, and the "fire" function is to sound the alarm. A combination of blocks from the standard Simulink library and from the add-on blockset are combined to create the program. In Fig. 2, the blue master block is used to configure the specific microcontroller device targeted, to set the processor clock source and frequency, and instruction execution speed (MIPS). The yellow block configures the code generation software so that the code generated is suitable for the specific microprocessor. The Toggle block in Fig. 2 converts the momentary press of the "arm" pushbutton to a maintained state. The Triggered Subsystem contains the "fire" function which rapidly alternates two LEDs. The AND block is the implementation of the logic in this simple algorithm. Pushbuttons and LEDs are wired on the development board to digital input-output (IO) pins of the microcontroller.

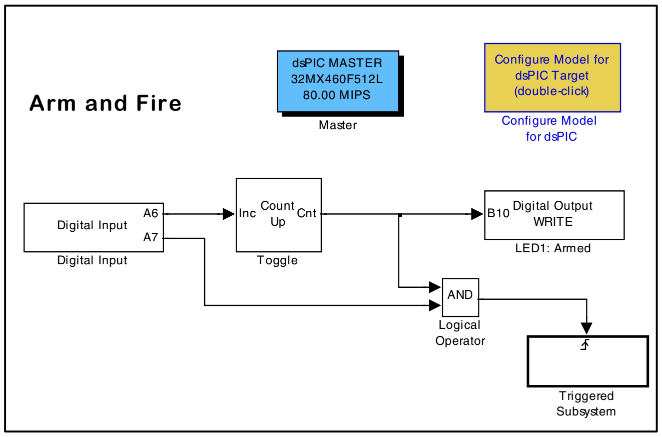

Figure 2. Student-written arm-and-fire algorithm.

A. Supporting Software and Hardware

Several software components must be installed to execute an algorithm such as "arm and fire" on a microcontroller, see Table 1. L. Kerhuel produces a free version and a full version of the blockset; the free version allows six input/outputs maximum whereas the full version is not limited by the software. The blockset is downloaded from the web site and installed in MATLAB. Those familiar with Simulink will recognize the addition of the special blocks to the Simulink Library Browser. The blockset allows the programmer to access all the functions of the microcontroller chip such as serial communications, analog to digital converters, PWM generation, and interrupts.

TABLE I. REQUIRED SOFTWARE

Software	*Vendor*	*Purpose*
MATLAB	The Mathworks	Programming
Simulink	The Mathworks	Graphical Interface
Simulink Coder and Embedded Coder	The Mathworks	C code generation
LK Blockset	L. Kerhuel	Microcontroller IO Interface
MPLAB	Microchip	Loading compiled code on Microcontroller (also used as a C programming development environment)
C32	Microchip	C compiler specific for microcontroller family

In addition to software, microcontroller hardware must be set up and connected to the computer for program-loading. The hardware used in West Point's mechatronics course is the Cerebot32MX4 development board by Digilent Inc. This board is convenient for the course because it has a good balance of size, cost, and functionality. The board includes a PIC32MX460F512L microcontroller from Microchip Corporation, and has provisions for peripheral modules. The blockset created by L. Kerhuel supports many of Microchip's PIC microcontrollers.

A Simulink model is constructed specifically for execution on the microcontroller. Digital inputs and outputs require selection of the port letter and pin number. These are determined from the development board schematic or reference manual. The model is "built," that is, C code is generated and compiled. The free MPLAB integrated development environment (IDE) software from Microchip may be used to inspect and modify the generated code if necessary. MPLAB is used to load the compiled program on the microcontroller via USB connection.

Examples of more complex algorithms programmed using in Simulink exist in the literature. The Arizona State University SPARKy robotic ankle [7-8] and The West Point Bionic Foot control algorithms were developed with Simulink. Both devices are motor-driven and use sensors for feedback. The programs contain nested subsystems (functions) and other advanced programming structures. In the experience of the authors the structures of these algorithms are readily understood by looking at the Simulink diagrams, whereas if the programs were written in a text-based programming language, the same level of understanding would take much longer, and would not be possible without text-based programming knowledge.

B. *Comparison with Text-Based Programming Methods*

Programming in the C language is by far the most common means of developing microcontroller programs. In order to compare the graphical Simulink method to hand-written C, a hand-coded version of the same Arm and Fire algorithm is shown, Fig. 3. Coding in this manner requires detailed knowledge of the C language and knowledge of the microcontroller. This program took an experienced programmer about one hour to write, debug, and revise. References to documents on the internet and the ability to locate header files were necessary. The Arm and

```
// Arm & fire code by Konstantin Avdashchenko
#include <p32xxxx.h>
#include <plib.h>

//Definitions
#define Arm PORTAbits.RA6
#define Fire PORTAbits.RA7
#define ArmedLED PORTBbits.RB10
#define Firing() PORTB = 0x0C00;
#define Hiring() PORTB = 0x1400;

//void Armed(void);
void initports(){
AD1PCFG = 0x0000; //no analog pins
PORTB = 0x0000;
PORTA = 0x0000;
TRISB = 0x0000; //All B ports outputs
TRISA = 0xFFFF; //All A ports Input
PORTB = 0x0000;
PORTA = 0x0000;
}

void Delay200ms(){
int i,r;
    Firing();
    for (i=0;i<3000;i++){
    r++;
    }
    Hiring();

    for (i=0;i<3000;i++){
    r++;
    }
return;
}

void Armed(){
            while (Fire){
            //__delay_ms(200);
            Delay200ms();
            }
    return;
}

int main(){
char Armz=0,Arml,Armd;
    initports();

    while(1){
            Armd=Arm;
            if (Armd&&(Arml==0)) Armz = !Armz;

            if (Armz){
                    ArmedLED=1;
                    Armed();
            }
            else ArmedLED=0;
            Arml=Armd;
    }
```

Figure 3. Arm-and-fire in the C language.

Fire Simulink model was reproduced in about half the time and with less specialized knowledge. It is clear by comparing the code above to Fig. 2 that visual learners, as most engineers are, would likely understand the algorithm more quickly by looking at the Simulink model than by looking at the C code. In the same manner, troubleshooting of common problems is easier in Simulink than in the C-based code. As with C, the programmer using Simulink can easily test his system by visually building his/her program incrementally to verify that different parts work. This can lead to a functioning program quickly with fewer errors.

IV. Results and Conclusions

A. Implementation in the Classroom

In the 2009 and 2010 iterations of the Mechatronics course at West Point, programming was done in C for an Atmel ATMega 128 microcontroller (Robostix). An autonomous vehicle platform based on the radio controlled E-Maxx truck. Only one of four 2009 student groups and none of the 2010 groups were able to develop algorithms on their own in C. This failure is attributed to insufficient student background and insufficient time in the course rather than to lack of ability. Adding programming content to the mechanical engineering program to remedy this deficiency is not feasible due to the many constraints at the Academy.

The method described in this paper was implemented in the course for the first time in spring 2011. The same vehicle platform was used, but with the PIC32 microcontroller. At first, during the spring semester, the Simulink implementation required persistent work to overcome errors, software version incompatibilities, and communication issues. The initial learning of the method was a burden, and this was borne largely by the course instructor. In spite of heavy coaching by the instructor, the students' view of the installation of the software and the struggle to get the first program running on the hardware was that it was burdensome. The instructor maintained motivation in the face of the startup challenges by framing the notion that we were at the frontier of microcontroller programming education, and that the class effort would make it easier for the next semester's class. This view was acceptable to the students and ultimately proved correct. Course credit was given for correct installation of all the software components on the student's computer. Credit was also given to students for documenting questions and solutions in a Frequently Asked Question file. After this "overhead," programming the microcontrollers was satisfying for the students.

The spring semester instructor observed significantly more progress by the students using Simulink for programming than by the students coding in C of prior years. Students were able to write simple programs somewhat more complex than the "arm and fire" example. Student work was archived for future semesters. During the most recent fall semester the students starting building working "arm and fire"-type programs by the fourth lesson. They proved more willing to experiment with the different functions of the microcontroller on their own and were more open to complex programming routines. This allowed the instructor to make assignments and labs that built upon each other enabling all the students to complete their autonomous vehicle by the semester's completion.

Once the students completed their initial simple designs they were quickly able to augment and increase the complexity, and thus the capability, of their designs through an iterative process. The students' first exposure to the microcontroller required them to make two LEDs alternately blink. The next labs required them to incorporate sensors, then actuators, and then finally they were required to control their autonomous vehicle through closed-loop control. They quickly developed a prototype of their semester design project. Several students in the most recent semester have readily extended their programming experience beyond the mechatronics course to build intelligent capstone projects. This fact shows a significant advance in mechatronics education at West Point attributable to the graphical method.

Other graphical methods of programming microcontrollers exist. Microchip Corporation also has a Simulink blockset comparable to the L. Kerhuel blockset. The Microchip blockset does not yet support the PIC32 chip. Scicos is open-source graphical programming software similar to Simulink. Scicos is free, but is not as extensive as Simulink, and the number of chips supported is smaller than the other options. LabVIEW also is a graphical programming method that can be used to program microcontrollers. Flowcode is software in which the programmer builds a flowchart to develop the algorithm. The authors do not have enough experience with these alternative methods to compare them meaningfully. A thorough analysis of the relative benefits would be useful, but is outside the scope of this paper. The authors would expect similar benefits for other graphical programming software.

B. The Programming Abstraction Continuum: Advantages amd Disadvantages of the Method

Programming languages vary in their degree of abstraction. The least abstract is binary machine code; at this level every aspect of the silicon machine hardware is relevant and visible. High-level languages such as Java, C++, and MATLAB, are at the opposite end of the abstraction continuum; these languages are designed for problem solving application. An engineering trade-off is made when a programming language is selected. High level languages are best for quick problem solving and analysis. For example, MATLAB is frequently utilized for visualization of scientific data. The language has hundreds of special purpose built-in commands. In the case of MATLAB, details such as data type and memory allocation are hidden from the programmer. These make for fast algorithm iteration, but the development speed has a cost: the execution speed and memory utilization will not be optimized; when such considerations dominate, a low-level language like assembly or a mid level language like C is called for.

The Simulink method of microcontroller programming is at a very high level of abstraction. Simulink hides the low level details such as configuration registers and the detailed function of microcontroller peripherals such as timers and analog-to-digital conversion.

A benefit of the Simulink method is its ability to enable the student to troubleshoot code. Inherent within Simulink is the ability to visually display data types of variables as the program is built. It also has the capability to send data through the microcontroller's UART modules over serial communication lines to a computer. The computer uses terminal software to display variable data at different points within the program while it is running. This added benefit has reduced student frustration, increased student understanding of basic microcontroller peripherals, and decreased time to troubleshoot software and hardware.

A drawback of graphical programming is that the automatically generated C code is not as efficient as hand-written in terms of program memory and computational efficiency. In the academic environment the trading efficiency away for ease of use is sensible.

C. Student Perception

In general, the graphical programming appealed more to the mechanical engineering students than to the electrical engineering students. The mechanical engineers had previously used Simulink in other courses while the electrical engineers had developed text-based code in Arduino in their prior courses. Based on end-of-course evaluations the mechanical students liked the familiarity, ease of use, and intuitive nature of the graphical programming. The electrical engineers disliked the added layer of abstraction and lack of existing workable code available to them for use as a reference. The following is a quote from end-of-course student feedback:

> "I prefer to program in C, and not using Arduino or MATLAB. High-level methods of programming obscure the details of how things are being done in the microcontroller and make trouble-shooting more difficult. Additionally high-level programming takes up space and processor time on the microcontroller, degrading performance (as shown by the fact we had to use a really slow loop interval in Simulink of 1ms)."

Programming in C is the norm for mainstream microcontroller programming and makes sense in academic environments where C programming experts reside as a student resource. In future semesters the students will be allowed to program using Arduino or Simulink.

D. Conclusion

Observing how much more the 2011 and 2012 mechatronics students achieved when using graphical programming instead of hand written C code has convinced the course instructors that the graphical method makes microcontrollers accessible to those who would otherwise be overwhelmed. Students were motivated to learn more.

The Simulink method enabled students to achieve results with microcontrollers whereas the attempt to program in C had near-zero success. This success is attributed to the visual aspect of the program interface. The fact that success left the students feeling satisfied and motivated was a clear indictor of the benefit of the programming method used. The Simulink method can be expected to work well in circumstances similar to the situation described in this paper: individuals with little traditional programming experience who are willing to spend the time to self train, and a computing environment where MATLAB and Simulink are present. If MATLAB and Simulink are already present, the cost to get started is low. The Digilent boards are not costly, MPLAB is free from Microchip, the C compiler (C32 lite) is free for academic use, and the Kerhuel blockset free version is sufficient to get started. The Simulink method is especially attractive for those who want to understand microcontroller capabilities, and who have difficulty learning, or don't have time to learn, C programming.

Graphical programming at West Point will increase cadet understanding of mechatronic applications and facilitate a more rapid prototyping process. It is the authors' hope that the introduction of this method will enable cadets to produce more meaningful projects and apply the method in both the military environment and in industry after they graduate.

Acknowledgment

The authors express their thanks to Chris Korpela for his assistance with C coding in the mechatronics course, to Ben Flanick for his contribution of the Arm-and-Fire Simulink model, and to Konstantin Avdashchenko for the C code version of the Arm-and-Fire program.

References

[1] Bucks, G., Oakes, W., 2010 "Integration of Graphical Programming into a First-Year Engineering Course," ASEE Annual Conference 2010-1431.

[2] Burns, D., Sugar, T., 2002, "Rapid Embedded Programming in the Mathworks Environment," Journal of Computing and Information Science in Engineering, 2, 237-241.

[3] Giurgiutiu, V., Lyons, J., Rocheleau, D., Liu, W., 2005, "Mechatronics/Microcontroller Education for Mechanical Engineering Students at the University of South Carolina," Mechatronics 15, 1025-1036, doi: 10.1016/j.mechatronics.2006.06.002

[4] Bhat, S., Glavic, M., Pavella, M., Bhatti, T.S., Kothari, D.P., "A Transient Stability Tool Combining the SIME Method with MATLAB and Simulink," International Journal of Electrical Engineering Education, 43(2), 119-133.

[5] Zobaa, A.F., Boghdady, T.A., "Integration into Undergraduate Courses of Transformer Tests using MATLAB/Simulink," International Journal of Electrical Engineering Education, 44 (4) 319-332.

[6] Duran, M.J., Gallardo, S., Toral, S.L., Martinez-Torrez, R., Barrero, F.J., 2007, "A Learning Methodology using Matlab/Simulink for Undergraduate Electrical Engineering Courses Attending to Lerner Satisfaction Outcomes," International Journal of Technology and Design Education, 17 55-73.

[7] Bellman, R.D., Holgate, M.A., Sugar, T.G., 2008, "SPARKy 3: Design of an Active Robotic Ankle Prosthesis with Two Actuated Degrees of Freedom Using Regenerative Kinetics," Proceedings of the 2nd Biennial IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics, Scottsdale, AZ, USA, 511-516.

[8] Holgate, M.A. Bohler, A.W., Sugar, T.G., 2008, "Control Algorithms for Ankle Robots: A Reflection on the State-of-the-Art and Presentation of Two Novel Algorithms", Proceedings of the 2nd Biennial IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics Scottsdale, AZ, USA, 97-10