Brick Sorting Revisited[*]

M. L. Neilsen

Department of Computing and Information Sciences
234 Nichols Hall
Kansas State University
neilsen@cis.ksu.edu

Abstract

With the rapidly advancing capabilities of computing hardware, it is now possible to embed computing capabilities in virtually all manufactured devices. Consequently, there is an increased demand for professionals trained to develop embedded electronic systems. However, the design and implementation of such systems requires a broad knowledge in areas traditionally not covered in any one discipline.

This paper discusses how a conceptually simple brick sorting problem can be used to solidify and unify some of the theoretical concepts covered in a traditional real-time systems course. The problem is to sort Lego™ Bricks of different colors into separate bins using a standard set of sensors and actuators controlled using Lego Mindstorms™ Robotic Command Explorer (RCX) bricks. Although conceptually simple, the problem enables students to connect abstract design concepts with a concrete implementation and to better understand the importance of using an iterative design methodology.

[*] In part, this material is based upon work supported by the National Science Foundation under NSF Grants No. 9980321 and 0227709. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

1 Introduction

The number of embedded electronic systems used in automobiles, industrial automation, and other control systems continues to increase dramatically. These systems typically include subsystems with separate processors. The processors must communicate to coordinate their activities. A typical system consists of an interconnected collection of distributed processors connected by a real-time network. As these systems become even more complex, the need for real-time embedded systems research and students trained in embedded system development become even more critical.

Design and implementation of embedded systems requires a *broad knowledge* in areas traditionally not covered in any one discipline. These areas include electrical and computer engineering, computing sciences, mechanical engineering, and other engineering disciplines. As a result, it is very difficult to train students and engineers within a single discipline to effectively design and implement complex real-time embedded systems. Thus, we felt that it was important to first establish an *interdisciplinary framework* of structured courses for education in real-time embedded system design [5]. One of the major goals of this new curriculum is to expose students to industrial and commercial quality implementations and *bridge the gap between conceptual understanding and concrete implementations*. When undergraduate and graduate students are able to apply abstract knowledge in concrete implementations, subsequent higher-level, theory-oriented concepts have more relevance.

The same problem exists wihin typical upper undergraduate and beginning graduate-level real-time systems courses where students lack an understanding of the connection between the underlying abstract theory and concrete implementations, and professors struggle to motivate students to appreciate the importance of learning those abstract concepts. One solution that has been used in our traditional beginning graduate-level real-time systems course is to develop innovative, challenging design problems that can be implemented using cost-effective hardware and software. This paper describes one such problem called *brick sorting*.

A Brick Sorter is used to sort 2x2 Lego™ Bricks of different colors into separate bins. Although this is a fairly well-known design problem for Lego Mindstorms™ Robotic Command Explorers (RCX Bricks), this paper describes how this problem can be integrated with theoretical design methodology for embedded systems and describes two new design challenges that can be used to further challenge students in a real-time systems course. The first challenge is to sort bricks of two distinct colors into two different bins at the fastest rate possible. The second challenge is to sort different colored bricks with the most precision. From the second challenge, students confront the limitations of the simple color sensor in the standard Lego Mindstorms™ Robotics Invention System. Consequently, the most effective solutions involve both hardware and software design.

Three steps are required to solve this problem. In the first step, a real-time design methodology using Rational Rose RealTime™ or some other design tool is used to specify requirements and develop a real-time model using the Unified Modeling Language extended with capsules and ports to capture real-time requirements and describe the interaction between key system components. The second step is the actual code generation, testing, and debugging using the RCX Brick, a real-time operating system, and (optionally) custom sensors. The final step is to perform real-time validation (via graphical simulation) and verification (via automatic model checking) using a tool such as UPPAAL or Real Time Spin (RT Spin). The system is modeled using timed-automata which are finite state machines with time implemented using real-valued clocks. The simulation is performed by interactively running the system to validate that it works as advertised. Then, the verifier can be used to check reachability properties by exhaustively searching all possible dynamic behaviors of the system. The verifier checks for simple invariants and reachability properties. This simple problem unifies several themes that are emphasized in our real-time systems course, including real-time design methodologies, scheduling theory, testing, validation, and verification.

The next section provides some background information. Then, Section 3 describes the design project completed by students as part of our real-time systems course. Finally, the paper concludes in Section 4 with a summary and recommendations for future work.

2 Background

Traditional approaches to system design in computing sciences have focused primarily on software design, whereas system design in other engineering disciplines has focused primarily on hardware design. With the introduction of inexpensive microprocessors, it becomes possible to provide students with hands-on laboratory experiences to construct simple embedded systems. As these systems have evolved in commercial applications, the number and complexity of embedded controllers has also increased. A significant portion of the design process must now focus on software engineering and the integration of hardware and software. However, most microprocessor-based system courses still emphasize hardware construction [7,8]. In order to address both software and hardware issues, it becomes essential to apply an *interdisciplinary approach* [5].

Many microcontrollers are used in real-time control systems such as automotive electronics and factory automation. To be practical for industry, the per-unit cost must be strictly controlled, but the development platform can be fairly expensive as long as the development cost can be amortized over thousands or millions of units. However, in an academic environment, the cost per development platform must be controlled to fit within a typically constrained laboratory budget. Early in the development process, this was a limitation in trying to establish a collection of inter-departmental laboratories. More recently, we have benefited by the foresight of many leading development platform vendors, both software and hardware. Development environments should support source-level debugging, simulators, profiling, and analysis tools. Many developers are now offering Educational Partner Programs to enable the integration of these sophisticated development tools into the curriculum.

Another frequently required technology is a real-time operating system (RTOS). We currently use both commercial (VxWorks) and open source (ERIKA, LeJOS, BrickOS, etc.) operating systems. We also built an RTOS that provides an efficient and extensible set of services. The functionality of the RTOS includes scheduling and thread context switching capability, synchronization primitives, and micro-interrupt handlers for interruptible peripheral devices. On top of the RTOS, various functions can be implemented as independent threads. All of these real-time operating systems can be used in either *simulation* or *execution* mode.

Due to the lack of time and facilities, traditional university education tends to emphasize theory and concepts. Even though implementation (laboratory) projects are associated with many courses, these projects tend to be more abstract than real implementations that can be used directly in industrial and commercial products. Typically, there is a large gap in students' understanding between theory (conceptual understanding) and implementation (concrete understanding). As a result, many students who have a good understanding of theory and concepts do not have confidence to map their knowledge onto implementations. One of the goals of this curriculum is to expose students to industrial and commercial quality implementations and *bridge the gap between conceptual understanding and concrete implementations*. After students are able to apply abstract knowledge in concrete implementations, subsequent higher-level, theory-oriented courses have more relevance. This paper describes a project that was included as a part of our traditional, theorectical, real-time systems course. Even though the focus in this course is on theoretical concepts, we felt that it was important for student to apply those concepts to small design problems so that subsequent courses that include large, capstone design problems could be managed efficiently and also, so that subsequent theory courses would have more significance.

The availability of powerful microprocessors and development environments supporting high-level languages and formalisms has allowed complex features to be incorporated into embedded systems. In turn, this sophistication has enabled the development of embedded systems to control complex applications having real-time, reliability and safety constraints by utilizing theoretical research advances made in a number of areas such as real-time computing, hardware interfacing, networking, fault-tolerance, and verification. Hence, theoretical research from these areas needs to be applied in practice for the development of high assurance, state-of-the-art, real-time embedded systems. We *incorporate recent advances* in methodologies, development tools and design techniques to develop practical new design methodologies, and apply those techniques in the design of real-time embedded systems. At the same time, the rapid evolution of embedded systems in industry also drives new directions for theoretical research.

Finally, the curriculum allows us to *accelerate applied research* in engineering, and produce significant new embedded systems for numerous applications including variable rate technology for precision farming. This transfer of technology has enabled us to develop even stronger linkages with industry.

The overall objective is to provide opportunities for students with varying engineering backgrounds to gain knowledge and experience in the design and implementation of real-time embedded systems, and to advance the state-of-the-art in design methodologies and real-time applications.

The core curriculum consists of the following four courses:

1. A *remedial* course consisting of three independent modules, intended to bring students with varying backgrounds up to speed,
2. An *implementation* course that allows students to work with state-of-the-art design tools, embedded development environments, and target platforms to interconnect a variety of sensors and actuators in complete real-time embedded systems,
3. A *theory* course, which is the focus of this paper, covering both real-time scheduling theory and contemporary design methodologies, and
4. A *project-based capstone design* course to complete a comprehensive design for a complex embedded system.

This section discusses the layout of our curriculum to provide training to embedded systems designers and programmers. The embedded systems curriculum consists of four semester-long courses at the upper undergraduate/beginning graduate level.

2.1 Remedial course

The first course is designed to be a remedial course for students who do not have a proper background for the subsequent courses in the proposed course sequence. The course consists of three five-week modules; students can take only the necessary modules and earn one credit each.

2.2 Implementation course

In this course, students implement simple but complete real-time embedded systems. The course consists of three modules: real-time programming fundamentals, real-time operating systems, and real-time embedded systems.

2.3 Theory course

This course teaches techniques used in the design and analysis of real-time embedded systems. It also provides students with a strong theoretical foundation for those techniques and a solid background in real-time scheduling theory. In additional to traditional scheduling theory, this course covers elements of the requirements phase, the design phase, and the implementation phase for the design of embedded systems. The requirements phase includes an introduction to the Unified Modeling Language (UML) and object-oriented design methodology, use case analysis, and specification of real-time properties. The design phase covers various aspects such as design patterns, use-case realization, and verification. An emphasis is placed on verification and model checking techniques. This is the course in which the Brick Sorting Problem described in this paper is used to motivate and unify the underlying theory.

2.4 Capstone design course

This course is intended to teach techniques that allow engineering an embedded system to satisfy certain performance requirements. The students must be able to evaluate various design choices and make design decisions accordingly. A major component of this course is an industrial-sized *team project* involving the design and implementation of a complete embedded system. The team ideally consists of students from different disciplines (CIS, EECE, MNE, and BAE). As team structures are increasingly emphasized in industry, this is a valuable experience for students. In particular, since the students are from different disciplines, they learn to work in a synergetic manner, exploiting the strengths of each discipline.

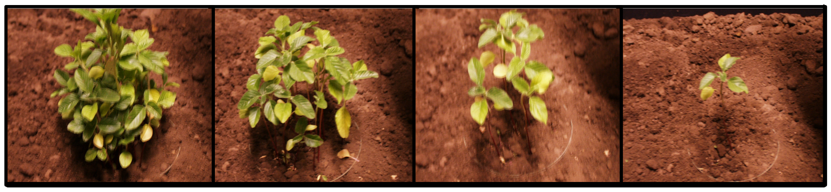


Figure 1. Redroot pigweed at different density levels.

For example, one capstone project focuses on agricultural applications involving variable-rate technology (VRT). Infrared sensors are used to collect information (Figure 1). Then, distributed controllers evaluate the input and generate variable-rate application recommendations in real-time. All sensors, controllers, and actuators are networked together using a real-time controller area network (Figure 2). Applications of embedded systems in industrial and agricultural applications usually involve a large number of various types of *sensors* and *actuators* connected by a real-time network. The rapid increase of such applications requires in-depth research to correctly interface multiple sensors and actuators. These applications serve as excellent case studies to motivate team-based research.

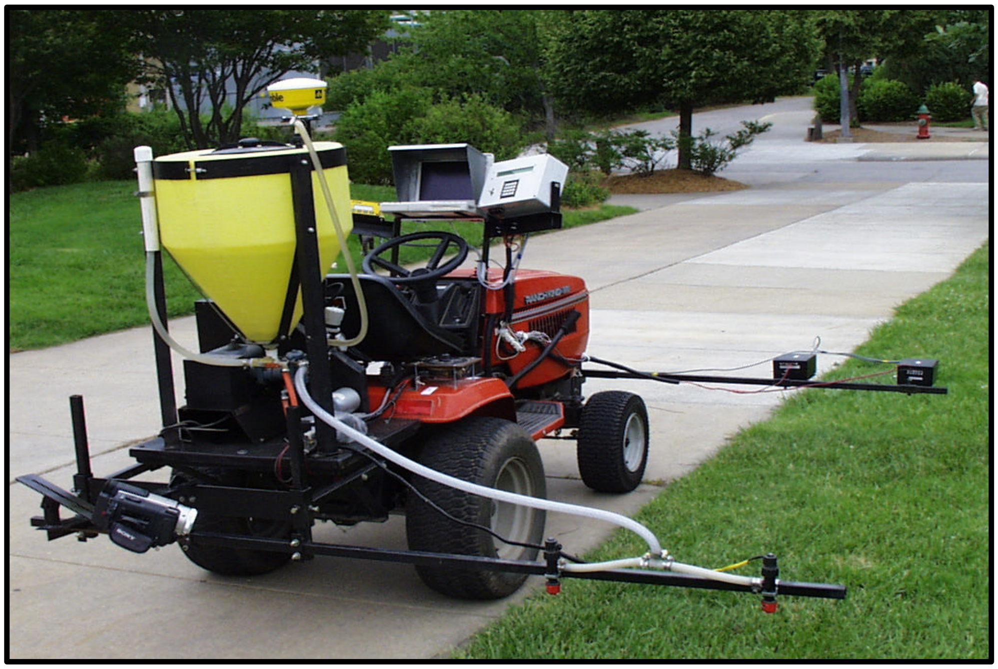


Figure 2. Weed detection system with variable rate applicator.

With the rapid development of new technologies for precision agriculture, more sensors and actuators with sophisticated control algorithms will be added to the system. This requires more complex and reliable networking techniques. We are currently conducting research on real-time image and optical weed sensors, particle flow sensors [11], soil moisture sensors [13], and standing wave sensors. Numerous other sensors have been developed for precision agriculture. Many of these sensors may be linked with a real-time network to log sensory data and provide feedback for real-time control.

The goal of the Brick Sorting Problem is to provide a simple design problem that can be used to motivate concepts covered in the theory-based course on real-time systems.

3. Brick Sorting Revisited

The goal of this project is to provide students with a concrete design experience covering several important aspects of real-time system design. The requirements phase includes an introduction to object-oriented design methodology and the Unified Modeling Language (UML), use case analysis, and specification of real-time properties. The design phase covers various aspects such as design patterns, use-case realization, and verification. An emphasis is placed on verification and model checking techniques. Design methodologies are introduced using IBM's Rational Rose RealTime. Properties of system designs can be verified using several different tools including a freely-available tool from Uppsala and Aalborg Universities in Denmark (UPPAAL) [3, 7] and a real-time extension to SPIN, called Real-Time Spin (RT Spin). Both tools are based on the theory of timed automata [1,2] and include the addition of real-valued clocks to specify real-time constraints. To give students a simple, but relatively challenging, design experience, they were assigned the problem of sorting Lego 2x2 bricks, with a twist.

The sorter should comply with the following requirements:

1. The sorter should be able to sort 2 x 2 Lego bricks of different colors.
2. The sorter should be able to sort bricks of at least two different colors.
3. The sorter should not damage the bricks during sorting.
4. The sorter should count the number of bricks sorted and display the total number sorted.
5. To the extent possible, the sorter should respond to abnormal events.

Students are required to develop two different designs to sort the bricks:

1. **Speed:** In the first design, the brick sorter is only required to sort bricks of *two* different colors, black and yellow, but the goal is to sort them as fast as possible. Each design was tested by loading their chutes with the same "random" sequences of bricks – some of the sequences used in the test were not very random; e.g., black, yellow, black, yellow, etc. Then, an average run time was computed for the same 10 sequences. The best design was able to sort 20 random bricks in an average of 4.6 seconds by flipping the bricks either left or right as the fell from the chute.

2. **Precision:** In the second design, the brick sorter is required to correctly sort as many different colored bricks as possible by placing bricks of the same color into the same bin. In this case, the best design, shown in Figure 3, was able to sort up to 8 different colors consistently. The bins were moved on a track and every brick of a color seen for the first time was placed in the first empty bin. Whenever a brick of the same color was seen again, then the bins were repositioned under the chute so that the brick would fall into the correct bin. In this case, it was important to carefully calibrate or train the sensor to detect changes in color.

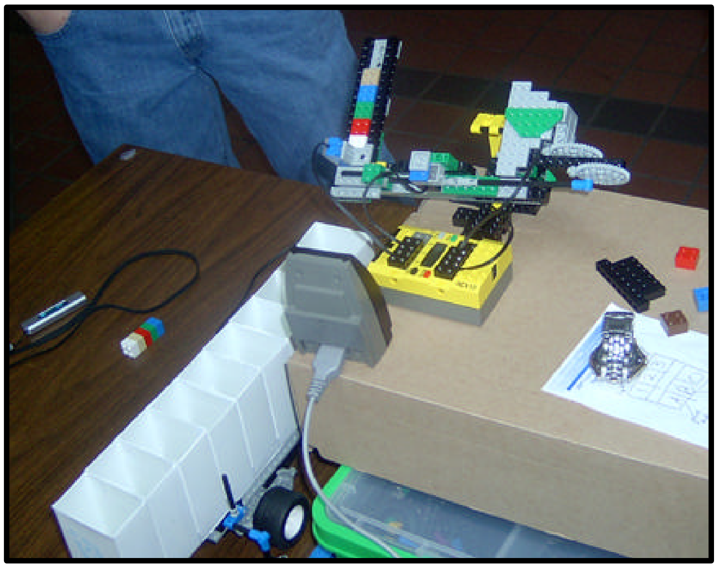


Figure 3. Most precise brick sorter.

To provide a cost-effective solution for a complete embedded system that is complex enough to solve interesting problems, the Lego Mindstorms™ Robotics Invention System (RIS 2.0) is used. RIS 2.0 is a very powerful educational tool disguised as a toy. The heart of the system is the RCX Programmable Brick (RCX 1.0), a microprocessor-based embedded controller housed in an over-sized Lego™ brick (see Figure 3). Built into the battery-powered RCX are three A/D inputs, three 9-volt outputs and an infrared (IR) link for communication with a host computer or other controllers. RIS 2.0 also includes 2 motors, 2 touch sensors, one light sensor, over 800 Lego™ pieces (plates, blocks, gears, axles, etc.) and software designed to allow students to program the robots designed using an object-oriented design methodology and programmed using Java or some other high-level programming languages.

For each design problem, the students had to complete the following three steps using an iterative design methodology:

1. Complete the Requirements Analysis and develop a Design Model using Rational Rose Real-Time. Verify the feasibility of the resulting task set based on the Scheduling Theory covered in class.

2. Design an UPPAAL or RT Spin Model for the brick sorter designed in Step 1. Verify both safety and liveness properties to ensure the correct operation of the system. Include all queries used to verify system properties.
3. Implement and Test the Design using Lego™ Mindstorms Robots. The programming language and operating system could be selected by the developers.

The following sections discuss each of these steps in more detail.

3.1 Design Methodology

Requirements can be specified by using Use Case Diagrams augmented with real-time requirements. Then, a Design Model can be developed using Capsules and Ports.

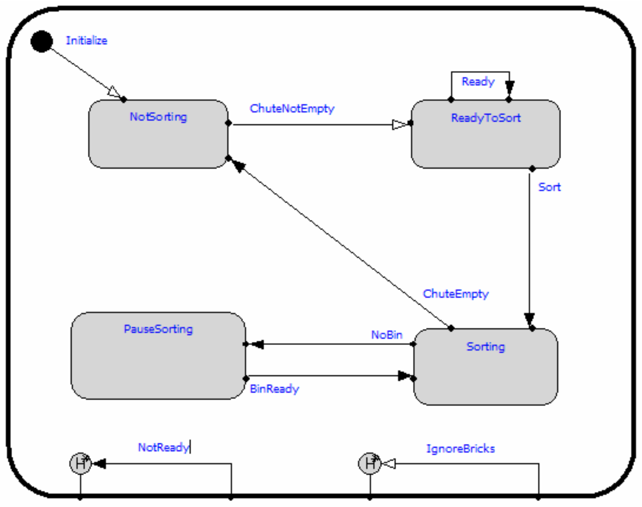
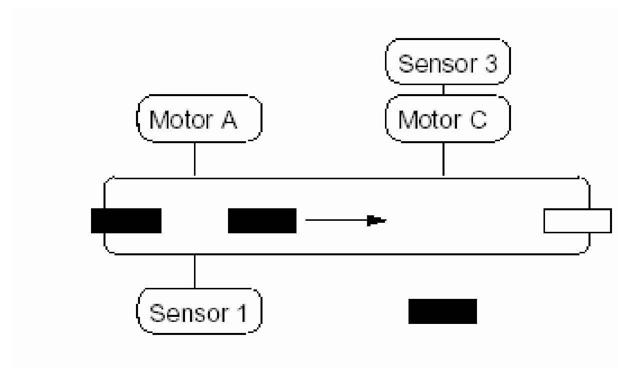
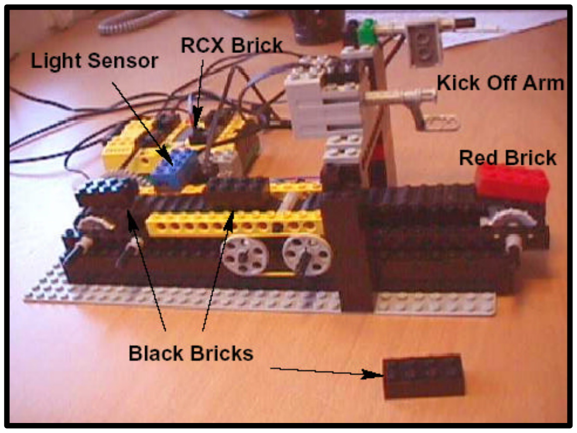


Figure 4. Statechart.

A capsule is just a lightweight concurrent class with ports. Capsules are highly encapsulated and can only communicate via message passing through their end ports. Capsule behavior is defined graphically (as shown in Figure 4) through the use of hierarchical state machines or statecharts. These state machines support run-to-completion semantics which simplifies synchronization constraints.

For example, a typical solution to the brick sorting problem might involve a small set of capsules, one to control the Brick Chute, one for the Color or Light Sensor, one for the Track, Belt or Arm Revolver to get the brick lined up with the correct bin, and one to control the Push Piston or Kick Off Arm to eject a brick.





```
int b=0, active1=0, active2=0;
int DELAY=25;
int LIGHT_LEVEL=42;

task main{
   Sensor(IN_1, IN_LIGHT);
   Sensor(IN_3, IN_SWITCH);
   Fwd(OUT_A,1);
   start kick_off;
   while(true){
      wait(IN_1<=LIGHT_LEVEL);
      if(b==0){
         ClearTimer(1);
         active1=1;
      }
      if(b==1){
         ClearTimer(2);
         active2=1;
      }
      b=-b+1;
      wait(IN_1>LIGHT_LEVEL);
   }
}
```

```
task kick_off{
while(true){
   wait(Timer(1)>DELAY && active1==1);
   active1=0;
   Fwd(OUT_C,1);
   Sleep(6);
   Rev(OUT_C,1);
   wait(IN_3==1);
   Off(OUT_C);
   wait(Timer(2)>DELAY && active2==1);
   active2=0;
   Fwd(OUT_C,1);
   Sleep(6);
   Rev(OUT_C,1);
   wait(IN_3==1);
   Off(OUT_C);
}
}
```

Figure 5. Brick Sorter with defective NQC code [4].

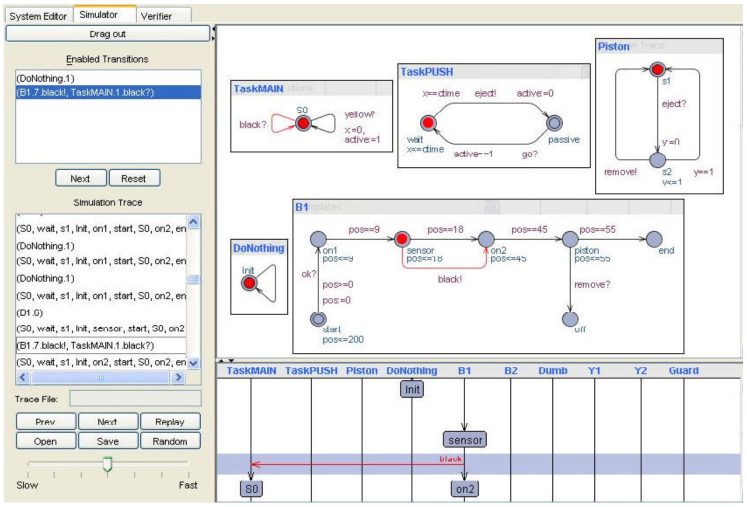
Developers generally focus on techniques that allow an embedded system to satisfy certain performance requirements. They must be able to evaluate many different design choices and make design decisions accordingly. Developers must understand techniques and patterns to satisfy constraints related to real-time, fault-tolerance, and correctness. These goals can only be accomplished by using a well-defined engineering approach starting with a high-level design specification which leads to a well-structured component-based implementation.

After developing a model, the designer can step through the execution sequences generated by randomly or intentionally interjecting events into the simulation of the model. The same model can be used as the basis for a model for validation and verification in the next step.

3.2 Validation and Verification

An important step is to validate the design and verify that the design satisfies safety and liveness properties to ensure its correct operation. The Brick Sorter Problem provides an opportunity to bridge the gap between theory and practice so that participants can apply theoretical knowledge to practice.

UPPAAL2k is a modeling, simulation, and verification tool for real-time systems modeled as networks of timed automata extended with real-valued clocks [3,7]. UPPAAL can be used to check both reachability and invariance properties of a combination of automata locations, clocks, and integer constraints. In UPPAAL, E<>f expresses that it is possible to eventually reach a state satisfying f , and A[ ]f expresses the invariance that f is satisfied over all time.



```
(DoNothing.1)
(B1.7.black!, TaskMAIN.1.black?)
```

```
(S0, wait, s1, Init, on1, start, S0, on2, en
(DoNothing.1)
(S0, wait, s1, Init, on1, start, S0, on2, en
(DoNothing.1)
(S0, wait, s1, Init, on1, start, S0, on2, en
(D1.0)
(S0, wait, s1, Init, sensor, start, S0, on2
(B1.7.black!, TaskMAIN.1.black?)
(S0, wait, s1, Init, on2, start, S0, on2, en
```

Figure 6. UPPAAL simulation of brick sorter.

For example, a user might want to test the invariant that it is always the case that Yellow (or Red) bricks get kicked off; that is they never arrive in the "end" state. This can be specified using the query: `A[] (not Y1.end).` Likewise, Black bricks should always be kicked off, so test the invariant: `A[] (not B1.off).`

3.3 Results

Students were given a total of four weeks in the middle of the course to complete the project. They were also enticed to participate in a friendly competition during the Open House festivities at Kansas State University, with small prizes for the top 3 teams. There were a total of 13 teams, with 1-3 students each, completing the project. A total of 28 students were in the class. All of the teams were able to sort at least 4 bricks and sort the set of 20 bricks within 2 minutes. The fastest design, shown below in Figure 7, sorted all 20 bricks in 4.6 seconds. There were some interesting designs from both a hardware and software perspective. Some of the designs were not very structurally sound, and at the other extreme, some of the designs incorporated very simple software designs. An important aspect was the discussion that ensued after each team presented their results. This year, we plan to use a more iterative approach to let teams receive more feedback after each design phase.

Students were not required to use any particular operating system or programming language. Most of the designs were completed using C and BrickOS (a C-based RTOS), or Java and LeJOS (a Java-based RTOS). A few of the teams chose to use no RTOS; e.g., NQC. The most successful designs incorporated an operating system.

Figure 7. Robotics competition at KSU Open House.

4 Conclusions

With the rapid advances in technology, it is now possible to embed computing capabilities in virtually all manufactured devices. To realize the full potential of this technology, embedded system developers must be trained to manage the complex design problems that are entailed. An important factor is the recognition that sound solutions require an understanding of concepts not covered in any one discipline, and that students with the preparation and desire are needed to acquire the technological knowledge required to be successful in embedded design programs.

This paper presents a practical design problem that can be easily incorporated into existing real-time system design courses. It also shows how the Brick Sorting Problem was integrated into our real-time systems course to enable students to have an opportunity to practice their abstract skills in a concrete setting. Two new variants of the problem based on Speed and Precision are suggested for an additional challenge.

Our embedded systems curriculum is still a work-in-progress, and will certainly evolve, just as the technology and needs of industry evolve. Specific details for each of the designs are available on-line at http://www.cis.ksu.edu/chert. The continued rapid evolution of real-time embedded systems and development tools will provide us with interesting challenges and unprecedented opportunities.

Bibliography

[1] R. Alur and D. L. Dill, “Automata for modelling real-time systems”, In Proceedings of the International Colloquium on Automata, Languages, and Programming, Vol. 443 of Lecture Notes in Computer Science, pages 322-335. Springer-Verlag, 1990.

[2] J. Bengtsson and W. Yi, “Timed automata: Semantics, algorithms and tools”, In Lecture Notes on Concurrency and Petri Nets, W. Reisig and G. Rozenberg (eds.), LNCS 3098, Springer-Verlag, 2004.

[3] A. David, M. Oliver Möller, and W. Yi, “Verification of UML statecharts with real-time extensions”, Technical Report, Uppsala University, 2003.

[4] T.K. Iversen, K.J. Kristoffersen, K.G. Larsen, M. Laursen, R.G. Madsen, S.K. Mortensen, P. Pettersson, C.B. Thomasen, “Model checking real-time control programs verifying LEGO Mindstorms systems using UPPAAL”, In Proceedings of the 12th Euromicro Conference on Real-Time Systems (ECRTS'00).

[5] M.L. Neilsen, D.H. Lenhert, M. Mizuno, G. Singh, N. Zhang, and A.B. Gross, “An interdisciplinary curriculum on real-time embedded systems”, In Proceedings of the 2002 American Society for Engineering Education (ASEE) Annual Conference and Exposition, Montreal, Quebec, 2002.

[6] M.L. Neilsen, “A flexible real-time transport protocol for controller area networks”, In Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA’01), pp. 250-256, June 25-28, 2001.

[7] P. Pettersson and K.G. Larsen,“UPPAAL2k”, In Bulletin of the European Association for Theoretical Computer Science, Vol. 70, pages 40-44, 2000.

[8] J. Wei, N. Zhang. N. Wang, D. Lenhert, M. Neilsen, M. Mizuno, and G. Singh, “Design of an embedded weed-control system using Controller Area Network (CAN)”, ASAE Paper No. 01-3033, American Society of Agricultural Engineers, ASAE 2001 Annual International Meeting, July 29 – August 1, 2001.

[9] W. Wolf, “Rethinking embedded microprocessor education”, In Proceedings of the 2001 American Society for Engineering Education Annual Conference and Exposition, Albuquerque, NM, 2001.

[10] W. Wolf and J. Madsen, “Embedded systems education for the future”, In Proceedings of the IEEE, 88(1), pp. 23-30, January 2000.

[11] N. Zhang, “DSP signal processing for a particle velocity sensor”, In Proceedings of the American Society of Agricultural Engineering, Number 98-3036, 1998.

[12] N. Zhang, R. Taylor, S. Runquist, E. Runquist, M. Schrock, and S. Staggenborg, “A field-level geographic information system (FIS) and precision agriculture”, In Proceedings of the International Conference on Agricultural Engineering, Dec. 1998.

[13] N. Zhang and N. Wang, “Effectiveness of a polarized laser light in soil moisture-content measurement”, In Proceedings of the American Society of Agricultural Engineering, Number 99-3113, 1999.

Biographical Information

MITCHELL L. NEILSEN is an Associate Professor in the Department of Computing and Information Sciences at Kansas State University. His research interests include real-time embedded systems, distributed systems, and distributed scientific computing.