Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

THE DESIGN OF A SOLID-STATE PHYSICAL MODEL OF AN AUTOMATED SYSTEM TO BE USED AS A TEST BED FOR CONTROL APPLICATIONS

Fernando G. Gonzalez
Alicia Helton
Douglas Helton
Jeffrey Smith
Eileen Thompson
Gerry Walterscheild

Department of Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL. 32816, U.S.A.

ABSTRACT

In order to develop, test, and validate control software for managing automated systems, laboratories have traditionally constructed experimental test beds using actual physical equipment (small scale). These experimental systems typically occupy a large amount of lab space, cost thousands of dollars to construct, and require considerable human expertise to operate. Using dedicated micro-controllers (programmable logic controllers), we have proposed the use of a solid-state physical model of an automated system which faithfully replicates the operating characteristics of an ensemble of physical equipment that would typically comprise an automated system. In this paper we present the design of a solid-state physical model of a Flexible Manufacturing System (FMS). Solid-state models have several unique advantages over the traditional models. First, they are inexpensive and can easily be replicated at other laboratories. Second, they can be easily reconfigured to consider alternative scenarios. Third, they can consider an emulated environment that is far more complex than those that are typically addressed by models using actual equipment. Finally, they are totally reliable and safe, and require minimal expertise to operate. This paper discusses the design and operational characteristics of the solid-state model along with its anticipated uses and current limitations.

1 INTRODUCTION

To test software that can control an FMS or other automated systems, laboratories typically use a small-scale, relatively inexpensive analog of an automated system like an FMS. See Davis et al. (1994). In developing a model of an FMS, the material handling system can be modeled with an electric train (replicating an automated guided vehicle), a robotic arm, or perhaps a small conveyor belt. The actual processing, for example the cutting of a part, may not be performed in the model. These physical models are usually much smaller than a real manufacturing plant. Often, they are constructed on a table or within a small room. Furthermore, the model is much cheaper than a real plant. Using a physical model, control software developers can test their software on these small-scale automated systems so that the software is not run on a real system without proper testing see Gonzalez and Davis (1997a) and Gonzalez and Davis (1997b). This approach provides an advantage over simply testing the software on a computer, since it requires the developed software to actually interact with physical equipment, as will be the case when the control software is implemented upon a real system.

Developing and testing control software using physical models, however, presents several problems. The first is the cost of the model. Even though these models are relatively cheap when compared to a real manufacturing plant, they are still expensive, particularly when compared to the alternative of simulating the system with software. Another problem is the size of the physical model. Again while the model is much smaller than a real manufacturing plant, it still requires a large laboratory. This can pose a problem if laboratory space is scarce. The last major problem with physical models—and perhaps the biggest problem—is their poor reliability. Since these models are much cheaper than a real system, the quality—and therefore the reliability—of the components is relatively low. Some models use reliable off-the-shelf robotic arms. But in a real manufacturing plant, robotic arms are seldom used as the primary material handling systems except for moving material in a local area, such as within a cell. Instead, conveyor belts or automatic guided vehicles (AGVs) are used to move material between machines or different parts of the manufacturing plant.

We have previously built a physical model of an FMS. In this model, to emulate an AGV, we chose to employ an HO-scale electric train. Figure 1 shows a picture of our first model and Figure 2 shows its diagram. Note in Figure 1 the AGV is implemented with an electric train. The structure in the front part of the model represents a fixturing station. The four similar structures shown in the rear part of the model represent four machine stations. In Figure 2 the train track connects each machine to each other. This model has 4 machine stations and 1 fixturing station. When one attempts to miniaturize a system, more precision is often needed. Since our electric train was never designed to provide the required precision, it became difficult to run the model for several hours without a failure. In order to conduct extended experiments that would require the model to run for weeks, or perhaps months, at a time without stopping, one must constantly monitor and service the system.

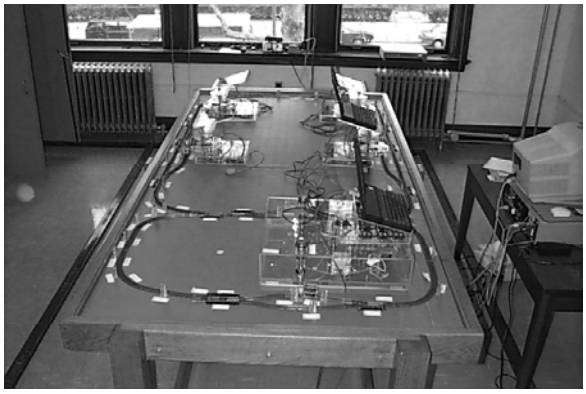

Figure 1: A Picture of our First Physical Model

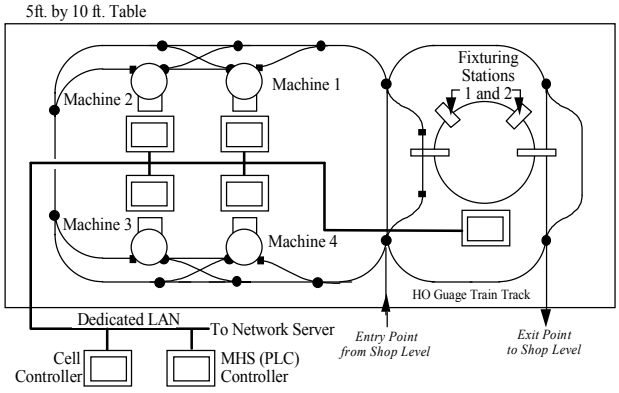

Figure 2: The diagram of our first model

This presents the control system designer with a dilemma. On one hand, one can use a computer simulation of the system to test the control software under development. See Davis et.al. (1993) and Kelton et.al. (1998). This option is cheap and very reliable, but the results are not as credible as the results derived from using the software to control an actual physical system. On the other hand, if one employs a physical model of a real system, it may provide credible results, but the reliability problems may prevent one from conducting experiments of a duration that is long enough to produce usable data.

In this paper, we propose a solution to this dilemma. A physical model of a flexible manufacturing system has been designed that is completely composed of solid-state electronics with no moving parts. This approach represents a compromise between a physical model, which has many unreliable moving parts, and a software simulator, which allows a minimal reality check. In the first part of this paper, we will justify the use of solid-state electronics instead of real physical parts in the model. We will accomplish this by showing why the results of running the control software using simulation are not as credible as the results from running the software on a real system and by carefully studying what parts of a physical model are necessary to the development and testing of control software. In the second part, we will present our solid-state physical model that provides all of the facilities of a conventional physical model and yet is much cheaper, reliable, flexible, smaller, and even portable.

2 BACKGROUND INFORMATION

The following section describes the concepts that allow us to eliminate all of the moving parts in our model and yet perform as a conventional physical model with moving parts. This model is used in research relating to the development of control algorithms such as that performed by our research team. See Gonzalez and Davis (1999).

2.1 Why We Need Physical Models

Why not simulate the system in software instead? Certainly, testing the software on a computer-simulated system is often the first step. However, there are many facets associated with the operation of a physical system that simply are difficult to test in software only. Among these are the issues arising from the fact that the control architecture is usually distributed across a network of computers and communication requirements among the distributed computing processes are a major concern. It is often difficult to adequately model the communications requirements using software alone. Furthermore, it is often difficult to foresee all potential deadlock situations that can arise and include these within the software simulation of the system. At the same time one might ask why not develop the system using the actual manufacturing plant? At some point, one will need to migrate the control software to the real system. However testing and debugging the control software on the actual system can be time consuming and complex. While this testing occurs, often production must be either interrupted or postponed.

Employing a physical model to test control software provides an intermediate step allowing the software to be tested on a physical system as it is being developed. The model replicates many of the factors that are difficult to model in software only, including the communication requirements associated with the distribution of the control processes and some timing issues. Using the model, the researcher has a realistic understanding of how the system will operate which simply cannot be achieved with software simulations.

In order to demonstrate the need for a physical model, consider controlling an automated guided vehicle (AGV) that is being emulated with an electric train. To move an AGV (train) from one location to another in the physical model, one has to issue a command to the programmable logic controller (PLC) to apply a voltage on the specified segments of the train track. An electric train occupying the powered track segment will then start to move. If the step to apply the voltage is skipped, the train will not move. On the other hand, in a simulation model, the train is moved by redrawing its location as a function of time. Thus, a data structure is being modified to reflect the train's trajectory. The simulated control program may send commands to apply power to the tracks but the simulator basically ignores these commands since there are no actual train tracks to power. It is possible that the control program could accidentally turn on the wrong track segment, or perhaps simply not turn on any track segment at all, and yet the train would be moved correctly in the simulation because the controller's command to the simulator to move the train was correct. Even if the simulator checks for the correctness of the commands that are supposed to go to the physical system, it may still be possible for the simulator to have an error and not check properly. In this case, one is simply replicating the function of the controller in the simulator. If this is done, it becomes difficult to determine if a given control function is being executed correctly by the controller or being compensated for incorrectly by the simulation.

Another concern that arises in using a simulator alone is that physical processes must be sensed. In the real physical system there will be sensors located at various location in the plant to measure numerous state variables. For example, the location of an AGV is sensed as the AGV passes across a sensor. In Figure 3 note the three magnets on the train that are used to trigger the three Hall-effect switches on the post. There is only a finite time interval during which the AGV will be in front of a sensor. If the computer is not monitoring the sensor when the AGV is at the sensor post, its passing the sensor can go undetected. Again these are real-world timing concerns are difficult to assess using software testing alone.

2.2 What Part of the Physical Model is Necessary

The feature provided by the physical model, which prevents the control software from accidentally "cheating",

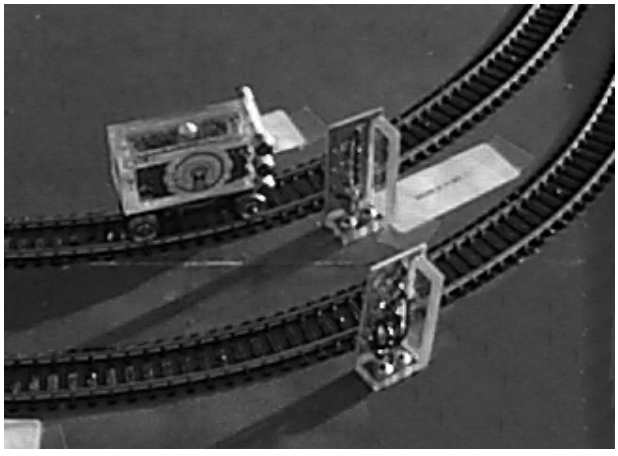

Figure 3: A Picture of an AGV Approaching a Sensor Post

is its real-world physics. For example, in order to move the train in the physical model, the control program must apply a voltage to the tracks. The train cannot move without this voltage. Furthermore, the model will detect the position of the train when the train moves across a sensor. Until the train is sensed, the controller does not know the state of the train. Once the train is sensed it may be necessary to stop the train to prevent it from passing its destination. The only way to stop the train is to cut the power to the track segment. This must be done quickly. The train will not stop and wait for the control program to execute the section of code that cuts the power to the track. A delay in cutting the power will cause the train to pass its destination. Timing is crucial.

Certainly, the physical model which employs the electric train to emulate the physics of the AGV can provide more realism than the simulation model. However, the train was demonstrated to be unreliable. Based upon our prior experience in constructing our first model, we have developed a compromise solution, which provides realism with reliability. In the case of emulating an AGV, the small-scale electric train has been abandoned. In its place, we have employed a specially programmed microcomputer chip to emulate the physics of the AGV. Like the train, the AGV accepts input voltages from the controller. Since our model has an independent circuit consisting of PLCs and other electronics, we will refer to the circuit used to model the physics as the physics PLC and the circuit controlling the model as simply the controller PLC or just controller. The key issue here is that there is absolutely no communication between the controller and the physics PLC. To control the model the controller must apply the correct voltage signals to the physics PLC. It can not simply "tell" the physics PLC to move the AGV, for example, as is the case with software simulation.

Once the controller applies voltage to the appropriate locations in the model, the physics PLC then senses these voltages and models the movement of the AGV according

to the location of the voltage signals. During the period of time that the AGV is in front of a sensor, the physics PLC outputs a signal in the same way the physical model consisting of a real electric train would do. The control must correctly detect this output signal and interrupt the voltage to the physics PLC in order to stop the train. If the controller does not detect the outputted sensor voltages, then the voltage is not interrupted and the physics PLC continues to move the train. In this way, the dedicated physics PLC faithfully replicates the physical dynamics of the train movement along the track. It is up to the controller to manage its movement using the same mode of operation that it would employ to manage a real system. That is by the application of voltage at specified locations and time and by the sensing of electrical pulses generated from the sensors. In the next generation model, all of the mechanical processes are replaced by dedicated PLCs that faithfully replicate the behavior of a physical component. In this manner, we can achieve realism and reliability. Note the key issue here is that there is no communication between the control software being tested and the PLC modeling the physics. This assures that the control software must correctly stimulate the model in order to have it respond correctly. Unlike software simulation, no accidental "cheating" by simply telling the physic PLC its desired outcome is possible. The following section presents the design of our emulator.

3 THE SOLID-STATE MODEL

3.1 Modeling Physical Movement

In our emulator all of the moving parts are implemented using solid-state electronics. Note that since no actual processing of material such as cutting, welding, bending, and so on is performed, our emulator like most others only models the movement of entities within the emulated system. In our previous emulator we modeled the AGV with an HO-scale electric train. The machine station was modeled with a plastic disk holding 6 electromagnets. The disk had an AC motor that turned the disk as 1 rpm and the electromagnets could be turned on or off in order to pick up and drop the entity which was modeled using a steel washer. Notice that all of the movement was modeled with real movement of a real physical entity. In our solid state emulator movement is modeled with a wall of LEDs turning on and off in sequence.

In the material handling system the movement of the AGV is represented by a single LED turning on and off in sequence along the path which the AGV follows. The single lit LED represents the actual vehicle. The material handling system in side of the machine center is modeled with a set of LEDs forming a circle. The LED at position one is the only LED that is lit. To model the disk turning the lit LED travels along the circle of LEDs indicating that as the disk turns position one moves. Figure 4 shows the diagram of a machine center in our model. The AGV's and the disk's movement is represented by the single LED turning on and off in sequence. The sensors are represented by a seven-segment display. It displays the AGV number when the AGV passes in front of the display. The lines are only drawn on the boards and to not represent any electronic devise. The black dots are the single LEDs that model movement and the seven-segment displays are used to give information to a human observer. All of the other physical devices in the model are modeled using LEDs in much the same fashion as the AGV and the disks. There are also some seven-segment displays used to indicate the entity number as it moves within the model. This is important because we must enforce the consistency of the entity's location. That is if entity 1 is going from location A to B and entity 2 is going from C to D then after the movement finishes location B will have entity 1 and location D will have entity 2. If the entity number is not maintained by the emulator then the controller may move entity 1 to location B then simply say B has entity 2. Since this is physically impossible the physics PLC must keep track of what entities are where. This information like all other information maintained by the physics PLC is displayed.

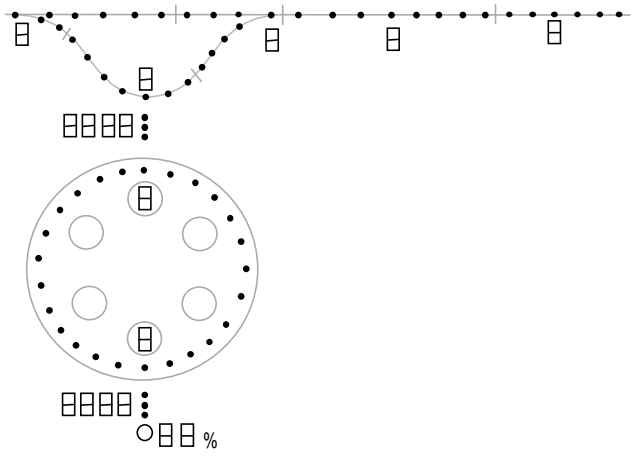

Figure 4: Diagram of a Machine Station in our New Model

It is important to note, however, that the controllers are still interacting with each electronically emulated device in a manner that is identical to the controller interacting with the actual device, i.e. with the application and sensing of voltage. The way we emulate the system has been changed; not the way we control it.

3.2 The Building Blocks

In order to increase the flexibility of the model, the model is decomposed into building blocks that can be put together to form a variety of realistic functional models. The

number of blocks and the types of blocks used in a model are variable. The blocks also referred to as boards are designed so that one can quickly build a model that closely resembles the actual system being investigated. Note that while the physical appearance may not be similar to the real system it is similar in the number and types of components. The basic modeling building blocks are shown in Figure 5. In this figure block (A) is used to represent the fixturing center and the queue for incoming and outgoing jobs. Block (B) represents a pair of machine centers, block (C) is used to allow the model to grow vertically as well as horizontally, block (D) allows the AGV to make a U-turn without having to go all the way to the end of the model, and blocks (D and E) are used to close the two ends of the model.

The model is built by putting boards together side by side. Since the AGV must have access to every machine station breaking the model into building blocks is not trivial. The building blocks must be designed in such a way such that regardless of how they are put together the MHS will have tracks reaching every station. The tracks forming the MHS must not become its own building block or this will decompose the model in too many small pieces. In order to provide tracks for the AGV that access all of the components regardless of how the block are put together we placed tracks along the top and bottom of each block. The model is then built horizontally. See Figure 6 for an example configuration. Note the configuration in Figure 6 is the same we used in our previous emulator shown in Figure 2. When the boards are placed side by side the track align and forms a long continuous segment of tracks. At the right and left end of the collection of boards a U-turn board (E) and (F) must be placed to connect the top and the bottom segments of track. This then produces a closed circuit of tracks segments. If the model consists of many boards one may want to insert an I-board (D) some where in the middle. This will allow the AGV to transfer between the top and the bottom track segment without having to travel to the left or the right end of the system. Board (C) has the capability to move an AGV to a similar board below or above it. This allows the complete model to expand vertically as well as horizontally. The software in the controller is designed so that the AGV goes in a clockwise direction. This allows for a much easier control scheme. The I-board (D) however is still bi-directional.

The controller can self adjust to any combination of arraignment of the blocks at time of power up. The MHS controller receives input from each board and among other information that is used during operation it also has a number identifying the board type. The relative location of the identification byte tells the controller where this board is located relative to the other boards. The software in the controller adjusts itself to the current configuration it detects at power up. Since the only part of the emulator that spans across several boards is the MHS, the locations of interest in the MHS must have a unique name regardless of the configuration adapted. In the naming convention we have adapted the first number in the location corresponds to the board. The boards are numbered from left to right from 1 to N. The next number represents the location within the board. The controller expects these names when referencing a location.

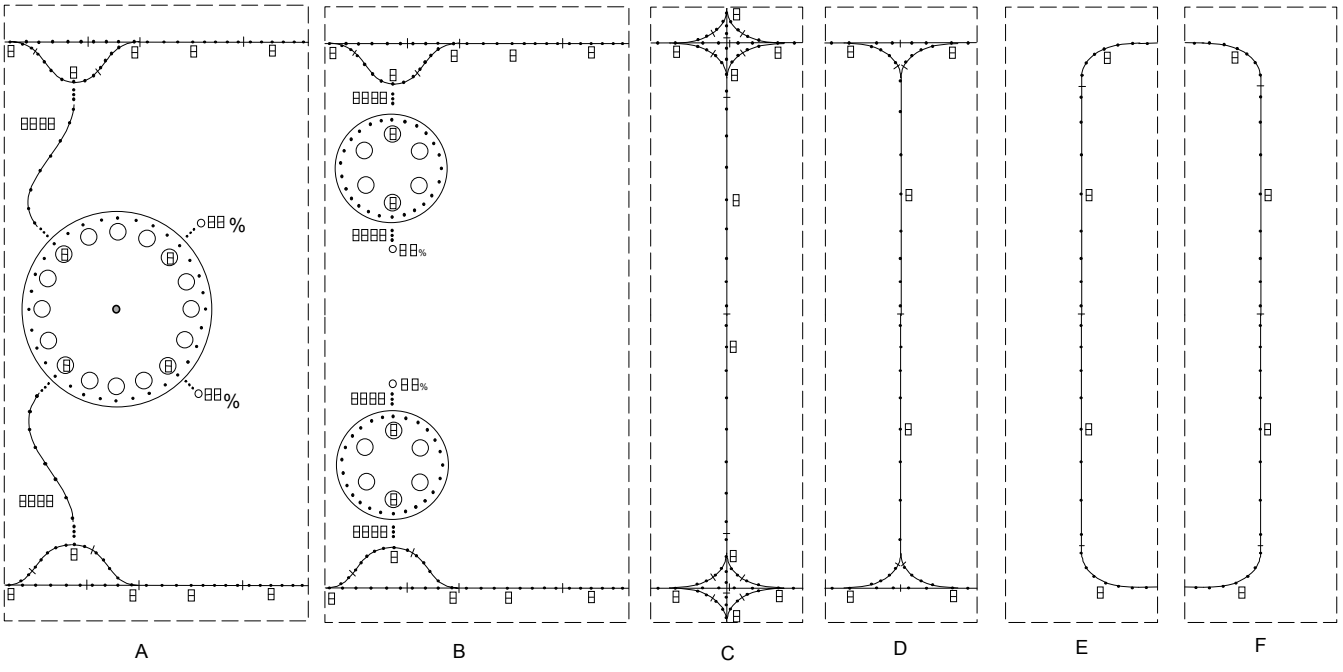

Figure 5: The Set of Basic Building Blocks

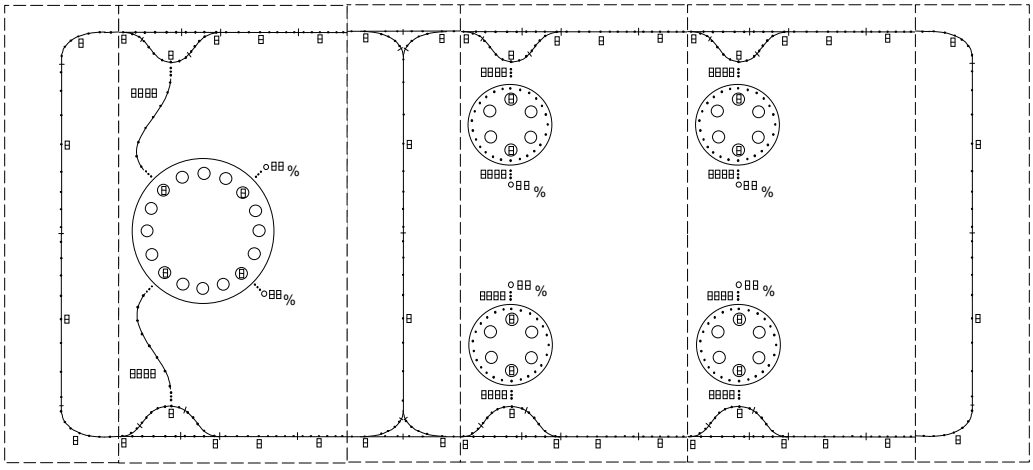

Figure 6: A Sample Configuration using One Fixturing Station and Four Machine Centers

In additional to the increased flexibility decomposing the model in to building blocks allows the model to be broken into pieces that are small enough to fit into an average suitcase. This allows easy transportation of the model. So for example if one is developing control software one may bring the model to a conference to demonstrate the software. The next section describes the boards in more detail.

3.2.1 The Machine Center Block

The machine-center board (B) models the machines that perform the required operations on the entities. Each machine consists of a disk with 6 holding position, two loader modeled as a small track consisting of 3 LEDs is used to move the entity between the AGV and the disk and between the disk and the actual machine. A large LED models the machine in progress. The block consists of two machines, one upside down on top and the other on the bottom. The seven-segment displays in the top machine are not upside down. This makes the top machine slightly different and is why they are paired. Note the machine center block like all other blocks includes a section of the main MHS. This was done so that the MHS does not have its own blocks.

There is only one PLC used to model the physics in this board. This PLC models the two machine stations as well as the portion of the main MHS tracks that are on the board. The controller for this board uses one PLC. This PLC is placed in this board as well although it is an entirely separate circuit only sharing power and the points of contact between the controller and the physics PLC where voltages are applies and sensed. The PLC that belongs to the controller however only controls the two. machine stations. The portion of the main MHS tracks that are on the board is controller by a dedicated PLC that only controllers the main MHS and is located in the fixturing center board.

3.2.2 The Fixturing Block

The fixturing-center board (A) models the storage of parts or jobs waiting to leave the floor or go to a machine or waiting to leave the system. It also models the fixturing operation that is necessary before a part can be sent to a machine for work. This board consists of a large disk with 16 holding positions, two fixturing stations, and four loaders. The board has two ports connecting the main MHS to the disk.

3.2.3 The Track Blocks

Blocks (E and F) are used to allow the MHS tracks to form a closed circuit, block (D) is used to form smaller loops within the track circuit and block (C) is used to allow vertical growth of the model. These boards do not have a PLC to model their physics. Instead their physical properties are modeled by a neighboring machine center (B) or fixturing center block (A). These track blocks do not have a controller PLC either. The controller for these blocks resides in the fixturing center block and controls all of the tracks within the entire model.

3.3 The Construction of the Building Blocks

Each block has a PLC dedicated to modeling the physical reaction the model has with the input voltage levels. We chose to use the PIC16F877 micro-controller since it contains an EEPROM and has many built in serial communication functions. To save on the number of I/O pins we need to handle the large amount of I/O, consider all of the LEDs that need to be handled, we use serial communication between the PLC and the I/O ports. Voltage sensing and application is modeled by reading or placing logic 0 or 1 on the pins of the ports. The input ports are implemented using an 8 bit parallel to serial buffer. All of the ports are tied together in series. Once per cycle the PLC

latches the data on all of the ports and inputs the string of serial data. This string of bits contains all of the input on the board. The outputs are handled in a similar fashion using a serial to parallel buffer. The data is sent to it serially and the buffer outputs the data in parallel. The LEDs are controlled using an 8-channel 7-segment display driver, MAX7219 by Maxim. This chip can handle 8 (7-segment) displays or 64 individual LEDs. The display drivers are also chained in series so only 3 pins in the PLC are used to control all of the LEDs on the board. While this method is slow in general, for our application it is sufficiently fast.

On the machine and fixturing boards, the physics PLC is responsible for modeling all of the machines and the portion of AGV tracks that are on the board. If a small board, one with only AGV tracks, is placed next to a larger board, one with a machine center or a fixturing center, then the larger board gains the responsibility of modeling the physics of the smaller board. This way the small boards that only model AGV movement does not need to have a PLC. Furthermore when an AGV moves from one board to another, the two boards communicate with each other to transfer control. Note, no communication between the PLC modeling the physics and the controller may exist however communication among the various PLCs modeling the physics may exist without any loss of functionality.

In order to minimize the number of PCs that must be connected to the emulator to communicate with it, the controller has a host PLC that communicates with one PC and sends the message to the corresponding PLC in the emulator. This allows only one PC to handle all of the communication with the emulator. The communication uses the RS-232 standard making it compatible with practically all types of PCs.

Each of the machine and the fixturing blocks measures 2 feet vertically and 1 feet horizontally. The smaller boards that are used for the main MHS only measure 6 inches horizontally. They are housed in wooden boxes each with a clear plastic face. The tint on the plastic will hide the electronics yet shows the LEDs when they are lit. See Figure 7.

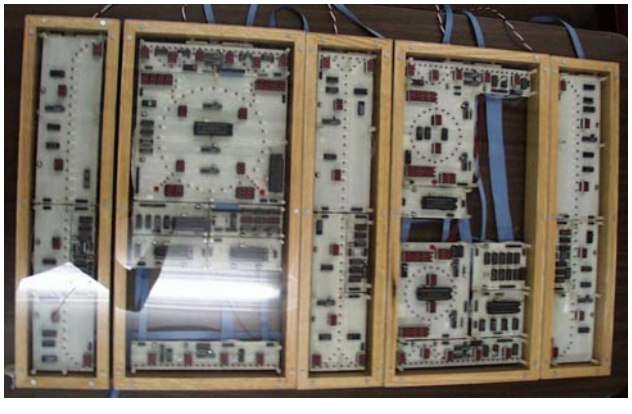

Figure 7: A Photograph of the Emulator Composed of Five Boards

In order for the solid-state model to interact with the controller in the same way a model with mechanically moving parts will, the model react to voltages applied and sensed at specific locations within the model. These locations are the electrical contact points where, in a real system, voltage will be applied to cause a device to react. For example in our first model, the electric train got its power from the tracks. To move the train, one applies a 12 volts source to the tracks. In our solid-state model, the application of a 5 volts source to a predefined contact point representing the tracks causes the physics PLC to model the movement of the AGV. The physics PLC models the activation of sensors by applying a voltage at a predefined location representing the sensor. The controller senses their locations to determine the status of the sensor. The emulator has modeled the presence of sensors at various locations so that the controller can determine the location of the emulated moving devices. The emulator is built so that the devices in the emulator periodically pass in front of a sensor. The physics PLC applies a voltage to the location designated as the sensor for the duration that the device is in front of the sensor. Once the device moves past the sensor the physics PLC drops this value to ground.

Since in most real-world systems, a facility is included to permit a human operator to interact directly with the device (this is essential when the real system gets into a deadlock state), we will provide a means for the human operator to interact directly with each process. This is accomplished through a liquid crystal display (LCD) and a small 16-key keypad. The user can physically type a message on the keypad and communicate directly with the process. We could also have a communication link through the control architecture so that the user can communicate with each emulated device remotely. However, by employing a keypad instead of a communication link, we are assured that this agent is totally isolated from any communication with any other devices in the model. The included keypad can be used to tell the device of any physical state change desired by a human operator. For example, a user may wish to add another AGV to the material handling system (MHS) or may wish to relocate an AGV to a different location. This is equivalent to a user of a conventional physical model walking up to the model, lifting a train and moving it to a different spot. Since one has to walk up to the model in a conventional model, having to walk up to the model to use the keypad poses no additional restrictions or inconvenience.

4 BENEFITS OF USING SOLID-STATE EMULATORS

Since our model is implemented by electronics and has no moving parts that can fall off, it is possible to hang the model on a wall in order to reduce the laboratory space needed to house it. Furthermore, the model can be

disassembled into smaller parts in order to permit it to be transported to different locations. Since the larger boards measure 1 by 2 feet while the smaller ones measure 6 inches by 2 feet, this allows the boards to be placed in a suitcase for traveling. Standardized electrical connectors are employed to connect one component of the model to another. Once its time to set up the emulator one simply lies the boards side by side in the desired configuration and connects the cables in the back of each block. When the power is turned on the software automatically adjusts itself to the current configuration and begins to execute. Instructions can then be sent to the emulator from the PC.

Not only does this approach make the model more reliable, it also makes it more realistic, i.e. more like the emulated entity's functionality. This approach allows us to economically emulate physical devices for which it would be difficult to build physical analogs. Since the model is now basically an electronic board with numerous displays and PLCs, we have the flexibility to implement fairly complex machines. For example, the design for our new model may include dedicated material-handling systems just for the handing of the tools. We were never able to implement this system on our previous physical model because of the high cost and space requirements. The new model may also include an Automated Storage and Retrieval System for storing fixtures and parts that are either waiting to enter the FMS or have finished processing. Thus, we can model a real manufacturing plant in a more comprehensive manner.

The model we have implemented has a cost under $2,000 in material. Since all of the physical devices are implemented using PLCs and display electronics, we avoid having to buy any expensive physical devices such as robotic arms. We can build several of these models for the cost of building only one conventional model. In fact, our goal is to construct several models at different locations and coordinate their operation over the World Wide Web. See Gonzalez and Davis (1997a), Gonzalez and Davis (1997b) and Tirpak et. al. (1992).

4 CONCLUSIONS

In this paper we showed how a physical model is implemented without the use of any moving parts. We showed that what distinguishes a physical model from a simulation is the fact that the controller does not tell the model to change its state but rather the state is changed by the application of control inputs to a physical device which then changes its state. The physical devices change their state using a dedicated PLCs that faithfully replicates each device's physics. Based upon the changes in the state, the same PLC will output signals to its controller PLC. This process faithfully replicates the process associated with sensors in the real system.

We showed that by using dedicated PLCs to emulate the devices, we eliminate most of the expensive, large, and unreliable hardware that is used to model the different physical components of a real automated system. In order for this PLC to properly model the physics of the model and, therefore, allow the sold-state model to function as a conventional model with moving parts, the physics PLC must be isolated from the rest of the model. The only exceptions are for its interfaces with its controller through the application and sensing of voltages at various contact points and an off-line communication port with the human operator. It is absolutely crucial that the controller has no way of altering the state of the model other than by applying appropriate inputs into the physics PLC. We then showed how by not having any moving parts our model can be built much cheaper, smaller, more reliable, and even portable.

We showed how to decompose the model into modeling blocks that can accommodate any configuration. The blocks are designed so that the main MHS has access to all of the points of interest regardless of the configuration. By using serial data transfer between the PLC and the ports were able to use only 7 pins for all the input and output including the control of all of the LEDs.

The developed model also has an educational function. Currently it is difficult to teach students to develop control architectures for systems as complex as an FMS. Many universities have developed laboratories using real equipment. However, these laboratories are expensive and can be dangerous for the novice to experiment with. In developing the electronic model, we have resolved all safety concerns. Furthermore, in our electronic emulation, we can include devices and subsystems that are seldom included in most academic laboratories. For example, most academic laboratories do not include tool handling or automated storage and retrieval systems. Once the student has mastered the control architecture for the model, he or she can then attempt to program the real system.

REFERENCES

Davis, W. J., B. Bauman, J. Macro, and D. Setterdahl, 1994. Constructing a model for research and education in the control of flexible automation. In *Proceedings of the ORSA Technical Section on Manufacturing Management Conference,* eds. J. Buzacott and C.A. Yano, 151–157.

Davis, W. J., D. Setterdahl, J. Macro, V. Izokaitis, and B. Bauman. 1993. Recent advances in the modeling, scheduling and control of flexible automation. In *Proceedings of the 1993 Winter Simulation Conference*, 143–155.

Gonzalez, F. G., and W. J. Davis. 1997a. A simulation-based controller for distributed discrete-event systems with application to flexible manufacturing. In *Pro-*

ceedings of the 1997 Winter Simulation Conference, 845–853.

Gonzalez, F. G., W. J. Davis. 1997b. A simulation-based controller for a flexible manufacturing cell. In *Proceedings of the 1997 International Conference on Systems, Man and Cybernetics.*

Gonzalez, F. G., W. J. Davis. 1999. An intelligent control architecture distributed across the network, for the control of large-scale discrete-event systems. In *Proceedings of the joint conference of the Third World Multiconference on System, Cybernetics and Informatics (SCI 99) and the Fifth International Conference on Information Systems Analysis and Synthesis (ISAS 99)*.

Kelton, W. D., D. A. Sadowski, and R. P. Sadowski. 1998. *Simulation with ARENA*, New York, NY: McGraw-Hill.

Tirpak, T. M., S. M. Daniel, J. D. LaLonde, and W. J. Davis. 1992. A fractal architecture for modeling and controlling flexible manufacturing systems. In *IEEE Transactions on Systems, Man and Cybernetics*, 22(5): 564–567.

AUTHOR BIOGRAPHIES

FERNANDO G. GONZALEZ is an Assistant Professor at the University of Central Florida. He received his B.S.C.S, and M.S.E.E. from Florida International University, and his Ph.D. from the University of Illinois at Urbana-Champaign. His interest includes real-time discrete-event control of distributed systems. His email and web addresses are <fgonzale@pegasus.cc.ucf.edu> and <http://pegasus.cc.ucf.edu/~fgonzale>.

ALICIA HELTON is an undergraduate student at the University of Central Florida, Electrical and Computer Engineering Department.

DOUGLAS HELTON is an undergraduate student at the University of Central Florida, Electrical and Computer Engineering Department.

JEFFREY SMITH is an undergraduate student at the University of Central Florida, Electrical and Computer Engineering Department.

EILEEN THOMPSON is an undergraduate student at the University of Central Florida, Electrical and Computer Engineering Department.

GERRY WALTERSCHEILD is an undergraduate student at the University of Central Florida, Electrical and Computer Engineering Department.