# CREATING AN ONLINE RFID LEARNING ENVIRONMENT USING AN AGENT-BASED ARCHITECTURE

Nabil Lehlou, Nebil Buyurgan, Justin R. Chimka
Industrial Engineering Department
University of Arkansas

## Abstract

Similar to many other emerging technologies, Radio Frequency IDentification (RFID) technology requires testing as well as proficient people to perform this task in order to achieve correct and effective deployment. This fact motivates efforts to build a remote automated system that makes testing easier and help increase the number of people who are educated about RFID. The authors of this paper take advantage of Web technology and agent-systems to (1) develop an online educational tool for RFID, (2) introduce an automated way to remotely control its laboratory hardware devices, and (3) construct an agent-based architectural model to support the underlying system as well as its flexibility and robustness. Moreover, a programming language is designed to enable experimenters to code hardware scenarios and hence, ease the task of configuring the testing setup in the laboratory. An implementation of the developed architecture is presented to demonstrate a real (vs. virtual) testing system and its use to conduct an experiment.

## Introduction

Even though the fields of science and engineering progressed in large part by conducting laboratory experiments, access to resources has been restricted to onsite students before the emergence of Internet. Now that this constraint is not an obstacle anymore, some institutions are promoting online educational tools from which students at different locations can greatly benefit by accessing remote laboratory equipment and obtaining hands-on experience[1,2,3,4,5,6,7]. Not only that, but Web technology is also able to provide new teaching techniques that are appealing to students[5].

On the other hand, RFID is one of the new emerging technologies that have a high potential of being used extensively in the near future. Some corporations such as Wal-Mart have shown interest in testing RFID equipment and developing applications for it, and now they sponsor research programs to promote such effort. One of the main objectives of such promotion is to assess the capability of RFID, which includes the quality of the RFID reads, the reliability of tags and readers in different environments, the impact of electromagnetic wave interference, etc. If an educational environment that is meant to convey knowledge about RFID technology contains a laboratory with RFID materials, then it is possible to provide testing results and conclusions, as well as give the involved students the opportunity to obtain hands-on experience, rendering them potential RFID experts and valuable assets to RFID stakeholders. Because the majority of employers who wish to adopt RFID believe that there are not enough RFID-skilled people to hire[8], attaining such experience by students is important.

Thus, it would be valuable to have an automated RFID laboratory whose apparatus can be remotely controlled, whose supporting system is flexible to reconfiguration and renovation, and whose graphical user interface (GUI) is interconnected with a knowledge bank about RFID technology and related topics. A learning environment of this sort has the potential of satisfying business needs, assisting collaborative educational programs, and endorsing RFID technology.

It is important however to mention that in general, the underlying architecture of technological laboratories may cause financial hindrance or structural limitations for instructional technologies in the long term. To be more specific, the learning process in laboratories can only last for so long before some of the used technologies become obsolete, a fact that makes the rapid pace of technological evolvement require the supporting system to be flexible to change, upgrade, and new-device-integration. Another issue is the non-standardized communication, in both software and hardware, between two or more technological ends. This worsens the situation of a system change that depends on replacing a component with a new one that is not compatible with other parts of the system. A large piece of the structure is then replaced, wasting by that a set of working assets and causing a considerable financial burden. It is therefore critical to take flexibility into account when engineering robust educational tools that depend heavily on technological means so that dealing with dynamic change is not only feasible, but efficient as well.

The contribution of this paper is the development of (1) a Web-based learning tool that targets teaching RFID with an emphasis on the practical aspect of the technology, (2) automated control of the laboratory hardware devices, and (3) an agent-based architectural model that supports the technological components of the RFID testing mechanism and endorses its system flexibility. An application example is given to illustrate the use of such educational tool, the ease behind its automated control, and some of the agent operations performed behind the scenes.

## Previous Work

The physical laboratory experiments have enriched the science and engineering areas; however, it is not easy for students to fully understand many modern systems due to the accessibility constraints associated with laboratory resources[1,3]. Recent research has revealed that through interactive examples and experiments, students are able to learn and retain information better[4,9,10,11]. As a result, many researchers from different fields are taking advantage of the evolution of technology and the Internet to create Web-based laboratories in order to better the learning of students by granting them the capability of studying anywhere and anytime[3,6,7]. Furthermore, online laboratories have the advantage of aiding researchers at stimulating the interest of learners with new teaching techniques provided by Web technology[5].

Kuester and Hutchinson[3] develop virtual laboratories in order to complement the limited classroom material with the laboratory adequate resources in earthquake engineering education. Jiang, Kurama, and Fanella[4] present an online laboratory for instructing the behaviour and design of reinforced concrete structures. Ozer et al.[5] construct Web-based modules to enhance the learning experience of students in thermal fluids. Anderson, Taraban, and Sharma[10] explore active-learning methods to influence the design of their Web-based modules on thermodynamics. Hsieh and Hsieh [1,2] create a Web-based system for Programmable Logic Controller (PLC) education by using “intelligent tutoring” to convey the knowledge, and they report very high success through the use of their tutoring system and virtual PLC, with the animation being the most popular instructional activity. Hamada[9] develops a Web-based educational tool that incorporates a set of visual modules to promote interactive and collaborative learning and meet learners’ preferences in the field of computation, including language processing, compiler design, and automata theory. The outcome of his experiments with students shows an increase in learner’s performance and motivation to independently acquire more knowledge. Finally, Gurbuz[6] discusses how the Internet aids the development of the manufacturing engineering curriculum in Turkey. The objective of developing Web-based material for such matter is to augment the students’ knowledge and sharpen their technical skills by giving them the

opportunity to access subjects that would otherwise be unavailable without the use of the Web.

From the perspective of RFID, the research efforts in this field were concentrated around the applications and uses of the technology rather than the education portion of it. Additionally, while only a select number of schools and universities teach RFID, the majority of these institutions instruct it through research or projects that are selected by students. Among the organizations that offer classes in RFID are Indiana University, University of California, Michigan State University, and University of Houston[12,13].

In 2004, Indiana University established its first working RFID instructive model. Students and professors are now able to experiment with RFID technology, build interfaces for RFID-related systems, generate metrics, and teach applications of RFID in conjunction with EPC systems[12].

In fall 2004 at the University of California, a mechanical engineering professor established an RFID-focused course called the Management of Technology, in which business students collaborate with students with technological background. In this class, students worked on designing and implementing novel business applications using RFID technology[13].

At Michigan State University, East Lansing, the School of Packaging offers a course on the utilization of RFID technology in packaging. Moreover, it has an RFID testing laboratory that both undergraduate and graduate students exploit to conduct independent testing for research projects. In 2005, "At least five Michigan State University students have completed their master's degrees in RFID research, in topics such as RFID in warehousing and supply chain applications packaging, and RFID systems design"[13].

In 2005, University of Houston offered a class in RFID Programming as an elective in the Management Information Systems (MIS) department, and it was under the form of a comprehensive survey of the RFID technology and its business applications. Most importantly, students carried out collaborative laboratory assignments, in which they experimented with different tag fixed-positions and mobile readers, and implemented a back-end software infrastructure using a developer's kit. Furthermore, they were able to encode data into RFID tags, deploy readers in a laboratory setting using TAVIS® middleware and Visual Basic©, and learn ways of managing the collected RFID data[13].

Albeit all these efforts, still not enough RFID skilled personnel are being supplied to the industry[8,12], and that is simply because the number of institutions providing the training and the proportion of students receiving it are significantly small. A way to promote RFID is by making it accessible to everyone as well as supporting collaboration between academic institutions, and that can be achieved by implementing the proposed system architecture of the online RFID learning environment.

## Technology Background

### *RFID Technology*

Radio Frequency Identification (RFID) is a data collection technology that utilizes wireless radio communication (radio frequency signals) to identify, track, and categorize objects. The basic RFID system consists of three main components (see Figure 1):

- The RFID tag, which is a microchip that is embedded into a miniature antenna and that transmits the data stored in it as the electromagnetic response to the RFID reader.

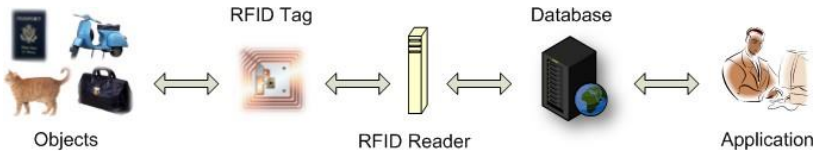


Figure 1. Object/device interactions
in an RFID system.

- The RFID reader, which is a subsystem that contains the processing unit and its connected antennas. Its main job is to emit electromagnetic waves to the surrounding setting and listen for electromagnetic responses from the RFID tags. It then forwards the captured RFID reads to the target database.
- The database where all the raw read data is to be collected and stored. Such data is usually translated to meaningful information that helps capture the business value added by RFID.

This system can be extended with a set of middleware devices, a variety of soft-controllers, a network of readers, and a powerful database management system (DBMS) to ease data-acquisition and data-management in a large information system.

The advantages that an RFID tag has over a barcode include: (1) the ability of being read from a further distance, (2) the ease of storing/representing more encoded data, (3) the identification of several items at once.

### *Everything is Alive*

"Everything is Alive"[14,15] (EiA) is a concept that says that all objects in the world can be connected to a network called the "Internet of Things"[16] and be intelligent enough to communicate with humans as well as other objects without human intervention. Once these abilities are acquired by an object, it is called an "EiA agent", and it is said to be "alive" because it can interact with humans and agents. Ideally, one can embed a microchip into an object, program it to create some kind of intelligence, then, plug this "thing" to the ubiquitous "Internet." This entity therefore becomes an agent that is ready to communicate with its outside world. However, this idea needs a while before being implemented because of the cost of the integrated microchips and their installation. Fortunately, there are cheaper ways to achieve the desired EiA environment. One solution is to use computers, for they are powerful, available, and can connect to the Internet, which represents the agent's network.

Now, consider XML, which is a structured programming language used in the Web technology, to be the standard language that every agent speaks. A computer program, usually called a "wrapper," is written to act as an interpreter from XML to the native language of a device (which can vary from encoded data to digital signals, to actual motion) and vice-versa. In other words, a hardware device (or software application) is wrapped with a software translator, and therefore, mutated into an agent that can speak XML to other agents.

One popular area in engineering that can benefit from EiA is system design. Systems that involve software can greatly benefit from EiA because by its nature, EiA leads to constructing agents that are independent of the outside world. To clarify this, think of a system as a set of independent, but interrelated, agents. If the system breaks, only the responsible agents are fixed or replaced. If the system needs to perform better, only the inefficient agents are upgraded. If the system requires an expansion, one or more agents are added to the system without affecting the existing ones. One can see that EiA provides engineers with a way to design a system that is simple, robust, easy to maintain, and flexible to change.

## System Architecture

The goal now is to construct, using an EiA approach, a tool that provides RFID learners with hands-on experience and gives them the opportunity to use their analytical skills. This section therefore discusses the software and hardware architecture of an easy-to-maintain laboratory system, and its soft-controllers that support the control of RFID equipment and hardware moving devices, in order to provide a high-access RFID testing environment to offsite students. Also, a feature that is added to this system is a programming language that can be used to code hardware scenarios so as to programmatically control the laboratory moving mechanisms.

### *Envisioned Setup*

Several testing requirements have to be considered when designing the hardware setup for the RFID laboratory system. The factors involved during experimentations are the motion of RFID tags, their distance from the RFID antenna, the tag density in the RFID envelope, and the angle of the RFID antenna. Allowing these factors to be treated as variables is of extreme value and the creation of degrees of freedom in the hardware system for such use is therefore necessary.

The hardware mechanism chosen to be implemented is in the form of a robotic system that has a set of motors and a control unit that cause the RFID tags to move linearly on parallel train tracks and the RFID antennas to change angles (see Figure 2.). For more control power, the tagged trains and the antenna motors can be programmed to move according to a certain scenario that is coded in a designed programming language.

### *Laborator y System Components*

#### RFID Reader Agent

Besides the previously mentioned benefits that a system enjoys when it is built using an EiA framework, there are two other advantages that are noticed when working with agents such as RFID reader agents. First, it is possible to control different types of readers, that are built by different manufacturers, using the same types of XML messages; and that is possible thanks to the agent wrappers that know which native language

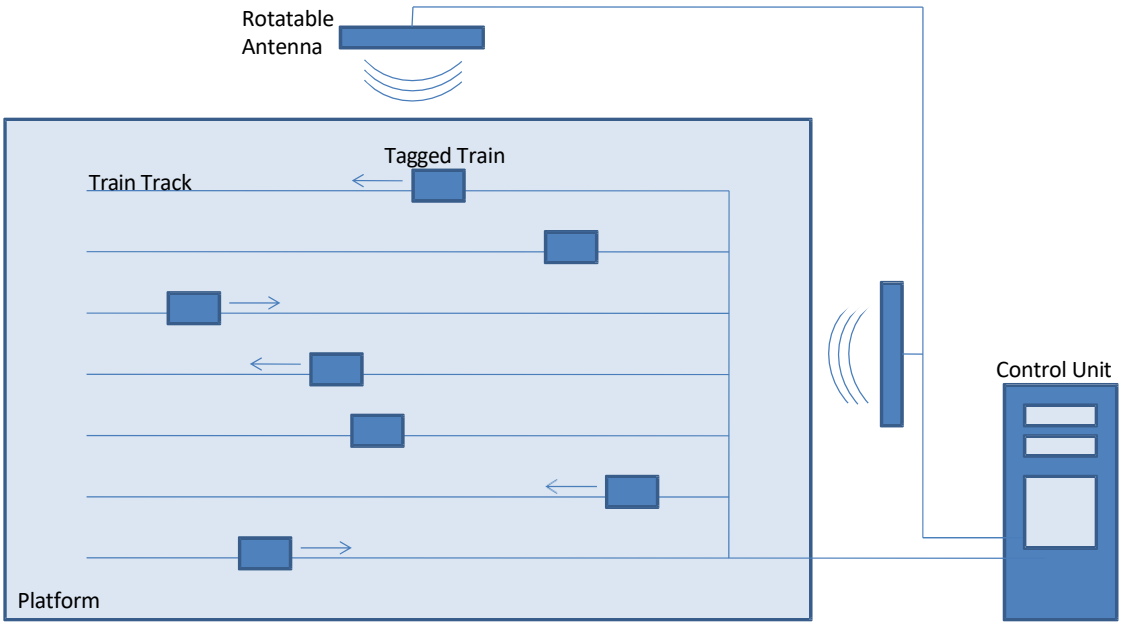


Figure 2. RFID laboratory testing system with tagged moving trains and rotatable antennas.

to translate to. Second, even though the RFID reader may carry out basic tasks only, it has the potential of becoming a much superior device if its agent is extended with more features. For example, the Alien® RFID reader has only one reading mode: reporting the RFID reads seen during an instant when a signal is sent to it. The RFID reader agent on the other hand can do the following:

- Continually report the RFID reads as long as the agent is enabled; that is achieved by having the wrapper send a signal to the reader at the end of a time frame called the "Poll Period," which is given by the user.
- Report the arrival of a tag to the RFID field, its departure from it, or both events.
- Be reconfigurable through one XML message instead of a set of line-commands that requires the user to be logged into the device.

## Database Agent

The DB agent receives from other agents data encoded in XML and stores it in its database, regardless of which database type is used (MS-Access, Oracle, mySQL, etc). The DB agent also receives queries and replies to their senders with the appropriate data. One advantage that this agent has over the regular database systems is the ability to stay alive and keep an open connection with the actual database itself, preventing connection overhead each time the database is to be queried or modified.

## Robotic System

The robotic system discussed in this section is a set of motors (which cause moving devices to translate or rotate) as well as the associated hardware equipment and software applications that help with the feasibility and efficiency of the moving mechanism. This system has a core structure that allows it to work just fine; however, it is not flexible to change since a simple modification may require editing the code and the circuit wiring. To solve this problem, the components of this mechanism are converted to EiA agents.

### *Core System Structure*

The core system structure is based on an electrical circuit that can yield three different states for a motor: idle, rotate clockwise, and rotate counter-clockwise. With the use of relays, such circuit can benefit from the fact that motors can be controlled by sending signals of 0 or 1 to the appropriate relays. For example, consider the circuit shown in Figure 3.: if relay R1 is closed (sent a signal of 1) and relay R2 is opened (sent a signal of 0), the motor rotates in the clockwise direction. Table 1 contains the mapping between all signal combinations and the motor states.

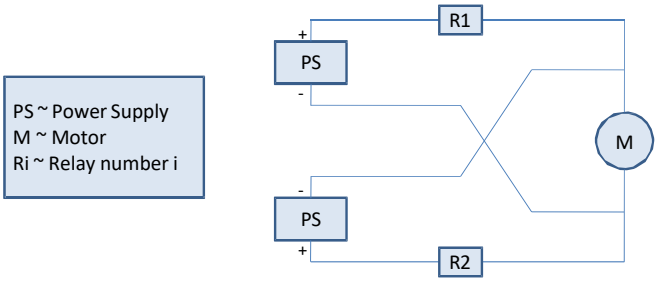


Figure 3. Circuit design that allows three different motor states.

Table 1: Relay signals and their associated motor states.

| R1 signal | R2 signal | State of motor |
|---|---|---|
| 0 | 0 | Idle |
| 0 | 1 | Counter-clockwise rotation |
| 1 | 0 | Clockwise rotation |
| 1 | 1 | Idle |

One way to send relay signals is by: (1) connecting a relay circuit board to a computer, (2) installing the software library of the board in the machine, and (3) constructing a GUI whose soft-buttons hide the digital details from the user and submit the appropriate relay signals according to the desired motion to be seen.

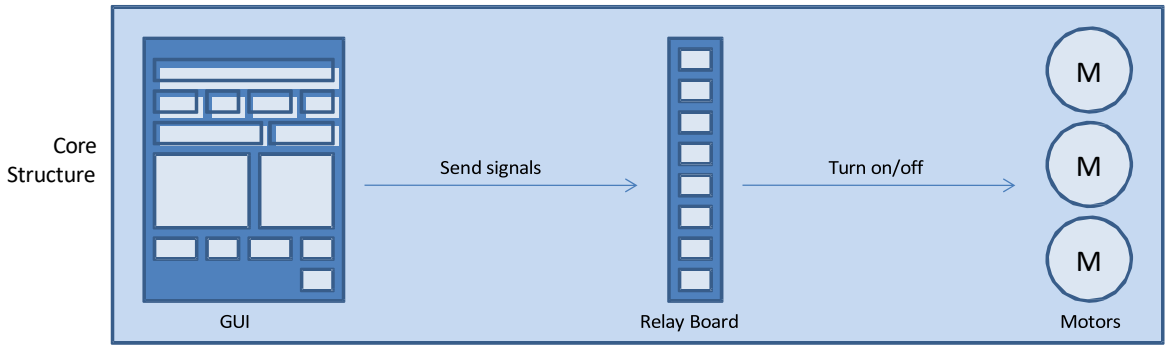


Figure 4. The components involved in the process of controlling a robotic system.

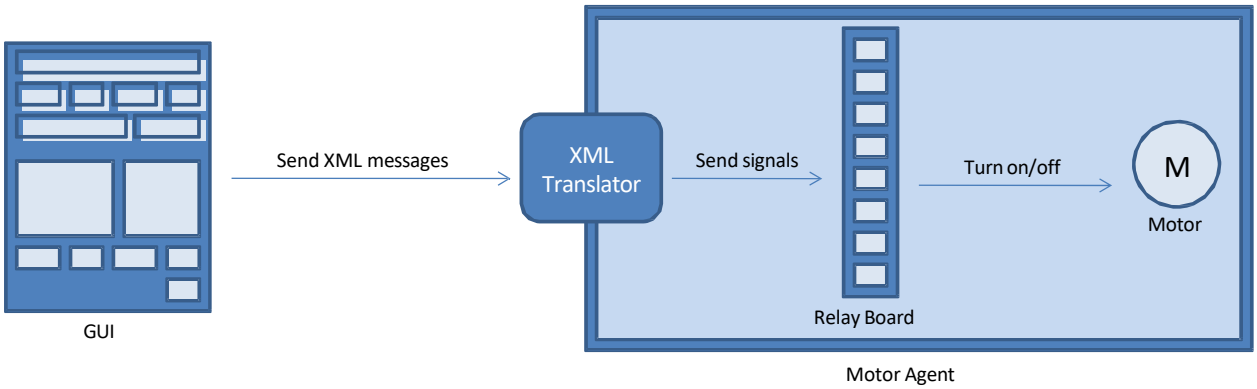


Figure 5. The components involved in the process of controlling a motor agent.

Figure 4 depicts these elements as well as the process of controlling a set of motors that constitutes a robotic system.

Note that for this structure to work, the GUI has to be a local application that runs on the machine that is directly connected to the relay board. It would be far more preferable to have a more flexible system where the motors can be controlled over a network. For that reason, motors are turned to motor agents, and the robotic system is transformed to a robot agent.

### *Motor Agent*

The strategy behind building a motor agent is grouping all interrelated components that serve the purpose of rotating a motor to make one independent unit; that includes the motors themselves, the power supplies, the relay board, and the associated middleware. This unit is then wrapped with an XML translator that converts XML messages into appropriates signals for certain time periods. Figure 5 shows the process of controlling a motor in the new improved system structure.

Due to this change in the system structure, the GUI has to be adjusted to send XML messages rather than digital signals. After this task is carried out, the motor can be controlled remotely over a local network. This can be accomplished by having GUI buttons generate XML messages for the motions a user wants to see. Nonetheless, for complicated motion scenarios, the task of pressing multiple soft-buttons several times becomes tedious and may lead to inaccurate results. As a response to this inconvenience, a programming language called “NBL” is developed to allow the controller to code an entire scenario, and then submit it to the appropriate agent with one button click. The grammar of the NBL language can be

found in Appendix B. Notice that NBL also allows other computational features such as loops, conditions, function calls, lists… just like the majority of programming languages. As an example, see the following NBL code:

```
var x = true;
if(x == false)
  motor "engine1" clockwise 2 seconds;
else
  motor "engine1" counterclockwise 2
  seconds;
```

This is a simple program that declares a variable x; then depending on the x value, the motor agent, engine1, will either move clockwise or counter-clockwise for two seconds before it returns to its original state (the state in which the motor was before the scenario was received by the associated agent).

One important point is that any programming language is just an abstract syntax, and for a scenario/program to be translated into action, the presence of an interpreter or compiler is compulsory. In an EiA environment, an NBL interpreter would transform a scenario to an XML message before the latter is submitted to the appropriate agent. A question then arises: in the case of a scenario where several motors are to be rotated, which agent should receive the XML message? One solution to this issue is to have a supervisory agent that encapsulates a number of agents as one system unit; this container agent can hence forward the different sub-scenarios to the associated sub-agents at the proper times.

### *Robot Agent*

In EiA, a set of agents that collectively represent a certain system or serve a specific purpose can be grouped under one container agent. As there can be many types of container agents, a special one for the robotic system case is the robot agent, which is primarily responsible for making the intended motor agents move according to a received scenario. For instance, consider the following NBL scenario:

```
var i = 1;
var sign = 1;
display("Starting scenario...\n");
while(i <= 2)
{
  motor "train2" forward 2*sign seconds;
  motor "antenna3" clockwise 3*sign
  seconds;
  sign = sign * -1;
  i = i + 1;
}
display("Scenario done!\n");
```

It is expected from this program to yield the following sequential motions:

1. train2 moves forward for 2 seconds
2. antenna3 rotates in the clockwise direction for 3 seconds
3. train2 moves backward for 2 seconds
4. antenna3 rotates in the counter-clockwise direction for 3 seconds

In this particular instance, the robot agent knows that it should tell train2 to move forward for two seconds, wait for two seconds, then tell antenna3 to rotate in the clockwise direction for 3 seconds, so on and so forth. The sequential set of commands can be carried out in a perfect fashion, and that is because all motor agents in the system are accessing the (common) relay board card one at a time. On the other hand, commands that are to be executed in parallel face the challenge of motor agents not being able to access the relay card simultaneously, a fact that leads to the creation of a waiting queue of motors that needs to utilize the shared resource. Hence, there is always a time delay between the different motors that are to be run in parallel, which negatively affects the quality of the supposedly synchronous motions.

There is a way to not only solve this problem, but take advantage of this shortcoming as well. One thing to know first is that motor agents are randomly queued while waiting for the availability of the relay board. This can be used when randomness is needed in the laboratory experiment. In NBL, if the earlier scenario is to be executed in a randomly asynchronous parallel fashion, the following code can be used:

```
var i = 1;
var sign = 1;
display("Starting scenario...\n");
while(i < 2)
{
  parallel async
  {
    motor "train2" forward 2*sign
    seconds;
    motor "antenna3" clockwise 3*sign
    seconds;
  }
  sign = sign * -1;
  i = i + 1;
}
display("Scenario done!\n");
```

Conversely, to achieve synchronous parallel movements, the robot agent has to be upgraded by adding the relay board agent. Such an agent enables the access of all the relays directly, and thusly, controlling all motors linked to them at the same time, leading to perfectly parallel motor motions. The previous scenario example can be executed in this manner simply by removing the keyword `async` from the NBL code.

The robot agent can be sophisticated in many other different dimensions where the only limitation is the imagination of the builder. One dimension would be adding sensor agents that let the users know the statuses of the motors or their associated moving devices. By receiving feedback about the location/position of a hardware unit, the NBL coder can even surpass the need for visual system troubleshooting.

## Graphical User Interface

### *Component Agents*

The graphical user interface is one of the most flexible project pieces to construct, and its look depends purely on the user's requirements and the programmer's imagination. Nevertheless, regardless of the way it is designed, the GUI needs two agents embedded in it if its functionality is to be complete in an RFID laboratory system that is built on top of an EiA framework. The first one is the control agent whose responsibility is to send XML messages that speak for the user's intentions, such as moving a tagged train on a track, reconfigure an RFID reader, terminate an agent, etc. Moreover, the control agent should also be able to receive XML messages that carry specific awaited information like statuses of controllable devices, confirmation of agents' actions, and replies to other sent messages. Furthermore, it has to be linked to the NBL interpreter so it can send the hardware scenarios (encoded in XML by the NBL interpreter) to the right system receiver. The second agent that complements the operation of the GUI is the listener agent whose main purpose is to capture unexpected messages such as RFID reads and agents disconnection confirmations. In other words, because these types of messages are not awaited by the control agent, it is necessary to have a listener mechanism take care of receiving them and displaying their content to the GUI user at all times.

The functionality of the GUI application is not bound to these two agents though. Similar to the robot agent, the GUI can be expanded and sophisticated through different kinds of agents and there is no limit on the operability that can be added.

### *Migration to the Web*

The EiA framework provides agents with a mean of communicating over a network, which is, in this specific system, the internet network used to support the Web technology. Even though it is conceptually feasible for two computers to send each other XML messages, it is not realistically always possible because of the network router architecture, the security barriers, the configuration of the local networks, etc. A way to overcome this constraint can be achieved by following these steps:

1. Build a local secure network for the RFID laboratory, adjust its configuration to satisfy the needs of agents, and make sure that messaging occurs as expected.
2. Connect a Web server to this network and run a website on it.

3. Have the website host the GUI that controls and listens to the agents that exist in the laboratory system.

This way, people that have access to the ubiquitous Web technology are able to reach the RFID laboratory agents and interact with them.

One final important complementary addition to the system is the presence of a good quality Web-camera that prompts the user to view and zoom on specific parts of the laboratory. For example, Figure 6. shows a Web browser view of an RFID laboratory obtained with a Sony® Web-camera. Such a device helps users in many ways such as obtain a visual of the laboratory and the RFID equipment, control the hardware with ease, and take snapshots of the laboratory view for future documentation.


Figure 6 . A Web-browser real-time view of an RFID laboratory.

***Laboratory System Architecture***

At this point, all the critical components of the RFID laboratory system have been discussed and its architecture can now be described at a high level: After the reader agent has its reading mode configured, it starts reading RFID tags accordingly. It then broadcasts the captured reads throughout the local network so that all interested parties can obtain such data. Two of these parties are the GUI listener agent, which displays the read data to the user, and the database agent, which stores it in the appropriate tables. Through the GUI and its control agent, the user can also discover the connected agents in the network, modify the configuration of the reader agent, control the robot agent, and query the database. Because the robot agent allows moving RFID tags and rotating RFID antennas, changing testing setups for experimentation on RFID technology can be acquired through multiple degrees of freedom. Figure 7 summarizes these interactions in an architectural model. Note that while a system can have many agents of the same type, the interaction with the other sorts of agents is the same as described above. That is in this EiA system for instance, one or more GUIs can control several robot agents, a reader is able to send read data to all DB agents, and a DB agent is capable of receiving from all the reader agents.

This agent-based architectural model is also not saturated and can be extended and modified in several ways. The fine thing about it is that it is empowered by three different features that give it an advantage over other architectural system designs. The first one is the ability of agents to be smart and their capability of handling system issues without the intervention of humans. For illustration, when the robot agent receives a scenario, it knows how to distribute the tasks over its motor agents in a synchronously parallel, asynchronous parallel, or serial manners. Hence, an aspect of this kind omits some of the dependency on humans and promotes more automation.

The second feature touches upon the structure of the system and its flexibility to be modified or extended. First, introducing a new device to the system is as simple as wrapping it inside an agent and connecting it to the agent network, a task that might need the creation or modification of the agent wrapper, but may be performed without affecting other components in the environment. Second, if a device is to be replaced with a more superior one to achieve the desired system upgrade, then the replacement is as trivial as

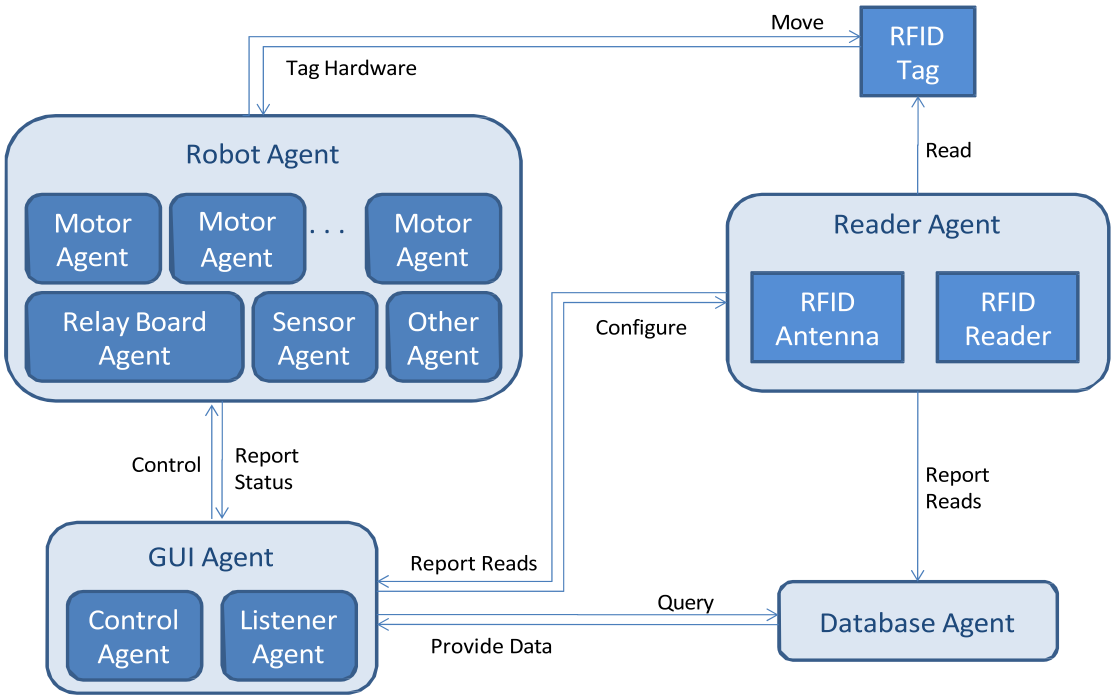


Figure 7 High-level RFID laboratory system architecture.

removing the old agent and introducing the replacement as a new agent. Finally, an agent can be made even smarter, and that is just by updating its wrapper while keeping its proper operating device as it is. Thus, one can see how inexpensive changes can be when the system is constructed on top of an EiA framework.

The final feature on which the architectural model is based is the fact that the system is composed of agents that speak and understand the language of XML; i.e. each one of these entities is able to send and receive messages encoded in XML through the local network. This yields standardization in communication between the laboratory technological devices in both hardware (through Internet network) and software (through XML messaging). In other words, non-Internet cables that connect electronic devices such as parallel-port and serial-port cables do not constitute a restriction in building flexible systems that are independent of the type of cabling anymore. Also, the software aspect of the system gains a similar advantage from the XML messaging that occurs between the agents, and that is due to the fact that the transferred data is recognized by all receiving parties, leading to error-free agent communication. Furthermore, it opens the door for agents to be developed in different programming languages without losing the ability to interact with each other. Application developers can therefore program in their favourite languages and use their most convenient software framework without problems, a fact that creates another dimension of flexibility.

Overall, the developed RFID laboratory system benefits from the EiA architecture in three notable ways. One is the ability of a device to interact with any other device, without human intervention. Secondly, any component in the system can easily be replaced by a substitute, even if the two are very different in the way they operate, resulting in inexpensive reconfiguration, upgrade, and agent integration. Finally, the standard fashion agents use to interact in an EiA network offer a set of benefits to hardware assemblers and application developers to construct systems with error-free communication.

## Application

This architectural model is applied to an RFID testing system at the Industrial Engineering RFID laboratory of the University of Arkansas, Fayetteville. The hardware setup is composed of one Alien® RFID reader with two antennas as well as sixteen moving RFID tagged Lego® trains on sixteen parallel rail tracks, which are set on a 16x16 ft (4.88x4.88 m) carpeted wooden platform (see Figure 6). The top part of each rail is made out of metal in which an electrical current can pass and cause the trains to move. Each Lego® system is powered up by two Lego® speed-adjusters, which are connected to the QUANCOM® USBOPTOREL32 relay-board, just as explained in Figure 3. This board is connectable to a computer through a USB port, is powered up through the same USB port, and has an Application Programming Interface (API) provided by the QLIB software library. The USBOPTOREL32 device also has thirty-two relays, where each one is responsible for the forward or backward motions of one of the sixteen trains, and thirty-two input-bits that can report the (on/off) status of every relay.

Given this setting, a sample experiment is presented in order to show the usefulness of this educational instrument and its programmable control as well as the response of some agents to users' requests. Such experiment first starts with aligning the trains to the far left, away from one of the antennas (see the left 15x15 grid of Figure 8, where the side of each cell is 1 foot (0.3m) long). This can be accomplished in several manners; one quick and easy way is by executing the following NBL statement:

```
Motor "" backward 6 seconds;
```

Note that in the implemented system, when the name of the agent is not specified in a request, the associated XML message is broadcast to all the connected agents in the local network. Also, the reason for which six seconds is used is because it takes that long for a train to be displaced from the extreme right to the extreme left of the platform on the 15 ft (4.57m) long rail, a fact that implies a velocity of 2.5 grid-cell per second (0.76 m/s). Second, the setup depicted on the right of Figure 8 is to be achieved. By using NBL, the following scenario (where the variable t represents the time to traverse a cell) performs the required task:

```
var t = 1/2.5;
display("Start scenario...\n");
motor "engine6"  forward 12*t;
motor "engine7"  forward  8*t;
motor "engine8"  forward 10*t;
motor "engine9"  forward  8*t;
motor "engine10" forward  6*t;
motor "engine11" forward  6*t;
display("End scenario.\n");
```

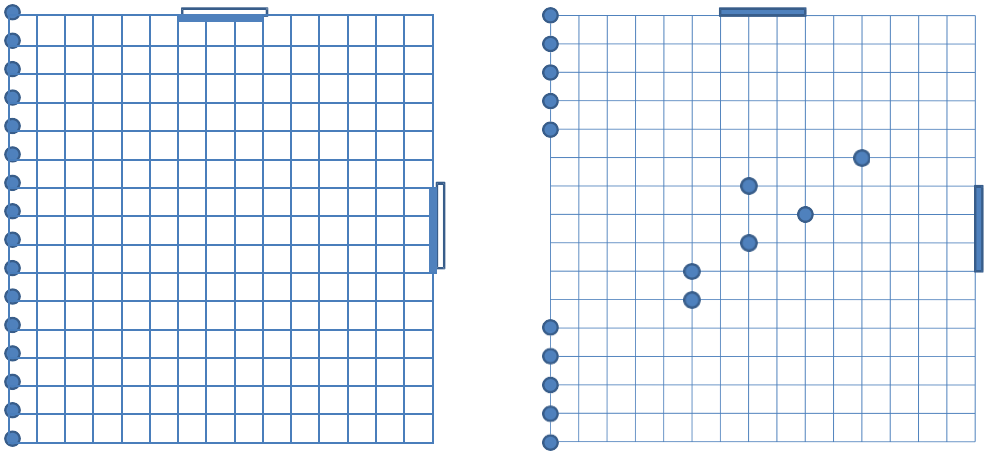

Figure 8. The initial setup (on the left) and the final setup (on the right).

Upon execution, this scenario is sent to the robot agent whose IP address and port number are 192.168.0.2 and 4142 respectively. Before it is sent however, the scenario is compiled and converted to the following XML message:

```
<?xml version="1.0" encoding="UTF-8"?>
<Message SrcAddr="192.168.0.3"
      SrcPort="3566"
      DstAddr="192.168.0.2"
      DstPort="4142"
      SeqNo="1" InRe="0">
  <Scenario Name="" Agent="RobotAgent"
      Parallel="False"
      Synchronous="False">
    <Command Motor="engine6"
      Direction="FORWARD"
      Size="4.8" Unit="SECOND"/>
    <Command Motor="engine7"
      Direction="FORWARD"
      Size="3.2" Unit="SECOND"/>
    <Command Motor="engine8"
      Direction="FORWARD"
      Size="4" Unit="SECOND"/>
    <Command Motor="engine9"
      Direction="FORWARD"
      Size="3.2" Unit="SECOND"/>
    <Command Motor="engine10"
      Direction="FORWARD"
      Size="2.4" Unit="SECOND"/>
    <Command Motor="engine11"
      Direction="FORWARD"
      Size="2.4" Unit="SECOND"/>
  </Scenario>
<Message/>
```

After the desired setup is attained, the experimenter can start reading the RFID tags for a certain period of time. Nevertheless, before s/he does so, the RFID reader agent has to be configured according to the needs of the experiment. Configuring an RFID reader agent involves choosing specific options for three different features. The first one is the Poll Period, which refers to the time frame the agent wrapper waits before it asks the reader device for the next set of RFID reads. The second feature is whether or not the read data is to be filtered according to arrival and departure events before it is reported. That is RFID reads are only reported when a tag arrives to or departs from the RFID field of a reader. If the user opts to enable such filtering, then the third feature is to be set to log either the departure event, arrival event, or both.

Consider now that a poll period of 200 milliseconds is chosen, and no filtering is required since the trains are static and the experiment does not involve motion. When this configuration is selected and executed at the GUI level, the following XML message is generated and sent to the targeted reader agent:

```
<?xml version="1.0" encoding="UTF-8"?>
<Message SrcAddr="192.168.0.3"
      SrcPort="3566"
      DstAddr="192.168.0.2"
      DstPort="3185"
      SeqNo="3" InRe="0">
  <ReaderOn LogArrivals="True"
      LogDepartures="True"
      PollPeriod="200"
      FilteredReads="False"
      TagTypeMask="4"/>
<Message/>
```

Upon receipt of the message, the RFID reader agent starts broadcasting RFID read data to the interested agents (such as GUI listener agents and DB agents), and does not stop doing so until it receives the ReaderOff message (see below), which can be generated and sent manually or programmatically.

```
<?xml version="1.0" encoding="UTF-8"?>
<Message SrcAddr="192.168.0.3"
      SrcPort="3566"
      DstAddr="192.168.0.2"
      DstPort="3185"
      SeqNo="5" InRe="0">
  <ReaderOff/>
<Message/>
```

Note: a sample of XML messages that are used in this system is provided in Appendix A.

After the data collection process is complete, the user can export the read data to an Excel sheet or any software package that can allow him/her to perform data analysis.

## Results of Implementation

As mentioned earlier, the developed EiA agent-based architecture is used to support an implemented remote RFID testing system at the University of Arkansas. From a Web-browser, users can access the RFID laboratory, open a laboratory view streamed by an onsite Web-camera, zoom on the target devices, move them according to the desired test setup, acquire the reads, and finally export the data to a file for future analysis. Furthermore, since a website is used to host the hardware control GUI, the site can include other pages that broaden the navigator's knowledge about different aspects of the used technologies. For example, specialized articles about RFID and its components can be posted, tutorials of the NBL programming language can be incorporated, and detailed technical discussions can be shared. Note that the hosting website is currently being expanded in a continual fashion.

From the system performance and usability perspectives, feedback from developers shows that the mechanism not only works well, but it is also easy to maintain and upgrade. Moreover, survey data and its analysis demonstrate that students who used the associated learning environment have increased their understanding of RFID technology and its relevant areas, improved their attitudes about engineering education, and enhanced their confidence towards any instructed technology. The results of the associated student assessment can be found in [19].

## Conclusions

This document describes the effort of developing a real (vs. virtual) Web-based educational tool that exploits the power of the *Everything is Alive* paradigm to achieve remote accessibility of RFID laboratory resources in a reliable and easy-to-maintain manner. A significant emphasis is placed on its agent-based architecture that represents the underlying framework of the laboratory system due to the importance of the flexibility of its structure in the long term. System agents include RFID reader agents, database agents, GUI listener and controller agents, motor and robot agents, and others. Segregating the system components into independent, but interrelated, agents grants the system the flexibility to dynamic changes and upgrades. Furthermore, the ability of objects to interact through a network by exchanging XML messages provides standardization of device communication and omits the human intervention when a person is needed as an intermediary between two or more devices. Another feature that is developed for the testing system is the NBL programming language, which can not only allow the experimenter to design and code complex hardware scenarios before their execution, but enable him/her to perform logical and computational tasks as well. One last important point is that the developed EiA agent architecture can also serve as a model for laboratories that support other kinds of technologies such as robotics and smart-home devices.

Now that such architecture is implemented, individuals from any place in the world can perform experiments with RFID, obtain and analyze data, and deduce conclusions. This not only helps gain hands-on experience in the technology and support collaborative programs, but it also permits to evaluate the equipment's performance and find its flaws. Due to their high access to laboratory resources, learning environments of this kind have considerable potentials to help improve targeted technologies as well as rapidly increase the number of people who are proficient in them.

## Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No. DUE0633334. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

## Bibliography


1. S. Hsieh and P. Hsieh, "Animations and Intelligent Tutoring Systems for Programmable Logic Controller Education," *International Journal of Engineering Education*, vol. 19, no. 2, pp. 282-296, 2003.

2. S. Hsieh and P. Hsieh, "Web-Based Modules for Programmable Logic Controller Education," *Computer Applications in Engineering Education*, vol. 13, no. 4, pp. 266-279, 2005.

3. F. Kuester and T. Hutchinson, "A Virtualized Laboratory for Earthquake Engineering Education," *Computer Applications in Engineering Education*, vol. 15, no. 1, pp. 15-29, 2006.

4. H. Jiang, Y. Kurama, and D. Fanella, "WWW-Based Virtual Laboratories for Reinforced Concrete Education," *Computer Applications in Engineering Education*, vol. 10, no. 4, pp. 167-181 2002.

5. T. Ozer, M. Kenworthy, J.G. Brisson, E.G. Cravalho, and G.H. McKinley, "On Developments in Interactive Web-Based Learning Modules in a Thermal-Fluids Engineering Course," *International Journal of Engineering Education*, vol. 19, no. 2, pp. 305-315, 2003.

6. R. Gurbuz, "Web-Based Curriculum Development of a Manufacturing Technology Programme," *International Journal of Engineering Education*, vol. 20, no. 4, pp. 566-577, 2004.

7. S. Huang, Q. Su, N. Samant, and I. Khan, "Development of a Web-Based Integrated Manufacturing Laboratory," *Computer Applications in Engineering Education*, vol. 9, no. 4, pp. 228-237, 2001.

8. CompTIA, "RFID Adoption Trends in the IT Channel," White Paper, May. 2008.

9. M. Hamada, "An Integrated Virtual Environment for Active and Collaborative e-Learning in Theory of Computation," *IEEE Trans. Learning Technologies*, vol. 1, no. 2, Apr/Jun 2008, doi: 10.1109/TLT.2008.3.

10. E. Anderson, R. Taraban, and M.P. Sharma, "Implementing and Assessing Computer-Based Active Learning Materials in Introductory Thermodynamics," *International Journal of Engineering Education*, vol. 21, no. 6, pp. 1168-1176, 2005.

11. D. Diong, R. Wicker, C. Della-Piana, and R. Quintana, "A Laboratory Designed to Enhance Students' Interest in and Learning of Controls," *International Journal of Engineering Education*, vol. 20, no. 4, pp. 628-636, 2004.

12. IU News Room, "IU's Kelley School Receives Gift from Zebra Technologies for RFID Lab," http://newsinfo.iu.edu/news/page/normal/3154.html. 28 Mar. 2006.

13. M.C. O'Connor, "RFID Makes the Grade," *RFID Journal*, http://www.rfidjournal.com/article/articleview/1538/1/1/. 27 Apr. 2005.

14. C. Thompson, "Everything is Alive," *Architectural Perspectives Column, IEEE Internet Computing*, vol. 8, no. 1, pp. 83-86, 2004.

15. J. Hoag and C. Thompson, "Architecting RFID Middleware," *Architectural Perspectives column, IEEE Internet Computing*, vol. 10, no. 5, pp. 88-92, 2006.

16. International Telecommunication Union, "Internet of Things," http://www.itu.int/osg/spu/publications/Internetofthings/ InternetofThings_summary.pdf. 2005.

17. M. Wooldridge, "Agent-Based Computing," *Interoperable Communication Networks*, vol. 1, no. 1, pp. 71-79, 1998.

18. M.R. Genesereth and S.P. Ketchpel, "Software Agents," *Communication of the ACM*, vol. 37, no. 7, pp. 48-53, 1994.

19. N. Lehlou, N. Buyurgan, J.R. Chimka, "An Online RFID Laboratory Learning Environment," *IEEE Transactions on Learning Technologies*, accepted, 10 Aug. 2009.

## Appendix A

```
<!--Scenario: a set of commands to be executed-->
<Scenario Name="" Agent="" Parallel="False"
    Synchronous="False">
  <Command Motor="train2" Direction="FORWARD"
    Size="2" Unit="SECOND"/>
  <Command Motor="antenna3" Direction="CLOCKWISE"
    Size="3" Unit="SECOND"/>
</Scenario>

<!--LaunchAgent: to launch an agent in the receiver agent-->
<LaunchAgent FullClassName="Agentlib.Diagnostic"
    TranslatorPath="Agentlib"
    StringParam="DiagAgent">
</LaunchAgent>
<LaunchAgent
    FullClassName="Agentlib.Motors.QLibMotorAgent"
    TranslatorPath="Agentlib.Motors"
    StringParam="Engine1">
  <MotorAgentConfig>
    <!--Caution: value is in hexadecimal-->
    <Action key="" value="FFFFFFFC"/>
    <Action key="INWARD" value="00000001"/>
    <Action key="OUTWARD" value="00000002"/>
    <Action key="FORWARD" value="00000002"/>
    <Action key="BACKWARD" value="00000001"/>
    <Action key="CLOCKWISE" value="00000002"/>
    <Action key="COUNTERCLOCKWISE"
       value="00000001"/>
  </MotorAgentConfig>
</LaunchAgent>

<!--ACK: for agent-acknolegments-->
<ACK/>

<!--AgentConnected: to report an agent's connection coordinates-->
<AgentConnected Name="DiagAgent"
    Type="Diagnostic"
    IPAddr="192.168.0.4"
    IPPort="983"/>

<!--AgentDisconnected: to declare an agent's termination-->
<AgentDisconnected Name="ControlInterface"
    Type="GUI" IPAddr="10.0.0.2" IPPort="763"/>

<!--DiagnosticPacket: to report byte-transfer diagnostics-->
<DiagnosticPacket Payload="10" SeqNo="2"/>

<!--DiscoveryRequest: to discover connected agents-->
<DiscoveryRequest Type="" Name=""/>
<DiscoveryRequest Type="Diagnostic" Name=""/>
<DiscoveryRequest Type="Diagnostic"
    Name="DiagAgent"/>

<!--Ping: to ping an agent-->
<Ping/>

<!--PingReply: to reply to an agent's Ping-->
<PingReply/>

<!--Terminate-->
<Terminate/>
```

## Appendix B

```
scenario -> *nothing* | statement scenario

statement -> command | parallelStatement
| expressionStatement | ifStatement
| whileStatement | forStatement
| variableDefinition | displayStatement

expressionStatement -> expression SEMICOLON

forStatement ->     FOR O_PAREN [assignmentExpression] SEMICOLON expression SEMICOLON
[assignmentExpression] C_PAREN statement
| FOR O_PAREN [assignmentExpression] SEMICOLON expression SEMICOLON [assignmentExpression]
block

displayStatement -> DISPLAY O_PAREN argumentList C_PAREN SEMICOLON
| CELLDISPLAY O_PAREN argumentList C_PAREN SEMICOLON
| CARDISPLAY O_PAREN argumentList C_PAREN SEMICOLON
| CDRDISPLAY O_PAREN argumentList C_PAREN SEMICOLON

argumentList -> *nothing*
| expression
| expression COMMA argumentList

whileStatement -> WHILE O_PAREN expression C_PAREN block
| WHILE O_PAREN expression C_PAREN statement

ifStatement -> IF O_PAREN expression C_PAREN statement [elseStatement]
| IF O_PAREN expression C_PAREN block [elseStatement]

elseStatement ->    ELSE statement | ELSE block

block -> O_CURLY_BRACE scenario C_CURLY_BRACE

assignmentStatement -> assignmentExpression SEMICOLON

assignmentExpression -> VARIABLE ASSIGN expression
| valExpression ASSIGN expression
| carExpression ASSIGN expression
| cdrExpression ASSIGN expression

expression -> primary | primary operat expression

primary -> NULL | VARIABLE | INTEGER | DOUBLE
| STRING | valExpression | cdrExpression
| carExpression | NOT primary | MINUS primary
| O_PAREN expression C_PAREN | functionCall

functionCall -> VARIABLE argumentLists

argumentLists ->    O_PAREN C_PAREN
| O_PAREN C_PAREN argumentLists
| O_PAREN argumentList C_PAREN
| O_PAREN argumentList C_PAREN argumentLists

valExpression ->    VAL O_PAREN expression C_PAREN

cdrExpression ->    CDR O_PAREN expression C_PAREN

carExpression ->    CAR O_PAREN expression C_PAREN

variableDefinition -> VAR VARIABLE SEMICOLON
| VAR VARIABLE ASSIGN expression SEMICOLON
| VAR VARIABLE O_PAREN C_PAREN block
| VAR VARIABLE O_PAREN parameterList C_PAREN block
```

```
parameterList ->    VARIABLE
| VARIABLE COMMA parameterList

parallelStatement -> PARALLEL O_CURLY_BRACE commandList C_CURLY_BRACE
| PARALLEL ASYNC O_CURLY_BRACE commandList C_CURLY_BRACE

commandList -> *nothing* | command commandList

command -> MOTOR motor direction size [unit] SEMICOLON
| MOTOR motor IDLE size [SECOND] SEMICOLON
| IDLE [SECOND] SEMICOLON

Motor -> VARIABLE | STRING

direction -> IN | OUT | FORWARD | BACKWARD
| CLOCKWISE | COUNTERCLOCKWISE | FW | BW | CW | CCW

Size -> expression

Unit -> SECONDS | SECOND | SEC | METERS | METER | M
| FEET | FOOT | FT | DEGREES | DEGREE | DEG

operat -> PLUS | MINUS | MULTIPLY | DIVIDE
| MODULUS | POWER |EQUAL | DIFFER | SAME
| DIFFERENT | LESS_THAN | MORE_THAN
| LESS_OR_EQUAL | MORE_OR_EQUAL | CONDITIONAL_AND
| CONDITIONAL_OR | LOGICAL_AND | LOGICAL_OR

true -> 1

false -> 0
```

## Biographical Information

Nabil Lehlou obtained his Honors Bachelor in computer science from the University of Arkansas, Fayetteville, USA, in 2007, and his Masters in Industrial Engineering in 2008 from the same institution. He worked with Wal-Mart Information System Division as a programmer for one year starting in summer 2005. He joined the graduate program of the Industrial Engineering Department at the University of Arkansas in summer 2007, where he was assigned an NSF project of developing an online testing system for RFID technology. He is currently pursuing a Ph.D. in Industrial Engineering and working as a graduate assistant at the University of Arkansas. His research interests span RFID technology, agent-systems, heuristics and optimization, and renewable energy.

Nebil Buyurgan is an assistant professor of Industrial Engineering, director of the AT&T Material Handling Laboratory, and co-director of the AT&T Manufacturing Automation Laboratory at the University of Arkansas. After receiving his PhD degree in Engineering Management from the University of Missouri-Rolla, he joined the Industrial Engineering Department at the University of Arkansas in 2004. His research and teaching interests include modeling and analysis of discrete event systems, supervisory control systems and distributed control, and Auto-ID technologies. He has directed several projects funded by National Science Foundation, Air Force Research Lab, and Wal-Mart Stores.

Justin R. Chimka is an assistant professor in the Department of Industrial Engineering at the University of Arkansas. His PhD with a major in industrial engineering is from the University of Pittsburgh. Justin's academic interests include education research, and statistical quality control.