Utilizing Lego Mindstorms as a Teaching Platform for Industrial Automation

Carolyn Oates, Alois Zoitl
Automation and Control Institute (ACIN)
Vienna University of Technology
Vienna, Austria
oatesc@acm.org , zoitl@acin.tuwien.ac.at

Abstract—**Industrial control systems are taught best using real systems. Such systems can be expensive, dangerous, and may break easily. In the other side simulations often do not react like the real system. IEC 61499 automation standard supports the current control system trend toward networks of event-driven distributed devices. Support for event driven control applications is new in IEC 61499 as are the tools supporting it. Three tutorials are presented to teach developing IEC 61499 event driven applications along with control theory basics using open source tools with the Lego™ Mindstorms hardware. This inexpensive training system can be used for teaching industrial control methods for students, as well as industrial professionals.**

Keywords: automation; control systems; robotics

I. INTRODUCTION

Real control systems, such as an industrial robot arm, are expensive; can be dangerous [3]; and may break easily. In a simulator timings and physical modeling often do not react like the real system, teaching the students only the software. Additionally industrial automation systems are undergoing a major transition towards distributed control systems adding new development paradigms. The problem of industrial automation education has been summarized by [12] as follows:

"During the last few years the education in engineering and mainly the control engineering, has suffered multiple changes due to the fast technological development and the current demands of the field."1

In order to support industrial automation engineers, the IEC developed standards to define how distributed control systems should be developed. The result of this standardization activity is the IEC 61499 [3], which provides a framework for networks of event-driven distributed industrial control systems. IEC 61499 applications are built using networks of new kinds of functions blocks (FBs), supporting event as well as data connections. Support for both event driven and distributed control applications are newly supported and required for the first time industrial automation by IEC 61499. The new kinds of FBs which support distributed control applications need to be learned. Although the standard is available now for nearly five years, little tutorial information is available. As new open source based tools like the 4DIAC–Framework for Distributed Industrial Control–are becoming available the gateway hurdle for adopting the new technology is greatly reduced.

[1] [12] II p. 3432

However a key open point for learning IEC 61499 based distributed control systems is the missing availability of cheap easily available training systems. Lego™ Mindstorms (LMS) offers with its building kit a flexible way of building small automation problems. Furthermore with the new system NXT it provides about the same computing performance as typical control devices used in the domain of industrial automation. With this work we like to show how LMS can be used together with 4DIAC to teach IEC 61499.

LMS has a great history for teaching robotics and control programming also with block like programming languages. However none of the available tools provides languages suitable for industrial automation engineers.

Lego™ Mindstorms software (a subset of Labview) allows sequential commands. So when using LMS software to blink the LED located on the light sensor, there is typically one light sensor block for *on* and another for *off* for the same light sensor. The same sensor may be tested in different phases of an application using different blocks. Telling the motor to move occurs via multiple TurnMotor blocks. Labview has data connections, but no event connections [7].

Lejos, Java on LMS, is object oriented so there is only one instance of a physical sensor, but the method to reads a sensor can be used multiple times. Behavior programming described in the Lejos tutorial can still reference the same instance in multiple behaviors [9]. In comparison FB instances are restricted by the standard to the one physical existence.

This article is structured as follows. In Section 2 we give a short introduction to IEC 61499. The environment is described in Section 3, followed by a description on how we developed the tutorials. The developed tutorials are described in Section 5. Finally we conclude the article and describe our next planed steps.

II. SHORT INTRODUCTION TO IEC 61499

The standard IEC 61499 defines several models–the application model, the system model, the device model, the resource model, and the Function Block (FB) model–that allow the control engineer developing distributed control applications in a graphical manner. This short introduction to IEC 61499 should serve as basis for the rest of this thesis. A full description of IEC 61499's architecture may be found directly in the standard

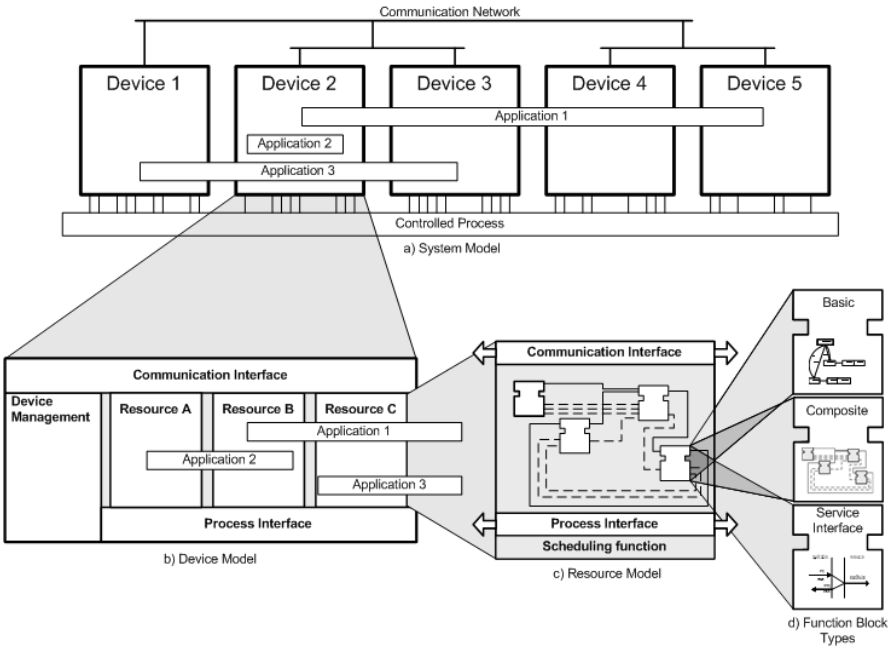

Figure 1. Overview on the main models of IEC 61499

IEC 61499-1 [6] or in a more comprehensible form in the books from Lewis [4] and Vyatkin [5].

The base model of IEC 61499 is the FB. A FB is a software component that is self contained and provides its functionality through a defined interface. This model has been adopted from the preceding standard IEC 61131-3 [11] and extended in its interface with an additional event interface. A trigger on one of the event inputs starts the execution of a FB. During the execution of the FB the input data will be processed, output data will be generated (depending on the functionality of the FB), and/or output events will be triggered. IEC 61499 defines three different FB types (schematically shown in Figure 1d):

Basic FBs (BFB) contain as main element a state machine that controls the internal execution on an input event arrival. This state machine is called Execution Control Chart (ECC) and is based on the *Sequential Function Charts* of IEC 61131-3. The ECC consists of three main parts: *ECC-states* with associated *ECC-actions* and *ECC-transitions* connecting the states. ECC-transitions are guarded by conditions. On an input event arrival the conditions of the current state's outgoing transitions are evaluated. The first true condition results in a state change. On state entry the associated actions of the state are executed. Actions consist of the execution of algorithms and/or triggering of output events. Algorithms may be programmed in any programming language. The main restriction is that algorithms can only access data inputs, data outputs, and internal variables.

Composite FBs (CFB) serve as container for FBs and may contain a whole set of FBs and their event connections and data connections. Incoming event connections and data connections are passed on to the internal FBs and vice versa for outgoing connections.

Service Interface FBs (SIFBs) provide a FB interface to functionality which is beyond the means of IEC 61499. Typical functionality encapsulated within SIFBS is the access to the control device's hardware, like the I/O interface or the communication interface. But also existing libraries that provide functions needed for the control system may be used through SIFBs. With SIFBs, this functionality can be encapsulated and the usage can be documented with so called service primitives. These service primitives allow to model event/data sequences explaining the usage of the SIFB. IEC 61499 distinguishes two general types of SIFBs. One is the requester SIFB, the other is the responder SIFB. The requester SIFB is an application triggered FB which remains passive until an event arrives at one of its event inputs. The responder type is a resource or hardware triggered FB. That means that it can send output events resulting on actions in the resource or the hardware (e.g.\ interrupts).

Through interconnecting the FBs with event connections and data connections to Function Block Networks (FBNs) the control functionality can be modelled in the application model. Applications are in general modelled without any device or control infrastructure in mind. The control equipment with their communication networks used for the data exchange between the distributed controllers is specified in the system model. A second part of the system model is the so called mapping. The mapping regulates which parts of the application are located on which control device. For example in Figure 1a Application 1 is mapped to the Devices 2, 3, 4, and 5; whereas Application 2 is mapped only to Device 2.

IEC 61499 models control equipment that is capable of executing IEC 61499 applications as devices. A device consists of a communication interface, a process interface, a device management, and may contain resources (see Figure 1b). The communication interface provides communication services for the device and the application parts residing in this device. The process interface provides the services for accessing the sensors and actuators needed to control the process (e.g. read the current motor position).

A resource is a functional unit that serves as containment for applications or application parts residing in the specific device and has independent control of its operation. Within a device resources can be created, deleted, configured, etc. without interfering with other resources and their contained applications. For applications a resource has to provide an execution environment (Figure 1c). That means it has to deliver event notifications to FBs and has to allow FBs to process the incoming events corresponding to their internal structure. A resource gets access to the communication interface and process interface from the device. SIFBs are the means to provide these services to the applications.

The management functionality within a device has the main task to administrate all applications and all resources located in this device. The management also provides an external interface for engineering tools allowing engineering tools downloading and uploading applications to (from) the device. This external interface is provided through the communication interface. Therefore the management needs an access to the communication interface (Figure 1b). At device level it provides the services to create, initialise, start, stop, kill, and delete the instances of resources and to query the attributes of resources. At resource level the same services allow the handling of FB instances and their interconnections.

III. Environment

The environment uses only open source applications. The 4DIAC-IDE is used to develop IEC 61499 standard compliant systems, applications and FB types. The standard provides portability and plug&play for controller applications. Applications are uploaded on to the Lego™ "controller" hardware [8] running the 4DIAC RunTime Environment (FORTE) under eCos operating system.

Figure 2. shows two IEC 61499 development tools, 4DIAC and FBDK. The tools generate XML files which comply with the IEC 61499 standard and can be exchanged.

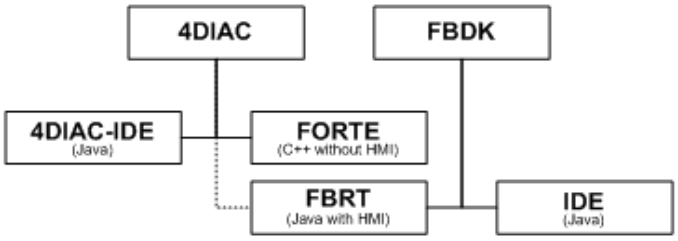

Figure 2. Development Environment

As shown in Figure 2. , the function blocks and applications are developed and mapped to device resources via 4DIAC-IDE. FBs are then exported to FORTE. The 4DIAC FB type export translates the FB's IEC 61499 XML representation into C++ code suitable for FORTE.

Figure 3. Upload to Lego™ Mindstorms NXT

FBDK FBs are reusable, so a simulation using FBDK HMI FBs to display the output is possible. This is useful for unit testing FBs inputs and outputs by event. The "device" is a Java window.

After the FBs are developed the LMS firmware must be flashed with the eCos+FORTE using SAM-BA [1] (see Figure 3). SAM-BA is provided by Atmel, the maker of the at91sam7s (ARM7) chip in the LMS [8]. At this point FORTE is running a simple Ethernet over USB program to upload and run an application.

eCos is an reconfigurable embedded operating system, so only the resources that exist in a device must be included. Control systems are typically embedded systems. Students who learn to work with LMS with eCos have a head start using eCos on other control devices.

IV. Course Development

The tutorials assume no automation background. The tutorials build up concepts stepwise. Beginning FBs and IEC 61499 applications developed are reused and refined in following steps and tutorials. A simple example is presented and then the student must create or refine the presented example.

Research by Lego™ and MIT encourages the use of the freer explorative constructivist philosophy of education [2] by letting students explore rather than directed learning. However the problem solving cognitive philosophy is also popular for teaching control theory [12]. Teaching of basic concepts to model the problem need to be more guided. Once the student has framework to model the problem, they can be given more freedom and still communicate their work using IEC 61499 standard.

These tutorials are a mixture of cognitive and constructivism teaching philosophies. The first tutorial is guided problem solving learning, because specific control theory concepts using IEC 61499 standard are to be taught. After a basic example a related task, but slightly harder task is assigned. The second tutorial is meant to allow the student more freedom to use what they have learned. The only new concept is composite FBs. The third tutorial is a mixture and has the goal to teach the concepts of buffering and use of a bus.

Figure 4. shows a typical control loop. A line follower uses a controller to stay on the line. Calibration and software connection to hardware are also typical tasks in automation.

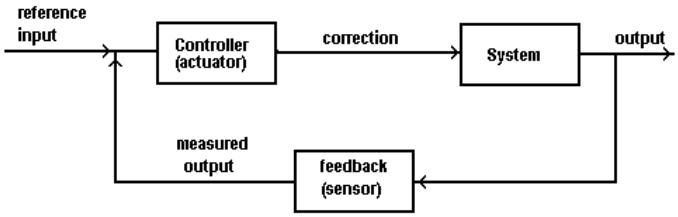

Figure 4. Feedback control loop

Tasks were chosen to teach control theory concepts as well as development of control application using the standard and tool.

In [5] three tutorials are presented for IEC 61499. The first tutorial modifies an LED application with 4 LEDs. The LEDs blink, or "chase" up or down. Turning an LED on and then blinking the LED was used as the first real test case for 3 different devices including LMS this semester. The NXT LabView Configuration VI also uses setting the light sensor's LED as part of an example NXT software block [7].

In [5] the second tutorial used simple equation as the first full application. We tried a similar internal tutorial with 4DIAC/Forte for $x^2 + y^2$ with a network-like interface between FORTE (C++) and FBDK (Java). However there were many questions from students afterwards, especially about FORTE. There were fewer questions after developing an application to blink an LED. The blink application was first simulated with FBDK in a Java window and then applied to the actual hardware. The toggling the LED FB was reused in later applications.

We identified a set of key concepts of IEC 61499 and automation engineering for which it is important the trainee grasps in the first tutorials:

- How to represent feedback and feed forward control in IEC 61499
- IEC 61499 devices represent control hardware
- Sensors and actuators are represented by SIFB. Typically you have one instance of an SIFB per sensor or actuator.
- Error handling is performed through Boolean FB interface variables and appropriate events.
- Boolean input qualifier named QI is used for turning event processing on and off.

V. TUTORIAL APPLICATIONS

The three tutorials developed teach the use of IEC 61499 function blocks to build three working applications. First the environment (4DIAC, [10]), hardware (LMS, [8]), and standard with a simple application are taught. A LMS FB library is provided with FBs to directly interface with the LMS hardware. This includes sensors, motor, and shutdown, plus hardware status (battery power status). A LMSUtil library will be developed during the tutorials.

Sensor FBs are associated with physical sensors or motors. Students must be careful to send and receive events to the FBs associated with exactly one physical resource. The second tutorial application uses a different kind of sensor hardware. The third tutorial application teaches timing and using buffers to send information between applications.

A. Tutorial 1: Line Follower

The first tutorial is a line follower application with on- and off-the-line calibration. If multiple light sensors are available the application can be expanded to use 2 or 3 light sensors.

We want to test if it more understandable to start with Basic FB (BFB) or a Service Interface FB (SIFB). A greater than BFB will be explained first. Then a two point controller (hysteresis) basic FB is assigned. So they go from a "one point" controller" to a two point controller.

For teaching how to provide an interface via a FB to the hardware the student is asked to develop a SIFB for the Lego™ light sensor. This should also help the student understand what the purpose FORTE C++ Eclipse compilation is for. The sample FB will be the touch sensor, which reads data the same as the light sensor. The light sensor ports must be initialized, which shows the direct connection to the ARM7 processor. The test application is a light blinking application utilizing the light sensor's LED. The Boolean data input QI is initialized to true. Errors indicated by the output variable QO=false are ignored for the moment.

The light blinking application also introduces the important and often needed Event FB library, which provides FBs for manipulating the event flow as well as timed events (i.e., cyclic triggers or delayed events).

Next, the boundary between on-the-line and off-the-line for the environment and a light sensor must be found. First the light sensor must be read and connected into the calibration calculation. There is no single way to do two-color calibration, but it's important how the process knows and handles reading different colors. The top part of Figure 5 shows calibration where all samples of one color are read and all samples of the second color. All samples are averaged together.

Next the boundary between the two colors is used to turn on an LED if the light sensor is over "black" and off when it is over "white" as shown by DarkTst FB and Led4 FB in bottom part of Figure 5. Here it is emphasized that a port should only be used once.

Without error checking it is possible that a second FB for the same port/light sensor will be erroneously used. So error handling and Service Sequence diagrams explained must be explained. IEC 61499 uses + events to indicate no error and – events indicate error condition.

The application should now react when a port is allocated twice, since only one resource can be connected to a port. The INITO event combined with QO, event qualifier is split via an event switch into INIT- (QO=false) and INIT+ (QO=true). If the student previously used the same port/light sensor their design error will cause their application to no longer work.

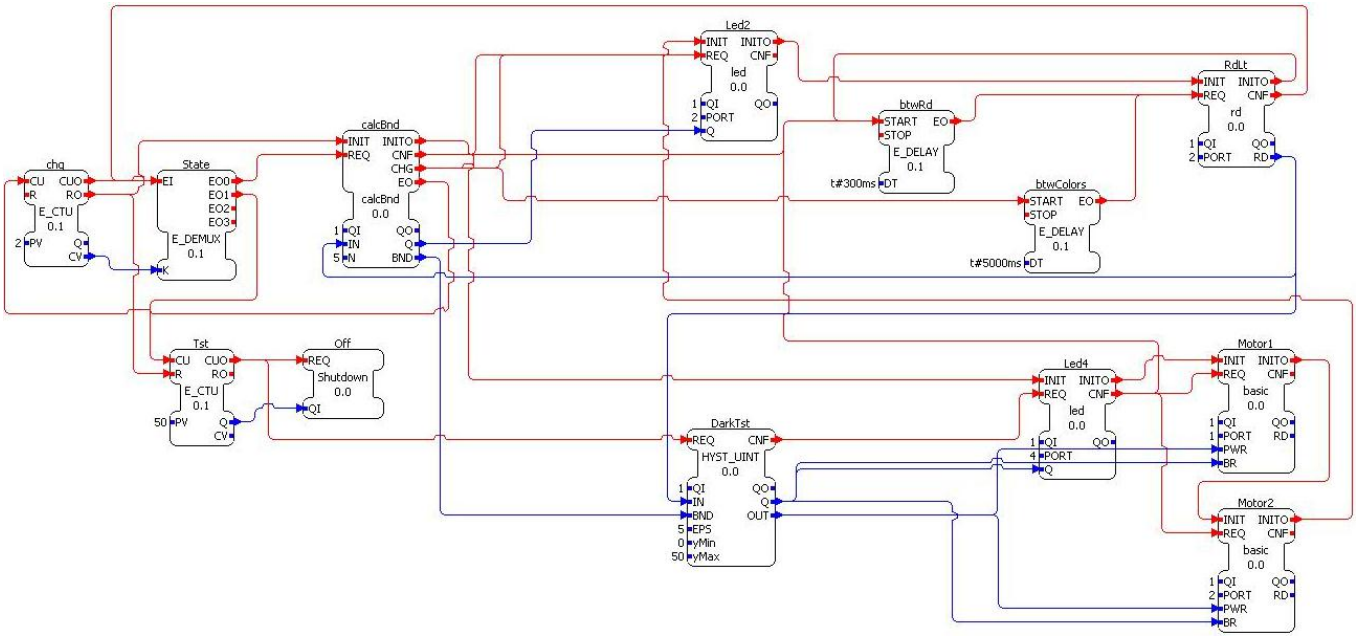

Figure 5. Simple Line Follower with light calibration

From personal experience making this failure helps the student remember each FB represents one real physical resource.

Finally the state of the light sensor can be used to tell the motors how to move can now replace toggling an LED. This final application should also be developed stepwise. First instead of just turning the LED on and off, the 2 motors can be set. The LED toggle is left for debugging. Toggling one motor off when not over the dark line allows the application to follow the line in one direction only. This simple line follower is shown in Figure 5. When a basic version is working, then the application can be expanded to a general line follower with feedback control and finding the line. Instead of just one light sensor to detect if the robot is over the line, multiple light sensors could be used. For example if three light sensors are used, the middle light sensor is over the line and other two light sensors straddle the line.

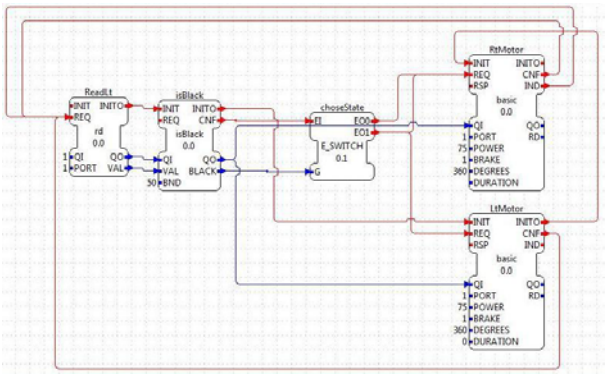

Figure 6. Simple Line Follower using composite FBs

B. *Further Tutorials*

The second tutorial uses the ultrasonic sensor as part of a simple Cartesian robot to keep a certain distance from the car. A Lego™ robot must be built to move forward and backwards based the "car's" length and up and down based on the feedback from the ultrasonic sensor. The student must develop their own control loop and Lego™ robot. Developing composite FBs is introduced to combine FBs together. Figure 6 shows how composite FBs would simplify the simple line follower by encapsulating the light calibration. The student must be careful to include error checking when needed in the composite FBs. Since only the input/ output events and variables are seen care must be given to not accidently reuse the same port.

The third tutorial uses stations to detect an object, its color, accept or reject it, and optionally deliver it. A pick-and-place robot is suggested. The application stations detect information and pass it on ahead of time via buffers, so the next station is prepared when the object arrives. This application teaches buffering data with time deadlines

The IEC 61499 tutorial examples can build on each other if multiple LMS NXT kits are available. The car is used in car wash and the Cartesian robot as one station in the assembly line.

VI. Conclusions and Future Work

Industrial automation is phased with major paradigm changes. First distributed control systems require a complete rethinking of how control applications are developed. By providing cheap and available tutorial systems control engineers can move up to the new paradigm much faster. With this work we showed how Lego™ Mindstorms NXT can be such a train-

ing platform for the new standard IEC 61499. We are developing tutorials which on the one hand utilize the LMS hardware and on the other side are representatives for typical industrial automation tasks. The final tutorial versions will appear on the 4DIAC website [10] under the development wiki.

Our next steps are to test the tutorials on different user groups in order to validate the contents and the structure of the tutorials. The first tutorial will be tested with students doing a practice work for the institute this summer. The last two tutorials will be tested with students in the fall. We also plan to use wireless communication between Lego™ Mindstorms NXT to teach using devices and applications across device boundaries.

REFERENCES

[1] ATMEL, AT91 ISP/SAM-BA® User Guide, http://www.atmel.com/dyn/resources/prod_documents/6421B.pdf.

[2] Mindell, D., Beland, C., Wesley, C., Clarke, D., Park, R., Trupiano, M. 2000. LEGO mindstorms, the structure of an engineering (r)evolution, 6.933J Structure of Engineering Revolutions, http://web.mit.edu/6.933/www/Fall2000/LegoMindstorms.pdf.

[3] S. Greengard, *Making automation Work*, Comm.ACM, Dec.2009, Vol.52,No.12, pp.18-19, http://doi.acm.org/10.1145/1610252.1610261.

[4] R. Lewis, "Modelling Control Systems Using IEC 61499 – Applying Function Blocks to Distributed Systems", The Institution of Electrical Engineers, London, 2001.

[5] V. Vyatkin, IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design, O3neida Publ.

[6] IEC TC65/WG6, IEC 61499: Function Blocks, Parts 1 – 4, International Electrotechnical Commission IEC Std., Rev. 1.0, 2004/2005.

[7] Labview, Creating Lego Mindstorms NXT Software Blocks, ftp://ftp.ni.com/evaluation/mindstorms/NXT_Creating_MINDSTORMS_Blocks.pdf

[8] LEGO™ MINDSTORMS NXT Hardware Developer Kit, http://mindstorms.lego.com/en-us/support/files/default.aspx

[9] LeJos Tutorial, http://lejos.sourceforge.net/nxt/nxj/tutorial/index.htm

[10] 4DIAC – Framework for Distributed Industrial Automation and Control, www.fordiac.org

[11] IEC TC65/WG6, IEC 61131-3: Programmable controllers – Part 3: Programming languages. International Electrotechnical Commission, Geneva, 1993

[12] Paja, C., Scarpetta, J.. and Mejia, E. Platform for Virtual Problem-Based Learning in Control Engineering Education, p-3432-3437, IEEE EDUCON 2010, April 2010.