Int. J. Engng Ed. Vol. 20, No. 2, pp. 000–000, 2004
Printed in Great Britain.

0949-149X/91 $3.00+0.00

Robot Control Teaching with a State Machine-based Design Method*

ILYA LEVIN, ELI KOLBERG and YORAM REICH
Tel Aviv University, Israel. E-mail: i.levin@ieee.org

Mechatronics design provides an excellent project-based learning activity in engineering education. It weaves together the Computer, Mechanics, and Electrical Engineering Curriculums, forming one of the key issues in all of them. This paper proposes a design method for control of a robot that can be used as a core part of a mechatronics course. This method includes: a) a universal formal notation including the concepts of ASM (algorithmic state machine) and FSM (finite state machine), as a basic aspect of designing a mechatronics control system, and b) an interactive learning environment developed on the basis of the formal notation. In this paper, both of the above components are presented in the context of a specific mechatronics design course based on a mobile robot contest. The proposed approach decreases the gap between theoretical and practical skills of students in mechatronics thus leading to a better robot design with a better contest-related performance; improves the real robot performance; and opens up a way to enrich mechatronics lessons by increasing the number of possible tasks and projects in a class.

GLOSSARY

ASM: algorithmic state machine
CPR: counts per revolution
DC: direct current
FSM: finite state machine
IR: infra-red
LED: light-emitting diode
PID: proportional integral derivative
PWM: pulse width modulation
RAM: random access memory
SMILE: state machine interactive learning environment
UV: ultra violet
VHDL: very high-speed integrated circuits hardware description language.

INTRODUCTION

IT IS a widely known phenomenon, that students experience difficulties while establishing a comprehensive interconnection between theoretical knowledge of a complex subject and a practical knowledge of the same subject. In particular, there is a gap between theoretical university courses from corresponding engineering curricula on one hand, and practical courses of design on the other hand. This gap is natural and fundamental, since the theoretical courses are focused on analytical and optimization skills, while the practical courses are oriented towards developing an ability to synthesize technical solutions.

It goes without saying that bridging such a gap would be a desirable purpose for a teacher and a fruitful achievement for a student during the educational process. It has become possible owing to both modern technological means, and newly developed design methods.

Designing the control part of robots is a known type of learning activity. Started in MIT (LEGO-LOGO, 6.270 MIT Contest) [1, 2], robot control design has become an accepted educational practice. The idea to introduce the state transition methodology into control curricula was proposed by Lewis [3]. The state machine-based approach is also widely used, for example by Brooks [4].

In the present paper, we develop a state transition methodology for the field of control design and show some directions of introducing this methodology into education practice.

In the frame of this novel educational approach, we propose to use a specialized means for designing control systems, which is a specific toolkit called ControlWare [5]. Blocks of such a specialized toolkit, when being presented and given to students for designing a control system in a class, enable construction of the system, while synchronously displaying it both as a state diagram of a finite state machine (FSM) and as a flow-chart of an algorithmic state machine (ASM). The simultaneous construction and display of the control system in these representations demonstrate unity of the theoretical and practical approaches to the control logic design. The teacher is therefore able to give, and the student is invited to acquire, both the theoretical knowledge and the practical design skills concerning the subject.

CONTROLWARE VERSUS SOFTWARE

The concept of ControlWare was first proposed in [5]. ControlWare was defined as a special toolkit

* Accepted 5 October 2003.

specifically created for designing the control part of any equipment in interactive learning environments. ControlWare differs from traditional software in the following significant ways:

1. While software is a universal means, ControlWare is oriented to programming of control units.
2. All traditional programs manipulate data flows while ControlWare employs control flows.
3. ControlWare exhibits a high degree of transparency; for instance, a student can immediately see the role and importance of behavior of the equipment.
4. ControlWare offers both a rich set of learning activities and direct proximity between the definition of equipment behavior and their implementation environment.
5. While traditional software is usually oriented on standard sequential computer architecture, ControlWare is based on a parallel architecture, which is characteristic for control units.
6. A software program consists of a set of instructions, where each instruction causes the computer to carry out a certain operation. A ControlWare procedure comprises a non-algorithmic set of intelligent bricks. Creation of a complex control system is then a process of connecting such bricks.

Any ContolWare toolkit presents the following three issues: 1) a subject matter, 2) a technological means of teaching, and 3) a formal model used during the teaching. In [5], ControlWare comprises a logical control concept as a subject matter, a spreadsheet software environment and educational mobile robots as technological means, and state machines (FSM and ASM) as formal models of the control unit to be designed.

In the present paper, we introduce ControlWare based on different components. We will deal with mechatronics as the subject matter. The formal model will be a state machine notation. The technological means will be a specially developed interactive learning environment.

MODEL OF MECHATRONICS SYSTEM

We consider a mechatronics system as a composition of *control* and *operational* units [6]. The operational unit of the system contains such building blocks as motors, sensors, lamps, manipulators, etc. A control unit receives information from the operational unit and produces the sequence of control signals that leads to executing desired operations by the operational unit.

Let micro-operation be an elementary step of processing in the operational unit of the mechatronics system, and let $Y = \{y_1, \ldots, y_N\}$ be a set of micro-operations initiated by the binary signals $y_1, \ldots, y_N$ from the control unit. In turn, the control unit receives binary signals $X = \{x_1, \ldots, x_L\}$ arriving from the operational unit.

In [7, 8], two main paradigms were suggested as conveyors of very different cognitive approaches to designing control units: the programming and the design paradigms.

The key formal concept of the programming paradigm is the Algorithmic State Machine (ASM) [6]. An ASM is a directed connected graph containing an initial vertex (Begin), a final vertex (End), a finite set of operator vertices, and conditional vertices. The final, the operator, and the conditional vertices have one input each, and the initial vertex has no input. The initial and operator vertices have only one output each, and each conditional vertex has two outputs marked by '1' and '0.' The final vertex has no outputs. One of the logical conditions (input binary variables of the control unit) of the set $X = \{x_1, \ldots, x_L\}$ is written in each conditional vertex. The micro-instruction $Y_t = \{y_{t_1}, \ldots, y_{t_U}\}$, which is a subset of the set of all micro-operations $Y = \{y_1, \ldots, y_N\}$ that may be performed concurrently, is written in each operator vertex; $y_{t_u} \in Y$, $u = 1, \ldots, U_t$.

An example of a specific ASM is presented in Fig. 1. In this figure: $X = \{x_1, \ldots, x_6\}$ is a set of input variables of the control unit, $Y = \{y_1, \ldots, y_{10}\}$ is a set of micro-operations, and $F = \{Y_1, \ldots, Y_4\}$ is a set of micro-instructions, where, for example, $Y_1 = \{y_1, y_3, y_5\}$. Notice, that here we relate to the ASM just as a formal notation, while the content of this ASM and the description of the corresponding control unit is done below.

The alternative design paradigm is based on the idea that the control unit is presented as a FSM. The control unit can be characterized by its state and may perform different functions (e.g., changing to other states) depending upon its current state. A formal construct, which we consider the most appropriate for the formal-model definition, is the state diagram. The state diagram is a representation of the system's possible states and possible transitions between them. Nodes in the diagram indicate states, and arrows indicate transitions between the states caused by specific input values. Also, the FSM can be represented in the form of a state table (i.e. a tabular form of the state diagram).

An example of a specific FSM corresponding to ASM from Fig. 1 is presented in Table 1. Columns of the table indicate in sequence: h number of a transition of the FSM; a_m a current state; a_s the next state; $X(a_m, a_s)$ an input signal, which is a logical function equal to one of the transitions from a_m to a_s; and $Y(a_m, a_s)$ a corresponding output (micro-instruction).

We will say that FSM implements a corresponding ASM. Any ASM can be transformed to the FSM form, and vice versa. To perform the transformation from ASM to FSM, the following steps have to be taken. First, the ASM has to be marked by marks reflecting states of FSM. The second step is the searching for paths between the marks within the ASM. Every such path has to

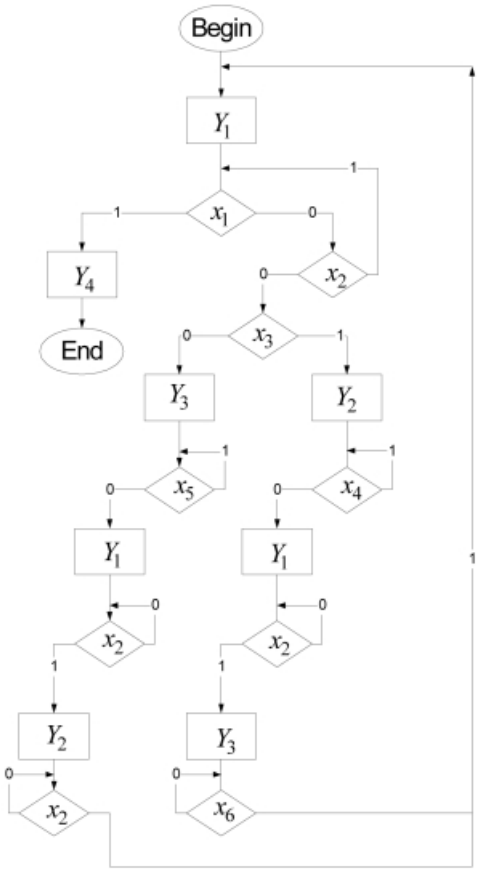


Fig. 1. ASM of the right wall navigation of the robot.

include one operator vertex. Each path can be interpreted as a transition within the FSM. We will represent the FSM as a list of transitions, which are paths of the initial ASM.

MECHATRONICS PROJECT 'FIRE FIGHTING ROBOT'

The aim of this section is to show an application of the proposed method of control design by an example of a particular mobile robot used in a classroom. For this goal, we have chosen an autonomous robot [9] built by high school students for the Fire Fighting Home Robot Contest (Trinity College, USA, contest rules can be seen in: http://www.trincoll.edu/~robot).

Table 1. FSM of the ASM from Fig. 1

h	a_m	a_s	$X(a_m, a_s)$	$Y(a_m, a_s)$
1	a_1	a_2	1	Y_1
2	a_2	a_1	x_1	Y_4
3		a_2	$\overline{x_1}x_2$	Y_0
4		a_4	$\overline{x_1}x_2x_3$	Y_2
5		a_3	$\overline{x_1}x_2\overline{x_3}$	Y_3
6	a_3	a_3	x_5	Y_0
7		a_5	$\overline{x_5}$	Y_1
8	a_4	a_4	x_4	Y_0
9		a_6	$\overline{x_4}$	Y_1
10	a_5	a_5	$\overline{x_2}$	Y_0
11		a_7	x_2	Y_2
12	a_6	a_6	$\overline{x_2}$	Y_0
13		a_8	x_2	Y_3
14	a_7	a_7	$\overline{x_2}$	Y_0
15		a_1	x_2	Y_0
16	a_8	a_8	$\overline{x_6}$	Y_0
17		a_1	x_6	Y_0

In this contest, a robot navigates through a maze with four 'rooms' and searches for a room with a lit candle in it. When it finds the room, it should enter, scan for candle position, go to a distance of 30 cm or less from the candle, and extinguish it. There are white lines in each room entry and also an arc of 30-cm radius around the candle.

There are optional bonus points for more difficult modes like non-dead-reckoning navigation, 'furniture' (obstacle) avoidance, sound activation, return trip to starting point and more. There are penalty points as well, for hitting the wall, and others. Run time for each trial is limited to 5 minutes.

The arena size is $2.5\,\text{m} \times 2.5\,\text{m}$, four different rooms, and 46-cm wide corridors. The robot size is limited to a maximum of $31\,\text{cm} \times 31\,\text{cm} \times 31\,\text{cm}$ cube. The robot must be autonomous, must not change its shape during the contest, and must not harm humans, or damage other robots or the arena.

One robot designed for this contest is shown in Fig. 2. According to the proposed approach, we concentrate on developing the robot's ControlWare taking into account the composition and the structure of the robot's operation part.

Operation part of the robot

The operation part of the robot includes sensors, actuators, and drivers, assembled for performing the robot's goal. We show here a plurality of components that comprise both the control and the operational units of the robot. All these components have to be assembled in a form that enables to achieve the general complex goal of the robot.

As can be seen from figures presented below, sensors, actuators, and drivers are quite delicate components of the system that could be damaged during normal operation of the robot or even

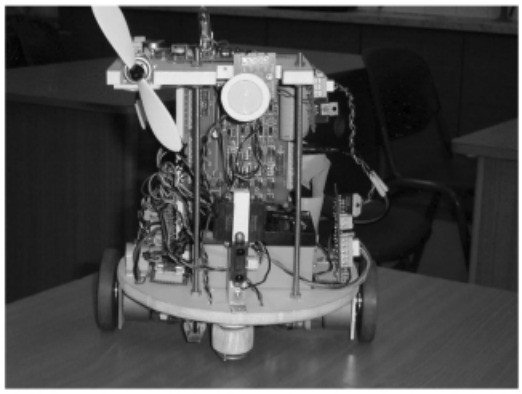


Fig. 2. Fire fighting robot.

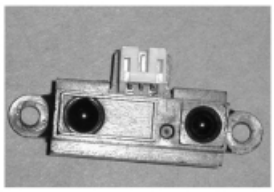


Fig. 3. IR analog distance sensor (GP2D12).

during any treatment performed by students. Consequently, improved robust control should be applied to overcome such incidences. One can also see that careful design should be considered for optimizing the use and operation of this robot's components.

Below we show the components of the robot's operation and control units.

Robot sensors include:

1. Six IR analog distance sensors (GP2D12) (see Fig. 3) are used: two on the right and left sides, one in the front, and one in back side of the robot. The sensor's output varies from 0.6 V for a distance of 80 cm or larger from an object up to 2.6 V for a distance of 10 cm.
2. One ultrasonic distance sensor (see Fig. 4) is placed in front of the robot. When a 10-μs pulse is fed to its trigger pin, a positive pulse relative to the distance from the object appears at an echo output pin of the sensor.
3. A digital UV sensor (see Fig. 5) is used for detecting a lit candle when a robot is passing the room entrance; the sensor output produces a 10-ms positive pulse every 30 ms when the lit candle is present.
4. An analog pyroelectric sensor (see Fig. 6) is intended for aligning the robot towards the candle. Its output is stabilized at 2.5 V. During the relative movement between the robot and the candle, the sensor passes against the candle, its output goes down to 0 V and then to 5 V and back to 2.5 V while rotating in one direction (e.g. right to left). When rotating in the other direction, the output of the sensor is reversed, first going to 5 V, and then down to 0 V and back to 2.5 V.
5. A digital white line sensor with its driver (see Fig. 7) is used for detecting a room entrance, and the white line around the candle. Its output goes from high to low when a white line is detected.

Fig. 4. Ultrasonic distance sensor.

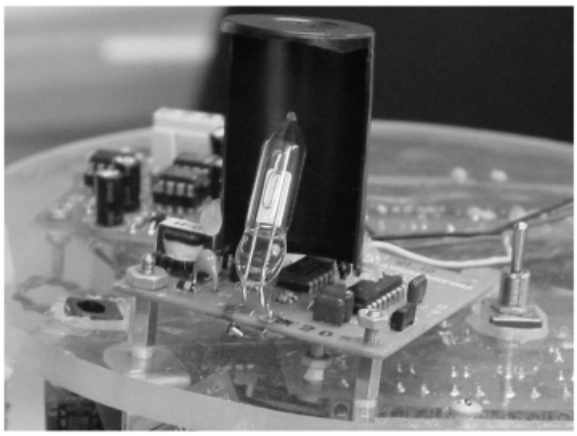


Fig. 5. Digital UV sensor.

6. An analog microphone sensor (see Fig. 8) is intended for detecting the 3.5-KHz start signal. This sensor makes frequency to voltage conversion and issues an analog voltage, which is proportional to the frequency depicted by the microphone. A frequency of 1 KHz will cause a voltage of 1 V at the output, 2 KHz will give 2 V at the output, etc., up to 5 KHz.

We will discuss, as an example, building and using the above sensor. This sensor may be used for a relatively low cost frequency catcher, like a hand clamp or a bark or a whistle etc., a tuning device, tone code key, software filter, etc. Figure 9 schematically shows such a circuit. Its three stages are based on three chips: LM386 low voltage audio power amplifier, LM2917 frequency-to-voltage converter, and TC7660 DC-to-DC voltage converter.

The first stage is an amplification stage, which takes the microphone output and amplifies it with a gain of about several tens. The next stage is a frequency-to-voltage converter using the LM2917. This chip is divided to a tachometer section and an operational amplifier (op-amp) section. The chip is used here with a ground-referenced tachometer input and an internal connection between the tachometer output and the op-amp non-inverting input.

Following a data sheet recommendation for the resistors and capacitors, with small changes, and

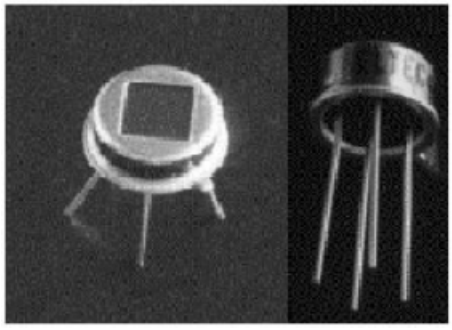


Fig. 6. Analog pyro-electric sensor.

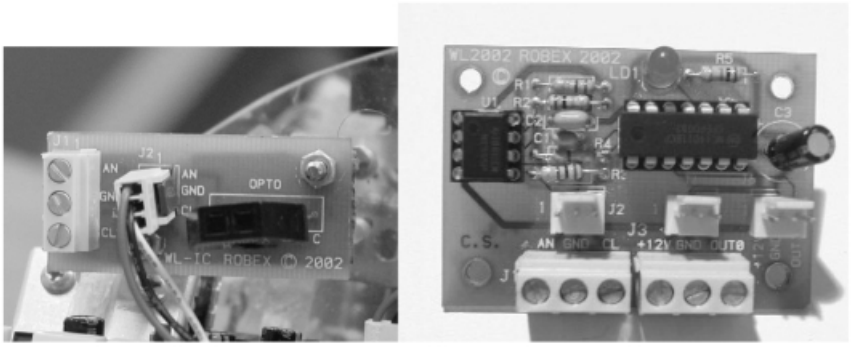
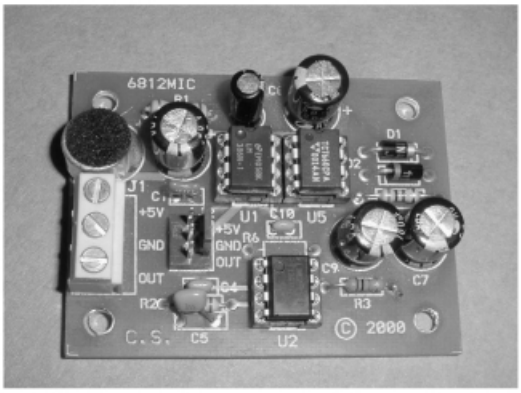


Fig. 7. Digital white line sensor and driver.



disconnecting the output transistor collector from the op-amp inverting input, the chip's output is a linear function where for 1 KHz it outputs 1 V and for input of 2 KHz it outputs 2 V, etc. Note that the chip cannot handle more then 8 KHz and will output a constant voltage of 8 V for a frequency of 8 KHz or above.

The LM2917 needs 10 V for a proper operation in this mode. In order to make the circuit as simple to the user as possible, we found the TC7660 DC-to-DC converter to be useful. It gets 5 V in pin 8 and following connection to pin 2 through a capacitor and two diodes, it gives $(2 \times 5\,\text{V} - 2 \times 0.7\,\text{V}) = 8.3\,\text{V}$ at the cathode of D2. This is less then 10 V but it works well for the LM2917. The analog voltage output from this circuit is fed to one of the microcontroller analog inputs that convert it to a binary value for further processing.

The output of the microphone circuit can also be converted to a digital binary bit that will indicate whether the input frequency is above or below a threshold one. It can be done by adding a simple comparator circuit, as shown in Fig. 10, and calibrating it to a desired frequency, so that when it reaches the desired frequency it will output the logic '1'. Below that frequency, it will output the logic '0'. Input is fed from Fig. 9's output. The new output is a digital one. As can be seen, LM339 is used for the comparator. Any other one will fit.

Digital encoders are used for a PID control feedback and for estimating the rotation degree. The robot has two driving/steering gear DC motors (Pittman) with built-in encoders (500 CPR), so that with the gear ratio of 1:19.2 we get 9600 encoder pulses per revolution. That is quite enough to take

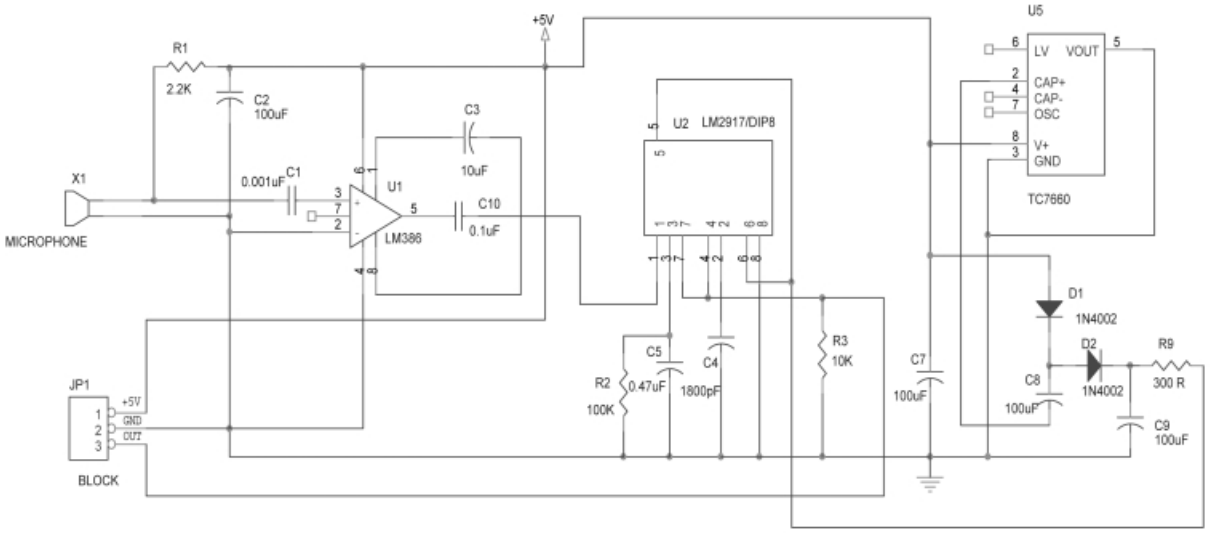


Fig. 9. Circuit of analog microphone sensor.

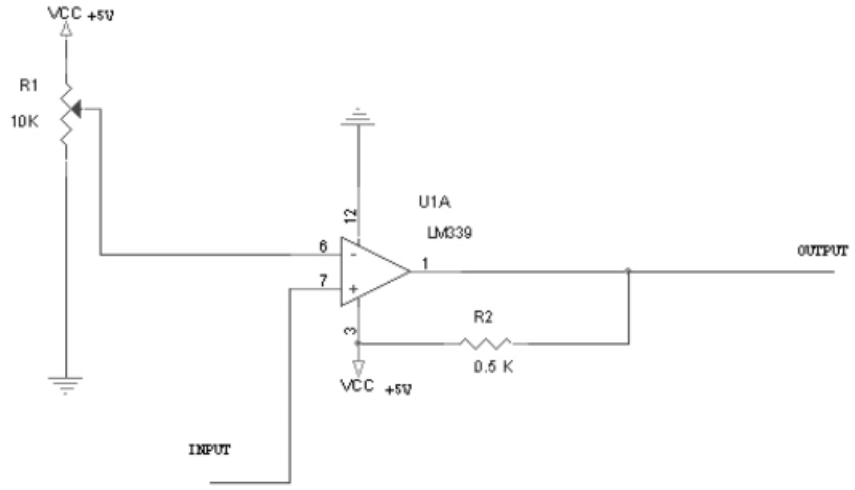


Fig. 10. Conversion of output signal (Fig. 9) to binary form.

care with a clock interrupt handler (every 8.125 ms), and to obtain a solid PID control. At 100 RPM, it will have 130 pulses every 8.125 ms. The encoders are built in the backside of the motors (see Fig. 11).

Robot actuators include:

1. Two driving/steering gear DC motors (see Fig. 11).
2. A fan DC motor with a propeller is intended for extinguishing the candle.

Robot drivers include:

1. Two h-bridge based DC motor drivers are used to control drive/steer motors, using the sign and magnitude PWM. These drivers handle the direct current up to 3 A continuous and transient current of up to 6 A. Control lines include (1) PWM for speed determination, (2) direction line, and (3) brake line (see Fig. 12).
2. The driver of the interface board connects all sensors and logic data lines to the main controller.
3. Drivers for voltage regulators (based on LM338), which are capable of handling up to 5 A, help distribute the power.
4. The main power board driver with protection was used to switch various sub-systems on and off.
5. Indication LED boards are placed for testing and debugging purposes.
6. A fan for the motor driver.

Fig. 11. Gear DC motor with encoder.

Control part of the robot

For implementing the proposed ControlWare, the main controller was used (see Fig. 13). It is based on the Motorola 69HC912b32 32K RAM chip and includes: a 8-MHz clock, eight analog to digital channels, four PWM channels, eight timer channels, and one pulse accumulator. There are more then 15 interrupt channels. The student assembly program resides in RAM. All the data from sensors is analyzed in this control unit, and then the appropriate micro-instructions are sent to the drivers.

Implementation of robot's ControlWare

The basic mission of the robot requires navigation through a corridor with four 'rooms', finding a lit candle in one of the rooms, approaching to a distance of up to 30 cm from the candle (marked by a white line in a radius of 30 cm around it), and extinguishing a flame. There are bonus points for more sophisticated modes, such as 'furniture' avoidance, non-dead-reckoning navigation, and more.

We present here one of the robot control algorithms, namely: non-dead-reckoning robot navigation in a corridor while aligning against the

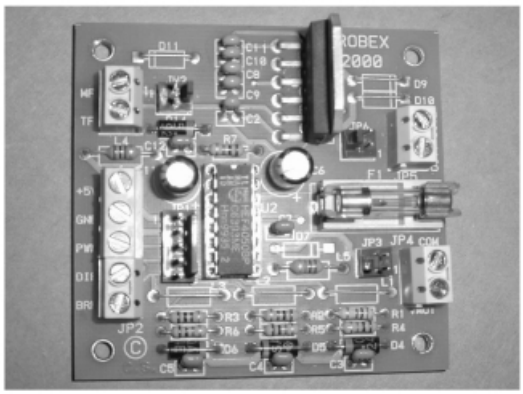


Fig. 12. H-bridge based DC motor driver.

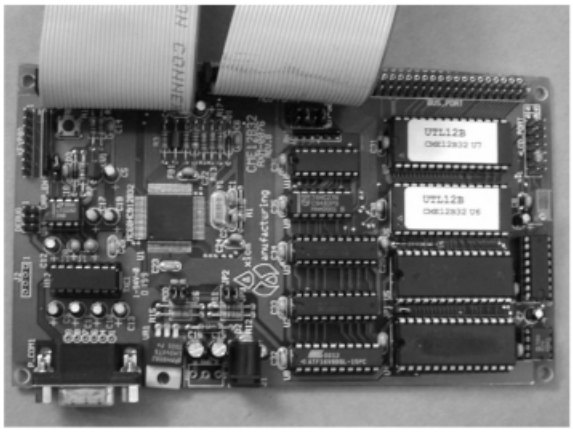


Fig. 13. The Motorola 69HC912b32 main controller.

right wall. The robot should stop when a wall appears in front of it. The relevant sensors for this task are right side and front analog distance sensors; the actions are performed by two driving/steering DC motors through their appropriate controllers.

For testing purposes, each micro-operation has its identifying LED that enables to follow the normal operation of the robot and visualizing its errors.

In this example, we use the logical variables $X = \{x_1, \ldots, x_6\}$ where:

x_1 is 'obstacle proximity has been detected in front of the robot' (which means that the analog voltage of the front sensor goes beyond 2 V);
x_2 is 'robot in center of corridor as seen by right sensors' (based on front or back right sensors' voltage of about 1.3 V);
x_3 is 'robot is close to right wall' (based on front and back right sensors' voltage of more than 2 V in at least one of them);
x_4 is 'direction of robot is less then 45 to the left from right side wall' (based on the positive difference between front and back right sensors' voltage of more than 0.9 V and less then 2 V);
x_5 is 'direction of robot is less then 45 to the right from left side wall' (based on the negative difference between front and back right sensors' voltage of more than 0.9 V and less then 2 V);
x_6 is 'robot is aligned to the right wall within $\pm 3^\circ$ (based on front and back right sensors' voltage of about 1.3 V each).

Micro-operations $Y = \{y_1, \ldots, y_{10}\}$ of the robot are the following:

y_1 is 'turning on right motor forward direction';
y_2 is: 'turning on right motor backward direction';
y_3 is: 'turning on left motor forward direction';
y_4 is: 'turning on left motor backward direction';
y_5 is: 'turning on led1';
y_6 is: 'turning on led6';
y_7 is: 'turning on led7';
y_8 is: 'turning off right motor';
y_9 is: 'turning off left motor';
y_{10} is: 'turning on led14'.

The above micro-operations form the set $F = \{Y_1, \ldots, Y_4\}$ of the following micro-instructions:

$Y_1 = \{y_1, y_3, y_5\}$ is 'move forward' (rotating two motors forward);
$Y_2 = \{y_1, y_4, y_7\}$ is 'pivot turn left' (rotating right

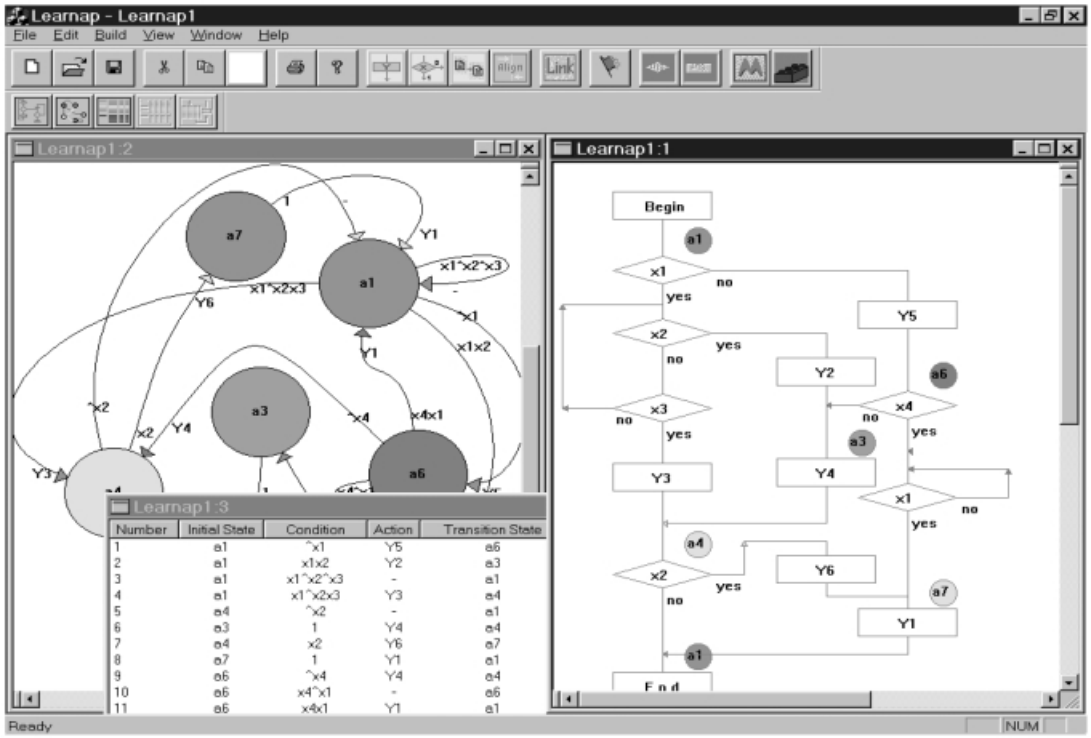


Fig. 14. The SMILE-environment screen shot.

motor forward while rotating left motor backward);

$Y_3 = \{y_2, y_3, y_6\}$ is 'pivot turn right' (rotating right motor backward while rotating left motor forward);

$Y_4 = \{y_8, y_9, y_{10}\}$ is 'stop' (stopping both motors).

The ASM corresponding to the above algorithm was shown in Fig. 1. The FSM implementing the ASM was presented in Table 1.

Both ASM and FSM descriptions of the robot's behavior can be considered as high-level descriptions of the corresponding system's behavior and according to [7] represent different approaches to designing control schemes. At the same time, these descriptions can be easily implemented by various hardware and software means. In the 'Robot Fire Fighting Project,' this control algorithm was implemented as a program of the above-mentioned microcontroller. The program for the microcontroller was produced by the specialized interactive learning environment SMILE that is discussed next.

INTERACTIVE LEARNING ENVIRONMENT

An interactive learning environment was developed in the framework of SMILE-project (State Machine Interactive Learning Environment) in the School of Education of the Tel-Aviv University [10]. The SMILE-environment implements the aforementioned ideas, i.e.:

- constitutes the ControlWare toolkit;
- supports both the programming and design paradigms of teaching control concepts; and
- is capable of controlling various kinds of real equipment.

SMILE comprises two editors with dedicated display windows shown in Fig. 14: the ASM editor (on the right) and the FSM editor (on the left). The environment enables students to design a control unit both in the form of a FSM by the FSM editor, and in the form of an ASM by the ASM editor. When the control unit is designed in one of these editors, it can be visualized simultaneously in the windows of both editors. Any change made in one of the editors is propagated immediately to the second editor.

Both state machine editors support a number of very powerful features for describing virtually any type of state machine. For example, a student can describe a state machine hierarchically, i.e. if it has a large number of states, it can be described in multiple levels. For debugging the state machine, two very important features are available. The first is an animated simulation where any step in the control process is depicted by a change in the color of the ASM (and/or FSM) symbolic states, which can be seen in the appropriate display windows. The second feature is that the process of control of the real equipment can also be visualized.

It should be noted that when the control unit is debugged, SMILE can generate a state machine file in various forms: software for a specific microcontroller; computer program; standard hardware description language (VHDL, Verilog), etc.

SMILE has not been fully tested with the fire fighting autonomous robot presented above. Nevertheless, several simulations have been made. ASM of the robot right wall navigation was written, and the software issued the corresponding FSM and state table. These allowed for better debugging, where missing states were easily discovered and corrected, some paths were found to be duplicated and canceled, making the code shorter. It took several minutes for a teacher to learn and start using the software with all its key features, therefore, we estimate that it would take students up to one hour to learn and use.

The duality of the representation and the transition from the algorithmic to the finite state machine, and vice versa, allows the students to solve a problem in the more convenient representation for them. Having such a friendly and powerful environment to design the control scheme at a high level is a big advantage since about half of the crucial failures of robots during contests are caused by failures of control of well-designed robots. In many cases, some states were ignored or misinterpreted. We believe that SMILE will improve significantly the students' understanding of the robot function and the control theory used to program the robot's behavior.

We base this believe also upon one experience in which one team, used FSM as the robot function representation and programmed it accordingly. They called it 'action-event' programming. It was the first time, that all possible situations could be easily discussed, and tested separately. When all the states were examined, most of the programming was over. Some of the team members, though, had a cognitive gap and encountered some difficulties to switch from ASM, which they were used to program with, to FSM programming.

For majority of the students, ASM is the 'natural' way to represent the robot function. Using SMILE will allow them to improve the robot control, better understand its functions, and more easily debug their software. FSM can be seen as a parallel activity, and thus improving students' parallel thinking, which has a significant value to students' problem-solving skills [11].

DISCUSSION AND CONCLUSIONS

We described an approach for designing and implementing a control part of a mechatronics system. This approach called 'the state machine-based design method' utilizes the concept of a state machine. Being presented in a form of the algorithmic state machine the specification of a mobile robot from the one hand becomes highly appropriate for regular software implementation, and

from the other hand, can be used as a ControlWare of the mobile robots.

The design method is supported by specific software, SMILE (State Machine Interactive Learning Environment), allowing the programming of a robot control part by using various representations of state machines.

We presented a specific application of the proposed approach by an example of a mobile robot built by high school students involved in a robot contest. This example shows a high level of transparency of the control algorithms developed by students. It demonstrates that the approach can lead to the significant improvement in the performance of the robot in real contest setting, thus filling an existing gap between design and actual performance of robots [12]. Given this consequence and its demonstrated ease of use, it can support the teaching and utilization of control concepts to high school and undergraduate students and has already become a standard part of high school mechatronics teaching.

Notice that the presented example of the 'fire fighting robot' application of the proposed approach demonstrates its relevance for high school students. At the same time, a similar approach was successfully used by the authors also for undergraduate students at the School of Engineering, Tel Aviv University and in the Computer Systems Department, Holon Academic Institute of Technology. In both of these institutions, the method was applied in mechatronics lessons.

Applicability of the approach both to high school and undergraduate students can be considered as an advantage. Indeed, the same methodological basis being introduced once in a high school can be extended in subsequent stages of technology education. Differences between the high school and the undergraduate lessons reflect different degrees of understanding of the theoretical basis of state machines and practical implementations thereof. Nevertheless, the main teaching method remains the same in both cases.

We believe that the proposed design methodology will permit the reduction of the gap between theoretical and practical components in learning mechatronics.

REFERENCES

1. F. Martin, A toolkit for learning: technology of the MIT LEGO Robot Design Competition, *Workshop on Mechatronics Education hosted at Stanford University* (1994). http://www.media.mit.edu/publications/.
2. M. Resnick, O. Stephen and S. Papert, LEGO, logo, and design, *Children's Environments Quarterly*, **5**(4), 1988.
3. P. Lewis, Introducing discrete-event control concept and state transaction methodology into control system curricula, *IEEE Trans. Education*, **37**(1) 1994, pp. 65–70.
4. A. R. Brooks, *A Robust Layered Control System for a Mobile Robot*, Massachusetts Institute of Technology, Artificial Intelligence Laboratory (1985).
5. I. Levin, V. Levit, ControlWare for learning with mobile robots, *Computer Science Education*, special issue: Robotics in Computer Science and Engineering Education, **8**(3) 1998, pp. 181–196.
6. S. Baranov, *Logic Synthesis for Control Automata*, Kluwer Academic Publisher, Dordrecht/Boston/London (1994).
7. I. Levin and D. Mioduser, A multiple-constructs framework for teaching control concepts, *IEEE Trans. Education*, **39**(4) 1996, pp. 488–496.
8. D. Mioduser and I. Levin, Cognitive-conceptual model for integration robotics and control into the curriculum, *Computer Science Education*, special issue: Robotics in Computer Science and Engineering Education, **7**(2) 1996, pp. 199–210.
9. J. L. Jones, B. A. Seiger and A. M. Flynn, *Mobile Robots, Inspiration to Implementation*, A. K. Peters, Massachusetts (1999).
10. I. Levin and E. Lieberman, Developing analytical and synthetic thinking in technology education, *Proc. Int. Conf. Technology Education*, Braunshweig, Germany, September 2000.
11. S. Waks, Lateral thinking and technology education, *J. Science Education and Technology*, **6**(4) 1997, pp. 245–255.
12. E. Kolberg, Y. Reich and I. Levin, Project-based high school mechatronics course, *Int. J. Eng. Educ.*, in press.

Eli Kolberg is a Ph.D. student in the Department of Solid Mechanics, Materials and Systems, Faculty of Engineering, Tel Aviv University, Israel. He received his B.Sc. in Mechanical Engineering from Tel Aviv University in 1980 and M.Sc. cum laude in Education in Technology and Science from the Technion in 2001. During 1980–1987 he practiced mechanical and aeronautical engineering in the Israeli Air Force. During 1987–1995 he practiced design and hardware engineering in the computers industry. During 1991–1994 he developed a robotics program for high school students and initiated the program in 1994. Since then, he has been involved in integrating successfully the robotics curriculum in high schools in Israel and abroad. He serves as a member in the Ministry of Education Robotics steering committee.

Yoram Reich is an associate professor in the Department of Solid Mechanics, Materials and Systems, Faculty of Engineering, Tel Aviv University, Israel. He received his B.Sc. (Summa Cum Laude) and M.Sc. (Magna Cum Laude) in Mechanical Engineering from Tel Aviv University in 1980 and 1984, respectively. Before obtaining the Ph.D. degree in Civil Engineering from Carnegie Mellon University, Pittsburgh, PA, in 1991, he practiced engineering design for over 7 years in the audio, structures, and marine industries. Dr. Reich has authored or co-authored over 100 papers and is a member of the editorial board of the journal *Advanced Engineering Informatics.* His research focuses on several inter-related topics: knowledge management; design methods, theories, philosophy, research methodology, and education; collaborative design; and knowledge discovery techniques for engineering applications.

Dr. Ilya Levin received the Ph.D. degree in Computer Engineering from the Latvian Academy of Science. During 1985–1990 he was the Head of the Computer Science Department in the Leningrad Institute of New Technologies (Russia). During 1993–1996 he was the Head of the Computer Systems Department of the Center for Technological Education, Holon (Israel). Being presently a faculty of the School of Education of Tel Aviv University, he is a supervisor of Engineering Education program. He is an author of more then 50 papers both in Design Automation and in Engineering Education fields.