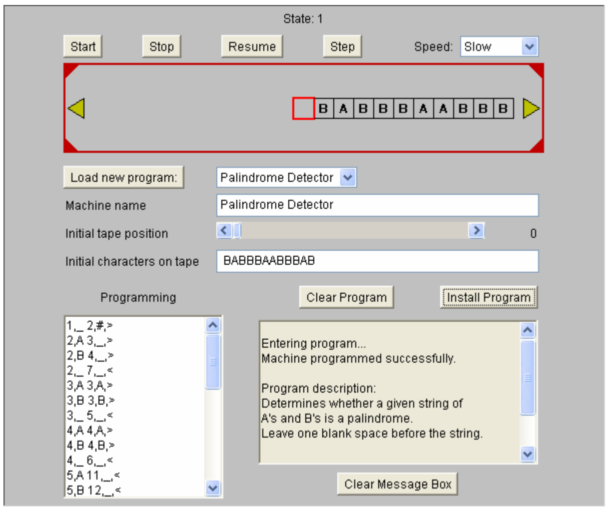
A Physical Implementation of the Turing Machine Accessed trough Web

doi:10.3991/ijoe.v4i4.688

Marijo Maracic and Slobodan Ribaric
University of Zagreb, Zagreb, Croatia

***Abstract*—A Turing machine has an important role in education in the field of computer science, as it is a milestone in courses related to automata theory, theory of computation and computer architecture. Its value is also recognized in the Computing Curricula proposed by the Association for Computing Machinery (ACM) and IEEE Computer Society. In this paper we present a physical implementation of the Turing machine accessed through Web. To enable remote access to the Turing machine, an implementation of the client-server architecture is built. The web interface is described in detail and illustrations of remote programming, initialization and the computation of the Turing machine are given. Advantages of such approach and expected benefits obtained by using remotely accessible physical implementation of the Turing machine as an educational tool in the teaching process are discussed.**

***Index Terms*— client-server architecture, e-learning, Turing machine, web interface.**



State: 1

Speed: Slow

Palindrome Detector

Machine name: Palindrome Detector

Initial tape position: 0

Initial characters on tape: BABBBAABBBAB

Programming

```
1,_ 2,#,>
2,A 3,_,>
2,B 4,_,>
2,_ 7,_,<
3,A 3,A,>
3,B 3,B,>
3,_ 5,_,<
4,A 4,A,>
4,B 4,B,>
4,_ 6,_,<
5,A 11,_,<
5,B 12,_,<
```

```
Entering program...
Machine programmed successfully.

Program description:
Determines whether a given string of
A's and B's is a palindrome.
Leave one blank space before the string.
```

Figure 1. Interface of a Turing machine program simulator [3]

I. INTRODUCTION

A Turing machine [1] has an important role in education in the field of computer science, as it is a milestone in the courses related to automata theory, theory of computation and computer architecture. Its value is also recognized in the Computing Curricula proposed by the Association for Computing Machinery (ACM) and IEEE Computer Society [2].

At present time the Turing machine is mainly implemented as a program simulator available on the Web [3-6]. Fig. 1 represents an example of the interface of a Turing machine program simulator [3]. The main characteristics of such program simulators are:

- they are widely accessible,
- they execute entirely on a computer,
- output of the machine is visually displayed,
- program is a character string whose semantics and syntax are often unintuitive,
- symbol set is very limited, usually restricted to alphanumeric symbols.

Based on our former experience [7], we believe that a physically implemented Turing machine supported by the web interface would add on the educational value of the Turing machine while retaining the positive characteristics of program simulators mentioned previously. The physical implementation of the Turing machine will enable students not only to observe the Turing machine computations but also to perceive the challenges of implementing a system in the physical environment.

The physical implementation of the Turing machine based on the yields of pattern recognition, computer vision and robotics will motivate students to study and research these areas of artificial intelligence in more detail.

It is very important for a successful educational tool to be easily accessible. The physical implementation is unique and static, i.e., tied to the laboratory so a client-server architecture was built for it to be remotely reachable from a classroom or home. The client is a Java applet available on the World Wide Web with a friendly user interface as its core component. The interface uses visual programming concepts that allow intuitive programming and a broader symbol set. The server was designed as a software module and built into the physical implementation of the Turing machine.

II. THE TURING MACHINE

A. *The formal model*

Formal structure of a Turing machine [1], [8] is shown in the Fig. 2. It consists of infinite tape, read/write (R/W) head, logical block L and internal memory cells S and P. An infinite tape, which represents the external memory, is divided into cells. Each cell contains one symbol from the tape alphabet Γ. At any time the R/W head is positioned over one particular cell that it is said to scan. The R/W head can alter the content of the scanned cell if so specified by the program of the Turing machine. The program is written as a set of transitions, defined by the transition function δ, and kept in the logical block L. Memory cells S and P are used to memorize the internal state of the machine and the command for the R/W head movement, respectively. During the computation process the Turing machine chooses a transition based on its internal state and the symbol in the scanned cell. According to the transition it replaces the symbol with a new symbol (the new symbol can be the same as the old one), changes the internal state of the machine and moves the R/W head.

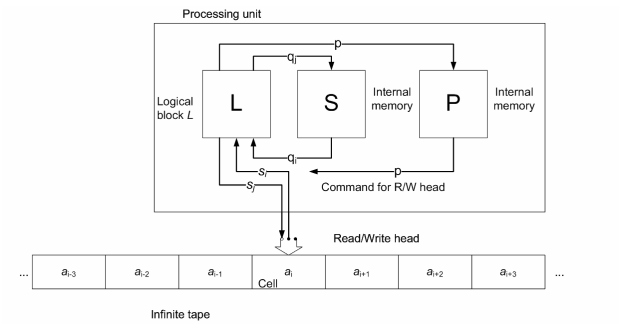


Figure 2. A Turing machine

Formally, Turing machine TM is defined as a 7-tuple [8]:

$$TM = (Q, \Sigma, \Gamma, \delta, B, q_0, q_f), \tag{1}$$

Where:

- Q is a finite set of states: $Q = \{q_0, q_1, \ldots, q_i, \ldots, q_f\}$, where q_0 denotes initial state and q_f denotes final and acceptable state of the machine.
- $\sum$ denotes a finite set of symbols consisting of symbols written on the tape at the beginning of the computation but excluding the "blank symbol" denoted with B. This set is also called input alphabet.
- Γ denotes a finite set of all the symbols that can appear on the tape of the machine during the computation. $\Sigma \subseteq \Gamma$, $B \in \Gamma$. This set is also called tape alphabet.
- δ denotes a transition function δ: $(Q, \Gamma) \rightarrow (Q, \Gamma, P)$, where P denotes a set consisting of valid moves of the R/W head, $P = \{L, R, H\}$, were R denotes movement of the R/W head one cell to the right, L denotes movement one cell to the left, and H denotes that the head remains at the same cell. The transition function δ is kept inside of the logical block L as a set of transitions. Transition function is often referred to as program.
- B denotes blank symbol that represents an empty cell.

Turing machine is a time-discrete machine that computes in cycles. In the first cycle the R/W head is positioned over the start cell. A finite array of cells starting with the start cell is filled with symbols from the input alphabet Σ. Content of that array of cells is called input string. During the first cycle, the machine is in the initial state q_0 and it scans the starting cell. Based on the scanned symbol $s_i \in \Sigma$ and the internal state of the machine $q_0 \in Q$, a transition is chosen from the transition function δ: $(q_0, s_i) \rightarrow (q_j, s_j, p)$ where $q_j \in Q$, $s_j \in \Gamma$, $p \in P$. Based on the chosen transition, symbol s_i is replaced with the symbol s_j, the machine changes its internal state into q_j and the R/W head takes action described by p. After executing the chosen transition, a current cycle ends and a new cycle begins. During the subsequent cycles, the process of choosing and executing transitions is analogue to the above-described procedure. If there is more than one possible transition for some input pair (q_i, s_i) the Turing machine is non-deterministic.

The Turing machine refines the input string into the output through series of cycles. The machine stops in the cycle where the transition for the current input pair (q_i, s_i) is not defined. If at that moment the internal state of the machine is an acceptable state q_f, the machine has computed desired output. The output is written on the tape in the form of the finite symbol sequence from Γ with length $n < \infty$.

If the internal state is not acceptable when the machine stops then the machine failed to compute the desired result and content of the tape is not relevant.

It is possible that the machine cannot compute the desired output in the finite number of steps (the machine never stops).

During each cycle the Turing machine is described with a configuration. A k-configuration is defined at the beginning of the k-th cycle and it consists of:

- content of the tape,
- internal state of the machine,
- position of the R/W head.

B. *The physical implementation*

Formal model of the Turing machine had to be modified as a prerequisite for physical implementation:

- Formal symbols, as elements of tape alphabet, are replaced by plates with symbol images (colored geometrical shapes).
- Pool is introduced to the model. The pool is an area used by the robot arm to store the spare plates.
- The movement of the R/W head has been replaced with the equivalent conveyer movement that is simpler to implement.

The physical implementation consists of hardware components and software modules. Fig. 3 depicts the main hardware components.

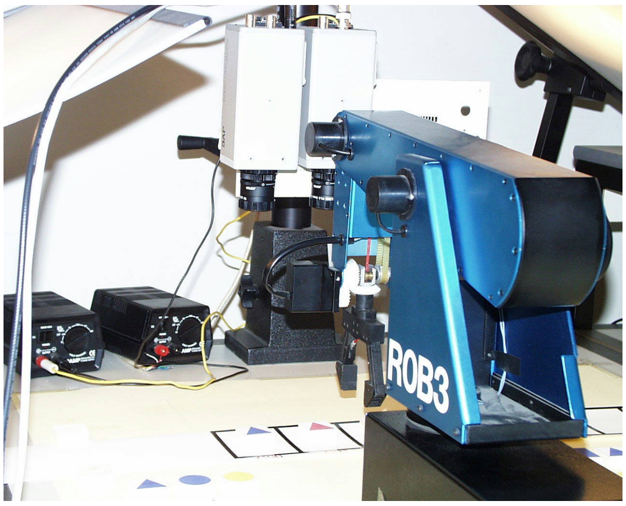


Figure 3. A detail of the physical implementation of the Turing machine

The main hardware components are:

- A personal computer is the most important hardware component of the implementation. It executes software modules that simulate the Turing machine, communicates with all other physical components and interacts with the user.
- A conveyer is a physical implementation of the tape of the Turing machine. It is a band segmented into rectangles that represent cells. Motion of the conveyer is achieved using two synchronized step-motors.
- A tape camera is mounted above the conveyer and its optical axis is orthogonal on the conveyer plane. Three cells of the conveyer are in the field of view of the camera: the scanned cell and its nearest neighbors to the left and to the right. This camera performs as a Read head of the Turing machine.
- The robot arm performs as a Write head of the Turing machine and manipulates the plates contained within the conveyer cells and in the pool. The robot arm, together with the tape camera, forms the R/W head.
- A pool camera monitors the content of the pool.
- A panoramic camera oversees the whole Turing machine. It provides live video feedback about interaction of physical components of the Turing machine to the client.

The software modules are:

- Performer module: Performs as the modified formal model of the Turing machine. It communicates with the control modules and the user interface. It transmits commands to the hardware components and sends Turing machine status to the user interface. It receives information about symbols present on the conveyer and in the pool.
- Control modules: They map commands received from the performer module to the hardware specific protocol. Robot arm, conveyer and cameras each have their own control module.
- Image analysis module: Processes the images provided by the tape camera and the pool camera to identify the symbols present on the conveyer and in the pool. The processing procedure includes image preprocessing, image segmentation and classification of objects i.e. symbols. It also provides information about position and orientation of the plates with symbol images.

III. CLIENT – SERVER ARCHITECTURE

An important measure of success of any product is the number of its users. The Turing machine implementation described above consists of several hardware components. These components are immobile, sophisticated and expensive so it is unlikely that this implementation would be widely used regardless of its potential educational value. Furthermore, in direct contact with the user, it is likely that some of the components would be damaged due to improper usage. To make the implementation widely accessible while still kept in the controlled and safe environment we have built in a client-server architecture allowing remote access to the physical implementation of the Turing machine.

The server is a software module built into the Turing machine implementation. It receives the initialization information for the Turing machine from the client. The server encapsulates and starts the computation of the Turing machine. It relays communication between the Turing machine and the client. The server transmits the output of the Turing machine, represented by the configuration, as well as live overview from the panoramic camera to the client. The server also relays client signals to the machine. Turing machine protocol - a specially designed protocol at the application level, is used for the communication between the client and the server.

The client provides a graphical user interface to the Turing machine. Through this interface the initialization data for the Turing machine can be defined and uploaded to the server. The client is written in Java and published in the World Wide Web (http://www.pattern-recognition.zemris.fer.hr/physicalTM/). The client interface uses visual programming paradigm to simplify the programming of the Turing machine and the analysis of its output.

IV. WEB INTERFACE

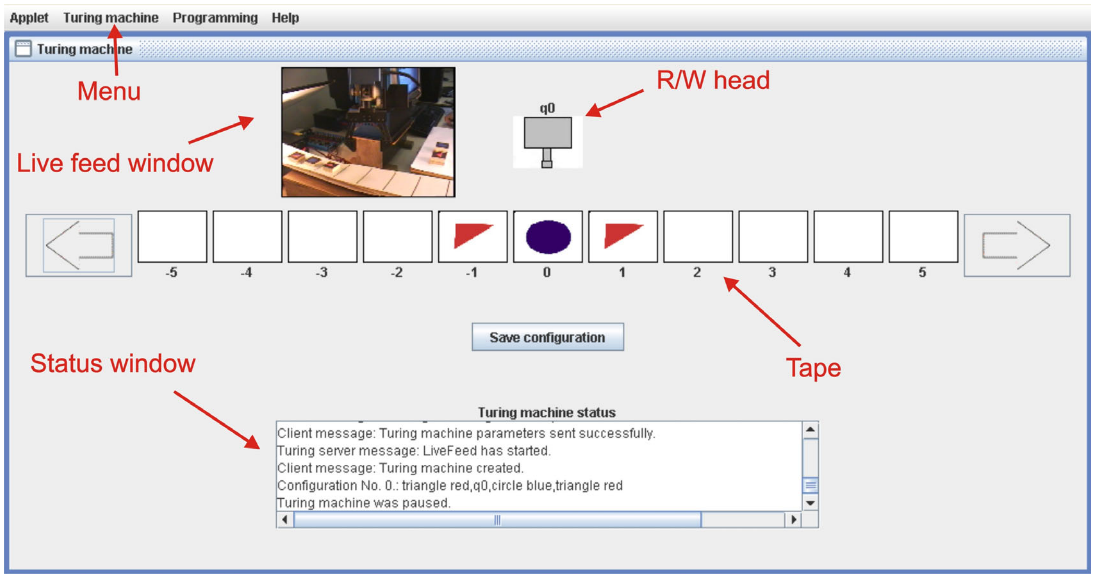
A. *Visual programming paradigm*

Web interface is based on the visual programming paradigm. Image of the symbol is used within the interface to program the Turing machine and display the results of computations. Internally, the physical implementation uses textual representation of the symbol because it consumes less storage space.

Since the internally used representation of symbol is textual, a simple hybrid visual programming language is used [9]. All elements of the visual programming language are present although very simple due to nature of the Turing machine itself. The tape alphabet is a dictionary of symbols (i.e. icons) represented by images and logical, textual description. The transition function δ represents the grammar. Only horizontal concatenation of symbols is allowed. Each configuration of the Turing machine is an iconic sentence. There is no formal domain-specific knowledge base because a Turing machine is a general-purpose machine that can solve broad class of problems when given enough time and space. The user who proposed the problem is expected to have enough domain-specific knowledge about the problem to interpret the result.

The web interface is shown in the Fig. 4. The interface consists of following components:

- menu allowing programming and running of the Turing machine,
- live feed window,
- representation of the Turing machine R/W head,
- representation of the Turing machine tape,
- status window.



Client message: Turing machine parameters sent successfully.
Turing server message: LiveFeed has started.
Client message: Turing machine created.
Configuration No. 0.: triangle red,q0,circle blue,triangle red
Turing machine was paused.

Figure 4. The web interface

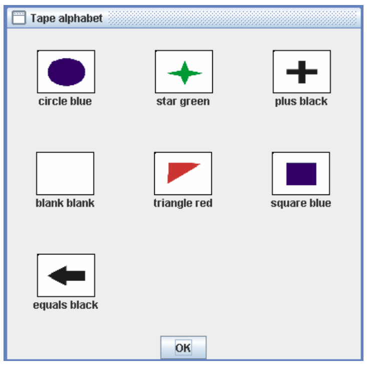
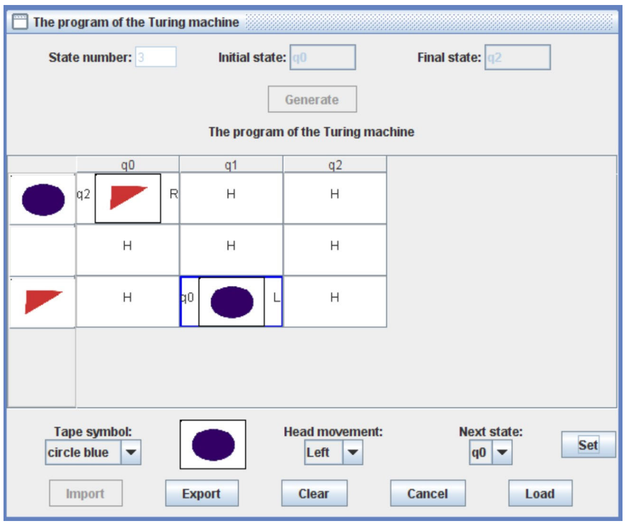


Figure 5. Available tape alphabet


The program of the Turing machine
Generate
The program of the Turing machine
Set
Import
Export
Clear
Cancel
Load

State number: 3
Initial state: q0
Final state: q2

	q0	q1	q2
●	q2 🟥 R	H	H
	H	H	H
🟥	H	q0 ● L	H

Tape symbol: circle blue
Head movement: Left
Next state: q0

Figure 6. An example of filled program form with triplets

B. Programming the machine remotely

User needs to provide the following information to start the computation of the Turing machine: tape alphabet Γ, state set Q, transition function δ and input string. This information formally and completely describes the problem of interest. A user, accessing the physical implementation of the Turing machine through the client interface, cannot use arbitrary tape alphabet. The tape alphabet that the remote user can use is sent to the client in advance and it depends on the plates with symbol images present in the pool. The user can preview the available tape alphabet using the form accessible through the menu Programming →Alphabet. An example of the form is shown in the Fig. 5.

The user has to specify the rest of the data using the web interface: state set Q, transition function δ, and input string. The first two can be defined using the form accessible through the menu Programming→Program. The form is shown in the Fig. 6.

The user is required to specify number of states contained in the state set Q, initial state and final state, respectively. After pressing the "Generate" button an empty table, designed for the specification of the Turing machine program, is generated.

The first column of the table represents scanned symbols, while the first row represents internal states of the Turing machine. The entry points of the table contain triplets (new state of the machine, new cell symbol, command for R/W head movement) which describe actions to be taken during execution of the Turing machine when the correspondent scanned symbol-state pair is encountered. The task of the user is to enter the triplets that describe the procedure (i.e. transition function δ) that the Turing machine should follow while computing an assignment. The parameters of a triplet are selected in the comboBoxes below the table and written in the table when the button "Set" is pressed. The input of the triplet is illustrated by the Fig. 6. The "Load" button uploads the state set and the program into the Turing machine that is about to be executed on the server side.

An input string can be defined using the form accessible through the menu Programming →Input string. The form is shown in the Fig. 8.

Within this form a user can also choose whether to run a program simulator (by choosing the radio button “Virtual tape”) or the physical implementation of the Turing machine (by choosing the radio button “Physical tape”). In the simulation mode, the user can use unlimited number of symbols while in the physical implementation mode the user can use only the symbols present in the pool. The Fig. 8 depicts an example of the initial string.

C. *An execution example*

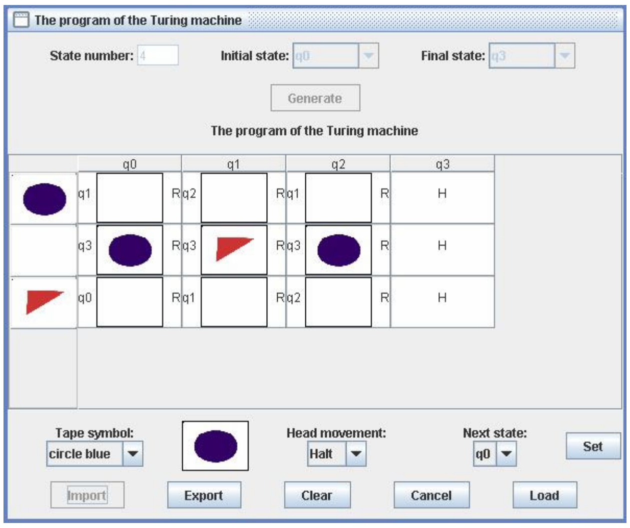
After the user provides the Turing machine parameters to the server through the client, the machine is ready to start. The machine can be run in systematic (i.e. step-by-step) mode, where the machine pauses after each cycle and waits for user confirmation to continue, or continuous mode where the machine continuously runs until it reaches the end of the computation.

The work of the physical implementation of the Turing machine connected with the web interface will be demonstrated by the following simple example:

The Turing machine has to determine if there is odd or even number of blue circles in the input string.

The state set Q and the program used to solve the described problem are shown in the Fig. 7.

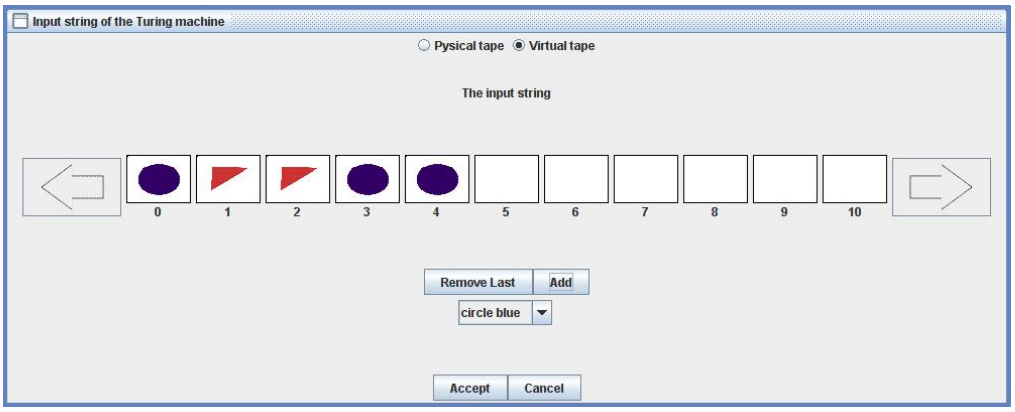
The solution utilizes the following idea: The Turing machine will have four internal states: initial state q_0, state q_1 symbolizing that odd number of blue circles has been scanned in the input string so far, state q_2, symbolizing that even number of blue circles has been scanned in the input string so far and the final and acceptable state q_3. The conveyer moves to the left in each cycle independently of the input pair (q_i, s_j) and the R/W head progresses through the input string, writing the blank and changing the internal state of the machine according to the program (Fig. 7). After the input string has been scanned and deleted, the R/W head places a blue circle in the cell if the machine is in the state q_2 (denotes that there was even number of blue circles in the input string) or it places a red triangle if the machine is in the state q_1 (denotes that there was odd number of blue circles in the input string). The machine then changes its state to the final state q_3, moves the R/W head to the right and stops.



	q0	q1	q2	q3
●	q1, , R	q2, , R	q1, , R	H
	q3, ●, R	q3, ◤, R	q3, ●, R	H
◤	q0, , R	q1, , R	q2, , R	H

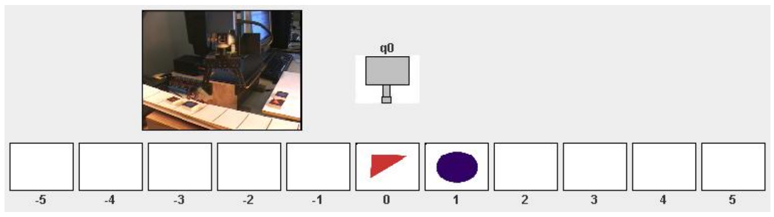
Figure 7. Program used to solve the problem. Note that the triplets specify R/W head movement while the conveyer moves in the opposite direction.

Fig. 9 shows an example of computation of physical implementation of the Turing machine while solving the problem as explained above. The initial string at the beginning of the computation (1st configuration) for this particular example consists of one red triangle and one blue circle as shown in the Fig. 9(a). The initial internal state of the machine is q_0. The R/W head scans the start cell marked with the index 0. The computation progresses through several cycles depicted with the Fig. 9(b-c). At the end of the computation (depicted with the Fig. 9(d)) the Turing machine scans the cell indicated with the index 3, the internal state of the machine is final and acceptable meaning that the red triangle on the conveyer is the solution of the problem.

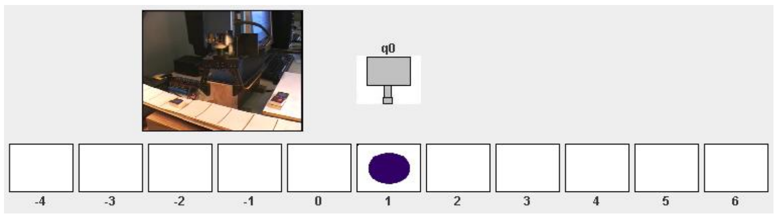


0	1	2	3	4	5	6	7	8	9	10
●	◤	◤	●	●						

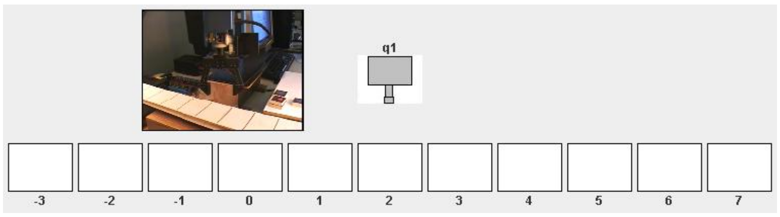
Figure 8. An example of specified input string



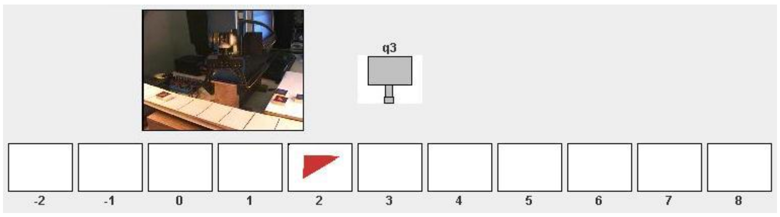
a)



b)



c)



d)

Figure 9. a) 1st configuration of the Turing machine b) 2nd configuration c) 3rd configuration d) 4th and final configuration, the solution of the problem is displayed on the conveyer of the machine.

V. CONCLUSION

In this paper we have described an attempt to broaden educational significance of the Turing machine. We achieved the goal by:

- designing the physical implementation of the Turing machine,
- interweaving the Turing machine implementation with the fields of artificial intelligence and real-time system design,
- introducing graphical programming language,
- preserving positive feature of the existing program simulators of the Turing machine available on the web.

The physical implementation has two main positive effects on the educational value of the Turing machine. First is that the physical implementation includes yields of pattern recognition, computer vision and robotics. It is a demonstration of techniques and algorithms from these areas allowing students to get an early glimpse on those areas. There is no reason why this implementation could not be used even in the secondary education. This could popularize mentioned areas, motivating pupils and students to pursue further education or career in these areas. The second positive effect is the experience obtained from the design process of system in the real physical environment. Parameters of such environment are constantly varying and the system has to be carefully designed and fault-tolerant to function within parameters in such environment. The limitation of the physical implementation is the fact that hardware components can perform only one task at any given moment, meaning that only one user can use the Turing machine at the same time although it is reachable by many potential users.

The graphical user interface that incorporates visual programming language paradigm eases the programming and analysis of the Turing machine. Syntax of visual programming language is more intuitive and therefore easier for students to use.

The main positive feature of program simulators is availability; they are available to the vast number of users at minimal or no cost. Availability, in our approach, was preserved by building in a client-server architecture that made the physical implementation of the Turing machine available on the World Wide Web through a client user interface.

Additional feature brought in by introducing the client-server architecture was the detachment of the user from the implementation hardware to prevent possible damage caused by improper use. A problem that can arise from making the implementation available on the Internet is the problem of network security.

Future work includes evaluation and further improvements of the implementation. The students of our faculty will initially test the Turing machine implementation, but we will be expanding the tests if the initial feedback and results show that our predictions about educational value were true. We consider number of overall users and average usage rate important indicators about the value of our implementation.

It is necessary to improve the hardware components: The robot arm has very constrained area of reach so we plan to replace it with a better model. A framework for the components will have to be built ensuring stable positioning. The implementation has to be tested on the broader set of symbols and problems.

We would like to stress that the physical implementation of the Turing machine includes all initially designed features and we hope to improve it to the benefit of all potential users.

REFERENCES

[1] Turing, A. M.: On computable numbers with an application to the Entscheidungsproblem, *Proceedings of the London Mathematical Society, 2nd Series, 42., pp. 230-265, 1936.*

[2] The Joint Task Force on Computing Curricula - IEEE Computer Society and Association for Computing Machinery: Computing Curricula 2001 for Computer Science, Final Report, December 15, 2001., http://www.acm.org/education/education/education/curric_vols/cc2001.pdf

[3] Britton, S.: Turing machine Simulator, http://www.ironphoenix.org/tril/tm/

[4] Schweller, K. G.: Turing Machine, http://web.bvu.edu/faculty/schweller/Turing/Turing.html

[5] Vinokur, A.: Turing and Post Machines: C++ Simulators, http://sourceforge.net/projects/turing-machine/

[6] Hodges, A.: Alan Turing Internet Scrapbook, http://www.turing.org.uk/turing/scrapbook/tmjava.html

[7] Ribarić, S., Krleža, D., Pavešić, N.: A Turing machine with Robot Arm and Eye, *Proceedings of the 5th IEEE Conference on Intelligent Engineering Systems, INES 2001, Helsinki pp. 273-276*

[8] Atallah, M. J.: Algorithms and Theory of Computation Handbook, *CRC Press, 1998., New York*

[9] Boshernitsan, M., Downes, M.: Visual Programming Languages: A Survey, Report No.UCB/CSD-04-1368, December 2004, Computer Science Division (EECS), University of California, Berkeley, California 94720, http://nitsan.org/~maratb/pubs/csd-04-1368.pdf

AUTHORS

Marijo Maracic is with the Faculty of electrical engineering and computing, University of Zagreb. Marijo Maracic earned a B. Sc. degree in computer science from the Faculty of electrical engineering and computing, University of Zagreb, Croatia in 2007. His research interests include computer vision and pattern recognition. He is a graduate student member of IEEE.

Slobodan Ribarić received the B.Sc. degree in electronics, the M.Sc. degree in automatics, and the PhD. degree in electrical engineering from the Faculty of Electrical Engineering, Ljubljana, Slovenia, in 1974, 1976, and 1982, respectively. He is currently a Full Professor at the Department of Electronics, Microelectronics, Computer and Intelligent Systems, Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia. His research interests include Pattern Recognition, Artificial Intelligence, Biometrics, Computer Architecture and Robot Vision. He has published more than one hundred and fifty papers on these topics, and he is author of four books (Microprocessor Architecture, The Fifth Computer Generation Architecture, Advanced Microprocessor Architectures, CISC and RISC Computer Architecture) and co-author of one (An Introduction to Pattern Recognition). Dr. Ribarić is a member of the IEEE, ISAI and IAPR.

This article was modified from a presentation at the International Conference on Computer Aided Learning ICL2008 in Villach, Austria, Sptember 2008. Manuscript received 10 October 2008. Published as submitted by the authors.