Design Hardware Simulation Using an Embedded System Find Shortest-Path Based on Dijkstra's Algorithm

Zina Abdul Lateef **Mohammed Ali Obaid**
communications Engineer
Musayyib Technical Institute

Abstract

In computer science, graphs are used to represent networks of communication, data organization, computational devices, the flow of computation, etc. One practical example: The link structure of a website could be represented by a directed graph. The vertices are the web pages available at the website and a directed edge from page A to page B exists if and only if A contains a link to B. A similar approach can be taken to problems in travel, biology, computer chip design, and many other fields. The proposal design is a simulation of one algorithm in graph theory to find shortest path called Dijkstra algorithm. The implementation of routing algorithm in hardware design has return practical solution benefit to find the shortest path in many fields (communication, search shortest road, and engineering design, etc.)

The hardware simulation process start by specify the source and destination of the topology consist of four nodes, and specify length between each connection. Then, microcontroller using in embedded system is ATML89c51 will start solving the problem of Dijkstra, and find the shortest path.

The experimental results show process of the hardware and the Dijkstra's algorithm, and the results shows on LCD implemented with microcontroller.

Keywords: Graph theory, Dijkstra's algorithm, ATML89c51, LCD.

المستخلص

في علوم الحاسوب وخصوصاً في كرافك المستخدم في تمثيل الاتصال الشبكي، لتنظيم البيانات، وحساب التدفق لهذه البيانات وغيرها من المسائل الخاصة في تصميم الشبكات، فان ايجاد الطرق المختصرة بين المصدر والهدف تعتبر من المسائل المهمة في مجالات متعددة. ومثال على ذلك الاتصال بين صفحات الوب كتوجيه الاتصال بين الصفحة (أ) وصفحة (ب) اذا كان الاتصال موجود بينهما. وهذه العملية مشابهة في حقول اخرى مثل البايلوجي، الترحال والسفر، وتصاميم الخرائط المايكروية وغيرها من الحقول الاخرى.

الغرض من البحث هو تصميم محكاة لاحدى خوارزمية التوجيه وهي (ديجكسترا) لايجاد اقرب طرق للوصول وتحديد عقدة الهدف باستخدام مايكروكونترولر. بالرغم من اغلب الدراسات والبحوث في هذا المجال وظفت هذه الخورازميات بشكل برامج حاسوبية. ويعتبر توظيف المايكروكونترولر في تنفيذ خوارزميات التوجيه ذا فائدة عملية في ايجاد الطرق المختصرة في كثير من المجالات (الاتصالات، البحث عن الطرق المختصرة، تصاميم الهندسية وغيرها).

عمليات الجارية اثناء عمل التصميم المقترح تبدأ بتحديد المصدر والهدف للتصميم التبولوجي والذي يحتوي على اربعة عقد، وكذلك يحتاج الى تحديد المسافة لكل ربط بين العقد. وبالتالي يقوم المعالج المستخدم (ATML89c51) بحل خوارزمية الدايجكسترا وذلك بايجاد اقرب مسار بين المصدر والهدف.

النتائج المستحصلة في هذا البحث تبين مراحل عمل الخوارزمية على شاشة LCD والمرتبطة مع المعالج في الدائرة الالكترونية.

1. Introduction

In recent years, graph theory has established itself as an important mathematical tool in a wide variety of subjects, ranging from operational research and chemistry to genetics and linguistics, and from electrical engineering and geography to sociology and architecture. At the same time it has also emerged as a worthwhile mathematical discipline in its own right.

In view of this, there is a need for a hardware simulation design on the subject, suitable for mathematicians taking courses in graph theory and also for nonspecialists wishing to learn the subject as quickly as possible, beside for networking study.

To implement kind of this algorithms in hardware, need to used a microcontroller as an important part to solving these issues.

2. Graph Theory

In the network communication can implement some of useful algorithm of graph theory. One of these algorithms used to find shortest path among nodes; nodes determining the clients connected somehow in topology with assigned weights (lengths) of each connection.

By characterizing the structure of an optimal solution, for the all-pairs shortest-paths problem on a graph G D .V;E, have to proven that all subpaths of a shortest path are shortest paths. Suppose that represent the graph by an adjacency matrix $W = (w_{i,j})$. Consider a shortest path *p* from vertex *i* to vertex *j*, and suppose that *p* contains at most m edges. Assuming that there are no negative-weight cycles, *m* is finite. If *i=j*, then *p* has weight *0* and no edges [1].

If vertices *i* and *j* are distinct, then decompose path *p* into $i \overset{p'}{\rightarrow} k \rightarrow j$, where path *p'* now contains at most *m-1* edges.

Now, let $l_{i,j}^{(m)}$ be the minimum weight of any path from vertex *i* to vertex *j* that contains at most m edges. When *m=0*, there is a shortest path from *i* to *j* with no edges if and only if *i=j*. Thus,

$$l_{i,j}^{(m)} = \begin{cases} 0 & if\ i = j \\ \infty & if\ i \neq j \end{cases} \ldots\ldots\ldots\ldots\ldots\ldots (1)$$

For $m \geq 1$, compute $l_{i,j}^{(m)}$ as the minimum of $l_{i,j}^{(m-1)}$ (the weight of a shortest path from *i* to *j* consisting of at most *m-1* edges) and the minimum weight of any path from *i* to *j* consisting of at most m edges, obtained by looking at all possible predecessors *k* of *j*. Thus, recursively define[2]

$$l_{i,j}^{(m)} = min\left(l_{i,j}^{m-1)}, \min_{1k \leq n}\left\{l_{i,k}^{m-1)} + w_{k,j}\right\}\right) \ldots\ldots\ldots (2)$$
$$= \min_{1k \leq n}\left\{l_{i,k}^{m-1)} + w_{k,j}\right\}$$

The latter equality follows since $w_{i,j} = 0$ for all *j*.

If the graph contains no negative-weight cycles, then for every pair of vertices *i* and *j* for which $\delta(i,j) < \infty$, there is a shortest path from *i* to *j* that is simple and thus contains at most *n-1*edges. A path from vertex *i* to vertex *j* with more than *n-1*edges cannot have lower weight than a shortest path from *i* to *j* . The actual shortest-path weights are therefore given by

$$\delta(i,j) = l_{i,j}^{(n-1)} = l_{i,j}^{(n)} = l_{i,j}^{(n+1)} = \cdots \qquad \ldots\ldots\ldots (3)$$

3. Process of Dijkstra Algorithm

In general the shortest-path problems are concerned with finding shortest paths between vertices. Many interesting problems arise, and the variety depends on the type of graph in application and the exact question wants to answer. Some of the characteristics which may help in defining the exact problem are as follows [3]:

1. The graph is finite or infinite.
2. The graph is undirected or directed.
3. The edges are all of length 1, or all lengths are non-negative, or negative lengths are allowed.
4. May be interested in shortest paths from a given vertex to another, or from a given vertex to all the other vertices, or from each vertex to all the other vertices.

5. May be interested in finding just one path, or all paths, or counting the number of shortest paths.

First, by consider the case of a finite graph *G* in which two vertices *s* and *t* are specified. The task is to find a path from *s* to *t*, if there are any, which uses the least number of edges. Clearly this is the case of the finite, undirected graph, with all length of edges being equal to 1, and where all want is one path from *a* given vertex to another. In fact, the digraph case is just as easy and can be similarly solved.

4. Sequence to Solve Dijkstra's Algorithm

One of shortest path algorithm in graph theory is Dijkstra's algorithm, and it's described as following:

1. $\lambda(s) \leftarrow 0$ and for all $v \neq s$, $\lambda(v) \leftarrow \infty$.
2. $T \leftarrow V$.
3. Let *u* be a vertex in *T* for which $\lambda(u)$ is minimum.
4. If $u = t$, stop.
5. For every edge $u \rightarrow v$, if $v \in T$ and $\lambda(v) > \lambda(u) + l(e)$ then $\lambda(v) \leftarrow \lambda(u) + l(e)$.
6. T-T-{u} and go to step 3.

The denote distance of vertex *v* from *s* by $\delta(v)$. And want to show that upon termination $\delta(t) = \lambda(t)$; that is, if $\lambda(t)$ is finite than it is equal to $\delta(t)$ and if $\lambda(t)$ is infinite then there is no path from *s* to *t* in the digraph [4].

Dijkstra's algorithm can be carried out more easily if keep track of the necessary information on tables. This illustrate this graph constructing an optimal A-rooted spanning tree in this graph as shown in figure (4)

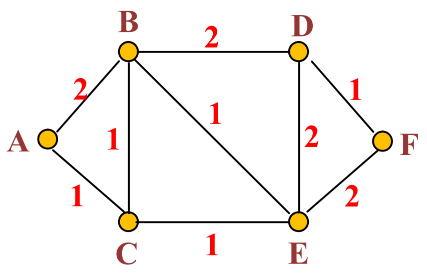

Figure (4) Example of Dijkstra's algorithm Topology

Clearly the first edge is AC. The first tree, with vertex labels, is this one shown in figure (5) below:

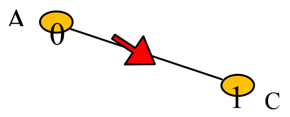

Figure (5) "AC" check path

to determine the next edge, found in below constructed table 3.1

Table 1. "AC" edges

		B	D	E	F
A	0	2			
C	1	1		1	

In which the first column shows the vertex labels at vertices already reached, and the remaining columns show the weights of all edges that are candidates for inclusion on the next step. By adding each vertex label to all weights in the same row as in table 2, and obtain the values that have to be compared in order to determine which edge to add.

Table 2. "AC" adding vertex label

	B	D	E	F
A	2			
C	2		2	

This shows that any one of the edges "AB", "CB", or "CE" can be added on the next step as shown in figure (6). Arbitrarily selecting AB, and obtain this labeled tree and the corresponding table as in table 3:

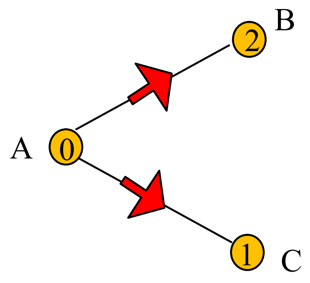

Figure (6) "AB" Path

The table below will determine the edged arrived or reached for both nodes "B" and "C".

Table 3. "ABC" edges

		D	E	F
A	0			
B	2	2	1	
C	1		1	

And finally the "B" and "C" will shortest one distance is from "C" to "E" will be labeled and as shown in below table 4.

Table 4. "ABC" vertex label

	D	E	F
A			
B	4	3	
C		2	

This time the only choice is "CE", corresponding to the minimum value on the last table. The remaining steps of the algorithm are shown below, with the circled entry indicating the selected edge and the new vertex label on each step.

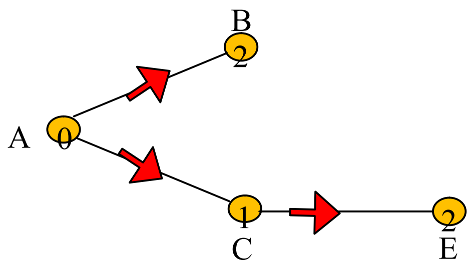

Figure (7) "CE" path

The corresponding table of indicating for path "ACE" is constructed in table below.

Table 5. "ACE" edges

		D	F
A	0		
B	2	2	1
C	1		1
E	2	2	2

And the "ACE" will be the shortest one distance from "B" to "D" will be labeled and as shown in below table 6.

Table 6. "ABCE" vertex label

	D	F
A		
B	4	
C		
E	4	4

As constructed in table above, the selected node from "B" is "D", and be shown in figure (3.6) as below.

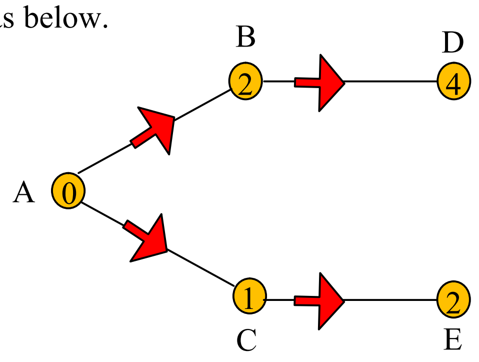

Figure (8) "BD" path

Table 7. "ABCDE" edges

		F
A	0	
B	2	
C	1	
D	4	1
E	2	2

Table 8. ABD vertex label

	F
A	
B	
C	
D	5
E	4

Finally, the shortest path from node "A" to "F" is get through "C" and then "E" and will be finally "A, C, D, and F" as shown below.

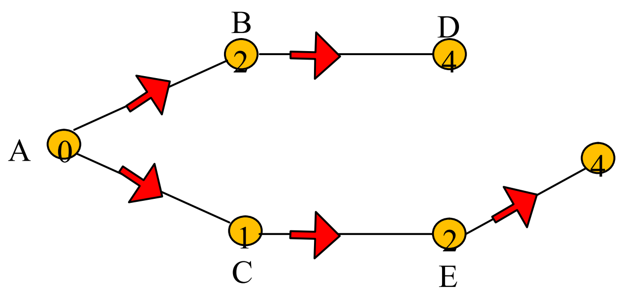

Figure (9) "EF" path

4. Experimental Results

As shown in figure (4)., the schematic of the controller that proposed in this paper to solve Dijkstra algorithm, and this design used a microcontroller 8051 family (AT89c51) with LCD (16*2) to display the message[5].

A good language program used called "Bascom 8051" as a basic language to simulate a family of ATML 8051 [5].

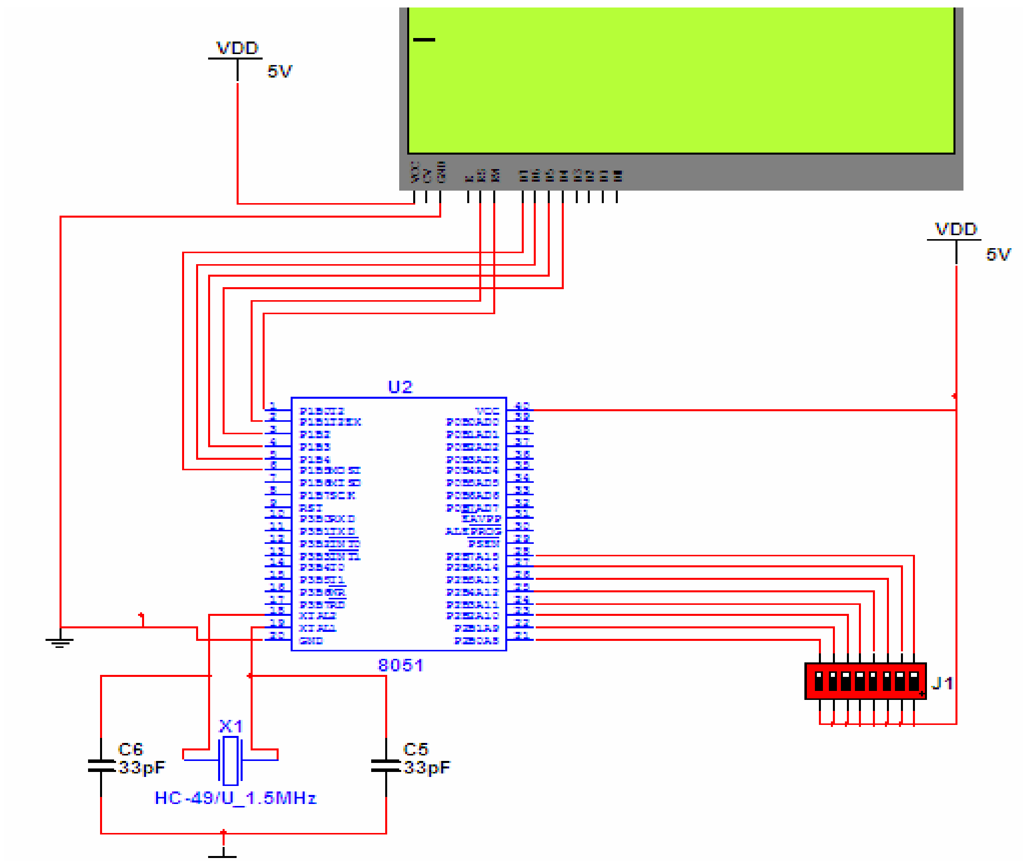

Figure (10) Design Schematic of Proposed work

The messages displayed on LCD are determining the sequence of process of the Dijkstra algorithm.

The figure (11) shows the flow chart of Dijkstra's algorithm which implemented in microcontroller

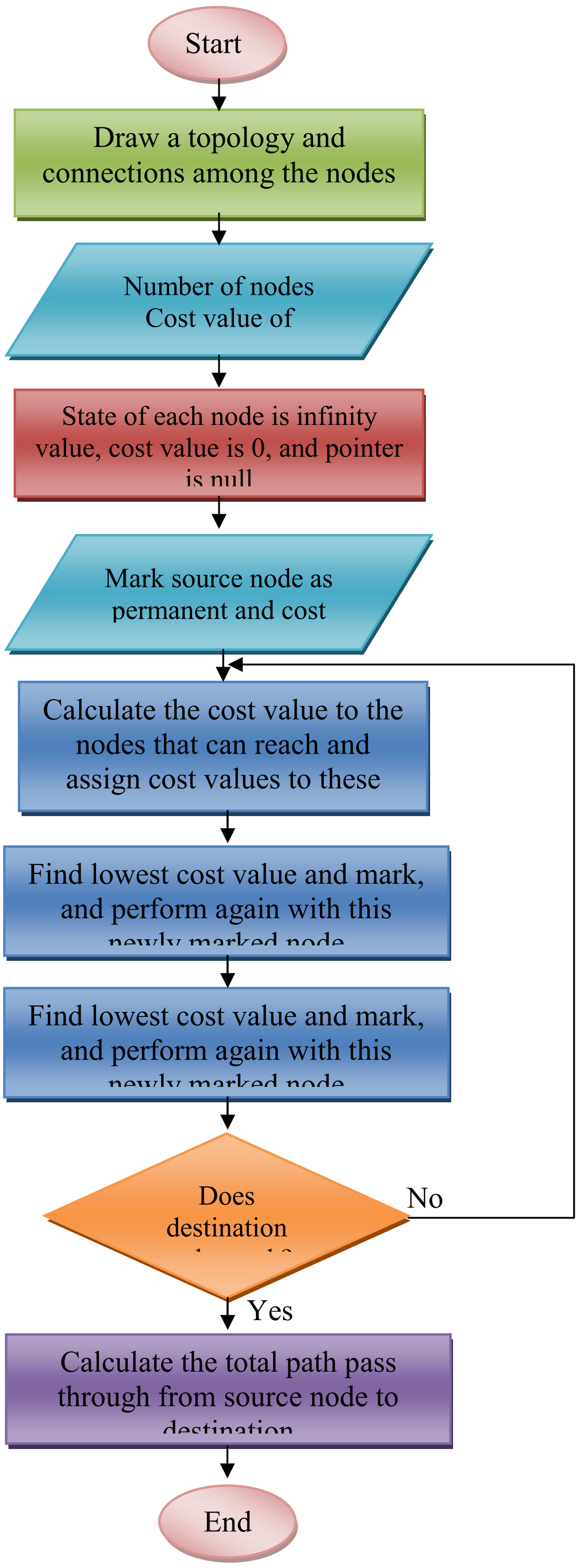
Start
Draw a topology and connections among the nodes
Number of nodes Cost value of
State of each node is infinity value, cost value is 0, and pointer is null
Mark source node as permanent and cost
Calculate the cost value to the nodes that can reach and assign cost values to these
Find lowest cost value and mark, and perform again with this newly marked node
Find lowest cost value and mark, and perform again with this newly marked node
Does destination
No
Yes
Calculate the total path pass through from source node to destination
End

Figure (11) Flow chart of Dijkstra's algorithm

The steps of operation of circuit are mentioned below:

1. Before power on, the DIP switch of 8 pins must be set off, which mean there is no input data to microcontroller, and all paths has an infinity set value as shown in figure (12) .

2. When power on, initialization messages showed up on LCD, as shown in figure (13).

3. Later, a message showed up asking for the source and destination node, as shown in figure (14).

4. After specify the source and destination node, a weights (lengths) for each connection must be set, as shown in figure (15).

5. Finally, results of shortest path will be calculated, beside that, a path will be showed from specified source to specified destination, as shown in figure (16).

5. Conclusion

Our research can solve one of an important algorithm in graph theory which used to find the shortest path called Dijkstra's Algorithm. Our experimental result has been focus on implementation of the Dijkstra's algorithm to find the shortest-path from source node to destination node of various connections on different topologies.

This hardware simulation is considered very useful tool kit which studies the theory of graph and shortest path in real.

This work has focus on explaining the steps of one of graph theory to search for shortest path. The work has also revealed that there is considerable variation of lengths and weights for each link of proposed design topology exhibited by different work. Compared to other approaches that has only has design an applications to analysis a graph theory especially in Dijkstra's algorithm, this work has extended to display the results on LCD for each step of process and in each link has to choose a weight from node to others.

6. References

1. Jean Walrand; 2010 ."Path Problems in Networks"; Morgan and Claypool.
2. C. Vasudev; 2006. "Graph Theory in Applications"; New age international publishers.
3. Jorgen Bang-Jesen, Gregory Z.Gutin; 2009. "Digraphs, Theory, Algorithms and Applications"; Springer Monographs in Mathematics.
4. Robin J. Wilson; 1996. "Introduction to Graph Theory"; Addison Wesley Longman.
5. Chris Braithwaite, Fred Cowan, and Hassan Parchizadeh; 2004. "8051 Microcontrollers, An Applications-Based Introduction"; Elseveir.
6. Claus Kuhnel; 2001."Bascom Programming of Microcontrollers with Ease".