Computer and Information Science; Vol. 6, No. 3; 2013
ISSN 1913-8989 E-ISSN 1913-8997
Published by Canadian Center of Science and Education

# An Efficient Cursor Search Algorithm

Yongping Huang[1], Shufan Yang[1] & Yushan Jin[1]

[1] College of Computer Science and Technology, Jilin University, Changchun, China

Correspondence: Yongping Huang, College of Computer Science and Technology, Jilin University, 130012 Changchun, China. E-mail: jinys@jlu.edu.cn

Received: April 13, 2013 Accepted: May 30, 2013 Online Published: June 28, 2013

doi:10.5539/cis.v6n3p96 URL: http://dx.doi.org/10.5539/cis.v6n3p96

**Abstract**

Table-based searching method is always used to improve the operating speed in embedded system measurement applications. Efficiency of the searching is a key factor in improving system performance. Based on the analysis of traditional search algorithms and characteristics of measured physical sensor signals, a simple and efficient cursor search algorithm is proposed, and is compared with traditional search algorithms. The simulation experiments were made, and the results proved that the cursor search algorithm, which the time complexity is $O(1)$, is superior to the traditional search algorithms. The cursor search algorithm is suitable for sensor signal measurement of embedded system applications.

**Keywords:** sensor signal, physical quantity, search algorithm, embedded system, continuity, measurement

## 1. Introduction

In terms of the measurement of physical quantities, the efficiency of the table searching method affects the performance of system. Because of low efficiency or other reasons, the traditional search algorithms are not applicable.

As an information processing system embedded in particular object systems, embedded system is often closely related to specific physical processes (Peter, 2011). Processing time and cooperativity are main technical difficulties (Michael & Massa, 2006). As cyber-physical system (CPS), embedded system is an integration of computation with physical processes. It monitors and controls the physical processes, usually with feedback loops where physical processes affect computations and vice versa (Edward & Sanjit, 2011).

Typically, the relationship of the detection module's outputs and the measured physical quantities are non-linear. There are two main reasons contributing. One is that the structural principle of the sensor is non-linear. The other is the use of the non-linear circuits (George, 1989).

Some circuits are used to solve the non-linear problems. But, this means high costs, complexity, debugging difficult, low accuracy and poor interoperability. A better solution is linearization during the processing of the programs by the microprocessor (George, 1989), which has the following advantages:

a) Simplifying the hardware structure, cost and complexity

b) Improving the linearization precision

c) Being versatile and producing different linearization correction features

Because of the nonlinearity of the sensor signals, the conversion formulas are very complicate. It's not suitable for embedded systems with limited resources. The common way is to quantify the theoretical formula to extract the standard data conversion table. And then, search the table to get the converted value (George, 1989). What needs to be done is to establish the conversion table between measured values and actual values, and find a certain search algorithm.

Linearization of the measured value and getting the actual value can be regarded as a process of searching table. For embedded system, the efficiency of the system depends on the speed of the search process. A simple, efficient searching algorithm becomes the focus of system design.

In computer science, there are several classical search algorithms. Linear search algorithm is the simplest one and its time complexity is $O(n)$. “Begin at the beginning, and go on till you find the right key; then stop”. This sequential procedure is the obvious way to search, and many of the more intricate algorithms are based on it. In

later will see, the sequential search involves some very interesting ideas, in spite of its simplicity (Knuth, 1999).

For an ordered table, a binary search algorithm is particularly rapid. The probe tells which half of the table should be searched next, and the same procedure can be used again, comparing key value to the middle key of the selected half, etc. After at most $\lg N$ comparisons, the algorithm will find the key value or it will establish that it is not present (Knuth, 1999). Therefore, its time complexity is $O(\log_2 N)$.

Hashing search is to avoid all the rummaging around by doing some arithmetical calculation on key value, computing a function h(k) that is the location of k. The time complexity of searching on a well-constructed hashing table is $O(1)$. But there are two issues required to address: a good hash function and a method for resolving hash conflicts (Thomas, 2001).

In this paper, based on the non-linear and time continuity of the sensor signals in embedded system, we propose a cursor search algorithm with simple, appropriate and efficient characteristics. In this field, its time complexity is $O(1)$.

This paper is organized as follows. In Section 2, we focus on research works associated with the search algorithms in computer science. Characteristics of embedded systems and sensor signals are described in the Section 3. In Section 4 we analyze the four kind of traditional search algorithms and summarize some of their characteristics. In Section 5 we present and elaborate the cursor search algorithm and analyze its average cost of searching. We compare it with traditional search algorithms through the simulation in Section 6. Section 7 provides concluding remarks.

## 2. Related Works

During the pre-computer era, many books of logarithm tables, trigonometry tables, etc., were compiled, so that mathematical calculations could be replaced by searching. With the development of calculating machine and random access memory, searching tables or information storage and retrieval issues have increasingly become an important aspect in computer science.

In the 50's, many computer scholar's have clearly represented the traditional search algorithm. The first surveys of the searching problem were published by Dumey (1956, pp. 6-9); Peterson (1957, pp. 130-146); Booth (1958, pp. 159-164); Douglas (1959, pp. 1-9). More extensive treatments were given later by Kenneth (1962, pp. 133-158), and by Werner (1963, pp. 86-111). During the early 1960s, a number of interesting new search procedures based on tree structures were introduced (Knuth, 1999). Now, research about search is still actively continuing. Its research and application have been gradually expanding to the most fields of computer science, such as the operating systems (Nicolescu, Ignat, Savaria, & Nicolescu, 2006), network communications (Behnamian & Fatemi, 2012), database systems, artificial intelligence (Balwinder et al., 2012), image processing, information security, etc.

However, in the area of sensor signal detection in embedded systems, no one analyze its characteristics and propose an applicable search algorithm. This paper will do these and present a new and useful cursor search algorithm for sensor signal detection.

## 3. Measurement in Embedded System

Embedded system with a principal role of interacting with the physical world must, of necessity, acquire some properties of the physical world. Quantities of the physical world tend to have two significant characteristics: continuity and locality.

The science of computation has systematically abstracted away the physical world. Embedded systems, however, engage the physical world. Time, concurrency, liveness, robustness, continuum, reactivity, and resource management must be remarried to computation (Edward, 2002).

Time constraints: Almost every task is associated with some time constraints (Vijaykrishnan & Yuan, 2006). One very common form of time constraints is deadlines associated with tasks (Hermann & Gunther, 2003). It is a responsibility of the embedded system to ensure that all tasks meet their respective time constraints (Rajib, 2007).

Resource constraints: Embedded system is used to accomplish a specific task and embedded in specific objects (Peter, 2011). Therefore, embedded systems is constrained for their size, power capacity (may be battery operated), limited memory capacity (especially RAM size), CPU speed and function capacity (Kai, David, & Li, 2009).

Interactivity: Embedded system is closely linked with the physical world, or even part of the physical world. Its principal role is not the transformation of data, but rather the interaction with the physical world. It executes on

machines that are not computers. They are cars, airplanes, telephones, audio equipment, robots, appliances, toys, security systems, pacemakers, heart monitors, weapons, television sets, printers, scanners, climate control systems, manufacturing systems, and so on. It's necessary to acquire some properties of the physical world. Consequently, the designer of that system should be the person who best understands that physical world (Edward, 2002). Only in this way, can really proceed from the view of the physical world, choose the appropriate method, and design a practical system.

From the interactive features of the embedded system, we must understand the physical world in the field, in order to propose a suitable method. With limited computing resources and storage resources, real-time embedded systems need a simple and efficient search algorithm.

Quantities of the physical world tend to have two significant characteristics: continuity and locality (Boi & Peter, 1992; Brain, 1984; Jan, 2005).

Physical quantities are measured values detected by sensors. Continuity refers to that physical quantities vary slowly. Therefore, compared with the last, the newly measured value changes only a little. In other words, if represented in a curve, the curve is derivable. Locality refers to that the possible measured values within a certain time are only in a small range. Figure 1 shows the 24-hours temperature curve of a city in north China.

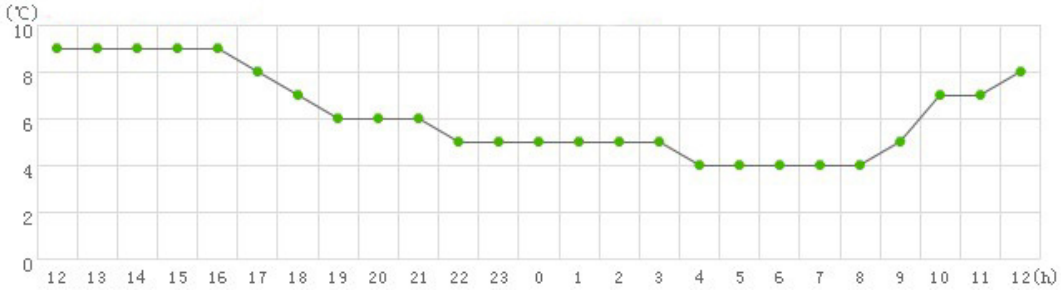


Figure 1. A 24-hours temperature curve of a city

Continuity and locality are the basic properties of the physical quantities such as displacement, force, temperature, velocity, humidity and flow (Brain, 1984). These continuous-time signals can often be modeled using differential equations, or equivalently, integral equations (Edward & Sanjit, 2011).

**4. Summary of Traditional Search Algorithms**

Although the linear search algorithm is very simple, it has some interesting advantages:

a) In linear search process, the location of the next element to be compared with is always known. And it is only to get partial block of the table in several comparisons.

b) If the searching probabilities of the elements in the table are learned and elements are placed in the right locations in the table, search efficiency will significantly increase (Knuth, 1999).

Given a table of records $R_1,R_2,\ldots,R_N$, whose respective keys are $K_1,K_2,\ldots,K_N$, the algorithm searches for a given K. In a general situation, key $K_i$ will occur with probability $p_i$, where

$$p_1 + p_2 + \cdots + p_N = 1. \tag{1}$$

The time required to do a successful search is essentially proportional to the number of comparisons, C, the average value of which is

$$\overline{C}_N = p_1 + 2p_2 + \cdots + Np_N. \tag{2}$$

If $p_1 = p_2 = \cdots p_N = 1/N$, formula (2) reduces to $\overline{C}_N = (N+1)/2$; we have already derived this. If we have the option of putting the records into the table in any desired order, this quantity $\overline{C}_N$ is smallest when

$$p_1 \geq p_2 \geq \cdots \geq p_N, \tag{3}$$

that is, when the most frequently used records appear near the beginning. Suppose, on the other hand, that

$$p_1 = \frac{1}{2}, p_2 = \frac{1}{4}, \cdots, p_{N-1} = \frac{1}{2^{N-1}}, p_N = \frac{1}{2^{N-1}}. \tag{4}$$

Then $\overline{C}_N = 2 - 2^{1-N}$; the average number of comparison is less than two, for this distribution, if the records

appear in the proper order within the table (Knuth, 1999).

So, if we find the probability distribution of sequential searching, there will be a much reduce of cost.

Binary search algorithm is a logarithmic algorithm for searching in ordered table. After at most $\lg N$ comparisons, we will have found the key or we will have established that it is not present (Knuth, 1999).

In practical use, binary search has some inherent disadvantages:

a) In some platforms, tail recursion is not eliminated and the recursive version of binary search algorithm requires more stack space or even causes a memory overflow.

b) Binary search can interact poorly with the memory hierarchy (i.e. caching), because of its random-access nature. For external searching, care must be taken of that each of the first several probes will lead to disk seeks.

Search in hash table is fast and the time complexity is $O(1)$. But it's hard to achieve in microprocessors, because of its obvious shortcomings (Thomas, 2001):

a) The cost of a good hash function can be significantly higher than the inner loop of the search algorithm for a sequential list or search tree.

b) The hash tables are not effective when the number of entries is very small.

c) The cost of a single operation may be quite high. In particular, if the hash table uses dynamic resizing, an insertion or deletion operation may occasionally take time proportional to the number of entries. This is a serious drawback in real-time or interactive applications.

d) Hash tables in general exhibit poor locality of reference-that is, the data to be accessed is distributed at random in memory. This can trigger microprocessor cache misses that cause long delays.

**5. Cursor Search Algorithm**

The pros and cons of an algorithm are closely related to its application environments. An algorithm may be theoretically good, but tends to ignore a lot of factors. The fact is that a match to the requirements of embedded systems is the best.

According to these, a cursor search algorithm is proposed. It is simple, efficient and suitable for the measurement of physical quantities.

For the first search, cursor search algorithm gets the element corresponding to the key value, and the cursor is positioned for the first time. For future searches, according to the continuity and locality of the measured value, as well as the characteristics of linear search method, the algorithm compares the searched value with the key value at cursor position. If the searched value is smaller than the key value of cursor position element, cursor left shifts, until it is not. Conversely, if the searched value is greater than the key value of cursor position element, the cursor right shifts, until it is not. After the process, if the cursor is not out of bounds, the algorithm will get the actual value corresponding to the measured value. Or if the cursor is out of bounds, mark that the next search is a first search and return "not found".

Suppose an array $A=\{a_1,a_2,\cdots,a_N\}$, with probability $p_k$ that record $A_k$ will be sought. For the first search, the algorithm define that the cursor is *i* using binary search method or a linear search method.

Because of the static characteristic of the cursor variable, cursor is still *i* until the next search. Near the cursor, the searched value (assuming it's $A_j$) have a greater probability, according to the continuity and locality of the measured value. When $|i-j|$ is increasing, $p_j$ is getting smaller, as Figure 2 shows.

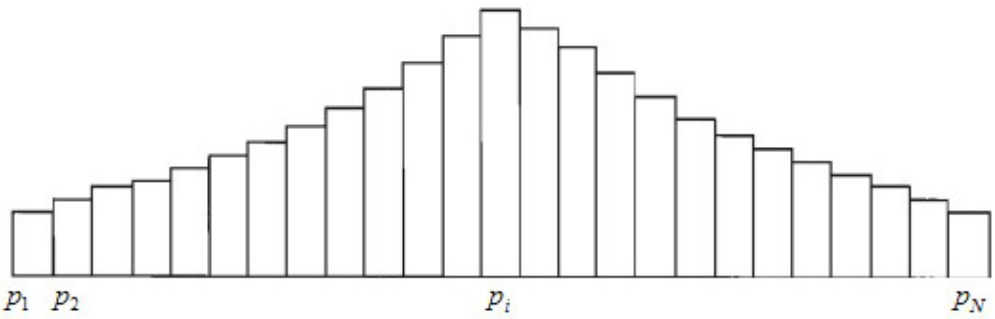


Figure 2. Probabilities of measured value base on the last searching

In other words, the probabilities of measured value meet

$$p_1 \le p_2 \le \cdots \le p_i, \quad p_i \ge p_{i-1} \ge \cdots \ge p_N \quad \text{and} \quad p_1 + p_2 + \cdots + p_N = 1. \tag{5}$$

Consider a simple case, supposing

$$p_{i-k} = p_{i+k} = c^k p_i, \quad c < 1. \tag{6}$$

Then

$$c = \frac{1 - p_i}{1 + p_i}. \tag{7}$$

The average comparison times of the algorithm can be expressed as

$$\begin{aligned} \overline{C}_N &= p_i + 2(p_{i-1} + p_{i+1}) + 3(p_{i-2} + p_{i+2}) + \cdots \\ &= p_i + 2cp_i + 3c^2 p_i + \cdots \\ &= p_i(1 + 2c + 3c^2 + \cdots) \\ &= p_i \cdot \frac{1}{(1-c)^2} \\ &= \frac{(1+p_i)^2}{4p_i}. \end{aligned} \tag{8}$$

That is

$$\overline{C}_N = \frac{p_i}{4} + \frac{1}{2} + \frac{1}{4p_i}. \tag{9}$$

Figure 3 shows the relationship of $\overline{C}_N$ and $p_i$ when N is 100. $\overline{C}_N$ is the average comparison times in Formula 9. $p_i$ is probability of measured value based on the last value. And N is the size of array.

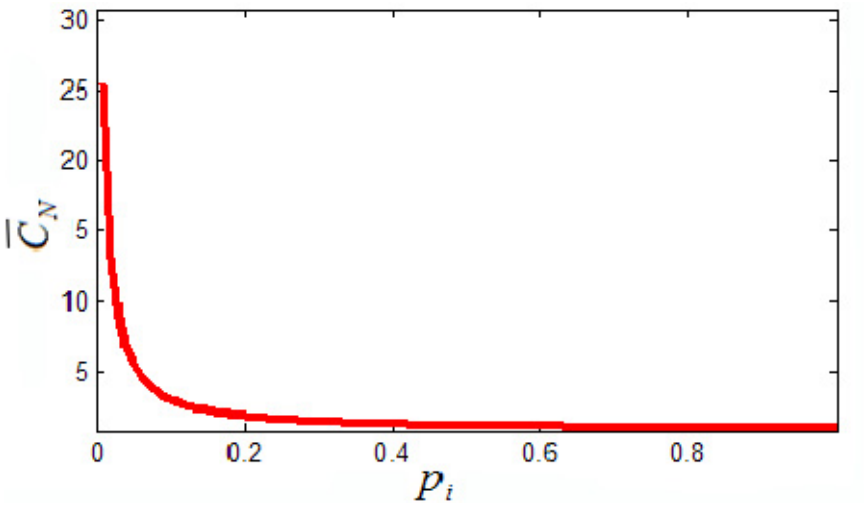


Figure 3. The relationship of $\overline{C}_N$ and $p_i$

Therefore, even if $p_i$ is 0.1, average number of comparison $\overline{C}_N$ is only 3. When $p_i$ is large enough, it's slightly larger than 1. For the sensor measurement, because of the continuity and locality, $p_i$ can be large enough to guarantee that the time complexity of the algorithm is $O(1)$.

Here, based on temperature detection system, use cursor search algorithm to search in a table and get the actual temperature value (an approximate value is OK). Assume the table is array A, with each element is a data structure consist of a measured value and an actual temperature value. Array A is sorted by measured values. The variable "cursor" is must be declared as a static or a global variable. The parameter "key" is the measured value to be searched. Figure 4 shows the flowchart of the algorithm.

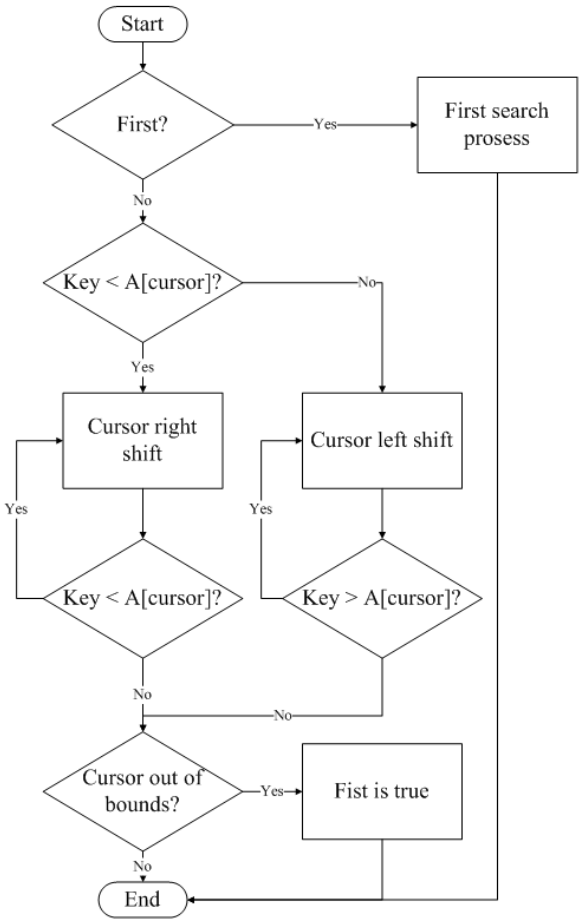


Figure 4. The flowchart of cursor search algorithm

The pseudo code of the algorithm is:

*Initialize:*

*(i)Variable cursor:= 0, and flag is First := true;*

*(ii)Array A is t the table, with each element is an instance of the structure {MeasuredValue m; TemperatureValue t;}c;*

*(iii)Variable maxLength is the size of A.*

```
Cursor_Search(MeasuredValue key){
if isFirst = true then
cursor := First_Search(key)
isFirst := false
return cursor
end
if key < A[cursor].m then
do
cursor := cursor - 1
while key < A[cursor].m and cursor is not out of bound
else if key > A[cursor].m then
do
cursor := cursor + 1
while key > A[cursor].m and cursor is not out of bound
end
if cursor is not out of bounds
```

```
return cursor
else
isFirst := true
return not found
end
}
```

The First_Search function is used to position the cursor for the first time, and it can be one of these:

a. a linear searching from the middle of table

b. a linear searching from the head of table

c. a binary searching

Compared with linear search algorithm, cursor search algorithm makes the searches to be catenated using a static cursor. Because of the features of the physical quantities, the probability of an element to be sought is maximum when it's near the cursor. The cost is significantly reduced by using a static cursor.

Compared with binary search algorithm which disregards the features, cursor search algorithm produces a better performance with a cost of $O(1)$ in theory. In addition, the random-access of binary search algorithm sometimes causes negative effects.

Compared with hash table search method, cursor search algorithm is also $O(1)$. But the hash table search method is a random-accessed and has difficulties in application. First, a good hash function is complex to compute, and is too costly for the microprocessors in embedded systems. Second, hash table need to be dynamically maintained at run time, which also increase the response time of the programs.

Theoretically, in the field of sensor detection, cursor algorithm is superior to other traditional algorithms. It makes use of the probability characteristics and locality. It is a catenated search algorithm, which begins where the last one left off. Its time complexity is $O(1)$. Furthermore, because of the locality of the algorithm, it has a good performance in memory hierarchy.

**6. Simulation and Comparison**

A simulation experiment is performed to prove its superiority.

In the simulation, a temperature detection process is constructed. Pt100 is a most common resistance temperature detector (RTD) used in industry, which have a nominal resistance of 100 ohms at 0 °C. Its resistance changes with temperature. So the temperature can be measured according to the change of its resistance. The resistance is measured by a microprocessor. And then, the resistance value needs to be converted into temperature value. Therefore, a temperature-resistance table is needed to seek the right temperature by the microprocessor. In simulation, only a part of them is extracted as the table, as shows in Table 1.

Table 1. The temperature-resistance table of Pt100

| T (°C) | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| | Resistance (Ω) | | | | | | | | | |
| -20 | 92.16 | 91.77 | 91.37 | 90.98 | 90.59 | 90.19 | 89.80 | 89.40 | 89.01 | 88.62 |
| -10 | 96.09 | 95.69 | 95.30 | 94.91 | 94.52 | 94.12 | 93.73 | 93.34 | 92.95 | 92.55 |
| 0 | 100.00 | 99.61 | 99.22 | 98.83 | 98.44 | 98.04 | 97.65 | 97.26 | 96.87 | 96.48 |
| 0 | 100.00 | 100.39 | 100.78 | 101.17 | 101.56 | 101.95 | 102.34 | 102.73 | 103.12 | 103.51 |
| 10 | 103.90 | 104.29 | 104.68 | 105.07 | 105.46 | 105.85 | 106.24 | 106.63 | 107.02 | 107.40 |
| 20 | 107.79 | 108.18 | 108.57 | 108.96 | 109.35 | 109.73 | 110.12 | 110.51 | 110.90 | 111.29 |
| 30 | 111.67 | 112.06 | 112.45 | 112.83 | 113.22 | 113.61 | 114.00 | 114.38 | 114.77 | 115.15 |
| 40 | 115.54 | 115.93 | 116.31 | 116.70 | 117.08 | 117.47 | 117.86 | 118.24 | 118.63 | 119.01 |
| 50 | 119.40 | 119.78 | 120.17 | 120.55 | 120.94 | 121.32 | 121.71 | 122.09 | 122.47 | 122.86 |
| 60 | 123.24 | 123.63 | 124.01 | 124.39 | 124.78 | 125.16 | 125.54 | 125.93 | 126.31 | 126.69 |
| 70 | 127.08 | 127.46 | 127.84 | 128.22 | 128.61 | 128.99 | 129.37 | 129.75 | 130.13 | 130.52 |
| 80 | 130.90 | 131.28 | 131.66 | 132.04 | 132.42 | 132.80 | 133.18 | 133.57 | 133.95 | 134.33 |
| 90 | 134.71 | 135.09 | 135.47 | 135.85 | 136.23 | 136.61 | 136.99 | 137.37 | 137.75 | 138.13 |

A collection of resistance data is detected in a day of a city. And the number of it is 63, as Figure 5 shows.

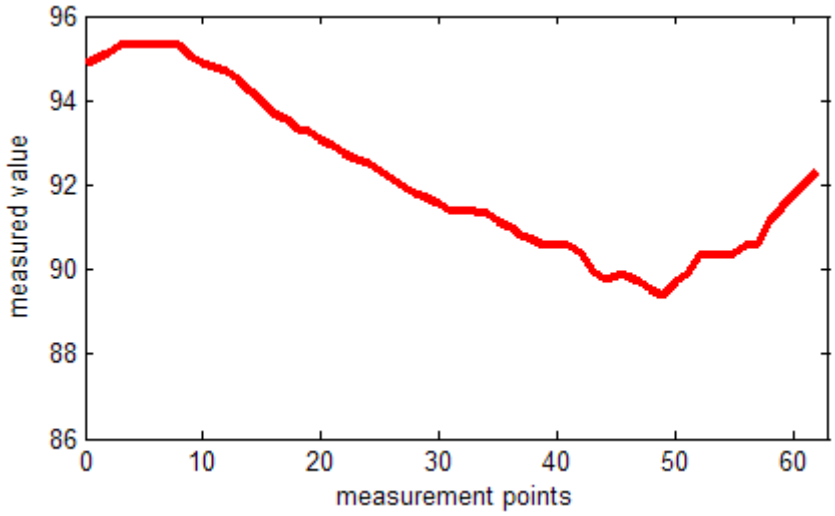


Figure 5. A collection of resistance data detected by Pt100

In simulation, these data is the input of the cursor search algorithm, linear search algorithm and binary search algorithm. Through simulation, temperature curve is generated by the three algorithms, as Figure 6 shows.

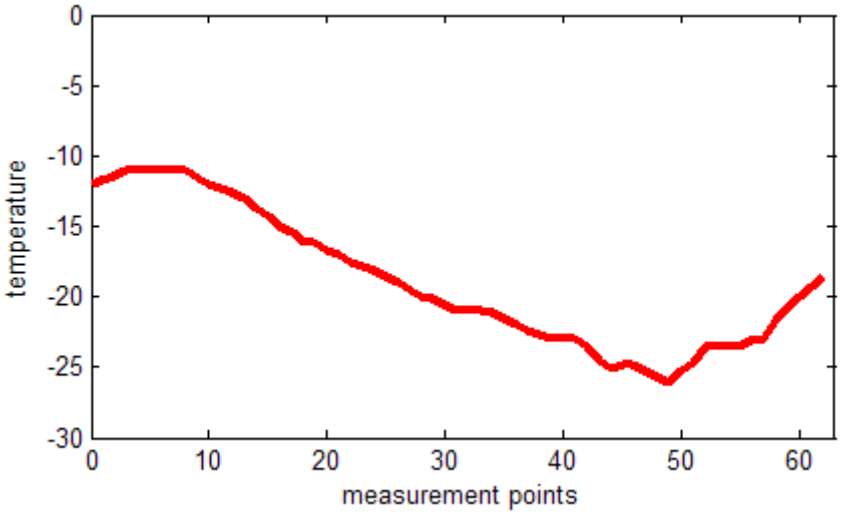


Figure 6. Temperature curve achieved by algorithms

Figure 7 shows the number of comparisons of each searching by linear search algorithm in the simulation. Figure 8 shows the cumulative number of comparisons. Because most of the data is gathered in the front of the table, its actual efficiency is better than theoretical. The total number of comparisons remains 724 and the numbers of each searching is not stable.

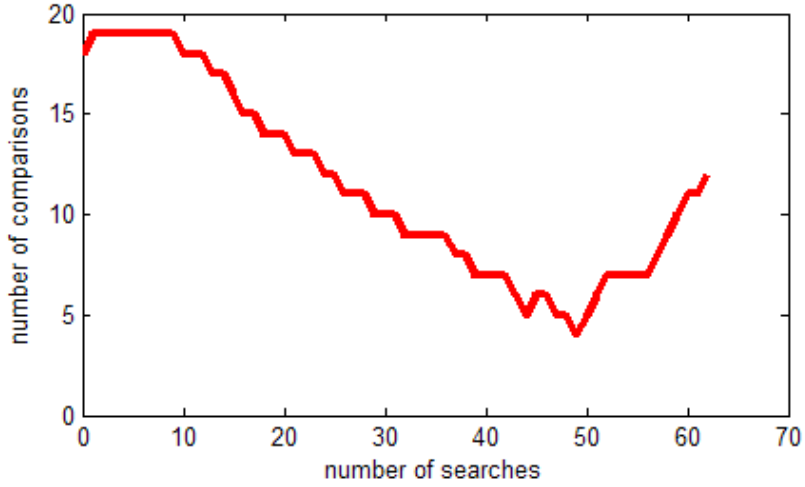


Figure 7. The curve of comparison times of each searching by linear search algorithm

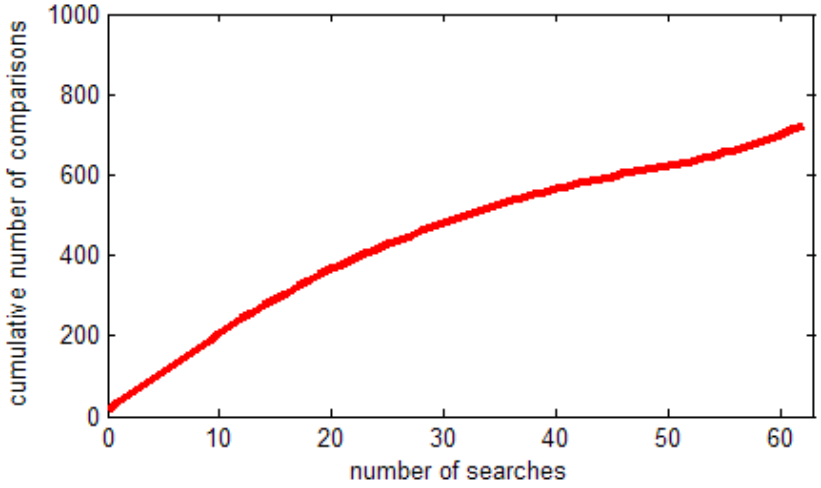


Figure 8. The curve of cumulative comparison times of searching by linear search algorithm

Figure 9 and Figure 10 show the performance of binary search algorithm. The number of comparisons of each searching by the algorithm is stable, and the total number of comparisons is 383.

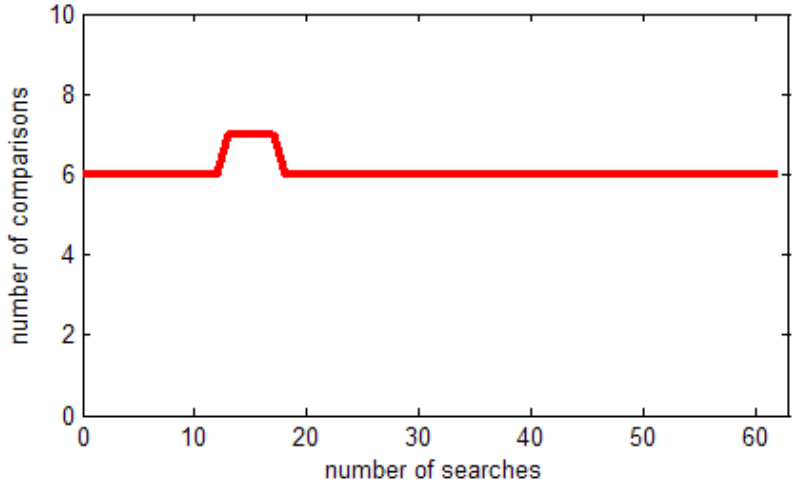


Figure 9. The curve of comparison times of each searching by binary search algorithm

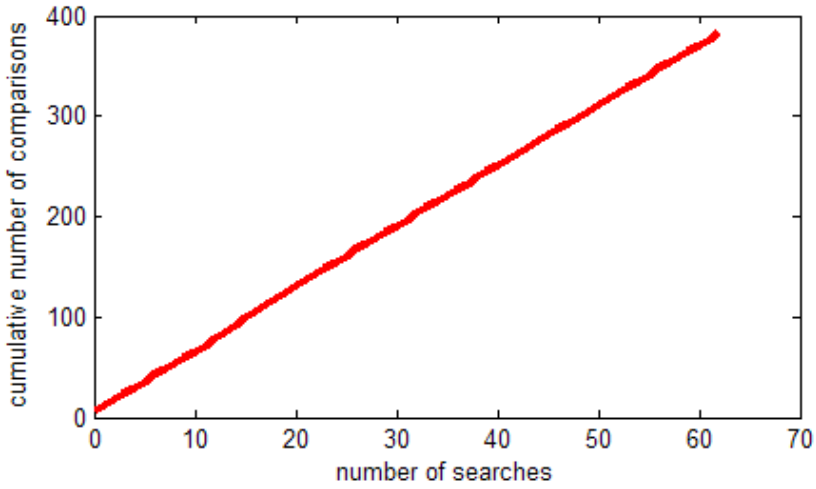


Figure 10. The curve of cumulative comparisons times of searching by binary search algorithm

The performance of cursor search algorithm is shown as Figure 11 and Figure 12. It appears that the comparison number of each searching (except the first searching to position the cursor) is only a few times (1 or 2 as Figure 10 shows). The total number of comparisons is 94, which is significantly than above two algorithms'.

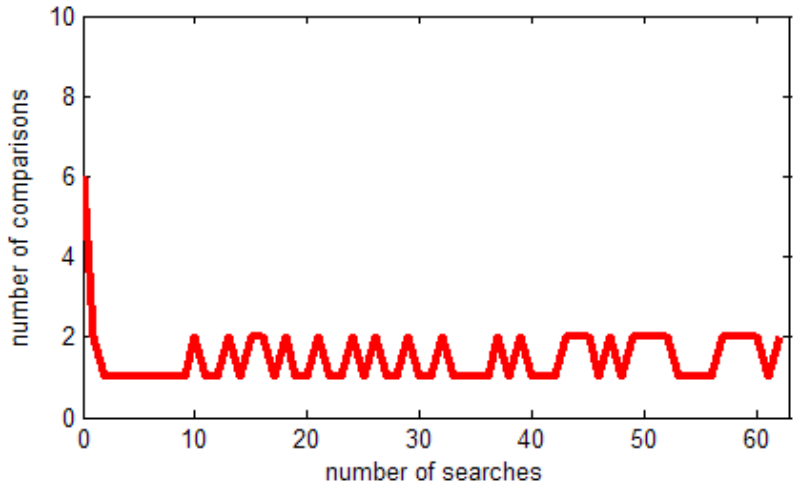


Figure 11. The curve of comparison times of each searching by cursor search algorithm

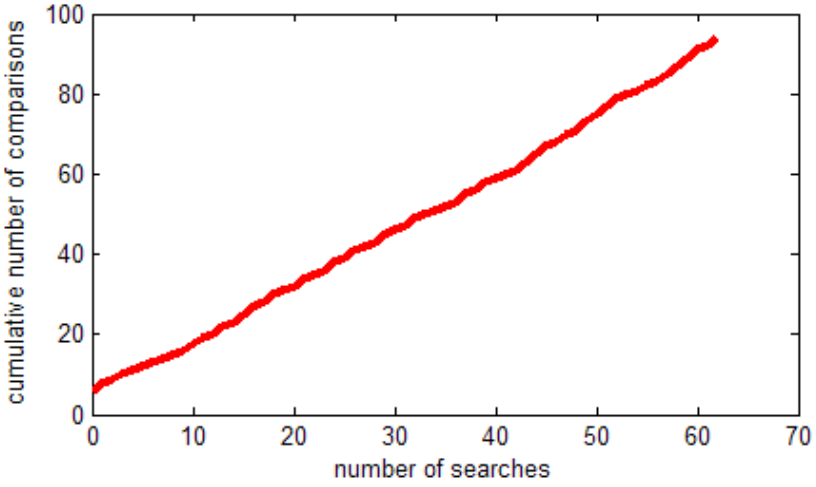


Figure 12. The curve of cumulative comparison times of searching by cursor search algorithm

Besides, the curve of the probabilities of the distance between the sought element and the cursor at that time is generated as Figure 13. It appears that the sought element is always near the cursor, which is why the cursor search algorithm is so efficient in measurement of continuous physical quantities.

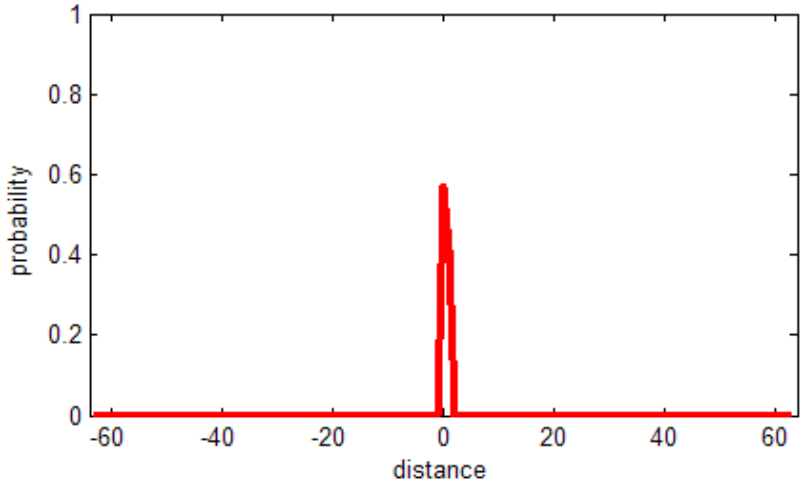


Figure 13. The probabilities of the distance between the sought element and the cursor

The performances of three search algorithms are concluded as Table 2. The efficiency of cursor search algorithm is four times of binary search algorithm and eight times of linear search algorithm.

Table 2. The performances of three search algorithms

| | Time complexity | Space complexity | Table size | Number of searching | Total number of comparisons | Average number of comparisons |
|---|---|---|---|---|---|---|
| Linear search | O($n$) | O(1) | 140 | 63 | 724 | 11.5 |
| Binary search | O(lg $N$) | O(1) | 140 | 63 | 383 | 6.1 |
| Cursor search | O(1) | O(1) | 140 | 63 | 94 | 1.5 |

## 7. Conclusions

Based on the sensor signal detection, a cursor search algorithm is proposed. According to the continuity and locality of the measured value, the algorithm catenates the successive searches, and has an average time complexity of O(1) . The simulation shows that the efficiency of cursor search algorithm is four times of binary search algorithm. It is superior in the measurement system. At the same time, the design philosophy of the algorithm provides a useful reference.

## References

Barr, M., & Massa, A. (2009). *Programming embedded systems: with C and GNU development tools*. O'Reilly Media.

Behnamian, J., & Fatemi, G. S. M. T. (2012). The heterogeneous multi-factory production network scheduling with adaptive communication policy and parallel machine. *Information Sciences, 219*(10), 181-196. http://dx.doi.org/10.1016/j.ins.2012.07.020

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2001). *Introduction to algorithms*. MIT press.

Donald, E. K. (1999). The art of computer programming. *Sorting and searching, 3*, 426-458.

Faltings, B., & Struss, P. (1992). *Recent advances in qualitative physics*. The MIT Press.

Kondraske, G. V. (1989). *Sensor conditioning method and apparatus*. Washington, DC: U.S. Patent and Trademark Office.

Kopetz, H., & Bauer, G. (2003). The time-triggered architecture. *Proceedings of the IEEE, 91*(1), 112-126. http://dx.doi.org/10.1109/JPROC.2002.805821

Lee, E. A., & Seshia, S. A. (2011). *Introduction to embedded systems: A cyber-physical systems approach*. Lee & Seshia. http://dx.doi.org/10.1145/1719010.1719011

Lee, E. A. (2002). Embedded software. *Advances in computers, 56*, 55-95. http://dx.doi.org/10.1016/S0065-2458(02)80004-3

Mall, R. (2007). *Real-Time Systems: Theory and Practice*. Prentice Hall.

Marwedel, P. (2011). *Embedded system design: Embedded systems foundations of cyber-physical systems*. Springer Science+ Business Media.

Mycielski, J. (2005). Pure mathematics and physical reality (continuity and computability). *Fundam. Prikl. Mat, 11*(5), 151-168. http://dx.doi.org/10.1007/s10958-007-0368-y

Narayanan, V., & Xie, Y. (2006). Reliability concerns in embedded system designs. *Computer, 39*(1), 118-120. http://dx.doi.org/10.1109/MC.2006.31

Nicolescu, B., Ignat, N., Savaria, Y., & Nicolescu, G. (2006). Analysis of Real-time systems sensitivity to transient faults using MicroC kernel. *Nuclear Science, IEEE Transactions on, 53*(4), 1902-1909. http://dx.doi.org/10.1109/TNS.2006.880940

Qian, K., Den Haring, D., & Cao, L. (2009). *Embedded software development with C*. Springer. http://dx.doi.org/10.1007/978-1-4419-0606-9

Singh, B., Singh, A., Ahmed, A., Wilson, G. A., Pickering, B. W., Herasevich, V., & Li, G. (2012). Derivation and Validation of Automated Electronic Search Strategies to Extract Charlson Comorbidities From Electronic Medical Records. In *Mayo Clinic Proceedings* (Vol. 87, No. 9, pp. 817-824). Elsevier. http://dx.doi.org/10.1016/j.mayocp.2012.04.015

Williams, B. (1984). The use of continuity in a qualitative physics. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-84)* (pp. 350-354).