Real Time Multitasking In Ardiuno Using Look-Up Table

G.Nandha
Department of ECE
MIT
Puducherry, India

V.Gokularaman
Department of ECE
MIT
Puducherry, India

P.S.Harikumar
Department of ECE
MIT
Puducherry, India

P.Malarvizhi
Department of ECE
MIT
Puducherry, India

E.Ramassamy
Department of ECE
MIT
Puducherry, India

***Abstract* - This paper propose to RTM-LUT algorithm (Real Time Multitasking using Look Up Table) to create real time multitasking instead of using external RTOS in Arduino resulting in resources allocation optimization. Multitasking is a process of creating an environment where multiple independent operations/tasks take place in concurrent time.**

I. INTRODUCTION

Arduino, a complete open source embedded system based prototyping device which provides the developers a platform to create embedded system. The software for Arduino is also an open sources that software is named as Arduino IDE (Integrated Development Environment). The codes or programmes created in this software are called sketches. To create a sketch a developer needs only the basics skills of c++, a programming language and need not know the hardware and underlying architecture of the Arduino and the Atmel microcontrollers used in those boards. The biggest advantage in these boards is power from the USB is more than sufficient to drive the low power loads and input peripherals. Arduino uses framework method in which one cyclic operation takes place at a time.

II. EXISTING WORK

In the present world microcontrollers are made for doing one operation at a time. But in recent trends microcontrollers are made to do multitasking operation using a special operating system called RTOS (Real Time Operating System). These RTOS uses scheduling process for dividing the times according to the jitter time in the given time slotted evenly. If 'N' number of operation is to take place operation one is carried for a period of time specified as soon as the time exceeds the next operation is started or continued without any intimation while during this operation the old operation that as been carried out will be halted. Thus the paused operation has to wait for the specified jitter slot to begin. Thus in a cyclic order all the operations will be carried out in an order continuously. When this RTOS is booted with an Arduino board then that process will be called as RT-Arduino. In this RT-Arduino there will be more than one main loop present in the sketch of a single Arduino sketch created by the Arduino IDE.

DRAWBACKS

RTOS when it is implemented in any microcontrollers it occupies more memory spaces which is the biggest drawback of the RTOS. As microcontrollers need more memory space for the RTOS only 32bit or 64bit and higher versions can be used, while small microcontrollers with 16 bit or 8 bit and lower version does not support any external RTOS.

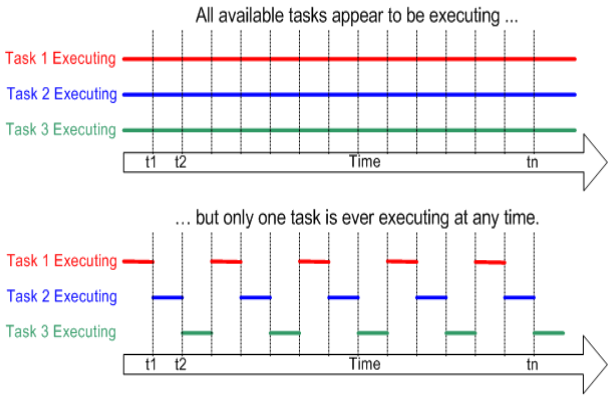


Fig 1: Representation of multitasking using RTOS.

Through this it is said to do multi operation in real time, it works under the principle and algorithms of time scheduling which actually does only one operation at a time under jitter time slotting and as the time exists the jitter time slot the operation which would be halted. RTOS follows the process of creating separate main loop for each operation. And each operation are carried out different frequencies, thus the operating speed of the microcontrollers would be reduced considerably. These are the notable drawbacks of this RTOS which affects the performance of the microcontrollers.

III. PROPOSED WORK

This paper approaches to follow a simple seven step algorithm to create a real time multitasking in any Arduino boards without using RTOS.

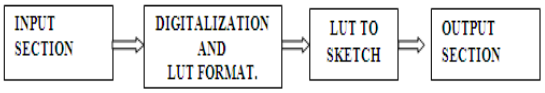


Fig 2: Block diagram of RTM-LUT algorithm.

A. INPUT SECTION

In this section the total number of input peripherals or lines to be connected in the Arduino board is counted and the total number of input peripherals is said to be

'N' (variable) and input peripherals are divided as digital and analog devices according to the type of data given by the Arduino board from the input peripherals. This forms the input section for this RTM-LUT algorithm.

B. DIGITALIZATION AND LUT FORMAT

This block consists of the most important part of peripherals algorithm, In this section the process of digitalization is done for analog input peripheral and the data is achieved at the result of this digitalization will be either '0' or '1' for both digital and analog input devices. These values are directly used in the Look up table to get a tabular column that will give all the possibilities in the operation to take place, which would be directly used in the Arduino sketch to achieve multitasking in real time.

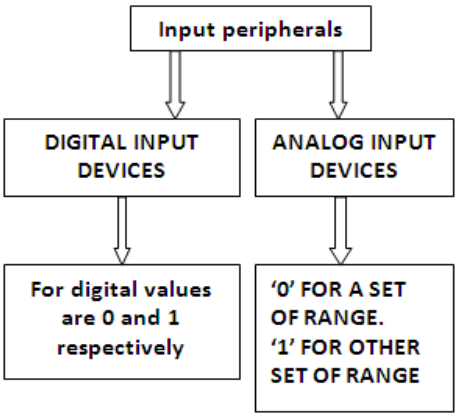


Fig 3: Digitalization of analog data

This paper proposes an algorithm named RTM-LUT (Real Time Multitasking using Look up Table). The main strength of this algorithm is that it does not need any RTOS to be booted in the microcontrollers, which saves most of the spaces of any microcontrollers and this RTM-LUT algorithm can be used for any small space allocated microcontrollers. In this all the spaces can be fully utilized for the sketch created in the Arduino IDE with single main loop and with only full operating frequency rather than getting various slots of frequency which affects the operating speed of the microcontrollers.

Peripheral1	Peripheral2
0	0
0	1
1	0
1	1

Table 1: sample array of LUT (Look Up Table).

C. LUT TO SKETCH

The LUT draw provides basic logics to be coded to create a sketch. For each sequence in an LUT there will be a set of multitask. By using this LUT as reference the developers can create the code following the algorithm. Basically LUT gives an array which shows the run compilation of any program in a tabular column.

```
Void setup()
{
    // setup code
}
Void loop()
{
 If((ip1 == 0)&&(ip2 == 0)&&.....( ip 'N' ==0/1)) //#1
    {
      //Set of various operations          //Sequence1
    }
If(.....) //#2
    {
      //Set of various operations          //Sequence2
    }
...../#n
                          //where ip = input peripherals
```

Fig 4: sketch format using LUT

D. OUTPUT SECTIONS

Output section compresses of output lines for the output peripherals from the Arduino board. These lines may be directly connected to output devices in case of output device being a low power load, whereas for a heavy load devices like motors the lines should be subjected to amplification nor it can be given to the relay to produce trigger pulse turn ON or OFF the high powered load.

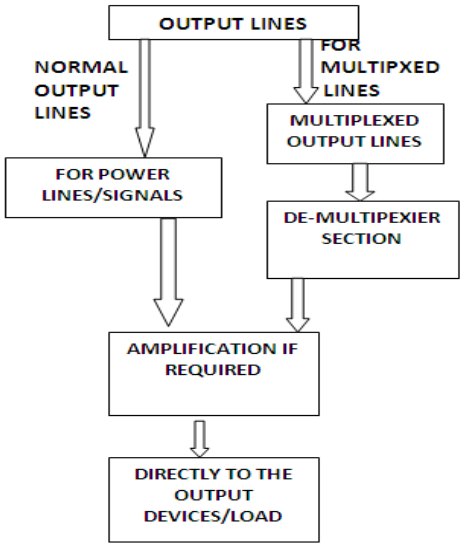


Fig 5: Flow diagram of output section

IV. FLOW CHART AND ALGORITHM OF RTM-LUT.

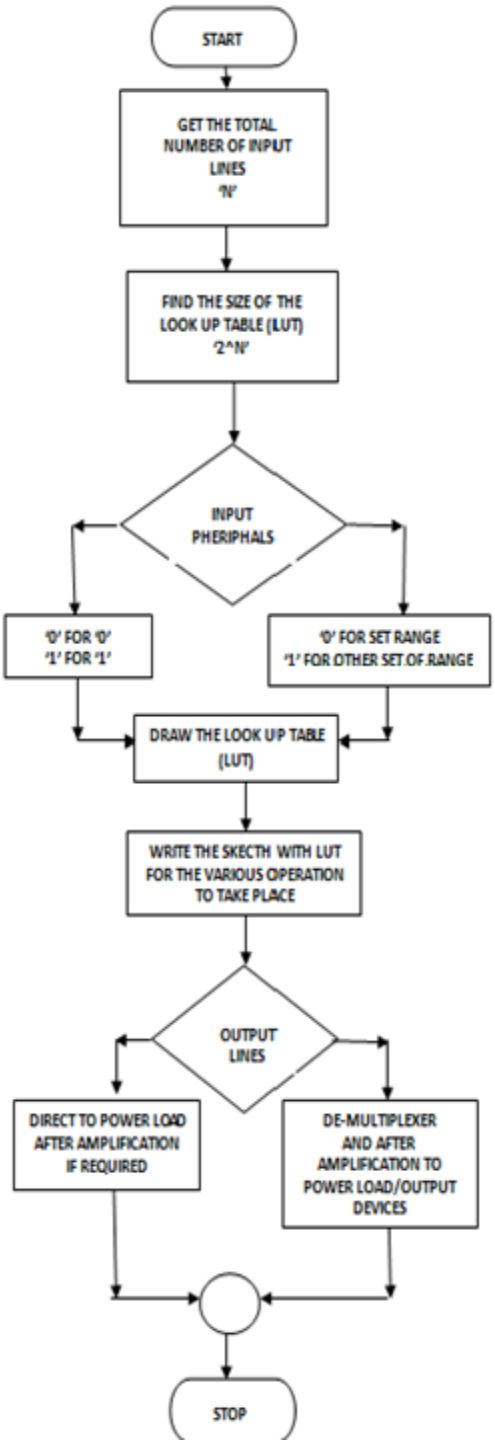


Fig 6: Flow chart of RTM-LUT algorithm

Flowchart

The flowchart is pictographically representation of this whole algorithm which gives an outlook of how this RTM-LUT works and helps to develop RTM-LUT sketch easily as per the blocks in the flowchart.

Algorithm for RTM-LUT:

Step1: Start.

Step2: Get the total number input peripherals to be connected with the Arduino board, let the total number of input peripherals be 'N'.

Step3: Find the size for the look up table using the value 'N', the size of the look up table is 2^n.

Step4: Digitalization of all the input peripherals,

For a digital input peripheral the values are '0' and '1'. For an analog input peripheral a "set of range of value is taken as 0", "The other set of range is taken as 1".

Step5: Using the digital values got from the above step, draw the Look up table.

Step6: Write the sketch according to the look up table achieved got from the above steps, to get the desired multiple outputs accordingly.

Step7: If the output lines are under multiplexed state then it should be de-multiplexed before it is used to drive the output peripheral loads. Else if it the output lines are not in multiplexed then it can.

V. OPERATIONAL BEHAVIOUR

The operational behaviour of the Arduino board for the RTM-LUP algorithm is very similar to a normal Arduino board doing one operation at any time. This method creates a pathway where all the different operations are mixed using the look up table and using the data taken from that table gives the sketch developers to code various operations to take place at any time. The strength of this RTM-LUT is this algorithm does not divide the time into many jitter slots. Thus all operations will occur in real concurrent time period.

Example

To explain this method more efficiently we present an example , which is composed up with 4 input peripheral devices among which 2 sensors are digital while 1 sensor is an analog device and 1 push button which can be taken as digital input peripheral device . The output from the Arduino board will be given through 4 lines which can be directly used to drive the output lines, for a multiplexed signal, there would be a total of 16 output peripheral devices from these which could be achieved using de-multiplexer where the signal after amplification or giving it to a relay may allow driving high end devices.

The aim of this example application is to create 4 different independent operation using RTM-LUT algorithms. The various for 4 different operations are

- **Operation1** – LED ONE (output peripheral1) should turn to ON state if external light falls on LDR sensor otherwise the output peripheral load should be in OFF state.
- **Operation2** – LED TWO (output peripheral 2) should turn ON if any object comes near crossing the threshold length specified by the programmers , otherwise the led should remain in OFF state.

- **Operation3** – LED THREE (output peripheral 3) should turn ON if the value crossed the boundary set by the developer, otherwise the output should remain in OFF state.
- **Operation4** – LED FOUR (output peripheral 4) and push button (input peripheral) in this , the total number of times the push button should be saved and for every even count the led load must remain in OFF state and for every odd count it has to remain in ON state These are the various set of operations which are independent should take place in real time.

Input peripherals	Digital/analog
Senor 1	Digital
Sensor 2	Digital
Sensor 3	Analog
Peripheral 4 (push button)	Digital

Table 2: Given data for example.

Step 1: Start

Step 2: Get the total number input peripherals to be connected with the Arduino board, let the total number of input peripherals be 'n'. So, according to 2nd step of the algorithm, we get the total number of input peripherals

$$'N' = 4$$

Step 3: Find the size for the look up table using the value 'n', the size of the look up table is 2^n forming R X n table where $R = 2^N$. Therefore we know the value of 'N's substituting it in '2^n'

$$2^N = 2^4$$

Thus the total size of the LUT is 16 in length. Therefore 16 is the Row size while 4 is the column size.

Step 4: Digitalization of all the input peripherals, For a digital input peripheral values will be '0' and '1' respectively.

For an analog input peripheral a "set of range of value is taken as 0". "The other set of range is taken as 1".

Therefore, for sensors 1, 2 and 4 there will be no effort to be done, for the 3 sensors being analog conversion should be done.

In case of many partition needed then it can be done using the orders 2^N. Where N is an integer 1, 2, 3...,

Step 5: Using the digital values got from the above step, draw the Look up table.

For the table, values between minimum and the specified data is taken as '0' and values form the mentioned data to maximum is taken as '1'

Sen1 (analog)	Sen2 (analog)	Sen3 (analog)	Sen4 (digital)
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1

Table3: LUT for example explained

Step 6: Write the sketch according to the look up table derived from the above steps, to get the desired multiple outputs accordingly. Each lines of the LUT could be used to create a set operation which as a combination forms multitasking.

To explain briefly we have presented a source code along, in this the 4 sensors are LDR, Ultra sonar, pushbutton, force sensor where LDR, Ultra sonar, force sensor are analog devices, while push button can be taken as digital input peripherals

Thus for the above mentioned specification, the sketch to burn on the Arduino board is given below.

Step 7: Multiplexer / de-multiplexer.

If the output lines are under multiplexed state then it should be de-multiplexed before it is used to drive the output peripheral loads. Else, if the output lines are not in multiplexed then it can be directly connected to the output peripheral device respectively.

Thus using an application with 4 different operations using simple led for load at output peripherals are explained.

VI. RESULTS AND DISCUSSION:

This paper proposes to create multitasking operation in Arduino using an algorithm named RTM-LUT which is a 7 step process. This algorithm saves more memory spaces when compared with other RTOS, this algorithm can be used for low memory Arduino boards and also to any microcontrollers. Thus this RTM-LUT algorithm is more efficient than other RTOS. This algorithm has been successfully implemented on the Arduino boards to get real time multitasking without any time slotting.

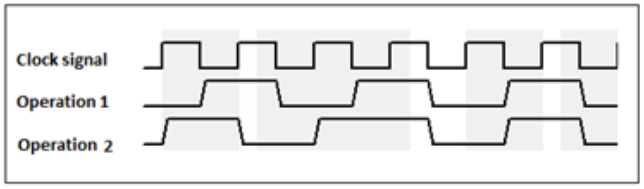
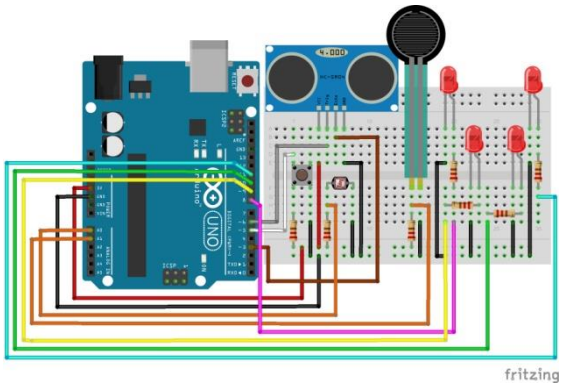


Fig.7: Timing graph for Multi-tasking using LUT algorithm.

VII. CONCLUSION

According to our paper, we proposed to create real time multi-tasking without an RTO'S, time scheduler and jitter time algorithms. Hence, the main advantage of using this RTM-LUT algorithm is more efficient then the above discussed algorithms. Therefore , the real time multi-tasking under single main loop obtain an result, instead of occupying more storage spaces in the microcontrollers are been saved . Hence this algorithm can be successfully suitable to implement it in the small scale microcontrollers like ATMEGA8.



REFERENCES

[1] G. Buttazzo. Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications, Third Edition. Springer, New York, 2011.

[2] P. Gai, G. Lipari, L. Abeni, M. di Natale, and E. Bini. Architecture for a portable open source real-time kernel environment. In Proceedings of the Second Real-Time Linux Workshop and Hand's on Real-Time Linux Tutorial, November 2000.

[3] C. Liu and J. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. Journal of the Association for Computing Machinery, 20(1):46–61, January 1973.

[4] L. Sha, R. Rajkumar, and J. Lehoczky. Priority inheritance protocols: An approach to real-time synchronization. IEEE Transactions on Computers, 39(9):1175–1185, September 1990.

[5] Pasquale Buonocunto, Alessandro Biondi. real-time operating systems for small microcontrollers. Tran nguyen bao anh, Singapore engineering centre for small microcontrollers. [1]T. Bonny,J. Henkel ,Efficient Code Density Through Look-up Table Compression Nice,Dept. of Comput. Sci., Karlsruhe Univ.

[6] M. Gotz, F. Dittmann, Reconfigurable Microkernel-based RTOS: Mechanisms and Methods for Run-Time Reconfiguration, Heinz Nixdorf Inst., Paderborn Univ.

[7] F. Vargas ,D. Silva,L. Bolzani,An intellectual property core to detect task schedulling-related faults in RTOS-based embedded systems Athens,Electr. Eng. Dept., Catholic Univ. - PUCRS, Porto Alegre, Brazil.

[8] Sungchul Lee, Juyeon Jo,Yoohwan Kim, Haroon Stephen, A Framework for Environmental Monitoring with Arduino-Based Sensors Using Restful Web Service

[9] Alexander,G.Dean, Efficient Real-Time Fine-Grained Concurrency on Low-Cost Microcontrollers, North Carolina State University.

[10]Taghi Mohamadi, Real Time Operating System for AVR microcontrollers, Iran University of Science and Technology (IUST), Tehran, Iran.