Vol 06, Article 12522; April 2015
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

DESIGN FLOW CYCLE AND SIMULATION OF LOAD MOVER

ASHISH MISHRA[1], ARSHAD JAVED[2], KARAN BHARGAVA[3], KAUSTUBH TANMANE[4], HARDIK SHAH[5]
[1,2,3,4,5]Department of Electrical and Electronics Engineering,
BITS-Pilani, Pilani, Rajasthan, India

ABSTRACT
In this paper we attempt to design an embedded device named as Load Mover (LM). The primary purpose of this LM will be to move objects from one place to another. LM will have its own intelligence to move automatically on a pre-defined track. The development of LM traverses the various stages in the embedded design cycle. The functionality has been verified using finite state machines (FSM) in Stateflow using Simulink. The experimental testing has been done by creating a device around arduino running on a created user grid.
Keywords: *FSM, robotics, Matlab, design cycle, embedded systems;*

1. INTRODUCTION

The field of embedded systems has grown at a rapid pace over the last decade, as is evident by the ever increasing demand for cellular phones, PDAs, digital camcorders, smart cards, and other intelligent portable devices. Embedded systems are designed to achieve specific applications and explore the best trade-off between hardware (HW) and software (SW)[1]. The design metrics should be efficient in terms of energy consumption, code-size, run-time efficiency, weight and cost [2]. Typical applications of embedded devices include vehicle control, communication systems, remote sensing, consumer electronics, Robotics and household appliances. Many such systems have been demonstrated till now to perform activities on behalf of humans for better performance and accuracy.

This paper describes the design flow of an embedded system called Load Mover (LM). A conceptual drawing of the device is shown in Figure 1. The primary purpose of this LM will be to move objects from one place to another as specified as an input by the user. The scenario kept in mind during the design cycle of the LM is typical of an airport or railway station where the luggage needs to carried over a large distance. Successful implementation of this product will ease the burden of all travellers especially lady travellers and those travelling with small children. LM has been designed with its own intelligence to move automatically on a pre-defined track. We explore the development of LM for best HW-SW trade-off by using finite state machine (FSM). The various stages in the design cycle are requirements, specification, design, testing and verification. We start from the requirement phase of LM and refine the design by justifying each stage of design cycle.

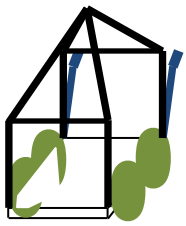

Figure 1.Conceptual design of the Load Mover

2. REQUIREMENTS

The requirements of an embedded system are usually specified in a natural language like English. This stage states what are the objectives to be met from the user perspective. The requirements for this LM are inspired from trolleys used in airports. This stage is crucial as it is the input to the subsequent stages. Better description of requirements leaves least margin for unsatisfactory product. The descriptions of LM requirement are as follows: 1. Weight Constraints: It should move maximum weight of 40 KGS and the total weight of the machine should not be more than 60 KGS 2. It can be operated manually, semi-automatically and automatically. In manual mode, it works as a trolley. In semiautomatic mode the user can enter the destination to move in different directions and in fully automatic mode, it can receive wireless command for destination .There is a predefined grid which has six points. These points are stored as a map of 2 x 3 points to move on (A, B, C, D, E, F).The map is shown in figure 1. We have assumed the strip is black and background is white for movement. It can detect hindrances and alarm them to move them to move away from the path. This map can be updated and it should follow the shortest path. It can take photo of each load and store it in memory. It can climb a slope of maximum 4 feet. The weight panel should be around 1.5 x 3 feet. It can talk to the customer and respond to

2010-2015 – IJVES

Vol 06, Article 12522; April 2015
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

fixed five questions. 3. Cost Constraints: INR 50000. 4. Energy Constraints: battery chargeable in intervals of 5 hours.

A		B		C
D		E		F

Figure 2.Grid Map for the path of Load Mover

3. SPECIFICATION

The requirements are refined in the specification phase. In this stage mathematical analysis of the design parameters and selection of components should be done accurately. The feasibility of the requirements and changes if any also should be reflected properly. The specifications can be functional and non-functional in nature. Functional specification includes time constraints which determine the functionality of the design. Non functional specifications include reliability, dimensions etc. In traditional design flow, the components are selected and design starts. But with the advent of large class of processor/system-on-chip-(SOC)/Field programmable gate arrays (FPGAs), the design space exploration becomes very large. Hence to make this step more comprehensible few questions can be raised. First question is what class of people are going to use the system? On this basis what are the I/O devices required can be answered. Second question is what the sub-systems in the design are. The recognition of sub-systems can be done on the orthogonality of the functionalities. From the specifications viewpoint, we have divided the design into following sub-systems1. Weight sensor interface (SEN-10245, 50Kg Max Wt) 2. Infrared Sensor interface 3.Camera Interface 4.Audio Section 5.Battery Charger (12 Volt & Amp Hour Alarm Battery) Section. 6. Motor Control Section: DC Motors provide high starting torque and good RPM values which would be the basic requirement of the LM. Usage of DC motors will eliminate the power supply converters as required in case of AC Motors with portable supply. Four motors can be used to obtain 90° turn by rotating both motors on one side in direction opposite to the other two motors.

System-on-Chip (SoC) have emerged as the design paradigm for designing scalable on-chip communication architectures, providing better structure and modularity. System-On-Chips are the common platform for audio camera interfaces and they also offer high computation ability. So the next step is to select a suitable SOC. Many SOCs are present in the market like OMAP series from TI, PSOC from Cypress, Blackfin from TI, Tricore from Infineon. We have chosen OMAP series as the platform as it has all the necessary interfaces required and the design can be simulated with beagle board. Next question to be answered is whether the design has real time features or not. If yes, do we require real time algorithms. The LM has the real time features, but it is not having critical features. So it has soft deadlines. Hence complex real time algorithms are not required.

The next point to be considered is whether the kernel is required, if yes which is the best kernel required. Since we have chosen the platform as Beagle board , hence Angstrom, Android, Linux all can be used as operating system. As the product is not user application based, so Linux will be a better choice. Since we have fixed Beagle board, memory choices are also fixed now. Next part is the decision of motors for cost effectiveness. Since the speed of the LM should be near to the human walking speed (4.5kmph) , so speed has been taken as 2 kmph. Following torque calculations are made for the selection of motor.

Total torque required,

$$Tt=Tf+Ta+Ts \qquad (1)$$

Where Tf is frictional torque, Ta= Torque to overcome the inertia of entire load mover, and Ts=Torque due to slope.

Calculation for Tf:

$$Tf=r\times\mu\times M \qquad (2)$$

r=radius wheel
μ=coefficient of friction
M=overall mass of load mover with pay load

Calculation for Ta:

$$Tar=Jt\times v\times N/(t\times r) \qquad (3)$$

$$Jt=Js+Jg/N2+Jw/N2 \qquad (4)$$

Jt=Total moment of inertia on reflected on motor side

Where Js, Jg, Jw is the moment of inertia of motor shaft, gears and wheel. Their values can be calculated by using the respective radius of gyration and masses. N is the gear ratio provided by the gearbox attached to the motor. 't' is the time required for the wheels to achieve a maximum velocity 'v'. For the inertia of masses having linear motion i.e. whole platform, pay load etc.

2010-2015 – IJVES

Vol 06, Article 12522; April 2015
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

$$Tal=M\times(v/t)\times r\times(1/N) \quad (5)$$

Thus the overall torque required to overcome inertia will be,

$$Ta=Tar+Tal \quad (6)$$

Calculation for Ts:

$$Ts=\mu\times M\times \sin\Theta \quad (7)$$

Above equations are used to compute the torque iteratively. Finally, four dc-geared motor of following specification is selected for the load mover.

100 RPM 12V DC motors with Gearbox
180gm weight
35kgcm torque
No-load current = 800 mA,
Load current = upto 7.5 A(Max)

For the selected motor, the specification of load mover is coming as,

Over all mass with pay load=60Kg
Max linear velocity =1.885 kmph
Torque required by each motor=12.15 kg-cm
Wheel radius= 5 cm
Wheel mass = 800 gm
Platform mass=450 gm

The required torque of the motor is coming as 12.15 kg-cm, however motors of 35 kg-cm are selected to ensure the proper functioning of load mover even in some worst condition.

4. DESIGN

There are three sections in which automatic behaviour can be divided: grid recognition, hindrance detection and shortest path recognition. All three require an algorithm for effective performance. The workplace shall have white floor with black grid for the LM to follow. There will be 3 Bump Switches / IR Sensors placed to detect the obstacle in front, left or right of the LM. The LM will use IR based 4 sensors (LIN1 to LIN4) for the line following process. The line follower logic has 4 IR based sensors beating at 38 KHz frequency whose output by reflection from white surface will be sensed by TSOP 1738 IR receiver. It is because this TSOP accepts only 38 KHz IR frequency and thus immunizes our system from the DC infrared signals present in the artificial/natural lights. Two sensors will be placed centrally (LIN1 and LIN2) with respect to the vertical axis of the LM to sense the presence of the Black Line of the Grid. Two other sensors (LIN3 and LIN4) will be placed spaced with respect to the central sensors to detect the horizontal lines of a grid as the LM moves as it can help the robot assess the arrival of intersection on the grid.

The obstacle avoider logic will check for any object present in front of the LM, if not, it will continue to reach the co-ordinate that it was set to reach in the automatic mode. However, if any object arrives in front , then it will go to the previous intersection point, take a 90° turn left/right depending on which direction gives lesser distance by checking the magnitude of the co-ordinate set by the user.

The workspace consists of white background with black grid on it. Each intersection on the grid will be associated with a (X, Y) co-ordinate. The central intersection on workspace will have co-ordinates (0, 0) becoming the centre of 4 quadrants in the workplace. The user will set the Co-ordinates of the destination and the current position of LM via a 4 X 4 keypad connected to the SOC. Also, the user can fix the path by storing 6 Co-ordinates of the grid intersections along with LM's current position. Accordingly the SOC logic will identify the quadrant and then move the LM by counting the number of intersections it crosses using the Line-Follower and Obstacle Avoider Logic passing through all those intersections if no objects are present in the path.

The LM will have information about maximum number of intersection points available in all 4 directions. The MCU will have SD-MMC memory to store the co-ordinates of 6 intersections which the LM has to cross and will also store the co-ordinates of the destination and LM's current position too, in the automatic mode. Figure 3 displays the physical positioning of sensors in order to enable proper detection of obstacles using L, R, M and to detect line/grid using LIN1 to LIN4. Table 1 displays the commands used to formulate the directions for motor control. Table 2 uses actions from Table 1 to generate final code word to control the four motors. Table 3 indicates the motor code words used in design.

2010-2015 – IJVES

Vol 06, Article 12522; April 2015
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

Table 1: Motor Control Signals

T1	T2	Action
0	0	Stop
0	1	Reverse
1	0	Forward
1	1	Stop

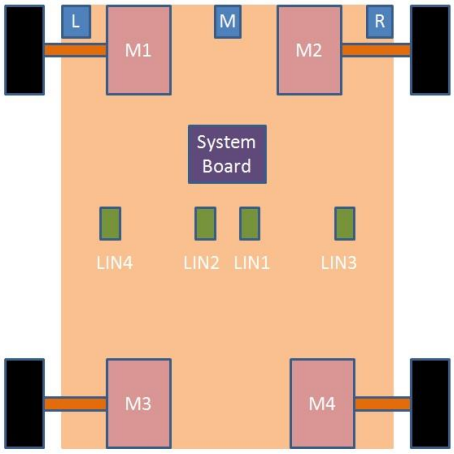

Figure 3. Physical positioning of sensors on LM

Table 2: Motor Codeword Design

M1 T1	M1 T2	M2 T1	M2 T2	M3 T1	M3 T2	M4 T1	M4 T2

Table 3: Motor Code-Words

Direction	Hex
Forward	0xAA
Back	0x55
Left	0x66
Right	0x99
Stop	0x00

The flow chart shown in Figure 4 describes the algorithm used for load movement. The flowchart gives a basic view of the embedded Matlab function that is used for obstacle avoiding and line following. It takes input as the co-ordinates from the earlier state and checks whether the source co-ordinates are equal with respect to destination co-ordinates. If yes, then it stops otherwise, it tries to run in a loop till source (src) and destination (dest) are same. Once it determines that src and dest are not equal, then it scans the obstacle avoiding sensors (L, M, R).

If they sense an obstacle in front of the Load-mover they then try to find an alternate path to destination else the Load mover remains oriented in the same direction. Once this is checked, the load mover scans the line follower sensors. If the load mover is not properly aligned to the line, then the LM aligns itself automatically over the line else it continues to move forward. Now when the LM is moving forward, it keeps on counting the grid-intersections to see whether it has encountered the total no. of intersections or not, if yes then it stops else it continues to run in loop till it reaches destination.

We choose to develop the finite state machine for the algorithm developed since FSM are easier and better way to capture the process flow. There are various tools for the development of FSM like Statemate, Stateflow, Yakindu etc. The extension of FSM are Statecharts [5]. Statecharts are combination of state-diagrams + depth + orthogonality + broadcast mechanism. We have chosen the State flow [6] in Simulink as our platform for implementing the state chart.

The flowchart given in Figure 5 below shows the software design for LM .The system is initialized and three modes of operation are checked. The device then checks the load and then enters the destination. The path destination algorithm then runs. Initially, the device is in the off state. On receiving the start input, it goes into the ON superstate. It enters into the mode select state where AMbar is (Automatic/manual control), since this machine is about automatic control, when AMbar is '1', the machine goes into the reading co-ordinate state

2010-2015 – IJVES

Vol 06, Article 12522; April 2015
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

where it reads the values from the user for source as well as destination. After that, it checks the weight of the Load-Mover, if it is lesser than 40 Kg, the system goes to snap state and takes a photo of the load and goes into the obstacle and line following mode. The code has been developed in Stateflow

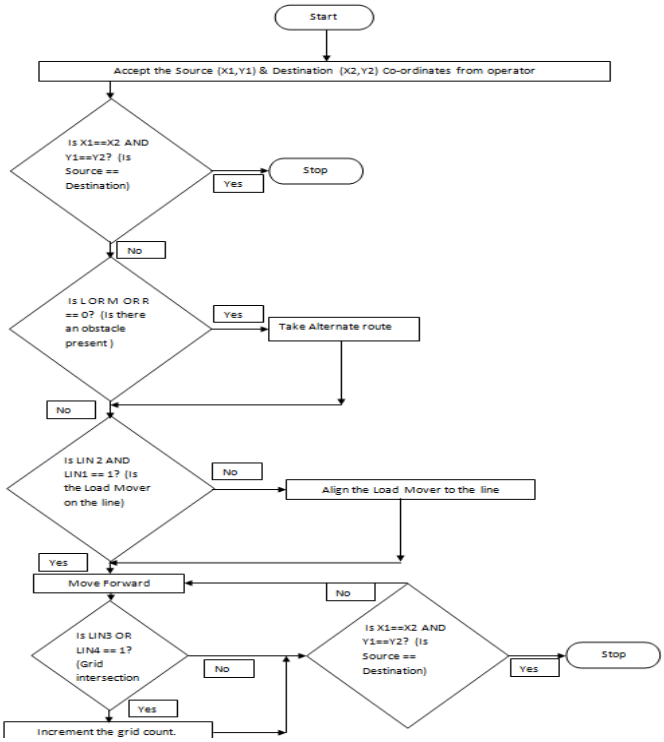

Figure 4.Algorithm used for load movement

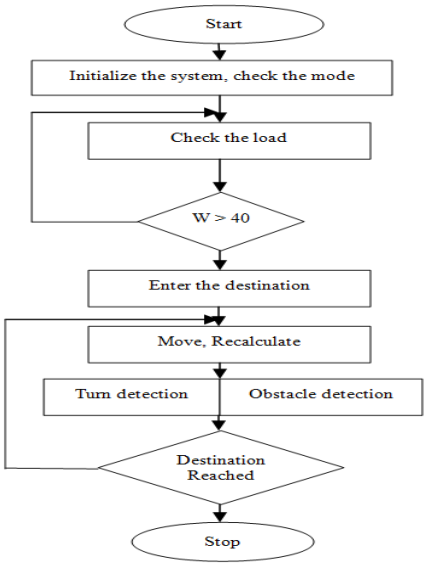

Figure 5 .The software design for LM

5. RESULTS

The simulation of the LM has been verified in Simulink. The code can be generated in C or HDL using the HDL codes. The implementation can be carried out in FPGA or can be compiled to a processor depending on the requirement. The algorithm was written in SystemC to check the functionality. The result given below show the path followed for a given co-ordinates.

Enter source and destination co-ordinates as per your wish
L=1 , M=1 ,R=1 are conditions for obstacle absent any other conditions are obstalce present
Enter the value of Source Co-ordinate x1 0
Enter the value of Source Co-ordinate y1 0
Enter the value of Destination Co-ordinate x2 10
Enter the value of Destination Co-ordinate y2 10

2010-2015 – IJVES

Vol 06, Article 12522; April 2015
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

Enter the value of left L 1
Enter the value of middle M 1
Enter the value of righty R 1
load mover started10 ms to detect obstacle there or not As per your L , M , R values Obstacle Absent
Now the time is 10 ms
The path which is taken by load mover is it will cover x distance first and then y distance
The co-ordinates entered by you are source co-ordinates x1 =0 y1=0 Destination co-ordinates x2 =10 y2=10
The x1 co-ordinate is 0 and x2 co-ordinate is 10 since x2 is greater than x1 it takes right turn
Turning Right, Motor Code0x99
Now the time is 10 ms
one right turn so 5 ms delay
time is 15 ms
Stopping, Motor Code0x00

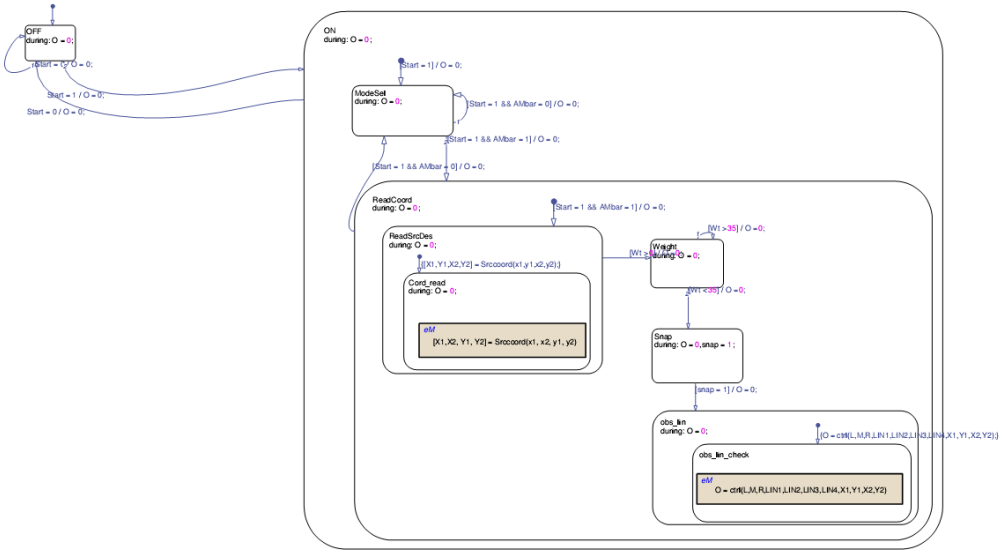

Figure 6 FSM design using Stateflow in Simulink Matlab

In order to test the algorithm a simpler hardware model was used which is shown below with following Microcontroller : Rapid prototyping platform Arduino, Motor Driver : L293D IC, Motors : Metal geared micro motors (30:1) gear ratio, 440 rpm at 6 V, Li-ion battery 2S - 7.4 V 1500 mAh, Infrared sensor array,Proximity sensor, Track (Black track on white surface) made by using electrical tape of 2mm.

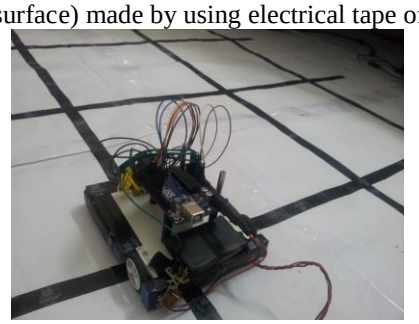

Fig. 7 Model and Arduino kit

REFERENCES

[1] Raj Kamal "Embedded Systems",Tata McGraw-Hill Education (2008)
[2] Peter Marwedel , Embedded system Design, Springer, Kluwer Academic Publishers, 2003.
[3] J. Straunstrup, W. Wolf ,Hardware/Software Co-design Principles and Practices. Springer, Kluwer Academic Publishers, 1993
[4] W. Wolf, "A Decade of Hardware/Software Co-Design, "in Proc. of the 5th International Symposium on Multimedia Software Engineering (MSE2003), Dec.2003, pp.38–43
[5] D. Harel, "Statecharts: A visual formalism for complex systems" Sci comput. prog. vol. 8, pp. 231-274, 1987.
[6] http://www.mathworks.in/products/stateflow/ last accessed on 15 Octber 2014.

2010-2015 – IJVES