PROGRAMMING AND CONSTRUCTION OF AHMEDULLAHBOT-A FAST GRID-SOLVING ROBOT

Ahmedullah Aziz, Md. Shafayat Hossain and Mohammad Wahidur Rahman

Department of Electrical and Electronic Engineering, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh
rumi3.1416@gmail.com

ABSTRACT

This paper presents AHMEDULLAHBOT which is a 4DOF self-driven vehicle capable of localizing itself in a grid and planning a path between two nodes. It can avoid particular nodes and plan path between two allowed nodes. Breadth-first search & Dijkstra's Algorithm have been used for finding the path between two allowed nodes. The vehicle is also capable of transferring blocks from one node to another. In fact, this vehicle is a prototype of a self-driven vehicle capable of transporting passengers and it can also be used in industries to transfer different items from one place to another.

KEYWORDS

Breadth-first Search, Dijkstra's Algorithm, DOF, Microcontroller, Path planning

1. INTRODUCTION

A key ability needed by an autonomous, mobile robot is the possibility to navigate through the space [1]-[2]. The problem can basically be decomposed into positioning and path planning. Especially if the robot is severely resource-constrained, simple schemes are favorable to elaborated algorithms. Rather simple sensors and actuators as well as a limited computing platform also demand simple, robust techniques due to inaccuracy and the lack of resources. Autonomous navigation of mobile robot has been appeared in literature many times [3]-[7].
AHMEDULLAHBOT is a 4DOF autonomous mobile robot capable of exploring 2D world, finding shortest path between source and destination. It can also transfer objects of different sizes and shapes from one place to another.

The paper is structured in following way: At first the nature of the grid and programming technique required to solve it is discussed and in the next section the mechanical construction is detailed.

2. GRID DESCRIPTION

The grid used as a reference is a 6×6 square grid having 270mm×270mm inner dimension of 30mm thick white lines on a black surface. The lines are spaced equally. There are nodes at the intersection of white lines at some places and the nodes are black squares of 30mm×30mm dimension (Fig.1).

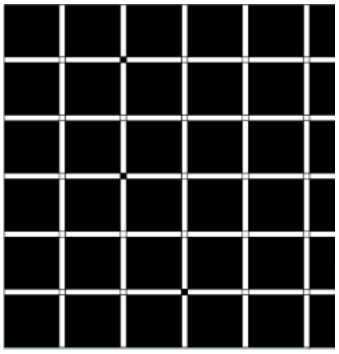


Figure 1. Grid

3. SOLVING ALGORITHM

The grid is represented in a 2D system. Starting node is assumed as the origin. During the first scan of the grid, the robot comes across every crossing in the grid and finds out whether there is a black node in it.

All the four directions are earmarked in which the bot can move so as to know to where the bot is going to move next. Referring the 2d system in geological directions NEWS, we will take NORTH or +y axis as 1 and the following directions as 2, 3 and 4. In this way the robot can be guided precisely to the destination node. In the code a direction counter is kept for this purpose. The value of the counter is changed whenever needed.

For graph-traversal and finding shortest path, at first Breadth-First Search (BFS) has been used and then we have moved on to Dijkstra's Algorithm [8] for lowering the time-complexity.

3.1. Breadth-First Search

Breadth-First Search is a simple technique for graph-traversal which has been used primarily for finding path with the minimum number of edges between two given vertices in the grid. BFS on a grid having n vertices and m edges takes O(n+m) time [9]. Pseudocode for BFS is shown in Table 1.

Breadth first search is complete as long as the branching factor is finite. This means that the algorithm will always return a solution if a solution exists. In a path planning sense, breadth first search will always find the path from the starting position to the goal as long as there is an actual path that can be found. Breadth first search is only optimal if the path cost is the same for each direction. In this case, the path cost would be the direction and optimal means the shortest distance path from the start to the goal. Every node generated must remain in memory and the number of nodes generated is at most O(7^(d+1)) where 7 represents the maximum branching factor for each node and d is the depth one must expand to reach the goal. For very large workspace where the goal is deep within the workspace, the number of nodes could expand exponentially and demand a very large memory requirement.

Table 1. Pseudocode for Breadth-First Search

Algorithm BFS(G)	**Algorithm *BFS*(*G, s*)**
Input graph G **Output** labeling of the edges and partition of the vertices of G **for all** u ∈ G.vertices() setLabel(u, UNEXPLORED) **for all** e ∈ G.edges() setLabel(e, UNEXPLORED) **for all** v ∈ G.vertices() **if** getLabel(v) = UNEXPLORED BFS(G, v)	$L_0 \leftarrow$ new empty sequence L_0.*insertLast*(*s*) *setLabel*(*s,VISITED*) $i \leftarrow 0$ **while** ¬L_i.*isEmpty*() $\boldsymbol{L_{i+1}} \leftarrow$ new empty sequence **for all** $v \in L_i$.*elements*() **for all** $\boldsymbol{e} \in G$.*incidentEdges*(v) **if** *getLabel*(e) = *UNEXPLORED* $w \leftarrow$ *opposite*(v,e) **if** *getLabel*(w) = *UNEXPLORED* *setLabel*(*e, DISCOVERY*) *setLabel*(*w,VISITED*) $\boldsymbol{L_{i+1}}$.*insertLast*(w) **else** *setLabel*(*e,CROSS*) $i \leftarrow i + 1$

3.2. Dijkstra's Algorithm

For a given source vertex (node) in the graph, the algorithm finds the path with lowest cost (i.e. the shortest path) between that vertex and every other vertex. It can also be used for finding costs of shortest paths from a single vertex to a single destination vertex by stopping the algorithm once the shortest path to the destination vertex has been determined [10]. A min-priority queue based implementation by a Fibonacci heap has been used running in O (m+|n|log|n|) which is asymptotically fastest shortest-path algorithm [11]-[12]. Pseudocode for Dijkstra's Algorithm is shown in Table 2.

Table 2. Pseudocode for Dijkstra's Algorithm

```
function Dijkstra(Graph, source):

for each vertex v in Graph:  // Initializations

dist[v] := infinity ;  // Unknown distance function from source to v

previous[v] := undefined ;  // Previous node in optimal path from source

end for

dist[source] := 0 ;  // Distance from source to source

Q := the set of all nodes in Graph ;

// All nodes in the graph are not optimized - thus are in Q


while Q is not empty:      // The main loop

u := vertex in Q with smallest distance in dist[] ;

// Start node in first case

remove u from Q ;

if dist[u] = infinity:

break ;

// all remaining vertices are inaccessible from source

end if

for each neighbor v of u: // where v has not yet been removed from Q.

alt := dist[u] + dist_between(u, v) ;

if alt < dist[v]:          // Relax (u,v,a)

dist[v] := alt ;

previous[v] := u ;

decrease-key v in Q;  // Reorder v in the Queue

end if

end for

end while
```

return dist;

if u = target

S := empty sequence

u := *target*

while previous[*u*] is defined:

// Construct the shortest path with a stack S

insert *u* at the beginning of *S*

// Push the vertex into the stack

u := previous[*u*] *// Traverse from target to source*

end while

The algorithm could be more efficient if A^* search could have been used which is too complex to implement within system-resource.

4. HARDWARE DEVELOPMENT

4.1. Structure Design

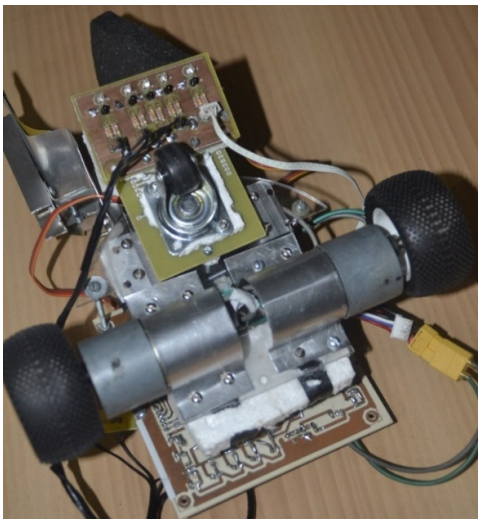


Figure 2. Bottom view of the robot showing the line-sensors and wheel arrangement

The chassis of the robot is built with light-weight but durable Acrylic and glass-fiber. The wheel are coupled with two different but identical DC motors for differential drive (Fig.2). For

maintaining proper weight-balance, an omnidirectional wheel is used in the front. Glass-fiber PCB boards are used for maintaining light-weight. Rubber-wheels with spikes ensure smooth movement in the rough surface. Line sensor and wheel arrangement are shown in Fig.2.

The hand is constructed using Aluminum sheet and pipe to ensure light-weight, strength and durability. The hand has 2 DOF powered by two servo-motors (Fig.3).

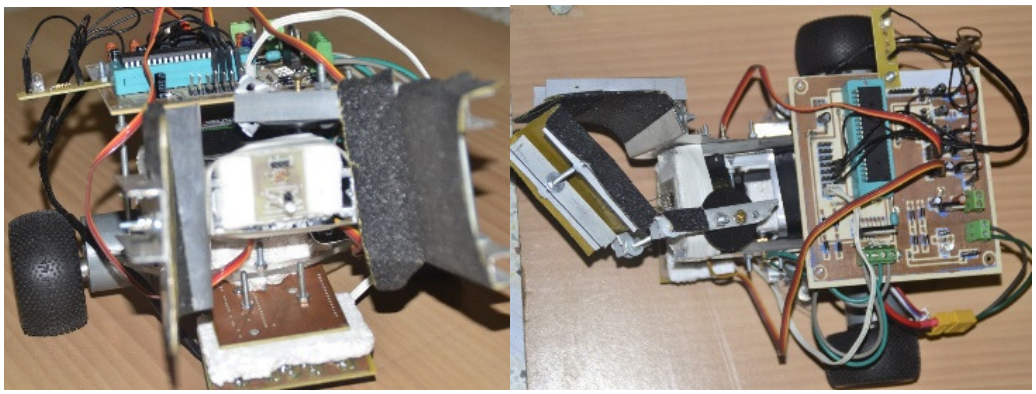


Figure 3. DOF hand showing different movements

4.2. Arrangement of the sensors for line-tracking and crossing-detection

The line sensor are arranged in such a way that the middle 3 sensors are equally apart (0.35 inch) and the side sensors are 0.55 inch apart from their nearest sensor pairs. The PCB layout of the line-sensor arrangement (Fig.4) clarifies the arrangement pattern.

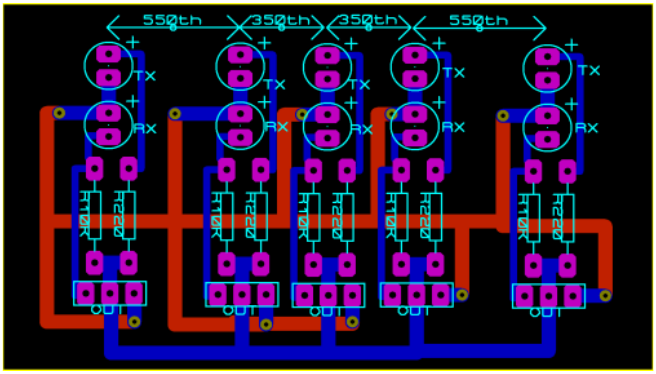


Figure 4. Line sensor arrangement

Fig.5 shows the sensor arrangement for following line and cross detection. Fig.6 shows the different scenario arising in case of line tracking and actuator action according to the relative position of the line-sensors with respect to the white line. Detection and counting process of the crossings are demonstrated in Fig.7.

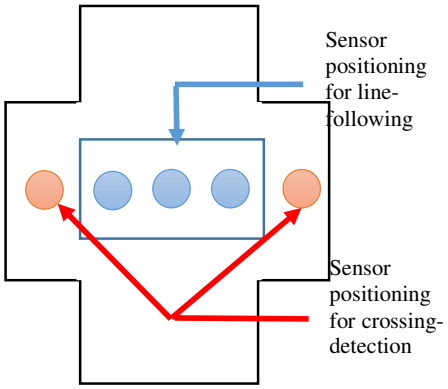


Figure 5. Sensor arrangement

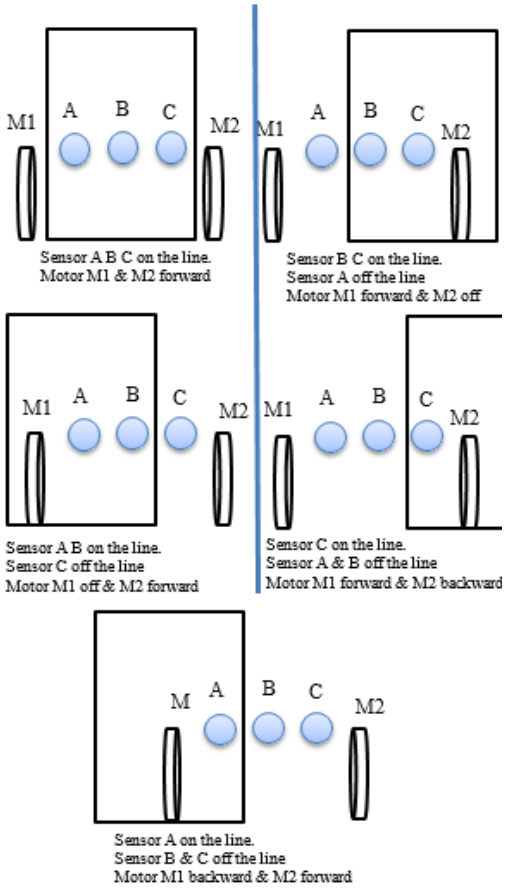


Figure 6. Line following mechanism

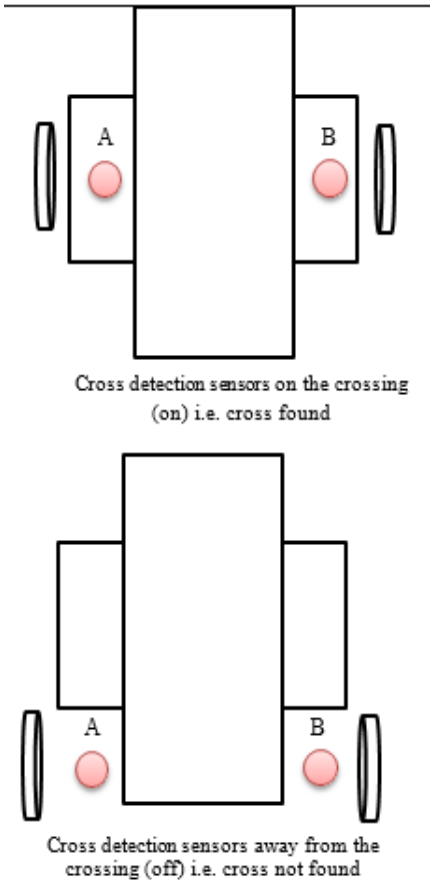


Figure 7. Cross-counting mechanism

5. CIRCUIT DESIGN

5.1. DC Motor Drive

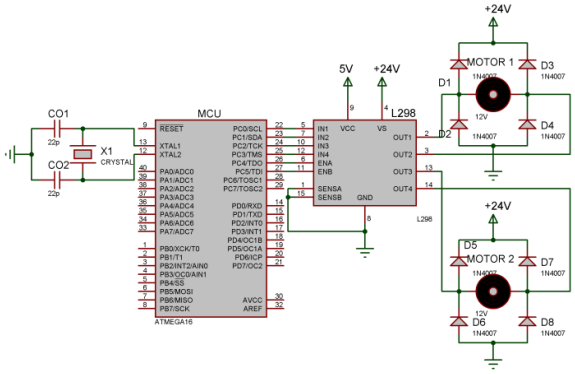


Figure 8. Circuit Diagram for DC-motor control using L-298

For high voltage and high current drive of the DC-motors from ATMEGA16 microcontroller [13], L-298 IC [14] has been used which is a dual-bridge controller for motor drive and can be controlled by sending PWM from the microcontroller into its Enable pin. It supports bi-directional motor-drive with about 46 volt and 3.5 Ampere. Diode-protection using 1N4007 [15] has been deployed for protecting the motor driver from back electromotive force. The necessary circuit to drive DC motor is shown in Fig.8.

5.2. Servo motor Control

Servo-motors have been used to construct the hand. Low-torque (8 kg-cm), 3-pole, plastic-gear, dual-bearing TowerPro SG-5010 - Standard Servo motors [16] have been used. For light-weight finger movements low-torque (1.8 kg-cm) TowerPro SG91R - Micro Servo motors [17] have been used. They are shown in Fig. 9.

(a) (b)

Figure 9. Servo-motors (a) SG-5010 (b) SG 91R

Servo-motors require approximately 5V and 500mA supply each. So, linear voltage regulator IC 7805 [18] has been used for constant 5V supply, to power-up the 2 servo-motors used in this robot. In normal operation of 7805, it provides only 500mA current. So, two 7805s are used for driving two servo-motors.

Servo-motors are controlled by pulse from microcontroller. A signal of 20ms period is sent continuously. For 0 degree position the duty period of the pulse is 1.5ms. For -45 degree and +45 degree the duty periods are 1ms and 2ms respectively.

2.5. Line sensor Interfacing

Short range IR transmitter-receiver pair (Fig.11) is used as line sensor. Interfacing pair shown in Fig.12 with the external logic circuitry has been mentioned in [19]. The same circuitry (Fig.11) is used here. In case of white lines, the output is high and for the black surface (out of the line), the output is low.

(a) (b)

Figure 10. IR transmitter-receiver pair (a) transmitter (b) receiver

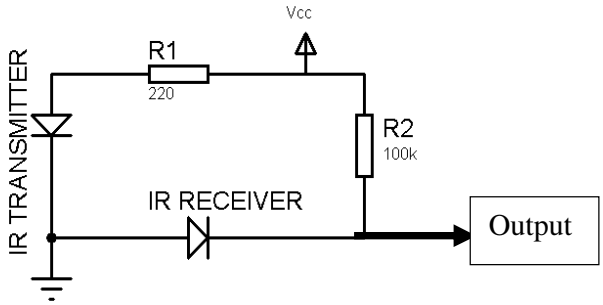


Figure 11. Circuit diagram for IR transmitter-receiver

2.6. Overall Electrical System

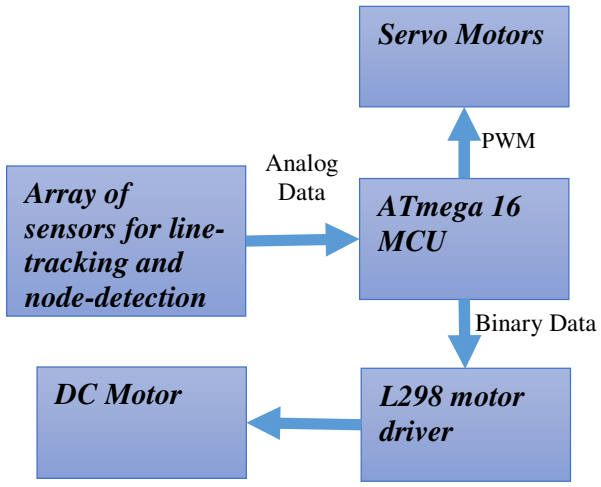
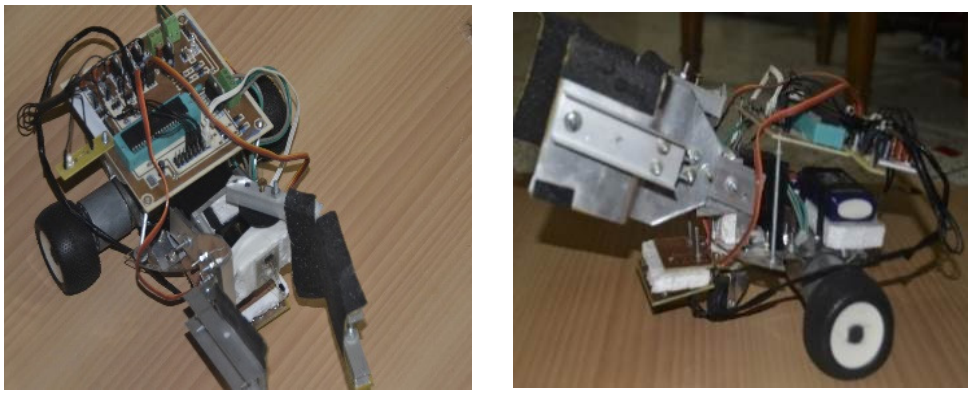


Figure 12. Block diagram of the total electrical system

The complete system is shown in a block diagram in Fig.12.

6. SYSTEM PROTOTYPE

The complete structure of the grid solving robot is shown in Fig.13.



(a) (b)

Figure13. The grid-solving robot (a) isometric view (b) side view

7. FUTURE WORK

This robot is run by an 8 bit microcontroller with limited processing power. In future, FPGA will be used for robust processing and control. A* search will be implemented for robust grid-solving.

8. CONCLUSIONS

AHMEDULLAHBOT is an autonomous mobile robot capable of localizing itself in a 2D world and planning its path towards destination by means of BFS & Dijkstra's Algorithm implemented in ATmega16 microcontroller. It can follow the tracks using feedback from the line-sensors and also capable of detecting crossings. This 4DOF robot has 2DOF hand for grabbing objects lying on its path which enables it to serve in industries for specific object-fetching and gathering. A large scale prototype customized for factories will be very promising.

REFERENCES

[1] J. Borenstein, B. Everett, and L. Feng. Navigating Mobile Robots: Systems and Techniques. A. K. Peters, Ltd., Wellesley, MA, 1996.
[2] I.J. Cox and G.T. Wilfong, editors. Autonomous Robot Vehicles. Springer,Verlag, 1990.
[3] Joachim Buhmann, Wolfram Burgard, Armin B. Cremers, Dieter Fox, Thomas Hofmann, Frank Schneider, Jiannis Strikos, and Sebastian Thrun. The mobile robot RHINO. AI Magazine, 16(2):31{38, Summer 1995.
[4] Wolfram Burgard, Dieter Fox, and Sebastian Thrun. Active mobile robot localization. In Proc. of the Fifteenth International ConferenceonArtificial Intelligence (IJCAI-97), 1997.
[5] Joachim Hertzberg and Frank Kirchner. Landmark-based autonomous navigation in sewerage pipes. In Proceedings of the First Euromicro Workshop on Advanced Mobile Robots (EUROMICRO '96), pages 68{73. IEEE Computer Society Press, 1996.
[6] Hans P. Moravec. Sensor fusion in certainty grids for mobile robots. AI Magazine, pages 61{74, Summer 1988.

[7] Illa Nourbakhsh, Rob Powers, and Stan Birch_eld. DERVISH an office-navigating robot. AI Magazine, 16(2):53{60, Summer 1995.
[8] Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs". Numerische Mathematik 1: 269–271. doi:10.1007/BF01386390. http://www-m3.ma.tum.de/twiki/pub/MN0506/WebHome/dijkstra.pdf.
[9] Knuth, Donald E. (1997), The Art Of Computer Programming Vol 1. 3rd ed., Boston: Addison-Wesley, ISBN 0-201-89683-4, http://www-cs-faculty.stanford.edu/~knuth/taocp.html
[10] Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2001). "Section 24.3: Dijkstra's algorithm". Introduction to Algorithms (Second Ed.). MIT Press and McGraw-Hill. pp. 595–601. ISBN 0-262-03293-7.
[11] Fredman, Michael Lawrence; Tarjan, Robert E. (1984). "Fibonacci heaps and their uses in improved network optimization algorithms". 25th Annual Symposium on Foundations of Computer Science (IEEE): 338–346.
[12] Fredman, Michael Lawrence; Tarjan, Robert E. (1987). "Fibonacci heaps and their uses in improved network optimization algorithms". Journal of the Association for Computing Machinery 34 (3): 596–615.
[13] Atmel. (2010, Oct. 20). "ATMEGA 16 datasheet." [On-line]. Pp. 1-356. Available: www.atmel.com/Images/doc2466.pdf [Sept. 1, 2012].
[14] STMicroelectronics. (2000, January). "L298 Dual full-bridge driver". [On-line], pp.1-12. Available: noel.feld.cvut.cz/hw/st/1918.pdf. [September 2, 2012].
[15] Fairchild Semiconductor. (2009, May). "1N4001-4007 General Purpose Rectifiers". [On-line]. pp. 1-3. Available: www.fairchildsemi.com/ds/1N/1N4001.pdf [September 2, 2012].
[16] TowerPro "TowerPro SG-5010 Servo Specifications and Reviews" [On-line] Available: www.servodatabase.com/servo/towerpro/sg-5010 [October 3, 2012].
[17] TowerPro "TowerPro SG91R Servo Specifications and Reviews" [On-line] Available: www.servodatabase.com/servo/towerpro/sg91r [October 3, 2012].
[18] National Semiconductor (2000, May 2). "7805 datasheet". [On-line]. pp. 1-3. Available: pira.cz/pdf/78xx.pdf. [September 1, 2011].
[19] Aziz, A.; Hossain, M.S.; "Inherent Inter-vehicle Signaling Using Radio Frequency and Infra-red Communication," Computer Modelling and Simulation (UKSim), 2012 vol., no., pp.211-215, 28-30 March 2012, doi: 10.1109/UKSim.2012.

AUTHOR

Ahmedullah Aziz was born in Dhaka, Bangladesh. He is currently an undergraduate student of Bangladesh University of Engineering and Technology (BUET) in department of Electrical and Electronic Engineering. His current research interests include Robotics, Novel semiconductor based thin film characterization, embedded system design etc.

Md. Shafayat Hossain was born in Dhaka, Bangladesh. He is currently pursuing his B.Sc. degree in electrical and electronic engineering from Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh. His research interests include analytical modeling & simulat ion of Nano-scale electron device, Robotics, computer vision and embedded system design.