MRL Extended Team Description 2013

Amin Ganjali Poudeh, Siavash Asadi Dastjerdi, Saeid Esmaeelpourfard, Hadi Beik Mohammadi, and Aras Adhami-Mirhosseini

Islamic Azad University of Qazvin, Electrical Engineering and Computer Science Department, Mechatronics Research Lab, Qazvin, Iran
a.adhami@ece.ut.ac.ir

Abstract. MRL Small Size Soccer team, with more than five years experience, is planning to participate in 2013 world games. In this paper, we present a more detailed overview of MRL small size team and especially major contributions at this year. After attaining acceptable performance to reach the third place in 2010 and 2011 competitions and achieving reliable robots in 2012, we focuse on the electrical and software parts. The main electrical board and charger board are redsigned. Finalizing our restructed decision systems, improving robot's motion and moving to more dynamic and learnning/adaptive based methods this year are the major rethis year, aided the team to reach at his highest level when participates in 2013 world games.

1 Introduction

MRL team started working on small size robot From 2008 and after three years we could qualified to be in semifinal round and attaining the third place in 2010 and 2011 RoboCups. This means that our last years plan was achieved. The main problem in MRL robots in 2011 competitions was its unreliable behavior. Mainly, slow and inaccurate motion together with software problems gave opportunities to the opponents to score. These problem were solved for 2012 RoboCups. The results of 2012 competitions illustrates the team required more felixiable strategies to creat opportunities agaist different teams. In general the MRL team shows a good and felixable defending tactics, but not in attacking strategies. Main target in this year plan is resolving minor harware problems, improvement on the electrical structure and moving toward the reliable and more dynamic AI mechanism.

Some requirements to reach the more fast and accurate motion are satisfied with electrical restructuring. Adaptive methods in control are employed and evaluated by software tools like online debugging tools and simulator, for more details see [1]. Iran open 2013 was an opportunity to evaluate our new contributions. Although, the modified robots are faster than the old ones, there are still some problems that should be solved to reach more reliable robots.

This paper is organized as follows: Firstly, our software architecture including game planner, strategy manger and high-level control strategies are described in section 2. The modified electrical design and new hardware architecture, are

explained in section 3. A short description of mechanical configuration of the robots, which elevates the capabilities of the robots' smooth and reliable motion is the subject of section 4.

2 Software

In this part the software main objects are presented. It is shown that how our new architecture provides us a safe and flexible game. In this year MRL software team has changed the AI structure and built up a new architecture. The new game planner as the core unit for dynamic play and strategy manager layer are introduced in this section. After these major changes, minor modifications on the other parts like visualizing systems are presented. Following paragraph, sketch an overview to the MRL software modules and their connections.

The software system is consisted of two modules, AI and Visualizer. The AI module has three sub-modules being executed parallel with each other: Planner, STP Software and Strategy Manager. The planner is responsible for sending all the required information to each section. The Visualizer module has to visualize each of these sub-modules and the corresponding inputs and outputs. The visualizer also provides an interface for online debugging of the hardware. Considering the vision subsystem as an independent module, the merger and tracker system merges the vision data and tracks the objects and estimates the world model by Kalman Filtering of the system delay. Figure 1 displays the relations between different parts. In this diagram, an instance of a play with its hierarchy to manage other required modules are depicted. The system simulator is placed between inputs and outputs and simulates the entire environments behavior and features. It also gets the simulated data of SSL Vision as an input and proceeds with the simulation. Last year we add a new feature to our simulator that uses the kinematic modeling of robot motions. In the following subsections we introduce each layer of the AI mechanism. Note that, the arrangement of the introduced layers is to increase tractability.

2.1 Role assignment

MRL AI architecture is based on the STP platform introduced by James Bruce in [2]. Our game strategy aims at dynamic assignment of some pre-defined roles to the robots in the field. The dynamic assignment tries to assign the best role to each robot at each time frame, making the strategy more flexible to the new events and positions in the field.

Dynamic role assignment is much more complex than static role matching. Therefore a new role matching algorithm was required to effectively assign roles to the robots at each time frame. Our algorithm addresses the role matching problem by considering two different states for each robot at each time frame; whether it stays at its previous role or switches to a new, overall more efficient role. The algorithm requires taking into account the set of all robots and all roles and should select the highest overall priority for robot-role matching. A

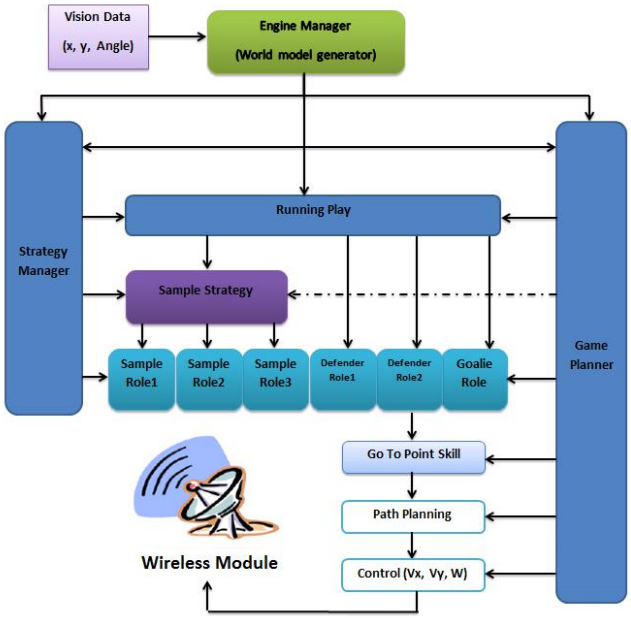

Fig. 1. Block diagram of AI structure

cost function for each role has been designed and each frame cost of a role for a specific robot has been calculated. Also a *SwitchTo* function has been embedded for each role. This function in each role specified roles that this role allowed to switch to them. We have considered this function to avoid some bad switches specially switches from defense roles to active role, For example by this function in dangerous situation like when ball and opponent are both near the danger zone some defense roles not allowed to switch to active role. Each role must have permission to switch to itself. After calling cost and *SwitchTo* functions for each role we map the intended problem into the bipartite graph. Then we try to find maximum flows in the matrix.

2.2 Strategy management

In last year we introduce a new layer of MRL AI hierarchy, the Strategy Layer. In the strategy layer, the AI system learns to select the best game strategy for some specific time frames. Each strategy is a heuristic game playing for certain number of attendees. "Field region","game status" and"minimum score to be activate" are parameters pertaining to each strategy. For instance, *Sweep and Kick* strategy with three attendees works the best in the middle of the field is activated after score one, and requires"Indirect Free Kick" game status. If all the four parameters are satisfied, the strategy becomes "applicable" at certain time frame. We model each strategy as a Finite State Machine (FSM). Consecutive

states of strategy FSM indicate the chain of actions required to be performed in that strategy. The transition conditions between states reflect the prerequisite conditions for the actions. The FSM has got an initial state with which the "applicability" is verified. It also has got Trap and Finish states indicating "failed" and "successful" ending of the strategy, respectively. A dynamic score is designed for each strategy. After completion of each strategy (either failed or successful), the strategy score is updated also strategy score will updated with result of game analysis from game planner and opponent defense analyzer during the game.

Strategy manager operates as the highest component of the Strategy Layer. This component is responsible for selecting the best strategy at each time frame. The strategy manager has got three different selection policies:

1. **Random Selection**: The manager randomly selects one of the applicable strategies.
2. **Higher Score with a Probability of Random Selection**: The manager tends to select the strategy with the highest score as of now, trying to apply the best strategy which has proved to have the best performance. Also, for the sake of giving the chance to some lower scored strategies to make progress, the manager randomly selects a strategy with probability of P.
3. **Weighted Random Selection**: The manager randomly selects one of the strategies, each of which has a weight corresponding to the probability to be selected.

The Strategy Manager selects one of the applicable strategies in one of the three mentioned ways and the attendee robots are assigned roles for performing the strategy. When the strategy traps or successfully ends, robots are reassigned roles for the normal play. The strategy layer helps us to avoid a share data or blackboard for agents. Therefore we can design a cooperative game of agents, dynamically.

PassShoot strategy Simple pass and shoot is one of the most common strategies in SSL because of its high execution speed. On the other hand due to the improvement of SSL teams in marking and defense tactics, it is very hard to achieve success with simple pass and shoot strategy with two attended robots. The proposed pass and shoot strategy has been implemented with three robots, a passer and two positioners, figure 2. Moreover, to increase the chance of success, the strategy has been implemented in a dynamic way as much as possible. In this strategy one of the positioners is selected to get the pass. The selection is made at the last moment of the passing state of the strategy. It depends on situations (position, clearance and ...) of the two positioners. Because of this type of selection, it is not clear witch robots shoot the ball. In order to make the strategy dynamic, we use a synchronizer module. This module synchronizes passer with the positioners in the way that pass point has been reached by the ball and the selected positioner simultaneously. The synchronization method considers the conflicts and obstacles. This module calculates the wasted time and then make

up this time by changing operations time line. Thanks to the synchronizer, it is not necessary for the shooter robot to be in the pass point from the beginning. This increases the chance of success. The pass point is determined by the game planner module using a grid based algorithm according to some parameters like goal view angle, opponents density and in different parts of field. Type of pass (direct or chip) depends on the obstacles in the way of pass point. It can change until kicking the ball.

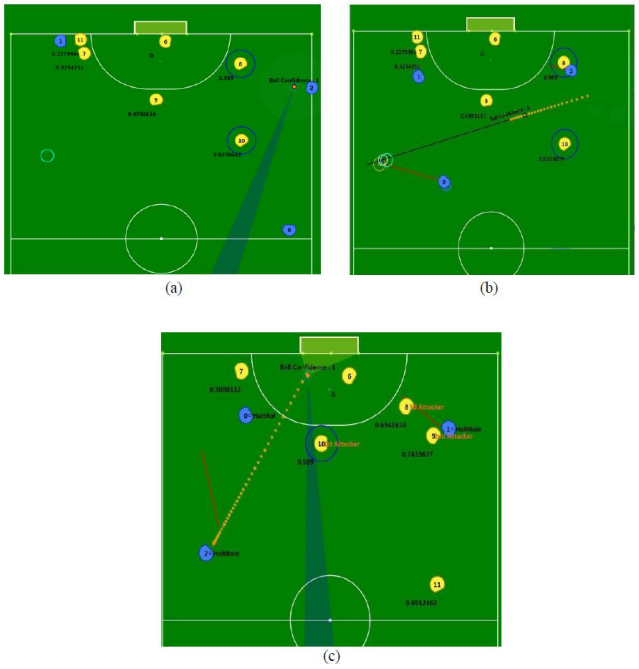

Fig. 2. Pass shoot strategy: (a) Initial state: positioning and pass point (aqua circle) selection (b) Pass state, (c) Final state.

2.3 Game planner

Game Planner is a part of our AI, which provides the information about the game. We use that information to choose good strategies for attacking and defending. In fact Game Planner analyzes the game to use in other part of the AI mechanism. Computational complexity and parallel essence of algorithms are used in the planner, make us to use "GPU programming" to speed up the calculations. There are many interfaces that can be used to do that, e.g. OpenAcc, OpenCL, Direct Compute and Cuda. Among them, "Cuda" is using C programming language that make it more suitable in our task. NVidia has created and

optimized "Cuda" only for GPUs produced by itself. We use Cuda 4.2 and a GTX 580 graphic card. Following, we describe Game Planner most important parts.

Prioritizing opponents robots This part plays the main role in defensive tactics. It is very important to cover opponent attack strategy with a reliable defense. In defensive tactics, due to the robots high speed and accuracy, each mistake can make a goal chance for opponent. On the other hand, because of the various attacking strategies used by different teams, covering all possible game conditions needs a lot of parameters and that raises the chance of mistakes. Collecting all these parameters and connecting them together is a very hard or even impossible task, to do that we calculate a score between 0 and 1 for each opponent robot in each frame. This score shows each robot importance in opponent offensive movements. This score must be very reliable in all opponent attack conditions. To increase scores reliability we have divided this operation to two parts, online and offline calculations. In offline calculation by preparing a lot of samples and scoring in different situations of the game, we create a function of initial conditions that assigns each robot a score that depends to some static parameters like location of the opponent most important robot (the robot owning the ball) and location of other opponent robots. In the online calculation, some dynamic parameters like direction and speed of the ball and robots, time it takes for each robot to reach the ball are used to calculate the score, then the results of online and offline calculations are combined to get the final score.

Regioning Another important part is Regioning. In this part, using different algorithms we recognize empty spaces of the field. And we use those spaces to choose a good strategy. Also using this part in different times of the game, executable strategies are suggested to strategy manager. To recognize empty spaces of the field, we use the idea of light being blocked by objects in field. We consider light sources on certain points on the field .The shadows created by objects on field are analyzed for each light source. We combine those using parallel algorithms to divide the field into regions. These regions are mostly used to find the best spot for direct or chip passes, also by analyzing changes of these regions area over time, planner can suggest a suitable way to adjust important regions in a way that implemented strategies chance of success increase by some dynamic off ball movements. Figure 3 illustrates an example adjusting regions by off ball movement.

2.4 Defense Analyzer

One criterion (maybe the most important one) that changes a specific strategy score is its success probability against the opponent defense tactic. We design an online defense analyzer to find suitable attacking strategies based on the opponent defense weak points. Our analyzer runs some movements pattern in a constant period of time during a game in “Free Kick” or “Stop” states of

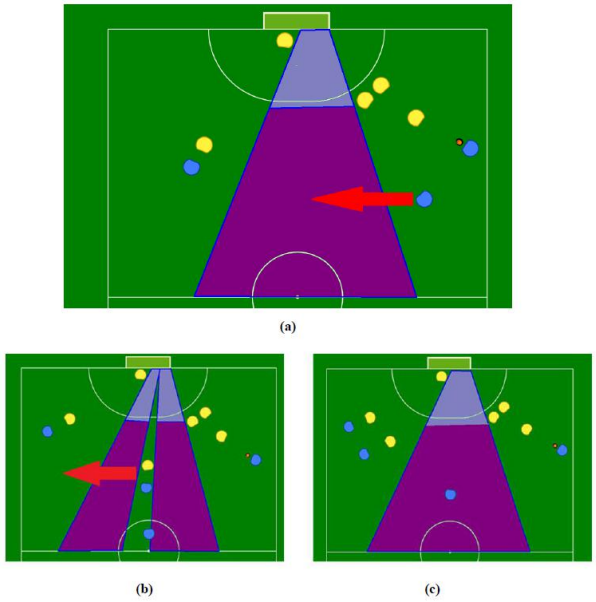

Fig. 3. Adjusting regions by off ball movement

the game. These patterns are programmable and dynamic to reveal important information aboout the defending tactics.

As an example, to find the maximum distance that opponent markers pursuit our attackers, we send two robots to the opponent field. After ensuring from marking these robots, we turn back the robots and measure maximum distance of pursuit.

Some important information, gathered from these patterns, are as belows:

- Which Robots are most important from opponent's point of view in different condition of our attack.
- Type of opponent "Stop-cover", e.g. moves according to the angle of the passer robot or in static form supports the goal.
- Behavior of the opponent defense in contrast with different number of attacker robots.
- Method of marking attackers, this information includes the marker distance from the attacker and maximum distance of pursuit in the field.
- Whether the opponent markers and defenders are aggressive or not.
- How the goal is covered by the goal keeper and the defenders. If the goal is closed by the goal keeper and two defenders or one defender. In the case of using two defenders, the goal keeper is always in the middle of the defenders or beside them.
- Whether opponents use path planning to move on defense line or not.
- If a robot switches between different defending roles (e.g. marker and defender roles).

This information send to the strategy manager. The strategy manager, according to this information and its knowledge about our attack strategies type, changes scores of the related attack strategies.

2.5 Visualizer and online internal debugging

As stated before, to debug onboard control modules such as wheels speed and controller parameters a comprehensive debugging tool is required. Simultaneous investigation of the commanded and the robot velocities (computed via vision and encoder data) is desired. Using this new approach we can easily debug and analyze our PID controller, wireless module data or any of our internal components. We have designed an online link between our microprocessor and AI systems in order to debug , maintain all controllers , speed problems easily and in a time optimal fashion. Also, in the visualizer module we have many other tuning tabs such as high level control parameters, "Active Role" parameters for ease of access and tuning. Figure 4 shows active parameter tab in the visualizer module.

Previously, we had a unique configuration states for all of the robots without considering differences between them. This year, we have embedded a sub-section to our AI system which stores specific properties of each robot. These properties (e.g. include controlling issues, kick speed and ...) later would be used for systems calibrations.

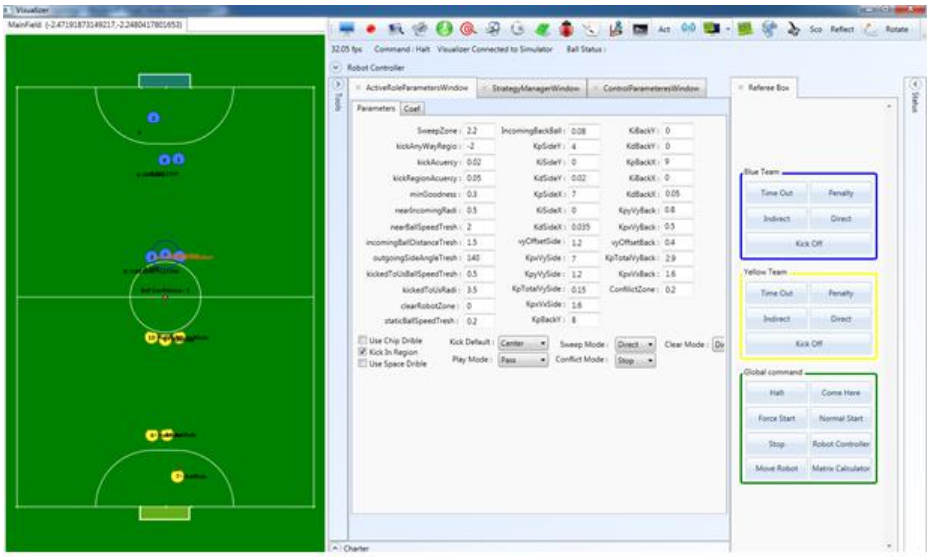

Fig. 4. Active role tuning tab and related parameters in the visualizer

2.6 Navigation system

We are using Rapidly Random Tree (RRT) as a part of our navigation subsystem. A simple RRT path planner tries to generate a tree from an initial state to a goal state using random points in a way that the tree does not encounter obstacles. In the procedure of expanding the tree all the random points that may are in the region of obstacles will be eliminated. Moreover, if there is a reachable path from each random point to goal, the desired tree will be completed. After reaching the goal, usually the generated tree is too rugged for tracking. Before starting to move on path, the generated path must be made smoother. For this purpose we benefit from a heuristic algorithm. In figure 5, the RRT path and smoothed one is depicted. The last experience at Robocup 2012, shows many collisions in

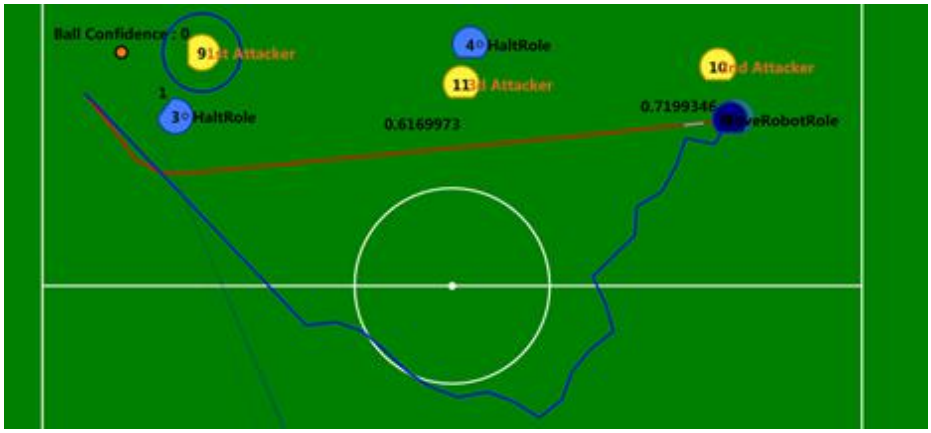

Fig. 5. Sample of the generated path before and after smoothing

each game. This is mainly because of the robots high speed, acceleration and robots high density in the small regions of play. Collisions may cause damages for robots and especially wheels electrical motors. Also, collisions reduce the prediction accuracy that causes performance reduction of some complicated and multi-states strategies. After the path has been generated well to avoid collisions, almost our case, the next step is moving on the generated path. This year we care more about the motion planner. Finding a motion planner that uses the maximum robot ability of motion while following the path is our aim.

To move on the generated path, we use a heuristic algorithm. This algorithm has two steps:

1. Calculating a target point that is a point on the path with length d from the robot current location. The length d depends on the path length l_p and the robot speed size v_r.

$$d = Dl_p v_r \tag{1}$$

where

- l_p: the path length from the current location to the end point,
- v_r: The robot speed size,
- D: Constant

2. Finding an achievable speed vector at the target point v_f. Direction of this speed vector always is along the robot-target vector, $\overrightarrow{p_r p_t}$. Size of the speed vector is calculated as a function of the path curvature κ and the nominal straight line speed v_s as:

$$|v_f| = \cos\left(\kappa(P)\right) v_s \tag{2}$$

where $\kappa(P)$ is the curvature of the path. The path is defined as a set of way-points as $P = \{p_0, p_1, \ldots, p_{n-1}\}$. The curvature is calculated by:

$$\kappa(P) = \sum_{i=\max(i_r-10,0)}^{i_t} \frac{|\angle(\overrightarrow{p_{i+1,i}}, \overrightarrow{p_{i,i-1}})|}{2\sum_{j=i}^{n-2} |\overrightarrow{p_{j+1,j}}|} \tag{3}$$

where in this relation:
- p_t: Calculated Target Point,
- i_t: Nearest node index before p_t,
- n: Nodes count,
- p_r: Robot Location $= p_{n-1}$

Finally, the robot is forced to reach all the target points sequentially with the calculated speed vector in each step. Figure 6 illustrates a schematic of the path following method.

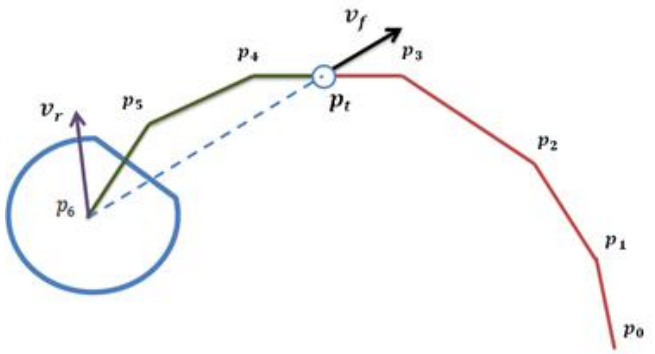

Fig. 6. schematic of the path following method.

3 Electronics

This year the MRL team has changed the main board design and motor driving method. Moreover, the low level controller is implemented on FPGA instead of ARM microcontroller. The other parts are as previous that may be found at [3].

3.1 Main board

The robot has a main board contains processor devices, motor drivers and solenoid driver circuit. A wireless board is also designed to send and receive data between robots and AI system. The current main board is the product of five years designing and evaluating. Figure 7 shows the current mainboard. As men-

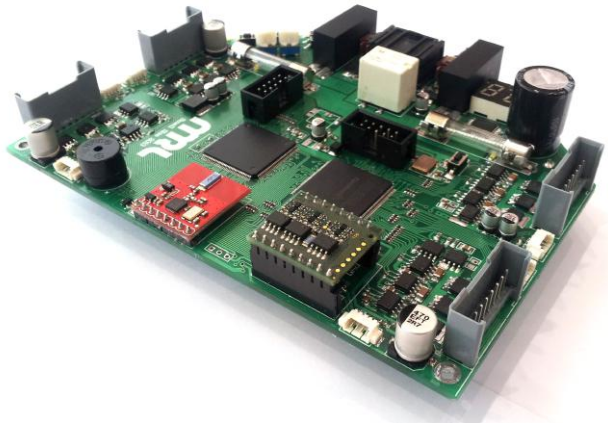

Fig. 7. MRL mainboard

tioned in previous ETDPs we described the advantage of bootstrap drivers. In this chapter some experiences that had been gained through the implementation of these drivers and points which are necessary to be taken into consideration which are useful to cover all aspect of the design are presented, for details see [4].

Principle of bootstrap gate driver: Signals created from FPGA should turn-on the power MOSFETs, but the voltage level of the FPGA pins is not adequate. As a result MOSFET driver should be used to amplify these signals. The previous MOSFET driver has voltage supply limitations also these drivers can be implemented in case the maximum input voltage level is less than the gate-to-source breakdown voltage. While the input voltage level prohibits the use of these drivers principle of bootstrap gate driver can be taken into consideration. Also previously signals were divided into two parts, logic and power. To transfer the signals between these parts, optocouplers are used. Due to photo-transistor structure of this device, temperature which rises in other parts affects the output voltage level of this device. On the other hand the total delay between transitions is high. Therefore it is essential to replace this part of circuit. Direct driver with ground considering, improve the reliability and increase the switching frequency [3].

Improvement of the Turn-off speed: The turn-off speed of MOSFETs only depends on the gate driver circuit. Regarding the mentioned fact, at first a lower resistance should be utilized in output of MOSFET drivers, because a higher current turn-off circuit can discharge the Capacitance of gate-source (Cgs) faster and provide shorter switching time and consequently lower switching losses. However, the using of this circuit increases the ringing of the waveform due to the $\frac{di}{dt}$ of the MOSFET. The simplest technique to achieve higher current in turn-off switching is applying an anti-parallel diode as shown in figure 8. The above cir-

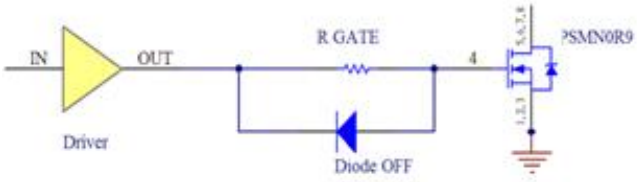

Fig. 8. Turn-off speed improvement circuit

cuit limits the current in turn-on MOSFETs by R-gate and the diode acts like a shunt resistor in turn-off switching. As a result, the turn-off delay time is decreased but still current must flow through the MOSFET driver impedance. The magnitude of the current limited by driver impedance is still lower than currents limited by R-gate. On the other hand, this circuit does not occupy much space to be implemented on printed circuit board (PCB). As it is shown in figure 9, the turn-off time when the anti-parallel diode is used is 120 Nanosecond and without the diode it is more than 1 microsecond but as mentioned before it can be seen that the ringing in waveform when the diode exists is more than the time which it does not.

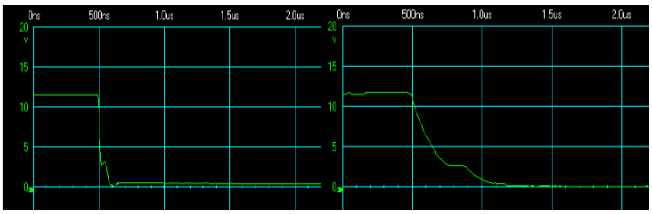

Fig. 9. Turn-off transient voltage: with diode (left). Without diode (right)

Calculate C_{BST} and C_{DRV} values: The most important components in design of the bootstrap drivers is bootstrap capacitor, C_{BST}. In every cycle of

switching, bootstrap capacitor most provide the total gate charge Q_g that is necessary to turn-on the MOSFET in normal operation.

The value of the bootstrap capacitor should be selected in a way to provide the required energy to turn-on and turn-off MOSFET continuously and rapidly and also need relatively short time to charge Q_g against the switching frequency. Therefore the challenge of this part is choosing the best value of capacitor which is attributed to various factors such as T_{onmax} , Q_g and some more. Also the voltage level of the capacitor must never be less than the under voltage lockout threshold of the driver V_{UVLO}.

Another capacitor which is as important as the bootstrap capacitor is C_{DRV}. Since the charge and discharge of the C_{BST} happens in a short time interval, it involves high peak current and reduces potential in driver supply voltage. Therefore it is necessary to utilize a capacitor in the supply of the driver and its magnitude should be larger than the C_{BST}. Also minimizing the whole loop area in PCB is important, see figure 10.

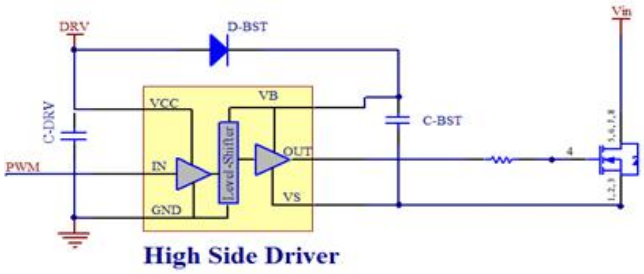

Fig. 10. High side bootstrap gate driver circuit

MOSFET selection: Considering the fact that the most of the power losses is related to the R_{DSon}, at first step to reduce power losses we should choose a MOSFET with lowest R_{DSon} but as a result Q_g of the MOSFET is increased. Q_g factor defines the time transition in switching an reduces the losses in the gate drive circuit owing to the fact that less energy is required to turn-on or turn-off. In optimal design the trade off between these two factors is important. Since the motor driver frequency is low, R_{DSon} is more important than Q_{gin} our design. According to the following table which is owned by NXP^{TM} cooperation the PSMN0R9 high performance N-channel MOSFET is selected. Figure 11 lists different MOSFET specifications.

3.2 Low level controller

The electronic part consist of FPGA and ARM7 microcontroller. All of signal processing are managed in FPGA and other tasks like wireless communication,

Type	Voltage (V)	R_{DSon} (typ) V_{GS} = 4.5 V (mΩ)	Q_G (typ) V_{GS} = 4.5 V (nC)
PSMN0R9-25YLC	25	0.95	51
PSMN1R1-25YLC	25	1.2	39
PSMN1R2-25YLC	25	1.35	31
PSMN1R7-25YLC	25	2	28
PSMN7R5-25YLC	25	8.4	7
PSMN9R0-25YLC	25	10.5	5.6
PSMN010-25YLC	25	11.9	5
PSMN012-25YLC	25	14.1	3.8

Fig. 11. List of MOSFET specifications

algebraic operation and control task are done in ARM7 core. Among these task, the control loop is the most important one.This architecture faces some problems as:

- Time limitation in perform of control loop due to the high computational cost.
- Suppressing the control loop interrupts by some interrupt event with higher priority.

According to the drawbacks mentioned above, redesign of this structure seems essential.

Implementation of PID control loop in FPGA, eliminates these constraints whit lowest software and hardware cost. There are two ways to solve the problems.

- Utilizing a soft core like Nios embedded-processor for Altera FPGA
- Self design architecture and implementation for PID in FPGA

The first way causes over hardware designing. It is not an optimal way, since the ARM architecture is more powerful than other soft cores. Thus, to design and impelemt PID control loop in FPGA, we select the second way based on [5].

4 Mechanical Design and construction

The mechanical system of the small size robot consists of the wheels, stands, chasis, kickers, the dribbler and the robot cover. Some problems in the last version of MRL small size robots encouraged us to change the materials and mechanical design. The diameter of the new robot is 179mm and the height is reduced to 140mm. The spin back system conceals 20% of the ball diameter in maximum situation. Different parts of our new mechanical design are described in [3]. Figure 12 shows our designed robots that are attended in Iran Open 2013 competitions.

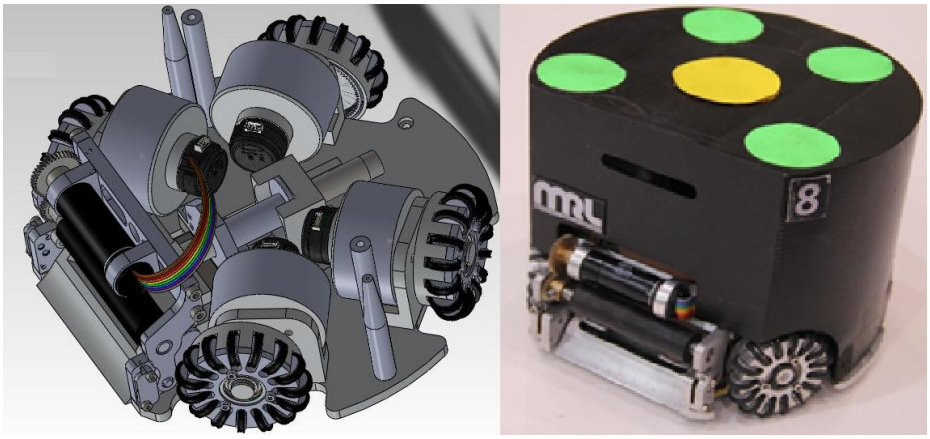

Fig. 12. MRL robots: Design (left) and real (right)

4.1 Kickers

The robot uses two kinds of kicking system, direct kick and chip kick. Each of them is divided in two part, solenoid and plunger. The magnetic plunger material is pure iron $ST37$. Because of the electromagnetic effect two separate parts are used in the cylindrical plunger. The custom-made cylindrical solenoid is used for direct kick which has ability to kick the ball up to $12m/s$. Last year our direct kicker was made from Aluminum alloy but the kickers were broken frequently during the matches. To solve this problem, we replaced it by Titanium Alloy for the new robot. Direct kick solenoid is located between kicking plates which are made from poly-amide and aluminum.

As a second kicking system, MRL2013 has a custom-made flat solenoid. Because of space limitation with high performance chip kick we decided to reshape the solenoid from cylindrical to flat rectangular and placed in the front part of the robot. The chip kick has a 45 degree hinged wedge front of the robot which is capable of kicking the ball up to 6m before it hits the ground. The chip kicker is made from Aluminum Alloy 7075 which is enough strong to kick the ball. Chip kick system has a different plunger from direct kick; chip kick plunger is made from Steel with the thickness of $3.70mm$.

4.2 Dribbling System

Dribbling system is a mechanism to improve the capability of ball handling. Dribbler is a steel shaft covered with a rubber and connected to high speed brushless motor shaft, Maxon EC16 Brushless, figure 13. We examined several materials for dribbler bar, like Polyurethane, Silicon and carbon silicon tube. Carbon Silicon is selected for its higher capability in ball handling. The spin-back motor was in the front of the robot and it was exposed to any strike whether due to ball hit or robots collisions. To solve this problem, we have taken the spin

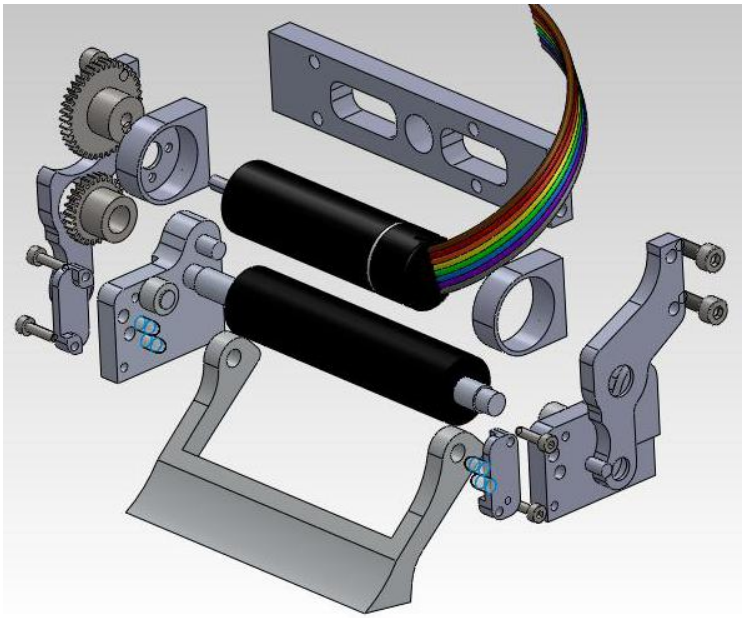

Fig. 13. MRL robot dribbling system

back motors position a little back. Additionally, by attaching the ending point of the spin-back motor with a fastener to spin-back system we fixed the structure. This led to a higher resistance in front of any hit and it also helped the opening and fastening of the dribbling system.

The major change in dribbling section in the following year was the sensors position in the dribbling system which a rotation in spin-back system around the connecting pin or any vibration due to spin-back motor created a movement in the spin-back system and gradually led to sensors movement. With a change in sensors position and fixing them with a specific structure we increased the accuracy and also by placing a cover on the sensors connections we saved them from any unwanted damage.

References

1. Bakhshandeh O., Sharbafi, M.A.: Modeling and Simulating of Omni Directional Soccer Robots. IEEE 3rd International Conf. on Computer Modeling and Simulation, Mumbai, India, (2011)
2. Browning, B., Bruce, J.; Bowling, M., Veloso, M.M.: STP: Skills, Tactics and Plays for Multi-Robot Control in Adversarial Environments. Robotics Institute, (2004)
3. Adhami-Mirhosseini, A., Bakhshande Babersad, O., Jamaati, H., Asadi, S., Ganjali, A.: MRL Extended Team Description 2012. Proceedings of the 15th International RoboCup Symposium, Mexico city, Mexico, (2012)

4. Balogh, L.: Design and application guide for high speed MOSFET gate drive circuits. Texas Instruments/Unitrode Corporation, Power Supply Design Seminar, SEM, (2001).
5. Wei, Z., Kim, B.H., Larson, A.C., Voyles, R.A.: FPGA implementation of closed-loop control system for small-scale robot. Proceedings 12th International Conference on Advanced Robotics, pp. 70–77. IEEE Press, (2005).