MRL Extended Team Description 2014

Amin Ganjali Poudeh, Hadi Beik Mohammadi, Adel Hosseinikia, Saeid Esmaeelpourfard, and Aras Adhami-Mirhossein

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 of experience, is planning to participate in 2014 world games. In this paper, we present an overview of MRL small size team. Having attained the third place in 2010, 2011 and 2013 competitions,this year we enhanced the reliability and achieved higher accuracy. Now, finalizing our debugging tools like 3D simulator, comprehensive user interface and decision systems restructuring, aided us to evaluate the entire system software from low level control to high level strategies. Finally, by overcoming electronic and mechanical structure problems,we promoted the robots ability in performing more complicated tasks.

1 Introduction

MRL team started working on small size robots from 2008. In 2013 Robocup, the team was qualified to be in semi-final round and achieved the third place. In 2014 the team goals are, first: preparation for double-sized field games and, second: Having more dynamic and intelligent behavior. This year, the main structure of the robots is the same as 2013, see [1] for details.

Some requirements to reach this target are achieved by redesigning the electrical and mechanical mechanisms. Moreover, simple learning approaches and advanced control methods are employed in the way of more dynamic play. Evaluation by software tools, like online debugger and simulator which is detailed more in [2], made the design procedure and verification faster.

This paper is organized as follows: First of all, the software architecture which includes our approaches in high level strategies, navigation subsystem and verification tools is described in section 2. The Electrical design including ARM micro controller together with FPGA, and other accessories of robots onboard brain, is explained in section 3. Description of new mechanical structure, which modifies the capabilities of the robots with 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 (see [5]) 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

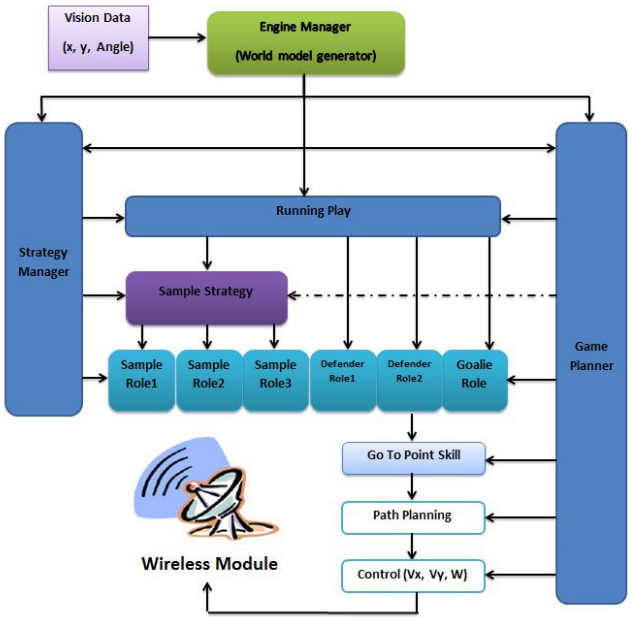

Fig. 1. Block diagram of AI structure

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 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.

Synchronization algorithm This module is used for synchronizing between passer and shooter robot while unexpected causes such as obstacles in the way of shooter robot does not make any disturbance in the pass-shoot timeline. To this end parameter t_m is defined as the time it takes the shooter robot to reach the pass target in an obstacle free path and t_p is defined as the time it takes the ball to reach the pass target from beginning of the pass procedure. t_p and t_m are determined as below:

$$t_m = k_m \cdot motionTime\left(p_{s(0)}, p_t\right)$$
$$t_p = kp \cdot \left(passTime\left(passSpeed, p_t\right) + motionTime\left(p_{p(0)}, p_b\right)\right) + t_{wo}$$

where
t_{wo}: An offset waiting time
$p_{s(i)}$ Shooter robot position at frame i of the procedure
$p_{p(i)}$ Passer robot position at frame i from beginning of the procedure
p_b: Ball first position
p_t: Pass target position
k_m and k_p: Constants
motionTime is an experimental/heuristic function that estimates the time it takes a robot to reach a target from an initial point.
passTime function calculates the time it takes a passed ball to reach a target with an initial pass speed considering both rolling and slipping parts of the ball motion.

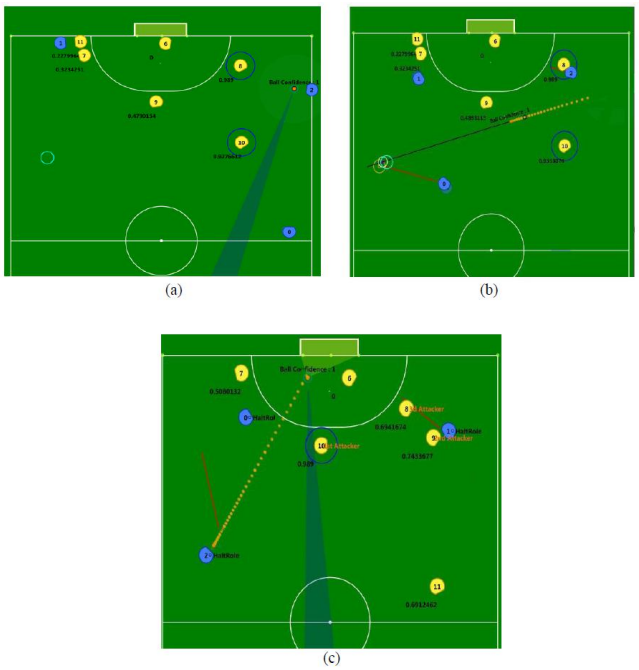

Fig. 2. Pass shoot strategy: (a) Initial state: positioning and pass point (aqua circle) selection (b) Pass state, (c) Final state.

Considering these definitions the pseudo code of synchronization function is as below:

$isInPassState = goPass(passTarget, t_w)$;
if $isInPassState$ **then**
  $counter\ ++$;
  if $t_p > t_m$ **then**
    if $counter \geq t_p - t_m$ **then**
      $goToPoint(shooterRoboto, passTarget)$;
      $determineWaitingTime = true$;
    else
      $determineWaitingTime = false$;
      $stop(shooterRobot)$;
    end
  else
    $gotoPoint(shooterRobot, passTarget)$;
    $determineWaitingTime = true$;
    if $counter < t_m - t_p$ **then**
      $t_{wo}\ ++$;
    end
  end
else
  $counter = 0$;
  $stop(shooterRobot)$;
end
$t_w = t_{wo}$;
if $determineWaitingTime$ **then**
  $t_{we} = calculateExtendedWaitingTime()$;
  $t_w = t_w + t_{we}$;
end

goPass function generates proper commands for robot to go behind the ball and throw a pass to an specific *passTarget* after waiting for t_w frames. It returns a Boolean flag which indicates if there is any obstacle between passer robot and the ball.

calculateExtendedWaitingTime function determines extended waiting time depends on deviation of shooter robot from the straight (obstacle free) path. For this purpose $\boldsymbol{v_r}$ is defined as the reference coordinate as below:

$$\boldsymbol{v_r} = p_t - p_s\,(0)$$

We use a near-time optimal trajectory planner implemented as the function of

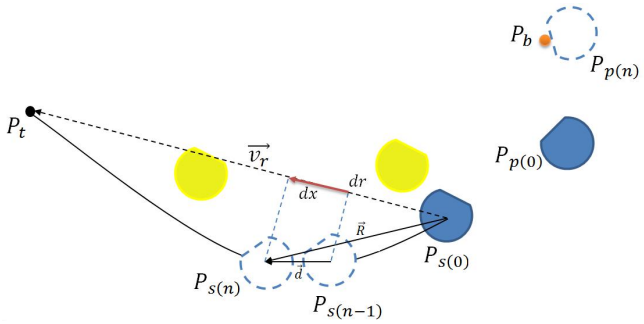

Fig. 3. Pass-shoot synchronization

$\langle V^*_{ideal}, X^* \rangle = motion1DinRefrence\left(P_{s(0)}, p_t, \boldsymbol{r}\right)$ to compute the sequence of velocity commands V^*_{ideal} and the position X^* required to navigate from initial location $p_{s(0)}$ to p_t along $\boldsymbol{v_r}$ with final velocity of zero, see Figure 3. To compare the motion of the shooter robot with an ideal motion along $\boldsymbol{v_r}$, d_x and d_r are determined as belows:

$$\begin{aligned}
\boldsymbol{d} &= P_s(n) - p_s(n-1) \\
\boldsymbol{R} &= P_s\,(n) - p_s\,(0) \\
dr &= |\boldsymbol{R}| \cdot Cos\left(\widehat{\boldsymbol{R}, \boldsymbol{v_r}}\right) \\
dx &= |\boldsymbol{d}| \cdot Cos\left(\widehat{\boldsymbol{d}, \boldsymbol{v_r}}\right)
\end{aligned}$$

where
$\boldsymbol{d}$: Displacement vector of shooter robot in the last frame
$\boldsymbol{R}$: Displacement vector of shooter robot from the initial point
dx: Projection of $\boldsymbol{d}$ on $\boldsymbol{v_r}$
dr: Projection of $\boldsymbol{R}$ on $\boldsymbol{v_r}$

Finally, the extended waiting time t_{we} is calculated from the following equations.

$$\begin{aligned}
v_d &= V^*_{ideal}\,(dr) \\
t_{w_e} &= \textstyle\int_{t_n}^{t_0} 1 - \frac{d_x}{v_d}
\end{aligned}$$

where:
v_d: Desired velocity at distance of dr from $p_{s(0)}$ along $\boldsymbol{v_r}$
t_{we}: Extended waiting time of the passer

2.2 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 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.

2.3 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 4, the RRT path and smoothed one is depicted. The experience at Robocup 2012, shows many collisions in each game.

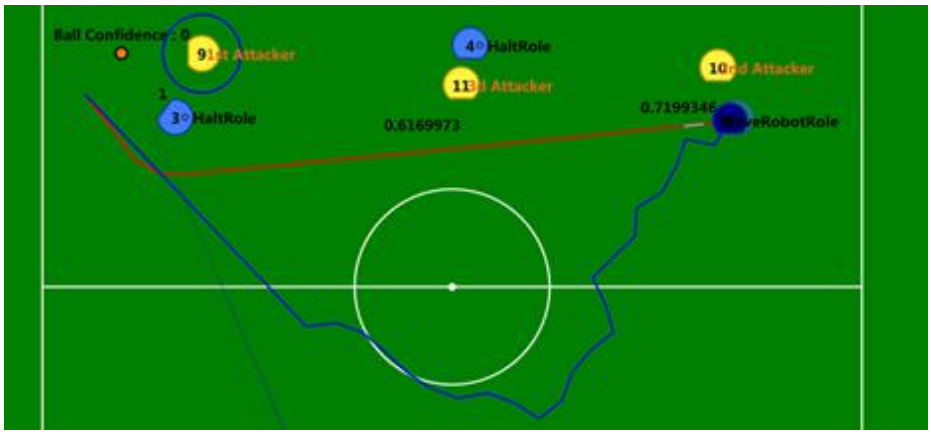

Fig. 4. Sample of the generated path before and after smoothing

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. Last year we cared more about the motion planner. Finding a heuristic algorithm that uses the maximum robot ability of motion while following the path was the aim, for more details on the algorithm see [1].

This year, our main focus is on achieving the method to perform a fast and accurate point regulation of robot. In small size robot soccer the main concern is to make the robot to reach at the target point in the minimum possible time. Like the most small size teams, we employ the practical bang-bang control in the open-loop manner to have the fastest motion between two points. Although this motion is fast enough, it is not precise enough at the end point. Usually, a closed-loop method like PID (or PD) controller are used to achieve precise regulation from a predefined distance from the target point. This year instead of the PID controller, a model based predictive control is employed as the end phase control. In recent years, the prediction concept in mobile robots is highly important because the robots are used more and more in dynamic environments (e.g. robotic soccer, manufacturing plants, and agriculture), [3].

Model predictive control is utilized a reference trajectory which is defined by the best trajectory from the current state (position and velocity) to the desired state. Then, based on the reference, two velocity are generated in X and Y coordinates. This strategy can be shortly explained in the following sub-section.

2.4 Model predictive control

In this section, a linear MPC (LMPC) scheme applied to the problem of trajectory tracking is introduced. MPC can compensate the effect of dead-time by predicting the future output of system while it can be used to overcome some other deficiency of system model. To design the MPC controller for trajectory tracking, the linearized system model will be written in a state space form as:

$$X(k+1) = AX(k) + BU(k-k_0) \tag{1}$$

where $A \in \mathbb{R}^{2\times 2}$ and $B \in \mathbb{R}^{2\times 2}$. By creating a state space model (1) of a system, it is possible to utilize MPC technique. The proposed approach is to find the control-variable values that minimize the quadratic objective function by solving a quadratic program (QP). The quadratic objective function associated with MPC at each sampling time k can be stated as below:

$$J(k) = \sum_{i=1}^{N} z'_{k+i|k} Q z_{k+i|k} + v'_{k+i-1|k} R v_{k+i-1|k} \tag{2}$$

Here $z_{k+i|k} = \hat{x}_{k+j|k} - x_{r_{k+j}}$ represents the error with respect to the reference robot and $v_{k+i|k} = \hat{u}_{k+j-k_0|k} - u_{r_{k+j-k_0}}$ is its associated error control input; N is the prediction horizon; Q and R are symmetric weighting matrices with proper dimensions.

By defining the following matrix, the future state of system can be formulated in (3).

$$Z(k+1) = G(k)Z(k|k) + S(k)V(k) \tag{3}$$

Where

$$S(k) = \begin{bmatrix} B_d(k) & 0 & \cdots & 0 \\ A_d(k)B_d(k) & B_d(k) & \cdots & 0 \\ \vdots & \ddots & \ddots & 0 \\ A_d(k)^{N-1}B_d(k) & A_d(k)^{N-2}B_d(k) & \cdots & B_d(k) \end{bmatrix}, G(k) = [I, A(k), \cdots, A^N(k)]^T \tag{4}$$

Thus, defining $\bar{Q} = diag(Q, \cdots, Q)$ and $\bar{R} = diag(R, \cdots, R)$, the cost function of the QP problem can be rewritten as fallows:

$$\begin{aligned} J(k) &= \sum_{i=1}^{N} z'_{k+i|k} Q z_{k+i|k} + v'_{k+i-1|k} R v_{k+i-1|k} \\ &= X^T \bar{Q} X + V^T \bar{R} V \\ &= (Gz(k|k) + SV)^T \bar{Q} (Gz(k|k) + SV) + V^T \bar{R} V \end{aligned} \tag{5}$$

After some algebraic manipulations, the cost function (2) is rewritten in a standard quadratic form (6):

$$J(k) = \frac{1}{2}\ \bar{U}^T(k)H\bar{U}(k) + F^T\bar{U}(k) + d \tag{6}$$

with

$$\begin{cases} H = 2\big((S(k)^T\bar{Q}S(k)) + \bar{R}\big) \\ F = 2S(k)^T\bar{Q}G(k)z_{k|k} \\ d = z_{k|k}^T S(k)^T\bar{Q}S(k)z_{k|k} \end{cases} \tag{7}$$

where $H \in \mathbb{R}^{2.N\times 2.N}$ and $F \in \mathbb{R}^{2.N}$. Since all constants, which do not consist the variable $\bar{u}$, do not affect the optimization problem,d is independent of v and the QP problem can be represented:

$$J'(k) = \frac{1}{2}\ \bar{U}^T(k)H\bar{U}(k) + F^T\bar{U}(k) \tag{8}$$

After yielding (8), which is a standard expression used in QP problems, the optimization problem to be solved at each sampling instant is represented as (9):

$$v^*(k) = \underset{v}{argmin}[J'(k)] \tag{9}$$

3 Electronics

MRL robot electronic consists of an Altera Cyclone FPGA linked to an ARM core the same as previous years. The major change during last year in this section is implementation of parallel motor controllers in FPGA, since calculation of PID controllers in software requires a lot of CPU time. Moreover, moving controllers to FPGA, the ARM processor can be dedicated to other tasks with less interrupts.

3.1 Main Board

Main board of the robot, which mainly drives wheels and dribbler motors, is illustrated in Figure 5. The board is the same as MRL 2013, [1].

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 the case that 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 previous signals were divided into two parts, logic and power. To transfer the signals between these parts, optocouplers are used. Due to

Fig. 5. MRL mainboard

photo-transistor structure of this device,the temperature which rises up 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, improves the reliability and increases the switching frequency.

Improvement of the Turn-off speed The turn-off speed of MOSFETs only depends on the gate driver circuit. Regarding the mentioned fact,a lower resistance should be firstly utilized in output of MOSFET drivers,since 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,using this circuit increases the ringing of the waveform due to the $di\frac{}{dt}$ of the MOSFET. The simplest technique to achieve higher current in turn-off switching is applying an anti-parallel diode as shown in Figure 6.

The above circuit 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).

circuit.png

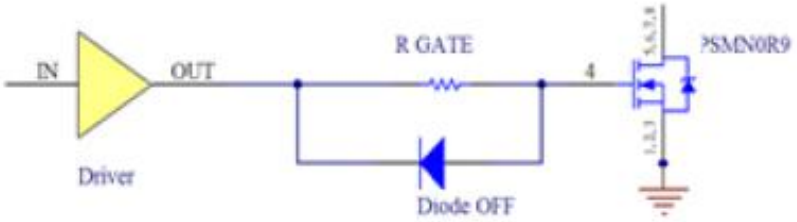

Fig. 6. Turn-off speed improvement circuit

As shown in Figure 7, 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 when it does not exist.

transient.png

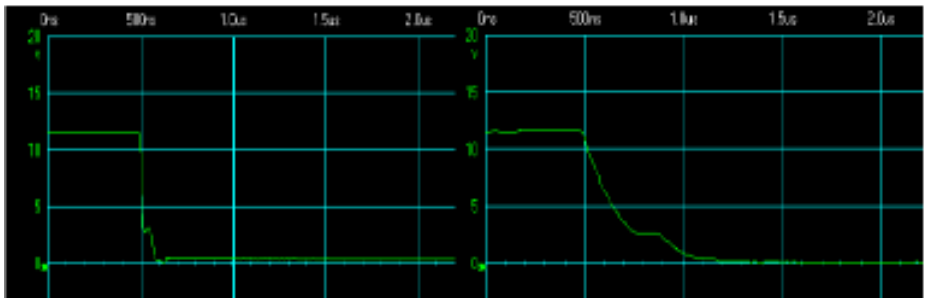

Fig. 7. Turn-off transient voltage: with diode (left), without diode (right)

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 must 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 that provides the required energy turn on and off MOSFET continuously and rapidly. Also it relatively needs short time to charge Q_g against the switching frequency. Therefore the challenge of this part is to choose 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. Another capacitor which is as important as the bootstrap capacitor is V_{UVLO}. 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 8.

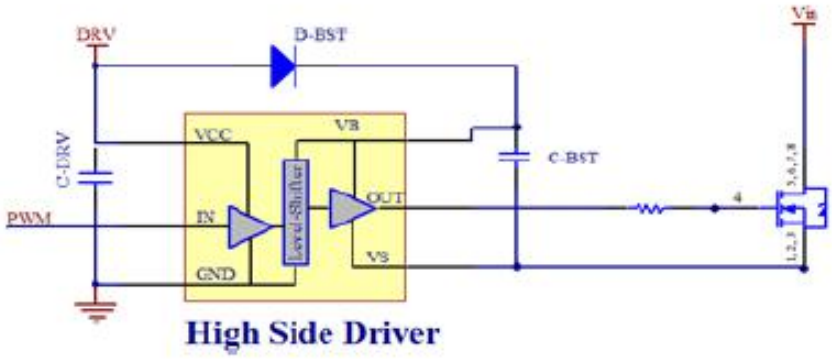

Fig. 8. High side bootstrap gate driver circuit

MOSFET selection Considering the fact that the most of the power loss is related to theR_{DSon},to reduce power loss we should first choose a R_{DSon} but as a result Q_g of the MOSFET is increased. Q_g factor defines the time transition in switching and reduces the losses in the gate drive circuit owing to the fact that less energy is required to turn on or 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} in our design. According to the following table which is owned by NXP^{TM} cooperation the PSMN0R9 high performance N-channel MOSFET is selected. Figure 9 lists different MOSFET specifications.

3.2 Low level Controller

The electronic part consists of FPGA and ARM7 microcontroller. signal processing is managed in FPGA and other tasks like wireless communication, algebraic operation and control task are done in ARM7 core. Among these tasks, the control loop is the most important one.

This architecture faces some problems as:

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. 9. List of MOSFET specifications.

- 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, redesigning of this structure seems essential. Implementation of PID control loop in FPGA, eliminates these constraints with 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 implement PID control loop in FPGA, we select the second way based on [7].

3.3 IR sensor

To recognize the ball position in dribbler we used IR sender sensors with specific frequency which is sync with two special frequency withdifferent measurments. To this end, the transmitter sensor is driven by a square wave with specific frequency. Receiver sensor output voltage is filtered by a narrow band-pass filter, Figure 10. To protect sensors and have better ball positioning, the location and structure of sensors were reshaped.

3.4 Kikers optimization

This year we optimize robots direct and chip kick systems again to overcome main drawbacks of the previous kicking systems. Direct kick designed last year faced two problems:

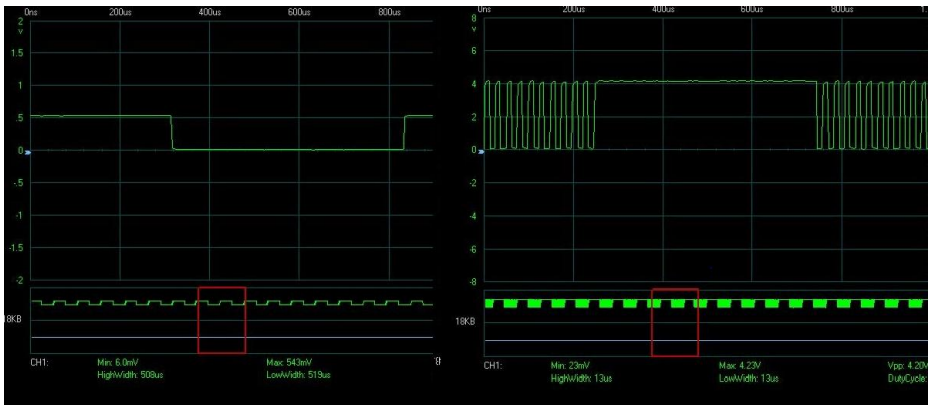

Fig. 10. Sensor signals: Receiver(left),Transmitter(Right)

- *Temperature problem*: Because of the heat loss in the solenoid coil, the kicking systems temperature goes up about $19°c$ after each kick. This causes different ball speeds in the future shoots/passes. More important, heat exchange through the robot may decrease the performance of the wheel motors.
- *Conservative design*: Because of the probable difference between theoretical and experimental results, a conservative optimization was done last year. When the selected set-up was implemented, it showed similar kick speed to the theoretical results.

Considering new constraint on the increasing temperature in each kick, we optimize the direct solenoid specification to reach more than 8 m/s kick speed with minimum weight. Simulations are done by *Matlab* and *FEMM* softwares.

For the chip kick system, we decide to replace the flat solenoid with a cylindrical one which is a more efficient structure. Lack of an appropriate simulator considering flat solenoids with moving plunger, is the second reason encourages us to this replacement. The new chip kick system (solenoid, plunger and capacitors) is 100 gr lighter than the old one. Specifications of the optimized solenoids are listed in Table 1.

Table 1. Specifications of the optimized kicking systems

Type	Solenoid length	Plunger diameter	Number of turns	Wire diameter	Coil resistance	Increasing temperature	Speed or distance
Direct	$33mm$	$8mm$	5	$0.6mm$	1.7Ω	$10°c$	$8.4m/s$
Chip	$29mm$	$11mm$	7	$0.6mm$	1.7Ω	$12°c$	$4m$

4 Mechanical Design and construction

Typically, the main portions of mechanical structure of a small size robot, include 4 wheels, two kickers, a dribbler and the motion transformer system, Figure 11. Regarding the league rules, diameter of the robot is $179mm$ and the height is $140mm$. The spin back system conceals 20% of the ball diameter in the maximum situation.

Due to some drawbacks in the previous proposed design, we have decided to improve both the mechanical design and the construction materials. Main changes in the mechanical structure of the robot are described in the following paragraphs. The other parts are the same as the 2012 robot described in [2].

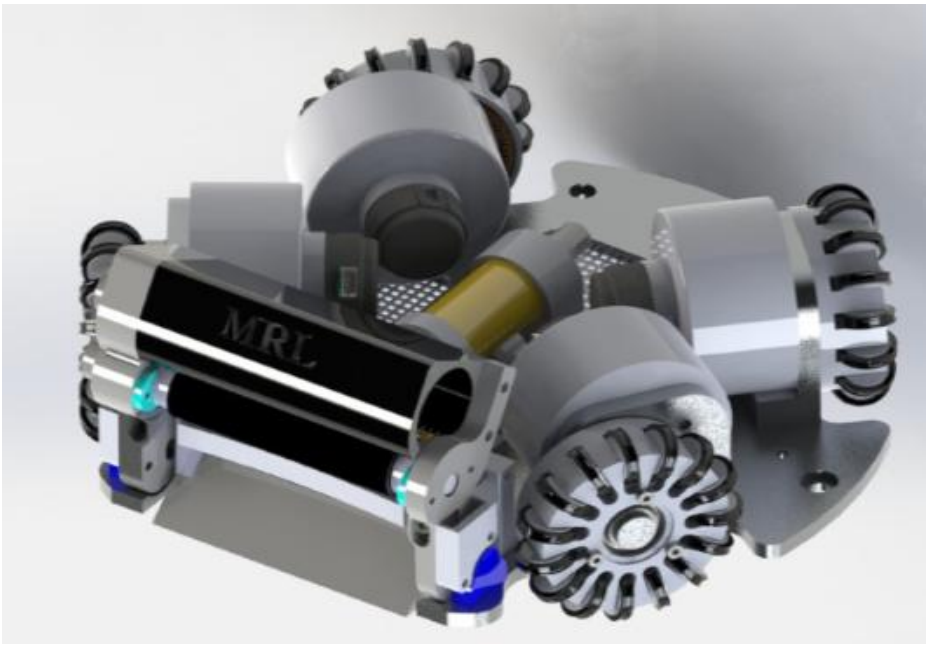

Fig. 11. Robot 2014 mechanical structure

4.1 Kicking systems

We use two kinds of kicking systems, direct kick and chip kick. Each kick system consist two parts, solenoid and plunger. The custom made cylindrical solenoid is used for direct kick similar to last year which has ability to kick the ball up to 9ms. This year as a chip kicking system, because of performance problem we decided to reshape the solenoid from rectangular to cylindrical. Because of the

electromagnetic effect in the cylindrical plunger two separate parts are used, one part is made from pure iron (ST37) and another one is made from Aluminum Alloy (7075). 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.

4.2 Dribbling System

Dribbling system is a mechanism to improve the capability of ball handling. As it is shown in Figure 12, dribbler is a steel shaft covered with a rubber and connected to high speed brushless motor shaft, Maxon EC16 Brushless. 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 took the spin back motors position a little back and designed a shield for spin back motor. To improve the capability of spin back for ball control we made a construct in which the amount damping is controlled.

Due to the changes and increased engine power, we had to change the gear ratio. Also we strengthened the bearing classification. Placing a cover on the sensors connections we saved them from any unwanted damage.

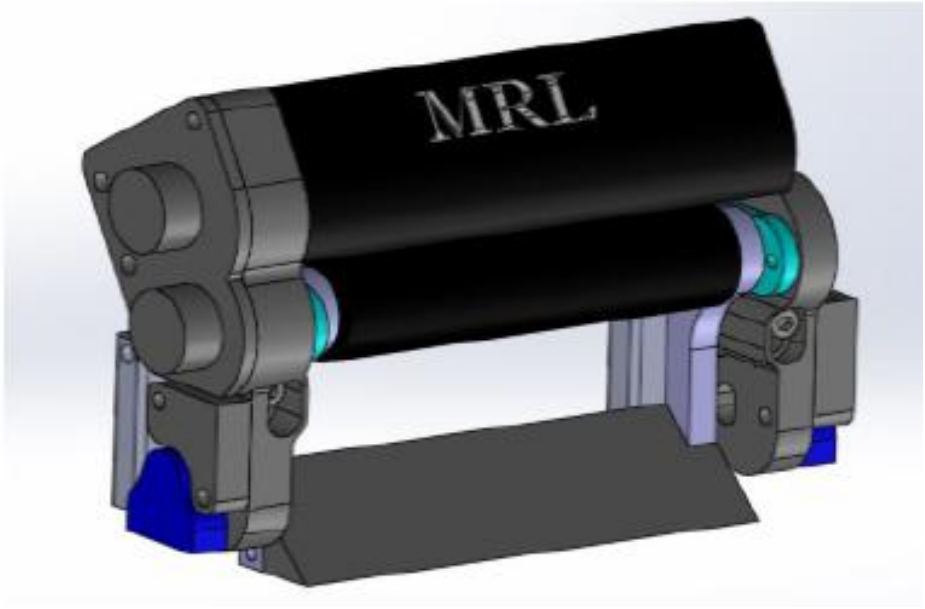

Fig. 12. New dribbling system of the MRL robot

References

1. Ganjali Poudeh, A., Asadi Dastjerdi, S., Esmaeelpourfard, S., Beik Mohammadi, H., Adhami-Mirhossein, A.: MRL Extended Team Description 2013. Proceedings of the 16th International RoboCup Symposium, Eindhoven, Netherlands, (2013)
2. 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)
3. Zarghami, M., Fakharian, A., Ganjali-Poudeh, A., Adhami-Mirhosseini, A.,:Fast and Precise Positioning of Wheeled Omni-directional Robot With Input Delay Using Model-based Predictive Control. Proceedings of the 33th Chinese Control Conference (CCC), Nanjing, china, (2104)
4. 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)
5. Browning, B., Bruce, J.; Bowling, M., Veloso, M.M.: STP: Skills, Tactics and Plays for Multi-Robot Control in Adversarial Environments. Robotics Institute, (2004)
6. Balogh, L.: Design and application guide for high speed MOSFET gate drive circuits. Texas Instruments/Unitrode Corporation, Power Supply Design Seminar, SEM, (2001).
7. 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).