NEUIslanders 2015 Team Description Paper

Prof. Dr. Rahib H.ABIYEV[1], Assist. Prof. Dr. Irfan GUNSEL[2], Nurullah AKKAYA[1], Murat ARSLAN[1], Mustafa ARICI[1], Ahmet CAGMAN[1], Seyhan HUSEYIN[1], Fatih EMREM[3], Gorkem SAY[3], Ersin AYTAC[4]

[1]Department of Computer Engineering
[2]Chairman of the Board of Trustees
[3]Department of Electrical and Electronic Engineering
[4]Department of Mechanical Engineering
Near East University, Lefkosa, TRNC

Abstract. This paper presents the 2015 design of NEUIslanders robotic team of small size league in RoboCup 2015. This year, we focused on all parts of our robots. Improvements include mechanical design, electronic design and software design. Especially, we achieved mechanical and electronic improvements.

1. Introduction

NEUIslanders is an interdisciplinary team of graduate students at the Near East University. The has been taking regular place in RoboCup events since 2012. This paper will outline the progress of our new generation robots since the last RoboCup.

In section 2 we give the information of our electronics design. This year we are giving all the parts that we are using in our robots in detail. In section 3 we explain how our mechanics work and their materials. And in section 4 we give details about our decision making process in our software.

NEU ISLANDERS Robot Specifications	
Dimensions	φ178x145mm
Weight	3200gr
Driving Motors	Maxon EC-45 Flat 30Watt
Driving Gear Ratio	72:20
Dribbler Motor	Maxon EC-16 15Watt
Dribbler Gear Ratio	24:48
Kick Speed	Up to 8m/s (electronically limited)
Communication	XBEE 1mW

Table 1. Robot Specifications

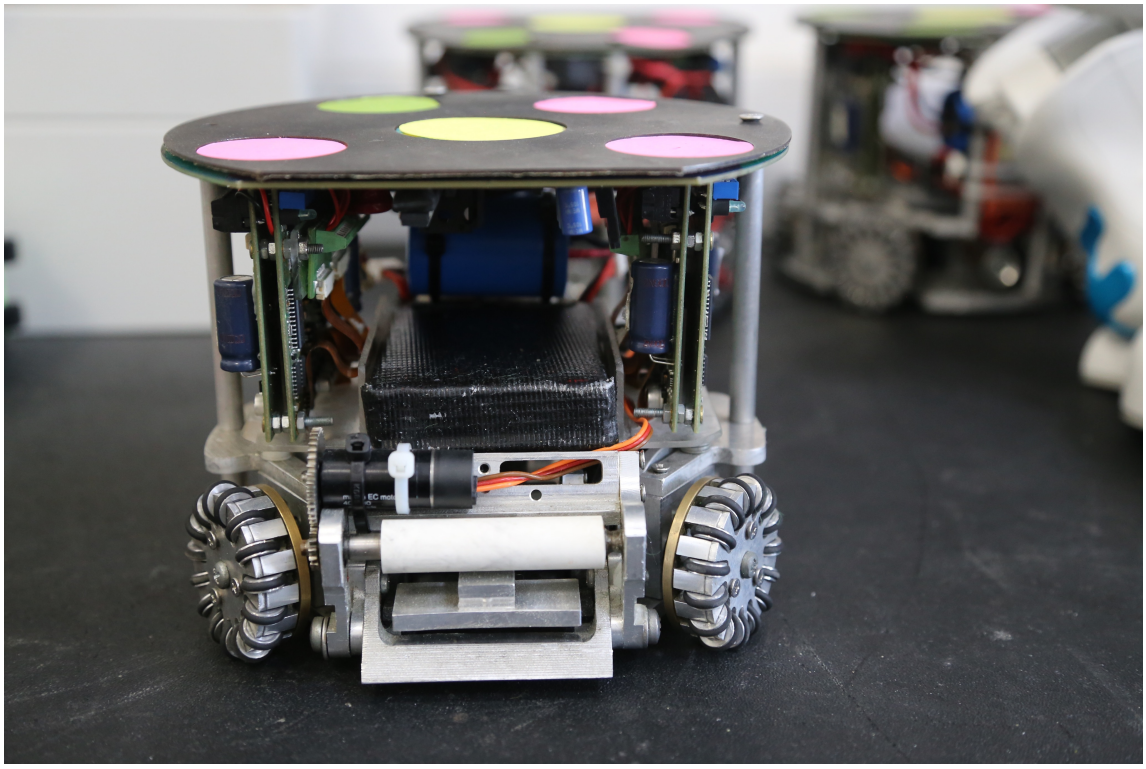

Fig. 1. Front View of NEUIslanders SSL Robot

2. Electronic Design

The electronic circuits of robots include main control units of wheel's motors, communication system, dribbler's motor and kicking parts. This year, we achieved a lot of improvements on electronic design. These changes include main CPU, kicking circuit, motors, motor drivers, dribbler motor drivers. We made these improvements because we faced some problems.

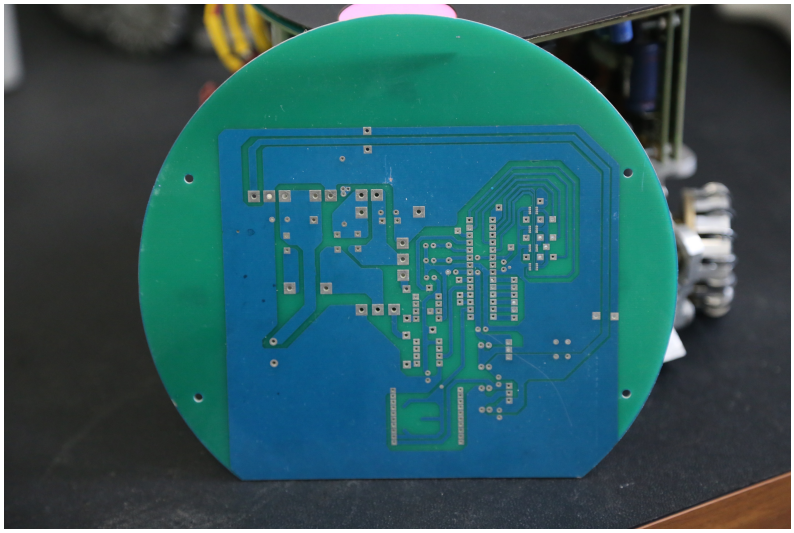

Fig. 2. Main Circuit Board

2.1. Battery

Last year, we used two 2650 mAh high discharge 35-70C lithium polymer batteries and all of them connected series to each other. This year, we have got new batteries. They are 12.6V 1800 mAh 25C lithium polymer batteries. All batteries are placed in li-po safes for fire or explosion accidents. we decreased weight of our new generation robots so we determined new specification of our batteries. Since we have observed that we had more power than we needed for one game in RoboCup for weight reduction purposes we made this decision.

2.2. Main CPU

One of the major changes is the main CPU. Since we had started to RoboCup, we changed our main CPU first time. We faced some problems so we decided to change. Our old CPU was ATmega328p and we started to use Atmel SAM3X8E ARM Cortex-M3. In ATmega328p, there are 14 digital input/output pins, 6 analog inputs, 16 Mhz clock speed so we didn't have enough digital pin and analog pin that was the major problem, second major problem was the Atmega didn't run our software anymore, other problem was transmitting and receiving signals. There are a lot of transmitting and receiving signals so our ATmega is too slow to process all signals and it misses some of them. In the new CPU, there are 54 digital input/output pins ,12 analog inputs, 84 Mhz clock speed and 2 DAC. We aim to overcome all problems and achieve maximum performance with new microprocessor Atmel SAM3X8E ARM Cortex-M3.

2.3. Motors&Motors Drivers

In our new generations robots we also change our motors and motor drivers. Old motors was 36V 30 Watts 3 phase brushless DC Maxon Ec-45 Flat motors. In this year, we use 30 Watts 3 phase brushless DC Maxon Ec-45 flat with MILE-Encoder.

For control more efficient, we selected this motor. The MILE encoder uses an inductive angle measurement system to generate incremental quadrature output signals. The encoder is designed for highest robustness in application. It can be operated in the open environment of an EC flat motor and is equipped with additional ESD protection circuitry. Due to the robustness of the MILE technology in terms of magnetic interference it was possible to integrate the encoder into the flat motor with minimal change dimensions with respect to a motor with our encoder. Depending on motor, we choose a motor driver. It's an EPOS2 24/2 digital positioning controller, 2A, 9-14 Vdc. Maxon motor control's EPOS2 24/2 is a small-sized, full digital smart motion control unit. Due to its flexible and high efficient power stage, the EPOS2 24/2 drives brushed DC motors with digital encoder as well as brushless EC motors with digital Hall sensors and encoder. The sinusoidal current commutation by space vector control offers to drive brushless EC motors with minimal torque ripple and lod noise. The integrated position, velocity and current control functionally allows sophisticated positioning application.

For control more efficient, we used AD5061 16-Bit DAC. The AD5061 is a low power, single 16-bit buffered voltage-out DAC that operates from a single 2.7 V to 5.5 V supply. The DAC will be driven by a micro-controller. The DAC will receive the input data code and convert the data code into current outputs to appropriate motor driven circuitry. The motor driver circuit can be implemented in several ways.

During the operation, the motor with an encoder sends velocity and position signals to the micro-controller. Depending on the encoder these signals may also include an index pulse signal. The micro controller then adjusts by changing the data codes sent to the DAC. The DAC, just like the ADC and the operational amplifier, plays a key role in a myriad of applications. If one considers the ubiquitous op amp as the "glue" between mixed-signal components, then it can be concluded that the three central components on a signal path are the op amp, ADC, and DAC. As the signal passes from the analog domain to digital and back again, the DAC could be considered the dénouement of a circuit. The signal, having done its work, returns to the analog domain. DAC's necessary to use a galvanically-isolated interface to protect and isolate the controlling circuitry from any hazardous common-mode voltages that may occur in the area where the DAC is functioning. That's why we used ADuM series digital isolator in circuit for DAC.

For dribbler motor, we use EC 16 brushless 30 Watt sensorless motors and for driver, we use ESCON Module 50/4 EC-2 4-Q Servo controller for sensorless EC motors.

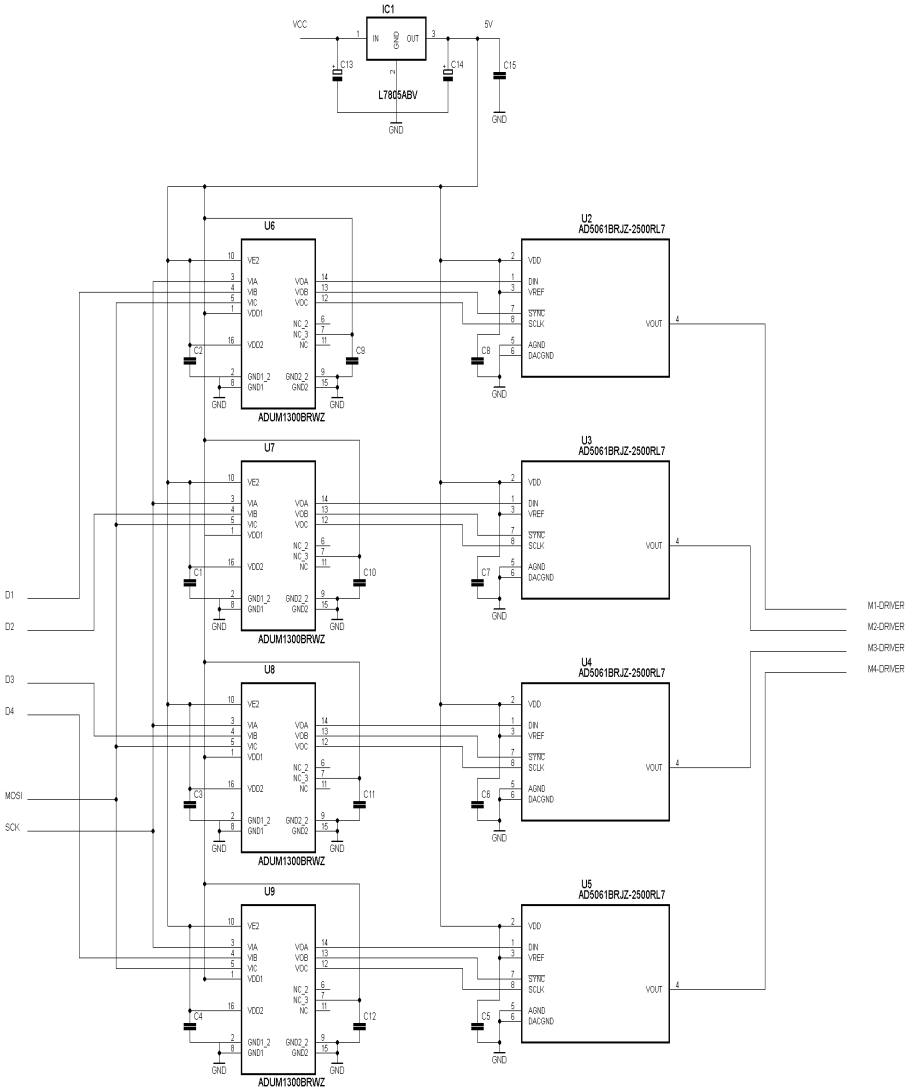

Fig. 3. Digital Isolators and DAC's Schematics

2.4. Kicking Circuit

We also changed our kicker circuit. We used high speed MOSFET IRFP450 to charge our 2200uF. We give 30 kHz PWM to MOSFET. For drive a MOSFET we use IR2106 MOSFET Driver. It provides high speed, high current MOSFET drive capability for efficient switching conversion in dc to dc converter. It adds over temperature and prevision enabled protection features to enhance overall system reliability and performance. MOSFET boosts voltage up and charge the capacitor to

240Vdc. To stop capacitor charging, we designed our comparator. In comparator part we used a 10 Mohm and 68 Kohm. We read voltage of resistors via micro-controller and when capacitor voltage become 240, the controller stops pwm . as a result we designed confidently boost circuit. To achieve various kicking forces, we use IGBT G27N120BN. To drive IGBT, we use IR2106 IGBT driver. When we apply PWM with different intervals, we create various kicking forces.

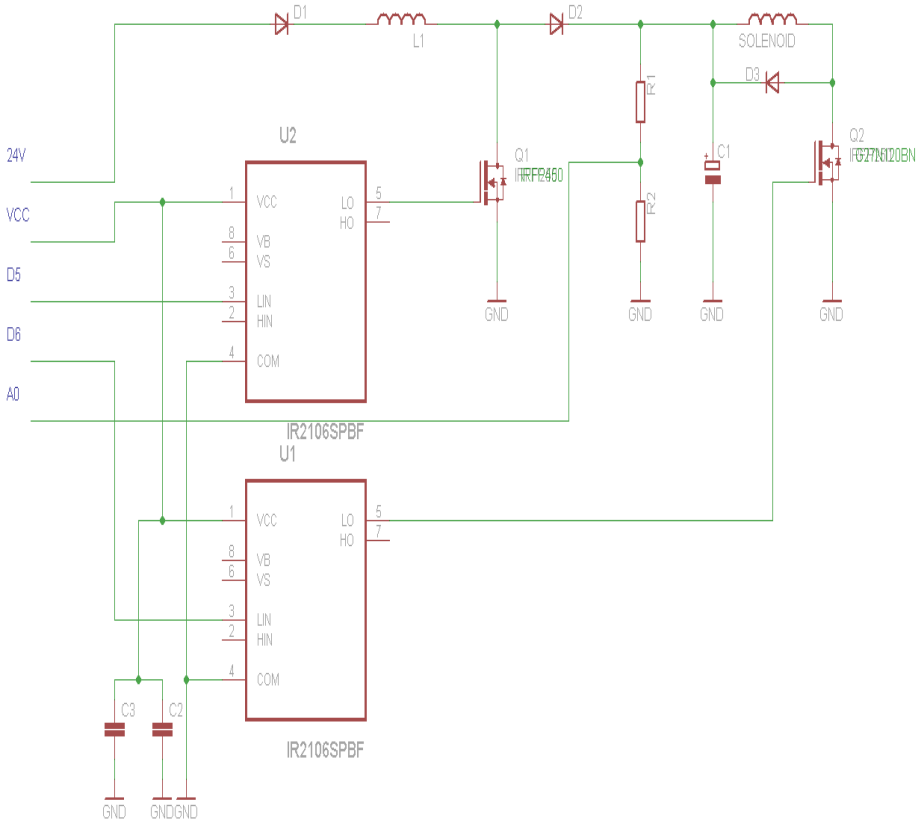

Fig. 4 Design of kicker circuit

2.5. Communication

In this year, we didn't change anything in communication because we didn't face any problem with our RF modules. We use XBEE-PRO RF module and it was working perfectly in communication area.

Fig. 5 Atmel SAM3X8E ARM Cortex-M3

3. Mechanical Design

3.1. Chassis

We are going to use new 3mm titanium for the chassis of our robots this year. Although we are going to build our robot from titanium this year we are going to keep the same wheel orientation and all the other dimensions of our robot regarding to last year. We will have 45 degrees at the rear and 33 degrees at the front wheels with respect to the horizontal axis.

3.2. Omni-Wheels

We have 15 rollers on our omni-wheels and they are 10mm in diameter. The pinion that is connected to the motor has 20 teeth and the internal gears which are mounted on the back side of each wheel has 72 teeth. As a result we will have 3.6:1 gear ratio on our wheels where we will have more torque to achieve the desired acceleration.

Fig. 6 Inside view of our Omni-Wheel

3.3. Kicking Mechanisms

We can hit the ball in very large range of speeds from 0.1 m/s to 8.0 m/s. The kicking mechanism will not kick faster than 8.0 m/s in any circumstances. As a new part to our robots this year we implemented chip kicker to the robots. While chip kicking the ball will not exceed 50cm height at the top point.

3.4. Dribbling Mechanism

Dribbling mechanism uses Maxon EC-16 30Watt motors. We have 1:2 gear ratio in our dribbler to reach around 12000rpm. With a screw set up we can change the height of the dribbling mechanism to adapt different types of carpet.

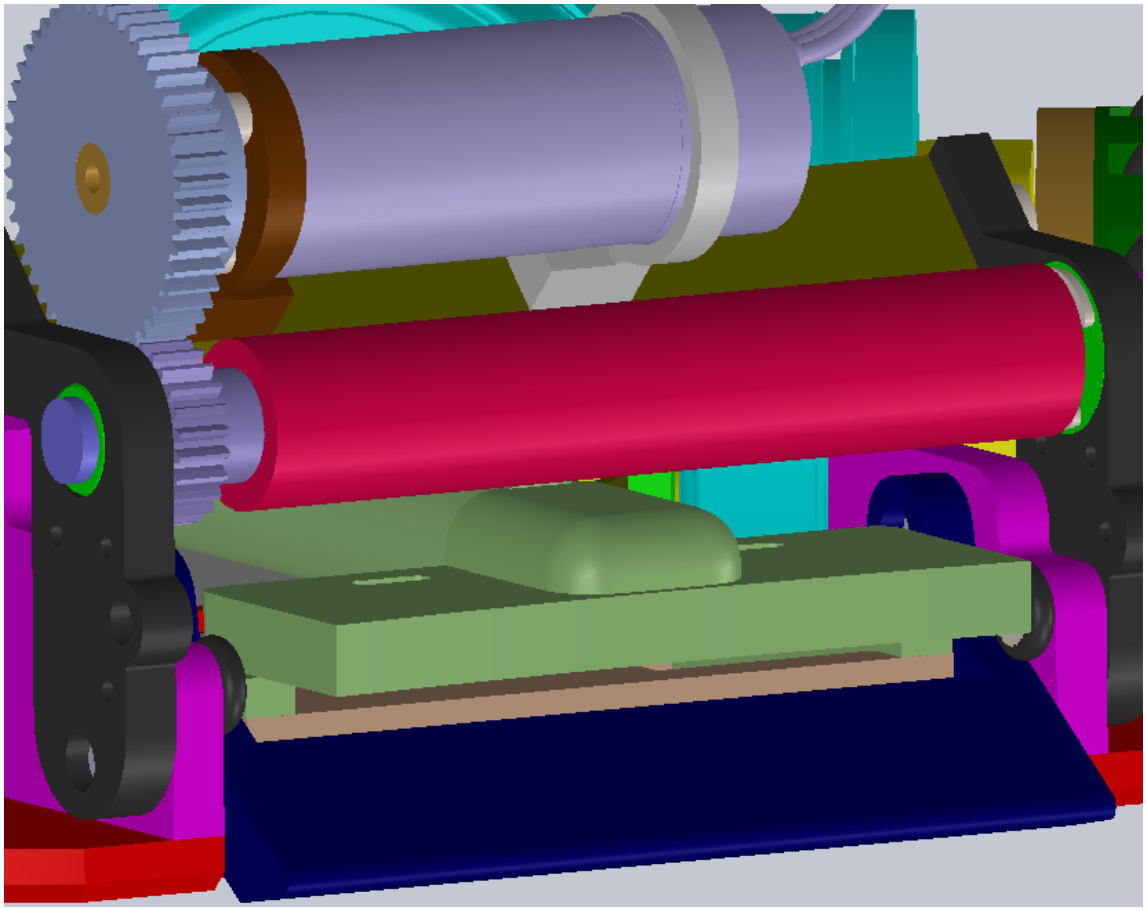

Fig 7. Kicking, Dribbling and Chip-Kicking Mechanisms

4. Software Design

4.1 Behavior Tree Based Control

Decision Making (DM) is one of basic blocks of soccer robot navigation system. DM analyses the current state of the world model and makes decisions about new positions of robots. In the paper behavior tree based control is proposed for decision making. BT is goal oriented. Each tree is assigned a goal, that will be achieved. The robot behavior is a control law that satisfies a set of constraints to achieve a particular goal. Each behavior is defined by the set of actions. BTs perform a number of artificial intelligence (AI) techniques such as Finite State Machines, Scheduling, Planning, and Action Execution.

A BT enables modularity, making states nested within each other and thus forming a tree-like structure, and restricting transitions to only these nested states. The root node branches down to the tree until the leaf nodes are achieved. The leaf nodes are the base actions that define the behaviors. As noted above the use of FSM for robot games increases the number of states required to encode the behavior of the robots, along with the number of transitions between the states. This grows the complexity of algorithm. Behavior trees replace the growing mess of state transitions of finite state machines (FSMs) with a more restrictive but also more structured traversal defining approach. Because of this, BTs easily define complex states. Behavior trees are formed by hierarchically organizing behavior sub-trees. The strength of BT comes from the fact that it is very easy to see logic, they are fast to execute and easy to maintain. In the paper BTs for control of soccer robots playing a football game are designed. A BT is made up of three types of nodes, action, decorator, composite. Composite and decorator nodes are used to control the flow within the three and action nodes are where the code is executed, they return success or failure and their

return value is then used to decide where to navigate next in the tree. Actions are used to change state such as calculating a new path or kicking the ball. Composite nodes include set of nodes such as selector, sequence nodes and their parallel and random versions.

Selector and sequence nodes are workhorse internal nodes. A sequence represents a series of behaviors that needs to be accomplished. A sequence will try to execute all its children from left to right. If all of its children succeed, sequence will also succeed. If one of its children fails, sequence will stop and return failure (Fig.4.1(a)). Fig.4.1(a) describes a hypothetical “Pass” sequence which will start executing its children from left to right, first checking if the receiver is at the correct assist spot waiting for a pass then make sure a pass is safe. If both these conditions are met, only then it will execute the pass and return success up the tree.

Selector node will try to execute its first child. If its first child returns success, it will also return success. If the child fails, it will try executing its next child until one of its children returns success, or the node runs out of children at which point the node will return failure. This property allows us to choose which behavior to run next. The tree in Fig.4(b) chooses between shooting at the goal directly or executing a pass. Selector will start executing its children from left to right beginning with “Shoot Goal” sequence. If both “Can Shoot to goal?” and “Shoot” actions succeeds, sequence will succeed which will cause the selector to succeed. If “Shoot Goal” sequence fails it will keep trying to execute its children until one succeeds in which case selector will also succeed. If all fails selector will also fail.

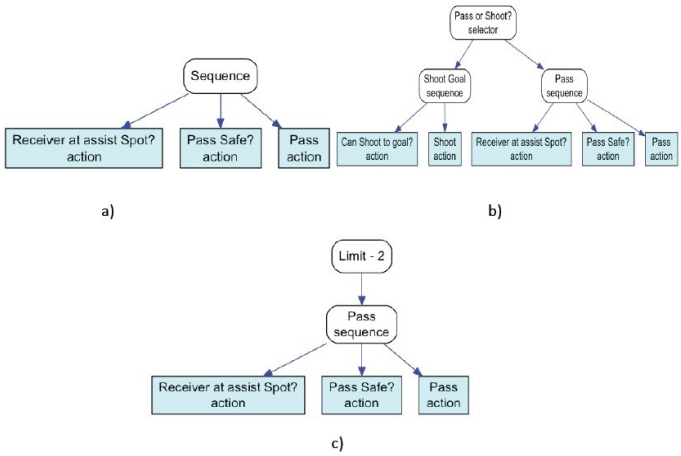

Fig 8. Behaviour Tree

As an example in Fig.4.2 “Pass” behavior is described. “Pass” sequence begins by taking control of two robots using a parallel sequence which acts like a sequence but executes its children in parallel aligns, passer and receiver for a pass, then the tree checks if a pass is safe, if a pass can be made passer shoots the ball. Next, the sequence waits for the ball to start moving, once that happens, we again wait for either the ball to stop or the ball gets within a robot diameter to the receiver at which point receiver moves and captures the ball.

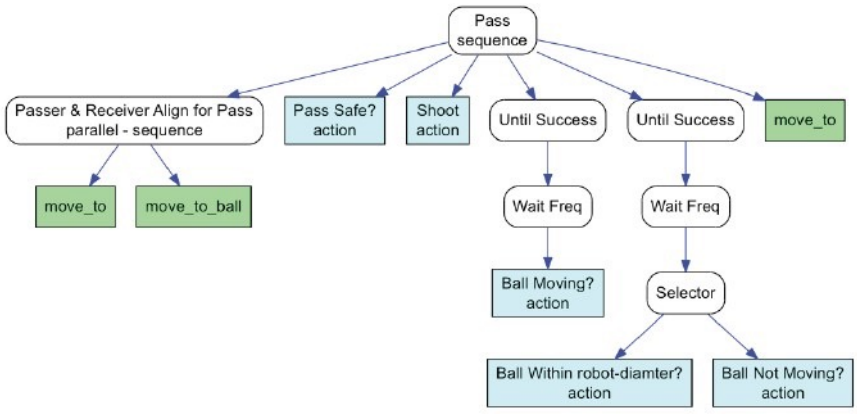

Fig. 9. Pass Behavior Tree

The low level behaviors are composed to form high level behaviors. Some of the high level behaviors include,

- Formations behavior places robots on various hard coded spots on the field.
- Offensive Game behavior focuses on scoring goals against the opponent, as opposed to defense plays, which focus on preventing goals being scored into home goal.
- Defensive Game concerns plays within game that focus on preventing the opponent from scoring goals, as opposed to offense plays that focus on scoring against the opponent

- Game Selection behavior used to select the main tactic.

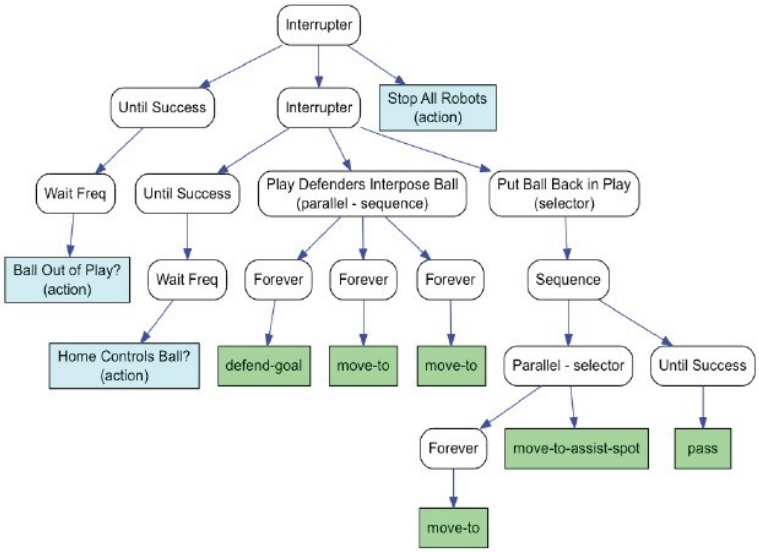

Fig. 10. Defensive game behavior

5. References

1. Rahib H. Abiyev, Nurullah Akkaya, Ersin Aytac. *Navigation of Mobile Robot in Dynamic Environment.* IEEE CSAE 2012 Conference, China

2. Rahib H. Abiyev, Nurullah Akkaya, Ersin Aytac, Mustafa Arici, Muhlis Bayezit. *NEUIslanders Team Description Paper 2012*, Robocup SSL, MexicoCity, Mexico

3. Rahib H. Abiyev, Senol Bektas, Nurullah Akkaya, Ersin Aytac. *Behavior Tree Based Control of Holonomic Robots*. International Journal of Robotics and Automation. WSEAS Conference 2013, Limasol, Cyprus

4. Rahib H. Abiyev, Nurullah Akkaya, Ersin Aytac. *Control of Soccer Robots Using Behaviour Trees*. ASCC 2013

5. Rahib H. Abiyev, Senol Bektas, Nurullah Akkaya, Ersin Aytac. *Behavior Trees Based Decision Making for Soccer Robots.* Recent Advances in Mathematical Methods, Intelligent Systems and Materials 2013

6. Rahib H. Abiyev, Nurullah Akkaya, Ersin Aytac, Dogan Ibrahim. *Behavior Tree Based Control For Efficient Navigation Of Holonomic Robots*. International Journal of Robotics and Automation

7. Ali Erdinc Koroglu, Rahib Abiyev, Nurullah Akkaya, Ersin Aytac, Mustafa Arici, Kamil Dimililer. *NEUIslanders Team Description Paper 2013*, Robocup SSL, Eindhoven, Netherlands

8. Rahib H. Abiyev, Nurullah Akkaya, Ersin Aytac, Irfan Gunsel, Ahmet Cagman *Improved Path-Finding Algorithm for Robot Soccer.* International Conference on Control, Robotics and Informatics. Hong Kong 2014

9. Rahib Abiyev, Nurullah Akkaya, Ersin Aytac, Gorkem Say, Fatih Emrem, Mustafa Arici. *NEUIslanders Team Description Paper 2014*, Robocup SSL, Joao Pessoa, Brazil