Middle Robots League, Team *CS-Sharif*, pages 118–126
http://www.ep.liu.se/ea/cis/1999/006/20/

# CS-Sharif ROCS99 team in middle-sized robots league

**CS-Sharif**

*Mansour Jamzad*

MJ: Sharif University of Technology, Iran

**Abstract.** **Arvand** *is a robot specially designed and constructed for playing soccer according to RoboCup rules and regulations for the medium size robots. This robot consists of three main parts: mechanics (motion and kicker), hardware (image acquisition, processing and control unit) and software (image processing, motion control and decision making). The motion mechanism is based on a drive unit, a steer unit and a castor wheel. Robot machine vision system uses a CCD camera and a frame grabber. Two microcontroller based boards are specially designed for carrying out the software system decisions and transferring them to the robot mechanics. The software system can perform real time image processing and object recognition. It implements algorithms for playing soccer. These algorithms are written in a high level language "ArvandLan" specially designed for mobilizing these robots. By changing the algorithms our robots can play as goal keeper, attacker or defender. We have constructed 5 such robots and successfully tested them in a soccer field defined according to RoboCup regulations.*

## 1 Mechanics Architecture

In the following we describe the steps of design and construction of our special purpose robot for soccer playing.

According to the motion complexity of a soccer player robot, proper design of its mechanics can play a unique role in simplifying the playing algorithms. For attaining this goal, there is a need to a mechanism that can easily and with the most accuracy provide the robot with its motion demands. In this regard, after performing several experiments on the motion mechanisms of mobile robots a specific mechanism was designed and implemented that together with the sensors and control feedbacks, to a good extent, verified our expectancy. The first experiment in producing **Arvand** was making a model car chassis that did not provide us with expected motion capabilities such as dribbling, penalty kick, intercepting and preserving the ball. For this reason, the mechanism described below, was designed and implemented.

## 1.1 Motion Mechanism

**Arvand** consists of two motion units in front of the robot and one castor wheel in the rear. Each motion unit has a drive unit and a steer unit. The functionality of drive unit is moving the robot and that of steer unit is rotating the drive unit round the vertical axis of its wheel. The drive unit consists of a wheel which is moved by a DC motor and a gearbox of 1:15 ratio [1]. The steer unit uses a DC motor and a gearbox of 1:80 ratio. Drive unit and steer unit use the same kind of DC motors. For controlling the steer unit, the optical encoders are mounted on the respective motor shaft and their resolutions are such that one pulse represents 0.14 degrees of drive unit rotation. The castor wheel consists of a spherical ball that roles in an special purpose ball bearing. By this structure **Arvand** can move in any direction freely.

This mechanism has the following capabilities:

1. By rotating the drive unit round its vertical axis the rotation center of the robot changes accordingly and this allows the robot to turn around any point in the plane. This point can be selected inside or outside the robot. It is necessary to adjust the speed of two drive units according to the following formula [2]:

   $$v_1.r_2 = v_2.r_1 \tag{1}$$

   In the above formula $v_1$ and $v_2$ is speed of the left drive motor and right drive motor respectively, $r_1$ is the distance of the left drive unit from the rotation center and $r_2$ is the distance of the right drive unit from the rotation center. Therefore, the robot rotation center will not depend on the robot center of gravity and on the position of drive units in the robot. For instance if we consider the center of ball the rotation center, the robot is able to turn around this center point in a way that it does not lose its sight of the ball and make the appropriate direction according to the opponent team goal position. One of the advantages of our design is the possibility of making the robot rotate around its geometrical center. This kind of rotation is done in a minimum amount of area which reduces the chance of accidental bump to wall or other robots.

2. In our software system we can set the drive units to be parallel to each other and also have a specific angle related to robot front. This mechanism is useful for taking out the ball when stuck in a wall corner and also dribbling other robots.

## 1.2 Kicker Mechanism

Appropriate use of a kicker in robot plays an important role in team play algorithms and individual technics. Therefore, we have designed a kicker with controllable kicking power. For instance, it can be applied to passing in the team play. We tested several kickers using a motor unit and also a solenoid. The solenoid was selected because of its efficiency in power usage. The kicker consists of a solenoid and a simple crowbar connected to a kicking arm. The kicking power is controlled by duration of 24 DC voltage applied to it.

# 2 Hardware Architecture

The goal of our hardware architecture is to have a kind of hardware control on the robot that would be independent of software system as much as possible and also reduce the robots mechanical errors.

**Arvand** hardware system consists of three principal subsystems which are described in the following:

## 2.1 The Image Acquisition Unit

The image acquisition unit output is a digitized color RGB image. For the first experimental implementation of **Arvand**, a Connectix Color QuickCam2 was utilized. It could transfer the digitized captured image via the parallel port. Because of its low resolution and low capture speed, a faster imaging system was needed. Consequently, we chose a PixelView CL-GD544XP+ capture card which provides **Arvand** with images having a resolution of 704x510 with the frame rate of 30 frames per second. The camera which we use is a Topica PAL color CCD camera with a 6mm lens. The PixelView card can be utilized under Linux, Windows and Dos.

## 2.2 The Processing Unit

The robot main processing unit consists of an Intel Pentium 233 MMX together with a main board and 32MB RAM. There are two serial ports onboard that are used as communication means with the control unit. A floppy disk drive is installed on the robot from which the system boots and runs the programs. Because of power supply problems and also hit sensitivity, a hard disk drive could not be applied.

## 2.3 The Control Unit

The control unit has been designed such that it can sense the robot, can inform the processing unit of the status and also fulfill the processing unit commands. Because of reduction in the number of wires which increases the robot robustness, communication between the control unit and the processing unit is done via two serial ports with RS-232 standard [3]. The control unit consists of two similar Intel 89C51 microcontroller based boards which we have designed. For further information about the Intel 89C51 microcontroller specifications, you can refer to [4]. One control unit board is represented in figure 1.

As it is shown in the figure 1, an Intel 89C51 microcontroller has been utilized to control the respective motors, kicker, encoder and limit switches. There are three power amplifiers which amplify the pulses which are generated by the microcontroller to control the speed of the motors and to drive the solenoid. The PWM pulse frequency is about 70kHz. The amplifier output is applied to the respective motor or kicker.

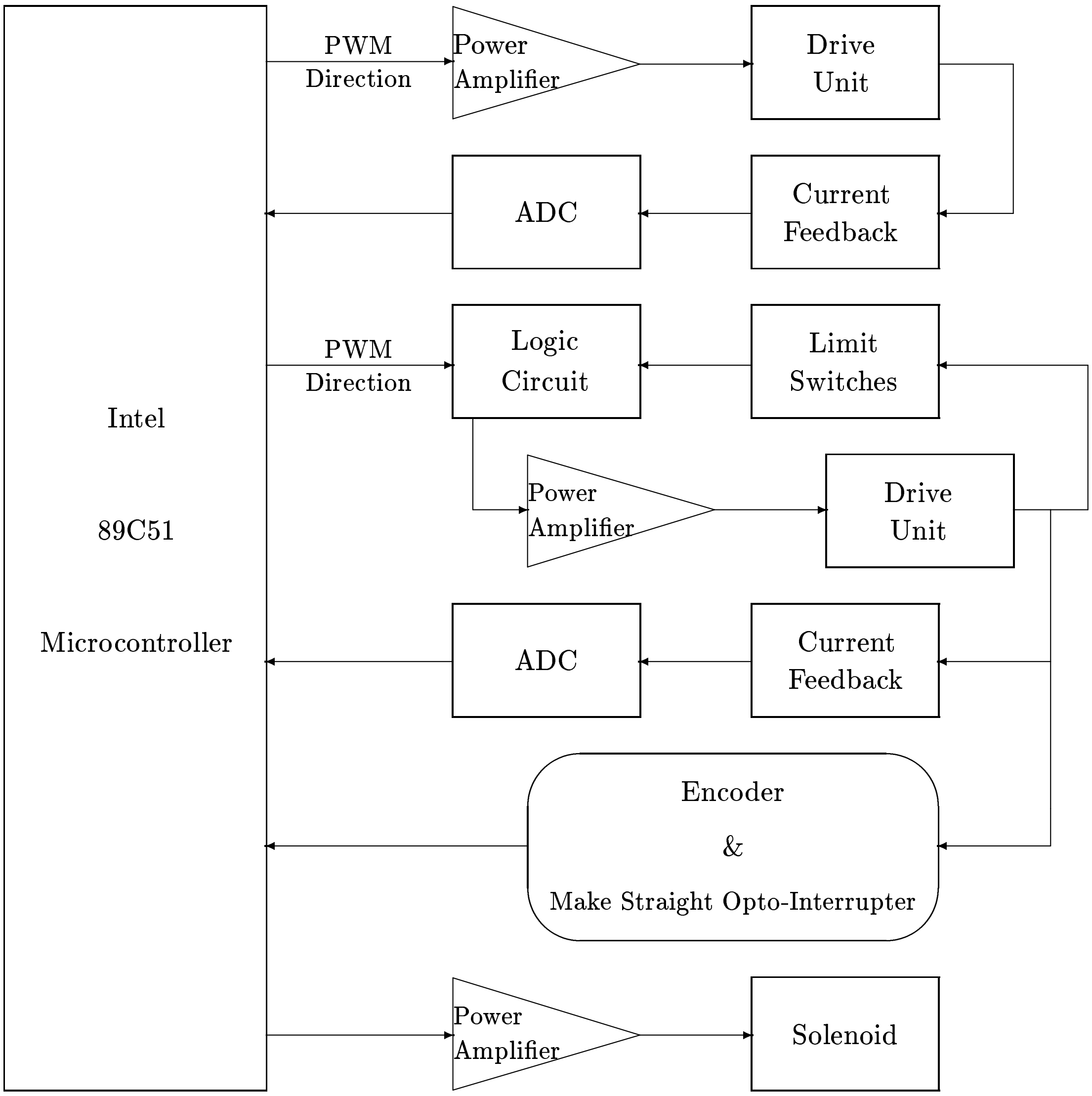


Figure 1: The **Arvand** control diagram. Each robot has two of these circuits, one for each side of the robot.

As the robot size is limited, drive units must not exceed the robot boundary limits. Therefore, two limit switches have been devised for each steer unit. These two limit switches confine the rotation domain of the drive unit by not allowing the power amplifier to apply any pulses to the steer motor in the respective directions when the drive unit has reached its limit.

The current drawn by each motor can be sampled by the microcontroller using an A/D Converter. This gives the microcontroller the capability of realizing the motors status and informing the processing unit of the robot status.

As it is mentioned in the mechanics architecture section, the rotation center point critically depends on the angle of the drive units. For the accuracy of controlling the angle of a drive unit, an encoder has been mounted directly on the respective motor shaft. One of the microcontroller duties is counting the pulses generated by this encoder. Each pulse represents 0.14 degrees of the drive unit rotation.

Hence, this architecture can provide the robot with the following capabilities:

1. Rotating around a point in the plane (formula 1) can be achieved by controlling speed of the drive units together with rotating them to an appropriate angle.
2. Controllable motor speed allows **Arvand** to move smoothly in the field. This augments the quality of **Arvand** individual technics. For instance, if **Arvand** can reduce its speed when it approaches a still ball in the field, the chance of gaining the ball increases. In addition, controllable kicking power can be useful for team play skills such as passing and stopping the ball.
3. It has been observed that a mobile robot may bump into barriers in its work field. It is an advantage for a robot to detect when it is stuck. **Arvand** control unit detects when it is stuck and alerts the processing unit to make an appropriate decision. This is done by checking the drive motors current feedback.

Finally, we are going to add a communication unit to **Arvand** which directly helps the team play algorithms.

## 3 Software Architecture

Software architecture of **Arvand** consists of three parts:

- Image Processing
- Motion Control
- Decision Making

In constructing each of the above parts and in linking them together, our developing approach is based on O.O.P. Consequently, there are three main classes. As it is described in the hardware architecture section, we could not use a hard disk. Therefore, we had two choices for our robot operating system, Linux and Dos. Because there are numerous programming tools in Dos, we chose the latter.

### 3.1 Image Processing Module

In our system, object detection is based on detecting the object color. We have experimented RGB [1] and HSI [2] color models [5] and chosen HSI model because of its advantage in representing approximately each color in a cube in HSI space. In HSI model a color can be detected by determining its domain in HSI space. To find all objects in a scene the image matrix is processed from top to bottom only once. In order to speed up this routine, instead of examining each single pixel in the image matrix, only one point from sub windows of size $m_w \times m_h$ [3] is tested. If this point has the desired color, then moves upward in one pixel step until hitting a border point. At this point a clockwise contour tracing algorithm is performed and the border points of the object are marked. If the object size is larger than a predefined amount it is recognized as an object, otherwise it is taken as a noise.

To find the next object the search is continued from the start point from which the previous object was found. In our search for the next object the marked points are not checked. At the end of this step, all objects are detected. A second algorithm will determine which objects are noise and will eliminate them. A third routine calculates all necessary information such as object type (i.e. there can be more than one object with the same color) distance and angle for "Decision Making Module" [6].

### 3.2 Motion Control Module

This module is responsible for receiving the motion commands from the "Decision Making Module" and putting the hardware to work. As it is mentioned in the hardware architecture section, the communication between the processing unit and the control unit is via two onboard PC serial ports using RS-232. So, just some basic computations are done in this module and commands are sent via serial ports to the microcontroller. The program loaded in the microcontroller is responsible for performing them. For example, some commands are kick, go(forward), go(backward), rotate(left), rotate(right), rotate_round(left, 10) (this stands for rotation around a point 10 centimeters straight from the **Arvand** geometrical center) and etc.

### 3.3 Decision Making Module

Principally, the decision making module is referred to that part of **Arvand** software that processes the results of image processing module, decides accordingly and finally commands the motion control software.

For designing this part, the first work was to develop simple **Arvand** skills such as finding an object, move towards an object up to a distinct distance and etc. After that, for simplifying the high level programming, we designed and implemented **ArvandLan** high level language. In addition to the previously implemented skills and also low level skills, some facilities were devised for creating automata and some other advanced capabilities.

[1] Red, Green, Blue

[2] Hue, Saturation, Intensity

[3] $m_w$ and $m_h$ can be the sizes of the smallest object

The time needed for a frame acquisition and its processing period is called a *time step*. The automaton state transition happens at the end of each time step, if no interrupt node has been executed in that time step.

In **ArvandLan** there are some instructions that are concisely as follows:

- **MakeNode** that is used to introduce a new node in our automaton. Each node can contain a piece of **C** code that is exactly what is executed in that node.
- **TransitionFile** that shows the file that contains the automaton transitions. Each transition determines the source and target node and the condition under which it is applied.
- **MakeGlobalNode** that is used to define a global node which is a node that has an execution condition and is executed every time its condition is satisfied (independently of the automaton status). For instance, a usage of these nodes is when the robot sees a fixed object in the field and resets its position variables.
- **MakeInterrupt** which defines an interrupt node that has some execution conditions and is executed when its conditions are satisfied. If an interrupt node is executed during one time step then the automaton transition will not happen at the end of that time step. These nodes can be used for handling exceptional conditions. For example, when the robot is stuck.

The following example is a simple program using **ArvandLan**. By this program, **Arvand** will first find the ball then catch it (move towards the ball) and finally carry it ahead.

```
// Sample Program

/MakeNode find_ball in Free.Nde
/MakeNode follow_ball in Ball.Nde
/MakeNode adjust_ball in Ball.Nde
/MakeGlobalNode chng_str in ChStr.Nde
/MakeNode init_node in Free.Nde
/MakeInterrupt go_out in Free.Nde

/TransitionFile Test.Trn

/Define DefTest 132
/GlobalVariable {Begin}
float g1;
int g2, s0, s1;
unsigned long g3;
/GlobalVariable {End}

/Define {Begin}
DFN 0
FRW 1
/Define {End}

/GlobalVariable double strv, temp_chk;
```

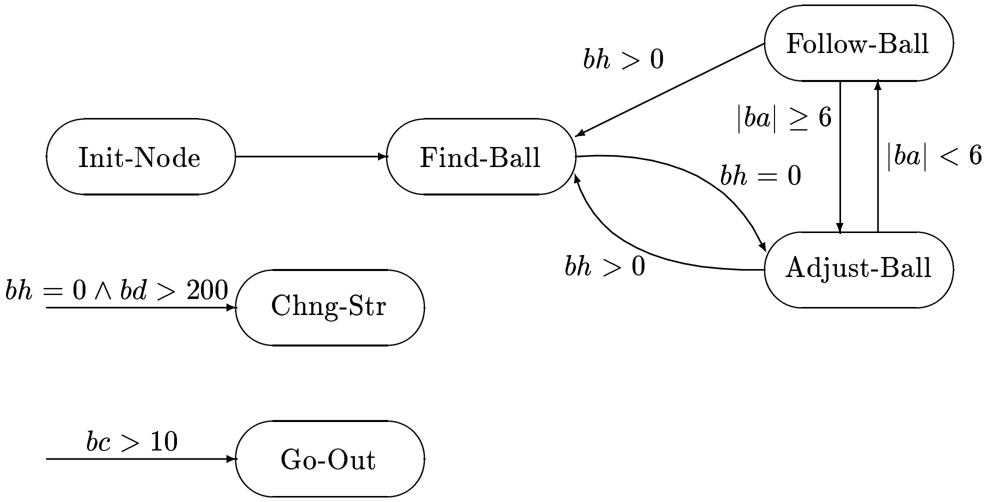


Figure 2: A sample program

```
// Sample Transition Function

[find_ball, ball_hist == 0, adjust_ball]
[adjust_ball, ball_hist > 0, find_ball]
[adjust_ball, abs(ball_angle) < 6, follow_ball]
[follow_ball, ball_hist > 0, find_ball]
[follow_ball, abs(ball_angle) >= 6, adjust_ball]
[,(ball_hist == 0) && (ball_dist > 200), chng_str]
[init_node, true, find_ball]
[,ball_cte > 10, go_out]
```

To put it in a nutshell, we omit details about modules which are called in each node. Figure 2 represents the scheme of this example (To simplify the figure we use abridged names for variables).

# 4 Conclusion

**Arvand** is the 2nd generation of robots constructed by our team. The 1st generation participated in RoboCup 98 in Paris. This new version is more advanced because of its image acquisition system, real time image processing, microcontroller based controller boards, the kicker mechanism and also its specially designed high level programming language. One advantage of **Arvand** is its mechanics capability to rotate around any point in the plane. This makes it possible for the robot to rotate around ball center while finding the goal position. In practice, this capability enabled us to implement special individual playing technics in dribbling, coming out when stuck and taking out the ball from a wall corner. Another advantage of our robot is its use of MS/DOS operating system. Because it can be executed on a floppy disk which is a reliable device on a mobile robot. Our robots showed a good performance in real test games and we are going to improve our software algorithms based on individual technics and team play. A wireless LAN system is under construction which enables our robots to communicate with each other and perform team play.

# References

[1] Shigley, J.E., *Mechanical Engineering Design*, McGraw-Hill, 1986.

[2] Meriam, J.L., *Dynamics*, John Wiley, 1993.

[3] Mazidi, M.A., and Mazidi, J.G., *The 80x86 IBM PC and Compatible Computers, Volume II*, Prentice Hall, 1993.

[4] MacKenzie, I.S., *The 8051 Microcontroller*, Prentice Hall, 1995.

[5] Gonzalez, R.C., and Woods, R.E., *Digital Image Processing*, Addison-Wesley, 1993.

[6] Jamzad, M., and others, *An Autonomous Mobile Soccer Player Robot*, Submitted to RoboCup, 1999.