MCT Susanoo Logics
2014 Team Description

Satoshi Takata, Yuji Horie, Shota Aoki, Kazuhiro Fujiwara, Taihei Degawa

Matsue College of Technology
14-4, Nishiikumacho, Matsue-shi, Shimane, 690-8518, Japan
beppu@matsue-ct.jp
http://www.matsue-ct.ac.jp/

Abstract. Our robots and systems are designed under the RoboCup SSL 2014 rules in order to participate in the RoboCup competition in Brazil. This year, we improve the dribble device of the robot and the AI architecture. This paper describes them and other systems of the robots.

1 Team outline

The MCT Susanoo Logics consists of the members of Department of Control Engineering, Electrical Engineering, and Information Engineering of Matsue College of Technology (Kosen). Our Robots with four Maxon 30 watts flat motors were totally designed by the team members and manufactured in the MCT factory. Electrical circuits boards were designed with Eagle PCB software and cut with Mits PCB Milling System. The artificial intelligence system of the robots was programed by C++ and C#.

The team has participated in the RoboCup SSL in Japan since 2011. Last year, the team participated in the RoboCup SSL 2013 in the Netherlands. That was our first world competition.

This year, we redesigned the dribble device to improve the success rate of the ball trap, and the AI architecture to shorten the developing time.

2 Hardware outline

Figure 1 shows the MCT Susanoo Logics' 2014 model robot.

The dribble device is the most important modification point of this year. The absorption method of the kinetic energy of the ball was modified from the vertical motion of the dribble bar of the dribble device to the rotary motion of the dribble device (Fig. 2). The modification improves the success rate of the ball trap. We changed the structure of the dribble device to minimize the assembly time (Fig. 3).

To prevent from the fiber or the field, the shape of the bottom plate was modified to cover the gear of the wheels (Fig. 4).

The nylon Omni wheels of the 2013 model were not robust. We changed the material of the wheel to duralumin. Table 1 shows specifications of the robot.

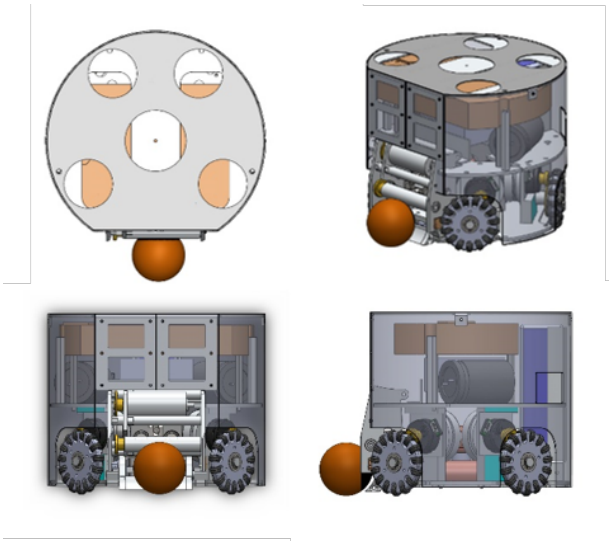
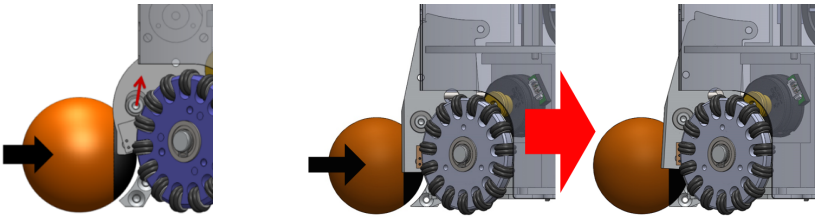
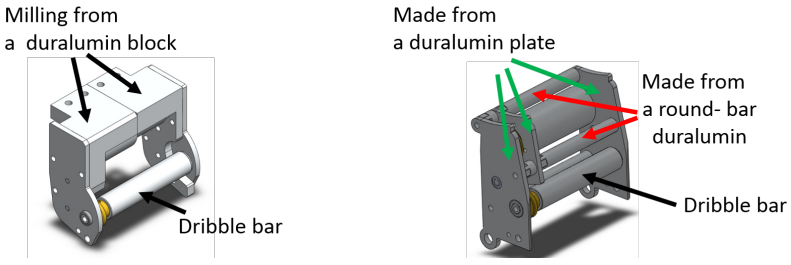


Fig. 1: External shape of a robot



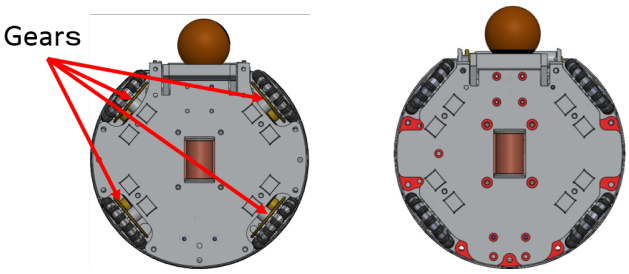
(a) 2013 model (b) 2014 model

Fig. 2: Ball receiving process



(a) 2013 type (b) 2014 type

Fig. 3: Dribble device



(a) 2013 type (b) 2014 type

Fig. 4: Bottom view of the robot

Table 1: Specifications of the robot

Height mm	: 148.5
Diameter mm	: 177
Weight kg	: 2.5
Maximum robot speed m/s	: 3.0

3 Electronics

A schematic block diagram of the machine is shown in Fig.5.

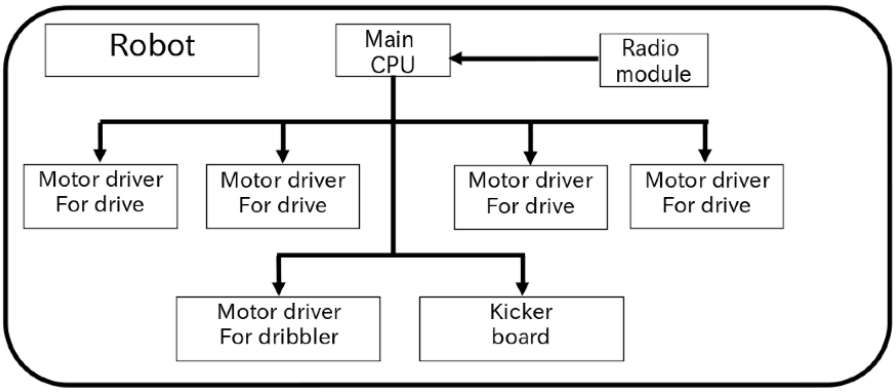


Fig. 5: Schematic Layout showing the main components

Battery Electrical energy is supplied to the circuit boards by a 14.8 V 4-cell LiPO battery.

Main board A microchip (dsPIC33FJ32GP202) is implemented to the machine as the main CPU. The main CPU clock is internal oscillator with a frequency of 80 MHz.

Communication Communication from the AI to each machine is via a 2.4GHz band radio module Xbee 802.15.4. The vision system sends data every 1/60 th of a second. Therefore, the communication time length has to be shorter than 1/60 th of a second. The Xbee's baud rate is set at 115200. It means that the AI sends data to all machines in 8 ms. The AI sends instruction data to the machines by broadcast. Instruction data include machine velocity, angular velocity and command data. Command data is to control the kicking and dribbling devices.

Ball sensor The ball sensor senses when a ball comes in front of the machine and indicates the possibility that the machine can kick the ball. Infrared LED and photo-transistor pair are implemented in front of the machine.

Velocity calculation for each wheel Each machine has four omni-wheels. The AI sends machine velocity as instruction data. Therefore, the machine must calculate the velocity for each wheel from machine velocity. The main CPU calculates the velocities and sends the data to each motor driver board. We apply I2C to communication between the main CPU and the motor driver boards.

Power supply The main board has a voltage converter for power supply. A DC-DC converter V-INFINITY V7805-1000 converts the 14.8 V voltage from the battery to 5V. The 5 V supply is used by the MOS-FET gate driver IC and rotary encoder. A linear regulator converts part of the 5 V supply to 3.3 V to power the dsPIC and radio module XBee.

4 Kicker board

Figure 6 shows the diagram of the kicker system. The system consists of a main CPU, a kick driver, a voltage booster, a drive capacitor (C_D) of 2 * 2200 uF, a IGBT, a solenoid, and a kick device. The voltage booster boosts the battery voltage of 14.8 V to 210 V and charges the drive capacitor (C_D). The IGBT drives the solenoid to kick the ball with the kick device. The voltage booster charges the drive capacitor (C_D) from 14.8 V to 210 V within 3.4 seconds (Fig. 7). The voltage of the drive capacitor (C_D) after the kick is usually 160 V, thus the recharge time is 1.2 seconds. A capacitor charger (Linear Technology LT3750) is used for the boost converter. For adjusting the strength of the ball kick, the main CPU controls the gate-on time of the IGBT for the drive capacitors.

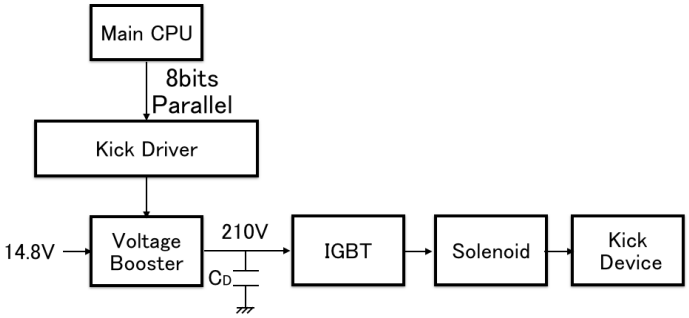


Fig. 6: Kicker system

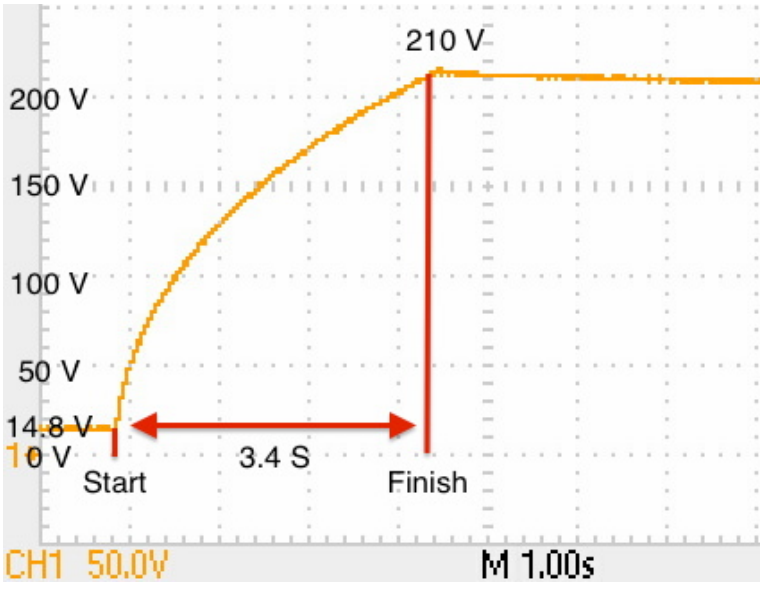


Fig. 7: The charge response of the drive capacitor (C_D)

5 Motor control

Figure 8 shows the control system of the MCT susanoo logics' robot. Where, M is the motor. The system have 4 Motor drive units for brush-less DC motors. The main CPU receives speed vector of the robot from AI computer, and calculates 4 motor speeds.

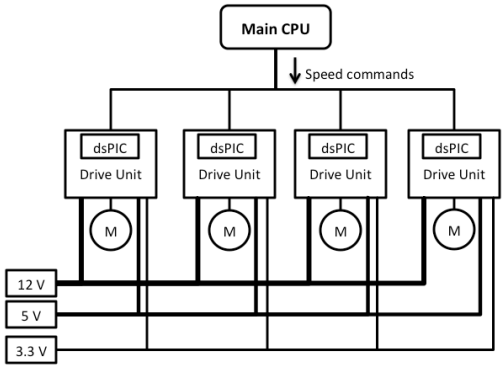


Fig. 8: The control system of the MCT Susanoo logics' robot

Figure 9 shows the motor drive unit of the MCT Susanoo logics' robot. The motor drive unit consists of a dsPIC33FJ12MC202 and a 3-phase H-bridge with MOSFETs. The dsPIC receives speed commands from main CPU and outputs the 3-phased PWM signal to the 3-phase H-bridge. The 3-phase H-bridge drives the brush-less DC motor (Maxon EC45 Flat 30W). For the detection of the rotation speed of the motor, a rotary encoder (US Digital E8P 1440 PPR) was attached to the back of the motor.

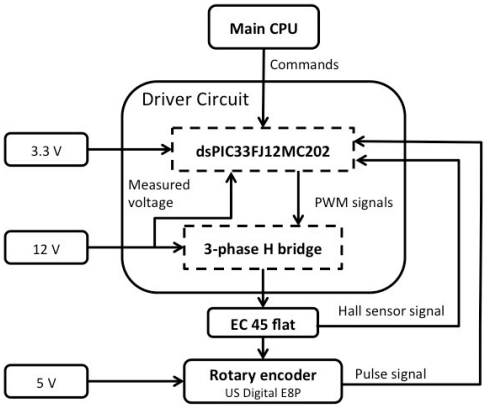


Fig. 9: The motor drive unit of the MCT Susanoo logics' robot

Figure 10 shows the torque controller. We introduce the torque control methods of Skuba's ETDP 2011 [1] into the torque controller. Equations (1) through (3) describe the control law.

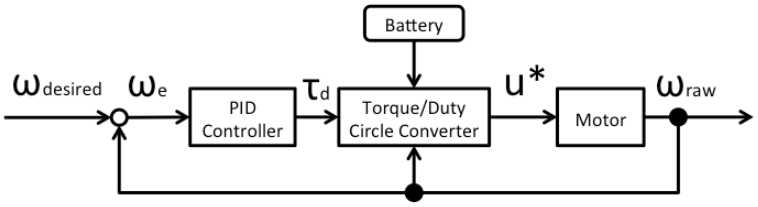


Fig. 10: The torque controller of the MCT Susanoo logics' robot

$$\omega_e[i] = \omega_{desired}[i] - \omega_{raw}[i] \tag{1}$$

$$\tau_d[i] = k_p \cdot \omega_e[i] + k_i \cdot \sum_{i=1}^{N}(\omega_e[i]) + k_d \cdot (\omega_e[i] - \omega_e[i-1]) \tag{2}$$

$$u^*[i] = \frac{\tau_d[i]}{(\frac{k_m}{R}) \cdot V_{cc} - (\frac{k_m}{R \cdot k_n}) \cdot \omega_{raw}} \tag{3}$$

where,

k_m is the motor torque constant
k_n is the motor speed constant

R	is the motor coil resistance
N	is the number of samples
V_{cc}	is the driver supply voltage
u^*	is the duty cycle of the PWM control signal

6 Robot control method

Figure 11 shows the velocity calculater. The AI computer calculates the target position. The velocity calculater decides the speed vector of a robot from the target position P_{target}, current position $P_{current}$, and current robot's speed vector $V_{current}$.

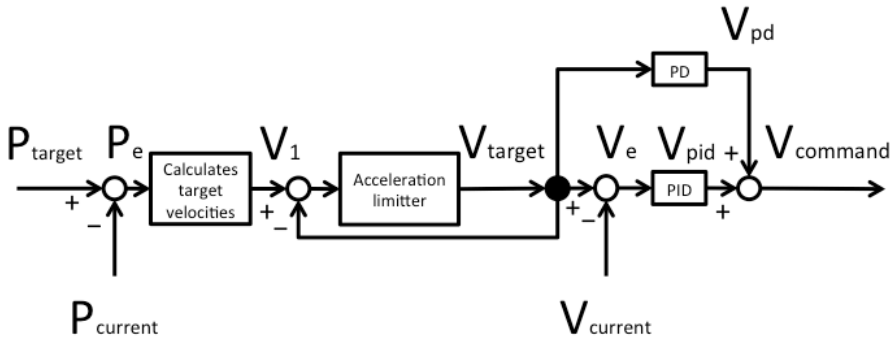


Fig. 11: The velocity calculater

The position error P_e is calculated from equation (4).

$$P_e[i] = P_{target}[i] - P_{current}[i] \tag{4}$$

The speed vector V_1 is calculated from equation (5).

$$V_1[i] = K \cdot P_e[i] \tag{5}$$

The target speed vector V_{target} is calculated from equations (6) through (8). These equations prevent the robot from moving by rapid acceleration.

$$Acc[i] = V_1[i] - V_{target}[i-1] \tag{6}$$

$$Acc[i] = \begin{cases} AccLimit & \text{if } Acc[i] > \text{threshold}, \\ Acc[i] & \text{otherwise} \end{cases} \tag{7}$$

$$V_{target}[i] = V_{target}[i-1] + Acc[i] \tag{8}$$

The speed vector error V_e is calculated from equation (9).

$$V_e[i] = V_{target}[i] - V_{current}[i] \tag{9}$$

The velocity calculater uses PD and PID controller for reducing the error V_e. Equation (10) describes the method for calculating the output of PID controller V_{pid}. PID controller calculates V_{pid} from the error V_e.

$$V_{pid}[i] = k_{p1} \cdot V_e[i] + k_{i1} \cdot \sum_{i=1}^{N}(V_e[i]) + k_{d1} \cdot (V_e[i] - V_e[i-1]) \tag{10}$$

Equation (11) describes the method for calculating the difference V_{diff} between current and previous V_{target}.

$$V_{diff}[i] = V_{target}[i] - V_{target}[i-1] \tag{11}$$

Equation (12) describes the method for calculating the output of PD controller V_{pd}. PD controller calculates V_{pd} from the difference V_{diff}.

$$V_{pd}[i] = k_{p2} \cdot V_{diff}[i] + k_{d2} \cdot (V_{diff}[i] - V_{diff}[i-1]) \tag{12}$$

The speed vector $V_{command}$ is calculated from equation (13).

$$V_{command}[i] = V_{pid}[i] + V_{pd}[i] \tag{13}$$

7 AI architecture

Our AI is mainly composed of Behavior Tree and STP (skills, Tactics and Play). Behavior Tree is a system that divides AI behavior into units named 'Behavior' and a tree of Behavior nodes controls the AI. Conceptual diagram of Behavior Tree is shown in Fig. 12.

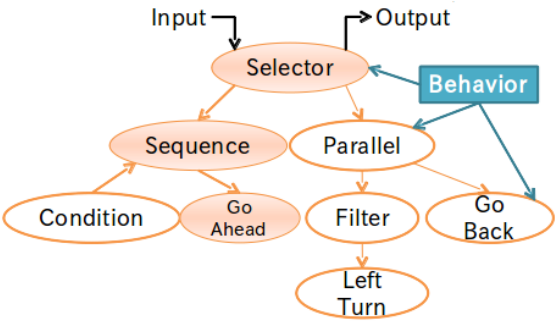


Fig. 12: Conceptual diagram of Behavior Tree

The Behavior consists of three states, Ready state, Finish state and Running state. Finish state can be divided into two states, Success state and Failure state. A Behavior can change a state of children node into Ready state. Behavior can classify into Action, Decorator and Composite. The Action must be a leaf node. “Go Ahead”, “Left Turn” and “Go Back” Behavior in Fig. 12 are Action. The Composite must has one or more children nodes and it expresses sequence of plural Behaviors. “Sequence” and “Parallel” in Fig. 12 are The Composite. The Decorator must have only one child node and it processes behavior of the child node. “Filter” in Fig. 12 is the Decorator.

There are two reasons why we use the Behavior Tree. One: It is easy to extend and to develop on a team. Two: GUI tools can be introduced to create Behavior Tree, so to write programs is unnecessary.

We could easily create Behavior Tree by using Behavior Builder shown in Fig. 13.

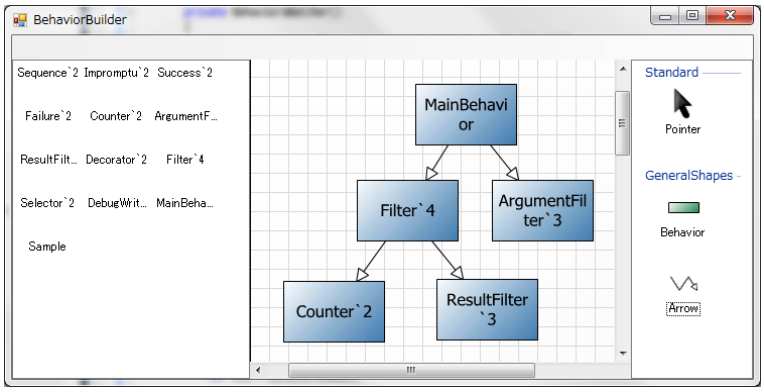


Fig. 13: Behavior Builder

Nevertheless, it is hard to management it if a Behavior Tree grows too much. Therefore, we use STP to solve this problem. This architecture is shown in Fig. 14.

Team Behavior Tree (Team BT) assigns Behavior Tree to each Agent and outputs its role and end condition. Agent’s Behavior Tree outputs the commands that robots can execute. Each agent continues to input until fulfilling the end condition.

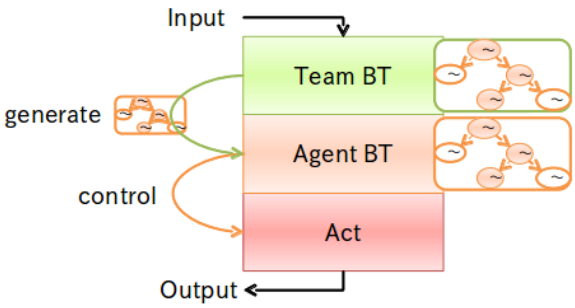


Fig. 14: Our architecture

References

1. Krit Chaiso, Kanjanpan Sukvichai :
Skuba 2011 Extended Team Description, Kasetsart University, Thailand, 2011.