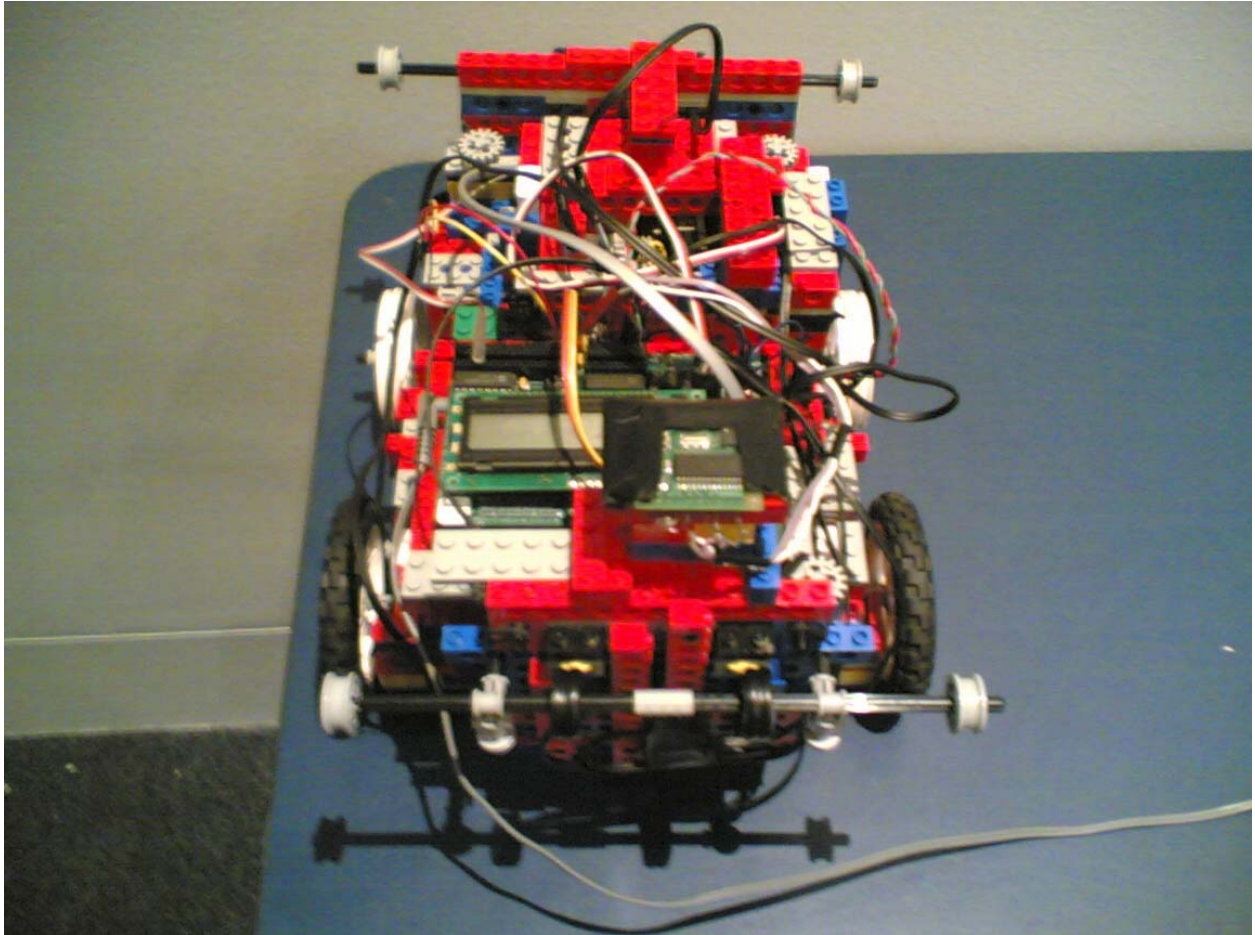


**CSCI 445 Fall 2004
RoboSoccer Competition Final Report**

COMPASS LOCALIZATION



Gautam Dandavate
Siu Yu Kwok
(Team: Robots 1, Humans 0)

1 RoboSoccer Localization

In robotics, localization is necessary to determine the robot's position relative to its environment, and thereby utilize the robot more efficiently. With this information, a robot can decide on where it should go or what it should do based on its task. For a RoboSoccer competition, which was the final project of the CSCI 445 course, localization is critical to a team's success. The competition was a soccer match played by two teams of robots. Each team had one striker, to attack the opposing team's goal, and a goalie, to defend its own goal. The game was played on a bounded rectangular field with the dimensions shown in Figure 1.

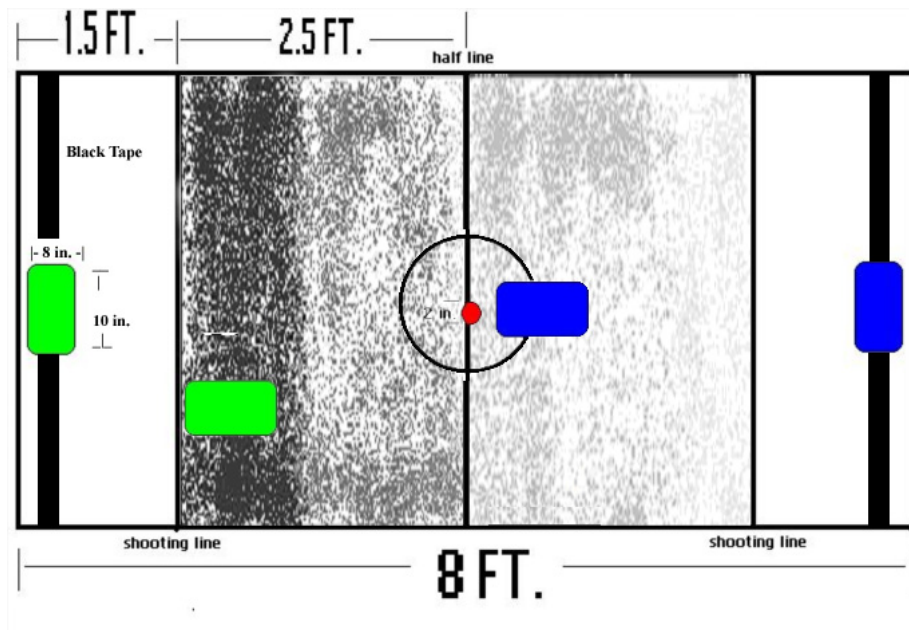


Figure 1: RoboSoccer Field

A team's performance is significantly enhanced if the striker has some method of localization. The three parameters that completely define the striker's position on the field are its x -position, y -position and heading θ . Determining the striker's heading is important because it can then identify which side it needs to attack.

2 Sony IR Emitters and IR Sensors

In order to assist in localization, Sony IR emitters were placed on the four corners of the field, each emitting unique frequencies. An IR sensor, present on the HandyBoard controller, could then be used to orient the robot in the attacking direction by looking for the frequencies corresponding to the attacking sides.

However, testing the IR emitter/sensor setup showed that the Sony-format IR transmission does not function well. The striker's IR sensor was sometimes unable to detect the emitted frequency from a corner even though it was pointing straight at it. It was also observed that the transmitter IR signals reflected off the boundaries of the field, thereby misguiding the robot. Additionally, the

range of the emitted signal was found to be poor. To solve this problem, a Devantech magnetic compass was used by the striker instead.

3 Magnetic Compass

The Devantech magnetic compass CMPS03, shown in Figure 2, is manufactured by Robot Electronics and used in robots for navigation purposes. It uses a Philips KMZ51 magnetic field sensor to detect the Earth's magnetic field. The compass gives the direction of the horizontal component of the prevailing magnetic flux, and provides a unique number that signifies the compass's heading. The compass is available for purchase from the Acroname website:

<http://www.acroname.com>



Figure 2(a): Compass (Front)



Figure 2(b): Compass (Back)

3.1 Compass Specifications

The Devantech magnetic compass requires a 5 V power supply at 15 mA in order to work. The compass's heading can be obtained either from a pulse width modulated (PWM) signal or through an I2C interface. The PWM signal technique was used and is described here.

Figure 3 shows the sample output of the compass's PWM pin (Pin 3). The width of the pulse corresponds to the heading; 1 ms corresponds to 0°, 36.99 ms corresponds to 359.9°. The resolution of the compass is 0.1°, but the listed accuracy is 3-4°. Therefore, although the values returned by the compass change increments of 0.1°, the actual bearing can differ from the measured bearing up to 3-4°.

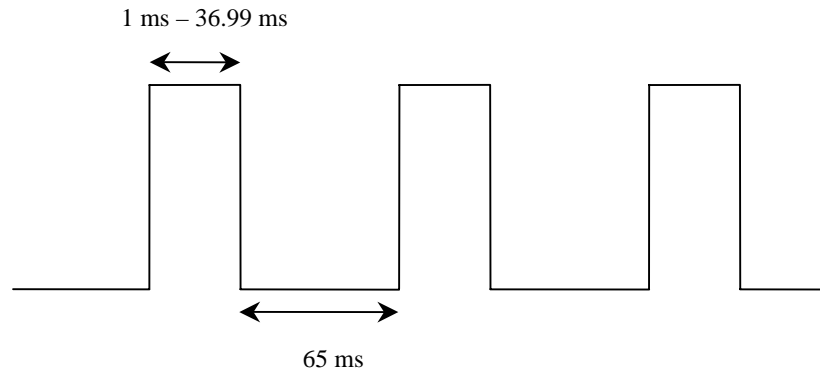


Figure 3: PWM Signal output from compass

Another issue regarding the compass is that it is affected by motors, magnets, or any ferrous objects. The compass will return different bearing measurements near a motor or away from a motor for the same orientation. Robot designers can minimize this problem by mounting the compass as far away from any magnetic or ferrous objects as possible.

3.2 Using the Devantech Magnetic Compass with the HandyBoard and Interactive C

In order to interface the compass with HandyBoard, the controller's internal TIC (Timer Input Capture) feature is used to measure the length of the PWM signal. A driver written by Andrew Chanler, a student from the University of Massachusetts-Lowell in Fall 2003, was used. The Interactive C driver files can be obtained from his website:

<http://www.cs.uml.edu/~achanler/robotics/compass/compass.html>

The workings of the driver are also described on his website. Restating what is mentioned on the website, the PWM output signal is sent to the HandyBoard's digital input 7 because it is directly connected to one of the 68hc11's TIC units. An interrupt is enabled when the signal on digital input 7 goes from low to high and the time for the low to high transition is saved. Next, the TIC unit is enabled to respond to a high to low transition and this time is also recorded. The difference between is the length of the pulse in E clocks, where 1 E clock is 0.5 μ s. This number is converted to milliseconds, and then to degrees by subtracting one and multiplying by 10. The internal timer that is measuring the pulse wraps around every 32 ms and the maximum pulse can be 37 ms.

3.3 Instructions for using compass

The following steps should be followed in order to use the Devantech magnetic compass with the HandyBoard:

Step 1: Wire the compass, as shown in Figure 4. Soldering of the pins to wires will be necessary.

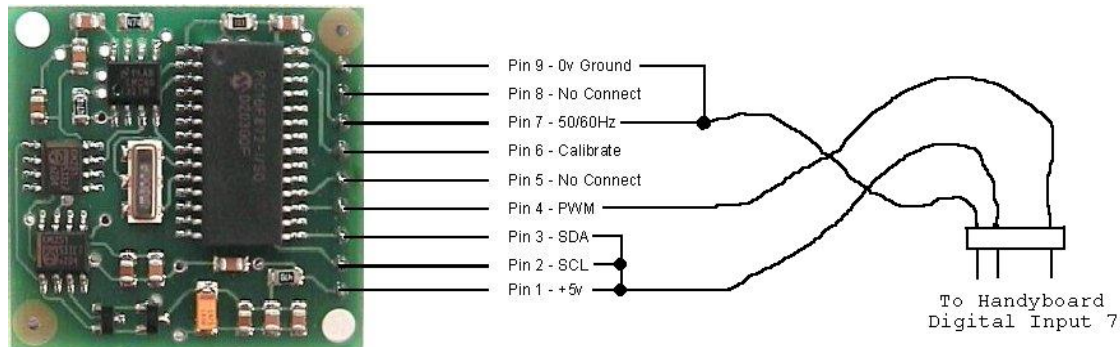


Figure 4: Interface to HandyBoard

(image taken from <http://www.cs.uml.edu/~achanler/robotics/compass/compass.html>)

Step 2: Connect the wire to the HandyBoard's digital input 7.

Step 3: Mount the compass on the robot. The compass should be kept as far away from any motors or solenoids as possible. The compass module should be horizontal with the PCB, parallel to the earth's surface. Also, the IC's should be on top. Figure 5 shows the mounting of the compass on the striker used by the team "Robots 1, Humans 0" in the CSCI 445 Fall 2004 RoboSoccer competition.

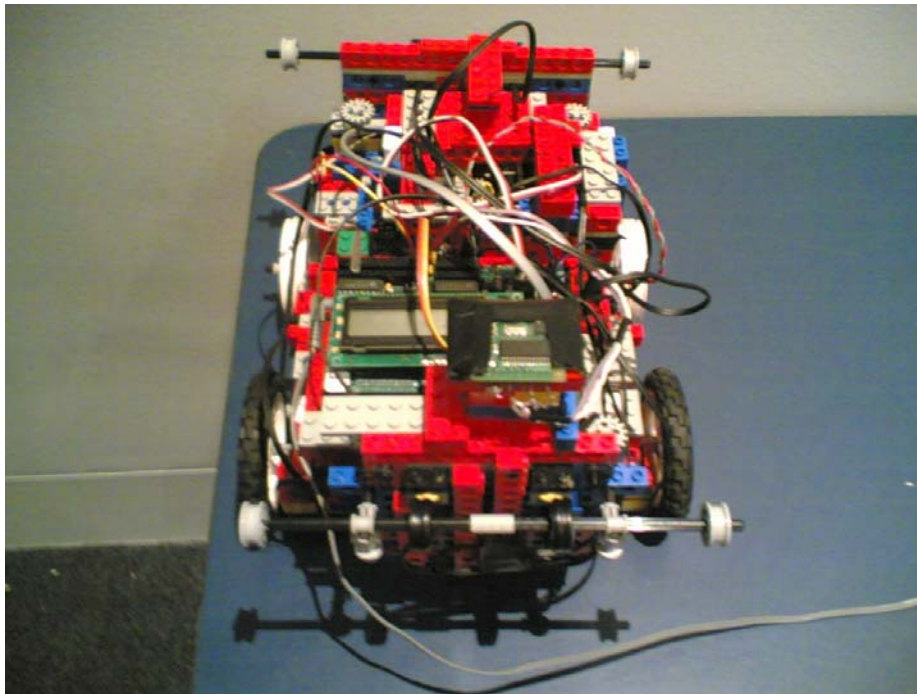


Figure 5: Compass mounted on striker

Step 4: Download Andrew Chanler's Interactive C compass driver files from his website:

<http://www.cs.uml.edu/~achanler/robotics/compass/compass.html>

Download the files "alc_cmps03.ic" and "alc_cmps03.icb" by clicking on the appropriate links, and save them to the same folder as the Interactive C program that will use the compass's measured heading (Refer to appendix to see code).

Step 5: Add the line "#use "alc_cmp03.ic" at the top of the Interactive C program. In order to obtain the compass's heading, simply make the function call "compass()". This function returns an integer from 0-359, referring to the bearing angle in degrees.

3.4 RoboSoccer Localization using Magnetic Compass

Calibration of the compass was done prior to the contest to determine the headings for the attacking direction and the defending direction. The compass was found to work much better than the IR sensors in obtaining the robot's orientation. The compass readings were not as noisy as the IR sensors. While the IR sensors could not satisfactorily guide the robot in the correct direction while attacking, the compass was successful in providing the direction to the opposition's goal. Although the striker faced several setbacks in other issues, such as ball finding, shooting, etc., no problems were obtained for localization by means of the compass.

3.5 Other Uses for Magnetic Compass

As mentioned before, magnetic compasses are commonly used for localization and navigation purposes. Since these two tasks are required in almost all mobile robot projects, magnetic compasses in several applications. Examples of such uses are in maze navigation...

4 References

1. *Deveantech Ltd. CMPS03 Magnetic Compass.*
http://www.robot-electronics.co.uk/shop/Compass_CMPS032004.htm
2. *Devantech Magnetic Compass - CMPS03. Andrew Chanler.*
<http://www.cs.uml.edu/~achanler/robotics/compass/compass.html>

5 Appendix

alc_cmps03.ic

/*

By Andrew Chanler - Fall 2003

Handyboard driver for using Devantech Magnetic Compass - CMPS03
refer to <http://www.cs.uml.edu/~achanler/robotics/>
*/

#use "alc_cmps03.icb"

int compass()

{

 int x;
 float z;
 int y;
 float degree;

 x = compass_driver(0);
 z = ((float) (0x0F & x))+(256.0*((float) (x >> 8))) ;

 if((z < 7440.0)&&(compass_wrap))
 z+=65536.0;

 degree = ((z - 1792.0)*360.0)/71183.0;
 y = (int) (degree+.5);
 y = y % 360;
 return y;

}