Inverted Pendulum Design With Hardware Fuzzy Logic Controller

Eric Minnaert
Electrical and Computer Engineering, South Dakota School of Mines and Technology
Rapid City, SD 57701, USA

Brian Hemmelman
Electrical and Computer Engineering, South Dakota School of Mines and Technology
Rapid City, SD 57701, USA

Dan Dolan
Mechanical Engineering, South Dakota School of Mines and Technology
Rapid City, SD 57701, USA

ABSTRACT

An inverted pendulum robot has been designed and built using a fuzzy logic controller implemented in a Field Programmable Gate Array (FPGA). The Mamdani fuzzy controller has been implemented using integer numbers to simplify its construction and improve system throughput. An accelerometer and rate gyroscope are used along with a complementary filter to monitor the state of the robot. Using angular velocity and angle error the fuzzy controller can successfully balance the inverted pendulum robot.

Keywords: Fuzzy logic, fuzzy controller, robot, inverted pendulum, FPGA, accelerometer, gyroscope, complementary filter, real-time computing.

INTRODUCTION

The inverted pendulum problem is a classic control systems problem [1,2]. Maintaining an equilibrium position of the pendulum pointing up is a challenge as this equilibrium position is unstable. As the inverted pendulum system is nonlinear it is well-suited to be controlled by fuzzy logic. Typically a fuzzy controller is implemented in software running on some form of microprocessor. Here, however, we demonstrate how an integer-based fuzzy controller can be directly implemented as hardware in a Field Programmable Gate Array (FPGA). Sensor signal preprocessing has also been integrated into the FPGA.

SYSTEM DYNAMICS

The balancing robot designed uses two planetary gear motors mounted under a two circuit board frame with the battery pack attached between the two PCBs (Figure 1.) As the battery is a significant portion of the mass of the robot it strongly influences the position of the center of gravity.

To properly understand the behavior of the inverted robot we need to develop the system equations that define its dynamic behavior. Figure 2 illustrates the free body diagrams of the robot's wheel and the body of the robot. The diagrams show all the forces and moments acting on the bodies so it is possible to write the system equations. A summary of all system equation variables is given in Table 1. The sum of forces on the pendulum about an axis perpendicular to the pendulum can be

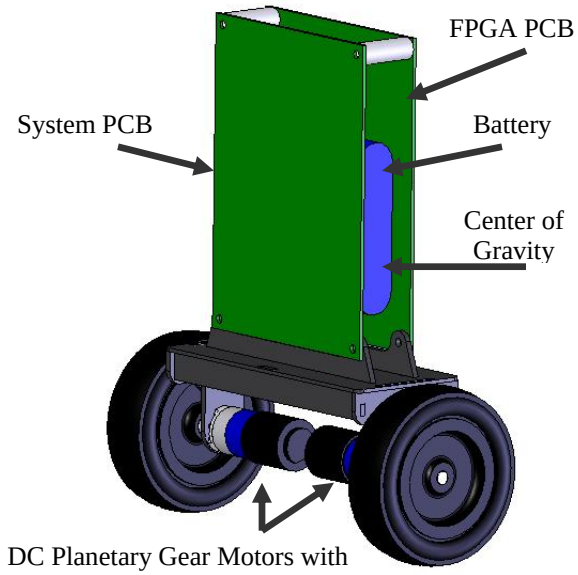

Figure 1. CAD Drawing of Balancing Robot.

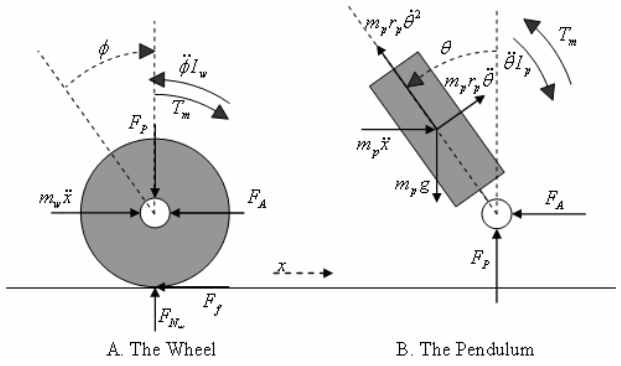

Figure 2. Free body diagram of the robot's body and wheel.

Table 1. List of terms and variables.

ϕ	angular position of the wheel
$\dot{\phi}$	angular velocity of the wheel
$\ddot{\phi}$	angular acceleration of the wheel
θ	angular position of the pendulum
$\dot{\theta}$	angular velocity of the pendulum
$\ddot{\theta}$	angular acceleration of the pendulum
x	position of the robot
$\dot{x}$	velocity of the robot
$\ddot{x}$	acceleration of the robot
F_N	normal force of the pendulum
F_{N_w}	normal force of the wheel
F_P	force P
F_A	force A
g	acceleration of gravity
I_p	pendulum inertia with respect to theta
I_w	wheel inertia with respect to phi
m_w	mass of the wheel
m_p	mass of the pendulum
r_p	radius of the pendulum
r_w	radius of the wheel
T_m	torque produced by the motor

written as

$$m_p \ddot{x}\cos(\theta) + m_p r_p \ddot{\theta} + F_N \sin(\theta) + F_A \cos(\theta) = g m_p \sin(\theta). \quad (1)$$

The sum of moments about the center of gravity of the pendulum is

$$F_N r_p \sin(\theta) + F_A r_p \cos(\theta) = I_p \ddot{\theta} - T_m \quad (2)$$

and linear acceleration of the robot can be related to angular acceleration of the wheel by

$$\ddot{x} = \ddot{\phi} r_w. \quad (3)$$

The DC motor torque is related to the applied voltage by

$$T_m = \frac{k_m V}{r_m} - \frac{k_m k_e V}{r_m} \quad (4)$$

allowing us to solve for pendulum angular acceleration as

$$\ddot{\theta} = \frac{g r_p m_p r_m \sin(\theta) - \ddot{\phi} r_p m_p r_w r_m \cos(\theta) - k_m k_e \dot{\phi} + k_m V}{r_m (I_p + m_p r_p^2)}. \quad (5)$$

The sum of forces on the pendulum with respect to the x axis can be written as

$$F_A + m_p \ddot{x} + \ddot{\theta} m_p r_p \cos(\theta) = m_p r_p \dot{\theta}^2 \sin(\theta). \quad (6)$$

The sum of moments of the wheel about its center is

$$\ddot{\phi} I_w = T_m + F_f r_w \quad (7)$$

and the sum of forces on the wheel about the x axis is

$$F_A = m_w \ddot{x} - F_f \quad (8)$$

allowing us to solve for wheel angular acceleration as

$$\ddot{\phi} = \frac{\ddot{\theta} r_p m_p r_w r_m \cos(\theta) - \dot{\theta}^2 r_p m_p r_w r_m \sin(\theta) - k_m k_e \dot{\phi} + k_m V}{r_m (I_w + m_p r_w^2 + m_w r_w^2)}. \quad (9)$$

Equations 5 and 9 form the system equations for the robot.

SENSORS AND SENSOR FUSION

An Analog Devices rate gyro (ADXRS150) and accelerometer (ADXL202E) were used to make the inertial measurements. The rate gyro has a single analog output for the rotation in the z axis. The gyro is mounted to match the rotation of the robot. The rate gyro measures angular velocity and outputs a voltage of

$$v_g = \dot{\theta} + f(T) + e_g \quad (10)$$

where f(T) represents the effect of temperature and e_g represents error, which is not known. As the rate gyro is sensitive to temperature it provides an analog output where temperature can be measured. A PIC microcontroller reads the temperature of the rate gyro regularly, converts it to the offset, f(T), and updates the FPGA with the offset value (Figure 3). The FPGA then subtracts the offset from the sampled value of the rate gyro. Since e_g is not known it cannot be subtracted from the signal and so the remaining value of angular velocity will be known as $\dot{\theta}_g$ which is not guaranteed to be the same as the actual angular velocity $\dot{\theta}$. The values are left in the measured units to save computation time in the FPGA.

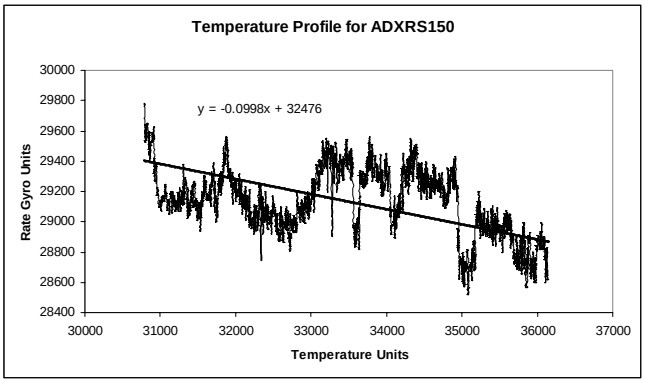

Figure 3. Temperature profile for ADXRS150.

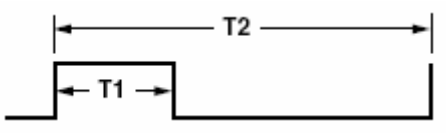

Figure 4. Accelerometer output wave (one channel).

The ADXL202E has two PWM outputs (Figure 4); one represents the acceleration in the x axis and the other represents the acceleration in the y axis. The acceleration on a given axis is computed from the PWM signal by

$$a = \left(\frac{T_1}{T_2} - 0.5 \right) 8 . \tag{11}$$

A counter in the FPGA measures the high and low time of each of the inputs and converts them to acceleration. Figure 5 shows how the two input axes of the accelerometer are oriented with respect to the robot and gravity vector. Assuming the only acceleration on the robot is the acceleration of gravity, the angle of the robot can be calculated directly from the two outputs

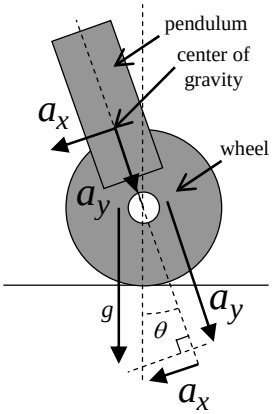

Figure 5. Accelerometer configuration.

of the accelerometer using the arctangent function. Because of this assumption the angle computed from the accelerometer will be referred to as θ_a:

$$\theta_a = \tan^{-1}\left(\frac{a_x}{a_y} \right). \tag{12}$$

The arctangent is computed with a CORDIC (**Co**ordinate **R**otation **Di**gital **C**omputer) algorithm. The CORDIC algorithm uses a series of shifts and adds (Equations 13 through 15) which are significantly less demanding than computing products and powers needed by an infinite series approach [3].

$$x_{i+1} = x_i - y_i d_i 2^{-i} \tag{13}$$

$$y_{i+1} = y_i + x_i d_i 2^{-i} \tag{14}$$

$$z_{i+1} = z_i - d_i \tan^{-1}\left(2^{-i}\right) with\ d_i = \begin{cases} -1 & if\ z_i < 0 \\ +1 & otherwise \end{cases} \tag{15}$$

COMPLEMENTARY FILTER

With the rotation, $\dot{\theta}_g$, and angle, θ_a, information a complementary filter algorithm in the FPGA computes a close estimate of the real angle of the robot. For clarity, the subscripts a, g, and c denote outputs from the accelerometer, rate gyro, and complementary filter respectively. A complementary filter is used because the rate gyro signal, $\dot{\theta}_g$, cannot be integrated directly without diverging to infinity because e_g is not known. The ideal discrete integration of the rate gyro output would be

$$\theta_g(k) = \theta_g(k-1) + \frac{\dot{\theta}_g(k)}{\Delta t} \tag{16}$$

where k represents the discrete sample number. Also the angle signal , θ_a, cannot be used directly because it is sensitive to disturbances and centripetal acceleration of the pendulum [2].

The complementary filter removes elements of two signals, θ_g and θ_a, that are dissimilar and keeps elements that are similar. It does this be removing a small amount of error, e_c, on each computation cycle of the filter. As mentioned previously, most, but not all, of the offset has been removed from the rate gyro signal and so the amount of error removed by the complementary filter, e_c, must be larger than the maximum amount of remaining offset, e_g. Without meeting this criterion the error will accumulate and diverge to infinity. The complementary filter computes the angle of the robot as

$$\theta_c(k) = \theta_c(k-1) + \frac{\dot{\theta}_g(k)}{\Delta t} - e_c(k) \tag{17}$$

where

$$e_c(k) = \theta_c(k-1) - \theta_a(k-1). \tag{18}$$

Figure 6 illustrates how the complementary filter is able to successfully track the actual angle of the robot without diverging to infinity.

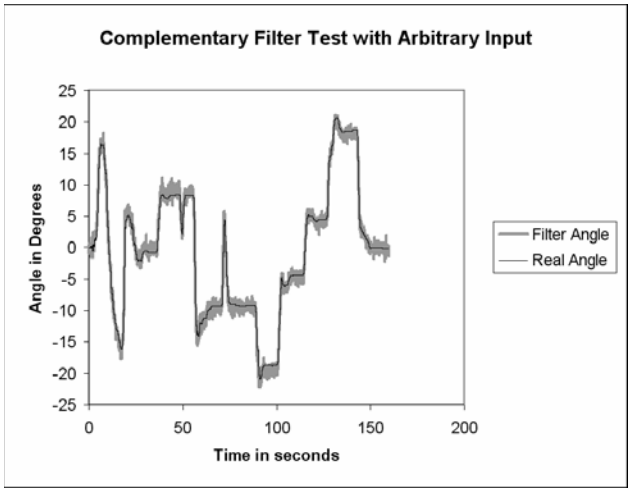

Figure 6. Complementary filter test data.

FUZZY LOGIC CONTROLLER

A fuzzy logic controller is an excellent choice for an inverted pendulum because it can easily be adapted to the system. There are several different fuzzy logic models. We have used the Mamdani model. Fuzzy controllers deal with inputs as if they are inexact quantities or "fuzzy" inputs [4]. An example of this would be temperature. A room may be 68°F, but in terms of a fuzzy controller it is considered to be a small amount "cold", a large amount "warm", and a medium amount "hot".

In most fuzzy systems, the degrees of membership that each input belongs to the various membership functions is represented as a fractional value between 0 and 1. For the example above, the temperature degrees of membership might be 0.05 "cold", 0.7 "warm", and 0.25 "hot". The precision of the degree of membership may be a 32, 64, or 80 bit floating point number. The hardware needed to deal with these numbers is not simple or small. By using degrees of membership in a range other than 0 to 1, the fuzzy logic controller can be made with significantly less hardware and can run at much higher sample rates [5, 6].

If the degrees of membership ranged from 0 to 80 in integer values, the temperature of the previous example would be 4 "cold", 56 "warm", and 20 "hot". Although the numbers will not be represented as accurately, the fuzzy logic controller will perform the same basic operations. The concept of a fuzzy controller maintains that the inputs are inexact quantities anyway, so the numerical precision may not be as important as the rules and update rate of the fuzzy controller. When using integer numbers, the fuzzy controller will only need integer arithmetic operations instead of floating or fixed point arithmetic. Also, the registers used to hold the numbers will be smaller.

For the inverted pendulum robot, there are two inputs to the fuzzy logic controller. The first is the angular velocity or the first derivative of angle with respect to time. The second input is the angle error of the robot. The error is the difference between the angle of the robot and the desired angle of the robot. For the purpose of this paper the desired angle is zero degrees, so the angle error is -θ. As can be seen from Figure 2, θ is 0 when the pendulum is upright.

The inputs θ and $\dot{\theta}$ must be adjusted to fit the fuzzy controller before they are fed into it. Figures 7 and 8 show the membership functions for the θ and $\dot{\theta}$ inputs. The fuzzy controller utilizes 8-bit unsigned integers for inputs so the actual θ and $\dot{\theta}$ values must be adjusted accordingly to fit within a range of 0 to 255. After some testing it was determined that the robot would be able to maintain control over a window of $-10 < \theta < +10$ degrees. Thus the θ input was simply multiplied by a factor of 12. This would center the "Neu", or neutral membership function at 0 though, so an offset of 130 is added to keep all of the membership functions within the 0 to 255 range.

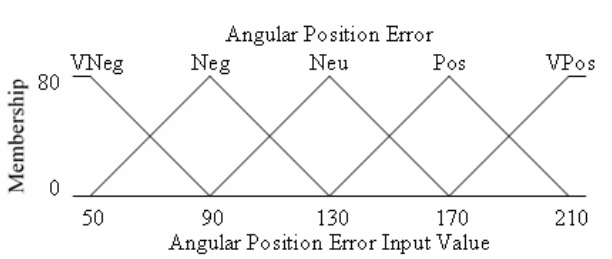

Figure 7. Fuzzy logic controller input membership functions for angular position error.

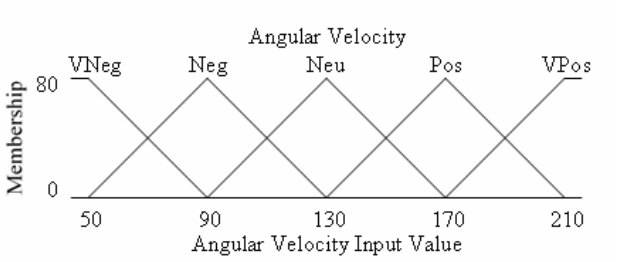

Figure 8. Fuzzy logic controller input membership functions for angular velocity.

Adjusting $\dot{\theta}$ was not as straightforward. Initial membership function locations were determined by simulation and then later tuned by experimentation. The end results is that $\dot{\theta}$ is scaled and offset such that when the robot is acted on by an external force it tends to remain at the same angle θ as it was before the disturbance.

Each of the inputs has five membership functions: VNeg, Neg, Neu, Pos, and VPos. The "Neg" represents negative values, and the "Pos" represents positive values. If a "V" precedes "Pos" or "Neg" it means that it is "Very" positive of "Very" negative. "Neu" simply means it a neutral value or near zero. The maximum degree of membership of each of these functions is 80 and the minimum is 0.

Much of the configurability and power of a fuzzy logic controller is in the evaluation function or the rule inference engine. Figure 9 shows the fuzzy rules matrix for this fuzzy controller. Each column shows the membership functions of angular position error and each row shows the membership functions of angular velocity. If the robot is experiencing a "VNeg" angular position error and a "VPos" angular velocity, this is represented by the lower left square of the matrix, and the corresponding output for the fuzzy controller is "Pos". This represents the evaluation of one fuzzy rule. Typically, each input will have a nonzero degree of membership simultaneously in more than one membership function. Thus, multiple rules must be evaluated during each cycle in the fuzzy controller.

Mamdani rule inference is used in the fuzzy controller. That is, when evaluating a rule where two input membership functions intersect in the rule inference matrix, the minimum degree of membership of the two input membership functions is assigned as the degree of membership to the corresponding output membership function. For example, if the angular velocity input has a degree of membership of 25 "VPos" and the angular position error has a degree of membership of 55 "Neg" then the rule evaluation will result in the output membership function "Pos" being assigned a degree of membership of 25.

Angular Position Error (theta)

Angular Velocity (theta_dot)	VNeg	Neg	Neu	Pos	VPos
VNeg	VNeg	VNeg	VNeg	Neg	Neu
Neg	VNeg	VNeg	Neg	Neu	Pos
Neu	VNeg	Neg	Neu	Pos	VPos
Pos	Neg	Neu	Pos	VPos	VPos
VPos	Neu	Pos	VPos	VPos	VPos

Figure 9. Fuzzy logic controller rule inference matrix.

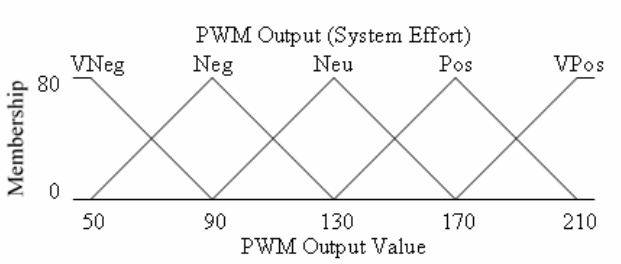

Figure 10. Fuzzy logic controller output membership functions.

The last stage of fuzzy control is defuzzification in which all of the output membership function degrees of membership must be combined into a single output control value. In this case the output is a PWM control signal as seen in Figure 10. Here the output membership functions have been centered on 130 in order to keep them within the fuzzy logic controller's data range of 0 to 255. The offset of 130 is subtracted from the fuzzy controller's output before being given to the actual motor controller circuitry.

The defuzzification technique used in this fuzzy controller is a variation of the weighted average method. After all Mamdani rule inferences have been evaluated the maximum degree of membership of each of the output membership functions is kept and the others are discarded. The defuzzifier output is then computed according to

$$Q_{HA} = \frac{\sum_i [\max(\mu(z_i)) * C_{z_i}]}{\sum_i \max(\mu(z_i))} \tag{19}$$

where

$i \equiv$ the number of output membership functions
$\mu(z_i) \equiv$ degree of membership in z_ith membership function
$C_{zi} \equiv$ center of z_ith membership function.

SIMULATION RESULTS

The ability of the fuzzy logic controller to balance the inverted pendulum robot was first simulated based on the system equations for the robot (Equations 5 and 9). To keep things simple the robot's wheels and motors were lumped together and modeled a single wheel and a single motor. The actual masses of the wheel/motor combinations and the body were measured and their respective inertias computed. The entire system was then modeled in Simulink. In the simulation shown in Figure 11 the desired position of the robot is straight up at 0 degrees for the first one second. Then from 1.0 to 1.5 seconds the robot should maintain an angle of +5 degrees followed by an angle of -5 degrees from 1.5 seconds to 2.0 seconds. Without friction, a negative angle is needed to keep the velocity of the robot low,

otherwise the required speed of the robot will exceed the motor's maximum velocity and the pendulum will fall over. Finally, the robot should return to a straight up position after the 2.0 second mark. The time to peak response is about 0.1 seconds for a 5 degree movement and about 0.25 seconds for a 10 degree movement.

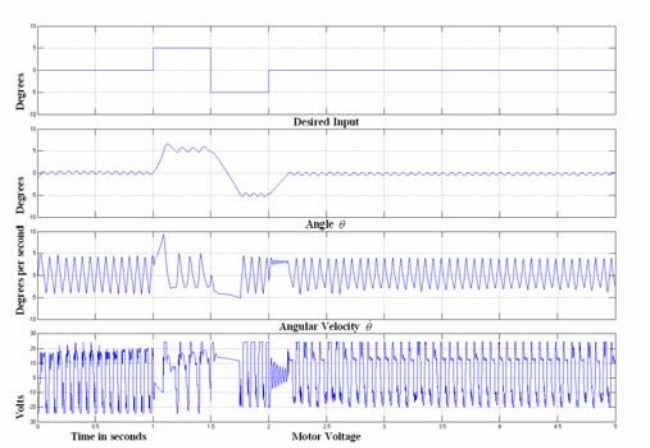

Figure 11. Simulation of the inverted pendulum and the fuzzy logic controller.

HARDWARE RESULTS

An actual inverted pendulum robot has been built utilizing an FPGA board to implement the fuzzy logic controller, complementary filter, as well as much of the signal pre- and post-processing (Figure 12). A second system board contains the motor driver circuitry and a PIC microcontroller to handle some auxiliary functions. The battery pack is mounted between the two boards. After initialization in an upright position the robot is able to successfully balance and maintain its equilibrium even if disturbed by an outside force as long as the disturbance is within limits. As previously mentioned, as long as the robot's angle is within $-10 < \theta < +10$ degrees it is able to successfully control itself and keep from falling over.

CONCLUSION

An inverted pendulum robot capable of balancing itself and maintaining equilibrium has been designed and implemented in hardware using an FPGA-based fuzzy logic controller. Integer inputs, outputs, and degrees of membership are used in the fuzzy logic controller to reduce computational complexity and the amount of FPGA resources needed. A two-axis accelerometer and a rate gyroscope used in combination with a complementary filter make an effective and simple solution for computing the actual angle of the robot. The robot's angular error and angular velocity are used as the inputs to the fuzzy controller. As long as the robot is not severely disturbed it is able to successfully maintain control over a window of $-10 < \theta < +10$ degrees.

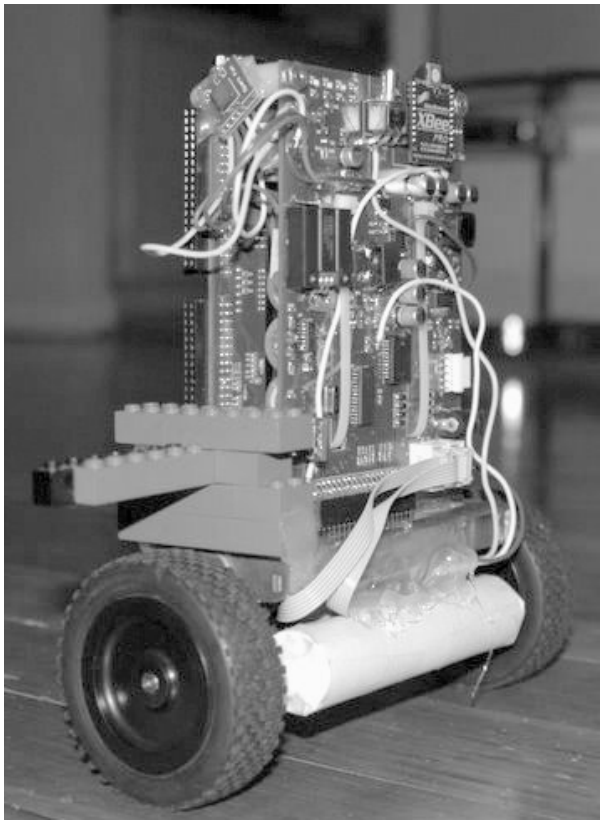

Figure 12. Photo of actual inverted pendulum robot.

REFERENCES

[1] J. Lam, "Control of an Inverted Pendulum", University of California, Santa Barbara, 10 June 2004, http://www.ece.ucsb.edu/~roy/student_projects/Johnny_Lam_report_238.pdf.

[2] R. Ooi, "Balancing a Two-Wheeled Autonomous Robot", University of Western Australia, 3 Nov. 2003, http://www.cs.cmu.edu/~mmcnaugh/kdc/as7/2003-Balance-Ooi.pdf.

[3] R. Andraka, "A Survery of CORDIC Algorithms for FPGA Based Computers", International Symposium on Field Programmable Gate Arrays, 1998.

[4] J. Yen, and R. Langari, **Fuzzy Logic: Intelligence, Control, and Information**, Upper Saddle River, NJ: Prentice Hallm Inc., 1999.

[5] B. Hemmelman, "High Performance Real-Time Fuzzy Logic System Using Field Programmable Gate Arrays and VHDL", Artificial Neural Networks In Engineering (ANNIE) Conference, St. Louis, MO, 2004.

[6] L. Terum, and B. Hemmelman, "A One GigaFLIPS Fuzzy Logic Control Chip Using Only Combinational Logic and Field Programmable Gate Arryas, IEEE Region 5 Conference, San Antonio, TX, 2006.