Proceedings of the ASME 2013 International Mechanical Engineering Congress and Exposition
IMECE2013
November 15-21, 2013, San Diego, California, USA

IMECE2013-62245

NOVEL, INEXPENSIVE ROBOT FOR TEACHING CONTROLS ENGINEERING

Andrew B. Wright
University of Arkansas at Little Rock
Little Rock, AR USA

Ann M. Wright
Hendrix College
Conway, AR USA

ABSTRACT

A novel Control and Sensor System (CASSY) has been developed to teach controls engineering to electrical and mechanical students. The inexpensive platform, which can be built for under $1500, has a first order velocity loop and a first order yaw rate loop with friction. A detailed model of the robot allows students to perform system identification and compare with the model. Students can implement PID, digital filter, and state space controllers on the robot, vary constants, measure performance, identify stability, and perform step and sine based system identification on the open and closed loop system. Wireless telemetry between the robot and a host computer allow all the control signals to be saved for later analysis. Fabrication guides and training videos are located on robotics.ualr.edu, and the robot has been fabricated by students at UALR and Hendrix College, demonstrating the ease with which the platform can be integrated into a curriculum. The CASSY platform has been used in both undergraduate and graduate control courses at the University of Arkansas at Little Rock. The practical robot experiments have improved learning outcomes of the largely theoretical material.

INTRODUCTION

Dr. Ann Wright is a professor in the physics department of Hendrix College, a private liberal arts undergraduate college. She first introduced robotics into the curriculum in a Lego Mindstorms-based robotics course for non- science majors. Dr. Andrew Wright is a mechanical engineer at the University of Arkansas at Little Rock (UALR) in the department of Systems Engineering. Together, they mentored a FIRST robotics competition team during 2000-2004. Dr. Andrew Wright incorporated robotic design into a UALR course called FIRST in Engineering: Elements of Mechanical Design [1]. While the Lego Mindstorm kits and the FIRST robots certainly allowed a nice introduction to robotics, the platforms had limitations in both the classroom and in the research lab. As a result, a new robot platform was created and successfully implemented at both schools. The goal of designing the Control and Sensor System (CASSY) was to create an inexpensive, but fully featured dynamic system (see Figure 1). Such a system would be useful for education and could support research.

Fabrication guides similar to Lego Constructapedia (e.g., http://education.lego.com/en-us/preschool-and-school/secondary/11plus-machines-and-mechanisms/constructopedia/) were prepared to allow collaborators to build a CASSY. These fabrication guides are available at http://robotics.ualr.edu.

In summer and fall 2010, independent study exercises with engineering students at UALR and physics students at Hendrix

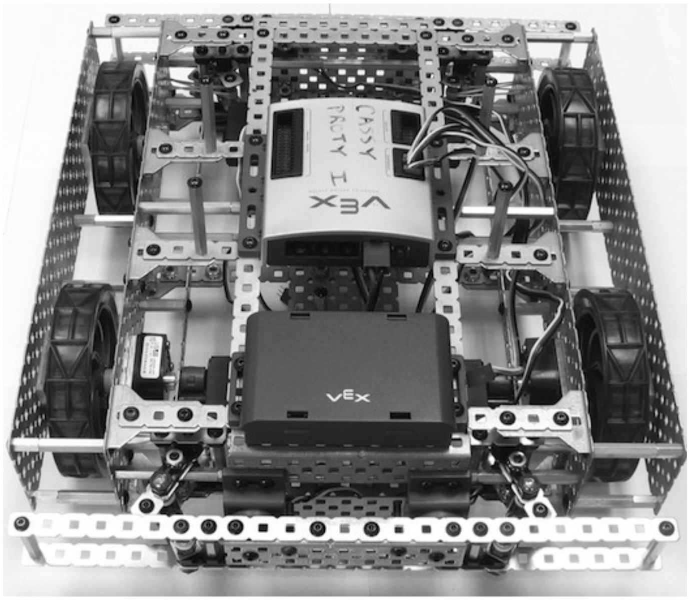
Figure 1: Control and Sensor System (CASSY) Robot

were offered. These exercises proved more valuable than expected, as the student response to the fabrication exercises developed surprisingly useful skills in the students. The fabrication exercise developed a detailed understanding of the robot in a way that reading manuals could not. The students were inspired to get started because the initial exercises were not intimidating. By the end of a semester, students had built basic skills that can be used to enhance research activities. The by-product for the instructor was a fully functioning mobile robotic platform which can be used to carry on educational and research activities.

The students reported that this exercise gave them a feeling of being a real robot engineer. It augments an engineer’s training (especially if the student has been theoretically trained) and helps physics students prepare for engineering graduate school. These activities are also very useful in the recruiting and retention of students.

The platform has enough expandability that it offers flexibility to explore many different types of problems that lead the student to conducting introductory research.

Details of the benefits to students of fabricating a CASSY platform are summarized in [2]. In the short time since the mechanical design was completed, UALR has built three CASSY's, Proty I, Jason, and Benny with plans to build three more. Hendrix has built two CASSYs.

The CASSY project was started originally to provide a project-based illustration of the largely theoretical concepts introduced in undergraduate control theory classes, using a digital control textbook such as Franklin and Powell [3]. The modern undergraduate student has a regrettable disinterest in theory for the sake of theory and needs some “to the point” examples to help them get into the material. The CASSY robot provides practical problems in system identification and linear control that allow the theoretical material to be paralleled with experimental assignments.

With the rise of Massive Open On-line Courses (MOOC) [4], the power-point slide presentation, matlab laboratory, traditional homework assignment paradigm for teaching control theory is not adequate. Some value added for in-class experience is essential. The kind of laboratory described in this paper not only improves the learning of the theoretical material, but also cannot be easily duplicated in a MOOC environment.

CONTROL AND SENSOR SYSTEM (CASSY)

The basic mobility platform (see Figure 1) is made from Vex mechanical parts (http://vexrobotics.com) and the Vex PIC18F8520 microcontroller, running a 55 Hz real time control loop. The finished dimensions are 33 cm (13 in.) wide by 41 cm (16 in.) long by 10 cm (4 in.) tall. CASSY has four independently driven wheels, the front two of which are instrumented with HEDS 100 counts per revolution quadrature encoders (http://www.usdigital.com). A custom-designed timing measurement algorithm combines the two signals to give a velocity signal in the direction of travel. Currently, the cost of all materials to build a complete CASSY is approximately $1500, which is far less than most commercially available robots or robot kits. In fact, it is inexpensive enough that most schools could afford to build more than one, which would open up programming opportunities to include cooperative behaviors or allow for student competitions or use in student lab classes.

A novel six-bar bumper design allows the right and left bumper switches to be independently toggled, when the bumper contacts a rigid obstacle on either the right or left of the CASSY. The bumper sensor outputs can give an indication as to the location of contact which can inform an autonomous algorithm to resolve the collision.

A custom circuit board [2], mounted on a Radio Shack printed circuit board, uses a layout that is identical to standard solder-less bread-board, which allows bread-boarded prototype circuits to be directly transferred to soldered connections. The circuit board contains an Analog Devices ADXRS613 inertial yaw rate sensor (http://www.analog.com) aligned with the turning axis. A Roving Networks-based Bluetooth serial port on the custom circuit board gives wireless telemetry to the host computer with a Bluetooth connection (http://www.sparkfun.com).

The chassis has room for a second custom circuit board and there is resource expansion for at least one additional mechanical module or sensor module containing perhaps one additional motor and encoder.

The encoder interface requires custom parts to be machined to adapt between the Vex component constraints and the encoder constraints [2]. It was discovered that the Vex Delrin bearings provided too much nonlinearity and a custom roller-bearing design was developed to occupy the same foot-print as the Delrin bearings. These parts have recently been printed on a 3D printer, which further reduces the expense for building the system.

The robot can be programmed using Microchip’s proprietary MPLAB integrated development environment; however, that can be superseded by the open-source Small Device Compiler (sdcc). A custom linux-based GUI program, ABWComm4, was designed to communicate with CASSY [2]. This program includes a downloader to load user code onto CASSY. This program can stream up to ten channels of data in real-time from the CASSY over the Bluetooth connection. This program is being ported over to Mac OSX to make it more accessible and simply start-up.

SYSTEM MODEL

A state space model of the CASSY robot has been developed from first principles [5]. The robot's POSE in inertial coordinates is (X,Y,θ) (see Figure 2). The local coordinates, (x,y) are related to the inertial coordinates through the coordinate transformation matrix, $\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix}$. The velocity in the direction of travel (x-direction) is $V_x = \dot{X}\cos\theta + \dot{Y}\sin\theta$ and the acceleration in the direction of travel is $A_x = \ddot{X}\cos\theta + \ddot{Y}\sin\theta$. The velocity perpendicular to the direction of travel (y-direction) is $V_y = -\dot{X}\sin\theta + \dot{Y}\cos\theta$ and the acceleration in that direction is $A_y = -\ddot{X}\sin\theta + \ddot{Y}\cos\theta$. The wheel parameters are shown in Figure 3.

With physical constants $a = 0.137\ m$ (half the width of the robot), $b = 0.100\ m$ (the location of the rear wheels relative to the center of mass), $d = 0.078\ m$ (the location of the front wheels relative to the center of mass), $r = 0.0508\ m$ (the wheel radius), $m_W = 0.09\ \text{kg}$ (the wheel mass), $I_{Wy} = 1.33\,x\,10^{-4}\ \text{kg}\cdot m^2$ (the wheel's moment of inertia about its rotation axis (the y-axis)) and $I_{Wx} = 0.75\,x\,10^{-4}\ \text{kg}\cdot m^2$ (the wheel's moments of inertia perpendicular to its rotation axis (due to symmetry, both the x and z axes)), $K_e = 0.589\ V\cdot s$ (the motor's electric constant), $m_{RB} = 3.29\ \text{kg}$ (the robot body's mass without wheels), and $I_{RBz} = 7.57\,x\,10^{-2}\ \text{kg}\cdot m^2$ (the moment of inertia of the robot body about the z-axis), the equations of motion of the four driven wheel, slip-steer mobile robot are:

$$\begin{bmatrix} m_T + m_J & 0 & 0 \\ 0 & \mu & 0 \\ 0 & \mu & I_C + m_J a^2 \end{bmatrix} \begin{bmatrix} A_x \\ A_y \\ \ddot{\theta} \end{bmatrix} = \frac{2\mathrm{K}_m^2}{rK_e} \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & a \end{bmatrix} \begin{bmatrix} V_{Rm} + V_{Lm} \\ V_{Rm} - V_{Lm} \end{bmatrix} - \frac{4\mathrm{K}_m^2}{r^2} \begin{bmatrix} V_x \\ 0 \\ a^2\dot{\theta} \end{bmatrix} + \begin{bmatrix} (\mu\dot{\theta} - m_J V_y)\dot{\theta} \\ \mu V_x \dot{\theta} \\ 0 \end{bmatrix}, \tag{2}$$

where V_{Rm} and V_{Lm} are the voltages applied to the right and left motors, $m_T = m_{RB} + 4\mathrm{m}_W$ is the total mass of the robot body and the wheels, $m_J = \left(\frac{4\mathrm{I}_{Wy}}{r^2}\right)$, $\mu = 2\mathrm{m}_W(d - b)$ and $I_C = I_{RBz} + 4\mathrm{I}_{Wx} + 2\mathrm{m}_W(2\mathrm{a}^2 + b^2 + d^2)$. The motor constant, $K_m = \sqrt{0.062\ N\cdot m\cdot s} = \frac{K_t}{\sqrt{R}}$, is the slope of the motor's torque-speed curve and can be written in terms of the torque constant, K_t and the resistance of the motor's coils, R [6].

For a robot only traveling in a straight line, the velocity can be determined by setting $\theta = \dot{\theta} = \ddot{\theta} = 0$ in equation (1), which results in a first order system in velocity, V_{Rm}

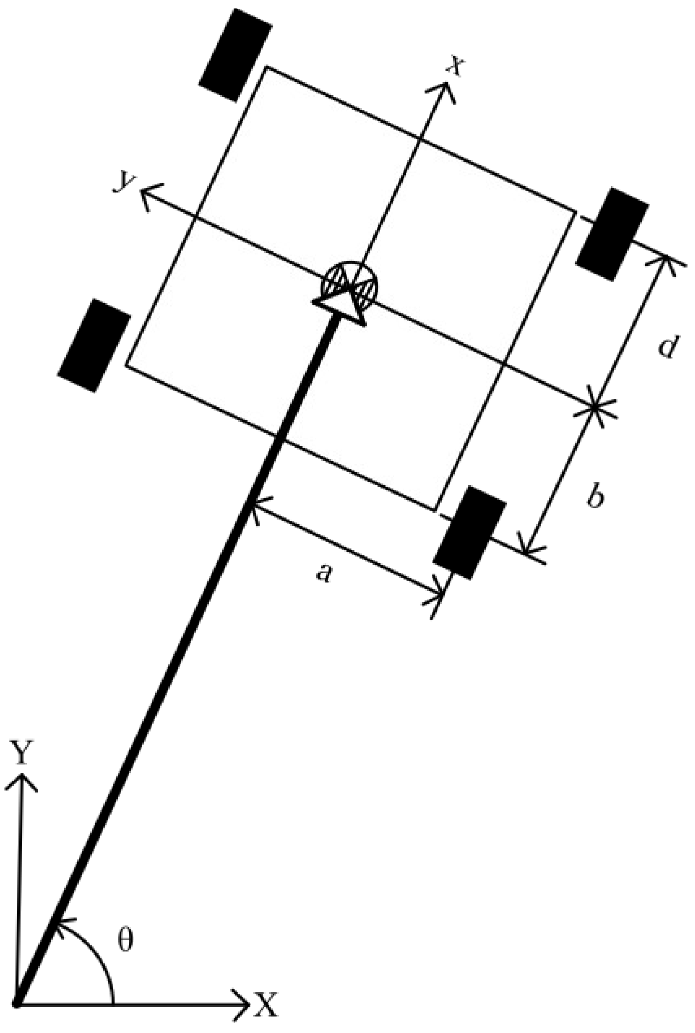

Figure 2: Top View of CASSY

VELOCITY MEASUREMENT SYSTEM

Each of CASSY's front wheels is instrumented with an 100 Count per Revolution (CPR) quadrature encoder (US Digital part number E2-100-375-H). At maximum velocity, the encoder generates a waveform such as seen in Figure 4. Two physical quantities can be measured directly from this waveform, angular speed and change in angular position.

Determining the direction in which the encoder is turning requires a few logic gates. Inspection of the waveform shows

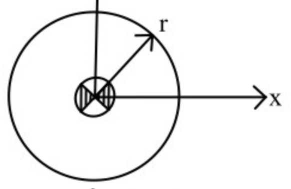

Figure 3: Side View of Wheel

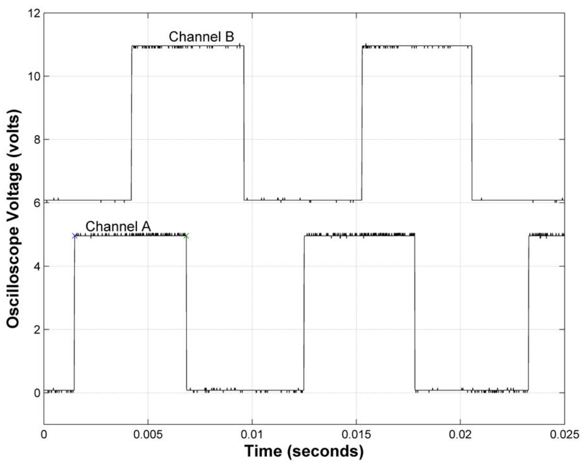

Figure 4: Encoder at Maximum Velocity

that Channel A is leading Channel B when rotation is forward. One half cycle after Channel A transitions from low to high, Channel B transitions from low to high. One half cycle after Channel A transitions form high to low, Channel B transitions from high to low. If the encoder were running in reverse, the pattern would be reversed (i.e. Channel B would lead Channel A.)

The state of the encoder direction can be uniquely determined when one or the other channel transitions. If the value of both channels were read at the instant when Channel A transitions, forward is indicated by Channel A = 1, Channel B = 0 or by Channel A = 0, Channel B = 1. Reverse is indicated by the opposite pattern (Channel A = 1, Channel B = 1 or Channel A = 0, Channel B = 0). If forward is represented as true and reverse as false, then the direction, for a Channel A transition, is given by

$$\text{direction(Channel A transition)} = \text{chA XOR chB.} \qquad (3)$$

As long as enough transitions occur in the sampling interval, then the resolution is set by the free running timer's resolution, rather than by the number of counts per revolution or the interrupt handling architecture.

The angular speed is the ratio of the net angular distance traveled to the amount of time required to travel the distance. If a timer were started upon a low to high transition of Channel A and stopped upon the next low to high transition on Channel A, then the angular distance traveled is known $\left(\frac{2\pi\ \text{radians}}{CPR}\right)$ and the angular speed would be

$$\omega(k) = \left[\frac{\text{RATE}}{(\text{tick\_count}(\text{k-1}))}\right](-1)^{\text{direction}} , \qquad (4)$$

where tick_count is the number of timer ticks which have occurred over the previous time interval (k-1) and $\text{RATE} = \left[\frac{2\pi\ \text{radians}}{(\text{CPR})(\text{RES})}\right]$. The variable, RES, is the "timer resolution" which is the amount of time for one tick on the counter. It can be seen that the timer resolution directly affects the speed resolution in the same fashion as the number of counts per revolution (i.e. both occur in the denominator).

Table 1: Variables Used in Velocity Algorithm

Variable	Meaning
tick_count	The number of free-running timer ticks which have occurred since the last velocity measurement was processed. This value is only updated upon a transition, and it does not include the number of ticks which may have occurred after the last transition.
Nticks	The net number of transitions which have occurred since the last real time interrupt occurred. Forward transitions increment N_ticks. Reverse transitions decrement N_ticks.
memCount	The number of free-running timer ticks which have occurred after the last encoder transition.
TMRvalue	The last value stored by the capture module.
last_dir	The direction of travel on the last velocity measurement. This is only used if no encoder transitions have occurred during the sample time.

If this method were implemented directly, it would give an accurate measure of the instantaneous speed experienced by the encoder. Unfortunately, for a digital system, the speed measurement is assumed to be constant over the fixed sampling interval and, for control purposes, instantaneous speed, which reflects the many imperfections in the mechanical system, is not desirable. Some form of averaging or filtering must be employed. Rather than making many instantaneous measurements at intervals within the sampling interval and averaging, it is desirable to incorporate the averaging into the measurement.

If multiple ticks occur during a sampling interval, then the change in position which occurs during the sampling interval is $\left[\frac{2\pi\ \text{radians}}{\text{CPR}}\right]\sum_{i=1}^{\text{Nticks}}(-1)^{\text{direction}_i}$ and the average angular speed over this sampling interval is

$$\text{updateValue} = \text{MaxTimerValue} - \text{lastTMRvalue}$$

$$\text{memCount} = min(\text{memCount} + \text{update\_value}, 0\text{xFFFFFFFF})$$

$$\text{tick\_count} = min(\text{tick\_count} + \text{update\_value}, 0\text{xFFFFFFFF})$$

$$\text{lastTMRvalue} = 0$$

Code 1: Process Timer Interrupt

$$\omega(k) = \left[\frac{\text{RATE}}{(\text{tick\_count}(\text{k-1}))}\right] \sum_{i=1}^{\text{Nticks}} (-1)^{\text{direction}_i} . \qquad (5)$$

Equation (5) requires that tick_count and Nticks be measured. For CASSY's velocity measurement scheme, Channel A is connected to an interrupt which triggers on a low-to-high transition (process_capture_interrupt, see Code 1) and Channel B is connected to a digital input. A free-running timer, which generates an interrupt on overflow (process_timer_interrupt, see Code 2), is needed because of the practical limitation that an infinitely large timer register does not exist. A real time timer, which generates an interrupt at the sampling time (process_realtime_interrupt, see Code 3), are used. This architecture requires three separate Interrupt Handlers to be used and measures the parameters shown in Figure 4. Variables used in the interrupt routines are described in Table 1.

Process Capture Interrupt

When the Capture Interrupt Handler (see Code 1) is called, a transition on the encoder has occurred. The first action is to reset memCount to zero, so that this variable will always be counting up from the previous encoder transition. The number of counts since the last time the CAPTURE Interrupt Handler was called must be added to Tick_count. This is accomplished by subtracting previously stored value of the free-running timer (last_TMR_value) from the value stored by the CAPTURE circuitry (TMR_value). The value of TMR_value is then stored in last_TMR_value for the next iteration.

It is critical that tick_count not overflow. In this example, an unsigned long variable is used to store Tick_count. It is desired that $\text{tick\_count} + \text{update\_value} < 0\text{xFFFFFFFF}$.

The final action in the routine is to update Nticks. If the direction is determined to be forward, then Nticks is incremented. Otherwise, it is decremented. In this way, Nticks contains a sense of the net direction of travel. The only way in which both an increment and a decrement can occur (i.e. a change in direction) is if the velocity is very low or if the sampling time is absurdly long or if a sinusoidal velocity well outside the band-width of the system occurs.

```
$\text{memCount} = 0$
$\text{update\_value} = \text{TMR\_value} - \text{last\_TMR\_value}$
$\text{last\_TMR\_value} = \text{TMR\_value}$
$\text{tick\_count} = min(\text{tick\_count} + \text{update\_value}, 0\text{xffffffff})$
if ChA XOR Chb = 1
    $\text{Nticks} = \text{Nticks} + 1$
else
    $\text{Nticks} = \text{Nticks} - 1$
end
```

Code 2: Process Capture Interrupt

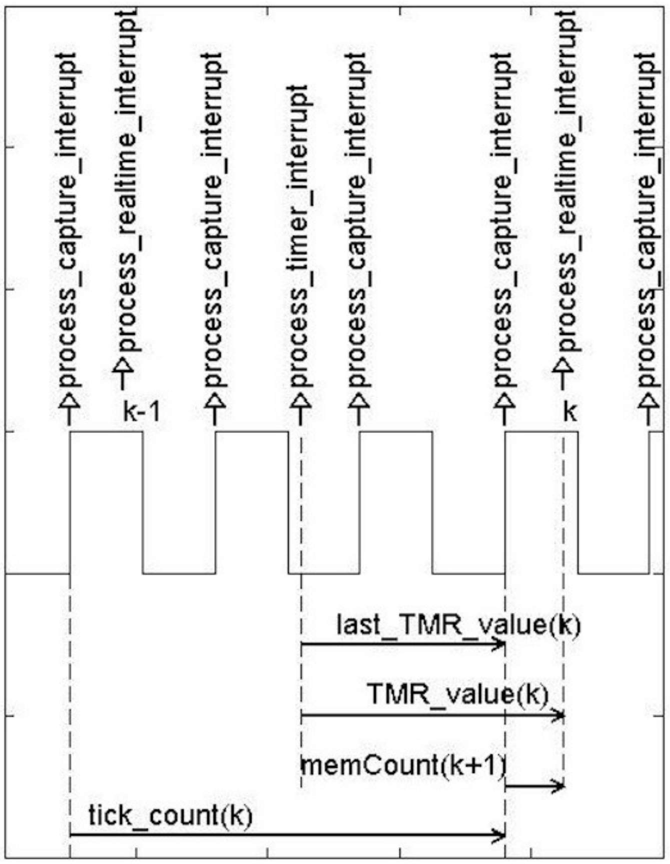

Figure 5: Encoder Velocity Timing Chart

In order for $\text{tick\_count} = 0\text{xFFFFFFFF}$, a very long time has passed between ticks in the encoder. In other words, the encoder is moving very slowly or stopped. In this design, the divisor in such a case will be a very large number and the output of the velocity measurement will be zero.

Process Timer Interrupt

The Interrupt Handler processes the free-running timer's overflow. If there were such a thing as a timer with an infinite register, then this routine would not be necessary. However, when the free-running timer reaches its maximum value, MAX_COUNTER, the routine shown in Code 2 will be called.

The interaction between "Process Timer Interrupt" and "Process Capture Interrupt" occurs through the variable last_TMR_value. The known value of the timer when this routine is called is MAX_COUNTER. The update value for both memCount and Tick_count will be the difference between the current timer value and the previous value when the timer was updated (last_TMR_value).

Both memCount and tick_count are updated with this update_value, with the protection that they cannot overflow.

```
If tick_count > 0
    If Nticks = 0
        $\omega = \frac{\text{RATE} * \text{lastdir}}{(\text{tick\_count} + \text{TMRvalue})}$
    else
        $\omega = \frac{\text{RATE} * \text{Nticks}}{\text{tick\_count}}$
        $\text{tick\_count} = \text{memCount}$
        if Nticks>0
            $\text{lastdir} = 1$
        else
            $\text{lastdir} = 0$
        end
    end
end
```

Code 3: Process Real Time Interrupt

The last_TMR_value is reset to zero (which is the new current timer value).

Process Real Time Interrupt

Since tick_count is primarily incremented when the Channel A transitions, it should never be zero during the real-time loop, unless no transitions are occurring (very low speed). Since division by zero would result in abnormal operation, this value must be tested. The only conclusion that can be drawn if it is zero is that velocity is zero. This test is run by the first decision in "Process Real Time Interrupt" shown in Code 3.

The next test determines whether any net encoder transitions have occurred during the sampling time (i.e. Nticks is non-zero). If Nticks is non-zero, at least one full pulse has occurred and the value in tick_count is an accurate representation of the number of free-running timer ticks which have occurred from the start to the end of the pulse. The design equation for measuring velocity can be used, $\omega = \frac{RATE * Nticks}{\text{tick\_count}}$. The value of tick_count is then reset for use in the next sampling time, by assigning it to memCount, which is the number of counts which have occurred *after* the last transition. The number of ticks is also reset (Nticks = 0). The value of last_dir is set based on the net value of Nticks.

If Nticks is zero, then velocity is slow relative to the sampling time, and an estimation of the velocity must be done, since the value in tick_count will be indeterminate. The variable, tick_count, will contain any residual time stored in memCount and will have been incremented when the free-running timer overflows. However, there may be counts on the free-running timer since the free-running timer overflowed. Therefore, the best guess at the number of ticks between transitions is to assume that the encoder is on the verge of transitioning and incrementing Nticks to one. Any time on the free-running timer (TMRvalue) and the current value of tick_count (which is non-zero based on the decision block) can be used. Therefore, $\omega = \frac{\text{RATE}}{(\text{tick\_count} + \text{TMR\_value})}$.

A test of the velocity measurement system was implemented on CASSY. Three open loop values (minimum speed, middle speed, maximum speed) for motor command were given and the velocity measured, stored, and printed (see Figure 6). This measurement system performs reasonably well for the experimental conditions. It is normal to see an oscillation, such as the one between 1.5 and 3.5 seconds. These oscillations are associated with "wheel run-out" and represent the encoder disk moving radially in and out relative to the measurement section due to imperfections in the bearings and the code-wheel. As the code-wheel moves radially in and out, the width of the opaque strips will appear to get longer and shorter, yielding changes in the pulse width that appear as a change in velocity.

BACKLASH COMPENSATION

The gearboxes and the connection of the drive axle to the wheel has a severe backlash that interfere with the sinusoidal system identification. This provides an opportunity to develop techniques in compensating for backlash.

The first approach involved designing a drive coupler that removed the backlash between wheel and axle. Although this greatly improves the system, it is cumbersome to build the coupler and it does not account for the backlash in the gearbox.

A technique which was developed in observatories (see [7] from 1966 for an example) and which has seen some use in robotic systems (see RSI robotics products for an example of this technique applied to driving a track in an inverted pendulum [8]) is the dual-opposed torque motor concept. This technique can be used in mobile robots, especially for skid-steer systems with all four wheels independently driven.

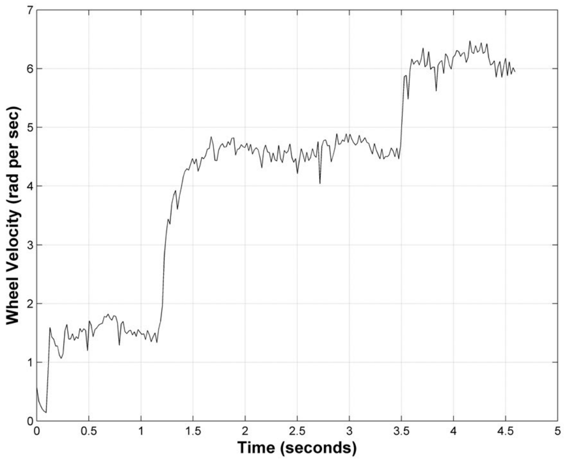

Figure 6: Velocity Measurement at From Minimum to Maximum

With two independently drive wheels that provide driving force along the same side of the vehicle, the output of each motor is additive. The net force between wheel and ground is all that matters, and the motors do not have to output the same torque. Hence, as torque approaches zero, one motor can output a positive torque and the other motor a negative torque. The sum of the two torques will be zero; however, the gear system of at least one motor will always be engaged. Once the positive motor has passed a certain minimum torque, the negative motor can reverse direction from a negative offset to a positive offset, while the positive motor continues to remain positive and its gear system fully engaged. As the negative motor switches to positive, it experiences a loss of (small) torque as it traverses through the backlash. For this period, the full load is carried by the positive motor.

The Vex motor also has a dead zone as signal goes through zero. With a slight modification to the dual-opposed command signal, dead zone can be removed as well.

PID CONTROL

A continuous proportional-integral-derivative (PID) controller

$$U(s)=\left(K_P+\frac{K_I}{s}+K_D s\right)E(s) \ , \qquad (6)$$

cascaded with a Zero Order Hold can be converted to a difference equation using the continuous to discrete transform [3]

$$\begin{aligned} u[k]=u[k-1]+\left(K_P+K_I+K_D\right)e[k]- \\ \left(K_P+2K_D\right)e[k-1]+K_D e[k-2] \end{aligned} \ , \qquad (7)$$

where K_P is the proportional gain, K_I is the integral gain, K_D is the derivative gain, $u[k]$ is the control signal, and $e[k]$ is the error signal.

Equation 7 is implemented in the CASSY embedded code and the GUI, ABWComm4, allows users to adjust the parameters of the equation. An example step response with $K_P=2.5$, $K_I=0.5$, and $K_D=0.5$ is shown in Figure 7. Students can vary parameters and demonstrate under-damped, over-damped, and critically-damped closed loop systems.

Each parameter can be varied until the system becomes unstable, allowing students to explore the stability boundary. When CASSY goes unstable, the system either goes to maximum speed or exhibits a bang-bang oscillation between zero speed and maximum speed. The design of the mechanical system is such that these instabilities cannot harm the system. This allows students to comfortably play around with gains in a low consequence environment.

SYSTEM IDENTIFICATION – STEP RESPONSE

The CASSY robot is programmed with an embedded code which allows a reference step input velocity to be commanded. A system identification algorithm computes the rise time, settling time, overshoot, time to peak, and steady state error. These values are displayed on the screen and can be recorded.

The rise time computes the time between $.1y_{ss}$ and $.9y_{ss}$ and the settling time computes the last time that the signal gets to $.99y_{ss}$, where y_{ss} is the steady state value of the measurement (in Figure 7, y_{ss} is the robot velocity in the direction of travel). Because of surface roughness, the settling time is not reliable, but this gives students a chance to see real data and draw conclusions.

An example of both an open loop step and a closed loop step of $0.2\frac{m}{s}$ are shown in Figure 7. The data clearly shows the time delay inherent in a digital system. The open loop trace shows the first order (over-damped) nature of the velocity loop. The closed loop system, with a PID controller, is under-damped and therefore second order (or higher). The closed loop system has an integrator, converting it from a Type 0 system to a Type 1 system, as seen by the zero steady-state error.

In these experiments, the surface on which the CASSY was driven was a grid of interlocking foam mats. Because the CASSY frame is rigid, some compliance needs to be introduced in the floor surface so that all four wheels contact the ground at all times. The seam at the intersection of the mats creates a bump, which can be seen in the data as a sudden decline in

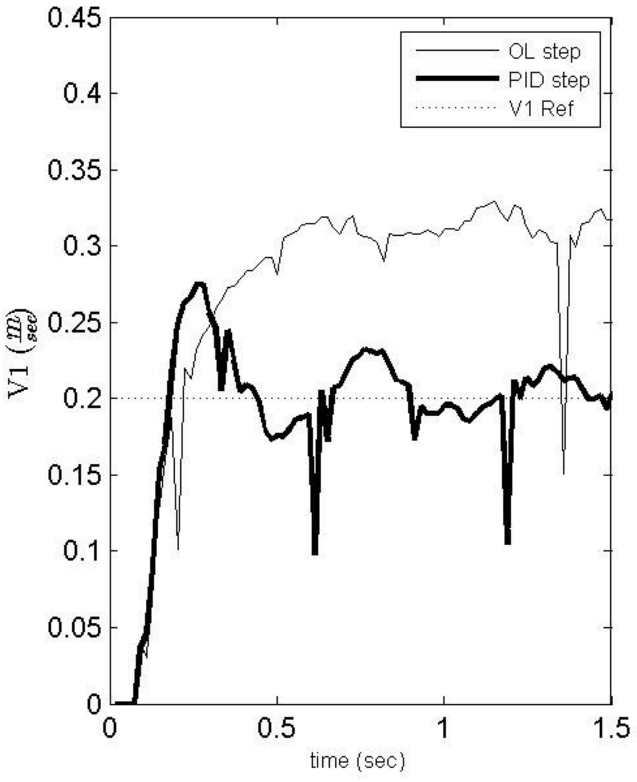

Figure 7: Open Loop and Closed Loop Step Responses

velocity. The closed loop system sees these bumps as impulses, and the convergence of the closed loop system to these impulses can be seen.

These experiments were used to illustrate the section of the undergraduate controls course covering performance specifications. Unlike purely theoretical or matlab-simulation-based illustrations of the material, students can see for themselves exactly what these specifications mean when a real system changes its behavior under feedback control. The physical system gives visceral appreciation that cannot be taught through paper-and-pencil exercises or simulations.

The GUI, ABWComm4, allows students to gather the step response data in about 10-20 minutes of laboratory time. By automating the calculations, students can focus on interpretation.

A matlab script file was provided that allows the raw data to be imported into matlab for off-line data analysis and presentation.

SYSTEM IDENTIFICATION – BODE PLOT

The CASSY robot is programmed with an embedded program that allows a sinusoidal reference signal to be commanded (see Figure 8). A system identification algorithm determines the frequency, gain, and phase between the reference signal and the measurement.

The frequency of the sinusoid can be varied and the data accumulated into a Bode plot (see Figure 9). The data presented in Figure 9 is for the CASSY named Jason. Although there are minor differences between the three robots (Proty I has anti-backlash hubs, Benny has a line-tracker on its underside), the data for these robots are very similar to Jason's.

Due to the automation in the GUI, ABWComm4, it requires about 30 minutes to complete one experimental Bode plot. Requiring students to plot and analyze their own data (versus using Matlab's Bode plot command for simulations) students are led through most of the minute details associated with the Bode plot.

Closed loop Bode Plots can be generated, demonstrating changes in steady-state gain and placing of the closed loop pole. Phase and gain margins can be computed, especially for systems close to the stability margin.

FUTURE WORK

A first order state space controller with an added integral controller has been implemented. This controller is used to support the graduate control class. These results will be presented in a future paper.

An inverted pendulum that mounts to the top of CASSY with potentiometer feedback has been designed. This converts the velocity loop to a third-order loop with an unstable pole. CASSY plus the inverted pendulum allows study of the classic “inverted pendulum on motorized cart” problem. Robust

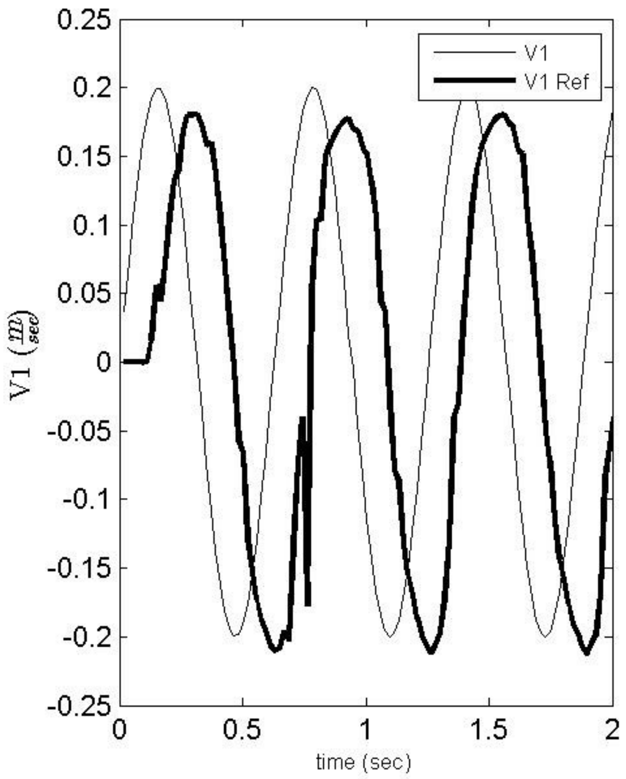

Figure 8: Sinusoidal Input Signal and Response

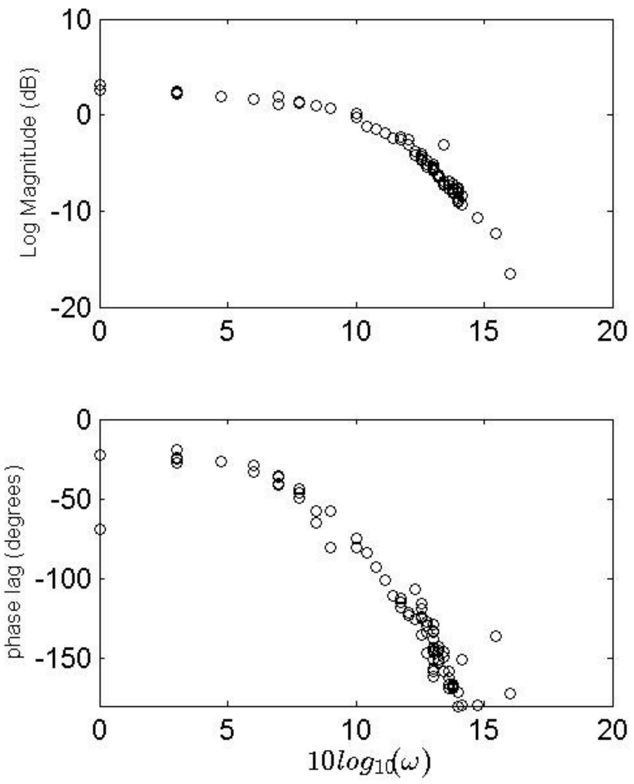

Figure 9: Bode Plot of the CASSY, Jason

control computations are required to fully implement this module, making this a very good exercise for a graduate class. This problem is used in conjunction with the graduate class and those results will be presented in a future paper.

The current CASSY is designed around the PIC-based Vex controller. A few years ago, a more advanced, Cortex-based, Vex controller was introduced. This project is being brought to completion using the original processor, since many universities have purchased an extensive supply of this platform, and this allows these institutions to easily re-purpose those assets to teaching control theory. However, conversion of the embedded system to the Cortex processor is underway.

Although the yaw loop allows friction to be explored, the skid-steer system demonstrates why such a drive system is undesirable in mobile robots. The drive system is being redesigned to a center drive with steerable front and rear wheels so as to provide zero-turn radius, frictionless turning.

The backlash compensation algorithm has been implemented and demonstrated; however, some work is required with the torque curve to get optimum performance. This improved performance is needed with the inverted pendulum implementation.

Both PID and state space controllers have been implemented on both velocity and yaw rate loops. The PID needs to be extended to include a general discrete pole-zero controller.

REFERENCES

[1] Wright, A. B., Wright, A. M., "FIRST in Engineering: Elements of Mechanical Design," 2003 ASEE Annual Conference.

[2] Wright, A. M., Wright, A. B., “An Inexpensive Dynamic System for Teaching Measurement and Controls, 51st AIAA Aerospace Sciences Meeting, AIAA 2013-0981.

[3] Franklin, G. F., Powell, D. J., Workman, M. L., Digital Control of Dynamic Systems, 3rd Edition, Addison-Wesley, 1998.

[4] Brown, A., “MOOCs Make Their Move,” The Bent, v. 104, n. 2, pp. 13-17, 2013.

[5] Wright, A. B., Introduction to Controls: Digital Design for Mobile Robotic Applications, Appendix A, Draft, http://calliope.ualr.edu/page, 2013.

[6] Globe Motors technical note, E-10, DC Motors, www.globe-motors.com/dc_motor.pdf

[7] Fisher, R. C., Hoard, H. S., Onians, F. A., Radio telescope for millimeter wavelengths, J. Spacecraft, v. 3, n. 1, 1966.

[8] RSI Robotic Systems Integration, www.roboticsys.com