Embedded Control Laboratory 3.3

Proportional Control

SIMULATED LABORATORY

→ Readings

- Lab Manual: *Chapter 7 - Control Algorithms* Up to and including Proportional Control.

→ Laboratory Goals

- Provide an introduction to open and closed loop control.
- Integration of sensors and actuators to perform automated functions.

→ Motivation

This laboratory serves as preparation for driving the full "Smart Car" in Labs 4 and 5 as well as the Gondola in Lab 6. In lab 3.1, the basic functionality for operating the steering servo and drive motor were built using the *Programmable Counter Array* (PCA0) to produce a *Pulse Width Modulated* (PWM) control signal. This control signal allows for the generation of a digital signal that conveys more than just True and False: by varying (modulating) the pulse width of the signal, the effective average value of the PWM may be changed in an analog fashion. In lab 3-2, development of the communication routines required for measuring the car's behavior was completed. In this [sub]Lab, the pieces made in lab 3-1 and lab 3-2 are put together such the the sensors control the motors through a proportional control routine.

→ General Lab Description

In this lab, the control routines needed to incorporate the sensor readings and the motor control outputs into a functional unit to form a building block of the "Smart Car" control algorithm will be done. Of course, this consists of two independent components:

1. The speed controller PWM routine developed in Lab 3-1 must be combined with the ultrasonic ranger sensing routine from Lab 3-2 to cause the speed controller, and therefore the drive motor, to essentially be controlled by the ranger such that low values measured by the ranger will cause the drive motor to spin "forward" at full speed, high values measured by the ranger will cause the drive motor to spin "backwards" at full speed, and linearly vary in between. The specific limiting values required to complete this are listed in the specifications table.
2. The servo motor PWM routine developed in Lab 3-1 must be combined with the electronic compass sensing routine from Lab 3-2 to cause the steering wheels to turn left and right appropriately. A "desired heading" is specified such that the wheels will always attempt to point in that direction - that is, if the car is aligned with the desired heading, the wheels should be straight forward, if the

car is turned left, the wheels should turn right, etc. It must be understood that a compass heading is a "circular" measurement; that is, when rotated past 360° (3600), the angle reverts to 0° and begins going up again. Please refer to the lecture for a comprehensive discussion of how to deal with this.

The crossbar must be configured such that UART0, SMB0, and CEX0-CEX2 are enabled. The servo motor is connected to P0.4 and the drive motor is connected to P0.5. The CEX*n* connections must be determined.

Finally, three slide switches are provided in the simulation, attached to P3.5, P3.6, and P3.7. P3.6 should be used to enable/disable the control of the drive motor and P3.7 should be used to enable/disable the control of the steering servo. *When either of these devices are disabled, the pulse widths should be set to neutral or center, respectively.*

Specifications

Specification	Value	Unit
Drive Motor and Ranger Mapping		
Full Speed Forward	100	cm
Neutral	200	cm
Full Speed Reverse	300	cm
Steering Servo and Compass *Error* Mapping		
Where $error = desired\_heading - actual\_heading$		
Turned Completely Right	-750	°/10
Straight Forward	0	°/10
Turned Completely Left	750	°/10

→ Laboratory Success Conditions

- Drive motor is fully controlled by the ranger reading with appropriate limits and scaling
- Steering servo is fully controled by the compass reading with appropriate limits and scaling
- Slide switch functionality is implemented correctly
- Steering servo capable of full range of left/right without failure
- Pulsewidths have appropriate limits placed upon them to avoid values outside of acceptable range
- The submitted source code is documented, indented, and modular[1].

[1]Modular code: [Simplified] Implementation of systm is broken into functions that perform specific tasks; that is, all code cannot exist in the `main()` function.

→ Pseudocode

The following pseudocode should result from combining Lab 3-1 and Lab 3-2. The only completely new lines that exist in this pseudocode are marked with an *. Note that keyboard control of the actuators has been removed.

```
    Initializations
    ...
    Trigger a ranger reading to ensure first downloaded measurement is not erroneous
    Initialize motor to neutral for 1 second
    ...
    Infinite Loop
    ...
    if 40 ms have passed (new_heading flag)
        clear the new_heading flag
        Read appropriate registers from the compass to get ...
                ...heading in tenths of degrees
        Combine data bytes and store into variable "heading"
        *Calculate the heading error and correct it (remember: circular)
        *Apply control equation to calculate desired Servo PW
        Set limits on calculated PW
        if slide switch is off, set PW to CENTER
        Assign PW to PCA
    if 80 ms have passed (new_range flag)
        clear the new_range flag
        Read appropriate registers from the ranger to get ...
                ...to get the first echo
        Combine data bytes and store into variable "range"
        Write 0x51 to the ranger control register to start ...
        ...another ultrasonic measurement, or "ping"
        *Calculate the range error
        *Apply control equation to calculate desired Motor PW
        Set limits on calculated PW
        if slide switch is off, set PW to NEUTRAL
        Assign PW to PCA
    if 100 ms have passed (new_print flag)
        clear the new_print flag
        Print out the current heading, range, and PW values
...
End Loop
```

Hints

1. If Lab 3-1 and Lab 3-2 were done in a modular fashion, assembly of the Lab 3-3 code should be fairly straight forward.
2. No template code is provided (again, merge 3-1 and 3-2 codes)
3. If the steering servo is turning away from the desired heading but scaling appropriately, there is an incorrect sign in the control equation.
4. Calculation of the proportional control equation should only happen when a new measurement is available. Running the equation on old, or "stale," measurements is unnecessary and inefficient and in some cases may lead to errors.
5. Ensure that all the CCMs and associated Port Pins are initialized correctly.
6. A common problem seen when calculating the PWM is that the value is calculated properly but never applied to the PCA0 CCM! Remember to apply a pulse width, `Npw`, *especially during the motor initialization.* The following line of code may be used:

   ```
   PCA0CPn = 0xFFFF - Npw;  \\ n is the CCM/CEX module used.
   ```

7. Neither the drive motor or steering servo will function without the correct PWM period. Ensure that the start value as applied in the `PCA_ISR()` function is implemented properly.
8. As always, don't forget `Sim_Update()`!