Lab 5 - Controls and Feedback

ECE 4330-5330 - Fall 2018

There are many examples of feedback in everyday life, such an example is a speeding ticket. Consider the ticket as an “error term” where a large ticket means that you exceeded the limit by a greater amount compared to a small ticket. Hopefully, a ticket motivates, you to drive slower and a larger ticket convinces you to drive slower, faster.

The use of feedback to control a system functions on similar principles. The device built in this lab is a Proportional Controller. The exact reason for its name will be evident by lab’s end. The idea is to develop a controller that uses feedback from a sensor to drive a closed-loop system that dynamically adjusts operations to compensate for external disturbances or interferences.

A. Background:

There are two basic types of control systems: open loop and closed loop. Both types of control systems use reference, r(t), and control, u(t), signals. Open loop systems do not use feedback, while closed loop systems rely on feedback.

For demonstration purposes think of an assembly plant and a controller that controls operations in the plant. In an open-loop system the controller does not recognize that the plant might be generating output other than what it requested when it set u(t), therefore it is not possible for the controller to modify u(t) to account for variations in y(t). One reason that y(t) might vary from u(t) is because the mathematical model of the plant does not account for a state or event in the plant, in which case u(t) does not directly correlate to y(t). Another possible reason is that external, uncontrollable, events are affecting the plant’s output.

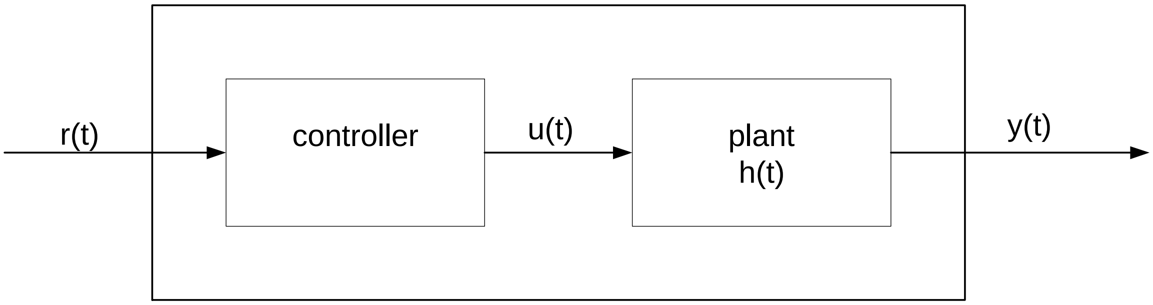

Figure 1: Block Diagram of an Open-Loop System

Figure 1 shows a reference signal, r(t), a control signal u(t), and an output signal y(t). h(t) is the plant’s transfer function and $y(t) = u(t) * h(t)$ where $*$ is the convolution operator. Usually a reference signal is a human controlled input that indicates the desired output to the controller. A volume control or gas pedal qualifies as a reference signal.

In closed-loop system a Proportional Controller dynamically adjusts u(t) based on the error term, e(t), where: $e(t) = r(t) - ky(t)$ where $k = 1$ for unity feedback. A proportional controller computes the adjustment to the control signal, u(t), by simply adding the error term to u(t). In an analog closed loop system the error term e(t) is continuously calculated. Digital systems compute e(t) and u(t) at discrete time intervals, such as: $u(t_2)=u(t_0) + e(t_1)$, where $t_2>t_1>t_0$ and $e(t_1)=r(t_0)-y(t_0)$. Iterative logic in a digital controller easily implements these calculations.

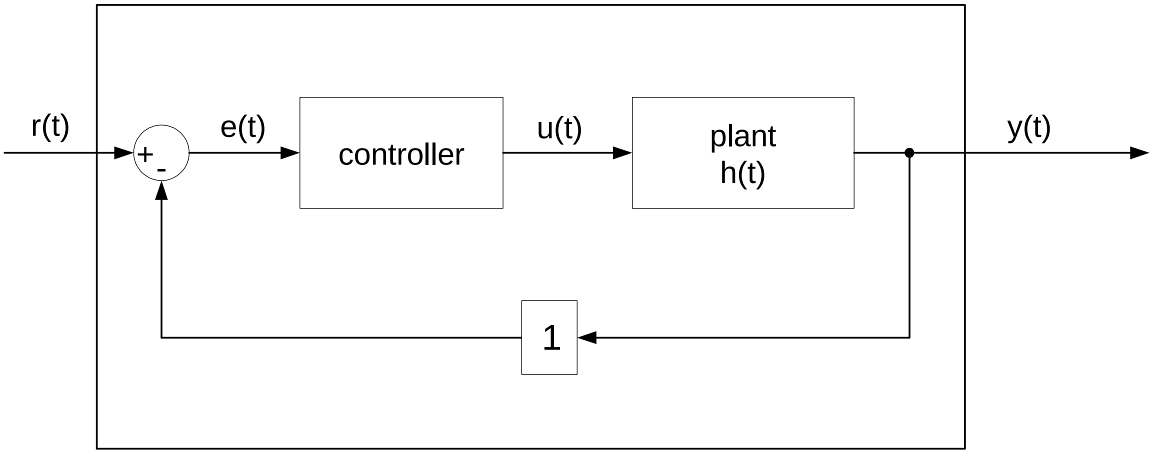

Figure 2: A Closed-Loop System with Unity Feedback

Feedback enables a system to become stable and to return to a stable condition when external events disrupt or perturb a system. In return for being stable a system exhibits a slowed response to changes in a reference signal or to external events. The mechanisms for a control system using feedback are: proportional, derivative, integral or of these mechanisms combined into a PID controller. The PID controller is very common in industry.

The choice of a controller depends on the application's requirements. This lab builds an application that uses the duty cycle of the PWM signal to dynamically adjust ambient light levels. The controller increases the duty cycle of the PWM in order to apply more power to the LED and thereby increase its intensity. The PWM is the control, u(t), that is fed into the plant (that is, LED).

This particular product experiences perturbations on a regular basis. Someone might open a curtain or turn on another light. The controller should respond to increases in ambient light by decreasing the PWM signal's duty cycle which in turn dims the LED (plant) such that the total ambient light matches the reference signal.

Changes to the LED's intensity can occur as fast as the MCU computes and updates the PWM duty cycle register (OCR2). For this application one update every 100 milliseconds is more than sufficient. One milli-second is very slow compared to how fast ambient light fills a room. At speeds of human perception 100 milliseconds is fast but detectable. These timing considerations drive the requirement for only needing to use a proportional controller. Hint: A maximum change of 2% to 4% to the duty cycle every 50 - 100 ms implements a nice smooth transition of the LED's intensity.

As you can imagine this is just the surface of a deep topic. If this material is interesting consider taking other courses that discuss feedback and specifically control systems, such as ECE4510 or

ECE4520 (ECE3510 is a pre-requisite for both courses). However, this lab is a practical hands-on approach. We need just enough information to know that the design is grounded on theory.

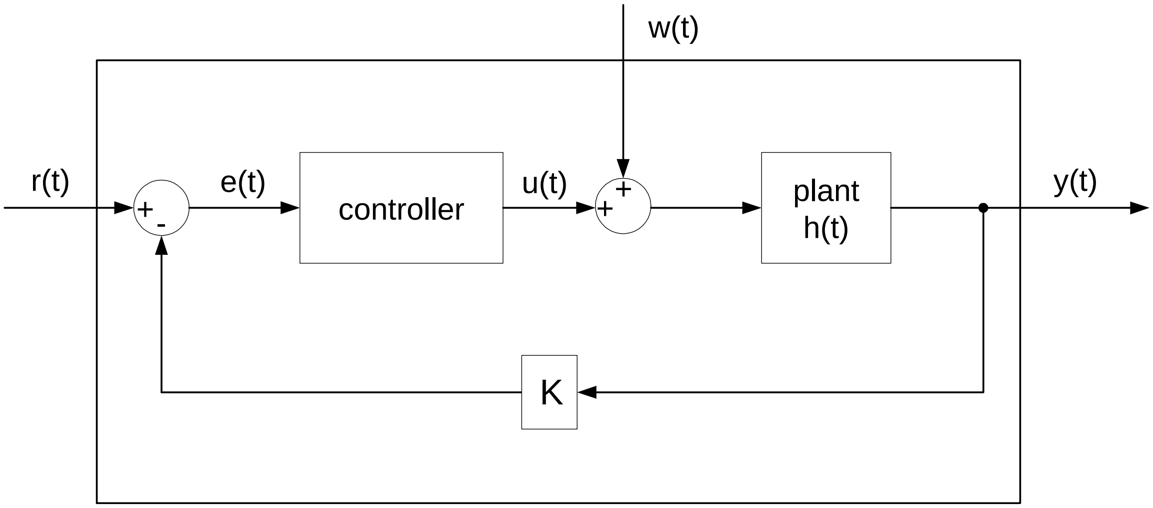

Figure 3: A Closed-Loop System with Feedback "K" and Perturbation w(t)

The error term and output are now defined as: $e(t) = r(t) - ky(t)$ and $y(t) = [u(t) + w(t)] * h(t)$. If the system is linear then the setting of "**K**" is a practical matter that depends on the digital representations for r(t), u(t) and y(t).

There are more complicated feedback systems than shown by Figure 3. Integral or differential controllers perform their respective mathematical functions on the error term, e(t) when computing a updates to u(t). These controllers have different block diagrams than the proportional controller shown in Figure 3. However, from the point of the hardware circuitry only the output term y(t) is fed back into the hardware, typically through an A/D converter.

Once y(t) is back in the controller then the software can perform any calculation. A careful analysis of your code for this lab might reveal elements of integral or differential control logic. There's no problem with this, just be aware that the physical feed of the sensor, y(t), back into the controller doesn't define or restrict the specific type of control logic implemented by your software.
***

In this lab, the plant's function is to generate a desired level of ambient light. The sensor measures ambient light with a Cadmium Sulfide (CdS) photocell. The sensor measures the plant's performance. However, it also measures outside perturbations such as light from other sources that are *not* controlled by the plant.

A proportional control system computes the error term or difference of the reference signal, r(t) and output, y(t), to dynamically adjust u(t). That is, e(t)=r(t)-y(t) and u(t)=u(t)+e(t). If a user adjusts r(t) then the error term exhibits the same adjustment and the re-calculation of u(t) still works.

The circuit, in Figure 4, depicts how we will implement a closed-loop system with the MCU. Port A, bit 15 controls the PWM (channel 3) to the LED and the feedback is controlled by the built in ADC (channel 6) on Port A, bit 6. Remember that you need the keypad so do not use an ADC channel that is also used by the Keypad. Choose the R that provides the best range vs. sensitivity (1K, 2.7K, 22K) for your light sensor. For the best results, the LED should be very close to the light sensor.

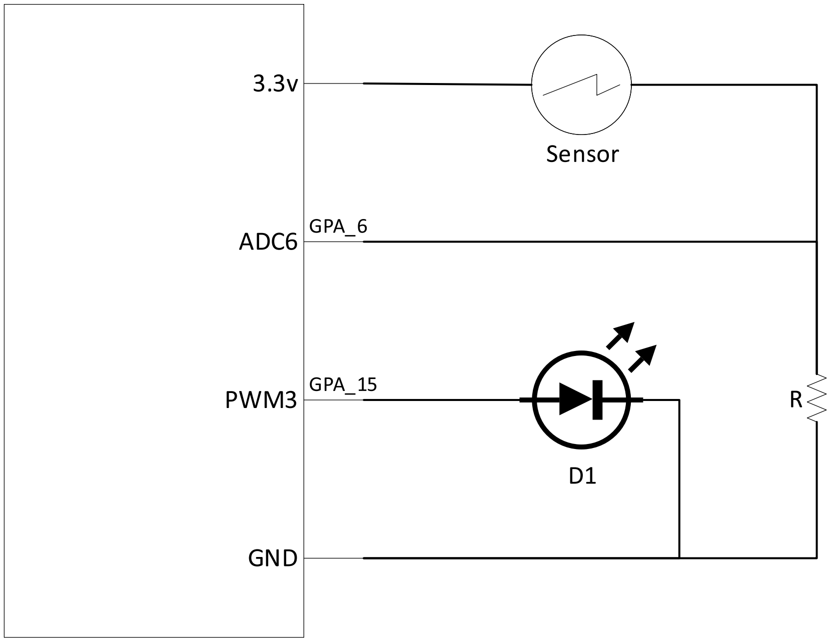

Figure 4: A Closed-Loop Control System

1. Pre-Lab Assignments:

1. TRM, read about the PWM.
2. Driver reference manual, read about the PWM driver.
3. Read the Lecture notes/slides.
4. Complete the Pre-Lab Questions.

Note: Each student must conduct the labs by themselves. You are not allowed to copy codes/content or to pair-up with other students to complete the labs. Each student must demonstrate individual tasks (after completion) to the professor and teaching assistant to get the full marks for the demo.

2. In-Lab Tasks:

1. Construct the Circuit in Figure 4 and test its functionality before connecting it to the MCU.
2. Connect the Circuit to the MCU and write code to implement the closed-loop design. The buttons on the keypad are to be used as reference values for the intensity of the light.

THE VALUES CANNOT JUST BE HARDCODED INTO THE PROGRAM. THE LIGHT INTENSITY IS TO CHANGE ACCORDING TO THE FEEDBACK FROM THE CIRCUIT. The selections (targets) are 0%, 25%, 50%, 75%, and 100% intensity.

3. You should limit the error to +/- 100 and ensure the PWM duty cycle output does not exceed 0xFFFF nor go lower than 0.
4. Display on the LCD the intensity commanded (in percentage), the measured intensity (ADC input), and the duty cycle (PWM output).
5. If a button is selected, change the target duty cycle to the new value.
6. The program is also to output to the LCD the percentage of intensity that the light is actually lit up.
   Hint: You are going to have to experiment with the hardware and values that are returned to correctly adjust the light output.
7. Go through the Lecture Notes for Lab 5, and complete any additional requirements presented during the lecture.

3. Parts List:

1. CdS Photocell
2. Resistors (you select for best range for the sensor 1K-22K Ohm)

4. Deliverables:

1. **To sign-off:** Demonstrate to your professor/lab instructor how your circuit functions and steps through the different intensities of light.
2. **Lab write-up.** Follow the Lab Report Format and the Lab 5 Rubric. The report should include:
   a. Introduction including summary of the lab tasks.
   b. In lab tasks, problems encountered, your solutions, including hardware and software.
   c. Describe your closed-loop system in the lab report.
   d. Include all the data obtained from different intensities of light. There should be 5 data points.
   e. Discussion and conclusion.
   f. Program/code listing for each task.

NOTE: All the lab reports should be typed and turned in as a hard copy on time (check the syllabus for the due date), and a soft copy should be forwarded to the Teaching Assistant: Srikanth Ramadurgam on the due date and time.

Pre-Lab Questions for Lab 5

Name: ______________________________

10 points total.

1. (3pts) Describe the 3 basic components of a PID and give an example of how each is used.

2. (4 pts) From the PWM section of the TRM, list the steps of the PWM Timer Start procedure

3. (4 pts) Here's a C-programming question. Let's say your code has defined a 16-bit signed integer, and reads a 12-bit value from the ADC. What is the maximum value that your code can multiply the ADC value by before storing the result in the 16-bit signed, integer variable, without losing any bits? Now, suppose that for whatever reason you need to multiply the ADC value by 74 but that you also need to divide it by a normalization factor of 165. Write the code, including the variable and constant (if any) definitions, to perform this calculation with ***minimal*** loss of accuracy in the final, signed, integer formatted answer.