ELEC 361 Measurement and Analysis
Lab 6. Fun with Arduino #1 :: Process Control

This **1-week lab** requires you to control a mains heater to maintain a constant temperature.

First steps with Arduino (common all Arduino labs)

Background
Arduino is an open-source electronics prototyping platform that provides flexible, easy-to-use hardware and software. The main Arduino boards are based on one of the Atmel AVR RISC-based microcontrollers, while a vast range of 'shields' (boards that can be plugged on top of the Arduino PCB extending its capabilities) are available to implement sensors and interfaces of just about any kind. Programming is via the Arduino IDE (integrated development environment) using the 'Wiring' language that is a cut-down version of C++.

We will be using the Seeeduino V4.2 that is a breadboard friendly version of the Arduino Uno (important information for set-up) running the Atmega328 microcontroller.

Pre-Lab Reading
Required reading is really skimming through material so you know where to find things:

- Wikipedia background information: http://en.wikipedia.org/wiki/Arduino
- Browse the Arduino website http://arduino.cc/ and look for information on how to program Arduinos.
- Seeeduino V4.2 blurb and circuit diagram at http://www.seeedstudio.com/wiki/Seeeduino_v4.2#Introduction
- Atmel Atmega328 specifications at http://www.atmel.com/devices/atmega328.aspx
- Getting Started with Arduino on Linux at http://arduino.cc/en/Guide/Linux (or Windows, etc)
- '10 Ways to Destroy an Arduino' http://ruggedcircuits.com/html/ancp01.html – make sure you avoid these!
- Arduino language reference http://arduino.cc/en/Reference/HomePage (useful for looking up examples of program statements)

Getting started in the lab

- Connect the computer and Seeeduino via USB – this supplies power to the Arduino.
- Launch the Arduino IDE. Under the Tools menu, ensure the correct Arduino bootloader is selected (Uno), and the correct COM port (not COM 1). Open the blink example, and upload the blink 'example' program. You should see the onboard LED blink (connected to PIN 13), though this program may already be loaded on your Arduino.
- Modify the blink program to make the onboard LED blink on for 2 seconds, off for 0.5 seconds.
- Plug the Arduino into the ELVIS breadboard -- careful please – and connect an ELVIS LED to an i/o pin other than 13. Modify the code to make one ELVIS LED blink.

  (Note that a sketch has two parts: The routine "void setup()" is run first and only once. This is where you define variables. The loop "void loop()" routine is run repeatedly after that.)

Process Control Lab

Control systems are used to keep a desired *process variable* at a *set-point* using measurements on the process variables and adjusting process variables that can be manipulated. There are two main systems of control: *feedback control* and *feedforward control*.

Pre-Lab Reading (for extra edification)
Process Dynamics and Control, 3rd edition, Seborg et. al. (Text for EMAN 403)
General: Sections 1.1, 1.2. Control strategies: Section 1.3 and Section 8.4.

Overall Objectives

Dylan likes to brew beer. He needs help designing a simple ON/OFF controller that will keep his brewing barrel at the desired temperature, which is around 26° C.

In this experiment, you will be using the Arduino, a voltage relay, the analog voltage reading function of the Arduino, and a temperature sensor to measure the temperature of the environment and then program the Arduino to automatically decide whether to turn on a lamp (heater), that is, you will build an ON/OFF controller thermostat. The knowledge attained from this experiment can be applied more widely and creatively, and even sold to Dylan. Please read through this lab sheet thoroughly.

Equipment required

Arduino and programming cable
Voltage relay box and IEC power cable
Temperature sensor
Lamp
LCD Display shield, with buttons

Lab Tasks

Take screenshots of your work throughout the lab and hand in your program code with the final report. You can take photographs of your ELVIS/Arduino setup to make nice documentation for your lab-book.

Mains Switching

1. Connect 5V and GND the ELVIS, and the signal pin to digital pin 11 of the Arduino. Figure 1 shows the inner working of the voltage relay box. The Active mains wire (brown) of the inlet and outlet are connected via the relay. When the signal wire is high (~5V) they are connected together and current flows, when the signal is low they are not connected and so the circuit is open. The Neutral (blue) and Earth (green with yellow stripe) wires are connected straight through. The Earth is also connected to the case for safety.

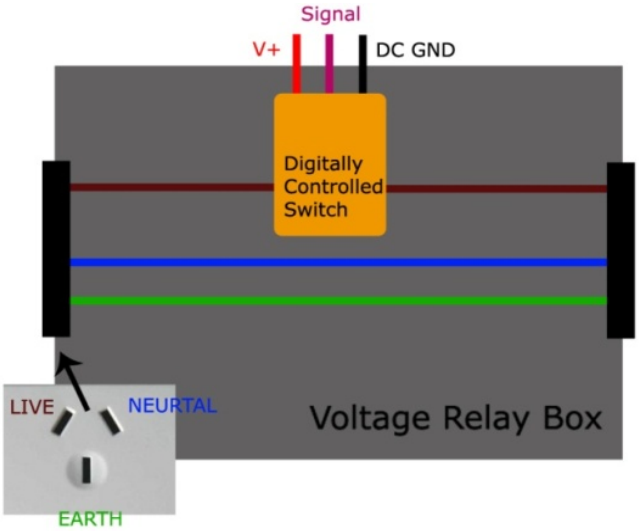

Figure 1 - Relay Box Wiring Diagram

   For more information on the 'digital controlled switch', see reference [2].
2. Connect the lamp and power to the relay box.
3. Using the BLINK sketch to make the lamp flash off and on periodically. How fast can you switch the lamp on and off (is it annoying?).

Temperature Sensor

4. Attach the temperature sensor module to the Arduino. Vcc to 5V, GND to GND of the ELVIS, and signal to ANALOG 1.

5. Search the Internet for code for this particular temperature sensor; make sure that you edit the analog read pin to ANALOG 1. (The A0 pin will be used for the LCD display later on.) This sensor is a "Grove temperature sensor".

6. In your writeup, explain the operation of the temperature sensor.

7. How accurate is the analog read function? (What is the smallest voltage difference it can differentiate?)

8. Now run the sketch, open up the serial monitor and make sure it is reporting a sensible temperature.

User Interface

9. Plug in the LCD display shield to the Arduino and get it to run. To do this, find the name (type) of the LCD shield, download the appropriate sketch that allows input from the buttons, and run it. This code also initializes the LCD and has code to print text to the LCD display.

10. Alter the code so that the LCD screen displays the current temperature. Units please.

ON/OFF Controller

11. Create an algorithm, and write a sketch, that turns the lamp on below a 'low' temperature threshold and off above a 'high' temperature threshold.
    You may wish to define more variables, and use ‘if’, ‘if else’ and ‘else’ statements.
    Test your controller by setting temperature thresholds above room temperature, and pointing the lamp towards the sensor. Try breathing on the temperature sensor.

12. Sketch a graph of what you believe the temperature will be as a function of time.

13. Incorporate the code for the LCD buttons into your program. Edit the buttons code to increase or decrease the temperature set points. Display this variable on the screen along with the current temperature. Now you have built a thermostat.

Further Notes

You have implemented a “bang-bang” (ON-OFF) controller. For a smarter control system, it is desirable to keep the actual value of the variable at the set point value constantly, with no overshoot. One way of doing this is implementing a PID (proportional—integral—derivative) control which measures the rate of change of the variable as well as the error in the variable over time to keep the actual value at the set point. For more information see references [3] and [4].

References

[1] - http://www.seeedstudio.com/depot/Grove-Temperature-Sensor-p-774.html
[2] - http://www.seeedstudio.com/depot/Grove-Relay-p-769.html
[3] - http://en.wikipedia.org/wiki/PID_controller#Proportional_term
[4] - https://www.youtube.com/watch?v=ng2PVOePN68