Experiment 12 – Introduction to the Arduino Microcontroller: Seven-Segment Display Controller with Serial Interface

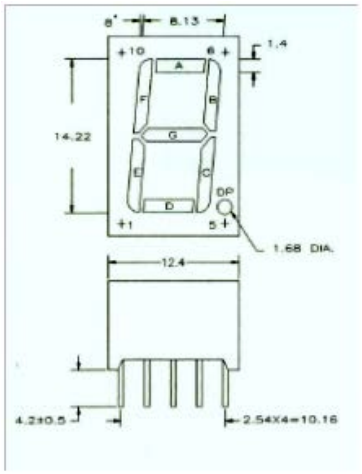
Physics 242 – Electronics

Introduction

The Arduino microcontroller is an inexpensive, versatile computer based on the Amtel ATmega328 microcontroller chip. The version we'll use, called the Arduino Uno (Revision 3), has 14 pins that can be configured as digital inputs or outputs, 6 pins that are analog voltage inputs, and comes equipped with a USB port to connect to a computer for programming or control via the serial port. Arduino can be programmed in a high-level computer language (called Processing, which is a dialect of C++), making it relatively easy to implement complicated logic and control functions through software. The website http://www.arduino.cc contains documentation of the Arduino board and the programming language, with code examples. The free download of the Arduino IDE (Integrated Development Environment) from the Arduino site comes with a library of example programs, which can be readily adapted for your own use.

Procedure

Your object in this lab is to use Arduino to control a seven-segment digital display. Your Arduino should run connected to a computer with the Serial Monitor window open, so that you can send instructions from the computer to the Arduino. You should arrange is so that you can type "1" and the seven-segment display will light up "1", you can type "2" and the display will show "2", and so on.



PIN NO.	FUNCTION
1	ANODE E
2	ANODE D
3	COMMON CATHODE
4	ANODE C
5	ANODE DP
6	ANODE B
7	ANODE A
8	COMMON CATHODE
9	ANODE F
10	ANODE G

The pin connections for the seven-segment display are shown in the diagram above. It is a common-cathode display, which means that the negative side of the diode elements are connected together, and should be connected to ground. The positive side of each diode should be connected, through a 1 kΩ resistor, to the appropriate digital output of your Arduino board. Be sure to use a resistor: if 5 V is applied directly across a diode, it will be destroyed by the large current that flows.

You can use any of the digital outputs on the Arduino board, except pins 0 and 1, which are reserved for serial communications with the computer. So pins 0 and 1 should be left unconnected.

This project has several parts that all need to work together, which can be challenging. In other words, if it doesn't work, might the problem lie in the computer serial connection, the Arduino connections, your circuit breadboard, or your code? Rather than building your complete design and praying fervently that it works before applying power, it will be helpful to **build and test in stages**, which is a general precept for bigger projects. First, get the Blink program to run on your Arduino—then you'll know that the Arduino software and the serial connection between Arduino and your computer is working. Then, construct your circuit and check that you can light up the seven-segment display manually, by connecting to 5 V or ground (be sure to use the resistor!) This verifies that your seven-segment display works. Then, write a simple Arduino program to light up all the elements of your display, or to cycle through all the numbers. Finally, add the code that allows you to select which number is displayed via the serial interface. Figuring out a good sequence of sub-problems to solve (and debug) is the key to getting a more complicated project working.

When your project is working, demonstrate it for the instructor. No lab report required!

Arduino Programming Hints:

1. You might find it helpful to use subroutines (called functions) to do some of the tasks. An example function is defined below:

```
void myFunc() {               //The function does not return a value, so it is of type "void."
   digitalWrite(2, HIGH);   //All this function does is write a HIGH to pin 2.
}
```

This function definition can be placed either before or after (but not within) the loop() function. To invoke or call your function, within the loop() function or another function, just type:

```
myFunc();
```

You can find out more about functions from the Arduino site at http://www.arduino.cc/en/Reference/FunctionDeclaration.

2. Here is example code that makes Arduino read the serial port, and take an action depending on what command you have entered:

```
if (Serial.available() > 0) {
```

```
   int command = Serial.read();
   if (command == '1') {       //Note that single quotes are need to read the character "1"
       takeAction1();          //takeAction1() is your function that is called if the command 1is read
       Serial.println("Command 1 received");      //Arduino can report back via the serial interface
    } else if (command == '2') {
      takeAction2();
      Serial.println("Command 2 received");
   }
}
```