LAB 3: TRADEOFFS BETWEEN HARDWARE & SOFTWARE

Objective

1. Become familiar with implementing Boolean logic in hardware and software.
2. Investigate methods for comparing voltage levels in hardware and software.
3. Reflect on relative merits of implementing solutions in either hardware or software.

Prelab Assignment

1. Read the Arduino code in Appendix A and determine what logic function it implements. Explain how you could speed up the rate at which the function is executed.
2. Write an Arduino program to read in two analog values, then turn on (or off) an I/O pin depending on whether the first analog input is higher or lower than the second analog input.

Boolean Logic

Today's modern computers use binary values (1's and 0's) to implement Boolean logic, which is the algebra of variables that take on only "true" or "false" values. These functions can be implemented in several ways, each of which has their own advantages and disadvantages. A single function is often implemented as a "gate," in which one or more binary inputs are passed through a Boolean function to produce a single binary output. These gates can be comprised of discrete transistors and resistors, or may be comprised of transistors and resistors within an integrated circuit (IC). Boolean logic can also be implemented in software code, thus allowing for true/false branching (often implemented as IF-THEN statements) within programming languages.

Voltage Comparator

Sometimes we desire to see if an analog voltage is above or below some reference level. For instance, we may wish to slow or turn our autonomous vehicle if the forward distance to a stationary object becomes less than some predetermined amount. If information about the forward distance is provided to us as an analog voltage, it would be nice to have an "alert" signal that turns on when that voltage becomes "too high" or "too low." While it is possible to build an op-amp circuit to accomplish this task, there are specialized circuits that have been built for this very purpose; these integrated circuits (ICs) are known as **voltage comparators**. One of the most common voltage comparators is the National Semiconductor LM339 quad comparator.

Rather than use a custom chip like the LM339, we will build our own voltage comparator from a 741 op-amp in this lab exercise. You will find a schematic for the 741 IC in Figure 1. Since a "comparator" circuit compares two input voltages, we connect our reference voltage (V_{ref}) to the negative input a pin 2, and our input voltage (V_{in}) to pin 3. When input signal V_{in} rises above, or falls below, reference voltage V_{ref}, the output voltage (V_{out}) on pin 6 changes polarity.

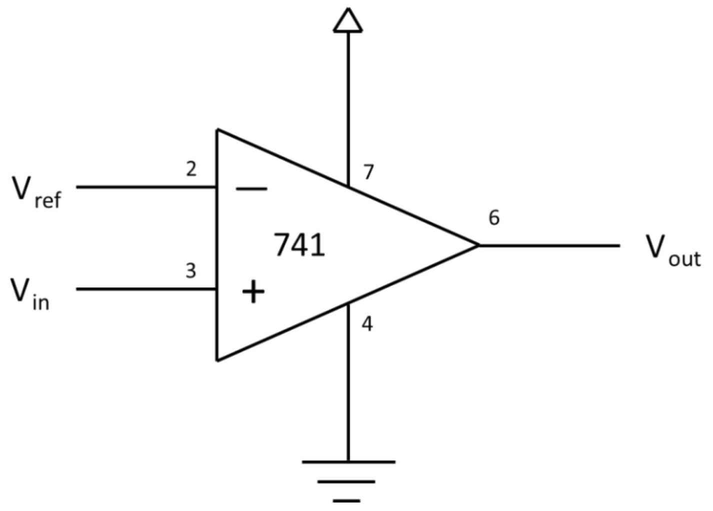


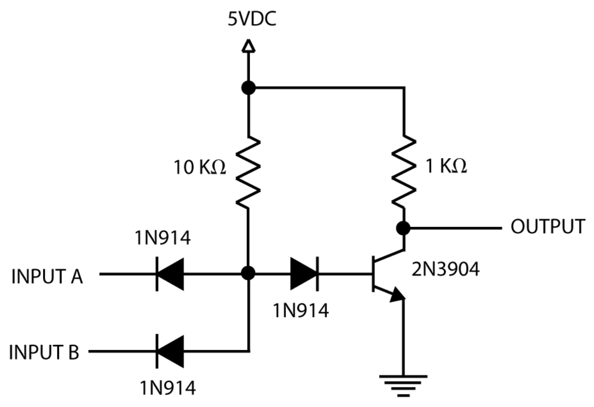
Figure 1: Schematic of the 741 op-amp

Procedure

1. **Boolean operators in hardware and software**

   In this step you will investigate and compare the implementation of Boolean operators in hardware and software.

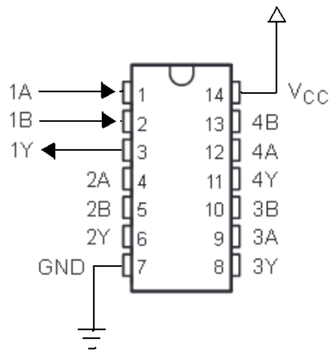
   (a) Build the following circuit on your breadboard. You will use jumper wires coming from GND or 5 VDC to select LOW or HIGH values for INPUT A and INPUT B. (Hint: You might wish to insert an LED between the OUTPUT and GND to easily identify the output state.)



   (b) Complete the truth table for the circuit. What Boolean operator is this?

A	B	Output
0	0	
0	1	
1	0	
1	1	

(c) Construct the following circuit on your breadboard with a 7400 integrated circuit. How does the behavior of this circuit compare to your previous circuit?



(d) Apply a 0-5 VDC square wave from a function generator as a simultaneous signal source for inputs 1A and 1B. Using an oscilloscope, determine the delay between polarity changes for the input and output signals. Plot this delay as a function of frequency.

(e) Write an Arduino program to replicate the circuit in part (a) using DIO 2 and 3 as inputs and DIO 4 as output. Does your code work as expected?

(f) Repeat step (d) for your Arduino program.

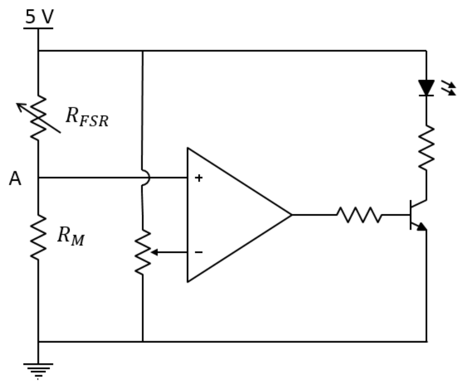
(g) Identify two advantages and disadvantages for the hardware implementation.

(h) Identify two advantages and disadvantages of the software implementation.

2. **Force-sensitive alarm in hardware and software**

In this section you will use a force sensitive resistor (FSR) to provide a source of variable resistance. You will then identify when a critical force has been reached and alert the user with an LED, using either hardware (a comparator) or software (Arduino code).

(a) Build the following circuit on your breadboard. Design appropriate current limiting resistors for the transistor and LED, assuming the 741 op-amp has a 10 VDC upper "rail," and is tied to ground at the lower "rail." Explain how you determined your resistor values. Use the potentiometer to select the force at which the LED lights. Does your circuit work as expected?



Hint: You will be using an FSR with a force sensing range of 0.2 to 20 N. As greater force is exerted on the device, its output resistance will decrease. Appropriate values for R_M are in the range of 3 to 100 kΩ.

(b) Explain how you might accomplish the same task using software instead. Do not implement.

(c) Identify two advantages and disadvantages for the hardware implementation.

(d) Identify two advantages and disadvantages of the software implementation.

Appendix A: Sample Truth Table Code

```
/*
Implements a two variable truth table
*/

// initialize pin 4 as an output
// (it's not necessary to initialize input pins)
void setup() {
pinMode(4, OUTPUT);
}

// the loop routine runs over and over
void loop() {
// read the input pins
int A = digitalRead(2);
int B = digitalRead(3);
// determine output value
int Z = !(A || B)
// write to output pin
digitalWrite(4, Z)
delay(1000);
```