LAB 2: INTRODUCTION TO COMPUTER SYSTEMS

Objective

1. Become familiar with the Arduino Uno microcontroller and Arduino IDE
2. Practice implementing discrete transistors as switches.

Prelab Assignment

There are two basic ways to program an Arduino microcontroller to blink an LED at a particular rate. One method uses Arduino's `delay()` function to halt computation for a specified number of milliseconds. The other method uses the `millis()` function to return the number of milliseconds since the Arduino board began running the current program.

1. Download the code available from the following URLs:
   http://arduino.cc/en/Tutorial/Blink
   http://arduino.cc/en/Tutorial/BlinkWithoutDelay
2. Modify each downloaded code segment so that the output pin stays on for 0.5 seconds, then turns off for 1.5 seconds.
3. Further modify each code segment so that Pin 12 is used as the output pin.
4. Rewrite the comments, putting them in your own words, so that you get a better understanding of the programming language.

Reference sheets, along with other code examples, are available through the following link:
http://arduino.cc/en/Tutorial/HomePage

Microcontrollers

Microcontrollers are devices that contain the same basic components as a computer but are built for the purpose of carrying out a small or specific task within a system. The image in Figure 1 shows the internal components of a microcontroller chip. The central Processing Unit (CPU) is part of the microcontroller that carries out the programming. There are typically two types of memory within a microcontroller: read-only memory (ROM) and random-access memory (RAM). ROM storage contains data that is not meant to be modified. Nonetheless, certain forms of ROM have been occasionally used for writable storage. RAM storage is designed for retaining and accessing user data. The clock signal is what determines how fast the chip operates, as the CPU depends on it to know when to change state and carry out functions. The input/output (I/O) control unit is how the user sends and receives data from the microcontroller.

This lab uses the Arduino Uno microcontroller board, which is programmed using a language similar to C or C++. External features of an Arduino board are shown in Figure 2. Digital Input/Output (I/O) pins can be configured as either input or output ports for communicating with external digital devices. Digital I/O pins marked with a tilde (∼) are able to generate a pulse width modulation (PWM) output that simulates an analog value. Digital I/O pins 0 and 1 double as the USB data lines, so try to avoid using these pins. The RX/TX LEDs will illuminate when an external device communicates with the Arduino board through the USB connection. Power can be supplied to the Arduino in one of three methods: the USB cable, the power jack, or the 5V barrel connector. *Use only ONE of these methods at a time!*

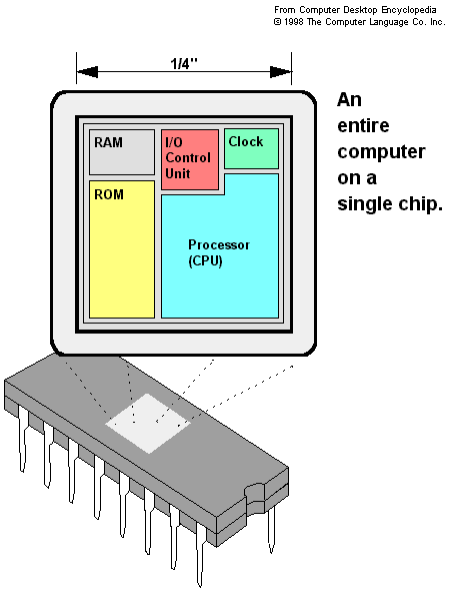

Figure 1: Internal Components within a Microcontroller
(Computer Desktop Encyclopedia, 1998 Ed.)

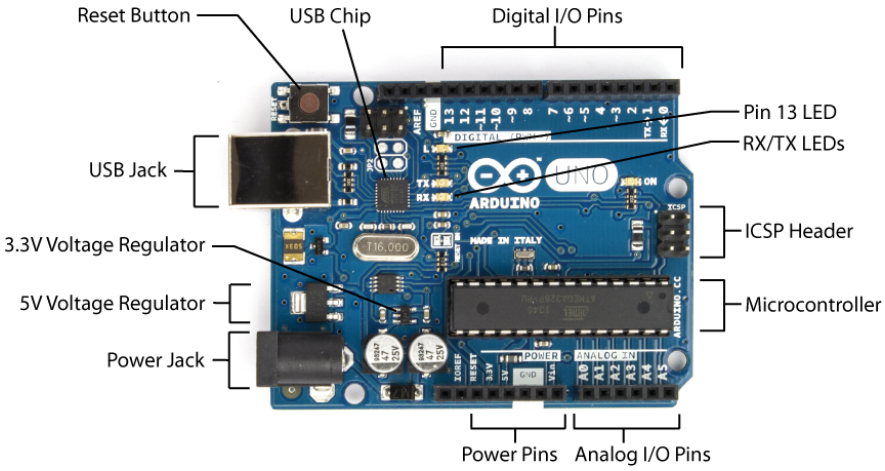

Figure 2: Arduino Uno R3 Photo

Transistors

A transistor is one of the basic building blocks in understanding digital logic. Transistors are incorporated in all sorts of electronic devices. Each transistor can be used as an amplifier or switch. The diagram in Figure 3 shows an NPN transistor in a standard "common-emitter" configuration (so called because the emitter is connected to ground). When sufficient voltage (typical greater than 0.65 V) is applied to the base, current flows freely from the collector to emitter. Digital outputs from an Arduino board supply a voltage of approximately 5 volts. Thus, when Pin 7 goes HIGH, the base-emitter junction of the NPN transistor is "forward biased," and current flows through the light emitting diode (LED). This causes the LED to illuminate. Currents through the transistor base and LED are limited by the 1 kΩ and 150 Ω resistors, respectively.

How does one choose such resistors? Well, let's look first at the base (1 kΩ) resistor. Arduino specifications indicate that its digital outputs supply 5 V, assuming the output current is limited to less than 40 mA. Using Ohm's Law, $V = IR$, we can estimate the base current. If Pin 7 delivers 5 V, and the voltage drop across the forward-biased base-emitter junction is 0.7 V, then the voltage drop across the base resistor is $(5 - 0.7 = 4.3)$ V, and the base current is 4.3 V/1 kΩ = 4.3 mA. This is well below the Arduino's current limit, and is a very conservative choice. You may opt for a less conservative value (lower resistance) if desired. However, note that a base resistor smaller than 107.5 Ω will cause the Arduino's current limit (40 mA) to be exceeded!

Similarly, the 150 Ω resistor limits current through the LED. Each LED model has different specifications, but the LEDs listed in the syllabus have a forward voltage rating of 2 volts. Thus, the voltage drop across the collector resistor (150 Ω) is $(5 - 2 - 0.2 = 2.8)$ V. (The 0.2 V term is the voltage drop across the C-E junction when the transistor is operating in its saturated, or "closed switch" region. It is common to ignore this voltage, as it is quite close to zero.) This tells us that the current flow through the resistor, LED, and C-E junction is going to be around 2.8 V/150 Ω ≈ 19 mA. This is very close to the current flow of 16 to 18 mA suggested by the manufacturer for "normal" operation. (Remember that each model of LED will have its own operating characteristics; appropriate current limits can be found in the device's spec sheets.)

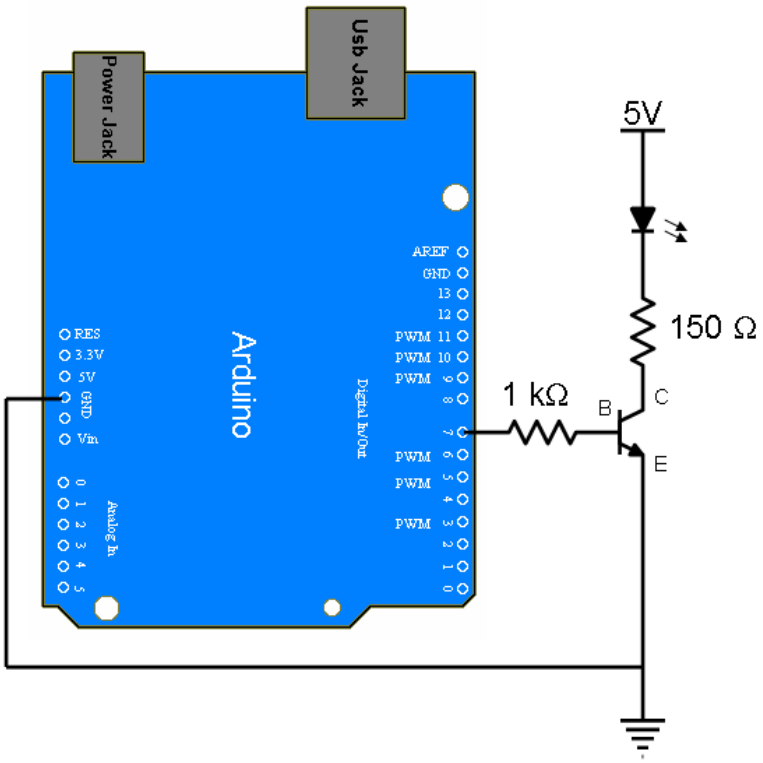

Figure 3: NPN Transistor Operating as a Switch

Procedure

1. **Connecting the Arduino**

   Before starting the Arduino integrated development environment (IDE), connect your Arduino to the computer. It is probably easiest to connect it to one of the USB ports on the front of the computer. If this is the first time using the board on the computer, it will prompt you to install the driver.

   If you experience trouble installing the driver on a lab computer, ask your TA for assistance.

2. **Blinking an LED**

In this task, you will use the two code segments you modified as part of the Prelab Assignment. Repeat the following steps for both programs you created:

(a) From the Arduino IDE, use the "Verify" command to compile your code.

(b) Use the "Upload" command to transfer your program into the Arduino board. (If you get a "not in sync" error, you may need to change the serial port. The Arduino is on a COM port that is NOT COM1, typically COM3 or greater. Also make sure that nothing is externally connected to pins 0 and 1 while uploading.)

(c) Build a circuit that connects one of the Arduino's digital outputs with an LED using an appropriately sized current-limiting resistor. What is the value of your current-limiting resistor? Include a circuit diagram in your lab report.

(d) Does the program work as expected?

(e) How fast can you make the LED blink before it appears to always be on? Express your answer in Hertz (Hz). *Hint: what type of waveform have you created at the digital output?*

3. **Using a Switch with Pull-Up or Pull-Down Resistors**

There are two methods you can use to connect a switch to digital input pins on the Arduino UNO, as shown in Figures 4 and 5. In each case, V_{CC} is a logic voltage supply (you may assume 5 V) and DI is where a high input impedance measurement device will connect. *Answer these questions based on the circuit diagrams, do not build the circuits.*

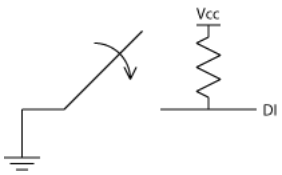

Figure 4: Switch with a "Pull-Up" Resistor

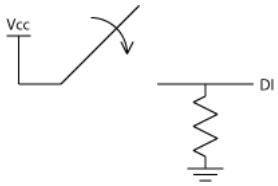

Figure 5: Switch with a "Pull-Down" Resistor

(a) Consider a bare wire on a work surface, unconnected to a circuit.

i. Is the wire voltage zero? *Recall: the wire is not connected to ground.*

ii. Is the wire voltage 5 V? *Recall: the wire is not connected to power.*

iii. Can you predict the voltage on the wire?

(b) With the pull-up resistor, what is the voltage at DI when the switch is open? When the switch is closed?

(c) Without the pull-up resistor (open-circuit between V_{CC} and DI), what is the voltage at DI when the switch is open?

(d) With the pull-down resistor, what is the voltage at DI when the switch is open? When the switch is closed?

(e) Without the pull-down resistor (open-circuit between GND and DI), what is the voltage at DI when the switch is open?

(f) When might pull-up or pull-down resistors be useful in building circuits that convey digital information?

4. **Working with a Transistor**

   Build the circuit shown in Figure 3 on a breadboard. While wiring your circuit, make sure one of the ground pins on your microcontroller is connected to ground on the breadboard. Also verify that you have included a current limiting resistor between the output of your Arduino board and the transistor base. Modify one of the blink codes from Part 1 to turn on the LED for 1 second and off for 0.5 seconds. Use the oscilloscope to examine the switching behavior of the transistor.

   (a) Measure the delay time and rise time of the transistor. What is the turn-ON time?

   (b) Measure the storage time and fall time of the transistor. What is the turn-OFF time?

   (c) Are the turn-ON and turn-OFF times within the transistor specifications?

   (d) What current is applied to the transistor base when the digital output is HIGH? Is this within specification for the Arduino output? *Hint: use Ohm's law to determine this value.*

   (e) What would happen without the current-limiting base resistor? *Answer analytically, do not test!*

   (f) What is the voltage drop across the LED when the LED is on?

   (g) What current flows through the LED when it is turned on? *Hint: Apply Ohm's Law on the collector resistor, not the LED.*

   (h) What would happen without the current-limiting collector resistor? *Answer analytically, do not test!*

5. **Working with Simulink**

   Familiarize yourself with Arduino Target on Simulink.

   http://www.mathworks.com/help/simulink/arduino.html

   Open a new Simulink Model and insert a Digital Output block from the 'Target for Use with Arduino Hardware Library' (not the 'Arduino Target' Library!). Insert a Pulse Generator block from Sources under the Modeling Block Libraries.

   Change the properties of the Pulse Generator to be sample based rather than time based, and modify it such that it produces a signal that turns on the LED for 1 second and off for 0.5 seconds. Save your model.

   Run the model by first clicking Tools, then selecting Run on Target Hardware → Prepare to Run. In the 'Run on Target Hardware' pane that opens, set the Target hardware parameter to Arduino Mega 2560 or Arduino Uno. Finally, click the Tools menu, and select Run on Target Hardware → Run. This action automatically downloads and runs your model on the Arduino hardware. Once setup, changes can be made to the model and the model can be again downloaded to the Arduino by selecting Run on Target Hardware → Run, or pressing Ctrl-B.

   Does the program work as expected? How does this compare to writing the code in the Arduino software? How might you use this in the future?

6. **Bonus: Variable LED Brightness (+5)**

   Place a potentiometer in series with the 150 Ω resistor in Figure 3. What is the effect of changing the potentiometer setting? For several resistance values determine the voltage drop across, and the current through, the LED. Compare these to the perceived brightness of the LED and include your results as a table in your report. Now that you've seen its ability to dim an LED, how else might you take advantage of a potentiometer's behavior?