HANDS-ON LAB INSTRUCTION SHEET – Learning Kit MODULE 5

DIGITAL AND ANALOG INPUTS

NOTES:
If you did not finish Modules 1 through 4, be sure to finish them NOW *before starting this Module or you will fall behind the rest of the class.* ***Labs MUST be done in order.***

BILL OF MATERIALS

(1) **Arduino UNO R3 Microcontroller & USB Cable**
(1) **RED** Light-Emitting Diode (LED) - *or any other color*
(1) **Single-Pole, Single-Throw Push Button Switch**
(1) **Photoresistor (sometimes called a Light Detecting Resistor or LDR)**
(1) **1KΩ (1000 Ohm)** Resistor (*actually any value **220Ω** through **1KΩ***)
(1) **10KΩ (10,000 Ohm)** Resistor
(1) **10KΩ (10,000 Ohm)** Potentiometer – if available

Lab 5A: REVISING THE BLINK SKETCH EXAMPLE CODE

What do you need to change in the LOOP FUNCTION code section of the BLINK sketch to make the code turn the LED 'ON' for five (5) seconds and then 'OFF' for two (2) seconds?

Starting with the code BlinkLED, with **led** = I/O Port #13, revise the delay code lines for
Line #10: **delay(5000);** *// wait for 5000 milliseconds = 5 seconds*
Line #12: **delay(2000);** *// wait for 2000 milliseconds = 2 seconds*

QUESTION 5A.1 and 5A.2 : What are the duty cycles of the #10 and #12 LEDs?
ANS 5A.1: #10 LED Duty Cycle = _____ %
ANS 5A.2: #12 LED Duty Cycle = _____ %

DIGITAL INPUT TECHNIQUES:

*Simple Switches; **digitalRead**, **while**, and **if** commands*

An **INPUT** is any signal *entering* an electrical system. Both **digital** and **analog** devices can be inputs. A digital signal in our case is either a ZERO (0v) OFF input or a 1 (+5v) ON input. Digital inputs can come from switches and Keyboards or O/1 outputs from specially designed sensor devices.

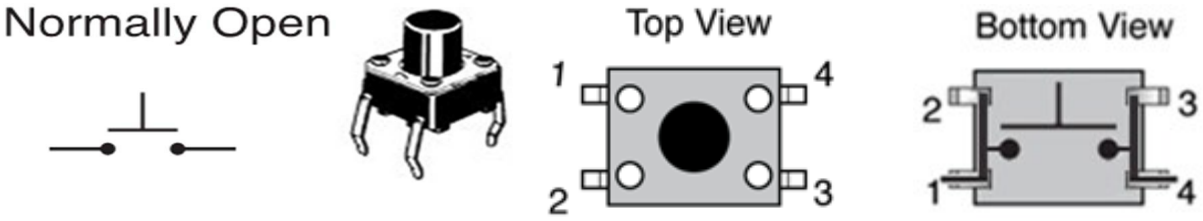


Figure 5.1 Normally Open Single-Pole, Single-Throw Push Button Switch Schematic, pictorial, and Wiring views

Revision 211021: This document is available electronically at:
http://www.CharlesRubenstein.com/222/**05Lab_LK.pdf**

This part of the lab investigates how we can use the Arduino to sense digital ON/OFF inputs. The simplest digital ON/OFF input device would be one that can be approximated by a switch closure. Included in your components kit is a Push-Button Switch. **Push Button Switches** are *Single Pole, Single Throw* **(SPST) inputs** that can be either Normally Open (**NO**) – meaning that they are initially **OFF** and no current flows through them as seen in Figure 5.1. Or, they can be Normally Closed (**NC**) – meaning they are initially **ON** and current flows easily through them as seen in Figure 5.2:

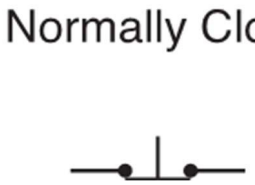


Figure 5.2 Normally Closed Single-Pole, Single-Throw Switch

ALL SPST switches use similar schematic symbols*, which, for normally open (NO) switches in addition to those above include:*

Figure 5.3 Normally Open (NO) Single-Pole, Single-Throw Switch Schematic Symbols

Pushing a push button switch closes the circuit and changes its 'state' (*e.g., **OFF** to **ON***) as noted in Figure 5.4:

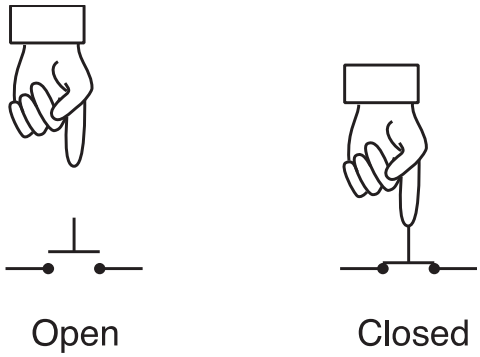


Figure 5.4 Single-Pole, Single-Throw Switch Actions

Releasing the push button returns it to its original state (*e.g., **ON** to **OFF***) – *often with a bounce! We will discuss the challenges of a spring switch 'bouncing' after you remove your finger from it in high-speed logic circuits.*

*A **slide switch** or **toggle switch** is able to be set as either open or closed depending on the position of the toggle or slider. Most ON-OFF switches are either toggle or slide switches that maintain their position. Other switches (double-pole, double-throw, etc.) can control multiple circuits.*

READING DIGITAL INPUTS

Each Digital Input needs a pinMode command in the setup function:

pinMode (pinNumber, INPUT); // *Make sure to use ALL CAPS for **INPUT***

Then, inside the loop function, we can read the value of a digital input:

"NAME" = digitalRead(pin#); // Note the spelling of **digitalRead**

Example:

val = digitalRead(7); // read value of I/O Pin #7, store in val

*Please note that when we name variables and functions using two-word names the first name is NOT capitalized, but the second name is – e.g., **firstSecond**.*

*We can connect digital inputs using any of the 14 digital I/O pins # 0 through #13. The Digital Input at I/O Port "pin#" is read and its value stored in the variable space reserved for 'NAME' Although a digital I/O can ONLY be a 0 or a 1 = integers – the '**int**' declaration is not required,*

We can always set the digital input as an **int** or integer variable. In the following example we set **buttonState** (*NOTE THE SPELLING!!!)* equal to the function **digitalRead** *which returns a zero or a one depending on the state of the **pushButton**:*

int buttonState = digitalRead(pushButton);

THE while() FUNCTION

Once we define the digital I/O Pin in a push button circuit, we can also define the state of the push button as global variables before we start the Setup function:

```
int pbPin = 2;              // Digital input on I/O Pin #2
int pbPressed = FALSE;      // Status of the button is: initially not pressed
```

Then, in the setup function we can both define the **pinMode** and also create a loop that instructs the program to wait until the push-button is pressed using a **while** command. If we used a normally CLOSED switch, we would be looking at

```
pinMode( pbPin, INPUT);
while (!pbPressed)  // Note: !pbPressed means the push-button is NOT pressed.
```

*(recall that the "**!**" is a Boolean "NOT"operator)*

We continue cycling and cheking the status of digial I/O Pin #2 UNTIL the switch is pressed and becomes a "1" – THEN we continue with:

```
{   pbPressed = digitalRead ( pbPin );   }
```

which stores the value "**1**" in **pbPressed**, and then closes/continues the Setup function, after which the Loop function begins.

THE if( ) / else FUNCTION

We could also define the digital I/O Pin state as the global variable buttonState before we start the Setup function:

```
int buttonState = digitalRead(2);
```

In this case we read status of a switch at digital I/O Pin#2 and store it as **buttonState**. As above, a normally Open (NO) switch would report a "0" to the Arduino.

Then, in the Loop function we would expect to see something like:

```
if(buttonState == LOW) {
// Check to see if buttonState equals "0" (closed).
// YES?  Perform the operations coded here!
        /* all the code for buttonState = 0 goes here
        And then return to the if statement.
        */
// NO? skip these program statements and go to 'else'
}
else {              // the other set of codes goes here!
}                   // at this point, end the Loop and executes the Loop function again.
```


Revision 211021: This document is available electronically at:
http://www.CharlesRubenstein.com/222/**05Lab_LK.pdf**

Lab 5B

Pull-Up and Pull-Down Resistors

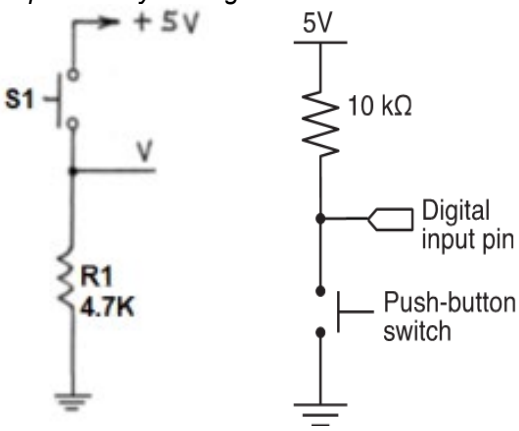
We can use a resistor (*either 4.7K or 10K – it doesn't matter which*) as a **pull-down resistance** to establish a resistance across which we placed +5 volts when a switch was closed as in Figure 5.5.

When the switch is open or NOT pressed, the voltage at v is **0 volts** (**logical 0**) and when the switch is closed or pressed, the voltage at v became **+5 volts** (**logical 1**). The resistor is used to limit the current flowing in the input circuit.

QUESTION 5B.1 and 5B.2: ***Calculate*** the current that would be flowing in the Pull-Down Resistor circuit of Figure 5.5 when the switch is closed for both a 4.7K and a 10K resistor:
ANS 5B.1 for 4.7K resistor: _____ **mA**
ANS 5B.2 for 10K resistor: _____ mA

Arduino circuits typically use a **Pull-Up Resistor** circuit for digital inputs based on limiting the current flowing from our +5v power source into the I/O Port. In this case the input to the digital I/O Port is a "**1**" when the switch is **OFF** and a "**0**" when the switch is pressed or **ON**. ***NOTE: Once again, the resistor value is not critical as its only function is to limit current flowing into the Uno.***

"Download" the Button example code in the IDE:
File → Examples → 02.Digital → Button
And use a Digital Sensor (a.k.a. a simple push button switch and 'Pull-up' Resistor) to control the onboard **L** LED *NOTE: The 10K resistor value is not critical…*
Your component layout might look like this:



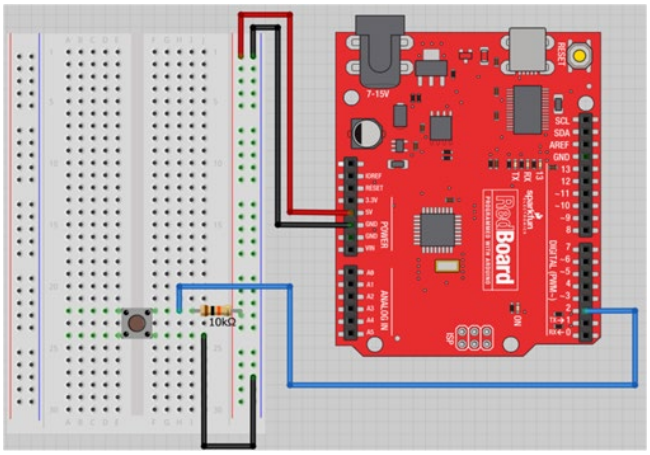


Figure 5.5 Switch Circuits with Pull-Down Resistor; Pull-Up Resistor, and Arduino Layout

The code might look like:

```
        int buttonPin = 2; int ledPin = 13; int buttonState = 0;
void setup( ) {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);  }
void loop( ) {
  buttonState = digitalRead(buttonPin);
        if (buttonState == HIGH) {
                digitalWrite(ledPin, HIGH); }
        else {
                digitalWrite(ledPin, LOW);    }
}
```


Revision 211021: This document is available electronically at:
http://www.CharlesRubenstein.com/222/**05Lab_LK.pdf**

Lab 5C: Pull-Down Resistor Button Input

Revise your circuit to use a Pull-DOWN resistor as shown:

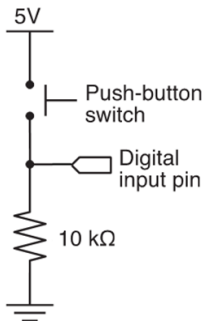


Figure 5.6 Switch Circuits with 10K Pull-Down Resistor

QUESTION 5C.1: Without changing the basic "Button" code, what – if any - changes did you find?
5C.1 ANS: _____

QUESTION 5C.2: What do you need to do to revise the code to turn the onboard "**L**" LED **ON** by pressing the switch?
5C.2 ANS: _____

ANALOG INPUTS

Analog inputs can come from Keyboards, a computer Mouse, and a variety of variable sensors including infrared sensors and biometric sensors that provide a variable voltage output depending on the condition of the sensors, or are created from other parts of an electronic circuit.

The Uno converts analog inputs from each of the six analog input pins **#A0** through **#A5** into a binary number using a built-in 10-bit A/D Converter analog to digital converter (**ADC**).
An analog input is converted from 0 volts to +5 volts into digital numbers 0 through 1023 - a total of 1024 values.

Recall that the output of a basic Voltage Divider circuit depends on a ratio between the resistance of individual resistances, a potentiometer, or any variable resistance device:

$$\mathbf{Vout = Vin\ [\ R_2 / (R_1 + R_2)\ ]}$$

We can use individual, discrete, resistors to make a voltage divider, or, if we have a potentiometer, a single pot allows use to vary the resistance without replacing resistors:

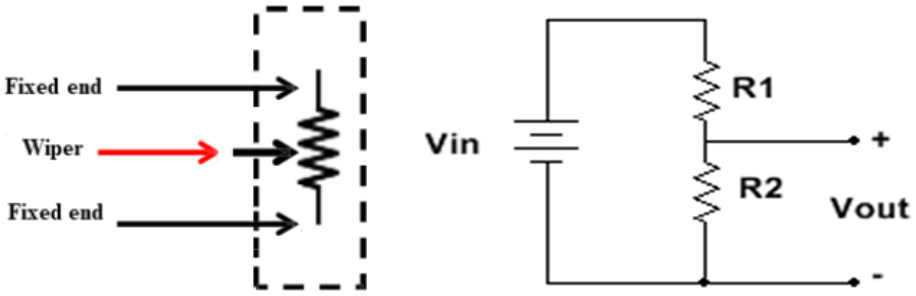


Figure 5.7 potentiometer = Variable Resistor; becomes Voltage Divider

Revision 211021: This document is available electronically at:
http://www.CharlesRubenstein.com/222/**05Lab_LK.pdf**

We could also use a variable resistor sensor as R2 (or R1) in the voltage divider circuit. Those devices include variable resistor sensors whose values change with temperature (thermistors), light (light detecting resistors = LDRs, or photoresistors or photocells), and strain gages,

Lab 5D: Current Limiting

Resistors in LED Circuits
Since we are familiar with LEDs, we can easily see that **LEDs** can be dimmed by adding a second current limiting resistor (**R2**) in series with their normal current limiting resistance (**R1**).

Once again, using a potentiometer would allow a wider range of dimming possibilities, but using discrete resistor pairs will do the same job. R1 is a 1K Ohm current limiting resistor.

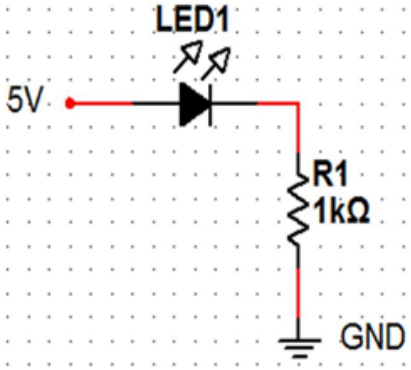


Figure 5.8 LED with 1K Current-limiting Resistor

QUESTION 5D.1: To what value does R1 (alone) limit the LED current flowing in the circuit?
5D.1 ANS: _____ mA

Adding several different resistor values for R2 (or a 10K Ohm potentiometer) allows for much lower currents…

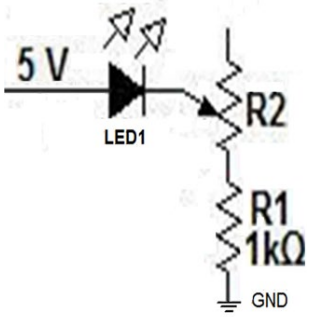


Figure 5.8 LED with Potentiometer and 1K Current-limiting Resistor

QUESTION 5D.2: If R2 is 10K what value is the current flowing in the LED circuit ?
5D.2 ANS: _____ mA

Wire the circuit and use a few different values of discrete resistors – or a 10K pot - and test out this concept.

Reading Analog Inputs

Before we can use an analog input, much like we had to do with our digital inputs, we have to read it! We can read the value of an analog input which is digitized by the Arduino's built-in 10-bit A/D Converter (ADC). *This means that you get input values from 0 (zero volts) to 1023 (5 volts) from the six analog input pins **#A0** through **#A5**.*

"NAME"=analogRead(analogPin#);

The **Analog Voltage Input** at analog input port "**analogPin#**" is read and its value stored in the variable space reserved for '**NAM**E'

Examples:

val = analogRead(0); // **read** value of analog Pin #**0**, store in **val**

int sensorValue = analogRead(0);

Lab 5E: Resistive Sensor-Controlled "L" LED Blinking

We can create a Variable Rate LED Blinking using either the Potentiometer or Resistor pairs. Presuming that you have already loaded the USK Guide examples; Download the code at:

Examples > USK Guide > Circuit #2

NOTE: *The potentiometer - or discrete resistor - voltage divider is connected to analog I/O Pin #A0 and we have an LED with 1K current limiting resistor connected to I/O Pin #12 but we will initially use the onboard LED (**L** = I/O Pin **#13**) for testing:*

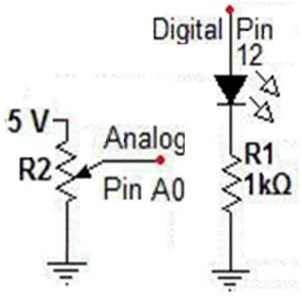


Figure 5.9 Analog (Potentiometer-controlled) Voltage Input; LED and 1K Resistor Output

Global variables:

```
    int sensorPin = 0;     // analog pin #A0 gets input
    int ledPin = 13;       // Use onboard "L" LED = I/O Pin #13
void setup() {
    pinMode(ledPin, OUTPUT); // Onboard "L" LED used
}
void loop() {
    int sensorValue;
    sensorValue = analogRead(sensorPin);
    digitalWrite(ledPin, HIGH); delay(sensorValue); // Turn ON LED, wait
    digitalWrite(ledPin, LOW);  delay(sensorValue); // Turn OFF LED, wait
}
```

*NOTES: The analog voltage input is an integer variable named "**sensorValue**" whose value will be used to set A0 to be used for the delay (in milliseconds).*

QUESTION 5E.1: Can you see the blinking at the highest analog input voltage?

5E.1 ANS: _____YES _____ NO

Revision 211021: This document is available electronically at:
http://www.CharlesRubenstein.com/222/**05Lab_LK.pdf**

Lab 5F: Resistor Controlled "#12" LED Blinking

The digital output in the last lab was sent to onboard "**L**" LED (***pin #13***). *Now let's "activate" the external LED at I/O Pin #12:*

Revised Sketch – *USING LED @ I/O Pin #12:*

```
// Variable Rate Blink – Potentiometer/Resistor Controlled
        int analogPin = 0;      // val = pin #A0
        int val;                // val = integer
void setup() {
        pinMode(12, OUTPUT); // LED at digital I/O pin #12
}
void loop() {
        val = analogRead(analogPin); // Read A0, set delay value
        digitalWrite(12, HIGH); delay(val); // Turn ON LED, wait
        digitalWrite(12, LOW);  delay(val); // Turn OFF LED, wait
}
```

NOTES: ***Compile and upload to your Uno.***
Vary your resistor ratios (or the potentiometer setting).

QUESTION 5F.1: Can you see the blinking at the highest analog input voltage?
5F.1 ANS: _____YES _____ NO

Cadmium Sulfide CdS Photoresistor or Photocells
Photoresistors – or Light Detecting Resistors (LDR) are specially designed resistors whose 'dark' and 'light' resistance values are great enough for the device to act as a light detector, or even as an ***amount of light*** **detector.**
A photoresistor is typically put into a voltage divider circuit with either a fixed **Pull-down** resistor circuit or a **Pull-up** resistor circuit:

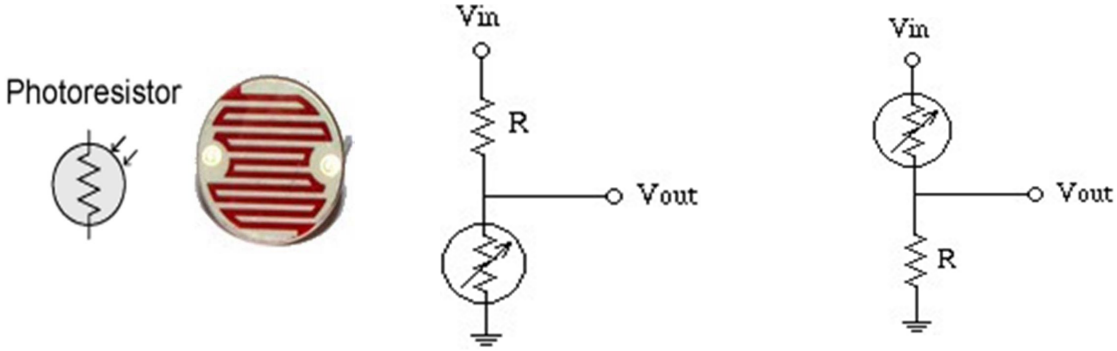


Figure 5.10 Photoresistor Inputs with Pull-Up and Pull-Down Resistor Schematics

In the Pull-up resistor circuit at the right, the *Voltage output* **DECREASES with light:**
(Rphoto*Vin) / (Rphoto+R) = Vout
Conversely,
for the Pull-down resistor circuit at the left, the *Voltage output* **INCREASES with light:**
(R*Vin) / (R+Rphoto) = Vout

Best results are returned when **R= sqrt (Rphoto_dark*Rphoto_bright)**

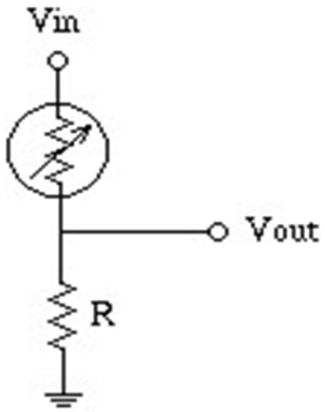
Revision 211021: This document is available electronically at:
http://www.CharlesRubenstein.com/222/**05Lab_LK.pdf**

Lab 5G: Resistive Sensor-Controlled LED Blinking

Now let's replace either **R1** (pull-down) or **R2** (pull-up) with a photocell.
(If you are using a potentiometer, replace it with a 1K resistor and a photocell)

5G Part 1 - PULL-DOWN SENSOR CIRCUIT

Change the analog input to the **pull-down resistor circuit**:
Vary the amount of light falling on the LDR by covering the LDR with your finger for 'dark' mode and using a smart phone flashlight for 'light' mode.



QUESTION 5G.1: What happens to the blink rate in 'dark' mode??
5G.1 ANS: _____

QUESTION 5G.2: What happens to the blink rate in 'light' mode??
5G.2 ANS: _____

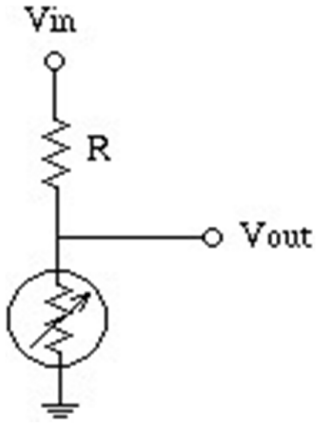
QUESTION 5G.3: Can you actually see the blinking at the highest rate?
5G.3 ANS: _____**YES** _____ **NO**

Figure 5.11 Photoresistor with Pull-Down Resistor Analog Voltage Input

5G Part 2 - PULL-UP SENSOR CIRCUIT

Change the analog input to the **pull-up resistor circuit:**

Vary the amount of light falling on the LDR by covering the LDR with your finger for 'dark' mode and using a smart phone flashlight for 'light' mode.



QUESTION 5G.4: What happens to the blink rate in 'dark' mode?
5G.4 ANS: _____

QUESTION 5G.5: What happens to the blink rate in 'light' mode?
5G.5 ANS: _____

QUESTION 5G.6: Can you actually see the blinking at the highest rate?
5G.6 ANS: _____**YES** _____ **NO**

Figure 5.12 Photoresistor with Pull-Up Resistor Analog Voltage Input

5H: SUMMARY:

QUESTION 5H.1: Which do you think is the best circuit to use, and why?
5H.1 ANS: _____

Revision 211021: This document is available electronically at:
http://www.CharlesRubenstein.com/222/**05Lab_LK.pdf**