ECE102 In-class LabJack Exercise Guide #2

There are 2 parts in this in-class LabJack exercise guide: Part 1: *Control LabJack from MATLAB,* Part 2: *Design a Counter with a Switch*. In this guide, there are two working examples to help students work on the questions in the worksheet and the class project. Record your answers on the worksheet and submit the worksheet at the end of the class.

<u>LabJack Kit for Today's Exercise</u>

1. LabJack Box – one LabJack U3 (either LV or HV), one USB cable, and one Screw Driver
2. One Breadboard
3. One Red LED and One Green LED
4. Connection Wires
5. MATLAB Sample script files from the class website

<u>Objectives of Today's In-Class Exercises</u>

Part 1. Review Labjack Hardware and Run LabJack from MATLAB

1. Take a LabJack, USB cable, screw driver out of the box and connect the LabJack to the computer
2. Review all the channels on the LabJack
3. Run a MATLAB program to control the LabJack
4. Take voltage measurements using the LabJack
5. Use a console prompt to generate an analog voltage

Part 2. Switch Circuit with LEDs

1. Learn how a push-button switch works
2. Learn about switch bouncing
3. Count switch events with a "while" loop

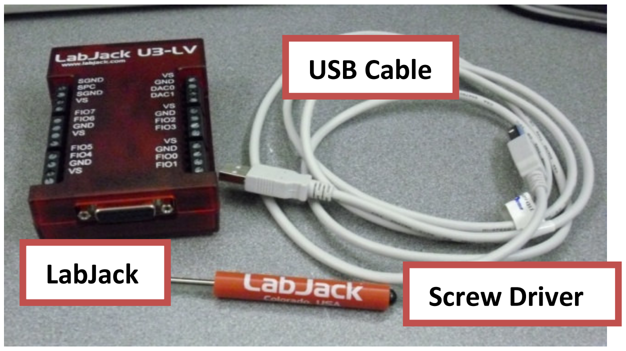
<u>LabJack Tutorial</u>

Before you start working on these exercises, you need to install MATLAB drivers on your computer. Follow the instructions described in the file "Installing_MATLAB_Driver.pdf", which is available on the course web site.

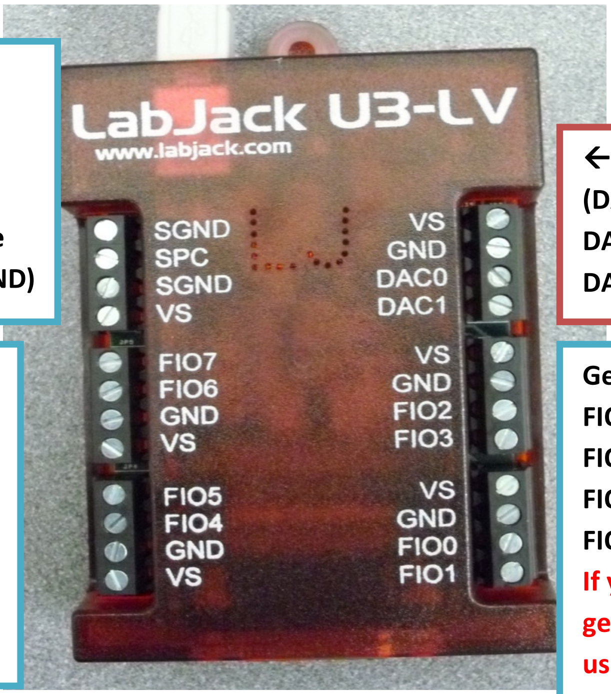
Part 1. Review LabJack Hardware and Control LabJack from MATLAB

1. Review LabJack

LabJack is a data acquisition device (DAQ) used to measure analog and digital (Logic High or Low) voltages and generate analog and digital voltages to compute current, power, temperature, force, and etc. The LabJack is connected to a computer through USB to share data and perform instructions. This time, we will use **MATLAB** to control LabJack.



Let’s review LabJack hardware again.
The channels which we will use today are VS (= constant 5V), GND, DACx, FIOx channels.



VS = supply voltage (always 5V)
GND = common ground

SPC = Reset Pin
SGND = Self-Reset GND with Fuse
(Use this pin for power supply GND)

General Purpose I/Os
- **can read analog in, digital in and write digital out**

FIO4 = channel 4
FIO5 = channel 5
FIO6 = channel 6
FIO7 = channel 7

← Digital-to-Analog Converter (DAC)
DAC0 = channel 0
DAC1 = channel 1

General Purpose I/Os
FIO0 = channel 0
FIO1= channel 1
FIO2 = channel 2
FIO3 = channel 3
If you use DAC0 and DAC1 to generate a voltage, you can NOT use FIO0 and FIO1 because DACs take over FIO0 and FIO1.

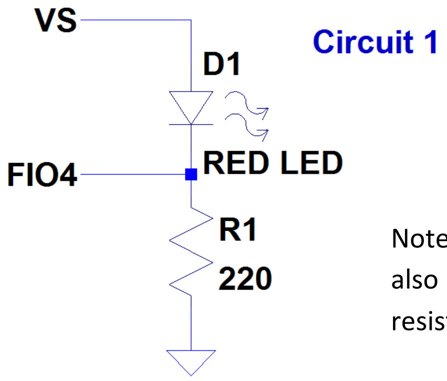
Let’s learn some useful MATLAB commands to control LabJack by a simple exercise.

Exercise 1: Take voltage measurements with LabJack and MATLAB

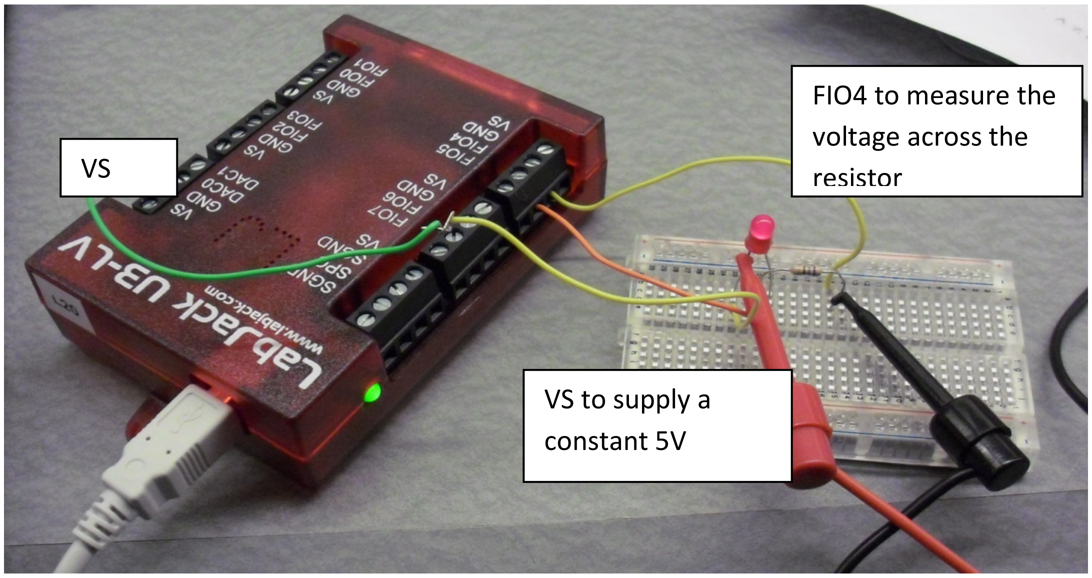
In this exercise, we will supply a constant 5V with the VS channel on the LabJack and measure the voltage across the resistor R1.

We use FIO4 as an analog input to measure the voltage with reference to the ground.

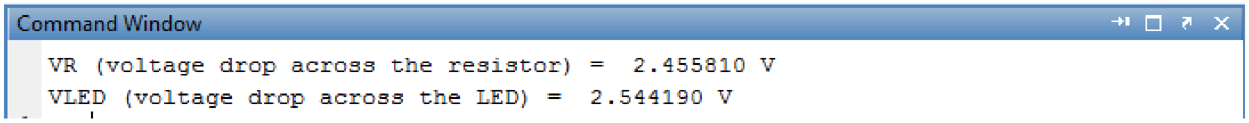
► Construct Circuit 1 on the protoboard.



Note: You may also use a 330 Ω resistor.

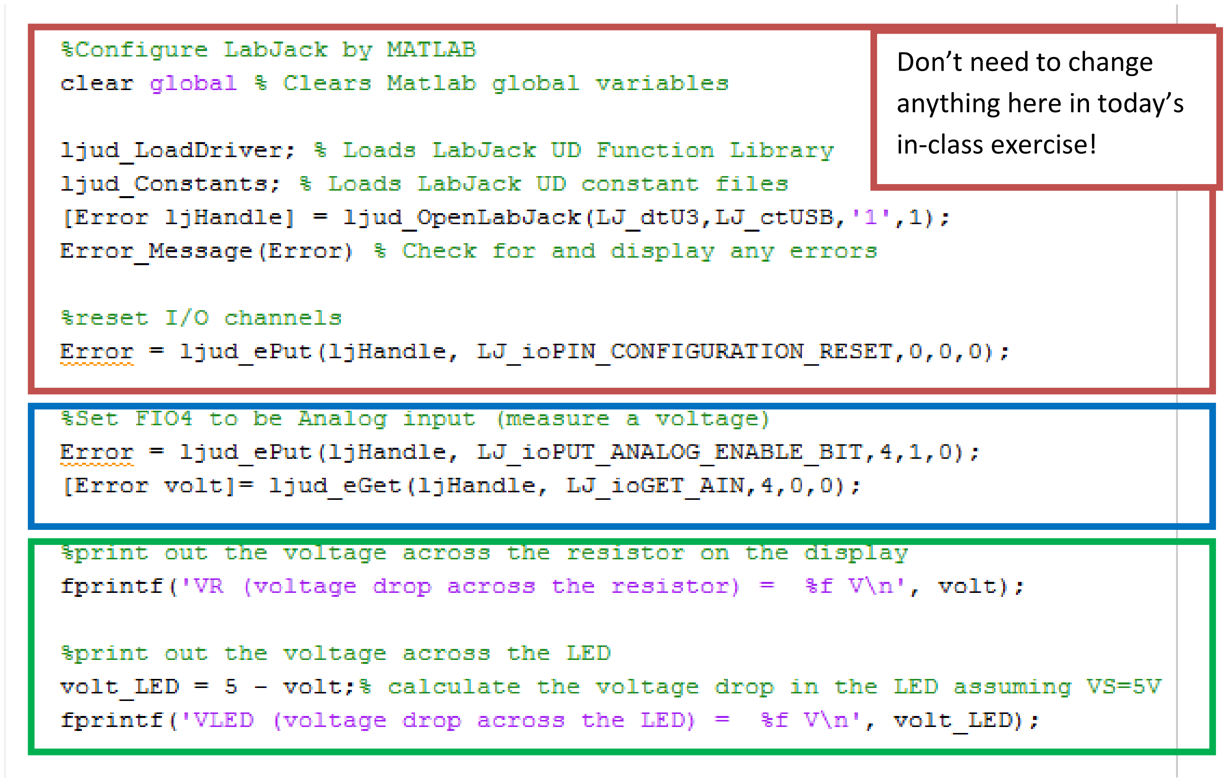

VS
FIO4 to measure the voltage across the resistor
VS to supply a constant 5V

► Download the file "ECE102_Script1.m" from the course website.
After you run the script, you will see the output in the MATLAB command window.



```
VR (voltage drop across the resistor) =  2.455810 V
VLED (voltage drop across the LED) =  2.544190 V
```

There are mainly 3 parts in this code: (1) Initiate the LabJack, (2) Set FIO4 as an Analog input (3) Print out readings using "fprintf" function.



```
%Configure LabJack by MATLAB
clear global % Clears Matlab global variables

ljud_LoadDriver; % Loads LabJack UD Function Library
ljud_Constants; % Loads LabJack UD constant files
[Error ljHandle] = ljud_OpenLabJack(LJ_dtU3,LJ_ctUSB,'1',1);
Error_Message(Error) % Check for and display any errors

%reset I/O channels
Error = ljud_ePut(ljHandle, LJ_ioPIN_CONFIGURATION_RESET,0,0,0);
```

Don't need to change anything here in today's in-class exercise!

```
%Set FIO4 to be Analog input (measure a voltage)
Error = ljud_ePut(ljHandle, LJ_ioPUT_ANALOG_ENABLE_BIT,4,1,0);
[Error volt]= ljud_eGet(ljHandle, LJ_ioGET_AIN,4,0,0);
```

```
%print out the voltage across the resistor on the display
fprintf('VR (voltage drop across the resistor) =  %f V\n', volt);

%print out the voltage across the LED
volt_LED = 5 - volt;% calculate the voltage drop in the LED assuming VS=5V
fprintf('VLED (voltage drop across the LED) =  %f V\n', volt_LED);
```

First of all, we need to find the LabJack device through the USB and reset all I/O channels for use, which is the first part of the MATLAB code. The second part enclosed in blue on the picture above is to configure the FIO4 to act like an analog input and take the voltage measurement one time. The voltage value is assigned into the "volt." The third part in green is to print out the result. We can calculate the voltage drop across the LED because we know the supply voltage and the voltage drop across the resistor.

Let's learn some useful MATLAB commands to control LABJACK

1. Analog Output with DAC#	Command Syntax:
	Error = ljud_ePut (ljHandle, LJ_ioPUT_DAC, DAC#, Volt_value, 0)
	
	Example: Generate 2.5V at DAC0 channel
	Error = ljud_ePut (ljHandle, LJ_ioPUT_DAC, 0, 2.5, 0)

2. Analog Input with FIO#	Command Syntax: Error = ljud_ePut (ljHandle, LJ_ioPUT_ANALOG_ENABLE_BIT, FIO#, 1, 0) [Error Value] = ljud_eGet (ljHandle, LJ_ioGET_AIN, FIO#, 0, 0) Example: Read an analog voltage from FIO4 and assign the voltage into "volt" Error = ljud_ePut (ljHandle, LJ_ioPUT_ANALOG_ENABLE_BIT, 4, 1, 0) [Error volt] = ljud_eGet (ljHandle, LJ_ioGET_AIN, 4, 0, 0) *After this command, "volt" holds the voltage measured by the FIO4 channel

3. Digital Output with FIO#	Command Syntax:
	Error = ljud_ePut (ljHandle, LJ_ioPUT_DIGITAL_BIT, FIO#, State, 0)
	
	Example: Output logic 'High' at FIO5
	Error = ljud_ePut (ljHandle, LJ_ioPUT_DIGITAL_BIT, 5, 1, 0)

4. Digital Input with FIO#	Command Syntax: Error = ljud_ePut (ljHandle, LJ_ioPUT_ANALOG_ENABLE_BIT, FIO#, 0, 0) [Error Value] = ljud_eGet (ljHandle, LJ_ioGET_AIN, FIO#, 0, 0) Example: Read an digital voltage from FIO6 and return 0 (Low) or 1 (High) with "volt" Error = ljud_ePut (ljHandle, LJ_ioPUT_ANALOG_ENABLE_BIT, 6, 0, 0) [Error volt] = ljud_eGet (ljHandle, LJ_ioGET_AIN, 6, 0, 0) *After this command, "volt" holds the 0 (=Logic Low) or 1 (=Logic High) measured by the FIO6 channel

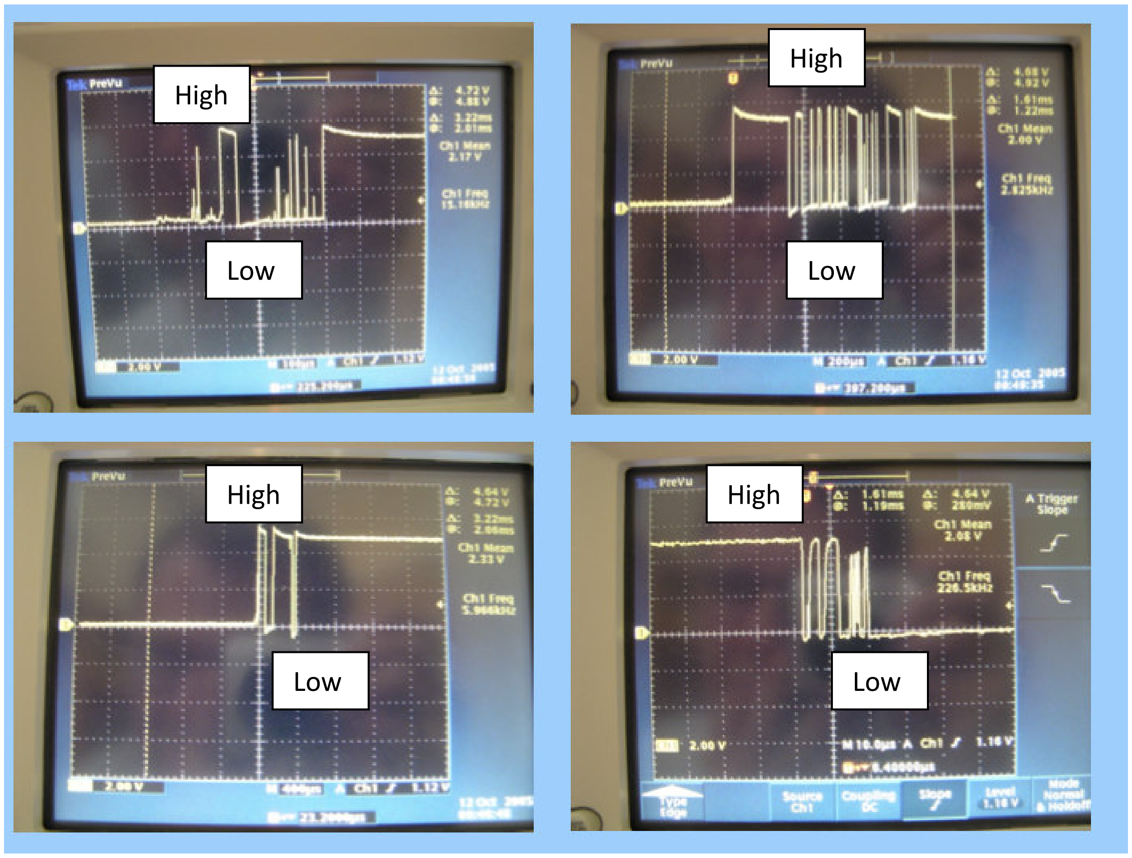
These LabJack commands are enough to complete all the questions in the worksheet. If you want to learn these commands in more detail, download the file "Accessing_LabJack_MATLAB.pdf".

To complete Part 1, please answer Question 1 in the separate worksheet.
That's all for Part 1.

Part 2. Design a Counter with a Switch

In this part, we will use a switch to make an event that will wake up the LabJack to take an action. As an exercise, we will design a counter that counts how many times the switch is pressed.

In general, when we press the switch, we want to count it as one event. However, in reality, pressing the switch initially causes bouncing between logic High and Low. Eventually, the signal settles down to a stable level. The LabJack is fast enough see all the transitions, and it acts on multiple transitions instead of one transition. The picture below shows the switch bouncing waveforms captured by an oscilloscope.



This is a big problem for a counter because the analog input treats this single button push as multiple events due to the bounce between High and Low. So, when you count the events of this switch, it will be very likely to miscount the numbers. This bounce period is typically 1m to 100m sec, depending on the switch and how you press the switch. One approach to "de-bouncing" the switch is to ask the LabJack to wait for 1 sec until all bounces are cleared out after taking the first voltage measurement.

MATLAB has a built-in function "pause." This function stops execution for a specific time. **pause (time_in_seconds)** will wait for a specific time period in seconds before executing the next instruction.

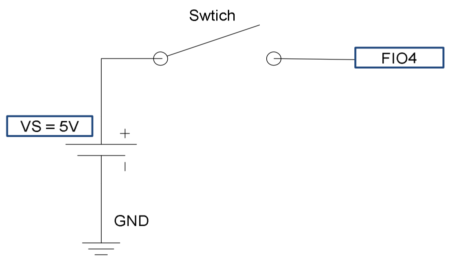
For example, pause(5) causes MATLAB to wait for 5 seconds before performing the next instruction.

We will use this “pause” function to avoid the switch bouncing.

Exercise 2: Design of a Counter: count how many times the switch is pressed

In this exercise, we will design a counter with the LabJack using a MATLAB script. Every time the switch is pressed, the LabJack counts the switch event.

Circuit Schematic 2

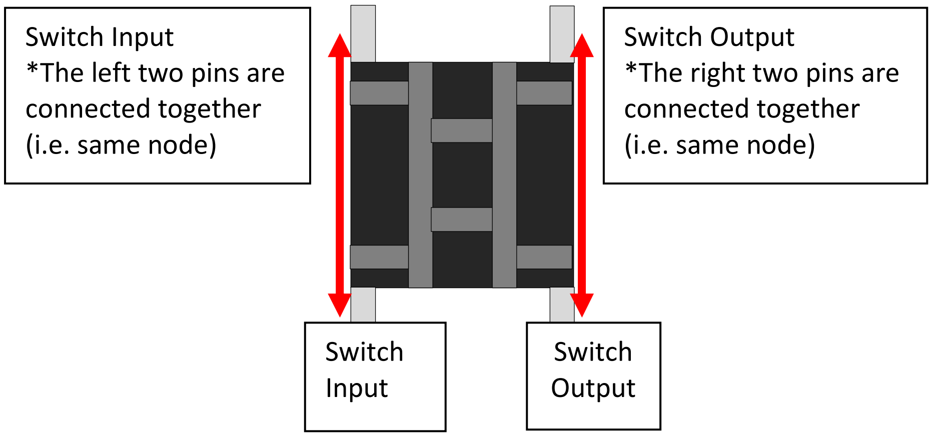


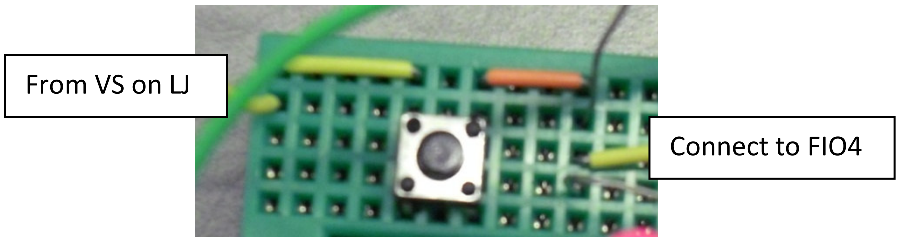
How does it works?

- The VS channel on the LabJack supplies a constant 5V to the input of the switch
- When the switch is not pressed (open), there is no current flowing into the FIO4. So, no event happens.
- When the switch is pressed (short), there is current flowing into the FIO4 from the VS. So, this event counts as a single event.
- To avoid switch bouncing, we will force the LabJack to wait for 1 sec after seeing a voltage of more than 2V at FIO4 (🡐 This value happened to be a nice threshold voltage not to see debouncing from our switch).

► Construct the switch circuit on the protoboard.

The back side view of the switch is shown in the diagram below.


Switch Input
*The left two pins are connected together
(i.e. same node)
Switch Output
*The right two pins are connected together
(i.e. same node)
Switch Input
Switch Output

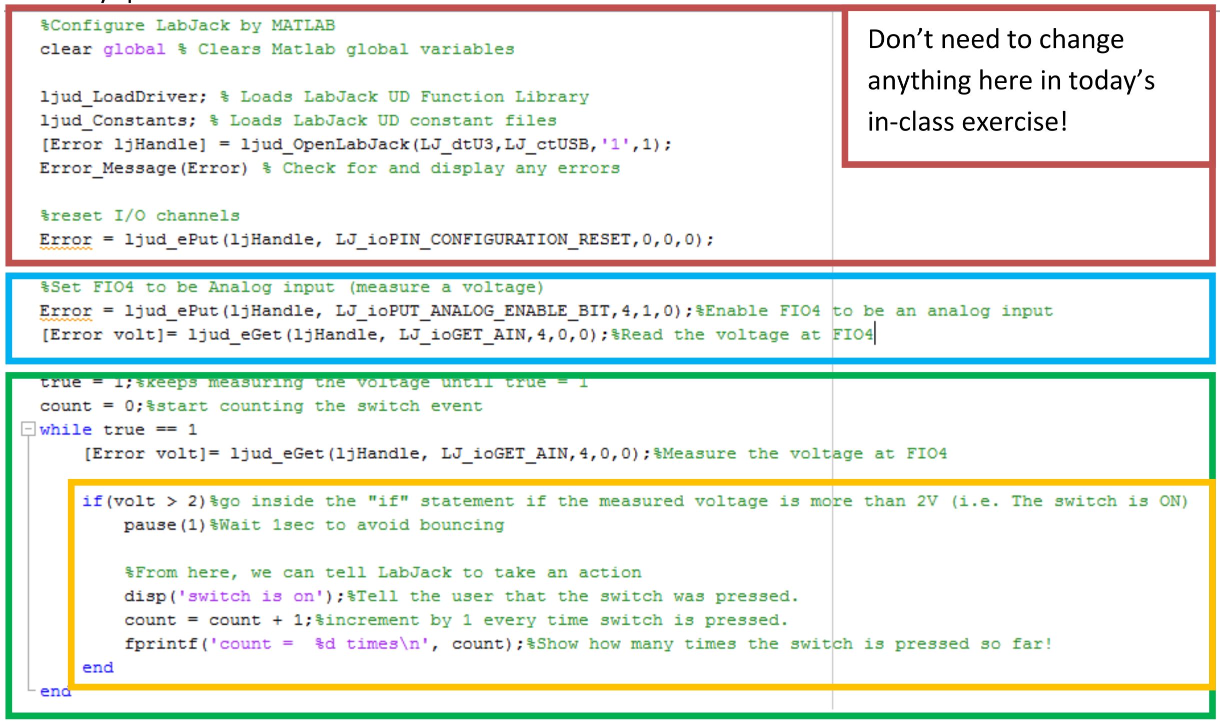


Be careful with placing the switch on the breadboard. There are four legs on the switch, and each two of the legs are connected together (short). Look at the back side of your switch and place the switch so that you will have open connection when it's not pressed between the switch input and output as shown in the picture above. When the switch is pressed, the switch becomes just a short connection between the switch input and output.

MATLAB Code

► Download the file "ECE102_ Script2.m" from the course website.

There are 4 parts in this code. (1) Initiate the LabJack, (2) Enable FIO4 to read a voltage, (3) Keep measuring the voltage with a loop, and (4) When the switch is pressed, take an action. In this case, count the event. Read through the code and try to understand what is going on. Ask the professor or TA if you have any questions.



```
%Configure LabJack by MATLAB
clear global % Clears Matlab global variables

ljud_LoadDriver; % Loads LabJack UD Function Library
ljud_Constants; % Loads LabJack UD constant files
[Error ljHandle] = ljud_OpenLabJack(LJ_dtU3,LJ_ctUSB,'1',1);
Error_Message(Error) % Check for and display any errors

%reset I/O channels
Error = ljud_ePut(ljHandle, LJ_ioPIN_CONFIGURATION_RESET,0,0,0);

%Set FIO4 to be Analog input (measure a voltage)
Error = ljud_ePut(ljHandle, LJ_ioPUT_ANALOG_ENABLE_BIT,4,1,0);%Enable FIO4 to be an analog input
[Error volt]= ljud_eGet(ljHandle, LJ_ioGET_AIN,4,0,0);%Read the voltage at FIO4

true = 1;%keeps measuring the voltage until true = 1
count = 0;%start counting the switch event
while true == 1
    [Error volt]= ljud_eGet(ljHandle, LJ_ioGET_AIN,4,0,0);%Measure the voltage at FIO4

    if(volt > 2)%go inside the "if" statement if the measured voltage is more than 2V (i.e. The switch is ON)
        pause(1)%Wait 1sec to avoid bouncing

        %From here, we can tell LabJack to take an action
        disp('switch is on');%Tell the user that the switch was pressed.
        count = count + 1;%increment by 1 every time switch is pressed.
        fprintf('count =  %d times\n', count);%Show how many times the switch is pressed so far!
    end
end
```

Don't need to change anything here in today's in-class exercise!

In this case, we use a "while" loop to keep measuring the voltage at FIO4. As long as the variable named "true" is equal to 1, it continues to measure the voltage. If FIO4 sees a voltage greater than 2V, the "pause(1)" causes a 1 second wait in order to avoid the switch bouncing that we discussed earlier.

After 1 second has elapsed, you can tell what you want the LabJack to do within the “if” statement. For example, if you want to quit the voltage measurement, you can set “**true = 0;**” to get out of the loop. Likewise, you can tell the LabJack to generate a voltage with DAC0 with **Error = ljud_ePut (ljHandle, LJ_ioPUT_DAC, DAC#, Volt_value, 0)**. This is helpful for Question 2 and Question 3 in the worksheet.

Please answer Question 2 and 3 in the separate worksheet to complete Part 2.

That’s all for today’s in-class exercise.
Submit the separate worksheet document with your answers by the end of the class.