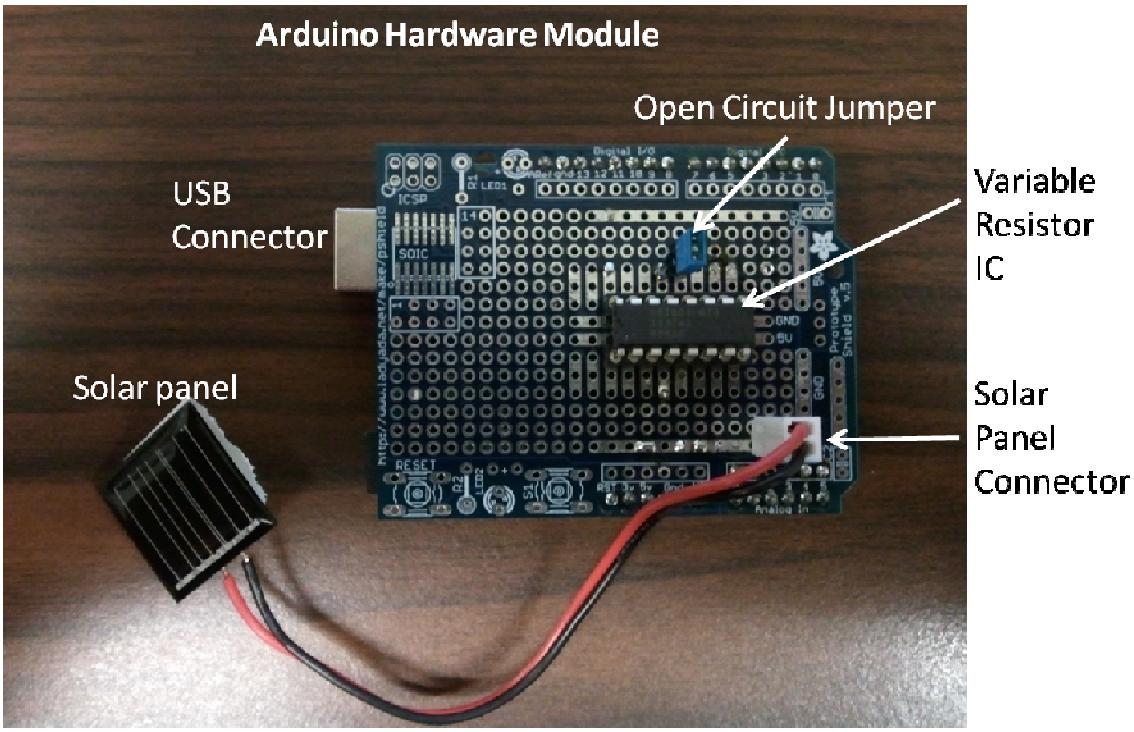
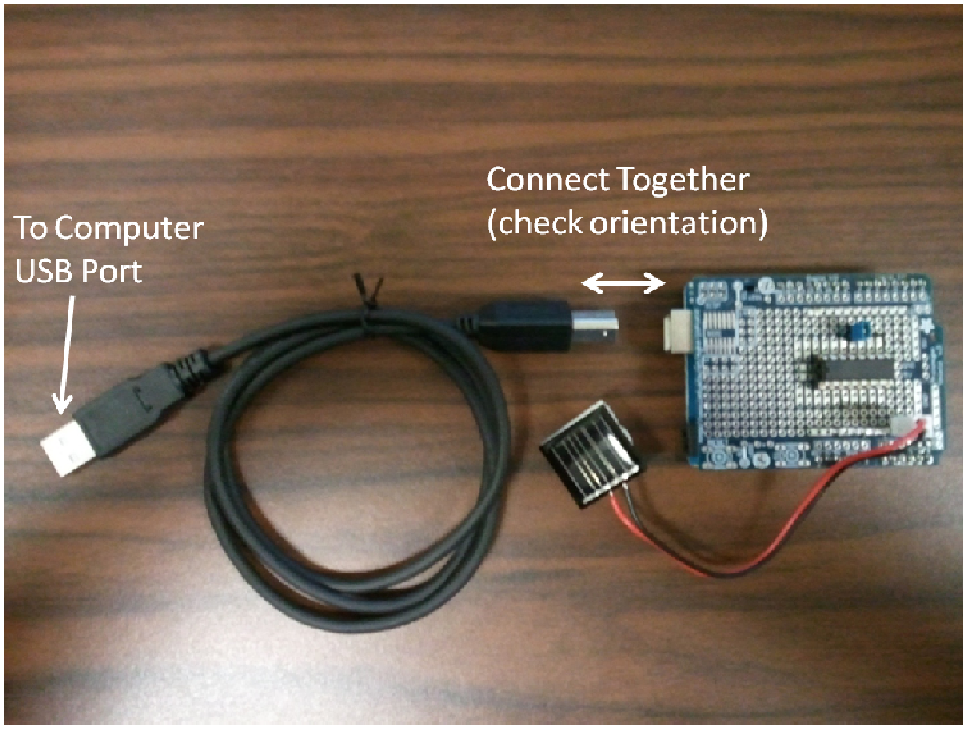
Lab: Solar Cell Characterization

In this lab, you will first use the Arduino Hardware Module to gather some solar cell data, then analyze the gathered data.

Below is a picture of the hardware setup with important parts labeled.





Lab: Solar Cell Characterization

Follow steps below to get started with the hardware.

<u>If you are using the lab's computer, start from Step 4.</u>

Step 1: Download and unzip 'ArduinoIO.zip' and 'arduino-1.0-windows.zip' onto anywhere on your computer. Recommended locations include C:/ (the C drive), your home directory, or the "My Documents" folder.

Step 1.5: Inside the folder "ArduinoIO", double click on the file "install_arduino.m". At this point, MATLAB will start and the installation script will run.

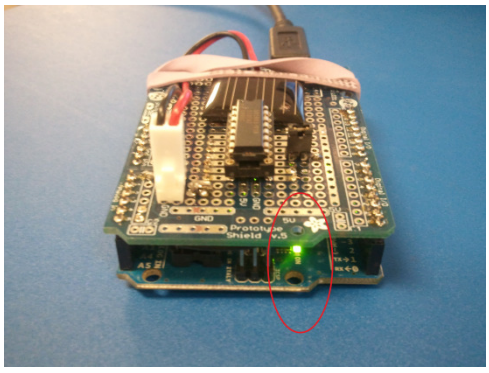
Step 2: Connect the solar panel onto the Arduino Hardware Module. Watch out for the orientation of the connector. You should hear a 'click' when the connector is fully inserted into the receptacle.

Step 3: Make sure the Open Circuit Jumper is inserted in place.

Step 4: Connect the USB cable to the Arduino Hardware Module. Watch out for the orientation of the connector.

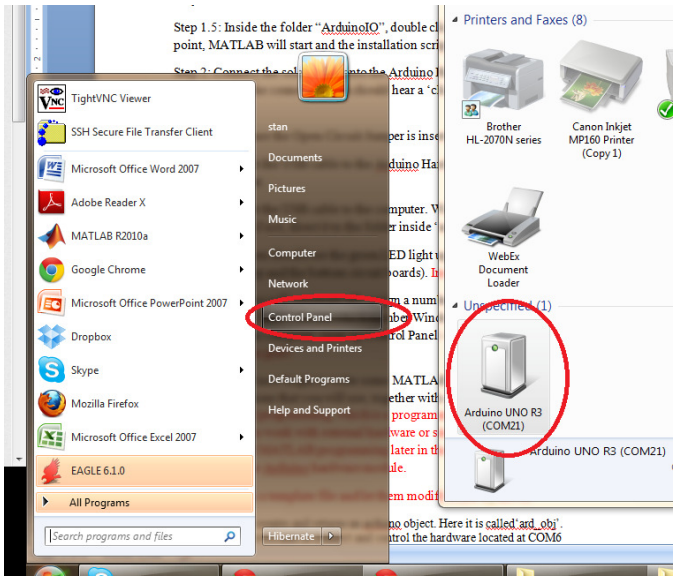
Step 5: Connect the USB cable to the computer. Windows should find the USB driver automatically, if not, direct it to the folder inside 'arduino-1.0-windows.zip'.

At this point, you should see the green LED light up (See figure below) inside the Arduino Hardware Module (between the top and the bottom circuit boards).

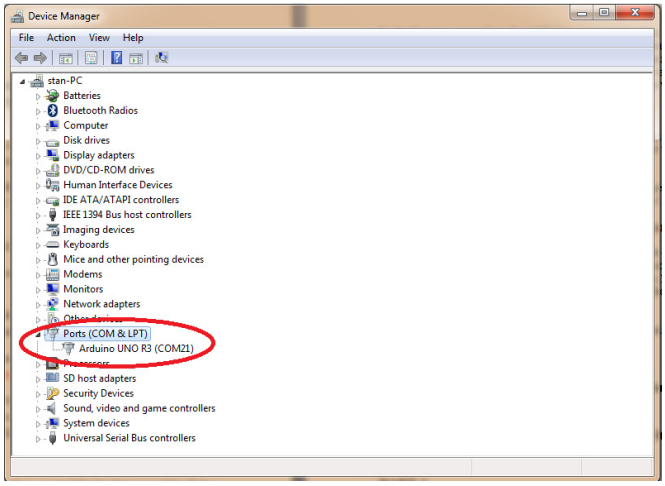


Step 6: Windows will automatically assign a number to the hardware module. For example: "COM2" or "COM9". To see which number Windows assigned to the Arduino Hardware Module on your computer, open up Control Panel -> Devices and Printers. See red circle in image below.

Lab: Solar Cell Characterization



In Windows XP, go to the Device Manager, and it will be listed under Ports. See red circle in figure below. (In the figure, Arduino device is at COM21)



Step 7: We will now begin to write some MATLAB code to control the hardware module. Below are the four functions that you will use in this lab, together with explanations.

Note that this code relies on object-oriented programming, which is a programming technique which makes it easy to write modular code to work with external hardware or simulate different systems. You will cover object-oriented MATLAB programming later in the term but we will use objects in this lab to interact with the Arduino hardware module.

Lab: Solar Cell Characterization

```
% This function creates and returns an arduino object. Here it is called‘ard_obj’.
% This object allows user to connect and control the hardware located at COM6
ard_obj = arduino(‘COM6’);

% This function reads the voltage on an analog input pin. The value returned by
%  the function ranges from 0 to 1023 that corresponds to 0 to 5 Volts on the analog pin.
%  Example: if voltage on analog input pin is 0 Volts, the value returned by the function is 0.
%                 if voltage on analog input pin is 5 Volts, the value returned by the function is 1023.
%                 if voltage on analog input pin is 2.5 Volts, the value returned by the function is 511
%  In general, to convert from the function’s return value to a voltage, use the formula:
%  Voltage =  return_value * 5 /1024
solar_cell_value = ard_obj.analogRead(0);

%  There are two variable resistors (each can vary from 60 ohms to 10 Kohms) on the hardware module.
% Use ‘pWrite(res_num, res_code)’ to set the resistance for either variable resistors.
%
% Arguments:
% ‘res_num’  -- The resistor you wish to change the resistance code, ‘res_code’. Can be either 0, 1, or -1.
%  0 or 1 sets the resistance for variable resistor number 0 or 1.
%  -1 sets the resistance for BOTH variable resistors.
% ‘res_val’ --  The resistance code you would like for ‘res_num’. Range from 0 to 255.
%  For example:
%  pWrite(1,0) sets variable resistor No.1 to a resistance code of 0
%  pWrite(0,255) sets variable resistor No.0 to a resistance code of 255
%  pWrite(-1,128) sets BOTH variable resistors No.0 and No.1 to a resistance code of 128
% To translate from resistance code to resistance value, use the formula: Resistance = 60 + res_code*43
ard_obj.pWrite(res_num, res_code);

% delets the arduino object. Call this function when script ends.
delete(ard_obj);
```

Important:

1) **Do NOT run the code you write for any tasks below. You will be explicitly instructed when to run your code.**
2) **All tasks below will build into a single script. It is important you complete the tasks in order.**

Lab: Solar Cell Characterization

Task 1: Open a new script and name it "acquire.m"Write code to create an Arduino object that connects and controls the hardware module at the appropriate COM port of your computer.

Task 2: Write code to set the variable resistor number 0 to a resistance code of 128, and insert a delay of 0.1 second immediately after this line of code by using "pause(0.1)".

Task 3: Write code to set the variable resistor number 1 to a resistance code of 255, and insert a delay of 0.1 second immediately after this line of code by using "pause(0.1)".

Task 4: Write a loop to acquire 256 data points from analog input pin number 0 by completing the four steps below.

1) Obtain a data point by calling the 'analogRead()' function,

2) Convert the data point into a voltage quantity using the formula below: ('code' contains the value returned by 'analogRead()')

Voltage = code * 5 / 1024;

3) Store each voltage into a vector called 'solar_cell_dat', and insert a delay of 0.1 second after this line of code.

4) Inside the loop, print a message, saying "Data acquisition in progress…" using disp().

Task 5: Now that you have a vector containing 256 data points, write a line of code to plot the 256 data point vector using 'plot()'.

Task 5.5: Write code to label the plot using 'xlabel()', 'ylabel()', and title()'. X-axis is the "Sample number", y-axis is the "Solar cell output voltage", and the title is "Solar cell output voltage sampled 256 times".

Task 6: Write code to delete the arduino object.

Task 7: The script is complete. Make sure the solar cell attached to the hardware module is facing up, towards the ceiling. **Run the code**. Save the plot as "light1.png".

Lab: Solar Cell Characterization

Change the name of 'solar_cell_dat' to 'light1'. Save 'light1' as "light1.mat".

Task 8: Turn the solar cell 90 degrees onto its side by turning the hardware module. **Re-run the code.** Save the plot as "light2.png".

Change the name of 'solar_cell_dat' to 'light2'. Save 'light2' as "light2.mat".

Compare this plot to the plot from Task 7. Depending on what the solar cell is facing, the y-values here should be higher/lower due to the increased/decreased number of photons incident on the solar cell.

Task 9: Take a flashlight (or download and open a flashlight app on your smartphone), turn it on, and point it directly at the solar cell at a distance about 3 inches. **Re-run the code.** Save the plot as "light3.png".

Change the name of 'solar_cell_dat' to 'light3'. Save 'light3' as "light3.mat".

Compare this plot to the plots from previous two tasks. The y-values here should be much higher due to the increased illumination.

Task 10: Now open a new MATLAB script. Name this script "process.m" Write code to load and compute the average and the variance of the three datasets, "light1.mat", "light2.mat", and "light3.mat", using 'mean()' and 'var(dataset,1)'. Print out the average and variance for the three datasets using fprintf().

Task 11: Zip the following five files into a single ZIP file: "light1.mat", "light2.mat", "light3. mat", "acquire.m", and "process.m". Do not submit the plots.

Name the ZIP file "LQ7_XXX.zip" where XXX are your initials. Submit this ZIP file onto SmartSite.

Lab: Solar Cell Characterization

PART 2

Task 1: Open a new MATLAB script. Write code to create an arduino object that connects and controls the hardware module at the appropriate COM port of your computer.

Task 2: Write code that sets both variable resistors to a resistance value of 255, and insert a delay of 0.1 second immediately after the code by using "pause(0.1)".

Task 3: Write a loop to acquire 256 data points from analog input pin number 0. At each loop iteration, write code to perform the following:

1) Store each data point into a vector called 'solar_cell_dat'.
2) Increment (by one) the resistance code of resistor 0 (resistance code for resistor 1 will remain fixed). So at the first loop iteration, the resistance code is 0, and at the last iteration, the resistance code is 255. Insert a delay of 0.1 second immediately following this line of code by using "pause(0.1)".

Task 4: Now that you have a vector that contains 256 data points, write a line of code to plot the 256 data point vector using 'plot()'.

Task 5: Write code to delete the arduino object.

Task 6: The script is complete. Take a flash light (or download and open a flashlight app on your smartphone), turn it on, and point it directly at the solar cell at a distance about 2 inches. **Run the code.** Save the plot. Save the vector 'solar_cell_dat' which contains the 256 data points as "sweep1.mat".

Task 7: Move the flash light about one inch farther from its current position. **Re-run the code.** Save the plot. Save the vector 'solar_cell_dat' which contains the 256 data points as "sweep2.mat".

Compare this plot to the plot from the previous task. The y-values here should be lower due to the decreased illumination.

Task 8: Move the flash light another inch farther from its current position. **Re-run the code.** Save the plot. Save the vector 'solar_cell_dat' which contains the 256 data points as "sweep3.mat".

Compare this plot to the plots from the previous two tasks. The y-values here should be lower due to the decreased illumination.

Task 9: For this part (part 2), you will submit four files onto SmartSite. The files are: “sweep1.mat”, “sweep2.mat”, “sweep3.mat”, and the MATLAB code for this part. You will NOT submit the plots.