LAB # 6:

Current Sensor – Battery Monitoring

Maria E. Meza-Bruguera
Anh Nguyen

Microprocessor Based System
EE 444
Spring 2013
Prof: Gary Hill

Table of Contents

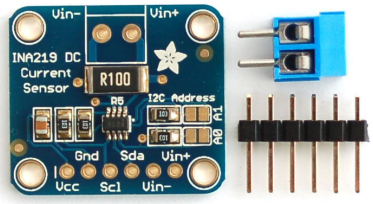
Objective

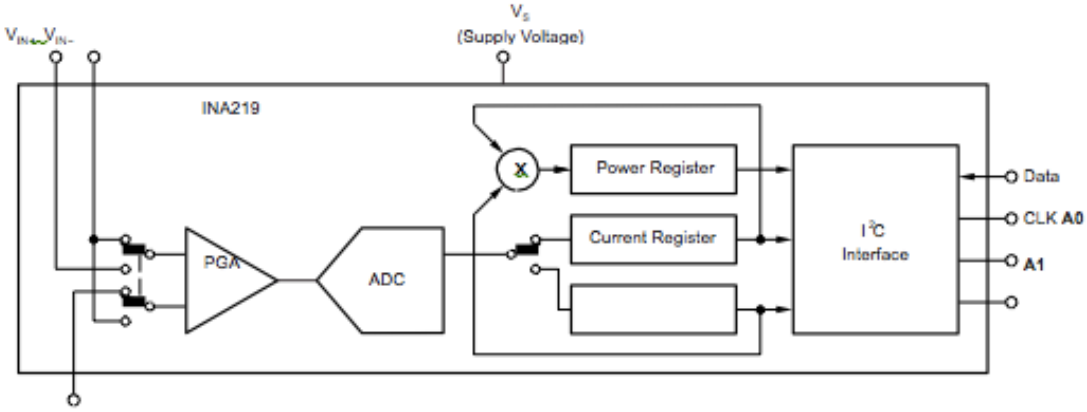
The purpose of this lab is to measure the current of the Arduino and plot the data using Matlab GUI. The current data will let us approximate when the 7.2V battery that powers the Arduino will run out.

Reference Material

2.1 Description

The current sensor that we use in this lab is the Adafruit INA219. It can measure both high side voltage and DC current draw over I2C with 1% precision. It can handle high side current measuring, up to +26VDC, even though it is powered with 3 or 5V. Inside the current sensor, a precision amplifier measures the voltage across the 0.1ohm, 1% sense resistor. Since the amplifier maximum input difference is ±320mV this means it can measure up to ±3.2 Amps. With the internal 12 bits ADC, the resolution at ±3.2A range is 0.8mA. With the internal gain set at the minimum of div8, the max current is ±400mA and the resolution is 0.1mA.





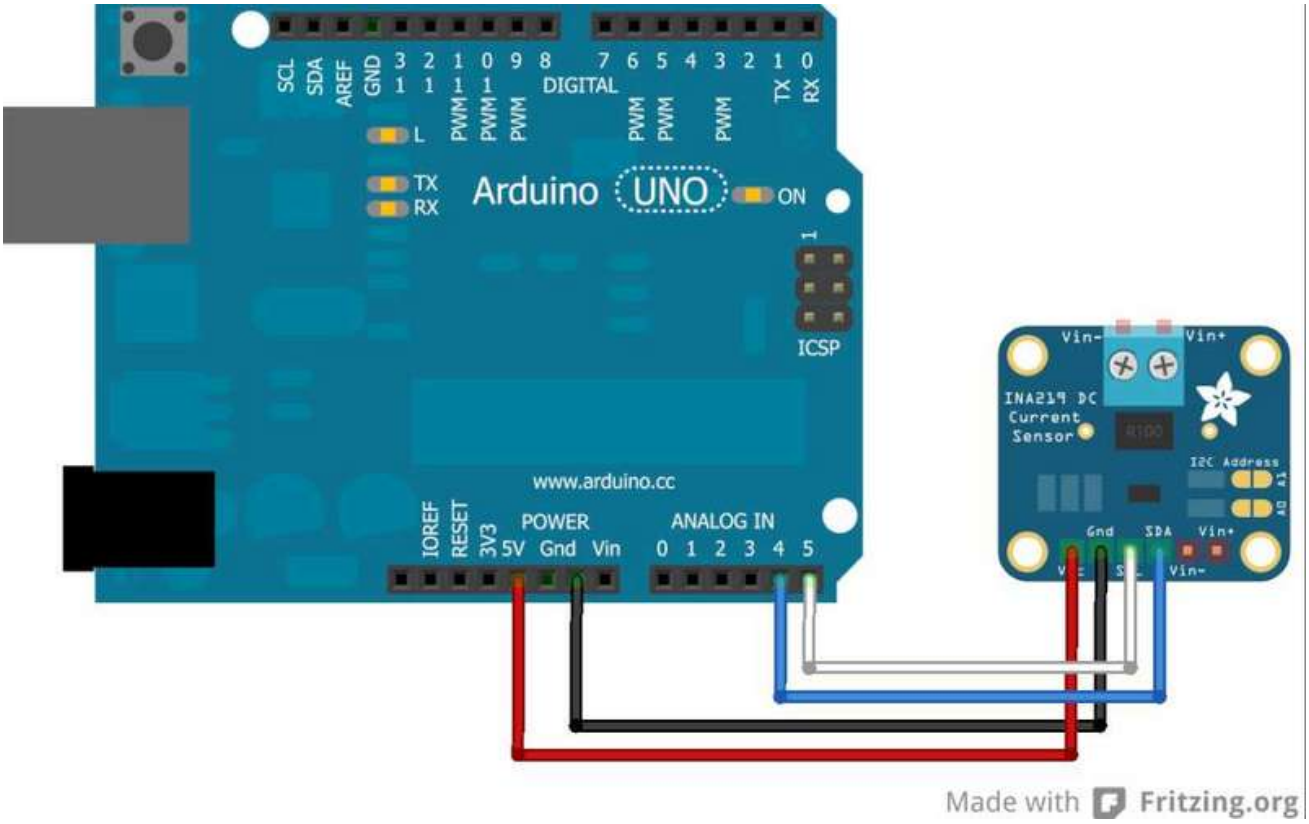
The link below is the data sheet for INA219 current sensors:

http://www.adafruit.com/datasheets/ina219.pdf

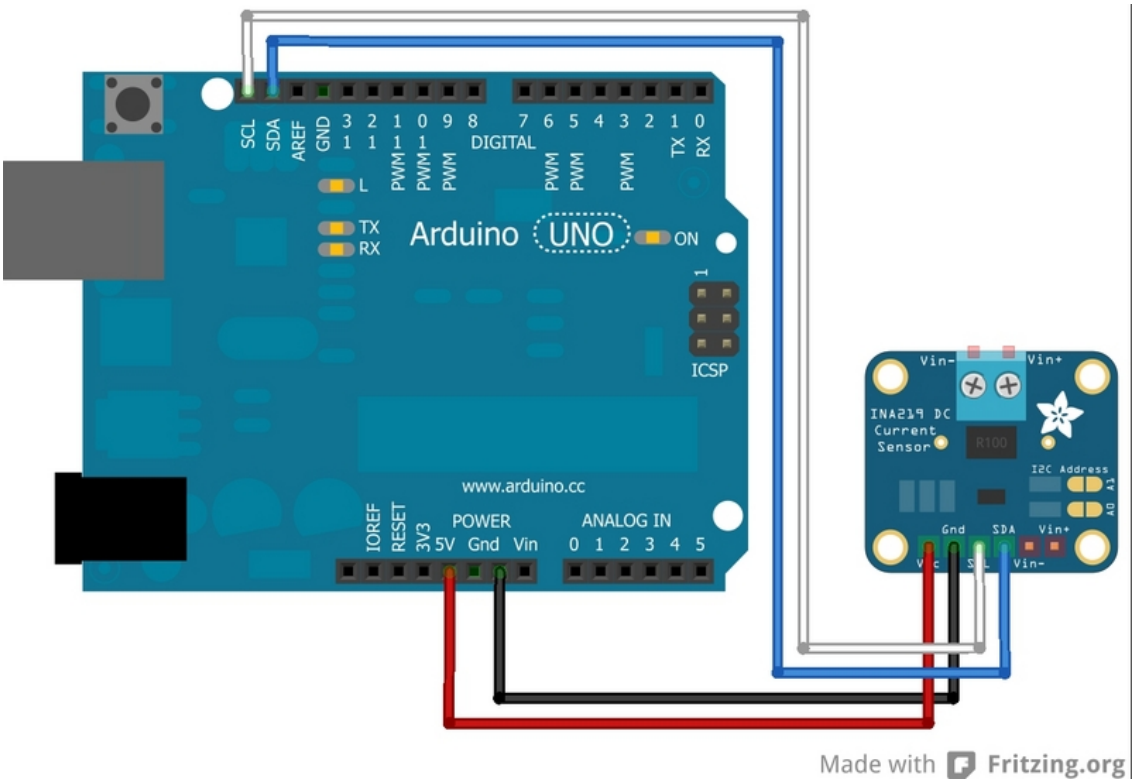
2.2 Current Sensor Wiring

The INA219 current sensor can be powered by the 5V pin on the Arduino and communicate via I2C.

- Connect GND to GND
- Connect VCC to 5V
- Connect SDA to SDA (analog pin 4 on pre- R3 Arduino)
- Connect SLC to SLC (analog pin 5 on pre-R3 Arduino)



On Arduino Due we can connect to the new SDA and SCL pins next to AREF pin.



The Experiment

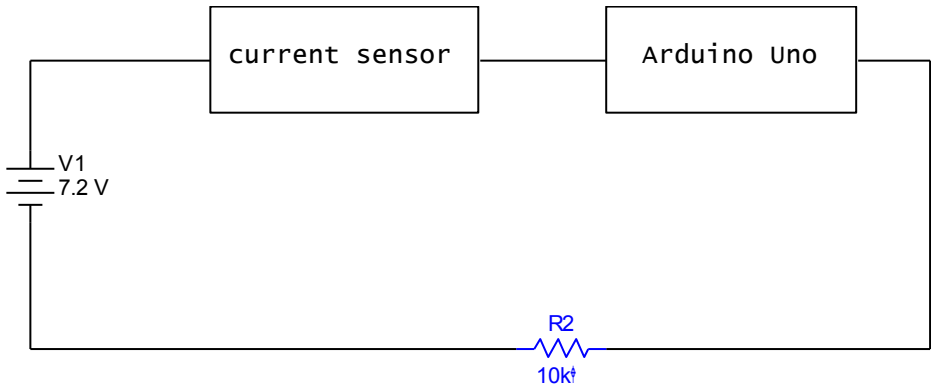
How to connect the current sensor to the circuit:

- Connect V+ to the positive terminal of the power supply for the circuit.

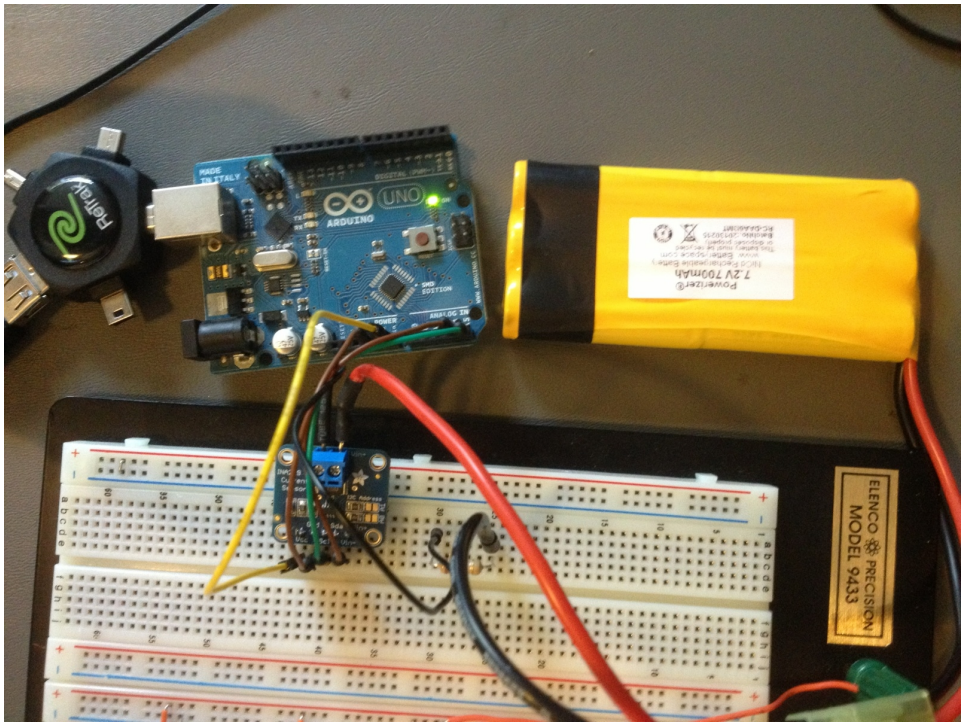
- Connect the V- to the positive terminal or lead of the load. This puts the sensor resistor in line with the circuit.

We will build a voltage divider circuit, which allows us to measure the current going through the Arduino and from those data we can know when the 7.2V battery will run out. We will use 7.2V battery to power the circuit and 2 resistors so that the voltage between those two resistors is 3.3V

Schematic:



Circuit:



Arduino Programming

4.1 Installing the Library

-Download the Library from the link below:

http://learn.adafruit.com/adafruit-ina219-current-sensor-breakout/downloads

- Expand the .zip file in the Libraries folder in the Arduino sketchbook folder. (To find the sketchbook folder, open File → Preferences in the IDE and it will show the location of the Sketchbook folder.)

- Rename the folder to Adafruit_INA219.

- Close all instances of the IDE, then re-open one, so that the IDE will recognizes the new library.

We then can use this code **to test the current sensor**.

Select "File → Examples → Adafruit_INA219 → getcurrent" to open the code for testing the current sensor. There should be four values being printed in the serial monitor; **make sure to close before running Matlab script.**

```
#include <Wire.h>
#include <Adafruit_INA219.h>

Adafruit_INA219 ina219;

void setup(void)
{
  uint32_t currentFrequency;

  Serial.begin(115200);
  Serial.println("Hello!");

  Serial.println("Measuring voltage and current with INA219 ...");
  ina219.begin();
}

void loop(void)
{
  float shuntvoltage = 0;
  float busvoltage = 0;
  float current_mA = 0;
  float loadvoltage = 0;

  shuntvoltage = ina219.getShuntVoltage_mV();
  busvoltage = ina219.getBusVoltage_V();
  current_mA = ina219.getCurrent_mA();
  loadvoltage = busvoltage + (shuntvoltage / 1000);

  Serial.print("Bus Voltage:   "); Serial.print(busvoltage); Serial.println(" V");
  Serial.print("Shunt Voltage: "); Serial.print(shuntvoltage); Serial.println("
mV");
  Serial.print("Load Voltage:  "); Serial.print(loadvoltage); Serial.println("
V");
  Serial.print("Current:       "); Serial.print(current_mA); Serial.println("
mA");
  Serial.println("");

  delay(2000);
}
```

4.2 New Arduino Code with Modification

To run the current sensor and get the data in the Arduino, we will have some modification of the code. We will create a Serial Event while loop to display shunt voltage, bus voltage, current and

total voltage (sum of shunt voltage and bus voltage), **make sure to close before running Matlab script.**

```
#include <Wire.h>
#include <Adafruit_INA219.h>

Adafruit_INA219 ina219;

void setup(void)
{  Serial.begin(9600);

  ina219.begin();
}

void loop(void)
{
}

void serialEvent() {

  while (Serial.available()) {
    char inputByte = Serial.read();

    switch (inputByte) {
    case 'S':  // S command returns shunt voltage
      Serial.println(ina219.getShuntVoltage_mV());
      break;
    case 'B': // B command returns bus voltage
      Serial.println(ina219.getBusVoltage_V());
      break;
    case 'C':
      Serial.println(ina219.getCurrent_mA());
      break;
    case 'L':
      Serial.println((ina219.getBusVoltage_V() + (ina219.getShuntVoltage_mV()
/ 1000)));
      break;
    }
    delay(1000);
  }
}
```

Matlab Programming

5.1 Matlab Coding and GUI

A GUI (Graphical User Interface) is a way to graph your program data. It also makes it easier to adjust parameters and visualize your program. The GUI allows you to create functions that will be called upon pressing the button(s) within the interface. In our case the functions call and receive the data directly from the serial port so there is no need for a separate Matlab script. A figure must also accompany the script, the file can be found inside the Lab folder in our dropbox account

For more instruction of building a GUI:

http://www.math.ucla.edu/~wittman/reu2008/matlabGUI.pdf

http://www.mathworks.com/videos/creating-a-gui-with-guide-68979.html

5.2 Matlab GUI Script

```
%============================================================%
% Author:                                                    %
% Brian Kiernan                                              %
% Application Engineer                                       %
% The MathWorks, Inc.                                        %
% August 27, 2004                                            %
%============================================================%
%============================================================%
% Edited to monitor battery stats by:                        %
% Maria Elena Meza-Bruguera                                  %
% April 10th, 2013                                           %
%============================================================%

% End initialization code - DO NOT EDIT
function varargout = RTviewer(varargin)
% RTVIEWER GUI to monitor the prices of stocks using the Datafeed Toolbox
%
% This demo shows how TIMER objects can be used to create a "real-time"
% trading application with MATLAB.
%
% Using the Datafeed Toolbox (Bloomberg or Yahoo) to import live market
% data or MATLAB's random number generation, this tool plots the data as it
% is imported or created, and displays whether the last price is higher or
% lower than the previous price.  If the price is higher, the point on the
% graph is plotted in green.  If it is lower, it is plotted as red.
% Otherwise it is plotted blue.
%
% This program could be used as a starting point to creating more detailed
% real-time trading applications.
%
% Author:
% Brian Kiernan
% Application Engineer
% The MathWorks, Inc.
% August 27, 2004
```

```
% Edit the above text to modify the response to help RTviewer

% Last Modified by GUIDE v2.5 19-Apr-2013 13:42:44

% Begin initialization code - DO NOT EDIT
gui_Singletona = 1;
gui_State = struct('gui_Name',       mfilename, ...
'gui_Singleton',  gui_Singleton, ...
'gui_OpeningFcn', @RTviewer_OpeningFcn, ...
'gui_OutputFcn',  @RTviewer_OutputFcn, ...
'gui_LayoutFcn',  [] , ...
'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end


% --- Executes on button press in tglRun.
function tglRun_Callback(hObject, eventdata, handles)
% Start and stop collecting data

% Get booling start of stopping value of toggle button
stopgo = get(hObject,'Value');

% Get period and ticker
period = get(handles.sldPeriod,'value');

% Find all timers
alltimers = timerfind;

% Start and stop the timer, but only if it exists
if ~isempty(alltimers) && isfield(handles,'t')
% Start Viewing Data
if stopgo
        set(hObject,'string','Stop')
        set(handles.t,'Period',period)
        start(handles.t)
        set(hObject,'BackGroundColor','g')
        set(handles.edtTicker,'enable','off')
else
        set(hObject,'string','Run')
        stop(handles.t)
        set(hObject,'BackGroundColor','g')
        set(handles.edtTicker,'enable','on')
end

else

% Timer doesn't exist, this will force the user to
```

```
% create one using the other controls
    set(hObject,'string','Run','BackGroundColor','m', ...
'value',0)
end

%update handles structure
guidata(hObject,handles)


% --- Groups and select data from given options
function edtTicker_Callback(hObject, eventdata, handles)

% Delete timer objects
handles = timerconncleanup(handles);

% Clear Plot and Reset Values
dataplotreset(handles.ticker)

% Fetch Parameters
period = get(handles.sldPeriod,'value');
ticker = upper(get(hObject,'string'));

% Find which data source has been selected
btngrpchil = get(handles.btngrpData,'children');
btngprchil = findobj(btngrpchil,'style','radio');
vals = get(btngrpchil,'value');
datasource = find(cell2mat(get(btngrpchil,'value')));

if datasource == 3;  % Shunt Voltage

% Update Timer Object to use Shunt Voltage with new ticker
    handles.t = createtimer(@shuntRT,ticker,period,handles,handles.Conn);

elseif datasource == 2  % Bus Voltage

% update Timer Object to use Bus Voltage with new Ticker
    handles.t = createtimer(@busRT,ticker,period,handles,handles.Conn);

else% Current

% Update Timer Object with new ticker for the plot
    handles.t = createtimer(@currentRT,period,handles,handles.Conn);

end

% Update handles structure
guidata(hObject,handles)


% --- Executes on button press in radCurrent.
function radCurrent_Callback(hObject, eventdata, handles)

% Turn off existing timer object and delete it
handles = timerconncleanup(handles);
```

```
% Clear Plot and Reset Values
dataplotreset(handles.ticker)

% Fetch Parameters
period = get(handles.sldPeriod,'value');
ticker = upper(get(handles.edtTicker,'string'));
handles.t = createtimer(@currentRT,ticker,period,handles);

guidata(hObject,handles)


% --- Current Real-Time processing function
function currentRT(obj,event,ticker)

arduinoPort = getappdata(0,'ardPath');
arduino = instrfind('Port',arduinoPort);

if isempty(arduino)
    arduino = serial('/dev/tty.usbmodemfa131');
else
    fclose(arduino);
    arduino = arduino(1);
end

fopen(arduino);
pause(2);
fprintf(arduino, 'C');
LastPrice.Last = fscanf(arduino,'%e',2);
TickTime = datenum(event.Data.time);

currentVector = [getappdata(gcf,'currentValue'); LastPrice.Last];
setappdata(gcf,'currentValue',currentVector)
timeVector = [getappdata(gcf,'currentTime'); TickTime];
setappdata(gcf,'currentTime',timeVector)

plotRT(ticker,currentVector,timeVector)


% --- Executes on button press in radShunt.
function radShunt_Callback(hObject, eventdata, handles)
% Shunt Radio Button Callback. Define timer to use Shunt Voltage data

% Turn off existing timer object and delete it
handles = timerconncleanup(handles);

% Clear Plot and Reset Values
dataplotreset(handles.ticker)

% Fetch Parameters
period = get(handles.sldPeriod,'value');
ticker = upper(get(handles.edtTicker,'string'));
handles.t = createtimer(@currentRT,ticker,period,handles);

guidata(hObject,handles)
```

```
% --- Shunt Voltage Real-Time processing function
function shuntRT(obj,event,Conn,ticker)

arduinoPort = getappdata(0,'ardPath');
arduino = instrfind('Port',arduinoPort);

if isempty(arduino)
    arduino = serial('/dev/tty.usbmodemfa131');
else
    fclose(arduino);
    arduino = arduino(1);
end

fopen(arduino);
pause(2);
fprintf(arduino, 'S');
LastPrice.Last = fscanf(arduino,'%e',2);
TickTime = datenum(event.Data.time);

currentVector = [getappdata(gcf,'shuntValue'); LastPrice.Last];
setappdata(gcf,'shuntValue',currentVector)
timeVector = [getappdata(gcf,'shuntTime'); TickTime];
setappdata(gcf,'shuntTime',timeVector)

plotRT(ticker,currentVector,timeVector)


% --- Executes on button press in radBus.
function radBus_Callback(hObject, eventdata, handles)
% Bus Voltage Radio Button Callback. Define timer to use Bus Voltage data

% Turn off existing timer object and delete it
handles = timerconncleanup(handles);

% Clear Plot and Reset Values
dataplotreset(handles.ticker)

% Fetch Parameters
period = get(handles.sldPeriod,'value');
ticker = upper(get(handles.edtTicker,'string'));
handles.t = createtimer(@currentRT,ticker,period,handles);

guidata(hObject,handles)


% --- Bus Voltage Real-Time processing function
function busRT(obj, event, ticker)

arduinoPort = getappdata(0,'ardPath');
arduino = instrfind('Port',arduinoPort);

if isempty(arduino)
    arduino = serial(arduinoPort);
```

```
else
    fclose(arduino);
    arduino = arduino(1);
end

fopen(arduino);
pause(2);
fprintf(arduino, 'B');
LastPrice.Last = fscanf(arduino,'%e',2);
TickTime = datenum(event.Data.time);

currentVector = [getappdata(gcf,'busValue'); LastPrice.Last];
setappdata(gcf,'busValue',currentVector)
timeVector = [getappdata(gcf,'busTime'); TickTime];
setappdata(gcf,'busTime',timeVector)

plotRT(ticker,currentVector,timeVector)


% --- General Plotting function for real-time data
function plotRT(ticker,currentVector,timeVector)

% Get handle to the status text box
circs = getappdata(gcf,'circs');

if length(currentVector) == 1

% Plot first Value
    set(circs,'xdata',timeVector,'ydata',currentVector,'cdata',[0 0 1]);
    setappdata(gcf,'circs',circs);
    axispos = axis;
    set(gca,'xtick',timeVector,'xticklabel',datestr(timeVector,13))

else
% Test for up or down movement
if currentVector(end) > currentVector(end-1)
% Create a green RGB color for the point on the graph
        dotcolor = [0 1 0];

elseif currentVector(end) < currentVector(end-1)
% Create a red RGB color for this point on the graph
        dotcolor = [1 0 0];

else
% Create a blue RGB color for this point on the graph
        dotcolor = [0 0 1];
end

% Fetch current x(time) and y(price) data from the graph
    ctime = get(circs,'xdata');
    cprice = get(circs,'ydata');

% Fetch current colormapping, and update it with the color
% of the next dot, which was defined above.
    colormat = get(circs,'Cdata');
    colormat = [colormat;dotcolor];
```

```
% Update scatter plot with new price point
    set(circs,'xdata',[ctime,timeVector(end)], ...
'ydata',[cprice, currentVector(end)],'Cdata',colormat)

% Find and delete line plot so as not to plot too many lines
    dummyline = findobj(gca,'type','line');
if ~isempty(dummyline)
        delete(dummyline)
end

% Plot Connecting Line
    plot(timeVector,currentVector)

% 5 percent of the data range
    x5per = .05*(max(timeVector) - min(timeVector));

% Calculate the lowest and highest price so far
    lowprice = min(currentVector);
    highprice = max(currentVector);
    Range = highprice - lowprice;
%update axes size for better viewing
if Range == 0
        axispos = axis;
else
        axispos = [timeVector(1)-x5per,timeVector(end)+x5per, ...
        max(0,lowprice-Range*.05),highprice+Range*.05];
end

if length(timeVector) > 2
% Create appropriate axis viewing range
        axis(axispos)
        datetick('x',13)
end
end

% Add Title and grid to graph
title(['Real Time Rover Battery Stats , ',datestr(timeVector(1),1)])
ylabel('Value')
grid on
hold on


% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

handles = timerconncleanup(handles);

delete(hObject);

% --- Resets timer connection
function handles = timerconncleanup(handles)
```

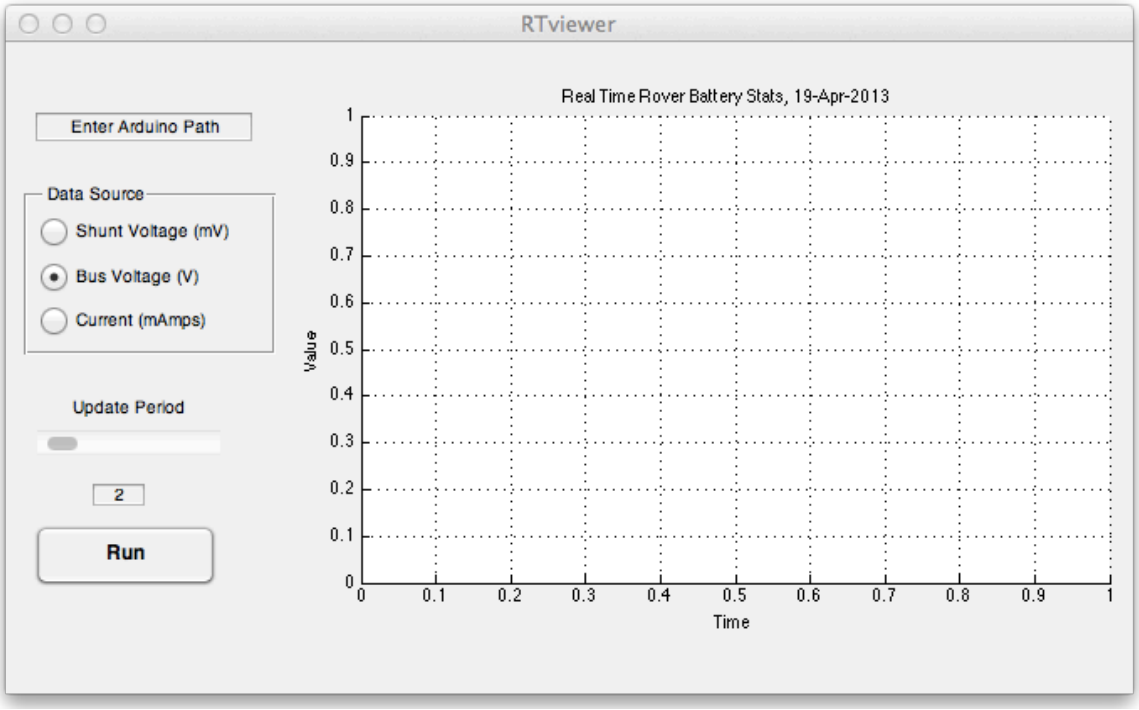
```
alltimers = timerfind;

if ~isempty(alltimers) && isfield(handles,'t')
if isequal(get(handles.t,'Running'),'on')
        stop(handles.t)
end
% use TIMERFIND function to find all timers, and delete them
    delete(alltimers)
    rmfield(handles,'t');
elseif isfield(handles,'t')
    rmfield(handles,'t');
end

if isfield(handles,'Conn')
    close(handles.Conn)
end

set(handles.tglRun,'value',0,'backgroundcolor','g','string','Run')


% --- Resets plotted data
function dataplotreset(ticker)

% Get Today's date
todaydate = getappdata(gcf,'todaydate');

reset(gca)
% Create Empty Graph
circs = scatter([],[],'filled');
grid on
title(['Real Time Rover Battery Stats, ',datestr(todaydate)])
xlabel('Time')
ylabel('Value')
setappdata(gcf,'circs',circs)

% Data Reset
setappdata(gcf,'currentValue',[]);
setappdata(gcf,'currentTime',[]);
setappdata(gcf,'shuntValue',[]);
setappdata(gcf,'shuntTime',[]);
setappdata(gcf,'busValue',[]);
setappdata(gcf,'busTime',[]);


% --- Creates real time
function tim = createtimer(realtimefun,ticker,period,handles,Conn)
% Create timer object with the datasource defined by the realtimefun
% function handle, and with ticker symbol and update period defined buy
% ticker and period respectively.

if nargin == 4
    tim = timer('TimerFcn',{realtimefun, ticker},'ExecutionMode', ...
'fixedRate', 'Period', period);
else
% Create timer object with data source TimerFcn
    tim = timer('TimerFcn',{realtimefun, Conn, ticker},'ExecutionMode', ...
```

```
'fixedRate', 'Period', period);
end


% --- Executes during object creation, after setting all properties.
function sldPeriod_CreateFcn(hObject, eventdata, handles)

usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor',[.9 .9 .9]);
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Updates the desired period
function edtPeriod_Callback(hObject, eventdata, handles)

% Fetch period value from the edit box
period = str2double(get(hObject,'String'));

% Set slider period value to period
set(handles.sldPeriod,'value',period)

% if the timer is running, it needs to be turned off before
% updating the period
if isfield(handles,'t')
if isequal(get(handles.t,'Running'),'on')
% Stop timer
        stop(handles.t)

% change period
        set(handles.t,'Period',period)

% Start it back up
        start(handles.t)
end
end

handles.period = period;
guidata(hObject,handles)


% --- Executes during object creation, after setting all properties.
function edtPeriod_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes on slider movement.
function sldPeriod_Callback(hObject, eventdata, handles)
% Change updating period via the slider control
```

```
% Fetch updating period from the slider
period = get(hObject,'Value');
handles.period = period;

% pass the period defined with the slider to the edit box
set(handles.edtPeriod,'string',num2str(period))

% If the timer is running, it needs to be turned off before
% updating the period
if isfield(handles,'t')
if isequal(get(handles.t,'Running'),'on')
% Stop timer
        stop(handles.t)

% change period
        set(handles.t,'Period',period)

% Start it back up
        start(handles.t)
end
end

% update handles structure
guidata(hObject,handles)


% --- Executes just before RTviewer is made visible.
function RTviewer_OpeningFcn(hObject, eventdata, handles, varargin)

% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to RTviewer (see VARARGIN)
% Choose default command line output for RTviewer

handles.output = hObject;

% Get Ticker
ticker = upper(get(handles.edtTicker,'string'));

% Find today's date
clk = clock;
todaydate = datenum(clk(1),clk(2),clk(3));
setappdata(gcf,'todaydate',todaydate)

% Create Empty Graph
circs = scatter([],[],'filled');
grid on
title(['Real Time Rover Battery Stats, ',datestr(todaydate)])
xlabel('Time')
ylabel('Value')

% Define timer object based on Random Data as default
```

```
handles.t = timer('TimerFcn',{@busRT, ticker},'ExecutionMode', ...
'fixedRate');
handles.ticker = ticker;

% Pre-define the current and time vectors for the data
setappdata(gcf,'currentValue',[]);
setappdata(gcf,'currentTime',[]);

setappdata(gcf,'circs',circs)

guidata(hObject, handles);


% --- Outputs are returned to the command line.
function varargout = RTviewer_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes during object creation, after setting all properties.
function edtTicker_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Allows for port to be changed inside GUI
function arduinoPath_Callback(hObject, eventdata, handles)
% hObject    handle to arduinoPath (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of arduinoPath as text
%        str2double(get(hObject,'String')) returns contents of
%        arduinoPath as a double

setappdata(0,'ardPath',get(hObject,'String'))
```

5.3 Matlab GUI Figure



5.4 Steps to Follow

In order for the GUI to function properly, it was designed tostart from the top to the bottom of the options on the left. If not followed in the correct manner, the program will display an error message in the Matlab command window.

- ✓ Enter the port being used by the Arduino. If the incorrect path is used, the plot will not run!
- ✓ Choose the data you want to be displayed.
- ✓ Update the desired period for the measurements.
- ✓ FINALLY: Click on Run. Give it a second to start plotting and voila!!

5.5 Troubleshooting

- ✓ **Double check that you are using the right port**. A lot of grief over this!
- ✓ **Double check thatthe port is closed or not being used by another program.** Two extremely useful commands to be typed inside Matlab command window
  >> delete(instrfindall)
  >> instrfindall
  The first one will close all ports and the second will show that this is case, it should return an empty matrix.
- ✓ Whenever in doubt, unplug everything and if necessary close Matlab and start fresh.

- ✓ Make sure your battery's charge does not exceed the 7.2V…Learnt this the hard way!!! ☹
- ✓ The figure and the script files must be within the same folder otherwise Matlab will return an error.

Lab Deliverables:

1. Test to make sure the Arduino working properly.
2. Connect the Arduino to Matlab to obtain the voltage and current data and be able to plot the data with Matlab Gui