MIAMI UNIVERSITY

A MATLAB Approach To Home Automation And Detection

Winter 2018

College of Engineering and Computing

ECE 302: MATLAB and Its Engineering Applications

Q. Zhou, PhD

Jonathan Kosir & Jacob Iarve

January 26, 2018

Introduction

Among many of MATLAB's applications is that for embedded systems. MATLAB is a very powerful tool and can used to quickly prototype and develop models for real time signal processing without some of the constraints of platforms such as Arduino. Furthermore, with many of the advantages of MATLAB such as easy to use API. In fact, while MATLAB was designed for help solve complex problems, it contains toolboxes that even allow a user to write code in MATLAB and then generate standalone C, C++ files. Using MATLAB, our group has interfaced an Arduino using a serial connection and coded it, in MATLAB, to be used as a motion detection system for home security. This system uses an embedded light sensor to determine if motion is detected via changes in light, however is programmed to be self adapting and not trigger for natural changes in light such as during sunrise and sunset. Using MATLAB, programming the microcontroller was made easy and allowed for the development of this smart motion detection system.

Approach

Our approach consisted of three definite steps to complete the project. The first step was interfacing the Arduino in MATLAB and creating the Arduino object that could, afterwards, be used to program the embedded system. This step, which seemed to be the most difficult when doing the project research and writing the proposal, was actual very simple. Creating an arduino object, in MATLAB, is very simple and proves that MATLAB benefits users with simple to read and understand documentation. In fact, this step, was made simple and actual resulted in just one line of code:

```
myArduino = arduino(port,board);
```

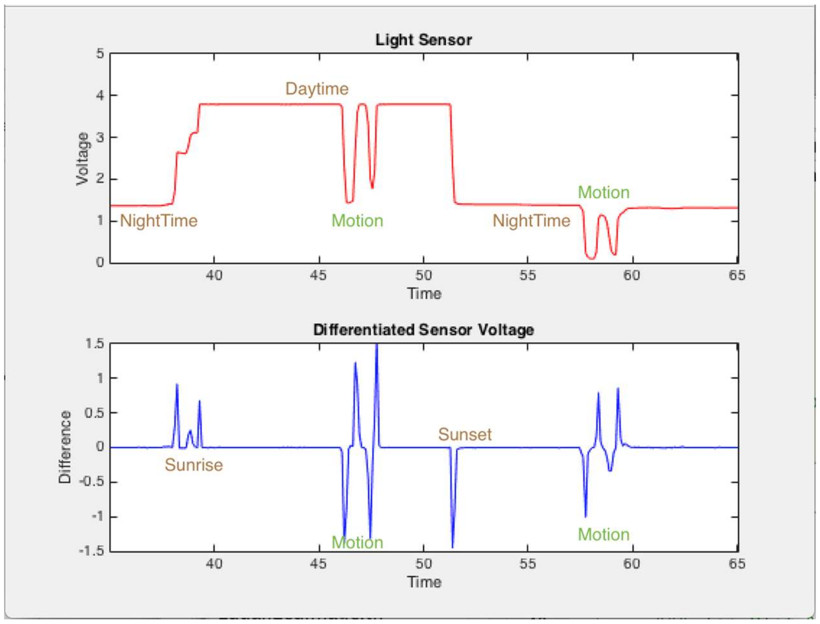
Thus, after instantiating the Arduino object, our group was able to move on to the second step which was processing signals in real time from different sensors attached to the arduino. In fact, included in our file submission is a file that allows a user to simply connect an Arduino and process up to four signals at a time from active sensors. Furthermore we made a tool that allows for quick comparison of different sensors, named *sensorTester.m* which was including in our submission file as well. Using MATLAB plotting voltage can be done at a much greater advantage than inside of Arduino platforms, or even excel, because you can control the axis values and range of the plot. This makes data analysis much easier and does not involve any data extraction, just a simple plug and chug.

Lastly, our final step was developing a system that responded to different changes in the voltage values read from our light sensor, and adding the specific instructions for what todo when motion is detected. This step is where we including the smart programming so that our system would be adaptable and not send false alarms due to natural changes of light during the day. Making the system smarter and more adaptable as to not trigger when unnecessary.

In conclusion, to discuss the applicability of coding this embedded device in MATLAB to respond to motion detection, our approach was to first interface the microcontroller, plot the sensor data in real time, and then code the response to different changes in voltage and alert the user. In the next section, we will discuss the solution process from the final stage in our approach, where we determine if changes in light are due to motion or not.

Solution Process

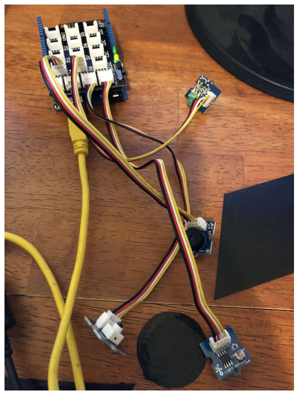
To begin coding actuators to respond and alert users of security threats from motion detection, first we needed to analyze the signal read from the light sensor. This signal was plotted and stays static in the environment for the most part until the change from daytime to night time is made. If we simply coded the light sensor to trigger from a threshold value, natural changes in light could result in triggering of the alarm. Thus, using the differentiated set of data from the original values we could loop for a specific signature in the signal that identifies motion (and works in both daylight and nighttime). The following is an annotated plot of the signal and its derivative:

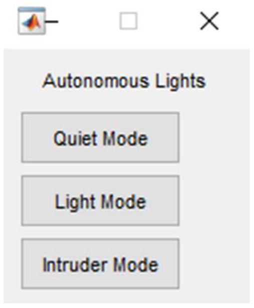


The top plot, labeled in red, is the voltage values read by the sensor at all times. The room where we tested this project started out with the lights off mimicking nighttime. At about 37 seconds,we flipped the lights on and the voltage responded by getting larger. This is seen at the increase in voltage towards the second plateau which is labeled as Daytime. At about 47 seconds in to the test, we moved a finger over the sensor and triggered the motion detection. This resulted in the dip in voltage, and the pulse curve shown in the differentiated graph. This pulse curve is the specific signature that our system looks for to determine if there is movement. Furthermore, to illustrate this phenomena and test the system further, we turned the lights back off again to mimic sunset. Here, you can see the drop in voltage value on light sensor curve and then we tested the motion detection again and once again saw the signature for motion detection. Finally, we were able to code a response specifically to this signature and alert the user of intrusion.

Results and Discussion

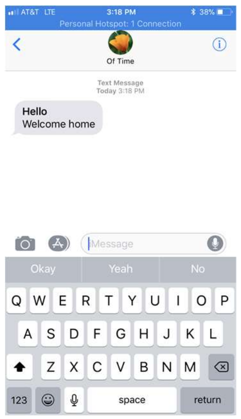
As stated above now our program is able to sense change in light and effectively motion. With this ability we are now able to begin to add responses to these detections. To do this we have three separate "modes" all which respond differently when the arduino senses motion. These modes can be chosen two different ways. First when the program is started a menu pops up (this simple menu is shown below). This menu has all three of the options that can be chosen from. If the user changes their mind, but wants to keep the program running they can switch modes by a simple button press which cycles through the three choices. Our Arduino setup is shown below with our two sensors, light and button press, and our two response components, LED for light and the buzzer for sound.





The first setting is a quiet setting. This setting was mostly used in testing but can also be used for other purposes. With this the Arduino simply runs the data analysis but when it senses a trigger it does nothing. This way we were able to test to make sure the graph was correct and everything was responding accordingly without our alerts going off.

Next we have the light mode, or setting two. This mode is simply for when a user may be home or is not worried about an intruder. It works similar to our classroom lights. When motion is sensed the lights turn on and a text is sent to the user saying hello. Once triggered a timer for the length you would like your lights to stay on is defined in the code. After that set

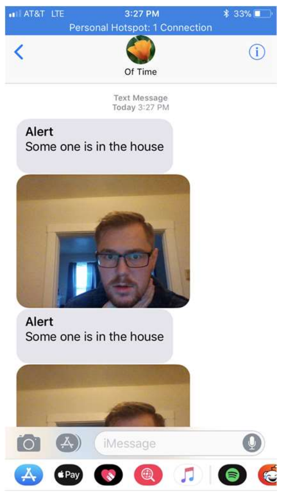


Text Message
Today 3:18 PM

Hello
Welcome home

time and no movement has been sensed the lights go off.

Lastly there is an intruder mode. This would ideally be set when you leave the house and would like to know if someone is breaking in. When motion is sensed a buzzer goes off in attempt to scare the intruder into leaving. Not only does it try to scare the intruder but a picture is then snapped without alerting anyone, the the intruder would never even know it was taken. This is done using the computers webcam and then a text is sent with the image to the user. This way you know that someone is in your house and they triggered your motion alarm. The picture comes in super handy when working in this setting. When sent and you are not home the picture can be used to see if the trigger was a false alarm. A false alarm could easily be caused by a spouse who returned home early or even a pet. This way you know if it is something serious and you can call the cops and you even have a picture to give them of the person who broke in!



Alert
Some one is in the house

Alert
Some one is in the house

With all of this three packages were used: The arduino package, the OS generic video interface, and the us webcams MATLAB package. These are packages which helped us seamlessly interact with the arduino and the webcams. Then with some simple code and a little bit of research we were able to send text messages via the email function provided in MATLAB.

Conclusion

Working with MATLAB to program gave us many options that we wouldn't have had if we programmed the Arduino directly. The ease of use being number one. We were able to easily install packages so we could work with many different peripherals such as the webcam and sending emails. Not only this but MATLAB excels in analyzing data, and being able to analyze our sensor information easily and efficiently using MATLAB was key to give us the ability to add more functionality into our embedded system.

If given more time we would have added a couple more things which would have made our Arduino even more user friendly and useful in everyday life. We would of added an online control where a user can control the modes via an online website GUI. We would have made the GUI menu more extensive and allow the user to have more control over what they want. Maybe an extra mode or two would have been added such as a morning wake up alarm, or a sound sensor for if someone breaks in through a window. For our intruder mode we would have also liked to included some sort of biometric feature. If the user who set off the alarm matches a facial recognition the alarm would turn off, as well a fingerprint scanner to turn off the alarm would have been good too. Lastly, we would have liked to build a chassi or some sort of housing for our device to make it more market ready.

Appendix

Code:

Main Arduino Function

```
function [] = arduino3()

%% Written By: Jacob Iarve && Jonathan Kosir
% This code is written to continuously plot voltage from an arduino
% given an input for frequency.  Takes in data and sends alerts

% Needed dependencies
% Webcam that works with matlab
% Packages/ Apps
      % Arduino package
      % OS generic video interface
      % us webcams matlab package
% Arduino with sensors and components
      % Light
      % button
w
      % led
      % sound

%% Hardware information
port = 'com3';
board = 'Uno';

%%% PORTS
lightSensorPort = 'A0';
soundSensorPort = 'A1';
lightPin = 'D8'
noisePin = 'D4'
buttonPin = 'A1'

%% Create arduino object
myArduino = arduino(port,board);

%% Grab webcam info
list = webcamlist
list =list{1};
cam = webcam(list);

%% Define Function Variables
time = 0;
data1 = 0;
diffData = 0;
```

```
count = 0;
peak = 0;
saveTime = 0;
onOff = 0;
senseMode = 1;
buttonTime = 0;
freq = 5000;
delay = 1/freq;
threshhold =  .05;
%%Plot Dimension
interval = 5;

% Create menu for alert type
menuFrame = menu('Autonomous Lights', 'Quiet Mode', 'Light Mode', 'Intruder
Mode', 'Wake up Mode');
senseMode = menuFrame

%Variables for email definition 9Set prefs and definitions found online
% at Mathworks and creators information was used
setpref('Internet', 'SMTP_Server',   'smtp.mail.yahoo.com');
setpref('Internet','E_mail','jmkosir@sbcglobal.net');
setpref('Internet', 'SMTP_Username', 'jmkosir@sbcglobal.net');
setpref('Internet', 'SMTP_Password', 'J32yahooK44');

props = java.lang.System.getProperties;
props.setProperty('mail.smtp.auth',              'true');  % Note: 'true' as
a string, not a logical value!
props.setProperty('mail.smtp.starttls.enable',  'true');  % Note: 'true' as a
string, not a logical value!
props.setProperty('mail.smtp.socketFactory.port',  '465');   % Note: '465'
as a string, not a numeric value!
props.setProperty('mail.smtp.socketFactory.class',
'javax.net.ssl.SSLSocketFactory');

%% Setup
figure

%%% Top Plot %%%
top = subplot(2,1,1);
plotGraph = plot(time,data1,'R-');
title('Light Sensor');
xlabel('Time');
ylabel('Voltage');
axis([0 interval 0 5]);

%%% Bottom Plot %%%
bottom = subplot(2,1,2);
diffGraph = plot(time,diffData,'B-');
title('Differentiated Sensor Voltage');
```

```
xlabel('Time');
ylabel('Difference');
axis([0 interval -1.5 1.5]);

% Make sure both x axis are the same
linkaxes([top, bottom],'x')

%%%INFINITE LOOP%%%%%%%
%% Real Time Plotting Alogrithm
tic % start timer

while ishandle(plotGraph)

      % Statement to switch modes with button sensor
      % If button is pressed and enough time has elapsed go in
      if(readVoltage(myArduino,buttonPin) > 1 && time(count) - buttonTime  >
1)
      buttonTime = time(count);

      % Count up through the modes until 3 and then reset
      if(senseMode ~= 3)
            senseMode = senseMode + 1
      else
            senseMode = 1
      end
      end

      % Read sensor, increment counter and timer
      readLightSensor = readVoltage(myArduino,lightSensorPort);
      count = count +1 ;
      time(count) = toc;
      data1(count) = readLightSensor;

      % start recording differential data after first loop
      if(count > 1)
      diffData(count) = data1(count) - data1(count-1);
      end

      % Create our two graphes
      set(plotGraph,'XData',time,'YData',data1);
      set(diffGraph,'XData',time,'YData',diffData);

      % set peak to one meaning our threshhold was broken time to check
      % for oppisite peak
      if(diffData(count) > threshhold)
      saveTime = time(count);
       peak = 1;
      end
      % if statement for derivative values if the difference value is
      % meets the threshhold positively and negatively and the time was
      % with in reason it enters statement
      if(peak == 1 && (time(count) - saveTime) <= 1.2)

      % switch statment to have response to each mode
      switch senseMode
            case 1 % Quiet mode nothing happens

            case 2 % Light mode/ LED come on and sends a hello text
                  peak = 0;
                  onOff = 1;
                writeDigitalPin(myArduino,lightPin,onOff);
                  sendmail('4402202880@mms.att.net', 'Hello', 'Welcome
home');

            case 3 % Introduer mode alarm sounds and text is sent
                  peak = 0;
                  onOff = 1;

                  % Take picture of intruder!
                  img = snapshot(cam);
                  imwrite(img, 'pic.jpg');

                writeDigitalPin(myArduino,noisePin,onOff);
                  sendmail('4402202880@mms.att.net', 'Alert', 'Some one is in
the house', 'pic.jpg');

                  %{
            Not using for this project can ass a wake up mode
            at a later date.
            case 4 % Wake up mode (work in progress)
                  peak = 0;
                  onOff = 1;
                writeDigitalPin(myArduino,lightPin,onOff);
                %sendmail('4402202880@mms.att.net', 'Alert', 'Some one is in
the house', );
                  %}

            otherwise % Otherwise something went wrong
                  disp 'How did you get here'
      end
      end

      % After 5 seconds turn off things that were turned on
      if((time(count) - saveTime)   >  2)
      onOff = 0;
        writeDigitalPin(myArduino,lightPin,onOff);
        writeDigitalPin(myArduino,noisePin,onOff);
      end
      pause(delay);

      % Plot the x axis correctly
      if time(count) > interval
      axis([time(count)-interval time(count) -1.5 1.5]);
      end
end
end % End function
```

```
function[] = sensorTester()

%% Written By: Jacob Iarve
% This code is written to continuously plot voltage from sensors attached
% to an Arduino. Made for the quick and simple testing of sensors at
% different sampling frequencies for comparison.
%
%
% Steps to use this program:
% 1. Change Hardware information to your computer.
%          For Mac: use '/dev/cu.usbmodem****' for your port
%          For Windows: use 'com*'
% 2. Change board type
% 3. Input Sensors Ports for each sensor (i.e. A0 for analog 0)
% 4. Change Plot titles
% 5. Change frequency
% 6. Change x axis interval



%% freq = input frequency as an argument

%% Hardware information

  port = '/dev/cu.usbmodem1411';
  board = 'Uno';

  %%% PORTS
  sensor1 = '';
  sensor2 = '';
  sensor3 = '';
  sensor4 = '';

%% Plot Titles

  title1 = '';
  title2 = '';
  title3 = '';
  title4 = '';

%% Frequency
  freq = 5000;  %Set to 5000 Hz

%% X Axis Interval
   interval = 5; %Set the x axis to continuously show x second intervals
                 %% Starting is 5 seconds


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% DO NOT CHANGE CODE BELOW %%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Create arduino object
    myArduino = arduino(port,board);

%% Define Function Variables
   time = 0;
   data1 = 0;
   data2 = 0;
   data3 = 0;
   data4 = 0;
   count = 0;

   delay = 1/freq/60;




%% Setup
    figure


    %%% subplot 1 %%%
    sub1 = subplot(2,2,1);
    plotGraph1 = plot(time,data1,'R-');
    title(title1);
    xlabel('Time');
    ylabel('Voltage');
    axis([0 interval 0 5]);



    %%% subplot 2 %%%
    sub2 = subplot(2,2,2);
    plotGraph2 = plot(time,data2,'B-');
    title(title2);
    xlabel('Time');
    ylabel('Voltage');
    axis([0 interval 0 5]);


    sub3 = subplot(2,2,3);
    plotGraph3 = plot(time,data3,'O-');
    title('title3');
    xlabel('Time');
    ylabel('Voltage');
    axis([0 interval 0 5]);

    sub4 = subplot(2,2,4);
    plotGraph4 = plot(time,data3,'Y-');
    title('title4');
    xlabel('Time');
    ylabel('Voltage');
    axis([0 interval 0 5]);



    linkaxes([sub1, sub2, sub3, sub4],'x')

%%%INFINITE LOOP%%%%%%%
%% Real Time Plotting Alogrithm
tic

while ishandle(plotGraph1)
```

```
        readSensor1 = readVoltage(myArduino,sensor1);
        readSensor2 = readVoltage(myArduino,sensor2);
        readSensor3 = readVoltage(myArduino,sensor3);
        readSensor4 = readVoltage(myArduino,sensor4);
        %readSoundSensor = readVoltage(myArduino,soundSensorPort);

        count = count +1 ;
        time(count) = toc;

        data1(count) = readSensor1;
        data2(count) = readSensor2;
        data3(count) = readSensor3;
        data4(count) = readSensor4;

       set(plotGraph1,'XData',time,'YData',data1);
       % set(plotGraph1,'XData',time,'YData',data2);
       set(plotGraph2,'XData',time,'YData',data2);
       set(plotGraph3,'XData',time,'YData',data3);
       set(plotGraph4,'XData',time,'YData',data4);



        %axis([time(count) time(count) 0 5]);

        pause(delay);


        disp([time(count) count])
        if time(count) > interval
           %% data1 = data1(2:end);
            axis([time(count)-interval time(count) 0 5]);
        end

end



end
```