ENR261 Final Project

Help Reference

Controlling an Arduino

Listing Available Ports

In order to connect to a device, you will need to know what ports are available on your system. In order to acquire this list we will use the `serialportlist` command in MATLAB.

To get the list of available com ports on your system, run the following command

```
availablePorts = serialportlist'
```

Now you have a variable named `availablePorts` which contains an nx1 string array of available ports on your system where n represents the number of ports which may be different for each computer.

Connecting to an Arduino

To connect to an arduino board, there are a number of possible arguments but at the minimum we need the type of board and the com port we will connect to.

Obtaining the correct COM port

Unless you know which port your board is connected to, you'll have to guess and check but generally each USB port has its own name and COM1 is most commonly **not** a USB port so you may want to skip over that one entirely. In general you will only have two ports available where COM1 is one of those ports so guess and check is fairly quick.

Another option is to print `serialportlist` before and after connecting your Arduino and note the newly available port.

Obtaining the correct board name

The connection function for Arduino accepts a number of supported board names and can be found in the help documentation. For your project you will be using an Arduino Nano and the board name for that is 'Nano3'.

Making the connection

Now that you have the board name and the appropriate COM port, we can run the function that creates the connection to the Arduino board.

NOTE: The initial connection can take a few minutes as MATLAB must first upload a special program onto the board in order to manipulate it. Please be patient and wait for MATLAB to finish.

Run the following command to create a connection to your Arduino board substituting <COM_PORT> for your appropriate COM port of course.

```
a = arduino('<COM_PORT>', 'Nano3');
```

Controlling a digital pin

To control (set) a digital pin's output we will use a function called `writeDigitalPin()` which accepts three arguments. We must supply the handle to our arduino object, the name of the digital pin we would like to control, and the value we would like to set that pin to. In the case of a digital pin the value is a binary one, digital pins can be either on or off which is represented by 1 for on and 0 for off or true for on and false for off. The choice is yours although generally 0 or 1 is a better representation of a pin's state.

So, to *set,* for example, pin 13's output to HIGH or ON, we use the following command

```
writeDigitalPin(a, 'D13', 1);
```

And to turn that pin off we would use the following command

```
writeDigitalPin(a, 'D13', 0);
```

Notice that `writeDigitalPin` is very similar to a function called `set` where set accepts an object handle, a property name, and a property value. We are basically doing the same thing here, we are setting the value of the digital pin.

An Arduino Example

Now that we've covered how to connect to an Arduino and how to control a digital pin, let's go through an example of blinking an LED a number of times.

First, we'll need to connect to the Arduino so we add the code

```
a = arduino('COM4', 'Nano3');
```

Next we want to loop "a number of times" so that should be a dead giveaway that we want a for-loop since for loops are the simplest way to loop for some number of times. Let's loop ten times and inside that loop we'll turn our pin on and off.

```
for i = 1:10
     writeDigitalPin(a, 'D13', 1);
     writeDigitalPin(a, 'D13', 0);
end
```

Now, what you may not already know is that digital pin 13 (D13) on nearly all Arduinos and Arduino compatible boards is a pin that has an LED built onto the board. So, when you run this code you should see the LED labeled L flash.

If you notice though, the LED doesn't really seem to blink, it just turns on and off really fast for what appears to be one time. There is a reason for this and this reason is something you have to always keep in mind when developing embedded software. The microcontroller will literally do whatever you tell it to and it doesn't know what to do unless you explicitly tell it to do so.

The issue you're seeing here is that we've told the microcontroller to turn the pin on and off ten times but the microcontroller is doing this so fast that it's imperceptible to the human eye. Because of this, we have to put in some time delays so that we can see this pin change happening. To add a time delay we can use the `pause()` function which accepts a number (which can be a decimal) that represents the number of seconds to wait before performing the next line of code.

The final example should look like this:

```
if exist('a', 'var') == 0
```

```
    a = arduino('COM4', 'Nano3');
end

for i = 1:10
    writeDigitalPin(a, 'D13', 1);
    pause(0.1);
    writeDigitalPin(a, 'D13', 0);
    pause(0.1);
end
```

One more thing you should notice is that we included two pauses, this is because we need to not only pause after turning the LED on but we need to pause after we turn it off as well, otherwise we won't see the off portion of the change.

It is extremely important that you understand why we need the pause here. It is also extremely important to understand that while paused, the microcontroller is in what is called a *busy-wait* state. A busy-wait is a state in which the microcontroller literally does nothing but waste time until the desired time has elapsed. During this pause, the microcontroller cannot do anything else. There are ways around this which we will get into later but for now, understand that there are times when we may not only want the microcontroller to wait for a specified time but also wait for a condition to be met other than the time or that, instead of waiting for some time we simply want to wait until a certain condition is met (a pin goes high or low, etc) and for these cases we cannot use a pause.