

Programming a web page for arm control using the web serial API?

How it works:

We're going to write a very basic web app using the Flask framework in python with a few buttons that will control our Arduino through a serial interface with python. What does this mean? It means that when we send a request to our webpage, our web server (aka our computer) will then send a message to our Arduino via the serial port/USB port on our computer, our Arduino device will interpret that message and perform said task.

Requirements:

Arduino Uno

1 LED + Resistor, anything between 220 ohm and 1k ohm

1 Photoresistor + 10k Ohm Resistor

Wires

Breadboard

Arduino

USB cable

Computer

Procedure:

1. Set up our Arduino circuit
2. Make sure we have Pyduino set up
3. Create our webserver and test it out
4. Link our Arduino to our webserver
5. Control our Arduino with our mobile phone
6. Profit!
7. Create basic web-api interface

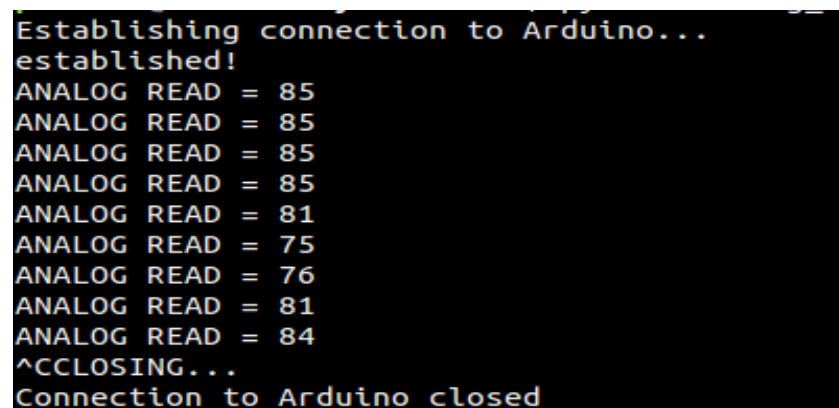
Step 1: Setting Up the Arduino Circuit

We are going to adopt a similar circuit as most tutorials out there that have you control an LED with a photoresistor. There are multiple ways to do this, check out the two images above to see what your board should look like. The two circuits do the same thing yet they're wired in a different sequence which is ok! Now its up to you on what you want to do next, I like to bend my photoresistor towards the LED so we can pick up some of its light.

Things to notice: I have the LED wired to pin #3 and the analog input for our photoresistor wired to A0 on the Arduino board.

Make sure your Arduino is connected to your PC via USB

Step 2: Setting Up Pyduino + Checking Circuit



```
Establishing connection to Arduino...
established!
ANALOG READ = 85
ANALOG READ = 85
ANALOG READ = 85
ANALOG READ = 85
ANALOG READ = 81
ANALOG READ = 75
ANALOG READ = 76
ANALOG READ = 81
ANALOG READ = 84
^CCLOSING...
Connection to Arduino closed
```

Now that we have our circuit set up lets test it to make sure everything is good before we create our webpage. To do this we need to make sure that we have our pyduino sketch loaded onto our arduino board and that we have the pyduino library. To set up our arduino to work with the pyduino library follow this instructable before continuing!! -> [LINK](#) otherwise you cannot do the rest of this instructable

Assuming that you all have followed the instructable above we are now ready to test our circuit. You should have your pyduino.py library file in the same directory that we're going to make this next python script to test our circuit.

What do we want our script to do?

1. Establish serial connection to our arduino device
2. Turn on LED

3. Obtain analog reading from photoresistor
4. Close connection to arduino device at the end

Save the piece of code below as: **analog_read_test.py**

To run the piece of code type in terminal: **python analog_read_test.py**

When you run the code you'll see your analog reading being printed every second in your terminal. To make sure your photoresistor works, place your hand or other opaque object in front of photoresistor and watch how your analog values change! See the image above for what your terminal output should look like.

```
from pyduino import *
import time

if __name__ == '__main__':

    print 'Establishing connection to Arduino...'

    # if your arduino was running on a serial port other than '/dev/t
    tyACM0/'

    # declare: a = Arduino(serial_port='/dev/ttyXXXX')

    a = Arduino()

    # sleep to ensure ample time for computer to make serial connecti
    on

    time.sleep(3)

    print 'established!'

    # define our LED pin
```

```

PIN = 3

# initialize the digital pin as output

a.set_pin_mode(PIN,'O')


# allow time to make connection

time.sleep(1)


# turn LED on

a.digital_write(PIN,1)

for i in range(0,1000):

    try:

        # Read the analog value from analogpin 0

        analog_val = a.analog_read(0)


        # print value in range between 0-100

        print 'ANALOG READ =',int((analog_val/1023.)*100)

        time.sleep(1)


    except KeyboardInterrupt:

        break # kill for loop


# to make sure we turn off the LED and close our serial connection

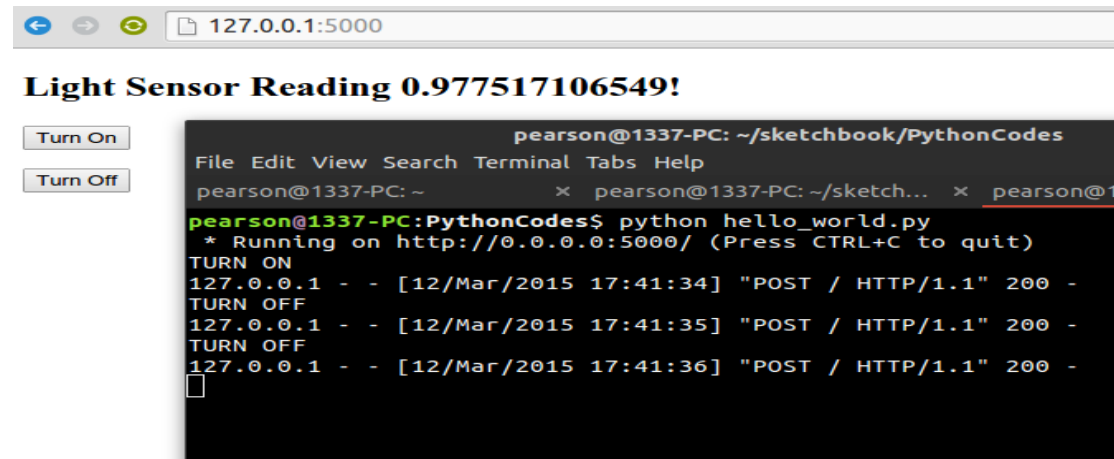
print 'CLOSING...'

a.digital_write(PIN,0) # turn LED off

```

```
a.close()
```

Step 3: Creating the Webserver with Python



I am no expert in web programming or specific networking protocols yet I managed to control my arduino over the web so I have faith that all of you can too! We're going to create a very basic website and have our computer be the host. To do this we are going to use the web framework Flask to create a web based API for controlling our arduino. I highly encourage you all to read through some examples on using Flask so that you can understand the basics of how it works and because I may not be the best teacher.

What do we want to do?

1. Create a webpage with 2 buttons
 1. Turn Off LED Button
 2. Turn On LED Button
2. When we turn on/off the LED we want to read the value on our photoresistor and display it
3. Create a specific URL for turning our LED on/off - Necessary for web API

To create a dynamic webpage that updates with our photoresistor value when we load the page we need to create a template page in Flask. Flask will use the value we get from our photoresistor and send it to our template to display. Flask automatically follows a specific directory format when searching for our template so be vigilante about where you place certain files. In the directory you're working in you will want to create a directory called templates. We will

put our template page in there. Our template html page will look very similar to a normal HTML page just with an added option to place a variable somewhere. Copy the snippet of html code on this link: <http://codepen.io/theown1/pen/RNeYMG> into a new file called index.html and place that file in your templates directory.

Copy the code below to a file named **hello_flask_world.py** The piece of code below is our basic Flask webpage. Its pretty simple to understand, we just have one webpage at the '/' address which is the home page. The URL for this webpage will be your ip address on the network or 127.0.0.1:5000. When someone connects to our webpage we just render our template we had earlier with some value that we're going to change to our photoresistor output later. If our webpage gets a POST request aka someone presses a button on our page then our server will perform the specific task of printing out what happened and redirect you back to the main page. You can run this piece of code below like you would a normal python program. (\$ python hello_world.py)

```
from flask import Flask, render_template, request, redirect, url_for
app = Flask(__name__)
# we are able to make 2 different requests on our webpage
# GET = we just type in the url
# POST = some sort of form submission like a button
@app.route('/', methods = ['POST', 'GET'])
def hello_world():
    # variables for template page (templates/index.html)
    author = "Kyle"
    readval = 10
    # if we make a post request on the webpage aka press button then
    do stuff
    if request.method == 'POST':
        # if we press the turn on button
        if request.form['submit'] == 'Turn On':
            print 'TURN ON'

        # if we press the turn off button
        elif request.form['submit'] == 'Turn Off':
            print 'TURN OFF'
        else:
            pass

    # the default page to display will be our template with our
    template variables
    return render_template('index.html', author=author,
value=100*(readval/1023.))
if __name__ == "__main__":
    # lets launch our webpage!
    # do 0.0.0.0 so that we can log into this webpage
    # using another computer on the same network later
    app.run(host='0.0.0.0')
```

When you run this program you should get an output that is similar to the picture above. Make sure you connect to the webpage on your browser at <http://127.0.0.1:5000/> after you run the python script (and keep it running!) Then play with the buttons and verify you're getting them to print something in the terminal.

The folder you're working in should now have all of the files below in the corresponding directories.

```
ARBITRARY_WORK_FOLDER/
    |-- pyduino.py           # pyduino library file
    |-- analog_read_test.py  # pyduino code to check circuit
    |-- hello_flask_world.py  # your first Flask webpage
    |-- templates/           # directory to store flask templates
    |-- index.html           # our html template
```

Next it's time to integrate our Arduino controls into the webserver!

Step 4: Integrating Pyduino with Flask

Now that we verified that our circuit is set up properly and that our first webpage works, it's time to add the pyduino commands that control our Arduino into the Flask webpage! We're going to add a few python commands to our previous `hello_flask_world.py` program. Go ahead and copy your **hello_flask_world.py** program to a new file called **pyduino_website.py**

```
a.digital_write(LED_PIN,1)

# if we press the turn off button
elif request.form['submit'] == 'Turn Off':

    print 'TURN OFF'

    # turn off LED on arduino
    a.digital_write(LED_PIN,0)

else:

    pass
```

```
# read in analog value from photoresistor

readval = a.analog_read(ANALOG_PIN)

# the default page to display will be our template with our template variables

return render_template('index.html', author=author, value=100*(readval/1023.))

if __name__ == "__main__":

    # lets launch our webpage!

    # do 0.0.0.0 so that we can log into this webpage

    # using another computer on the same network later

    app.run(host='0.0.0.0')
```

Let's run the program and see what we get!! Check out the video above for what you should hopefully get!

Step 5: Controlling Your Arduino with Your Mobile Device!

Now that you successfully have your website up and running and its controlling your arduino it's now time to see if we can control it with our mobile device.

The Steps

1. Connect computer and phone to same wifi network
2. Disable firewall on computer
3. Figure out network address of computer on wifi network
4. Run python website on computer
5. Connect to ip address of computer with phone browser

In order to connect to your computer that is hosting the website, you will need to disable the firewall on your computer or make an exception for a connection on a certain port. I can not tell you how to do this exactly because

most people have different computers so just google it! I am creating this instructable from Ubuntu so I have a graphical window to control ufw (my firewall and I can just disable it).

To figure out your network ip, google it if you're on a windows machine or mac otherwise type: **ifconfig** into terminal and figure out what your inet addr is on your wlan0 connection, it will probably start with 192.168.XX.XXX

That is the address you will type in on your phone. See the picture above what I did.

If you want to connect to this arduino device from a computer outside your network you will have to set a port forwarding connection on your router and link it up to your computer.

Step 6: Creating a Basic Web Api with Python

If we want to control our arduino device without having to interact with an interface we can create a very basic web api with our python code. Keep in mind that this connection is not secure, you will not have to validate your credentials or anything so be careful if you decide to upload and use this code for real.

Add this snippet of code below to your **pyduino_website.py** file just above the main block of code

```
# unsecure API urls
@app.route('/turnon', methods=['GET'] )

def turn_on():

    # turn on LED on arduino

    a.digital_write(LED_PIN,1)

    return redirect( url_for('hello_world') )

@app.route('/turnoff', methods=['GET'] )

def turn_off():

    # turn off LED on arduino

    a.digital_write(LED_PIN,0)
```

```
return redirect( url_for('hello_world') )
```

This will allow us to now go to the url: <http://127.0.0.1:5000/turnon> to turn our led on and subsequently <http://127.0.0.1:5000/turnoff> to turn our led off. If you want to make this connection secure by creating an api key before the arduino will execute the task look into the API controls in Flask on this page: <http://flask.pocoo.org/snippets/category/apis-and-microformats/>

