CISC 340 Project – Illuminated WiFi Range Finder

*Works anywhere in the world! ™*

Written by Will Thompson and Steve Emmanuel

# Overall Device Purpose

The overall purpose of our component of the project was to transmit information from one Raspberry Pi connected with a sensor, to another Raspberry Pi connected to a display device, over the internet. The Raspberry Pi’s needed to be able to communicate even if they are connected to different WiFi networks in different physical locations.

# WiFi Connection

The WiFi dongles connected to each Pi appear as standard Linux WiFi network interfaces and no further driver setup is required. Upon startup, a user must configure the WiFi connection of each Pi using the WiFi Setup tool included with Raspian. We assume that the Pi’s will always be connected to a network with DHCP so no static IP needs to be assigned; instead, it will be assigned one automatically by the DHCP server for that network.

# Client-Server Architecture

A requirement of our project is that the two Pi’s must be able to communicate even when they are not on the same network; we supported this requirement by implementing a client-server architecture with a third-party server (not on either Pi) at a publicly accessible address visible to both Pi. This requires that each Pi be connected to a WiFi connection, regardless if they are on separate networks. Our third-party server also needs to be connected to a WiFi network. The beauty of our design is that all components can be used on WiFi connections across the world, and only experience slight connection delays.

It is also worth noting that our third-party server design is demonstrated in many real life applications in the technology world. Similar to how a Google search is sent through servers, our Pi data will always flow through the server to allow for a consistent and stable operation. Additional perks of a server-client design include:

* Increased security between the Pi’s data transfer. Access rights can be defined at the time of server setup.
* The centralization of all data flowing through the server. This allows all the data to be securely stored in one place, accessible for authorized users to detect bugs, and further improve operation.
* Increased scalability by simply upgrading the server at a later time.

We wrote the server using Python and the Flask web server framework. It hosts a REST API with two endpoints, “/message” and “/message/latest”. Upon receiving a POST request on “/message”, the server places the contents of the POST body into a buffer. Requests to “/message/latest” return the newest message in the buffer. The server is publicly accessible at <http://104.247.231.22/>.

Each Pi runs a python-based web client built using the standard library “urllib2” and “json”. The sensor Pi uses urllib2 to send POST requests to the server containing the sensor value and a timestamp in JSON format every one second. We have designed the sensor value to be an integer ranging from 1 to 16, because of there being only 16 lights on the display. The sensor itself is able to give a range from 0 to 255, allowing for greater accuracy. However, due to the nature of the light display we are using, 1 will indicate a very close distance, and 16 will indicate a max distance. The display Pi also uses “urllib2” and “json” to retrieve the messages the sensor Pi posts to the server every 1 second. Only the latest message is returned to the display Pi.

# Demo Code

|  |
| --- |
| **server.py** |
| from flask import Flask, request  app = Flask(\_\_name\_\_)  # Buffer of most recent messages  messages = ['']  # Receive a new message, and store it in the buffer  @app.route('/message', methods=['POST'])  def new\_message():  print('Data:'+str(request.data))  messages.append(request.data)  return 'Received!'  # Get the latest message from the buffer when requested  @app.route('/message/latest')  def get\_message():  print('request received')  return messages[-1]  # Start the server, hosting on all local ip addresses  app.run(host='0.0.0.0') |

|  |
| --- |
| **sensor.py** |
| import urllib2  import time  import json  ip = "192.168.1.170"  url= "http://192.168.1.170:5000/message"  for i in range(10):  sample = {"Value": 0.5, "time": i}  data = json.dumps(sample)  req = urllib2.Request(url, data, {'Content-Type': 'application/json'})  varRec = urllib2.urlopen(req)  varRec.read()  varRec.close()  time.sleep(1) |

|  |
| --- |
| **light.py** |
| import urllib2  import time  import json  ip = "192.168.1.170"  for i in range(10):  varRec = urllib2.urlopen("http://"+ip+":5000/message/latest")  data = json.load(varRec)  print data  time.sleep(1) |