**Objective:** This lab is to simulate measuring water level using an ultrasonic sensor. The measured value must be stored locally as a text file with time stamp and sent to the server via the given API.

**Apparatus:** Raspberry Pi 1

SD Card 1

Power Adaptor 1

PC or Mac 1

Ultrasonic Sensor (HC-SR04) 1

**Preconditions that I completed:**

1. I have completed Labs 1, 2, 3, 4, and 5
2. The following is the information on Raspberry Pi pins

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Raspberry Pi Model B** | | | | | |
|  |  |  |  |  |  |
|  | 3.3V | 1 | 2 | 5V |  |
|  | I2C1 SDA | 3 | 4 | 5V |  |
|  | I2C0 SCL | 5 | 6 | Ground |  |
|  | GPIO 4 | 7 | 8 | UART TXD |  |
|  | Ground | 9 | 10 | UART RXD |  |
|  | GPIO 17 | 11 | 12 | GPIO 18 |  |
|  | GPIO 27 | 13 | 14 | Ground |  |
|  | GPIO 22 | 15 | 16 | GPIO 23 |  |
|  | 3.3V | 17 | 18 | GPIO 24 |  |
|  | SP10 MOSI | 19 | 20 | Ground |  |
|  | SP10 MPSO | 21 | 22 | GPIO 25 |  |
|  | SP10 SCLK | 23 | 24 | SP10 CEO N |  |
|  | Ground | 25 | 26 | SP10 CE1 N |  |
|  |  |  |  |  |  |

**Task 1: Default cmdline.txt**

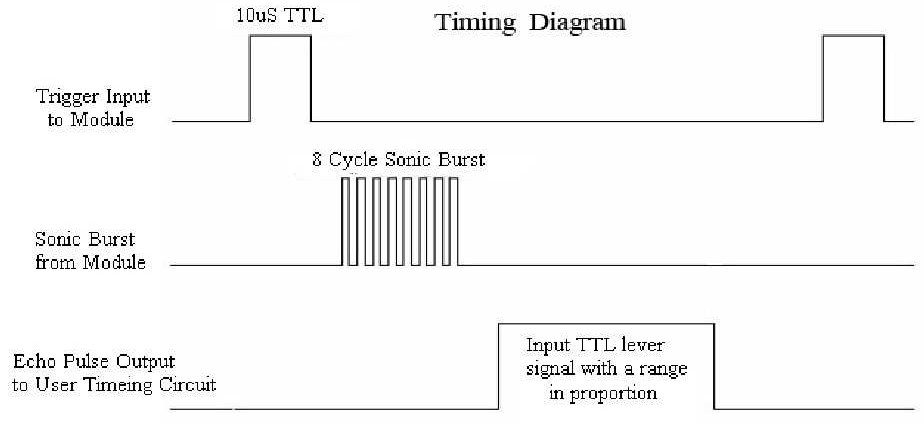
1. I connected the provided ultrasonic sensor with GPIO of Raspberry Pi as follow:





2. I wrote this python code to measure the distance of an object (water level). This task is to test that my connections and ultrasonic sensor was working properly. Here is the information on how to measure (calculate) the distance:

Test distance = high level time×velocity of sound (340M/S) / 2



*Note: more information can be found from datasheet:*

*http://e-radionica.com/productdata/HCSR04.pdf*

**My completed Python code:**

import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BCM)

TRIG = 17

ECHO = 27

print "Distance Measurement In Progress"

GPIO.setup(TRIG,GPIO.OUT)

GPIO.setup(ECHO,GPIO.IN)

GPIO.output(TRIG, False)

print "Waiting For Sensor To Settle"

time.sleep(2)

#Send 10us pulse to trigger

GPIO.output(TRIG, True)

time.sleep(0.00001)

GPIO.output(TRIG, False)

while GPIO.input(ECHO)==0:

start = time.time()

while GPIO.input(ECHO)==1:

stop = time.time()

#calculate pulse length

elapsed = stop - start

#multiplied by the speed of sound (cm/s)

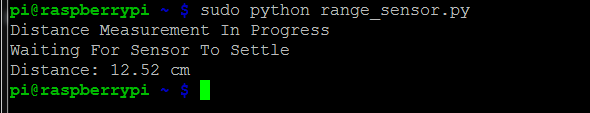
distance = elapsed \* 34000

distance = distance/2

print "Distance:%lf" % distance

GPIO.cleanup()

I saved this program as range\_sensor.py and when I run this program it shows up the distance calculated. The sensor will settle for a few seconds, and then record the specific object distance!



1. **Application:**

* **I measured the distance (water level) every 5 minutes (using cron job)**

**CRON AND CRONTAB**

Cron is a tool for configuring scheduled tasks on Unix systems, used to schedule commands or scripts to run periodically and at fixed intervals; tasks range from backing up the users' home folders every day at midnight, to logging CPU information every hour.

The command crontab (cron table) is used to edit the list of scheduled tasks in operation, and is done on a per-user basis; each user (including root) has their own crontab.

**EDITING CRONTAB**

I run crontab with the -e flag to edit the cron table:

crontab –e

The layout for a cron entry is made up of six components: Minute, hour, day of month, month of year, day of week, and the command to be executed.

# m h dom mon dow command

# \* \* \* \* \* command to execute

# ┬ ┬ ┬ ┬ ┬

# │ │ │ │ │

# │ │ │ │ │

# │ │ │ │ └───── day of week (0 - 7) (0 to 6 are Sunday to Saturday, or use names; 7 is Sunday, the same as 0)

# │ │ │ └────────── month (1 - 12)

# │ │ └─────────────── day of month (1 - 31)

# │ └──────────────────── hour (0 - 23)

# └───────────────────────── min (0 - 59)

I added this line to crontab:

\*/5 \* \* \* \* /home/pi/lab6/range\_sensor.py

The water level was measured every five minutes after that.

* **Right after the measurement record write (append) a line to a text file called log.txt with timestamp, tab, and value**

For this requirement, I added the following code to the original code:

**#Append lines to log.txt**

**with open("/home/pi/lab6/log.txt","a") as text\_file:**

**text\_file.write("Distance:%1f " % distance )**

**text\_file.write(strftime("%Y-%m-%d %H:%M:%S\n"))**

**text\_file.close()**

#!/usr/bin/python

#Author: Riz

#Date: 19/11/2015

import datetime

import time

import RPi.GPIO as GPIO

import cookielib

import socket

import urllib

import urllib2

GPIO.setmode(GPIO.BCM)

GPIO\_TRIGGER=17

GPIO\_ECHO=27

print "Ultrasonic Distance Measurement"

GPIO.setup(GPIO\_TRIGGER,GPIO.OUT)

GPIO.setup(GPIO\_ECHO,GPIO.IN)

GPIO.output(GPIO\_TRIGGER,False)

#time.sleep(0.5)

#while 1:

#Send 10us pulse to trigger

GPIO.output(GPIO\_TRIGGER,True)

time.sleep(0.00001)

GPIO.output(GPIO\_TRIGGER,False)

start=time.time()

while GPIO.input(GPIO\_ECHO)==0:

start=time.time()

while GPIO.input(GPIO\_ECHO)==1:

stop=time.time()

#Calculate pulse length

elapsed=stop-start

#Distance pulse travelled in that time is time

#Multiplied by the speed of sound (cm/s)

distance=elapsed\*34000

distance=distance/2

print "Distance:%1f" % distance

from time import gmtime, strftime

print strftime("%Y-%m-%d %H:%M:%S")

**#Append lines to log.txt**

**with open("/home/pi/lab6/log.txt","a") as text\_file:**

**text\_file.write("Distance:%1f " % distance )**

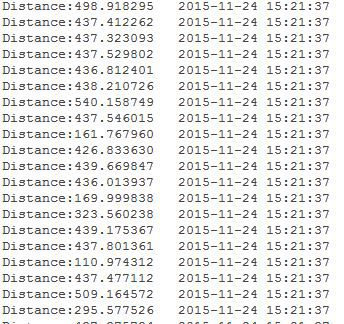
**text\_file.write(strftime("%Y-%m-%d %H:%M:%S\n"))**

**text\_file.close()**

#time.sleep(5)

#GPIO.cleanup()

So the log.txt file looks like this:

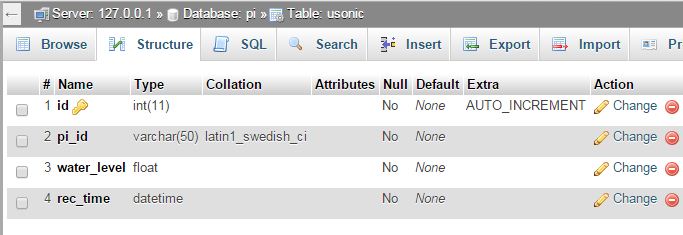


* **After writing (appending) to the text file I sent the value to the localhost server using the API I developed for CE4001 (Software Lab).**

API, an abbreviation of application program interface, is a set of [routines](http://www.webopedia.com/TERM/R/routine.html), [protocols](http://www.webopedia.com/TERM/P/protocol.html), and tools for building [software applications](http://www.webopedia.com/TERM/A/application.html).

The API specifies how software components should interact and APIs are used when programming graphical user interface ([GUI](http://www.webopedia.com/TERM/G/Graphical_User_Interface_GUI.html)) components.  A good API makes it easier to develop a [program](http://www.webopedia.com/TERM/P/program.html) by providing all the building blocks. A [programmer](http://www.webopedia.com/TERM/P/programmer.html) then puts the blocks together.

The API I developed for my localhost server (usonic.php) is basically a database-driven API. The MySQL database ‘pi’ was created containing one table ‘usonic’ as the following:



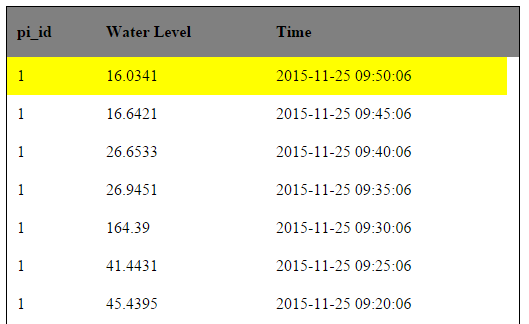
Google Chrome has a REST/HTTP API Client called ‘DHC’. It’s easy to use and configurable. I used DHC as the API client.

The following PHP files were created in order to develop the API:

**usonic.php:** I developed this REST API called “usonic.php” which accepts two POST parameters “pi\_id” and “water\_level” witht the following requirements:

* + Content-Type: application/x-www-form-urlencoded
  + Check that “pi\_id” and “water\_level” are not empty or NULL. If one or both them is empty or NULL then echo “Failed: input mismatch” otherwise proceed to next step.
  + Insert a record “pi\_id”, “water\_level”, and “rec\_time” into the database
  + If successfully add a record then echo “Success” otherwise echo “Failed: cannot add a record”

**monitor.php:** I developed a web page called “monitor.php” to list the records in the database in descending order as shown below:



I just modified the Python code I wrote to measure the distance (water level) and added the following lines at end of the code:

url='http://192.168.0.60:8080/usonic3/usonic.php'

#localhost IP 192.168.0.60 and 8080 is port no.

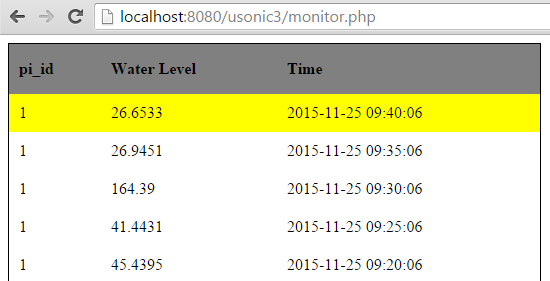
data=urllib.urlencode({'pi\_id' : '1',

'water\_level' : distance})

content=urllib2.urlopen(url=url, data=data).read()

print content

I run the program distance2.py and check the output every five minutes (cron job) browsing to my localhost server. It’s working perfectly.



**Lab 7**

**(This is a revised Lab 6)**

Same objective and apparatus list.

In this lab, I sent the measured value to Dr. Anand Dersingh's website: http://168.120.143.220/pi/monitor.php

I have given an specific pi\_id=1. Using this pi\_id my data can be tracked into the website.

I just modified the previous code as following and used cron job to send the measured value every five minutes:

#!/usr/bin/python

#Author: Riz

#Date: 19/11/2015

import datetime

import time

import RPi.GPIO as GPIO

import cookielib

import socket

import urllib

import urllib2

GPIO.setmode(GPIO.BCM)

GPIO\_TRIGGER=17

GPIO\_ECHO=27

print "Ultrasonic Distance Measurement"

GPIO.setup(GPIO\_TRIGGER,GPIO.OUT)

GPIO.setup(GPIO\_ECHO,GPIO.IN)

GPIO.output(GPIO\_TRIGGER,False)

#time.sleep(0.5)

#while 1:

#Send 10us pulse to trigger

GPIO.output(GPIO\_TRIGGER,True)

time.sleep(0.00001)

GPIO.output(GPIO\_TRIGGER,False)

start=time.time()

while GPIO.input(GPIO\_ECHO)==0:

start=time.time()

while GPIO.input(GPIO\_ECHO)==1:

stop=time.time()

#Calculate pulse length

elapsed=stop-start

#Distance pulse travelled in that time is time

#Multiplied by the speed of sound (cm/s)

distance=elapsed\*34000

distance=distance/2

print "Distance:%1f" % distance

from time import gmtime, strftime

print strftime("%Y-%m-%d %H:%M:%S")

with open("/home/pi/lab6/log.txt","a") as text\_file:

text\_file.write("Distance:%1f " % distance )

text\_file.write(strftime("%Y-%m-%d %H:%M:%S\n"))

text\_file.close()

#time.sleep(5)

#GPIO.cleanup()

**url='http://168.120.143.220/pi/usonic.php'**

**#values={'pi\_id':'1','water\_level':distance}**

**data=urllib.urlencode({'pi\_id' : '1',**

**'water\_level' : distance})**

content=urllib2.urlopen(url=url, data=data).read()

print content

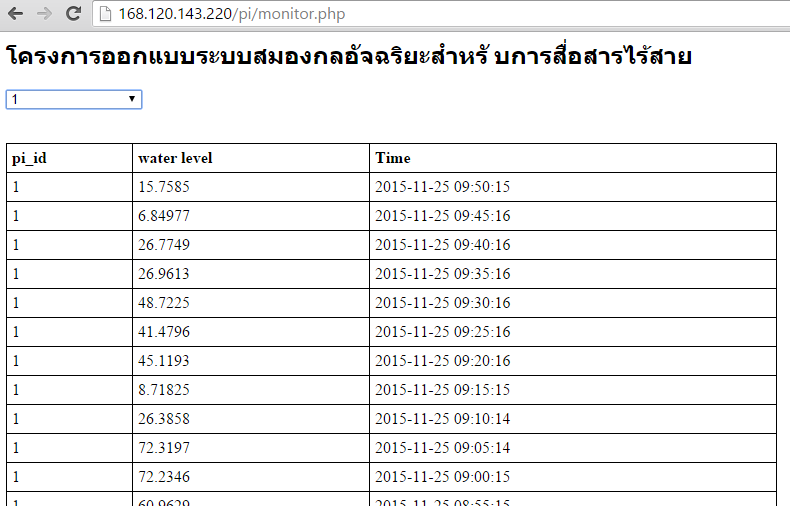
Look carefully at the bold texts that I mentioned above. The url contains the website IP address and the API (usonic.php). Every 5 minutes data is being sent to this server via usonic.php.

I modified the crontab -e and added the following line:

\*/5 \* \* \* \* /home/pi/lab6/distance.py

Now every five minutes measured data will be sent to the website's server.

Here's the most recent screenshot of the website:



THE END