Formal Project Documentation #1

Embedded Linux

Erin Rivera, Mahmoud Jawharji, Ivan D'Alcantara, Fabricio Goncalves, Kevin Han, Theodore Thueson

April 6 2015

Project Description

The project scope is to create a buoy that will hold the raspberry pi, along with other materials necessary to be able to record temperatures at different depths using temperature sensors. The buoy will hold the pi and be connected to multiple temperature sensors at different depths. It will also be connected to a motor that will control our pulley system. Our pulley system is being created as a way for us to elevate and descend our temperature sensors into the water. For example having the first sensor reading the temperature 1 foot below sea level, and the second reading it from 2 feet below sea level and so on. After the pulley lowers the sensor to its location it will remain there for however long is needed, record the data, then be elevated back into its original position. It will be recorded at fixed time intervals of about 1 hour or so. After the data is recorded it will be send the information to the web server. The web server will be able to map out the data on a graph and display it.

Project Goals

The goal of our project is to be able to create a buoy that will record temperature of water. Another one of our goals is to create a design in our buoy with our raspberry pi, temperature sensors, motor, and pulley system where the design will have a fully working waterproof system and be able to monitor and record temperature at multiple depths. The final goal includes being able to have our raspberry pi record the data, save the data, then send the to a web server. Along with creating the web server, we will also need to have the web server receive the data and graph it in a user-friendly form where anyone will be able to monitor our data.

Implementation Plans

Components

Hardware

Buoy Design

The buoy has been built using a plastic case (yellow), which contains a box inside it. All the spaces that are between the box and plastic case are covered with three layers of two types of heavy-duty tapes (black and brown), which hold the whole structure as one solid piece, see Figure 1a & 1b.

Figure 1a

Figure 1b

Electronic Components

The next step is to get all the electronics parts, (Breadboard, Raspberry pi B+, DC & Stepper Motor Hat for Raspberry pi, Stepper Motor, waterproof DS18B20 Digital temperature sensor, and jumper wires), and connect them together, see Figure 2. Note, two holes have been made, one on the side to insert the temperature sensor wire, and the other one in the middle in order to connect the motor to a “Pulley” from the other side. This will help to manage and control the

depth of the temperature sensor, in which it will pull it either upward or downward, see Figure 3a & 3b.

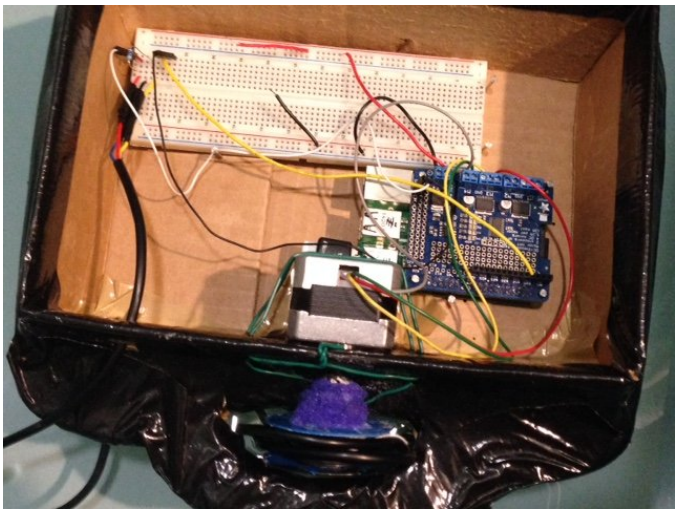

Figure 4

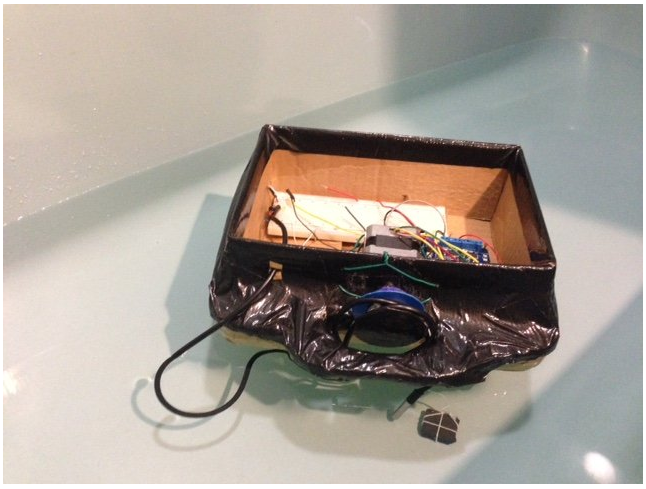

Figure 2a

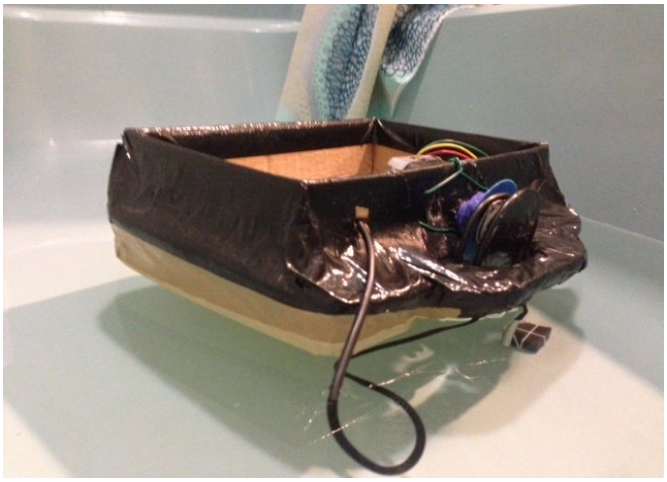

Figure 3b

- Connecting the Temperature Sensor to the Raspberry pi

  In order to connect the DS18B20 temperature sensor, we followed the same steps as in Figure 4, which from the tutorial on adafriut.com. However, we did not use the extendable wire for the GPIO since we are fully using the GPIO for the Stepper Motor Hat.

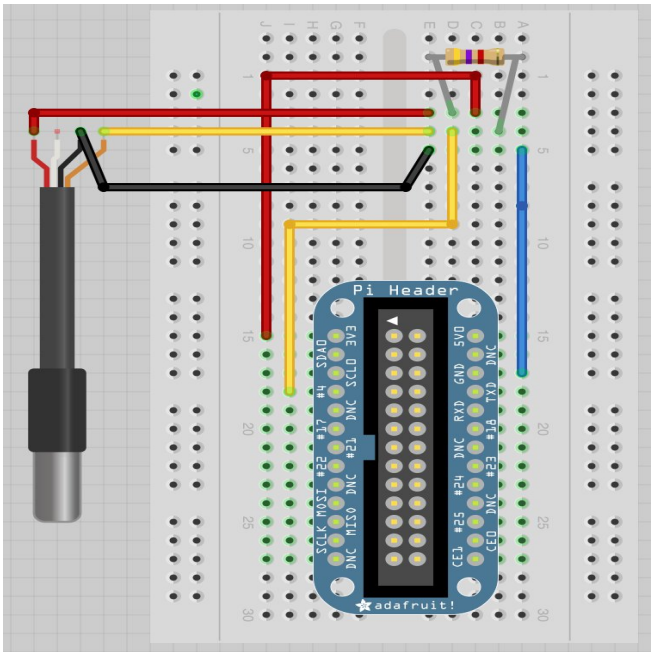

Figure 4

Then we developed the following code (monitor.py), which will capture the temperature and the current time and save them inside the “templog.db” file, see Figure 5.

```
#!/usr/bin/env python

from datetime import datetime
import sqlite3 as mydb
import os
import time
import glob


# main function
# This is where the program starts
def main():

    tempfile = open("/sys/bus/w1/devices/28-000006574c70/w1_Slave")
    tempfile_text = tempfile.read()
    tempfile.close()
    tempC = float(tempfile_text.split("\n")[1].split("t=")[1])/1000
    tempF = tempC*9.0/5.0+32.0

    con = None

    con = mydb.connect('templog.db')
    cur = con.cursor()
    now = datetime.datetime.now()
    cur.execute("INSERT INTO Temp VALUES(?, ?)", (now, tempC))
    con.commit();
    con.close()

if __name__=="__main__":
    main()
```

Figure 5

- Connecting the Stepper Motor to the Raspberry pi

In order to connect the Stepper Motor, we first need to attach the Stepper & DC Motor Hat, which will make it easier for us to control the Motor from the python script. We followed the tutorial on the Adafruit.com to make the correct connection, see Figure 6.

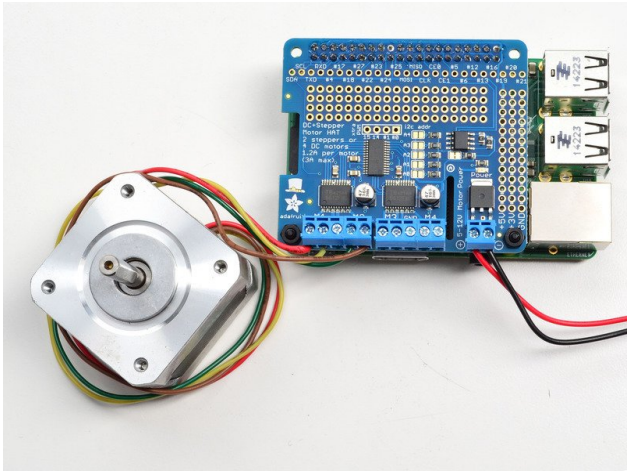

Figure 6

Then, we developed the code for the stepper motor, which will control the movement of the Stepper Motor, FORWARD or BACKWARD. Also, since we have attached pulley on the other side of the stepper motor, the pulley will then control the movement of the string that is attached to the temperature sensor, which will move either UPWARD or DOWNWARD, see Figure 2a & 2b. However, for the purpose of our project, we need the string, or the temperature sensor, to go down on three fixed depths (7 cm, 14 cm, 21 cm). First, the temperature will go down 7 cm and them it will stop, at this point we will call the (monitor.py) file to capture the current temperature and current time and save them on “templog.db”. Then, the temperature will wait for little while it collect the right data and then will move down 7 cm more, which will be at depth 14 cm and then stop. We do the same thing as we did in the first stage, and then continue to 7 cm more to reach at depth 21 cm and then stop. After we collect the data (temperature and current time), we scroll back up to the starting point, in which the temperature sensor will move up 21 cm until it reaches the starting point. Then we wait for fixed time then repeat the same steps to get an update, see Figure 7.

```
#!/usr/bin/python
from Adafruit_MotorHAT import Adafruit_MotorHAT, Adafruit_DCMotor, Adafruit_StepperMotor

import time
import atexit

# create a default object, no changes to I2C address or frequency
mh = Adafruit_MotorHAT()

# recommended for auto-disabling motors on shutdown!
def turnOffMotors():
    mh.getMotor(1).run(Adafruit_MotorHAT.RELEASE)
    mh.getMotor(2).run(Adafruit_MotorHAT.RELEASE)
    mh.getMotor(3).run(Adafruit_MotorHAT.RELEASE)
    mh.getMotor(4).run(Adafruit_MotorHAT.RELEASE)

atexit.register(turnOffMotors)

myStepper = mh.getStepper(200, 2)   # 200 steps/rev, motor port #1
myStepper.setSpeed(200)         # 30 RPM

while (True):

    print("Double coil steps")
    myStepper.step(300, Adafruit_MotorHAT.BACKWARD,  Adafruit_MotorHAT.DOUBLE)
        # Measure Tempreture on Depth = 7 cm
        # execfile("monitor.py")
        time.sleep(5)
        myStepper.step(300, Adafruit_MotorHAT.BACKWARD,  Adafruit_MotorHAT.DOUBLE)
        # Measure Tempreture on Depth = 14 cm
        # execfile("monitor.py")
        time.sleep(5)
        myStepper.step(300, Adafruit_MotorHAT.BACKWARD,  Adafruit_MotorHAT.DOUBLE)
        # Measure Tempreture on Depth = 21 cm
        # execfile("monitor.py")
        time.sleep(5)
```

Figure 7

Reading component

Server communication component

Overview

The data read from the sensor are stored locally, so we need to send this data to the server when an internet connection is available. This component is responsible for checking for internet connection, reading the data from the local database and send it to the server. As soon as a internet connection is available, the component will read all the data stored locally and format it to a defined JSON format. The component them makes a request to the web server passing the JSON containing the data. The webserver gets the data and stores in the database sending back a response code.

Completed work

The method to send the data from raspberry will be by POST http request. The software to accomplish this is done. It basically reads the data from the SQLite on the raspberry pi, enfold it in a JSON format and send to the webserver that will receive and manipulate it. The code below was used to test this component.

```
#!/usr/local/bin/python
# -*- coding: UTF-8 -*-

import os
import time

#Http request and json Libraries
import httplib
import json

#Sqlite Library - Accessing data in the raspberry pi
import sqlite3 as db
import sys

def getData():
```

```
        con = db.connect('temperature.db')

        with con:
        cur = con.cursor()

        cur.execute("""SELECT date, roomTempC FROM TempData""")

        data = cur.fetchall()

        return data

rows = getData()

httpServ = httplib.HTTPConnection("cs.newpaltz.edu")
httpServ.connect()

js = json.dumps(rows)

httpServ.request('POST', "/~fernandi2/playground/t1.php", js)

response = httpServ.getresponse()

httpServ.close()

print response.read()
```

To be done

The next part of this component is to define the JSON data format to be sent. After it is defined, the code will be changed regarding the new data format.

The script to test wifi connection also needs to be done. As described above, its function is to test network connection and once a connection is established, run the python code to send the data.

Web site component

Overview

The website is used to display the data in form of a chart. It has a backend database with the data sent from the PI by the communication component. The web site gets the data from the database and draw charts for displaying.

Completed work

The server database was designed and created. We used MySQL Workbench for creating the database model. It was designed in such a way that it is highly extensible when new PIs and sensors are added to the system.

For drawing the charts we are using Google Charts. It is a powerful chart plotting tool by google. It is free, reliable and have many functionalities. The webpage we be also built on top of bootstrap, so that it has a solid css base and helps on the arrangement of the charts in the page, in addition to mechanisms to support mobile devices. We created a simple web page to learn how to use the google charts api, and have a basic filling of the charts arrangement on the website.

To be done

We need to develop the php code for getting the data from the webserver's database and displaying on the chart. Since the data is dependant of the other components, we need to create some dummy data to test the website. We need also to design the overall layout of the page for displaying the charts.

Project aspects

Next Goals

Member Contribution

Fabricio - He will be focusing on programming the raspberry pi to have the temperature sensors read the data and transmit it to a web server. Also making sure to include important details such as temperatures in fahrenheit and celsius, and details about the date and times of when the data is recorded. He will also need to specify which sensor is recording at what time and at what depth according to the pulley system.

Ivan - He will be focusing on programming the web server that will be able to receive data from our raspberry pi. The web server he is creating will also plot the data in a user-friendly graph where anyone could see. He will also record the data on when it was taken and which sensor it was taken from at that depth.

Erin - He will be focusing on hardware and wiring of the parallel sensors. Knowing which ports need to be connected where. Also thinking of the idea of having a similar representation of different sensors being implemented into the design and knowing how to work with them.

Mahmoud - He will be focusing on the architecture of the buoy, and being able to create a design that will house the materials needed to float on top of water. Make sure that the design covers anything that needs to be kept from water. Also having a location on the buoy where we will be able to hang the wires and sensors from the buoy without water entering. He will also need to keep in mind the fact that we will have a motor and pulley system attached to the buoy and figure out the location of where everything needs to go.

Theodore and Kevin- They will be focusing on the hardware and architecture of being able to implement a pulley system for the different sensors. The idea is to have a system where sensors will be lowered into the water at a specific depth, record the data after a short period of time, then have the pulleys return to their original location after collecting the data. They will need to keep in mind that they will be doing this working with the pi and the stepper motor.