Buoy Project

By:
Phelipe Fernandes
Anthony Giordano

Project Description

-Develop a system that can monitor temperature at different depths in a lake at regular intervals.

-The system should archive the data locally if no network connection is available. If a network connection is available, transmit the data to a central repository.

-Develop a web interface that can display the data collected at the central repository. Start with temperature sensor. Keep it simple, easily deployable and sturdy

Project Goals

Materials/ Software

Our goal is free, open source software,
Readily available, inexpensive hardware

Hardware:

- Raspberry Pi running Raspbian
- DC motor
- Low signal relay
- Mosfet
- Breadboard
- Batteries
- Temperature Sensor

Software

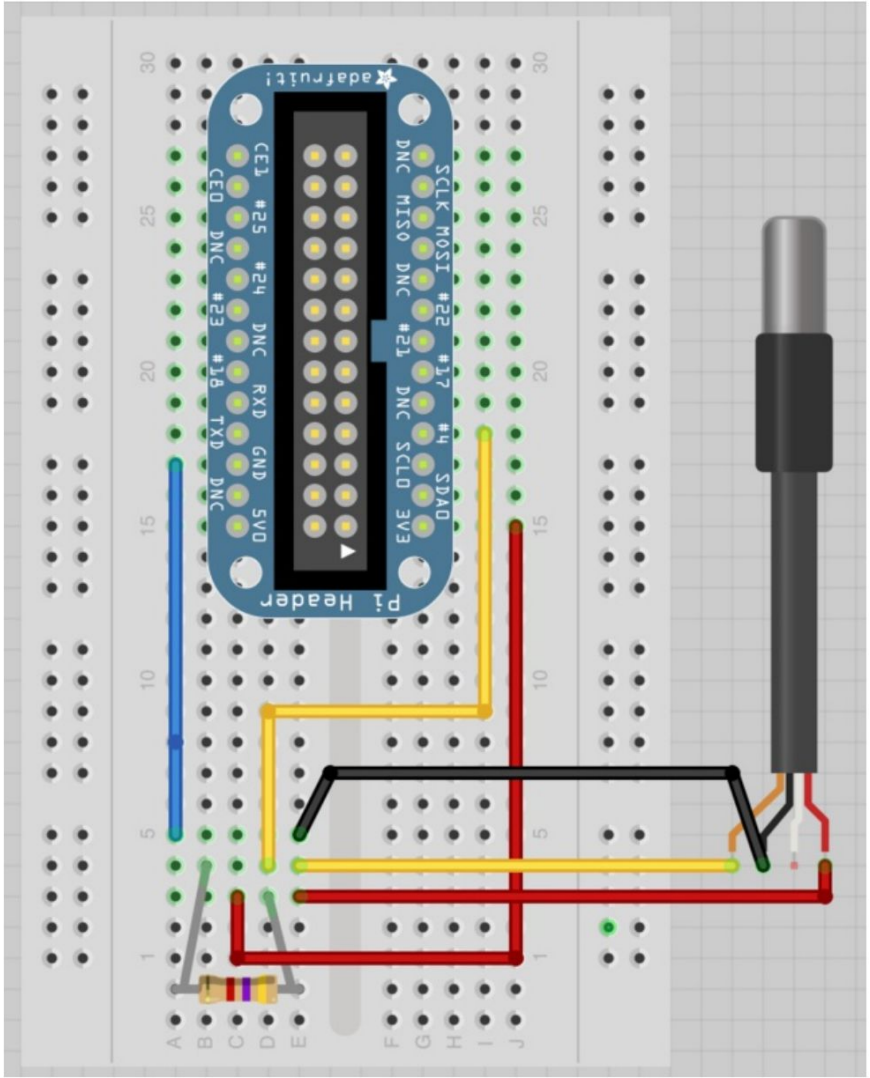
1. Flask - Handles HTTP requests in python environment ran via Virtualenv - Open source
2. JQuery - Retrieves data from web database (Hosted by SUNY New paltz) - Open source
3. Chart.js for serving the data in line graph form, powered by Javascript - Open source
4. Bootstrap for designing the webpage for all platforms - Open source

5. SQLite 3 - Used for database - Public Domain and free

Hardware Implementation

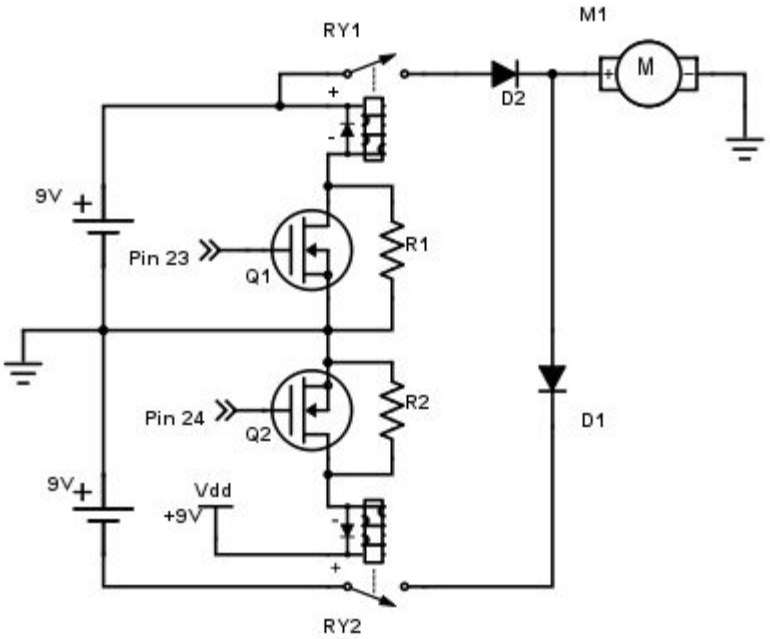
Getting the data

- Digital sensor
- One Wire protocol



Moving the sensor

- Two relays
- Only one DC Motor
- Two batteries



Saving the Data

- Sqlite3
- Copy DB to JSON file
- Send to cloud

```
#this function saves the temperature and the respective date/time
#and depth in the database
def saveTemp(date, depth, tempC, tempF):
    conn = sqlite3.connect("buoy.db")
    c = conn.cursor()
    c.execute("INSERT INTO buoy VALUES ('%s', '%s', '%s', '%s')"
        % (date, depth, tempC, tempF))
    conn.commit()
    conn.close()

#this is the main function. This function calls the others in order
#to make the measurements. It measures the temperature in the surface
# (position 0) and save it in the save it in the database. Then the
# function does the same for the positions 1 and 2
def controller():
    if getPosition() != 0:
        setPosition(0)
    saveTemp(getDateTime(), getPosition(), readTemp(0), readTemp(1))

    setPosition(1)
    time.sleep(DELAY_TEMP)
    saveTemp(getDateTime(), getPosition(), readTemp(0), readTemp(1))

    setPosition(2)
    time.sleep(DELAY_TEMP)
    saveTemp(getDateTime(), getPosition(), readTemp(0), readTemp(1))

    setPosition(0)
```

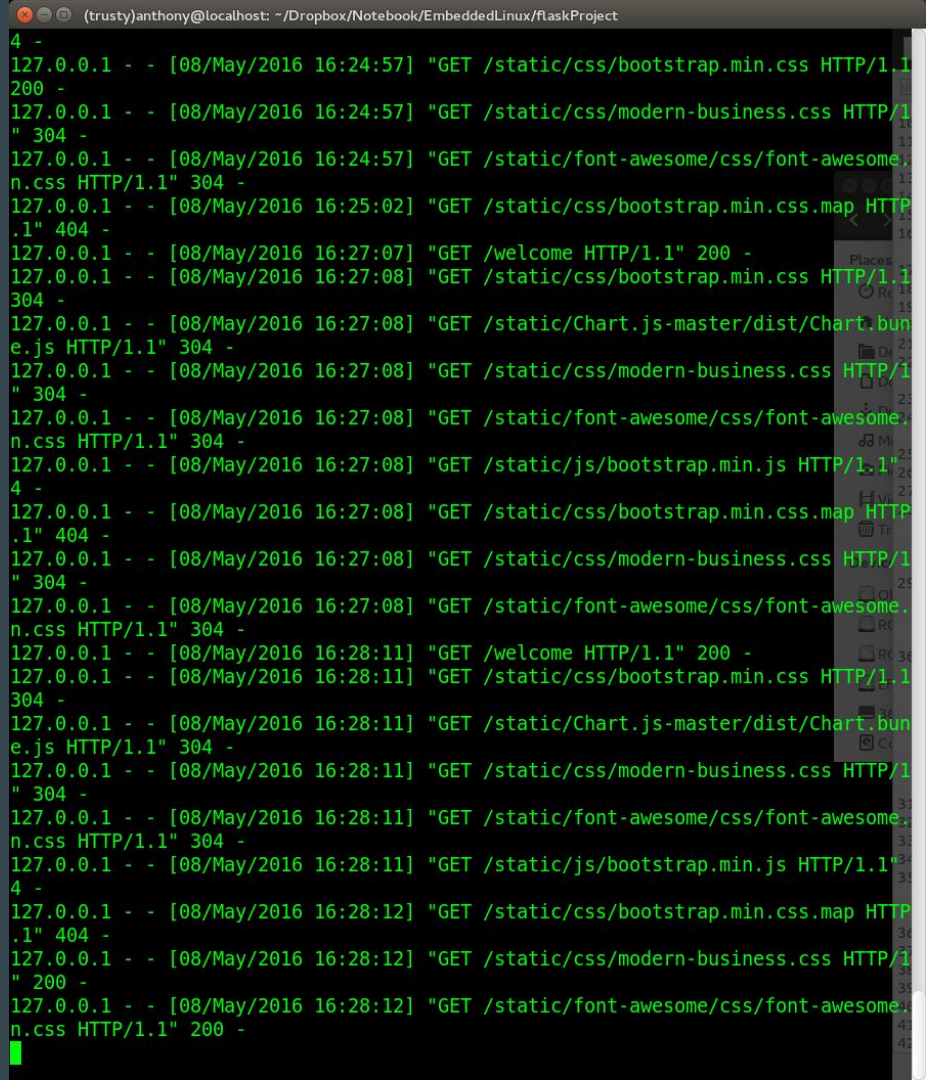
Software Implementation

Software Implementation

Back end

Flask takes HTTP requests and is programmed to serve an HTML page, with dependent files such as CSS, JS, Chart.JS.

For demo purposes, this is done on a local network, should you want to expand to a live website, with admin network access, just forward the port specified by Flask to the local IP of the hosting device. By default this is port 5000.

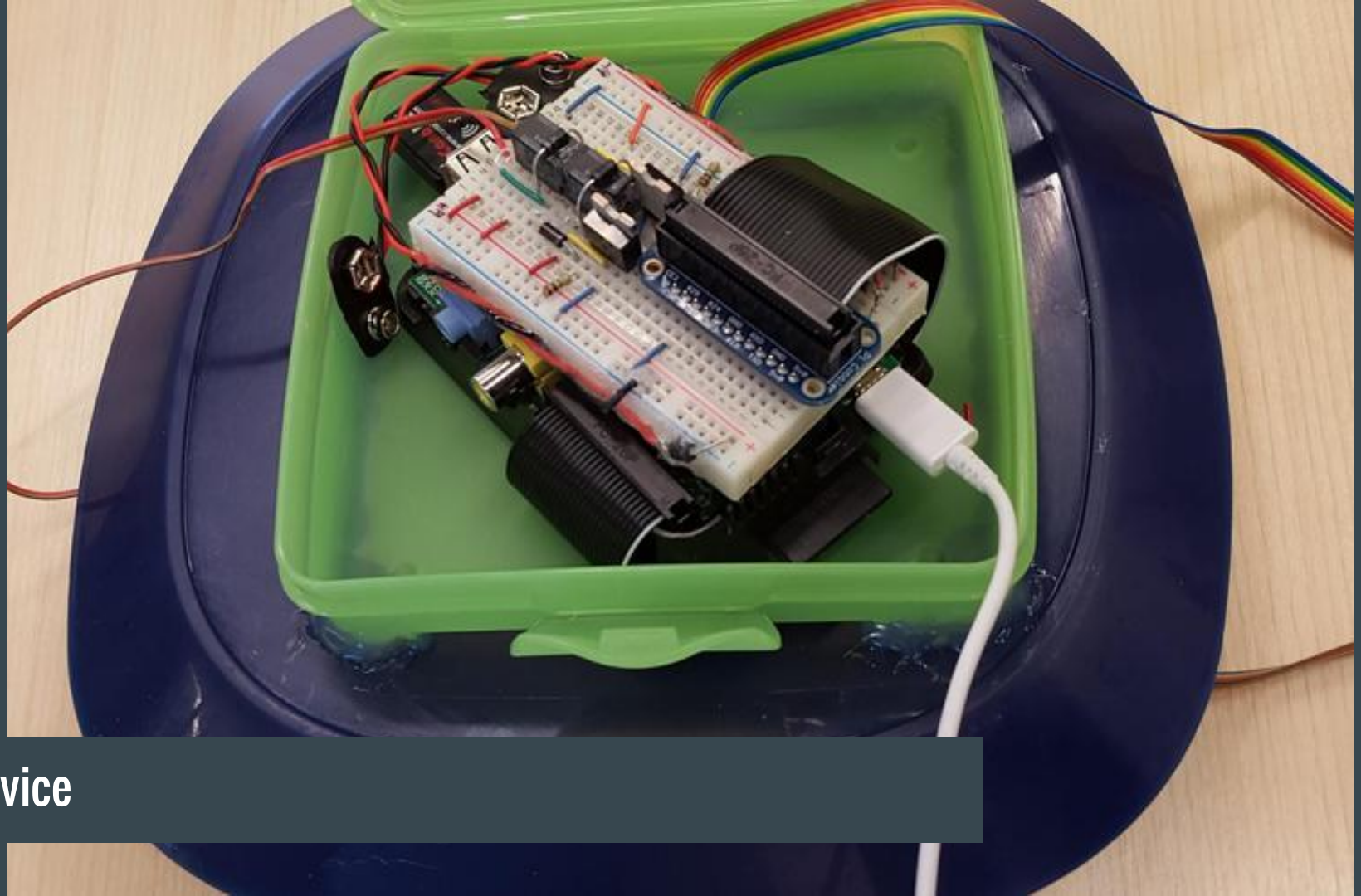


Software Implementation

Front End

On top of HTML and CSS, a JQuery.get request retrieves the data in array form from the SUNY New Paltz server (Which is synced to the device periodically, the school just provides a free web host). A JS function then parses it into arrays of each element, which are then passed to Chart.js to use as input.


```
107         //GET request
108         $.get('http://www2.newpaltz.edu/~fernandp5/buoy/buoy.txt', function(data){
109                 var content = data;
110                 var entryArr = content.split("]")
111                 //Removes unnecessary characters from each array element
112                 for(var i = 0;i < entryArr.length;i++){
113                         entryArr[i] = entryArr[i].replace( '[','' );
114                         entryArr[i] = entryArr[i].replace( '[','' );
115                         entryArr[i] = entryArr[i].replace( /"/g,"" );
116                         if(i != 0){
117                                 entryArr[i] = entryArr[i].replace( ' ','' );
118                                 entryArr[i] = entryArr[i].replace( /,/,'' );
119                         }
120                 }
121                 //Sorts into array of each element type
122                 var tempArr = new Array();
123                 var dateArr = new Array();
124                 var depthArr = new Array();
125                 var tempFArr = new Array();
126                 var tempCArr = new Array();
127                 for(var i = 0;i < entryArr.length;i++){
128                         tempArr = entryArr[i].split(",");
129                         for(var x = 0;x < 4;x++){
130                                 if(x==0){
131                                 dateArr.unshift(tempArr[x]);
132                                 }
```



Device

Conclusion

Temperature and/or acidity data can be recorded for research or academic purposes on a very low budget with full control of the hardware and software using a Raspberry Pi as a mobile, low power embedded device with strictly free software and moderate programming.

For our source code and extended documentation, see:

https://github.com/n02810216/ELSpring2016

https://github.com/N03324541/ELSpring2016