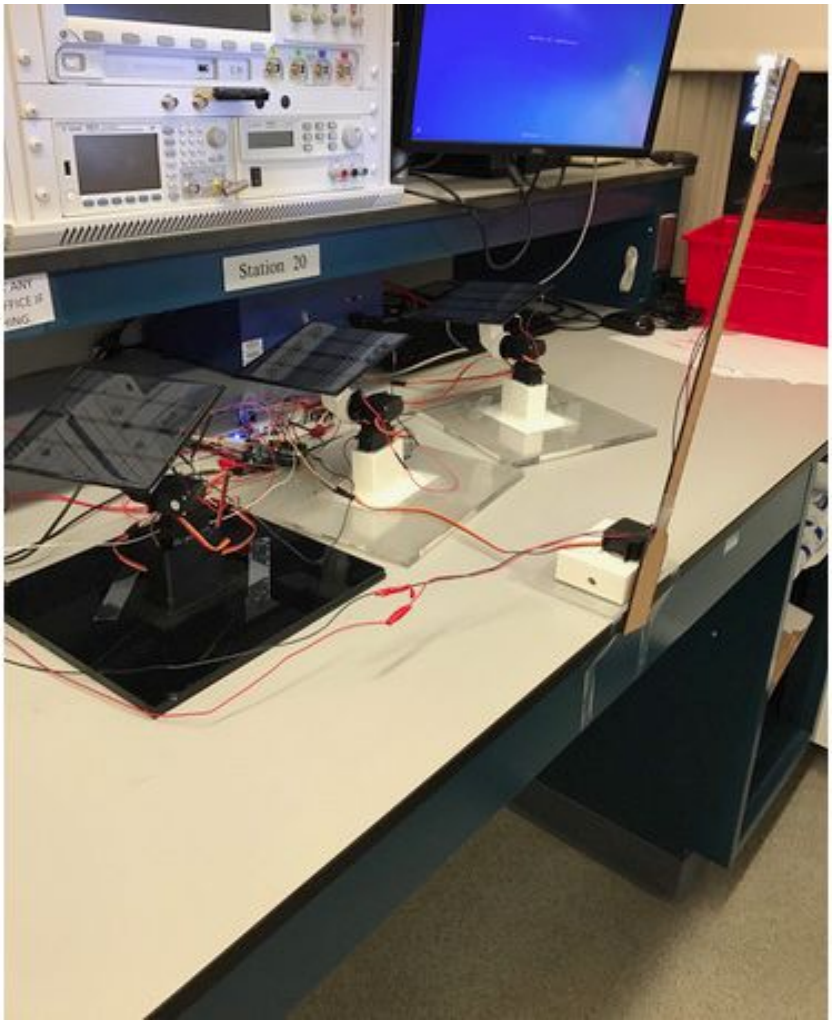
ESE 350
FINAL PAPER

MAY 5, 2017

"Sunny Energy"

Yuxin Pan, Ece Sahin



1. Introduction

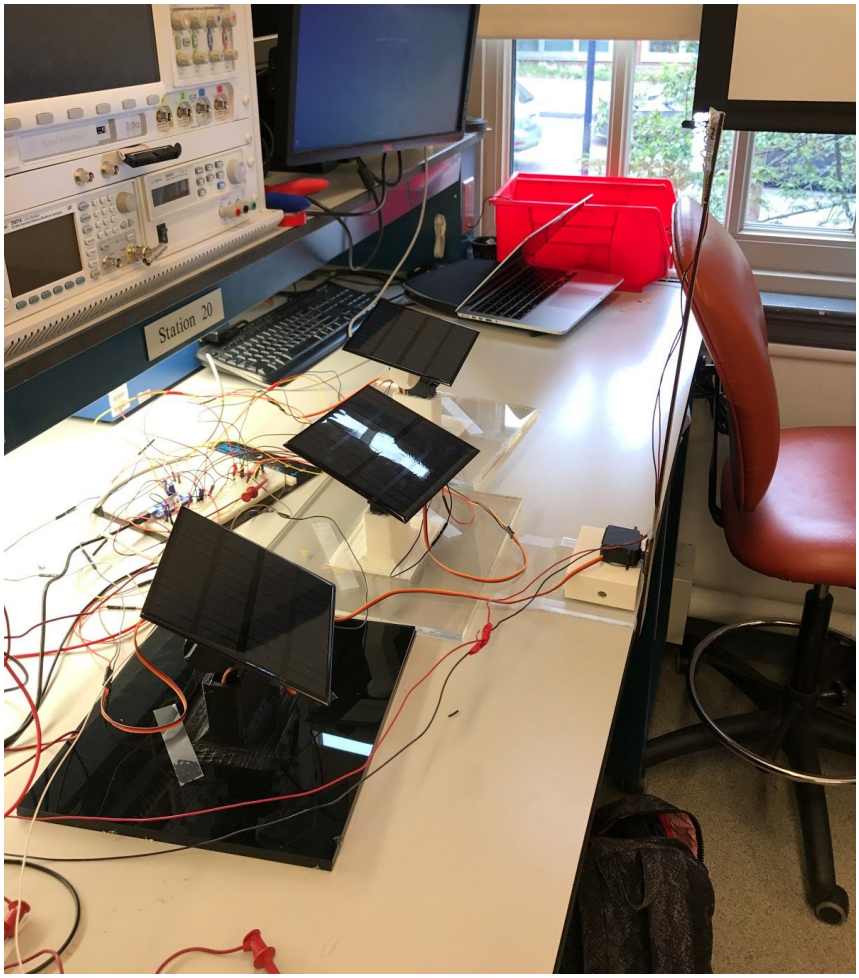
Motivation:

Solar power is the new rising trend in energy industries and currently there are a lot of research and development process being done to optimize the power output. Some optimize the solar energy capture with shape design approach. We wanted to design a setup that has a moving structure, in both horizontal and vertical plane using 2 standard servo motors that can turn 180 degrees in each plane. This as we guessed increased the voltage output. The non rotating solar panel average voltage output was 3.2 V whereas rotating solar panel reading on average was as high as 4.9 V which yields about 50 % more power output.

Given the growing trend in IOT device design we wanted to implement a Web App feature to our project. We designed our own Web App for that reason to keep track of the solar power that is produced and to have a remote control capability. A major advantage of having our own website was that we have now the full control over the system and thus our system is not dependent on any other 3rd party Web App suppliers. The data and the serial connection is all managed by us reducing the upfront cost and dependencies significantly.

Objective:

The purpose of this project was to create an efficient way to generate electric energy using solar power by adding mobility to solar panels combined with an IOT feature that gives the user control over the system.



The solar panel system (with artificial sun at edge of the table)

Hardware elements and pieces used in the project:

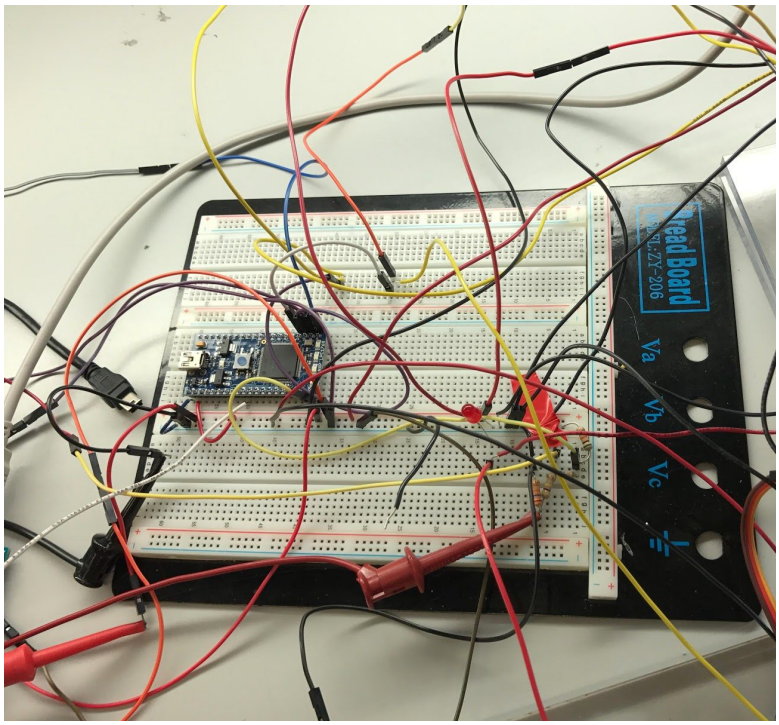
Solar Farm:

3 X Solar Panel 12V- D145X145-3
1 mbed LPC1768
USB Cable
6 X Standard Servo
3 X 3D printed custom design rotation arm
3 X 3D printed custom design base for the mounted servos
6 X Laser cut frames to fit the bases
Resistors 27K, 50K, 100K, 500K
Solderless Breadboard
5V power supply for the Mbed
Beaglebone black wireless

Artificial Sun:
Arduino Uno
USB cable
9x1K Ohm Resistor
breadboard
⅛” MDF Laser cut arm (custom design)
3D print platform
9x3 White LEDs
15 V power supply

For both:
Wires
Hot Glue
Solder

Diagram of the mbed LPC1768 circuit in development:



2.Timeline

Initial Iteration:

Using Mbed we created the first mounted unit of 2 servos, double plane rotating unit to be used in the project in order to have moving solar panels. We also tested our solar panel voltage outputs.

Second Iteration:

We created the 3D designed joint and the base using Solidworks. We assembled the solar panel, joint, servos and base by using a hot glue. After assembling we created our initial algorithm that makes it possible for us to keep track of light direction. We were able to create our first functional rotational solar panel unit that could rotate and collect voltage values to create a linear regression and to find the servo angle that yields maximum power.We also started making a research on adding a communication component to our project. We collected some potential ideas of using Mbed lora communication sensor, Raspberry pi 3, Arduino MKR1000 and Beaglebone black wireless.

Third Iteration - Baseline Demo:

By the time we had our baseline demo with the professor, we were able to create 1 master and 1 slave solar panel units. Master solar panel was giving the servo angle data to the slave unit. We also added a feature that enabled us to see the master solar panel stopping at the angle that yields maximum power output for 1 minute then start searching again. Professor suggested that we should make our max voltage search algorithm more efficient since previously our unit was turning in both planes at a unit time to search for the optimum angle. After the demo we worked on optimizing the algorithm in a way that enabled us to search each plane at a time; first the horizontal plane search and then the vertical plane.

Fourth Iteration - Final Demo:

As the final iteration we were able to get 1 master and 2 slave solar panel units working. We also created a Web App that has a serial connection to our solar farm and has a remote control capability. The Web App is totally independent to any third party API, which means all the components in the Web App are coded by ourselves. The Web App lets the user to retract the solar panels at once with a click. This is designed to let the users control the farm under severe weather conditions. Real time data can be shown on the Web App so the user will be aware of the condition the solar panel currently is. Total energy collection on the web page indicates how much energy it has collected since reset for research and development purpose.

We also laser cut base structures that made our solar panel structures more stable. We also fixed the algorithm to make the panels move more stable way. We fixed the delay durations to reduce the oscillations. We also designed an Artificial sun unit that is controlled by Arduino Uno, which has 27 white LEDs. The LED unit is attached to a laser cut rod that is 16" tall, which is attached to a standard servo motor that rotates 180 degrees simulating the rotation of sun. Every 2 seconds represented 1 hour in our demo so that the demo does not take too long. In normal circumstances after the master solar panel finds its ideal maximum power yielding output, it stops rotating for 1 hour and then starts rotation, but in order to make the demo faster we made the solar panel stop for 1 minute.

3. Solar Panel Algorithm Logic

One solar panel is in charge of searching for the optimum sunlight. Upon activated, it rotates 180 degrees to determine the best angle to start the search by measuring the output voltage from the solar panel, the angle that maximum voltage occurred is the angle to start the next process. From that point, it rotates a small angle (e.g. 5 degree) to either direction to test if the sunlight is stronger at that direction. Due to the error and precision by analog to digital converter and the error by the servo, reading only the start point (0 degree) and end point (5 degree) is not sufficient to provide a guide to the optimal direction. Experiment shows that the error generated during the process outweighs the change of the strength of the sunlight. The key to solve the problem is to note that the characteristics of the error is random, and after collecting a large amount of data and going through a proper filter, the trend of the true data can be seen and error will be filtered out.

To implement this concept, two methods are used. First, instead of collecting only at two points (0 and 5 degree), the panel now, for example, quickly stops at 0.5, 1, 1.5, 2 degree to provide plenty of datasets, and although the individual value of these datasets can vary, the trend, or in other words, the slope of the data, can be determined precisely. Since the angle of rotation is small, the change of the strength of the sunlight can be approximated as a linear function, therefore, applying linear regression to the dataset (voltage read by analog pin) to find the slope of the data would be a way to find the trend. Positive slope suggesting that the strength of the sunlight is increasing. Since linear regression takes every voltage value into account, and the error is canceled out during the process, the accuracy is improved significantly. Another method to reduce error is by repeating the analog voltage read process: instead of reading the voltage once at every particular angle, it reads the analog value for ten times. By doing so the error generated by analog to digital converter is minimized. Also, the test shows that more analog reading does not have an significant impact on the processing time.The second method is not able to reduce the error generated by the inaccurate positioning of servo rotation. But by

combining the two methods, the total error is reduced by a large amount, and the final result has met our expectation. The searching process is sensitive to the change of sunlight (or artificial light) and does not have an obvious lag on time.

4. Web App and User Interface

The Web App is completely independent to any third party API. We have coded everything in the app except using an open source Javascript library for plotting the graph. The advantage of coding our own app is that we have total control of the app, which is not subject to any subscription fee or API requests time limitation. Avoid a subscription fee is essential for a startup company to reduce the cost. Also, the app won't be impacted by the performance of third party API (e.g. service downtime of the third party API). Also, customization is made possible by coding our own app. For example, we tried a Matlab based API which only refreshed every twenty seconds. However, with our own app, we can code it to refresh for any given interval.

The Beaglebone black wireless acts as a relay between mbed and cloud (server). It reads the analog value of the solar panel voltage output and send it to cloud in real time. A Python program on the board is also polling the new command from cloud (server) and sends it to mbed via serial communication.

Our own cloud, a server running on Linux is based on KiwiVm and located somewhere in Arizona. The program on the server side to receive and store solar panel data is coded in PHP (io.php, the code is in appendix). This PHP program is also taking command (release or retract solar panel) from user and send the command to client (Beaglebone).

The user interface consists of HTML web page and Javascript that fetches the data and plots the graph. The web page layout is organized in a simple and clear way that won't confuse the users. The Javascript is embedded in the web page, and makes an XMLHttpRequest every second to fetch new voltage data from server, then Chart.js, an open source library, is used to plot the

graph based on the new fetched data. The estimation of the the real time power output and total energy collected is calculated by Javascript and updates when new data comes in. The estimation is calculated based on the voltage over time. The user input command is passed from web page to io.php and sent to client to release or retract the solar panel.

5. Conclusion

Challenges:

Throughout the project we found the algorithm design most challenging to accomplish. The reason for that was Mbed was not a familiar processor for us before and also we had mechanical pieces that we need to take into account in terms of oscillations and movements. Thus finding the optimum algorithm and making sure we spot the most power outputting angle was the most challenging part that we learned through completing the project.

Another biggest challenge was the Web App design for the project initially we looked in various methods to complete this reach goal of our project, such as the MKR 1000, Raspberry Pi3. However none of them worked as we desired. For example Raspberry Pi 3 was not easy to work with because it lacks the analog to digital converter that we need for the power output data representation. So we tried MKR 100 to see if it would work as we desired, and we saw that we were able to extract the analog readings and generate graphs using thingspeak MATLAB graphs. However because this still depends on a n external web service so we looked into ways to generate our own Web App and we were able to do that using Beaglebone Black WiFi. While we were working with the Beaglebone, we had some difficulties because the board was very fragile against any external forces or input voltage. In fact once the board got broken and we needed to fix it again, but in the end it all worked great.

Goals Achieved:

Overall, we were very happy with the end product. It performed as expected and it really creates a unique experience. We reached our baseline goals as well as our reach goals.

1-We were able to design a functional solar panel design that is controlled with an Mbed and which can move in 2 axis rotation to increase the voltage output. (BASELINE) (The non rotating solar panel

average voltage output was 3.2 V whereas rotating solar panel reading on average was as high as 4.9 V which yields about 50 % more power output.)

2- We generated a synchronized set-up of 2 solar panels: one being the master that rotates to spot the best angle the other being the slave to follow the commands given by the master.(BASELINE)

3- We were able create a Web App and a serial communication platform that shows power output (REACH)

4-We also created an artificial sun that simulates the 180 degree rotation of the sun (east to west)around to make our demo more presentable (REACH)

Looking Forward:

Even though we did not have enough time during this final project, we could implement LoRa to our project as we learned in the discussions we had in the class, We could implement that to create the communication between the solar panels in a city setting. In the long run our goal is to have a large network of solar panels that can create an alternative for the current electricity generation methods based on fossil fuels. There will be one master panel that will send the angle data to the slaves around the city through LoRa. The network will also make sure that all of the panels in the city is working. Let's say if the master panel is broken then the next working slave will become the temporary master and will start doing the job that the master one was doing, till the broken one gets fixed. In addition if any of the solar panels are not functioning properly there will be an alert system that will notify the maintenance service to fix the broken one. This will make the maintenance service much easier to control for both the service provider and the client. In addition we are planning on creating a personalized webpage for the solar panel owner to let him/her keep track of their power outputs and give them the chance to turn it on or off. This will give more freedom and better user experience through using an IOT feature.

Because we already have our Web App we are not dependent on an external platform thus we have the ability to design our product in the best way with very little cost.

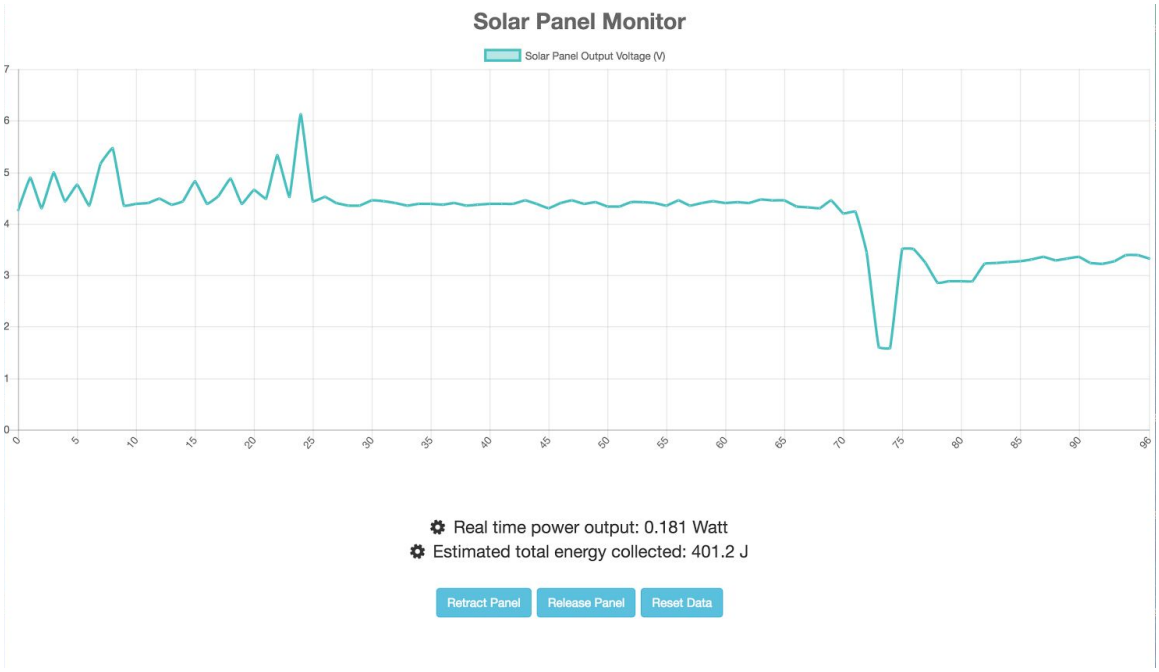
6. References

Open source Javascript plotting graph library Chart.js: http://www.chartjs.org/

Appendix:

1. The Webpage Screenshot:

Address link for the online web app: apps.panyuxin.com/test/plot/



2. Project Mbed Code:

```
#include "mbed.h"
#include<iostream>
#include<cmath>
#include <stdlib.h>
#include <math.h>
#define REAL double

DigitalOut led1(LED1);
DigitalOut led2(LED2);

PwmOut servo(p21);  //servo 0.0005 to 0.002
PwmOut servo2(p22); //servo 0.0005 to 0.002
PwmOut servo3(p23); //servo 0.0005 to 0.002
PwmOut servo4(p24); //servo 0.0005 to 0.002
AnalogIn voltage(p20);
Serial pc(USBTX, USBRX); // tx, rx
Serial device(p9, p10);  // tx, rx

float
lastVoltage=0,compareV=0,offsetTest,currentVoltage,lastVoltage2=
0,currentVoltage2,offset=0.0005,offset2=0.0005,offsetOld=0.0,off
set2Old=0.0,val1,val2;
int lastCommand=1,currentCommand=1;
bool sysOff=false;

void init_servo(){ // initializing the servo
    pc.printf("System initializing\r\n");
    servo.period(0.020);          // servo requires a 20ms
period
    servo2.period(0.020);          // servo requires a 20ms
period
    servo.pulsewidth(0.0005);
    servo2.pulsewidth(0.001);
    wait(1);
```

```
    for(float offsetTest=0.0; offsetTest<0.0015;
offsetTest+=0.0001) {
            servo.pulsewidth(0.0005 + offsetTest); // servo
position determined by a pulsewidth between 1-2ms
            wait(0.6);

currentVoltage=(voltage.read()+voltage.read()+voltage.read()+vol
tage.read()+voltage.read()+voltage.read()+voltage.read()+voltage
.read()+voltage.read()+voltage.read())/10;
            if (compareV<currentVoltage)
{compareV=currentVoltage;offset=offsetTest;}  // use the largest
voltage position
            pc.printf("offsetTest: %f    offset:
%f\r\n",offsetTest,offset);

    }

    compareV=0.0;
    servo.pulsewidth(0.0005 + offset);
    //servo2.pulsewidth(0.0005);
    for(float offsetTest=0.0; offsetTest<0.0015;
offsetTest+=0.0001) {
            servo2.pulsewidth(0.0005 + offsetTest); // servo
position determined by a pulsewidth between 1-2ms
            wait(0.6);

currentVoltage=(voltage.read()+voltage.read()+voltage.read()+vol
tage.read()+voltage.read()+voltage.read()+voltage.read()+voltage
.read()+voltage.read()+voltage.read())/10;
            if (compareV<currentVoltage)
{compareV=currentVoltage;offset2=offsetTest;} // use the largest
voltage position
            pc.printf("offsetTest: %f    offset2:
%f\r\n",offsetTest,offset2);

    }
    pc.printf("Initialization complete\r\n");
}
```

```
void callback() {  // receive from Beaglebone
    // read from the serial to clear the RX interrupt
    currentCommand=((int)(device.getc()- '0'));
    //pc.printf("currentCommand: %d\r\n", currentCommand);
    if (currentCommand!=lastCommand) {if (currentCommand==1)
{sysOff=false;pc.printf("Releasing Panels!\r\n");init_servo();}
else {sysOff=true;offset=0;offset2=0;pc.printf("Retracting
Panels!\r\n");}}
    if (currentCommand==0)
{offset=0;offset2=0;servo.pulsewidth(0.0005 +
offset);servo2.pulsewidth(0.0005 +
offset2);servo3.pulsewidth(0.0005 +
offset);servo4.pulsewidth(0.0005 + offset2);}
    //pc.printf("%c", device.getc());
    lastCommand=currentCommand;
    led2 = !led2;  // toggle led
}


inline static REAL sqr(REAL x) {
    return x*x;
}

// an open source math library for linear regression
int linreg(int n, const REAL x[], const REAL y[], REAL* m, REAL*
b, REAL* r)
    {
        REAL   sumx = 0.0;        /* sum of x */
        REAL   sumx2 = 0.0;       /* sum of x**2  */
        REAL   sumxy = 0.0;       /* sum of x * y */
        REAL   sumy = 0.0;        /* sum of y     */
        REAL   sumy2 = 0.0;       /* sum of y**2  */

       for (int i=0;i<n;i++)   // executing math calculation
          {
          sumx  += x[i];
          sumx2 += sqr(x[i]);
          sumxy += x[i] * y[i];
          sumy  += y[i];
```

```
            sumy2 += sqr(y[i]);
            }

        REAL denom = (n * sumx2 - sqr(sumx));
        if (denom == 0) { // rule out invalid data
            *m = 0;
            *b = 0;
            if (r) *r = 0;
            return 1;
        }

        *m = (n * sumxy  -  sumx * sumy) / denom;   // m is the
slope
        *b = (sumy * sumx2  -  sumx * sumxy) / denom;
        if (r!=NULL) {                  // calculating r
            *r = (sumxy - sumx * sumy / n) /
                  sqrt((sumx2 - sqr(sumx)/n) *
                  (sumy2 - sqr(sumy)/n));
        }

        return 0;
}
```

```
float search(bool direction,float val) // search for horizontal
direction
    {
    int n = 10,i;
    //float val;
    REAL x[10]= {1,2,3,4,5,6,7,8,9,10};
    REAL y[10];
```

```
    //val=offset;

    for (i=0;i<=9;i++){  // collecting the data sets
        if (sysOff==true) return 0.0;
        if (direction==true)
            val+=0.000015;
        else
            val-=0.000015;
        if (val<0) val=0;
        if (val>0.0015) val=0.0015;
        servo.pulsewidth(0.0005 + val); // execute servo command
        wait(0.2); // wait for command done

y[i]=(voltage.read()+voltage.read()+voltage.read()+voltage.read(
)+voltage.read()+voltage.read()+voltage.read()+voltage.read()+vo
ltage.read()+voltage.read())/10;
        //pc.printf("y=%g i=%g val=%g\r\n",y[i],i,val);
        pc.printf("val: %f    offset: %f\r\n",val,offset);

    }

    REAL m,b,r;
    linreg(n,x,y,&m,&b,&r);
    //pc.printf("m=%g b=%g r=%g\r\n",m,b,r);
    pc.printf("slope = %g\r\n",m);
    return m;
    }




float search2(bool direction,float val)   // search for vertical
direction
    {
    int n = 10,i;
    //float val;
    REAL x[10]= {1,2,3,4,5,6,7,8,9,10};
    REAL y[10];
    //val=offset2;
```

```
    for (i=0;i<=9;i++){  // collecting the data sets
        //temp=(float) i;
        if (sysOff==true) return 0.0;
        if (direction==true)
            val+=0.000015; // turn either right or left
        else
            val-=0.000015;
        if (val<0) val=0;
        if (val>0.0015) val=0.0015;
        servo2.pulsewidth(0.0005 + val); // execute servo
command
        wait(0.2); // wait for command done

y[i]=(voltage.read()+voltage.read()+voltage.read()+voltage.read(
)+voltage.read()+voltage.read()+voltage.read()+voltage.read()+vo
ltage.read()+voltage.read())/10;
        //pc.printf("y=%g i=%g val=%g\r\n",y[i],i,val);
        pc.printf("val: %f    offset2: %f\r\n",val,offset2);

    }

    REAL m,b,r;
    linreg(n,x,y,&m,&b,&r);
    //pc.printf("m=%g b=%g r=%g\r\n",m,b,r);
    pc.printf("slope = %g\r\n",m);
    return m;
}




int main() {

    device.attach(&callback);  // serial communication interrupt
    init_servo();
```

```
    while (1) {
        //servo.pulsewidth(0.0015);
        //wait(0.25);


        if (sysOff==false){ // disabled by the server command

currentVoltage=(voltage.read()+voltage.read()+voltage.read()+vol
tage.read()+voltage.read()+voltage.read()+voltage.read()+voltage
.read()+voltage.read()+voltage.read())/10;


        val1=search(true,offset);   // search for horizontal max
        val2=search(false,offset);  // search for horizontal
max, another direcion

        if ((val1>0.00005)&&(val2<-0.00005)) offset+=0.00005;
// check for slope to turn right
        if ((val2>0.00005)&&(val1<-0.00005)) offset-=0.00005;
// check for slope to turn left
        if ((val1>0.0001)&&(val2>0.0001))  {if (val1>val2)
offset+=0.00005;else offset-=0.00005;}   // if could turn either
way, turn with larger slope
        if (offset<0) offset=0;            // make sure not
exceed range
        if (offset>0.0015) offset=0.0015;  // make sure not
exceed range

        servo.pulsewidth(0.0005 + offset);  // execute servo
command



        val1=search2(true,offset2);   // search for vertical max
        val2=search2(false,offset2);  // search for vertical
max, another direction
```

```
        if ((val1>0.00005)&&(val2<-0.00005)) offset2+=0.00005;
// check for slope to turn up
        if ((val2>0.00005)&&(val1<-0.00005)) offset2-=0.00005;
// check for slope to turn down
        if ((val1>0.0001)&&(val2>0.0001))  {if (val1>val2)
offset2+=0.00005;else offset2-=0.00005;}  // if could turn
either way, turn with larger slope
        if (offset2<0) offset2=0;            // make sure not
exceed range
        if (offset2>0.0015) offset2=0.0015;  // make sure not
exceed range

        servo2.pulsewidth(0.0005 + offset2);  // execute servo
command

        servo3.pulsewidth(0.0005 + offset);  // execute servo
command
        servo4.pulsewidth(0.0005 + offset2);  // execute servo
command


        pc.printf("percentage: %3.3f% V    offset: %f
offset2: %f\r\n", currentVoltage*3.3*3,offset,offset2);
        wait(0.5);
        servo3.pulsewidth(0);servo4.pulsewidth(0);  // turn off
servo so not vibrate
        if
((abs(offsetOld-offset)<0.000001)&&(abs(offset2Old-offset2)<0.00
0001))
{servo.pulsewidth(0);servo2.pulsewidth(0);servo3.pulsewidth(0);s
ervo4.pulsewidth(0);wait(60.0);}
        offsetOld=offset;  // record data
        offset2Old=offset2;// record data

        }
    }
}
```

3. Project Web Server Code:

io.php:

```
<?php

// should add if (isset($_GET['act']))
if ($_GET['act']=='reset')
file_put_contents('data.json','{"time":[],"voltage":[]}');

if ($_GET['act']=='add')  {

    if
((file_get_contents('data.json')=='{"time":[],"voltage":[]}')&&(
$_GET['reset']==0))
        {echo 'reset';die();}

    $json=json_decode(file_get_contents('data.json'),true);
    $json[time][]=$_GET['x'];  // should add quotes
    $json[voltage][]=$_GET['y'];

    file_put_contents('data.json',json_encode($json));
}

if ($_GET['act']=='release')
file_put_contents('command.json','{"command": 1}');
if ($_GET['act']=='retract')
file_put_contents('command.json','{"command": 0}');


if ($_GET['q']=='command') {
    $json=json_decode(file_get_contents('command.json'),true);
    echo $json[command];
}

// end of PHP file
?>
```

HTML web page (including Javascript to fetch new data):

```
<!DOCTYPE html><html><head>

<meta charset="utf-8" /><title>Solar Panel Monitor</title><meta
name="viewport" content="width=device-width,
initial-scale=1.0"><meta name="keywords" content="Yuxin
Pan"><meta name="description" content="Yuxin Pan course
project."><meta name="author" content="Yuxin Pan"><link
rel="shortcut icon" href="https://www.panyuxin.com/favicon.ico">
    <link rel="stylesheet"
href="https://www.panyuxin.com/assets/css/bootstrap.min.css" >
    <!--<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/fon
t-awesome.min.css" >-->
<link rel="stylesheet" href="font-awesome.min.css" >

<script
src="https://www.panyuxin.com/assets/js/jquery.min.js"></script>
<script src="Chart.js"></script>
</head>
<body>
<canvas id="myChart" width="100px" height="40px"></canvas>
<span id="in"></span>
<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (xhttp.readyState == 4 && xhttp.status == 200) {
      //document.getElementById("cookie").innerHTML =
xhttp.responseText;
    }
  };
  xhttp.open("GET", "io.php?act=reset", true);
  xhttp.send();
  window.location.href = "./";

}

function loadDoc2() {
```

```
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (xhttp.readyState == 4 && xhttp.status == 200) {
      //document.getElementById("cookie").innerHTML =
xhttp.responseText;
    }
  };
  xhttp.open("GET", "io.php?act=retract", true);
  xhttp.send();
  //window.location.href = "./";

}

function loadDoc3() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (xhttp.readyState == 4 && xhttp.status == 200) {
      //document.getElementById("cookie").innerHTML =
xhttp.responseText;
    }
  };
  xhttp.open("GET", "io.php?act=release", true);
  xhttp.send();
  //window.location.href = "./";

}


var ctx = document.getElementById("myChart");
var data1='{"time":[],"voltage":[]} ';
var data1Old='';
var
panelNumber=3,resistor=182.5,tempN,sumEnergy,updateInterval=1;

plot = function() {


var client = new XMLHttpRequest();
client.open('GET', 'data.json?t='+ Date.now());
```

```
client.onreadystatechange = function() {
  data1=client.responseText;
}
client.send();

//document.getElementById("in").innerHTML = data1;
//while (data1==''){}
obj = JSON.parse(data1);

if (data1Old==data1) return;
data1Old=data1;

// calculate power at this time at insert into html
tempN=panelNumber*parseFloat((obj.voltage[obj.voltage.length-1])
)*parseFloat((obj.voltage[obj.voltage.length-1]))/resistor;
//sumEnergy=2+updateInterval*tempN;
sumEnergy=0;
for (var i = 0; i < obj.voltage.length; i++) {
    sumEnergy=sumEnergy+obj.voltage[i]*updateInterval;
}
tempN=tempN.toFixed(3);
sumEnergy=sumEnergy.toFixed(1);
document.getElementById("power").innerHTML = tempN.toString();
document.getElementById("energy").innerHTML =
sumEnergy.toString();




var data = {
    labels: obj.time,
    datasets: [
        {
            label: "Solar Panel Output Voltage (V)",
            fill: false,
            lineTension: 0.1,
            backgroundColor: "rgba(75,192,192,0.4)",
            borderColor: "rgba(75,192,192,1)",
            borderCapStyle: 'butt',
```

```
            borderDash: [],
            borderDashOffset: 0.0,
            //borderWidth: 40,
            borderJoinStyle: 'miter',
            pointBorderColor: "rgba(75,192,192,1)",
            pointBackgroundColor: "#fff",
            pointBorderWidth: 1,
            pointHoverRadius: 5,
            pointHoverBackgroundColor: "rgba(75,192,192,1)",
            pointHoverBorderColor: "rgba(220,220,220,1)",
            pointHoverBorderWidth: 2,
            pointRadius: 1,
            pointHitRadius: 10,
            data: obj.voltage,
            spanGaps: false,
        }
    ]
};



var myLineChart = new Chart(ctx, {  // create new chart
    type: 'line',
    data: data,
    options: {
animation : false,

title: {
            display: true,
            text: 'Solar Panel Monitor',
            fontSize: 26
        },

scales: {

xAxes: [{   // options to make the graph looks better
    ticks: {
        autoSkip: true,
        maxTicksLimit: 20
```

```
    }
}],
            yAxes: [{
                ticks: {
                    beginAtZero:true
                }
            }]
        }
    }

});};

setInterval(plot, 1000);

</script>
<br /><br /><br /><center> <p style="font-size:20px">
<i class="fa fa-cog fa-spin fa-fw"></i>
<span>Real time power output: </span><span id="power"></span>
Watt <br />
<i class="fa fa-cog fa-spin fa-fw"></i>
<span>Estimated total energy collected: </span><span
id="energy"></span> J <br /><br /></font><input type="button"
class="btn btn-info" value="Retract Panel"
onclick="loadDoc2();"> <input type="button" class="btn btn-info"
value="Release Panel" onclick="loadDoc3();"> <input
type="button" class="btn btn-info" value="Reset Data"
onclick="loadDoc();">
</center>
<!--
<script
src="https://cdnjs.cloudflare.com/ajax/libs/tether/1.4.0/js/teth
er.min.js"></script>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-alpha.6/js/
bootstrap.min.js"></script>
-->
</body></html>
```

Uno code for the artificial sun:

```
#include <Servo.h>

Servo myservo;  // create servo object to control a servo
                // a maximum of eight servo objects can be
created

int pos = 0;    // variable to store the servo position

void setup()
{
  myservo.attach(9);  // attaches the servo on pin 9 to the
servo object
}


void loop()
{
  for(pos = 0; pos < 180; pos += 1)  // goes from 0 degrees to
180 degrees
  {                                  // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to
position in variable 'pos'
    delay(1000);                       // waits 15ms for the
servo to reach the position
  }
  for(pos = 180; pos>=1; pos-=1)     // goes from 180 degrees to
0 degrees
  {
    myservo.write(pos);              // tell servo to go to
position in variable 'pos'
    delay(1000);                       // waits 15ms for the
servo to reach the position
  }
}
```

4. Project Beaglebone Code:

Serial communication between web server, beaglebone and mbed

```
import Adafruit_BBIO.UART as UART
import serial
import urllib2, urllib
import time

UART.setup("UART1") # establish the serial connection by Python

ser = serial.Serial(port = "/dev/ttyO1", baudrate=9600)
ser.close()
ser.open()

# client header
request_headers = {
"User-Agent": "Mozilla/5.0 (Windows NT 10.0; WOW64; rv:40.0)
Gecko/20100101 Firefox/40a.0",
}

# infinite loop
while True:
    request =
urllib2.Request("http://apps.panyuxin.com//test/plot/io.php?q=co
mmand", headers=request_headers)
    contents = urllib2.urlopen(request).read() # execute request
    if contents:
        print contents
        ser.write(contents)
    time.sleep(0.1)  # pause for 0.1 seconds
ser.close()
```

Polling voltage value and send to cloud:

```
var http = require('http');
var b = require('bonescript');  // running on Beaglebone

var inputPin = "P9_38";
var value = b.analogRead(inputPin);
var count=0;
var resetFlag=0;


callback = function(response) {
  var str = '';

  //another chunk of data has been recieved, so append it to
`str`
  response.on('data', function (chunk) {
    str += chunk;
  });

  //the whole response has been recieved, so we just print it
out here
  response.on('end', function () {
    console.log(str);
    if (str=='reset') {count=0;resetFlag=1;}
  });
}

process.on('uncaughtException', function(err) {
  console.log('Caught exception: ' + err);
});



loop();

function loop() {
    value = b.analogRead(inputPin);

    var options = {
```

```
        host: 'apps.panyuxin.com',
        path:
'/test/plot/io.php?act=add&x='+count+'&y='+value*18.1+'&reset='+
resetFlag,
        //if listening on a custom port, we need to specify it by
hand
        //port: '1337',
        //headers: {'custom': 'Custom Header Demo works'}
    };
    resetFlag=0;
    console.log(value);
    http.request(options, callback).end();

    count++;
    setTimeout(loop, 1000);
}
```