#include <Servo.h>

#include <math.h>

/\*

\* Link for solar information:(1) https://www.esrl.noaa.gov/gmd/grad/solcalc/ \*\*\*\*use this one

(2) https://www.esrl.noaa.gov/gmd/grad/solcalc/azel.html

($) https://www.timeanddate.com/sun/usa/las-cruces

\* Las Cruces Coordinate: 32.287989° N(+), -106.75385° W (-)

\* Azimuth and Elevation - an angular coordinate system for locating positions in the sky. Azimuth is measured clockwise from true North

to the point on the horizon directly below the object. Elevation is measured vertically from that point on the horizon up to the object.

\* offset\_azi\_testunit=3; offset\_alt\_testunit=3; (in degree)

\*/

/\*june 17 2021 - > day\_of\_year= 168\*/

int PTh=9; //present time in hours\*

int PTm=55; //present time in minutes\*

int day\_of\_year= 170; //day of the year of the experiment\*

double Hour=0; //initiation of variable "Hour" (hour corresponds to current time)

double gamma=0; //initiation of variable "gamma" (fractional year (in radian))

double eqtime=0; //initiation of variable "eqtime" (equation of time (in minute))

double decl=0; //intiation of variable "decl" (solar declination angle (in radians))

double time\_offset=0; //initiation of variable "time\_offset" (time offset (in minutes))

double tst=0; //initiation of variable "tst" (true solar time (in minutes))

double ha=0; //initiation of variable "ha" (solar hour angle (in radian))

double phi=0; //initiation of variable "phi" (solar zenith angle angle (in radian))

double alpha=0; //initiation of variable "alpha" (solar altitude angle angle (in radian))

double beta=0; //initiation of variable "beta" (solar azimuth angle angle (in radian))

double alpha\_d=0; //initiation of variable "alpha\_d" (solar altitude angle angle (in degree))

double beta\_d=0; //initiation of variable "beta\_d" (solar azimuth angle angle (in degree))

double snoon=0; //initiation of variable "snoon" (solar noon (in minute))

double t\_CT=0; //initiation of variable "t\_CT" (current time (in minutes))

double s\_x=0; //initiation of variable "s\_x" (x-component of sun\_vector)

double s\_y=0; //initiation of variable "s\_y" (y-component of sun\_vector)

double s\_z=0; //initiation of variable "s\_z" (z-component of sun\_vector)

double pi= 3.1415926; //value of 'pi'

double r2d= 180/pi; //convesion ratio of radian to degree conversion

double d2r= pi/180; //convesion ratio of degree to radian conversion

double h=23.625; //height(in inch) of convex mirror focal point from the origin(centre point of the base on which these mirror units are intalled)

int timezone=-6; //timezone in hour from UTC (Mountain Daylight Time)

double latitude= 32.287989; //latitude position (in degree) of the experimental station (Monte Vista Ave)

double longitude= -106.75385; //longitude position (in degree) of the experimental station (Monte Vista Ave)

double lat= (3.1415926/180)\*latitude; //latitude (in radian) for trigonometric functions

unsigned long myTime;

unsigned long last\_seen;

//-------------------------------------------------------------------------

//---------------UNIT#4-----------------

//------Focus\_Point--------

double x$= 0; //x position (in inch) of unit#$

double y$= 18.75; //y position (in inch) of unit#$

double z$=h; //z position (in inch) of unit#$

//target\_vector$: unit vector along reflected ray from flat mirror of unit#$

double f$\_x= -x$/(sqrt(x$\*x$+y$\*y$+z$\*z$)); //x-componenet of target\_vector$

double f$\_y= -y$/(sqrt(x$\*x$+y$\*y$+z$\*z$)); //y-componenet of target\_vector$

double f$\_z= z$/(sqrt(x$\*x$+y$\*y$+z$\*z$)); //z-componenet of target\_vector$

//-------------------------

//-----Bisector\_Vector$------

//bisector\_vector$: unit vector along bisector on fat mirror which will always be normal to the flat mirror of unit#$

double m$\_x=0; //initiation of variable "m$\_x" (x-component of bisector\_vector$)

double m$\_y=0; //initiation of variable "m$\_y" (y-component of bisector\_vector$)

double m$\_z=0; //initiation of variable "m$\_z" (z-component of bisector\_vector$)

//---------------------------

double a$=0; //initiation of variable "a$" (alt adjustment angle(in radian))

double b$=0; //initiation of variable "b$" (azi adjustment angle(in radian))

int a$\_d=0; //initiation of variable "a$\_d" (alt adjustment angle(in degree))

int b$\_d=0; //initiation of variable "b$\_d" (azi adjustment angle(in degree))

int offset\_alt$=3; //offset of altitude adjustment (in degree) for unit#$

int offset\_azi$=3; //offset of azimuth adjustment (in degree) for unit#$

int a$\_d\_offset= offset\_alt$ +a$\_d; //intial angle for altitude adjustment with offset for unit#$

int b$\_d\_offset= offset\_azi$ +b$\_d; //intial angle for azimuth adjustment with offset for unit#$

double theta$=0; // for test only. it's angle of projection of normal on xy plane

Servo myservo$\_alt; //create servo object to control a servo for altitude adjustment for unit#$

Servo myservo$\_azi; //create servo object to control a servo for azimuthal adjustment for unit#$

//-----------------------------------------

void setup()

{

//---------------UNIT#$--------------------

myservo$\_alt.attach(5); //attaches the 'altitude adjustment servo' of unit#$ on pin 5 to the servo object

myservo$\_azi.attach(6); //attaches the 'azimuthal adjustment servo' of unit#$ on pin 6 to the servo object

//-----------------------------------------

//-----------------------------------------

Serial.begin(9600);

Serial.print("myTime=");

Serial.print(myTime);

Serial.print("\t");

Serial.print("tst=");

Serial.print(tst);

Serial.print("\t");

Serial.print("time\_offset=");

Serial.print(time\_offset);

Serial.print("\t");

Serial.print("eqtime=");

Serial.print(eqtime);

Serial.print("\t");

Serial.print("decl=");

Serial.print(decl);

Serial.print("\n");

Serial.print("snoon=");

Serial.print(snoon);

Serial.print("\t");

Serial.print("t\_CT=");

Serial.print(t\_CT);

Serial.print("\n");

Serial.print("m$\_x=");

Serial.print(m$\_x);

Serial.print("\t");

Serial.print("m$\_y=");

Serial.print(m$\_y);

Serial.print("\n");

Serial.print("theta$=");

Serial.print(theta$);

Serial.print("\n");

Serial.print("alpha\_d=");

Serial.print(alpha\_d);

Serial.print("\t");

Serial.print("beta\_d=");

Serial.print(beta\_d);

Serial.print("\n");

Serial.print("a$\_d=");

Serial.print(a$\_d);

Serial.print("\n");

Serial.print("b$\_d=");

Serial.print(b$\_d);

Serial.print("\n");

//-----------------------------------------

}

void loop()

{

myTime= millis();

//----------------------------Solar\_Angles---------------------------------

Hour= (((PTh\*60+PTm)\*60000) +myTime)/3600000; //hour corresponds to current time

gamma= (2\*pi/365)\*(day\_of\_year-1+((Hour-12)/24)); //fractional year (in radian) [for leap year, use 366 instead of 365 in the denominator]

eqtime= 229.18\*(0.000075+0.001868\*cos(gamma)-0.032077\*sin(gamma)-0.014615\*cos(2\*gamma)-0.040849\*sin(2\*gamma)); //equation of time (in minutes)

decl= 0.006918-0.399912\*cos(gamma)+0.070257\*sin(gamma)-0.006758\*cos(2\*gamma)+0.000907\*sin(2\*gamma)-0.002697\*cos(3\*gamma)+0.00148\*sin(3\*gamma);

//solar declination angle (in radians)

time\_offset= eqtime+4\*longitude-60\*timezone; //time offset (in minutes)

tst= (PTh\*60+PTm+time\_offset) +(myTime/60000); //true solar time (in minutes)

ha= (d2r)\*((tst/4)-180); //solar hour angle(in radian)

snoon= 360-4\*longitude-eqtime; //solar noon for a given location is found from the longitude (in minutes)

t\_CT= (PTh\*60+PTm) +(myTime/60000); //current time (in minutes)

phi= acos(sin(lat)\*sin(decl)+cos(lat)\*cos(decl)\*cos(ha)); //solar zenith angle angle (in radian)

alpha= (pi/2 -phi); //solar altitude angle (in radian)

beta= acos((sin(lat)\*cos(phi)-sin(decl))/(cos(lat)\*sin(phi))); //solar azimuth angle (in radian)

if(t\_CT <= snoon)

{

beta= pi-(beta); //solar azimuth angle (in radian) before solar noon

}

if(t\_CT > snoon)

{

beta= pi+(beta); //solar azimuth angle (in radian) after solar noon

}

alpha\_d= (r2d)\*alpha; //solar altitude/elevation angle (in degree)

beta\_d= (r2d)\*(beta); //solar azimuth angle (in degree)

//------Sun\_Vector------

s\_x= cos(alpha)\*cos(beta); //x-component of sun\_vector)

s\_y= -cos(alpha)\*sin(beta); //y-component of sun\_vector)

s\_z= sin(alpha); //z-component of sun\_vector)

//----------------------

//-----------------------------------------------------------------------------

//--------------------($)------------------------

m$\_x=(((abs(s\_z+f$\_z))/(s\_z+f$\_z))\*(s\_x+f$\_x))\*(1/(sqrt((s\_x+f$\_x)\*(s\_x+f$\_x)+(s\_y+f$\_y)\*(s\_y+f$\_y)+(s\_z+f$\_z)\*(s\_z+f$\_z))));

m$\_y=(((abs(s\_z+f$\_z))/(s\_z+f$\_z))\*(s\_y+f$\_y))\*(1/(sqrt((s\_x+f$\_x)\*(s\_x+f$\_x)+(s\_y+f$\_y)\*(s\_y+f$\_y)+(s\_z+f$\_z)\*(s\_z+f$\_z))));

m$\_z=(abs(s\_z+f$\_z))\*(1/(sqrt((s\_x+f$\_x)\*(s\_x+f$\_x)+(s\_y+f$\_y)\*(s\_y+f$\_y)+(s\_z+f$\_z)\*(s\_z+f$\_z))));

//---test----

theta$= (r2d)\*atan2(m$\_y, m$\_x);

//-----------

//a$= pi/2 -acos(m$\_z); //elevation angle (in radian) of flat mirror of unit#$

if(m$\_x>0 && m$\_y>=0)

{

a$= pi/2 +acos(m$\_z);

b$= pi/2 -atan(abs(m$\_y)/abs(m$\_x));

}

else if(m$\_x<=0 && m$\_y>0)

{

a$= pi/2 -acos(m$\_z);

b$= pi/2 +atan(abs(m$\_y)/abs(m$\_x));

}

else if(m$\_x<0 && m$\_y<=0)

{

a$= pi/2 -acos(m$\_z);

b$= pi/2 -atan(abs(m$\_y)/abs(m$\_x));

}

else if(m$\_x>=0 && m$\_y<0)

{

a$= pi/2 +acos(m$\_z);

b$= pi/2 +atan(abs(m$\_y)/abs(m$\_x));

}

else {b$=b$;}

a$\_d= (r2d)\*a$; //elevation angle (in degree) of flat mirror of unit#$

b$\_d= (r2d)\*b$; //azimuth angle (in degree) of flat mirror of unit#$

if(myTime - last\_seen > 10000)

{

a$\_d\_offset= offset\_alt$ +a$\_d; //intial angle for altitude adjustment with offset for unit#$

myservo$\_alt.write(a$\_d\_offset);

b$\_d\_offset= offset\_azi$ +b$\_d; //intial angle for azimuth adjustment with offset for unit#$

myservo$\_azi.write(b$\_d\_offset);

last\_seen = myTime;

}

//-----------------------------------------------

Serial.print(">>");

Serial.print(myTime);

Serial.print("\t");

Serial.print(tst);

Serial.print("\t");

Serial.print(time\_offset);

Serial.print("\t");

Serial.print(eqtime);

Serial.print("\t");

Serial.print(decl);

Serial.print("\n");

Serial.print("\t");

Serial.print(snoon);

Serial.print("\t");

Serial.print(t\_CT);

Serial.print("\n");

Serial.print("\t");

Serial.print(m$\_x);

Serial.print("\t");

Serial.print(m$\_y);

Serial.print("\n");

Serial.print("\t");

Serial.print(theta$);

Serial.print("\n");

Serial.print("\t");

Serial.print(alpha\_d);

Serial.print("\t");

Serial.print(beta\_d);

Serial.print("\n");

Serial.print("\t");

Serial.print(a$\_d);

Serial.print("\n");

Serial.print("\t");

Serial.print(b$\_d);

Serial.print("\n");

delay(5000);

}