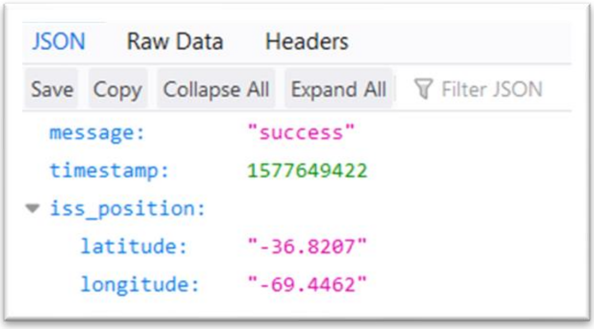
Mathematical Calculations for the ISS Tracker

Imagine drawing a line directly from the ISS to the centre of the Earth. The point on the surface is known as the nadir and these coordinates can be retrieved through an application programming interface (API) on the http://open-notify.org/ website. To find the ISS in the sky, we need to know the direction (bearing) and height (elevation) to point at, relative to our own location.

The data is downloaded in JSON format so it needs to be parsed (broken into meaningful chunks of information).



You will need to know your own location, measured as decimal latitude and longitude which you can do on Google maps if you right click on your location. This needs to be added to the programming code.

The first measurement we need to calculate is the bearing from the observer's location to the nadir point. Fortunately, we have the latitude and longitude coordinates for these two points, so we can calculate the angle between them.

I have set up a spreadsheet so you can experiment with different locations and see if the answers match up with the bearings that are calculated on an interactive map.

Let's look at the formula to calculate the bearing, β from one set of co-ordinates to another.

$X = \cos(\theta b) * \sin(\Delta\varphi)$
$Y = \cos(\theta a) * \sin(\theta b) - \sin(\theta a) * \cos(\theta b) \cos(\Delta\varphi)$
Bearing, $\beta = \mathrm{atan2}(X,Y)$

a = my location
b = ISS location
β = azimuth bearing from observer to ISS
θ = Latitude
φ = Longitude
Δφ = difference in longitude

Note that the program uses radians rather than degrees to measure the angles. To convert degrees into radians, you need to multiply by Pi and divide by 180.

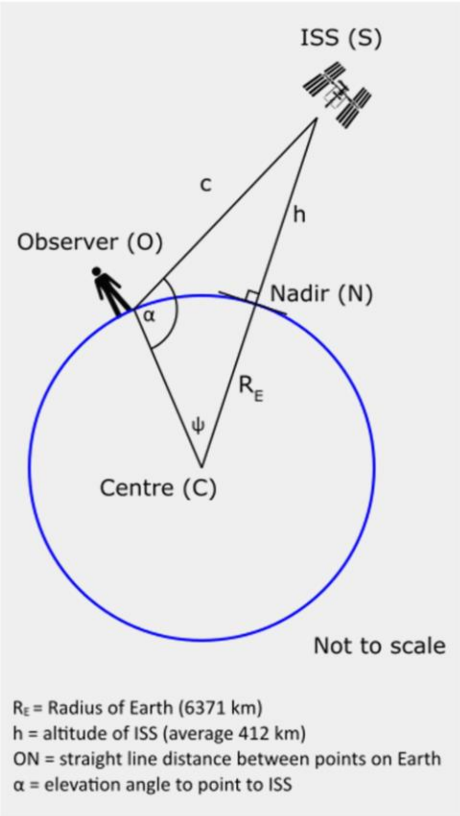
This is how the bearing calculation has been coded:

```
// ESP32 calculates angles in radians (not degrees)
double latitude_in_radians = latitude * PI / 180;
double longitude_in_radians = longitude * PI / 180;
double my_latitude_in_radians = my_latitude * PI / 180;
double my_longitude_in_radians = my_longitude * PI / 180;
// calculate azimuth bearing to ISS
double x = cos(latitude_in_radians)*sin(longitude_in_radians-my_longitude_in_radians);
double y = cos(my_latitude_in_radians)*sin(latitude_in_radians
          -sin(my_latitude_in_radians*cos(latitude_in_radians
          *cos(longitude_in_radians-my_longitude_in_radians);
double b = atan2(x,y);
//Serial.print("Bearing to ISS: ");
b=b*180/PI;// convert to degrees
//Serial.println(b);
```

Now that we have the bearing to the ISS, we need to work out how high to point the arrow, known as the elevation using some trigonometry.

When creating a mathematical model, we may need to make some assumptions to keep things simple. For this example, I have assumed that the Earth is a perfect sphere, and we know the radius of it (R_E = 6371 km). The ISS position is constantly being monitored, so when the orbital distance decays due to atmospheric drag, it is boosted to maintain its height. In 2019, this ranged from 405km to 418km but to keep the calculations simple, an average orbital distance has been used (h = 412 km).

Consider the diagram below. If we can calculate the angle α (alpha) it is possible to elevate the tracker's pointer to the right position. It can be seen from the diagram, that if the pointer is initially pointing straight down towards the centre of the Earth (which is set up when the device is calibrated), it needs to rotate through α degrees to point at the ISS.



We need to calculate the straight-line distance between the observer and the nadir (ON).

The first step is to convert the latitude and longitude coordinates to X,Y,Z cartesian coordinates.

$$X = R_E \cos(\theta)\cos(\varphi)$$

$$Y = R_E \cos(\theta)\sin(\varphi)$$

$$Z = R_E \sin(\theta)$$

We can use Pythagoras' theorem in three dimensions to calculate the distance. ON is the distance from the observer to the nadir.

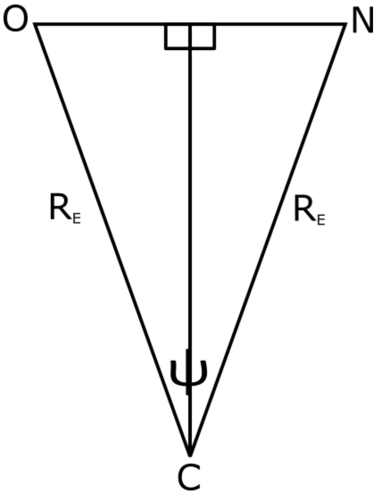
ON = $\sqrt{(X_{iss} - X_{me})^2 + (Y_{iss} - Y_{me})^2 + (Z_{iss} - Z_{me})^2}$

Where
X_{iss} = X coordinate for the ISS
Y_{iss} = Y coordinate for the ISS
Z_{iss} = Z coordinate for the ISS
X_{me} = X coordinate for my location
Y_{me} = Y coordinate for my location
Z_{me} = Z coordinate for my location

This is coded as:

```
double dist_to_nadir = sqrt(sq(x2-x1)+sq(y2-y1)+sq(z2-z1));
//Serial.print("Distance to nadir :");
//Serial.println(dist to nadir);
```

The next stage is to work out the geocentric angle ψ (psi) from the observer to the ISS. We now know distance ON, so we can split the triangle into two right angle triangles and use the sine function to calculate ψ.



For one triangle

$$\sin\left(\frac{\psi}{2}\right) = \frac{opposite}{hypotenuse}$$

$$\sin\left(\frac{\psi}{2}\right) = \left(\frac{½ON}{R_E}\right)$$

$$\psi = 2\ \sin^{-1}\left(\frac{ON}{2R_E}\right)$$

This is coded as follows:

```
double geocentric_angle_in_radians = 2 * asin(dist_to_nadir/(2*earth_radius));
//Serial.print("Geocentric angle in degrees :");
//Serial.println(geocentric_angle_in_radians * 180 / PI);
```

Using the Law of Cosines to calculate distance c from the observer to the ISS.

$$c^2 = (h + R_E)^2 + {R_E}^2 - 2(h + R_E).R_E \cos(\psi)$$

$$c = \sqrt{(h + R_E)^2 + {R_E}^2 - 2(h + R_E).R_E \cos(\psi)}$$

We have three sides and one angle of the triangle (ψ), so we can use the Law of Sines to calculate the angle α.

$$\frac{\sin(\psi)}{c} = \frac{\sin(\alpha)}{(h + R_E)}$$

$$\sin(\alpha) = (h + R_E)\left(\frac{\sin(\psi)}{c}\right)$$

$$\alpha = \sin^{-1}\left((h + R_E)\left(\frac{\sin(\psi)}{c}\right)\right)$$

This is coded as:

```
double ISS_angle = (180 / PI) * asin((orbit + earth_radius) * (sin(geocentric_angle_in_radians)/distance_to_ISS));
```

When the device starts, it needs to have a reference point, so it knows where the pointer is pointing. Initially the elevation angle is set to point straight down, to the centre of the Earth. It is then raised up to point at Polaris, the North Star which is easy to do, as the elevation angle is the same as your latitude angle. Once the pointer is aligned with true north, the azimuth position will also be known.

The angles for the azimuth and elevation have now been calculated, so they need to be converted into motion using the stepper motors.

```
// azimuth stepper motor has 4096 half steps per revolution and gear ratio 113/22
azimuth_motor.moveTo(4096/360 * 113/22 * b);
// elevation motor has 4096 half steps per revolution
elevation_motor.moveTo(4096/ 360 * ISS_angle);
```

References

European Southern Observatory (2019) *Is the altitude of Polaris equal to your latitude?.* Available at: https://www.eso.org/public/outreach/eduoff/aol/market/information/finevent/polmath.html

GISMAP (2020) *Formula to Find Bearing or Heading Angle Between Two Points: Latitude and Longitude.* Available at: https://www.igismap.com/formula-to-find-bearing-or-heading-angle-between-two-points-latitude-longitude/

Igismap (2020) *Bearing Angle.* Available at: https://www.igismap.com/map-tool/bearing-angle

Movable Type Scripts (2020) *Calculate distance, bearing and more between Latitude/Longitude points.* Available at: http://www.movable-type.co.uk/scripts/latlong.html?from=47.80423,-120.03866&to=47.830481,-120.00987

Open Notify (2020) *Open APIs From Space*. Available at: http://open-notify.org/