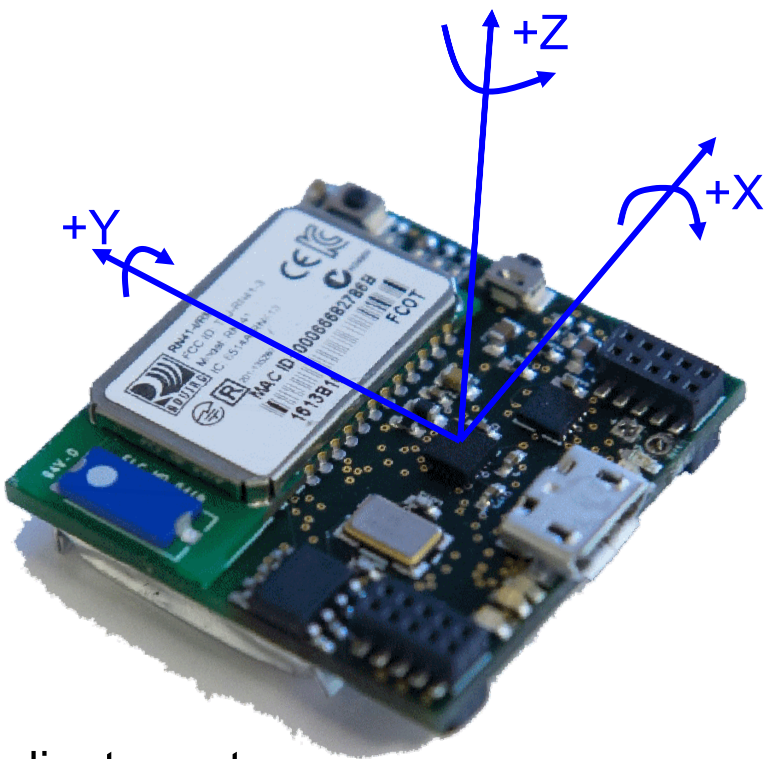
Sensor-fixed coordinate system (S)

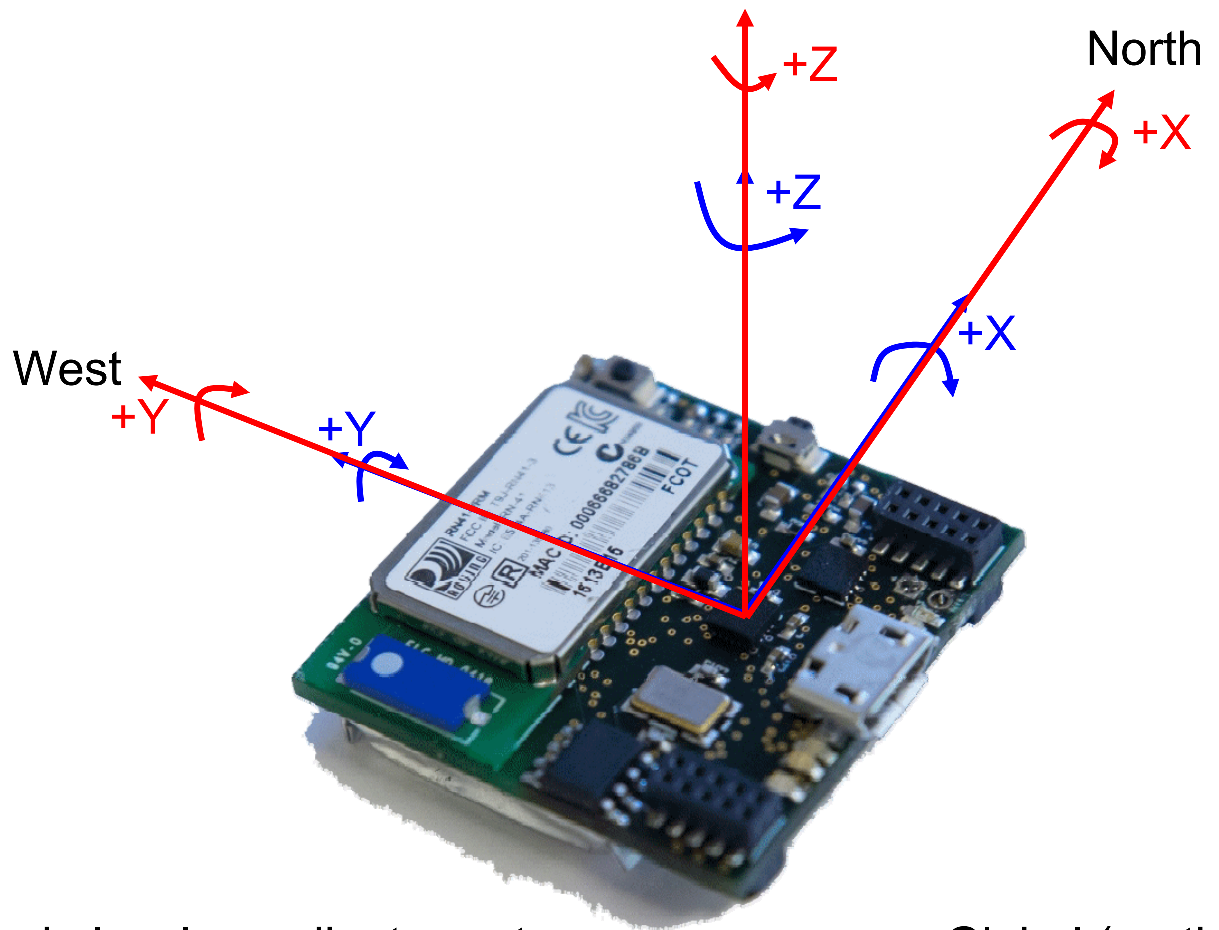
- Output of node's accelerometer, gyroscope verified to follow this convention
- Sensor coordinate system is right hand



Node local coordinate system

Earth-fixed coordinate system (G)

- Global coordinate system is right hand
- Node represented in the "zero" position (yaw=0, pitch=0, roll=0)

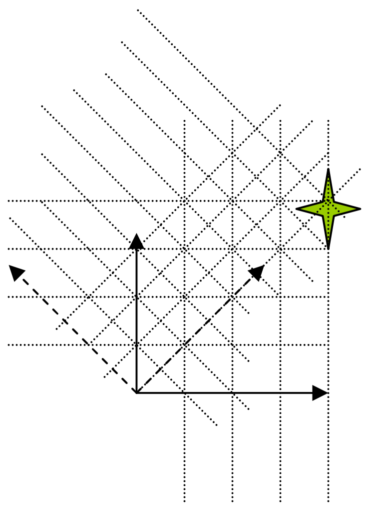


Node local coordinate system

Global (earth) coordinate system

Representation of rotations

- BlueSense reports the orientation of the sensor-fixed coordinate system (S) with respect to the earth-fixed coordinate system (G)
- The orientation is represented by a quaternion $q_{S \leftarrow G}$
- The quaternion $q_{S \leftarrow G}$ can be used to rotate a vector represented in earth-coordinates G into sensor-coordinates S

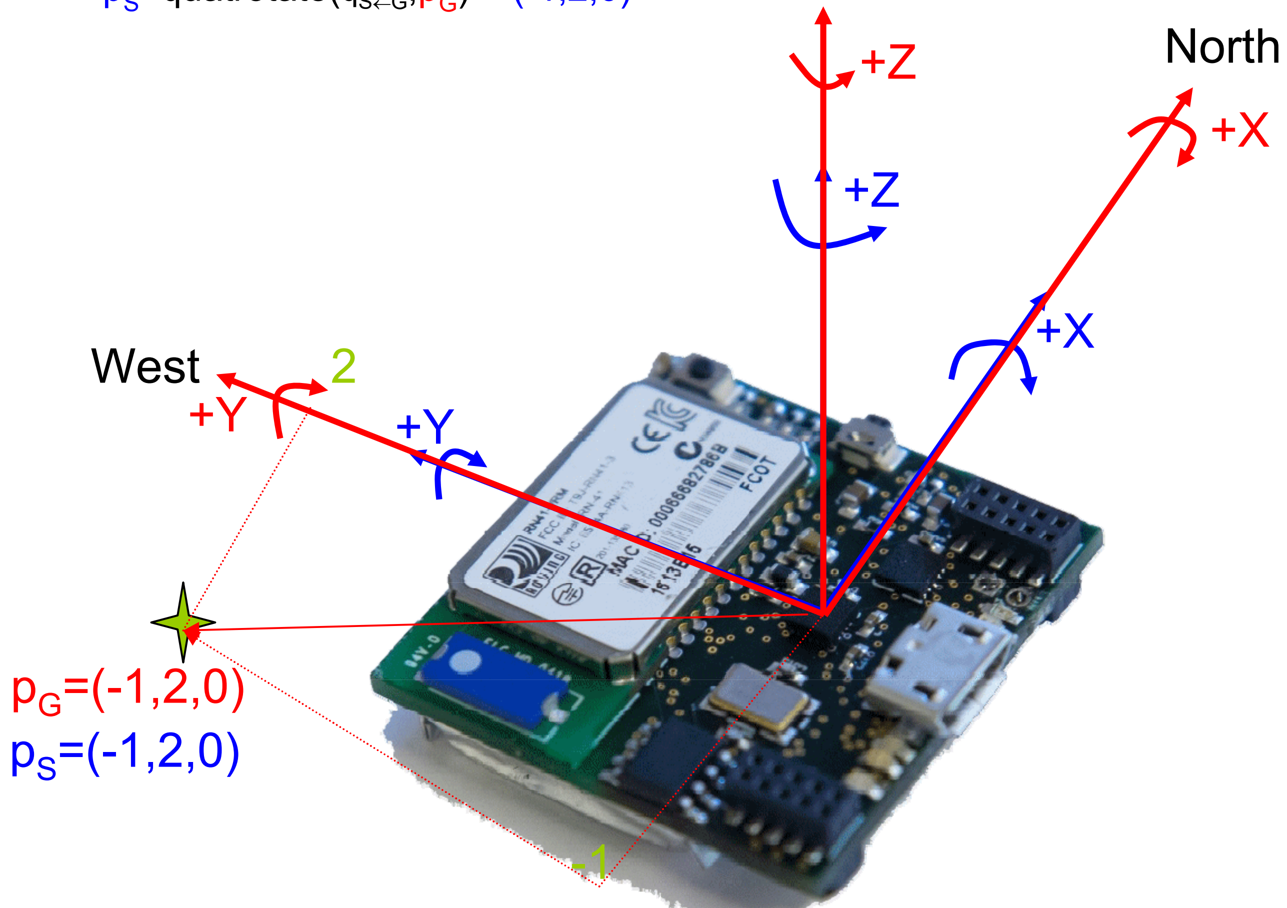


Initial sensor coordinate system
Star local coordinates ~(4,4,0)

New sensor coordinate system
Star local coordinates ~(8,0,0)

Earth-fixed (G) to sensor-fixed (S) mapping

- Application: finding local coordinates of a target provided by its absolute coordinates
- Example: star at coordinate p_G=(-1,2,0) in the earth-fixed (G) coordinate system
  - As the node is in the zero position, the star coordinate is also (-1,2,0) in the sensor-fixed coordinate system.
  - $q_{S\leftarrow G}$ = (1,0,0,0)
  - p_S=quatrotate($q_{S\leftarrow G}$,p_G) = (-1,2,0)

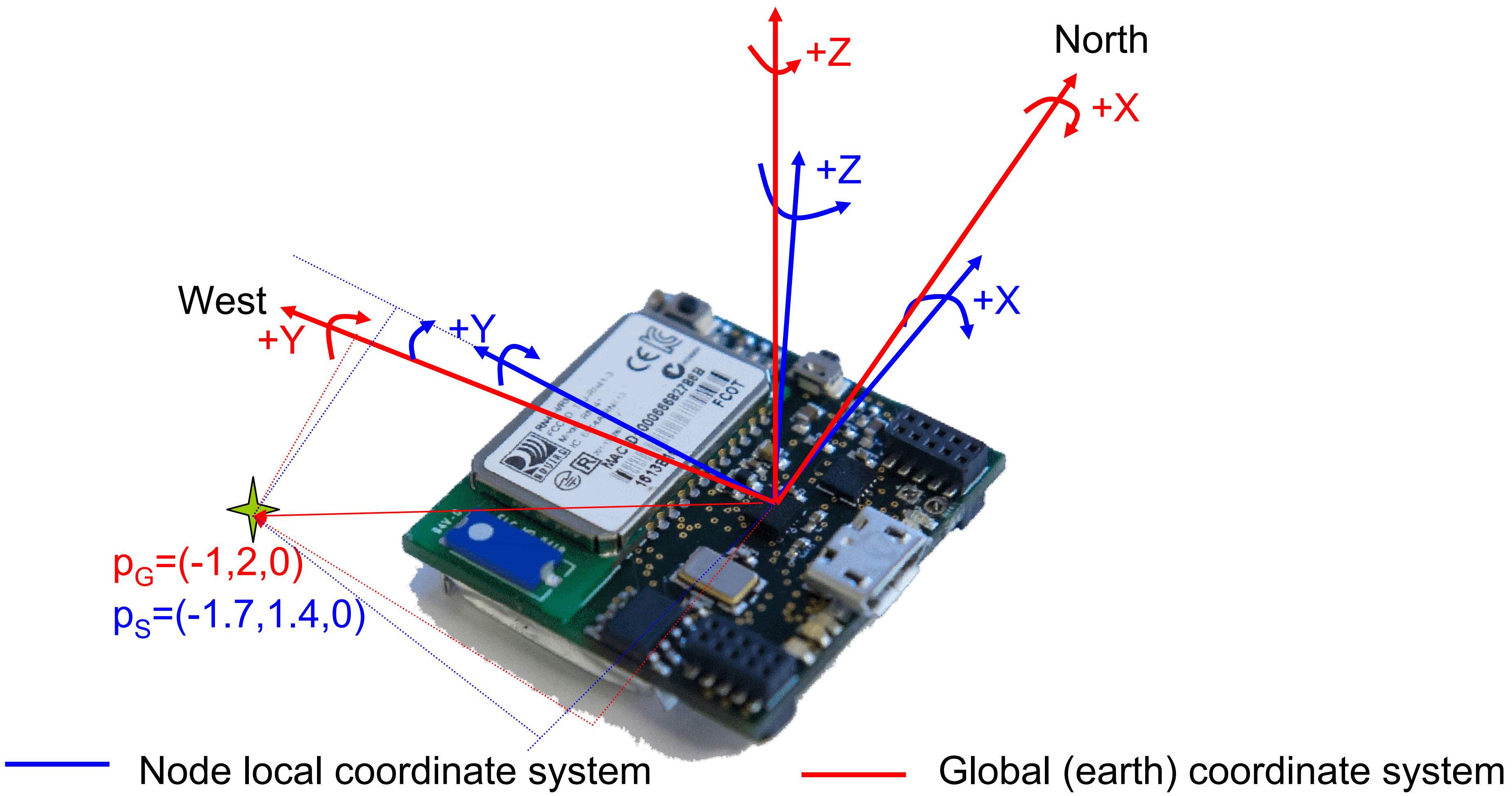


Node local coordinate system

Global (earth) coordinate system

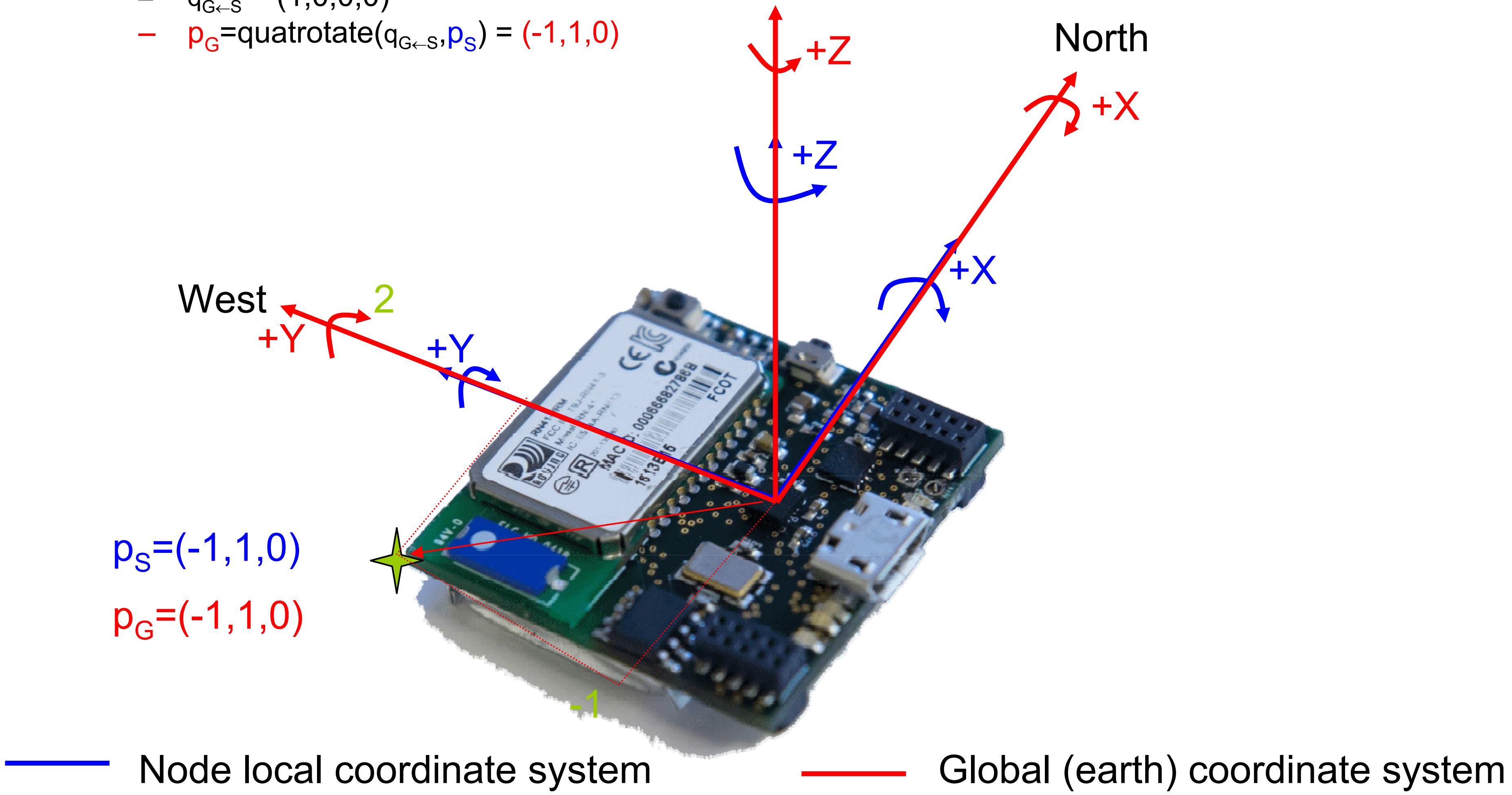
Earth-fixed (G) to sensor-fixed (S) mapping

- The sensor rotates by ~-20° along the Z axis
  - $q_{S \leftarrow G}$ = (+.98, 0, 0, -.2)
  - p_S=quatrotate($q_{S \leftarrow G}$,p_G) = (-1.7, 1.4, 0)



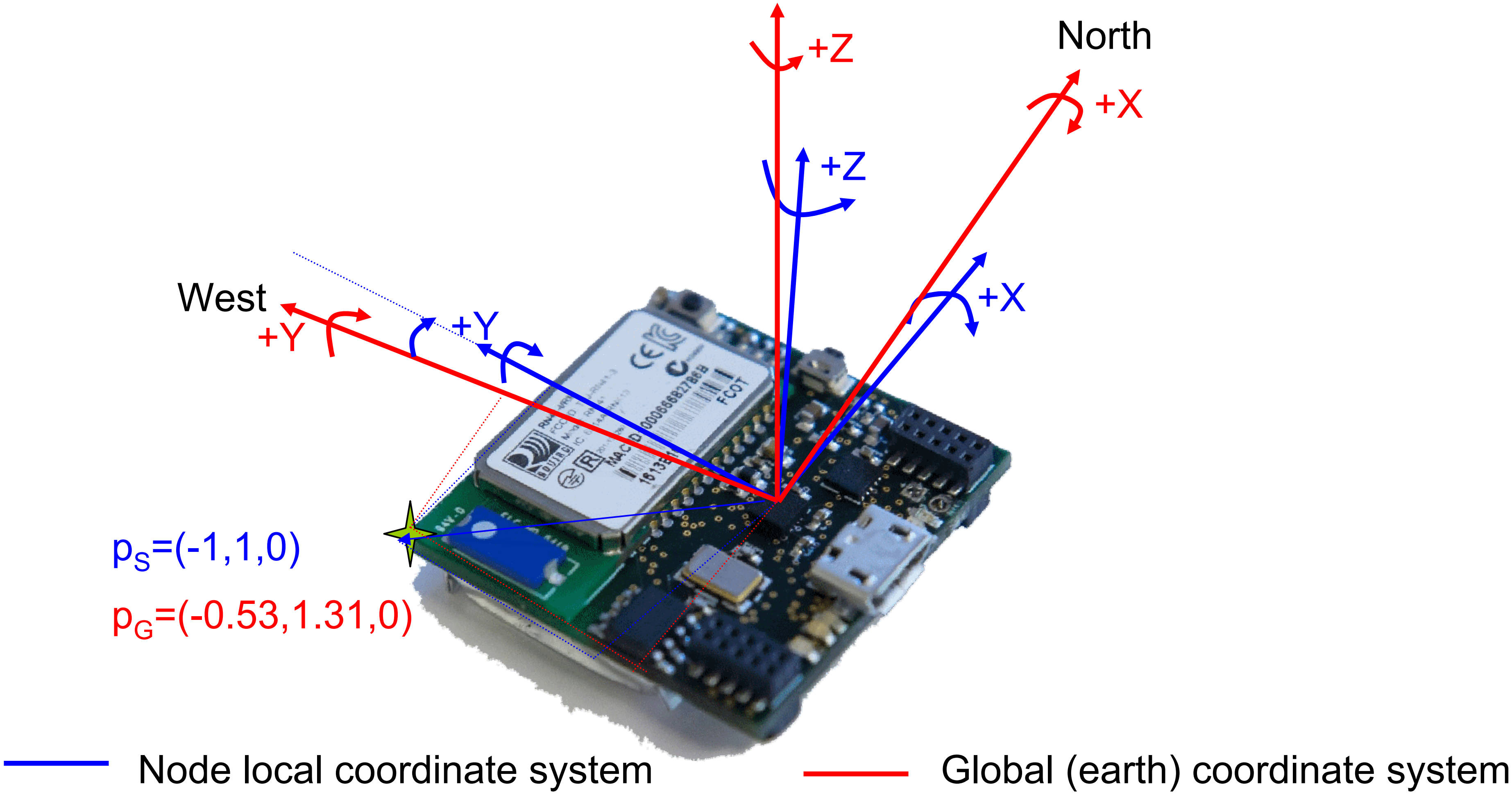
Sensor-fixed (S) to earth-fixed (G) mapping

- Application: find earth coordinates of a target located in sensor coordinates; rendering
- Example: rendering of the edge of the node at coordinate p_S=(-1,1,0) in the sensor-fixed (S) coordinate system
  - As the node is in the zero position, the star coordinate is also (-1,1,0) in the earth-fixed coordinate system.
  - $q_{G \leftarrow S}$ = (1,0,0,0)
  - p_G=quatrotate($q_{G \leftarrow S}$,p_S) = (-1,1,0)



Sensor-fixed (S) to earth-fixed (G) mapping

- The sensor rotates by ~-20° along the Z axis
  - $q_{S \leftarrow G}$ = (+.98, 0, 0, -.2) Provided by the sensor
  - $q_{G \leftarrow S} = q_{S \leftarrow G}$'=(0.98, 0, 0, +.2) Complex conjugate
  - p_G=quatrotate($q_{G \leftarrow S}$,p_S) = (-0.53, 1.31, 0)



Quaternion rotations

```
% Matlab/Octave code to rotate a vector along a quaternion
function result=quat_mult(q1,q2)
    result=[0 0 0 0];
    result(1) = (q1(1)*q2(1) -q1(2)*q2(2) -q1(3)*q2(3) -q1(4)*q2(4));
    result(2) = (q1(1)*q2(2) +q1(2)*q2(1) +q1(3)*q2(4) -q1(4)*q2(3));
    result(3) = (q1(1)*q2(3) -q1(2)*q2(4) +q1(3)*q2(1) +q1(4)*q2(2));
    result(4) = (q1(1)*q2(4) +q1(2)*q2(3) -q1(3)*q2(2) +q1(4)*q2(1));
end
%Quaternion multiplication without the .a component. Returns a vector
function result=quat_pointmult(q1,q2)
    result=[0 0 0];

    result(1) = (q1(1)*q2(2) +q1(2)*q2(1) +q1(3)*q2(4) -q1(4)*q2(3));
    result(2) = (q1(1)*q2(3) -q1(2)*q2(4) +q1(3)*q2(1) +q1(4)*q2(2));
    result(3) = (q1(1)*q2(4) +q1(2)*q2(3) -q1(3)*q2(2) +q1(4)*q2(1));
end
% Rotates vector v by quaternion q=[q0,q1,q2,q3]=[cos(a/2);sin(a/2)x,sin(a/2)y,sin(a/2)z]
% result = q*v*q' with q' the conjugate
function result = quat_rot(q,v)
    qconj = [0 0 0 0];                  % conjugate of the rotation quaternion
    qv = [0 0 0 0];                     % quaternion representation of the vector to rotate

    qv(1) = 0;
    qv(2) = v(1);
    qv(3) = v(2);
    qv(4) = v(3);
    qconj(1) = q(1);
    qconj(2) = -q(2);
    qconj(3) = -q(3);
    qconj(4) = -q(4);

    result =quat_pointmult(quat_mult(q,qv),qconj);
end
```

Quaternion rotations

- Matlab's aerospace toolbox **`quatrotate`** function behaves differently from the previous **`quat_rot`** function.
- Matlab's quatrotate rotates the coordinate system by the specified quaternion and returns the vector's coordinates in the new coordinate system
- Consider this positive rotation of ~20° around the Z axis:

```
>> q=[.98 0 0 .2];
>> quatrotate(q,[1 0 0])
ans =
    0.9200    -0.3918         0

>> quat_rot(q,[1 0 0])
ans =
    0.9204   +0.3920         0
```

- The two can be reconciled by passing the conjugate of the quaternion to the function.

```
>> quatrotate(quatconj(q),[1 0 0])
ans =
    0.9200   +0.3918         0
```