# Smarthub Sensor Data Calculation Functions & Equations

Kellen Gary

October 7, 2025

## Problem

When we go to calculate all time step based calculations (displacement, heading, trajectory), we lose a point of data due to us passing an array of length 4 into the function causing the resulting point length to be n-1 or 3. With this we are losing data that we need to preserve. Is there a way to update these equations so that this doesn't happen? Or is the velocity equation potentially wrong in that it shouldn't result in an array of length 4?

## 1 Displacement Calculation

### 1.1 Python Function

```python
def get_displacement_m(timeStamps, gyroLeft, gyroRight, diameter=WHEEL_DIAM_IN,
    dist_wheels=DIST_WHEELS_IN):
    gyroLeft = np.array(gyroLeft)  # Rotation of left wheel rps
    gyroRight = np.array(gyroRight)  # Rotation of right wheel rps
    timeStamps = np.array(timeStamps)  # Time (sec)

    dist_m = [0]
    for i in range(len(gyroRight) - 1):
        # Wheel rotation in time step:
        dx_r = (gyroLeft[i]+gyroRight[i])/2 * (timeStamps[i + 1] - timeStamps[i])
        # Change in displacement over time step:
        dx_m = dx_r * (diameter * IN_TO_M / 2)
        # Append last change to overall Displacement:
        dist_m.append(dx_m + dist_m[-1])
    return dist_m
```

### 1.2 Mathematical Equations

Displacement at each time step is calculated as:

$$\Delta x_r = \frac{\text{rot}_l[i] + \text{rot}_r[i]}{2} \times (t_{i+1} - t_i)$$
$$\Delta x_m = \Delta x_r \times \frac{d \times \text{IN\_TO\_M}}{2}$$
$$\text{displacement}[i+1] = \Delta x_m + \text{displacement}[i]$$

where $d$ is the wheel diameter and IN_TO_M converts inches to meters.

## 2 Velocity Calculation

### 2.1 Python Function

```python
def get_velocity_m_s(timeStamps, gyroLeft, gyroRight, diameter=WHEEL_DIAM_IN, dist_wheels=DIST_WHEELS_IN):
    gyroLeft = np.array(gyroLeft)  # Rotation of left wheel (converted to rps by Arduino)
    gyroRight = np.array(gyroRight)  # Rotation of right wheel (converted to rps by Arduino)
    timeStamps = np.array(timeStamps)  # Time (sec)

    vel_ms = [0]
    for i in range(len(gyroRight) - 1):
        # Right wheel velocity:
        v_r = (gyroRight[i]) * diameter/2*IN_TO_M
        # Left wheel velocity:
        v_l = (gyroLeft[i]) * diameter/2*IN_TO_M
        # Velocity of wheelchair over time:
        v_curr = (v_r+v_l)/2
        # Append last change to overall Displacement:
        vel_ms.append(v_curr)
    return vel_ms
```

### 2.2 Mathematical Equations

Velocity at each time step is calculated as:

$$\mathrm{v}_r = \mathrm{rot}_r[i] \times \frac{d}{2} \times \text{IN\_TO\_M}$$

$$\mathrm{v}_l = \mathrm{rot}_l[i] \times \frac{d}{2} \times \text{IN\_TO\_M}$$

$$\mathrm{velocity}[i] = \frac{\mathrm{v}_r + \mathrm{v}_l}{2}$$

where $d$ is the wheel diameter in inches, and IN_TO_M is the conversion factor from inches to meters.

## 3 Heading Calculation

### 3.1 Python Function

```python
def get_heading_deg(timeStamps, gyroLeft, gyroRight, diameter=WHEEL_DIAM_IN, dist_wheels=DIST_WHEELS_IN):
    gyroLeft = np.array(gyroLeft)  # Rotation of left wheel (converted to rps by Arduino)
    gyroRight = np.array(gyroRight)  # Rotation of right wheel (converted to rps by Arduino)
    timeStamps = np.array(timeStamps)  # Time (sec)

    heading_deg = [0]
    for i in range(len(gyroRight) - 1):
        # Angular Velocity in time step (rotating left is positive):
        w = ((gyroRight[i]-gyroLeft[i]) * diameter*IN_TO_M/2) / (dist_wheels*IN_TO_M)
        # Change in heading angle over time step:
        dh = w * (timeStamps[i + 1] - timeStamps[i])
```

```
        # convert to degrees:
        dh = dh*180/np.pi
        # Append last change to overall heading angle:
        heading_deg.append(dh + heading_deg[-1])
    return heading_deg
```

### 3.2 Mathematical Equations

Heading (in degrees) at each time step is:

$$\begin{aligned}
\omega &= \frac{(\text{rot}_r[i] - \text{rot}_l[i]) \times d \times \text{IN_TO_M}/2}{\text{wheelDistance} \times \text{IN_TO_M}} \\
\Delta t &= t_{i+1} - t_i \\
\Delta h &= \omega \times \Delta t \\
\Delta h_{\text{deg}} &= \Delta h \times \frac{180}{\pi} \\
\text{heading}[i+1] &= \Delta h_{\text{deg}} + \text{heading}[i]
\end{aligned}$$

# 4 Trajectory Calculation

### 4.1 Python Function

```
def get_top_traj(disp_m, vel_ms, heading_deg, timeStamps, diameter=WHEEL_DIAM_IN,
    dist_wheels=DIST_WHEELS_IN):
    x, y = [], []
    dx, dy = 0, 0

    for i in range(len(disp_m) - 1):
        '''
        dr = disp_m[i + 1] - disp_m[i]
        dh = heading_deg[i] * np.pi / 180  # radian
        dx += dr * np.cos(dh)
        dy += dr * np.sin(dh)
        '''
        dx += vel_ms[i]*np.cos(heading_deg[i]*np.pi/180) * (timeStamps[i + 1] -
            timeStamps[i])
        dy += vel_ms[i]*np.sin(heading_deg[i]*np.pi/180) * (timeStamps[i + 1] -
            timeStamps[i])
        x.append(dx)
        y.append(dy)
    traj = [[x[i], y[i]] for i in range(len(x))]
    return traj
```

### 4.2 Mathematical Equations

Trajectory $(x, y)$ over time is:

$$\begin{aligned}
\Delta t &= t_{i+1} - t_i \\
\theta &= \text{heading}[i] \times \frac{\pi}{180} \\
\Delta x &= \text{velocity}[i] \times \cos(\theta) \times \Delta t \\
\Delta y &= \text{velocity}[i] \times \sin(\theta) \times \Delta t \\
\text{x}[i+1] &= \text{x}[i] + \Delta x \\
\text{y}[i+1] &= \text{y}[i] + \Delta y
\end{aligned}$$

# 5 Data Length Analysis & Solutions

## 5.1 The n-1 Problem

The current implementation results in arrays of length $n - 1$ because:

- Input arrays have length $n = 4$ (4 time stamps and gyro values)
- Loop runs from $i = 0$ to $len(rot\_r) - 1 = 3$, so only 3 iterations
- Each iteration appends one calculated value
- Result: 3 calculated values + 1 initial value = 4 total values for some functions