invention bootcamp 2022

Using acceleration to count steps with the Circuit Playground Express

Learning objectives

These slides should help you to

- Explain the physical significance of the acceleration signal from the CPX
- Combine acceleration components to get the total acceleration
- Apply exponentially-weighted averaging to reduce high frequency noise
- Apply a simple algorithm to count steps from the total acceleration from a CPX in your pocket

What is acceleration?

Acceleration is the rate of change of velocity

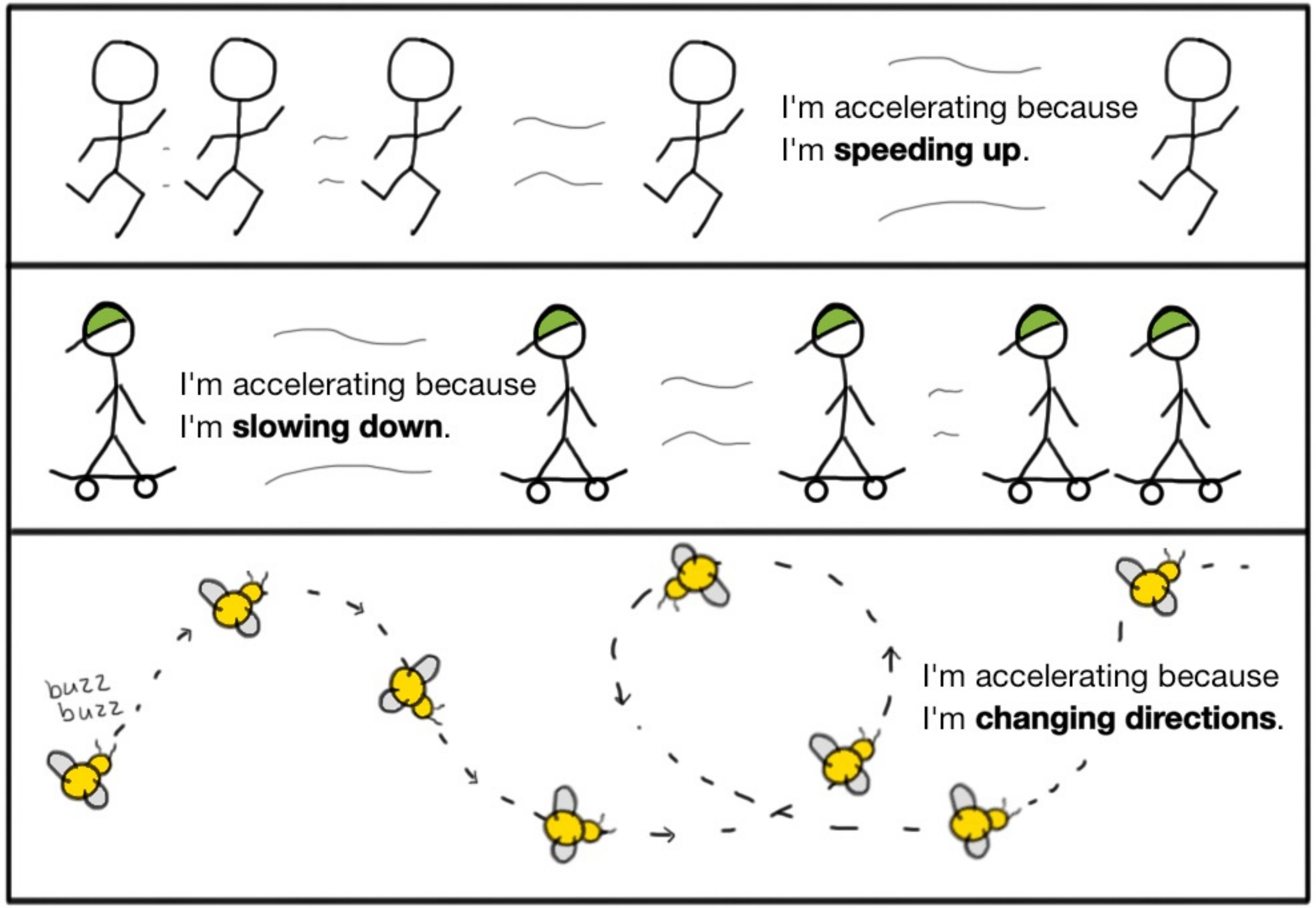
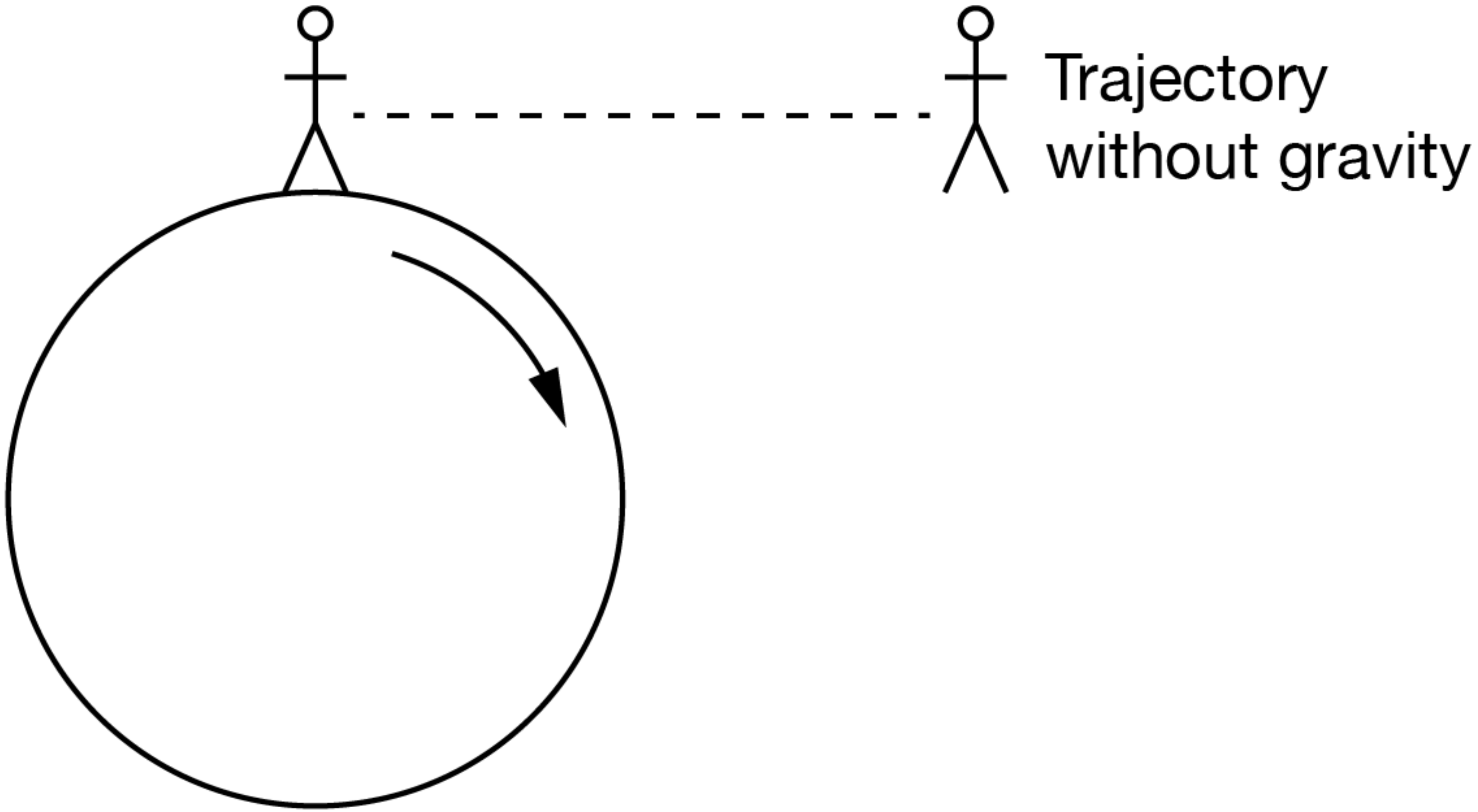
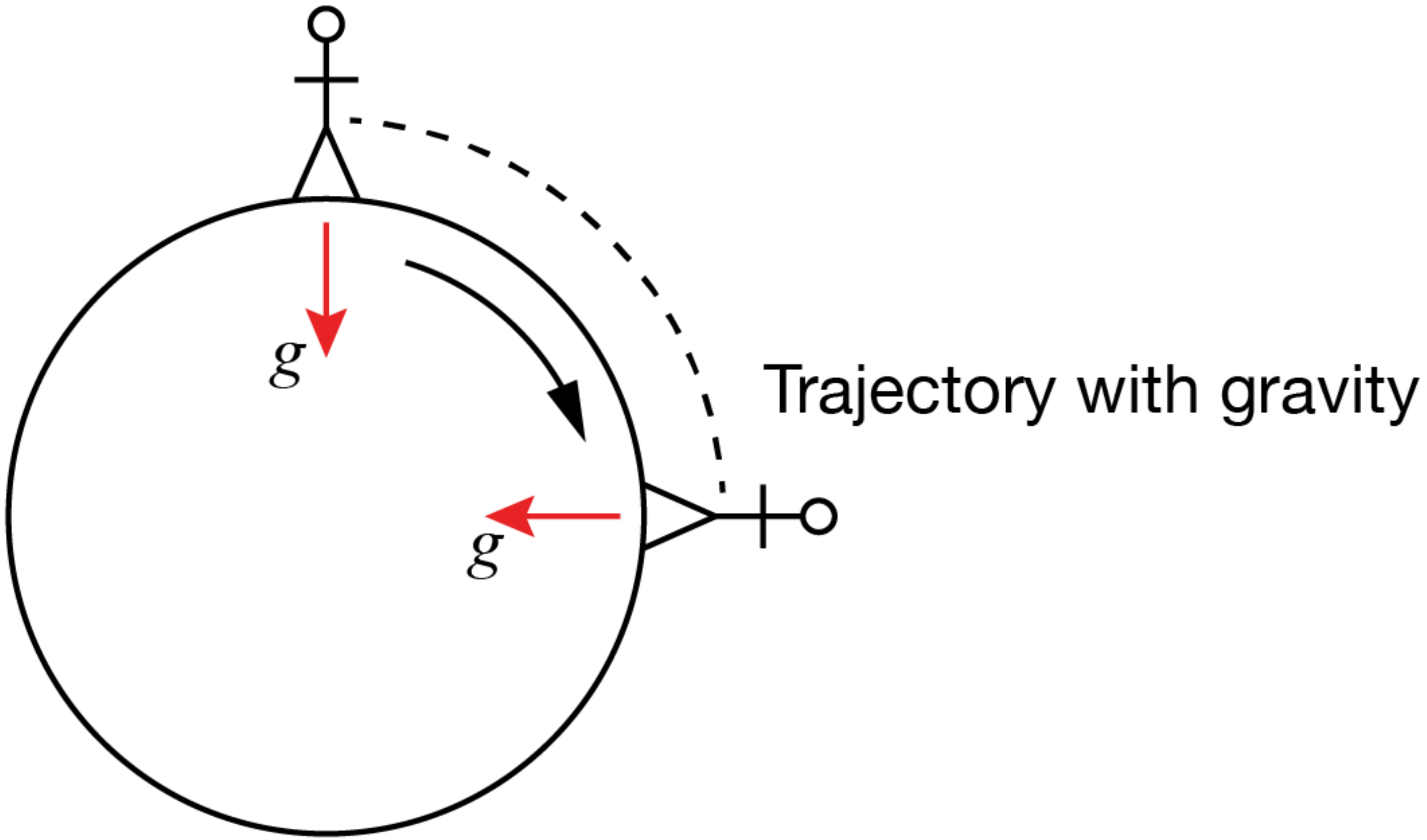


Image from
https://www.khanacademy.org/science/physics/one-dimensional-motion/acceleration-tutorial/a/acceleration-article

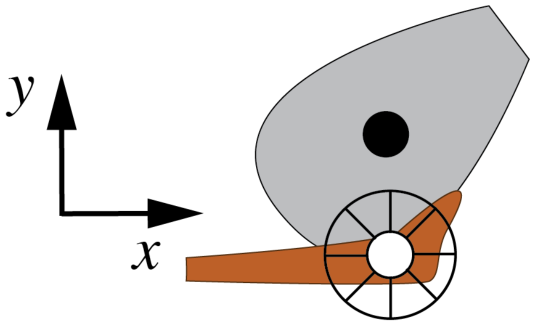
Gravity acts to accelerate us



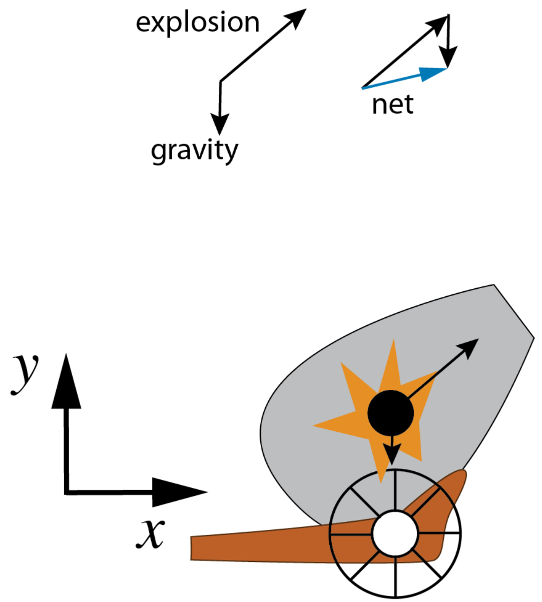
Gravity acts to accelerate us



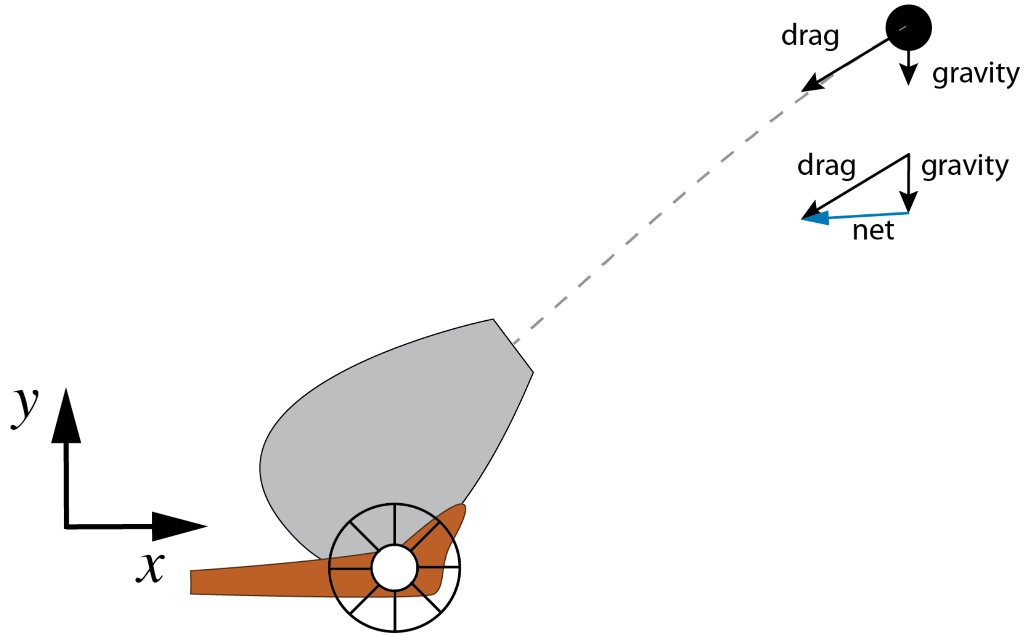
Acceleration is a vector



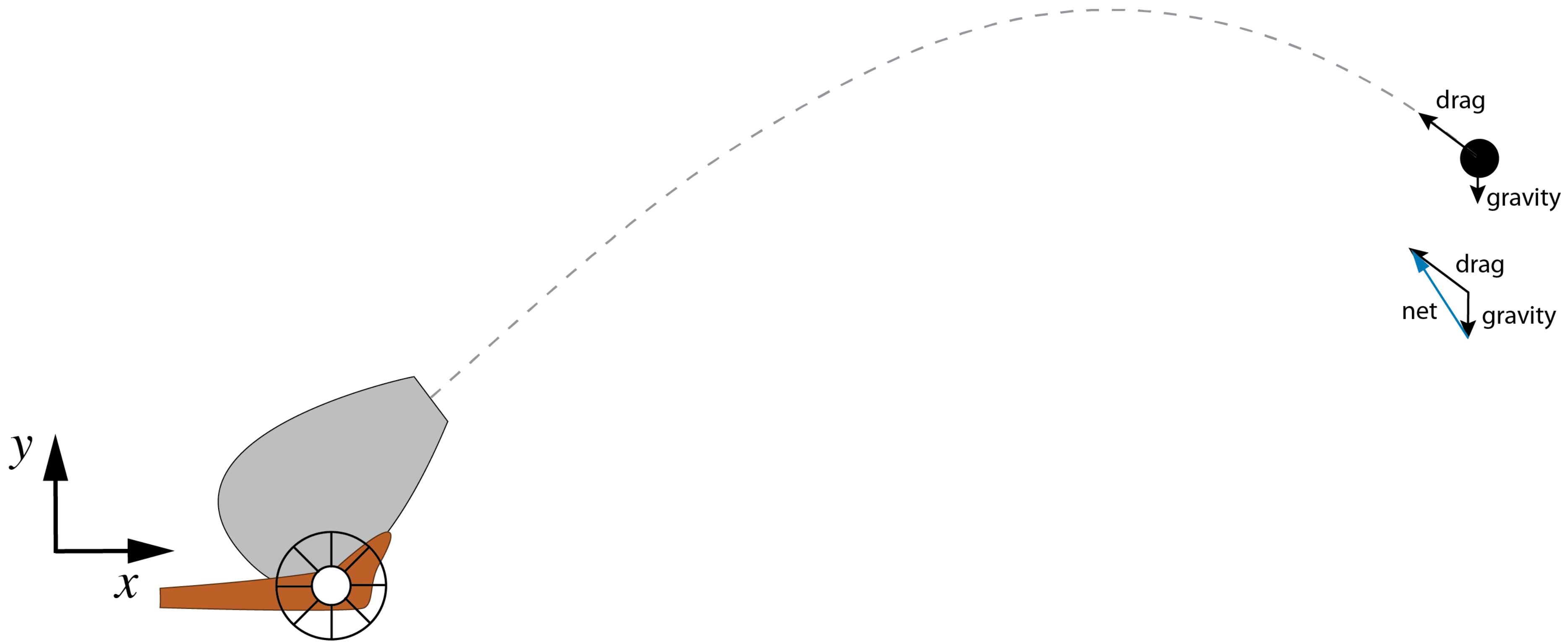
Acceleration is a vector



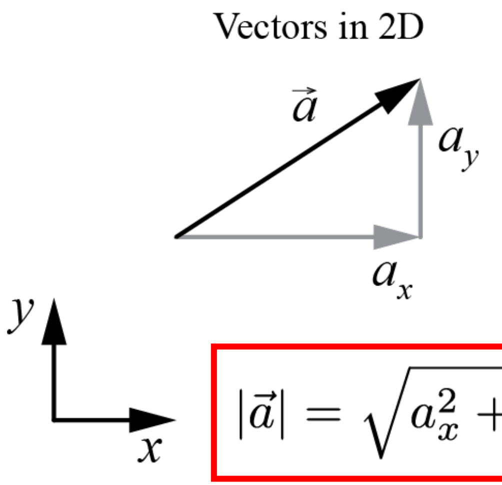
Acceleration is a vector



Acceleration is a vector

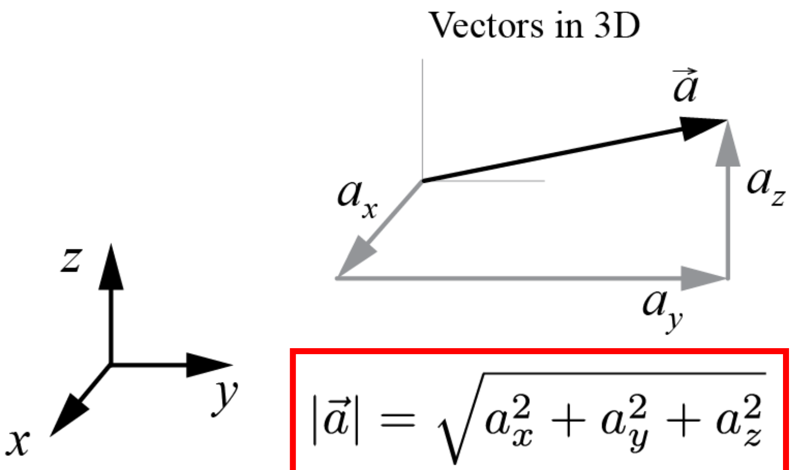


Vectors have direction and magnitude



$$|\vec{a}| = \sqrt{a_x^2 + a_y^2}$$

magnitude



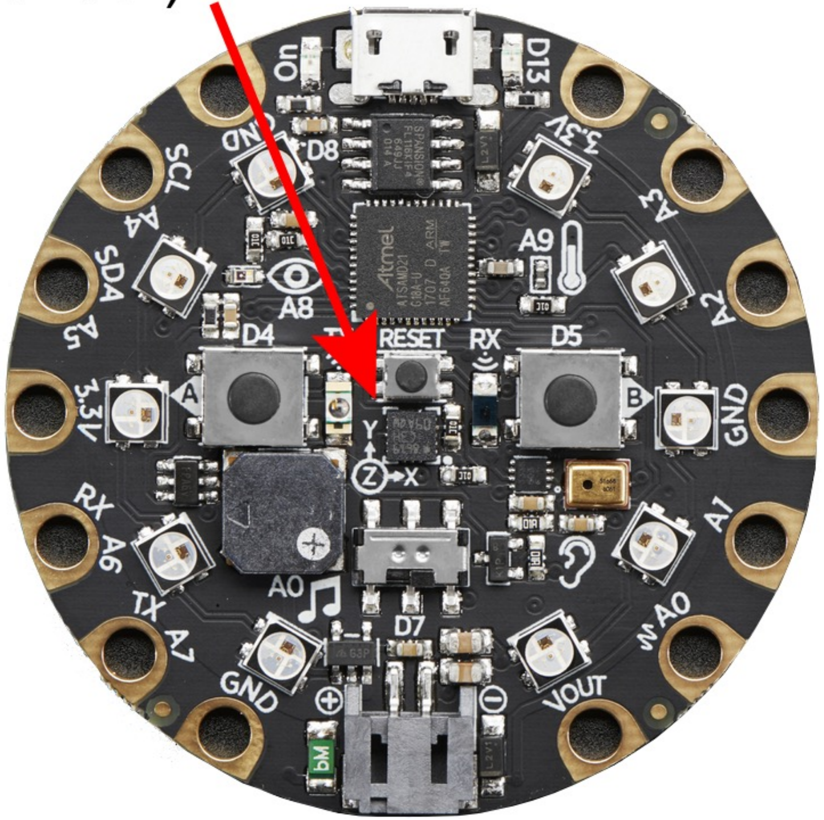
$$|\vec{a}| = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

magnitude

Measuring acceleration with the Circuit Playground Express

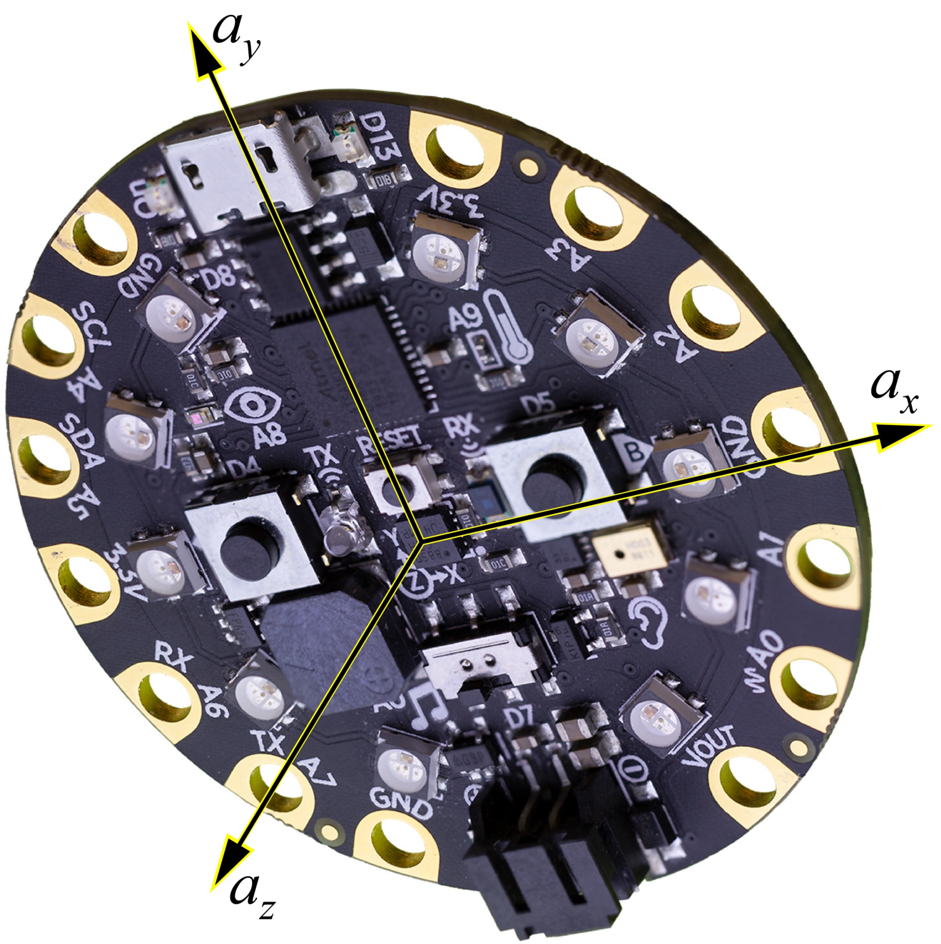
Accelerometer on the CPX

Accelerometer
(motion sensor)



Notice the coordinate triad (x,y,z)

Accelerometer on the CPX



Code to read acceleration components

```
float ax, ay, az, aTot;

ax = CircuitPlayground.motionX();
ay = CircuitPlayground.motionY();
az = CircuitPlayground.motionZ();

aTot = sqrt(ax*ax + ay*ay + az*az);
```

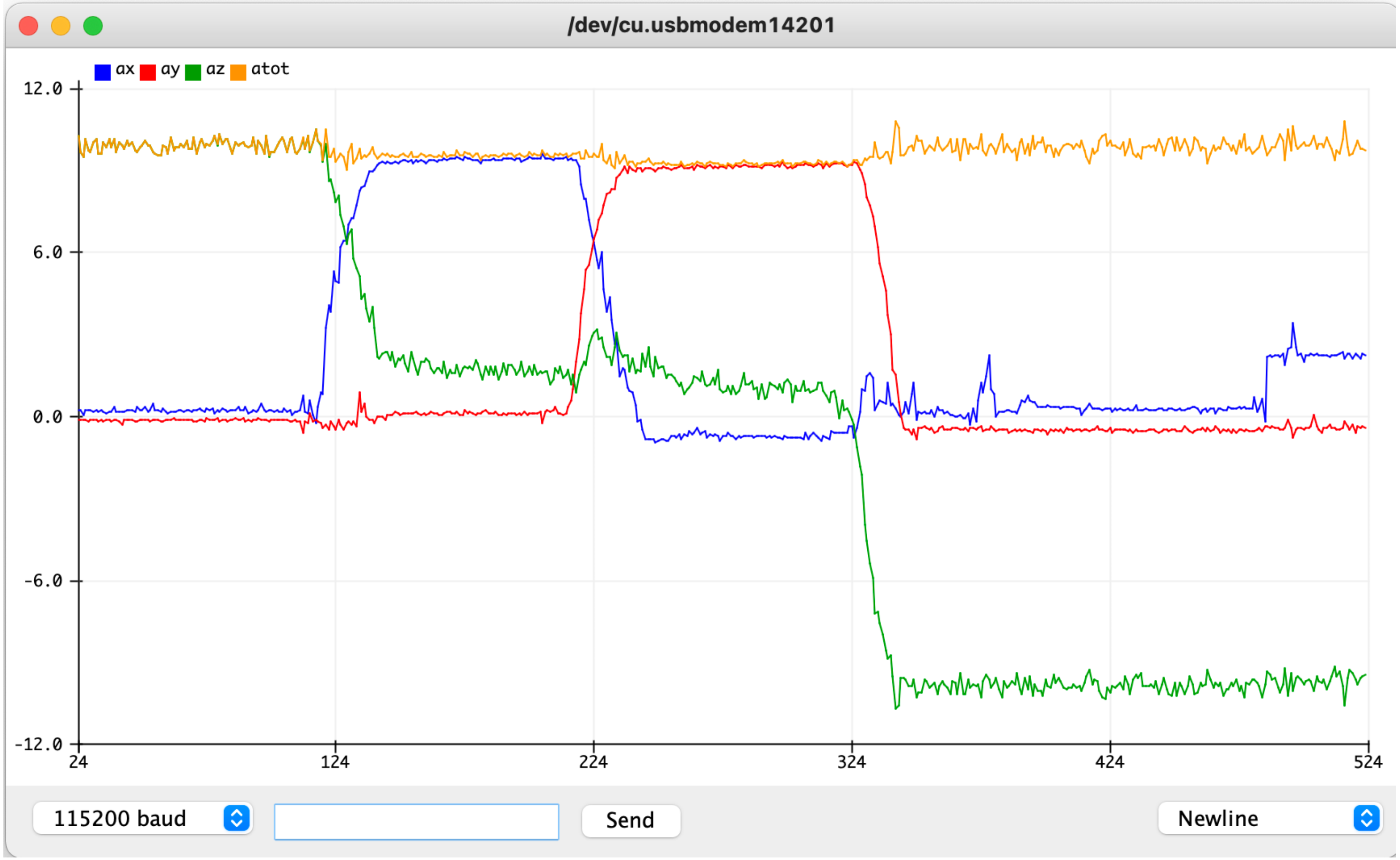
Look at accelerometer output

Download and run `demo_accelerometer.ino` from the public website for the camp

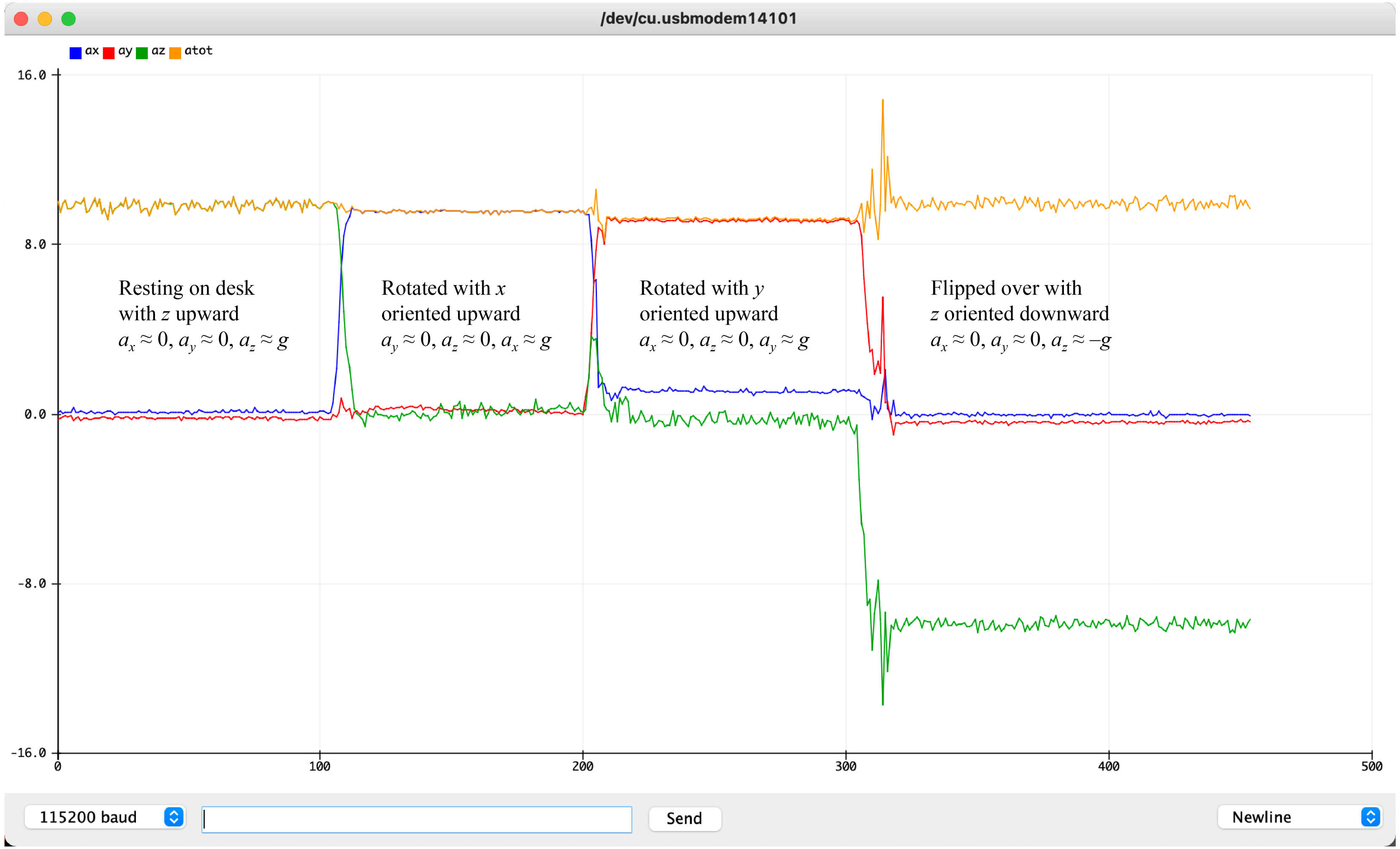
Text output to the Serial Monitor

ax	ay	az	atot
0.07	-0.31	9.93	9.94
0.06	-0.27	9.59	9.59
0.02	-0.28	9.74	9.75
0.01	-0.30	9.91	9.92
0.04	-0.33	9.79	9.80
0.04	-0.24	9.91	9.92
0.01	-0.32	9.66	9.66
-0.11	-0.14	10.03	10.03
0.03	-0.25	9.87	9.87

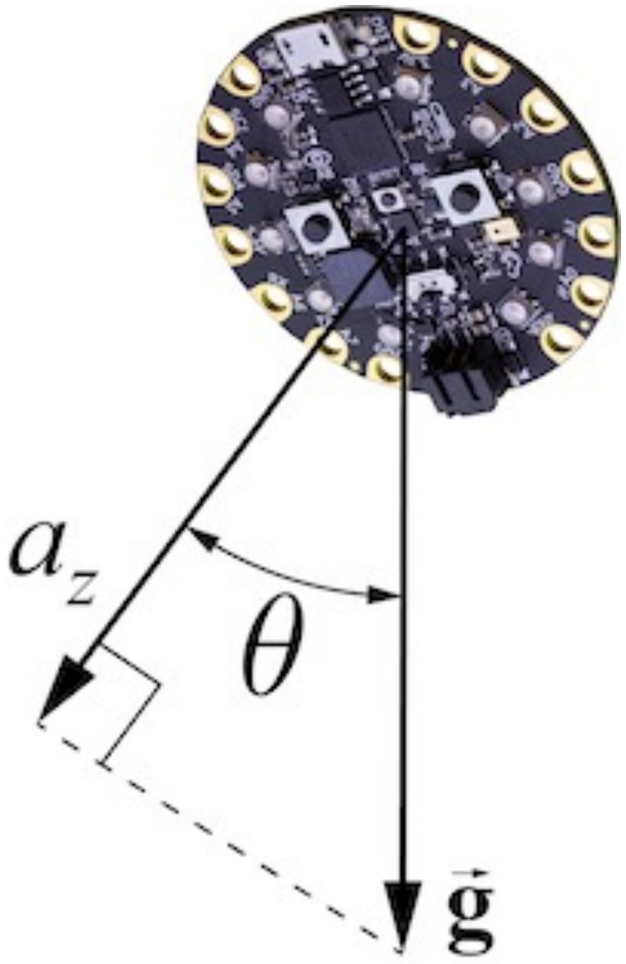
Watch dynamic data on the Serial Plotter



Watch dynamic data on the Serial Plotter



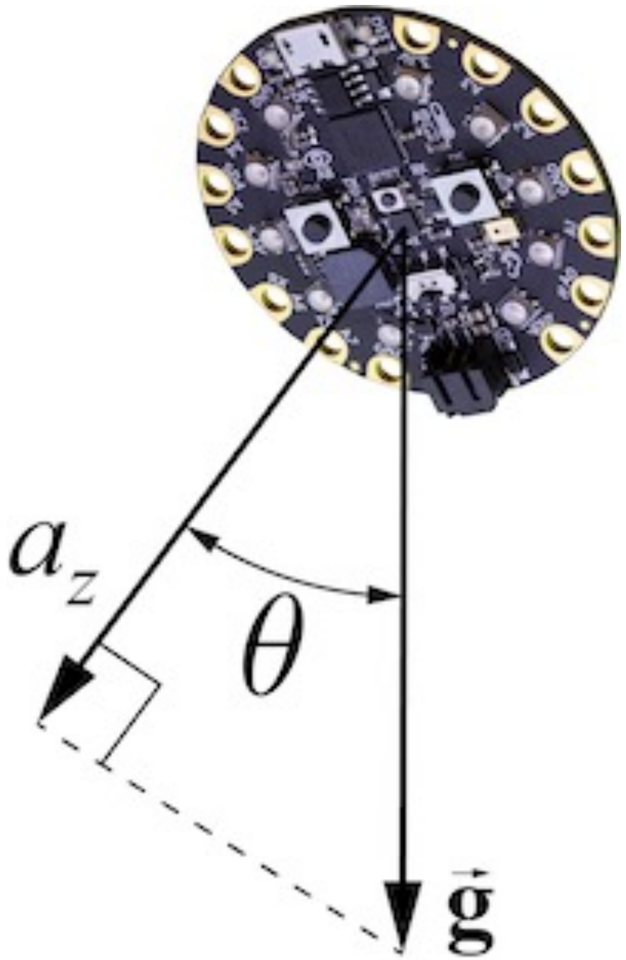
Gravity is an acceleration



The acceleration of gravity always acts down toward the center of the earth

The acceleration of gravity acts on all objects, even when they are stationary

Gravity is an acceleration



Use acceleration components to find the direction of “down” when CPX is stationary

```
float ax, ay, az, aTot;
float theta

ax = CircuitPlayground.motionX();
ay = CircuitPlayground.motionY();
az = CircuitPlayground.motionZ();
aTot = sqrt(ax*ax + ay*ay + az*az);

theta = (180.0/PI) * acos(az/aTot);
```

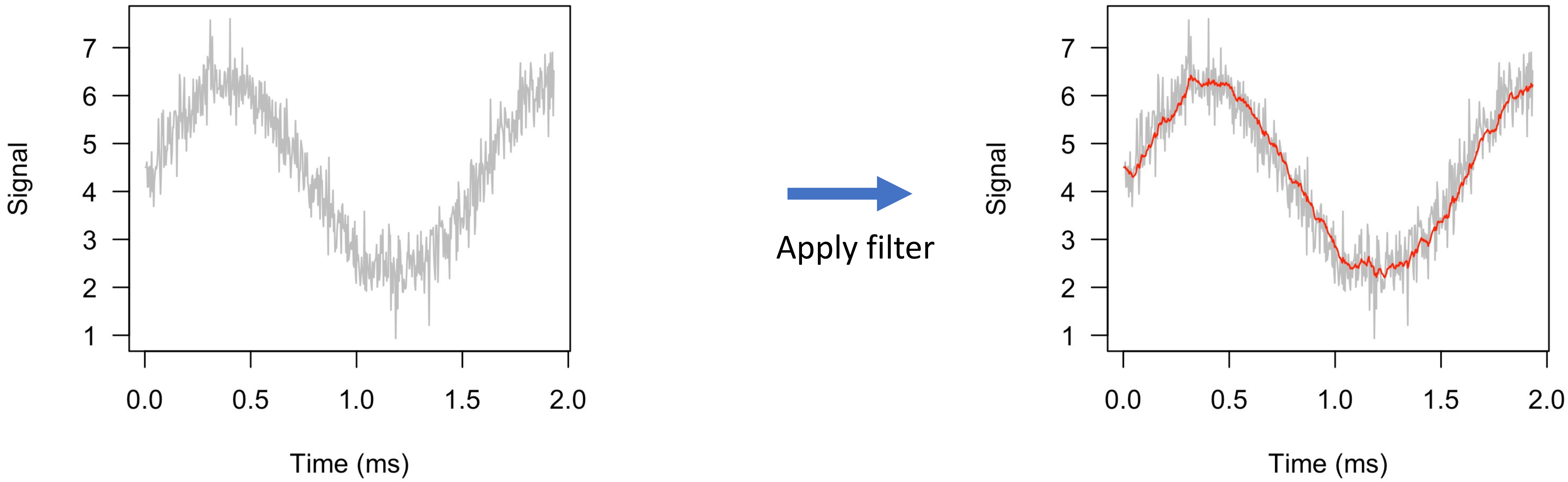
Smoothing accelerometer data

Smoothing is achieved by applying a filter that reduces high frequency parts of the signal

High frequency noise is not helpful

Filtering the acceleration data reduces noise

- A low-pass filter eliminates higher frequencies
- An exponentially-weighted average is efficient and easy to implement



A low-pass filter reduces high frequency noise

An exponentially-weighted average is efficient and easy to implement low-pass filter

- Average the latest reading with earlier readings
- Influence of older readings decreases with age of the reading

Let v_i be the value of reading i

Let n be the last reading taken

Let α be a parameter that controls the smoothing, $0 < \alpha \leq 1$

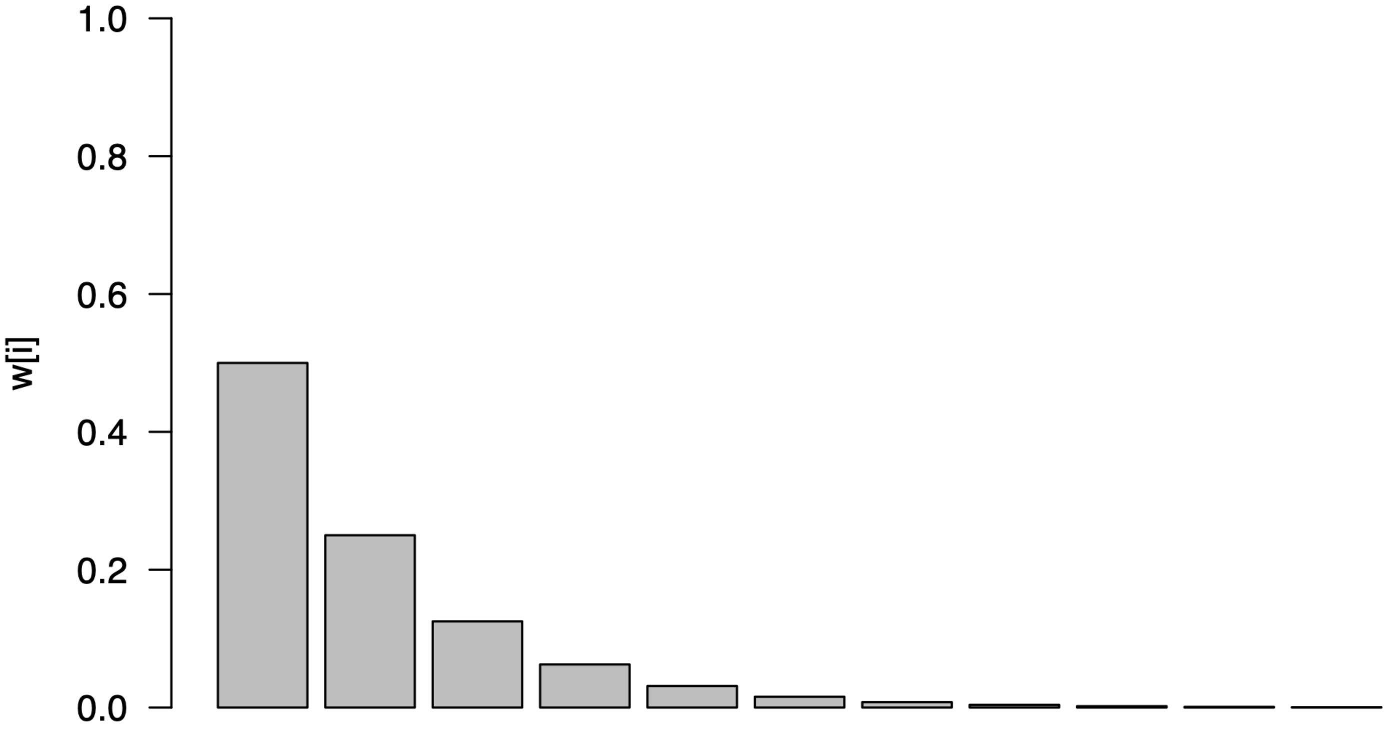
$$\bar{v}_n = v_n \qquad (i = n)$$

$$\bar{v}_i = \alpha v_i + (1 - \alpha)\bar{v}_{i-1} \qquad (i = n-1, n-2, \ldots)$$

Exponentially-weighted average

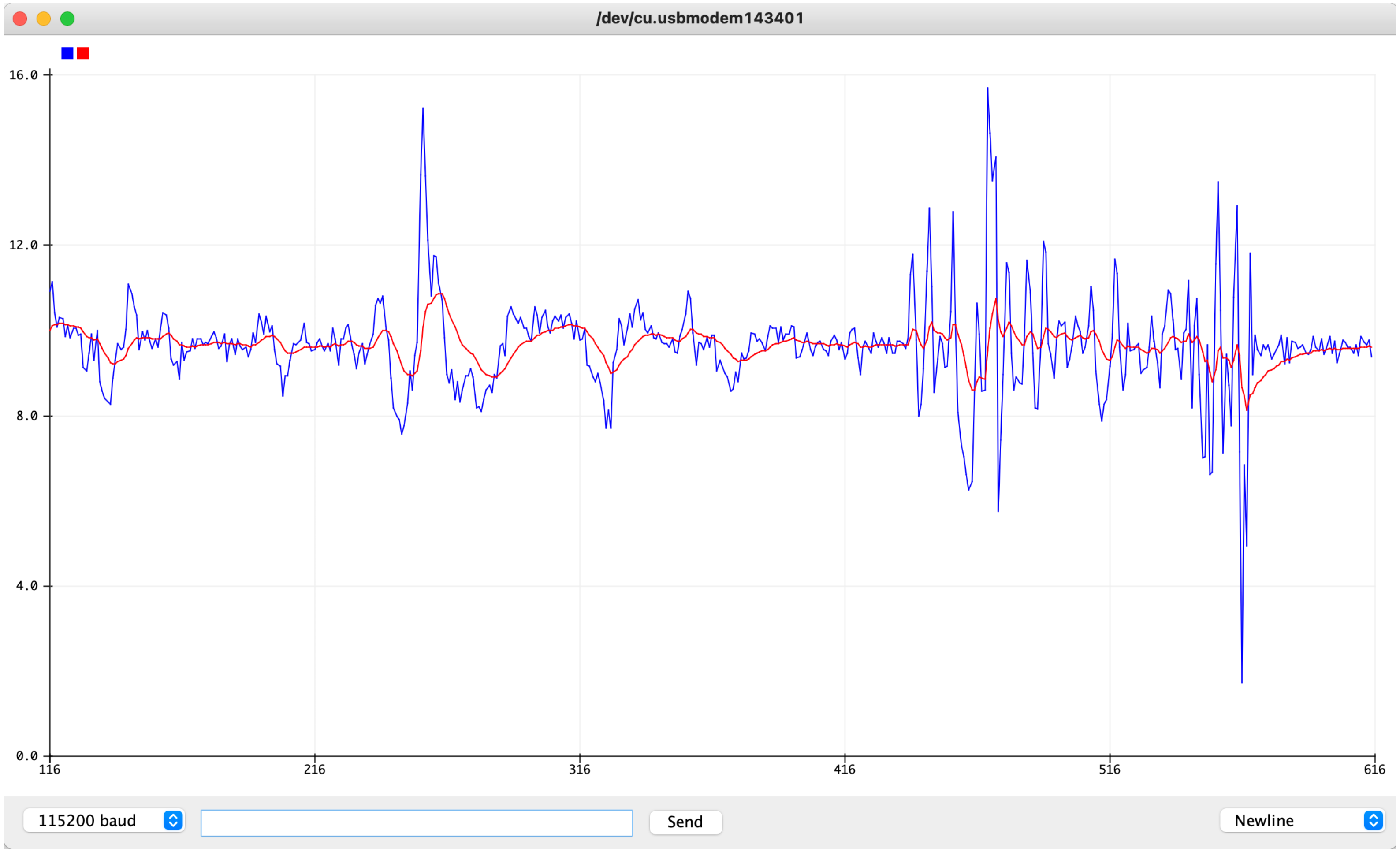
Example: $\alpha = 0.5$

$$\bar{v} = 0.5 \times v_i + 0.25 \times v_{i-1} + 0.125 \times v_{i-2} + 0.0625 \times v_{i-3} + \cdots$$

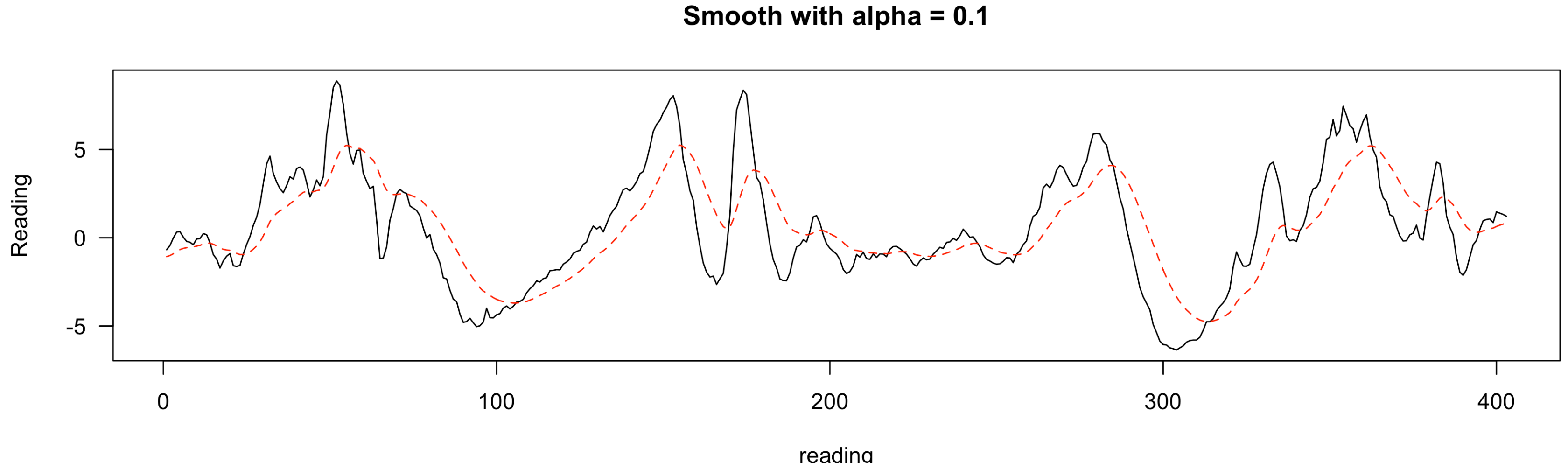


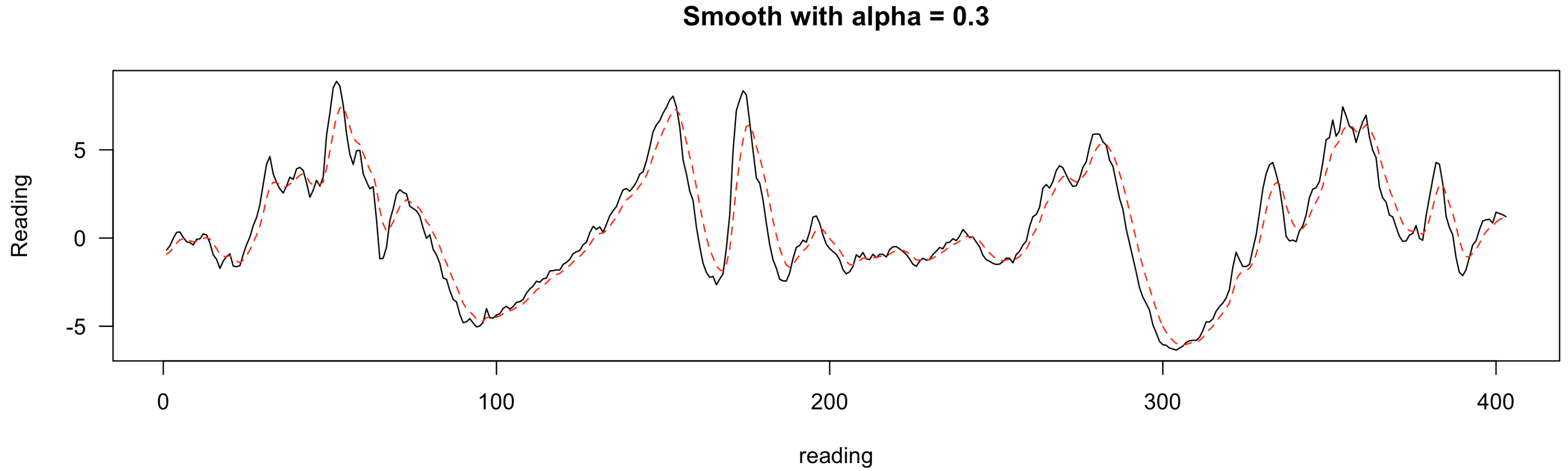
Practice

Run demo_accelerometer_smoothed.ino



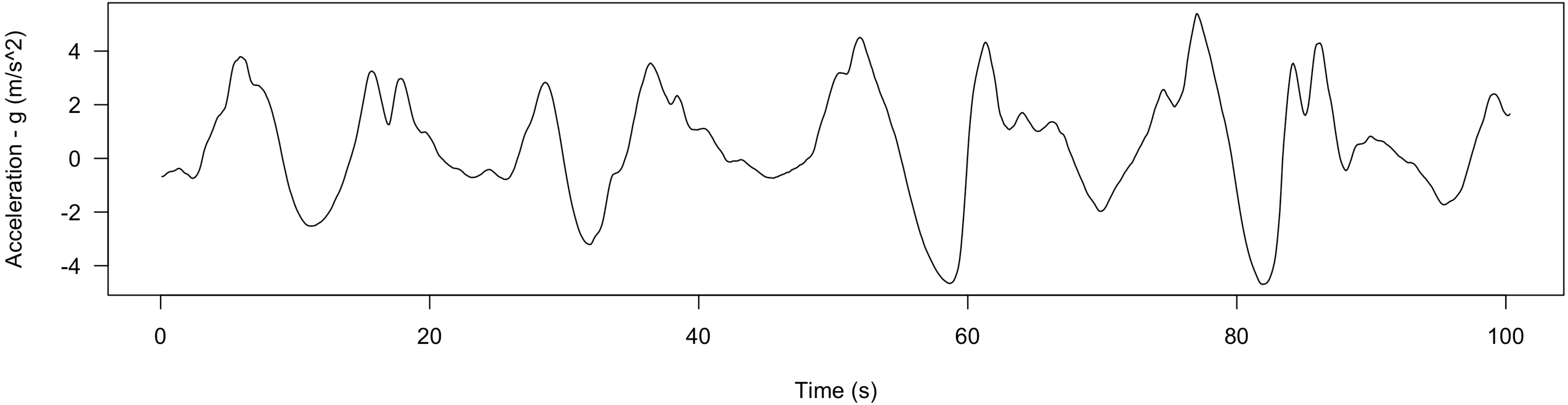
Smoothing can be adjusted with α



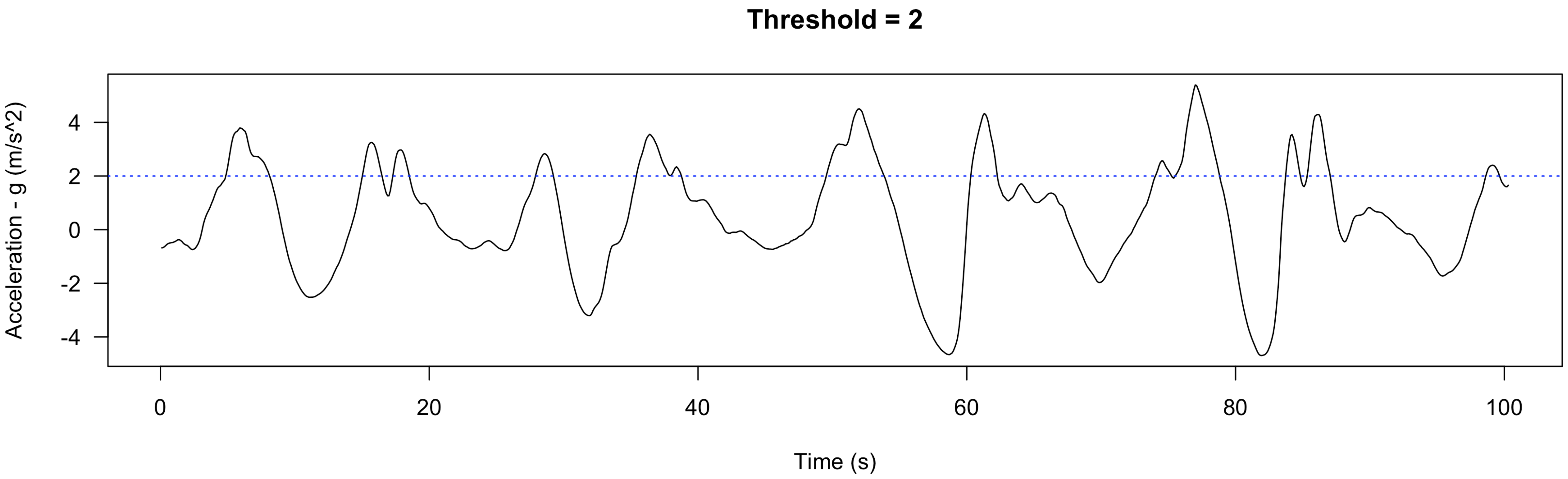


A simple pedometer algorithm

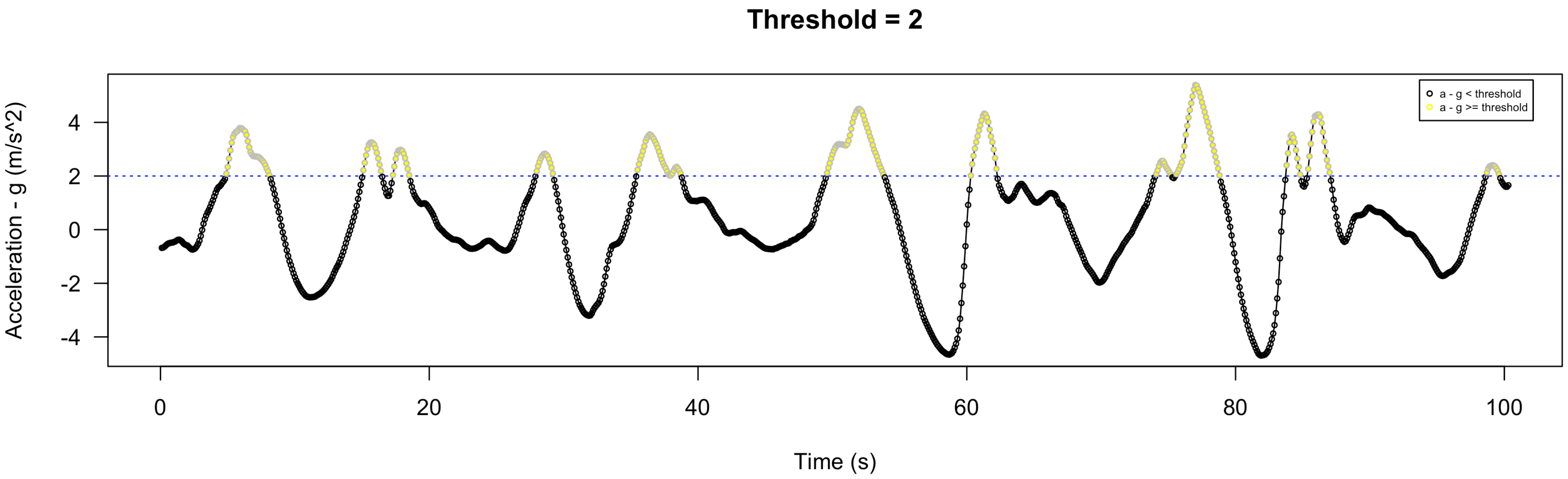
Raw data: a_{tot} - g



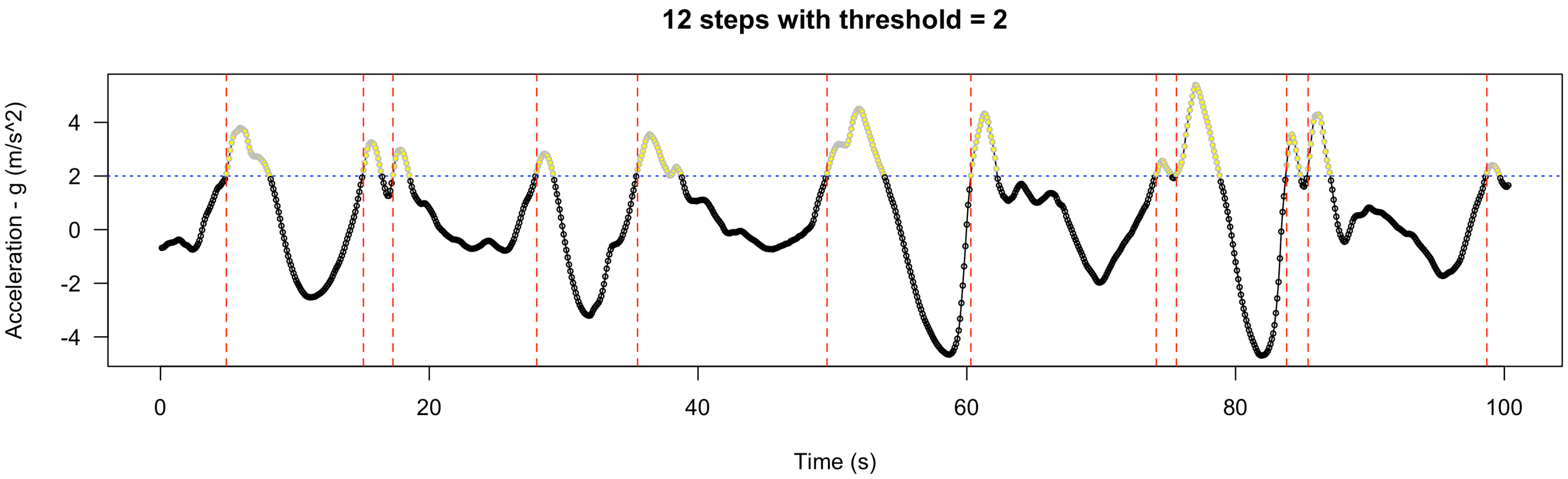
Define a threshold for high acceleration



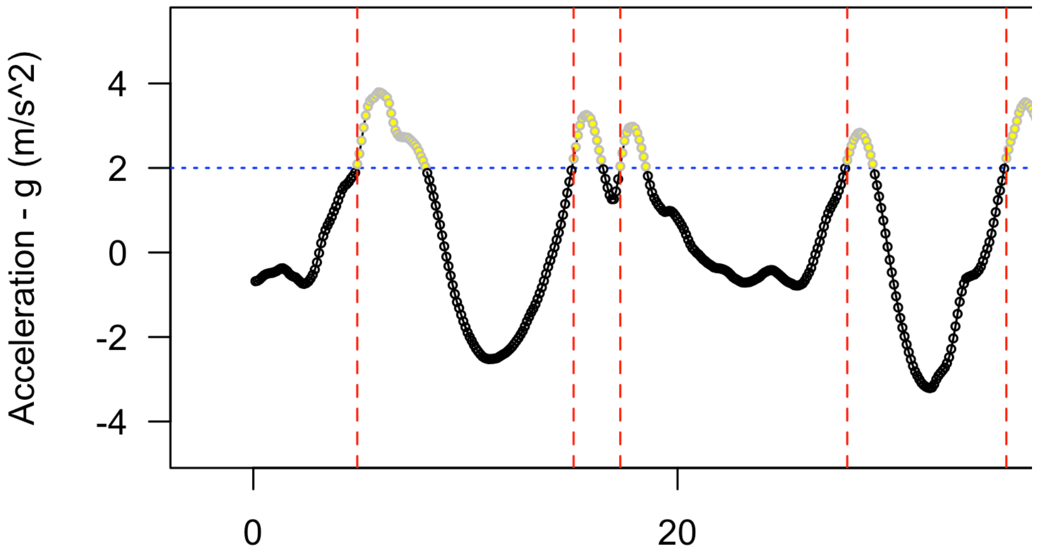
Identify points above and below the threshold



Count a step when acceleration first crosses above the threshold



Count a step when acceleration first crosses above the threshold

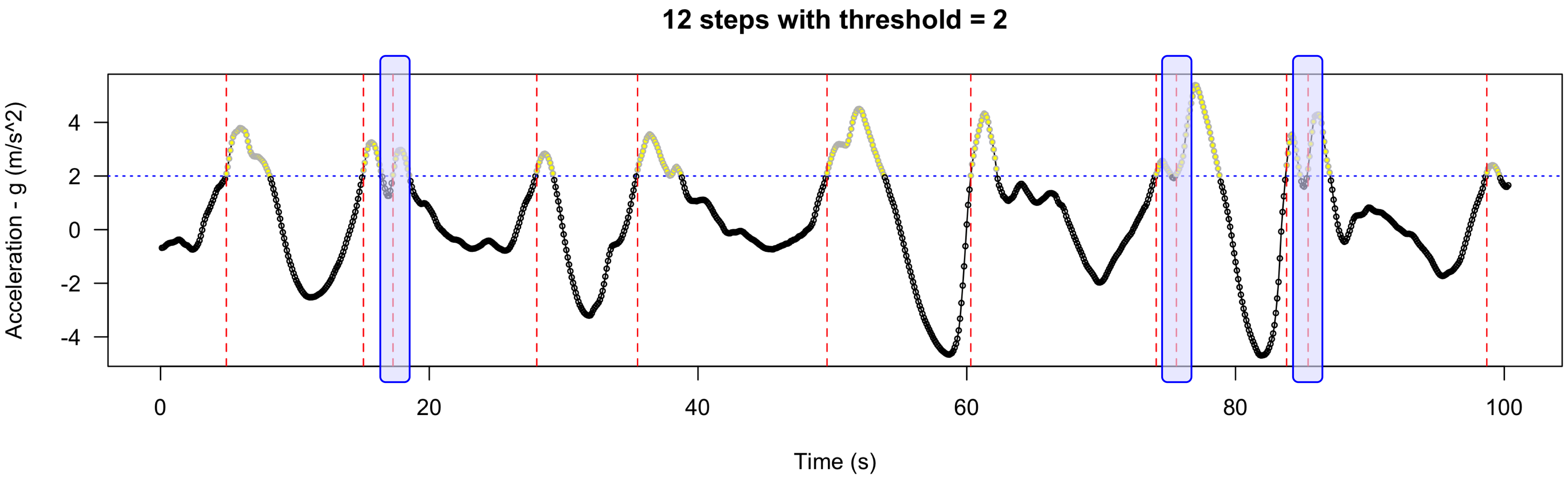



```
int count_step(float a, float threshold) {

  int n;
  static boolean lastWasLow = true;

  n = 0;
  if ( a > threshold  ) {
    if ( lastWasLow )
      n = 1;
    }
    lastWasLow = false;
  } else {
    lastWasLow = true;
  }
  return (n);
}
```

Modification to avoid counting quick changes



See code in OLED_pedometer.ino

Pedometer algorithm is affected by ...

- Type of walking: smooth, jumpy, slow, quick
- Algorithm variables
  - Frequency of reading the acceleration
  - Smoothing parameter, α
  - Threshold for counting steps
  - Time delay used to avoid counting quick changes

You will need to experiment