Python, Motion Sensors and Data Fusion

PyCon UK, 15 September 2016

Stephen Simmons

mail@stevesimmons.com

https://github.com/stevesimmons/pyconuk-motion-sensor-data-fusion

Goals for today…

Why play with motion sensors?

tiny sensors + fun applications + cheap

Goals for today…

Why play with motion sensors?

tiny sensors + fun applications + cheap
+ smart algorithms **in python**

Goals for today…

Why play with motion sensors?

tiny sensors + fun applications + cheap
+ smart algorithms **in python**

Motion sensors then and now – 1852, 1965, 2016

- Example #1 – Wireless motion logging with BBC micro:bit

Goals for today…

Why play with motion sensors?

tiny sensors + fun applications + cheap
+ smart algorithms **in python**

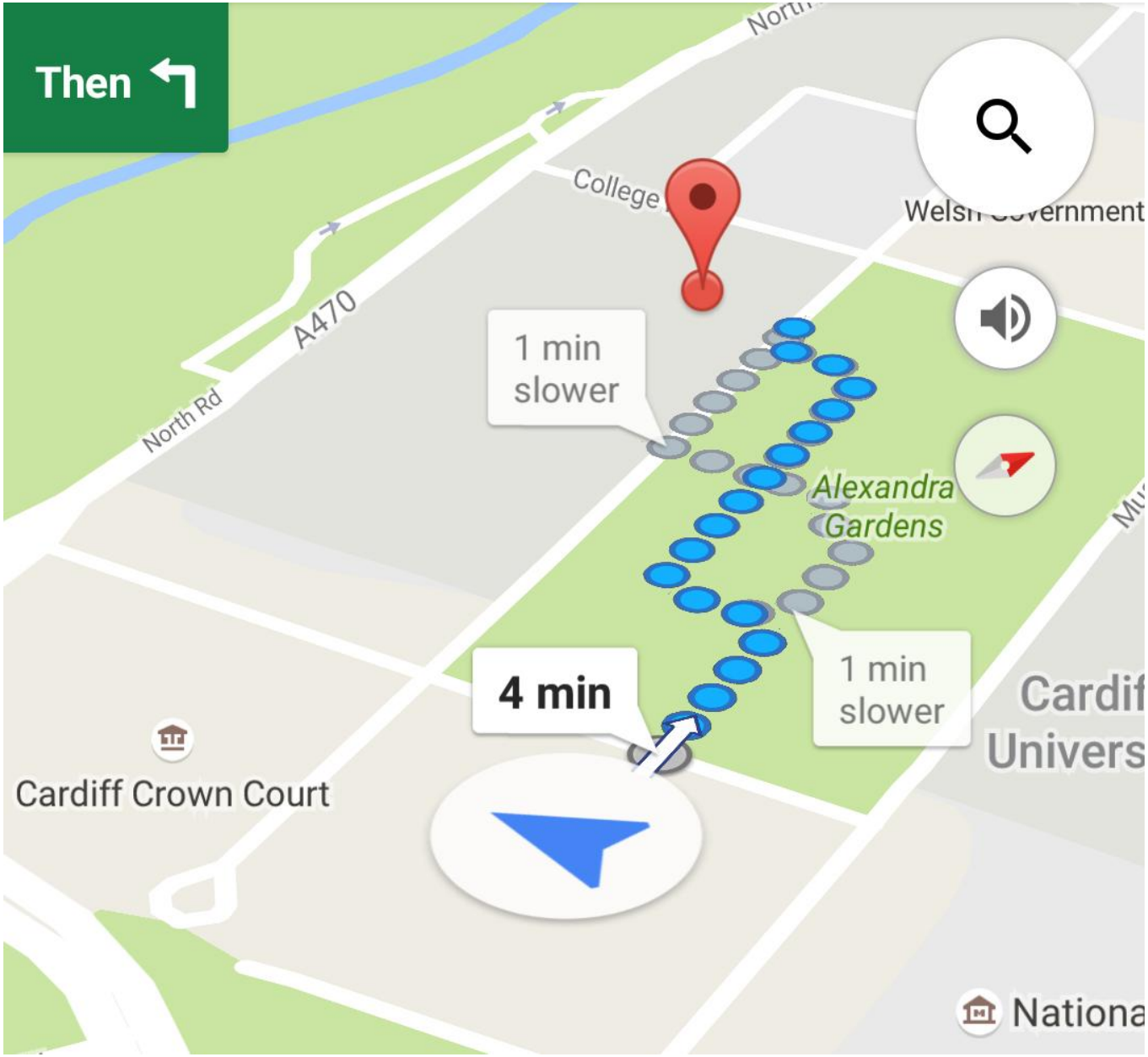
Motion sensors then and now – 1852, 1965, 2016

- Example #1 – Wireless motion logging with BBC micro:bit

Data fusion – combining data from multiple sensors

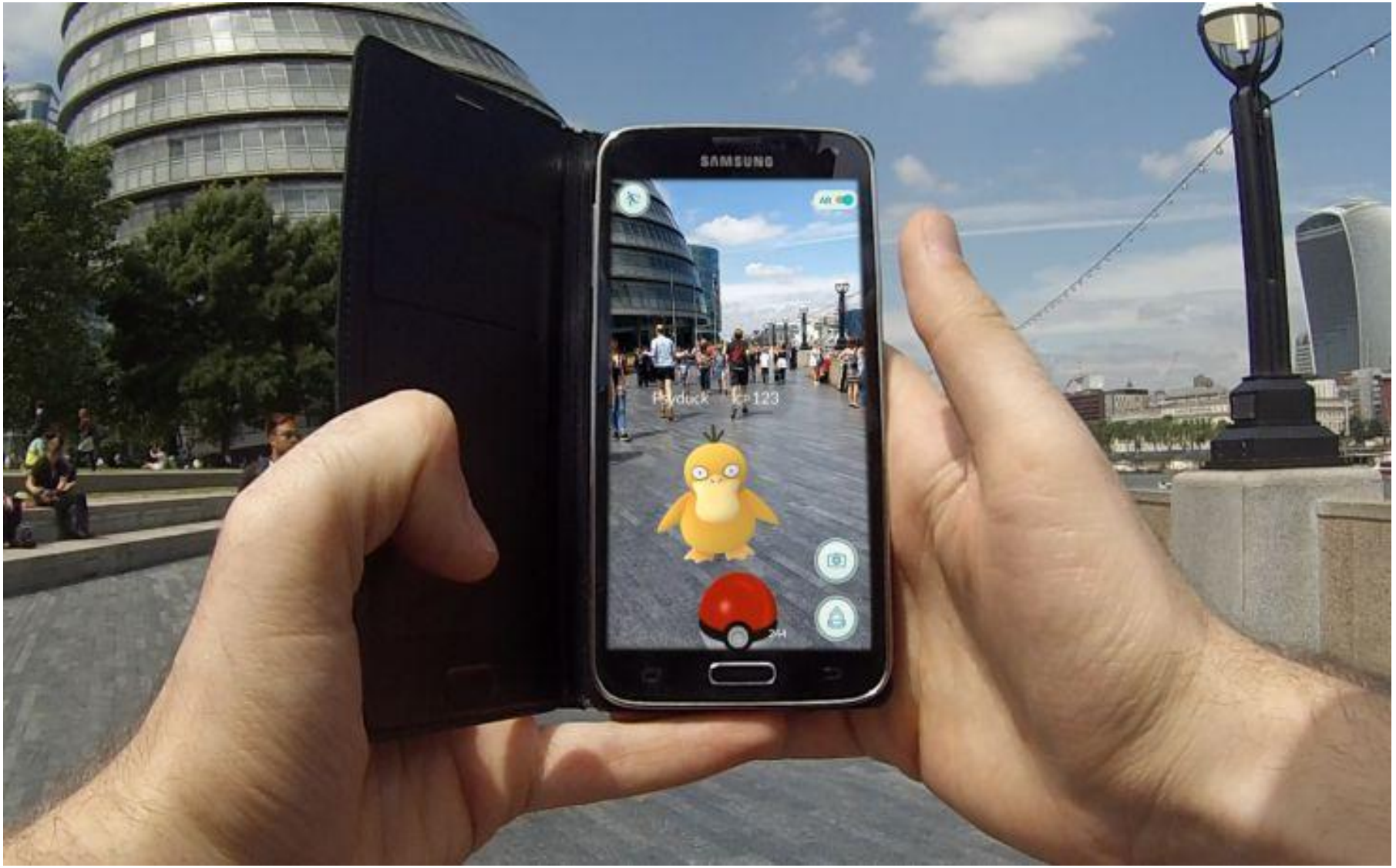
- Example #2 – tilt-compensated digital compass
- Example #3 – 9 DoF IMU from mobile phone data

Mapping and navigation



VR…

AR…



Motion capture…

Motion capture…

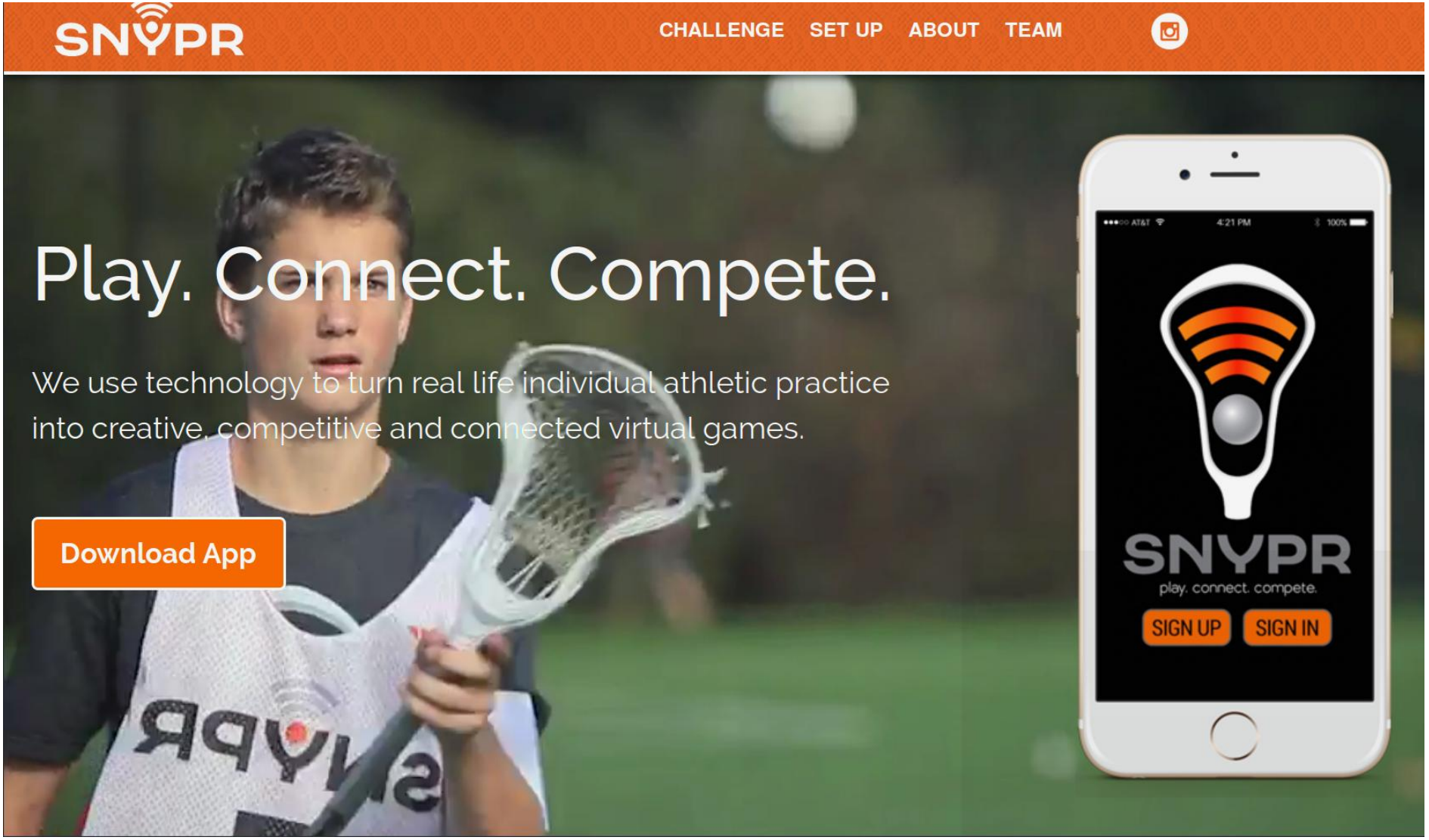
Control and stabilisation…

Automotive…

“Quantified self”…

Sports coaching…

Sports coaching…



Sports coaching…

Types of sensors

Accelerometer
Linear acceleration plus gravity

Magnetometer
Local magnetic field

Gyroscope
Angular rotation rate

GPS
Absolute location (outdoor)

Pressure
Altitude (outdoor); relative height (indoor)

+ light, sound, touch, temperature, humidity, heart rate, …

Types of sensors

Accelerometer

Linear acceleration plus gravity

Magnetometer

Local magnetic field

Gyroscope

Angular rotation rate

GPS

Absolute location (outdoor)

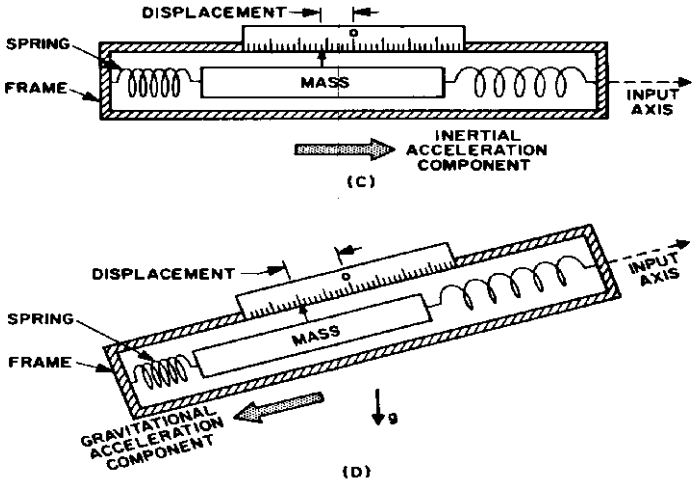
Pressure

Altitude (outdoor); relative height (indoor)

+ light, sound, touch, temperature, humidity, heart rate, …

$$A_{xyz} = A_{sensor} - R\ g$$

R is rotation matrix converting gravity vector g to sensor's local (x,y,z) coordinates



Types of sensors

Accelerometer
Linear acceleration plus gravity

Magnetometer
Local magnetic field

Gyroscope
Angular rotation rate

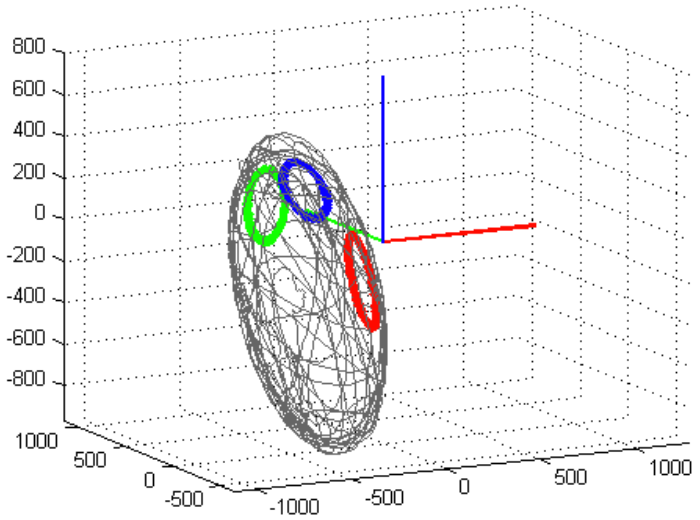
GPS
Absolute location (outdoor)

Pressure
Altitude (outdoor); relative height (indoor)

$$\mathbf{B}_{x,y,z} = \mathbf{W}\,\mathbf{R}_{\phi}\,\mathbf{R}_{\theta}\,\mathbf{R}_{\psi}\,\mathbf{B}_{earth} + \mathbf{V}$$

$\mathbf{R}_{\phi}\,\mathbf{R}_{\theta}\,\mathbf{R}_{\psi}$ rotates the earth's magnetic field by yaw (ψ), pitch (θ) and roll (ϕ) angles to the sensor's local (x,y,z) coordinates

W (3x3 matrix) and **V** (xyz-vector) are "soft iron" and "hard iron" distortions for ferromagnetic components that move with the sensor



+ light, sound, touch, temperature, humidity, heart rate, …

Types of sensors

Accelerometer

Linear acceleration plus gravity

Magnetometer

Local magnetic field

Gyroscope

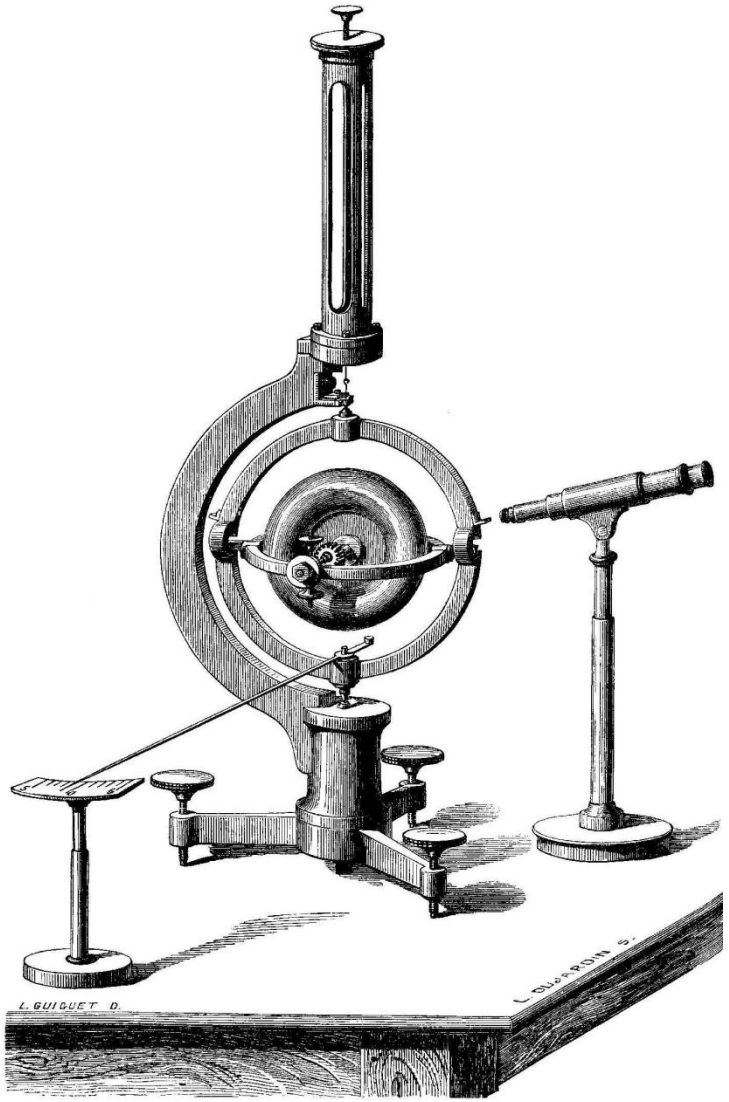
Angular rotation rate

GPS

Absolute location (outdoor)

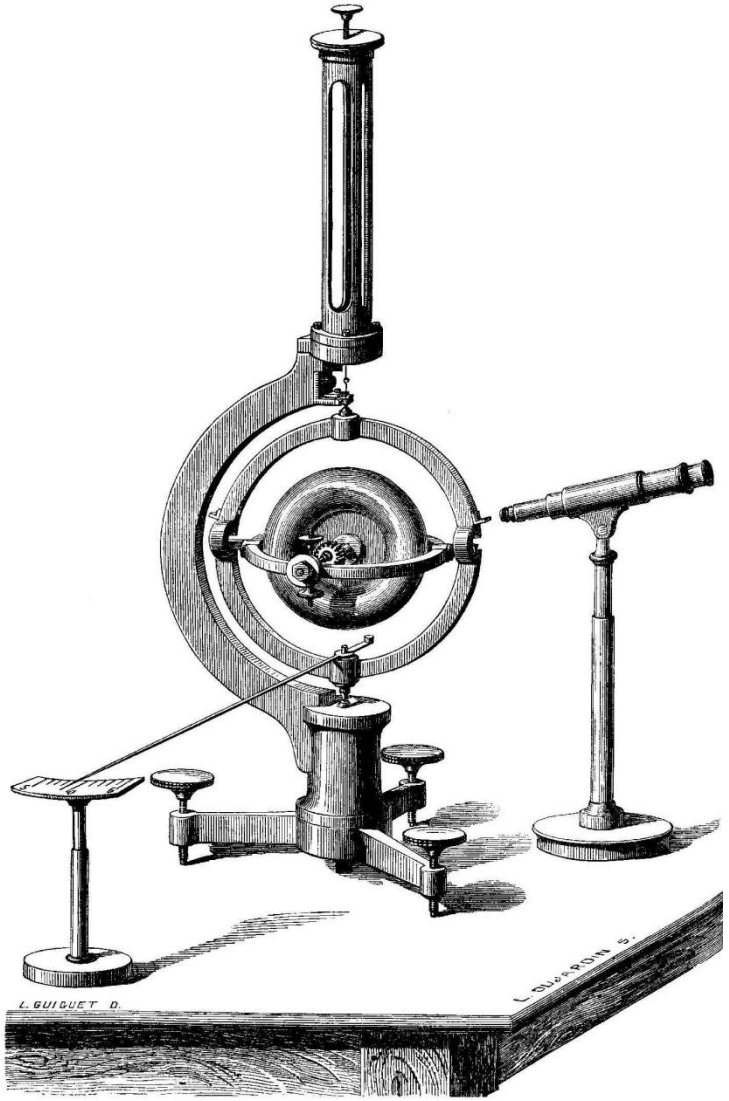
Pressure

Altitude (outdoor); relative height (indoor)

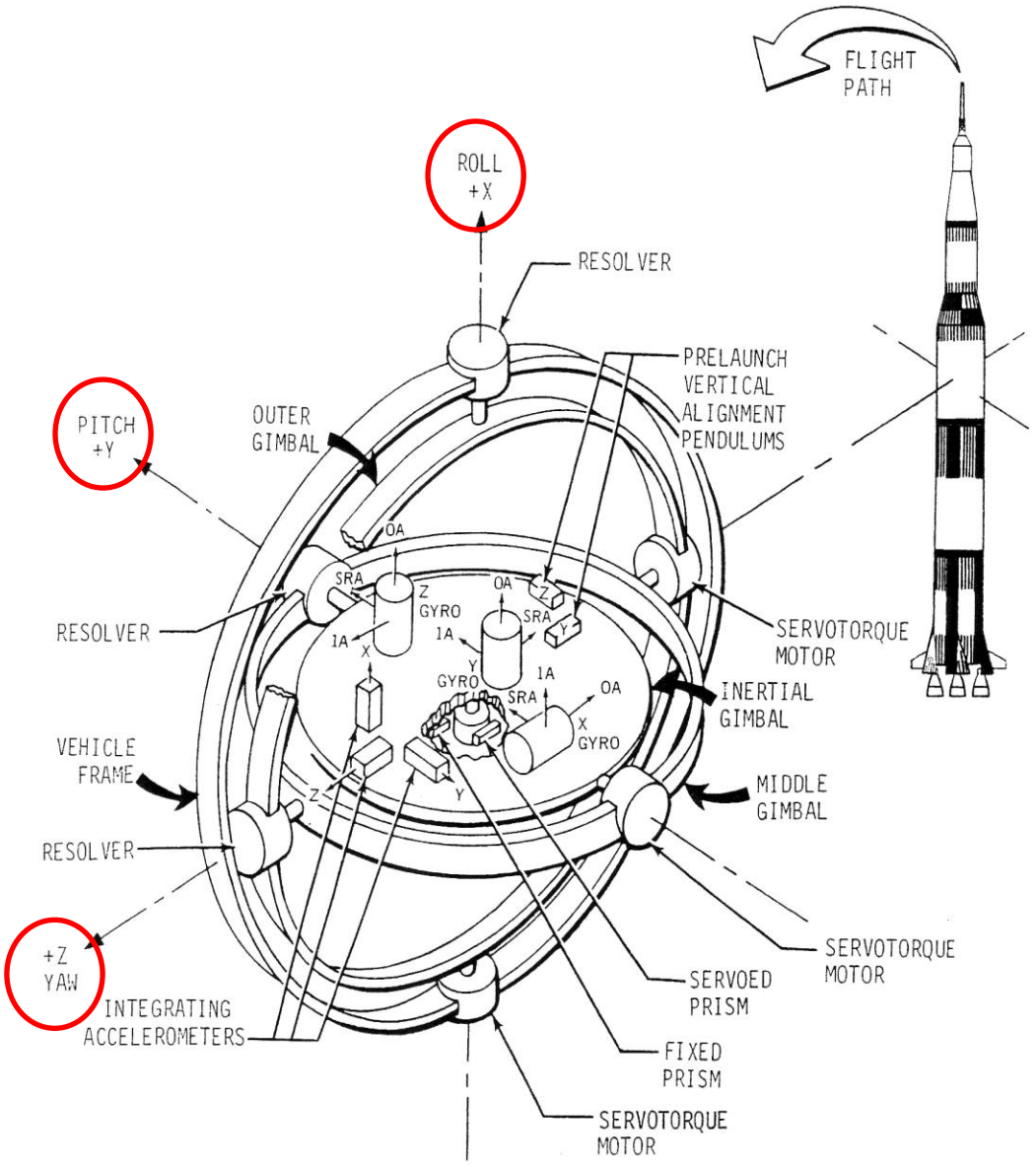


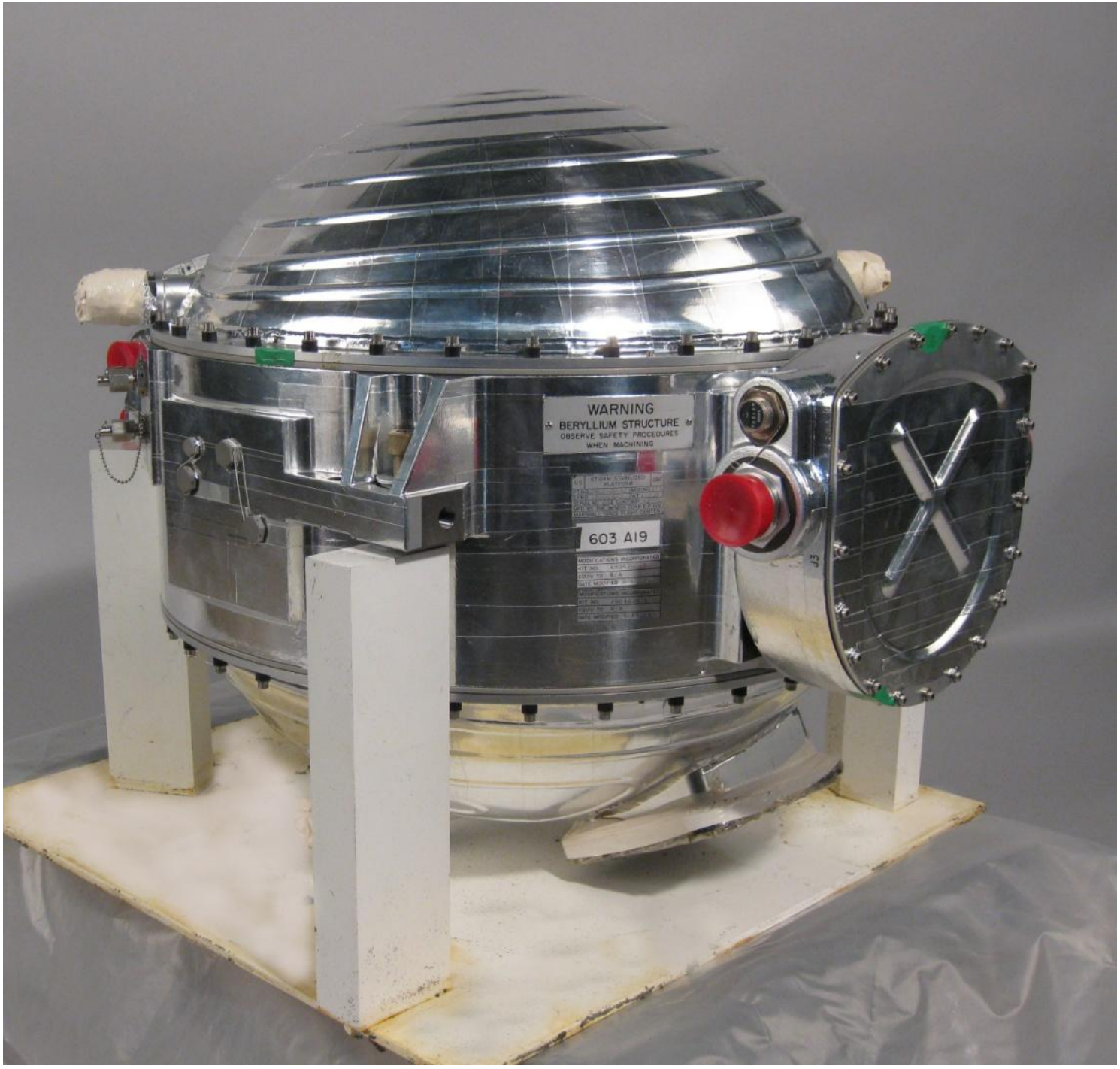
+ light, sound, touch, temperature, humidity, heart rate, …

Foucault’s gyroscope (1862)

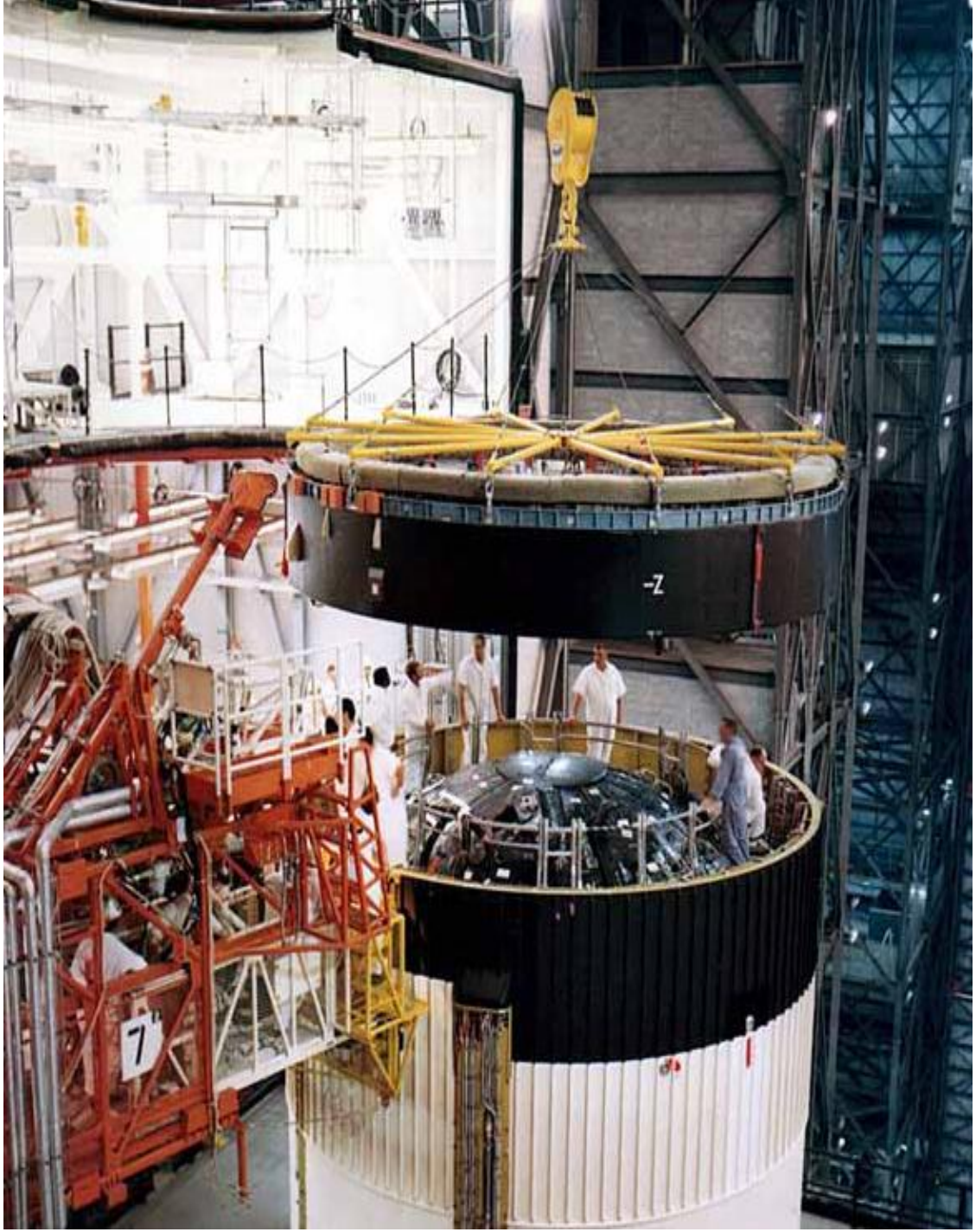


ST124-M inertial platform for Saturn V (1965)

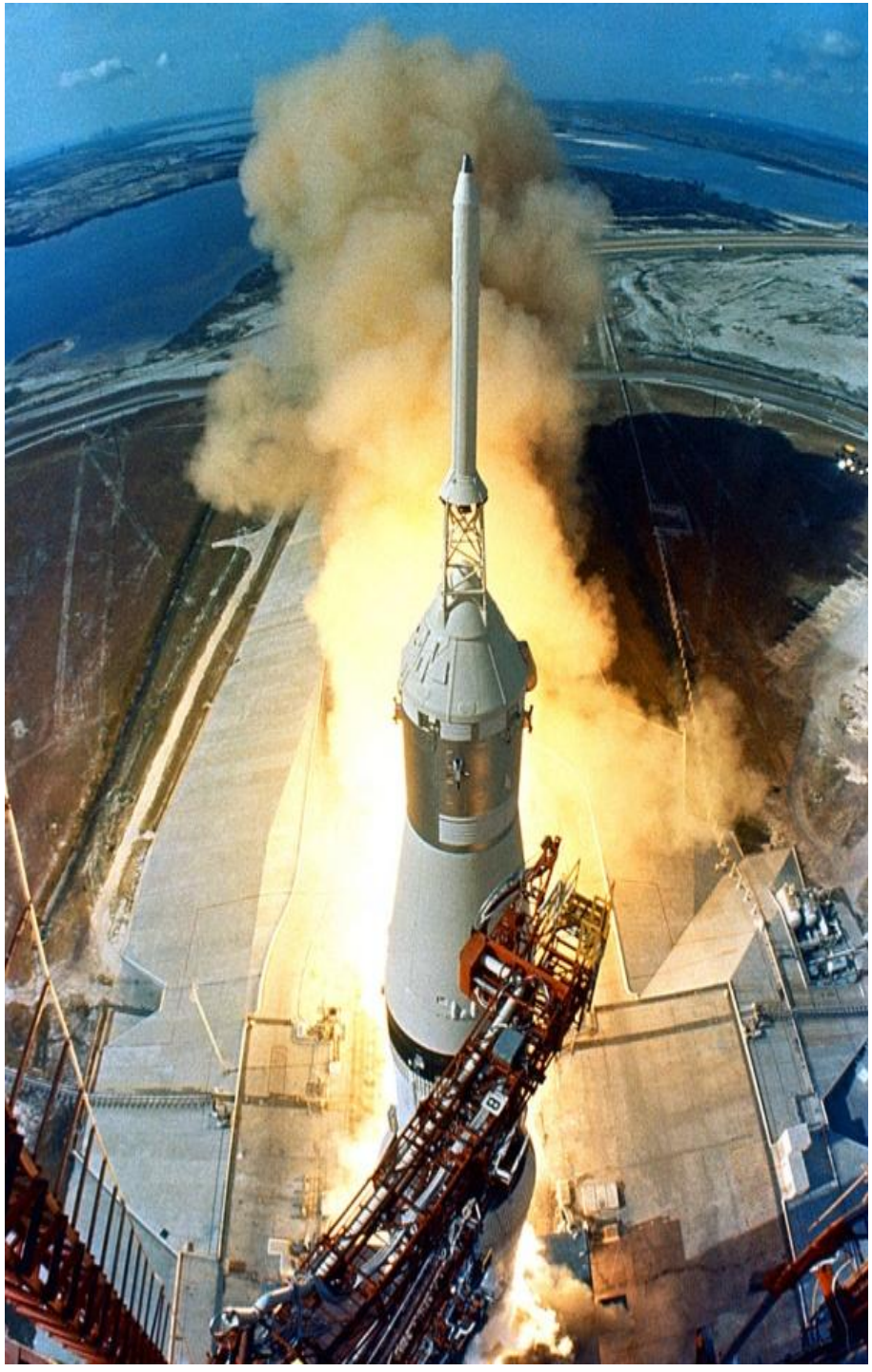



WARNING
BERYLLIUM STRUCTURE
OBSERVE SAFETY PROCEDURES
WHEN MACHINING
603 A19

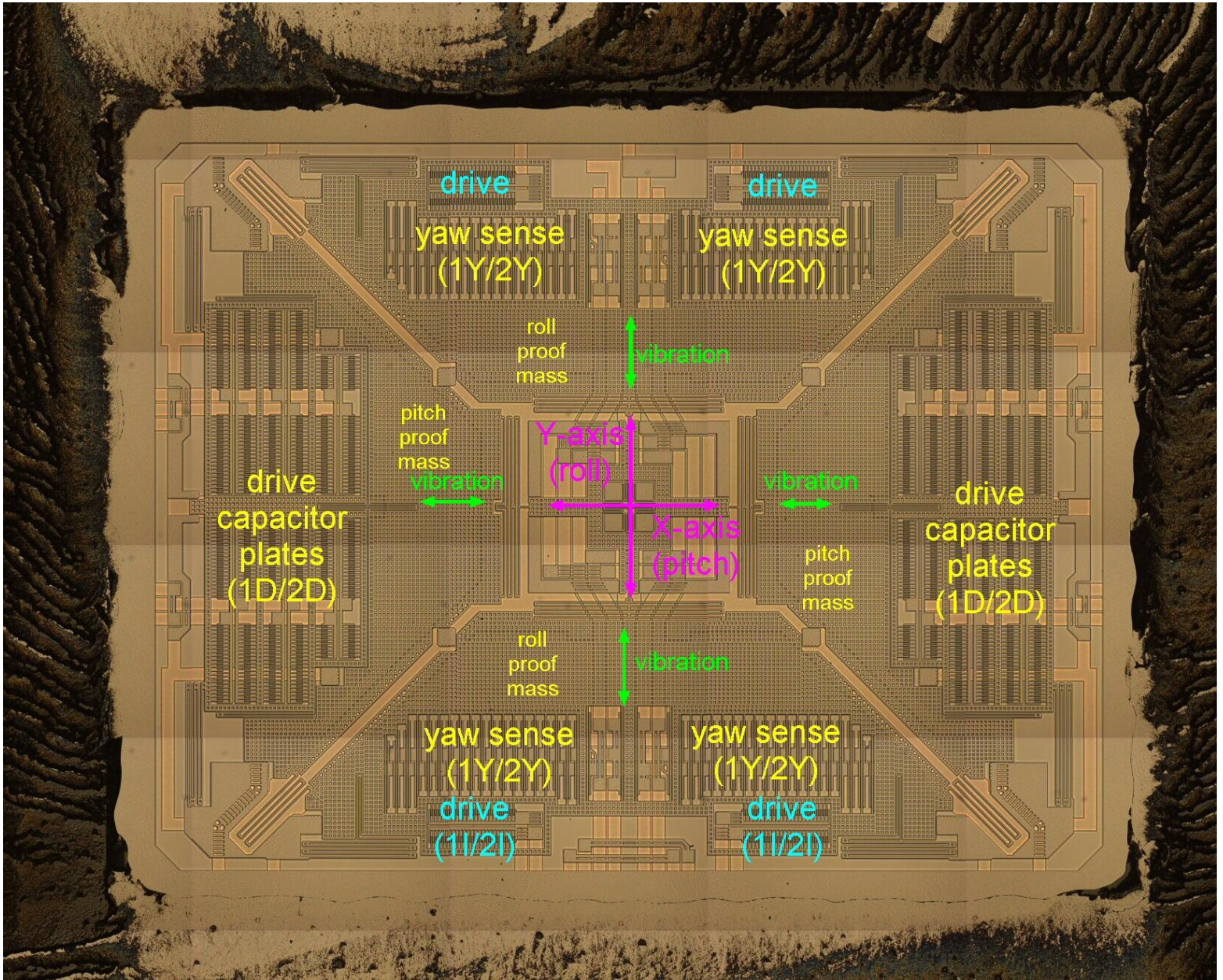

ACCESS
DOOR


-Z
7"

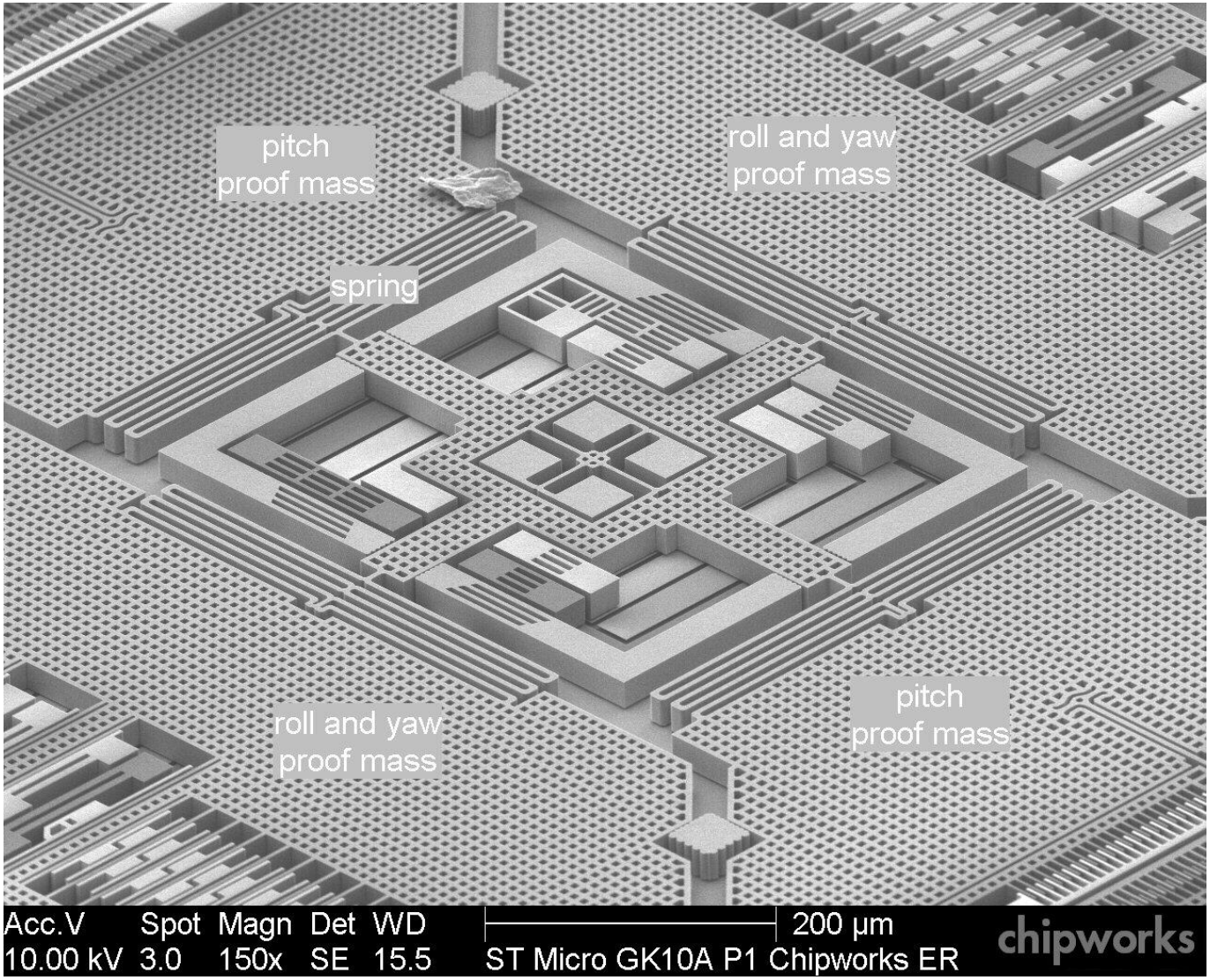

USA



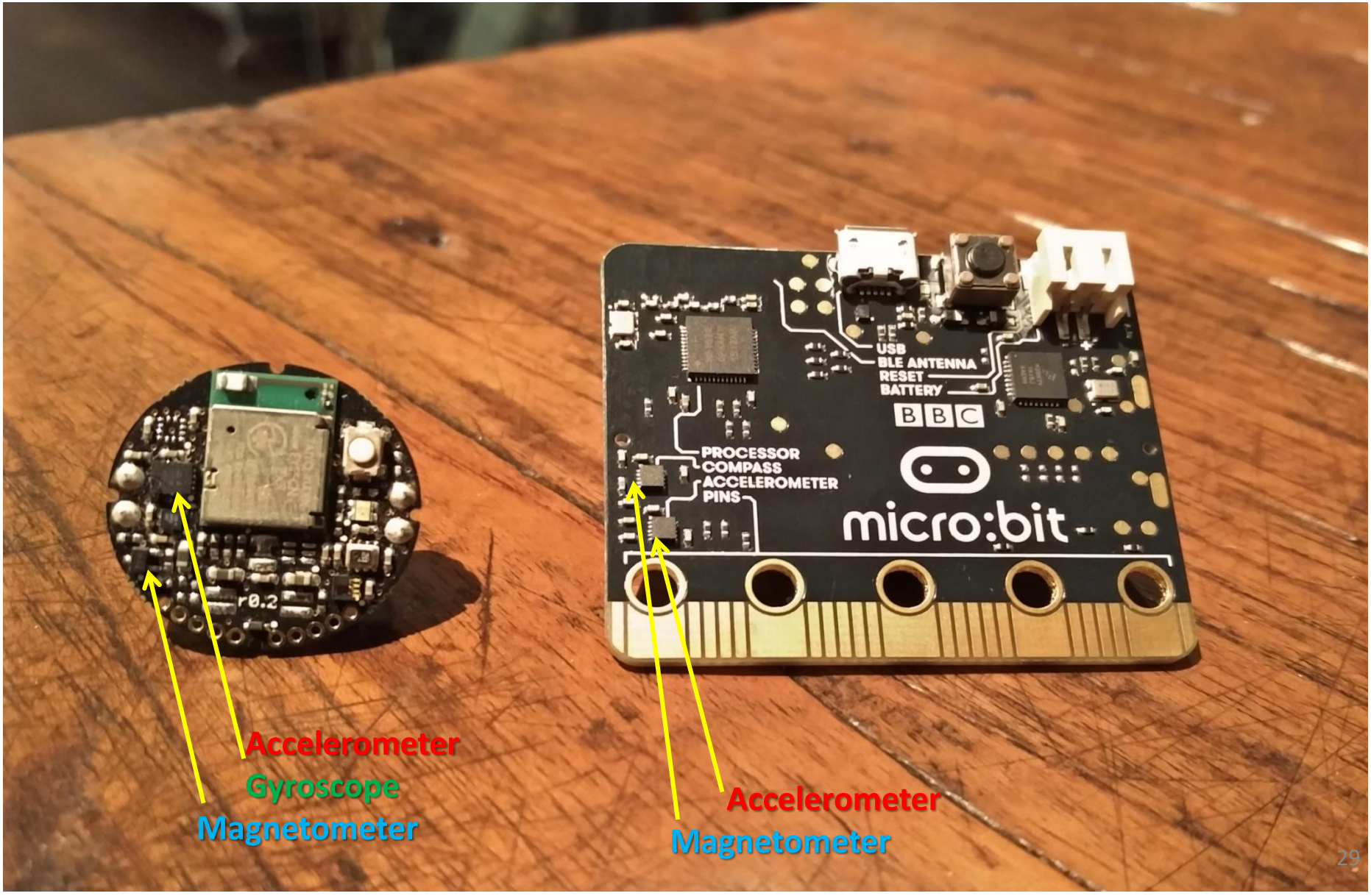
MEMS gyroscope (2016)



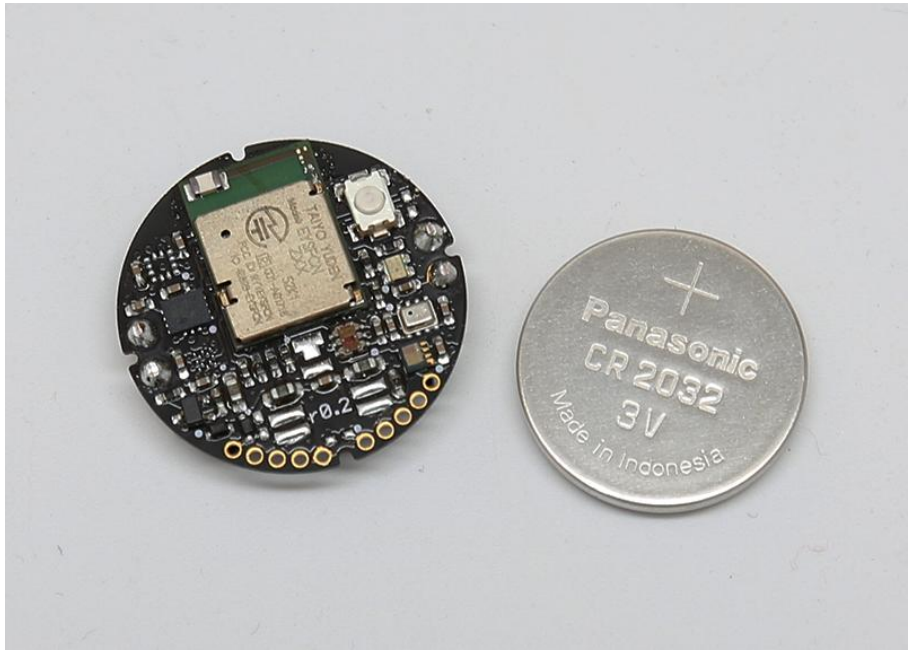
MEMS gyroscope (2016)



Sensors in Metawear CPRO and BBC micro:bit



Metaware CPRO – hardware



PCB: 24 mm diameter
sensors: 2 x 2 mm

CPU
nRF51822 - Nordic Semiconductor
ARM Cortex-M0, 32k RAM, 256k flash

Interfaces
Bluetooth LE (stream up to 100 Hz)
Push button; 1 LED

Sensors
BMI160 (Bosch)
- Accelerometer - 3 axis, 14-bit, up to 800Hz
- Gyroscope – 3 axis, 16 bit

BMI155 Magnetometer (Bosch)
- 3 axis, 16-bit, up to 80 Hz
- ±1300µT (sens 0.3µT)

Pressure: 300-1100 hPa
Temperature: -40 to 85 °C
Light: 0.01 to 64k lux

BBC microbit – hardware

PCB: 43 x 52 mm

sensors: 2 x 2 mm

CPU

nRF51822 - Nordic Semiconductor

ARM Cortex-M0, 16k RAM, 256k flash

Interfaces

Bluetooth LE; radio; USB

Push buttons

Sensors

Accelerometer MMA8652FC (NXP)

3 axis, 12-bit, up to 800Hz

6-180µA @ 2.4V

±2/4/8g (sens ±2.5%, offset ±33mg, noise 182µg/√Hz)

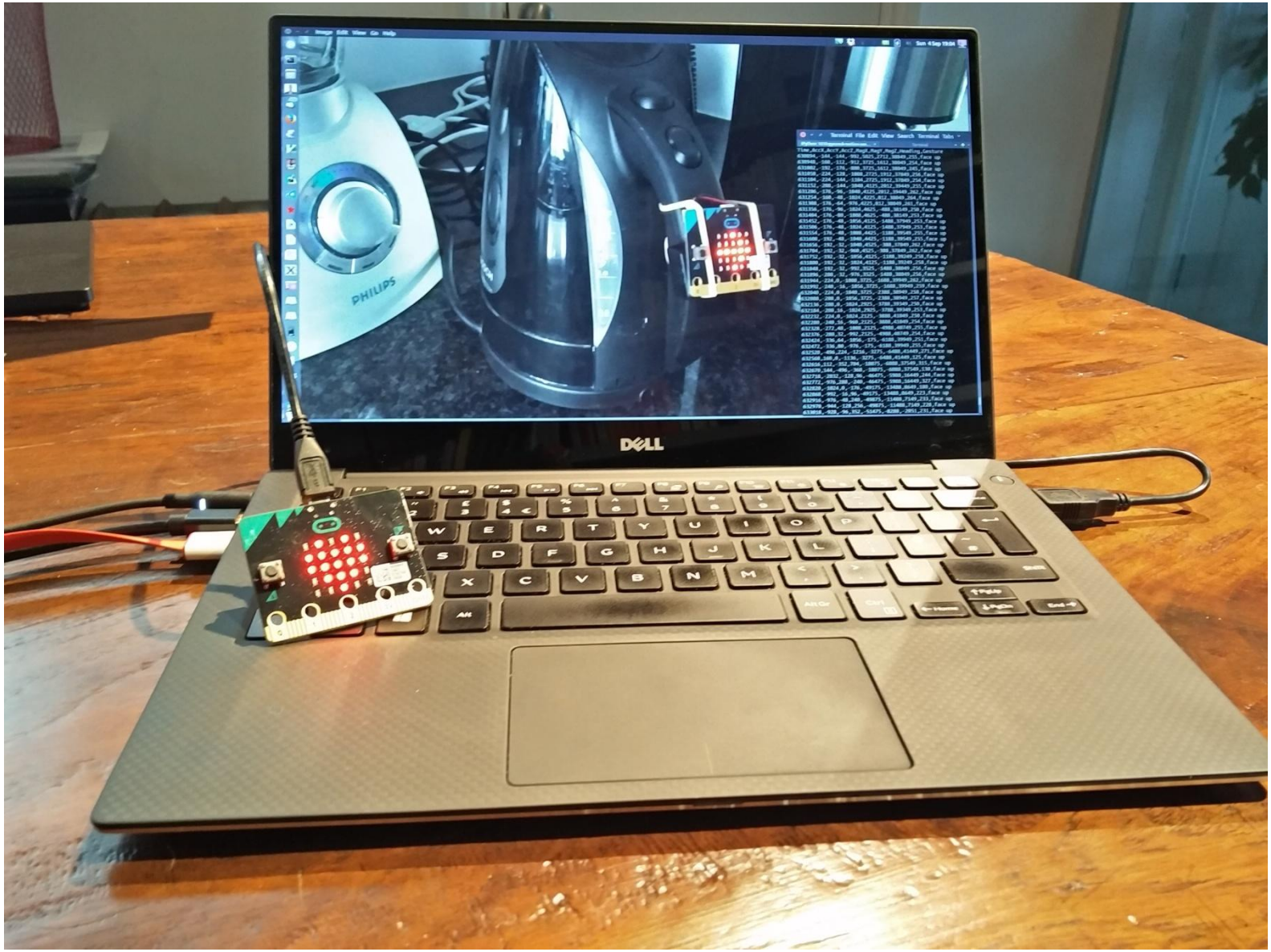
Magnetometer MAG3110 (NXP)

3 axis, 16-bit, up to 80 Hz

9-900µA @ 2.4V

±1000µT (sens 0.1µT, noise 0.25µT)

Example #1 – Wireless motion logging



Example #1 – BBC microbit code

```
# Transmit sensor data via radio
from microbit import *
import radio

header = "Time,AccX,AccY,AccZ,MagX,MagY,MagZ,Heading,Gesture\n“

def read_sensors():
    "Sensor data as a comma-separated string with a newline"
    t = running_time()
    ax, ay, az = accelerometer.get_values()
    g = accelerometer.current_gesture()
    cx, cy, cz = compass.get_x(), compass.get_y(), compass.get_z()
    h = compass.heading()
    msg = '%d,%d,%d,%d,%d,%d,%d,%d,%s\n' % (t,ax,ay,az,cx,cy,cz,h,g)
    return msg
```

Example #1 – BBC microbit code

```
# Transmit sensor data via radio
from microbit import *
import radio

header = "Time,AccX,AccY,AccZ,MagX,MagY,MagZ,Heading,Gesture\n“

def read_sensors():
    "Sensor data as a comma-separated string with a newline"
    t = running_time()
    ax, ay, az = accelerometer.get_values()
    g = accelerometer.current_gesture()
    cx, cy, cz = compass.get_x(), compass.get_y(), compass.get_z()
    h = compass.heading()
    msg = '%d,%d,%d,%d,%d,%d,%d,%d,%s\n' % (t,ax,ay,az,cx,cy,cz,h,g)
    return msg

radio.config(length=100)
radio.on()
radio.send(header)

while True:
    data = read_sensors()
    radio.send(data)
    sleep(25)
```

Example #1 – BBC microbit code

```
# Transmit sensor data via radio
from microbit import *
import radio

header = "Time,AccX,AccY,AccZ,MagX,MagY,MagZ,Heading,Gesture\n“

def read_sensors():
    "Sensor data as a comma-separated string with a newline"
    t = running_time()
    ax, ay, az = accelerometer.get_values()
    g = accelerometer.current_gesture()
    cx, cy, cz = compass.get_x(), compass.get_y(), compass.get_z()
    h = compass.heading()
    msg = '%d,%d,%d,%d,%d,%d,%d,%d,%s\n' % (t,ax,ay,az,cx,cy,cz,h,g)
    return msg

radio.config(length=100)
radio.on()
radio.send(header)

while True:
    data = read_sensors()
    radio.send(data)
    sleep(25)
```

```
# Relay radio data to host PC via USB
from microbit import *
import radio
radio.config(length=100)
radio.on()
uart.init(115000)

while True:
    data = radio.receive()
    if data:
        uart.write(data)
```

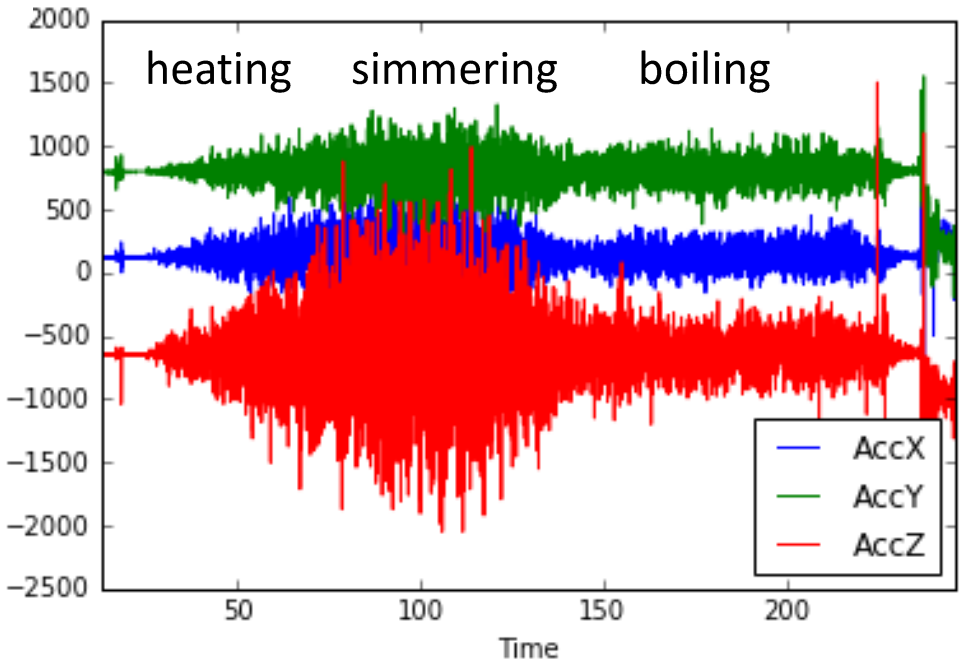
Example #1 – Analyse data in IPython

```
# See Jupyter notebook analysis-kettle.ipynb

%matplotlib inline
import matplotlib
import pandas as pd

>>> df = pd.read_csv('data.csv')
>>> df['Time'] /= 1000.0 # ms to sec
>>> df.set_index('Time', inplace=True)
>>> df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Float64Index: 5860 entries, 12.9 to 246.1
Data columns (total 8 columns):
AccX       5860 non-null int64
AccY       5860 non-null int64
AccZ       5860 non-null int64
MagX       5860 non-null int64
MagY       5860 non-null int64
MagZ       5860 non-null int64
Heading    5860 non-null int64
Gesture    3760 non-null object
dtypes: int64(7), object(1)
memory usage: 412.0+ KB
```

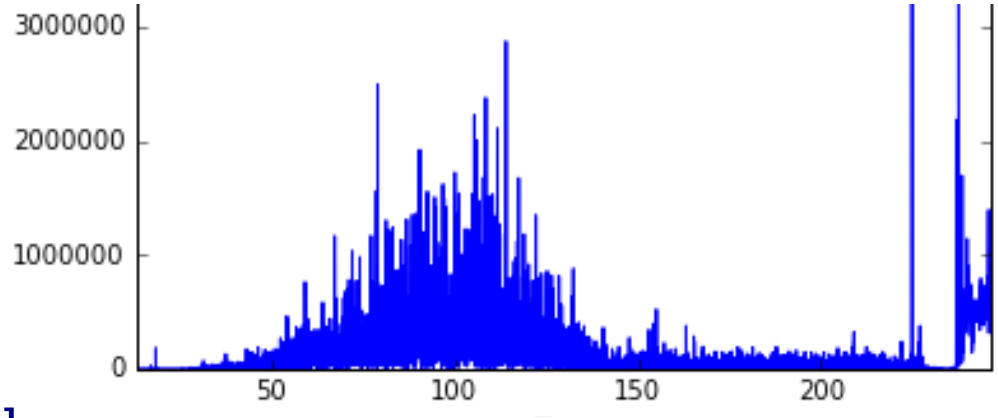
Example #1 – Analyse data in IPython

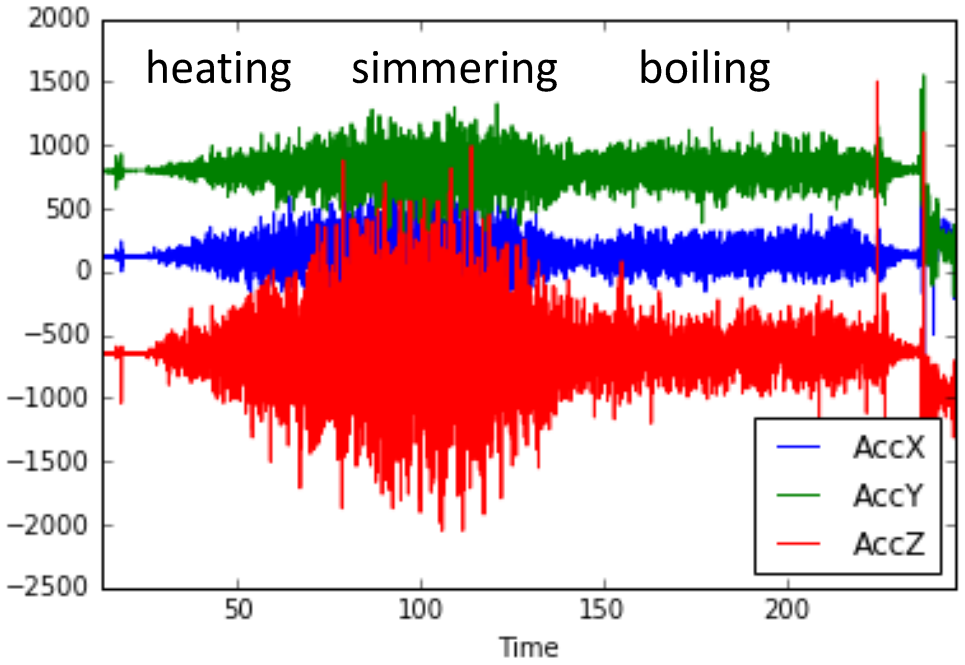
```
# See Jupyter notebook analysis-kettle.ipynb

%matplotlib inline
import matplotlib
import pandas as pd

>>> df = pd.read_csv('data.csv')
>>> df['Time'] /= 1000.0 # ms to sec
>>> df.set_index('Time', inplace=True)
>>> df.info()

>>> accXYZ = df[['AccX','AccY','AccZ']]
>>> accXYZ.plot()
```



Example #1 – Analyse data in IPython

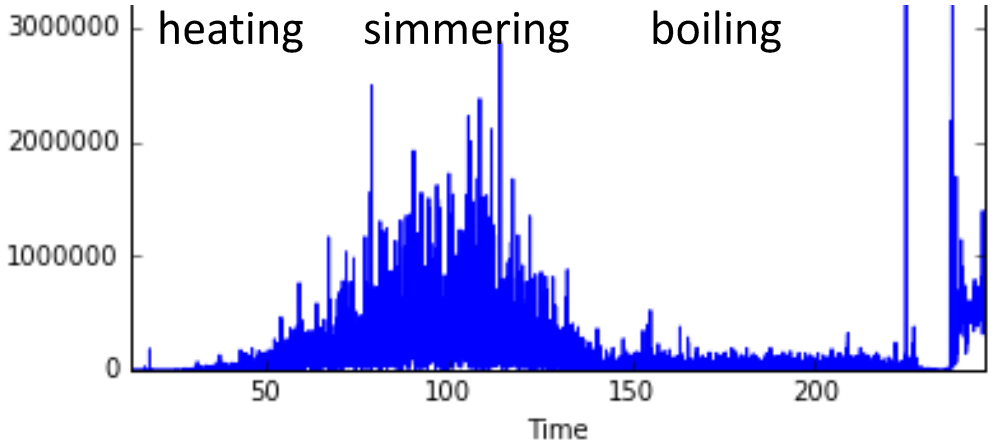
```
# See Jupyter notebook analysis-kettle.ipynb

%matplotlib inline
import matplotlib
import pandas as pd

>>> df = pd.read_csv('data.csv')
>>> df['Time'] /= 1000.0 # ms to sec
>>> df.set_index('Time', inplace=True)
>>> df.info()

>>> accXYZ = df[['AccX','AccY','AccZ']]
>>> accXYZ.plot()

>>> g = accXYZ[50.0:200.0].mean()
>>> g
AccX    128.733753
AccY    802.477987
AccZ   -636.926625
dtype: float64

>>> s = (accXYZ - g).pow(2).sum(axis=1)
>>> s.plot()
```

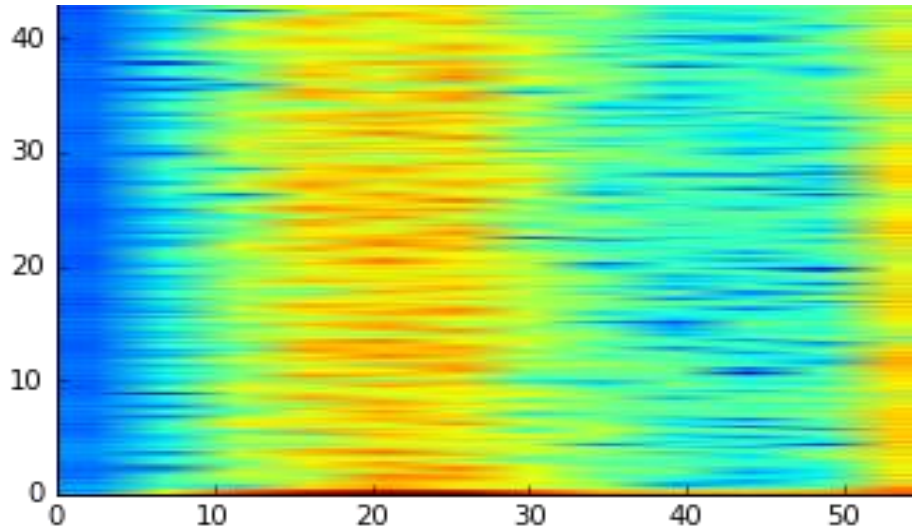




Example #1 – Analyse data in IPython

```
# See Jupyter notebook analysis-kettle.ipynb

%matplotlib inline
import matplotlib
import pandas as pd

>>> df = pd.read_csv('data.csv')
>>> df['Time'] /= 1000.0 # ms to sec
>>> df.set_index('Time', inplace=True)
>>> df.info()

>>> accXYZ = df[['AccX','AccY','AccZ']]
>>> accXYZ.plot()

>>> g = accXYZ[50.0:200.0].mean()
>>> g
AccX    128.733753
AccY    802.477987
AccZ   -636.926625
dtype: float64

>>> s = (accXYZ - g).pow(2).sum(axis=1)
>>> s.plot()

>>> import matplotlib.pyplot as plt
>>> plt.specgram(s, NFFT=512, Fs=100, noverlap=32)[3]
```





Project ideas

Orientation

- Read accelerometer data
- Figure out which simple gesture fits best ("falling", "left", "up"…)
- Display first letter of the active gesture on the screen

Sports stats

- Log accelerometer data while repeated a sport-related action (running, hitting tennis ball, …)
- Plot the time series
- Identify characteristic motion
- Find decision rules to pinpoint time when motion occurred
- Use decision rule to total sports stats (# actions, speed etc)

Data fusion algorithms

(1) Combine data from multiple sensors

(2) Reduce position errors when integrating accel/vel

(3) Correct for sensor bias, drift, noise, nonlinearities

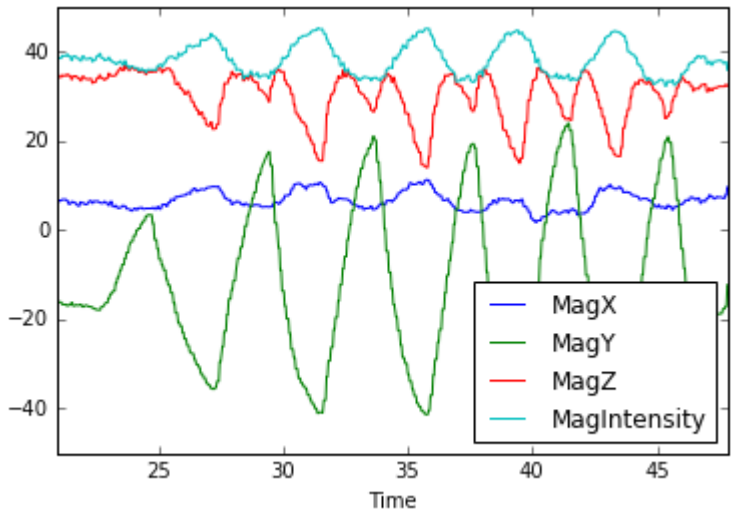
(4) Deal with missing data points

Accelerometer **Magnetometer**	**Tilt-compensated electronic compass** (if acceleration = 0)
Accelerometer **Gyroscope**	**Inertial Measurement Unit (IMU)** (6 DoF – orientation, no absolute yaw)
Accelerometer **Magnetometer** **Gyroscope**	**Attitude Heading Reference System (AHRS)** **Magnetic, Angular Rate, Gravity (MARG)** (9 DoF – orientation, absolute north)

Example #2 – Tilt-compensating compass

```
# See analysis-tilt-compensation.ipynb
import numpy as np
import pandas as pd
import math
rad2deg = 180.0 / math.pi

# Data is collected via microbit data logger as the
# microbit is tilted up and down while pointed in
# approximately a constant heading.
df = pd.read_csv('data-compass-heading-with-tilt.csv')
df['Time'] /= 1000.0    # ms (int) to sec (float)
df.set_index('Time', inplace=True)

def magXYZ_plus_intensity(df):
    '''Add column 'MagIntensity' to DataFrame df'''
    m = df[['MagX','MagY','MagZ']].copy()
    m['MagIntensity'] = m.pow(2).sum(axis=1).pow(0.5)
    return m

magXYZ_plus_intensity(df).plot()
```

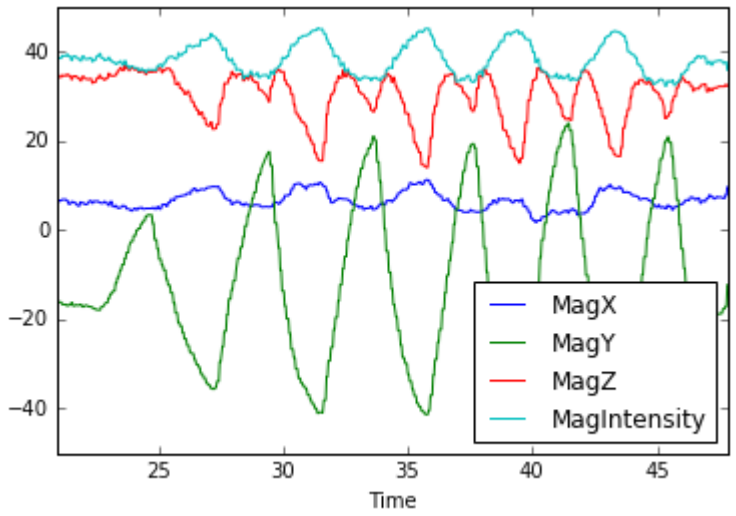
Raw magnetometer data



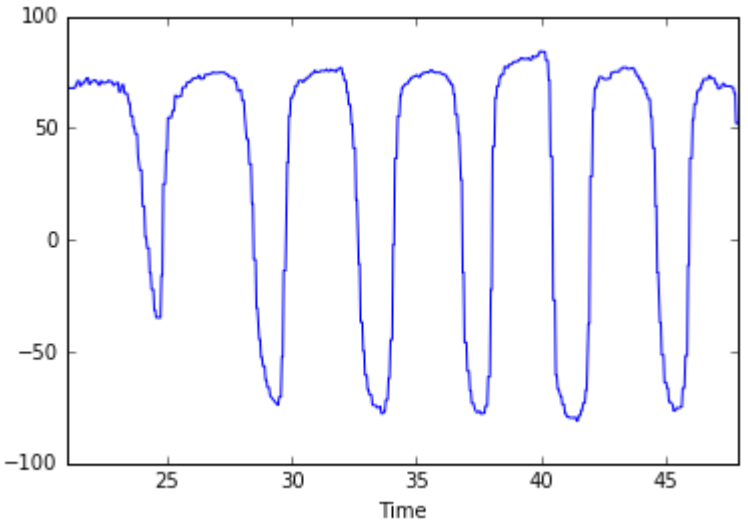
Example #2 – Tilt-compensating compass

```
# See analysis-tilt-compensation.ipynb
import numpy as np
import pandas as pd
import math
rad2deg = 180.0 / math.pi

# Data is collected via microbit data logger as the
# microbit is tilted up and down while pointed in
# approximately a constant heading.
df = pd.read_csv('data-compass-heading-with-tilt.csv')
df['Time'] /= 1000.0    # ms (int) to sec (float)
df.set_index('Time', inplace=True)

def magXYZ_plus_intensity(df):
    '''Add column 'MagIntensity' to DataFrame df'''
    m = df[['MagX','MagY','MagZ']].copy()
    m['MagIntensity'] = m.pow(2).sum(axis=1).pow(0.5)
    return m

magXYZ_plus_intensity(df).plot()
```

Raw magnetometer data



Calculated heading (degrees)

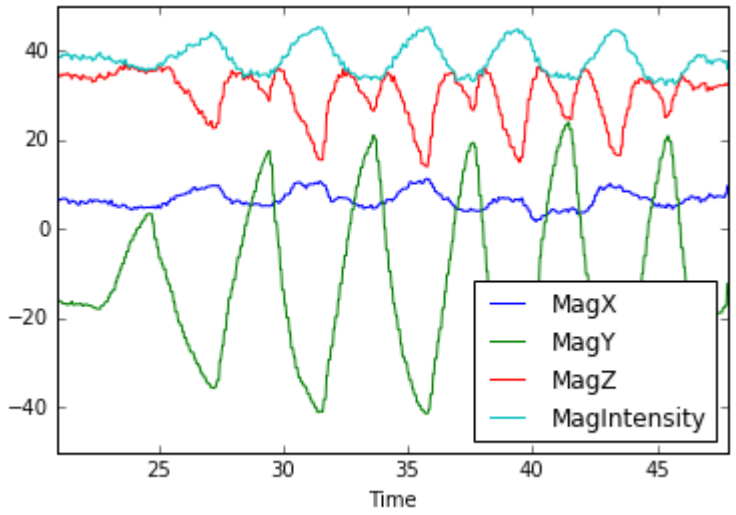



```
# Simple compass heading (yaw angle = psi)
psi = rad2deg * np.arctan2(-df.MagY, df.MagX)
s = pd.Series(psi, index=df.index)
s.plot()
```

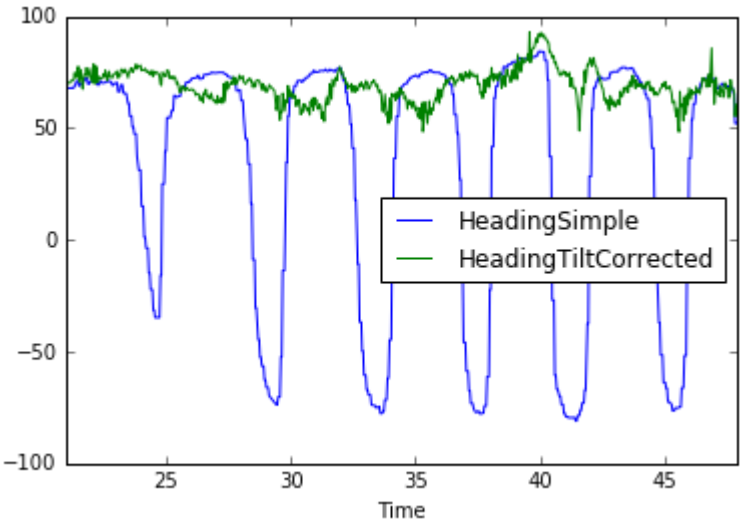
Example #2 – Tilt-compensating compass

```
def compass_heading_tilt_comp(df):
    """Pandas Series with heading corrected for tilt.

    This assumes the accelerometer is held still.
    From the angle of gravity, infer the magnetometer's
    tilt relative to the Earth's surface.
    Undo the effects of tilt in the roll and pitch
    directions, leaving a clean magnetic heading in
    the yaw (Z) direction.

    The input dataframe is assumed to have columns named
    MagX, MagY, MagZ, AccX, AccY, AccZ.
    """
    ax, ay, az = df.AccX, df.AccY, df.AccZ
    mx, my, mz = df.MagX, df.MagY, df.MagZ

    # Undo roll around X axis by angle phi
    phi = np.arctan2(ay, az)  # +/- 180 degrees
    s, c = np.sin(phi), np.cos(phi)
    az = ay*s + az*c
    my, mz = (my*c - mz*s), (my*s + mz*c)

    # Undo pitch around Y axis by angle theta
    theta = np.arctan(-ax / az) # +/-90 degrees
    s, c = np.sin(theta), np.cos(theta)
    mx, mz = (mx*c + mz*s), (-mx*s + mz*c)

    # Calculate yaw angle psi around Z axis
    psi = np.arctan2(my, mx)
    s = pd.Series(psi*rad2deg, index=df.index,
                  name='HeadingTiltCorrected')
    return s
```

Raw magnetometer data



Calculated heading (degrees)

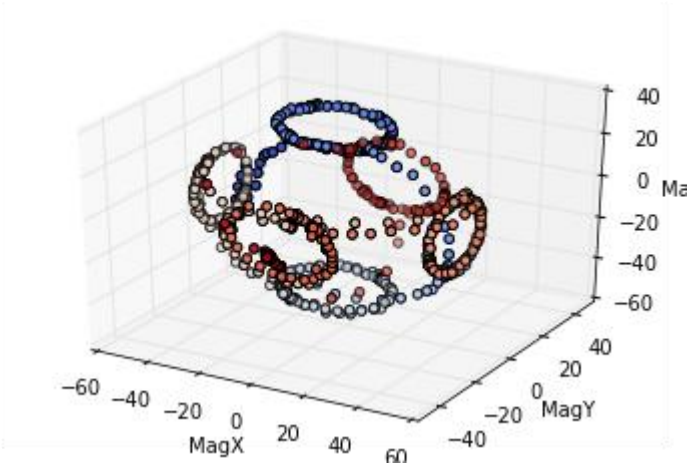


Example #2 – Simple magnetometer calibration

$$\mathbf{B}_{x,y,z} = \mathbf{W}\,\mathbf{R}_{\phi}\,\mathbf{R}_{\theta}\,\mathbf{R}_{\psi}\,\mathbf{B}_{earth} + \mathbf{V}$$

where **V** = “hard iron”; **W** = “soft iron”

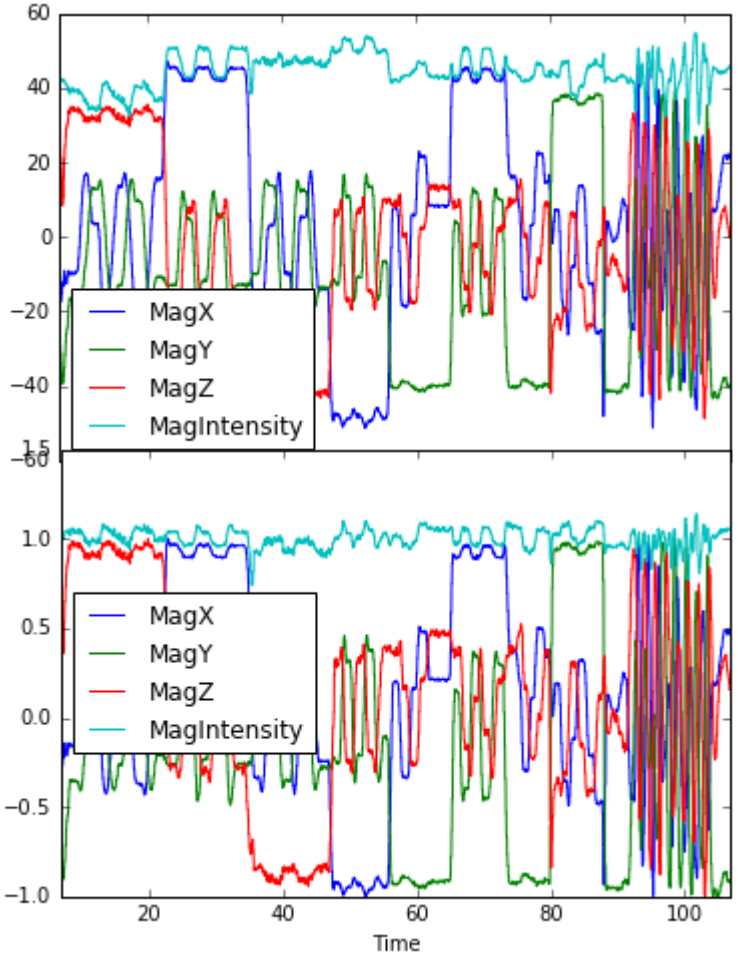
Ideal readings lie on a perfect sphere



Proper calibration fits **W** to an ellipsoid.
To simplify the example today, don’t worry about rotation, just offset **V** and scale **S**:

```
V = (magXYZ.min() + magXYZ.max()) / 2
S = (magXYZ.max() - magXYZ.min()) / 2
magXYZ_cal = (magXYZ - V) / S
```

Pre- and post- calibration data



Example #3 – Mobile phone sensor data

List sensors, manufacturers, specs, etc

- **My Sensors**

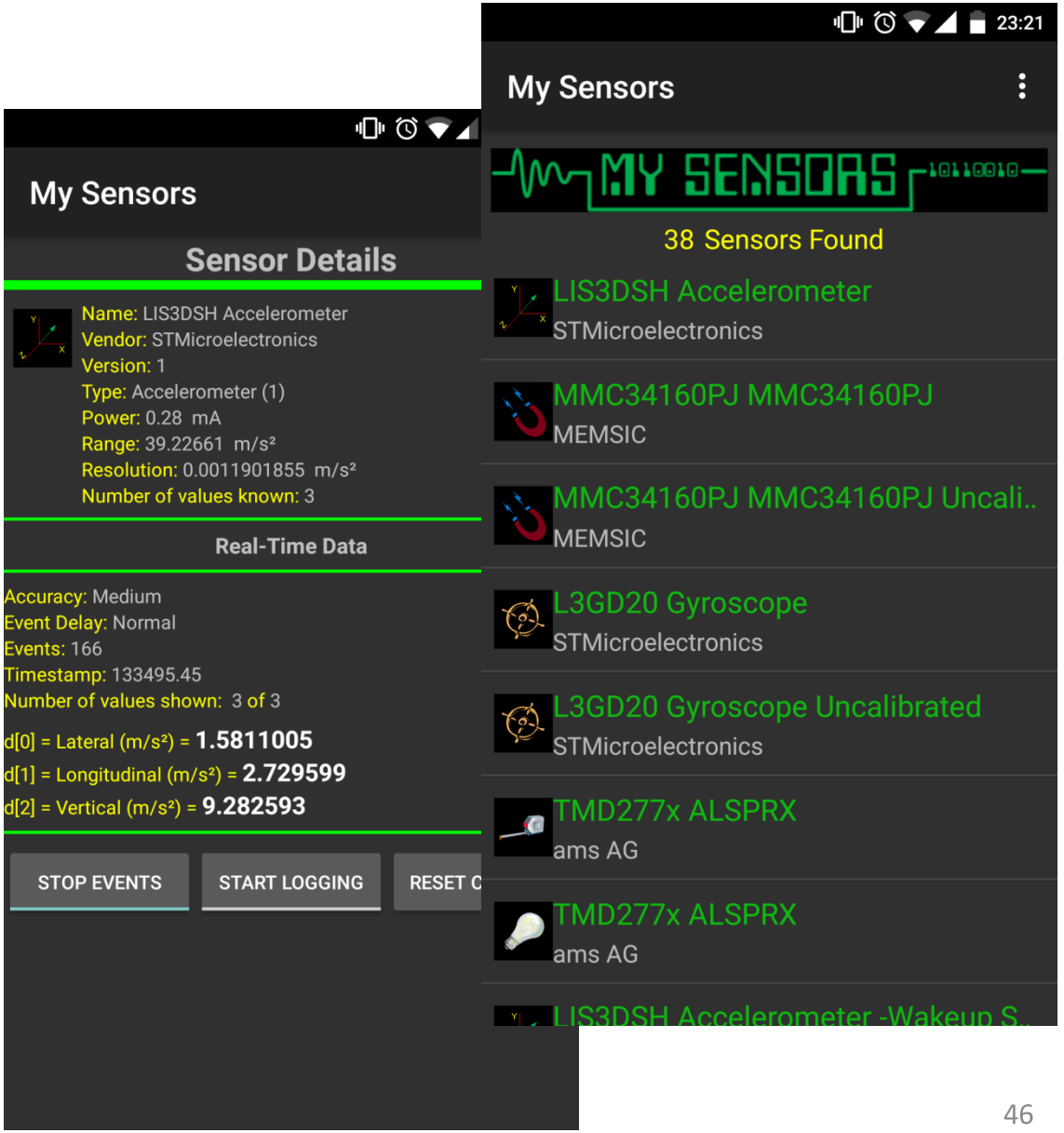
Stream sensor data over wifi

- IMU+GPS-Stream

Log sensor data to csv on phone

- Physics Toolbox Suite

Or write your own app for processing sensor data locally



Example #3 – Mobile phone sensor data

List sensors, manufacturers, etc

- My Sensors

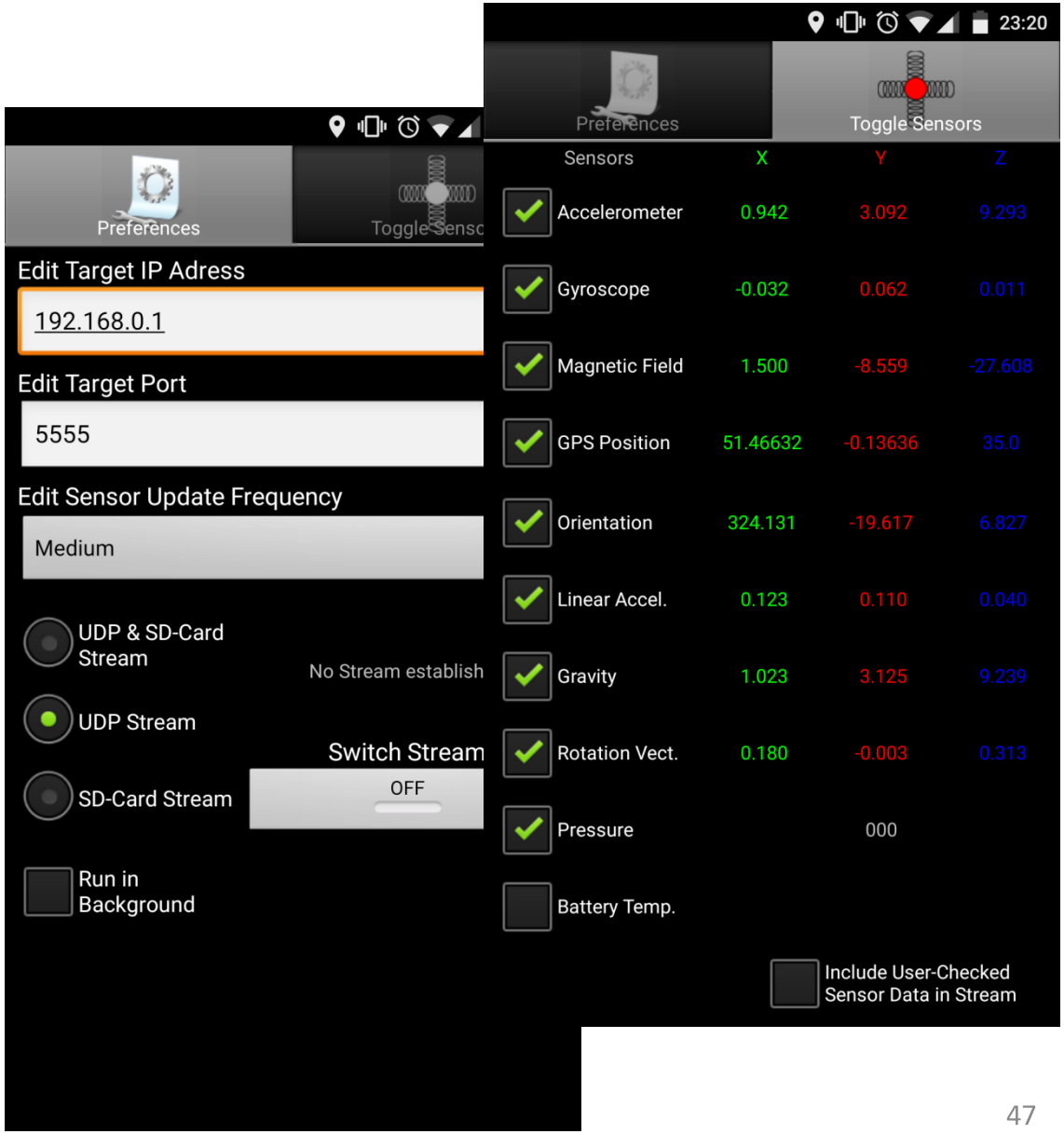
Stream sensor data over wifi

- IMU+GPS-Stream

Log sensor data to csv on phone

- Physics Toolbox Suite

Or write your own app for processing sensor data locally



Example #3 – Mobile phone sensor data

List sensors, manufacturers, etc

- My Sensors

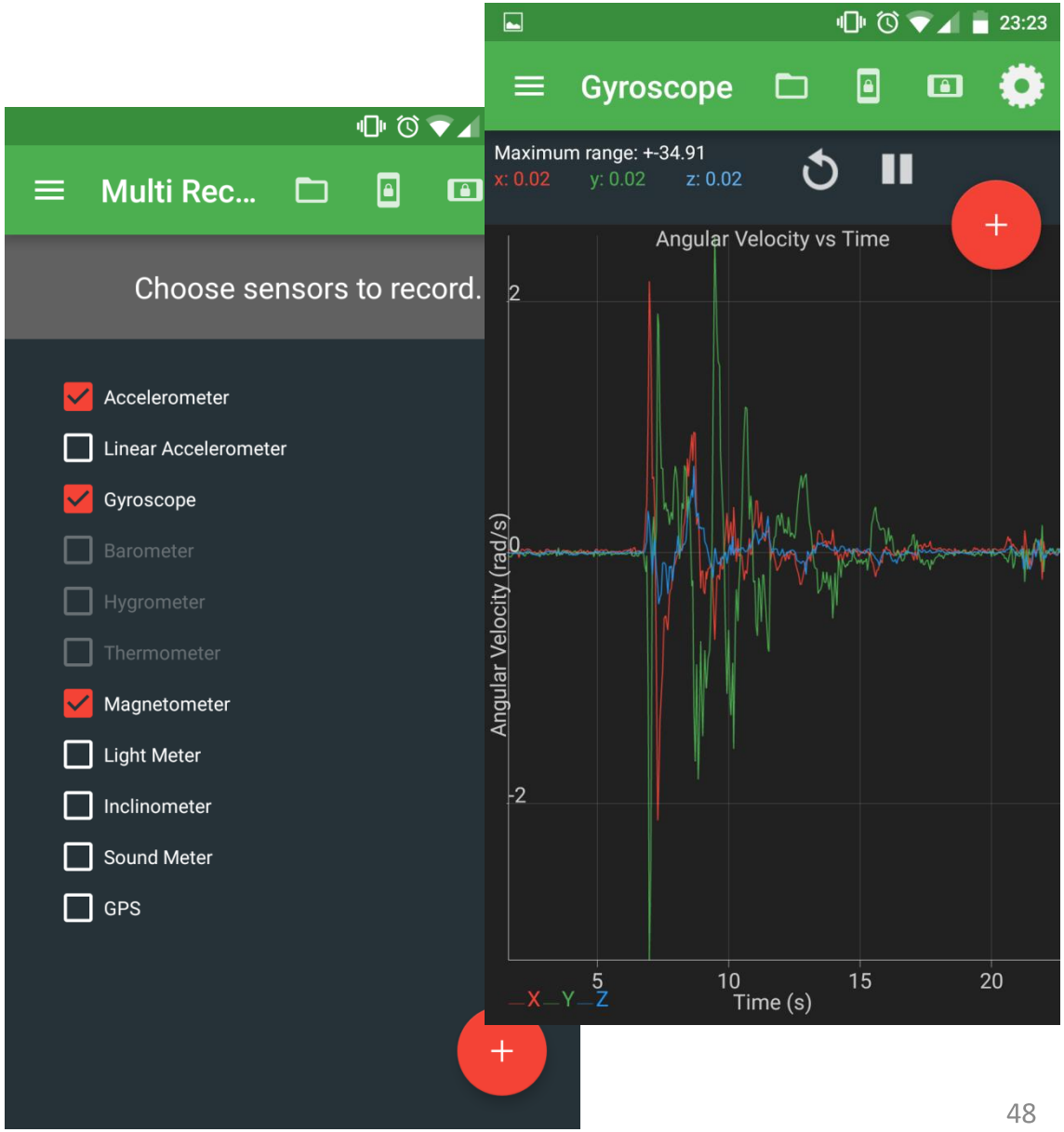
Stream sensor data over wifi

- IMU+GPS-Stream

Log sensor data to csv on phone

- **Physics Toolbox Suite**

Or write your own app for processing sensor data locally



Example #3 – Kalman filter for 9-DoF IMU

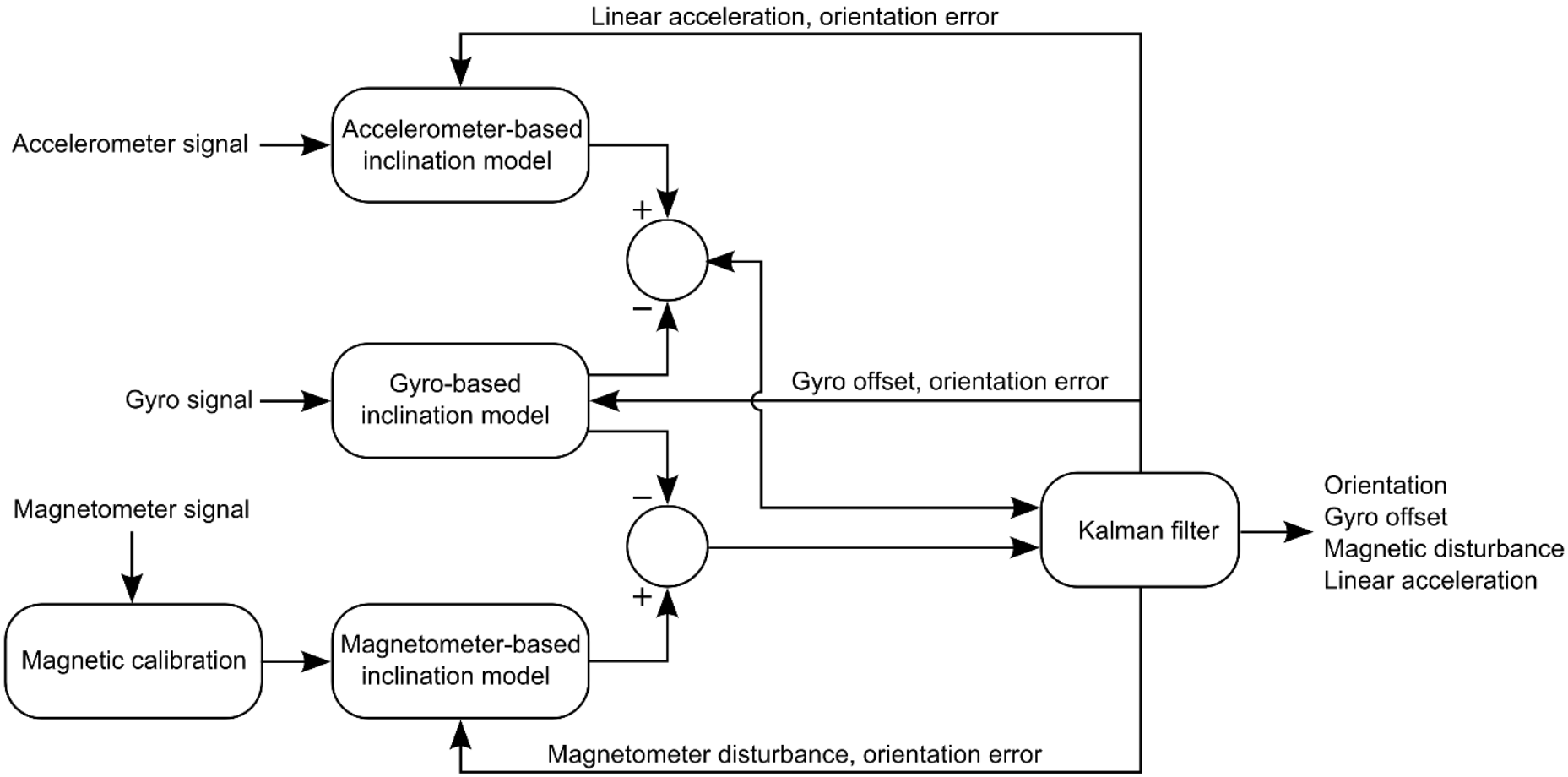
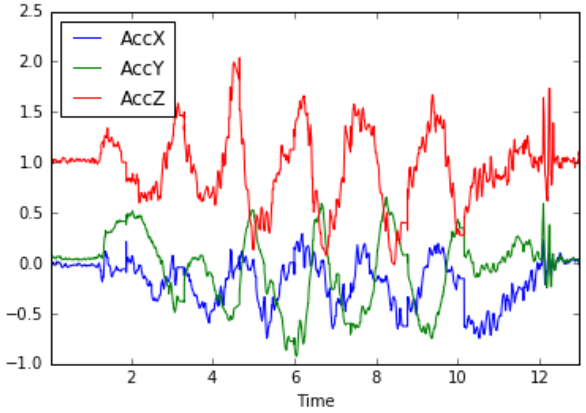
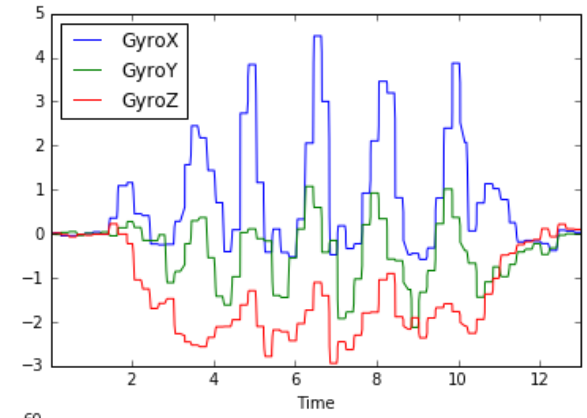
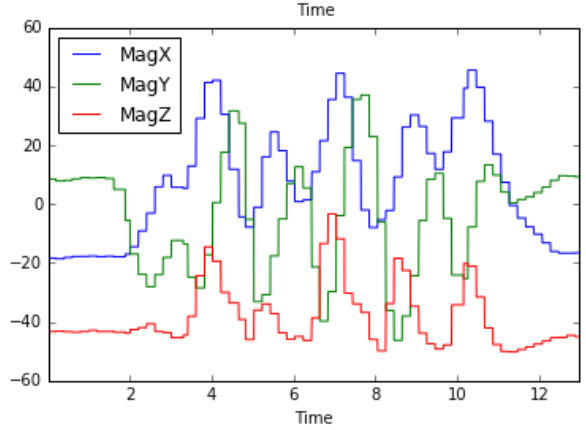


Diagram from NXP Sensor Fusion Library for Kinetis MCUs, release 7.00, August 2016

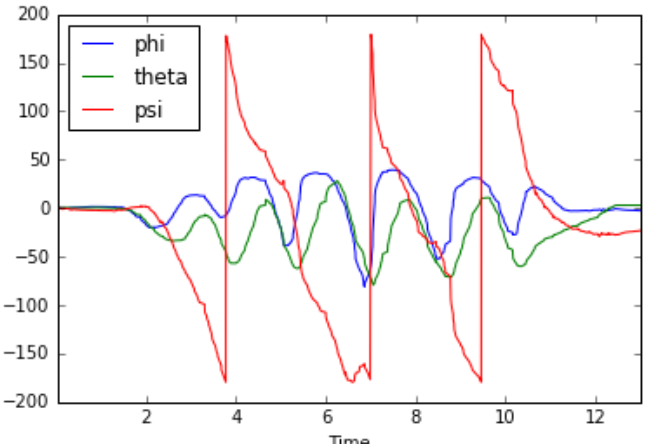
Example #3 – Kalman filter for 9-DoF IMU








```
# Load 9-DoF sensor data as pandas DataFrame
df = pd.read_csv("data-IMU.csv")
names = 'Time,AccX,AccY,AccZ,GyroX,GyroY,GyroZ,MagX,MagY,MagZ'.split(',')
df.rename(columns=dict(zip(df.columns, new_names)), inplace=True)
df.set_index('Time', drop=False, inplace=True)

# Get data as numpy arrays that scikit-kinematics expects
aXYZ = df[['AccX','AccY','AccZ']].values
gXYZ = df[['GyroX','GyroY','GyroZ']].values
mXYZ = df[['MagX','MagY','MagZ']].values

# IMU using Kalman filter from scikit-kinematics
# http://work.thaslwanter.at/skinematics/html/index.html
rate = len(df) / (df.Time.max() - df.Time.min())

import skinematics as skin
quaternion_arr = skin.imus.kalman_quat(rate, aXYZ*9.81, gXYZ, mXYZ)
quaternion_obj = skin.imus.quat.Quaternion(quaternion_arr)
angles_arr = quaternion_obj.export('Fick') * 180/math.pi

angles_df = pd.DataFrame(angles_arr,
                         columns=['phi', 'theta','psi'], index=df.index)
angles_df.plot()
```



Project ideas

Orientation

- Read accelerometer data
- Figure out which simple gesture fits best ("falling", "left", "up"…)
- Display first letter of the active gesture on the screen

Sports stats

- Log accelerometer data while repeated a sport-related action (running, hitting tennis ball, …)
- Plot the time series
- Identify characteristic motion
- Find decision rules to pinpoint time when motion occurred
- Use decision rule to total sports stats (# actions, speed etc)

9 DoF IMU (Inertial Measurement Unit)

- Find a Python IMU project on GitHub
- Capture accel/mag/gyro data from mobile phone
- Reformat the data and feed into the IMU
- See how accurately this can track orientation/position using a Kalman filter

Goals for today…

Why play with motion sensors?

tiny sensors + fun applications + cheap
+ smart algorithms **in python**

Motion sensors then and now – 1852, 1965, 2016

- Example #1 – Wireless motion logging with BBC micro:bit

Data fusion – combining data from multiple sensors

- Example #2 – tilt-compensated digital compass
- Example #3 – 9 DoF IMU from mobile phone data

References

Github report accompanying this talk

http://github.com/stevesimmons/pyconuk-motion-sensor-data-fusion

BBC micro:bit (has accelerometer and magnetomer, no gyroscope)

MicroPython	http://microbit-micropython.readthedocs.io	
Mu editor / uFlash	https://github.com/ntoll/mu	http://uflash.readthedocs.org
MMA8652 accelerometer	http://www.nxp.com/files/sensors/doc/data_sheet/MMA8652FC.pdf	
MAG3110 magnetometer	http://www.nxp.com/files/sensors/doc/data_sheet/MAG3110.pdf	
	http://www.nxp.com/files/sensors/doc/data_sheet/AN4083.pdf	

Data fusion concepts

NXP sensor fusion toolbox (version 7.00, 18 August 2016)

http://www.nxp.com/products/sensors/nxp-sensor-fusion:XTRSICSNSTLBOX?tid=vansensorfusion

Datasheet	http://cache.nxp.com/files/sensors/doc/data_sheet/XSFLK_DS.pdf
User guide	http://cache.nxp.com/files/sensors/doc/user_guide/NSFK_Prod_UG.pdf

Electronic compass projects (accelerometer and magnetometer)

Circuit Cellar (2012)	http://cache.nxp.com/files/sensors/doc/reports_presentations/ARTICLE_REPRINT.pdf

IMU/AHRS projects (accelerometer, gyroscope, magnetometer)

scikit-kinematics	http://work.thaslwanter.at/skinematics/html/index.html	
Madgwick's algorithm (2010)	https://www.samba.org/tridge/UAV/madgwick_internal_report.pdf	
IMU Data Fusing	http://www.olliw.eu/2013/imu-data-fusing/	
9DoF Sensor Fusion	https://github.com/kriswiner/MPU-6050/wiki/Affordable-9-DoF-Sensor-Fusion	
Experimental Comparison of Sensor Fusion Algorithms for Attitude Estimation (A Cavallo et al IFAC 2014)		
	http://www.nt.ntnu.no/users/skoge/prost/proceedings/ifac2014/media/files/1173.pdf	
BerryIMU	http://ozzmaker.com/berryimu-quick-start-guide	https://github.com/mwilliams03/BerryIMU
MicroPython 6/9-DOF fusion (needs gyro -> not for BBC micro:bit)	https://github.com/micropython-IMU/micropython-fusion	

Python, Motion Sensors and Data Fusion

PyCon UK, 15 September 2016

Stephen Simmons

mail@stevesimmons.com

https://github.com/stevesimmons/pyconuk-motion-sensor-data-fusion