# EE595 Final Presentation
# Trajectory Tracking & Service

Team 9

20217044 HyungJun Yoon

20160479 Seungjoo Lee

# Agenda

1. We will present Part 2 first (Mobile Service)
    a. Motivation & Problem scenario
    b. Our approach
    c. System implementation
    d. Evaluation

2. Part 1 as a subpart of Part 2 (Trajectory Tracking)
    a. Technical details behind Trajectory Tracking
    b. Evaluation for Trajectory Tracking

# Part 2
# Designing Mobile Service (System Part)

# Motivation

Trajectory tracking for **better video screen manipulation**!

# Target Scenario (1) - Usefulness

*When user is close to the screen,*
*user expects higher resolution*
*or Overview Beside Detail[1] with the video*

*When user gets far from the screen,*
*scaled-up screen size is preferred*

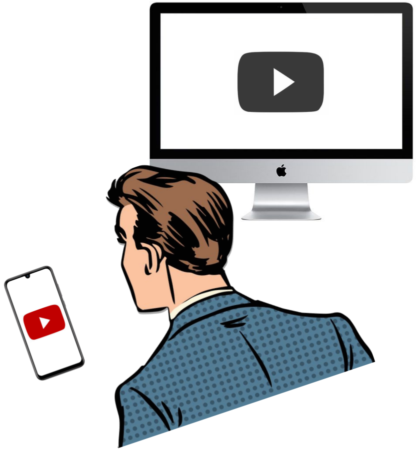
[1] http://www.cs.helsinki.fi/u/salaakso/patterns/Overview-beside-Detail.html

# Target Scenario (2) - Usefulness

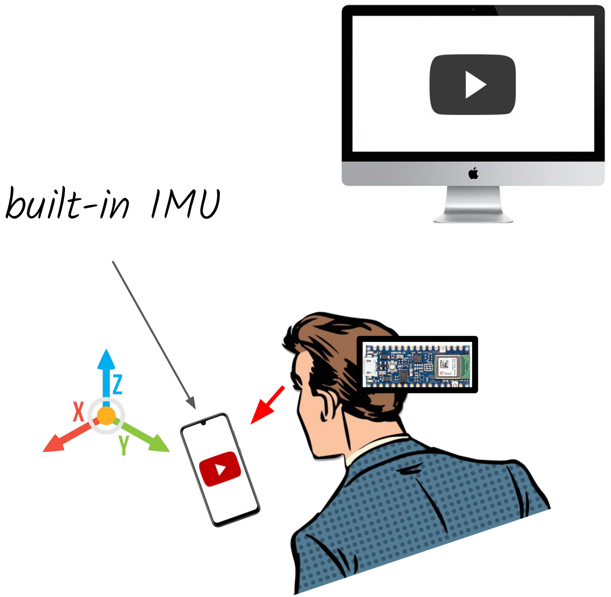
When user lies down,
rotated screen is expected

# Target Scenario (3) - Usefulness

*When the using device is changed,*
*user wants to continue watching video*



# Approach

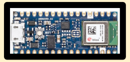



**IMU sensor** knows

- orientation of your head
- how far you are from monitor
- whether you are lying

→ Use **trajectory tracking** for **context-aware** video screen manipulation!

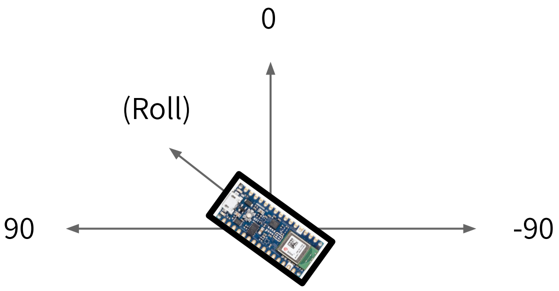
# System implementation - Technical Details



Trajectory tracking (part 1)
- Location tracking (x, y, z)
- +) Orientation tracking (**pitch, yaw, roll**)

Trajectory tracking (Android)
- Location tracking (x, y, z)
Tracks link of the video

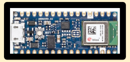
# System implementation - Technical Details




If roll is close to **0**, user is not lying

If roll is close to **90**, user is lying left

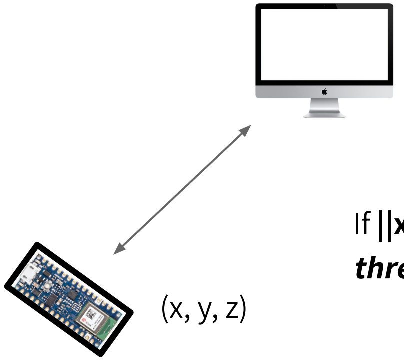
If roll is close to **-90**, user is lying right



Trajectory tracking (part 1)
- Location tracking (x, y, z)
- +) Orientation tracking (**pitch, yaw, roll**)

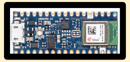
Trajectory tracking (Android)
- Location tracking (x, y, z)
Tracks link of the video

# System implementation - Technical Details




If **||x-monitor_x, y-monitor_y, z-monitor_z||**$_2$ is larger than ***threshold_scaleup***, the video should **be scaled up**

ARDUINO



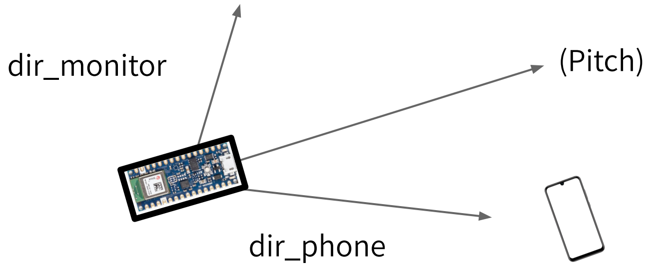
Trajectory tracking (part 1)
- Location tracking (x, y, z)
- +) Orientation tracking (**pitch, yaw, roll**)

android

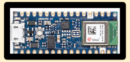
Trajectory tracking (Android)
- Location tracking (x, y, z)
Tracks link of the video

# System implementation - Technical Details

If the **angle between the direction of Pitch and dir_phone** has smaller value than the angle with dir_monitor, the focus **should be changed to phone**




ARDUINO



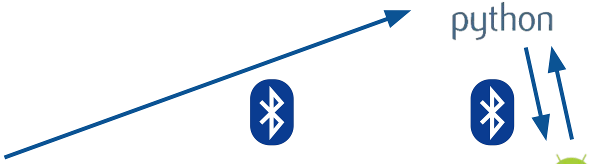
Trajectory tracking (part 1)
- Location tracking (x, y, z)
- +) Orientation tracking (**pitch, yaw, roll**)

ANDROID

Trajectory tracking (Android)
- Location tracking (x, y, z)
Tracks link of the video

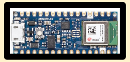
# System implementation - Technical Details

**Central** for the bluetooth connection
Combines all trajectory data
Generates user **context summary**



**Generated summary**

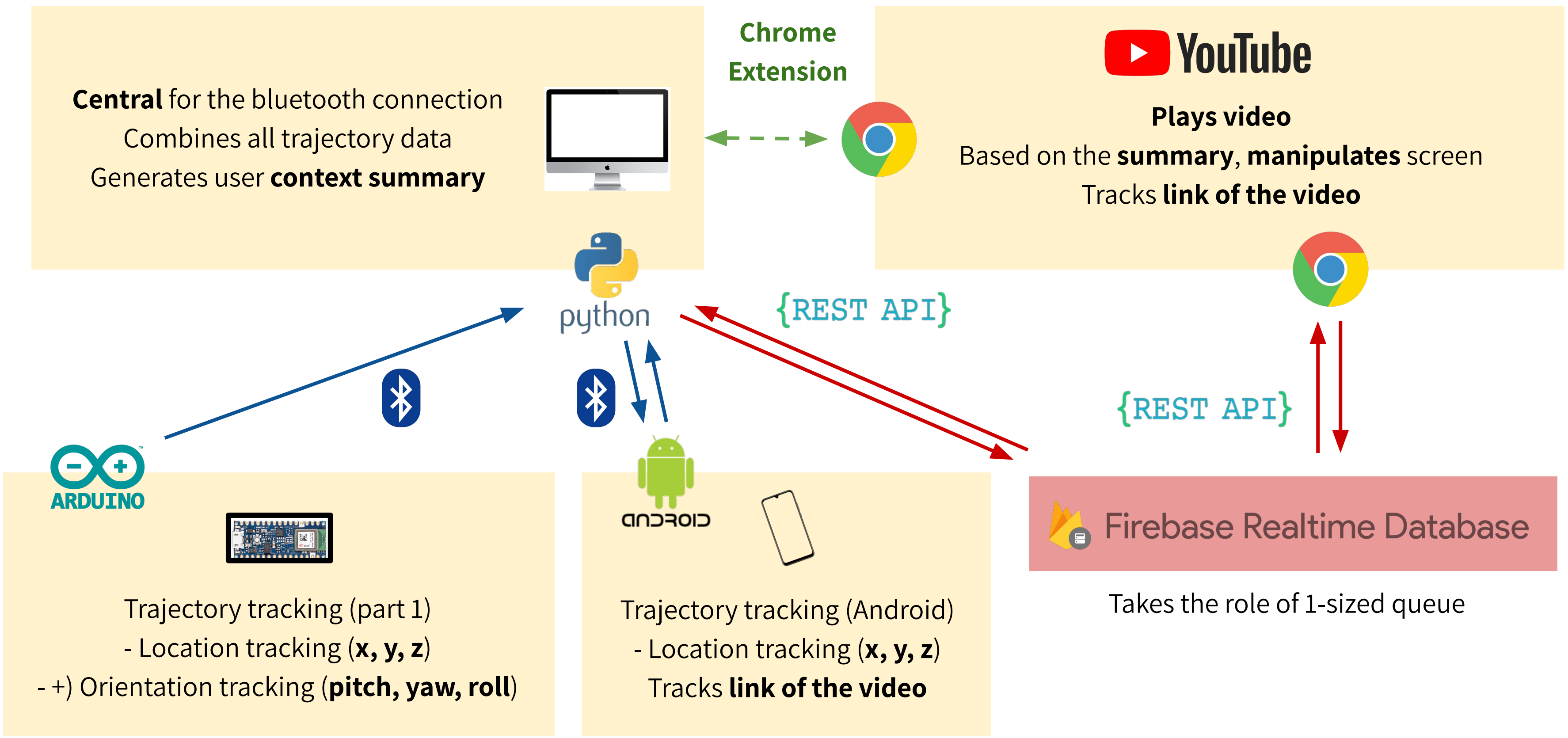
- distance_from_monitor: 42
- focus: "monitor"
- timestamp: 162342432
- user_direction: "left_lying"
- video_link_phone: "youtube.com/watch=abc?t=12"
- video_link_web: "youtube.com/watch=def?t=14"



Trajectory tracking (part 1)
- Location tracking (x, y, z)
- +) Orientation tracking (**pitch, yaw, roll**)

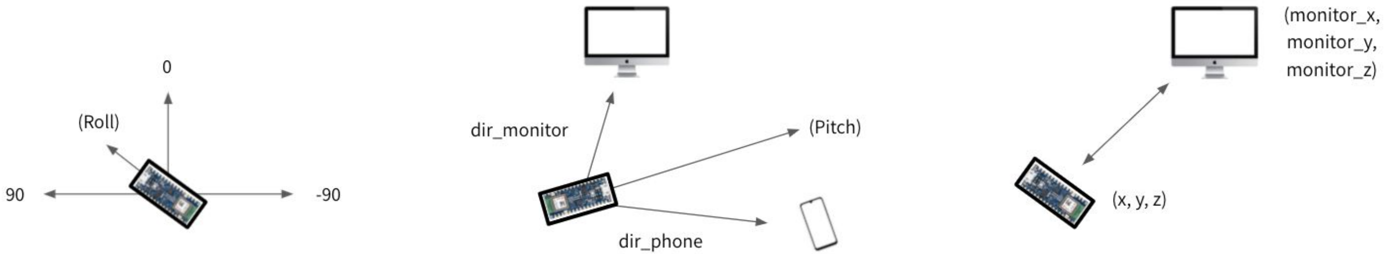
Trajectory tracking (Android)
- Location tracking (x, y, z)
Tracks link of the video

# System implementation - Technical Details



# Demo Video

# Evaluation - Performance




All of our systemic components use the value **derived from trajectory data**.

- location_x, location_y, location_z, pitch, roll, yaw

So we evaluate our system with the listed trajectory components

- What is new?
    - Evaluation about **orientation**
    - Evaluation about **location where change in orientation exists**

# Part 1
# Trajectory Tracking
# (Arduino Part)

# Providing the robust mobile service

- Accurate distance & orientation tracking
- Removing rotating effect from accelerometer in **real-time**
  - Rotating changes the gravity for each axis
  - Accelerometer cannot distinguish linear acceleration from the gravity

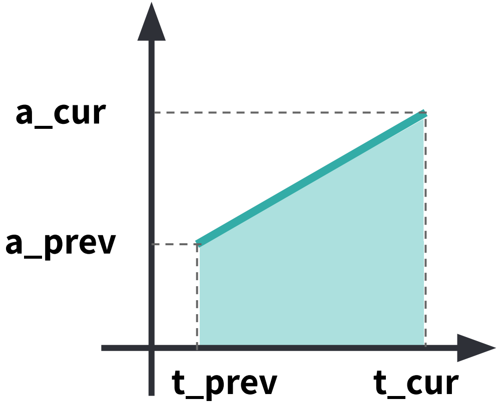
# Hardware configuration

- Changed sampling rate to **476Hz** by changing the register value
- Used IMU internal **FIFO queue** to store sensor value while doing other calculations
- Implemented library function that gives **average value of *N* sensor values** in FIFO queue
  - Calculation is slower than the sampling rate
  - Consider all collected sensor values in real-time

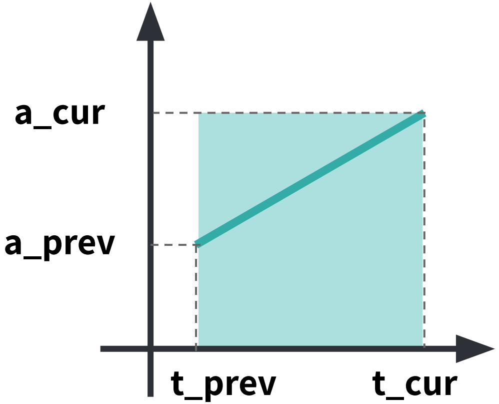
https://www.st.com/resource/en/datasheet/lsm9ds1.pdf

# Accurate distance tracking

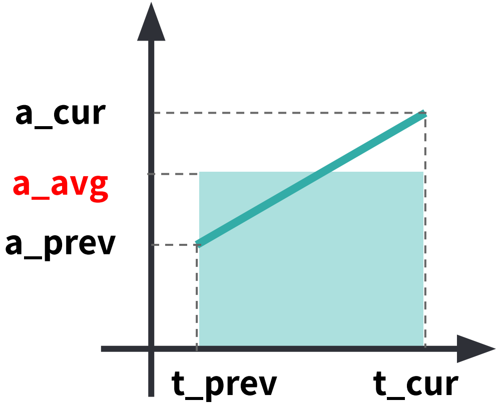
- Correcting integration error
  - Assume acceleration changes linearly in short time




Correct integration




Using **a_cur**




Using **a_avg**

Do the same when integrates *velocity* to *position*

# Orientation tracking

- **Integrate angular speed** of each axis from the gyroscope
    - Same setting with the accelerometer (sampling frequency, FIFO queue)
- **Calibrating** the gyroscope beforehand
    - Measuring the gyroscope value 200 times when the arduino is not moving
    - Average value is the error

# Removing effect of gravity change

- Getting orientation from the gyroscope (yaw, pitch, roll)
- Calculating the gravity in the arduino's coordinate using **rotation matrix**

$$R = R_z(\alpha)\, R_y(\beta)\, R_x(\gamma) = \overset{\text{yaw}}{\begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}} \overset{\text{pitch}}{\begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}} \overset{\text{roll}}{\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix}}$$

$$= \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix}$$

https://en.wikipedia.org/wiki/Rotation_matrix

# Removing effect of gravity change

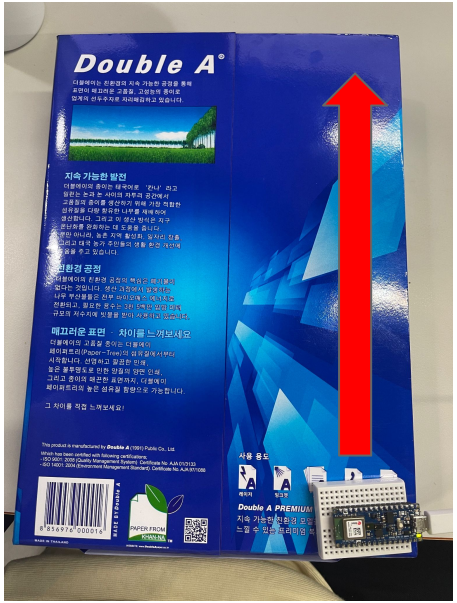
- Getting orientation from the gyroscope (yaw, pitch, roll)
- Calculating the gravity in the arduino's coordinate using **rotation matrix**

Gravity in the arduino's coordinate:

$$R^{-1}[0 \quad 0 \quad 1]^T = \begin{bmatrix} -\cos(\alpha)\sin(\beta)\cos(\gamma) + \sin(\alpha)\sin(\gamma) \\ \sin(\alpha)\sin(\beta)\sin(\gamma) + \cos(\alpha)\sin(\gamma) \\ \cos(\beta)\cos(\gamma) \end{bmatrix}$$

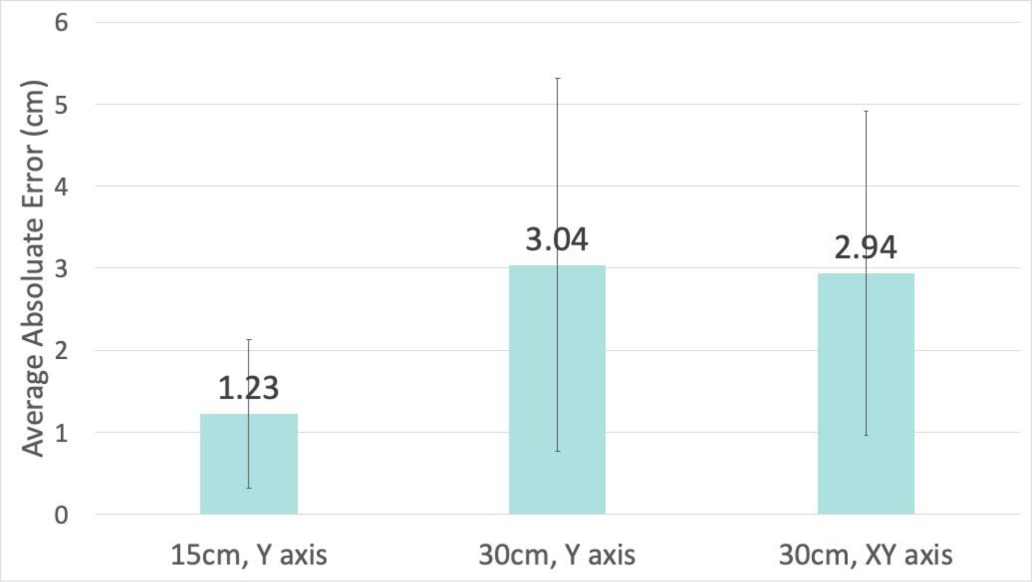
# Evaluation

- **Evaluates part 1 & 2 simultaneously**
  - Distance tracking
  - Orientation tracking
  - Gravity cancelling



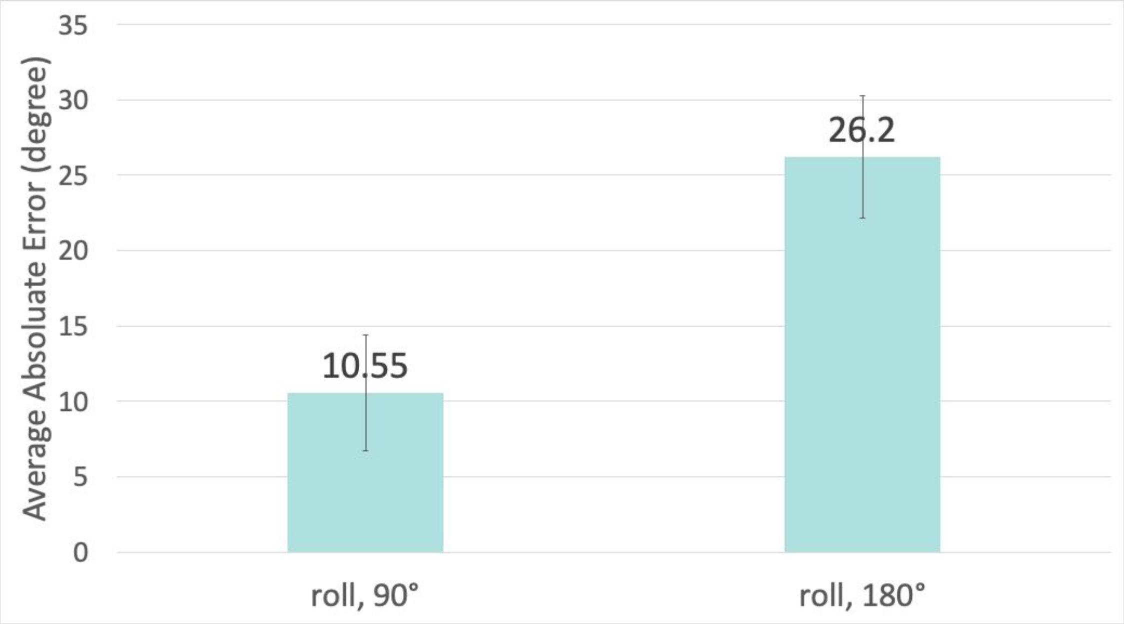

# Evaluation : distance tracking

- Tried 20 times for each
- Moved the arduino in straight line
  - XY axis means moved the arduino diagonally
- Distance is calculated by $\sqrt{x^2 + y^2 + z^2}$



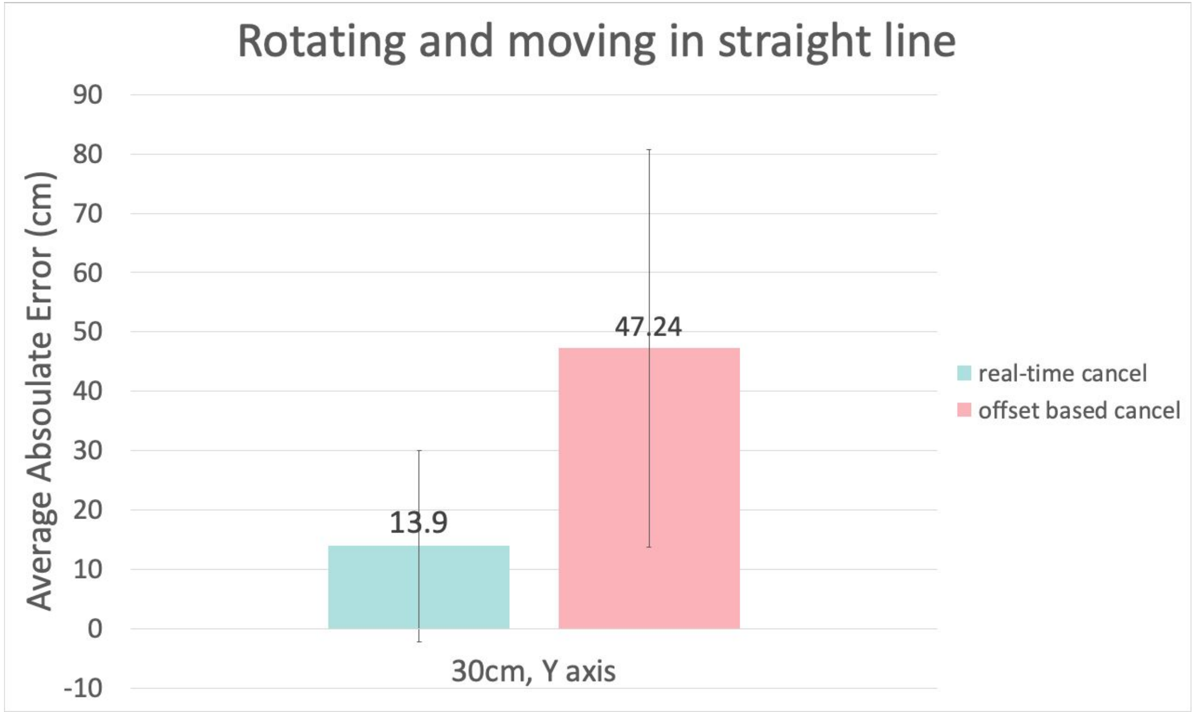
# Evaluation : orientation tracking

- Tried 20 times for each
- Evaluate only roll without loss of generality
  - Roll, yaw, pitch are calculated in the same way



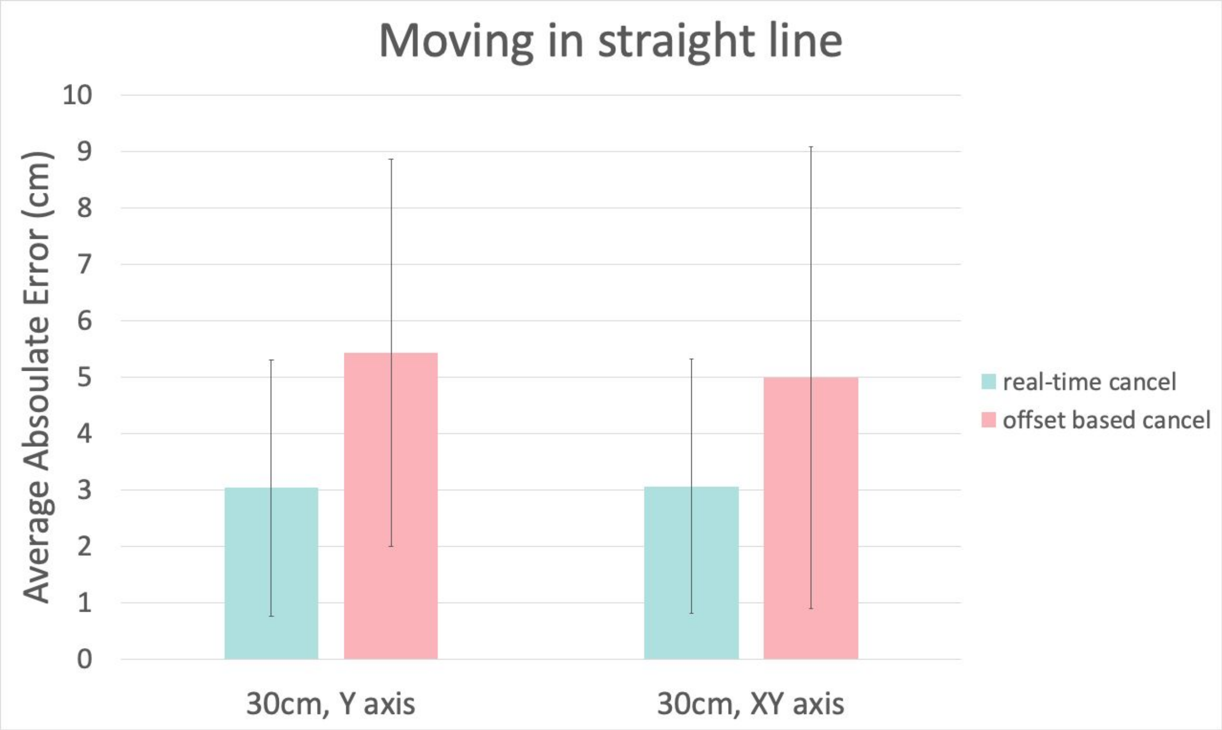
# Evaluation : gravity cancelling

- Tried 20 times for each
- Rotate roll +90°, roll -90°, and move 30cm



# Evaluation : gravity cancelling

- Tried 20 times for each
- Hand can shake while moving the arduino



# Roles of Each Teammate

1. Part 2
   a. HJ
      i. Implementing summary generation & system architecture
      ii. Youtube screen manipulation functionality (chrome extension)
   b. SJ
      i. Bluetooth connection and data transfer among devices (arduino, phone, PC)

2. Part 1
   a. HJ
      i. reducing algorithmic error (integration error)
   b. SJ
      i. hardware configuration
      ii. gravity cancelling, orientation tracking

# References


1. https://www.st.com/resource/en/datasheet/lsm9ds1.pdf
2. https://en.wikipedia.org/wiki/Rotation_matrix
3. http://www.cs.helsinki.fi/u/salaakso/patterns/Overview-beside-Detail.html
4. https://ux.stackexchange.com/questions/3643/should-video-always-play-full-screen-on-cell-phones

# Accurate distance tracking

```
void loop() {
  IMU.readAcceleration(AccX, AccY, AccZ);
  do_something();
}
```

- **Lose sensors values** while in `do_something()`
  - Use **FIFO queue**
  - IMU sensor puts the sensor value in the FIFO queue **regardless of the code execution**

# Accurate distance tracking

```
void loop() {
  IMU.readAcceleration(AccX, AccY, AccZ);
  do_something();
}
```

- **Lose sensors values** while in `do_something()`
  - Use **FIFO queue**
  - IMU sensor puts the sensor value in the FIFO queue **regardless of the code execution**
- However, **FIFO queue grows** if the calculation is slower than sampling rate

Implemented library function that gives **average value of *N* sensor values** in FIFO queue