PointBlank

B04901036 林厚均 B04901165 陳致維

Outline

- Motivation
- Goals
- Methods
  - Qt (for GUI)
  - Bluetooth Low Energy (for communication)
  - Motion Sensor (for mouse position)
- Results
- Reference

Motivation

- Replace traditional laser pointer
- Safer and more versatile
- Inspired by a commercial product

Goals

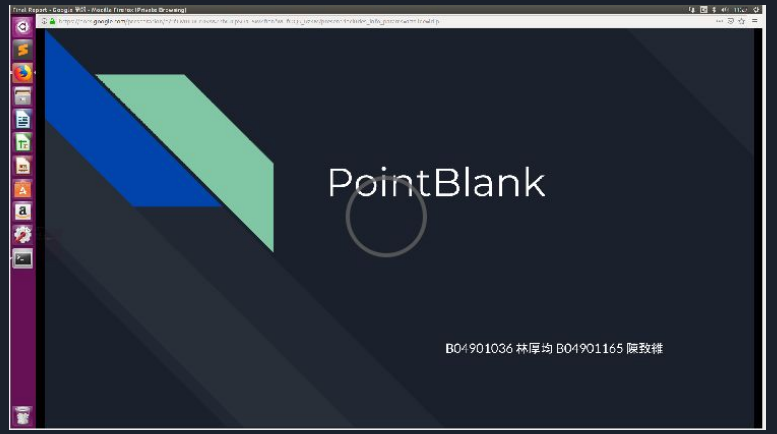
- Determine pointer’s projection on screen without cameras and physical (light) markers
- Basic functions of a presentation pointer
  - Next slide
  - Previous slide
  - Display the position of the projection
- Additional features
  - Control mouse position
  - Simulate mouse click
  - Act as magnifying glass
  - Customization

Methods (GUI)

- Using Qt
- Creates a full screen transparent window that ignores keyboard and mouse inputs
- Receives coordinates from the pointer and draws on the screen according to the current mode
- Also provides user interface for customization as a seperate program

Four modes (1)

1. drawEllipse()
   - center coordinates, radius
   - using pen width, pen color



Four modes (2)

1. fillRect()
   - fill the whole screen with color
2. drawEllipse()
   - fill with color (0,0,0,0)
   - full transparent circle around mouse position

Four modes (3)

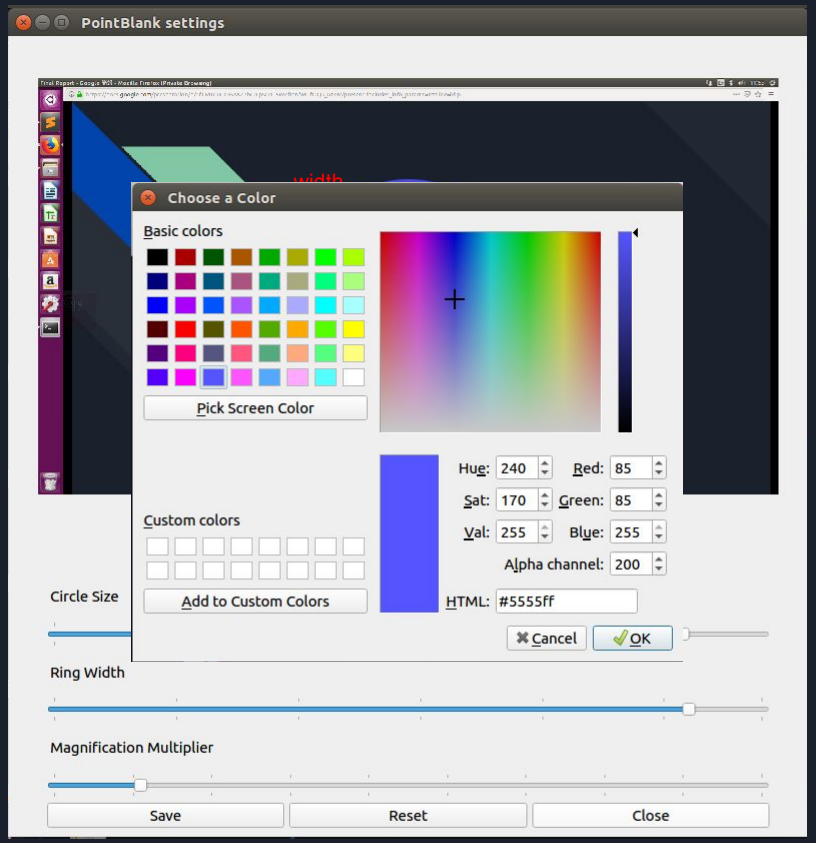
1. grabWindow()
   - take screenshot when button is pressed
2. scale and reposition the image
3. setClipRegion
   - the scaled image only appears inside the circle

Four modes (4)

- The previous modes are for slideshows
  - their left and right buttons correspond to keyboard UP and DOWN respectively
- This mode simulates mouse function
  - with mouse left click and right click

Customization

- settings.py
- Provides instant preview
- Four customization factors
  - color
  - circle size
  - ring width (under mode 1,3)
  - magnification (under mode 3)



Methods (Bluetooth)

- Using Qt for central
  - Discover advertising Raspberry Pi and connect to it
  - Enable notifications for Position and Buttons characteristics
  - Update GUI whenever a notification is received
- Using bluez for peripheral
  - Advertisement
  - Service: PointBlank
  - Characteristics: Position, Buttons
  - Position returns (x, y) with x and y being floats between 0 and 1
    - referring to the boundaries of the screen
  - Buttons returns an integer indicating which button is pressed or released

Establishing Bluetooth Connection

1. QBluetoothDeviceDiscoveryAgent.start()
    - Search for nearby low energy services
2. QLowEnergyController.connectToDevice()
    - connnect to device with service uuid “00003125-0000-1000-8000-00805f9b34fb”
3. QLowEnergyController.discoverServices()
4. QLowEnergyService.discoverDetails()
    - For each service, in this case we only have one
5. Write “0x0100” to the descriptors of Position and Buttons characteristic
    - to enable notification

Methods (Motion Sensor)

- MPU9250
- Self-defined operation state:
  - Sleep: all sensor power off
  - Active: read gyroscope and accelerometer at ~500Hz
  - Switch by button

- Data process:
  - Represent orientation: frame and transformation
  - Update orientation: 6-axis data fusion
  - Initialize orientation
  - Reduce vibration: vibration from hand and sensor noise

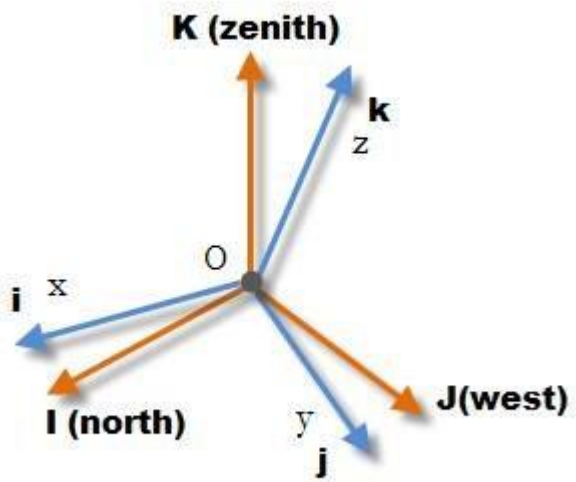
Orientation Representation

- Recording the difference between
  - Sensor frame
  - Earth frame

  by some representation

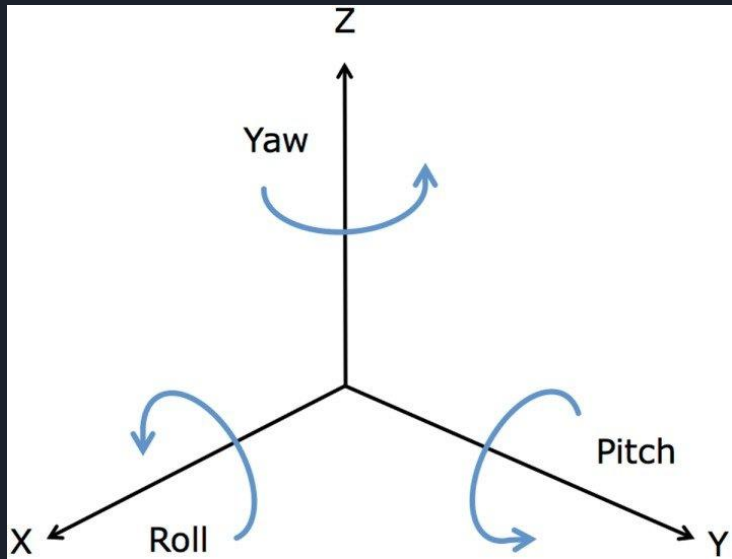
- Rotation matrix
  The same as recording the three axes of earth frame in sensor frame

- Numerical error
  Cannot ensure the three axes keep perpendicular to each other after several update



Roll-Pitch-Yaw (Euler Angles)

- Simple representation
  three rotations apply on x, y, z axis sequentially
- Consistent to gyroscope output
- Computational demanding:
  one vector transformation needs
  - three rotation matrix multiplication
  - and lots of sine, cosine computation

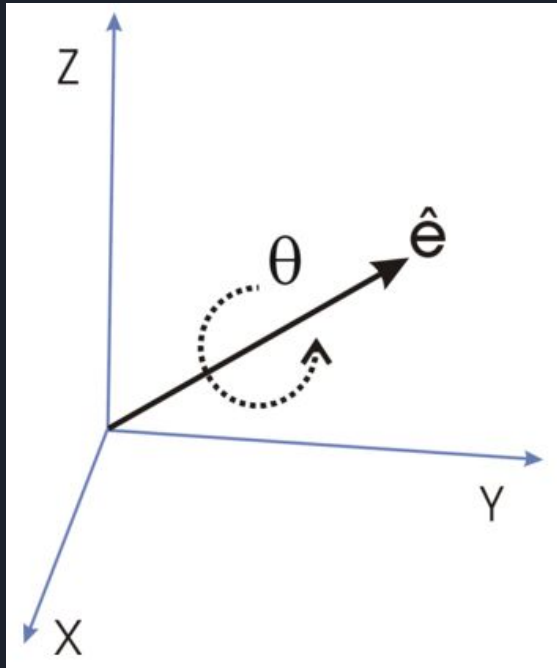


Quarternion

- Euler rotation theorem:
  Any 3D rotation can be represent as a rotation along some axis

- Quarternion catch this property by an indirect way:
  $[\cos \theta/2, e_x \sin \theta/2, e_y \sin \theta/2, e_z \sin \theta/2]$

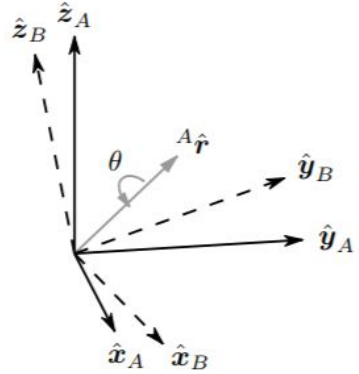
- All rotation operations become linear
  - operations like rotation matrix
  - smaller numerical error
  - faster computation



Quarternion Operation

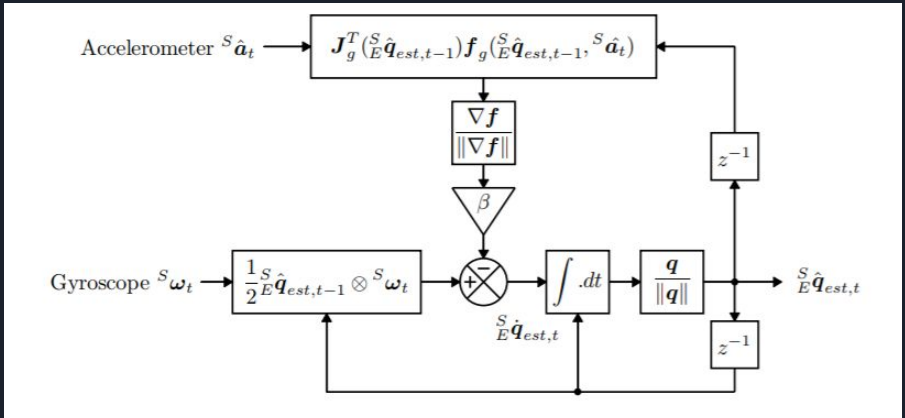
- Extention of complex number:
  - $i^2 = j^2 = k^2 = -1$
  - $ij = k$
  - $[w, x, y, z] = w + xi + yj + zk$
  - $\mathbf{a}^* = [a_1, -a_2, -a_3, -a_4]$
- Unit quarternion and 3D rotation are isomorphic:
  - Quarternion ${}^A_B\hat{\boldsymbol{q}}$ can represent rotation from frame A to frame B
  - sequential: ${}^A_C\hat{\boldsymbol{q}} = {}^B_C\hat{\boldsymbol{q}} \otimes {}^A_B\hat{\boldsymbol{q}}$
  - transformation: ${}^B\boldsymbol{v} = {}^A_B\hat{\boldsymbol{q}} \otimes {}^A\boldsymbol{v} \otimes {}^A_B\hat{\boldsymbol{q}}^*$, $\mathbf{v} = [0, v_1, v_2, v_3]$
  - backward: $\mathbf{q}^*$
- Both Unity and OpenGL support quarternion

$$\begin{aligned}\boldsymbol{a} \otimes \boldsymbol{b} &= \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \end{bmatrix} \otimes \begin{bmatrix} b_1 & b_2 & b_3 & b_4 \end{bmatrix} \\ &= \begin{bmatrix} a_1b_1 - a_2b_2 - a_3b_3 - a_4b_4 \\ a_1b_2 + a_2b_1 + a_3b_4 - a_4b_3 \\ a_1b_3 - a_2b_4 + a_3b_1 + a_4b_2 \\ a_1b_4 + a_2b_3 - a_3b_2 + a_4b_1 \end{bmatrix}^T\end{aligned}$$



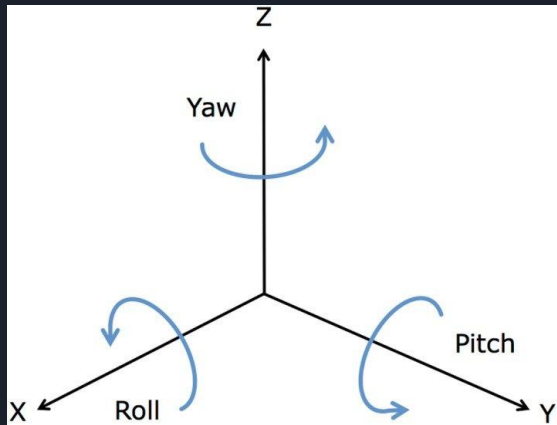
Update Orientation

- 6-axis data fusion
  - Gyroscope has drift problem from integral
  - Accelerometer itself cannot generate a sensor frame
- Madgwick orientation filter
  - Quarternion representation
  - use fixed vector (gravity) to adjust gyroscope



Initialize Orientation

- Problem
  - Madgwick filter assumes gravity is on positive z-axis
  - Correlation between initial sensor frame and horizontal movement
- Goal
  - Pointer usage is independent of pitch angle and roll angle
  - x-axis: front direction (pointer direction)
  - y-axis: horizontal direction
  - z-axis: vertical direction

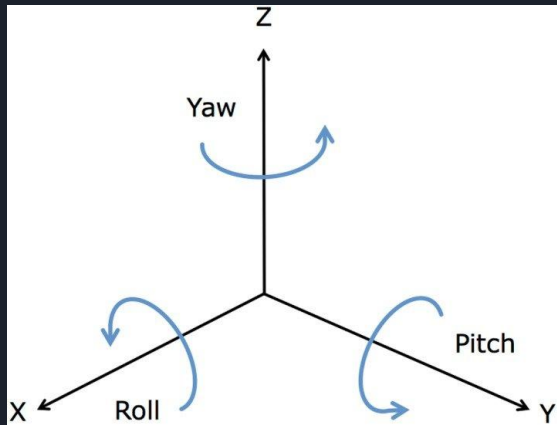


Initialize Orientation (Cont.)

- Add a beginning body frame
- From sensor frame to beginning frame: roll angle
  - horizontal direction: outer product of pointer direction and gravity
  - compute the angle between y-axis and horizontal direction
  - correct roll angle

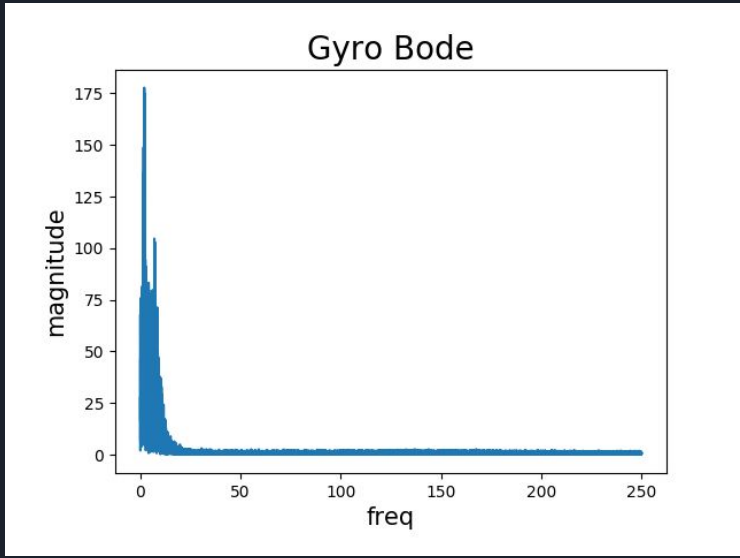
- From beginning frame to earth frame: pitch angle
  - compute the angle between z-axis and vertical direction (gravity)
  - correct pitch angle

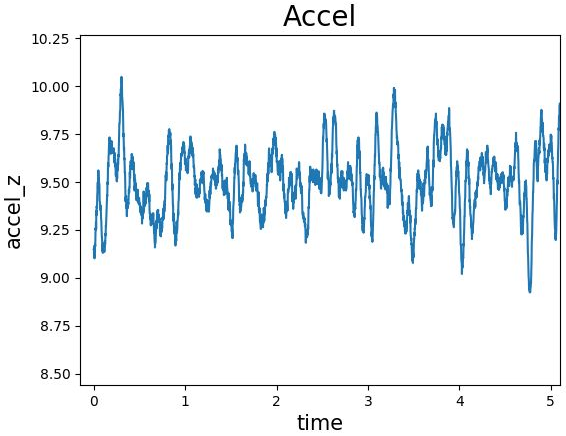
- Update sensor-to-earth quarternion by Madgwick filter
- Compute the difference between pointer direction and beginning direction in beginning frame

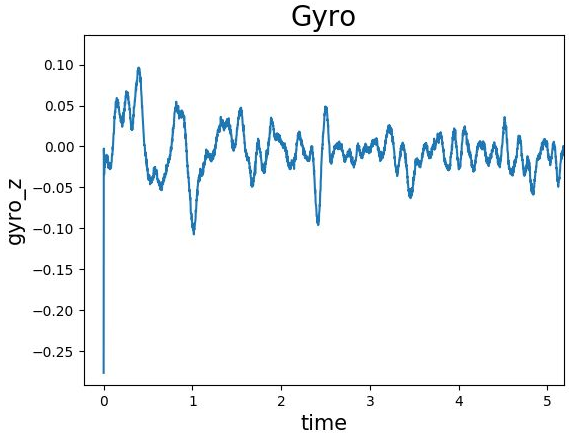


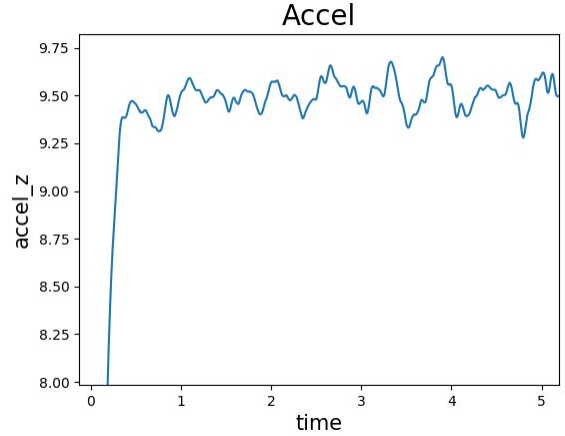
Reduce Vibration

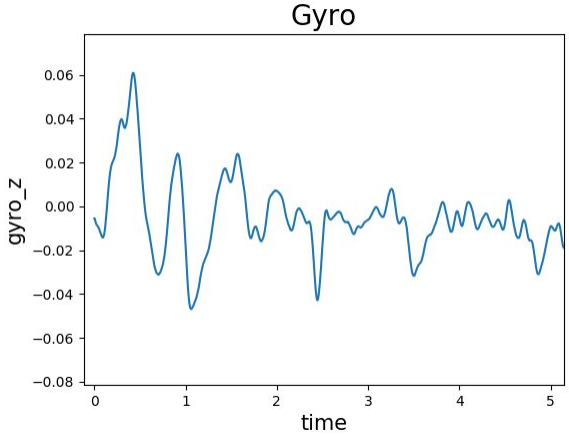
- Problem
  - Hand natural vibration (1~10Hz)
  - Sensor white noise
- Goal
  - Minimize the vibration of the pointer output
  - Reasonable response time (negligible delay)
  - Trade-off
- Solution
  - SImple first order IIR filter
  - output[n] = 0.02 * input[n] + 0.98 * output[n-1]
  - Reduce ~70% vibration
  - Delay: ~0.1s




Accel
accel_z
time
10.25
10.00
9.75
9.50
9.25
9.00
8.75
8.50
0
1
2
3
4
5

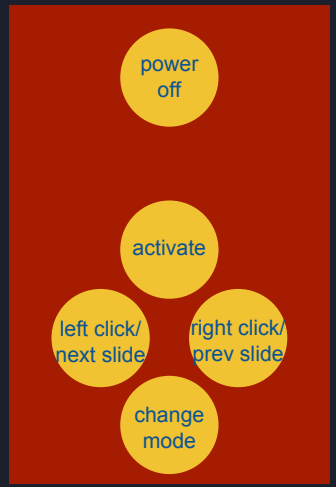

Gyro
gyro_z
time
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
0
1
2
3
4
5

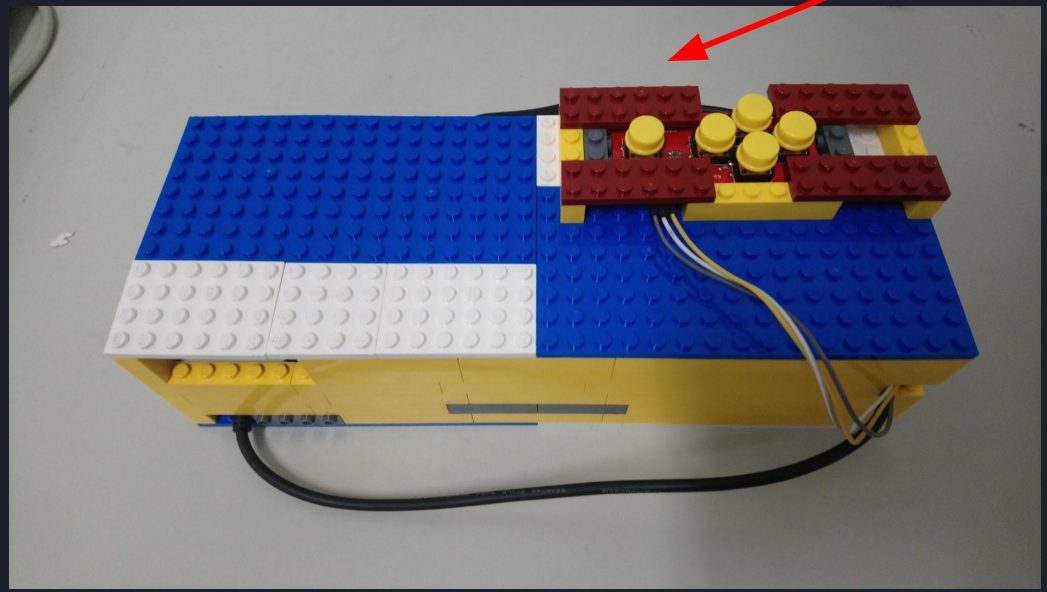

Accel
accel_z
time
9.75
9.50
9.25
9.00
8.75
8.50
8.25
8.00
0
1
2
3
4
5


Gyro
gyro_z
time
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
0
1
2
3
4
5

Results

- With Raspberry Pi, Arduino and a power bank inside
- Flip switch to power on
- Press button to shutdown normally, then flip switch to cut power
- Run PointBlank.py on PC
- Point at the center of the screen when activating





Reference

- http://doc.qt.io/
- Example codes in bluez folder
- MPU 9250 Datasheet
- Sebastian O. H. Madgwick, “Estimation of IMU and MARG orientation using a gradient descent algorithm”, 2011 IEEE International Conference on Rehabilitation Robotics

Thank you for your attention