University of Tehran

College of Engineering

School of Electrical and Computer Engineering (ECE)
School of Mechanical Engineering (ME)

Mechatronics & Robotics

Mini Project 1

Teaching Assistants:

Parsa Namazian (Questions 1 and 4)
Sina Kazemi (Questions 2 and 3)

Deadline: 24 March 2024 (5 Farvardin), 23:59

List of Problems

Problem 1: Angle Recording using MPU-6050 (25 points) 3

Problem 2: Angle Filtering (25 points) 6

Problem 3: MPU-6050 as a Game Controller (25 points) 8

Problem 4: Motion Capturing Using MediaPipe (25 points) 10

Problem 1: Angle Recording using MPU-6050 (25 points)

Introduction to Motion Sensors

Motion sensors are integral components of robotic systems, playing a vital role in tasks such as calibrating and controlling robotic arms through closed-loop mechanisms. These sensors are increasingly utilized in the field of rehabilitation robotics as well. Gyro sensors, also referred to as angular rate or angular velocity sensors, are MEMS-based devices designed to measure the angular velocity of moving objects.

In the market, there exists a variety of gyro sensors, including models like MPU-6050, MPU-9250, and GY-25. These sensors differ in precision, number of measurement axes, and additional features. The MPU-6050 stands out as a popular six-degree-of-freedom (6-DoF) inertial measurement unit (IMU) comprising a 3-axis accelerometer and a 3-axis gyroscope. Leveraging Micro Electro Mechanical Systems (MEMS) technology, it delivers precise measurements of acceleration, velocity, and orientation.

The MPU-6050 offers multiple full-scale ranges for both the accelerometer (±2g, ±4g, ±8g, ±16g) and the gyroscope (+/- 250°/s, +/- 500°/s, +/- 1000°/s, +/- 2000°/s). Equipped with a digital motion processor (DMP), this sensor can execute advanced signal processing tasks, making it suitable for diverse applications like smartphones, tablets, drones, robots, and gaming systems.

Compact in size at approximately 4 mm × 4 mm × 0.9 mm, the MPU-6050 operates on a 5V supply with low power consumption—drawing less than 3.6 mA during measurements and only 5 µA when idle. Its energy efficiency enables integration into battery-operated devices. Communication is facilitated through the I^2C bus interface, ensuring seamless compatibility with popular microcontrollers such as Arduino.

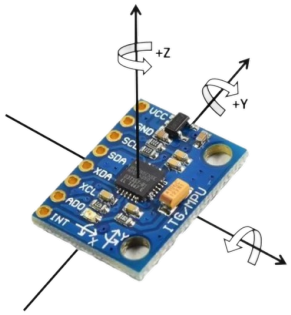

Figure 1: MPU-6050 module sensor.

Recording Data From MPU-6050

To leverage the capabilities offered by the MPU-6050 IMU, microcontroller platforms must be employed. One such option is the Arduino Uno board featuring the ATmega328P microcontroller.

In order to capture angular velocities and accelerations from the sensor, custom electronics design is necessary. Since the MPU-6050 supports communication via the I^2C protocol, circuits will be developed accordingly.

For streamlined interaction with the MPU-6050 sensor utilizing Arduino, library-based methods are recommended. Two prominent libraries, namely "Adafruit MPU6050" and "MPU6050 tockn," facilitate straightforward retrieval of calibrated sensor data. After integrating these libraries into your project, creating an instance of the MPU-6050 class becomes essential. With the object established, users may employ library functions to extract sensor data effortlessly. For instance, the "Adafruit MPU6050" library provides functions like 'getAcceleration()' and 'getGyro()', while the "MPU605 tockn" library supplies functions like 'getAccX()', 'getAccY()', 'getAccZ()', 'getGyroX()', 'getGyroY()', and 'getGyroZ()'. Additionally, these libraries offer functionalities such as sensor calibration, filtering, and sensor fusion, which further enhance the usability of the sensor data.

By leveraging these libraries, developers can rapidly obtain acceleration, gyroscope, and temperature data without having to deal with intricate register-level details. This approach is beginner-friendly, expedites implementation, and suits a broad range of applications that do not necessitate fine-tuning at the register level.

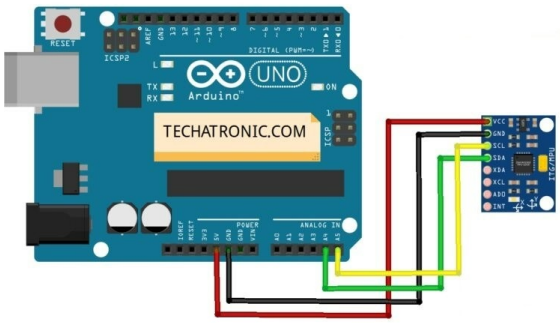

Figure 2: Designed circuit.

Calibration

To ensure accurate readings from MPU-6050 sensors, calibration is imperative to eliminate zero-error or inherent inaccuracies in the sensor data. Given the unique nature of each sensor, calibration becomes essential as the default readings from the MPU6050 may not be entirely precise. By calibrating the sensor, users can attain more reliable representations of angular acceleration values and effectively mitigate zero-error discrepancies.

The calibration procedure entails determining specific offset values for the sensor by placing it in a flat, level position and calculating three offsets that align the initial sensor readings closer to zero. This process can be facilitated through established libraries such as 'MPU6050 tockn' or 'Adafruit MPU6050', which offer user-friendly functions for retrieving calibrated sensor data. The obtained calibration values can then be integrated into the initialization code at the beginning of any sketch utilizing the MPU-6050 sensor.

Calibration serves as a fundamental step in achieving precision in sensor readings and is crucial for applications that demand accurate measurements. By calibrating the MPU-6050 sensor, users can enhance the reliability and accuracy of their data outputs, ensuring optimal performance across various applications.

In this link, there is a code uploaded for this section. Using this code, the quaternion value and roll, pitch, yaw angles of the sensor can be recorded.

Tasks

1. Implement the circuit using Arduino-Uno and MPU-6050 to record the yaw, pitch, and roll angles, also the quaternion value. Record a video file from the function of your system.

2. Plot each of the values above using the Serial-Plot application. Using this link, you can download and install the application. Record a video file from the function of your system.

Note: In the recorded videos, you have to show both your setup and the output in the software at the same time. Each video should be no longer than 15 seconds.

Problem 2: Angle Filtering (25 points)

Signal processing employs filters, which serve to eradicate undesirable elements or characteristics from signals. A distinguishing trait of filters lies in their ability to either completely or partially suppress specific aspects of the signal. Typically, filters target the removal of certain frequencies or frequency bands, although they aren't confined solely to the frequency domain. Particularly in the realm of image processing, correlations can be eliminated selectively for specific frequency components without necessarily acting in the frequency domain.

Filters permeate various domains, encompassing electronics, telecommunications, radio, television, audio recording, radar, control systems, music synthesis, image processing, computer graphics, and structural dynamics. Classification schemes for filters are multifarious and overlapping, rendering a concise taxonomy elusive. Filters may be categorized as linear or nonlinear, time-variant or time-invariant, causal or non-causal, analog or digital, discrete-time (sampled) or continuous-time, passive or active type of continuous-time filter. Commonly applied filters in signal processing consist of low-pass filters, high-pass filters, complementary filters, Kalman filters, finite impulse response (FIR) filters, and particle filters. Each filter type fulfills distinct purposes, and selecting the appropriate filter relies upon the specific requirements of the application.

First Order Complementary Filter

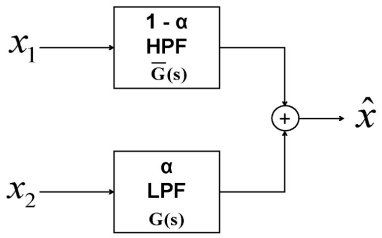

Figure 3: Complementary Filter Block Diagram.

A complementary filter is a signal processing technique used with IMUs like the MPU-6050 to combine data from the accelerometer and gyroscope to obtain more accurate orientation estimates while minimizing drift. The complementary filter blends accelerometer data (sensitive to linear acceleration and gravity) and gyroscope data (measuring angular velocity) to produce a more stable and accurate estimation of orientation. The filter typically involves calculating pitch, roll, and yaw angles using both accelerometer and gyroscope data. By combining these measurements with appropriate weighting factors, the complementary filter provides a more reliable orientation estimate compared to using either sensor data alone. The filter helps in reducing noise, improving accuracy, and minimizing drift in orientation estimation. By combining accelerometer and gyroscope data effectively, it provides a more stable output suitable for applications requiring precise orientation tracking. The implementation of the complementary filter involves processing raw sensor data, applying filtering algorithms, and updating orientation estimates in real-time. The filter can be

implemented in Arduino using code snippets that calculate pitch angles using gyroscope data and accelerometer readings. The formula for the output of this filter is as follows.

$$Angle = \alpha_1 * (angle + gyroscope * dt) + \alpha_2 * accelerometer$$

$$\alpha_1 + \alpha_2 = 1$$

Kalman Filter

The Kalman filter is a powerful recursive algorithm that can be used to fuse data from multiple sensors, including the MPU-6050 sensor, to obtain more accurate orientation estimates. The Kalman filter accounts for measurement uncertainty and process noise, making it suitable for improving accuracy and reducing errors in orientation estimation when combined with data from the accelerometer and gyroscope of the MPU-6050 sensor. The filter involves predicting the state of the system based on the previous state and the current measurement, and then updating the state estimate based on the difference between the predicted and measured values. The filter also estimates the covariance of the state estimate, which can be used to quantify the uncertainty in the estimate.

To implement a Kalman filter for the MPU-6050 sensor in Arduino, you can use pre-built libraries available on platforms like GitHub or the Arduino Library Manager. For example, the "Arduino-KalmanFilter" library provides a simple implementation of the Kalman filter for the MPU-6050 sensor. The library includes a sample code that demonstrates how to use the filter to estimate the pitch and roll angles of the sensor. The code reads the raw accelerometer and gyroscope data from the sensor, applies the Kalman filter to the data, and then outputs the filtered pitch and roll angles.

Keep in mind that implementing a Kalman filter requires more computational resources than a complementary filter or low-pass filter, so optimization techniques must be employed to prevent slowdowns. Additionally, tuning the filter parameters, such as the process noise covariance matrix and measurement noise covariance matrix, can be challenging and requires a good understanding of the system dynamics and noise characteristics.

Tasks

1. Perform the first order complementary filter algorithm on the code in the previous problem.
2. Perform the kalman filter algorithm on the code in the previous problem. Using these links, you can access proper kalman libraries in Arduino Uno: Link 1 and Link 2.
3. Compare the results of the both filters and discuss.
4. Record a video from the function of your system.

Note: In the recorded videos, you have to show both your setup and the output in the software at the same time. Each video should be no longer than 15 seconds.

Problem 3: MPU-6050 as a Game Controller (25 points)

Overview

The goal of this problem is to create an interactive maze game controlled by physically moving and orienting an MPU-6050 accelerometer/gyroscope sensor. You will use the motion data from the sensor to update and control the position of a ball within a maze environment generated in Python.

The provided Python code in this Link (use the notebook in MP1 folder) utilizes the PyGame library to render graphics and handle gameplay for the maze. The code generates walls (rendered in cool blue) comprising the simple square maze layout along with a movable gray ball for the player avatar. Currently, this gray ball is controlled by the WASD keyboard keys to demonstrate basic movement mechanics. Pressing W makes the ball move up, A makes it move left, S down, and D right. The player uses these keys to navigate the ball through the openings in the walls and try to reach the end of the maze.

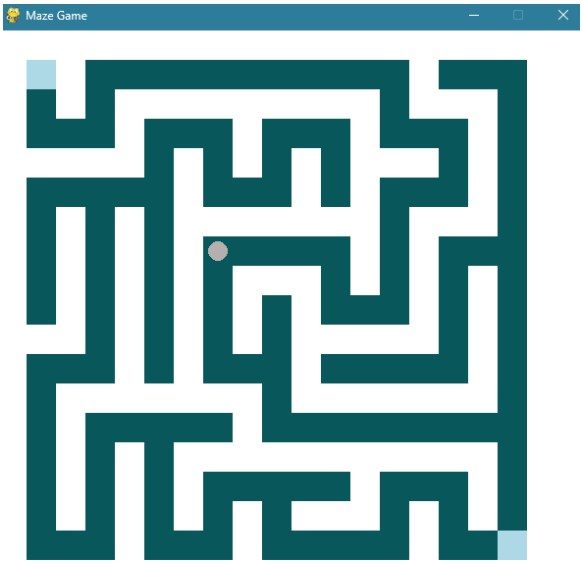

Figure 4: The Maze-game gameplay

Your task will be to replace this keyboard control scheme with input from the motion sensor. By tilting and rotating the MPU-6050, you will adjust the ball's position on screen. This creates a more physical and hands-on interaction for controlling the game compared to just pressing keys. The existing collision detection on the maze walls is still used to prevent moving through walls.

Processing this analog motion data and mapping sensor orientation to screen position requires calculating directionality vectors and scaling factors. The existing collision detection code can still be leveraged to prevent illegal passage through maze walls. Just the control mechanism is replaced from key presses to motion tilting.

Tasks

1. Connect the Arduino board to Python and read the data using Python.
2. Modify the Python game code to receive serial input from the Arduino instead of WASD keys. Map tilt motions to ball movement directions.
3. Demonstrate the working game by recording a video tilting the sensor to navigate the ball out of the maze (No loner than 30 seconds).

Note: In the recorded videos, you have to show both your setup and the output in the software at the same time.

Problem 4: Motion Capturing Using MediaPipe (25 points)

MediaPipe is an open-source framework developed by Google for creating customizable machine learning solutions for live and streaming media. It supports various platforms, including mobile (Android and iOS), web, desktop, edge devices, and Internet of Things (IoT). MediaPipe enables developers to rapidly prototype and deploy computer vision applications, offering ready-to-use solutions for tasks such as face detection, face mesh, iris detection, hands detection, and pose estimation. The framework is designed to be lightweight yet efficient, allowing it to run on battery-powered devices. MediaPipe uses a graph-based architecture consisting of nodes, packets, and graphs, providing flexibility and scalability for diverse applications.

MediaPipe Pose is a machine learning solution for high-fidelity body pose tracking, inferring 33 3D landmarks and background segmentation mask on the whole body from RGB video frames. It is a part of the MediaPipe framework developed by Google for creating customizable machine learning solutions for live and streaming media. MediaPipe Pose uses a graph-based architecture consisting of nodes, packets, and graphs, providing flexibility and scalability for diverse applications. The solution is designed to be lightweight yet efficient, allowing it to run on battery-powered devices. MediaPipe Pose is suitable for applications requiring precise body pose tracking, such as fitness tracking, augmented reality, and more. The solution is open-source and can be customized to meet specific requirements.

In this problem, you will be working with MediaPipe's pose detection to capture the human leg's motion and represent it using what you have learned from motion representation in the robotics course.

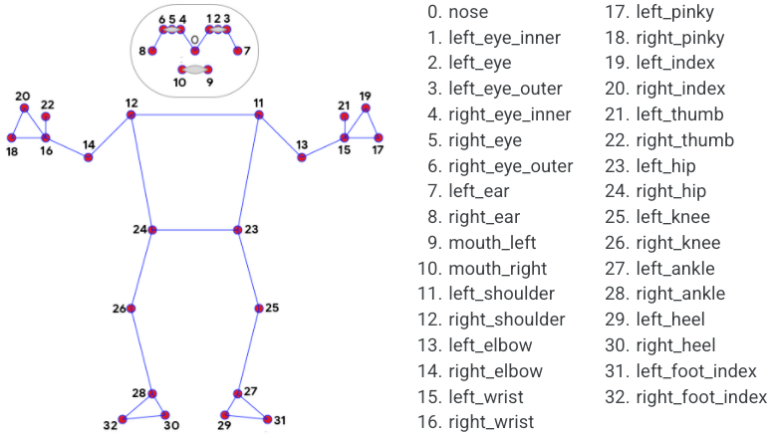

Figure 5: Details on the 33 landmarks detectable by MediaPipe Pose.

Figure 6: Sample of a picture being processed by MediaPipe Pose.

Tasks

Imagine that you want to capture and analyze the 2D rotational motion of a football player's leg in a video, using MediaPipe Pose. Due to the inaccuracy of Mediapipe in depth estimation, you have to record the data using two views. Therefore, it is suggested to record the motion from both lateral and frontal view of the leg simultaneously. Using these links, you can have a better knowledge on how this model should be implemented: Link 1, Link 2,

1. Record two videos from the lateral and frontal view of leg during a random rotational motion (No longer than 15 seconds). (**Note**: Take care to use two cameras with same frame rates.)
2. For the leg (link between ankle and knee), compute the values θ_x and θ_y for each frame and form the rotation matrix.
3. Compute the vector $\vec{e}$ and the value of quaternion at each frame.
4. Plot the values θ_x, θ_y, and quaternion with respect to time.
5. The plots in the previous item represents that your observation is noisy. Save the data and perform an offline smoothening process. Re-plot the results and compare them with the previous item. (**Note**: You can use functions in MATLAB for this process.)

Homework Guidelines and Instructions

- The deadline for sending this exercise will be until the end of Sunday, March 24th.
- This time cannot be extended and you can use time grace if needed.
- The implementation must be in Python programming language and your codes must be executable and uploaded along with the report.
- This exercise is done a group of two persons. Note that both members of a group will receive same mark for the project, and no complaints will be accepted for this. You must submit the members of your group in theGoogle Sheet Link.
- If any similarity is observed in the work report or implementation codes, this will be considered as fraud for the parties.
- Using ready-made codes without mentioning the source and without changing them will constitute cheating and your practice score will be considered zero.
- If you do not follow the format of the work report, you will not be awarded the grade of the report.
- Handwritten exercise delivery is not acceptable.
- All pictures and tables used in the work report must have captions and numbers.
- A large part of your grade is related to the work report and problem solving process.
- Please upload the report, code file and other required attachments in the following format in the system: `MP1_[Lastname]_[StudentNumber].zip`
  For example, the: `HW1_Ezati_12345678.zip`
- If you have questions or doubts, you can contact the assistants through the following e-mail with the subject 3HW_DGM. Stay in touch educationally:
  - The first and fourth question: `PNamazian@ut.ac.ir` (Parsa Namazian)
  - The second and third question: `SinaKazemi@ut.ac.ir` (Sina Kazemi)
- Be happy and healthy