# Wearable IMU-based Elbow Angle Estimation

Phu Nguyen, Rafael Perron, **James Walsh**

December 2025

The overview and team result **IS** the same for the entire group.

The source code, dataset and figures for this project are available at:
https://github.com/PhusNguyen/Elbow-Angle-Tracking-IMUs

## Overview

The goal of the elbow angle estimation project was to estimate human elbow angle using a pair of wearable IMU devices. Estimating the angles of the limbs in the human body is an extremely important practice in various health, rehabilitation, and performance applications, and better estimates in various environments and conditions can teach us more about how our bodies work. Although optical tracking systems are more accurate and reliable than IMU devices, they require calibrated laboratory environments, which limits the environments where data can be collected. IMUs and other wearable devices are cheaper and more accessible alternatives to optical tracking systems that also offer the ability to collect data outdoors and in more practical conditions. [1], [2], [3]

Our team set three project goals to meet:

1. Establish a working baseline system for elbow angle estimation and quantify fundamental limitations.
2. Implement and evaluate sensor fusion techniques and kinematics models to mitigate drift and improve long-term accuracy.
3. Create an intuitive visualization system for real-time monitoring and analysis of arm motion.

We met our first two goals; however, we did not have enough time to complete the visualization.

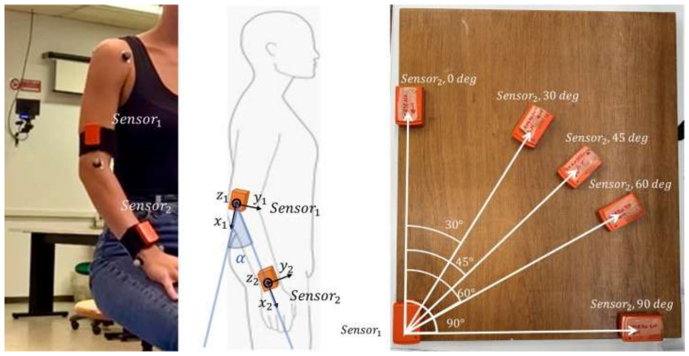

Figure 1: IMU Setup (a) and flexion/extension experiment on flat surface (b).

The IMU devices were strapped on the subject's forearm (near the wrist), and their upper arm between the elbow and the shoulder as shown in Fig 1a. In our version of the experiment, we set the sensors on a table, as shown in Fig 1b, to simulate controlled elbow flexion/extension rotation starting with the devices in a collinear position (0 degrees) and moving the second device into the 30-, 45-, 60- and 90-degree positions shown.

Our experiment used two Adafruit IMU BNO085 sensors and an ESP32-S2 microcontroller. The data were filtered and fused using a complementary filter, a Madgwick filter, and a filter from the SparkFun library.

## Team Results

### System architecture

Micro-ROS was implemented to bridge the microcontroller to the ROS 2 system. /dual_Imu_node node was created by micro-ROS to publish two topics – one for each IMU – that contain accelerometer and gyroscope data. Two filter nodes then subscribed to the raw IMU data and published fused orientations to /Imu1/data and /Imu2/data. An /angle_estimation node subscribed to get updated orientations from the 2 IMUs to calculate the relative angles.

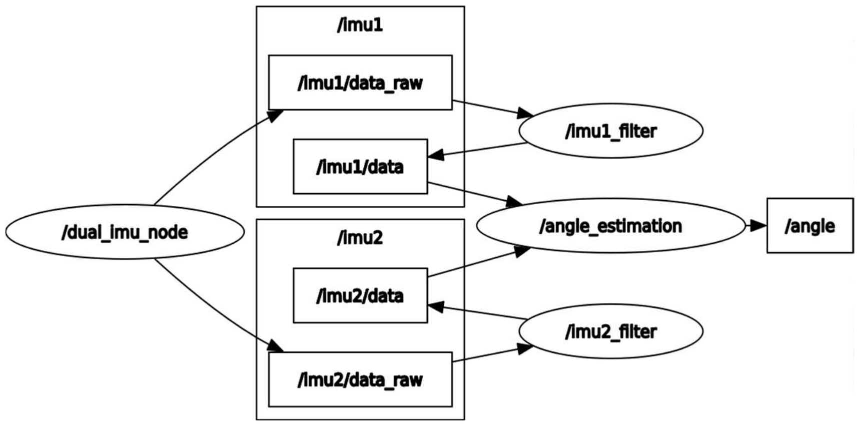

Figure 2: ROS2 node graph.

## Filters used and details

IMU sensors are prone to drift and inaccurate results. Three sensor fusion techniques were implemented before angle calculation. Separate rosbags were collected for each of those filters for comparison.

The first filter we used was a complementary filter which fuses accelerometer, gyroscope and sometimes magnetometer data to get a more accurate quaternion estimate using a weighted average. Imu_complementary_filter ros2 package was used for complementary implementation. use_magnetic_field_msg was set to false because we want to evaluate the combination of accelerometer and gyroscope.

The second filter we used was a Madgwick filter, which also fuses accelerometer and gyroscope data into a quaternion but aims to minimize gyroscope noise through gradient descent. imu_filter_madgwick ros2 package was used for the filter implementation; similarly, use_magnetic_field_msg was set to false.

The last filter we tried was from the SparkFun BNO080 Arduino Library. SparkFun GitHub repository has a rich set of examples for working with the same IMU devices that we had in conjunction with a microcontroller, and we decided to test the filtering practice they used in their examples to get fused quaternions.

## Angle Calculations

To calculate the elbow angle, we chose to use the IMU's orientation data, which is fused accelerometer and gyroscope data. The orientation is presented as a quaternion in a global/world frame. Before implementing an angle calculation algorithm, we first have to understand the setup of our two IMU sensors. As shown in Figure 1, one sensor is placed on the upper arm (above the elbow) and one on the forearm. Given this setup, we can begin to conceptualize how we will extract the elbow angle. To allow the most consistent data analysis, we placed the IMUs with the same orientation: both IMU's x-axis is pointing down the arm towards the hand, and both IMU's z-axis remain parallel to each other. For this phase of the project, we are performing pure flexion/extension movements of the arm and not rotation (single plane motion); this setup allows us to measure the relative yaw between the upper arm and forearm.

After understanding the setup, we can create an angle estimation algorithm. We began with initializing a bone axis $\hat{b}$, which is a unit vector in the sensor frame that points in the +x direction of the IMUs. We then rotate the bone axis into the world frame by the following calculation:

$$\hat{b} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\hat{u}_{upper} = q_{upper}\left(0, \hat{b}\right) q_{upper}^{*}$$

$$\hat{u}_{forearm} = q_{forearm}\left(0, \hat{b}\right) q_{forearm}^{*}$$

This results in two unit vectors pointing along the upper arm and forearm in the global frame. Given that the dot product between two unit vectors is the cosine of the angle between the two vectors, we simply perform this calculation to get the elbow angle. The calculation is shown below:

$$\hat{u}_{upper} \cdot \hat{u}_{forearm} = \cos(\theta)$$

$$\theta_{raw} = \cos^{-1}(\hat{u}_{upper} \cdot \hat{u}_{forearm})$$

After calculating the raw angle, the algorithm performs a simple calibration with the following assumption. At the beginning of data collection, the arm is fully extended, so the elbow angle is 0 degrees. After making this assumption, we calibrate the data, so the calculated elbow angle is always subtracted from the initial angle measured. The following calculation shows this:

$$\theta_{elbow} = \theta_0 - \theta_{raw}$$

This completes the algorithm: given the upper arm and forearm IMU orientations, we rotate a common bone axis into the world frame, compute the angle between the two segment directions, and apply a simple calibration to obtain the elbow flexion angle.

## Data collection details

To collect data, we set up the IMUs on a smooth table. We drew a dot on the table that would be the origin of our axes. From that point we drew a straight line approximately 10 centimeters in length; that line would be our X-axis. From the origin, we drew four more 10-centimeter lines at 30, 45, 60, and 90 degrees from the X-axis.

IMU1 was centered with its tip on the origin and its X-frame pointing collinear to the table's X-axis. IMU2 was placed along the X-axis so that the cable connecting it to IMU1 was stretched tightly and its X-frame collinear to IMU1. Every rosbag we recorded began with the IMUs in this configuration.

Once the IMUs were in starting position, we would run our launch files, check if the sampling rate was 40 Hz, and start recording our rosbag. Once recording began, we counted to 3 then slowly moved IMU1 toward the 30-degree lines so that it arched around the origin while the cable remained taut; we tried our best to ensure that the X-frame of IMU2 always remained collinear with the cable. This was then repeated for degrees 45, 60, and 90, pausing for 3 seconds each time we reached those angles. After reaching 90 degrees and waiting 3 seconds, we would pause our rosbags and ensure data were recorded properly. This procedure was repeated various times for each type of filter we used: complementary, Madgwick, and Sparkfun.

After plotting our data and determining the filter that gave us the best angle estimations, we moved to testing our arms. Using straps and sleeves, we mounted IMU1 on our upper arm, approximately 2 centimeters above the elbow and IMU2 on our forearm, approximately 5 centimeters above the wrist. We collected two rosbags: one where our arm was down by our side, and we flexed our elbow upwards, and one where our arm was fully stretched out perpendicular to our body, and we flexed our elbow inward.

## Plots and results

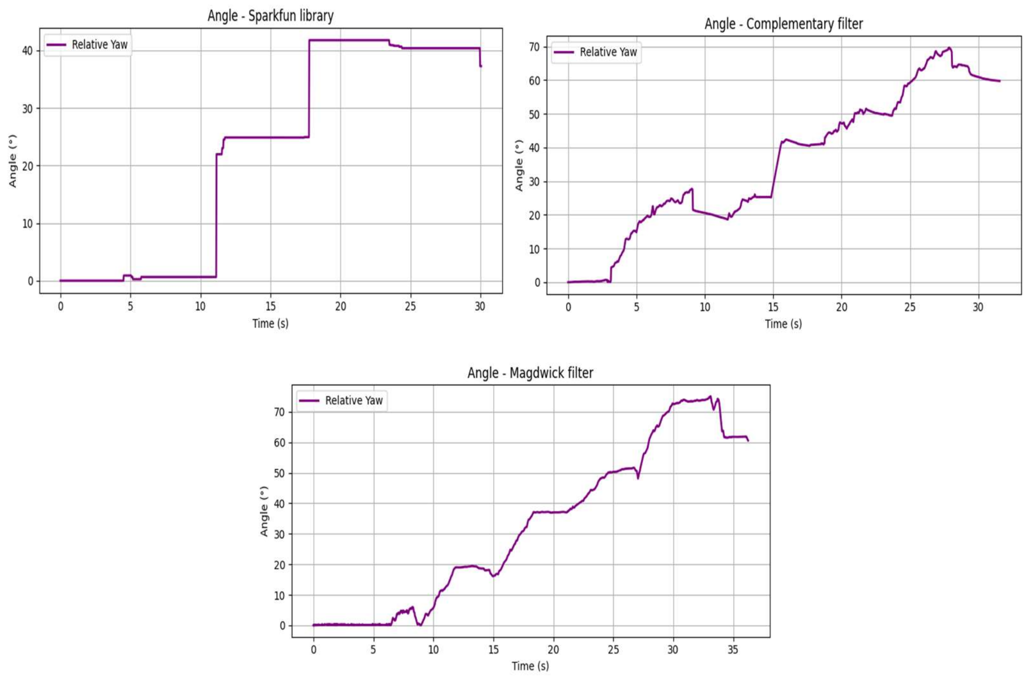

Figure 3: Angle comparison of SparkFun library, Complementary filter and Madgwick filter on flat surface experiment: 0° to 90° with 3 pauses in the middle.

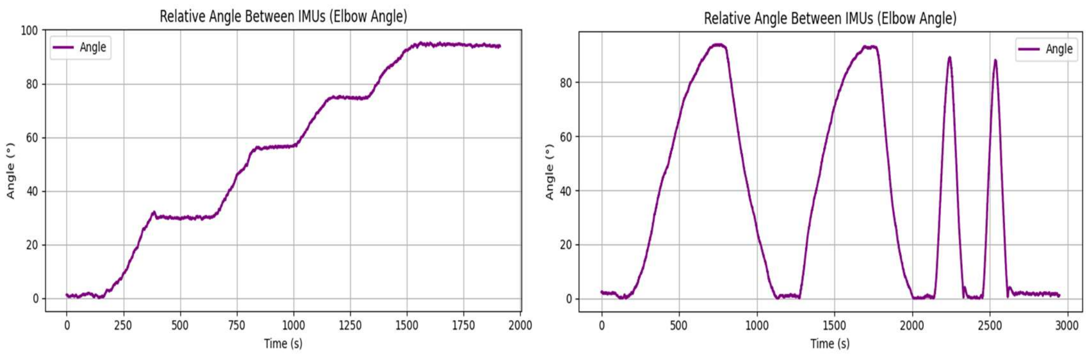

Figure 4: Angle estimation with real human arm on 2 scenarios: 0° to 90° with 3 pauses in the middle, slow and fast swing between 0° and 90°.

## Individual Analysis

My role in the team was contributing to designing a solution to the problem, and I was tasked with implementing the angle estimation node. One challenge I focused on was designing a data-collection setup that would provide accurate ground-truth angles, so I could verify that the angle estimation algorithm was producing correct results. The solution I contributed to was laying the IMU's on a flat surface and measuring the angle between the two exactly. Therefore, the real results can be compared to the angle estimator. This worked out very well and was able to verify our estimation algorithm and make any changes to improve accuracy.

After the sensor configuration was developed and decided which IMU data I was going to use to estimate the angle, I was tasked with creating the algorithm to do the angle estimation. I first began by doing research on calculating the angle with the IMU orientation data. I found simple quaternion operations would be the best way to do it. As explained in the **Angle Calculations** section, I implemented the algorithm to output the angle in degrees given the upper arm and forearm IMU orientations, rotating a common bone axis into the world frame, and computing the angle between the two segment directions. Before collecting IMU data, I created some sample quaternions to test angles of 0, 30, 45, 60, and 90 degrees. Originally, the plan was to collect IMU data, export a ROS bag, convert the data to a CSV file, then run the angle estimation algorithm on that. However, I developed a much more intuitive and practical approach. I created a ROS node, that subscribes to the two imu data topics (/imu<1, 2>/data), with a callback function that waits for the data and calculates the angle using the algorithm described above. Then, publishes the angle to a topic (/angle). This enabled the project to produce real-time angle estimation while collecting data.

When collecting data and simultaneously getting angle estimation from the node, I found that there were some inconsistencies with the accuracy of the angle. To fix this issue, I added a calibration portion to the algorithm that saves the first angle calculated and uses that as a 0-degree reference point. Then for every calculated angle after, the absolute value (to avoid negative results) was taken from the 0-degree reference point subtracted by the new calculated angle. This works with the requirement of when collecting data, the arm must be fully extended, which is a 0-degree elbow angle. This simple calibration resolved the error and provided very accurate angle estimation.

From the analysis of the data collection, I can strongly conclude that the Madgwick filter produced the best results. The Madgwick filter is a sensor-fusion algorithm that combines the IMU's gyroscope, accelerometer, and magnetometer data. As shown in the **Plots and Results** section, the bottom plot in figure 3 shows the angle starting at 0-degrees and approaching 90-degrees at the end, with pauses in between. Comparing this to the real angles observed when collecting data, the plot accurately depicts what was occurring. On the other hand, the Complementary and SparkFun filters do not accurately reflect the observed motion. Therefore, I recommend that any future data collection for this project should use the Madgwick filter to get the most accurate quaternions for the angle estimation algorithm.

In conclusion, I am not only satisfied with the accuracy of the final results, but with how the overall project was designed to reach the goals. I found that it was very important that I implemented an accurate algorithm and created a way to see real-time elbow angle estimation with use of ROS subscribers and publishers. If it was decided to go through with the original plan of converting the data to a CSV, it still would have produced the same results, but now the project has a much more practical data collection setup with more rewarding feedback. This project is something that could be extended and applied to more advanced solutions. For example, a team creating a robot that simultaneously mimics a human's arm movement. The design and algorithms created in this project can be integrated in a project like this. The team can follow the data collection setup used in this project to get accurate elbow angle estimation data, and have the robot subscribe to the angle topic. From there, the robot can mimic the human's elbow angle based off the estimation provided by this project's algorithm. This is just one example that expands the scope of the project.

## References

[1] M. El-Gohary and J. McNames, “Shoulder and elbow joint angle tracking with inertial sensors,” *IEEE Trans Biomed Eng*, vol. 59, no. 9, pp. 2635–2641, 2012, doi: 10.1109/TBME.2012.2208750.

[2] A. Bicchi and A. Colombo, “Improved Estimation of Elbow Flexion Angle from IMU Measurements Using Anatomical Constraints,” *IRBM*, vol. 45, no. 1, Feb. 2024, doi: 10.1016/j.irbm.2024.100820.

[3] J. Z. Nie, J. W. Nie, N. T. Hung, R. J. Cotton, and M. W. Slutzky, “Portable, open-source solutions for estimating wrist position during reaching in people with stroke,” *Sci Rep*, vol. 11, no. 1, Dec. 2021, doi: 10.1038/s41598-021-01805-2.