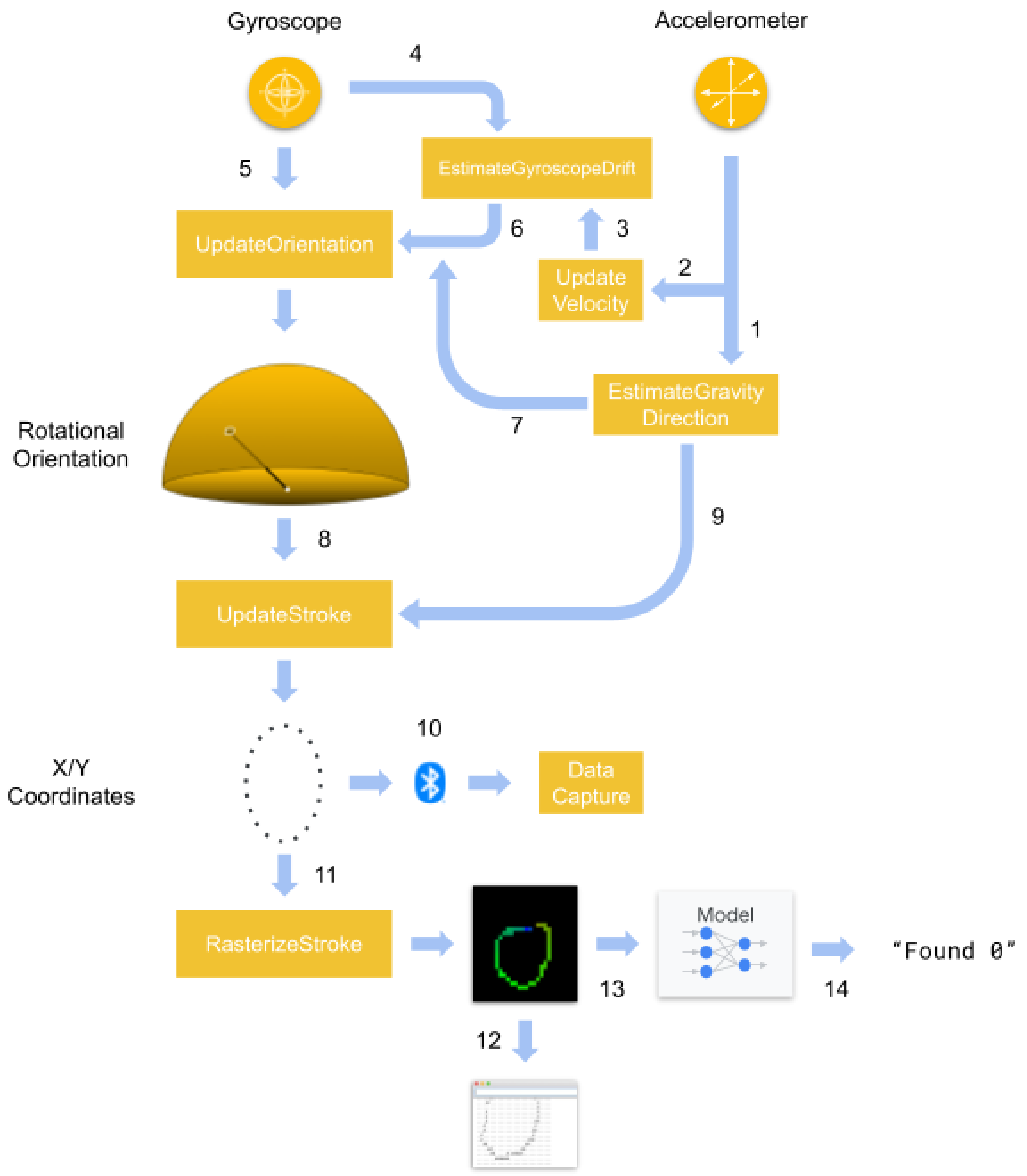
Understanding the Magic Wand Application

This reading is about understanding the components of the magic wand application and how they generalize to other IMU-based projects. By gaining familiarity with the data flow, from sensor to classification, you can apply similar concepts to other applications.



The Magic Wand application accomplishes a fairly complicated task (gesture recognition) by carefully crafting a 2D image from 3D IMU data. The dataflow displayed in the diagram above consists of 14 steps:

1. The accelerometer data is read and passed to the `EstimateGravityDirection()` where it is used to **determine the orientation** of the Arduino with respect to the ground.

2. The Accelerometer data is passed to `UpdateVelocity()` where it is used to **calculate the velocity** of the Arduino.

3. The direction of gravity is passed to `UpdateVelocity()` and is used to c**ancel out the acceleration due to gravity** from the accelerometer data.

4. The velocity is then passed to `EstimateGyroscopeDrift()` where it is used to **determine if the Arduino is stationary or moving**.

5. The gyroscope data is passed to `EstimateGyroscopeDrift()` where it is used to **calculate the sensor drift of the gyroscope** if the Arduino is not moving (velocity is 0).

6. The gyroscope data is passed to `UpdateOrientation()` where it is integrated to **determine the angular orientation** of the Arduino.

7. The gyroscope drift is also passed to `UpdateOrientation()` and subtracted from the gyroscope reading to **cancel out the drift.**

8. The 3D angular orientation is passed to `UpdateStroke()` transformed into 2d positional coordinates. `UpdateStroke()` also handles whether the **current gesture has ended or if a new gesture has been started** by analyzing both the length of the gesture and testing whether the orientation data is still changing.

9. The direction of gravity is also passed to `UpdateStroke()` to **determine the roll orientation** of the Arduino.

10. The 2d positional coordinates are sent over BlueTooth to the **data collection** application in the browser.

11. The 2d positional coordinates are passed to `RasterizeStroke()` which takes the 2D coordinates and **draws lines between them on a 2D image**. The color of the lines shifts from red to green to blue to indicate the direction of motion during the gesture.

12. The 2D image of the gesture is converted to ASCII art and **printed to the serial monitor**.

13. The 2D image of the gesture is **passed to the model**.

14. The model **predicts the label of the gesture** and the label is printed to the serial monitor.

For a deeper understanding of the Magic wand application, try following the flow diagram while reading the code. The comments should provide the context needed to understand the function of each section. Try to piece the application together, one piece at a time, as there is a lot going on in this application. That said, each piece by itself is fairly self-contained.