Мой первый околонаучный проект. Инерциальный трекер

Даниил Барков

Осень 2022

Оглавление

Введение **2**

Описание **3**
- Цели и задачи . . . 3
- Электронная начинка . . . 4
- Алгоритм работы и программная реализация . . . 4
  - Термины и определения . . . 4
  - Математика . . . 5
  - Прототип . . . 5
  - Программа для ПК . . . 6
  - Программа для МК . . . 6
  - Матричные операции . . . 6
  - Типа операционная система реального времени . . . 6
- Проблемы . . . 6
  - Корявые вычисления с плавающей точкой . . . 6
  - Частота измерений датчика и интерполяция . . . 6
  - Шумы и фильтрация . . . 6
  - Калибровка . . . 7

Планы на будущее **8**

Введение

Не буду рассказывать, когда и почему я захотел написать программу для работы с инерциальным сенсором, а начну с того момента, когда я начал изучать линейную алгебру и MATLAB.

Мне открылись новые возможноти и методики, и создать прототип программы удалось очень быстро. А ещё я понял, что стремлюсь к научному познанию, и через эту работу я могу получать результаты.

Текст поможет структурировать мысли и сохранить математические выкладки. Однако стили оформления и содержания будут не совпадать. Это потому, что я хочу заодно научиться техать. Вопреки желанию буду писать единицы измерения на английском, т.к. модуль записи математики не понимает кириллицу.

Описание

Сечас я делаю такую электронную штучку с набором датчиков, которая умеет измерять ускорения, угловые скорости, магнитные поля, и давление. То есть, 10 степеней свободы. Она умеет обрабатывть данные с датчика и выдавать набор чисел, которые отправляются на компбютер по USB. Там их считывает программа, написанная на матлабе и выдаёт такую картинку. При запуске программа определяте свой наклон при помощи акселерометра, затем хитрым образом суммирует повороты по осям, компенсирует дрейф гироскопов с помощью акселерометров и получает набор векторов x, y, z. Ещё оно умеет определять азимут, но сейчас я отключил эту функцию, тк магнетометр приходится калибровать, об этом расскажу в конце.

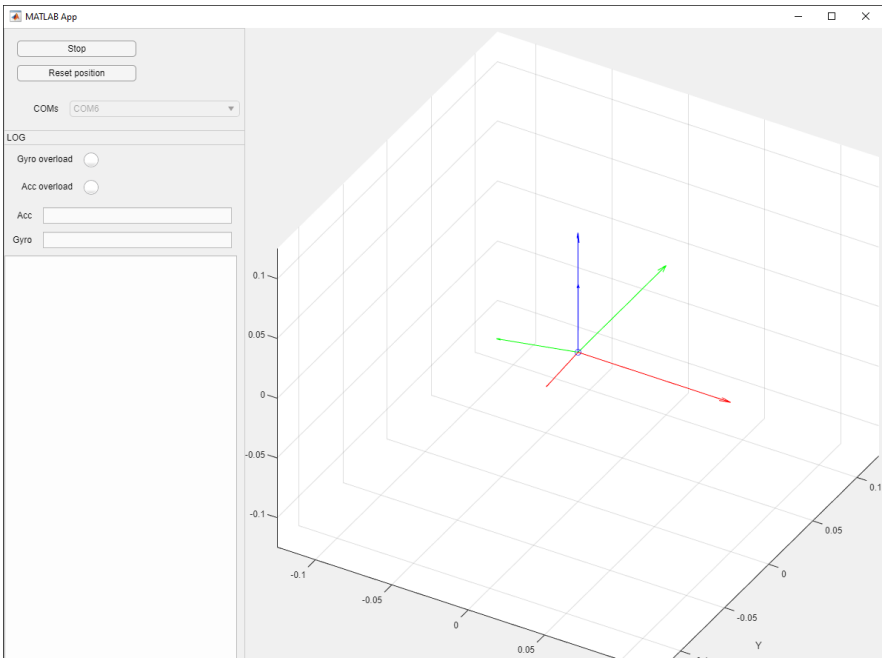

Рис. 1: Моё первое десктоп приложение

Цели и задачи

Пока что план такой - написать библиотеку, функции которой будут принимать данные с датчика, измерять время между приёмами и возвращать необходимые данные:

- трёхмерный базис, векторы которого сонаправлены с датчиком ($B_{3\times3}$)

- скорость относительно земли ($\overrightarrow{V}$)
- ускорение относительно земли ($\overrightarrow{A}$) и относительно датчика ($\overrightarrow{A'}$)
- азимут (θ)
- высота над уровнем моря или хотя бы над местом старта (alt)
- время между измерениями акселерометра и гироскопа (τ), магнетометра (τ_m) и барометра (τ_{alt})
- угловые скорости ($\varphi_x,\ \varphi_y,\ \varphi_z$, то есть $\overrightarrow{\phi}$)

$B_{3\times3}$ как бы отражает наклон датчика и ориентацию относительно севера. Он не должен дрейфовать со временем и наклоняться из-за линейного ускорения.

Затем подготовлю шаблон для настройки микроконтроллера (MK), сделать библиотеку для работы с конкретной моделью датчика.

И в конце концов выполню это всё в железе. Очев, начал я с конца, т.к. без устройства было бы тяжело проводить тесты.

Электронная начинка

Главное устройство тут - это модуль 10 DOF IMU sensor (B) от Waveshare, включающий в себя

- 3-х осевой акселерометр (до $\pm156\ m/s^2$)
- 3-х осевой гироскоп (до $\pm2000\ ^\circ/s$)
- 3-х осевой магнетометр (до $\pm4800\mu T$)
- барометр ($300 \sim 1100\ hPa$, т.е $+9000 \sim -500\ m$)
- термометр (даже)

Считыванием и обработкой данных занимается МК STM32F103C8T6, в народе Blue Pill. Работает на 72 MHz и не имеет ускорителя вычислений с плавающей точкой (FPU), поэтому считает относительно медленно. Он собирает по I2C данные с акселерометра и гироскопа каждые $\tau = 4500\mu s$, а магнетометра - $\tau_m = 15000\mu s$. С барометром я пока не работал. Результаты вычислений отправляются на комп через USB-UART конвертер, где их ловит матлаб и выводит их на экран в виде векторов и фонариков перегрузки.

Алгоритм работы и программная реализация

Термины и определения

Перед описанием алгоритма расскажу как и что я назвал. Как рассказал выше, датчик выдаёт такой набор величин:

- ускорение относительно датчика ($\overrightarrow{A'}$)
- давление (P)
- угловые скорости $\overrightarrow{\phi}$
- магнитное поле $\overrightarrow{M'}$

Также измеряются $\tau,\ \tau_m\ \tau_{alt}$.
На аслерометр всегда (если он не падает) действует сила тяжести, которую он также регистрирует своими внутренними грузиками - $\overrightarrow{G'}$. По-хорошему он должен иметь значение $\overrightarrow{G'} = (0, 0, -9.8)$, но когд датчик правильно понимает совою ориентацию. Так что оригинальную силу тяжести назовут отдельно - $\overrightarrow{G}$.
Главный термин здесь - базис $B_{3\times3}$. Под ним я подразумеваю три ортонормированных вектора датчика, как-то ориентированных в пространстве. Например, $\overrightarrow{z}$ направлен вверх, $\overrightarrow{x}$ - на север. Получится единичная матрица матрица. Если $\overrightarrow{x}$ смотрит на запад, получится так: $\begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$.
Также часто будут использоваться матрицы поворота:

вокруг $\overrightarrow{z}$: $rotz(\varphi) = \begin{pmatrix} cos(\varphi) & sin(\varphi) & 0 \\ -sin(\varphi) & cos(\varphi) & 0 \\ 0 & 0 & 1 \end{pmatrix}$

вокруг $\overrightarrow{x}$: $rotz(\varphi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\varphi) & sin(\varphi) \\ 0 & -sin(\varphi) & cos(\varphi) \end{pmatrix}$

вокруг $\overrightarrow{y}$: $rotz(\varphi) = \begin{pmatrix} cos(\varphi) & 0 & -sin(\varphi) \\ 0 & 1 & 0 \\ sin(\varphi) & 0 & cos(\varphi) \end{pmatrix}$

вокруг произвольного $\overrightarrow{l}$: $rotl(\varphi) = \begin{pmatrix} c + (1-c)l_x^2 & (1-c)l_xl_y - sl_z & (1-c)l_zl_x + sl_y \\ (1-c)l_xl_y + sl_z & c + (1-c)l_y^2 & (1-c)l_zl_y - slx \\ (1-c)l_xl_z - sl_y & (1-c)l_yl_z + sl_x & c + (1-c)l_z^2 \end{pmatrix}$,

где $c = cos(\varphi),\ s = sin(\varphi),\ \overrightarrow{l} = (lx, ly, lz),\ |\overrightarrow{l}| = 1$

Математика

Расскажу о алгоритме, математическич формулах, физических явлениях

Прототип

Кратко расскажу о том, как я написал прототип в матлабе, как он работал, и почему этого было недостаточно
Изначально все чудеса происходили в матлабе. Мобильное приложение собирало показания с датчиков в массив, сохраняло в облачный файл, который затем открывался в матлабовском скрипте. В нём я прототипировал алгоритм, что было очень удобно.

Программа для ПК

Сейчас приложение лишь принимает 10 байт, 9 из которых - это координаты трёх векторов, помноженные на 100 (векторы в длину меньше 1, а передавать single precision вчетверо дольше), а последний хранит 2 бита для индикаторов перегруза датчкиов.

Хорошо бы сделать проверку целостности, перезапуск общения, отправку команд на МК. Но сходу не получилось, а необходимости в этом нет.

Программа для МК

Расскажу о том, как происходит общение с датчиком, о структуре программы и о том, как перенёс матлабовский код в СИ. О скорости работы и о том, что можно улучшить

Матричные операции

Тут я расскажу, о том, как сделал функции удобные, как в матлабе, но они не завелись на МК, и пришлось доделывать.

Типа операционная система реального времени

Моё крутое испрользование micros(), прерываний и отсутствие делэев.

Проблемы

Расскажу о проблемах, связанных с работой датчика, МК и алгоритма

Корявые вычисления с плавающей точкой

Здесь я расскажу о том, то я много вычилсяю тригонометрические функции, которые по природе неидеальны.
О том, что вещественные числа в формате float подразумевают накопление ошибки.
И о том, что я хочу вычислять определитель базиса, чтобы проверть его ортонормированность. Ну и как-то чинить базис в дальнейшем.

Частота измерений датчика и интерполяция

Подумаю, какую интерполяцию использовать, и почему. (до прямых линий между точками)

Шумы и фильтрация

Построю гистограмму шумов покоящихся датчиков на разных диапазонах измерений. Подумаю, нужно ли использовать фильтр. Чего это будет стоить (скорость и память). Подберу варианты.

Калибровка

Объясню необходимость калибровки, опишу текущий метод, расскажу, почему хочу поменять.

Планы на будущее

Расскажу о том, что хочу улучшить, и куда добавить придуманный алгоритм