Virtual controller

Имеется: программа на питоне на компьютере, к компу подключен мощный радиопередатчик, соединенный с аналогичным на гусеничной тележке. Телега имеет ряд датчиков, которые агрегирует и посылает на компьютер. Комп в ответ говорит скорости гусениц в каждый момент времени.

Формат сообщений

Общий формат сообщений:

- заголовок - 0x7E
- тип - 1 байт
- длинна - 1 байт
- полезная нагрузка - 0-255 байт
- crc32 - 4 байта

Небольшая договоренность в типах пакетов:

Пакеты со стартовым байтом

- A - administration пакеты с настройками системы.
- B - команды на взаимодействие (запрос ответа)
- C - control пакеты с параметрами движения.
- D - данные от датчиков.
- E - пакеты с кодами ошибки.
- F - ответы на запросы
  Второй байт типа говорит о номере сообщения такого типа. Итоговые типы должны быть уникальны, т.е. тип пакета 0xD0, говорящий о данных с IMU, всегда должен нести эти данные, откуда бы он не пришел.

Сообщение от телеги

- Тип - 0xD0
- Payload - all IMU:
  - в формате uint32_t - временная метка пакета в миллисекундах
  - в формате float - X Y Z, акселерометр, магнитометр и гироскоп.
- Длинна - 4 *3* 3 + 4 = 40 байт
- Тип - 0xD1
- Payload - taho data:
  - в формате uint32_t - временная метка пакета в миллисекундах
  - в формате int32_t левая и правая гусеницы rpm,
- Длинна - 4 + 2 * 4 = 12 байт
- Тип - 0xD2
- Payload - motor data:
  - в формате uint32_t - временная метка пакета в миллисекундах
  - в формате int16_t ток левой и правой гусеницы
  - в формате int16_t напряжение левой и правой гусеницы
  - в формате int16_t температура левой и правой гусеницы
- Длинна - 4 + 2 *2 + 2* 2+ 2 * 2 = 16 байт
- Тип - 0xF0 - ответ на запрос синхронизации
- Payload
  - в формате uint32_t - время приема сообщения (максимально быстро и близко к моменту приема 0xB0)
  - в формате uint32_t - время отправки ответа, максимально близко к отправке сообщения
- Длинна = 8

Сообщения от компьютера:

- Тип - 0xC0
- Payload - скважность шима на левую и правую гусеницы
  - в формате uint32_t - временная метка пакета в миллисекундах
  - в формате int16_t на левую и правую гусеницы
  - в формате uint16_t сколько миллисекунд действительна команда
- Длинна = 10
- Тип 0xB0 - Запрос на синхронизацию времени
- Payload
  - в формате uint16_t - номер раунда
  - в формате uint32_t - время компа
- Длинна - 6 байт
- Тип 0xA0 - выключить отправку сообщений под типом....
- Payload
  - в формате uint32_t - временная метка пакета в миллисекундах
  - тип D сообщения, которое надо отключить (uint8_t)
- Длинна - 5
- Тип 0xA1 - включить сообщения с периодом
- Payload
  - в формате uint32_t - временная метка пакета в миллисекундах

- тип D сообщения, которое надо включить (uint8_t)
- в формате uint16_t - период отправки сообщения

Протокол синхронизации времени PC↔MCU (NTP-like, оценка a и b на стороне PC)

0. Обозначения

- PC ведёт монотонное время `t_pc` (uint64, например мкс).
- MCU ведёт монотонное время `t_mcu` (uint64, те же единицы, что и на PC, или любые — главное постоянный масштаб).
- Модель соответствия времени:
  - `t_mcu ≈ a * t_pc + b`
  - `a` — масштаб (1 + skew), `b` — смещение (offset).
- Все таймстампы берутся максимально близко к событию:
  - `t1_pc` — при формировании SYNC_REQ на PC
  - `t2_mcu` — при приёме SYNC_REQ на MCU (RX обработчик)
  - `t3_mcu` — при формировании SYNC_RESP на MCU
  - `t4_pc` — при приёме SYNC_RESP на PC (RX обработчик)

---

1. Форматы пакетов

1.1 SYNC_REQ (PC → MCU)

Поля:

- `type = SYNC_REQ` (u8)
- `seq` (u16/u32) — номер раунда
- `t1_pc` (u64) — время PC в момент формирования этого пакета

1.2 SYNC_RESP (MCU → PC)

Поля:

- `type = SYNC_RESP` (u8)
- `seq` (u16/u32) — эхо номера раунда
- `t2_mcu` (u64) — время MCU в момент приёма SYNC_REQ
- `t3_mcu` (u64) — время MCU в момент формирования SYNC_RESP

1.3 Рабочий пакет MCU → PC (DATA, необязательный формат)

Поля (минимум для привязки времени):

- `type = DATA` (u8)
- `ts_mcu` (u64) — таймстамп события/формирования на MCU (что именно — фиксируете в протоколе прикладного уровня)

---

2. Расчёт одной синхронизационной точки на PC (для раунда i)

После получения SYNC_RESP{seq=i, t2_mcu, t3_mcu}:

- PC фиксирует `t4_pc = now_pc()` (при приёме)
- `t1_pc` берётся из ранее отправленного SYNC_REQ с тем же seq

Вычисляются:

2.1 Оценка качества (задержка канала)

`delta_i` (используется только для фильтрации “плохих” раундов):

- `delta_i = (t4_pc - t1_pc) - (t3_mcu - t2_mcu)`

2.2 Mid-пара (реперная точка соответствия времен)

- `tmid_pc_i = (t1_pc + t4_pc) / 2`
- `tmid_mcu_i = (t2_mcu + t3_mcu) / 2`

Интерпретация: в момент `tmid_pc_i` по PC на MCU было примерно `tmid_mcu_i`.

---

3. Процедура первой инициализации (стартовая синхронизация)

Цель: получить первичные `a_hat`, `b_hat`.

1. PC выполняет N раундов SYNC (рекомендуемо N=10…30):
   - для i=0..N-1:
     - PC: сформировать SYNC_REQ, записать `t1_pc`, отправить
     - MCU: при приёме зафиксировать `t2_mcu`; сформировать ответ с `t3_mcu`; отправить SYNC_RESP
     - PC: при приёме зафиксировать `t4_pc`, посчитать `delta_i`, `tmid_pc_i`, `tmid_mcu_i`, сохранить запись
2. PC выбирает K лучших записей по минимальному `delta_i` (рекомендуемо K=5, K<=N).
3. Оценка `a_hat`, `b_hat` по двум крайним точкам среди этих K (максимально простой расчёт):

- выбрать среди K точек самую раннюю и самую позднюю по `tmid_pc`:
  - `(tmid_pc_first, tmid_mcu_first)`
  - `(tmid_pc_last, tmid_mcu_last)`
- вычислить:
  - `a_hat = (tmid_mcu_last - tmid_mcu_first) / (tmid_pc_last - tmid_pc_first)`
  - `b_hat = tmid_mcu_first - a_hat * tmid_pc_first`

4. С этого момента PC считает, что время MCU задаётся моделью `t_mcu ≈ a_hat * t_pc + b_hat`.

---

4. Процедура последующей инициализации (подстройка во времени)

Цель: не дать модели “уплыть” из-за изменения дрейфа (температура/режимы/питание).

Периодически (например, раз в 10 секунд) выполнить один раунд SYNC и сгладить модель:

1. Выполнить 1 раунд SYNC, получить `(delta, tmid_pc, tmid_mcu)`.
2. (Опционально) отбросить раунд, если `delta` слишком велик (локальный порог или относительно недавних минимумов).
3. Обновить `a_hat` и `b_hat` по последней принятой “хорошей” точке и предыдущей опорной точке:
   - пусть предыдущая опорная точка: `(tmid_pc_prev, tmid_mcu_prev)`
   - текущая: `(tmid_pc_cur, tmid_mcu_cur)`
   - сырые оценки:
     - `a_raw = (tmid_mcu_cur - tmid_mcu_prev) / (tmid_pc_cur - tmid_pc_prev)`
     - `b_raw = tmid_mcu_cur - a_raw * tmid_pc_cur`
   - сглаживание (0<beta<=1, например 0.02…0.1):
     - `a_hat = (1 - beta) * a_hat + beta * a_raw`
     - `b_hat = (1 - beta) * b_hat + beta * b_raw`
   - обновить опорную точку: `prev = cur`

---

5. Формулы использования на PC (после синхронизации)

5.1 Оценка текущего времени MCU на PC

Для текущего времени PC `t_pc_now = now_pc()`:

- `t_mcu_est = a_hat * t_pc_now + b_hat`

5.2 Перевод таймстампа из пакета MCU → PC в шкалу времени PC

Если MCU прислал `ts_mcu` (u64) в DATA-пакете:

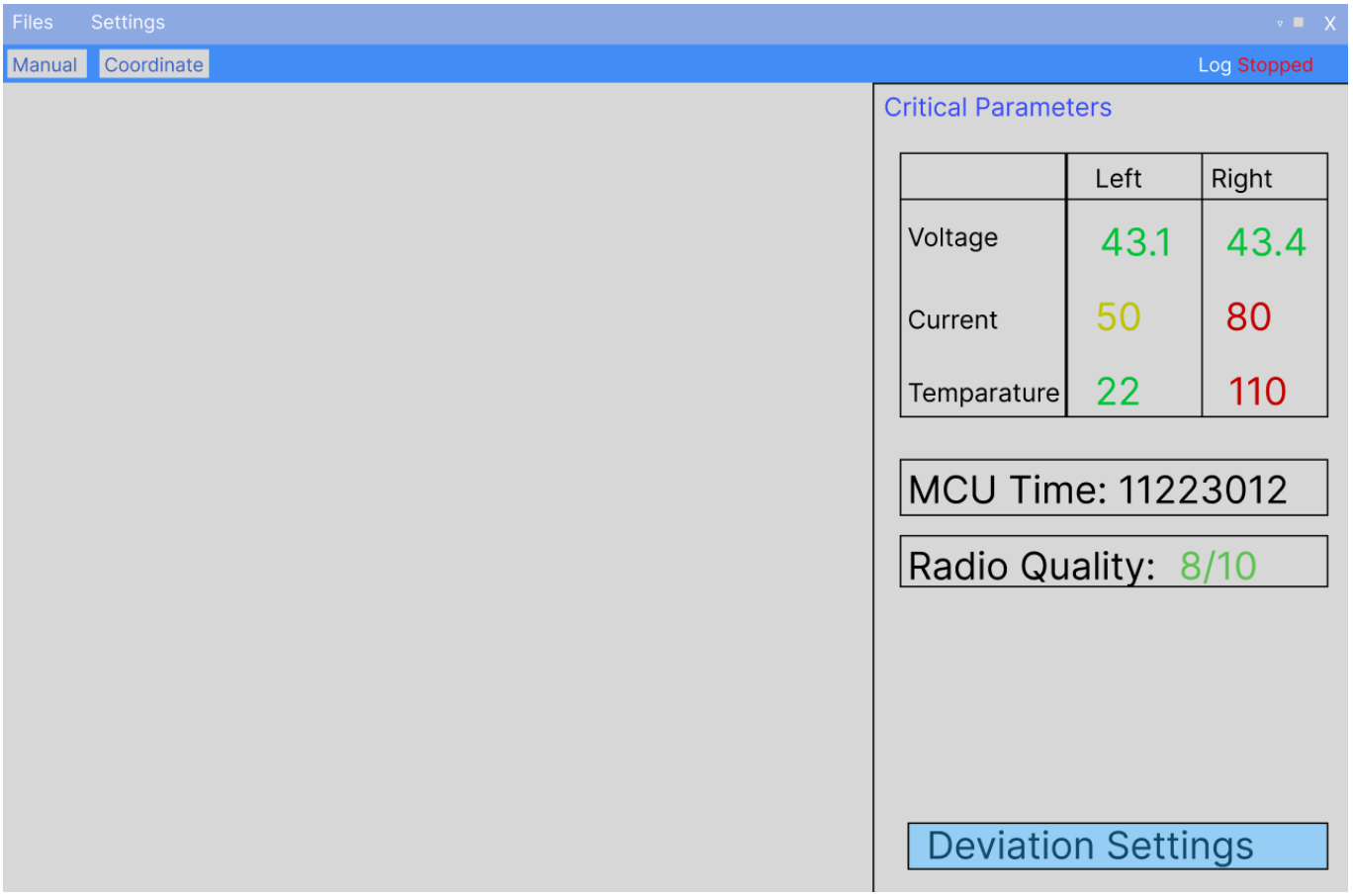
- `t_pc_est = (ts_mcu - b_hat) / a_hat`

(Дальше `t_pc_est` можно использовать как “оценку времени события по шкале PC”.)

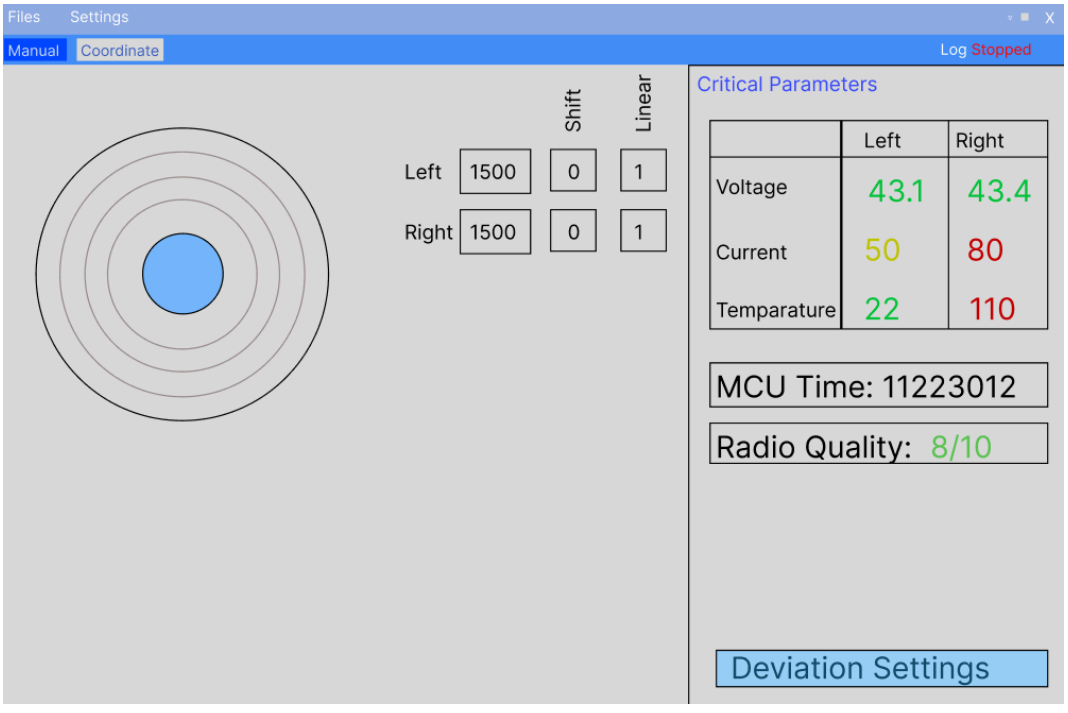
Общая концепция приложения

Кросплатформенное (Linux и Windows) графическое приложение на python, работающее с UART. Соответственно должна быть возможность задания настроек ком порта (выбор самого порта и скорости общения). Также должна быть возможность ведения и сохранения логов, полученных из ком порта. Сохраняться должны уже распарсенные данные в формате .log .

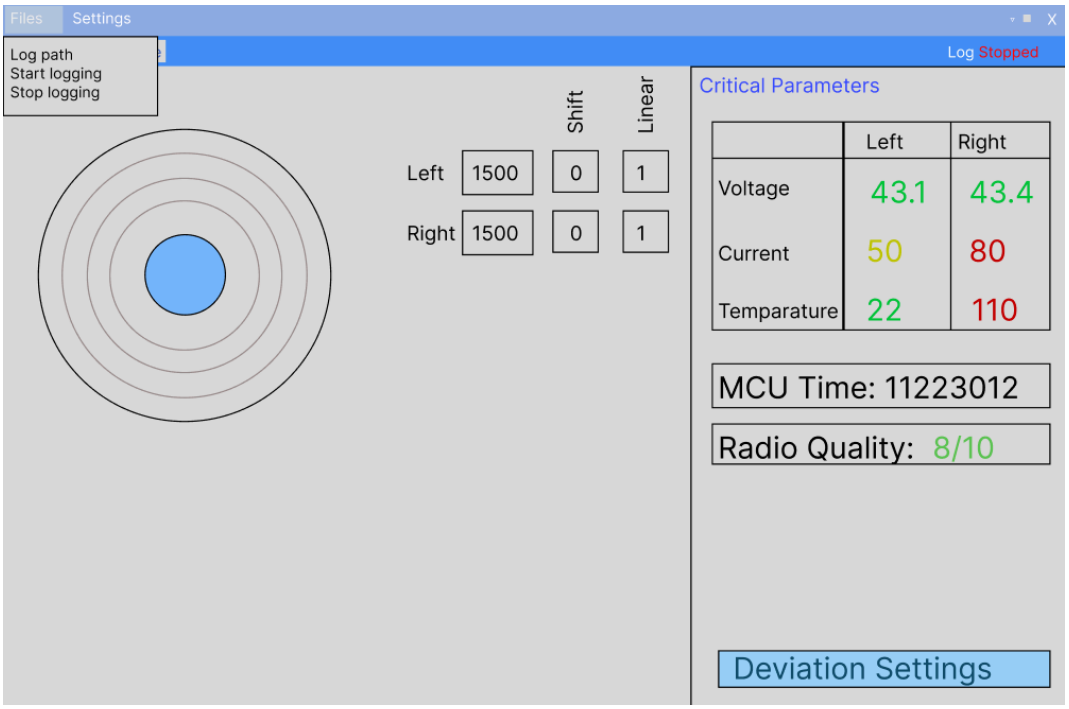
Есть несколько отдельных окон/вкладок с разными методами управления и реализуемой математикой.

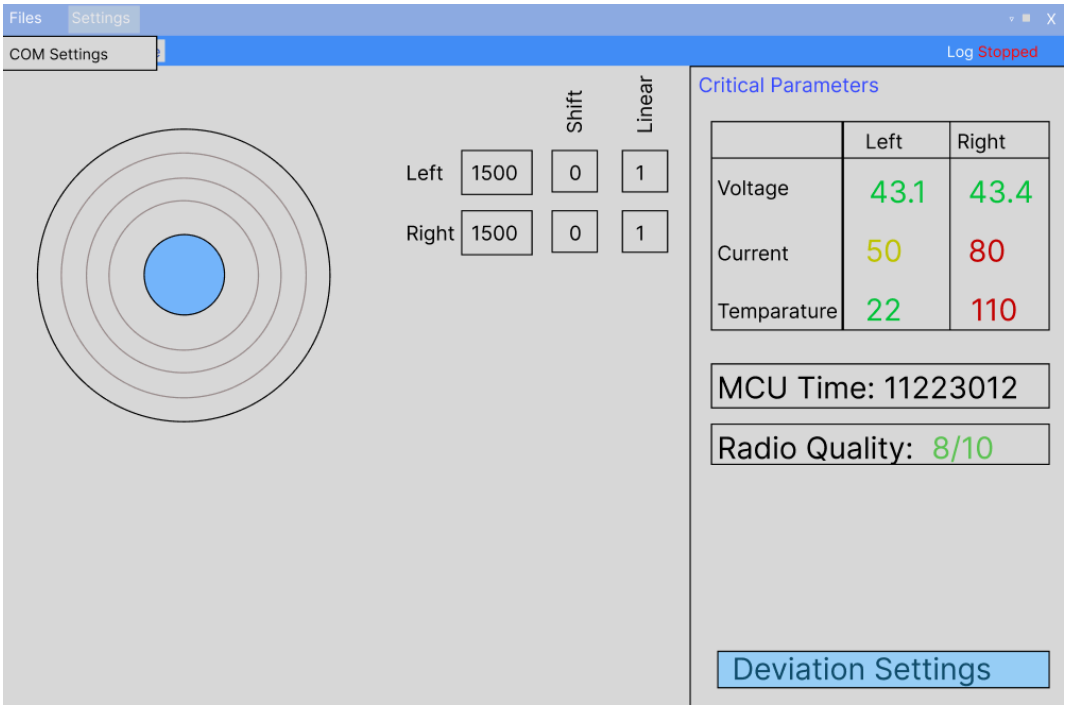


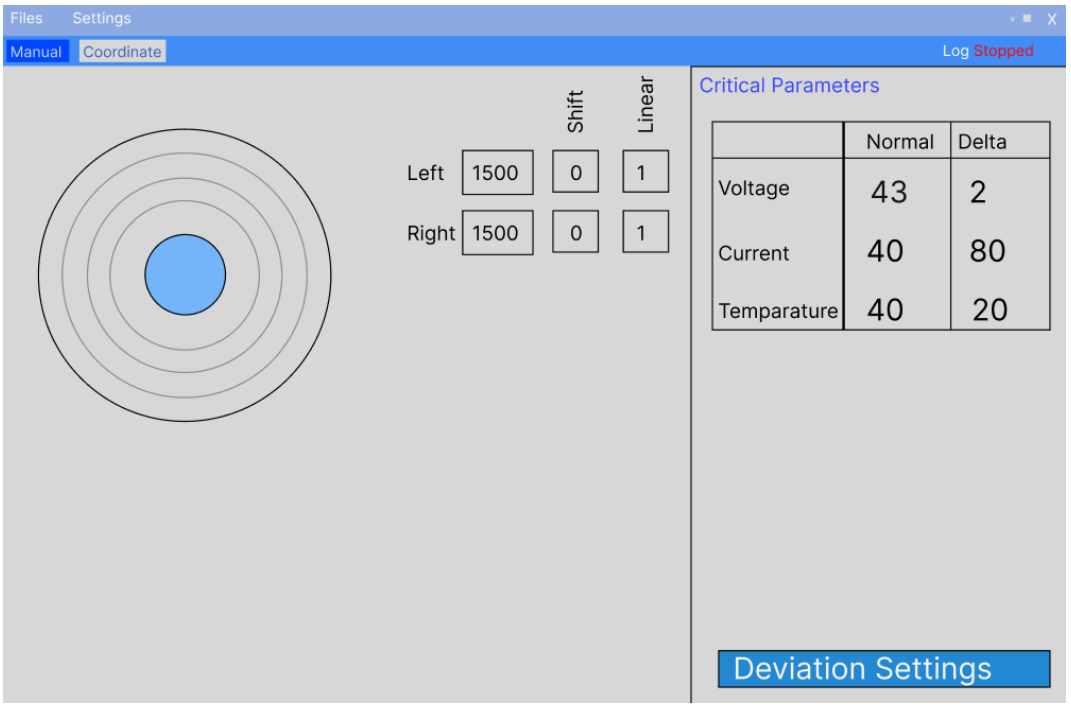
Можно переключаться между режимами, пока будем делать только Manual



Так же настройки на кнопках files, settings и Deviation Settings


Files
Settings
Log path
Start logging
Stop logging
Log Stopped
Shift
Linear
Left 1500 0 1
Right 1500 0 1
Critical Parameters
Left
Right
Voltage 43.1 43.4
Current 50 80
Temparature 22 110
MCU Time: 11223012
Radio Quality: 8/10
Deviation Settings


Files
Settings
COM Settings
Log Stopped
Shift
Linear
Left 1500 0 1
Right 1500 0 1
Critical Parameters
Left
Right
Voltage 43.1 43.4
Current 50 80
Temparature 22 110
MCU Time: 11223012
Radio Quality: 8/10
Deviation Settings



Виртуальный джойстик

На всех рисунках кроме первого есть джойстик, который можно двигать мышкой. Он задает параметры для Left и Right. Shift и Linear это коэффициенты коррекции, задаваемые вручную отдельно (просто окно для ввода числа). Цветные значения в критикал параметрс это значения, полученные от микроконтроллера по радио каналу. Зеленые, если значение находиться в нормал +- 0.5 дельта, желтое если больше +- 0.5 и до +- дельта, красное если больше дельта. Нормаль и дельта задаются в окне Deviation Settings для каждого параметра отдельно.

Логирование включается через вкладку файл. Если логгирование идет, то должна появиться надпись Log Running зеленым, вместо Log Stopped красным.

Задачи

Требуется реализовать считывание данных из ком порта, обработку всех описанных выше сообщений и алгоритма синхронизации времени. Реализовать графическое приложение по приложенным референсам. Весь код должен быть модульным, разбитым по нескольким файлам. Математику расчета итоговых значений вывести в отдельные функции для легкой замены.

Во время работы над кодом надо вести чейнджлог, список того, что было сделано, использованный фреймворк и идеи. Весь код тщательно комментировать и пояснять смысл и работу каждой функции.