Система управления оптоволоконным спектрографом 1-м телескопа

Емельянов Эдуард Владимирович

2021-10-25

1 Описание

Основой системы управления прибором является мини-компьютер под управлением ОС Gentoo Linux, расположенного на подвесной части. В подвесной части расположены три шаговых двигателя (управляются по CAN-шине): управление позиционированием волокна и подфокусировкой, а также камера подсмотра волокна. В стационарной части размещены система калибровочной засветки (может управляться как отдельно по USB, так и с общего интерфейса по CAN-шине), управление кросс-дисперсором и фокусировкой камеры.

Система управления состоит из трех демонов: `canserver` – работа с устройствами на CAN-шине через USB-CAN переходник, `loccorr` – анализ изображений и управление процессом коррекции положения объекта на волокне и `spec_server` – сервер веб-интерфейса системы управления.

2 Базовые демоны системы управления

2.1 Демон `loccorr`

2.1.1 Описание

Данный демон[1] предназначен для анализа изображений, определения координат центроидов и выдачи управляющих сигналов на корректирующую аппаратуру.

В качестве входных изображений могут быть файлы формата FITS, JPEG или PNG (посредством `inotify` производится мониторинг обновления одного файла либо появления новых файлов в указанной директории), CMOS-светоприемники Basler или Grasshopper. Возможно добавление других источников, для этого необходимо в директории проекта создать заголовочный файл и файл исходных текстов наподобие `basler.h` и `basler.c`, где будут реализованы обработчики функций структуры `camera` (файл `cameracapture.c`):

disconnect отключить камеру;

connect подключить камеру и произвести базовые настройки;

capture захватить изображение (возвращается указатель на структуру `Image`, выделенную в этой функции);

[1] `https://github.com/eddyem/astrovideoguide_v3/tree/main/LocCorr`

setbrightness установить настройку «яркость» (если таковая имеется);

setext установить экспозицию (в миллисекундах);

setgain установить усиление (в дБ, если таковое имеется);

getmaxgain получить предельное значение усиления (если таковое имеется);

setgeometry установить геометрию кадра (ширина, высота, смещение по X и Y);

getgeomlimits получить предельные значения геометрии и шаг изменения.

Алгоритм обработки изображений следующий. После считывания очередного изображения опционально выполняется его медианная фильтрация. Далее строится гистограмма, по которой (в точке перегиба после первого максимума) определяется уровень фона. Изображение бинаризуется по найденному фону. Производится заданное количество эрозий и дилатаций для очистки шума. Находятся и нумеруются 4-связные области. Каждая область проверяется на соотношение сторон описывающего прямоугольника и занимаемую площадь (они должны лежать в заданных рамках). Если после этого все еще обнаружено больше одного объекта, объекты сортируются по яркости либо близости к координатам оптического волокна. После обработки определенного количества изображений выполняется вычисление средних координат центроида звезды. Если среднеквадратичное отклонение не превышает пяти пикселей, определен модуль коррекции и его состояние позволяет выполнять коррекцию, посылаются команды коррекции положения.

Работа модуля коррекции (на основе контроллеров шаговых двигателей от фирмы PusiRobo) описана в `pusirobo.c`. Принимаемые команды передаются через открытый сокет демону `canserver` (см. стр. 9, п. 2.2). Возможно подключение любого другого модуля коррекции, для этого необходимо реализовать описанные в `improc.h` поля структуры `stepperspproc`:

proc_corr выполнить коррекцию положения звезды на заданное количество пикселей;

stepstatus получить состояние шаговых двигателей;

setstepstatus установить новое состояние;

movefocus переместить фокусер на заданное количество шагов;

moveByU переместить корректор на заданное количество шагов по оси U;

moveByV переместить корректор на заданное количество шагов по оси V;

relay выполнить команду на стационарной части (включить/выключить реле, установить значение ШИМ, считать состояние кнопок и т.п.);

stepdisconnect отключиться от устройства коррекции.

2.1.2 Аргументы командной строки демона

-A, –maxarea=arg максимальная площадь (в пикселях) объекта (по умолчанию 150000);

-C, –canport=arg порт локального сервера `canserver` (по умолчанию 4444);

-D, –ndilat=arg количество дилатаций при обработке изображения (по умолчанию 2);

-E, –neros=arg количество эрозий при обработке изображения (по умолчанию 2);

-H, –height=arg высота рабочего участка изображения;

-I, –minarea=arg минимальная площадь (в пикселях) объекта (по умолчанию 400);

-L, –logXY=arg файл для логгирования вычисленных координат центроидов;

-N, –naverage=arg количество изображений для вычисления средних координат центроида (от 1 до 25);

-P, –pidfile=arg файл с PID сервера (по умолчанию `/tmp/loccorr.pid`);

-T, –intthres=arg порог яркости изображений при сортировке ($(I_1 - I_2)/(I_1 + I_2)$, по умолчанию 0.01)

-W, –width=arg ширина рабочего участка изображения;

-X, –xtarget=arg координата X цели (оптоволокна, щели);

-Y, –ytarget=arg координата Y цели;

-b, –blackp=arg доля пикселей с низкой интенсивностью, которая будет отброшена при эквализации гистограммы (если включена эквализация обработанного кадра);

-c, –confname=arg имя файла конфигурации (по умолчанию `loccorr.conf`);

-e, –equalize выполнять эквализацию обработанного кадра;

-h, –help вызвать данную справку;

-i, –input=arg название объекта для мониторинга новых изображений (имя файла или директории, «grasshopper» или «basler» при захвате с КМОП-камеры);

-j, –jpegout=arg название файла, куда будет записано обработанное изображение (по умолчанию `./outpWcrosses.jpg`);

-l, –logfile=arg файл, в который будет вестись логгирование;

-p, –proc=arg имя модуля процессинга коррекций («pusirobo»);

-v, –verbose повысить уровень информативности логгирования (каждый `-v` повышает на 1);

-x, –xoff=arg сдвиг по оси X рабочего участка изображения;

-y, –yoff=arg сдвиг по оси Y рабочего участка изображения;

–**ioport=arg** номер порта сокета для подключения управления (по умолчанию 12345);

–**maxexp=arg** максимальная экспозиция (в миллисекундах, по умолчанию 500);

–**minexp=arg** минимальная экспозиция (в миллисекундах, по умолчанию 0.001).

2.1.3 Конфигурационные параметры

В файле конфигурации хранятся базовые параметры настроек демона. В случае, если в аргументах командной строки введены другие значения, нежели в настройках, преимущество будет за первыми. В случае отсутствия и конфигурационного файла, и аргументов, будут использоваться значения по умолчанию. При завершении работы в конфигурационный файл сохраняются текущие значения (в т.ч. полученные в результате указания пользователем во время работы).

Конфигурационный файл может иметь следующую структуру (после поля # указан комментарий):

```
maxarea = 10000     # максимальная площадь объекта
minarea = 100       # минимальная площадь объекта
minwh = 0.800       # минимальное отношение ширины к высоте объекта
maxwh = 1.300       # максимальное отношение ширины к высоте объекта
ndilat = 3          # количество дилатаций
neros = 3           # количество эрозий
xoffset = 528       # горизонтальное смещение изображения
yoffset = 0         # вертикальное смещение изображения
width = 1000        # ширина изображения
height = 800        # высота изображения
equalize = 1        # эквализация результата
expmethod = 0       # метод вычисления экспозиции (0-авто, 1-вручную)
naverage = 1        # количество усреднений
umax = 24000        # максимальное абсолютное значение отсчетов по U
vmax = 24000        # максимальное абсолютное значение отсчетов по U
focmax = 32000      # макимальное значение отсчетов F
focmin = -32000     # минимальное значение отсчетов F
stpservport = 4444  # порт сервера корректора
Kxu = 71.987        # коэффициенты перевода
Kyu = 54.506        #     координат изображения
Kxv = 22.750        #     в шаги моторов
Kyv = -90.607       #     по осям U и V
xtarget = 1170.900  # координаты центра
ytarget = 480.300   #     цели
eqthrowpart = 0.900 # доля отброшенных пикселей при эквализации
minexp = 50.000     # минимальная экспозиция
maxexp = 1000.000   # максимальная экспозиция
fixedexp = 1000.000 # экспозиция, введенная вручную
intensthres = 0.010 # порог интенсивностей при сортировке
gain = 36.000       # усиление, введенное вручную
brightness = 0.000  # яркость
```

```
starssort = 0       # метод сортировки звезд (0-по расстоянию до цели,
                                              1-по интенсивности)
medfilt = 0         # медианная фильтрация
medseed = 3         # радиус медианной фильтрации
fixedbg = 0         # 1-не считать фон автоматически
fbglevel = 100      # установленный вручную уровень фона
```

2.1.4 Сетевой протокол

В целях безопасности сетевое соединение для подключения клиентов открывается исключительно на локальном компьютере. Для осуществления безопасных удаленных подключений можно выполнить проброс портов посредством ssh.

«Общение» клиента с сервером выполняется по текстовому протоколу. В простейшем случае можно подключиться к указанному в параметрах командной строки порту при помощи утилиты `nc` и вручную передавать команды. Полный список команд с пояснениями появляется в ответ на запрос `help`:

```
maxarea=newval - maximal area (in square pixels) of recognized star image
                      (from 4 to 2.5e+06)
minarea=newval - minimal area (in square pixels) of recognized star image
                      (from 4 to 2.5e+06)
minwh=newval - minimal value of W/H roundness parameter (from 0.3 to 1)
maxwh=newval - maximal value of W/H roundness parameter (from 1 to 3)
ndilat=newval - amount of dilations on binarized image (from 1 to 100)
neros=newval - amount of erosions after dilations (from 1 to 100)
xoffset=newval - X offset of subimage (from 0 to 10000)
yoffset=newval - Y offset of subimage (from 0 to 10000)
width=newval - subimage width (from 0 to 10000)
height=newval - subimage height (from 0 to 10000)
equalize=newval - make histogram equalization (from 0 to 1)
expmethod=newval - exposition method: 0 - auto, 1 - fixed (from 0 to 1)
naverage=newval - calculate mean position by N images (from 1 to 25)
umax=newval - maximal value of steps on U semi-axe (from 100 to 50000)
vmax=newval - maximal value of steps on V semi-axe (from 100 to 50000)
focmax=newval - maximal focus position in microsteps (from 2.22045e-16 to 64000)
focmin=newval - minimal focus position in microsteps (from -64000 to -2.22045e-16)
stpservport=newval - port number of steppers' server (from 0 to 65536)
Kxu=newval - dU = Kxu*dX + Kyu*dY (from -5000 to 5000)
Kyu=newval - dU = Kxu*dX + Kyu*dY (from -5000 to 5000)
Kxv=newval - dV = Kxv*dX + Kyv*dY (from -5000 to 5000)
Kyv=newval - dV = Kxv*dX + Kyv*dY (from -5000 to 5000)
xtarget=newval - X coordinate of target position (from 1 to 10000)
ytarget=newval - Y coordinate of target position (from 1 to 10000)
eqthrowpart=newval - a part of low intensity pixels to throw away when histogram equalized
                      (from 0 to 0.9)
minexp=newval - minimal exposition time (from 0 to 4001)
maxexp=newval - maximal exposition time (from 0 to 4001)
fixedexp=newval - fixed (in manual mode) exposition time (from 0.1 to 4001)
intensthres=newval - threshold by total object intensity when sorting = |I1-I2|/(I1+I2)
                      (from 4.44089e-16 to 1)
```

```
gain=newval - gain value in manual mode (from 0 to 100)
brightness=newval - brightness value (from 0 to 10)
starssort=newval - stars sorting algorithm: by distance from target (0) or by intensity (1)
                    (from 0 to 1)
medfilt=newval - use median filter (from 0 to 1)
medseed=newval - median filter radius (from 1 to 7)
fixedbg=newval - don't calculate background, use fixed value instead (from 0 to 1)
fbglevel=newval - fixed background level (from 0 to 250)
help - List avaiable commands
settings - List current configuration
canbus - Get status of CAN bus server
imdata - Get image data (status, path, FPS, counter)
stpstate=newval - Set given steppers' server state
focus=newval - Move focus to given value
moveU=newval - Relative moving by U axe
moveV=newval - Relative moving by V axe
relay=newval - Send relay commands (Rx=0/1, PWMX=0..255)
```

Часть команд является сеттерами различных параметров конфигурации. Формат сеттеров: «параметр=значение». В случае, если команда–сеттер принята успешно, возвращается ответ `OK` (однако, это не гарантирует ее выполнения). Если же команда–сеттер имеет неправильное (например, выходящие за допустимые диапазоны) значение или же неизвестна, возвращается `FAILED`.

Полностью весь список конфигурационных параметров в формате JSON можно получить по запросу `settings`. Ответом будет строка вида

```
{ "messageid": "settings", "maxarea": 10000, "minarea": 100, "minwh": 0.800,
"maxwh": 1.300, "ndilat": 3, "neros": 3, "xoffset": 528, "yoffset": 0, "width":
1000, "height": 800, "equalize": 1, "expmethod": 0, "naverage": 1, "umax":
24000, "vmax": 24000, "focmax": 32000, "focmin": -32000, "stpservport": 4444,
"Kxu": 71.987, "Kyu": 54.506, "Kxv": 22.750, "Kyv": -90.607, "xtarget":
1043.600, "ytarget": 417.100, "eqthrowpart": 0.900, "minexp": 50.000, "maxexp":
1000.000, "fixedexp": 2000.000, "intensthres": 0.010, "gain": 36.000,
"brightness": 0.000, "starssort": 0, "medfilt": 0, "medseed": 3, "fixedbg": 0,
"fbglevel": 100 }
```

Общим для каждой JSON-строки является параметр `messageid`, характеризующий содержащуюся в строке информацию.

Запрос `canbus` возвращает JSON-строку вида:

```
{ "messageid": "canbus", "status": "ready", "Umotor": { "status": "stopping",
"position": 0 }, "Vmotor": { "status": "stopping", "position": 0 }, "Fmotor":
{ "status": "stopping", "position": 0 }, "relay": 0, "PWM0": 0, "PWM1": 0, "PWM2":
0, "button0": 0, "button1": 0, "button2": 0, "button3": 0 }
```

Здесь отображается состояние устройств (`status`): `disconnected` – соединение с сервером отсутствует, `ready` – пассивное состояние, `setup` – юстировка осей U и V, `gotomiddle` – перемещение подвижки и фокусера в изначальное положение, `findtarget` – определение центра волокна (при обратной засветке), `fixing` – корректор работает, `fixoutofrange` – корректор

не может работать, т.к. объект вышел за допустимые диапазоны его перемещения. У состояний `setup` и `gotomiddle` есть дополнительный параметр — текущее действие (например, `waituv0` – ожидание перемещения подвижек в крайнее отрицательное положение до упора). Каждому мотору соответствует свое поле значений: `Umotor`, `Vmotor` и `Fmotor`. Значения этих полей: `status` – `stopping` или `moving`; `position` – текущее положение в шагах. Параметр `relay` описывает состояние обоих реле: 0 – оба отключены, 1 – включено первое, 2 – включено второе, 3 – включены оба. Параметр `PWMx` характеризует заполнение ШИМ канала `x` ($x = 0 \div 3$): от 0 (нет сигнала) до 255 (полный сигнал). Параметры `buttonx` ($x = 0 \div 3$) характеризуют состояние соответствующей кнопки: 0 – события отсутствуют, 1 – кнопка была нажата в течение более 9 мс, 2 – кнопка была нажата в течение более 199 мс, 3 – кнопка была отпущена.

Запрос `imdata` возвращает состояние граббера:

```
{ "messageid": "imdata", "camstatus": "disconnected", "impath":
"/dev/shm/image.jpg", "imctr": 0, "fps": 0.000, "expmethod": "auto",
"exposition": 100, "gain": 0, "brightness": 0, "xcenter": -1.0, "ycenter": -1.0 }
```

`camstatus` может принимать значения `disconnected` (камера отключена), `watch directory` или `watch file` (мониторинг файловой системы), `connected` (камера подключена). `impath` содержит полный путь к сохраняемому обработанному файлу. `imctr` – счетчик отснятых изображений. `fps` – среднее количество обработанных кадров в секунду. `expmethod` – метод вычисления экспозиции (`auto` или `manual`). `exposition` – длительность последней экспозиции (в миллисекундах). `gain` – текущее значение усиления (в дБ). `brightness` – текущий уровень яркости. `xcenter` и `ycenter` – координаты последнего вычисленного центроида (или -1.0, если центроид вычислить не удалось).

Сеттер `stpstate` позволяет задать требуемый режим работы: `relax` или `disconnect` – выйти в режим бездействия, `middle` – вывести подвижки в среднее положение, `setup` – начать процесс калибровки, `fix` – включить режим коррекции.

Сеттеры `focus`, `moveU` и `moveV` дают возможность перемещать подвижки фокуса и корректора вдоль соответствующей координаты. Между ними есть разница: `focus` принимает **абсолютное** значение (в шагах), а `moveU` и `moveV` — относительные значения. Если требуемое значение за диапазоном перемещения, будет возвращено `FAILED`.

2.1.5 Калибровка корректора положения звезды

Калибровка выполняется автоматически в режиме `stpstate=setup`. Для начала ее проведения необходимо установить все подвижки в среднее положение, навести телескоп на относительно яркую звезду, сфокусироваться телескопом, установить звезду как можно более близко к метке рабочего оптоволокна и включить процедуру калибровки. Процедура заключается в следующем:

1. обе координаты выставляются в среднее положение;
2. подвижка по оси U выставляется в крайнее «отрицательное» положение и запоминаются усредненные координаты звезды;
3. подвижка по оси U выставляется в крайнее «положительное» положение и запоминаются усредненные координаты звезды;
4. аналогично пунктам 2 и 3 определяются крайние координаты по оси V;
5. после измерения координат звезды во всех крайних положениях подвижек коррекции вычисляются коэффициенты преобразования координат.

2.1.6 Коэффициенты преобразования координат

Система координат корректора, (U, V) априори считается неортогональной. Кроме того, по каждой из осей может быть свой масштабирующий коэффициент. Для преобразования смещений в координатном пространстве изображения, $(\Delta x, \Delta y)$, в шаги корректора положения, $(\Delta u, \Delta v)$, необходимо найти коэффициенты матрицы:

$$\begin{pmatrix}\Delta u\\ \Delta v\end{pmatrix} = \begin{pmatrix}K_{xu} & K_{yu}\\ K_{xv} & K_{yv}\end{pmatrix}\begin{pmatrix}\Delta x\\ \Delta y\end{pmatrix}.$$

Непосредственно измерить эти коэффициенты нельзя, т.к. при калибровке в процессе экрана получаются другие коэффициенты:

$$\begin{pmatrix}\Delta x\\ \Delta y\end{pmatrix} = \begin{pmatrix}K_{ux} & K_{vx}\\ K_{uy} & K_{vy}\end{pmatrix}\begin{pmatrix}\Delta u\\ \Delta v\end{pmatrix}.$$

Искомые коэффициенты связаны с данными обратным преобразованием:

$$\begin{pmatrix}K_{xu} & K_{yu}\\ K_{xv} & K_{yv}\end{pmatrix} = \begin{pmatrix}K_{ux} & K_{vx}\\ K_{uy} & K_{vy}\end{pmatrix}^{-1}.$$

Введем коэффициенты пропорциональности: $K_U = \Delta u/\Delta r$, $K_V = \Delta v/\Delta r$, где $\Delta r = \sqrt{\Delta x^2 + \Delta y^2}$. Пусть ось U наклонена по отношению к оси X под углом α, а ось V — под углом β. В этом случае получим:

$$\begin{pmatrix}\Delta x\\ \Delta y\end{pmatrix} = \begin{pmatrix}\cos\alpha & \cos\beta\\ \sin\alpha & \sin\beta\end{pmatrix}\begin{pmatrix}\Delta u/K_U\\ \Delta v/K_V\end{pmatrix} = \begin{pmatrix}\cos\alpha/K_U & \cos\beta/K_V\\ \sin\alpha/K_U & \sin\beta/K_V\end{pmatrix}\begin{pmatrix}\Delta u\\ \Delta v\end{pmatrix}.$$

Отсюда, обратная матрица

$$\begin{pmatrix}K_{xu} & K_{yu}\\ K_{xv} & K_{yv}\end{pmatrix} = \begin{pmatrix}\cos\alpha/K_U & \cos\beta/K_V\\ \sin\alpha/K_U & \sin\beta/K_V\end{pmatrix}^{-1} = K\begin{pmatrix}K_U\sin\beta & -K_U\cos\beta\\ -K_V\sin\alpha & K_V\cos\alpha\end{pmatrix}, \qquad (2.1)$$

где

$$K = \frac{1}{\cos\alpha\sin\beta - \sin\alpha\cos\beta}.$$

Для калибровки выставим корректор в центральное положение, а затем поочередно передвинем его в крайние положения по обеим осям. Таким образом, смещению по оси U на N_U шагов будут соответствовать изменения координат Δy_u и Δx_u, а смещению по оси V на N_V шагов — Δx_v и Δy_v. Коэффициенты пропорциональности определим как

$$K_U = \frac{N_U}{\Delta r_u}, K_V = \frac{N_V}{\Delta r_v}, \quad \text{где} \quad \Delta r_u = \sqrt{{\Delta x_u}^2 + {\Delta y_u}^2}, \Delta r_v = \sqrt{{\Delta x_v}^2 + {\Delta y_v}^2}.$$

А тригонометрические функции направляющих углов — как

$$\cos\alpha = \frac{\Delta x_u}{\Delta r_u}, \quad \sin\alpha = \frac{\Delta y_u}{\Delta r_u}, \qquad \cos\beta = \frac{\Delta x_v}{\Delta r_v}, \quad \sin\beta = \frac{\Delta y_v}{\Delta r_v}.$$

В результате вычислим искомые коэффициенты по уравнению (2.1).

2.1.7 Коррекция телескопом

В случае, если запрос `canbus` возвращает состояние `fixoutofrange`, необходимо сделать коррекцию телескопом для возвращения объекта в допустимую область. В данный момент автоматическая коррекция не реализована. Для работы данного модуля необходимо выполнить процедуру калибровки, аналогичную калибровке локального корректора положения звезды, чтобы вычислить коэффициенты перехода от экранных координат к экваториальным.

2.2 Демон `canserver`

Обеспечивает работу[2] с устройствами, подключенными по CAN-шине к преобразователю USB–CAN (см. стр. 12, п. 3.1). Устройство преобразователя можно определить по его PID/VID либо файлу устройства. Демон обеспечивает резервирование: можно подключить два одинаковых устройства преобразователя к общей CAN-шине, вставив их в разные порты USB. В случае, если базовое устройство перестанет корректно работать, автоматически произойдет переподключение к резервному.

2.2.1 Аргументы командной строки

-P, –pid=arg PID устройства преобразователя;

-V, –vid=arg VID устройства;

-e, –echo включить опцию «эха» введенных пользователем команд;

-i, –device=arg название файла устройства преобразователя;

-h, –help отобразить данную справку;

-l, –logfile=arg сохранять логи в указанный файл;

-p, –port=arg использовать для подключения указанный сетевой порт;

-s, –speed=arg скорость соединения по CAN-шине (в бодах);

-v, –verbose повысить уровень подробностей логгирования (каждая `-v` повышает уровень на 1);

–pidfile=arg имя PID-файла процесса (по умолчанию: `/tmp/canserver.pid`).

2.2.2 Сетевой протокол

В целях безопасности сетевое соединение для подключения клиентов открывается исключительно на локальном акомпьютере. Для осуществления безопасных удаленных подключений можно выполнить проброс портов посредством ssh. Протокол — текстовый. Для получения базовой справки можно выполнить запрос `help`:

[2] `https://github.com/eddyem/pusirobot/tree/master/canserver`

```
help> help - show help
help> list - list all threads
help> mesg NAME MESG - send message ‘MESG‘ to thread ‘NAME‘
help> register NAME ID ROLE - register new thread with ‘NAME‘, raw receiving ‘ID‘
                    running thread ‘ROLE‘
help> threads - list all possible threads with their message format
help> unregister NAME - kill thread ‘NAME‘
```

Формат передаваемых сообщений: первым словом идет имя потока или определенное ключевое слово (от имени потока отличается наличием знака »"в конце, поэтому не стоит именовать потоки подобным образом). Далее следует непосредственно команда `параметр=значение`. В случае неправильно введенной команды последует ответ сервера: `Wrong command`, если же команда принята, ответом будет `OK` с возможными дальнейшими сообщениями.

Чтобы обеспечить гибкость работы с разнообразными устройствами, подключенными к CAN-шине, каждым устройством управляет один поток. Новый поток нужно «зарегистрировать» командой `register`. Ее параметры: имя потока (должно быть уникальным, иначе в ответ получим ошибку), идентификатор потока (классический CANbus ID, который поток будет слушать, скажем, если нам нужно подключиться к CANopen устройству с NodeID=10, то ID=0x58A) и роль потока.

Например, зарегистрируем в системе два двигателя с именами «x» и «y»:

```
register x 0x58a stepper
OK
x maxspeed=OK
register y 0x58b stepper
OK
y maxspeed=OK
```

После регистрации устройства все принимаемые пакеты с указанным для него идентификатором будут перенаправлены в поток, обслуживающий это устройство. Соответственно, в зависимости от «роли» устройства, поступающие данные будут обрабатываться и сообщаться клиенту.

Командой `list` можно посмотреть, какие потоки запущены:

```
list
thread> name='x' role='stepper' ID=0x58A
thread> name='y' role='stepper' ID=0x58B
thread> Send message 'help' to threads marked with (args) to get commands list
```

Список всех возможных «ролей» можно получить при помощи команды `threads`:

```
role> canopen NodeID index subindex [data] - raw CANOpen commands with ‘index‘
and ‘subindex‘ to ‘NodeID‘
role> emulation (args) - stepper emulation
role> raw ID [DATA] - raw CANbus commands to raw ‘ID‘ with ‘DATA‘
role> stepper (args) - simple stepper motor: no limit switches, only goto
```

«Роль» `canopen` дает возможность работать с CAN-шиной в режиме CANopen, посылая туда пакеты и читая ответные данные. «Роль» `raw` дает доступ к «чистой» CAN-шине. В

случае указания нулевого идентификатора, будут приниматься все команды, передающиеся по CAN-шине, поэтому удобно использовать `raw` с нулевым идентификатором для полного мониторинга сети.

Команда `mesg` позволяет отправлять сообщения указанным потокам. Чтобы посмотреть, какие команды принимает конкретный поток, посылаем ему сообщение `help`:

```
mesg x help
OK
x> COMMAND   NARGS   MEANING
x> stop        0     stop motor and clear errors
x> status      0     get current position and status
x> relmove     1     relative move
x> absmove     1     absolute move to position arg
x> enable      1     enable (!0) or disable (0) motor
x> setzero     0     set current position as zero
x> maxspeed    1     set/get maxspeed (get: arg==0)
x> info        0     get motor information
```

Например, для получения полной информации о драйвере «x», отправим команду `info`:

```
mesg x info
OK
x errstatus=0
x devstatus=0
x curpos=0
x enable=1
x microsteps=32
x extenable=0
x maxspeed=3200
x maxcurnt=600
x gpioval=8191
x rotdir=1
x relsteps=0
x abssteps=0
```

Здесь перечислены: флаги ошибок, состояние устройства, текущее положение (в микрошагах), активность обмоток двигателя, количество микрошагов в одном шаге, останов двигателя по внешнему концевику, максимальная скорость (в микрошагах в секунду), максимальный ток (в микроамперах), текущее значение на пинах GPIO, направление вращения, последнее заданное относительное положение (в микрошагах), последнее заданное абсолютное положение (в микрошагах).

Для перемещения в заданное абсолютное положение дадим команду `absmove`:

```
mesg x absmove 3200
OK
x abssteps=OK
mesg x status
OK
x devstatus=0
```

```
x curpos=3200
x errstatus=0
```

В случае ненулевого значения errstatus, сбросить флаги ошибок можно командой `stop`.

Если во время движения попробовать дать новую команду движения без команды `stop`, придет сообщение об ошибке:

```
mesg x relmove 1000
OK
x abortcode='0x8000022' error='Data cannot be transferred or stored to the
                      application because of the present device state'
x abortcode='0x8000022' error='Data cannot be transferred or stored to the
application because of the present device state'
```

Т.о., следует проверять все сообщения, т.к. ответ `OK` лишь подтверждает правильность введенной команды и возможность ее отправки соответствующему устройству. В конкретном случае сообщений об ошибке два, т.к. `relmove` сначала пытается задать направление движения, а затем — указать, сколько шагов надо проехать.

Ответы `OK` на запросы видит лишь тот клиент, который отправлял данные запросы. Ответы от сервера отправляются всем подключенным клиентам.

3 Электронные компоненты

3.1 Преобразователь USB–CAN

Данный преобразователь[3] разработан на основе микроконтроллера STM32F042C6T6. При подключении к ПК эмулирует преобразователь USB–UART PL2303, так что нет необходимости настраивать нестандартное устройство. Кроме того, такой подход позволяет после конфигурации устройства (например, при помощи `stty`) работать с ним напрямую из bash-скриптов без утилит вроде screen и т.п.

3.1.1 Протокол последовательного интерфейса

Используется текстовый протокол с односимвольными командами. При вводе неправильной команды (например, `?`), пользователь получает справку:

```
'a' - add ID to ignore list (max 10 IDs)
'b' - reinit CAN with given baudrate
'd' - delete ignore list
'f' - add/delete filter, format: bank# FIFO# mode(M/I) num0 [num1 [num2 [num3]]]
'F' - send/clear flood message: F ID byte0 ... byteN
'I' - reinit CAN
'l' - list all active filters
'o' - turn LEDs OFF
'O' - turn LEDs ON
'p' - print ignore buffer
'P' - pause/resume in packets displaying
```

[3] `https://github.com/eddyem/stm32samples/tree/master/F0-nolib/usbcan`

```
'R' - software reset
's/S' - send data over CAN: s ID byte0 .. byteN
'T' - get time from start (ms)
```

По умолчанию устройство работает на скорости 100 кбод. Если необходима другая скорость CAN, следует сразу после включения задать ее командой `b`.

Численные данные выводятся в шестнадцатеричном формате, а ввод позволяют делать в двоичном (например, `0b110011`), восьмеричном (например, `0123`), десятичном или шестнадцатеричном (например, `0xdeadbeef`).

Реализован программный «черный список», позволяющий игнорировать до десяти идентификаторов, а также аппаратные фильтры STM32. Командой `l` можно получить информацию о всех активных фильтрах. По умолчанию их два:

```
Filter 0, FIFO0 in MASK mode: ID=0x01, MASK=0x01
Filter 1, FIFO1 in MASK mode: ID=0x00, MASK=0x01
```

Один фильтр размещает пакеты с нечетными идентификаторами в FIFO0, другой размещает четные в FIFO1. Удалив оба этих фильтра (`f 0`, `f 1`), можно установить необходимые фильтры (на конкретный список идентификаторов или же по маске). Например, `f 0 0 M 0 0x3fc` настраивает фильтр номер 0 на FIFO0 для идентификаторов с 0 по 7 (т.е. младшие 3 бита), а `f 1 1 I 11 15 20 30` настраивает фильтр 1 на FIFO1 для списка идентификаторов.

Команды `o` и `O` позволяют отключать или включать диагностические светодиоды (один светодиод, LED1, постоянно горит при стабильной связи; второй, LED0, вспыхивает в момент получения сообщений, прошедших фильтрацию).

При помощи команды `s` можно отправлять данные в шину. После команды указывается идентификатор сообщения за которым следуют от нуля до восьми байт данных. В случае ошибки выводятся соответствующие сообщения, например, если на шине нет ни одного устройства, либо настроена неправильная скорость, получим:

```
s 123 0b1010
Message parsed OK

Too much errors, restarting CAN!
Receive error counter: 0
Transmit error counter: 96
Last error code: Ack error
Error counter limit
```

Прошедшие фильтрацию принятые сообщения отображаются в формате `# ID [data]`: после символа решетки следует идентификатор сообщения, за которым перечисляются данные тела сообщения (если длина данных больше нуля).

Приостановить/возобновить отображение пришедших сообщений можно при помощи команды `P`.

Команда `F` позволяет отправлять в сеть каждые 5 мс заданное сообщение.

3.2 Модуль управления нагрузкой

Модуль[4] разработан на основе микроконтроллера STM32F042C6T6 и по сути является «наследником» преобразователя USB–CAN. Помимо приема команд по USB или CAN для ра-

[4] `https://github.com/eddyem/stm32samples/tree/master/F0-nolib/usbcan_relay`

боты с нагрузкой, модуль может выступать и в роли преобразователя USB–CAN.

CAN–идентификатор устройства задается переключателями на плате (возможно задать ID от 0 до 255). Скорость интерфейса конфигурируется USB-командой `C`, по умолчанию составляет 250 кбод.

3.2.1 Протокол последовательного интерфейса

```
'0' - turn relay0 on(1) or off(0)
'1' - turn relay1 on(1) or off(0)
'a' - add ID to ignore list (max 10 IDs)
'A' - get ADC values @ all 4 channels
'b' - get buttons' state
'C' - reinit CAN with given baudrate
'd' - delete ignore list
'f' - add/delete filter, format: bank# FIFO# mode(M/I) num0 [num1 [num2 [num3]]]
'F' - send/clear flood message: F ID byte0 ... byteN
'I' - read CAN ID
'l' - list all active filters
'm' - get MCU temp & Vdd
'o' - turn nth LED OFF
'O' - turn nth LED ON
'p' - print ignore buffer
'P' - pause/resume in packets displaying
'R' - software reset
's/S' - send data over CAN: s ID byte0 .. byteN
'T' - get time from start (ms)
'w' - get PWM settings
'W' - set PWM @nth channel (ch: 0..2, PWM: 0..255)
```

Команды включения/выключения реле: состоят из номера реле и аргумента (0/1), например, `0 1` включит реле номер 0.

Считать информацию со всех четырех каналов (два внешних, температура МК и Vdd) можно при помощи команды `A`.

Команда `b` позволяет получить информацию о состоянии кнопок, ответ выводится в виде человекочитаемой строки, например: `The key 2 is released at 145623` (т.е. в условное время145623 мс кнопка номер 2 была отпущена). Нумерация начинается с нуля.

Команда `I` отображает установленный переключателями идентификатор CAN.

При помощи команды `o x` можно отключить питание с внешнего светодиода номер `x` (от 0 до 2). Командой `O x` можно подать на него питание.

Команда `w` позволяет получить данные ШИМ со всех трех каналов (от 0 до 2), а команда `W x val` — установить на канале с номером `x` значение заполнения ШИМ `val` (от 0 – сигнал отсутствует до 255 – максимальный сигнал).

В отличие от преобразователя USB–CAN, данное устройство не позволяет изменять фильтр номер 0: в момент старта он настраивается на приме сообщений с идентификатором, соответствующим выставленному при помощи переключателей на плате модуля.

3.2.2 Протокол интерфейса CAN

Если устройство получает на свой идентификатор пустое сообщение, оно отправляет его обратно («ping»). У сообщения с ненулевой длиной анализируется байт 0 пришедших данных (это — команда). Команда может быть одна из:

```
typedef enum{
    CAN_CMD_PING,   // ping (without setter)
    CAN_CMD_RELAY,  // relay get/set
    CAN_CMD_PWM,    // PWM get/set
    CAN_CMD_ADC,    // get ADC (without setter)
    CAN_CMD_MCU,    // MCU T and Vdd
    CAN_CMD_LED,    // LEDs get/set
    CAN_CMD_BTNS,   // get Buttons state (without setter)
    CAN_CMD_TMS,    // get time from start (in ms)
    CAN_CMD_ERRCMD, // wrong command
    CAN_CMD_SETFLAG = 0x80 // command is setter
} CAN_commands;
```

Старший бит команды (`CAN_CMD_SETFLAG`) является маркером того, что команда — сеттер. Без этого маркера команда рассматривается как геттер. Т.е. номер команды определяется выражением `byte[0]&0x7f`.

Начиная с номера 8 (`CAN_CMD_ERRCMD`) команда считается ошибочной, в этом случае в сеть отправляется пакет с длиной, равной единице, где нулевой байт данных содержит код `CAN_CMD_ERRCMD`.

Идентификатор ответного сообщения совпадает с идентификатором устройства.

Большинство команд передает данные в формате little endian, кроме состояния АЦП.

CAN_CMD_ADC — получить значение всех каналов АЦП. Поле данных в байтах $1 \div 6$ содержит смешанные по 12 бит показания АЦП (каналы 5 В, 12 В и внутренние: температура МК и Vdd).

CAN_CMD_BTNS — получить параметры кнопок. В байте `data[1]` указан номер кнопки, `data[2]` – ее состояние (0 – не нажата, 1 – была нажата больше 9 мс и меньше 200 мс, 2 – была нажата больше 200 мс, 3 – была отпущена), `data[4..7]` (для выравнивания) – время наступления событий в условных мс от запуска МК или переполнения счетчика времени (`uint32_t`, little endian).

CAN_CMD_LED — установить или получить состояние светодиодов. В случае сеттера `data[1]` входящего потока задает состояние соответствующего светодиода (если n-й бит установлен, светодиод включается, нет – гаснет), в ответе `data[1]` указывает на текущее состояние светодиодов (аналогично сеттеру).

CAN_CMD_MCU — температура МК и значение Vdd: в `data[2,3]` содержится температура МК ($T \cdot 10°C$, little endian `int16_t`), в полях `data[4,5]` — значение Vdd ($V \cdot 100$, little endian `uint16_t`).

CAN_CMD_PWM позволяет задавать и читать заполнение соответствующих ШИМ каналов, поля у сеттера и геттера аналогичны: `data[1..3]` — соответствующее значение заполнения $(0 \div 255)$ ШИМ для каналов $0 \div 2$.

CAN_CMD_RELAY управляет реле, поля сеттеров и геттеров аналогичны: биты 0 и 1 в `data[1]` отражают состояние соответствующего реле (1 – включено, 0 – выключено).

CAN_CMD_TMS содержит в полях `data[4..7]` время Tms — условное время в миллисекундах с момента запуска микроконтроллера (little endian `uint32_t`).