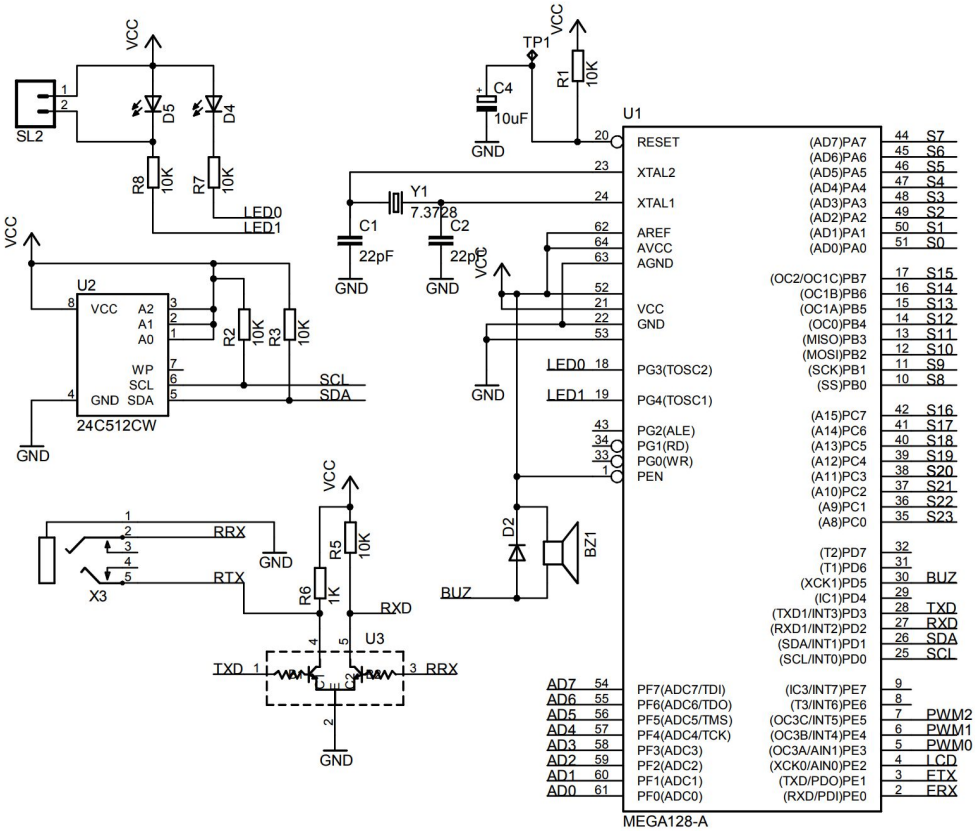
Черновик по наработкам с Robonova MF2.

Вводная:

Robonova MF является роботом-гуманоидом, ОДС которого состоит из 16 сервоприводов (+1 голова).
Контроллер(плата) MR-C3024FX - обычный ATmeaga128A.
Соль платы - в его бутлоадере (загрузчике).
Искать материалы лучше по словам “MR-C3024”, т.к. данная плата используется на многих роботах.

Бутлоадер засекречен разработчиком, но форумчане слили HEX.

Меняем бутлоадер на кастомный + описание HMI протокола, благодаря которому возможны чудеса (смотри внутрь):
Slave-architecture-for-the-Robonova-MR-C3024



Полная схема тут.

Архитектура MR-C3024:

ATmega128A имеет два UART-интерфейса:
- UART0 - для общения с внешними устройствами через контакты ETX, ERX.
- UART1 - интерфейс программирования с ПК, в котором аппаратно инвертируется RX (это видно на схеме ЭП).
- S0-S23 - выходы управления сервоприводами.

Стоковый бутлоадер обеспечивает исполнение HEX-команд, приходящих на UART1, согласно спец.протоколу.

Что-бы отработать команду, нужно открыть терминал на COM-порте (USB-UART преобразователя), и отправить HEX-значение. У меня получилось отправить 0xAF (reboot controller). Для отработки остальных команд протокола, нужно писать программу, т.к. команды выполняются с обратной связью (ожидают ответа от ПК).

Особенности языка RoboBASIC:

Преимуществом языка roboBASIC является параллельное управление сервоприводами по группам (как по 6, так и 24 сразу). **MOVE24** и **MOVE** [GROUP] [MOTOR POS]. Синтаксис всех команд описан в документе под ссылкой выше.

Изучив архитектуру платы и набор команд языка можно использовать такие возможности как:

- I2C
- ADC
- SPI
- UART

Использование вышеперечисленных интерфейсов ограничивает только один огромный нюанс: команды изменения положения сервоприводов (**MOVE** и **MOVE24**) не могут принимать за аргументы переменные. То есть, нельзя написать следующее:
DIM **POS** AS BYTE
MOVE G6A, 93, 76, 145, 94, **POS**, 100,

Из этого следует, что средствами roboBASIC **НЕВОЗМОЖНО** создать алгоритм изменения положения сервоприводов, в зависимости от динамически-изменяемого параметра.

Также хорошей, но ограниченной функцией является ETX, ERX (UART наружу). Подключаемся к одноименным контактам на плате, и начинаем общение:

ERX *[скорость UART в BAUD], [Receipt variable], [Receipt Error Process Label]*
Пример:

```
Retry:
        ERX 9600, A, Retry
        IF A = &HAA THEN GOTO Label1
        IF A = &HAB THEN GOTO Label2
Label1:
        MOVE G6A 100, 100, 100, 100, 100, 100
        GOTO Retry
Label2:
        MOVE G6A 100, 100, 100, 100, 100, 100
```

***GOTO** Retry*

Переходим на метку “Retry” пока не прочитаем по UART значение в переменную “A”. Если получаем по UART значение **0xAA**, то переходим на метку “**Label1**”, а при получении **0xAB** - переходим на “**Label2**”.

К входам **ERX** и **ETX** можно подключить **TX** и **RX** контакты (соответственно) платы Arduino, и удобно передавать HEX-значения с помощью команды:

```
Serial.write(0xAA); // send a byte with the value 0xAA
```

Таким образом, на **Arduino** можно возложить алгоритм формирования управляющих сигналов для робота. Также, с помощью Arduino удобно выполнять считывание и фильтрование значений с датчиков.

Использование гироскопа-акселерометра **GY-521** (МС **MPU-6050**):

При использовании микросхемы **MPU-6050**, нужно использовать алгоритмы калибровки от производителя и выполнять фильтрацию комплексных измерений, с целью получения точных значений положения датчика.

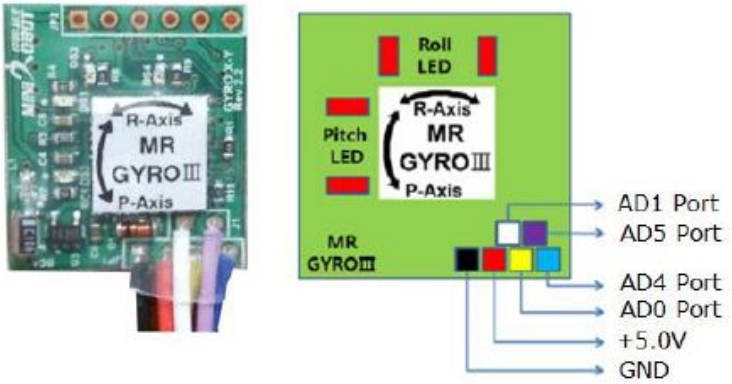
Для примера работы с датчиком, и визуализации данных, используем связку Arduino + Processing. Ссылки кликабельны.

Arduino-код считывает подключенный по I2C датчик, и выдает по UART данные.

Processing-код парсит данные с UART и рисует три прямоугольника: гироскоп, акселерометр и отфильтрованное их значение.

При использовании низкой скорости передачи по UART, видны торможения. Поэтому, выставляем максимально большой баудрейт, при котором обмен происходит без ошибок. Дальше можно сократить UART- сообщение, для увеличения скорости обмена и КПД.

Использование пьезоэлектрического двухосного датчика положения **MR-GYRO III**.



Документации на аппаратную часть нет, но выявлено, что на входы датчика подается меандр (**ШИМ**, м.б. даже **ШИМ** для сервопривода), а на выходах уже скорректированный **ШИМ**. Пример работы подобного датчика здесь. Кодом можно отрегулировать чувствительность датчика, направление реакции двигателя, и привязать выход датчика к двигателю.

Подключаем датчик (конечно же у нас его нет) согласно таблице.

Wire Color	Wire Function	AD Port	AD Port Pin	MR-C3024FX Port	Gyro Sensor # GYROSET command	Gyro Port Function
Black	GND	AD0	GND	32		
Red	+5V	AD0	+5V	32		
Yellow	Pitch Axis, Basic Signal X	AD0	Signal	32	1	Output Port
White	Roll Axis, Signal Y	AD1	Signal	33	2	Output Port
Blue	Pitch Axis, Result X	AD4	Signal	36	1	Input Port
Purple	Roll Axis, Result Y	AD5	Signal	37	2	Input Port

В документации на робот есть следующие модели гироскопов:

- gws pg-03 gyro
- FUTABA G190 MICRO PIEZO GYRO
- SPARKFUN - TRIPLE AXIS ACCELEROMETER

Датчик можно попробовать смоделировать с помощью Arduino.