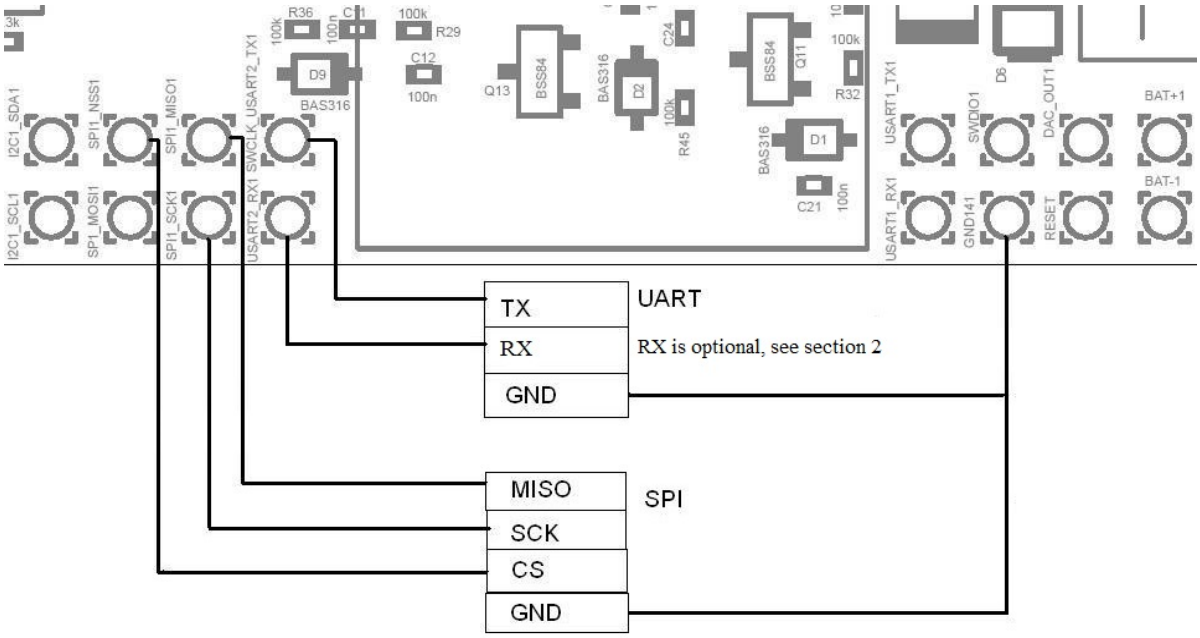
Hardware interface and protocol of data exchange with mobile beacon via USB, UART and SPI interfaces.

Version 2016.12.10
Valid for firmware v4.07 and newer

For communication with mobile beacon (hedgehog), it shall be connected to an external device (robot, copter, AGV, etc.) via any of the following interfaces:

1. Connect to USB-host as an USB device of CDC class (virtual COM port in Windows, ttyUSB or ttyACM in Linux). In the Windows, it requires driver - the same driver as for modem. In Linux, the driver is not required, since the required driver is integrated into Linux kernel. Because real RS-232 is not used in the interface, parameters of serial port opened on the host (baudrate, number of bits, parity, etc) may be any.
2. Connect to UART on a hedgehog – 2 wires soldering to pins required. See the picture of beacon interface below. To have the location data out, it is sufficient to connect only 2 wires: GND and USART2_TX. Logic level of UART transmitter is CMOS 3.3V. Default baudrate is 500 kbps, configurable from the Dashboard from following list: 4.8, 9.6, 19.2, 38.4, 57.6, 115.2, 500 kbps. Format of data: 8 bit, no parity, 1 stop bit.
3. Connect to SPI. Hedgehog acts as SPI slave device. Parameters of SPI: SPI mode 0, MSB inside each byte transmits first. Connection was tested on SCK speed up to 8 MHz. Be careful to provide quality wiring connections on high speeds (more than 500 KHz). Formats of packets transmitting via SPI are same as for serial ports (USB and UART).



The hedgehog constantly streams packets of location data, defined in section 1, to UART and USB without any request. To get data from SPI, master should provide signals of chip select (CS) and clock (SCK).

Marvelmind robotics

The section 2 defines the protocol for reading/writing common data from/to user device through the hedgehog's UART or USB. One of the applications is transmission of movement path to the robot, copter or any another vehicle.

Data is represented in binary format.

Multibyte numbers are transmitted starting from low byte (little endian format).

1. Streaming packet format

All streaming packets have same general structure:

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Destination address	0xff
1	1	uint8_t	Type of packet	0x47
2	2	uint16_t	Code of data in packet	See detail
4	1	uint8_t	Number of bytes of data transmitting	N
5	N	N bytes	Payload data according to code of data field	
5+N	2	uint16_t	CRC-16 (see appendix)	

1.1. Packet of hedgehog coordinates (code of data 0x0001).

This packet is transmitted every time new coordinates are measured or failed to measure.

1.1.1. Packet with cm resolution coordinates.

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Destination address	0xff
1	1	uint8_t	Type of packet	0x47
2	2	uint16_t	Code of data in packet	0x0001
4	1	uint8_t	Number of bytes of data transmitting	0x10
5	4	uint32_t	Timestamp – internal time of beacon measured in milliseconds from the moment of the latest wakeup event. See note.	
9	2	int16_t	Coordinate X of beacon, cm	
11	2	int16_t	Coordinate Y of beacon, cm	
13	2	int16_t	Coordinate Z, height of beacon, cm	
15	1	uint8_t	Byte of flags: Bit 0: 1 - coordinates unavailable. Data from fields X,Y,Z should not be used. Bit 1: timestamp units indicator (see note) Bit 2: 1 - user button is pushed (V5.23+) Bit 3: 1 - data are available for uploading to user device, see section 2 (V5.34+) Bit 4: 1 - want to download data from user device, see section 2 (V5.34+) Bit 5…7: – reserved (0)	
16	1	uint8_t	Address of hedgehog	
17	4	4 bytes	Reserved (0)	
21	2	uint16_t	CRC-16 (see appendix)	

1.1.2. Packet with mm resolution coordinates (firmware V5.35+).

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Destination address	0xff
1	1	uint8_t	Type of packet	0x47
2	2	uint16_t	Code of data in packet	0x0011
4	1	uint8_t	Number of bytes of data transmitting	0x16
5	4	uint32_t	Timestamp – internal time of beacon measured in milliseconds from the moment of the latest wakeup event. See note.	
9	4	int32_t	Coordinate X of beacon, mm	
13	4	int32_t	Coordinate Y of beacon, mm	
17	4	int32_t	Coordinate Z, height of beacon, mm	
21	1	uint8_t	Byte of flags: Bit 0: 1 - coordinates unavailable. Data from fields X,Y,Z should not be used. Bit 1: timestamp units indicator (see note) Bit 2: 1 - user button is pushed (V5.23+) Bit 3: 1 - data are available for uploading to user device, see section 2 (V5.34+) Bit 4: 1 - want to download data from user device, see section 2 (V5.34+) Bit 5…7: – reserved (0)	
22	1	uint8_t	Address of hedgehog	
23	4	4 bytes	Reserved (0)	
27	2	uint16_t	CRC-16 (see appendix)	

Note: For firmware versions before 5.20, timestamp is in alpha-cycle periods (1/64 sec). This is indicated by zero value of bit 1 in byte of flags. For firmware version 5.20+ the timestamp is in milliseconds and bit 1 in byte of flags has value 1.

1.2. Packet of all beacons coordinates (code of data 0x0002).

This packet is transmitted when system becomes frozen, and repeats every 10 sec.

1.2.1. Packet with cm resolution coordinates.

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Address	0xff
1	1	uint8_t	Type of packet	0x47
2	2	uint16_t	Code of data in packet	0x0002
4	1	uint8_t	Number of bytes of data transmitting	1+N*8
5	1	uint8_t	Number of beacons in packet	N
6	1	N*8 bytes	Data for N beacons	
6+N*8	2	uint16_t	CRC-16 (see appendix)	

Format of data structure for every of N beacons:

Offset	Size (bytes)	Type	Description
0	1	uint8_t	Address of beacon
1	2	int16_t	Coordinate X of beacon, cm
3	2	int16_t	Coordinate Y of beacon, cm
5	2	int16_t	Coordinate Z, height of beacon, cm
7	1	uint8_t	Reserved (0)

1.2.2. Packet with mm resolution coordinates (firmware V5.35+).

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Address	0xff
1	1	uint8_t	Type of packet	0x47
2	2	uint16_t	Code of data in packet	0x0012
4	1	uint8_t	Number of bytes of data transmitting	1+N*14
5	1	uint8_t	Number of beacons in packet	N
6	1	N*14 bytes	Data for N beacons	
6+N*14	2	uint16_t	CRC-16 (see appendix)	

Format of data structure for every of N beacons:

Offset	Size (bytes)	Type	Description
0	1	uint8_t	Address of beacon
1	4	int32_t	Coordinate X of beacon, mm
5	4	int32_t	Coordinate Y of beacon, mm
9	4	int32_t	Coordinate Z, height of beacon, mm
13	1	uint8_t	Reserved (0)

2. Protocol of reading/writing data from/to user device.

2.1. Packet from user device of confirmation readiness to transmit/receive data.

User device should monitor stream from hedgehog (see section 1.1). If the byte of flags contains in bits 3 or 4 values "1",this indicates ready data for uploading to user device or request to download data. The user device can reply by the following confirmation packet.

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Address of hedgehog (get from 0x0001 packet of streaming)	
1	1	uint8_t	Type of packet	0x48
2	2	uint16_t	Code of data in packet	0x0100
4	1	uint8_t	Number of bytes of data transmitting	0x04
5	1	uint8_t	Status byte: Bit 0: 1 - user device ready to receive data Bit 1: 1 - user device ready to send data Bit 2...7: reserved (0)	
6	3	3 bytes	Reserved (0)	0
9	2	uint16_t	CRC-16 (see appendix)	

After this reply, if hedgehog detects set bits of readiness, it should send requests for read/write as shown in following sections.

2.2. Reading data from user device to hedgehog.

This packet is transmitted from hedgehog if it wants to get data from user device and bit 0 in status byte of confirmation packet (see section 2.1) is set to 1.

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Destination address	0xff
1	1	uint8_t	Type of packet	0x49
2	2	uint16_t	Code of requested data	0x0200... 0x02ff
4	1	uint8_t	Number of bytes of data transmitting	0x04
5	4	4 bytes	Reserved (0)	0
9	2	uint16_t	CRC-16 (see appendix)	

For this command the codes of data from 0x200 to 0x2ff are reserved.

If the user device successfully processed the request, it sends response in following format:

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Address of hedgehog (get from 0x0001 packet of streaming)	
1	1	uint8_t	Type of packet	0x49
2	2	uint16_t	Code of data in packet	0x0200... 0x02ff
4	1	uint8_t	Number of bytes of data transmitting	N
5	N	N bytes	Payload data, depends from code of data	
5+N	2	uint16_t	CRC-16 (see appendix)	

If the user device failed to process the request, it sends response in following format:

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Address of hedgehog (get from 0x0001 packet of streaming)	
1	1	uint8_t	Type of packet	0xc9
2	2	uint16_t	Code of requested data	0x0200... 0x02ff
4	1	uint8_t	Code of error (see Appendix 3)	1
5	2	uint16_t	CRC-16 (see appendix)	

2.3. Writing data from hedgehog to user device.

This packet is transmitted from hedgehog if it wants to send data to user device and bit 1 in status byte of confirmation packet (see section 2.1) is set to 1.

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Destination address	0xff
1	1	uint8_t	Type of packet	0x4a
2	2	uint16_t	Code of data in packet	0x0200... 0x02ff
4	1	uint8_t	Number of bytes of data transmitting	N
5	1	N bytes	Payload data	
5+N	2	uint16_t	CRC-16 (see appendix)	

For this command the codes of data from 0x200 to 0x2ff are reserved.

If the user device successfully processed the request, it sends response in following format:

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Address of hedgehog (get from 0x0001 packet of streaming)	
1	1	uint8_t	Type of packet	0x4a
2	2	uint16_t	Code of data in packet	0x0200... 0x02ff
4	2	uint16_t	CRC-16 (see appendix)	

If the user device failed to process the request, it sends response in following format:

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Address of hedgehog (get from 0x0001 packet of streaming)	
1	1	uint8_t	Type of packet	0xca
2	2	uint16_t	Code of requested data	0x0200... 0x02ff
4	1	uint8_t	Code of error (see Appendix 3)	1
5	2	uint16_t	CRC-16 (see appendix)	

In the following sections (2.3.x) described the specific data writing requests.

2.3.1. Request of writing the movement path.

This packet contains one command of elementary movement. The hedgehog sends one after another all commands for elementary movements in the path.

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Destination address	0xff
1	1	uint8_t	Type of packet	0x4a
2	2	uint16_t	Code of data in packet	0x201
4	1	uint8_t	Number of bytes of data transmitting	0x0c
5	1	12 bytes	Payload data	
5+N	2	uint16_t	CRC-16 (see appendix)	

Format of payload data:

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Type of elementary movement: 0 - move forward 1 - move backward 2 - rotate right (clockwise) 3 - rotate left (counterclockwise) 4 - pause 5 - repeat program from start 6 - move to specified point 7 - setup speed	
1	1	uint8_t	Index of this elementary movement (0 is the first)	
2	1	uint8_t	Total number of elementary movements	
3	2	int16_t	Parameter of movement: Types 0; 1 - distance of movement, cm Types 2; 3 - angle of rotation, degrees Type 4: time of pause, ms Type 6: X target coordinate, cm Type 7: speed, %	
5	2	int16_t	Parameter of movement: Type 6: Y target coordinate, cm	
7	5	5 bytes	Reserved (0)	

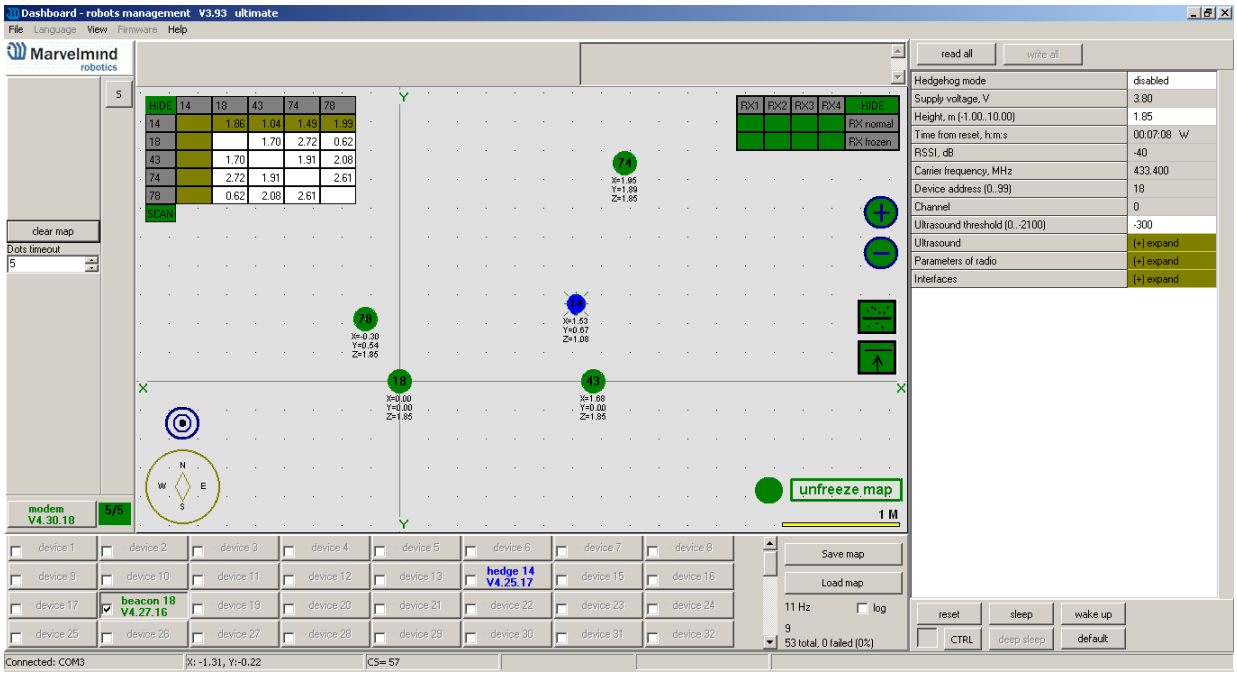
Appendix 1. Calculating CRC-16

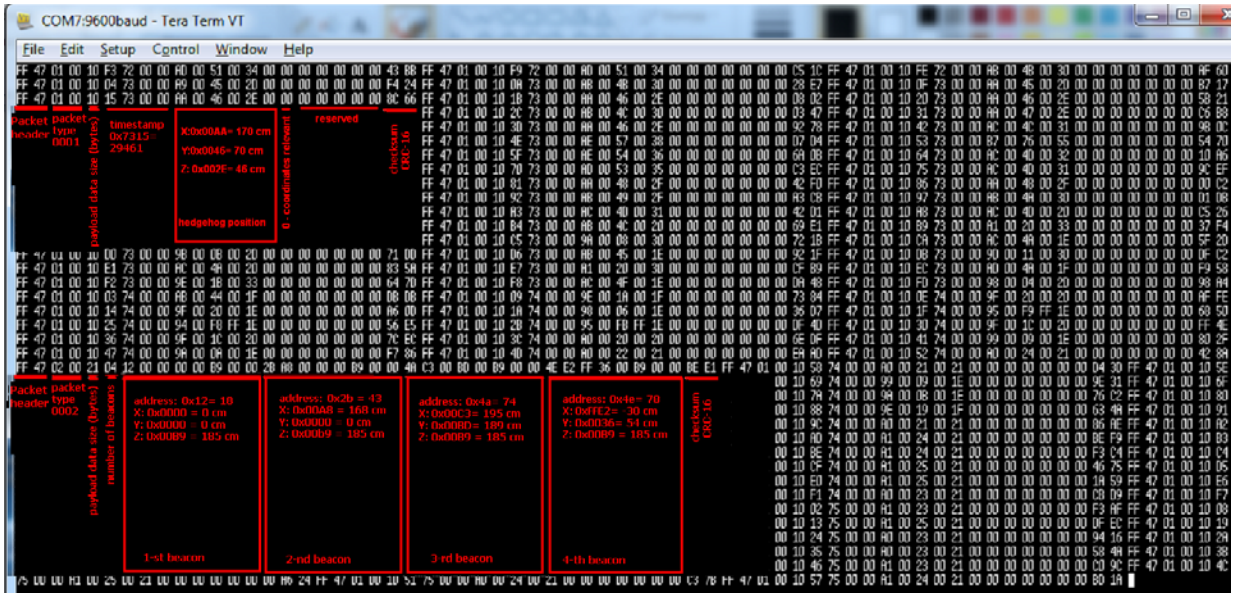
For checksum the CRC-16 is used. Last two bytes of N-bytes frame are filled with CRC-16, applied to first (N-2) bytes of frame. To check data you can apply CRC-16 to all frame of N bytes, the result value should be zero.

Below is the implementation of the algorithm in the 'C':

```
typedef ushort ModbusCrc;//  ushort – two bytes

typedef union {
        ushort w;
        struct{
                uchar lo;
                uchar hi;
        } b;
        ucharbs[2];
} Bytes;

static Modbus CrcmodbusCalcCrc(const void *buf, ushort length)
{
        uchar *arr = (uchar *)buf;
        Bytes crc;

        crc.w = 0xffff;

        while(length--){
                chari;
                bool odd;

                crc.b.lo ^= *arr++;
                for(i = 0; i< 8; i++){
                        odd = crc.w& 0x01;
                        crc.w>>= 1;
                        if(odd)
                                crc.w ^= 0xa001;
                }
        }
        return (ModbusCrc)crc.w;
}
```

Appendix 2. Data dump





Appendix 3. Codes of errors, reported by user device

If user device could not process request from hedgehog, it should send reply with one of following error codes:

1 - unknown field "type of packet" in request
2 - unknown field "code of data" in request
3 - incorrect payload data in request
6 - device is busy and cannot retrieve requested data now