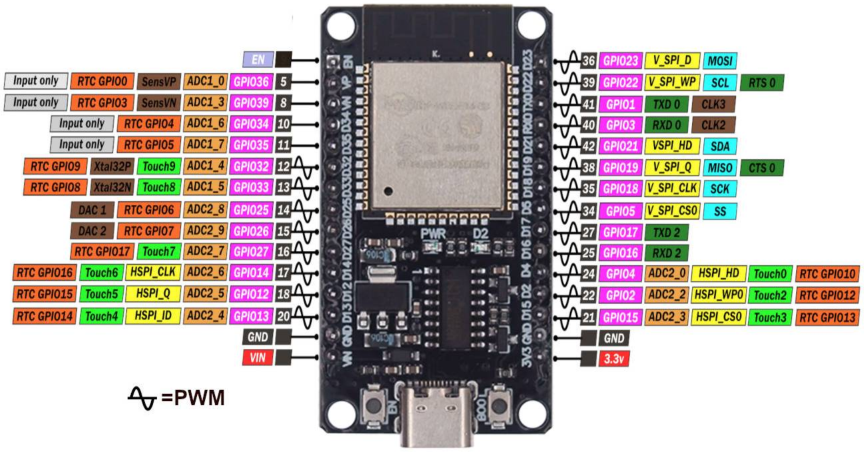
Part I

Telemetry Box

1 ESP32 Board

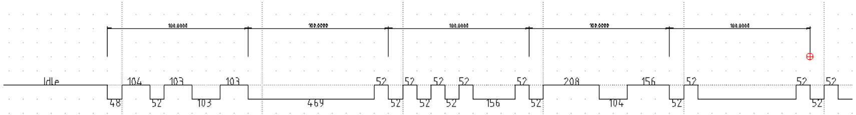
Telemetry box uses ESP32 board.

- VIN: 3.3-12V gives very wide voltage range.
- GPIO13: Connected to Hobbywing ESC for telemetry signal
- GPIO12: Used to receive SmartPort data. Connect it to model receiver
- GPIO14: Used to transmit SmartPort data. Use 120 ohm resistor and connect with Rx pin (GPIO12).


EN
Input only RTC GPIO0 SensVP ADC1_0 GPIO36 5
Input only RTC GPIO3 SensVN ADC1_3 GPIO39 8
Input only RTC GPIO4 ADC1_6 GPIO34 10
Input only RTC GPIO5 ADC1_7 GPIO35 11
RTC GPIO9 Xtal32P Touch9 ADC1_4 GPIO32 12
RTC GPIO8 Xtal32N Touch8 ADC1_5 GPIO33 13
DAC 1 RTC GPIO6 ADC2_8 GPIO25 14
DAC 2 RTC GPIO7 ADC2_9 GPIO26 15
RTC GPIO17 Touch7 ADC2_7 GPIO27 16
RTC GPIO16 Touch6 HSPI_CLK ADC2_6 GPIO14 17
RTC GPIO15 Touch5 HSPI_Q ADC2_5 GPIO12 18
RTC GPIO14 Touch4 HSPI_ID ADC2_4 GPIO13 20
GND
VIN
=PWM
EN VP VN D34 D35 D32 D33 D25 D26 D27 D14 D12 D13 GND VIN
D23 D22 TXD0 RXD0 D21 D19 D18 D5 D17 D16 D4 D2 D15 GND 3V3
PWR
D2
EN
BOOT
36 GPIO23 V_SPI_D MOSI
39 GPIO22 V_SPI_WP SCL RTS 0
41 GPIO1 TXD 0 CLK3
40 GPIO3 RXD 0 CLK2
42 GPIO21 VSPI_HD SDA
38 GPIO19 V_SPI_Q MISO CTS 0
35 GPIO18 V_SPI_CLK SCK
34 GPIO5 V_SPI_CS0 SS
27 GPIO17 TXD 2
25 GPIO16 RXD 2
24 GPIO4 ADC2_0 HSPI_HD Touch0 RTC GPIO10
22 GPIO2 ADC2_2 HSPI_WP0 Touch2 RTC GPIO12
21 GPIO15 ADC2_3 HSPI_CS0 Touch3 RTC GPIO13
GND
3.3v

2 Hobbywing ESC

2.1 Communication



- Speed: 19200 bps
- Parity bit: No

- Stop bit: 1
- Logic: Normal
- Big endian coding
- bit duration: 52 us
- two types of frames sent: data and signature
  - Data frame: 20 bytes
  - Hobbywing signature frame: ? bytes

2.2 Data frames

Length equal to 20 bytes.
0: Start byte 0x9B
1-3: Frame number
4-5: (0-100%). To calculate throttle percentage divide this value by 10.0
6-7: (0-100%) Real motor output. To calculate percentage divide by 10.0
8-10: data
11-12: data. To calculate real voltage divide this value by ~113,29. So for example value 2830 will be 24.98V (/113,29)
13-14: data
15-16:
17-18: Temperature 2
19: End byte 0xB9

	Start	Frame number	Rx throttle 0-1000	Output PWM 0-1000	RPM raw	Voltage raw
Byte	0	1-3	4-5	6-7	8-10	11-12
	9B	00 07 DC	01 2C	00 00	00 00 00	0A 49
	9B	00 07 DD	01 2C	00 00	00 00 00	0A 4A
	9B	00 07 DE	01 2C	00 00	00 00 00	0A 49
	9B	00 07 DF	01 2C	00 00	00 00 00	0A 4A
	9B	00 07 E0	01 2C	00 00	00 00 00	0A 48
	9B	00 07 E1	01 2C	00 00	00 00 00	0A 4C
	9B	00 07 E2	01 2C	00 00	00 00 00	0A 48
	9B	00 07 E3	01 2C	00 00	00 00 00	0A 48
	9B	00 07 E4	01 2C	00 00	00 00 00	0A 4B
	9B	00 07 E5	01 2C	00 00	00 00 00	0A 48
	9B	00 07 E6	01 2C	00 00	00 00 00	0A 49
	9B	00 07 E7	01 2C	00 00	00 00 00	0A 48

2.3 Signature frames

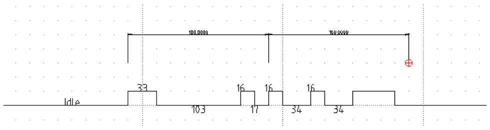
Length equal to 13 bytes. But as my software starts frame recognition from 0x9b, it sees it as 12 bytes frame.
0-1 or 0: Start byte 0x9B

1-10: Signature
11: End byte 0xB9

	Start	Start									
Byte	-1	0	1	2	3	4	5	6	7	8	9
V4LV/25/60/80A	0x9b	0x9b	0x03	0xe8	0x01	0x08	0x5b	0x00	0x01	0x00	0x21
V4HV200A OPTO	0x9b	0x9b	0x03	0xe8	0x01	0x02	0x0d	0x0a	0x3d	0x05	0x1e
V5HV130A OPTO	0x9b	0x9b	0x03	0xe8	0x01	0x0b	0x41	0x21	0x44	0xb9	0x21
HW HV 200A	0x9b	0x9b	0x02	0xd0	0x01	0x0b	0x41	0x21	0x7e	0x62	0x21
HW 120A	0x9b	0x9b	0x03	0xe8	0x01	0x08	0x5b	0x21	0x71	0x6e	0x21

3 FrSky Smart Port

3.1 Communication



- Speed: 57600 bps
- Parity bit: No
- Stop bit: 1
- Logic: Inverted
- Little endian coding

FrSky receiver asks for sensor with two bytes.

	Start	Sensor ID
Byte	0	1
Value	0x7e	ID

Pooled sensor IDs:

- 6A CB AC 0D 8E 2F D0 71 F2 53 34 95 16 B7 98 39 BA 1B 00 A1 22 83 E4 45 C6 67 48 E9

If sensor is present - it answers with:

	Head	Sensor type		Value				CRC
Byte	0	1	2	3	4	5	6	7
	0x10							
FLVV Cell sensor for 2S LiPo	0x10	0x00	0x03	0x20	0x2C	0xC8	0x82	
A3 Voltage sensor	0x10	0x00	0x90	Voltage * 100.0				

Known sensor types:

- RPM: 0x0500
- A3 (Voltage): 0x0900

- A4: 0x0910
- Current: 0x0200
- T1: 0x0400
- T2: 0x0410
- FLVV Cell sensor: 0x0300
  - 0x20 means:
    - Total numbers of cells is 2
    - Currently sent ID is 0
    - For 4S battery there will be frames with ID=0 (Cell0=C0 and Cell1=C1) and ID=2 (Cell0 refers to C2 and Cell1 refers to C3)
  - Two cells voltages are send using 24 + 24 bits.
    - Cell0: 0x82c -> when read as float it refers to 4.2V
    - Cell1: 0x82c
    - Transmitter will sum up all cells and show summed voltage

3.1.1 CRC calculation

It is not a simple sumup. Below is correct method:

```
uint8_t calcCrc(const uint8_t* buf, size_t len) {
    short crc = 0;
      for(int i=0;i<len;i++) {
        crc += buf[i]; //0-1FF
        crc += crc >> 8; //0-100
        crc &= 0x00ff;
        crc += crc >> 8; //0-0FF
        crc &= 0x00ff;
        }
  return ~crc;
}
```