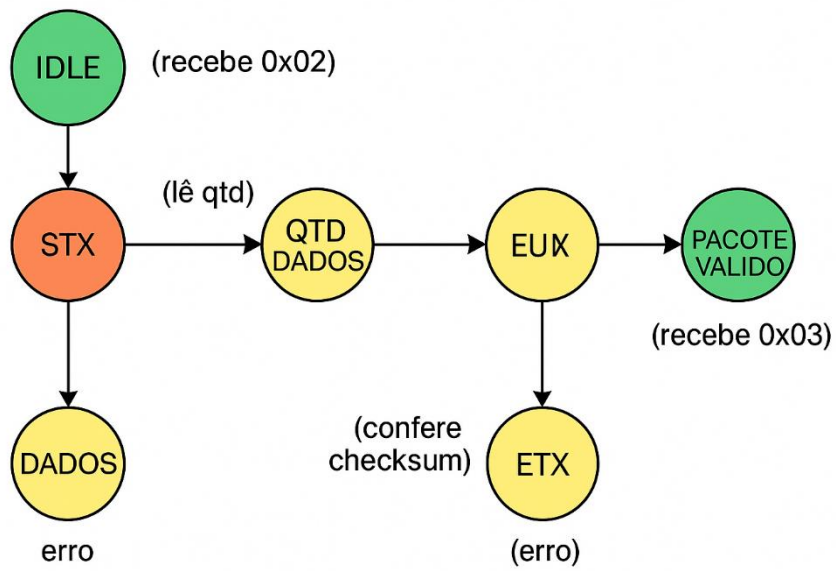


PROJETO DE SISTEMAS EMBARCADOS

26/08/2025

RAVI DE FARIAS

DIAGRAMA DE FUNCIONAMENTO DO SISTEMA:



CODIGO RECEPTOR:

```
1 #include <stdio.h>
2 #include <stdint.h>
3
4 #define STX 0x02
5 #define ETX 0x03
6
7 typedef enum {
8     ST_IDLE,
9     ST_STX,
10    ST_QTD,
11    ST_DADOS,
12    ST_CHK,
13    ST_ETX,
14    ST_OK,
15    ST_ERROR
16 } State;
17
18 static State fsm_state = ST_IDLE;
19
20 static uint8_t buffer[256];
21 static uint8_t qtd_dados = 0;
22 static uint8_t dados_count = 0;
23 static uint8_t checksum_calc = 0;
24 static uint8_t checksum_rx = 0;
25
26 void reset_fsm() {
27     fsm_state = ST_IDLE;
28     qtd_dados = 0;
29 }
30
31 Transmitindo bytes...
32
33 ...Program finished with exit code 0
34 Press ENTER to exit console.
```

```
5
6 void reset_fsm() {
7     fsm_state = ST_IDLE;
8     qtd_dados = 0;
9     dados_count = 0;
10    checksum_calc = 0;
11    checksum_rx = 0;
12 }
13
14 void fsm_receiver(uint8_t byte) {
15     switch(fsm_state) {
16     case ST_IDLE:
17         if (byte == STX) {
18             fsm_state = ST_STX;
19         }
20         break;
21
22     case ST_STX:
23         qtd_dados = byte;
24         dados_count = 0;
25         checksum_calc = 0;
26         fsm_state = ST_QTD;
27         break;
28
29     case ST_QTD:
30         buffer[dados_count++] = byte;
31         checksum_calc ^= byte;
32         if (dados_count == qtd_dados) {
33             fsm_state = ST_CHK;
34         }
35     }
36 }
37
38 Transmitindo bytes...
39
40 Program finished with exit code 0
41 Press ENTER to exit console.
```

```
8
9     case ST_QTD:
10         buffer[dados_count++] = byte;
11         checksum_calc ^= byte;
12         if (dados_count >= qtd_dados) {
13             fsm_state = ST_DADOS;
14         }
15         break;
16
17     case ST_DADOS:
18         checksum_rx = byte;
19         if (checksum_rx == checksum_calc) {
20             fsm_state = ST_CHK;
21         } else {
22             fsm_state = ST_ERROR;
23         }
24         break;
25
26     case ST_CHK:
27         if (byte == ETX) {
28             fsm_state = ST_OK;
29         } else {
30             fsm_state = ST_ERROR;
31         }
32         break;
33
34     case ST_OK:
35         printf("Pacote recebido com sucesso! Dados: ");
36         transmitindo_bytes();
37
38 Program finished with exit code 0
39 Press ENTER to exit console.
```

```
71     }
72     break;
73
74     case ST_OK:
75         printf("Pacote recebido com sucesso! Dados: ");
76         for (int i = 0; i < qtd_dados; i++) {
77             printf("%02X ", buffer[i]);
78         }
79         printf("\n");
80         reset_fsm();
81         break;
82
83     case ST_ERROR:
84         printf("Erro no pacote!\n");
85         reset_fsm();
86         break;
87 }
88
89
90 int main() {
91     uint8_t pacote[] = {0x02, 0x03, 0x10, 0x20, 0x30, (0x10^0x20^0x30), 0x03};
92
93     printf("Transmitindo bytes...\n");
94     for (int i = 0; i < sizeof(pacote); i++) {
95         fsm_receiver(pacote[i]);
96     }
97
98     input
99
100 transmitindo bytes...
101
102 Program finished with exit code 0
103 Press ENTER to exit console.
```

CÓDIGO TRANSMISSOR:

```
1 #include <stdio.h>
2 #include <stdint.h>
3
4 #define STX 0x02
5 #define ETX 0x03
6
7 uint8_t calcula_checksum(uint8_t *dados, uint8_t qtd) {
8     uint8_t chk = 0;
9     for (int i = 0; i < qtd; i++) {
10         chk ^= dados[i];
11     }
12     return chk;
13 }
14
15 void transmissor_send(uint8_t *dados, uint8_t qtd) {
16     uint8_t chk = calcula_checksum(dados, qtd);
17
18     printf("Enviando pacote:\n");
19
20     printf("%02X ", STX);
21
22     printf("%02X ", qtd);
23
24     for (int i = 0; i < qtd; i++) {
25         printf("%02X ", dados[i]);
26     }
27
28     printf("%02X ", chk);
29
30     printf("%02X\n", ETX);
31 }
```

```
Enviando pacote:
02 02 AB CD 66 03

...Program finished with exit code 0
Press ENTER to exit console.
```

```
15 void transmissor_send(uint8_t *dados, uint8_t qtd) {
16     uint8_t chk = calcula_checksum(dados, qtd);
17
18     printf("Enviando pacote:\n");
19
20     printf("%02X ", STX);
21
22     printf("%02X ", qtd);
23
24     for (int i = 0; i < qtd; i++) {
25         printf("%02X ", dados[i]);
26     }
27
28     printf("%02X ", chk);
29
30     printf("%02X\n", ETX);
31 }
32
33 int main() {
34     uint8_t dados1[] = {0x10, 0x20, 0x30};
35     uint8_t dados2[] = {0xAB, 0xCD};
36
37     transmissor_send(dados1, sizeof(dados1));
38     transmissor_send(dados2, sizeof(dados2));
39
40     return 0;
41 }
42
43
```

```
Enviando pacote:
02 02 AB CD 66 03

...Program finished with exit code 0
Press ENTER to exit console.
```

TRANSMISSOR + RECEPTOR:

```
#include <stdio.h>
#include <stdint.h>

#define STX 0x02
#define ETX 0x03

/* ----- RECEPTOR ----- */
typedef enum {
    RX_IDLE,
    RX_WAIT_QTD,
    RX_READ_DATA,
    RX_WAIT_CHK,
    RX_WAIT_ETX
} RxState;

static RxState rx_state = RX_IDLE;

static uint8_t rx_buf[256];
static uint8_t rx_expected = 0;
static uint8_t rx_len = 0;
static uint8_t rx_chk = 0;

static void rx_reset(void) {
    rx_state = RX_IDLE;
    rx_expected = 0;
    rx_len = 0;
    rx_chk = 0;
}

static void rx_byte(uint8_t b) {
    switch (rx_state) {
        case RX_IDLE:
            if (b == STX) {
                rx_state = RX_WAIT_QTD;
            }
            break;

        case RX_WAIT_QTD:
            rx_expected = b;
            rx_len = 0;
            rx_chk = 0;
            rx_state = (rx_expected > 0) ? RX_READ_DATA : RX_WAIT_CHK;
            break;

        case RX_READ_DATA:
            if (rx_len < sizeof(rx_buf)) {
                rx_buf[rx_len++] = b;
                rx_chk ^= b;
            }
            if (rx_len >= rx_expected) {
                rx_state = RX_WAIT_CHK;
            }
    }
}
```

```

        rx_state = RX_WAIT_CHK;
    }
    break;

case RX_WAIT_CHK:
    if (b == rx_chk) {
        rx_state = RX_WAIT_ETX;
    } else {
        printf("ERRO: checksum invalido\n");
        rx_reset();
    }
    break;

case RX_WAIT_ETX:
    if (b == ETX) {
        printf("OK: pacote recebido. Dados: ");
        for (uint8_t i = 0; i < rx_expected; i++) {
            printf("%02X ", rx_buf[i]);
        }
        printf("\n");
    } else {
        printf("ERRO: ETX invalido\n");
    }
    rx_reset();
    break;
}
}

```

```

3     }
4     rx_reset();
5     break;
6 }
7 }
8
9  /* ----- TRANSMISSOR ----- */
10 static uint8_t tx_calc_chk(const uint8_t *data, uint8_t n) {
11     uint8_t c = 0;
12     for (uint8_t i = 0; i < n; i++) {
13         c ^= data[i];
14     }
15     return c;
16 }
17
18 static void tx_send(const uint8_t *data, uint8_t n) {
19     uint8_t chk = tx_calc_chk(data, n);
20
21     rx_byte(STX);
22     rx_byte(n);
23     for (uint8_t i = 0; i < n; i++) {
24         rx_byte(data[i]);
25     }
26     rx_byte(chk);
27     rx_byte(ETX);
28 }
29

```

```

92     rx_byte(n);
93     for (uint8_t i = 0; i < n; i++) {
94         rx_byte(data[i]);
95     }
96     rx_byte(chk);
97     rx_byte(ETX);
98 }
99
100 int main(void) {
101     const uint8_t p1[] = {0x10, 0x20, 0x30};
102     const uint8_t p2[] = {0xAB, 0xCD};
103
104     tx_send(p1, (uint8_t)sizeof p1);
105     tx_send(p2, (uint8_t)sizeof p2);
106
107     const uint8_t bad[] = {0x01, 0x02, 0x03};
108     uint8_t chk_bad = tx_calc_chk(bad, (uint8_t)sizeof bad) ^ 0xFF;
109     rx_byte(STX);
110     rx_byte((uint8_t)sizeof bad);
111     for (uint8_t i = 0; i < (uint8_t)sizeof bad; i++) rx_byte(bad[i]);
112     rx_byte(chk_bad);
113     rx_byte(ETX);
114
115     return 0;
116 }
117
118

```

input

```

OK: pacote recebido. Dados: 10 20 30
OK: pacote recebido. Dados: AB CD
ERRO: checksum invalido

...Program finished with exit code 0
Press ENTER to exit console.

```