Проектирование устройств

Домашнее задание №5

Евгений Зеленин

30 ноября 2024 г.

1. Шина I^2C

Условия задачи. *Если у вас есть датчик, поддерживающий I2C, подключите его и воспользуйтесь программой из методички. Для удобства можно выводить каждый шаг программы через Serial.print(). Так будет больше понимания работы библиотеки Wire.h. Если вы являетесь обладателем нескольких датчиков, подключите их и обратитесь к каждому по адресу.*

Описание эксперимента

Объединим с помощью I^2C следующие устройства: два Aruino UNO, датчик температуры и влажности HTU21D и датчик освещенности и приближения APDS-9930. Суть взаимодействия в следующем: первая Arduino UNO опрашивает все устройства в сети, подключается к датчикам и снимает с них показания. Затем, упаковывает эти данные в структуру, вычисляет CRC8 и передает их во вторую Arduino UNO, которая проверят CRC и выводит эти данные в терминал. Адрес ведомой Arduino UNO - 01. Код сканера из лекции доработан и теперь он находит все устройства с адресами от 1 до 127 на шине I2C.

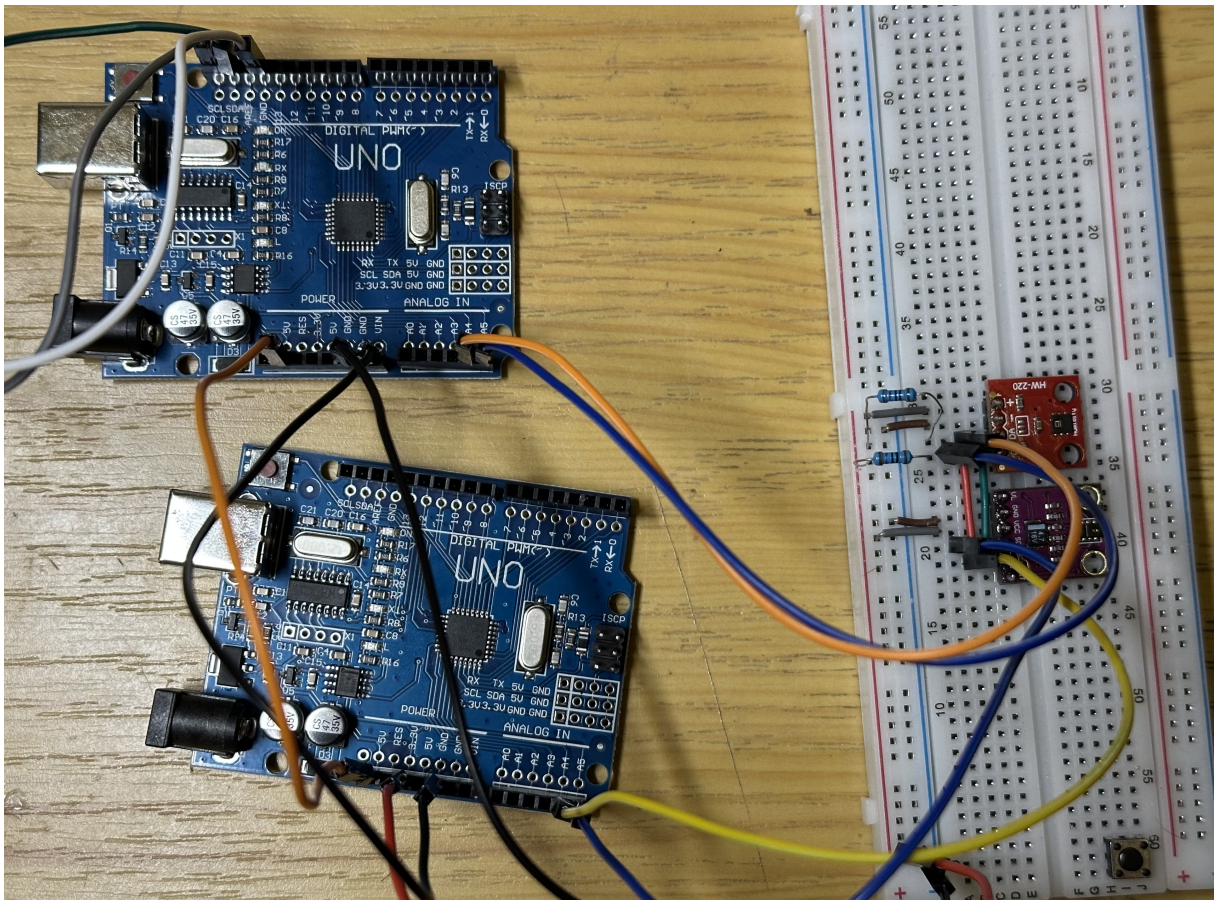


Рис. 1: Схема подключения

```
#include <GyverHTU21D.h>
#include <APDS9930.h>
//задаем адрес второй Arduino UNO
#define dev_id 1
GyverHTU21D htu;
APDS9930 apds = APDS9930();

//определяем новый тип данных - структура для передачи показаний датчиков
typedef struct sensor_data_t {
  int32_t time;
  float light;
  float hum;
  float temp;
  byte crc8;
};

const int data_size = sizeof(sensor_data_t); //определяем размер структуры

//для удобного доступа к структуре по байтам создаем новый тип данных через объедниение
typedef union data_packet_t {
  sensor_data_t data;
  byte byte_a[data_size];
};

//объявляем структуру
data_packet_t sensors;

//переменные для отсчета времени
uint32_t msr_time = 0;
uint32_t send_time = 0;
uint32_t cur_time = 0;

byte CRC8(const byte *data, byte len);
void scan_devices();

void setup() {
  Serial.begin(9600);
  //инициализация HTU21D
  if (!htu.begin()) {
    Serial.println(F("HTU21D error"));
  }

  //инициализация APDS-9930
  if (!apds.init()) {
    Serial.println(F("APDS ERROR!"));
  }  // Иначе, выводим сообщение об ошибке инициализации датчика
```

```
  if (!apds.enableLightSensor(false)) {
    Serial.println(F("Start light sensor ERROR!"));
  }  // Иначе, выводим сообщение об ошибке запуска режима определения освещённости
  htu.readTick();
  scan_devices();  //сканируем устройства
}

void loop() {
  cur_time = millis();
  //считываем показания датчиков раз в 500мс
  if (cur_time - msr_time > 500) {
    msr_time = cur_time;
    // функция опрашивает датчик
    if (htu.readTick()) {
      sensors.data.hum = htu.getHumidity();
      sensors.data.temp = htu.getTemperature();
    }
    if (!apds.readAmbientLightLux(sensors.data.light)) {
      Serial.println(F("Read light ERROR!"));
    }
  }

  cur_time = millis();
  //осуществляем передачу данных 1 раз в 1000мс
  if (cur_time - send_time > 1000) {
    send_time = cur_time;
    Wire.beginTransmission(dev_id);

    /* альтернативный вариант передачи через массив
    for (uint8_t i = 0; i < data_size; i++) {
      Wire.write(sensors.byte_a[i]);
    }
    */

    Wire.write(sensors.byte_a, data_size);

    int err = Wire.endTransmission();
    if (err) {
      Serial.println((String)err + F(" ERROR SEND FAILED"));
    }
  }
}

void scan_devices() {
  byte error, address;
  int nDevices;
  Serial.println(F("\nScanning I2C:"));
```

```
  nDevices = 0;
  //всего на шине I2C может быть до 127 устройств
  for (address = 1; address < 127; address++) {
    Wire.beginTransmission(address);
    error = Wire.endTransmission();
    switch (error) {

      case 0:
        Serial.print(F("I2C device found at address 0x"));
        if (address < 16)
          Serial.print(F("0"));
        Serial.println(address, HEX);
        nDevices++;
        break;

      case 4:
        Serial.print(F("Unknown error at address 0x"));
        if (address < 16)
          Serial.print(F("0"));
        Serial.println(address, HEX);
        break;
    }
  }
  if (nDevices == 0)
    Serial.println(F("No I2C devices found"));
  else
    Serial.println(F("Done"));
}

/*
  Name  : CRC-8
  Poly  : 0x31    x^8 + x^5 + x^4 + 1
  Init  : 0xFF
  Revert: false
  XorOut: 0x00
  Check : 0xF7 ("123456789")
  MaxLen: 15 байт (127 бит) - обнаружение
    одинарных, двойных, тройных и всех нечетных ошибок
*/

const unsigned char Crc8Table[256] = {
    0x00, 0x31, 0x62, 0x53, 0xC4, 0xF5, 0xA6, 0x97,
    0xB9, 0x88, 0xDB, 0xEA, 0x7D, 0x4C, 0x1F, 0x2E,
    0x43, 0x72, 0x21, 0x10, 0x87, 0xB6, 0xE5, 0xD4,
    0xFA, 0xCB, 0x98, 0xA9, 0x3E, 0x0F, 0x5C, 0x6D,
    0x86, 0xB7, 0xE4, 0xD5, 0x42, 0x73, 0x20, 0x11,
    0x3F, 0x0E, 0x5D, 0x6C, 0xFB, 0xCA, 0x99, 0xA8,
```

```
    0xC5, 0xF4, 0xA7, 0x96, 0x01, 0x30, 0x63, 0x52,
    0x7C, 0x4D, 0x1E, 0x2F, 0xB8, 0x89, 0xDA, 0xEB,
    0x3D, 0x0C, 0x5F, 0x6E, 0xF9, 0xC8, 0x9B, 0xAA,
    0x84, 0xB5, 0xE6, 0xD7, 0x40, 0x71, 0x22, 0x13,
    0x7E, 0x4F, 0x1C, 0x2D, 0xBA, 0x8B, 0xD8, 0xE9,
    0xC7, 0xF6, 0xA5, 0x94, 0x03, 0x32, 0x61, 0x50,
    0xBB, 0x8A, 0xD9, 0xE8, 0x7F, 0x4E, 0x1D, 0x2C,
    0x02, 0x33, 0x60, 0x51, 0xC6, 0xF7, 0xA4, 0x95,
    0xF8, 0xC9, 0x9A, 0xAB, 0x3C, 0x0D, 0x5E, 0x6F,
    0x41, 0x70, 0x23, 0x12, 0x85, 0xB4, 0xE7, 0xD6,
    0x7A, 0x4B, 0x18, 0x29, 0xBE, 0x8F, 0xDC, 0xED,
    0xC3, 0xF2, 0xA1, 0x90, 0x07, 0x36, 0x65, 0x54,
    0x39, 0x08, 0x5B, 0x6A, 0xFD, 0xCC, 0x9F, 0xAE,
    0x80, 0xB1, 0xE2, 0xD3, 0x44, 0x75, 0x26, 0x17,
    0xFC, 0xCD, 0x9E, 0xAF, 0x38, 0x09, 0x5A, 0x6B,
    0x45, 0x74, 0x27, 0x16, 0x81, 0xB0, 0xE3, 0xD2,
    0xBF, 0x8E, 0xDD, 0xEC, 0x7B, 0x4A, 0x19, 0x28,
    0x06, 0x37, 0x64, 0x55, 0xC2, 0xF3, 0xA0, 0x91,
    0x47, 0x76, 0x25, 0x14, 0x83, 0xB2, 0xE1, 0xD0,
    0xFE, 0xCF, 0x9C, 0xAD, 0x3A, 0x0B, 0x58, 0x69,
    0x04, 0x35, 0x66, 0x57, 0xC0, 0xF1, 0xA2, 0x93,
    0xBD, 0x8C, 0xDF, 0xEE, 0x79, 0x48, 0x1B, 0x2A,
    0xC1, 0xF0, 0xA3, 0x92, 0x05, 0x34, 0x67, 0x56,
    0x78, 0x49, 0x1A, 0x2B, 0xBC, 0x8D, 0xDE, 0xEF,
    0x82, 0xB3, 0xE0, 0xD1, 0x46, 0x77, 0x24, 0x15,
    0x3B, 0x0A, 0x59, 0x68, 0xFF, 0xCE, 0x9D, 0xAC
};

unsigned char CRC8(unsigned char *pcBlock, unsigned char len)
{
    unsigned char crc = 0xFF;

    while (len--)
        crc = Crc8Table[crc ^ *pcBlock++];

    return crc;
}
```

На ведомом устройстве запустим следующий код:

```
  #include <Wire.h>
  //определяем новый тип для хранения показаний с датчиков
  typedef struct sensor_data_t {
    int32_t time;
    float light;  // Определяем переменную для хранения освещённости общей в люксах
    float hum;
    float temp;
```

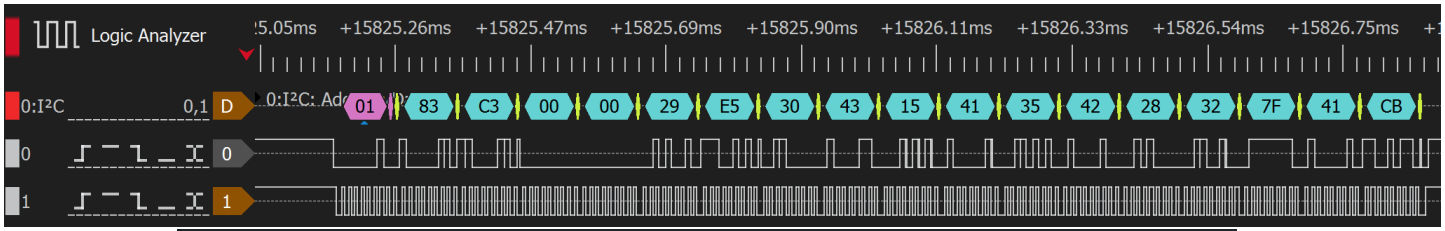
```
  byte crc8;
};

//определяем размер данных
const int data_size = sizeof(sensor_data_t);
//для доступа по байтам
typedef union data_packet_t {
  sensor_data_t data;
  byte byte_a[data_size];
};

data_packet_t sensors;

byte CRC8(const byte *data, byte len);
byte crc_res = 0;

//статус принятия данных
bool data_ok = 0;

void setup() {
  Wire.begin(1);  // подключиться к шине i2c с адресом #4
  Wire.onReceive(receiveEvent); //подключаем функцию для обработки события
  Serial.begin(9600);
}
void loop() {
  //если данные получены - вывод в терминал
  if (data_ok) {
    Serial.println("\n------ Recieved data: -------");
    Serial.println((String) "Time = " + sensors.data.time + "ms - "
    + sizeof(sensors.data.time) + " bytes");
    Serial.println((String) "Temp = " + sensors.data.temp + "C - "
    + sizeof(sensors.data.temp) + " bytes");
    Serial.println((String) "Hum = " + sensors.data.hum + "% - "
    + sizeof(sensors.data.hum) + " bytes");
    Serial.println((String) "Light = " + sensors.data.light + " lux - "
    + sizeof(sensors.data.light) + " bytes");
    Serial.print((String) "CRC8 = ");
    Serial.println(sensors.data.crc8, HEX);

    Serial.println(F("-->> HEX <<--"));
    //вывод значений в 16-ричном виде по байтам
    for (int i = 0; i < data_size; i++) {
      Serial.print(sensors.byte_a[i], HEX);
      Serial.print(" ");
    }
    Serial.println();
```

```
    data_ok = 0;
  }
  delay(500);
}

// прием данных
void receiveEvent(int howMany) {
  //считываем побайтово пока есть данные и буфер не заполнен
  for (uint8_t i = 0; Wire.available() && i < data_size; i++) {
    sensors.byte_a[i] = Wire.read();
  }
  //альтернативный способ чтения байтов
  //Wire.readBytes((byte*)&sensor_data, sizeof(sensor_data));

  //вычисление и проверка CRC8
  if (!CRC8(sensors.byte_a, data_size)) {
    data_ok = 1;
  } else {
    Serial.println((String) "CRC ERROR! " + sensors.data.crc8 + " != " + crc_res);
    data_ok = 0;
  }
}

/*
  Name  : CRC-8
  Poly  : 0x31    x^8 + x^5 + x^4 + 1
  Init  : 0xFF
  Revert: false
  XorOut: 0x00
  Check : 0xF7 ("123456789")
  MaxLen: 15 байт (127 бит) - обнаружение
    одинарных, двойных, тройных и всех нечетных ошибок
*/

const unsigned char Crc8Table[256] = {
  0x00, 0x31, 0x62, 0x53, 0xC4, 0xF5, 0xA6, 0x97,
  0xB9, 0x88, 0xDB, 0xEA, 0x7D, 0x4C, 0x1F, 0x2E,
  0x43, 0x72, 0x21, 0x10, 0x87, 0xB6, 0xE5, 0xD4,
  0xFA, 0xCB, 0x98, 0xA9, 0x3E, 0x0F, 0x5C, 0x6D,
  0x86, 0xB7, 0xE4, 0xD5, 0x42, 0x73, 0x20, 0x11,
  0x3F, 0x0E, 0x5D, 0x6C, 0xFB, 0xCA, 0x99, 0xA8,
  0xC5, 0xF4, 0xA7, 0x96, 0x01, 0x30, 0x63, 0x52,
  0x7C, 0x4D, 0x1E, 0x2F, 0xB8, 0x89, 0xDA, 0xEB,
  0x3D, 0x0C, 0x5F, 0x6E, 0xF9, 0xC8, 0x9B, 0xAA,
  0x84, 0xB5, 0xE6, 0xD7, 0x40, 0x71, 0x22, 0x13,
  0x7E, 0x4F, 0x1C, 0x2D, 0xBA, 0x8B, 0xD8, 0xE9,
  0xC7, 0xF6, 0xA5, 0x94, 0x03, 0x32, 0x61, 0x50,
```

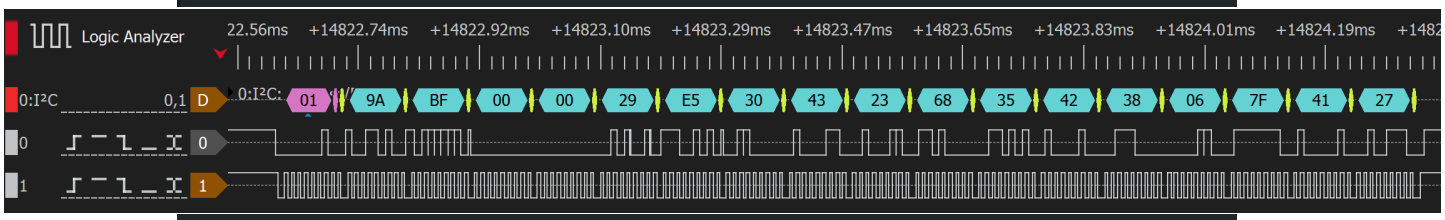
```
  0xBB, 0x8A, 0xD9, 0xE8, 0x7F, 0x4E, 0x1D, 0x2C,
  0x02, 0x33, 0x60, 0x51, 0xC6, 0xF7, 0xA4, 0x95,
  0xF8, 0xC9, 0x9A, 0xAB, 0x3C, 0x0D, 0x5E, 0x6F,
  0x41, 0x70, 0x23, 0x12, 0x85, 0xB4, 0xE7, 0xD6,
  0x7A, 0x4B, 0x18, 0x29, 0xBE, 0x8F, 0xDC, 0xED,
  0xC3, 0xF2, 0xA1, 0x90, 0x07, 0x36, 0x65, 0x54,
  0x39, 0x08, 0x5B, 0x6A, 0xFD, 0xCC, 0x9F, 0xAE,
  0x80, 0xB1, 0xE2, 0xD3, 0x44, 0x75, 0x26, 0x17,
  0xFC, 0xCD, 0x9E, 0xAF, 0x38, 0x09, 0x5A, 0x6B,
  0x45, 0x74, 0x27, 0x16, 0x81, 0xB0, 0xE3, 0xD2,
  0xBF, 0x8E, 0xDD, 0xEC, 0x7B, 0x4A, 0x19, 0x28,
  0x06, 0x37, 0x64, 0x55, 0xC2, 0xF3, 0xA0, 0x91,
  0x47, 0x76, 0x25, 0x14, 0x83, 0xB2, 0xE1, 0xD0,
  0xFE, 0xCF, 0x9C, 0xAD, 0x3A, 0x0B, 0x58, 0x69,
  0x04, 0x35, 0x66, 0x57, 0xC0, 0xF1, 0xA2, 0x93,
  0xBD, 0x8C, 0xDF, 0xEE, 0x79, 0x48, 0x1B, 0x2A,
  0xC1, 0xF0, 0xA3, 0x92, 0x05, 0x34, 0x67, 0x56,
  0x78, 0x49, 0x1A, 0x2B, 0xBC, 0x8D, 0xDE, 0xEF,
  0x82, 0xB3, 0xE0, 0xD1, 0x46, 0x77, 0x24, 0x15,
  0x3B, 0x0A, 0x59, 0x68, 0xFF, 0xCE, 0x9D, 0xAC
};

unsigned char CRC8(unsigned char *pcBlock, unsigned char len) {
  unsigned char crc = 0xFF;

  while (len--)
    crc = Crc8Table[crc ^ *pcBlock++];

  return crc;
}
```

Отдельно стоит отметить, что CRC помещается в последний байт последовательности. При подсчете контрольной суммы этот байт вычитается $CRC8(sensors.byte_a, data\_size - 1)$, а при проверке учитывается $CRC8(sensors.byte_a, data\_size)$, тогда в случае совпадения контрольной суммы, результат проверки CRC - 0.

Теперь разберемся с тем, какие байты в этой посылке за что отвечают и в каком порядке следуют.




```
------ Recieved data: -------
Time = 50051ms - 4 bytes
Temp = 15.95C - 4 bytes
Hum = 45.31% - 4 bytes
Light = 176.90 lux - 4 bytes
CRC8 = CB
-->> HEX <<--
83 C3 0 0 29 E5 30 43 15 41 35 42 28 32 7F 41 CB
```




```
------ Recieved data: -------
Time = 49050ms - 4 bytes
Temp = 15.94C - 4 bytes
Hum = 45.35% - 4 bytes
Light = 176.90 lux - 4 bytes
CRC8 = 27
-->> HEX <<--
9A BF 0 0 29 E5 30 43 23 68 35 42 38 6 7F 41 27
```

Рис. 2: Разбор пакетов данных

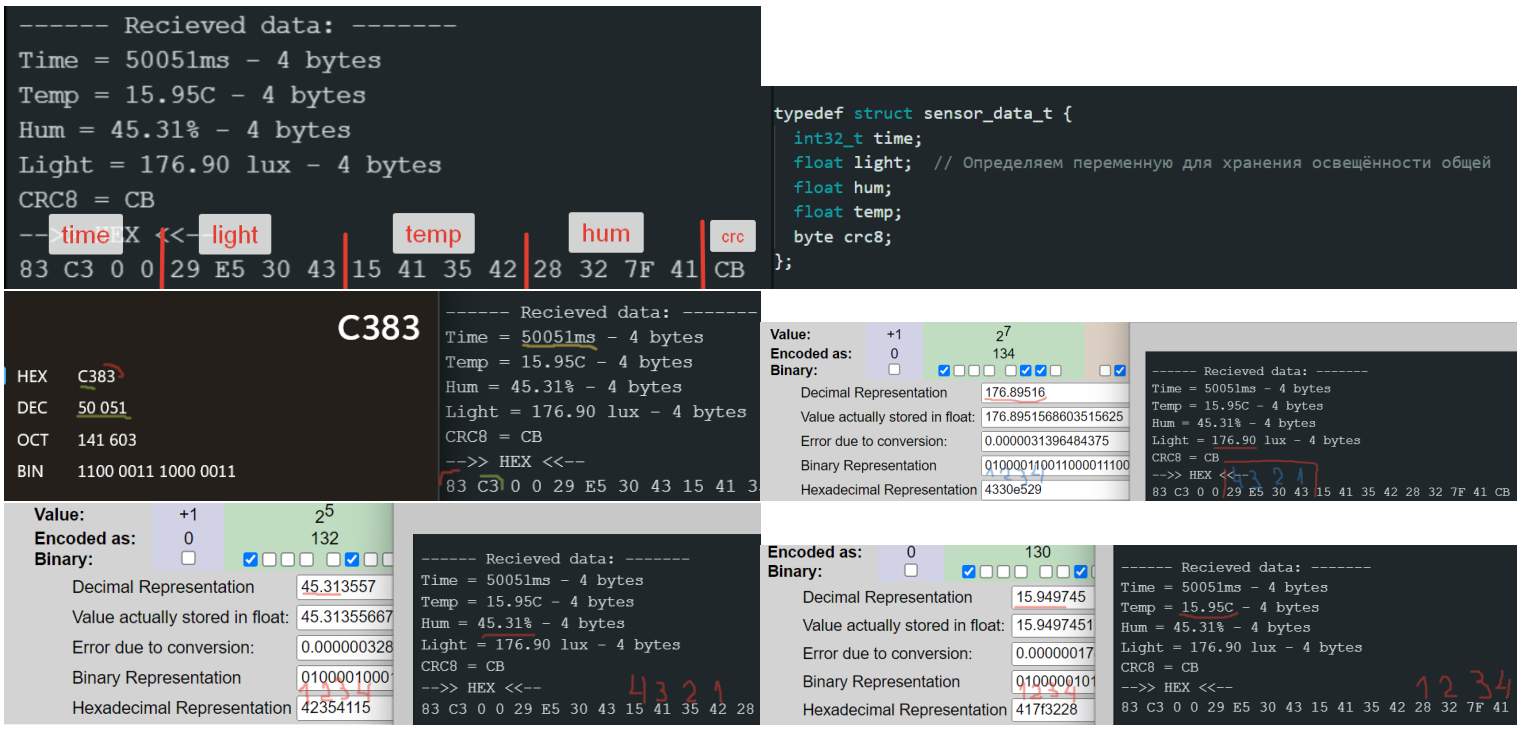


Рис. 3: Разбор пакета данных

Как видно из рисунка 3, поля данных передаются по направлению от старшего байта к младшему. В структуре первым идет поле time (время передачи), затем поля light, hum, temp, crc8. Поле time имеет 4-ех байтный тип uint32_t. Поля light, hum, temp имеют тип float он занимает 4 байта. Поле crc8 занимает 1 байт. Также, на рисунке показана расшифровка значений.

Отсоединим провод во время передачи данных и проверим, как среагирует устройство на ошибку в данных.

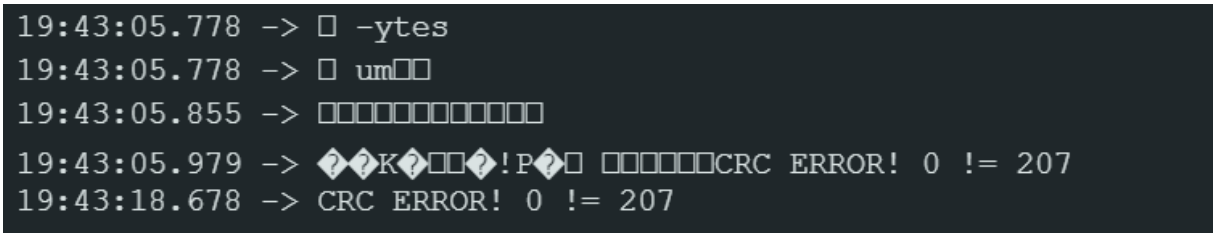


Рис. 4: Имитация ошибки

Как видно из рисунка 4, контрольная сумма не сошлась, данные не приняты.

2. Материалы к занятию

Дамп логического анализатора, вывод данных в терминал и прочие материалы к занятию расположены в папке на google диск по следующей ссылке: Материалы к ДЗ №5