```
/**
 * Copyright (c) 2015, Yaacov Zamir <kobi.zamir@gmail.com>
 * Copyright (c) 2017, Andrew Voznytsa <andrew.voznytsa@gmail.com>,
 *   FC_WRITE_REGISTER and FC_WRITE_MULTIPLE_COILS support
 *
 * Permission to use, copy, modify, and/or distribute this software for any
 * purpose with or without fee is hereby granted, provided that the above
 * copyright notice and this permission notice appear in all copies.
 *
 * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF  THIS SOFTWARE.
 */

#ifndef MODBUSSLAVE_H
#define MODBUSSLAVE_H
#include <Arduino.h>

#define MODBUS_MAX_BUFFER 256
#define MODBUS_INVALID_UNIT_ADDRESS 255
#define MODBUS_DEFAULT_UNIT_ADDRESS 1
#define MODBUS_CONTROL_PIN_NONE -1

/**
 * Modbus function codes
 */
enum {
  FC_INVALID = 0,
  FC_READ_COILS = 1,
  FC_READ_DISCRETE_INPUT = 2,
  FC_READ_HOLDING_REGISTERS = 3,
  FC_READ_INPUT_REGISTERS = 4,
  FC_WRITE_COIL = 5,
  FC_WRITE_REGISTER = 6,
  FC_READ_EXCEPTION_STATUS = 7,
  FC_WRITE_MULTIPLE_COILS = 15,
  FC_WRITE_MULTIPLE_REGISTERS = 16
};

enum {
  CB_MIN = 0,
  CB_READ_COILS = CB_MIN,
  CB_READ_DISCRETE_INPUTS,
  CB_READ_HOLDING_REGISTERS,
  CB_READ_INPUT_REGISTERS,
  CB_WRITE_COILS,
  CB_WRITE_HOLDING_REGISTERS,
  CB_READ_EXCEPTION_STATUS,
```

```
53    CB_MAX
54  };
55
56  enum {
57    COIL_OFF = 0x0000,
58    COIL_ON = 0xff00
59  };
60
61  enum {
62    STATUS_OK = 0,
63    STATUS_ILLEGAL_FUNCTION,
64    STATUS_ILLEGAL_DATA_ADDRESS,
65    STATUS_ILLEGAL_DATA_VALUE,
66    STATUS_SLAVE_DEVICE_FAILURE,
67    STATUS_ACKNOWLEDGE,
68    STATUS_SLAVE_DEVICE_BUSY,
69    STATUS_NEGATIVE_ACKNOWLEDGE,
70    STATUS_MEMORY_PARITY_ERROR,
71    STATUS_GATEWAY_PATH_UNAVAILABLE,
72    STATUS_GATEWAY_TARGET_DEVICE_FAILED_TO_RESPOND,
73  };
74
75  typedef uint8_t (*MobbusCallback)(uint8_t, uint16_t, uint16_t);
76
77  /**
78   * @class Modbus
79   */
80  class Modbus {
81  public:
82      Modbus(
83        uint8_t unitAddress = MODBUS_DEFAULT_UNIT_ADDRESS,
84        int transmissionControlPin = MODBUS_CONTROL_PIN_NONE);
85      Modbus(
86        Stream &serialStream,
87        uint8_t unitAddress = MODBUS_DEFAULT_UNIT_ADDRESS,
88        int transmissionControlPin = MODBUS_CONTROL_PIN_NONE);
89
90      void begin(uint64_t boudRate);
91      void setUnitAddress(uint8_t unitAddress);
92      uint8_t poll();
93
94      bool readCoilFromBuffer(int offset);
95      uint16_t readRegisterFromBuffer(int offset);
96      uint8_t writeExceptionStatusToBuffer(int offset, bool status);
97      uint8_t writeCoilToBuffer(int offset, bool state);
98      uint8_t writeDiscreteInputToBuffer(int offset, bool state);
99      uint8_t writeRegisterToBuffer(int offset, uint16_t value);
100     uint8_t writeStringToBuffer(int offset, uint8_t *str, uint8_t length);
101
102     uint8_t readFunctionCode();
103     uint8_t readUnitAddress();
104     bool isBroadcast();
```

```
105
106     uint64_t getTotalBytesSent();
107     uint64_t getTotalBytesReceived();
108
109     MobbusCallback cbVector[CB_MAX];
110 private:
111     Stream &_serialStream;
112     int _serialTransmissionBufferLength = SERIAL_TX_BUFFER_SIZE;
113     int _transmissionControlPin = MODBUS_CONTROL_PIN_NONE;
114     uint8_t _unitAddress = MODBUS_DEFAULT_UNIT_ADDRESS;
115
116     uint16_t _halfCharTimeInMicroSecond;
117     uint64_t _lastCommunicationTime;
118
119     uint8_t _requestBuffer[MODBUS_MAX_BUFFER];
120     uint16_t _requestBufferLength = 0;
121     bool _isRequestBufferReading = false;
122
123     uint8_t _responseBuffer[MODBUS_MAX_BUFFER];
124     uint16_t _responseBufferLength = 0;
125     bool _isResponseBufferWriting = false;
126     uint16_t _responseBufferWriteIndex = 0;
127
128     uint64_t _totalBytesSent = 0;
129     uint64_t _totalBytesReceived = 0;
130
131     bool readRequest();
132     bool validateRequest();
133     uint8_t createResponse();
134     uint8_t executeCallback(
135       uint8_t callbackIndex,
136       uint16_t address,
137       uint16_t length);
138     uint16_t writeResponse();
139     uint16_t reportException(uint8_t exceptionCode);
140     uint16_t calculateCRC(uint8_t *buffer, int length);
141 };
142 #endif
143
```