```
/*
    Modbus slave example.

    Control and Read Arduino I/Os using Modbus serial connection.

    This sketch show how to use the callback vector for reading and
    controleing Arduino I/Os.

    * Control digital pins mode using holding registers 0 .. 13.
    * Controls digital output pins as modbus coils.
    * Reads digital inputs state as discreet inputs.
    * Reads analog inputs as input registers.
    * Write and Read EEPROM as holding registers.

    The circuit: ( see: ./extras/ModbusSetch.pdf )
    * An Arduino.
    * 2 x LEDs, with 220 ohm resistors in series.
    * A switch connected to a digital input pin.
    * A potentiometer connected to an analog input pin.
    * A RS485 module (Optional) connected to RX/TX and a digital control pin.

    Created 8 12 2015
    By Yaacov Zamir

    https://github.com/yaacov/ArduinoModbusSlave

*/

#include <EEPROM.h>
#include <ModbusSlave.h>

/* slave id = 1, control-pin = 8, baud = 9600
 */
#define SLAVE_ID 1
#define CTRL_PIN 8
#define BAUDRATE 9600

#define PIN_MODE_INPUT 0
#define PIN_MODE_OUTPUT 1

/**
 *  Modbus object declaration.
 */
Modbus slave(SLAVE_ID, CTRL_PIN);

void setup() {
    uint16_t pinIndex;
    uint16_t eepromValue;

    /* set pins for mode.
     */
    for (pinIndex = 3; pinIndex < 14; pinIndex++) {
        // get one 16bit register from eeprom
        EEPROM.get(pinIndex * 2, eepromValue);

        // use the register value to set pin mode.
        switch (eepromValue) {
            case PIN_MODE_INPUT:
                pinMode(pinIndex, INPUT);
                break;
            case PIN_MODE_OUTPUT:
                pinMode(pinIndex, OUTPUT);
                break;
        }
    }

    // RS485 control pin must be output
    pinMode(CTRL_PIN, OUTPUT);

    /* register handler functions.
     * into the modbus slave callback vector.
     */
    slave.cbVector[CB_READ_COILS] = readDigital;
    slave.cbVector[CB_READ_DISCRETE_INPUTS] = readDigital;
    slave.cbVector[CB_WRITE_COILS] = writeDigitalOut;
    slave.cbVector[CB_READ_INPUT_REGISTERS] = readAnalogIn;
    slave.cbVector[CB_READ_HOLDING_REGISTERS] = readMemory;
    slave.cbVector[CB_WRITE_HOLDING_REGISTERS] = writeMemory;

    // set Serial and slave at baud 9600.
    Serial.begin( BAUDRATE );
    slave.begin( BAUDRATE );
}

void loop() {
    /* listen for modbus commands con serial port.
     *
     * on a request, handle the request.
     * if the request has a user handler function registered in cbVector.
     * call the user handler function.
     */
    slave.poll();
}

/**
 * Handel Read Input Status (FC=01/02)
 * write back the values from digital pins (input status).
 *
 * handler functions must return void and take:
 *      uint8_t  fc - function code.
 *      uint16_t address - first register/coil address.
 *      uint16_t length/status - length of data / coil status.
 */
uint8_t readDigital(uint8_t fc, uint16_t address, uint16_t length) {
    // read digital input
    for (int i = 0; i < length; i++) {
        // write one boolean (1 bit) to the response buffer.
        slave.writeCoilToBuffer(i, digitalRead(address + i));
    }

    return STATUS_OK;
}

/**
 * Handel Read Holding Registers (FC=03)
 * write back the values from eeprom (holding registers).
 */
uint8_t readMemory(uint8_t fc, uint16_t address, uint16_t length) {
    uint16_t value;

    // read program memory.
    for (int i = 0; i < length; i++) {
        EEPROM.get((address + i) * 2, value);

        // write uint16_t value to the response buffer.
        slave.writeRegisterToBuffer(i, value);
    }

    return STATUS_OK;
}

/**
 * Handel Read Input Registers (FC=04)
 * write back the values from analog in pins (input registers).
 */
uint8_t readAnalogIn(uint8_t fc, uint16_t address, uint16_t length) {
    // read analog input
    for (int i = 0; i < length; i++) {
        // write uint16_t value to the response buffer.
        slave.writeRegisterToBuffer(i, analogRead(address + i));
    }
}

/**
 * Handle Force Single Coil (FC=05) and Force Multiple Coils (FC=15)
 * set digital output pins (coils).
 */
uint8_t writeDigitalOut(uint8_t fc, uint16_t address, uint16_t length) {
    // set digital pin state(s).
    for (int i = 0; i < length; i++) {
        digitalWrite(address + i, slave.readCoilFromBuffer(i));
    }

    return STATUS_OK;
}

```

```
/**
 * Handle Write Holding Register(s) (FC=06, FC=16)
 * write data into eeprom.
 */
uint8_t writeMemory(uint8_t fc, uint16_t address, uint16_t length) {
    uint16_t value;
    uint16_t registerIndex;

    // write to eeprom.
    for (int i = 0; i < length; i++) {
        registerIndex = address + i;

        // get uint16_t value from the request buffer.
        value = slave.readRegisterFromBuffer(i);

        EEPROM.put(registerIndex * 2, value);

        /* if this register sets digital pins mode,
         * set the digital pins mode.
         */
        if (registerIndex > 2 && registerIndex < 14 && registerIndex != CTRL_PIN) {
            // use the register value to set pin mode.
            switch (value) {
                case PIN_MODE_INPUT:
                    pinMode(registerIndex, INPUT);
                    break;
                case PIN_MODE_OUTPUT:
                    pinMode(registerIndex, OUTPUT);
                    break;
            }
        }
    }

    return STATUS_OK;
}

```