```
//Add ArduinoModbusSlave-master.zip file from "Sketch>Include
Library>Add .ZIP Library"
#include <ModbusSlave.h>

/* Wiring diagram
 *
 *  Uno pin      RS-485 Breakout Board
 *    0(Rx)             TX
 *    1(Tx)             RX
 *    2                 RTS
*/

/*
    *** for more information ***
          www.usetechno.com
*/

/*   Uno  Pin       Modbus_Address        Read/Write
*    AI0   A0            40001                 R
*    AI1   A1            40002                 R
*    AI2   A2            40003                 R
*    AI3   A3            40004                 R
*    ***                  ***                  *
*    AO0   5             40005                 R/W
*    AO1   6             40006                 R/W
*/

#define SLAVE_ID 1           // The Modbus slave ID, change to the ID
you want to use.
#define RS485_CTRL_PIN 2     // Change to the pin the RE/DE pin of the
RS485 controller is connected to.
#define SERIAL_BAUDRATE 9600 // Change to the baudrate you want to use
for Modbus communication. Data Length = 8, Parity = None, Stop Bit = 1
or 2
#define SERIAL_PORT Serial   // Serial port to use for RS485
communication, change to the port you're using.

// Modbus object declaration
Modbus slave(SERIAL_PORT, SLAVE_ID, RS485_CTRL_PIN);
```

```
int16_t AI0 = 0, AI1 = 0, AI2 = 0, AI3 = 0;
int8_t AO0 = 0, AO1 = 0;

void setup()
{

// Set the analog input pins. (Range: 0~1023, 10 bits resolution )
    pinMode(A0, INPUT);
    pinMode(A1, INPUT);
    pinMode(A2, INPUT);
    pinMode(A3, INPUT);

// Set the analog Output pins. (Range: 0~255, 8 bits resolution)
// PWM pin must be selected
    pinMode(5, OUTPUT);
    pinMode(6, OUTPUT);


// Register functions to call when a certain function code is received.
   slave.cbVector[CB_READ_HOLDING_REGISTERS] = readMemory;
   slave.cbVector[CB_WRITE_HOLDING_REGISTERS] = writeMemory;


// Set the serial port and slave to the given baudrate.
   SERIAL_PORT.begin(SERIAL_BAUDRATE);
   slave.begin(SERIAL_BAUDRATE);

}



void loop()
{
    // Listen for modbus requests on the serial port.
    // When a request is received it's going to get validated.
    // And if there is a function registered to the received function
code, this function will be executed.
    slave.poll();
}



// Handle the function code Read Holding Registers (FC=03) and write
```

```
back the values from the Analog channels (holding registers).
uint8_t readMemory(uint8_t fc, uint16_t address, uint16_t length)
{
    if (address > 5 || (address + length) > 6){
       return STATUS_ILLEGAL_DATA_ADDRESS;
    }

    else{
        AI0 = analogRead(A0);
        AI1 = analogRead(A1);
        AI2 = analogRead(A2);
        AI3 = analogRead(A3);
        int16_t analog[] = {AI0, AI1, AI2, AI3, AO0, AO1};
        for (uint8_t i = address; i < address + length; ++i){
             slave.writeRegisterToBuffer(i - address, analog[i]);
          }

        return STATUS_OK;
    }
}


// Handle the function codes Write Holding Register(s) (FC=06, FC=16)
uint8_t writeMemory(uint8_t fc, uint16_t address, uint16_t length)
{
    if (address == 4) {
        if (length == 1){
           AO0 = slave.readRegisterFromBuffer(0);
            analogWrite(5,AO0);
            return STATUS_OK;
        }
        else if (length == 2){
           AO0 = slave.readRegisterFromBuffer(0);
           AO1 = slave.readRegisterFromBuffer(1);
            analogWrite(5,AO0);
            analogWrite(6,AO1);
            return STATUS_OK;
        }
        else{
           return STATUS_ILLEGAL_DATA_ADDRESS;
        }
    }
    else if (address == 5) {
        if (length == 1){
            AO1 = slave.readRegisterFromBuffer(0);
            analogWrite(6,AO1);
            return STATUS_OK;
        }
        else{
            return STATUS_ILLEGAL_DATA_ADDRESS;
        }
    }
    else{
        return STATUS_ILLEGAL_DATA_ADDRESS;
    }
}
```