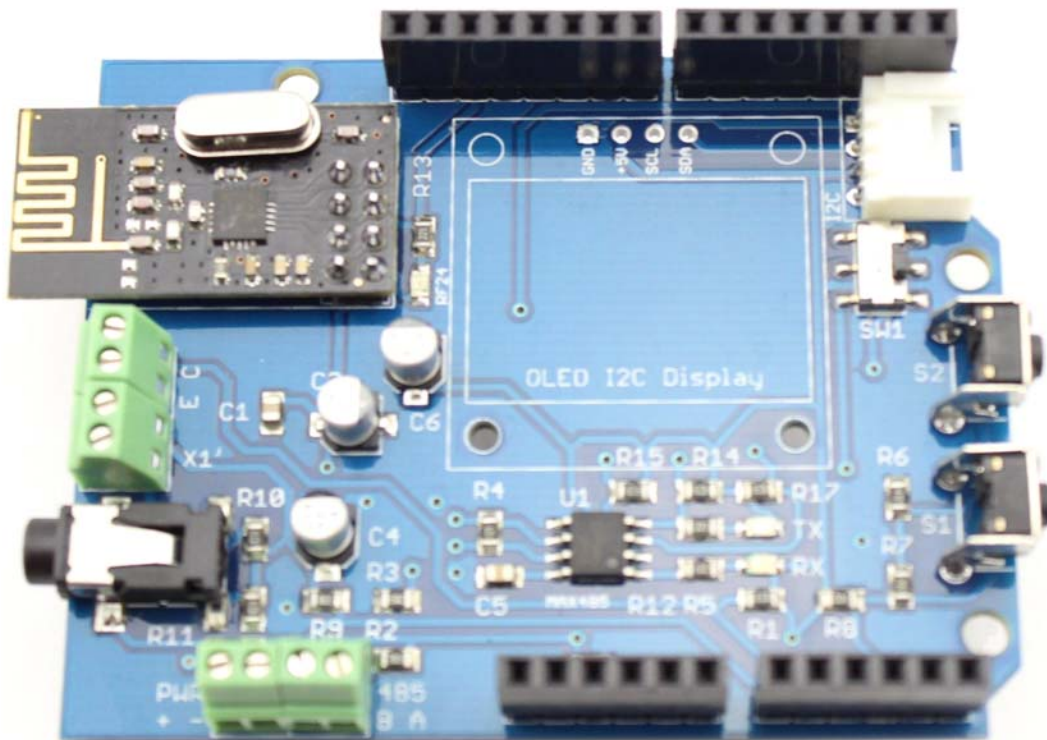
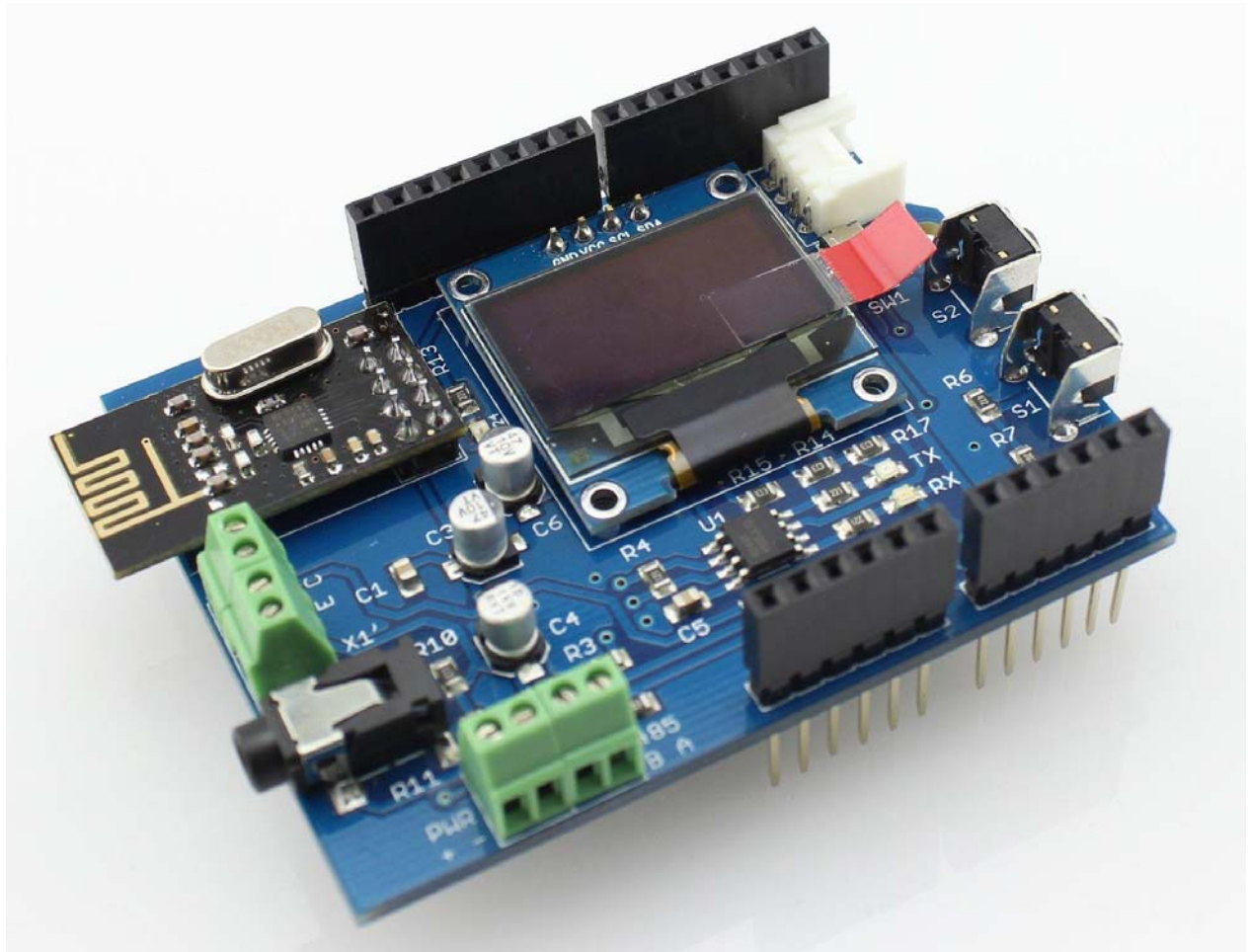


## Energy Monitor Shield V2



Energy Monitor Shield v2 is an Arduino-compatible expansion card designed for building energy monitoring system with support I2C OLED display, with the phototransistor to read data from the built-in LED for different energy meters, interface RS485 and built-in wireless transceiver nRF24L01 +.

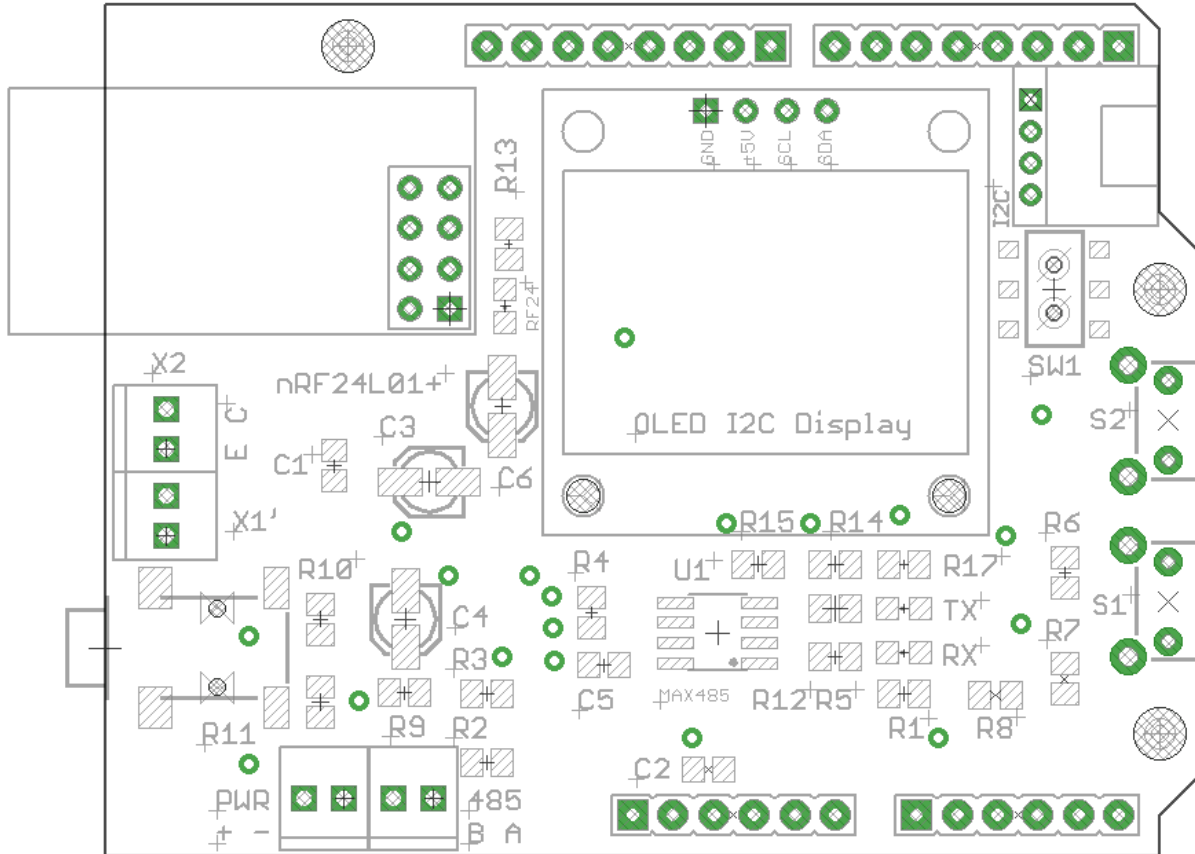
Model: 106990026



## Feature

- Connection current sensor
- Support for OLED-display I2C
- Terminal for connecting the phototransistor BPW85C
- RS485 interface
- Two buttons to control (enabled one analog pin)
- Transceiver for 2.4G nRF24L01 + soldered on the motherboard
- GROVE -compatible connector: I2C
- Full compatibility with (Wiznet 5100 + SD)

## Layout and schematics



The left side of EM Shield is a connector for current sensors (terminals X1 and X1' - duplicate each other).

X2 - connector for the phototransistor.

In the upper right corner - I2C-connector.

Just below the right are two buttons (labeled S1 and S2).

Left-bottom-placed terminals for interface RS485 (referred to as PWR and AB).

Below I2C connector is a switch that connects / disconnects Arduino-pins RX / TX on chip MAX485 (switching "programming" and "work with the interface RS485").



EM Shield v2 was designed to be fully compatible with Ethernet Shield (Wiznet 5100 + SD) - so you can use these two together Schild to create even more advanced device monitoring electricity (logging on SD-card and presenting the data on a web page).

## Interfaces

- A0 - engaged to connect the sensor AC
- A1 - to connect phototransistor
- A4 (SDA), A5 (SCL) - displayed on connector "I2C" (the other two pins in the connector - VCC and GND for sensor supply)
- Interface for connecting an RF-module nRF24L01 +:
  - D11 - MOSI
  - D12 - MISO
  - D13 - SCK
  - D7 - RF\_CE
  - D8 - RF\_CSN
  - D2 - RF\_IRQ
- Interface for MAX485:
  - D0 - RX (switched off switch for programming)
  - D1 - TX (switched off switch for programming)
  - D5 - Reception / Transmission
- A3 - Buttons

## Libraries

### Necessary library

To use the EM Shield requires the following libraries:

- Work with the transceiver nRF24L01 + - RF24
- Work with the display OLED (I2C 0.96" OLED 128x64 - Blue&Yellow)- U8glib
- Work with the display OLED (Grove - OLED Display 0.96)- OLED\_Display\_96X96
- Work with current sensors - EmonLib
  
- You must use the library from - [www.mysensors.org](http://www.mysensors.org)

### API

- MySensors Arduino Library (v1.5)

Additionally required libraries that are used when working RF24:

- SPI

### Features use libraries

Library has used examples that allow easy to understand how they work.

Initialization RF-module as follows:

...

```
//RF24 radio(CE,CSN);  
RF24 radio(7,8);
```

...

## Simple demonstration with wireless technology

```
/*  
This sketch is for a Energy Monitor v 2.0  
http://www.seeedstudio.com/depot/Energy-Monitor-Shield-V2-p-2393.html  
and MySensors 1.5  
http://www.mysensors.org/  
  
modified  
1 April 2016  
by greengo  
*/  
  
#include <SPI.h>  
#include "U8glib.h"  
#include <MySensor.h>  

```

```

int stateFloating = 0;

void setup()
{
  double cIrms = 0;
  // emon.current(0, 111.1);      // Current: input pin, calibration.
  emon.current(0, 71.2);

  double Irms[WINDOW];
  Irms[0] = emon.calcIrms(1480); // the first value in the measurement clearly "crooked"
  for (int i=0; i<WINDOW; i++) {
    Irms[i] = emon.calcIrms(1480);
    cIrms = cIrms + Irms[i];
    delay(100);
  }

#ifdef NODE_ADDRESS
  gw.begin(incomingMessage, NODE_ADDRESS, NODE_REPEAT);
#else
  gw.begin(incomingMessage, AUTO, NODE_REPEAT);
#endif
  gw.sendSketchInfo("Energy Monitor v2.0", "1.5.0");

  // Register all sensors to gw (they will be created as child devices)
  gw.present(CHILD_ID_POWER, S_POWER);
  gw.present(CHILD_ID_FLOATING_CUR_SENSOR, S_CUSTOM);
  sendPowerMeasurements(false);
}

void loop()
{
  gw.process();

  displayUpdate();

  unsigned long NOW_TIME = millis();

  if(NOW_TIME - CHECK_TIME >= SLEEP_TIME)
  {
    measureCount ++;
    bool forceTransmit = false;
    transmission_occured = false;

    if (measureCount > FORCE_TRANSMIT_INTERVAL) { // force a transmission
      forceTransmit = true;
      measureCount = 0;
    }

    sendPowerMeasurements(forceTransmit);

    // Serial.print(" measureCount: ");
    // Serial.println(measureCount);
    // Serial.print(" ");

    CHECK_TIME = NOW_TIME;
  }
}

void sendPowerMeasurements(bool force)
{
  bool tx = force;

  Irms = emon.calcIrms(1480); // Calculate Irms only
  realWatt = (emon.Irms * 220); // Extract Real Power into variable
  realkWt = (Irms * 0.220);

  // fixing minimum and maximum values
  if (realkWt>PMax) PMax = realkWt;
  if (realkWt<PMin && realkWt>0.05) PMin = realkWt;

  // Set relay to last known state (using eeprom storage)
  TEMP_TRANSMIT_THRESHOLD = gw.loadState(CHILD_ID_FLOATING_CUR_SENSOR);
  Floating = TEMP_TRANSMIT_THRESHOLD;

  // Serial.print("TEMP_TRANSMIT_THRESHOLD: ");Serial.println(TEMP_TRANSMIT_THRESHOLD);

  float diffIrms = abs(lastIrms - Irms);

  // Serial.print(F("IrmsDiff :"));Serial.println(diffIrms);

```

```

    if (diffIrms > TEMP_TRANSMIT_THRESHOLD) tx = true;

    if (tx) {
        measureCount = 0;

//      Serial.print("Watt: ");Serial.println(realWatt);
//      Serial.print("kWh : ");Serial.println(reakkWh);

        gw.send(IrmsMsg.set(realWatt, 1));
        gw.send(kWhMsg.set(reakkWh, 2));

        lastIrms = Irms;
        transmission_occured = true;
    }

// Serial.print("Irms : ");Serial.println(Irms);
}
// OLED SSD1306 128X64 (yellow line top)
void displayUpdate()
{
    u8g.firstPage();
    do {
        draw();
    } while( u8g.nextPage() );
}

void draw(void) {

    // graphic commands to redraw the complete screen should be placed here
    u8g.setFont(u8g_font_6x12r);
    // u8g.print("Energy Monitor");

    u8g.setPrintPos(0, 30); // Scale up all draw procedures
    u8g.print(reakkWh);
}

//*****
void incomingMessage(const MyMessage &message) {
    // We only expect one type of message from controller. But we better check anyway.
    if (message.isAck()) {
        // Serial.println("This is an ack from gateway");
    }

    if (message.type == V_VAR1) {
        // Change state
        stateFloating = message.getInt();

        // Store state in eeprom
        gw.saveState(CHILD_ID_FLOATING_CUR_SENSOR, stateFloating);

        // Write some debug info
//      Serial.print("Incoming change for sensor:");
//      Serial.print(message.sensor);
//      Serial.print(", Delta =: ");
//      Serial.println(message.getInt());
//      Serial.print(", New status: ");
//      Serial.println(message.getBool());
    }
}

```



## Version Tracker

Revision	Description	Release
1.0	Prototype V1	01.04.2014
2.0	Public version	29.12.2014

## Applications

- Monitoring of electricity consumption
- Get the data from the meters LED flash
- Receive data via RS485 interface
- Data transfer to other devices via nRF24l01 +
- Logging data to SD-card and display on a web page (via Ethernet Shield)

## Questions and Answers

- Blog Energy Monitor Shield - Wireless energy monitoring RU
- Ask a question by e-mail [support@devicter.ru](mailto:support@devicter.ru)

## How to buy

This product can be purchased via distributors:

Copyright (c) 2008-2016 Seeed Development Limited ([www.seeedstudio.com](http://www.seeedstudio.com) / [www.seeed.cc](http://www.seeed.cc))  
**This static html page was created from <http://www.seeedstudio.com/wiki>**