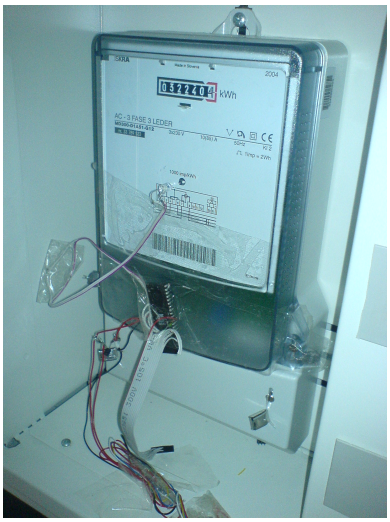
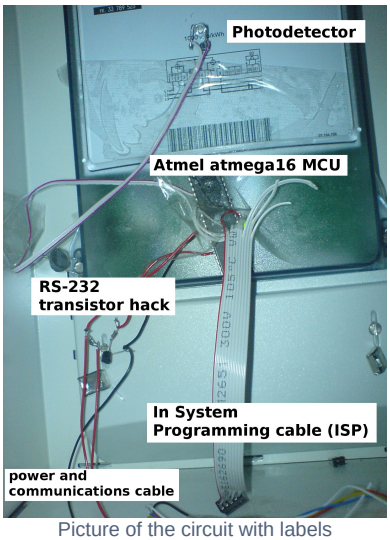
Household PowerMeter

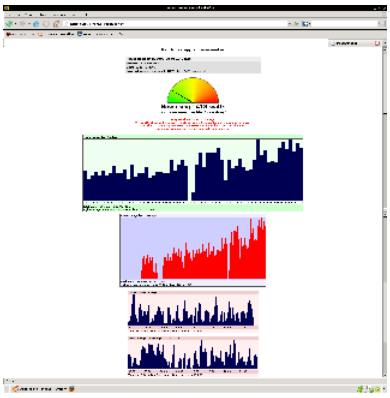
Archived Project

This is an old project, and probably no one currently at Omega Verksted knows anything more about it than what is provided on this page. You can try to recreate it if you want, but the documentation is provided as-is and it may be outdated.





Picture of the circuit with labels


PHP based web-interface displaying statistics

Current projectThe Household PowerMeter is a simple circuit with an AVR and a photodetector that detects flashes from the household wattmeter. The AVR software utilizes the on-chip UART serial port and sends arbitrary data to notify a host about the flashes. With this simple circuit and software one can log and analyze the total household power usage. A perl-script for logging the data into a mySQL database and a PHP page to display the data is also made.

Developers

Pål Driveklepp

Source code for ATmega16

```
/*
Source for software for atmega16 running at 8MHz sensing optical flashes
and notifying host via USART
*/



#include <avr/io.h>
#include <avr/signal.h>
#include <avr/interrupt.h>



unsigned char pulsecounter = 0;
unsigned int uptime = 0;
unsigned int downtime = 0;
unsigned char sufficientdowntime = 0;
unsigned char sendpointer = 0;

void flash(unsigned char pattern);
void notify(void);

int foo = 0;

void delay(unsigned int delayet){

        for (unsigned int i = 0; i < delayet; i++){

                foo++;

        }
}



/* AD-converter conversion complete interrupt routine */

ISR(ADC_vect){



        int sample = (ADCW>>2);



        if ((sample>32)&&(sufficientdowntime)){
```

```
                uptime++;
                downtime = 0;

        }

        else {

                if (uptime){

                        if((uptime>5)&&(uptime<60)){                //led
flash duration was measured to 16 samples

                                notify();
                                pulsecounter++;
                                PORTC = pulsecounter&0x01;

                        }
                }

                uptime = 0;

                downtime++;

                if(downtime>60){

                        sufficientdowntime = 1;

                }

                else{

                        sufficientdowntime = 0;

                }
        }



        delay(2500);
        ADCSRA |= (1<<ADSC);         //AD start conversion



}



int main(void){

        //LEDS
        DDRC = 0xFF;

        //ADC
        DDRA = 0x00;

        //serialport
        DDRD = 0xFF;

        //ADC READS ADC0 (pin1 on PortA)
        ADMUX = (1<<REFS0)|(1<<REFS1);

        //adc prefrences
        ADCSRA = (1<<ADEN)|(1<<ADSC)|(1<<ADIE)|(1<<ADPS2)|(1<<ADPS0);

        //setup USART
        UCSRB = (1<<TXEN);

        //8 databits see page 163 for writing to this register
        UCSRC = (1<<URSEL)|(3<<UCSZ0);//(1<<UCSZ1)|

        //baud-rate = 9600 (when mcu=8MHz)
```

```
        UBRRL = 51;

        flash(0x00);

        for (int i = 0; i < 256; i++){

                /* Wait for empty transmit buffer */
                while ( !( UCSRA & (1<<UDRE)) );

                /* Put data into buffer, sends the data */
                UDR = i;

        }




        //ENABLES GLOBAL INTERRUPTS
        sei();

        while(1){

        }
}



/* flash arbitrary signal */

void flash(unsigned char pattern){

        for (int i = 0; i < 10; i++){

                PORTC = pattern;

                delay(60000);

                PORTC = 0xFF;

                delay(60000);

        }
}



/* send signal to host */
void notify(void){



        /* Wait for empty transmit buffer */
        while ( !( UCSRA & (1<<UDRE)) );



        /* Put data into buffer, sends the data */
        UDR = 0x41 + sendpointer;



        /* Wait for empty transmit buffer */
        while ( !( UCSRA & (1<<UDRE)) );

        /* send end of text */
        UDR = 0x0A;



        if(sendpointer<25){
```

```
                sendpointer++;

        }

        else{

                sendpointer = 0;

        }
}
```

Perl script for logging to mySQL

```
#!/usr/bin/perl
#
#
# Author: Bruce S. Garlock
# Date: 2002-09-11
# Requirements: Device::SerialPort 0.12 (from cpan)
#
# Version: 0.1
#
#
# Description: This perl script is for logging of data from a serial
# port, to a specified logfile. The logfile can then be parsed with
# other programs for reporting purposes.
#
# This program was written for specifically logging Multitech's
# MTASR2-203 T1 Router. The router outputs text to the command
# port with 57.6k, 8-1-N, and No flow control.
#
#

use Device::SerialPort 0.12;

#$LOGDIR = "/var/log"; # path to data file
#$LOGFILE = "router.log"; # file name to output to
$PORT = "/dev/ttyS0"; # port to watch

#
#
# Serial Settings
#
#

$ob = Device::SerialPort->new ($PORT) || die "Can't Open $PORT: $!";
$ob->baudrate(9600) || die "failed setting baudrate";
$ob->parity("none") || die "failed setting parity";
$ob->databits(8) || die "failed setting databits";
$ob->handshake("none") || die "failed setting handshake";
$ob->write_settings || die "no settings";

#
# Send a string to the port
#

$pass=$ob->write("AT");
sleep 1;

#
# open the logfile, and Port
#

#open(LOG,">>${LOGDIR}/${LOGFILE}")
# ||die "can't open smdr file $LOGDIR/$LOGFILE for append: $SUB $!\n";
#

open(DEV, "<$PORT")
```

```
|| die "Cannot open $PORT: $_";

#select(LOG), $| = 1; # set nonbufferd mode
#
#
# Loop forver, logging data to the log file
#

use DBI;

$username = '*****';$password = '******';$database =
'*******';$hostname = 'localhost';
$dbh = DBI->connect("dbi:mysql:database=$database;" .
"host=$hostname;port=3306", $username, $password);



$watted = 0;
$SQL= "INSERT INTO ticktable VALUES ( '' , NOW( ) , '26')";

while($_ = <DEV>){ # print input device to file
if (($_ eq "X\n") || ($_ eq "Y\n") || ($_ eq "Z\n")){
$watted = 1;
print "reached end!";

}
else {
if($watted>0){

$dbh = DBI->connect("dbi:mysql:database=$database;" .
"host=$hostname;port=3306", $username, $password);
$InsertRecord = $dbh->do($SQL);

if ($InsertRecord){
print "SQL success!";
}

else{
print "SQL failed!: $DBI::errstr";
}

$watted = 0;
}
}
print $_;
}
undef $ob;
```