{ARTIKEL:news}

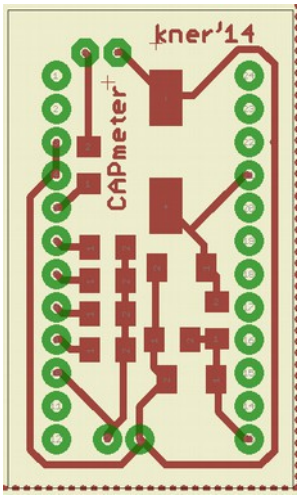
Measuring Cap's

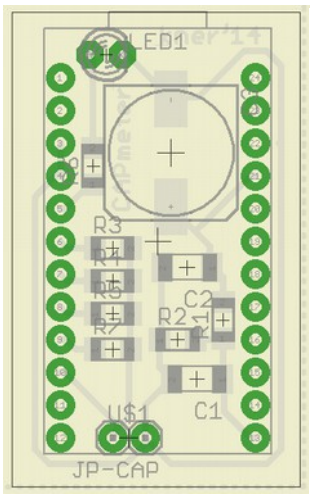
Cheap and simple Capacitance Meter.

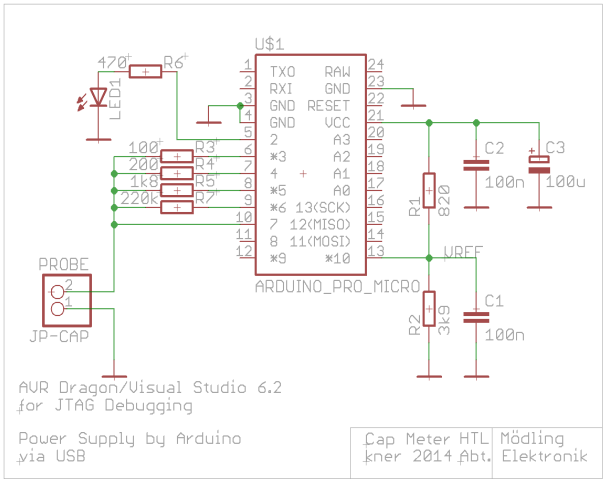
Principle: Count the time from start of loading to a reaching a reference voltage.

Auto Range: If loading time too short-> switch to higher load resistor

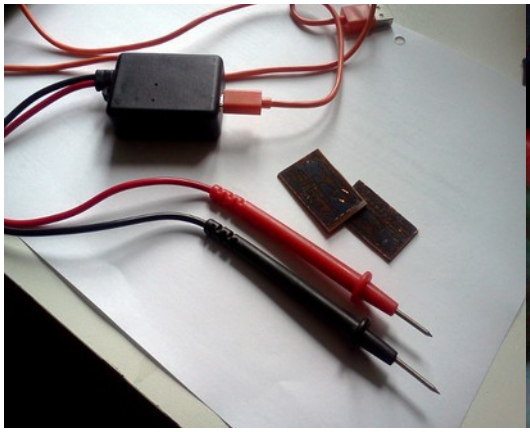
Problem: for very large Caps charge/discharge currents too high → use external transistor.





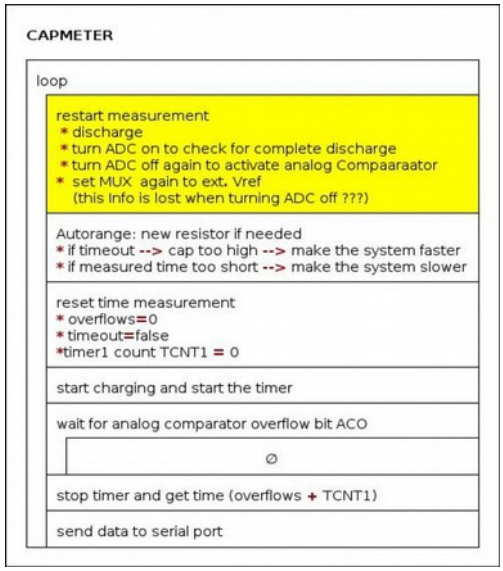


Attention: parts are placed on top layer, so soldering arduino pico pro's pins is difficult (see image below) – could be improved by 2 layer layout (or omit large cap C3 and place parts at bottom (new layout needed).



Software

Arduino




```
/*

* Main.cpp
*
* incomplete – open zip file attached
*/

#include <Arduino.h>
#define TIMER_START TCCR1B = (1<<CS10)
#define TIMER_STOP  TCCR1B = 0
#define ANALOG_COMP_INT_DISABLE  ACSR &= ~(1 <<
ACIE);
#define INPUT_HIGHER_THAN_REFERENCE (ACSR &
(1<<ACO))

#define THRESHOLD 5  //D10  ADC13

#define CHARGE 5                        //Arduino
Pin numbers
#define CHARGELONG 6
#define SENSE 4
#define DISCHARGE 3
#define TESTPIN 2                       //just for debugging - with LED

volatile int timeroverflows = 0;
volatile bool timeout = false;
int speed = 1; //fastest
unsigned long timePhase1;
unsigned int senseVoltage;

void sendSerial(); //output data to serial port
void dischargeC(); //discharge the cap
void restart(); //start new conversion

void chargeC(uint8_t speed1){
        pinMode(DISCHARGE,INPUT);
        pinMode(SENSE,INPUT);
        if (speed1==3){
                pinMode(CHARGE,INPUT);
                pinMode(CHARGELONG,OUTPUT);
                digitalWrite(CHARGELONG,HIGH);
        }
        else if (speed1==2){
                pinMode(CHARGELONG,INPUT);
                pinMode(CHARGE,OUTPUT);
                digitalWrite(CHARGE,HIGH);
        }
        else{
                pinMode(CHARGELONG,INPUT);
                pinMode(CHARGE,INPUT);
                pinMode(DISCHARGE,OUTPUT);          //fastest pin: discharge
                digitalWrite(DISCHARGE,HIGH);
                speed=1;
        }
}
```

```
uint16_t threshold;
uint16_t resistorHigh;
uint16_t resistorLow;

void setup()
{
        pinMode(TESTPIN,OUTPUT);
        Serial.begin(115200UL);

        getThresholdAndResistorinfo();

        // Analog Multiplexer: Enabled  high bit for Mux 8..15
        ADCSRB = (1 << ACME)|(1<<MUX5);

        ADCSRA &= ~(1<<ADEN);//ADEN disabled for Analog Mux to work
        ADMUX = THRESHOLD; // 5 ... ADC13
        //ACSR &= ~((1<<ACD)|(1<<ACBG));
        // Disable digital input (preserves power consumption)
        DIDR1 |= (1<<AIN0D);

        TCCR1A = 0;
        TCCR1B = 0;
        TIMSK1 = (1<<TOIE1);
        TIMER_STOP;

        speed = 1;

}

uint16_t nr =0;

#define TIMEOUT 1500
void loop()
{
        restart();
        TCNT1=0;
        if (timeout){
          speed--; //shorten time constant
          digitalWrite(TESTPIN,HIGH);
        }
        if (timePhase1<100){
                speed++;
             digitalWrite(TESTPIN,HIGH);
        }
        timeroverflows=0;
        timeout=false;
        chargeC(speed);
        digitalWrite(TESTPIN,LOW);
        TIMER_START;
        while((!INPUT_HIGHER_THAN_REFERENCE) &&(!timeout)); //wait for ACO
        TIMER_STOP;
        timePhase1=timeroverflows*0x10000+TCNT1;
        if (!timeout) sendSerial();
        else {
                Serial.print("Timeout:");
                Serial.println(speed);
        }
}

ISR(TIMER1_OVF_vect){
        timeroverflows++;
        if (timeroverflows>TIMEOUT) timeout=true;
}
```

```
void dischargeC(){
        pinMode(DISCHARGE,OUTPUT);
        digitalWrite(DISCHARGE,LOW);
        pinMode(SENSE,OUTPUT);
        digitalWrite(SENSE,LOW);
        pinMode(CHARGE,OUTPUT);
        digitalWrite(CHARGE,LOW);
        pinMode(CHARGELONG,OUTPUT);
        digitalWrite(CHARGELONG,LOW);
}
void restart(){
        dischargeC();
        ADCSRA |= (1<<ADEN); //turn ADC on
        senseVoltage = 100;
        while (senseVoltage > 0)  //wait for cap to discharge
        {
                delay(1);
                senseVoltage = analogRead(A6);
        }
        ADCSRA &= ~(1<<ADEN); //turn ADC off so Analog Mux can work
        ADMUX = THRESHOLD;
        delay(50);
}
```

Qt Creator

programming C++ and using Qt allows development for win/linux/android;

calculates capacitance and smooths results by means of FIFO (QQueue)

do the calculations on the pc side

```
QString processData(QByteArray data){
    QQueue<double> filter;
    int oldspeed=0;
    QString s = data;
    QStringList sl = s.split("/");
    if (sl.size()!=5) return ""; //there may be broken lines

    int speed=sl[0].toInt();

    if (speed<1 || speed>3) return "";
    int x2=sl[1].toInt();
    int x3=sl[2].toInt();
    int x4=sl[3].toInt();
    int timer1=sl[4].toInt();

    double T=1/16E6;            //Period @ 16MHz
    double Vthreshold=1.6;

    double Resistors[] = {150,200,2E3,200E3};
    int    corrections[]={0,    6,  6,   68};
    double R = Resistors[speed];
    //double timer1korr=timer1-68; //some pF for the equipment
    double timer1korr = timer1-corrections[speed];
    double t=timer1korr*T;
    double Cx=-t/(R*log(1-Vthreshold/5));

    if (speed != oldspeed) filter.clear();
```

```
    filter.enqueue(Cx);
    if (filter.size() > 10) filter.dequeue();
    double sum=0;
    for(int i=0; i<filter.size(); i++){
        sum += filter.at(i);
    }
    double Cy = sum/filter.size();
    Cxs = eng(Cx,2,0);
    QString Cys = eng(Cy,2,0);

    qDebug() << Cxs << " (" << Cys << ")";
    oldspeed=speed;

    //    }
    return Cys;
}
```