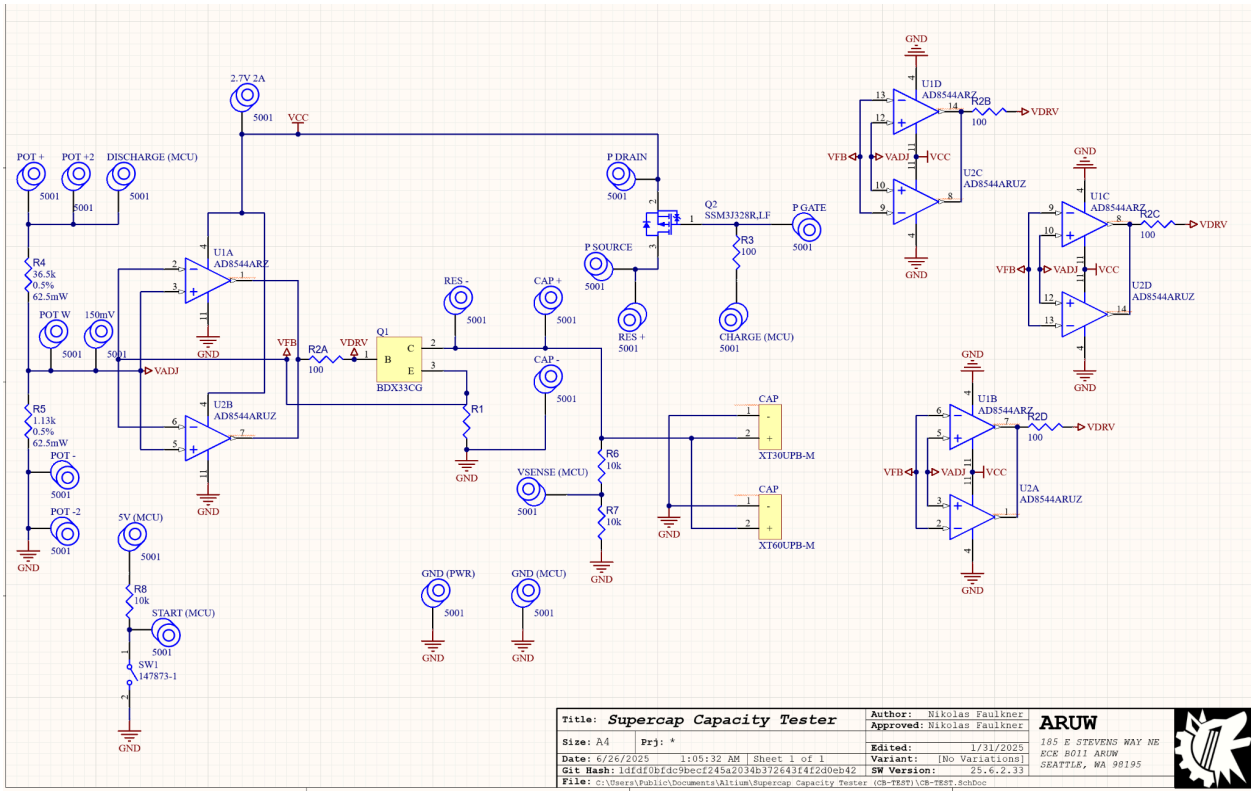
Supercapacitor Capacity Tester (CB4-TEST)

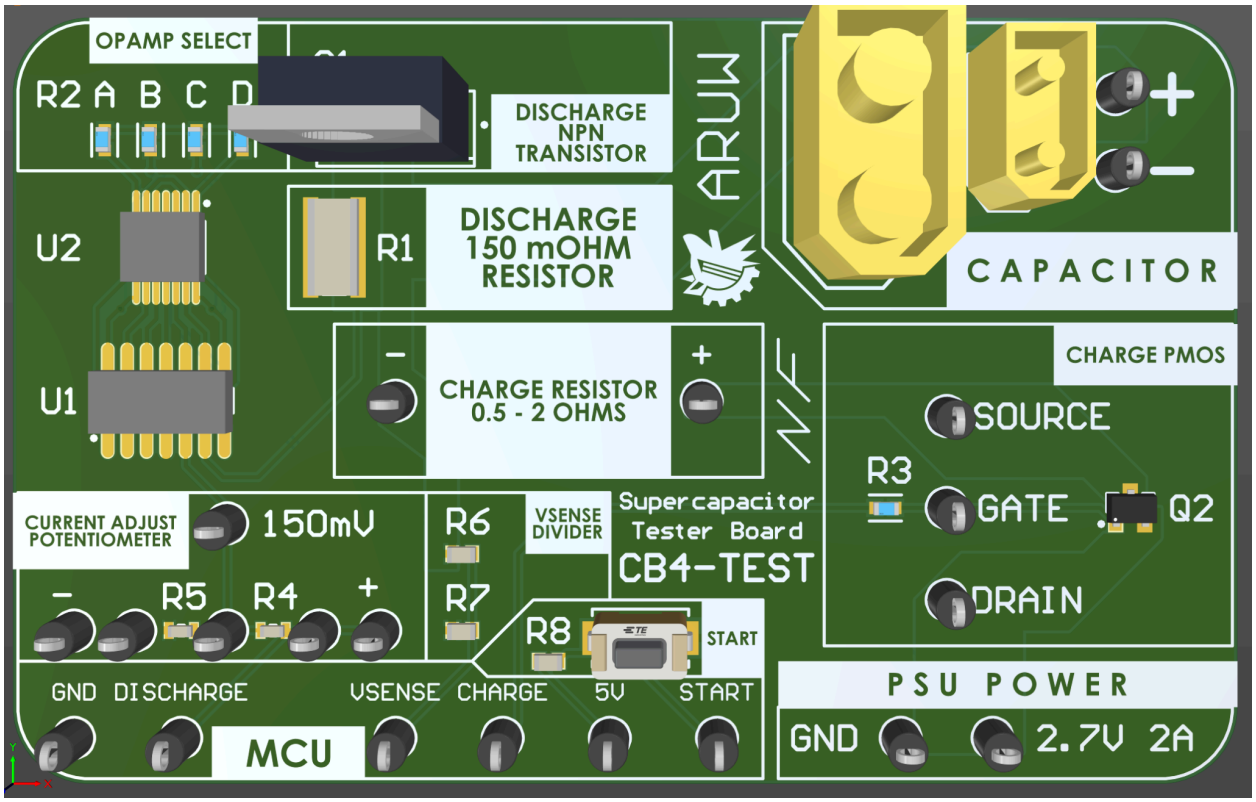
Nikolas Faulkner

Developed for the Mark 4 supercapacitor bank system (CB-4), the supercapacitor capacity tester is designed to automatically test and measure the capacity, ESR, and leakage current of a supercapacitor.

It uses a resistor and PMOS switch to charge the capacitor from a lab power supply, and a constant current discharge circuit consisting of an NPN darlington transistor, op-amp, NMOS MOSFET, and current sense resistor to discharge the capacitor at a very constant rate. The device is controlled by an Arduino, which controls both the charge and discharge circuits, and measures the voltage of the capacitor. The Arduino takes many voltage measurements across the complete charge and discharge cycle of the capacitor to try to obtain highly accurate measurements of the characteristics of the capacitor, with heavy focus on obtaining an accurate and precise measurement of the capacitance of the supercapacitor. This information is then used to group capacitors of similar capacity together when building the series supercapacitor arrays, in order to help with voltage balancing.

The schematic and board are designed to accomodate a wide variety of components and footprints, in order to be able to successfully build the tester circuit out of whatever components are on hand and available immmediately. The board is split into several clearly marked sections, which each serve a separate function as part of the cap tester.


2.7V 2A
VCC
POT +
POT +2
DISCHARGE (MCU)
P DRAIN
Q2
SSM3J328R,LF
P GATE
P SOURCE
R3
100
CHARGE (MCU)
RES +
RES -
CAP +
CAP -
U1A
AD8544ARZ
U2B
AD8544ARUZ
R4
36.5k
0.5%
62.5mW
POT W
150mV
VADJ
VFB
R2A
VDRV
Q1
BDX33CG
R1
R5
1.13k
0.5%
62.5mW
POT -
POT -2
5V (MCU)
R8
10k
START (MCU)
SW1
147873-1
VSENSE (MCU)
R6
10k
R7
10k
GND (PWR)
GND (MCU)
CAP
XT30UPB-M
XT60UPB-M
U1D
AD8544ARZ
U2C
AD8544ARUZ
R2B
100
U1C
AD8544ARZ
U2D
AD8544ARUZ
R2C
100
U1B
AD8544ARZ
U2A
AD8544ARUZ
R2D
100
GND
5001
Title: Supercap Capacity Tester
Author: Nikolas Faulkner
Approved: Nikolas Faulkner
Size: A4
Prj: *
Edited: 1/31/2025
Date: 6/26/2025 1:05:32 AM
Sheet 1 of 1
Variant: [No Variations]
Git Hash: 1dfdf0bfdc9becf245a2034b372643f4f2d0eb42
SW Version: 25.6.2.33
File: C:\Users\Public\Documents\Altium\Supercap Capacity Tester (CB-TEST)\CB-TEST.SchDoc
ARUW
185 E STEVENS WAY NE
ECE B011 ARUW
SEATTLE, WA 98195


OPAMP SELECT
R2 A B C D
DISCHARGE NPN TRANSISTOR
U2
U1
R1
DISCHARGE 150 mOHM RESISTOR
ARUW
CAPACITOR
CHARGE RESISTOR 0.5 - 2 OHMS
CHARGE PMOS
SOURCE
R3
GATE
Q2
DRAIN
CURRENT ADJUST POTENTIOMETER
150mV
R5
R4
R6
R7
VSENSE DIVIDER
Supercapacitor Tester Board
CB4-TEST
R8
START
PSU POWER
GND
2.7V 2A
GND DISCHARGE
VSENSE CHARGE
5V
START
MCU

The code has a series of tests it goes through throughout the testing sequence to ensure the test is progressing smoothly and a dangerous situation is not occurring. This works to prevent erroneous readings from being reported, and also helps to prevent overcharging of the capacitor and other undesireable situations.

Code:

```
//CB4-TEST supercap tester code V1
//Nikolas Faulkner
//2/11/25

//Discharger pin number
const int dischgPin = 7;
//Charger pin number
const int chargePin = 8;
//Cap voltage pin number
const byte vcapPin = A2; //A2 analog input
//Start button pin number
const int startPin = 11;
//Indicator light pin number (internal)
const int indicPin = 13;
//Calibration mode pin number
const int calibPin = 10;
//Beeper pin number
const int beepPin = 4;

//For making measurements and comparisons
int vhold = 0;
int vcomp = 0;
unsigned long thold = 0;
int vdelt = 0;

//For turning the indicator light on and off
bool light = false;

//Variable to tell charger to stop
bool stop = true;
//Variable to tell charger to skip last part of charging cycle
bool skip = false;

//NEW THRESHOLD NOTES:
//2.7V = 538
//0.0V = 000
//1 point = 5.019 mV
//2.615V = 521
```

```
//Upper threshold = 2.615V = 521 != 534
const int upThs = 521; //534;
//Lower threshold = 0.489V = 100 It is now 0.870V = 173 It is now 1.14 =
228
const int lwThs = 228; //173; //178;
//Emergency threshold = 2.713V = 541 != 555
const int emgThs = 541; //555;
//ESR+Leakage testpoint thresholds
const int tp1 = 250; // 1.25V
const int tp2 = 320; // 1.61V
const int tp3 = 460; // 2.31V
//Capacitance test thresholds (95 decrease per step)
const unsigned long Ths1 = 510; // 2.56V
const unsigned long Ths2 = 445; // 2.23V
const unsigned long Ths3 = 380; // 1.91V
const unsigned long Ths4 = 315; // 1.58V
const unsigned long Ths5 = 250; // 1.25V , used to be 880mV

//Integers for ESR tests
//Start point "stp"
int stp1 = 0;
int stp2 = 0;
int stp3 = 0;
//First instant point "insF"
int insF1 = 0;
int insF2 = 0;
int insF3 = 0;
//Second instant point "insS"
int insS1 = 0;
int insS2 = 0;
int insS3 = 0;
//Leakage point "lk"
int lk1 = 0;
int lk2 = 0;
int lk3 = 0;

//Integers for capacitance tests
unsigned long v0 = 0;
unsigned long t0 = 0;
unsigned long t1 = 0;
```

```
unsigned long t2 = 0;
unsigned long t3 = 0;
unsigned long t4 = 0;
unsigned long t5 = 0;

//Doubles for slope generation
unsigned long t_sum = 0;
unsigned long v_sum = 0;
//unsigned long t2_sum = 0;
unsigned long v2_sum = 0;
unsigned long tv_sum = 0;
double slope = 0.0;
//Doubles for leakage generation
double R = 0.0;
//Doubles for ESR generation
double ESR = 0.0;

//quick guess system
double QG = 0;
unsigned long tStartQG = 0;
unsigned long tStopQG = 0;
int vStartQG = 0;
int vStopQG = 0;
int shift = 20;

//***SETTINGS***
//Enable Quick Guess
const bool QGEnable = true;

//Enable beep
const bool beep = true;

//Debug mode
const bool debug = true; // Shows the internal behavior while charging and
discharging
const bool debug_advanced = true; // Shows raw final values used to
calculate the output


void setup() {
```

```
  // put your setup code here, to run once:
  pinMode(dischgPin, OUTPUT);
  pinMode(chargePin, OUTPUT);
  pinMode(vcapPin, INPUT);
  pinMode(startPin, INPUT);
  pinMode(indicPin, OUTPUT);
  pinMode(calibPin, INPUT);
  pinMode(beepPin, OUTPUT);
  Serial.begin(9600);
  Serial.println();
  Serial.println();
  Serial.println();
  Serial.println("**************************************");
  Serial.println("* CB4-TEST SUPERCAPACITOR TESTER V1.0 *");
  Serial.println("*      Nikolas Faulkner  4/05/25      *");
  Serial.println("**************************************");
  Serial.println();
  digitalWrite(dischgPin, 0); //Discharge off
  digitalWrite(chargePin, 1); //Charge off
  digitalWrite(indicPin, 0); //Indicator off
  light = false;
  stop = false;
  skip = false;
}

void loop() {
  // put your main code here, to run repeatedly:
  if(stop) {
    Serial.println("AN ERROR OCCURRED AND PROCESS WAS HALTED. HOLD START
TO OVERRIDE AND RUN TEST AGAIN");
    Serial.println();
    errorBeep(beep);
  } else {
    Serial.println("CONNECT CAPACITOR AND PRESS START");
    Serial.println();
    startBeep(beep);
  }
  skip = false;
  light = false;
  //Initial overvoltage test
```

```
  stop = overvoltTest(stop);
  //Idle, run calibration routine it enabled
  while(digitalRead(startPin) == 1) {
    delay(100);
    stop = overvoltTest(stop);
    if(stop) {
      digitalWrite(chargePin, 1); //Stop charging
      digitalWrite(dischgPin, 1); //Start discharging
    }
    if(!stop && digitalRead(calibPin) == 1) {
      Serial.println("ENTERING CALIBRATION MODE. WARNING: SAFETY SYSTEMS
DISABLED");
      while(!stop && digitalRead(calibPin) == 1) {
        delay(100);
        digitalWrite(dischgPin, 1); //Discharge on
        digitalWrite(chargePin, 0); //Charge on
        Serial.println(analogRead(vcapPin));
        stop = overvoltTest(stop);
      }
    }
  }
  //Clear fault code
  if(stop && digitalRead(startPin) == 0) {
    delay(500);
    if(digitalRead(startPin) == 0) {
      stop = false;
    }
  }
  //Start test when button pressed
  if(!stop && digitalRead(startPin) == 0) {
    delay(10);
    if(digitalRead(startPin) == 0) {
      //If capacitor is above lower threshold, discharge to lower
threshold
      if(analogRead(vcapPin) >= lwThs) {
        Serial.println("DISCHARGING TO START VALUE");
        digitalWrite(chargePin, 1); //Stop charging
        digitalWrite(dischgPin, 1); //Start discharging
        thold = millis();
        vhold = analogRead(vcapPin);
```

```
        while(!stop && analogRead(vcapPin) >= lwThs) {
          delay(1000);
          if(debug) {
            Serial.print("Cap Voltage: ");
            Serial.println(analogRead(vcapPin));
          }
          digitalWrite(indicPin, light ? LOW : HIGH); //Flash light
          light = !light;
          if(millis()-thold > 18000 && (vhold-analogRead(vcapPin)) < 2) {
//If it fails to discharge promptly after 18 seconds
            Serial.println("ERROR 5: FAILURE TO DISCHARGE - Check
connections and circuit");
            stop = true;
          }
          vhold = analogRead(vcapPin);
        }
      }
      //Charging cycle
      if(!stop) {
          digitalWrite(dischgPin, 0); //Stop discharging
          //Begin capacitor test
          Serial.println("TEST BEGIN, CHARGING");
          //Charging portion
          digitalWrite(indicPin, 1); //Indicator on
          digitalWrite(chargePin, 0); //Charge on
          //Quick guess system to approximate the capacitance to adjust
time scaling for charge and discharge cycle.
          delay(500);
          QG = quickGuess();
          Serial.print("Rough approximation of capacitance: ");
          Serial.print(QG);
          Serial.println(" millifarads");
          if(QGEnable) {
            //shift = (QG / 3000);
            shift = (QG / 4000);
            if(debug) {
              Serial.print("Time shift factor adjusted to ");
              Serial.print(shift);
              Serial.println(" from standard value of 20.");
            }
```

```
          } else {
            shift = 20;
          }
          delay(200);
          //Once voltage hits value 1, run first ESR/Leakage test
          while(!stop && analogRead(vcapPin) <= tp1) {
            stop = testForCharging(stop, 200, debug, shift);
            delay(50);
            digitalWrite(indicPin, light ? LOW : HIGH); //Flash light
            light = !light;
          }
          esrTest(1, debug); //Run ESR test 1
          while(!stop && analogRead(vcapPin) <= tp2) {
            stop = testForCharging(stop, 250, debug, shift);
            delay(50);
            digitalWrite(indicPin, light ? LOW : HIGH); //Flash light
            light = !light;
          }
          esrTest(2, debug); //Run ESR test 2
          while(!stop && analogRead(vcapPin) <= tp3) {
            stop = testForCharging(stop, 650, debug, shift);
            delay(50);
            digitalWrite(indicPin, light ? LOW : HIGH); //Flash light
            light = !light;
          }
          esrTest(3, debug); //Run ESR test 3
          while(!stop && !skip && analogRead(vcapPin) <= upThs) {
            stop = testForCharging(stop, 1000, debug, shift);
            if(analogRead(vcapPin) >= 800) {
              skip = testForCharging(skip, 600, debug, shift);
            }
            delay(50);
            digitalWrite(indicPin, light ? LOW : HIGH); //Flash light
            light = !light;
          }
          if(stop) {
            Serial.println("ERROR 0: FAILURE TO CHARGE - Check capacitor
and power supply");
          }
          if(skip) {
```

```
            Serial.println("ERROR 1: SLOW CHARGE - Test will continue, but
may be inaccurate. Check for excessive leakage");
          }
          //Cap should be at upper threshold now.

          //Discharging portion
          Serial.println();
          Serial.println("BEGIN DISCHARGING");
          Serial.println();
          digitalWrite(chargePin, 1); //Charge off
          digitalWrite(dischgPin, 1); //Discharge on
          //Save time and initial voltage (t0, v0)
          t0 = millis();
          v0 = analogRead(vcapPin);
          //Accumulator to allow leniency on a few errors.
          int accumilator = 0;
          //Save timestamps for various voltage points until fully
discharged (t1, t2, t3, t4, t5)
          while(!stop && analogRead(vcapPin) > Ths1) {
            stop = testForCharging(stop, -22, debug, shift);
            if(stop) {
              if(accumilator > 10) {
                Serial.println("ERROR 6: FAILURE TO DISCHARGE - Check
connections and circuit");
              } else {
                stop = false;
                accumilator += 1;
              }
            }
            delay(50);
            digitalWrite(indicPin, light ? LOW : HIGH); //Flash light
            light = !light;
          }
          t1 = millis();
          while(!stop && analogRead(vcapPin) > Ths2) {
            stop = testForCharging(stop, -22, debug, shift);
            if(stop) {
              if(accumilator > 20) {
                Serial.println("ERROR 7: FAILURE TO DISCHARGE - Check
connections and circuit");
```

```
              } else {
                stop = false;
                accumulator += 1;
              }
            }
            delay(50);
            digitalWrite(indicPin, light ? LOW : HIGH); //Flash light
            light = !light;
          }
          t2 = millis();
          while(!stop && analogRead(vcapPin) > Ths3) {
            stop = testForCharging(stop, -22, debug, shift);
            if(stop) {
              if(accumilator > 30) {
                Serial.println("ERROR 8: FAILURE TO DISCHARGE - Check
connections and circuit");
              } else {
                stop = false;
                accumilator += 1;
              }
            }
            delay(50);
            digitalWrite(indicPin, light ? LOW : HIGH); //Flash light
            light = !light;
          }
          t3 = millis();
          while(!stop && analogRead(vcapPin) > Ths4) {
            stop = testForCharging(stop, -22, debug, shift);
            if(stop) {
              if(accumilator > 40) {
                Serial.println("ERROR 9: FAILURE TO DISCHARGE - Check
connections and circuit");
              } else {
                stop = false;
                accumilator += 1;
              }
            }
            delay(50);
            digitalWrite(indicPin, light ? LOW : HIGH); //Flash light
            light = !light;
```

```
          }
          t4 = millis();
          while(!stop && analogRead(vcapPin) > Ths5) {
            stop = testForCharging(stop, -22, debug, shift);
            if(stop) {
              if(accumilator > 50) {
                Serial.println("ERROR 10: FAILURE TO DISCHARGE - Check
connections and circuit");
              } else {
                stop = false;
                accumilator += 1;
              }
            }
            delay(50);
            digitalWrite(indicPin, light ? LOW : HIGH); //Flash light
            light = !light;
          }
          t5 = millis();
          if(!stop) {
            Serial.println("TEST CONCLUDED");
            Serial.println();
          }


          //Results portion
          if(debug_advanced) {
            Serial.println();
            //Serial.print(stp insF insS lk v0 t0 t1 t2 t3 t4 t5);
            Serial.print("stp1 ");
            Serial.println(stp1);
            Serial.print("insF1 ");
            Serial.println(insF1);
            Serial.print("insS1 ");
            Serial.println(insS1);
            Serial.print("lk1 ");
            Serial.println(lk1);
            Serial.print("stp2 ");
            Serial.println(stp2);
            Serial.print("insF2 ");
            Serial.println(insF2);
```

```
            Serial.print("insS2 ");
            Serial.println(insS2);
            Serial.print("lk2 ");
            Serial.println(lk2);
            Serial.print("stp3 ");
            Serial.println(stp3);
            Serial.print("insF3 ");
            Serial.println(insF3);
            Serial.print("insS3 ");
            Serial.println(insS3);
            Serial.print("lk3 ");
            Serial.println(lk3);
            Serial.print("v0 ");
            Serial.println(v0);
            Serial.print("t0 ");
            Serial.println(t0);
            Serial.print("t1 ");
            Serial.println(t1);
            Serial.print("t2 ");
            Serial.println(t2);
            Serial.print("t3 ");
            Serial.println(t3);
            Serial.print("t4 ");
            Serial.println(t4);
            Serial.print("t5 ");
            Serial.println(t5);
            Serial.println();
          }
          if(stop) {
            Serial.println("Test aborted due to previous error, check
connections and run test again");
            Serial.println();
          } else {
            Serial.println("******* RESULTS: ********");
            //Calculate best fit line for capacitance discharge points
            //Shift times
            t1 = t1 - t0;
            t2 = t2 - t0;
            t3 = t3 - t0;
            t4 = t4 - t0;
```

```
            t5 = t5 - t0;
            //Make double sums
            t_sum = t1 + t2 + t3 + t4 + t5;
            //t2_sum = (t1 * t1);
            //t2_sum += (t2 * t2);
            //t2_sum += (t3 * t3);
            //t2_sum += (t4 * t4);
            //t2_sum += (t5 * t5);
            v_sum = v0 + Ths1 + Ths2 + Ths3 + Ths4 + Ths5;
            v2_sum = (v0 * v0);
            v2_sum += (Ths1 * Ths1);
            v2_sum += (Ths2 * Ths2);
            v2_sum += (Ths3 * Ths3);
            v2_sum += (Ths4 * Ths4);
            v2_sum += (Ths5 * Ths5);
            tv_sum = (t1 * Ths1);
            tv_sum += (t2 * Ths2);
            tv_sum += (t3 * Ths3);
            tv_sum += (t4 * Ths4);
            tv_sum += (t5 * Ths5);
            //Determine slope, t is y variable, v is x variable, slope is
milliseconds per 1/2 voltage point, 1 voltage point is 5.018 mV, not
0.4887 mV
            long a = (6 * tv_sum);
            a -= (v_sum * t_sum);
            long c = (6 * v2_sum);
            c -= (v_sum * v_sum);
            slope = a/c;
            //slope = ((6 * tv_sum) - (v_sum * t_sum)) / ((6 * v2_sum) -
(v_sum * v_sum));
            //Determine capacitance from slope: current = 1A, so 1C/sec,
so 1mC * slope per 5.018 mV (not 0.4887 mV), or [2.046F] 0.199F * slope
            slope = -0.199 * slope;
            //Calculate leakage from capacitance and leakage points
            //V(t) = V0*e^(-t/RC)
            //t = 10
            //C = slope
            //V0 = insS
            //V(t) = lk
            //lk = insS*e^(-10/(R*slope))
```

```
            //ln(lk/insS) = -10/(R*slope)
            //R = (-1.0) / (slope * ln((1.0*lk)/(1.0insS)))
            R = (-10.0) / (0.01 + (slope * log((1.0*lk1)/(1.0*insS1)))) +
(-10.0) / (0.01 + (slope * log((1.0*lk2)/(1.0*insS2)))) + (-10.0) / (0.01
+ (slope * log((1.0*lk3)/(1.0*insS3))));
            //Calculate ESR from instantaneous voltage deltas
            //For each test, there are 2 instantaneous voltage jumps:
stp->insF, and insF->insS.
            //Average stp and insS, then find voltage delta between that
and insF. 1/delta = ESR. Average these.
            //Multiply delta by 5.018mV per step. Then, 1/delta = ESR.
Average these from 3 tests.
            //We can combine previous steps all in the final ESR test.
            stp1 = (stp1 + insS1)/2;
            insF1 = stp1 - insF1;
            stp2 = (stp2 + insS2)/2;
            insF2 = stp2 - insF2;
            stp3 = (stp3 + insS3)/2;
            insF3 = stp3 - insF3;
            ESR = (0.199 * 3.0) / (0.0 + insF1 + insF2 + insF3);
            //Print out results
            if(debug_advanced) {
              Serial.print("t1 ");
              Serial.println(t1);
              Serial.print("t2 ");
              Serial.println(t2);
              Serial.print("t3 ");
              Serial.println(t3);
              Serial.print("t4 ");
              Serial.println(t4);
              Serial.print("t5 ");
              Serial.println(t5);
              Serial.print("t_sum ");
              Serial.println(t_sum);
              //Serial.print("t2_sum ");
              //Serial.println(t2_sum);
              Serial.print("v_sum ");
              Serial.println(v_sum);
              Serial.print("v2_sum ");
              Serial.println(v2_sum);
```

```
    Serial.print("tv_sum ");
    Serial.println(tv_sum);
    Serial.print("slope ");
    Serial.println(slope);
    Serial.print("R ");
    Serial.println(R);
    Serial.print("stp1 ");
    Serial.println(stp1);
    Serial.print("insF1 ");
    Serial.println(insF1);
    Serial.print("stp2 ");
    Serial.println(stp2);
    Serial.print("insF2 ");
    Serial.println(insF2);
    Serial.print("stp3 ");
    Serial.println(stp3);
    Serial.print("insF3 ");
    Serial.println(insF3);
    Serial.print("ESR ");
    Serial.println(ESR);
    Serial.println();
  }
  //Capacitance
  Serial.print("Capacitance: ");
  Serial.print(slope);
  Serial.println(" Farads");
  //Quick capacitance
  if(QGEnable) {
    Serial.print("Quick capacitance (less accurate): ");
    Serial.print(QG/1000.0);
    Serial.println(" Farads");
  }
  //Leakage
  Serial.print("Leakage:     ");
  Serial.print(R);
  Serial.println(" Ohms");
  //ESR
  Serial.print("ESR:         ");
  Serial.print(ESR);
  Serial.println(" Ohms");
```

```
            Serial.println("************************");
            Serial.println();
            Serial.println("READY FOR NEXT TEST");
            successBeep(beep);
          }
        }

    }
  }
  //Wait until start button is no longer pressed
  while(digitalRead(startPin) == 0) {
    delay(10);
  }
}

//Test for overvoltage
bool overvoltTest(bool in) {
  if(in) {
    return true;
  }
  if(analogRead(vcapPin) >= emgThs) {
    delay(10);
    if(analogRead(vcapPin) >= emgThs) {
      Serial.println("ERROR 3: OVERCHARGE - REMOVE CAPACITOR
IMMEDIATELY!");
      if(analogRead(vcapPin) >= (1000)) {
        Serial.println("ERROR 4: SEVERE OVERCHARGE - DROP AND RUN!");
      }
      return true;
    }
  }
  return false;
}

//Test for ESR and leakage
void esrTest(int num, bool debug) {
  if(debug) {
    Serial.print("ESR Test ");
    Serial.print(num);
    Serial.print(":  Voltage = ");
```

```
    Serial.println(analogRead(vcapPin));
  }
  if(!stop) {
    digitalWrite(chargePin, 1); //Charge off
    delay(100);
    //Save start point "stp"
    if(num == 1) {
      stp1 = analogRead(vcapPin);
    } else if(num == 2) {
      stp2 = analogRead(vcapPin);
    } else if(num == 3 ){
      stp3 = analogRead(vcapPin);
    }
    digitalWrite(dischgPin, 1); //Discharge on
    //Save first instantaneous point "insF"
    if(num == 1) {
      insF1 = analogRead(vcapPin);
    } else if(num == 2) {
      insF2 = analogRead(vcapPin);
    } else if(num == 3 ){
      insF3 = analogRead(vcapPin);
    }
    digitalWrite(dischgPin, 0); //Discharge off
    //Save second instantaneous point "insS"
    if(num == 1) {
      insS1 = analogRead(vcapPin);
    } else if(num == 2) {
      insS2 = analogRead(vcapPin);
    } else if(num == 3 ){
      insS3 = analogRead(vcapPin);
    }
    delay(10000);
    //Save leakage point "lk"
    if(num == 1) {
      lk1 = analogRead(vcapPin);
    } else if(num == 2) {
      lk2 = analogRead(vcapPin);
    } else if(num == 3 ){
      lk3 = analogRead(vcapPin);
    }
```

```
    digitalWrite(chargePin, 0); //Charge on
  }
  if(debug) {
    Serial.print("Final voltage: ");
    Serial.println(analogRead(vcapPin));
  }
}

//Test to ensure charging or discharging
bool testForCharging(bool in, int time, bool print, int adjustment) {
  if(in) {
    return true;
  }
  if(time < 0) {
    digitalWrite(chargePin, 1); //Charge off
    digitalWrite(dischgPin, 1); //Discharge on
  } else {
    digitalWrite(dischgPin, 0); //Stop discharging
    digitalWrite(chargePin, 0); //Charge on
  }

  vhold = analogRead(vcapPin);
  delay(abs(time*(abs(adjustment)+1)));
  vcomp = analogRead(vcapPin);
  if(time < 0) {
    if(print) {
      Serial.print(vhold);
      Serial.print(" -> ");
      Serial.print(vcomp);
      Serial.println(vcomp < vhold ? " GO" : " STOP");
    }
    return vcomp >= vhold;
  }
  if(print) {
    Serial.print(vhold);
    Serial.print(" -> ");
    Serial.print(vcomp);
    Serial.println(vcomp > vhold ? " GO" : " STOP");
  }
  return vcomp <= vhold;
```

```
}

//Automatically gets a rough estimate of capacitor size
double quickGuess() {
  digitalWrite(chargePin, 0); //Charge on
  digitalWrite(dischgPin, 0); //Stop discharging
  //Measure 2 points in charge cycle
  tStartQG = millis();
  vStartQG = analogRead(vcapPin);
  int overflow = 0;
  while(overflow < 30000 && analogRead(vcapPin) <= (vStartQG + 25)) {
    delay(1);
    overflow += 1;
  }
  if (overflow >= 30000) {
    return 0;
  }
  tStopQG = millis();
  vStopQG = analogRead(vcapPin);
  //Calculate capacitance from known quantities and 2 points
  //V(t) = V0*e^(-t/RC)
  //V(t) = V0*(1-e^(-t/RC))
  //t = tStopQG - tStardQG
  //R = 0.66 ohms
  //V0 = vStartQG
  //V(t) = vStopQG
  //vStopQG = vStartQG*(1 - e^((tStopQG - tStartQG)/(0.66*C)))
  //(1 - ln(vStopQG/vStartQG)) = (tStopQG - tStartQG)/(0.66*C)
  //C = (tStopQG - tStartQG) / (0.66 * (1 - ln(vStopQG/vStartQG)))
  //return 1000 * (tStopQG - tStartQG) / (0.66 * (1 -
log(vStopQG/vStartQG)));
  if(debug_advanced) {
    Serial.print("tStartQG: ");
    Serial.println(tStartQG);
    Serial.print("tStopQG: ");
    Serial.println(tStopQG);
    Serial.print("vStartQG: ");
    Serial.println(vStartQG);
    Serial.print("vStopQG: ");
    Serial.println(vStopQG);
```

```
  }
  return (1.0*tStopQG - 1.0*tStartQG) / (0.66 * log((1.0 -
(1.0*vStartQG/538.0)) / (1.0 - (1.0*vStopQG/538.0))));
}

//Beep for error
void errorBeep(bool in) {
  if(in) {
    tone(1, 3000);
  }
}

//Beep for start up
void startBeep(bool in) {
  if(in) {
    tone(1, 300);
    delay(200);
    tone(1, 100);
    delay(50);
    tone(1, 100);
  }
}

//Beep for test success
void successBeep(bool in) {
  if(in) {
    tone(4, 250);
    tone(2, 250);
    tone(1, 250);
  }
}

//Tone generator
void tone(int period, int duration) {
  int i = 0;
  while(i < duration) {
    digitalWrite(beepPin, 1);
    delay(period);
    digitalWrite(beepPin, 0);
    i += period;
```

```
    }
}
```