Tonuino_Power.ino

```
#include <SoftwareSerial.h>

#define debug_mode 0 //0 deactivate debug mode, 1 debug mode serial output; necessary because of 5 pins maximum

int Pin_Relais_PBI=0;
int Pin_Relais_PBO=1;
int Pin_Read_Cap=3;
int Pin_Read_Charger=4;
int Pin_Serial_Debug=2;//debug on PB wakeup pin

#if debug_mode == 1
  SoftwareSerial mySerial(99,Pin_Serial_Debug); // 99 is dummy since recieving is not necessary
  int Pin_PBWake=98;//dummyPin
#else
  SoftwareSerial mySerial(99,98);
  int Pin_PBWake=2;
#endif

class PowerStatus
{
  // Class Member Variables
  unsigned long previousMillis;   // will store last time Relais was updated
  unsigned long startMillis;   // will store last time Relais was updated
  int CapValues[10]={0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
  int i=0; //array index for values

  public:
  boolean StatusChanged;   //did Status change?
  int CapStatus;      // Status of the cap 0=unknown/init, 1=charging, 2=full, 3=discharging
int previous_CapStatus;      // Status of the cap 0=unknown/init, 1=charging, 2=full, 3=discharging

boolean runState;                //does it run?

boolean ChargerStatus;


PowerStatus()

{

  CapStatus = 0;

  previous_CapStatus=0;

  pinMode(Pin_Read_Charger, INPUT_PULLUP);

  ChargerStatus = !digitalRead(Pin_Read_Charger);

  StatusChanged = LOW;

  runState = LOW;

  previousMillis = 0;

  startMillis = 0;

}


void readStatus()

{

  unsigned long currentMillis = millis();

  int readIntervall=100; //in ms

  //int minChange=1; //minimal differntial change to sense charging or discharging in change times
100


  if (runState == LOW)

  {

    startMillis = currentMillis;

    previousMillis = currentMillis;

    runState = HIGH;


  }

  else

  {
      if ((currentMillis-previousMillis >= readIntervall) && (i <= 9) )
      {
        CapValues[i]=analogRead(Pin_Read_Cap); // 1024 means step-up voltage (@5.2V stepUp Cap
charges up to 4.2V, 4V=788 )
        if (debug_mode == 1)
        {
          mySerial.println(CapValues[i]);
        }
        previousMillis=currentMillis;
        i=i+1;
      }
      else
      {
        if (i == 10)
        {
          ChargerStatus = !digitalRead(Pin_Read_Charger);// read Charger
          //change status
          int
CapValues_Av=(CapValues[0]+CapValues[1]+CapValues[2]+CapValues[3]+CapValues[4]+CapValues[5]
+CapValues[6]+CapValues[7]+CapValues[8]+CapValues[9])/10;
          //int Cap_Values_Diff_Av=((CapValues[1]-CapValues[0])+(CapValues[2]-
CapValues[1])+(CapValues[3]-CapValues[2])+(CapValues[4]-CapValues[3])+(CapValues[5]-
CapValues[4])+(CapValues[6]-CapValues[5])+(CapValues[7]-CapValues[6])+(CapValues[8]-
CapValues[7])+(CapValues[9]-CapValues[8]))*10/9;
          if (debug_mode == 1)
          {
            mySerial.println(CapValues_Av);
            //mySerial.println(Cap_Values_Diff_Av);
          }
          //if (((CapValues_Av>=700) && (ChargerStatus==HIGH)) || ((CapValues_Av>=600) &&
(ChargerStatus==LOW)))
          if (((CapValues_Av>=700) && (ChargerStatus==HIGH)))
          {
            if (debug_mode == 1)
```

```
  {
    mySerial.println(F("State: Flull"));
  }
  if (CapStatus!=2)
  {
    previous_CapStatus=CapStatus;
    CapStatus=2; //Fully charged
    StatusChanged=HIGH;
  }
}
else if ((CapValues_Av<700) && (ChargerStatus==HIGH))
{
  if (debug_mode == 1)
  {
    mySerial.println(F("State: Charging"));
  }
  if (CapStatus!=1)
  {
    previous_CapStatus=CapStatus;
    CapStatus=1; //charging
    StatusChanged=HIGH;
  }
}
else if (ChargerStatus==LOW)
{
  if (debug_mode == 1)
  {
    mySerial.println(F("State: Discharging"));
  }
  if (CapStatus!=3)
  {
    previous_CapStatus=CapStatus;
```

```
              CapStatus=3; //discharging
              StatusChanged=HIGH;
            }
          }
          else
          {
            if (debug_mode == 1)
            {
              mySerial.println(F("State: Unknown"));
            }
            if (CapStatus!=0)
            {
              previous_CapStatus=CapStatus;
              CapStatus=0; //unknown or init
              StatusChanged=HIGH;
            }
          }
          runState = LOW;

          i=0;
        }
      }
    }
  }
};

class Relais
{
  // Class Member Variables
  // These are initialized at startup
  int RelaisPin;      // the number of the LED pin
```

```
unsigned long previousMillis;   // will store last time Relais was updated
unsigned long startMillis;   // will store last time Relais was updated

public:

boolean RelaisState;               // true means on, false means off
int runState;               //does it run? 0=fresh, 1=running, 2=finished

Relais(int pin)
{
  RelaisPin = pin;
  pinMode(RelaisPin, OUTPUT);
  RelaisState = LOW;
  runState = 0;
  startMillis = 0;
}

void Pulse(int delaytime, int duration)
{
  unsigned long currentMillis = millis();
  if (runState == 0)
  {
    startMillis = currentMillis;
    runState = 1;
  }
  else if (runState == 1)
  {
    if (currentMillis-startMillis >= delaytime)
    {
      if (currentMillis-startMillis >= duration+delaytime)
      {
        digitalWrite(RelaisPin, LOW);
```

```
     RelaisState = LOW;
     runState = 2;
     if (debug_mode == 1)
     {
      mySerial.println("PulseOff");
     }
    }
    else
    {
     if (RelaisState == LOW)
     {
      RelaisState = HIGH;  // turn it on
      digitalWrite(RelaisPin, RelaisState);   // Update the actual Relais
      if (debug_mode == 1)
      {
       mySerial.println("PulseOn");
      }
     }
    }
   }
  }
 }

void TurnOn(int delaytime)
 {
  unsigned long currentMillis = millis();
  if (runState == 0)
  {
   startMillis = currentMillis;
   runState = 1;
  }
  else if (runState == 1)
  {
   if (currentMillis-startMillis >= delaytime)
   {
    digitalWrite(RelaisPin, HIGH);
    RelaisState = HIGH;
    runState = 2;
    if (debug_mode == 1)
    {
     mySerial.println("TurnedOn");
    }
   }
  }
 }

void TurnOff(int delaytime)
 {
  unsigned long currentMillis = millis();
  if (runState == 0)
  {
   startMillis = currentMillis;
   runState = 1;
  }
  else if (runState == 1)
  {
   if (currentMillis-startMillis >= delaytime)
   {
    digitalWrite(RelaisPin, LOW);
    RelaisState = LOW;
    runState = 2;
    if (debug_mode == 1)
    {
     mySerial.println("TurnedOff");
```

```
        }
      }
    }
  }
};

Relais  Relais_PBI(Pin_Relais_PBI);
Relais  Relais_PBO(Pin_Relais_PBO);
Relais  PBWake(Pin_PBWake); //actually this is not a relais but an optocoupler
PowerStatus PowerStatusOb;

void setup() {
  // put your setup code here, to run once:
  delay(1000);
  if (debug_mode == 1)
  {
    mySerial.begin(9600);
    mySerial.println(F("PowerMuxer Tonuino Debugger"));
  }
}

void loop() {
  // put your main code here, to run repeatedly:

  if(PowerStatusOb.StatusChanged == HIGH)//runs until it is set LOW here
  {
    if(PowerStatusOb.CapStatus == 0)//unknown
    {
      Relais_PBI.TurnOff(0);//disconnect PB Input immediatly
      Relais_PBO.TurnOff(500);//connect Tonuino to Powerbank with safety delay
    }
  else if((PowerStatusOb.CapStatus == 1) && (PowerStatusOb.previous_CapStatus != 2))//charging
and not full before
  {
   Relais_PBI.TurnOff(0);//disconnect PB Input immediatly
   Relais_PBO.TurnOff(500);//connect Tonuino to Powerbank with safety delay
  }
  else if((PowerStatusOb.CapStatus == 1) && (PowerStatusOb.previous_CapStatus == 2))//charging
and full before (additional charge due to too much current)
  {
   //keep everything; it must do something to trigger runState=2
   Relais_PBO.TurnOn(500);//connect Tonuino to external power after safety delay
   if (Relais_PBO.RelaisState==HIGH) //safety: only activate charging PB when PB Output is
disconnected!
   {
    Relais_PBI.TurnOn(700);//charge powerbank with additional safety delay
   }
  }

  else if(PowerStatusOb.CapStatus == 2)//full
  {
   Relais_PBO.TurnOn(500);//connect Tonuino to external power after safety delay
   if (Relais_PBO.RelaisState==HIGH) //safety: only activate charging PB when PB Output is
disconnected!
   {
    Relais_PBI.TurnOn(700);//charge powerbank with additional safety delay
   }
  }
  else if(PowerStatusOb.CapStatus == 3)//discharging
  {
   Relais_PBI.TurnOff(0);//disconnect PB Input immediatly
   if (Relais_PBO.RelaisState==HIGH) //safety: only activate pulsing PB when PB Output is
disconnected!
   {
    PBWake.Pulse(1000,400);//wake up Powerbank before connecting to PB Output
   }
   Relais_PBO.TurnOff(2000);//connect Tonuino to Powerbank with safety delay
  }
  if((Relais_PBI.runState==2) && (Relais_PBO.runState==2))// only if all actions are finished
  {
   PowerStatusOb.StatusChanged=LOW;//leave if and go back to readStatus mode until next event
is triggered
   Relais_PBI.runState=0;
   Relais_PBO.runState=0;

   PBWake.runState=1;
   PBWake.TurnOff(0);//safety to really turn off OC in every case
   PBWake.runState=0;
  }
 }
 else
 {
  PowerStatusOb.readStatus();
 }
}
```