INVENTION DISCLOSURE

03/13/18

TITLE

"Thermal Fuse Breaker for Smart Socket"

INVENTORS

James G Hagerman, 1521 Alexander St Apt 1701 Honolulu HI 96822
Assigned to Ibis Networks, 841 Bishop St Ste 1601 Honolulu HI 96813

REDUCTION TO PRACTICE

Idea conceived on 7/20/17, working prototype the next day.

BACKGROUND

Electrical fires are common. Many are caused by heat generated from a poor electrical connection at the plug / outlet interface. Causes for bad conductivity include corrosion, small contact area, weak contact pressure, and partial disconnection. A failed contact becomes resistive, which leads to excessive heat dissipation during high current loads. Extension cords, power strips, and smart sockets all may succumb to such abnormal operating conditions resulting in a meltdown.

ABSTRACT

This invention seeks to minimize or prevent consequential damage due to electrical fires caused by failing electrical connections via elimination of load before temperatures become dangerous. In essence, the invention is a thermally triggered circuit breaker, contained within a smart socket.

PURPOSE

The purpose of this invention is to prevent socket failure, meltdowns, fires, or any such damage possible from a failed plug / outlet electrical connection. By providing early detection of a potentially hazardous condition, plugs and outlets can be replaced prior to any damaging event. This is of great value to the facility and maintenance of buildings.

DETAILED DESCRIPTION

Necessary for this invention is a fast and accurate thermal measurement of electrical contact materials. Both input and output contacts must be monitored. A poor connection leads to abnormal conditions of high contact resistance, resulting in lost power converted to heat. Power dissipation is given by:

$$P = I^2 \cdot R$$

Being proportional to current-squared, the problem only manifests at higher currents. Example high current loads would be space heaters, microwave ovens, vacuum cleaners, large vending machine, ice makers, etc. During a failure condition, excessive heat can melt solder, plastic enclosures, circuit boards, and emit smoke.

Key to this invention is shutting off load current before temperatures get too high. A smart socket can do this by turning off the output relay when thermal measurements exceed a given threshold. The threshold must be higher than what occurs during normal operation, and less than the melting point of solder or plastic. Given that the maximal temperature rise allowed by UL (Underwriters Laboratory) is 30C, and maximum operating ambient for a typical smart socket is 40C, we selected a threshold of 80C. The extra 10C margin is to reduce false positives.

Reduction to practice of this invention does not require direct measurement of contact temperature, but rather something indicative and proportional. In the case of a smart socket, the microprocessor that controls operation is mounted on a circuit board which is also soldered directly to the brass plug / outlet contacts. Via conduction, the processor will rise in temperature proportional to the contacts. Using the internal thermometer of the processor, an algorithm can be applied to report temperatures at regular intervals and automatically turn off the relay when needed.

An application (such as the www.ibis.io cloud service) can monitor smart socket temperatures on a regular basis and determine which sockets may be at risk. Additionally, when a smart socket exceeds the threshold, an alert will be sent that its load has been turned off. This alert supplies notification of said abnormal condition, whereupon corrective remedies can be taken.

A smart socket can be turned on again after tripping, which would just lead to another tripping event. This is acceptable. The smart socket remains undamaged and continues to operate and maintain safe conditions.

PRIOR ART

It is not known if other smart sockets employ such an algorithm.

DRAWINGS

Drawing 1: Photo of example smart socket internal structure

```
// check temperature
MyCelsius = ReadCelsius() - MyCelsiusOffset;
if(MyCelsius > CELSIUS_TRIP_LEVEL)
{
    // overheated!
    sprintf((char *)MyString, "!Overheated!\r\n");
    WRITE_STRING();
    MyState = 0x00;
    SetState();
    SendAck(GET_CELSIUS, (uint32)MyCelsius);
}
```

Drawing 2: Code snippet showing thermal sense and turn-off of relay

```
//////////////////////////////////////////////////////////////////////////////
// read internal temperature sensor

uint16 ReadCelsius(void)
{
   int16 reading;

   // select internal temp as input to adc, resolution 10 bits
   TR0 = 0x01;
   ATEST = 0x01;
   ADCCON3 = 0x2E;

   // wait for conversion to complete
   while (!(ADCCON1 & 0x80));

   // read result
   reading = ADCL;
   reading = reading | (ADCH << 8);

   // ignore negative
   if(reading < 0)reading = 0;

   // shift to 10 bits
   reading = reading >> 6;

   return(reading);
}
```

Drawing 3: Code snippet for reading temperature of processor chip