#include "math.h"

const int currentPin0 = A0;

const int currentPin1 = A1;

const unsigned long sampleTime = 100000UL; // sample over 100ms, it is an exact number of cycles for both 50Hz and 60Hz mains

const unsigned long numSamples = 250UL; // choose the number of samples to divide sampleTime exactly, but low enough for the ADC to keep up

const unsigned long sampleInterval = sampleTime/numSamples; // the sampling interval, must be longer than then ADC conversion time

//const int adc\_zero = 522; // relative digital zero of the arudino input from ACS712 (could make this a variable and auto-adjust it)

const int washerVoltageThreshold = 5; //number of volts that must be detected before it is considered active

const int dryerVoltageThreshold = 5; //number of volts that must be detected before it is considered active

const int washerAlertThreshold = 300; //number of seconds to wait for no current detection before sending alert

const int dryerAlertThreshold = 10; //number of seconds to wait for no current detection before sending alert

const int washerStartWaitTime = 5; //number of seconds to wait before considering the device to be on. Sometimes temporary voltage is detected when the device is not on.

const int dryerStartWaitTime = 5; //number of seconds to wait before considering the device to be on. Sometimes temporary voltage is detected when the device is not on.

int washerAlertThresholdCounter = washerAlertThreshold; //set Counter to threshold

int dryerAlertThresholdCounter = dryerAlertThreshold; //set counter to threshold

int washerStartWaitTimeCounter = 0;

int dryerStartWaitTimeCounter = 0;

int adc\_zero0; //autoadjusted relative digital zero

int adc\_zero1;

bool washerAlertPrimed = false;

bool dryerAlertPrimed = false;

float currentWasherReading;

float currentDryerReading;

void setup()

{

Serial.begin(9600);

adc\_zero0 = determineVQ(currentPin0); //Quiscent output voltage - the average voltage ACS712 shows with no load on plug 1

adc\_zero1 = determineVQ(currentPin1); //Quiscent output voltage - the average voltage ACS712 shows with no load on plug 2

delay(1000);

Spark.publish("DryerEvent", "Power On - Washer and dryer notification system is online!", 60, PRIVATE);

}

void loop(){

//Plug1 Washer

//get current reading

currentWasherReading = readCurrent(currentPin0,adc\_zero0);

//Check Current Washer Reading

if (currentWasherReading >= washerVoltageThreshold){

//Check to see if the alert is already primed

if(washerAlertPrimed == true){

//Reset countdown timer

washerAlertThresholdCounter = washerAlertThreshold;

} else {

//Increment Prime Counter

washerStartWaitTimeCounter = washerStartWaitTimeCounter + 1;

//Check to see if washer is now primed

if(washerStartWaitTimeCounter >= washerStartWaitTime){

//set prime to true

washerAlertPrimed = true;

}

}//end check if alert is primed

} else {

//Check to see if alert is already primed

if(washerAlertPrimed == false){

//Reset Prime Counter

washerStartWaitTimeCounter = 0;

} else {

//check to see if countdown is > 0

if(washerAlertThresholdCounter > 0){

washerAlertThresholdCounter = washerAlertThresholdCounter - 1;

} else {

//Send push notification

//Serial.println("Send Push Notification");

Spark.publish("WasherEvent", "Washer Is Done!", 60, PRIVATE);

//Set prime to false

washerAlertPrimed = false;

washerAlertThresholdCounter = washerAlertThreshold;

}

}

} //end check Current Washer Reading

//Plug2 Dryer

//get current reading

currentDryerReading = readCurrent(currentPin1,adc\_zero1);

//Check Current Dryer Reading

if (currentDryerReading >= dryerVoltageThreshold){

//Check to see if the alert is already primed

if(dryerAlertPrimed == true){

//Reset countdown timer

dryerAlertThresholdCounter = dryerAlertThreshold;

} else {

//Increment Prime Counter

dryerStartWaitTimeCounter = dryerStartWaitTimeCounter + 1;

//Check to see if dryer is now primed

if(dryerStartWaitTimeCounter >= dryerStartWaitTime){

//set prime to true

dryerAlertPrimed = true;

}

}//end check if alert is primed

} else {

//Check to see if alert is already primed

if(dryerAlertPrimed == false){

//Reset Prime Counter

dryerStartWaitTimeCounter = 0;

} else {

//check to see if countdown is > 0

if(dryerAlertThresholdCounter > 0){

dryerAlertThresholdCounter = dryerAlertThresholdCounter - 1;

} else {

//Send push notification

//Serial.println("Send Push Notification");

Spark.publish("DryerEvent", "Dryer Is Done!", 60, PRIVATE);

//Set prime to false

dryerAlertPrimed = false;

dryerAlertThresholdCounter = dryerAlertThreshold;

}

}

} //end check Current Dryer Reading

//Serial.print("V="); Serial.print(currentWasherReading);

//Serial.print(" Primed="); Serial.print(washerAlertPrimed);

//Serial.print(" ACount="); Serial.print(washerAlertThresholdCounter);

//Serial.print(" WCount="); Serial.print(washerStartWaitTimeCounter);

//Serial.print(" V="); Serial.print(currentDryerReading);

//Serial.print(" Primed="); Serial.print(dryerAlertPrimed);

//Serial.print(" ACount="); Serial.print(dryerAlertThresholdCounter);

//Serial.print(" WCount="); Serial.print(dryerStartWaitTimeCounter);

//Serial.println();

//wait one second before checking current again

delay(1000);

} //end of main loop

int determineVQ(int PIN) {

Serial.print("estimating avg. quiscent voltage:");

long VQ = 0;

//read 5000 samples to stabilise value

for (int i=0; i<5000; i++) {

VQ += analogRead(PIN);

delay(1);//depends on sampling (on filter capacitor), can be 1/80000 (80kHz) max.

}

VQ /= 5000;

Serial.print(map(VQ, 0, 1023, 0, 5000));Serial.println(" mV");

return int(VQ);

}

float readCurrent(int PIN, int adc\_zeroed)

{

unsigned long currentAcc = 0;

unsigned int count = 0;

unsigned long prevMicros = micros() - sampleInterval ;

while (count < numSamples)

{

if (micros() - prevMicros >= sampleInterval)

{

int adc\_raw = analogRead(PIN) - adc\_zeroed;

currentAcc += (unsigned long)(adc\_raw \* adc\_raw);

++count;

prevMicros += sampleInterval;

}

}

float rms = sqrt((float)currentAcc/(float)numSamples) \* (75.7576 / 1024.0);

return rms;

//Serial.println(rms);

}