```
C++ code ESP32freezer.ino to run on ESP32
#include <Arduino.h>
#include <string.h>
#include <stdint.h>
#include <WiFi.h>
#include <HTTPClient.h>
#define LED 4
#define AdcIn 36
#define AC_SurgeSensor 33
#define AC_SurgeInterrupt GPIO_NUM_33

void setup() { // all code is in setup
  char codefile[] ="ESP32freezer"; // on ezsbc board
  char webHostUrl[] = "http://webhost.com/dev/smallfreezer.php";
  char cstr[20];
  String myPost = " ";
  String minutesBetweenPostsStr = "";
  int delayNextSleep = 1000*60*20; //  20 minutes between
measurements
betweeen post
  long adcAccumulator = 0;
  int thermistorVal = 0;
  analogSetAttenuation(ADC_6db);
  pinMode (AdcIn, INPUT); //Thermistor analog input
  pinMode (AC_SurgeSensor, INPUT); // voltage src for resistor to adc
  Serial.begin(115200);
  delay(500);
  Serial.println(" ");
  Serial.println(codefile);

  // Flash LED a few times at start and every wakeup.
  pinMode(LED, OUTPUT);
  blinkLed(3);

  adcAccumulator=0;
  for(int i=0; i<20; i++ ){ // take the average of 20 readings
    delay(50);
    adcAccumulator=adcAccumulator+analogRead(AdcIn);
  }
  thermistorVal=adcAccumulator/20; //create average of 20 loops
  Serial.println(thermistorVal);
```

```
  mySetUpWiFi(); // this is done after measurment to minimize noise
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;

    //prep post message to website
    String adcLabel="&adc="; //labels for post message arguments
    http.begin(webHostUrl); //Specify request destination
    http.addHeader("Content-Type",
"application/x-www-form-urlencoded; charset=UTF-8");
    myPost=myPost + adcLabel;
    myPost=myPost + thermistorVal;
    Serial.print("myPost=");
    Serial.println(myPost); //Print post

    int httpCode = http.POST(myPost); //Send the request

    //Print message received back from website (payload) to debug
    String payload = http.getString(); //Get the response payload,
even though we do nothing with it.  Serial print for debug
    Serial.print("payload=");
    Serial.println(payload); //Print request response payload

    http.end(); //Close connection
  } else {
    Serial.println("Error in WiFi connection");
  }

  delay (delayNextSleep);  // wait before testing again
// And we are done this time through
  esp_sleep_enable_ext0_wakeup(AC_SurgeInterrupt,1); //1 = High, 0 =
Low esp_sleep_enable_ext0_wakeup(GPIO_NUM_33,1); //1 = High, 0 = Low
  Serial.println("GOING TO SLEEP ======================");
  esp_deep_sleep_start();
}
```