/\* How to use the DHT-22 sensor with Arduino uno

Temperature and humidity sensor

\*/

//Libraries

#include <DHT.h>

#include <HttpClient.h>

#include <Bridge.h>

#include <LiquidCrystal\_I2C.h>

#include <SoftwareSerial.h>

#include <LiquidCrystal.h>

#include <stdlib.h>

#include <Adafruit\_Sensor.h>

#include <Adafruit\_BMP280.h>

//Constants

#define DHTPIN 7 // what pin we're connected to

#define DHTTYPE DHT22 // DHT 22 (AM2302)

DHT dht(DHTPIN, DHTTYPE); //// Initialize DHT sensor for normal 16mhz Arduino

const int buzzer = 10;

//Temp vars

int chk;

float hum; //Stores humidity value

float temp; //Stores temperature value

//Light vars

int light;

//barometer vars

Adafruit\_BMP280 bmp; // I2C

//Dust vars

int pin = 8;

unsigned long duration;

unsigned long starttime;

unsigned long sampletime\_ms = 2000;

unsigned long lowpulseoccupancy = 0;

float ratio = 0;

float concentration = 0;

HttpClient client; // For talking with the thingspeak api

// Set the LCD address to 0x27 for a 16 chars and 2 line display

LiquidCrystal\_I2C lcd(0x27, 16, 2);

//Variables

int error;

int pulsePin = 1; // Pulse Sensor purple wire connected to analog pin 1

int blinkPin = 13; // pin to blink led at each beat

int redIndicatorPin = 4;

int greenIndicatorPin = 5;

// Volatile Variables, used in the interrupt service routine!

volatile int BPM; // int that holds raw Analog in 0. updated every 2mS

volatile int Signal; // holds the incoming raw data

volatile int IBI = 600; // int that holds the time interval between beats! Must be seeded!

volatile boolean Pulse = false; // "True" when heartbeat is detected. "False" when not a "live beat".

volatile boolean QS = false; // becomes true when Arduino finds a beat.

// Regards Serial OutPut -- Set This Up to your needs

static boolean serialVisual = true; // Set to 'false' by Default. Re-set to 'true' to see Arduino Serial Monitor ASCII Visual Pulse

volatile int rate[10]; // array to hold last ten IBI values

volatile unsigned long sampleCounter = 0; // used to determine pulse timing

volatile unsigned long lastBeatTime = 0; // used to find IBI

volatile int P =512; // used to find peak in pulse wave, seeded

volatile int T = 512; // used to find trough in pulse wave, seeded

volatile int thresh = 525; // used to find instant moment of heart beat, seeded

volatile int amp = 100; // used to hold amplitude of pulse waveform, seeded

volatile boolean firstBeat = true; // used to seed rate array so we startup with reasonable BPM

volatile boolean secondBeat = false; // used to seed rate array so we startup with reasonable BPM

void setup()

{

Bridge.begin();

pinMode(redIndicatorPin, OUTPUT);

pinMode(greenIndicatorPin, OUTPUT);

// LCD

lcd.begin();

lcd.setBacklight((uint8\_t)1);

lcd.print("Hello!");

delay(100);

lcd.setCursor(0,1);

lcd.print("Connecting...");

temp\_hum\_setup();

light\_setup();

dust\_setup();

barometer\_setup();

Serial.begin(9600);

delay(1000);

interruptSetup();

}

void loop()

{

temp\_hum\_loop();

light\_loop();

dust\_loop();

barometer\_loop();

update\_all();

lcd\_out();

}

void temp\_hum\_setup()

{

dht.begin();

}

void temp\_hum\_loop() {

delay(500);

//Read data and store it to variables hum and temp

hum = dht.readHumidity();

temp= dht.readTemperature();

//Print temp and humidity values to serial monitor

Serial.print("Humidity: ");

Serial.print(hum);

Serial.print(" %, Temp: ");

Serial.print(temp);

Serial.println(" Celsius");

delay(500); //Delay 2 sec.

}

void barometer\_setup() {

Serial.println(F("BMP280 test"));

while(!bmp.begin()) {

Serial.println(F("Could not find a valid BMP280 sensor, check wiring!"));

}

}

void barometer\_loop() {

Serial.print(F("Temperature = "));

Serial.print(bmp.readTemperature());

Serial.println(" \*C");

Serial.print(F("Pressure = "));

Serial.print(bmp.readPressure());

Serial.println(" Pa");

Serial.print(F("Approx altitude = "));

Serial.print(bmp.readAltitude(1013.25)); // this should be adjusted to your local forcase

Serial.println(" m");

Serial.println();

delay(2000);

}

void light\_setup() {

pinMode(A0,INPUT);

}

void light\_loop()

{

light=analogRead(A0);

Serial.println("Light = " + String(light));

delay(500);

if(light < 100){

digitalWrite(greenIndicatorPin, HIGH);

digitalWrite(redIndicatorPin, LOW);

}

else if (light > 100){

digitalWrite(greenIndicatorPin, LOW);

digitalWrite(redIndicatorPin, HIGH);

}

else {

digitalWrite(greenIndicatorPin, LOW);

digitalWrite(redIndicatorPin, LOW);

}

}

void dust\_setup(){

pinMode(8,INPUT);

starttime = millis();

}

void dust\_loop() {

duration = pulseIn(pin, LOW);

lowpulseoccupancy = lowpulseoccupancy+duration;

if ((millis()-starttime) >= sampletime\_ms) //if the sampel time = = 30s

{

ratio = lowpulseoccupancy/(sampletime\_ms\*10.0);

concentration = 1.1\*pow(ratio,3)-3.8\*pow(ratio,2)+520\*ratio+0.62;

Serial.print("Concentration = ");

Serial.print(concentration);

Serial.println(" pcs/0.01cf");

Serial.println("\n");

lowpulseoccupancy = 0;

starttime = millis();

}

}

void lcd\_out() {

lcd.backlight();

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Humidity: ");

lcd.print(hum);

delay(1000);

lcd.clear();

lcd.setCursor(0,0);

lcd.print(" %, Temp: ");

lcd.print(temp);

lcd.print(" Celsius");

delay(1000);

lcd.clear();

lcd.setCursor(0,0);

lcd.print(F("Pressure = "));

lcd.print(bmp.readPressure());

lcd.print(" Pa");

delay(1000);

lcd.clear();

lcd.setCursor(0,0);

lcd.print(F("Approx altitude = "));

lcd.print(bmp.readAltitude(1013.25)); // this should be adjusted to your local forcase

lcd.print(" m");

delay(1000);

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Light = " + String(light));

delay(1000);

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Concentration = ");

lcd.setCursor(0,1);

lcd.print(concentration);

lcd.print(" pcs/0.01cf");

delay(1000);

lcd.clear();

lcd.setCursor(0,0);

lcd.print("BPM = ");

lcd.setCursoor(0,1);

lcd.print(BPM);

lcd.print(" BPM");

delay(1000);

}

void update\_all(){

String request = "http://api.thingspeak.com/update?key=39A4LSNDUISGO0UJ&field1=" + String(temp)+"&field2=" +String(hum) + "&field3=" + String(bmp.readPressure()) + "&field4=" + String(light)+ "&field5="+ String(concentration)+"&field6=" +String(BPM);

client.get(request);

}

void interruptSetup(){

TCCR1A = 0x00; // DISABLE PWM ON DIGITAL PINS 3 AND 11, AND GO INTO CTC MODE

TCCR1B = 0x0C; // DON'T FORCE COMPARE, 256 PRESCALER

OCR1A = 0x7C; // SET THE TOP OF THE COUNT TO 124 FOR 500Hz SAMPLE RATE

TIMSK1 = 0x02; // ENABLE INTERRUPT ON MATCH BETWEEN TIMER2 AND OCR2A

sei(); // MAKE SURE GLOBAL INTERRUPTS ARE ENABLED

}

ISR(TIMER1\_COMPA\_vect){ // triggered when Timer2 counts to 124

// Serial.println("ENTER ISR");

cli(); // disable interrupts while we do this

Signal = analogRead(pulsePin); // read the Pulse Sensor

sampleCounter += 2; // keep track of the time in mS

int N = sampleCounter - lastBeatTime; // monitor the time since the last beat to avoid noise

// find the peak and trough of the pulse wave

if(Signal < thresh && N > (IBI/5)\*3){ // avoid dichrotic noise by waiting 3/5 of last IBI

if (Signal < T){ // T is the trough

T = Signal; // keep track of lowest point in pulse wave

}

}

if(Signal > thresh && Signal > P){ // thresh condition helps avoid noise

P = Signal; // P is the peak

} // keep track of highest point in pulse wave

// NOW IT'S TIME TO LOOK FOR THE HEART BEAT

// signal surges up in value every time there is a pulse

if (N > 250){ // avoid high frequency noise

if ( (Signal > thresh) && (Pulse == false) && (N > (IBI/5)\*3) ){

// Serial.println("Enter main calc");

Pulse = true; // set the Pulse flag when there is a pulse

digitalWrite(blinkPin,HIGH); // turn on pin 13 LED

IBI = sampleCounter - lastBeatTime; // time between beats in mS

lastBeatTime = sampleCounter; // keep track of time for next pulse

if(secondBeat){ // if this is the second beat

secondBeat = false; // clear secondBeat flag

for(int i=0; i<=9; i++){ // seed the running total to get a realistic BPM at startup

rate[i] = IBI;

}

}

if(firstBeat){ // if it's the first time beat is found

firstBeat = false; // clear firstBeat flag

secondBeat = true; // set the second beat flag

sei(); // enable interrupts again

return; // IBI value is unreliable so discard it

}

word runningTotal = 0; // clear the runningTotal variable

for(int i=0; i<=8; i++){ // shift data in the rate array

rate[i] = rate[i+1]; // and drop the oldest IBI value

runningTotal += rate[i]; // add up the 9 oldest IBI values

}

rate[9] = IBI; // add the latest IBI to the rate array

runningTotal += rate[9]; // add the latest IBI to runningTotal

runningTotal /= 10; // average the last 10 IBI values

BPM = 60000/runningTotal; // how many beats can fit into a minute? that's BPM!

QS = true; // set Quantified Self flag

// QS FLAG IS NOT CLEARED INSIDE THIS ISR

}

}

if (Signal < thresh && Pulse == true){ // when the values are going down, the beat is over

digitalWrite(blinkPin,LOW); // turn off pin 13 LED

Pulse = false; // reset the Pulse flag so we can do it again

amp = P - T; // get amplitude of the pulse wave

thresh = amp/2 + T; // set thresh at 50% of the amplitude

P = thresh; // reset these for next time

T = thresh;

}

if (N > 2500){ // if 2.5 seconds go by without a beat

thresh = 512; // set thresh default

P = 512; // set P default

T = 512; // set T default

lastBeatTime = sampleCounter; // bring the lastBeatTime up to date

firstBeat = true; // set these to avoid noise

secondBeat = false; // when we get the heartbeat back

}

sei();

// enable interrupts when youre done!

}// end isr