# MY SENSORS CODES FOR ARK-1

## FOR VEML6075 UV SENSOR

#include <Wire.h>

// VEML6075 I2C address

#define VEML6075\_ADDR 0x10

// Register addresses

#define VEML6075\_CONF 0x00

#define VEML6075\_UVA 0x07

#define VEML6075\_DUMMY 0x08

#define VEML6075\_UVB 0x09

#define VEML6075\_COMP1 0x0A

#define VEML6075\_COMP2 0x0B

#define VEML6075\_ID 0x0C

void writeRegister(uint8\_t reg, uint16\_t value) {

Wire.beginTransmission(VEML6075\_ADDR);

Wire.write(reg);

Wire.write(value & 0xFF); // Low byte

Wire.write((value >> 8) & 0xFF); // High byte

Wire.endTransmission();

}

uint16\_t readRegister(uint8\_t reg) {

Wire.beginTransmission(VEML6075\_ADDR);

Wire.write(reg);

Wire.endTransmission(false);

Wire.requestFrom(VEML6075\_ADDR, (uint8\_t)2);

uint16\_t low = Wire.read();

uint16\_t high = Wire.read();

return (high << 8) | low;

}

void setup() {

Serial.begin(115200);

Wire.begin(); // SDA, SCL

delay(100);

// Check device ID

uint16\_t id = readRegister(VEML6075\_ID);

Serial.print("VEML6075 ID: ");

Serial.println(id, HEX);

if (id != 0x26) {

Serial.println("ERROR: Wrong ID, check wiring!");

while (1);

}

// Configure sensor

// Bits: [13:12] = Integration time, [0] = Active mode enable

// Example: IT = 100ms (01), Active mode ON (bit0=0)

uint16\_t config = 0x00; // default

config |= (0x01 << 4); // Integration time = 100ms

config &= ~(0x01); // 0 = Active mode

writeRegister(VEML6075\_CONF, config);

Serial.println("VEML6075 Initialised");

}

void loop() {

uint16\_t UVA = readRegister(VEML6075\_UVA);

uint16\_t UVB = readRegister(VEML6075\_UVB);

uint16\_t COMP1 = readRegister(VEML6075\_COMP1);

uint16\_t COMP2 = readRegister(VEML6075\_COMP2);

// Apply compensation (simplified, proper coeffs from datasheet)

float UVA\_corr = UVA - (COMP1 \* 0.5) - (COMP2 \* 0.5);

float UVB\_corr = UVB - (COMP1 \* 0.5) - (COMP2 \* 0.5);

if (UVA\_corr < 0) UVA\_corr = 0;

if (UVB\_corr < 0) UVB\_corr = 0;

// Calculate UV Index (datasheet coefficients vary)

float UV\_index = (UVA\_corr \* 0.00229) + (UVB\_corr \* 0.00274);

Serial.print("UVA: "); Serial.print(UVA\_corr);

Serial.print(" UVB: "); Serial.print(UVB\_corr);

Serial.print(" UV Index: "); Serial.println(UV\_index);

delay(500); // 2 Hz sampling

}

## FOR MAGNETOMETER LIS2MDLTR

#include <Wire.h>

#include <LIS2MDL.h> // Magnetometer library (ST’s LIS2MDL)

// Create sensor object

LIS2MDL mag;

void setup() {

Serial.begin(115200);

// Start I2C on your custom pins

Wire.begin(36, 37); // SDA = IO36, SCL = IO37

Serial.println("I2C bus started on IO36(SDA), IO37(SCL)");

// Initialise LIS2MDL

if (!mag.begin(&Wire)) {

Serial.println("Failed to detect LIS2MDL magnetometer!");

while (1);

}

Serial.println("LIS2MDL initialised.");

}

void loop() {

// Read magnetic field values

sensors\_event\_t event;

mag.getEvent(&event);

Serial.print("Magnetic field [uT] X:");

Serial.print(event.magnetic.x);

Serial.print(" Y:");

Serial.print(event.magnetic.y);

Serial.print(" Z:");

Serial.println(event.magnetic.z);

delay(500);

}

## FOR SHT31-DIS-B-TEMPREATURE SENSOR

#include <Wire.h>

#define SDA\_PIN 36

#define SCL\_PIN 37

#define SHT31\_ADDR 0x44 // I2C address for SHT31-DIS

// Function to calculate CRC-8 (poly = 0x31, init = 0xFF)

uint8\_t calculateCRC(uint8\_t \*data, uint8\_t len) {

uint8\_t crc = 0xFF;

for (uint8\_t i = 0; i < len; i++) {

crc ^= data[i];

for (uint8\_t j = 0; j < 8; j++) {

if (crc & 0x80)

crc = (crc << 1) ^ 0x31;

else

crc <<= 1;

}

}

return crc;

}

void setup() {

Serial.begin(115200);

delay(1000);

Wire.begin(SDA\_PIN, SCL\_PIN);

Serial.println("ESP32-S3 + SHT31-DIS-B sensor test...");

// Soft reset SHT31

Wire.beginTransmission(SHT31\_ADDR);

Wire.write(0x30);

Wire.write(0xA2);

Wire.endTransmission();

delay(20);

}

void loop() {

// Send single-shot high repeatability measurement (0x2400)

Wire.beginTransmission(SHT31\_ADDR);

Wire.write(0x24);

Wire.write(0x00);

Wire.endTransmission();

delay(15); // Measurement time (typ ~15ms)

// Read 6 bytes (T\_MSB, T\_LSB, CRC\_T, RH\_MSB, RH\_LSB, CRC\_RH)

Wire.requestFrom(SHT31\_ADDR, (uint8\_t)6);

if (Wire.available() == 6) {

uint8\_t t\_msb = Wire.read();

uint8\_t t\_lsb = Wire.read();

uint8\_t t\_crc = Wire.read();

uint8\_t rh\_msb = Wire.read();

uint8\_t rh\_lsb = Wire.read();

uint8\_t rh\_crc = Wire.read();

uint8\_t t\_data[2] = {t\_msb, t\_lsb};

uint8\_t rh\_data[2] = {rh\_msb, rh\_lsb};

if (t\_crc == calculateCRC(t\_data, 2) && rh\_crc == calculateCRC(rh\_data, 2)) {

uint16\_t t\_raw = ((uint16\_t)t\_msb << 8) | t\_lsb;

uint16\_t rh\_raw = ((uint16\_t)rh\_msb << 8) | rh\_lsb;

// Conversion formulas

float temperature = -45.0 + (175.0 \* ((float)t\_raw / 65535.0));

float humidity = 100.0 \* ((float)rh\_raw / 65535.0);

// Print to Serial (can be replaced with LoRa / Echo comms)

Serial.print("Temperature: ");

Serial.print(temperature, 2);

Serial.print(" °C | Humidity: ");

Serial.print(humidity, 2);

Serial.println(" %RH");

} else {

Serial.println("CRC error in sensor data!");

}

} else {

Serial.println("Failed to read from SHT31.");

}

delay(1000); // Repeat every second

}

## FOR BMA400 ACCLEROMETER SENSOR

#include <Wire.h>

#define SDA\_PIN 36

#define SCL\_PIN 37

#define BMA400\_I2C\_ADDRESS 0x14 // Can also be 0x15 depending on pin config

// BMA400 registers

#define BMA400\_CHIP\_ID\_REG 0x00

#define BMA400\_CHIP\_ID 0x90

#define BMA400\_ACC\_X\_LSB 0x04

#define BMA400\_ACC\_CONFIG0 0x19

#define BMA400\_ACC\_CONFIG1 0x1A

#define BMA400\_PWR\_CONF 0x7C

#define BMA400\_PWR\_CTRL 0x7D

// Current g-range = ±2g → sensitivity = 0.000976 g/LSB

#define BMA400\_SENS\_2G 0.000976

void writeRegister(uint8\_t reg, uint8\_t value) {

Wire.beginTransmission(BMA400\_I2C\_ADDRESS);

Wire.write(reg);

Wire.write(value);

Wire.endTransmission();

}

uint8\_t readRegister(uint8\_t reg) {

Wire.beginTransmission(BMA400\_I2C\_ADDRESS);

Wire.write(reg);

Wire.endTransmission(false);

Wire.requestFrom(BMA400\_I2C\_ADDRESS, (uint8\_t)1);

return Wire.read();

}

void readAcceleration(float &ax, float &ay, float &az) {

Wire.beginTransmission(BMA400\_I2C\_ADDRESS);

Wire.write(BMA400\_ACC\_X\_LSB);

Wire.endTransmission(false);

Wire.requestFrom(BMA400\_I2C\_ADDRESS, (uint8\_t)6);

int16\_t rawX = (int16\_t)(Wire.read() | (Wire.read() << 8));

int16\_t rawY = (int16\_t)(Wire.read() | (Wire.read() << 8));

int16\_t rawZ = (int16\_t)(Wire.read() | (Wire.read() << 8));

// Convert raw values → g

ax = rawX \* BMA400\_SENS\_2G;

ay = rawY \* BMA400\_SENS\_2G;

az = rawZ \* BMA400\_SENS\_2G;

}

void setup() {

Serial.begin(115200);

Wire.begin(SDA\_PIN, SCL\_PIN);

delay(100);

uint8\_t chipID = readRegister(BMA400\_CHIP\_ID\_REG);

if (chipID == BMA400\_CHIP\_ID) {

Serial.println("BMA400 detected!");

} else {

Serial.print("BMA400 not found! Read ID: 0x");

Serial.println(chipID, HEX);

while (1);

}

// Set accelerometer configuration

writeRegister(BMA400\_ACC\_CONFIG0, 0x00); // ±2g range, OSR = 0

writeRegister(BMA400\_ACC\_CONFIG1, 0x14); // 100 Hz ODR

writeRegister(BMA400\_PWR\_CONF, 0x00); // Enable power

writeRegister(BMA400\_PWR\_CTRL, 0x04); // Enable accelerometer

delay(50);

}

void loop() {

float ax, ay, az;

readAcceleration(ax, ay, az);

Serial.print("X: ");

Serial.print(ax, 3);

Serial.print(" g, Y: ");

Serial.print(ay, 3);

Serial.print(" g, Z: ");

Serial.print(az, 3);

Serial.println(" g");

delay(200);

}