am2315c(parentSense) {}

AOSongAM2315\_Temp()

: Variable((const uint8\_t)AM2315\_TEMP\_VAR\_NUM,

(uint8\_t)AM2315\_TEMP\_RESOLUTION, AM2315\_TEMP\_VAR\_NAME,

AM2315\_TEMP\_UNIT\_NAME, AM2315\_TEMP\_DEFAULT\_CODE) {}

The AM2315C::begin() function is commented out in your code. This function is typically used to initialize the sensor and it's likely that the rest of your code depends on it. If it's not defined, you may get errors when you try to use the sensor.

Uncomment the AM2315C::begin() function:

bool AM2315C::begin() {

    if (!am2315ptr->begin()) {

        Serial.println("Could not find a valid AM2315C sensor, check wiring!");

        return false;

    }

    Serial.println("AM2315C sensor initialized successfully!");

    return true;

}

Also, the constructors for the AM2315C class are commented out. If you're trying to create an instance of AM2315C elsewhere in your code, this could be causing errors. Uncomment the constructors if they are needed:

AM2315C::AM2315C(signed char powerPin, unsigned char i2cAddress) {

    // Initialize your class members here

}

AM2315C::AM2315C(TwoWire\* wire) {

    // Initialize your object with the wire parameter...

}

In the addSingleMeasurementResult function, you have declared getTemperature and getHumidity as floats, but they seem to be functions. If they are functions, they should be declared like this:

float getTemperature(void);

float getHumidity(void);

And then used like this:

float temp = getTemperature();  // Declare and initialize temp

float hum = getHumidity();  // Declare and initialize hum

Finally, ensure that verifyAndAddMeasurementResult, \_millisMeasurementRequested, and \_sensorStatus are properly declared and defined in your class. If they are not, you will need to add their declarations to your AM2315C class definition and define them appropriately.

class AOSongAM2315\_Humidity : public Variable {

public:

AM2315C\* am2315c; // Add AM2315C as a member

explicit AOSongAM2315\_Humidity(

AM2315C\* parentSense, const char\* uuid = "",

const char\* varCode = AM2315\_HUMIDITY\_DEFAULT\_CODE)

: Variable((const uint8\_t)AM2315\_HUMIDITY\_VAR\_NUM,

(uint8\_t)AM2315\_HUMIDITY\_RESOLUTION,

AM2315\_HUMIDITY\_VAR\_NAME, AM2315\_HUMIDITY\_UNIT\_NAME, varCode,

uuid),

am2315c(parentSense) {} // Initialize AM2315C member in constructor

// ... /Users/brianjastram/Documents/PlatformIO/Projects/GreenHouse

Editing 8/28/2024 –

/\* clang-format on \*/

class AOSongAM2315\_Humidity : public Variable {

public:

AM2315C\* am2315c; // Add AM2315C as a member

explicit AOSongAM2315\_Humidity(

AM2315C\* parentSense, const char\* uuid = "",

const char\* varCode = AM2315\_HUMIDITY\_DEFAULT\_CODE)

: Variable((const uint8\_t)AM2315\_HUMIDITY\_VAR\_NUM,

(uint8\_t)AM2315\_HUMIDITY\_RESOLUTION,

AM2315\_HUMIDITY\_VAR\_NAME, AM2315\_HUMIDITY\_UNIT\_NAME, varCode,

uuid),

am2315c(parentSense) {} // Initialize AM2315C member in constructor

// ...

};

class AOSongAM2315\_Temp : public Variable {

public:

AM2315C\* am2315c; // Add AM2315C as a member

explicit AOSongAM2315\_Temp(

AM2315C\* parentSense, const char\* uuid = "",

const char\* varCode = AM2315\_TEMP\_DEFAULT\_CODE)

: Variable((const uint8\_t)AM2315\_TEMP\_VAR\_NUM,

(uint8\_t)AM2315\_TEMP\_RESOLUTION,

AM2315\_TEMP\_VAR\_NAME, AM2315\_TEMP\_UNIT\_NAME, varCode,

uuid),

am2315c(parentSense) {} // Initialize AM2315C member in constructor

// ...

/\*\*@}\*/

};

#endif // SRC\_SENSORS\_AOSONGAM2315\_H\_

After Lunch

/\*\* Start [ao\_song\_am2315] \*/

// #include </Users/brianjastram/Documents/PlatformIO/Projects/GreenHouse/.pio/libdeps/mayfly/AM2315C/AM2315c.h>

#include <Wire.h>

#include <sensors/AOSongAM2315.h>

// NOTE: Use -1 for any pins that don't apply or aren't being used.

const int8\_t AM2315Power = sensorPowerPin; // Power pin

// Create an AOSong AM2315 sensor object

TwoWire\* myI2C = &Wire;

int8\_t powerPin = AM2315Power;

uint8\_t measurementsToAverage = 3; // or any other value you want

// AM2315C am2315(myI2C);

/\*\*

\* @class AM2315C

\* @brief A class representing the AM2315C sensor.

\*

\* This class provides an interface to interact with the AM2315C sensor.

\* It allows reading temperature and humidity values from the sensor.

\*/

// AM2315C am2315(&Wire);

// Create an AM2315C object

AM2315C\* am2315 = new AM2315C(&Wire);

// Create an AM2315C\_Sensor object

AM2315C\_Sensor\* am2315Sensor = new AM2315C\_Sensor(am2315);

// Create the humidity and temperature variable pointers for the AM2315C

Variable\* am2315Humid = new AM2315C\_Humidity(am2315Sensor, "0af7a334-7200-4ade-853d-09c66afb6f58");

Variable\* am2315Temp = new AOSongAM2315\_Temp(am2315Sensor, "5c5f2d60-497c-4c34-b6fb-6c074a56b200");

class AM2315CWrapper {

public:

AM2315CWrapper(TwoWire \*wire, const char\* uuid) : am2315(wire), uuid(uuid) {}

float getHumidity() {

return am2315.getHumidity();

}

const char\* getUuid() {

return uuid;

}

private:

AM2315C am2315;

const char\* uuid;

};

Tuesday Sep. 3rd AOSongAM2315.cpp

AM2315C\_Sensor::AM2315C\_Sensor(signed char powerPin, unsigned char i2cAddress) {

// Initialize your class members here

}

AM2315C::AM2315C(TwoWire\* wire) {

// Initialize your object with the wire parameter...

}

AM2315C::~AM2315C() {}

String AM2315C::getSensorLocation(void) {

return F("I2C\_0x38");

}

bool AM2315C::begin() {

if (!am2315ptr->begin())

Serial.println("Could not find a valid AM2315C sensor, check wiring!");

return false;

// }

// Serial.println("AM2315C sensor initialized successfully!");

// return true;

//}

}

bool AM2315C::addSingleMeasurementResult(int varNum, float result) {

// float getTemperature();

// float getHumidity();

float temp = getTemperature(); // Declare and initialize temp

float hum = getHumidity(); // Declare and initialize hum

if (temp == -9999 || hum == -9999) {

Serial.println("Failed to read from AM2315C sensor!");

return false;

}

bool verifyAndAddMeasurementResult() {

// Implementation of the function goes here

// Return true if successful, false otherwise

return true;

}

verifyAndAddMeasurementResult(AM2315\_TEMP\_VAR\_NUM, temp);

verifyAndAddMeasurementResult(AM2315\_HUMIDITY\_VAR\_NUM, hum);

// Unset the time stamp for the beginning of this measurement

\_millisMeasurementRequested = 0;

// Unset the status bits for a measurement request (bits 5 & 6)

\_sensorStatus &= 0b10011111;

return true;

}