/\*\*

\* @file AOSongAM2315.h

\* @copyright 2017-2022 Stroud Water Research Center

\* Part of the EnviroDIY ModularSensors library for Arduino

\* @author Sara Geleskie Damiano <sdamiano@stroudcenter.org>

\*

\* @brief Contains the AOSongAM2315 sensor subclass and the variable subclasses

\* AOSongAM2315\_Humidity and AOSongAM2315\_Temp.

\*

\* These are used for the AOSong AM2315 capacitive humidity and temperature

\* sensor.

\*

\* This file is dependent on the Adafruit AM2315 Library.

\*/

/\* clang-format off \*/

/\*\*

\* @defgroup sensor\_am2315 AOSong AM2315

\* Classes for the AOSong AM2315 encased I2C capacitive humidity and

\* temperature sensor.

\*

\* @ingroup the\_sensors

\*

\* @tableofcontents

\* @m\_footernavigation

\*

\* @section sensor\_am2315\_notes Quick Notes

\* - Applies to both the AOSong AM2315 and CM2311 capacitive relative humidity

\* and temperature sensors

\* - Depends on the [Adafruit AM2315 Library](https://github.com/adafruit/Adafruit\_AM2315).

\* - Communicates via I2C

\* - only one address possible, 0xB8

\* - \*\*Only 1 can be connected to a single I2C bus at a time\*\*

\* - Requires a 3.3 - 5.5V power source

\*

\* @note Software I2C is \*not\* supported for the AM2315.

\* A secondary hardware I2C on a SAMD board is supported.

\*

\* @section sensor\_am2315\_datasheet Sensor Datasheet

\* [Datasheet](https://github.com/EnviroDIY/ModularSensors/wiki/Sensor-Datasheets/AOSong-AM2315-Product-Manual.pdf)

\*

\* @section sensor\_am2315\_ctor Sensor Constructors

\* {{ @ref AOSongAM2315::AOSongAM2315(int8\_t, uint8\_t) }}

\* {{ @ref AOSongAM2315::AOSongAM2315(TwoWire\*, int8\_t, uint8\_t) }}

\*

\* @section sensor\_am2315\_examples Example Code

\*

\* The AM2315 is used in the [double logger](@ref double\_logger.ino)

\* and @menulink{ao\_song\_am2315} example

\*

\* @menusnip{ao\_song\_am2315}

\*/

/\* clang-format on \*/

// Header Guards

#ifndef SRC\_SENSORS\_AOSONGAM2315\_H\_

#define SRC\_SENSORS\_AOSONGAM2315\_H\_

// Debugging Statement

// #define MS\_AOSONGAM2315\_DEBUG

#ifdef MS\_AOSONGAM2315\_DEBUG

#define MS\_DEBUGGING\_STD "AOSongAM2315"

#endif

// Included Dependencies

#include "ModSensorDebugger.h"

#undef MS\_DEBUGGING\_STD

#include "VariableBase.h"

#include "SensorBase.h"

#include <Adafruit\_AM2315.h>

#include <AM2315C.h>

/\*\* @ingroup sensor\_am2315 \*/

/\*\*@{\*/

// Sensor Specific Defines

/// @brief Sensor::\_numReturnedValues; the AM2315 can report 2 values.

#define AM2315\_NUM\_VARIABLES 2

/// @brief Sensor::\_incCalcValues; we don't calculate any additional values.

#define AM2315\_INC\_CALC\_VARIABLES 0

/\*\*

\* @anchor sensor\_am2315\_timing

\* @name Sensor Timing

\* The sensor timing for an AOSong AM2315

\*/

/\*\*@{\*/

/// @brief Sensor::\_warmUpTime\_ms; the AM2315 warms up in 500ms (estimated).

#define AM2315\_WARM\_UP\_TIME\_MS 500

/// @brief Sensor::\_stabilizationTime\_ms; the AM2315 is stable after 500ms

/// (estimated).

#define AM2315\_STABILIZATION\_TIME\_MS 500

/// @brief Sensor::\_measurementTime\_ms; the AM2315 takes 2000ms (2s) to complete

/// a measurement.

#define AM2315\_MEASUREMENT\_TIME\_MS 2000

/\*\*@}\*/

/\*\*

\* @anchor sensor\_am2315\_humidity

\* @name Humidity

\* The humidity variable from an AOSong AM2315

\* - Range is 0 to 100% RH

\* - Accuracy is ± 2 % RH at 25°C

\*

\* {{ @ref AOSongAM2315\_Humidity::AOSongAM2315\_Humidity }}

\*/

/\*\*@{\*/

/// @brief Decimals places in string representation; humidity should have 1 (0.1

/// % RH for the 16 bit sensor).

#define AM2315\_HUMIDITY\_RESOLUTION 1

/// @brief Sensor variable number; humidity is stored in sensorValues[0].

#define AM2315\_HUMIDITY\_VAR\_NUM 0

/// @brief Variable name in

/// [ODM2 controlled vocabulary](http://vocabulary.odm2.org/variablename/);

/// "relativeHumidity"

#define AM2315\_HUMIDITY\_VAR\_NAME "relativeHumidity"

/// @brief Variable unit name in

/// [ODM2 controlled vocabulary](http://vocabulary.odm2.org/units/); "percent"

/// (percent relative humidity)

#define AM2315\_HUMIDITY\_UNIT\_NAME "percent"

/// @brief Default variable short code; "AM2315Humidity"

#define AM2315\_HUMIDITY\_DEFAULT\_CODE "AM2315Humidity"

/\*\*@}\*/

/\*\*

\* @anchor sensor\_am2315\_temperature

\* @name Temperature

\* The temperature variable from an AOSong AM2315

\* - Range is -40°C to +125°C

\* - Accuracy is ±0.1°C

\*

\* {{ @ref AOSongAM2315\_Temp::AOSongAM2315\_Temp }}

\*/

/\*\*@{\*/

/// @brief Decimals places in string representation; temperature should have 1.

/// (0.1°C for the 16 bit sensor)

#define AM2315\_TEMP\_RESOLUTION 1

/// @brief Sensor variable number; temperature is stored in sensorValues[1].

#define AM2315\_TEMP\_VAR\_NUM 1

/// @brief Variable name in

/// [ODM2 controlled vocabulary](http://vocabulary.odm2.org/variablename/);

/// "temperature"

#define AM2315\_TEMP\_VAR\_NAME "temperature"

/// @brief Variable unit name in

/// [ODM2 controlled vocabulary](http://vocabulary.odm2.org/units/);

/// "degreeCelsius" (°C)

#define AM2315\_TEMP\_UNIT\_NAME "degreeCelsius"

/// @brief Default variable short code; "AM2315Temp"

#define AM2315\_TEMP\_DEFAULT\_CODE "AM2315Temp"

/\*\*@}\*/

/\* clang-format off \*/

/\*\*

\* @brief The Sensor sub-class for the [AOSong AM2315](@ref sensor\_am2315).

\*/

/\* clang-format on \*/

class AOSongAM2315 : public Sensor {

public:

/\*\*

\* @brief Construct a new AOSongAM2315 object using a secondary \*hardware\*

\* I2C instance.

\*

\* This is only applicable to SAMD boards that are able to have multiple

\* hardware I2C ports in use via SERCOMs.

\*

\* @note It is only possible to connect \*one\* AM2315 at a time on a single

\* I2C bus.

\*

\* @param theI2C A TwoWire instance for I2C communication. Due to the

\* limitations of the Arduino core, only a hardware I2C instance can be

\* used. For an AVR board, there is only one I2C instance possible and this

\* form of the constructor should not be used. For a SAMD board, this can

\* be used if a secondary I2C port is created on one of the extra SERCOMs.

\* @param powerPin The pin on the mcu controlling power to the AOSong

\* AM2315. Use -1 if it is continuously powered.

\* - The AM2315 requires a 3.3 - 5.5V power source

\* @param measurementsToAverage The number of measurements to take and

\* average before giving a "final" result from the sensor; optional with a

\* default value of 1.

\*/

bool begin();

AOSongAM2315(TwoWire\* theI2C, int8\_t powerPin,

uint8\_t measurementsToAverage = 1);

/\*\*

\* @brief Construct a new AOSongAM2315 object using the primary hardware I2C

\* instance.

\*

\* Because this is I2C and has only 1 possible address (0xB8), we only need

\* the power pin.

\*

\* @note It is only possible to connect \*one\* AM2315 at a time on a single

\* I2C bus.

\*

\* @param powerPin The pin on the mcu controlling power to the AOSong

\* AM2315. Use -1 if it is continuously powered.

\* - The AM2315 requires a 3.3 - 5.5V power source

\* @param measurementsToAverage The number of measurements to take and

\* average before giving a "final" result from the sensor; optional with a

\* default value of 1.

\*/

explicit AOSongAM2315(int8\_t powerPin, uint8\_t measurementsToAverage = 1);

/\*\*

\* @brief Destroy the AOSongAM2315 object - no action needed.

\*/

~AOSongAM2315();

/\*\*

\* @brief Report the I2C address of the AM2315 - which is always 0xB8.

\*

\* @return \*\*String\*\* Text describing how the sensor is attached to the mcu.

\*/

String getSensorLocation(void) override;

/\*\*

\* @brief Do any one-time preparations needed before the sensor will be able

\* to take readings.

\*

\* This sets the #\_powerPin mode, begins the Wire library (sets pin levels

\* and modes for I2C), and updates the #\_sensorStatus. No sensor power is

\* required.

\*

\* @return \*\*bool\*\* True if the setup was successful. For the AOSong AM2315

\* the result will always be true.

\*/

bool setup(void) override {

// Add your setup code here

// ...

return true;

}

/\*\*

\* @copydoc Sensor::addSingleMeasurementResult()

\*/

bool addSingleMeasurementResult(void) override;

private:

/\*\*

\* @brief An internal reference to the hardware Wire instance.

\*/

TwoWire\* \_i2c;

AM2315C\* am2315ptr; // create a sensor object

};

/\* clang-format off \*/

/\*\*

\* @brief The Variable sub-class used for the

\* [relative humidity output](@ref sensor\_am2315\_humidity) from an

\* [AOSong AM2315](@ref sensor\_am2315).

\*/

/\* clang-format on \*/

class AOSongAM2315\_Humidity : public Variable {

public:

/\*\*

\* @brief Construct a new AOSongAM2315\_Humidity object.

\*

\* @param parentSense The parent AOSongAM2315 providing the result

\* values.

\* @param uuid A universally unique identifier (UUID or GUID) for the

\* variable; optional with the default value of an empty string.

\* @param varCode A short code to help identify the variable in files;

\* optional with a default value of "AM2315Humidity".

\*/

explicit AOSongAM2315\_Humidity(

AOSongAM2315\* parentSense, const char\* uuid = "",

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

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

(uint8\_t)AM2315\_HUMIDITY\_RESOLUTION,

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

uuid) {}

/\*\*

\* @brief Construct a new AOSongAM2315\_Humidity object.

\*

\* @note This must be tied with a parent AOSongAM2315 before it can be used.

\*/

AOSongAM2315\_Humidity()

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

(uint8\_t)AM2315\_HUMIDITY\_RESOLUTION,

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

AM2315\_HUMIDITY\_DEFAULT\_CODE) {}

/\*\*

\* @brief Destroy the AOSongAM2315\_Humidity object - no action needed.

\*/

~AOSongAM2315\_Humidity() {}

};

/\* clang-format off \*/

/\*\*

\* @brief The Variable sub-class used for the

\* [temperature output](@ref sensor\_am2315\_temperature) from an

\* [AOSong AM2315](@ref sensor\_am2315).

\*/

/\* clang-format on \*/

class AOSongAM2315\_Temp : public Variable {

public:

/\*\*

\* @brief Construct a new AOSongAM2315\_Temp object.

\*

\* @param parentSense The parent AOSongAM2315 providing the result

\* values.

\* @param uuid A universally unique identifier (UUID or GUID) for the

\* variable; optional with the default value of an empty string.

\* @param varCode A short code to help identify the variable in files;

\* optional with a default value of "AM2315Temp".

\*/

explicit AOSongAM2315\_Temp(AOSongAM2315\* parentSense, const char\* uuid = "",

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

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

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

AM2315\_TEMP\_UNIT\_NAME, varCode, uuid) {}

/\*\*

\* @brief Construct a new AOSongAM2315\_Temp object.

\*

\* @note This must be tied with a parent AOSongAM2315 before it can be used.

\*/

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) {}

/\*\*

\* @brief Destroy the AOSongAM2315\_Temp object - no action needed.

\*/

~AOSongAM2315\_Temp() {}

};

/\*\*@}\*/

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