

My Project

Generated by Doxygen 1.16.0

1 Smartgreenhouse	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 SharedState Struct Reference	7
5 File Documentation	9
5.1 climateControl.h File Reference	9
5.1.1 Detailed Description	9
5.1.2 Function Documentation	10
5.1.2.1 climateControlBegin()	10
5.1.2.2 climateControlUpdate()	10
5.2 climateControl.h	10
5.3 dht11Sensor.cpp File Reference	11
5.3.1 Detailed Description	11
5.3.2 Function Documentation	11
5.3.2.1 dht11Begin()	11
5.3.2.2 dht11Read()	12
5.4 dht11Sensor.h File Reference	12
5.4.1 Detailed Description	12
5.4.2 Function Documentation	12
5.4.2.1 dht11Begin()	12
5.4.2.2 dht11Read()	13
5.5 dht11Sensor.h	13
5.6 heater.cpp File Reference	13
5.6.1 Detailed Description	14
5.6.2 Function Documentation	14
5.6.2.1 heaterBegin()	14
5.6.2.2 heaterSet()	14
5.7 heater.h File Reference	14
5.7.1 Detailed Description	14
5.7.2 Function Documentation	15
5.7.2.1 heaterBegin()	15
5.7.2.2 heaterSet()	15
5.8 heater.h	15
5.9 LightSensor.cpp File Reference	15
5.9.1 Detailed Description	16
5.9.2 Function Documentation	16
5.9.2.1 lightInit()	16

5.9.2.2 lightUpdate()	16
5.10 Lightsensor.h File Reference	16
5.10.1 Detailed Description	17
5.10.2 Function Documentation	17
5.10.2.1 lightInit()	17
5.10.2.2 lightUpdate()	17
5.11 Lightsensor.h	17
5.12 mister.cpp File Reference	17
5.12.1 Detailed Description	18
5.12.2 Function Documentation	18
5.12.2.1 misterInit()	18
5.12.2.2 misterState()	18
5.13 mister.h File Reference	18
5.13.1 Detailed Description	19
5.13.2 Function Documentation	19
5.13.2.1 misterInit()	19
5.13.2.2 misterState()	19
5.14 mister.h	19
5.15 sharedState.h File Reference	20
5.15.1 Detailed Description	20
5.16 sharedState.h	20
Index	21

Chapter 1

Smartgreenhouse

Regulate a greenhouse This is just for show

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

SharedState	7
-----------------------------	-------	---

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

climateControl.h	
Climate controller: applies control logic (heater + mister) from SharedState sensor values/targets	9
dht11Sensor.cpp	
DHT11 implementation	11
dht11Sensor.h	
DHT11 sensor module: init + read into SharedState	12
heater.cpp	
Heater output implementation	13
heater.h	
Heater output module (LED / relay abstraction)	14
LightSensor.cpp	
Light sensor (LDR) logic + lamp + daily light counter	15
Lightsensor.h	
Light sensor module (LDR) + lamp control + daily light counter	16
mister.cpp	
Mister output implementation	17
mister.h	
Mister output module (on/off)	18
sharedState.h	
Shared state container for sensors + actuators	20

Chapter 4

Class Documentation

4.1 SharedState Struct Reference

Public Attributes

- float **tempC** = NAN
Latest temperature in °C (NAN if invalid).
- float **humidityPct** = NAN
Latest humidity in % (NAN if invalid).
- bool **hasDht** = false
True if last DHT read succeeded.
- float **lightHoursToday** = NAN
Accumulated hours of effective light today.
- bool **lampOn** = false
True when lamp LED output is ON.
- bool **heaterOn** = false
True when heater output is ON.
- bool **misterOn** = false
True when mister output is ON.
- float **targetTempC** = 27.0f
Desired temperature (°C).
- float **targetHumidityPct** = 65.0f
Desired humidity (%).

The documentation for this struct was generated from the following file:

- [sharedState.h](#)

Chapter 5

File Documentation

5.1 climateControl.h File Reference

Climate controller: applies control logic (heater + mister) from `SharedState` sensor values/targets.

```
#include "sharedState.h"
```

Functions

- void `climateControlBegin()`
Initialize controller timers/state.
- void `climateControlUpdate(SharedState &state)`
Update climate control (heater + mister) based on sensor readings and targets.

5.1.1 Detailed Description

Climate controller: applies control logic (heater + mister) from `SharedState` sensor values/targets.

This module implements:

- Temperature control using hysteresis (prevents rapid toggling).
- Humidity control using hysteresis + safety timers (max ON time + minimum OFF backoff).

Typical use:

- call `climateControlBegin()` once in `setup()`
- call `climateControlUpdate(state)` repeatedly in `loop()`

5.1.2 Function Documentation

5.1.2.1 climateControlBegin()

```
void climateControlBegin ()
```

Initialize controller timers/state.

Call once in setup(). This does not change any outputs by itself.

5.1.2.2 climateControlUpdate()

```
void climateControlUpdate (
    SharedState & state)
```

Update climate control (heater + mister) based on sensor readings and targets.

Inputs (read from `state`):

- `state.hasDht`, `state.tempC`, `state.humidityPct`
- `state.targetTempC`, `state.targetHumidityPct`
- `state.heaterOn`, `state.misterOn` (used for hysteresis decisions)

Outputs (written to `state` and hardware):

- Heater output via `heaterSet(state, on)` (also updates `state.heaterOn`)
- Mister output via `misterState(on)` and `state.misterOn`

Fail-safe:

- If `state.hasDht` is false, heater and mister are turned OFF.

Parameters

<code>state</code>	Shared state containing latest sensor values, setpoints, and actuator states.
--------------------	---

5.2 climateControl.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include "sharedState.h"
00003
00016
00022 void climateControlBegin();
00023
00041 void climateControlUpdate(SharedState &state);
```

5.3 dht11Sensor.cpp File Reference

DHT11 implementation.

```
#include "dht11Sensor.h"
#include <DHT.h>
```

Macros

- #define **DHTPIN** D7
- #define **DHTTYPE** DHT11

Functions

- void **dht11Begin** ()
Initialize the DHT11 hardware/library.
- void **dht11Read** (**SharedState** &state)
Read temperature + humidity and write results into SharedState.

5.3.1 Detailed Description

DHT11 implementation.

5.3.2 Function Documentation

5.3.2.1 dht11Begin()

```
void dht11Begin ()
```

Initialize the DHT11 hardware/library.

Call once in setup().

5.3.2.2 dht11Read()

```
void dht11Read (
    SharedState & state)
```

Read temperature + humidity and write results into [SharedState](#).

On success:

- state.tempC, state.humidityPct updated
- state.hasDht = true

On failure:

- tempC/humidityPct set to NAN
- state.hasDht = false

Parameters

state	SharedState to update.
-------	--

5.4 dht11Sensor.h File Reference

DHT11 sensor module: init + read into [SharedState](#).

```
#include "sharedState.h"
```

Functions

- void [dht11Begin \(\)](#)
Initialize the DHT11 hardware/library.
- void [dht11Read \(SharedState &state\)](#)
Read temperature + humidity and write results into [SharedState](#).

5.4.1 Detailed Description

DHT11 sensor module: init + read into [SharedState](#).

5.4.2 Function Documentation

5.4.2.1 dht11Begin()

```
void dht11Begin ()
```

Initialize the DHT11 hardware/library.

Call once in setup().

5.4.2.2 dht11Read()

```
void dht11Read (
    SharedState & state)
```

Read temperature + humidity and write results into `SharedState`.

On success:

- `state.tempC`, `state.humidityPct` updated
- `state.hasDht` = true

On failure:

- `tempC/humidityPct` set to NAN
- `state.hasDht` = false

Parameters

<code>state</code>	<code>SharedState</code> to update.
--------------------	-------------------------------------

5.5 dht11Sensor.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include "sharedState.h"
00003
00008
00014 void dht11Begin();
00015
00029 void dht11Read(SharedState &state);
```

5.6 heater.cpp File Reference

Heater output implementation.

```
#include <Arduino.h>
#include "heater.h"
```

Functions

- `void heaterBegin ()`
Initialize heater output pin(s).
- `void heaterSet (SharedState &state, bool on)`
Turn heater output ON/OFF and update `SharedState`.

5.6.1 Detailed Description

Heater output implementation.

5.6.2 Function Documentation

5.6.2.1 heaterBegin()

```
void heaterBegin ()
```

Initialize heater output pin(s).

Call once in setup().

5.6.2.2 heaterSet()

```
void heaterSet (
    SharedState & state,
    bool on)
```

Turn heater output ON/OFF and update [SharedState](#).

Parameters

<i>state</i>	SharedState to update (writes state.heaterOn only).
<i>on</i>	True to turn heater on, false to turn it off.

5.7 heater.h File Reference

Heater output module (LED / relay abstraction).

```
#include "sharedState.h"
```

Functions

- void [heaterBegin \(\)](#)
Initialize heater output pin(s).
- void [heaterSet \(SharedState &state, bool on\)](#)
Turn heater output ON/OFF and update [SharedState](#).

5.7.1 Detailed Description

Heater output module (LED / relay abstraction).

5.7.2 Function Documentation

5.7.2.1 heaterBegin()

```
void heaterBegin ()
```

Initialize heater output pin(s).

Call once in setup().

5.7.2.2 heaterSet()

```
void heaterSet (
    SharedState & state,
    bool on)
```

Turn heater output ON/OFF and update `SharedState`.

Parameters

<code>state</code>	<code>SharedState</code> to update (writes <code>state.heaterOn</code> only).
<code>on</code>	True to turn heater on, false to turn it off.

5.8 heater.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include "sharedState.h"
00003
00008
00014 void heaterBegin();
00015
00022 void heaterSet(SharedState &state, bool on);
```

5.9 LightSensor.cpp File Reference

Light sensor (LDR) logic + lamp + daily light counter.

```
#include "LightSensor.h"
```

Functions

- `void lightInit ()`
Initialize light sensor module (pins + timing).
- `void lightUpdate (SharedState &state)`
Update light logic and write results to `SharedState`.

5.9.1 Detailed Description

Light sensor (LDR) logic + lamp + daily light counter.

5.9.2 Function Documentation

5.9.2.1 `lightInit()`

```
void lightInit ()
```

Initialize light sensor module (pins + timing).

Call once in `setup()`.

5.9.2.2 `lightUpdate()`

```
void lightUpdate (
    SharedState & state)
```

Update light logic and write results to `SharedState`.

Writes:

- `state.lightHoursToday`
- `state.lampOn`

Call repeatedly in `loop()`.

Parameters

<code>state</code>	<code>SharedState</code> to update.
--------------------	-------------------------------------

5.10 Lightsensor.h File Reference

Light sensor module (LDR) + lamp control + daily light counter.

```
#include <Arduino.h>
#include "sharedState.h"
```

Functions

- `void lightInit ()`
Initialize light sensor module (pins + timing).
- `void lightUpdate (SharedState &state)`
Update light logic and write results to `SharedState`.

5.10.1 Detailed Description

Light sensor module (LDR) + lamp control + daily light counter.

5.10.2 Function Documentation

5.10.2.1 lightInit()

```
void lightInit ()
```

Initialize light sensor module (pins + timing).

Call once in setup().

5.10.2.2 lightUpdate()

```
void lightUpdate (
    SharedState & state)
```

Update light logic and write results to [SharedState](#).

Writes:

- state.lightHoursToday
- state.lampOn

Call repeatedly in loop().

Parameters

<code>state</code>	SharedState to update.
--------------------	--

5.11 Lightsensor.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include <Arduino.h>
00003 #include "sharedState.h"
00004
00009
00015 void lightInit ();
00016
00028 void lightUpdate(SharedState &state);
```

5.12 mister.cpp File Reference

Mister output implementation.

```
#include "mister.h"
```

Macros

- `#define MISTERPIN D1`

Functions

- `void misterInit ()`
Initialize the mister output pin.
- `void misterState (bool on)`
Turn mister output ON/OFF.

5.12.1 Detailed Description

Mister output implementation.

5.12.2 Function Documentation

5.12.2.1 misterInit()

```
void misterInit ()
```

Initialize the mister output pin.

Intializing the pin to be used by the mister. Change pin in the define MISTERPIN.

5.12.2.2 misterState()

```
void misterState (
    bool on)
```

Turn mister output ON/OFF.

Turns the mister on or off. Handles unknown states by turning the mister off.

Parameters

<code>on</code>	can be true or false, to turn on or off the mister. It is inverted.
-----------------	---

5.13 mister.h File Reference

Mister output module (on/off).

```
#include <Arduino.h>
```

Functions

- void **misterInit ()**
Initialize the mister output pin.
- void **misterState (bool command)**
Turn mister output ON/OFF.

5.13.1 Detailed Description

Mister output module (on/off).

5.13.2 Function Documentation

5.13.2.1 misterInit()

```
void misterInit ()
```

Initialize the mister output pin.

Call once in setup().

Intializing the pin to be used by the mister. Change pin in the define MISTERPIN.

5.13.2.2 misterState()

```
void misterState (
    bool on)
```

Turn mister output ON/OFF.

Parameters

<i>command</i>	True = ON, False = OFF.
----------------	-------------------------

Turns the mister on or off. Handles unknown states by turning the mister off.

Parameters

<i>on</i>	can be true or false, to turn on or off the mister. It is inverted.
-----------	---

5.14 mister.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include <Arduino.h>
00003
00008
00014 void misterInit();
00015
00021 void misterState(bool command);
```

5.15 sharedState.h File Reference

Shared state container for sensors + actuators.

```
#include <Arduino.h>
```

Classes

- struct [SharedState](#)

5.15.1 Detailed Description

Shared state container for sensors + actuators.

Modules write/read to/from this struct to coordinate behavior.

5.16 sharedState.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include <Arduino.h>
00003
00010 struct SharedState {
00011     // --- DHT11: Temperature and humidity ---
00012     float tempC = NAN;
00013     float humidityPct = NAN;
00014     bool hasDht = false;
00015
00016     // --- Light module ---
00017     float lightHoursToday = NAN;
00018     bool lampOn = false;
00019
00020     // --- Actuator states ---
00021     bool heaterOn = false;
00022     bool misterOn = false;
00023
00024     // --- Targets (set by main / controller) ---
00025     float targetTempC = 27.0f;
00026     float targetHumidityPct = 65.0f;
00027 };
```

Index

climateControl.h, 9
 climateControlBegin, 10
 climateControlUpdate, 10
climateControlBegin
 climateControl.h, 10
climateControlUpdate
 climateControl.h, 10

dht11Begin
 dht11Sensor.cpp, 11
 dht11Sensor.h, 12
dht11Read
 dht11Sensor.cpp, 11
 dht11Sensor.h, 12
dht11Sensor.cpp, 11
 dht11Begin, 11
 dht11Read, 11
dht11Sensor.h, 12
 dht11Begin, 12
 dht11Read, 12

heater.cpp, 13
 heaterBegin, 14
 heaterSet, 14
heater.h, 14
 heaterBegin, 15
 heaterSet, 15
heaterBegin
 heater.cpp, 14
 heater.h, 15
heaterSet
 heater.cpp, 14
 heater.h, 15

lightInit
 LightSensor.cpp, 16
 Lightsensor.h, 17
LightSensor.cpp, 15
 lightInit, 16
 lightUpdate, 16
Lightsensor.h, 16
 lightInit, 17
 lightUpdate, 17
lightUpdate
 LightSensor.cpp, 16
 Lightsensor.h, 17

mister.cpp, 17
 misterInit, 18
 misterState, 18