NodeRedTime

1.0.0

Generated by Doxygen 1.8.18

1 Class Index

1.1 Class List

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

2 Class Documentation

2.1 NodeRedTime Class Reference

Class to obtain Unix epoch time values from a Node-Red server.

```
#include <NodeRedTime.h>
```

Public Member Functions

- **NodeRedTime** (const char *url, const unsigned int recall_s=3600, const time_t minEpoch_s=1262304000)

  NodeRedTime *(*p. *1) constructor.*
- bool **serverTime** (time_t *epoch) __attribute__((nonnull))

  Obtain Unix epoch time value from Node-Red.
- bool **syntheticTime** (time_t *epoch) __attribute__((nonnull))

  Synthesize updated epoch time value if possible.

Protected Attributes

- String **_url**

  url of Node-Red server. eg `http://host.domain.com:1880:/time/` *Initialized by constructor.*
- double **_recall_ms**

  the maximum time in milliseconds that ***syntheticTime()*** *(p. 4) can calculate updated epoch values by adding elapsed time derived from millis() to _epochLastSync_ms. Once this period has expired, the next call to* ***syntheticTime()*** *(p. 4) will force a call to* ***serverTime()*** *(p. 3). The value is constrained to the range 1 minute to 4 hours, and defaults to 1 hour. Initialised by constructor which converts seconds argument to milliseconds.*
- double **_minEpoch_ms**

  the earliest epoch value which can be considered valid. Initialized by constructor which converts seconds argument to milliseconds. Default value corresponds with 2010-01-01T00:00:00Z.
- double **_epochLastSync_ms** = 0.0

  epoch time in milliseconds last obtained from Node-Red. Initialized to zero which is considered a sentinel value meaning EITHER ***serverTime()*** *(p. 3) has never been called OR* ***serverTime()*** *(p. 3) has been called at least once but, thus far, has not been able to obtain a milliseconds value greater than _minEpoch_ms. While _epochLastSync_ms has a zero value,* ***syntheticTime()*** *(p. 4) will always call* ***serverTime()*** *(p. 3). Will only be updated if a new valid seconds value can be obtained from Node-Red. Updated when* ***serverTime()*** *(p. 3) is called. Used by* ***syntheticTime()*** *(p. 4).*
- double **_uptimeLastSync_ms** = 0.0

  the millis() value corresponding ***approximately*** *with the moment when _epochLastSync_ms was determined on the Node-Red server. Set by* ***serverTime()*** *(p. 3) but will only be non-zero if _epochLastSync_ms is also non-zero. Used by* ***syntheticTime()*** *(p. 4). Initialized to zero (implying millis() at system boot) but zero can potentially be a valid value when millis() wraps (every 49.7 days).*

2.1.1 Detailed Description

Remarks

Instance variables are mostly declared **double** but are only used to hold integer milliseconds values. The code *could* have made the integer nature of the milliseconds values explicit by using **uint64_t**. Unfortunately, **uint64_t** variables don't yet have full support throughout the Arduino API and tend to be slightly opaque when it comes to using them in Serial.print statements during debugging. You wind up having to cast to **double** anyway. On balance, declaring **double** seemed the better choice.

2.1.2 Constructor & Destructor Documentation

2.1.2.1 NodeRedTime() `NodeRedTime::NodeRedTime (`

```
const char * url,
const unsigned int recall_s = 3600,
const time_t minEpoch_s = 1262304000 )
```

Sample code:

```
#include <NodeRedTime.h>
NodeRedTime nodeRedTime("http://host.domain.com:1880/time/");
```

Parameters

in	*url*	well-formed Node-Red URL (eg "http://host.domain.com:1880:/time/").

Parameters

in	*recall_s*	the number of seconds between enforced calls to **serverTime()** (p. 3) within **syntheticTime()** (p. 4). Defaults to 1 hour. Any value passed is clipped to the range 60..14400 (one minute to 4 hours).
in	*minEpoch_s*	earliest seconds value which can be considered a valid date+time. Default value = 1262304000 (2010-01-01T00:00:00Z). Any value to the left of this on the number line will be considered invalid.

Warning

No constraints are applied to the minEpoch_s parameter. In theory, any **non-zero value** will work but testing of this assumption is up to the user.

Returns

nothing.

2.1.3 Member Function Documentation

2.1.3.1 serverTime() `bool NodeRedTime::serverTime ( time_t * epoch )`

Posts an http request to a Node-Red server. Expects a reply containing a string representation of a positive integer of the number of whole milliseconds that have elepsed since the Unix epoch on 1970-01-01T00:00:00.000Z.

Sample code:

```
time_t epochTime;
if (nodeRedTime.serverTime(&epochTime)) {
    tm timeinfo;
    if (localtime_r(&epochTime, &timeinfo)) {
        Serial.printf("time: %s",asctime(&timeinfo));
    }
}
```

Precondition

url set by constructor must be valid. Assumes Node-Red responds to URL with Unix Epoch milliseconds value.

Parameters

out	*epoch*	pointer to time_t, must not be nil.

Returns

true if a valid time value was able to be obtained from Node-Red. Otherwise **false**.

Remarks

time_t is declared "typedef uint32_t time_t" (an unsigned 32-bit quantity). The Node-Red response is interpreted by HTTPClient::getString().toDouble() which parses like this:

- Skips leading spaces.
- Handles leading "+" or "-" correctly (returns signed quantity).
- Stops parsing on the first non-numeric character or end-of-string.
- Understands scientific notation (eg "1E3" and "1E-3").
- Returns 0 if it cannot recognise a number.

The unlikely possibility of a negative number, combined with the slightly more likely possibility of a zero from either a server non-response or a failed parse is the reason for considering a seconds value to be invalid if it is less than _minEpoch_ms.

2.1.3.2 syntheticTime() `bool NodeRedTime::syntheticTime ( time_t * epoch )`

Calculates an updated epoch value by using millis() to determine the number of whole seconds that have elapsed since the last successful call to **serverTime()** (p. 3). Passes the request to **serverTime()** (p. 3) if:

- _epochLastSync_ms is zero (ie **serverTime()** (p. 3) never called successfully); or
- _recall_ms have elapsed since the last successful call to **serverTime()** (p. 3)

Sample code:

```
time_t epochTime;
if (nodeRedTime.syntheticTime(&epochTime)) {
    tm timeinfo;
    if (localtime_r(&epochTime, &timeinfo)) {
        Serial.printf("time: %s",asctime(&timeinfo));
    }
}
```

Parameters

out	*epoch*	pointer to time_t, must not be nil.

Returns

true if a revised time value was able to be synthesized based on a prior successful call to Node-Red **or** a successful call can be made to Node-Red. Otherwise **false**.