

mLink Library Reference Guide for

mLink NTC Temperature Sensor  
Module (HCMODU0186)

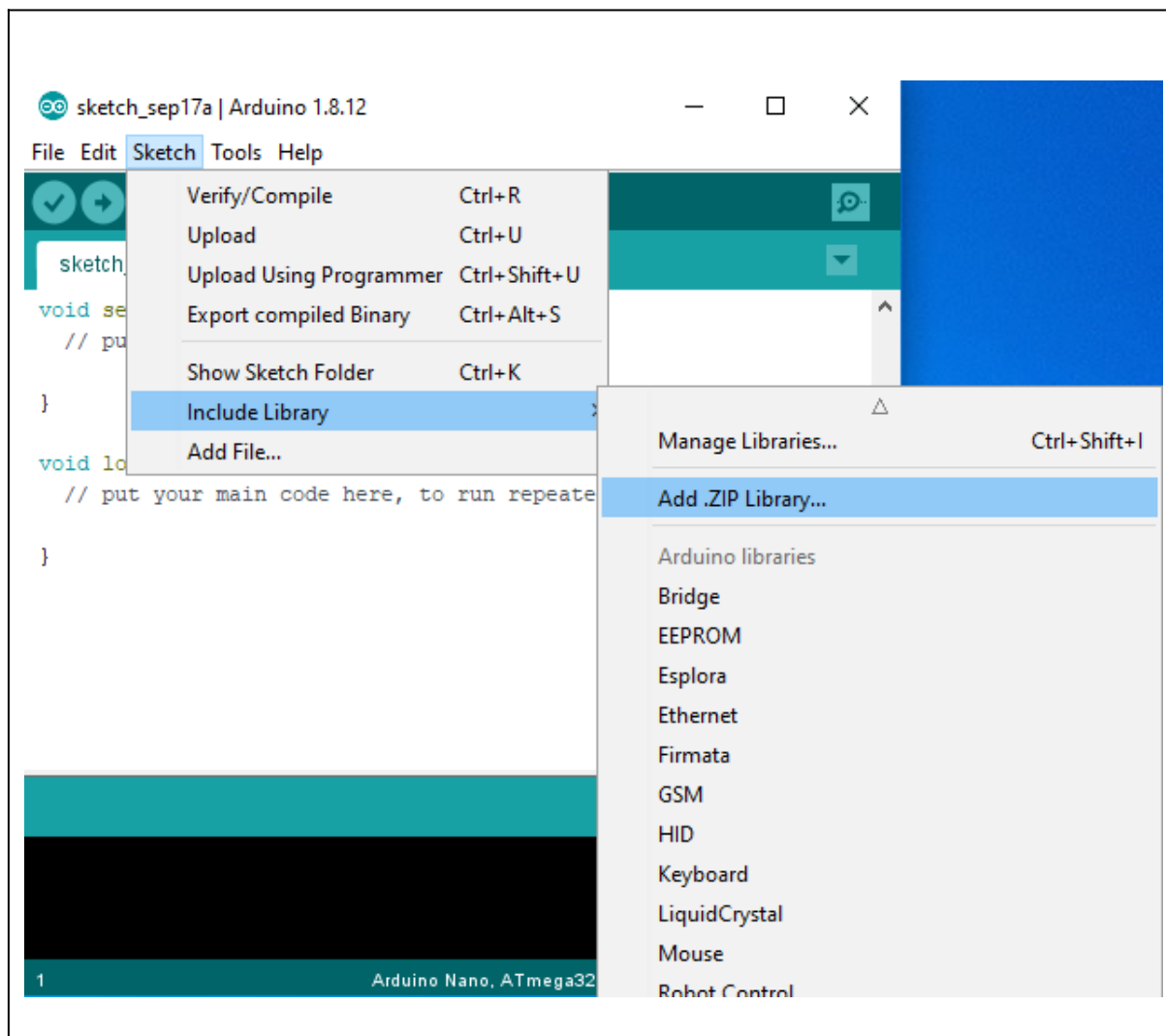
# Installing the mLink library

Adding the mLink library to your Arduino IDE can be done in the same way as any other Arduino library:

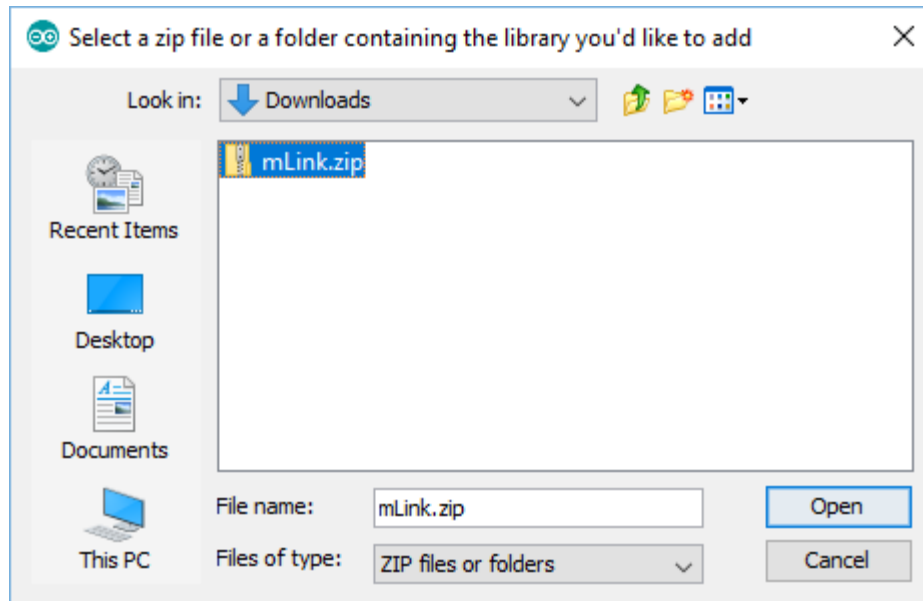
First download the mLink.zip file from the software section of our support forum here:

<https://hobbycomponents.com/mLink>

Once downloaded, open up your Arduino IDE and go to Sketch->Include Library->Add .ZIP Library.



In the file selection dialogue window that opens, navigate to wherever you downloaded the mLink .zip file and select it, then click the 'Open' button.



# Including the mLink library in your sketch

Adding the mLink library to your sketch consists of 3 steps; Firstly include the mLink header file (mLink.h) at the top of your sketch, create an instance of the library, then finally initialise the library inside the startup() function:

```
// Step 1: Include the mLink library
#include "mLink.h"

//Step 2: Create an instance of the library
mLink mLink;

void setup()
{
  // Step 3: Initialise the library
  mLink.init();
}

void loop()
{
}
```

## Quick library reference table

COMMAND		PARAMETERS	RETURNS
<code>init()</code>	Initialises the mLink library	None	n/a
<code>readBit(add, reg, bit)</code>	Reads the state of a bit from one of the mLink registers	<i>add</i> = byte value containing I2C address of mLink module <i>reg</i> = byte value containing register index <i>bit</i> = byte value containing the bit number to read (0 to 7)	<i>boolean</i> value containing the state of the bit
<code>read(add, reg)</code>	Reads the contents of one of the mLink registers	<i>add</i> = byte value containing I2C address of mLink module <i>reg</i> = byte value containing register index	<i>byte</i> value containing the state of the register
<code>readInt(add, reg)</code>	Reads the contents of 2 consecutive registers and returns the result as an unsigned integer	<i>add</i> = byte value containing I2C address of mLink module <i>reg</i> = byte value containing register index of the first register	<i>unsigned integer</i> containing the values of the two registers
<code>writeBit(add, reg, bit, state)</code>	Writes to a bit in one of the mLink registers	<i>add</i> = byte value containing I2C address of mLink module <i>reg</i> = byte value containing register index <i>bit</i> = byte value containing the bit number to write to (0 to 7) <i>state</i> = boolean value to set the bit to	n/a
<code>write(add, reg, data)</code>	Writes data to one of the mLink registers	<i>add</i> = byte value containing I2C address of mLink module <i>reg</i> = byte value containing register index <i>data</i> = byte value containing the data to write to the register	n/a
<code>sleep(add);</code>	Puts the module into a low power sleep mode.	<i>add</i> = byte value containing I2C address of mLink module	n/a
<code>busy(add);</code>	Checks the state of the busy bit in the status register	<i>add</i> = byte value containing I2C address of mLink module	Boolean value: 0 = ready 1 = busy
<code>NTC_Temp(add);</code>	Library macro that reads the temperature registers and returns the result in °C	<i>add</i> = byte value containing I2C address of mLink module	float value containing the temperature in °C to 1dp

# Library Commands

## mLink.init()

### Description

Initialises the mLink library

Add to the setup() section of your sketch to initialise the mLink library

### Syntax

```
mLink.init()
```

### Parameters

None

### Returns

Nothing

### Example Code

```
void setup()
{
  mLink.init();
}

void loop()
{
}
```

## mLink.readBit(add, reg, bit)

### Description

Reads the state of a bit from one of the mLink modules 8 bit registers and returns the result as a boolean value.

### Parameters

*add*: byte value containing I2C address of mLink module. Alternatively, if the mLink module is set to its default I2C address (0x54) you can use the predefined value:

NTC\_I2C\_ADD

*reg*: byte value containing the register number to read. You can either specify the register number (see register table) or you can use one of the following predefined values:

MLINK\_STATUS\_REG

*bit*: byte value containing the bit number within the specified register to read. Valid values are 0 to 7.

### Returns

A boolean value representing the state of the bit.

### Example Code

Reads the state of bit 0 (COM error bit) from the status register

```
boolean result = mLink.readBit(NTC_I2C_ADD, MLINK_STATUS_REG, 0);
```

## mLink.read(*add*, *reg*)

### Description

Reads the state of one of the mLink modules 8 bit registers and returns the result as a byte.

### Parameters

*add*: byte value containing I2C address of mLink module. Alternatively, if the mLink module is set to its default I2C address (0x54) you can use the predefined value:

NTC\_I2C\_ADD

*reg*: byte value containing the register number to read. You can either specify the register number (see register table) or you can use one of the following predefined values:

MLINK\_STATUS\_REG  
MLINK\_ADD\_REG  
MLINK\_MOD\_TYPE\_REG  
MLINK\_MOD\_SUBTYPE\_REG  
MLINK\_SW\_VER\_REG  
MLINK\_NTC\_TEMP\_H\_REG  
MLINK\_NTC\_TEMP\_L\_REG

### Returns

A byte value representing the state of the register.

### Example Code

Reads the contents of the software version register (register 4)

```
byte result = mLink.read(NTC_I2C_ADD, MLINK_SW_VER_REG);
```



## mLink.readInt(*add*, *reg*)

### Description

Reads the state of two consecutive 8 bit registers and returns the result as an unsigned int.

### Parameters

*add*: byte value containing I2C address of mLink module. Alternatively if the mLink module is set to its default I2C address (0x54) you can use the predefined value:

NTC\_I2C\_ADD

*reg*: byte value containing the first register number to read. You can either specify the register number (see register table) or you can use one of the following predefined values:

NTC\_READ\_TEMP

### Returns

An unsigned int containing both registers where the low byte is the first register and the high byte is the second register.

### Example Code

Reads the contents of the two temperature registers (register 11 & 12).

```
unsigned int result = mLink.readInt(NTC_I2C_ADD, NTC_READ_TEMP);
```

## mLink.write(*add*, *reg*, *data*)

### Description

Writes to one of the mLink modules 8 bit registers.

### Parameters

*add*: byte value containing I2C address of mLink module. Alternatively if the mLink module is set to its default I2C address (0x54) you can use the predefined value:

NTC\_I2C\_ADD

*reg*: byte value containing the register number to write to. You can either specify the register number (see register table) or you can use one of the following predefined values:

MLINK\_STATUS\_REG

MLINK\_ADD\_REG

MLINK\_SLEEP\_REG

*data*: byte value containing the data to write to the register

### Returns

None

### Example Code

Puts the module into low power sleep mode by writing a 1 to the MLINK\_SLEEP\_REG register.

```
mLink.write(NTC_I2C_ADD, MLINK_SLEEP_REG , 1);
```

`mLink.sleep(add);`

## Description

Puts the module into a low power sleep mode.

Sleep mode is automatically exited on the next register read or write.

## Parameters

*add*: byte value containing I2C address of mLink module. Alternatively if the mLink module is set to its default I2C address (0x54) you can use the predefined value:

`NTC_I2C_ADD`

## Returns

None

## Example Code

Puts the module into low power sleep mode.

```
mLink.sleep(NTC_I2C_ADD);
```

`mLink.busy(add);`

## Description

Checks the state of the busy bit in the status register. This bit will be set when exiting the sleep state and cleared once a new measurement is automatically made. Therefore after exiting sleep state this bit should be polled before attempting to read a new temperature measurement.

## Parameters

*add*: byte value containing I2C address of mLink module. Alternatively if the mLink module is set to its default I2C address (0x54) you can use the predefined value:

`NTC_I2C_ADD`

## Returns

None

## Example Code

Triggers a DHT22 measurement then uses the `mLink.busy()` function to wait until the measurement is complete.

```
mLink.sleep(I2C_ADD); // Go in to low power sleep mode
delay(10000);         // Wait for 10 seconds

while(mLink.busy(I2C_ADD)); // Wake module up and wait for busy to clear

float temp = mLink.DHT22_Temp(I2C_ADD); // Get the temperature in oC
```

`mLink.NTC_Temp(add);`

## Description

Library macro that reads the temperature registers and returns the result in °C.

## Parameters

*add*: byte value containing I2C address of mLink module. Alternatively if the mLink module is set to its default I2C address (0x54) you can use the predefined value:

`NTC_I2C_ADD`

## Returns

A float containing the last temperature measurement in °C to 1 decimal place.

## Example Code

Reads the current temperature.

```
float temp = mLink.NTC_Temp(I2C_ADD);    // Get the temperature in °C
```

# DISCLAIMER

The mLink range is a series of modules intended for the hobbyist and educational markets. Where every care has been taken to ensure the reliability and durability of this product it should not be used in safety or reliability critical applications.

This library and document is provided "as is". Hobby Components Ltd makes no warranties, whether express, implied or statutory, including, but not limited to, implied warranties of merchantability and fitness for a particular purpose, accuracy or lack of negligence. Hobby Components Ltd shall not, in any circumstances, be liable for any damages, including, but not limited to, special, incidental or consequential damages for any reason whatsoever.

# COPYRIGHT NOTICE

This manual, including content and artwork is copyright of Hobby Components Ltd and may not be reproduced without written permission. If you paid for or received a copy of this manual from a source other than Hobby Components Ltd, please contact us at [sales@hobbycomponents.com](mailto:sales@hobbycomponents.com)