mLink Library Reference Guide for

mLink Matrix 4x4 Keypad (HCMODU0188)

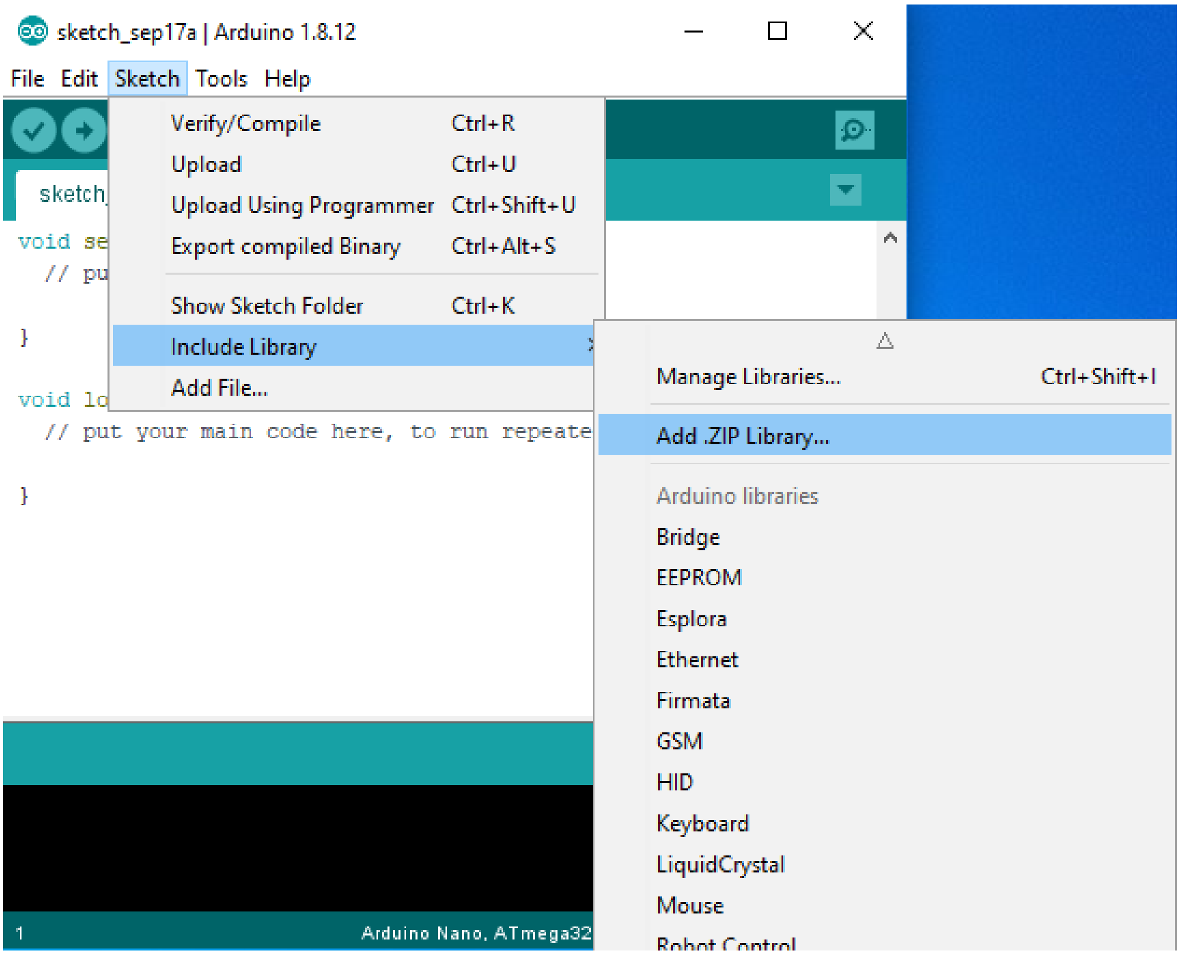
Installing the mLink library

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.



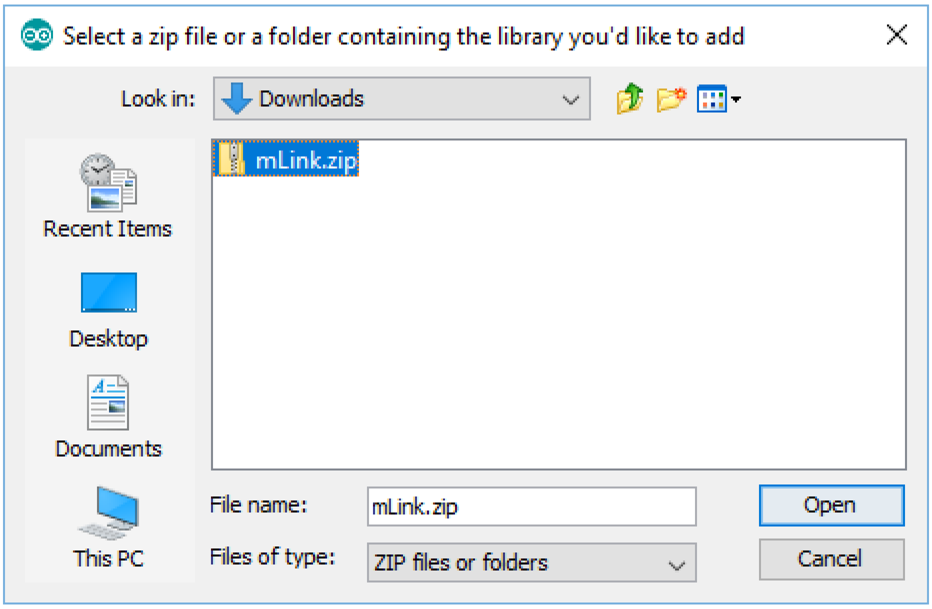
```
void se
  // pu

}

void lo
  // put your main code here, to run repeate

}
```

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.



Select a zip file or a folder containing the library you'd like to add

Look in: Downloads

mLink.zip

File name: mLink.zip

Files of type: ZIP files or folders

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
init()	Initialises the mLink library	None	n/a
readBit(*add*, *reg*, *bit*)	Reads the state of a bit from one of the mLink registers	*add* = *byte* value containing I2C address of mLink module *reg* = *byte* value containing register index *bit* = byte value containing the bit number to read (0 to 7)	*boolean* value containing the state of the bit
read(*add*, *reg*)	Reads the contents of one of the mLink registers	*add* = *byte* value containing I2C address of mLink module *reg* = *byte* value containing register index	*byte* value containing the state of the register
writeBit(*add*, *reg*, *bit*, *state*)	Writes to a bit in one of the mLink registers	*add* = *byte* value containing I2C address of mLink module *reg* = *byte* value containing register index *bit* = *byte* value containing the bit number to write to (0 to 7) *state* = boolean value to set the bit to	n/a
write(*add*, *reg*, *data*)	Writes data to one of the mLink registers	*add* = *byte* value containing I2C address of mLink module *reg* = *byte* value containing register index *data* = *byte* value containing the data to write to the register	n/a
Keypad_4x4_Key_Down(*add*);	Library macro that returns the current key pressed status.	*add* = *byte* value containing I2C address of mLink module	Boolean value: 0 = no key press 1 = a key is pressed

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 (0x55) you can use the predefined value:

KEYPAD_4X4_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
KEYPAD_4X4_KEY_STATE

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(KEYPAD_4X4_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 (0x55) you can use the predefined value:

KEYPAD_4X4_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
KEYPAD_4X4_KEY
KEYPAD_4X4_KEY_STATE
KEYPAD_4X4_DEBOUNCE

Returns

A byte value representing the state of the register.

Example Code

Reads the the key register to get the last key pressed (register 13)

```
byte key = mLink.read(KEYPAD_4X4_I2C_ADD, KEYPAD_4X4_KEY);
```

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 (0x55) you can use the predefined value:

KEYPAD_4X4_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
KEYPAD_4X4_DEBOUNCE

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

Returns

None

Example Code

Sets the amount of debounce used when reading the state of the keypad keys to 8 (default = 5).

```
mLink.write(KEYPAD_4X4_I2C_ADD, KEYPAD_4X4_DEBOUNCE, 8);
```

mLink.Keypad_4x4_Key_Down(*add*);

Description

Library macro that reads the last key pressed

Parameters

add: byte value containing I2C address of mLink module. Alternatively if the mLink module is set to its default I2C address (0x55) you can use the predefined value:

KEYPAD_4X4_I2C_ADD

Returns

A boolean value containing the key pressed state.
Low (0) = no key is currently pressed.
High (1) = one of the keypad keys is currently pressed

Example Code

Reads the current key pressed state.

```
boolean state = mLink.Keypad_4x4_Key_Down(KEYPAD_4X4_I2C_ADD);   // Get the key
                                                                 // pressed state
```

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