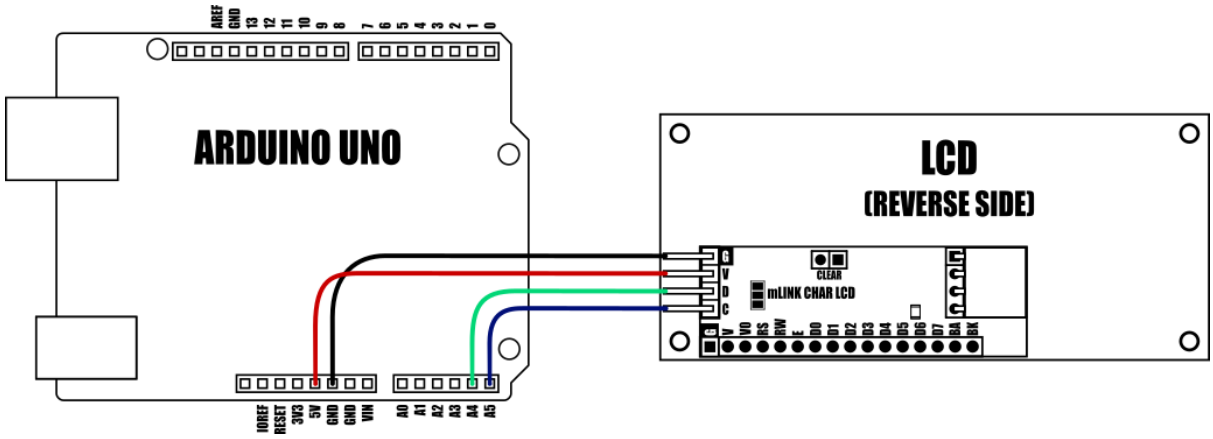
Arduino Quick Start Guide and Examples for

mLink Character LCD (HCMODU0190x)

Setting up your mLink module in 3 easy steps

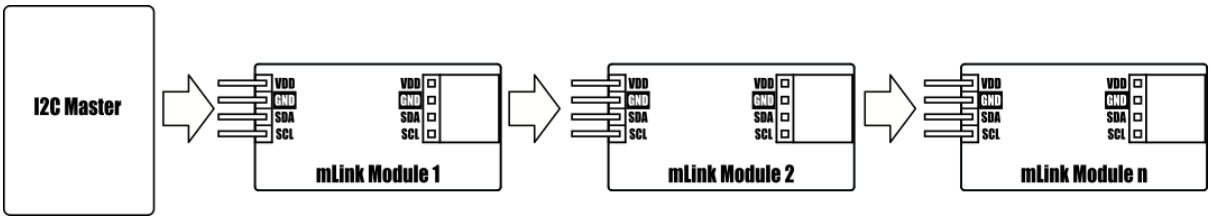
Step 1: Connecting to your microcontroller



DEVICE	VDD	GND	SDA	SCL
Uno/Nano	5V	GND	A4	A5
Pro Mini	5V	GND	A4	A5
Pro Micro	5V	GND	2	3
Mega	5V	GND	20	21
Due	5V	GND	20	21
Other microcontroller	5V	GND	I2C SDA	I2C SCL

mlink modules can be connected to any microcontroller with an I2C (IIC) serial master interface. The above example shows a mLink module connected to an Arduino Uno's I2C interface. In most cases it can be powered via the Arduinos 5V supply but please check power supply capabilities of your development board, especially when connecting multiple mLink modules.

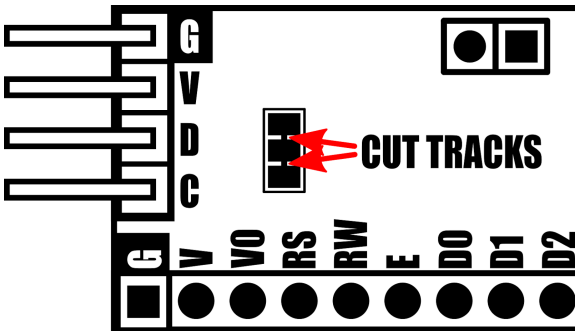
Connecting multiple mLink modules



Each mLink module includes pullup resistors (10K), which are required for the SDA and SCL data lines. This allows up to 5* mLink modules to be connected directly to an I2C master without any additional hardware or modifications.

*note maximum number of modules will be dependent on data cable lengths and module power requirements.

If more than 5 mLink modules are required to be connected to a single master interface then the built-in 10K resistors will need to be removed from the additional modules.



The two 10K pullups can be removed from the I2C bus by breaking the tracks between the 3 pads shown in the diagram above. Should you need to reconnect the 10K pullups at a later date this can be done by bridging the 3 pads with solder.

The mLink character LCD comes with a preset I2C address of 0x56 (hex). When connecting multiple modules of the same type each module's I2C address must be unique. Therefore you must change the address of any additional modules to a unique address (valid I2C addresses range between 0x08 and 0x77) before linking them together. Changing a module's I2C address can be done via the module's I2C interface. For examples of how to do this, see the Changing I2C address section within this document.

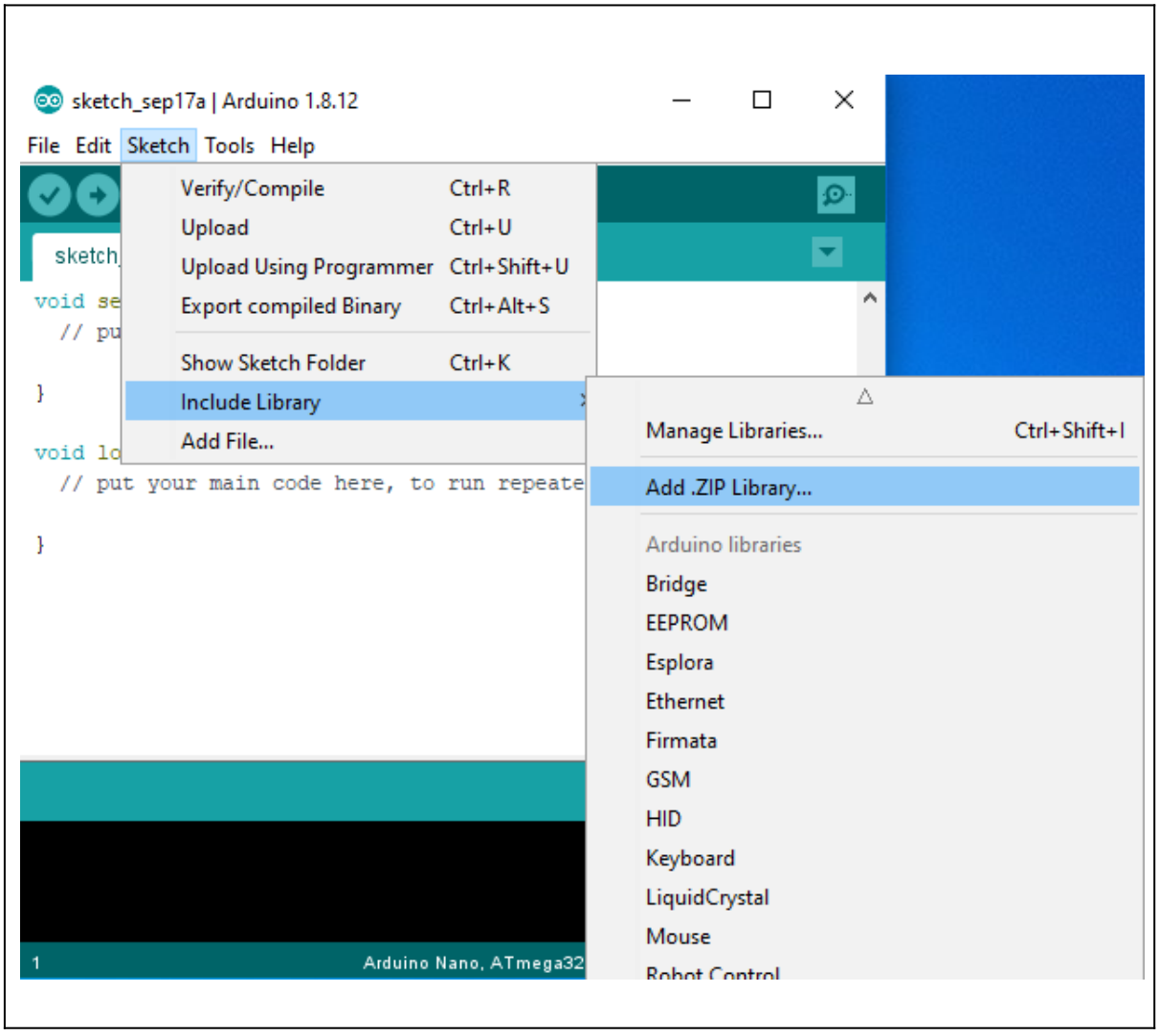
Step 2) Installing the mLink Library

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

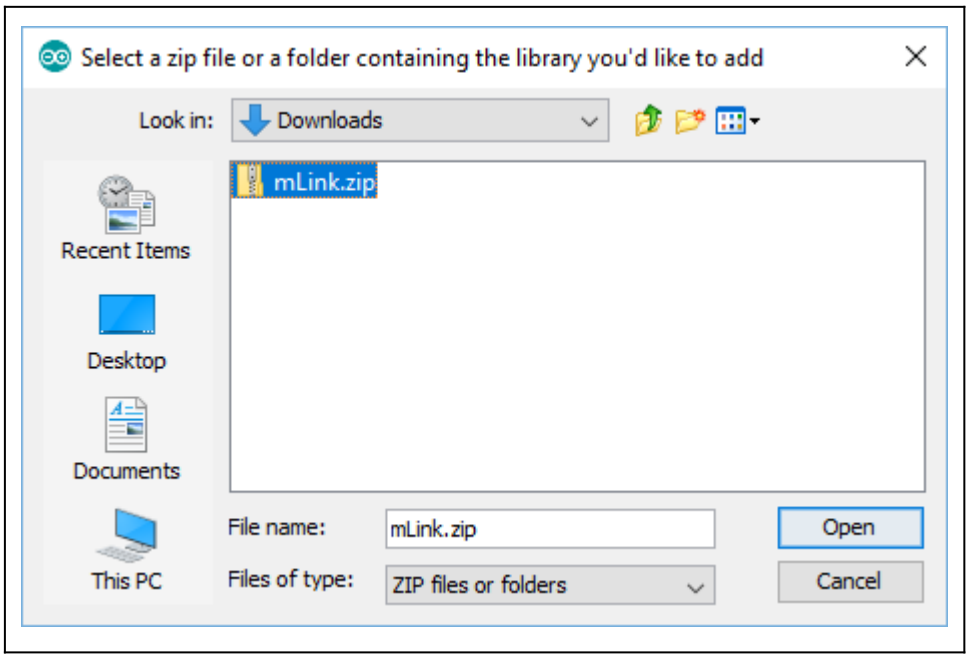
First download the mLink.zip file from the software section of our support forum here:

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 up, navigate to wherever you downloaded the mLink .zip file and select it, then click the ‘Open’ button.



Step 3) 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 Start Examples

Printing some text

Text can be printed to the screen using the mLink library's cLCD_print() macro function. When printing text, the first character to be printed will always start at the current cursor position. When the display is first powered up the cursor is always located at the top left corner of the display (column 0, row 0). However, the cursor can be moved at any point using the mLink library's cLCD_cursor() macro function.

The following example will print some text starting in the top left corner of the display:

```
#include "mLink.h"

mLink mLink;

#define I2C_ADD 0x56

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


void loop()
{
    mLink.cLCD_print(I2C_ADD, "Hello World");

    while(1);
}
```

The next example uses the cLCD_cursor() macro function to print some text stored in a character array to column 3, row 1 of the LCD:

```
#include "mLink.h"

mLink mLink;

#define I2C_ADD 0x56

char someText[] = "Hello World!";

void setup()
{
    mLink.init();

    mLink.cLCD_clear(I2C_ADD)            // Clear the screen
}
```

```
void loop()
{
   mLink.cLCD_cursor(I2C_ADD, 3, 1);        // Set the cursor to col 3 row 0
   mLink.cLCD_print(I2C_ADD, someText); // Print the character array
   while(1);
}
```

Printing a number

Bytes and integers

Signed and unsigned bytes and integers can be printed out using the same cLCD_print() function used in the previous example.

The following example will print a signed integer number to the display:

```
#include "mLink.h"

mLink mLink;

#define I2C_ADD 0x56

void setup()
{
   mLink.init();

   mLink.cLCD_clear(I2C_ADD)                  // Clear the screen
}


void loop()
{
   int number = -1234;

   mLink.cLCD_print(I2C_ADD, number);       // Print the number

   while(1);
}
```

Floating point numbers

Floating point numbers can be printed using the mLink library's cLCD_printFloat() macro function. The function takes two parameters, the floating point number to print, and number of decimal places to print it to.

The following example will print the value stored in a float variable to 2 decimal places:

```
#include "mLink.h"

mLink mLink;

#define I2C_ADD 0x56

void setup()
{
   mLink.init();

   mLink.cLCD_clear(I2C_ADD)                    // Clear the screen
}


void loop()
{
   float number = -123.456;

   mLink.cLCD_printFloat(I2C_ADD, number, 2);      // Print the number to 2 decimal places

   while(1);
}
```

Setting the backlight level

The mLink character LCD has a controllable backlight. Its brightness can be set using the mLink library's cLCD_backlight() function. The function takes a brightness level from 0 to 10 which sets the backlight brightness in 10% increments where 0 = backlight off and 10 = 100%.

The following example sets the backlight to 50% brightness:

```
#include "mLink.h"

mLink mLink;

#define I2C_ADD 0x56

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


void loop()
{
   mLink.cLCD_backlight(I2C_ADD, 5);       // Set the backlight to 50%

   while(1);
}
```

Setting the contrast

The display's contrast can be set using the mLink libraries cLCD_contrast() macro function. The function takes the contrast level as a byte value between 0 (min) and 255 (max).

Note that the contrast level is stored in the display's non-volatile memory and will be restored when the display's power is cycled.

The following example sets the contrast level to 0x55:

```
#include "mLink.h"

mLink mLink;

#define I2C_ADD 0x56

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


void loop()
{
   mLink.cLCD_contrast(I2C_ADD, 0x55);       // Set the contrast to 0x55

   while(1);
}
```

Creating and printing a custom character

The display contains 8 user definable 5x8 pixel custom characters. These 8 characters can be printed to the display using the mLink library's cLCD_printCust() macro function. This function takes the index number (0 to 7) of the custom character to print.

Before printing a custom character its bitmap must be uploaded by writing the bitmap as a sequence of 8 bytes using one of the 8 cLCD_setCustx macro commands (cLCD_setCust0 to cLCD_setCust7).

The following example shows how to define a 5x8 bitmap for a battery icon:

Bit Number	7	6	5	4	3	2	1	0	HEX
Byte 0	---	---	---	0	1	1	1	0	0h0E
Byte 1	---	---	---	1	1	1	1	1	0h1F
Byte 2	---	---	---	1	0	0	0	1	0h11
Byte 3	---	---	---	1	0	0	0	1	0h11
Byte 4	---	---	---	1	0	0	0	1	0h11
Byte 5	---	---	---	1	0	0	0	1	0h11
Byte 6	---	---	---	1	0	0	0	1	0h11
Byte 7	---	---	---	1	1	1	1	1	0x1F

The example below will write the 8 bytes defined in the previous table to custom character 0 and then print it out to the display:

```
#include "mLink.h"

mLink mLink;

#define I2C_ADD 0x56

byte bitmap = {0x0E, 0x1F, 0x11, 0x11, 0x11, 0x11, 0x11, 0x1F};  // Battery icon bitmap

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


void loop()
{
   mLink.cLCD_setCust0(CLCD_I2C_ADD, bitmap);      // Write the bitmap to custom char 0

   mLink.cLCD_printCust(CLCD_I2C_ADD, 0);          // Print custom character 0 to the display

   while(1);
}
```

Changing the I2C address

To change the module's address you can use the libraries write() function.

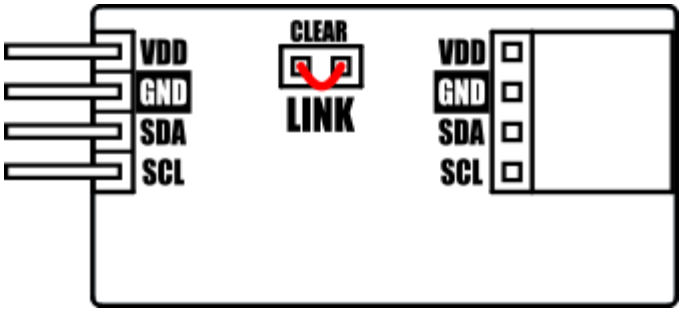
The following example changes the module's I2C address from the default 0x56 to 0x57. The new address will automatically be saved into non-volatile memory and so will retain the new address even after power has been removed from the module.

Before the module's address can be changed, the address register must first be unlocked by writing the byte value 0x55 followed by the byte value 0xAA to the register. The new address must then be written within 100ms of sending the 0xAA byte otherwise the unlock process will timeout and the process will then have to be restarted.

```
#include "mLink.h"

mLink mLink;

void setup()
{
  mLink.init();

  // Unlock the address register by writing 0x55 followed by 0xAA
  mLink.write(0x55, MLINK_ADD_REG, 0x55);
  mLink.write(0x55, MLINK_ADD_REG, 0xAA);

  // Change the I2C address from 0x56 to 0x57
  mLink.write(0x56, MLINK_ADD_REG, 0x57);
}

void loop()
{
}
```

Factory Reset

Should you wish to restore the module back to its factory default configuration, this can be done by manually forcing a factory reset. All mLink modules include a set of pads labelled clear:



Note, exact location of clear jumper may vary on your module

To perform a factory reset, carefully short the two pads together with a piece of wire or with something conductive, such as a paperclip.

Whilst shorted, connect power to the module via the VCC and GND connections.

Wait a few seconds and then remove the short from the pads.

The module's settings, including its I2C address, should now be restored back to factory defaults.

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 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