Arduino Quick Start Guide and Examples for

mLink 1,2 & 4 Channel Relay Module (HCMODU0182, HCMODU0183, & HCMODU0184)

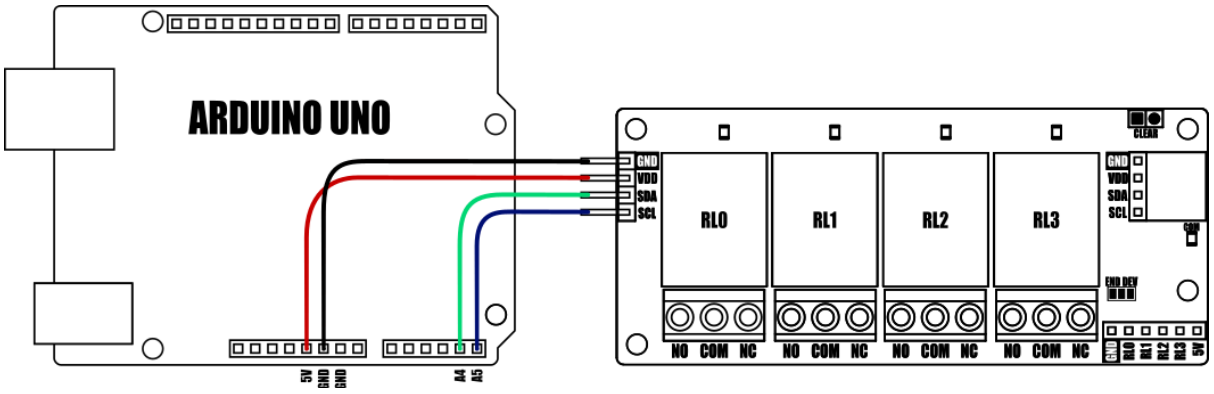
Version 1.10

REVISIONS

Date	Version	Details
7th Oct 2021	V1.00	Initial release
24th Mar 2025	V1.10	Updated for firmware version 1.01

Setting up your mLink module in 3 easy steps

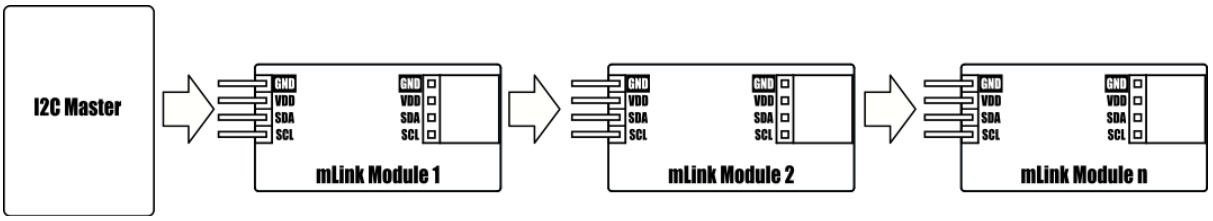
Step 1: Connecting to your microcontroller



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 mLink relay modules come with a preset I2C address of 0x52 (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.

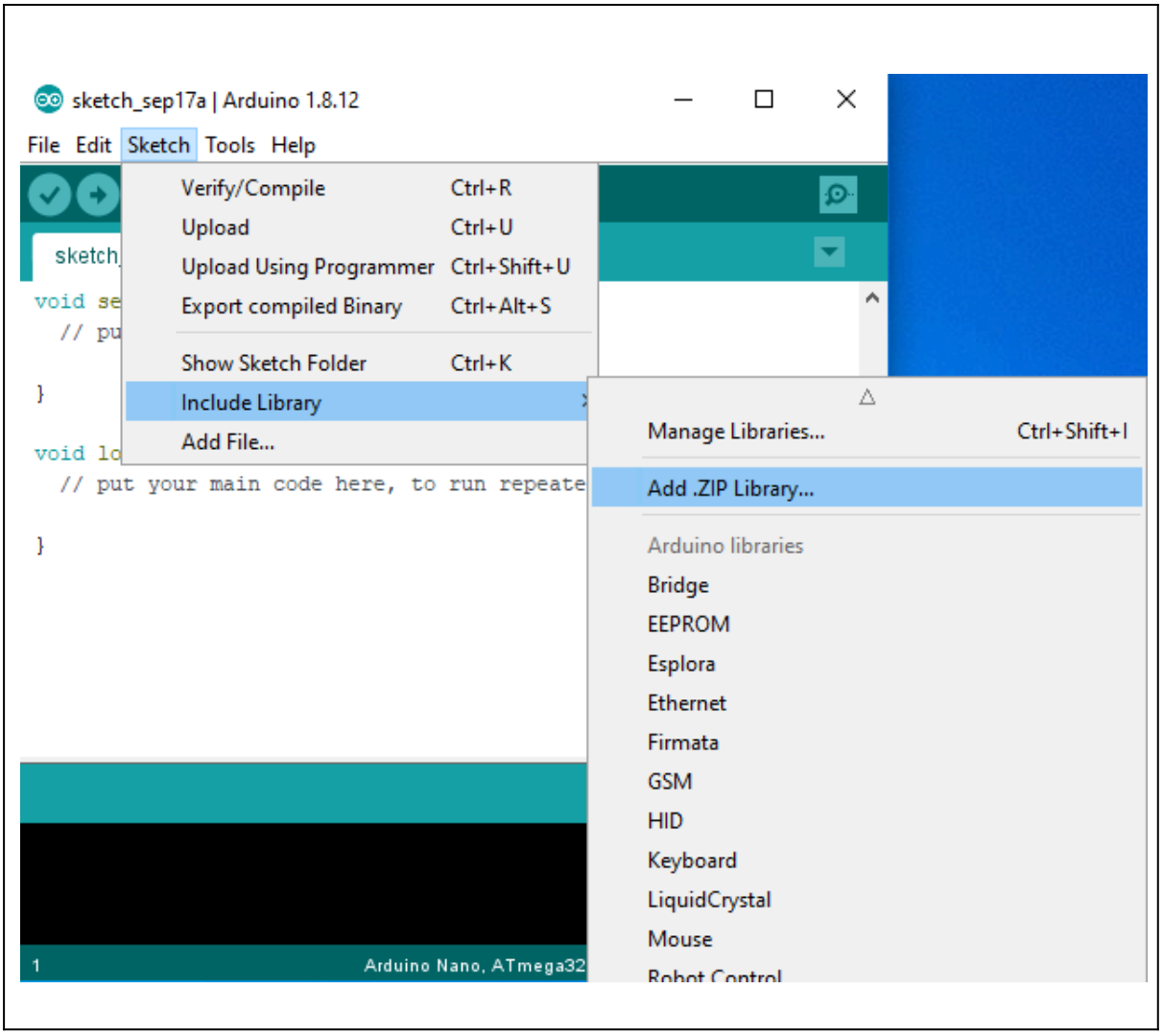
Step 2) Installing the mLink Library

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

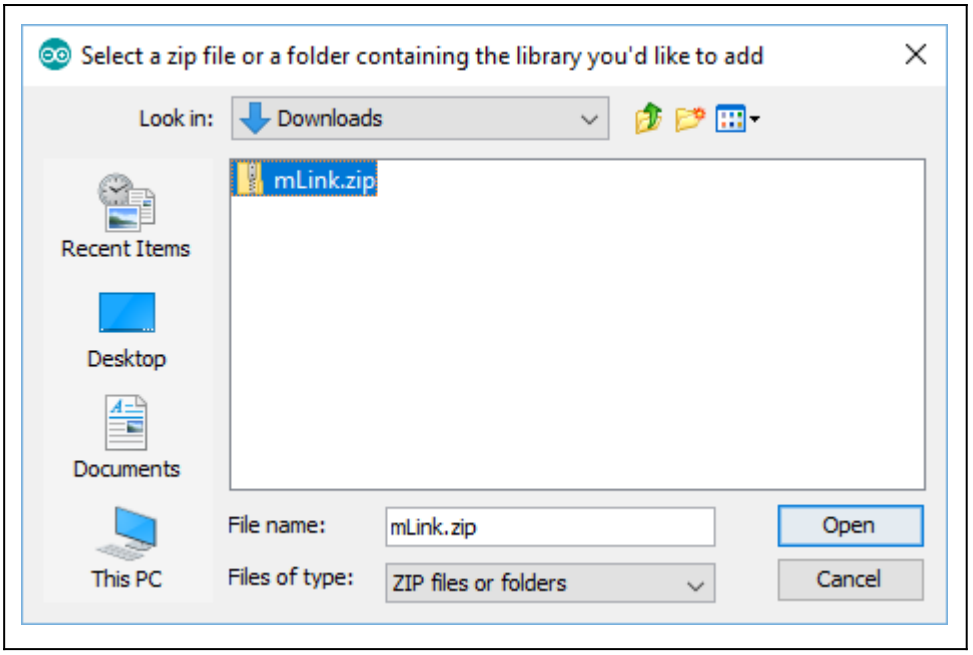
First download the mLink.zip file from the software section of our support forum here:

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

Controlling a relay via the I2C interface

Setting the state of one of the relay(s) via the I2C interface requires setting or resetting an appropriate bit in the modules relay status register. Fortunately, using the mLink libraries set relay macros this is almost as simple as controlling an digital pin on your Arduino:

mLink.SET_RLY0(add, state)	Turn relay 0 on or off
mLink.SET_RLY1(add, state)	Turn relay 1 on or off
mLink.SET_RLY2(add, state)	Turn relay 2 on or off
mLink.SET_RLY3(add, state)	Turn relay 3 on or off

Where res is a boolean representing the relays state. True / HIGH = on, false / LOW = off

The following example will toggle (blink) relay 0 on and off every 10 seconds. If your module has more than one relay you can use SET_RLY1 for relay 1, SET_RLY2 for relay 2, or SET_RLY3 for relay 3 instead.

```
#include "mLink.h"

mLink mLink;

#define I2C_ADD 0x52

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


void loop()
{
  mLink.SET_RLY0(I2C_ADD, HIGH);     // Turn relay 0 ON.
  delay(10000);

  mLink.SET_RLY0(I2C_ADD, LOW);     // Turn relay 0 OFF
  delay(10000);
}
```

Alternate method (firmware V1.01 28/11/24 and above)

Alternatively a relay can be turned on or off by specifying its index, 0 for relay 0, 1 for relay 1 etc. The following example will turn relay 0 on and off every 10 seconds.

```
#include "mLink.h"

mLink mLink;

#define I2C_ADD 0x52

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


void loop()
{
  mLink.RLY_ON(I2C_ADD, 0);     // Turn relay 0 ON.
  delay(10000);

  mLink.RLY_OFF(I2C_ADD, 0);     // Turn relay 0 OFF
  delay(10000);
}
```

Reading the state of a relay via the I2C interface

If you need to check the current state of a relay, for instance to check if its state has been changed via the external RL pin header, this can be done by reading the appropriate bit from the modules relay status register. As with the previous example the mLink library has macros to make this straightforward:

mLink.READ_RLY0(add)	Returns the state of relay 0
mLink.READ_RLY1(add)	Returns the state of relay 1
mLink.READ_RLY2(add)	Returns the state of relay 2
mLink.READ_RLY3(add)	Returns the state of relay 3

The above functions will return a boolean value representing the current relay state where: True / HIGH = on, false / LOW = off

The following example will read the current state of relay 0 and will output the state to the serial monitor. If your module has more than one relay you can use READ_RLY1 for relay 1, READ_RLY2 for relay 2, or READ_RLY3 for relay 3 instead.

```
#include "mLink.h"                                  // Include the library

mLink mLink;                                        // Create an instance of the library


#define I2C_ADD 0x52                                // Default I2C address


void setup()
{
  Serial.begin(9600);

  mLink.init();                                     // Initialise the library

  boolean state = mLink.READ_RLY0(I2C_ADD); // Get the current state of relay 0

  Serial.print("Relay 0 state = ");

  if(state == 1)
    Serial.println("ON");
  else
    Serial.println("OFF");
}


void loop()
{

}
```

Relay timer mode

Putting a relay into timer mode allows you to specify the minimum amount of time a relay will stay energised for when triggered via either the I2C interface or its external RL trigger pin. To put a relay into timer mode its ON time in seconds needs to be set by writing to the appropriate modules on time registers. Using the mLink library this can be easily done with one of its predefined macros:

```
mLink.RLY0_SetOnTime(add, time)        Sets the relay on for relay 0
mLink.RLY1_SetOnTime(add, time)        Sets the relay on for relay 1
mLink.RLY2_SetOnTime(add, time)        Sets the relay on for relay 2
mLink.RLY3_SetOnTime(add, time)        Sets the relay on for relay 3
```

Where time is a 16 bit unsigned integer containing the on time in seconds.

The following example sets the on time for relay 0 to 10 seconds. The maximum on time for a relay is 65535 seconds. Setting the on time to 0 will disable timer mode and the relay will operate as normal. If your module has more than one relay you can use RLY1_SetOnTime for relay 1, RLY2_SetOnTime for relay 2, or RLY3_SetOnTime for relay 3 instead.

```
#include "mLink.h"                          // Include the library

mLink mLink;                                // Create an instance of the library

#define I2C_ADD 0x52                        // Default I2C address

void setup()
{
  mLink.init();                             // Initialise the library

  mLink.RLY0_SetOnTime(I2C_ADD, 10); // Set the on time for relay 0 to 10 seconds
}


void loop()
{

}
```

Timer resolution (firmware V1.01 28/11/24 and above)

For firmware version 1.01 and above, the timer resolution for each relay can be set to either 1 second increments (default) giving a timer range of 1 to 65535 seconds, or 100ms increments giving a timer range of 100ms to 6553.5 seconds. The resolution can also be independently set for each relay with the following library macros:

mLink.RLY0_TIMER_RES(add, res)	Sets the timer resolution for Relay 0.
mLink.RLY1_TIMER_RES(add, res)	Sets the timer resolution for Relay 1.
mLink.RLY2_TIMER_RES(add, res)	Sets the timer resolution for Relay 2.
mLink.RLY3_TIMER_RES(add, res)	Sets the timer resolution for Relay 3.

Where res is a boolean value specifying the resolution. A value of 0 or RLY_RES_1S (default) will set the resolution to 1 second. A value of 1 or RLY_RES_100MS will set the resolution to 100 milliseconds.

The following example will set the timer resolution for relay 0 to 100ms and then set an on timer of 500ms

```
#include "mLink.h"

mLink mLink;

#define I2C_ADD 0x52

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


void loop()
{
  mLink.RLY_RLY0_TIMER_RES((I2C_ADD, RLY_RES_100MS);  // Set relay 0 timer res to 100ms

  mLink.RLY0_SetOnTime(I2C_ADD, 5);    // Set relay 0 ON time to 500ms
}
```

Note that once set, the module stores the new setting in its non-volatile memory and will remember the setting after the module's power has been cycled.

Changing the I2C address

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

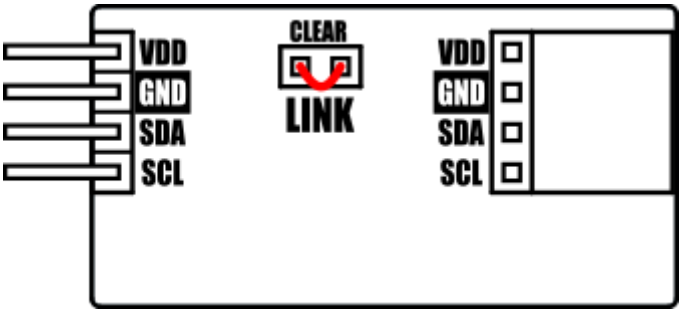
The following example changes the module's I2C address from the default 0x51 to 0x52. The new address will automatically be saved into non-volatile memory and so will retain the new address even after power has been removed from the module.

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(0x51, MLINK_ADD_REG, 0x55);
  mLink.write(0x51, MLINK_ADD_REG, 0xAA);

  // Change the I2C address from 0x50 to 0x51
  mLink.write(0x51, MLINK_ADD_REG, 0x52);
}

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