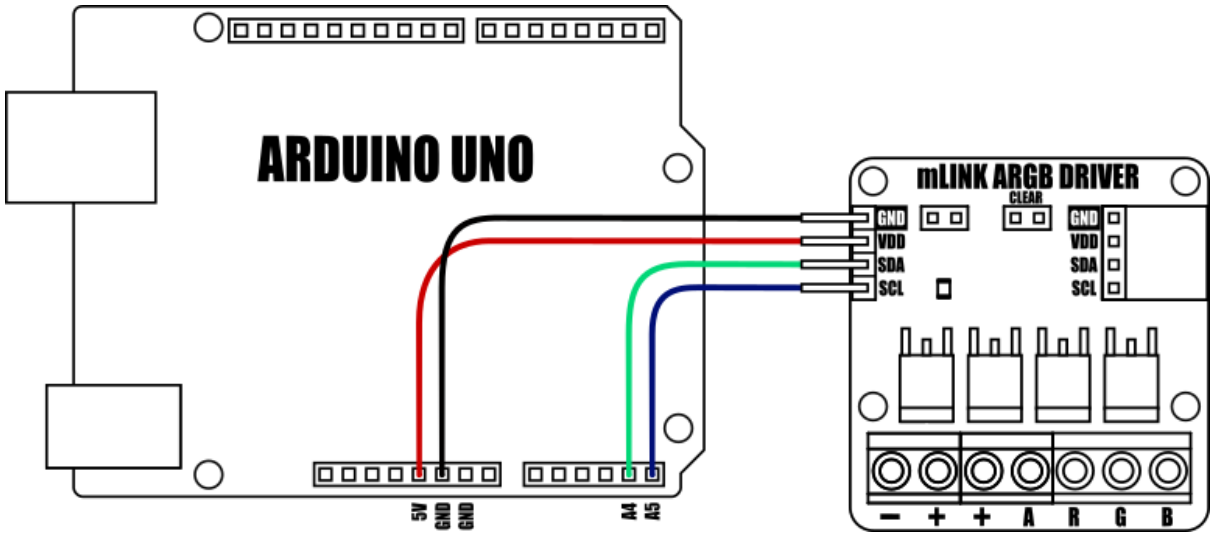
Arduino Quick Start Guide and Examples for

mLink RGBW LED Controller (HCMODU0185)

Setting up your mLink module in 3 easy steps

Step 1: Connecting to your microcontroller

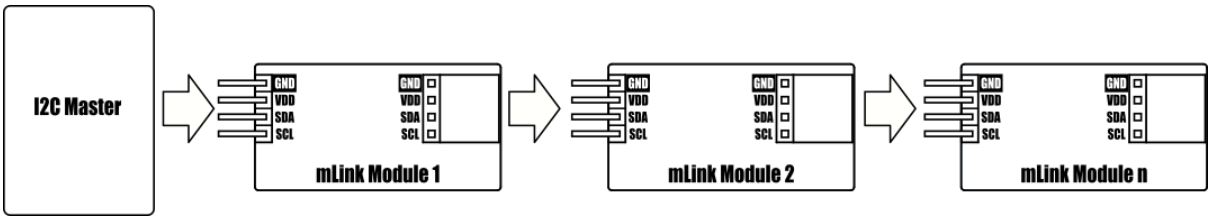


DEVICE	VDD	GND	SDA	SCL
Uno/Nano	3.3V/5V	GND	A4	A5
Pro Mini	3.3V/5V	GND	A4	A5
Pro Micro	3.3V/5V	GND	2	3
Mega	3.3V/5V	GND	20	21
Due	3.3V/5V	GND	20	21
Other microcontroller	3.3V/5V	GND	I2C SDA	I2C SCL

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. It can be powered via a 5V or 3.3V supply (depending on the logic levels you require the modules digital IO pins to operate at, and voltage requirements of any additional mLink modules connected to the modules I2C output).

© Hobby Components Ltd

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 RGBW controller comes with a preset I2C address of 0x53 (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

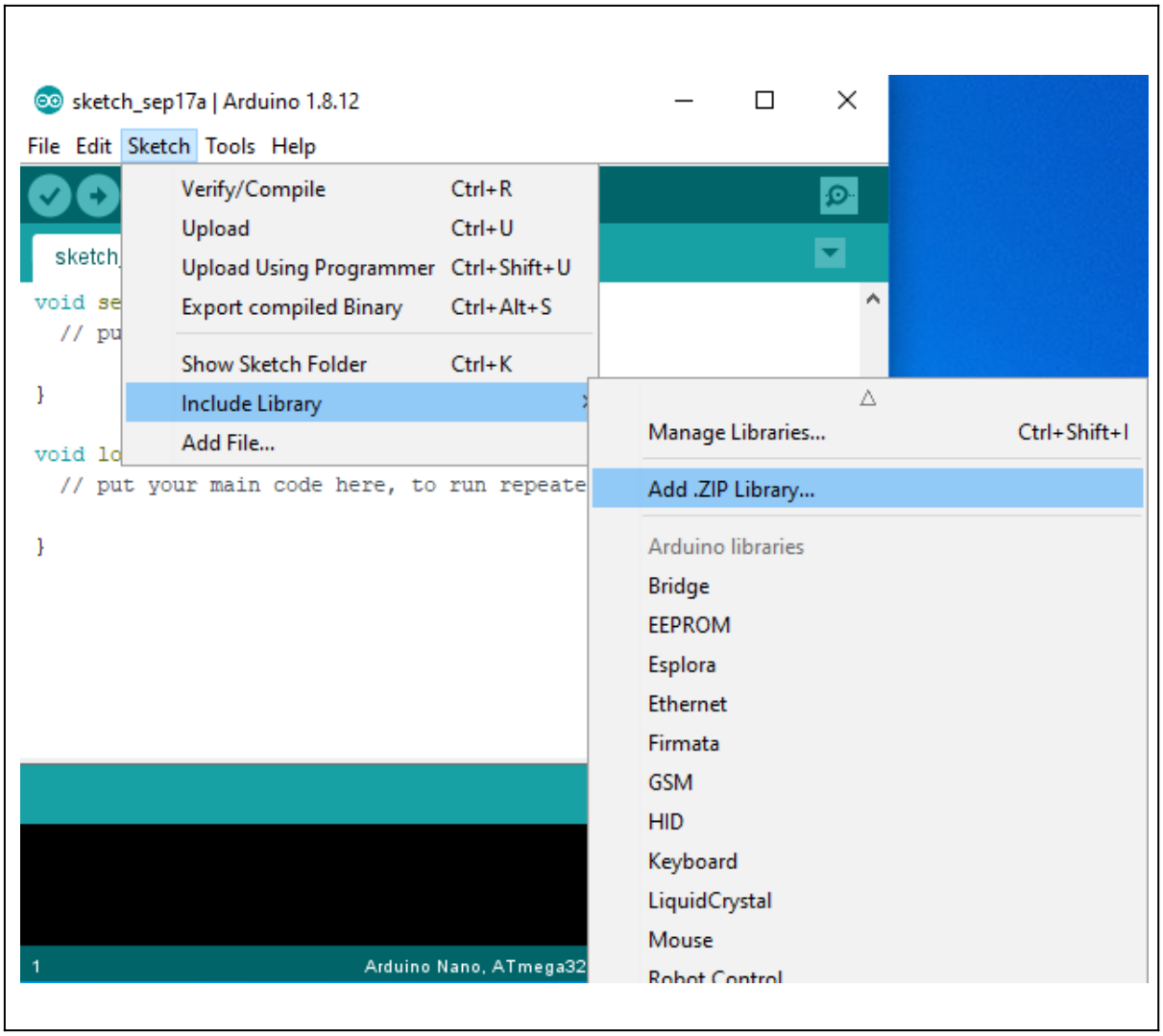
Adding the mLink library to your Arduino IDE can be done in the same way as any other standard Arduino library:

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

https://hobbycomponents.com/mLink

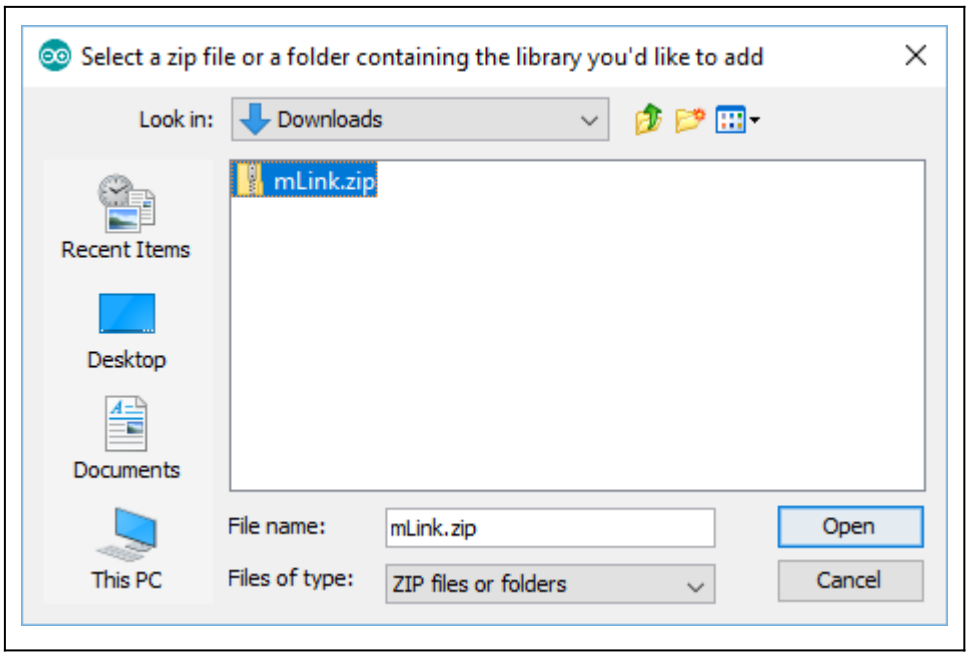
Once downloaded, open up your Arduino IDE and go to Sketch->Include Library->Add .ZIP Library.

© Hobby Components Ltd



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.

© Hobby Components Ltd



Step 3) Including the mLink library in your sketch

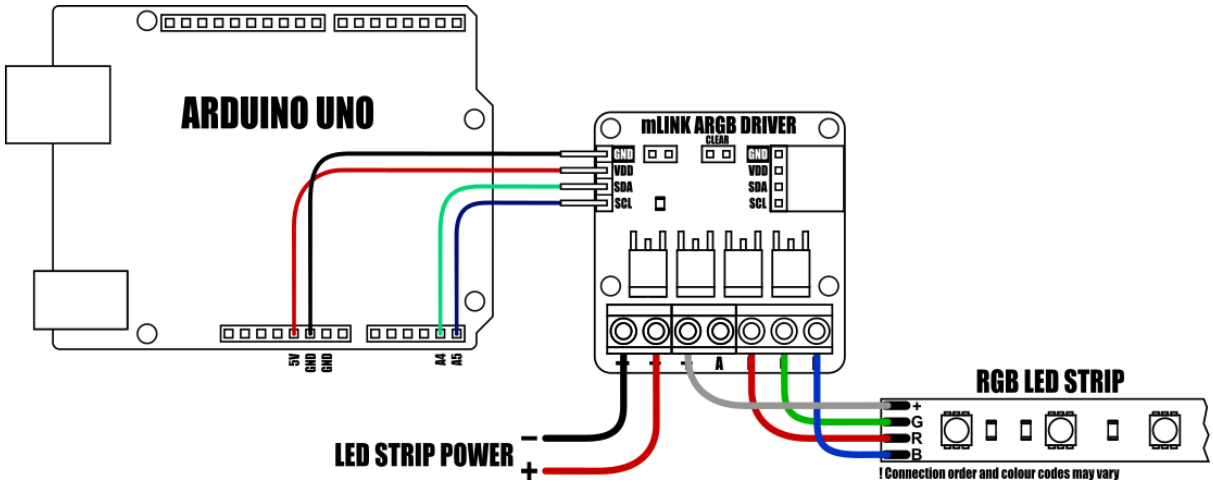
Adding the mLink library to your sketch consists of 3 steps; Firstly, include the mLink header file (mLink.h) at the top of your sketch, create an instance of the library, then finally initialise the library inside the startup() function:

```
// 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()
{
}
```


© Hobby Components Ltd

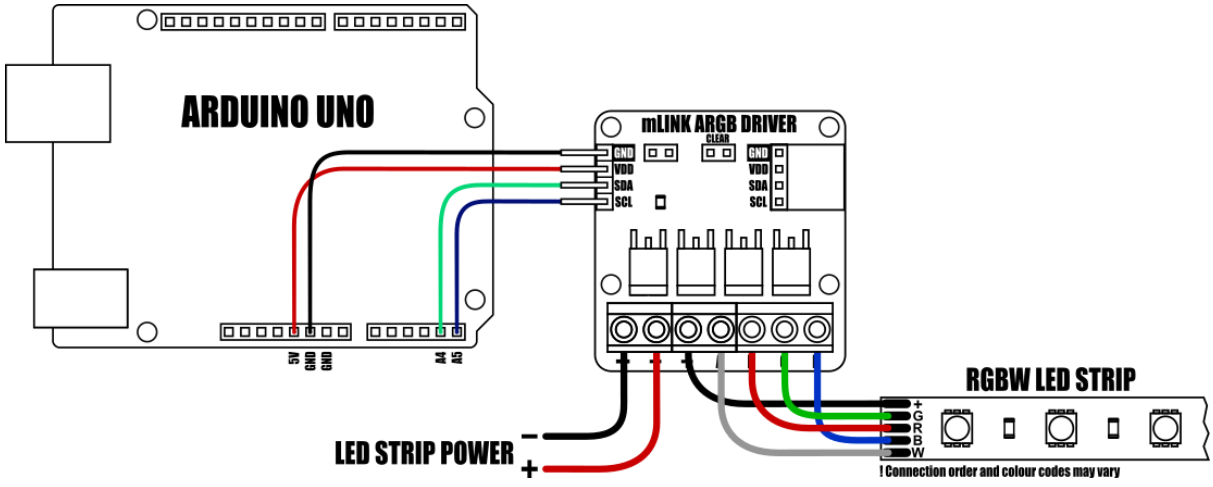
Quick Start Examples

RGB LED strip connections



The RGBW controller can directly control RGB LED strips that have +V (5V, 12V, or 24V), -red, -green, & -blue connections. Power for the LED strip can be connected to the module via the + & - screw terminals. Note that these terminals supply power to the LED strip only. To power the module itself, a 3.3/5V power supply must be connected via the VDD & GND header pins.

RGBW LED strip connections



Connecting to an RGBW LED strip is similar to an RGB strip but with the addition or a -white connection. As with the RGB strip the RGBW strip must be of the +V, -R, -G, -W type. The additional W connection can be connected to the modules A (aux/white) screw terminal.

© Hobby Components Ltd

Setting the colour

The RGBW LED controller has four 8 bit registers that set the brightness level of the red, green, blue, and white (aux) outputs. A writing value of 255 will set maximum brightness and writing a value of 0 will turn the output off. Writing to each register can be done using the mLink.write() function as shown in the example below.

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

mLink mLink;                                            // Create an instance of the library

#define I2C_ADD 0x53                                    // Default I2C address


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

  mLink.write(I2C_ADD, RGBW_BRIGHTNESS, 255);           // Set brightness to maximum

  mLink.write(I2C_ADD, RGBW_R_LEVEL, 255);              // Set red to 100%
  mLink.write(I2C_ADD, RGBW_G_LEVEL, 127);              // Set green to 50%
  mLink.write(I2C_ADD, RGBW_B_LEVEL, 63);               // Set blue to 25%
  mLink.write(I2C_ADD, RGBW_W_LEVEL, 255);              // Set white to 100%
}


void loop()
{
}
```


© Hobby Components Ltd

Selecting a predefined cycle mode

The module also has the ability to automatically cycle through a set of predefined RGB colour patterns. These can be selected by writing to the module RGBW_LOAD_CYCLE register with one of the following pattern labels.

Mode	Label	Description
0	RGBW_CYCLE_USER	User defined mode
1	RGBW_CYCLE_FAST_RGB_COLOUR_CYCLE	Cycle fast between red, green, & blue
2	RGBW_CYCLE_MED_RGB_COLOUR_CYCLE	Cycle medium between red, green, & blue
3	RGBW_CYCLE_SLOW_RGB_COLOUR_CYCLE	Cycle slow between red, green, & blue
4	RGBW_CYCLE_FAST_RG_CYCLE	Cycle fast between red & green
5	RGBW_CYCLE_ALARM_INTER_PULSE	Intermittent red pulse
6	RGBW_CYCLE_ALARM_CONT_PULSE	Constantly pulse red
7	RGBW_CYCLE_RGB_CONT_PULSE	One second step between each colour
8	RGBW_CYCLE_FLAME	Flame effect

Example puts the module into medium RGB cycle mode

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

mLink mLink;                                // Create an instance of the library

#define I2C_ADD 0x53                        // Default I2C address


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

  mLink.write(I2C_ADD, RGBW_BRIGHTNESS, 255);          // Set brightness to maximum

  // Set cycle mode to medium RGB cycle
  mLink.write(I2C_ADD, RGBW_LOAD_CYCLE, RGBW_CYCLE_MED_RGB_COLOUR_CYCLE);
}


void loop()
{
}
```


© Hobby Components Ltd

User defined cycle mode

Beside the predefined cycle modes a user defined cycle can be programmed into the module. To create a user defined cycle up to 8 different RGB levels can be loaded into the modules pattern buffer registers. Once the colours have been loaded the module can cycle through each colour by specifying how many intermediate colour steps to take between each colour and how many milliseconds to wait between each step.

The following example loads 3 RGB colours (red, green, & blue) into the pattern buffer then sets the number of intermediate steps to 255 with a delay of 10ms between each step.

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

mLink mLink;                                // Create an instance of the library

#define I2C_ADD 0x53                        // Default I2C address


// Create an array with 3 colours (maximum is 8) to load into the pattern buffer
byte colours[][3] = {{255,0,0},     // Colour 1 = Red
                     {0,255,0},     // Colour 2 = Green
                     {0,0,255}      // Colour 3 = Blue
                    };


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


  mLink.write(I2C_ADD, RGBW_CYCLE(colours));          // Load the colours into the buffer

  mLink.write(I2C_ADD, RGBW_BRIGHTNESS, 255);       // Set brightness to maximum
  mLink.write(I2C_ADD, RGBW_CYCLE_COLOURS, 3);  // There are 3 colors to cycle though
  mLink.write(I2C_ADD, RGBW_CYCLE_STEPS, 255);    // Make 255 steps between each colour
  mLink.writeInt(I2C_ADD, RGBW_CYCLE_STEP_SPEED, 10);   // Wait 10ms between each step
}


void loop()
{
}
```


© Hobby Components Ltd

Saving custom settings

Custom settings can be stored into the modules non-volatile memory allowing the settings to be automatically restored after power has been cycled. To store custom settings use the following command:

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

mLink mLink;                                // Create an instance of the library

#define I2C_ADD 0x53                        // Default I2C address


void setup()
{
  mLink.write(I2C_ADD, RGBW_SAVE);
}


void loop()
{
}
```


© Hobby Components Ltd

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 0x53 to 0x54. 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.

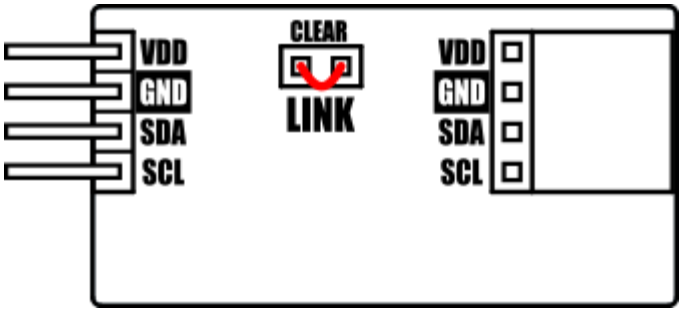
Before the module's address can be changed, the address register must first be unlocked by writing the byte value 0x55 followed by the byte value 0xAA to the register. The new address must then be written within 100ms of sending the 0xAA byte otherwise the unlock process will timeout and the process 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(0x53, MLINK_ADD_REG, 0x55);
  mLink.write(0x53, MLINK_ADD_REG, 0xAA);

  // Change the I2C address from 0x53 to 0x54
  mLink.write(0x53, MLINK_ADD_REG, 0x54);
}

void loop()
{
}
```


© Hobby Components Ltd

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 labeled 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.

© Hobby Components Ltd

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

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

© Hobby Components Ltd