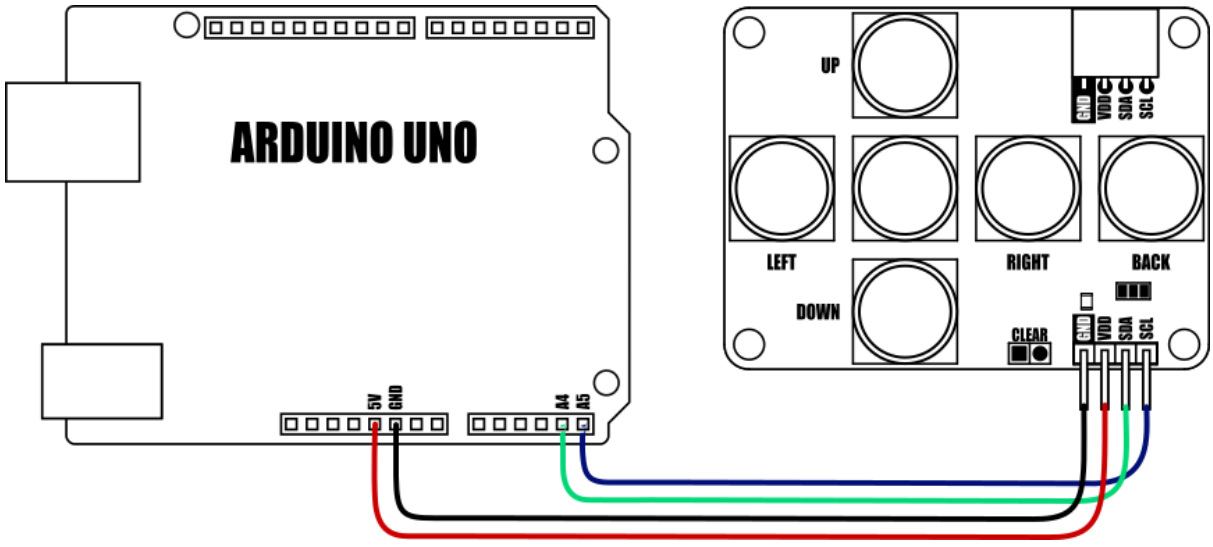
Arduino Quick Start Guide and Examples for

mLink 6 Button Pad (HCMODU0193)

Setting up your mLink module in 3 easy steps

Step 1: Connecting to your microcontroller

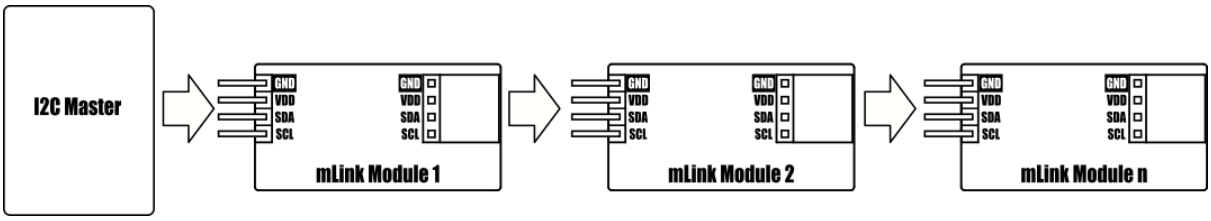


DEVICE	VDD	GND	SDA	SCL
Uno/Nano	5V	GND	A4	A5
Pro Mini	5V	GND	A4	A5
Pro Micro	5V	GND	2	3
Mega	5V	GND	20	21
Due	5V	GND	20	21
Other microcontroller	5V	GND	I2C SDA	I2C SCL

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.

© Hobby Components Ltd

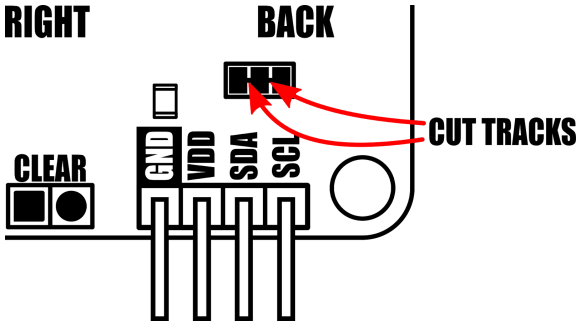
Connecting multiple mLink modules



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 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 button pad comes with a preset I2C address of 0x59 (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.

© Hobby Components Ltd

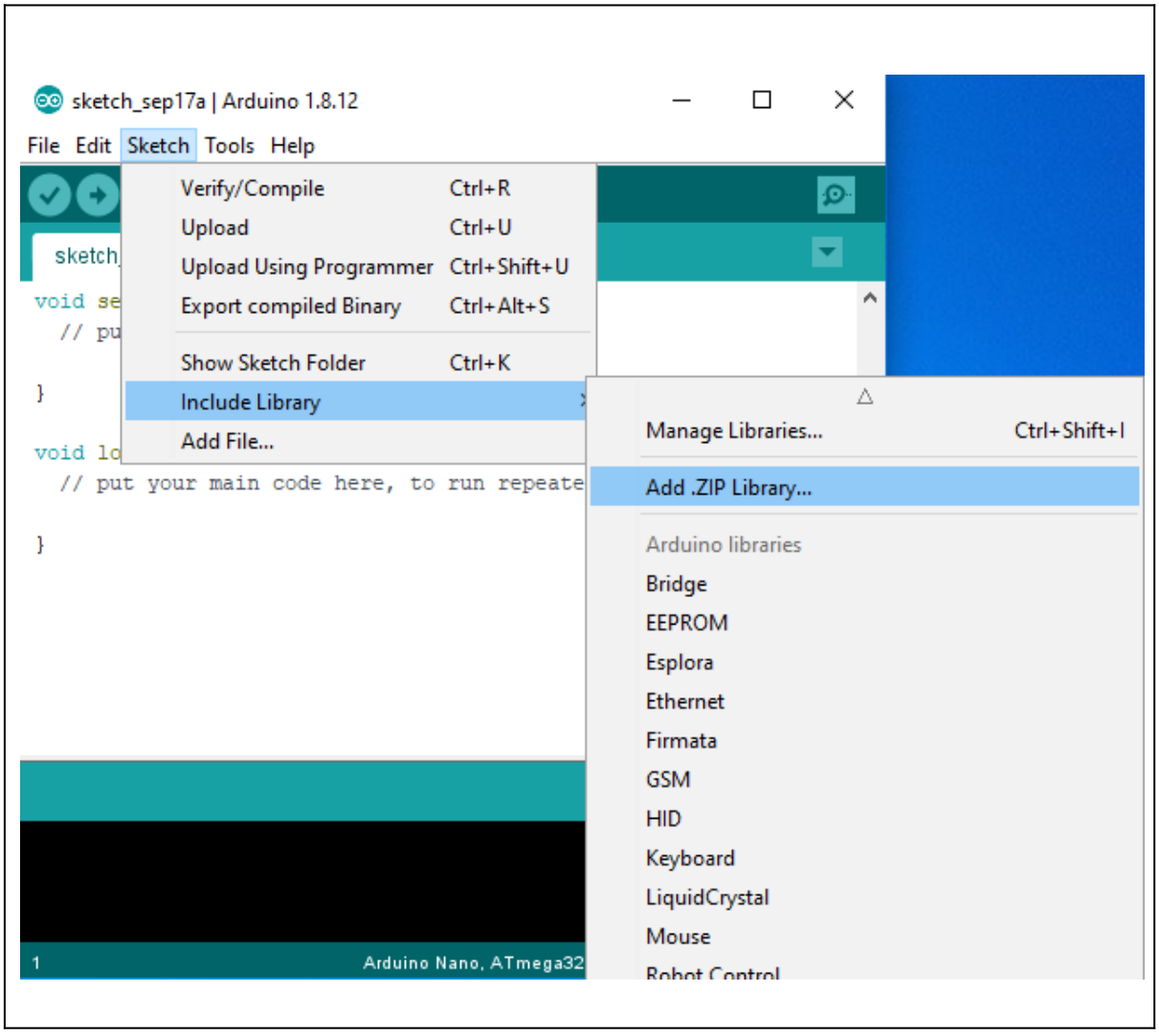
Step 2) Installing the mLink Library

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:

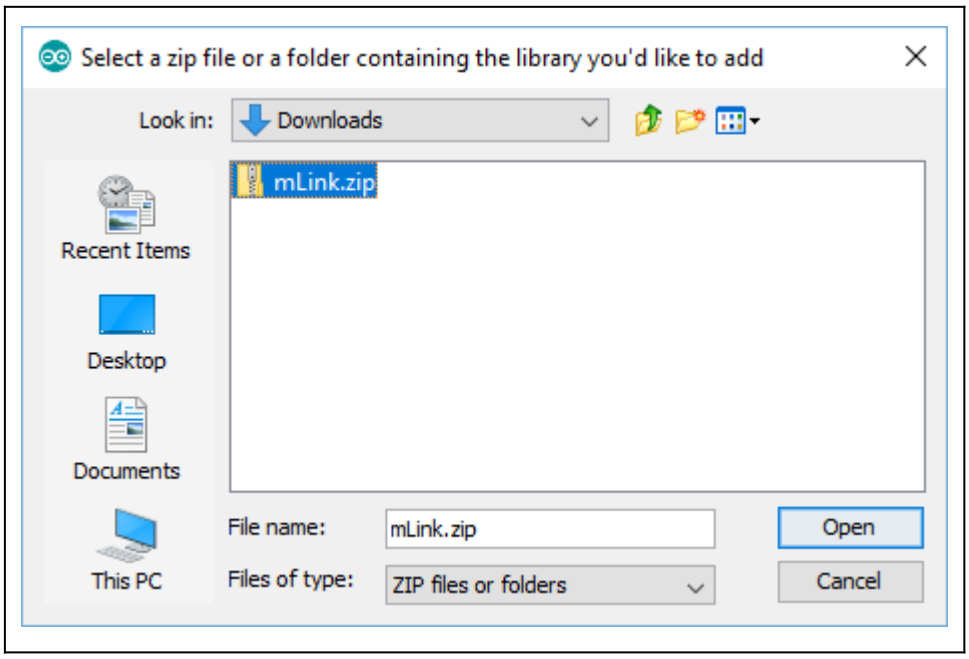
https://hobbycomponents.com/mLink

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.

© 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

Reading key presses from the keypad

When a button is pressed, the module will store that button's keycode in its buffer until it is read out of the buffer by the user. The module is capable of storing up to 16 key presses in the order they were pressed. By storing the button presses in a buffer it means that the keypad does not have to be constantly checked by your sketch to capture a keypress as the module will do this for you.

To read a key out of the buffer you must first check to see if there are any button presses currently stored in the buffer. This can be done by testing to see if the buffer is empty using the bPad_Empty() macro function. If the buffer is not empty, i.e., there is at least one button pressed stored in it, then the key code for the first button stored in the buffer can be read out using the bPad_Read() macro function.

The following example will check to see if there are any key presses currently stored in the buffer and if so will read the buffer and print the first one to the serial port. It will continually repeat this process until there are no more buttons stored in the buffer (buffer is empty):

```
#include "mLink.h"

mLink mLink;

#define I2C_ADD 0x59

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

  mLink.init();
}


void loop()
{
  boolean empty = mLink.bPad_Empty(I2C_ADD);  // Check to see if the buffer is empty

  if(!empty)
  {
    byte code = mLink.bPad_Read(I2C_ADD);          // If not then read the buffer and print it out
    Serial.println(code);
  }
}
```


© Hobby Components Ltd

Checking the current state of the buttons

If you wish to know the current state of any of the buttons you can use one of 6 button state macros (one for each button).

The following example will check the state of each button and if a button is pressed, will then print it to the serial port:

```
#include "mLink.h"

mLink mLink;

#define I2C_ADD 0x59

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

  mLink.init();
}


void loop()
{
  boolean state = mLink.bPad_UpState(I2C_ADD);   // Get the current state of the up button
  if(state)
    Serial.println("UP button is pressed!");              // If state = true then left button is pressed


  state = mLink.bPad_LeftState(I2C_ADD);             // Get the current state of the left button
  if(state)
    Serial.println("LEFT button is pressed!");           // If state = true then left button is pressed


  state = mLink.bPad_DownState(I2C_ADD);             // Get the current state of the down button
  if(state)
    Serial.println("DOWN button is pressed!");           // If state = true then down button is pressed


  state = mLink.bPad_RightState(I2C_ADD);            // Get the current state of the right button
  if(state)
    Serial.println("RIGHT button is pressed!");          // If state = true then right button is pressed


  state = mLink.bPad_SelectState(I2C_ADD);           // Get the current state of the select button
  if(state)
    Serial.println("SELECT button is pressed!");         // If state = true then select button is pressed


  state = mLink.bPad_BackState(I2C_ADD);             // Get the current state of the back button
  if(state)
    Serial.println("BACK button is pressed!");           // If state = true then back button is pressed
}
```


© Hobby Components Ltd

Setting the debounce level

Most switches are prone to an issue called bouncing. When a button is pressed its switch contacts can bounce a few times before finally making a permanent contact. This can result in multiple presses being registered. To mitigate this issue the module applies automatic debouncing by reading the switch multiple times to confirm the switch contacts have closed. The level of debouncing can be changed between 0 (no debouncing) and 254 (maximum debouncing). By default the module is set to a level of 200. Reducing this value will make the keypad more responsive to button presses, increasing it will apply more debouncing.

Note: The debouncing level is stored in the modules non-volatile memory and so does not need to be re-written each time you run your sketch.

The following example shows how to change the debouncing level:

```
#include "mLink.h"

mLink mLink;

#define I2C_ADD 0x59

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

  mLink.init();
}


void loop()
{
  mLink.bPad_Debounce(I2C_ADD, 100);                    // Set the debounce level to 100

  Serial.print("Debounce level now set to: ");          // Check the debounce level
  Serial.println(mLink.read(I2C_ADD, BPAD_DEBOUNCE));

  while(1);
}
```


© 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 0x59 to 0x5A. 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.

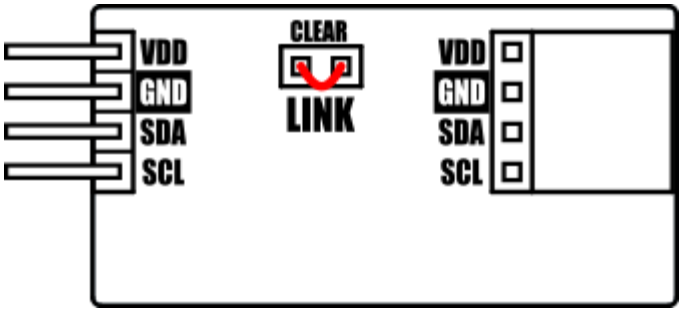
Before the module's address can be changed, the address register must first be unlocked by writing the byte value 0x55 followed by the byte value 0xAA to the register. The new address must then be written within 100ms of sending the 0xAA byte otherwise the unlock process will timeout and the process 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(0x59, MLINK_ADD_REG, 0x55);
  mLink.write(0x59, MLINK_ADD_REG, 0xAA);

  // Change the I2C address from 0x59 to 0x5A
  mLink.write(0x59, MLINK_ADD_REG, 0x5A);
}

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 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.

© 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