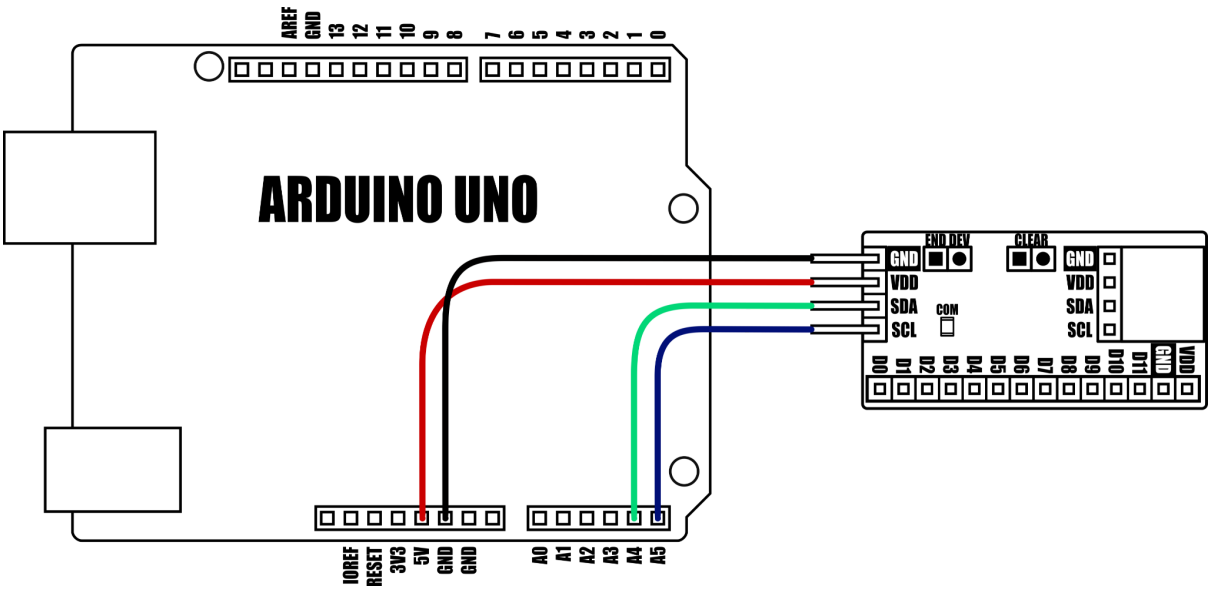
Arduino Quick Start Guide and Examples for

mLink 12bit Digital Port Expander Module (HCMODU0180)

Setting up your mLink module in 3 easy steps

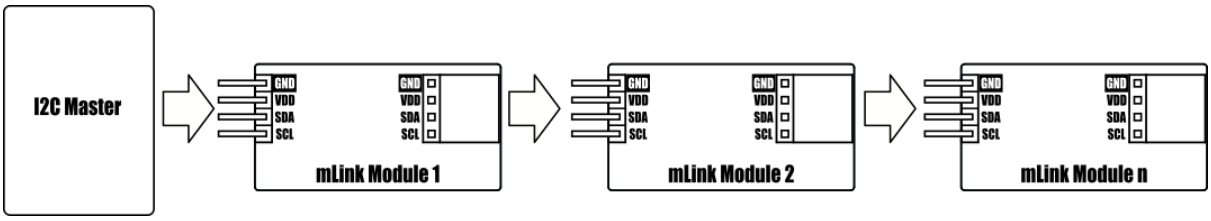
Step 1: Connecting to your microcontroller



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).

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 digital port expander comes with a preset I2C address of 0x50 (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.

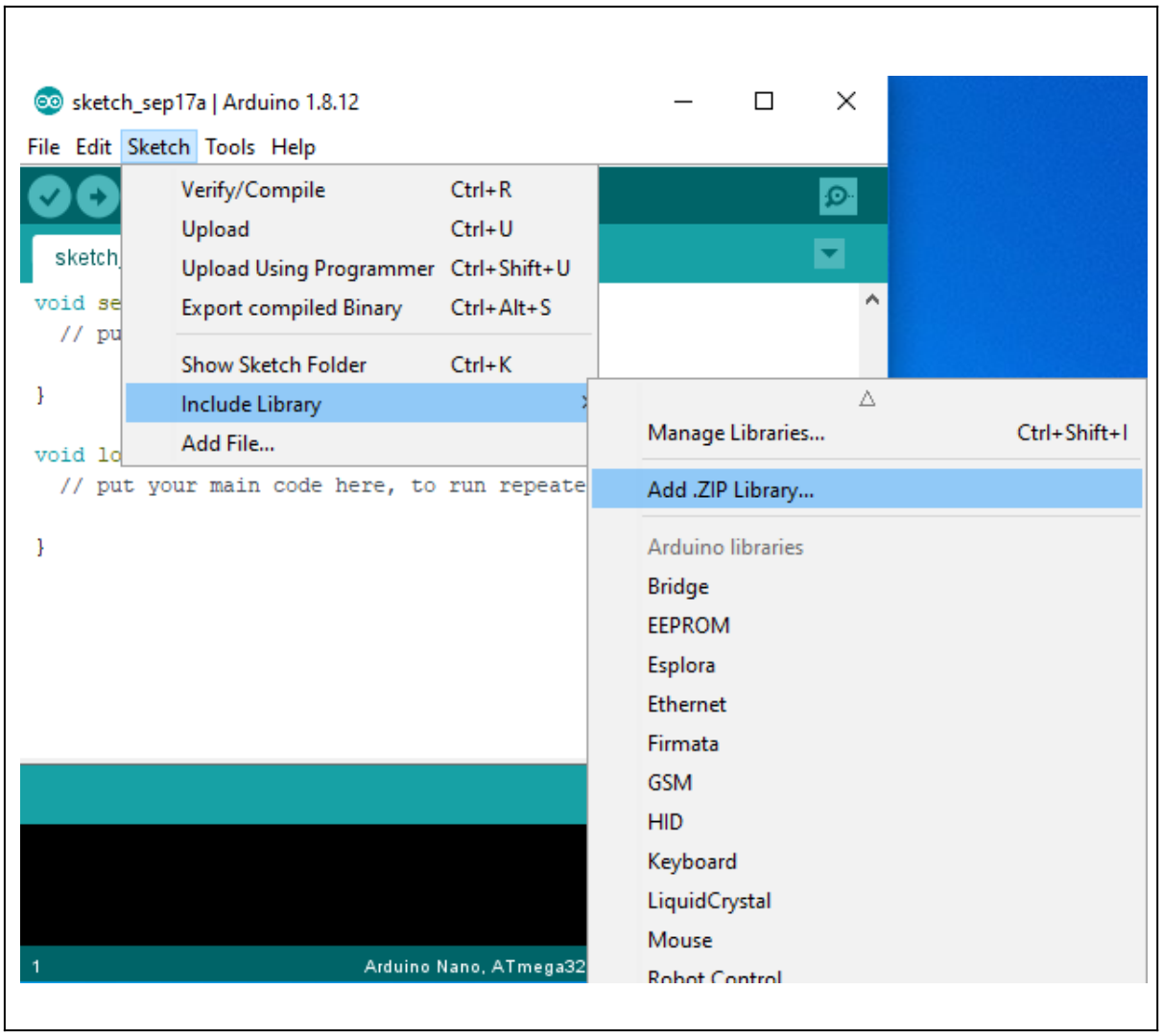
Step 2) Installing the mLink Library

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

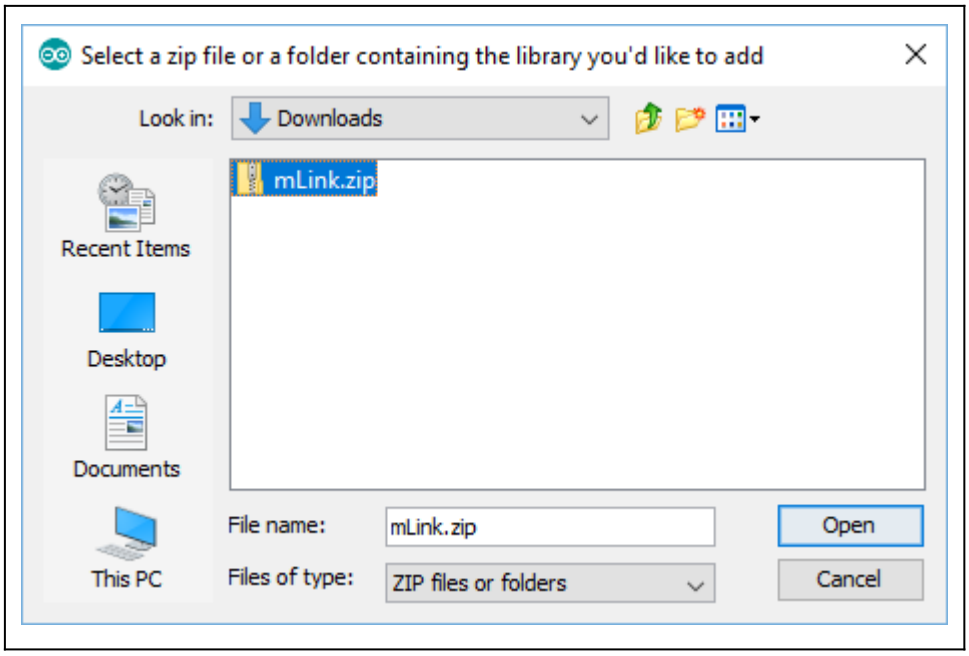
First download the mLink.zip file from the software section of our support forum here:

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

Configuring the port direction

The digital pins on the port expander module can be configured to be either inputs (with pullups) or outputs (push-pull). Note, by default all 12 pins are set to inputs.

To set the direction of a pin you must write to the appropriate bit in the module's data direction registers. Using the mLink library this can easily be done in one of two ways, by using the libraries writeBit() function to set the direction of an individual pin, or by using the writeInt() to set the direction of all 12 pins at once:

Setting the direction of individual pins

To set the direction of an individual pin you can use the writeBit() library function, just specify the modules I2C address (default 0x50), the pin you wish to set the direction of (you can specify the pin number or use one of the predefined library values DIO12_DIR_D0 to DIO12_DIR_D11), and the direction to set it to. 0 will set the pin to an output and 1 to an input or for readability you can use the libraries DIO12_OUTPUT and DIO12_INPUT definitions.

The following example will set pin D0 to an output and pin D1 to an input.

```
#include "mLink.h"

mLink mLink;

#define I2C_ADD 0x50

void setup()
{
  mLink.init();

  // Set pin 0 to an output
  mLink.writeBit(I2C_ADD, DIO12_D0_OUTPUT);

  // Set pin 1 to an input
  mLink.writeBit(I2C_ADD, DIO12_D1_INPUT);
}


void loop()
{
}
```

Setting the direction of all 12 pins at once

To set the direction of all 12 pins at once you can use the mLink libraries writeInt() function. The example below will set pins 0 to 5 as outputs and pins 6 to 11 as inputs.

```
#include "mLink.h"

mLink mLink;

#define I2C_ADD 0x50

void setup()
{
  mLink.init();

  // Set pins 0 to 5 as outputs and bits 6 to 11 as inputs
  unsigned int dir = 0b111111000000;

  mLink.writeInt(I2C_ADD, DIO12_DIR, dir);
}


void loop()
{
}
```

Reading input pins

Reading a single input pin

To read the state of a single bit you can use the libraries readBit() function, just specify the module's I2C address, and the input pin to read (DIO12_IN_PIN0 to DIO12_IN_PIN11). The following example will read the state of input pin 0 and output the result to the serial port.

```
#include "mLink.h"

mLink mLink;

#define I2C_ADD 0x50

void setup()
{
  mLink.init();
  Serial.begin(115200);

  // Set pin 0 to an input
  mLink.writeBit(I2C_ADD, DIO12_D0_INPUT);
}


void loop()
{
  // Read the state of pin 0
```

```
  boolean val = mLink.readBit(I2C_ADD, DIO12_D1);

  Serial.println(val);
  delay(100);
}
```

Reading all inputs at once

To read all input pins at once you can use the libraries readInt() function. The following example will read all 12 pins at once and output the result to the serial port.

```
#include "mLink.h"

mLink mLink;

#define I2C_ADD 0x50

unsigned int val;

void setup()
{
  mLink.init();
  Serial.begin(115200);

  // Set all pins to inputs
  val = 0b111111111111;
  mLink.writeInt(I2C_ADD, DIO12_DIR, val);
}


void loop()
{
  // Read all 12 inputs and save to an integer
  val = mLink.readInt(I2C_ADD, DIO12_DATA);

  Serial.println(val, BIN);
  delay(100);
}
```

Controlling output pins

Controlling a single output pin

To set the state of an output pin you can use the libraries writeBit() function, just specify the modules I2C address, the output pin to write to (DIO12_OUT_PIN0 to DIO12_OUT_PIN11), and the state to set the pin to. The following example will ‘blink’ pin 0.

```
#include "mLink.h"

mLink mLink;

#define I2C_ADD 0x50

void setup()
{
  mLink.init();
  Serial.begin(115200);

  // Set pin 0 to an output
  mLink.writeBit(I2C_ADD, DIO12_D0_OUTPUT);
}

void loop()
{
  // 'Blink' pin 0
  mLink.writeBit(I2C_ADD, DIO12_D0, HIGH);
  delay(1000);
  mLink.writeBit(I2C_ADD, DIO12_D0, LOW);
  delay(1000);
}
```

Writing to all output pins at once

To set the state of all output pins at once you can use the libraries writeInt() function. The following example will 'blink' all 12 pins at once.

```
#include "mLink.h"

mLink mLink;

#define I2C_ADD 0x50

unsigned int val;

void setup()
{
  mLink.init();

  // Set all pins to outputs
  val = 0b000000000000;
  mLink.writeInt(I2C_ADD, DIO12_DIR, val);
}


void loop()
{
  // Set all pins high
  val = 0b111111111111;
  mLink.writeInt(I2C_ADD, DIO12_DATA, val);
  delay(1000);

  // Set all pins low
  val = 0b000000000000;
```

```
  mLink.writeInt(I2C_ADD, DIO12_DATA, val);
  delay(1000);
}
```

Changing the I2C address

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

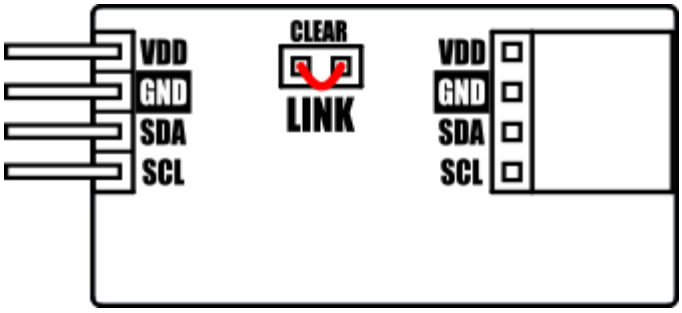
The following example changes the module's I2C address from the default 0x50 to 0x51. The new address will automatically be saved into non-volatile memory and so will retain the new address even after power has been removed from the module.

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(0x50, MLINK_ADD_REG, 0x55);
  mLink.write(0x50, MLINK_ADD_REG, 0xAA);

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

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 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.

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