Omega_Arduino_Expansion

Version 1.0.1 – 11 August 2016
Kit Bishop

Document History		
Version	**Date**	**Change Details**
Version 1.0.0	19 July 2016	First released version
Version 1.0.1	11 August 2016	• Added Vin track on board • Track layout improvements • Allow for usage of TSR 1-2540 regulator as alternative to L7805

Contents

1. Overview

Omega_Arduino_Expansion is an Omega expansion board that provides access to Arduino Shields.

The expansion board plugs in to the Omega Dock. It has the form factor and pin configuration of Arduino shields.

A brief summary of features:

- connects Omega GPIO pins to Arduino Digital pins
- has an on board A/D converter accessible from the Omega using I2C
- power supplied either from the Omega or via the 5v Arduino pin or from an on board power regulator to provide power from external supplies
- has power indicator
- has reset button

Some basic C++ code is provided for I2C access to the on board A/D converter.

The rationale for producing this expansion is to provide Omega access to Arduino shields via Omega GPIO pins while still retaining the function of the Omega GPIO pins and providing analog input functionality to the Omega.

2. Files Supplied

Omega_Arduino_Expansion information is supplied in files in a GitHub repository at https://github.com/KitBishop/Omega-Arduino-Expansion. This repository contains the following important directories and files:

- **Omega-Arduino-Expansion.pdf** – this documentation as a PDF file
- **Omega Arduino Expansion-v1.3.123** – a Sunstone Circuits PCB123 file containing the board circuit design and layout (see http://www.sunstone.com/pcb-products/pcb123 for information on the PCB123 program).
- **code** – directory containing the sample C++ code for access to the A/D converter

3. Expansion Board Details

The Omega_Arduino_Expansion board was created by Sunstone Circuits using the design provided in the PCB123 file. This section describes:

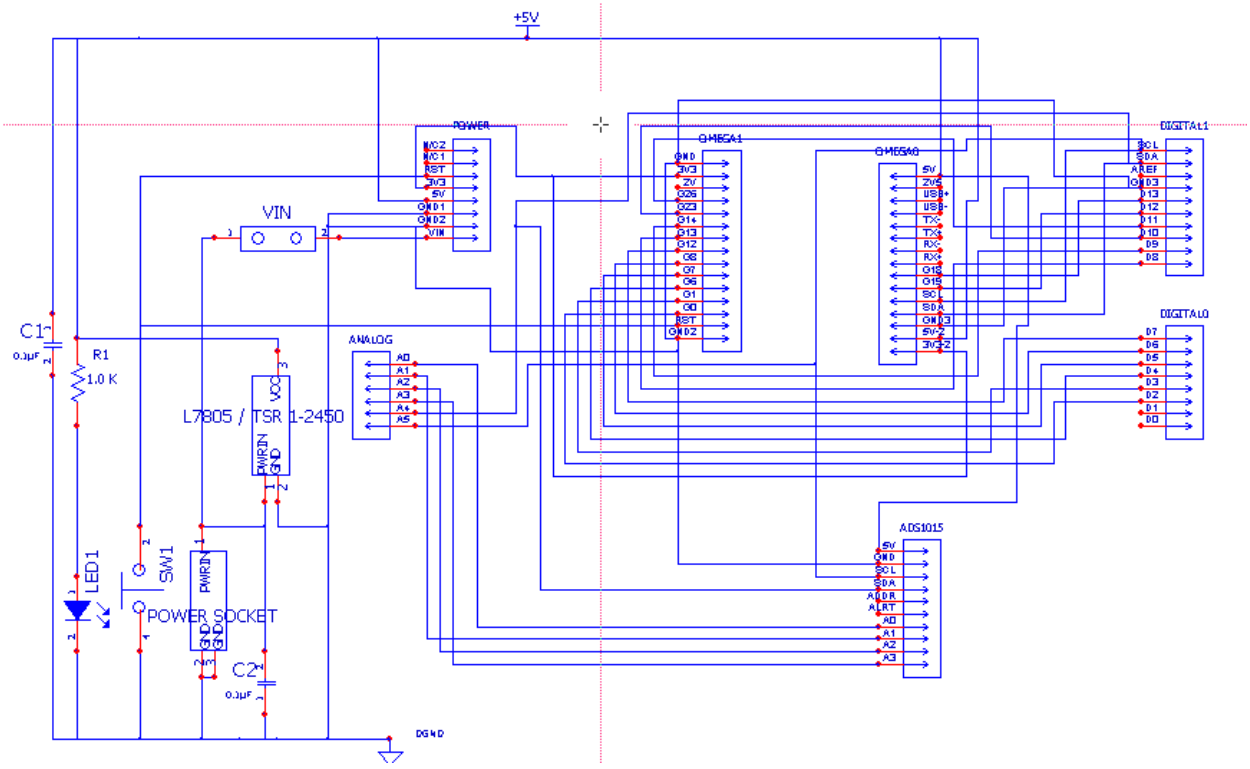
- The circuit design
- The board layout
- Component List
- Details of power supply
- Some photographs of the board and its usage

3.1. Circuit

The circuit design primarily connects the Omega GPIO pins to Arduino Shield pins (details of the pin mappings are given below). Additionally, the board provides:

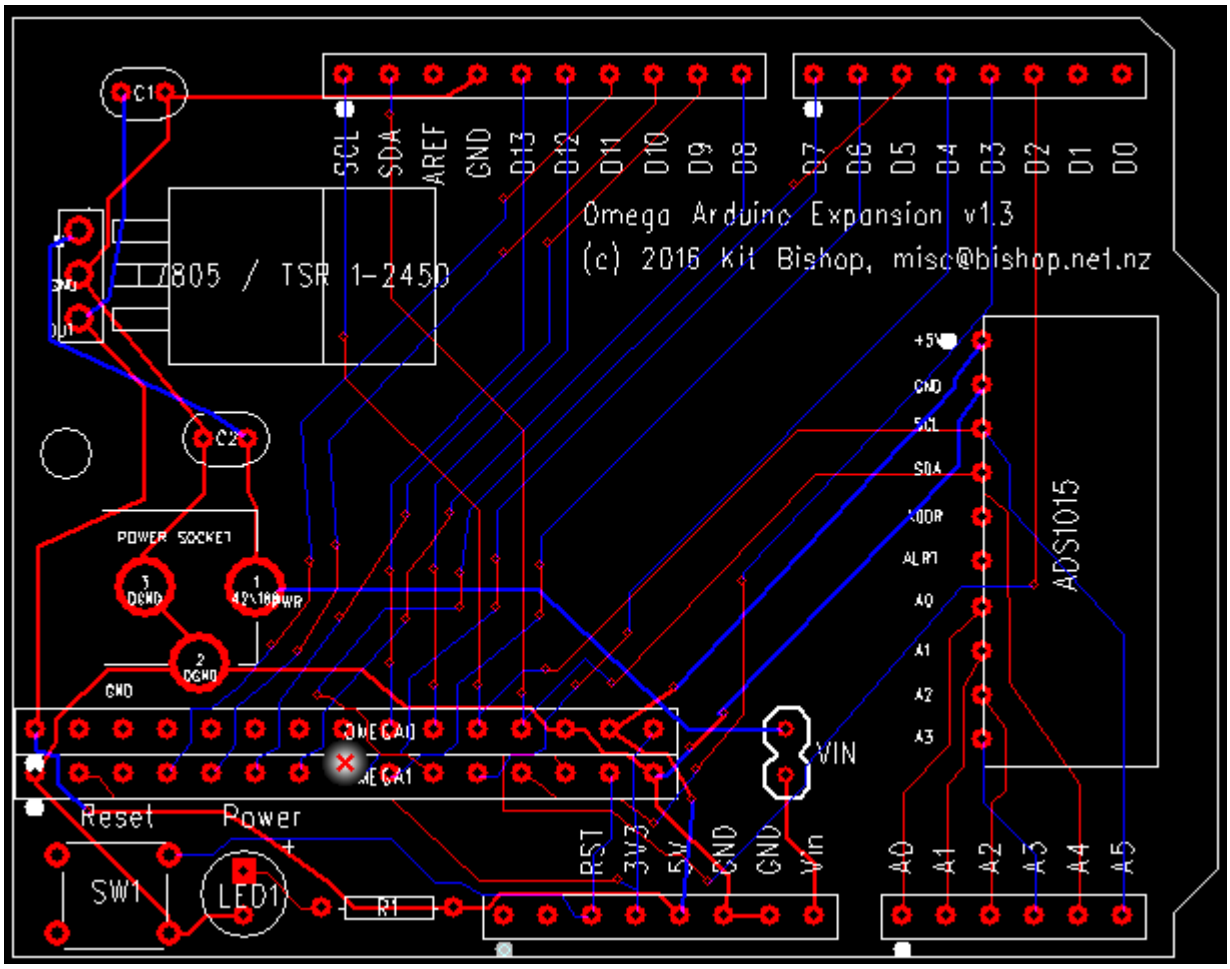
- A/D converter for I2C access from the Omega
- Power handling
- Power status indicator
- Reset button
- Jumper link for connection of Arduino Vin pin to external power supply

The circuit is shown in the following diagram (however note that the circuit can be better viewed using the PCB123 program on the supplied .123 file):



3.2. Board Layout

The board layout is shown in the following diagram (however note that the layout can be better viewed using the PCB123 program on the supplied .123 file):



3.3. Components

The components comprising the expansion are described in the following table:

Item	**Description**
Circuit Board	Circuit board as produced by Sunstone Circuits from the .123 file
Stacking Headers	Set of Arduino stacking headers as follows: • 8 pin header – POWER • 6 pin header – ANALOG • 8 pin header – DIGITAL0 • 10 pin header – DIGITAL1
Omega Connector	2 x 15 connector strip, 0.1in spacing – for connection to Omega Dock
A/D Connector	1 x 8 connector strip, 0.1in spacing – for mounting of A/D converter board
A/D Converter	Analog to Digital converter board – ADS1015 – available from https://www.adafruit.com/product/1083

Item	Description
	Alternatively the ADS1115 can be used – available from https://www.adafruit.com/products/1085
Power Regulator	Provides regulated 5v power to system from the Power Socket – supply from the socket should be greater than 7v and not more than 35v Either: • L7805 power regulator in TO220 package – e.g. https://www.sparkfun.com/products/107. Or: • TSR 1-2540 power regulator in TO220 package – e.g. https://www.adafruit.com/products/1065
Power Socket	A power barrel jack that provides power to the power regulator – e.g. https://www.sparkfun.com/products/119
Vin Jumper	2 pin jumper used to connect Arduino Power VIn pin to Power Socket and Power Regulator
Reset Button	Normally open button switch used to perform reset – e.g. http://www.digikey.com/product-detail/en/omron-electronics-inc-emc-div/B3F-1000/SW400-ND/33150
LED1	Power status LED – standard 3mm LED
Resistor R1	1K Ohm resistor – current limiting for LED
Capacitors C1 and C2	Power smoothing capacitors – 0.1 µF

3.4. Pin Mapping

The tables below show the mapping to/from the Omega Dock pins and the ArduinoShield pins

3.4.1. Omega Dock Pins

This table shows the pin connections listed by Omega Dock pin. The layout of the table corresponds to the Omega Dock connector as seen from the top.

Left Omega Dock Pins	Arduino Shield Pin	Right Omega Dock Pins	Arduino Shield Pin
Gnd	POWER – Gnd	5v	POWER – 5v
3v3	POWER – 3v3	2v5	N/C
2v	N/C	USB+	N/C
GPIO 26	DIGITAL1 – D11	USB-	N/C
GPIO 23	DIGITAL1 – D10	TX-	N/C
GPIO 14	DIGITAL1 – D9	TX+	N/C
GPIO 13	DIGITAL1 – D8	RX-	N/C
GPIO 12	DIGITAL0 – D7	RX+	N/C
GPIO 8	DIGITAL0 – D6	GPIO 18	DIGITAL1 – D13
GPIO 7	DIGITAL0 – D5	GPIO 19	DIGITAL1 – D12
GPIO 6	DIGITAL0 – D4	GPIO 20 / I2C SCL	ANALOG – A5 / DIGITAL1 – SCL
GPIO 1	DIGITAL0 – D3	GPIO 19 / I2C SDA	ANALOG – A4 / DIGITAL1 – SDA
GPIO 0	DIGITAL0 – D2	Gnd	POWER – Gnd
RST	POWER – RST	5v	POWER – 5v
Gnd	POWER – Gnd	3v3	POWER – 3v3

3.4.2. Arduino Shield Pins

This table shows the pin connections listed by Arduino Shield pin. The layout of the table corresponds to the Arduino Shield connectors as seen from the top.

Arduino Shield Pin	Connected To	Arduino Shield Pin	Connected To
		DIGITAL1 – SCL	Omega Dock – GPIO 20 / SCL Arduino ANALOG – A5
		DIGITAL1 – SDA	Omega Dock – GPIO 19 / SDA Arduino ANALOG – A4
POWER – N/C	N/C	DIGITAL1 – AREF	N/C
POWER – IOREF	N/C	DIGITAL1 – GND	Omega Dock – Gnd Arduino POWER – GND
POWER – RST	Omega Dock – RST Reset button	DIGITAL1 – D13	Omega Dock – GPIO 18
POWER – 3v3	Omega Dock – 3v3	DIGITAL1 – D12	Omega Dock – GPIO 19
POWER – 5v	Omega Dock – 5v Also: • Powered from power regulator • Powers LED	DIGITAL1 – D11	Omega Dock – GPIO 26
POWER – GND	Omega Dock – Gnd	DIGITAL1 – D10	Omega Dock – GPIO 23
POWER – GND	Omega Dock – Gnd	DIGITAL1 – D9	Omega Dock – GPIO 14
POWER – VIN	External power socket via Vin jumper	DIGITAL1 – D8	Omega Dock – GPIO 13
		DIGITAL0 – D7	Omega Dock – GPIO 12
		DIGITAL0 – D6	Omega Dock – GPIO 8
ANALOG – A0	A/D ADS1015 – A0	DIGITAL0 – D5	Omega Dock – GPIO 7
ANALOG – A1	A/D ADS1015 – A1	DIGITAL0 – D4	Omega Dock – GPIO 6
ANALOG – A2	A/D ADS1015 – A2	DIGITAL0 – D3	Omega Dock – GPIO 1
ANALOG – A3	A/D ADS1015 – A3	DIGITAL0 – D2	Omega Dock – GPIO 0
ANALOG – A4	Omega Dock – GPIO 19 / SDA A/D ADS1015 – SDA	DIGITAL0 – D1	N/C
ANALOG – A5	Omega Dock – GPIO 20 / SCL A/D ADS1015 – SCL	DIGITAL0 – D0	N/C

3.5. Power Supply

The power can be supplied to the Omega and the expansion board in one of four ways:

3.5.1. Power from Omega

In its simplest form, power can be supplied via the Omega USB connector. This provides power to the expansion as follows:

- 5v – the 5v supply from the USB connector is made available on the Arduino 5v pin
- 3.3v – the 3.3v supply generated by the Omega is made available on the Arduino 3.3v pin

3.5.2. Power on Arduino 5V Pin

Power can be supplied by applying 5v to the Arduino 5v pin. In this case, power is distributed as follows:

- 5v – the Arduino 5v pin supplies the power to the Omega via the 5v pin on the Omega Dock connector
- 3.3v – the 3.3v supply generated by the Omega is made available on the Arduino 3.3v pin

An example of providing power this way is via the usage of power supplied by an Arduino lithium battery shield (see: https://learn.adafruit.com/adafruit-powerboost-500-shield-rechargeable-battery-pack/overview) – see photograph below.

3.5.3. External Power

Power can be supplied by providing power from an external source of between 7v and 35v to the power connector on the expansion. In this case, power is distributed as follows:

- 5v – the Power Regulator supplies 5v to the Arduino 5v pin which the power to the Omega via the 5v pin on the Omega Dock connector
- 3.3v – the 3.3v supply generated by the Omega is made available on the Arduino 3.3v pin

3.5.4. Arduino VIn Pin

Power can be supplied by providing power to the Arduino Power VIn pin and inserting a VIn jumper. This connects the VIn pin to the power regulator. If this method of providing power is used, the External Power connector should not be used. In this case, as for External Power, power is distributed as follows:

- 5v – the Power Regulator supplies 5v to the Arduino 5v pin which the power to the Omega via the 5v pin on the Omega Dock connector
- 3.3v – the 3.3v supply generated by the Omega is made available on the Arduino 3.3v pin

3.6. VIn Jumper

The VIn Jumper, when inserted, connects the Arduino Power VIn pin to the External Power input.

This can be used in one of two ways:

When no power is supplied to the External Power connector, any power supplied on VIn is fed to the Power regulator to provide system power.

When power is supplied to the External Power connector, the external power is also supplied to the VIn pin for use by any Arduino Shields that need it.

3.7. Photographs

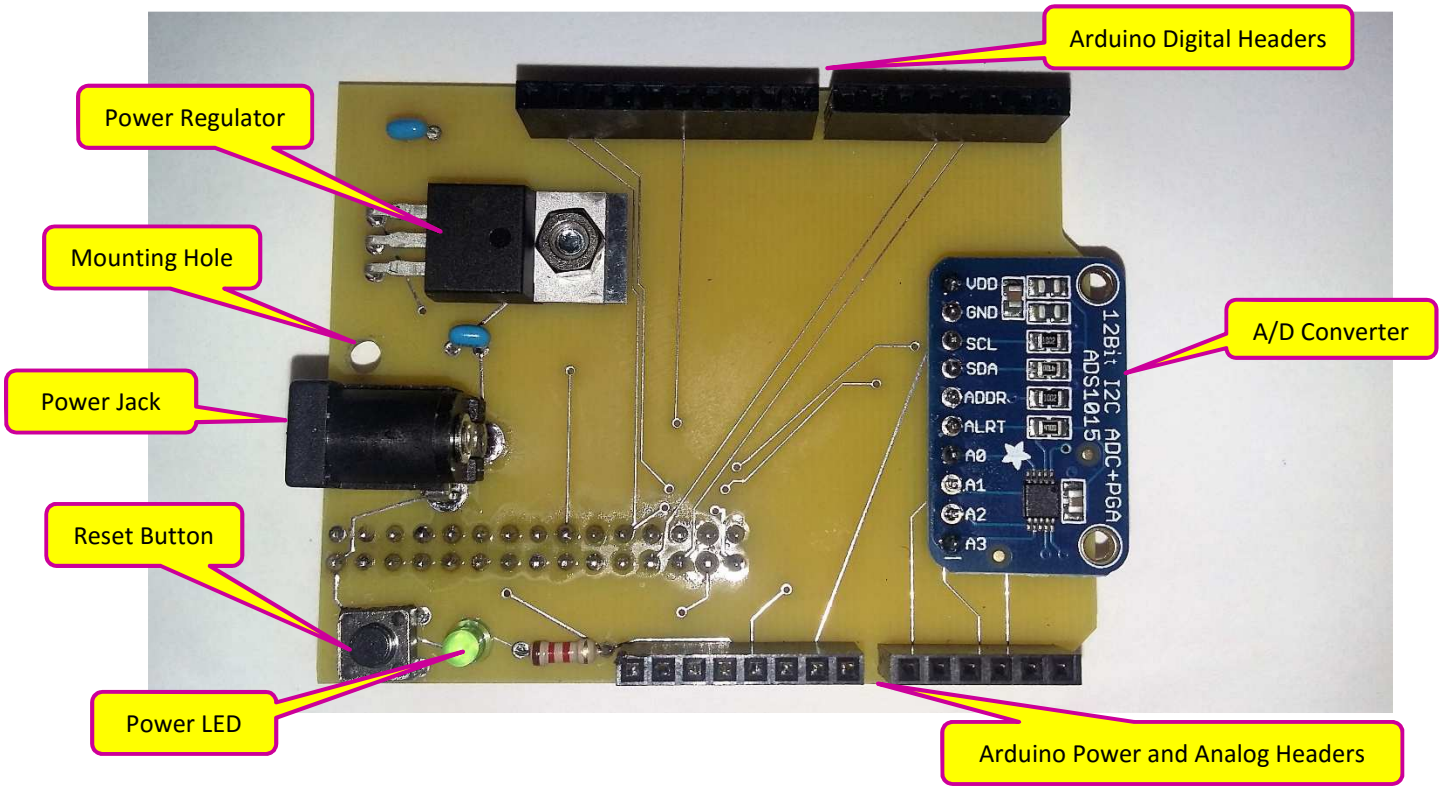
This section contains some photographs of the board and some examples of its usage.

Note that in the images of the expansion board, there is no silk screening on the board. Sunstone Circuits can produce the board with silk screening using the supplied .123 file but this substantially increases the cost per board when small quantities are produced.

3.7.1. Board Top

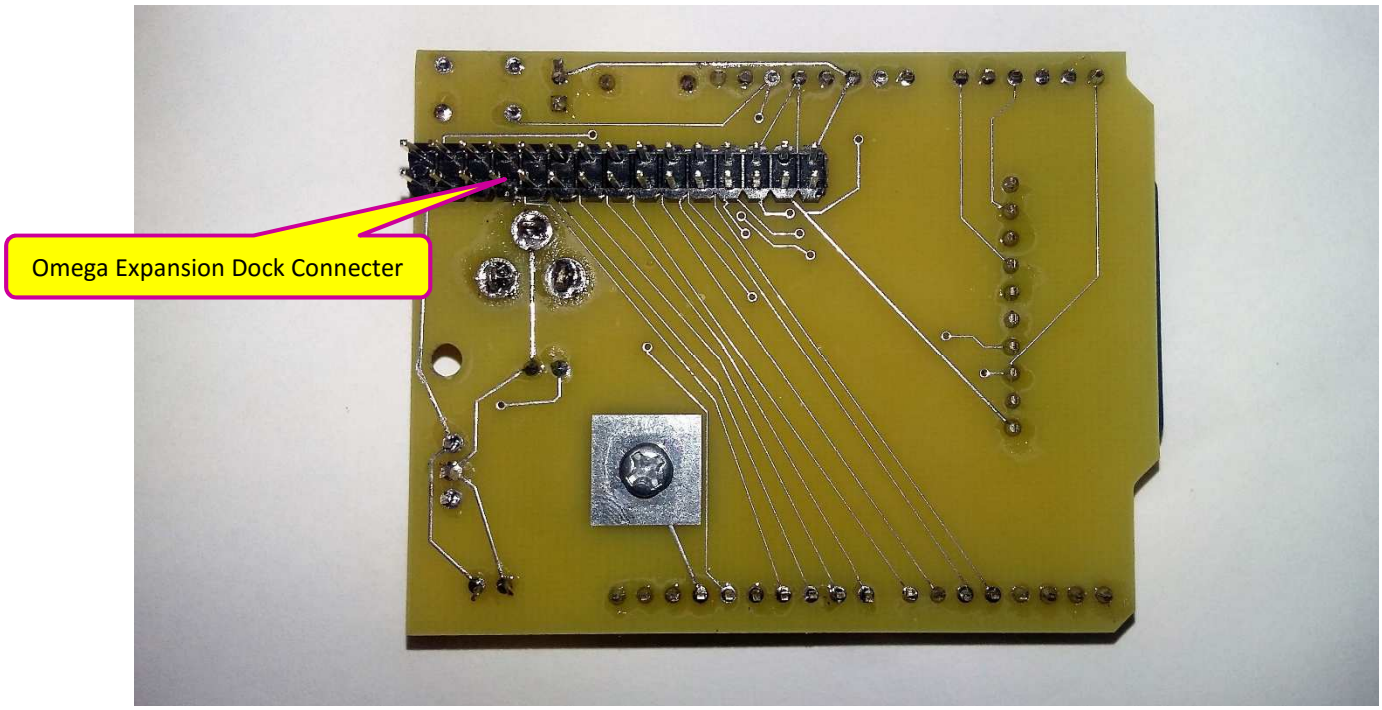
This photograph shows the top of the expansion board with all components mounted.

Note that the hole between the Power Socket and the Power Regulator matches the corresponding hole on the Omega Dock and can be used for mechanically securing the expansion to the dock.



3.7.2. Board Bottom

This photograph shows the bottom of the expansion board and shows the connector that plugs in to the Omega Dock.

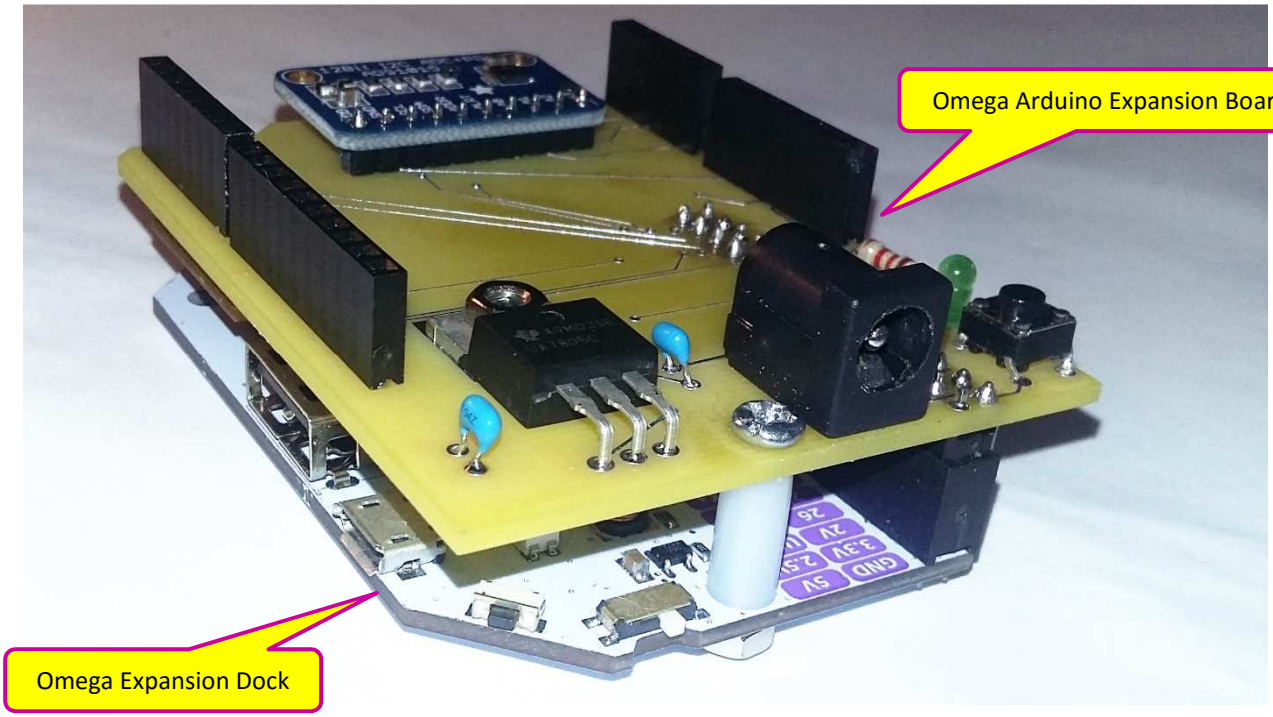


3.7.3. Usage

These photographs show various combinations of using the expansion board including with various Arduino Shields.

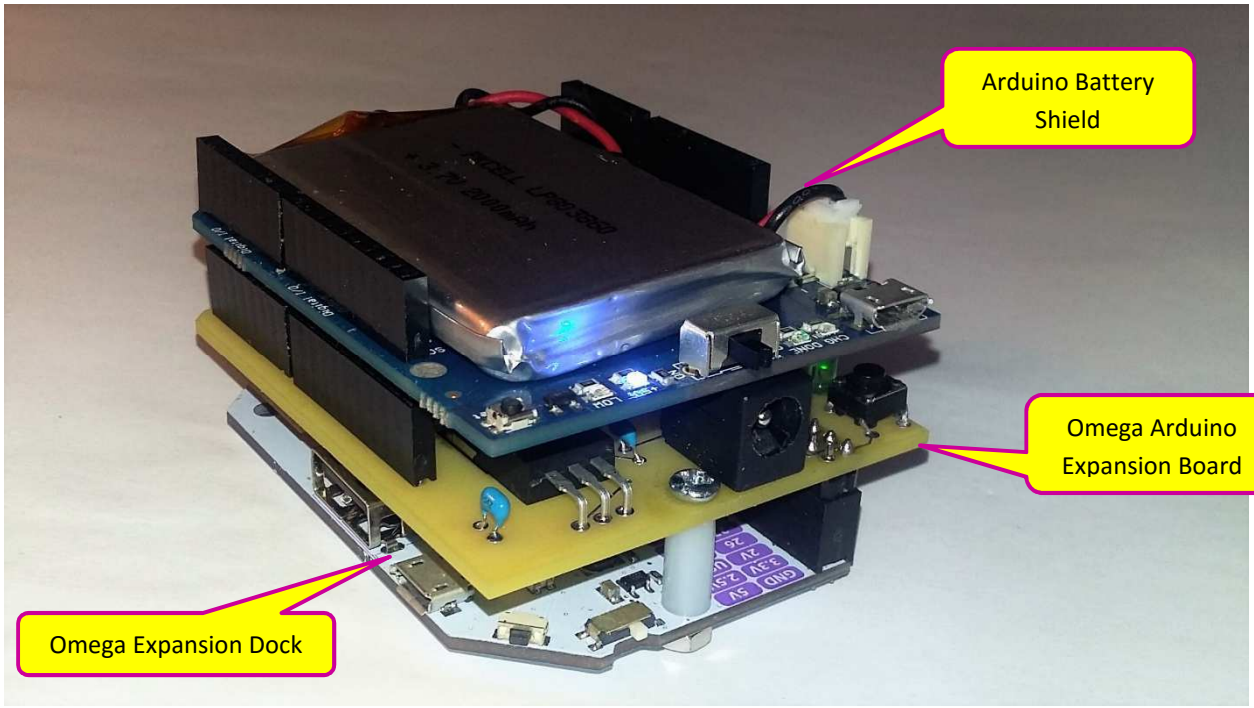
3.7.3.1. Omega with Board

This photograph show the expansion board plugged in to the Omega Dock. It also shows usage of mechanically securing the board to the dock.



3.7.3.2. Omega with Board and Arduino Battery Shield

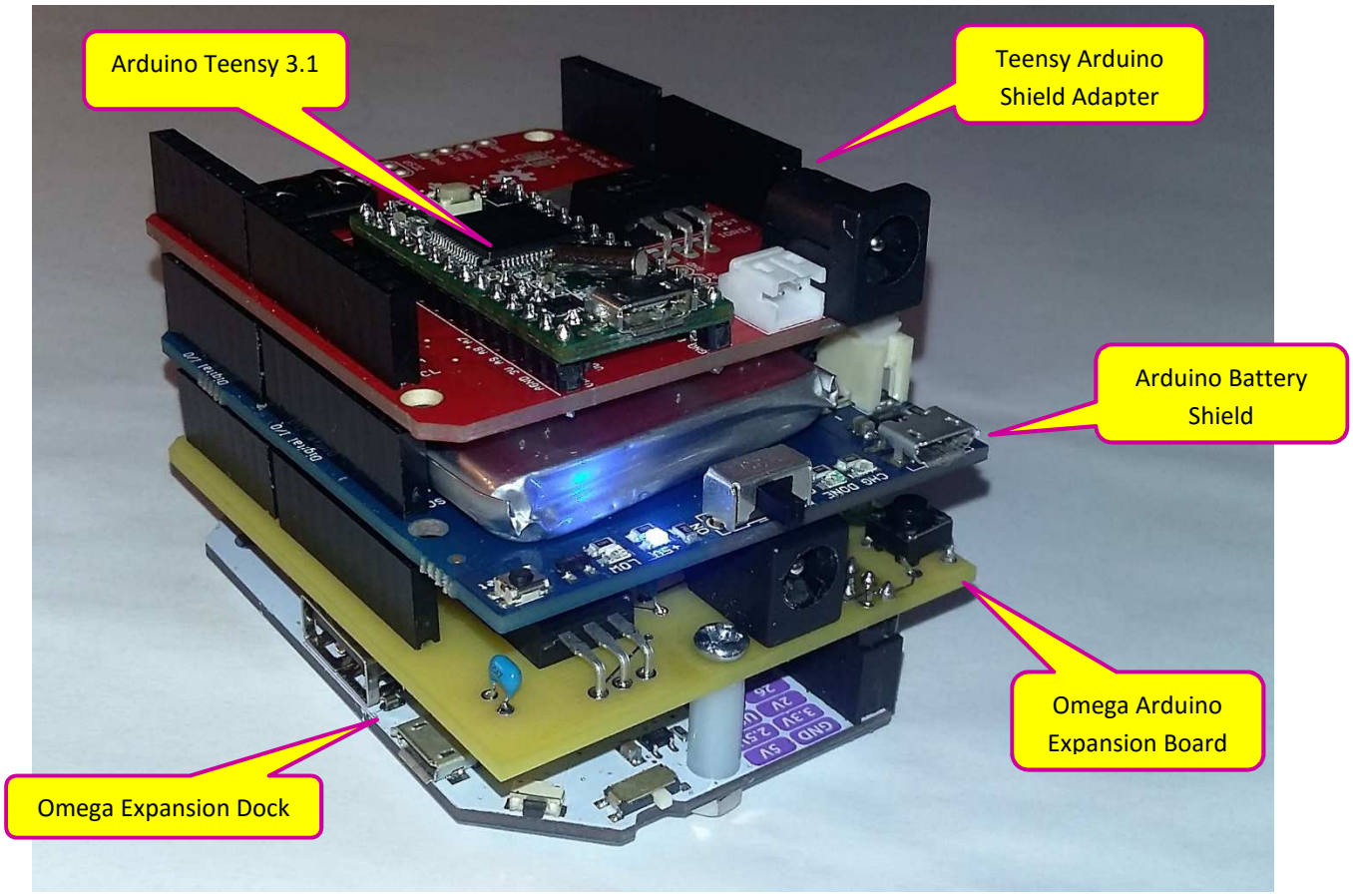
This photograph shows the expansion board connected to the dock with power supplied by an Arduino lithium battery shield (see: https://learn.adafruit.com/adafruit-powerboost-500-shield-rechargeable-battery-pack/overview)



3.7.3.3. Omega with Board and an Arduino System

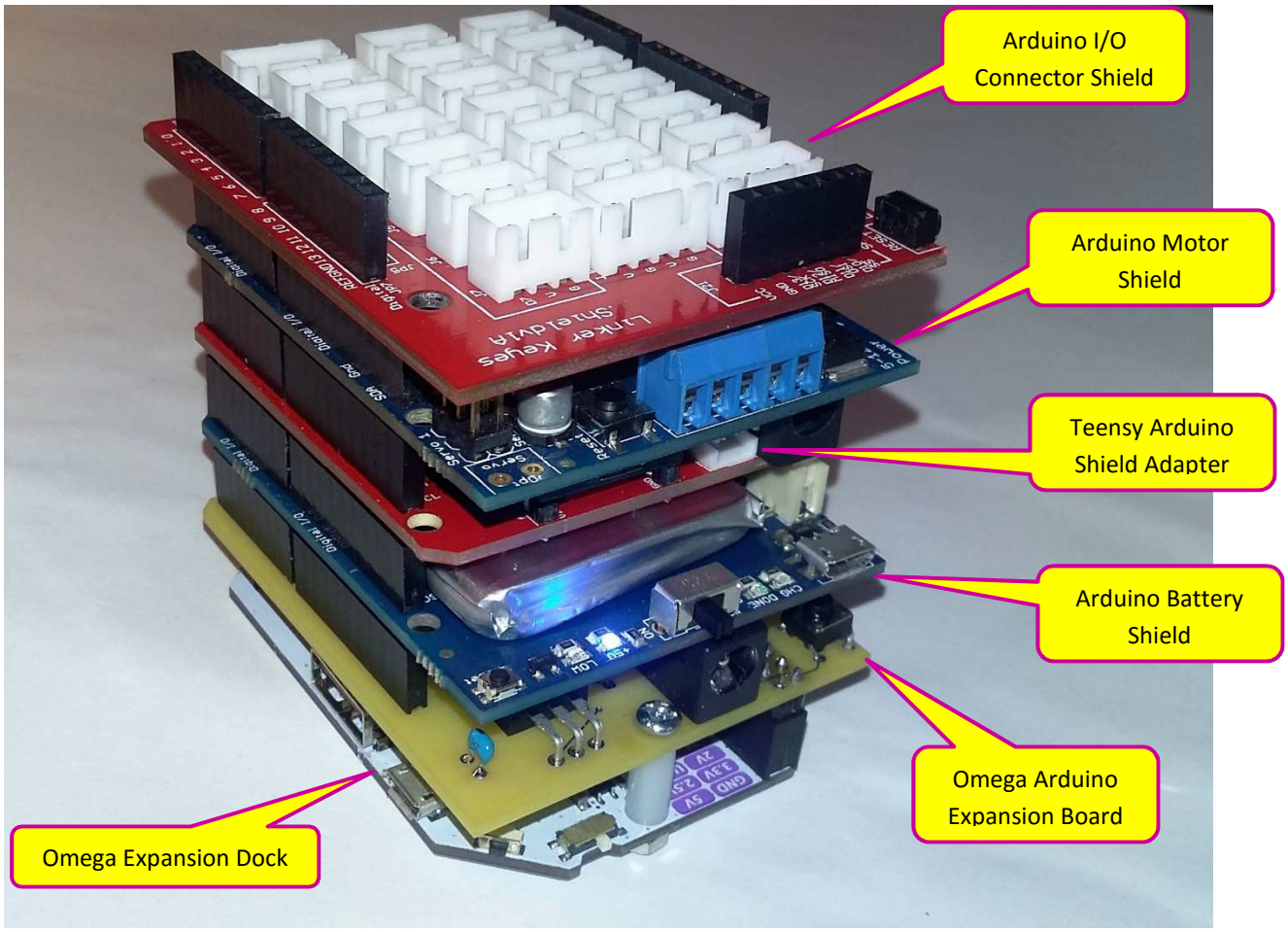
This photograph shows an Arduino System added. In this case it is an Arduino Teensy 3.1 on a Teensy Arduino Shield Adapter (see: https://www.sparkfun.com/products/13288). The Omega and the Arduino can communicate using I2C using code that is available at https://github.com/KitBishop/Omega-GPIO-I2C-Arduino/tree/master/libarduino and https://github.com/KitBishop/Omega-GPIO-I2C-Arduino/tree/master/arduino_omega.

Both the Omega and the Arduino can access the IO pins, though you should take care to set the input and output states of the pins appropriately for usage by each system to avoid possible conflicts.



3.7.3.4. Omega with a full stack of Arduino Shields

This photograph shows the addition of an Arduino I2C controlled motor shield and an Arduino IO connector shield.



4. Code for access to A/D

Some C++ code is provided for access to the A/Dconverter on the expansion board.

The files for this can be found at:

- **code/hdr/OAExpAnalog.h** – the header file for access
- **code/src/OAexpAnalog.cpp** – the source file for access

This code is documented in sections below.

This code makes use of the **libnewi2c** library which can be found at
https://github.com/KitBishop/Omega-GPIO-I2C-Arduino/tree/master/libnewi2c

4.1. OAExpAnalog Types

The file **OAExpAnalog.h** contains definitions of some basic types used in the code.

4.1.1. Define

The following #define item is provided for convenience:

- **#define DEFAULT_OAEXPANALOG_DEV_ADDR 0x48**
  Defines the default I2C device address for the A/D converter

4.1.2. enum OAExpAnalog_DeviceType

enum OAExpAnalog_DeviceType is used to represent the type of the A/D converter on the expansion board. It has values:

- **OAEXPANALOG_DEVICE_TYPE_1015** – represents an ADS1015 A/D converter
- **OAEXPANALOG_DEVICE_TYPE_1115** – represents an ADS1115 A/D converter

4.1.3. enum OAExpAnalog_DifferentialChanel

enum OAExpAnalog_DifferentialChanel is used to represent the A/D converter channels to use when performing a differential A/D read. It has values:

- **OAEXPANALOG_DIFFERENTIAL_CHANNEL_0_1** – read from channels 0 and 1
- **OAEXPANALOG_DIFFERENTIAL_CHANNEL_2_3** – read from channels 2 and 3

4.1.4. enum OAExpAnalog_Range

enum OAExpAnalog_Range is used to represent the maximum range to be used by the A/D converter when performing a read. Any input value greater than the selected range will return the maximum for that range. It has values:

- **OAEXPANALOG_RANGE_6_144V** – maximum range of 6.144 volts
- **OAEXPANALOG_RANGE_4_096V** – maximum range of 4.096 volts
- **OAEXPANALOG_RANGE_2_048V** – maximum range of 2.048 volts
- **OAEXPANALOG_RANGE_1_024V** – maximum range of 1.024 volts
- **OAEXPANALOG_RANGE_0_512V** – maximum range of 0.512 volts

- **OAEXPANALOG_RANGE_0_256V** – maximum range of 0.256 volts

4.2. Class OAExpAnalog

An object of type the **OAExpAnalog** class is used to provide access to the expansion A/D converter.

4.2.1. OAExpAnalog Constructors

Note that an **OAExpAnalog** instance uses an **I2CDevice** for the I2C access.

4.2.1.1. Constructor - OAExpAnalog(OAExpAnalog_DeviceType type = OAEXPANALOG_DEVICE_TYPE_1015);

Creates a new OAExpAnalog instance for the given device type with an I2CDevice with the default I2C address given by DEFAULT_OAEXPANALOG_DEV_ADDR.

Parameters:

- **OAExpAnalog_DeviceType type** – specifies the type of A/D converter used on the board. Optional, defaults to OAEXPANALOG_DEVICE_TYPE_1015 if not supplied.

4.2.1.2. Constructor - OAExpAnalog(byte devAddr, OAExpAnalog_DeviceType type = OAEXPANALOG_DEVICE_TYPE_1015);

Creates a new OAExpAnalog instance for the given device type with an I2CDevice with the given I2C address.

Parameters:

- **byte devAddr** – specifies the I2C device address to be used to access the converter.
- **OAExpAnalog_DeviceType type** – specifies the type of A/D converter used on the board. Optional, defaults to OAEXPANALOG_DEVICE_TYPE_1015 if not supplied.

*4.2.1.3. Constructor - OAExpAnalog(I2CDevice * i2cDev, OAExpAnalog_DeviceType type = OAEXPANALOG_DEVICE_TYPE_1015);*

Creates a new OAExpAnalog instance for the given device type with a given I2CDevice.

Parameters:

- **I2CDevice * i2cDev** – references the I2C device to be used to access the converter.
- **OAExpAnalog_DeviceType type** – specifies the type of A/D converter used on the board. Optional, defaults to OAEXPANALOG_DEVICE_TYPE_1015 if not supplied.

4.2.2. OAExpAnalog Public Methods

4.2.2.1. bool begin();

Initialises the A/D converter. Must be called after constructing the instance and before performing any read operations.

Parameters:

- <none>

Returns:

- **true** if result is OK, **false** on any error

4.2.2.2. bool read(float & val, unsigned char channel, OAExpAnalog_Range range = OAEXPANALOG_RANGE_6_144V);

Reads and returns the single ended analog value on a given channel.

Parameters:

- **float & val** – returns the value of the channel in volts
- **unsigned char channel** – specifies the channel to read from. Must be >= 0 and <= 3
- **OAExpAnalog_Range range = OAEXPANALOG_RANGE_6_144V** – specifies the maximum range to be used to perform the read. Any input value greater than this will return the specified maximum. Optional, default specifies a maximum of 6.144 volts.

Returns:

- **true** if result is OK, **false** on any error

4.2.2.3. bool readDifferential(float & val, OAExpAnalog_DifferentialChanel difChan, OAExpAnalog_Range range = OAEXPANALOG_RANGE_6_144V);

Reads and returns the differential analog value on the given pair of channels.

Parameters:

- **float & val** – returns the value of the channel in volts
- **OAExpAnalog_DifferentialChanel difChan** – specifies the pair of channels to read from.
- **OAExpAnalog_Range range = OAEXPANALOG_RANGE_6_144V** – specifies the maximum range to be used to perform the read. Any input value greater than this will return the specified maximum. Optional, default specifies a maximum of 6.144 volts.

Returns:

- **true** if result is OK, **false** on any error