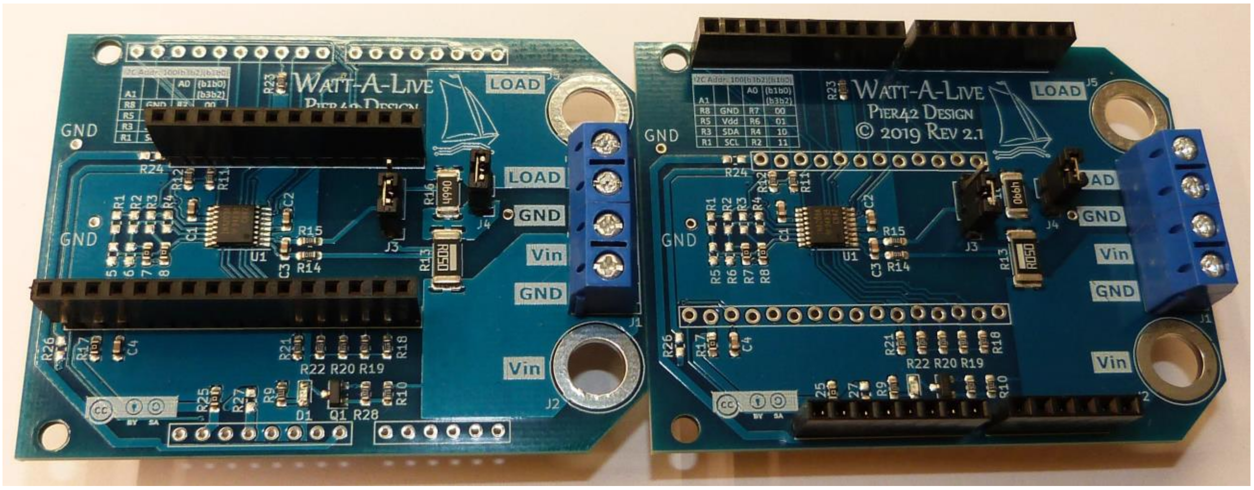


P42 Watt-A-Live Power Monitor
Arduino Shield Feather Wing
Rev2.1

Designed by Pier42 Electronics Design

Wolfgang Friedrich

https://www.tindie.com/stores/pier42/

https://hackaday.io/project/166326-watt-a-live-power-monitor-shield-wing

https://github.com/wolfgangfriedrich/P42-Watt-A-Live

Table of Contents

Introduction

The Watt-A-Live is a versatile embedded power monitor based on the Texas Instruments INA209 with full connectivity to the unit under test. It measures the load current through a shunt resistor and has separate GND connections to the power supply and the load to make it a true power monitor.

It is designed as an Adafruit Feather Wing and Arduino Shield. Alternatively it can be used as a breakout board with any other controller that has I2C and optional GPIO connectivity. 2 jumpers can select between 2 different shunts to measure different current ranges.

The board is populated with a 0.05 Ω 1% shunt to measure 6.4A to 200µA. Optionally a 499 Ω 1% shunt can be added for a current range of 640uA to 20nA. Maximum bus voltage is 26V.

Full feature set:

- Adafruit Feather Wing and Arduino Uno Shield connector option
- Current monitor with 2 different ranges for high current and sleep mode measurements
- Positive and negative current flow
- Bus voltage monitor with 2 dedicated GND terminals
- Communication interface: I2C up to 3.4MHz
- 16 I2C addresses selectable through resistor options
- 6 dedicated signaling pins (SMBus Alert, Warning, Overlimit, Critical, Convert and 1 GPIO)
- Operating supply voltage 3.0V to 5.5V
- Optional 4mm Banana plugs for high current measurement or to bypass the shunt.
- Size: 71mm x 54mm (2.8" x 2.1")
- LED showing voltage over 2V on Vin
- The rev2 Arduino library also works together with the rev1 board which has a 0.1 Ω shunt mounted for the high current range and 500 Ω for the low current range.

Hardware

The Watt-A-Live Power Monitor Wing/Shield is designed for the Adafruit Feather and Arduino Uno, but also runs on the Mega and the Due. It can also be used as a breakout board or on a breadboard (thanks to the normal Wing connector spacing.

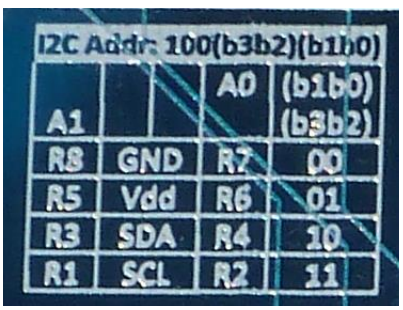
I2C address select

If more than one device is used, the I2C address on every board must be set to unique value. 16 addresses are available for the chip and fully supported by this board.

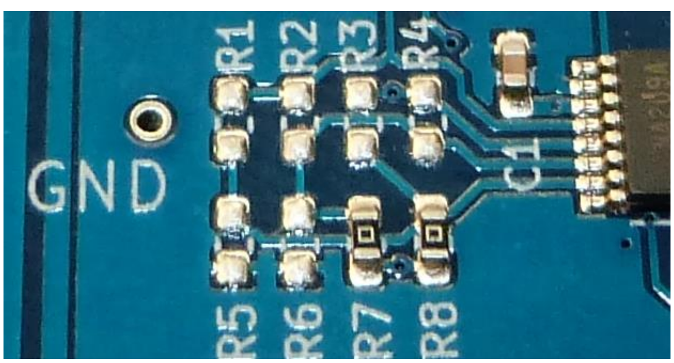
Here is a table showing all address options. The 7-bit address is used as parameter in the configuration function P42_INA209 ().

A1	A0	7bit addr		8bit addr		Populate Resistor	
GND	GND	0b100.0000	0x40	0x80	+R/W	R8	R7
GND	Vdd	0b100.0001	0x41	0x82	+R/W	R8	R6
GND	SDA	0b100.0010	0x42	0x84	+R/W	R8	R4
GND	SCL	0b100.0011	0x43	0x86	+R/W	R8	R2
Vdd	GND	0b100.0100	0x44	0x88	+R/W	R5	R7
Vdd	Vdd	0b100.0101	0x45	0x8A	+R/W	R5	R6
Vdd	SDA	0b100.0110	0x46	0x8C	+R/W	R5	R4
Vdd	SCL	0b100.0111	0x47	0x8E	+R/W	R5	R2
SDA	GND	0b100.1000	0x48	0x90	+R/W	R3	R7
SDA	Vdd	0b100.1001	0x49	0x92	+R/W	R3	R6
SDA	SDA	0b100.1010	0x4A	0x94	+R/W	R3	R4
SDA	SCL	0b100.1011	0x4B	0x96	+R/W	R3	R2
SCL	GND	0b100.1100	0x4C	0x98	+R/W	R1	R7
SCL	Vdd	0b100.1101	0x4D	0x9A	+R/W	R1	R6
SCL	SDA	0b100.1110	0x4E	0x9C	+R/W	R1	R4
SCL	SCL	0b100.1111	0x4F	0x9E	+R/W	R1	R2

A handy table on the board shows the address selection in a compact form.



Resistors R1-R8 are used to select the I^{2}C slave address.



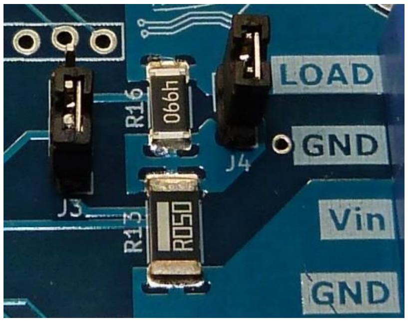
Dual current range select

The 2 current measuring ranges can be selected by 2 jumpers on the board.

The 3-pin header J3 selects the mode of operation.

1) Use of the high current range only. This gives a more accurate result because the measurement is done on the terminals of the shunt directly. Jumper in position 1-2 as shown in the picture.
2) Switchable operation between high current and low current mode. Jumper in position 2-3. Pin 3 is the open visible pin in the picture.

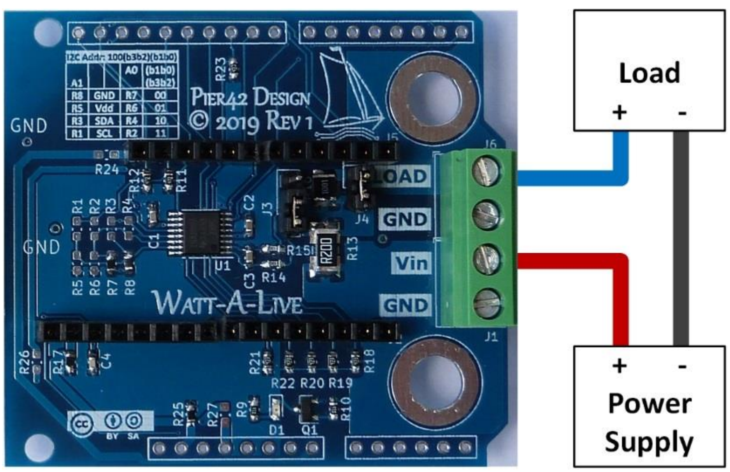
The 2-pin jumper J4 bypasses the low-current shunt when set.



Connection Diagrams

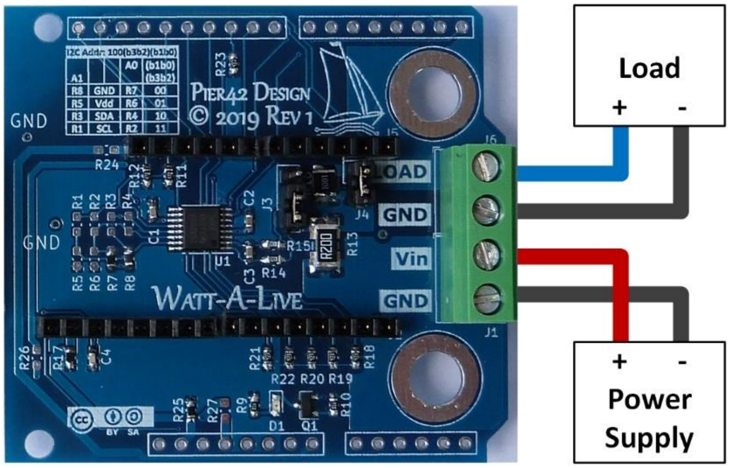
Current Measurement

If only the current to the load is of interest, a tap into the positive power wire is sufficient. Current is calculated from the voltage drop over the shunt resistor.



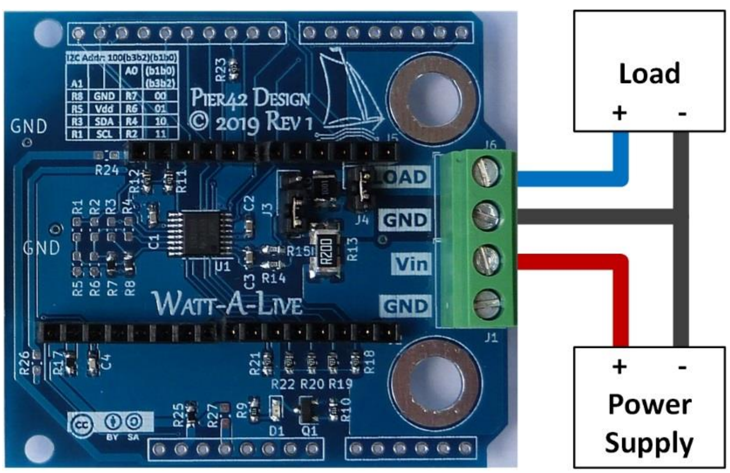
Power Measurement 1

For additional power or bus voltage measurements, GND of the target device must be connected to the sensor board. The board provides 2 GND terminals for a complete separation of source and load wiring.



Power Measurement 2

A minimum wire configuration for power or bus voltage measurements would look like this.



Software

Support for the 500Ω shunt is now available. And the warning register and pin state monitoring is highly experimental.

!!! For correct function the respective shunt value has to be enabled in the ti_ina209.h file. !!!

Depending whether the Arduino or Feather platform is being used, the ti_ina209.h file needs to be edited.

!!! For correct function the respective pin mappings have to be enabled in the ti_ina209.h file. !!!

Texas Instruments offers a tool to set-up the calibration value. Details can be found here: http://www.ti.com/lit/zip/sboc248

Constants

The following constants are provided by the library. They are useful to make the code adapting to other resolutions.

Name		Description
All register addresses		see datasheet and ti_ina209.h
All register bits		see datasheet and ti_ina209.h
CALIB_VALUE	0x4096	Calibration value if a manual calibration was performed. See Ti INA209 datasheet for details.
SHUNT_R	0.05 or 499	Shunt value that is enabled by the jumpers on the board

TI_INA209()

Board pinout and I2C address configuration. Multiple boards can be used with different class names and I2C addresses.

```
TI_INA9 (byte address);
```

Pin	Pin Nr Feather	Pin Nr Arduino	Direction	Default	Description
WARN_PIN	15	13	Out	OC with pull-up	Set when a SMBus warn condition is present.
ALERT_PIN	14	12	Out	OC with pull-up	Set when an alert condition is present.
GPIO_PIN	16	11	In/Out	In with pull-up	General purpose IO.

CONV_PIN	12	10	In	In with pull-up	Triggers a conversion.
OVER_PIN	1	2	Out	OC with pull-up	Set when an over/under condition is present.
CRIT_PIN	0	3	Out	OC with pull-up	Set when critical condition is present.

Example: setup board with I2C address 0x40

```
TI_INA209 ina209_40(0x40);          // instantiate ina209_40 of class INA209 with I2C address 0x40. Address depends on set resistors on the board.
```

readWord ()

Read a 16bit register value from the controller.

```
word readWord (byte reg_addr);
```

Value	Size	Description
Reg_addr	byte	register address
return value	word	Result of the read command: register content

Example: Read Status register

```
Result = ina209_40.readWord( STATUS_REG );
```

⇨ Result will be 0xXXXX.

writeWord ()

Read a 16bit register value from the video controller.

```
void SPIReadRegister16 (byte address, word data);
```

Value	Size	Description
address	byte	Opcode of the video controller command, also called register address
data	word	16 bit data word to write into specified register
return value	void	No return value

Example: Write Configuration register address 0x00

```
ina209_40.writeWord( CONFIG_REG, 0x3FFA );
```

⇨ No return result.

pinMode209 ()

Set the direction of the GPIO pin, with similar syntax as the Arduino digital pin commands.

```
void pinMode209 (uint8_t mode);
```

Value	Size	Description
mode	Uint8_t	Set either as INPUT or OUTPUT

Example: Set GPIO as an output.

```
ina209_40.pinMode209 ( OUTPUT );
```

digitalWrite209 ()

Set the GPIO pin to HIGH or LOW, with similar syntax as the Arduino digital pin commands. Its voltage level will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW.

```
void digitalWrite209 (uint8_t value);
```

Value	Size	Description
value	uint8_t	Set either to HIGH or LOW

Example: Set GPIO to a HIGH level.

```
ina209_40.digitalWrite209 ( HIGH );
```

digitalRead209 ()

Read the GPIO pin and return the value either HIGH or LOW, with similar syntax as the Arduino digital pin commands.

```
return digitalRead209 ( );
```

Value	Size	Description
value	uint8_t	HIGH or LOW

Example: Read GPIO pin value.

```
Return = ina209_40.digitalRead209 ( );
```

getCurrent ()

Read the current register value from the controller and returns the real world current measured in mA. With the variable shunt value, the 2 different shunt resistors on the board can be easily supported. Output value is given in mA for shunt values <= 1 Ω, output is given in µA for shunt values > 1 Ω.

```
float  getCurrent ( float shunt_f );
```

Value	Size	Description
shunt_f	float	Shunt value in Ohm.
return value	float	Measured current in mA (shunt_f <= 1) / uA (shunt_f>1).

Example: Read current value.

```
#define SHUNT_R  0.05
Serial.print( ina209_40.getCurrent( SHUNT_R ) );
```

getVoltage ()

Read the bus voltage register value from the controller and returns the real world voltage measured in V. The voltage is measured at the negative differential shunt voltage to GND, so the voltage drop across the shunt is not included in this value. The voltage is the value that the connected load is seeing as input voltage, which can be up to 320mV lower than the voltage display on the power supply.

```
float   getVoltage ( void );
```

Value	Size	Description
void	-	No parameter needs to be passed into this function
return value	float	Measured voltage in V.

Example: Read voltage value.

```
Serial.print( ina209_40.getVoltage ( void ) );
```

getPower ()

Read the power register value from the controller and returns the real world power calculated in W. With the variable shunt value, the 2 different shunt resistors on the board can be easily supported. Output value is given in W for shunt values <= 1 Ω, output is given in mW for shunt values > 1 Ω.

```
float   getPower ( float shunt_f );
```

Value	Size	Description
shunt_f	float	Shunt value in Ohm.
return value	float	Measured voltage in W(shunt_f <= 1) / mW (shunt_f>1).

Example: Read power value.

```
#define SHUNT_R  0.05
Serial.print( ina209_40.getPower ( SHUNT_R ) );
```

This is a living document. Any missing content will be added as required.

Revision Control

Version	Data	Changes
0.1	23. June 2019	Madman Chicken-Scratch Manifesto
1.0	9. Sept 2019	Updates for Board revision 2.1
1.1	15. Sept 2019	More updates and new pictures
1.2	6. Oct 2019	Updates for getCurrent, getVoltage, getPower to support 2 ranges better.