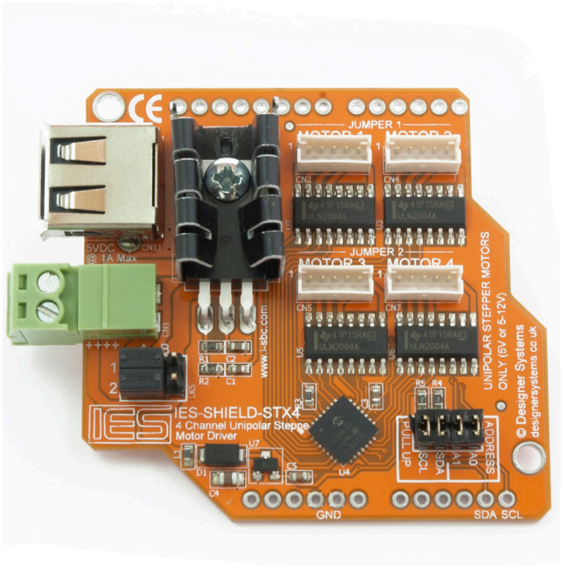
4 Channel Stepper Driver Shield

for Arduino™ and Raspberry-Pi™

Product Overview



IES-SHIELD-STX4
4 Channel Unipolar Stepper Motor Driver
JUMPER 1
JUMPER 2
MOTOR 1
MOTOR 2
MOTOR 3
MOTOR 4
5VDC @ 1A Max
UNIPOLAR STEPPER MOTORS ONLY (5V or 5-12V)
© Designer Systems designersystems.co.uk
PULL UP
ADDRESS
GND
SDA SCL
www.i-sbc.com

The IES-SHIELD-STX4 is a four [4] channel 5V or 6-12V (jumper selectable) Unipolar stepper motor driver with advanced control features.

Specifically targeted at the Arduino UNO board user [all other Arduino boards supported] and the Raspberry-Pi the STX4 features high speed I^{2}C communication for easy project integration and smooth speed control.

Once connected each stepper motor can be stepped in either direction a number of steps (with speed control), continually stepped (with speed control), stopped (with holding torque or completely off), with selectable 1 or 2 phase excitation, by simply writing a value to an internal register over the connected I^{2}C interface.

The STX4 supports all 5-12V unipolar (5 wire) stepper motors (with a minimum winding resistance of 20R [250mA]) and also provides global activation of new stepper positions & movement complete registers for superior control.

The STX4 provides a high power regulator capable of supplying 5V @ 3A to the connected servos from an external Li-Pol or Ni-MH battery pack of 7.2 to 8.4V and a 5V @ 1A USB A power output to power an externally connected Arduino board or Raspberry-Pi.

Alternatively a jumper selection allows servos with a maximum 12V operating voltage to be used.

The on-board I^{2}C pull-ups are jumper configurable to allow disconnection when connecting to the Raspberry-Pi, which has its own pull-ups

Applications

The STX4 has applications in robotics, including 2 and 4 wheeled rovers, process control & sensor manipulation when used in conjunction with standard unipolar stepper motors.

Technical Features

- ArduinoTM UNO Shield standard form factor for simple integration into any Arduino project
- Frees up the Arduino™ IO lines normally used for motor control
- I^{2}C interface for simple connection to Arduino or Raspberry-Pi
- On-board 5Volt 3Amp regulator, with heat-sinking, for stepper motor and external 5V supply
- 8bit (1 to 255) level step speed control for each motor
- 16bit (1 to 65535) number of steps control for each motor
- Movement complete status for each motor
- Global activation control ensures all motors start moving together, important for 2/4 wheel drive rovers
- I^{2}C address links allow up to four [4] shields to be used together to provide up to 16 motors
- LVD, RoHS and WEEE compliant product

Products

IES PART NUMBER	Description
IES-SHIELD-STX4	4 Channel Stepper Driver Shield

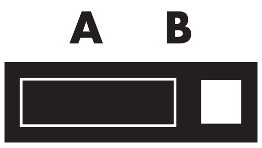
* Raspberry-Pi, Arduino, NANO, UNO & MEGA are trademark

Power requirements

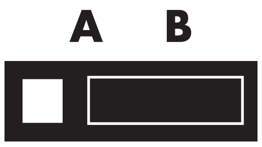
The IES-SHIELD-STX4 takes the power necessary for operation (approx. 2- 25mA) and the power for the USB type 'A' power output and stepper motors (1-3A) from an external battery / mains adaptor.

The input voltage range is 6 - 12VDC which can be applied directly to the stepper motors or through the on-board regulator which provides a clean regulated 5V supply.

The stepper motor supply for motors 1&2 (Jumper 1) and 3&4 (Jumper 2) can be selected as follows: 6-12VDC supply (A position):



5VDC supply (B position):



Refer to the voltage rating of the stepper motors you wish to use before configuring the jumpers. Stepper motors rated at 5V may be driven with a 6V supply to reduce power dissipation in the 5V 3A regulator.

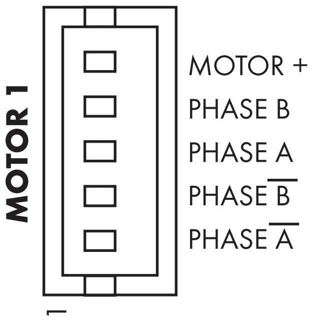
The USB type 'A' power output provides a 5V @ 1A maximum supply to power a Raspberry-Pi or Arduino platform.

Connection of the external supply - battery / mains adaptor - to the STX4 module is through a two (2) way pluggable screw terminal block marked '6-12VDC @ 3A Max'.

Note: This supply is NOT reverse connection protected but is marked with a series of '++++' signs to denote positive.

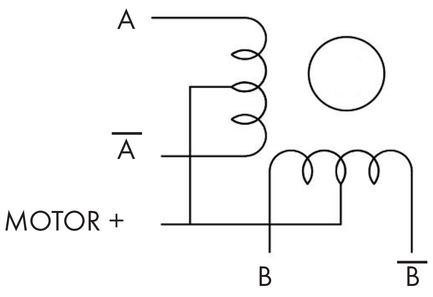
Stepper Motor Connection

The stepper motor connections are marked 'MOTOR 1' to 'MOTOR 4' and consist of a five (5) pin JST ZHR 1.5mm pitch connector pinned as follows:



This connection is standard for most 24mmD stepper motors, such as the DS-STMOT1, but there is also a pre-made 200mm JST ZHR cable available, DS-STCAB1, and the JST ZHR mating receptacle and pins are also available from Farnell electronics as items 2078162 and 1830898 respectively.

Most unipolar stepper motor phases are wired as follows:



It is recommended that you double check your motor phase connections before power application.

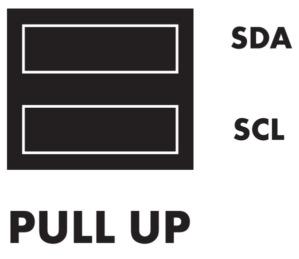
Six wire stepper motors are also supported by connecting the two centre taps both to MOTOR +.

I^{2}C Connection

The I^{2}C connections are marked 'SDA' and 'SCL' and allow connection to the Arduino UNO board

'ANALOG IN' pins 4 and 5 or the Raspberry-Pi GPIO port pins 3 and 5 (see Fig. 2.0) or another I^{2}C Master device. The IES-SHIELD-STX4 is fitted with pullup jumpers that can be configured to provide the source current necessary for I^{2}C communication. The following jumpers should normally be set when using the UNO board, as long as the I^{2}C bus does not have existing pull-up's provided by another device.

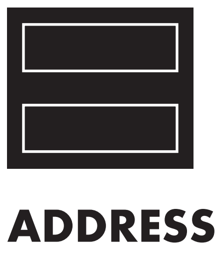
These jumpers MUST be removed when using the Raspberry-Pi:



I^{2}C Communication

Up to four IES-SHIELD-STX4 modules may be connected to the same UNO / Raspberry-Pi board or I^{2}C bus and accessed individually using their own individual address.

The address is configured with the following jumpers:



The following table shows how the jumpers are placed for the different binary addresses:

Address xx	A0	A1
00 (default)	ON	ON
01	OFF	ON
10	ON	OFF
11	OFF	OFF

The binary address (xx) above is used in conjunction with the device ID 11011xxD to form the complete device address i.e. if both jumpers are left connected (default) then the device address would be 1101100Dbinary.

The 'D' bit determines if a read or a write to the STX4 is to be performed. If the 'D' bit is set '1' then a register read is performed or if clear '0' a register write.

To access individual registers a device write must be undertaken by the I^{2}C Master which consists of a Start condition, device ID ('D' bit cleared), register to start write, one or more bytes of data to be written and a stop condition (see Figure 1.0 for I^{2}C write protocol).

There are 17 individual registers that can be written within the STX4.

N_7	N_6	N_5	N_4	N_3	N_2	N_1	N_0	

STX I2C address

1.	1	1	0	1	1	X	X	0	

XX = STX4 address

Register address

R0	U	U	U	B	B	B	B	B	

B..B = 0 to 17
U..U = unused on this implementation

Stepper 1 control

R1	U	U	U	E	D	C	B	A	

A = Operate (0 - Stepper disabled 1 - Stepper enabled)
B = Reverse (0 - Stepper normal 1 - Stepper reversed)
C = Rotate (0 - Single step 1 - Constant rotation)
D = Step (0 - Normal step 1 - Half step)
E = Hold (0 - No hold on stop 1 - Hold on stop)
U = Unused

Stepper 1 step number MSB

R2	N	N	N	N	N	N	N	N	

N..N = Stepper number of steps MSB (High byte)

Stepper 1 step number LSB

R3	N	N	N	N	N	N	N	N	

N..N = Stepper number of steps LSB (Low byte)

Stepper 1 speed

R4	S	S	S	S	S	S	S	S	

S..S = Stepper step speed in quarter mS (0.25mS)

Stepper 2 control

R5	U	U	U	E	D	C	B	A	

B = Reverse (0 - Stepper normal 1 - Stepper reversed)
C = Rotate (0 - Single step 1 - Constant rotation)
D = Step (0 - Normal step 1 - Half step)
E = Hold (0 - No hold on stop 1 - Hold on stop)
U = Unused

Stepper 2 step number MSB

R6	N	N	N	N	N	N	N	N

N..N = Stepper number of steps MSB (High byte)

Stepper 2 step number LSB

R7	N	N	N	N	N	N	N	N

N..N = Stepper number of steps LSB (Low byte)

Stepper 2 speed

R8	S	S	S	S	S	S	S	S

S..S = Stepper step speed in quarter mS (0.25mS))

Stepper 3 control

R9	U	U	U	E	D	C	B	A

A = Operate (0 - Stepper disabled 1 - Stepper enabled)
B = Reverse (0 - Stepper normal 1 - Stepper reversed)
C = Rotate (0 - Single step 1 - Constant rotation)
D = Step (0 - Normal step 1 - Half step)
E = Hold (0 - No hold on stop 1 - Hold on stop)
U = Unused

Stepper 3 step number MSB

R10	N	N	N	N	N	N	N	N

N..N = Stepper number of steps MSB (High byte)

Stepper 3 step number LSB

R11	N	N	N	N	N	N	N	N

N..N = Stepper number of steps LSB (Low byte)

Stepper 3 speed

R12	S	S	S	S	S	S	S	S

S..S = Stepper step speed in quarter mS (0.25mS)

Stepper 4 control

R13	U	U	U	E	D	C	B	A

A = Operate (0 - Stepper disabled 1 - Stepper enabled)
B = Reverse (0 - Stepper normal 1 - Stepper reversed)
C = Rotate (0 - Single step 1 - Constant rotation)
D = Step (0 - Normal step 1 - Half step)
E = Hold (0 - No hold on stop 1 - Hold on stop)
U = Unused

Stepper 4 step number MSB

R14	N	N	N	N	N	N	N	N

N..N = Stepper number of steps MSB (High byte)

Stepper 4 step number LSB

R15	N	N	N	N	N	N	N	N

N..N = Stepper number of steps LSB (Low byte)

Stepper 4 speed

R16	S	S	S	S	S	S	S	S

S..S = Stepper step speed in quarter mS (0.25mS)

Servo global enable register

R17	X	X	X	X	X	X	X	X

X..X = Any value

Each control register consists of five control bits defined as follows:

Bit (A) 1decimal is the operate bit which when set activates the stepper being controlled.

Bit (B) 2decimal is the reverse bit which reverses the direction of the stepper being controlled.

Bit (C) 4decimal is the rotate bit which makes the stepper motor constantly rotate rather than step the number of steps defined in the step number register.

Bit (D) 8decimal is the step bit which allows the stepper motor to be configured for normal or half-step operation.

Bit (E) 16decimal is the hold bit which allows the stepper motor to hold its position (holding torque) once the number of steps have been made or go free (no holding torque).

Once all the required step & control registers have been set a write to the R17 (Global enable register) must be made to activate all the new settings.

Example

To step motor 1 500 half-steps clockwise at 3mS per step and then hold in that position, first write:

Byte 1 (STX4 Adr) 1	1011000_{binary}
Byte 2 (Register 0)	$0_{decimal}$
Byte 3 (Register 1)	$25_{decimal}$, 19_{hex}
Byte 4 (Register 2)	$1_{decimal}$, 01_{hex}
Byte 5 (Register 3)	$232_{decimal}$, $E8_{hex}$
Byte 6 (Register 4)	$12_{decimal}$, $0C_{hex}$

then to activate write:

Byte 1 (STX4 Adr)	11011000_{binary}
Byte 2 (Register 0)	$17_{decimal}$
Byte 3 (Register 17)	$0_{decimal}$

The individual registers are broken down into:

Register 1 = 25 which sets bits A (enable motor), D (half-step) and E (hold on stop)

Register 2 & 3 = 500 (1E8 hex) split into register 2 = 01 and register 3 = E8.

Register 4 = 12 which configures the step to 3mS (12x0.25mS = 3mS)

To read the status registers a device write then read must be undertaken by the Arduino / Raspberry-Pi / I^2C Master. The write consists of a Start condition, device ID ('D' bit cleared), register to start read and a Stop condition.

This is followed by a read, which consists of a Start condition, device ID ('D' bit set), followed by data from the status register and terminated with a Stop condition (see Figure 1.1 for I2C read protocol).

Status registers

There are 5 registers that can be read within the STX4 as follows:

N_7	N_6	N_5	N_4	N_3	N_2	N_1	N_0	

SCX I2C Address

1.	1	1	0	1	1	X	X	1	

XX = STX4 address

Stepper 1 status

R0	A	0	0	0	0	0	0	0	

A = Movement (0 – Complete 1 – In-progress)

Stepper 2 status

R1	A	0	0	0	0	0	0	0	

A = Movement (0 – Complete 1 – In-progress)

Stepper 3 status

R2	A	0	0	0	0	0	0	0	

A = Movement (0 – Complete 1 – In-progress)

Stepper 4 status

R3	A	0	0	0	0	0	0	0	

A = Movement (0 – Complete 1 – In-progress)

Firmware version

R18	M	M	M	M	V	V	V	V	

M..M = Firmware major revision number 1-15
V..V = Firmware minor revision number 1-15

Bit (A) 128decimal is cleared to indicate if the current stepper motor stepping process has completed.

Electrical Characteristics (T_A = 25˚C Typical)

Parameter	Minimum	Maximum	Units	Notes
Supply Voltage (Vin)	6	12	V	1,2
Supply Current (operational)	2	25	mA	
Output Voltage (Stepper motors)	5	Vin	V	3
Output Current (Stepper motors)	1	2000	mA	4
Output Voltage (USB output)	4.8	5.2	V	5
Output Current (USB output)	0	1000	mA	5
I^2C pull-up resistance	-	4700	Ω	
I^2C speed	-	400	kHz	

Absolute Maximum Ratings

Parameter	Minimum	Maximum	Units	Notes
Supply Voltage (Vin)	-0.5	+15	V	5
Supply Current (All)	0	3.0	A	

Environmental

Parameter	Minimum	Maximum	Units
Operating Temperature	0	70	˚C
Storage Temperature	-10	80	˚C
Humidity	0	80	%
Dimensions	Length 56.25mm, Width 53.5mm, Height 15mm		
Weight	16g		
Immunity & emissions	See statement on page 8		

Notes:
1.Minimum voltage given is required for normal operation whether stepper motor regulator is used or not.
2.Force cooling of the heat-sinks may be required if the 5V regulator is enabled and supply voltage in high.
3.Maximum value given is when the 5V regulator is disabled ie. the stepper motor voltage is the same as the supply voltage.
4.Values given are for stepper motors being not driven and driven.
5.Values given are based on maximum and minimum loading.

Calculating binary bit values:

The registers used above use the binary notation to allow the control of stepper motor operation, reversal, rotation type, step type and hold. Each register is made up of eight (8) bits, which can be set or cleared to produce the desired operation, the individual bits having a value associated with them as follows:

128	64	32	16	8	4	2	1

If we take for example one of the stepper control registers we can see it is made up of five (5) separate bits A, B, C, D & E plus three unused bits UUU:

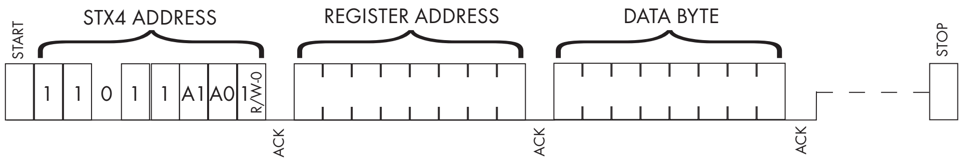
Stepper 1 control

R1	U	U	U	E	D	C	B	A

A = Operate (0 - Stepper disabled 1 - Stepper enabled)
B = Reverse (0 - Stepper normal 1 - Stepper reversed)
C = Rotate (0 - Single step 1 - Constant rotation)
D = Step (0 - Normal step 1 - Half step)
E = Hold (0 - No hold on stop 1 - Hold on stop)
U = Unused

Each bit is defined to control a particular function for the stepper it controls, so if for example we wanted to enable stepper 1 we would need to set bit 'A' which controls the stepper operation. We know from the bit values defined above that the value associated with the 'A' bit is 1, so by writing this value to register 1 we can enable stepper 1. If we need to enable additional functions such as the constant rotation - 'C' - as well as the stepper enable, the value of this bit is added to the value written to the register i.e. 1 + 4 = 5. In addition we could also reverse the direction of rotation that would make the total value 1 + 4 + 2 = 7.

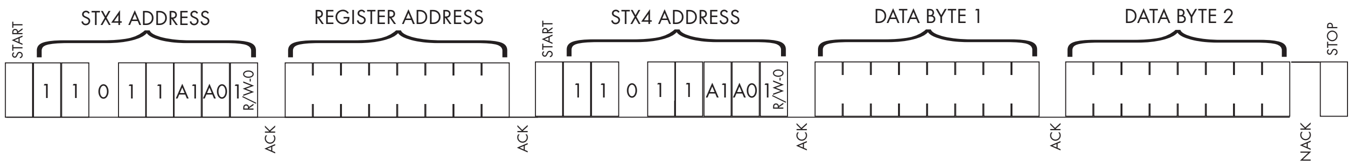
Figure 1.0 (I²C write protocol)



Multiple bytes may be written before the 'STOP' condition. Data is written into registers starting at 'REGISTER ADDRESS', then 'REGISTER ADDRESS' +1, then 'REGISTER ADDRESS' +2 etc.

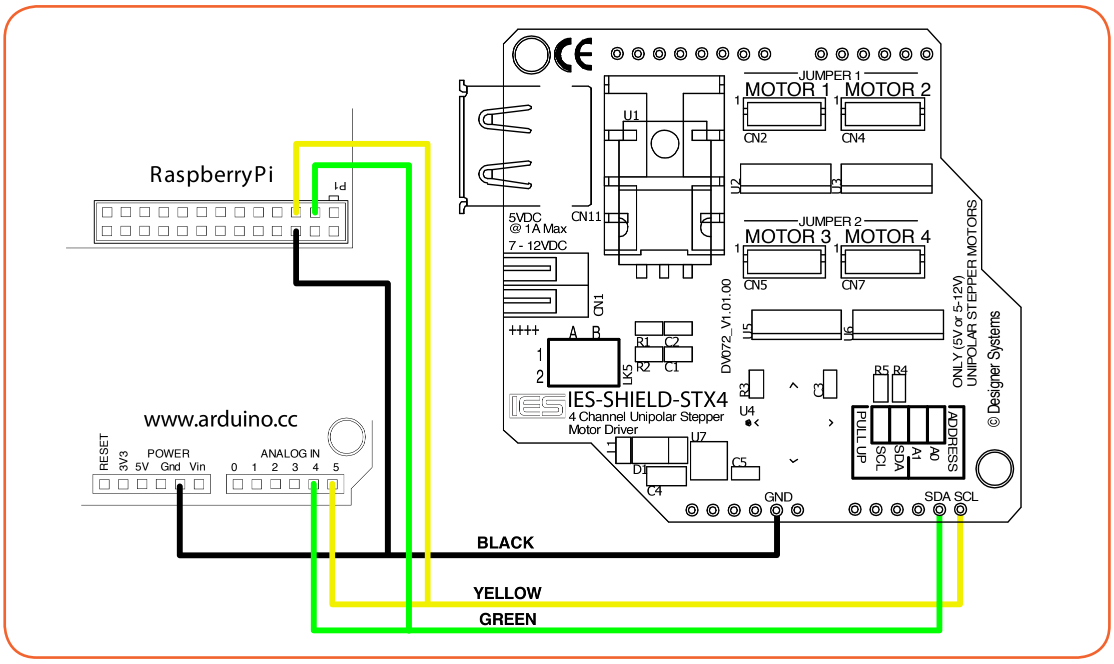
Each byte transfer is acknowledged 'ACK' by the STX4 until the 'STOP' condition.

Figure 1.1 (I²C read protocol)

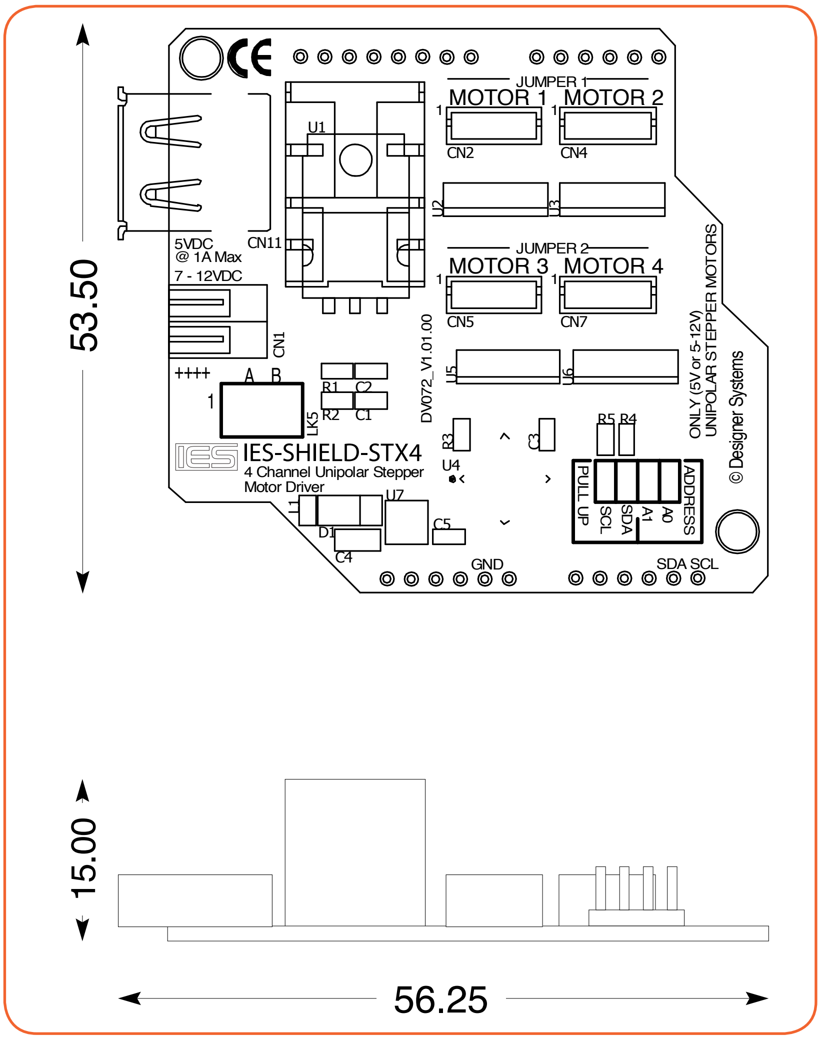


'DATA BYTE 1 & 2' are register values returned from the STX4. Each byte written is acknowledged 'ACK' by the STX4 , every byte read is acknowledged 'ACK' by the I2C Master. A Not-acknowledge 'NACK' condition is generated by the I2C Master when it has finished reading.

Figure 2.0 (Connection Schematic for Arduino UNO or Raspberry-Pi I^{2}C communication)



Mechanical Specifications – Units millimetres



WEEE Consumer Notice

This product is subject to Directive 2002/96/EC of the European Parliament and the Council of the European Union on Waste of Electrical and Electronic Equipment (WEEE) and, in jurisdictions adopting that Directive, is marked as being put on the market after August 13, 2005, and should not be disposed of as unsorted municipal/public waste. Please utilise your local WEEE collection facilities in the disposition and otherwise observe all applicable requirements. For further information on the requirements regarding the disposition of this product in other languages please visit www.i-sbc.com

RoHS Compliance

This product complies with Directive 2002/95/EC of the European Parliament and the Council of the European Union on the Restriction of Hazardous Substances (RoHS) which prohibits the use of various heavy metals (lead, mercury, cadmium, and hexavalent chromium), polybrominated biphenyls (PBB) and polybrominated diphenyl ethers (PBDE).

For further information please contact IES

The values contained in this data sheet can change due to technical innovations. Any such changes will be made without separate notification.