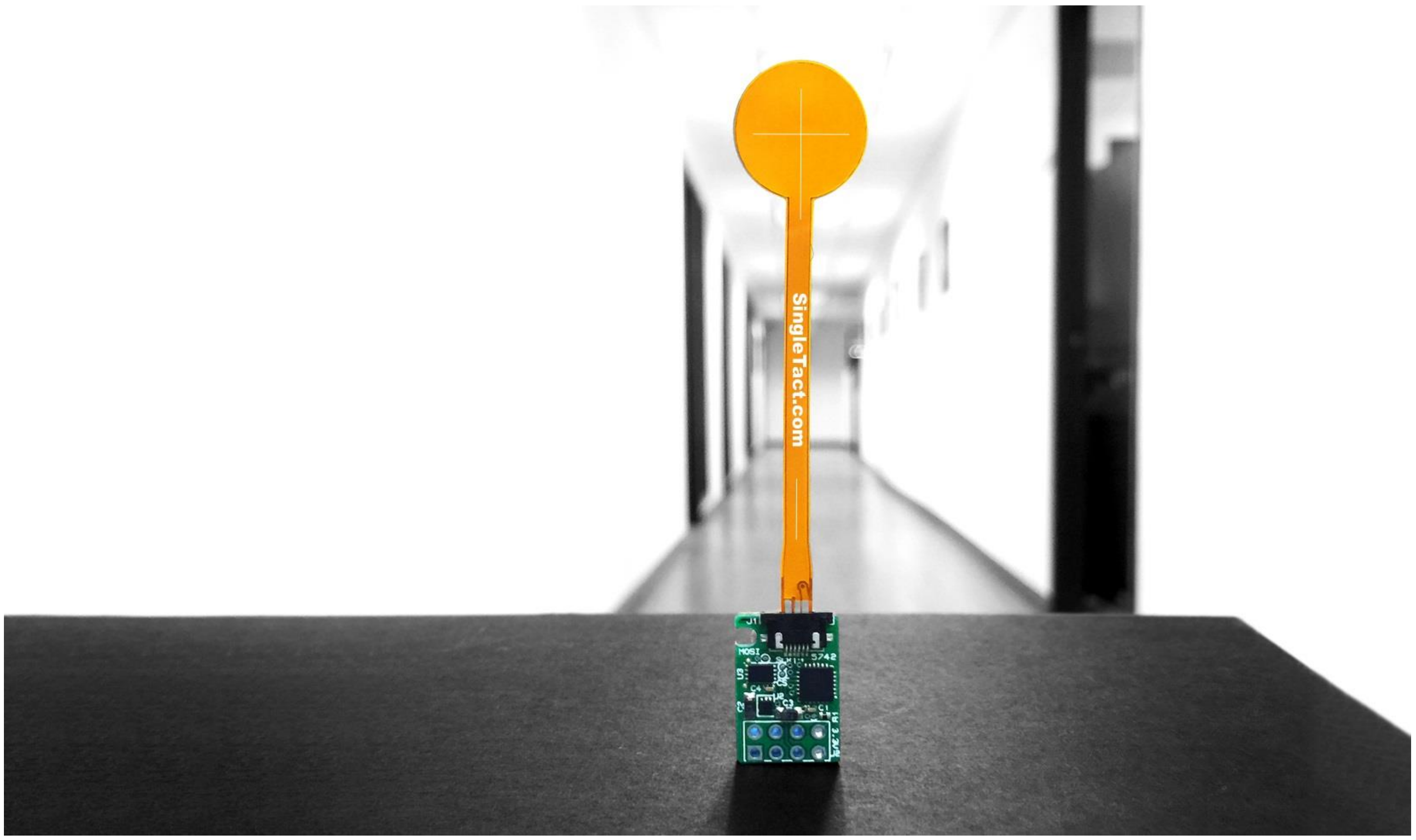
USER MANUAL

EXPERIENCE
INCREDIBLE
PERFORMANCE



CONTENTS

1 INTRODUCTION

SingleTact is a single element tactile pressure sensor that accurately and reliably quantifies applied force combined with a simple interface board offering a 0 to **2V** analog output for immediate Data Acquisition (DAQ) integration and an I^{2}C based interface for integration into embedded systems. Standard and Calibrated sensors (with matched pre-calibrated interface board) are available.

This document provides all the information necessary to interface with the SingleTact including a sample Arduino digital interface and simple C# PC DAQ software (see EXAMPLE USE CASE)

All demo and API source code is open source and can be downloaded from: www.singletact.com.

Figure 1 SingleTact Sensor and Interface Board

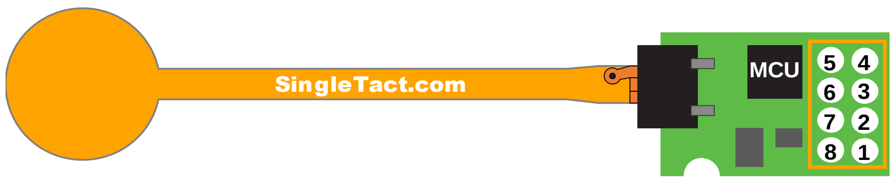
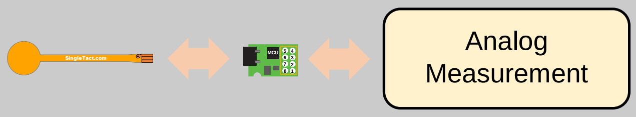
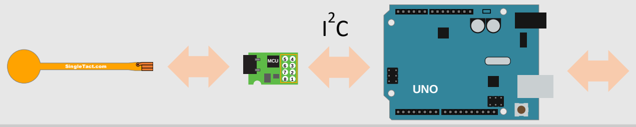
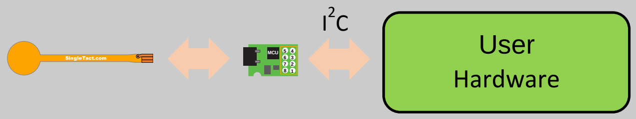
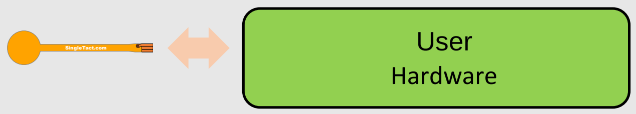


Figure 2 Use Case Configurations

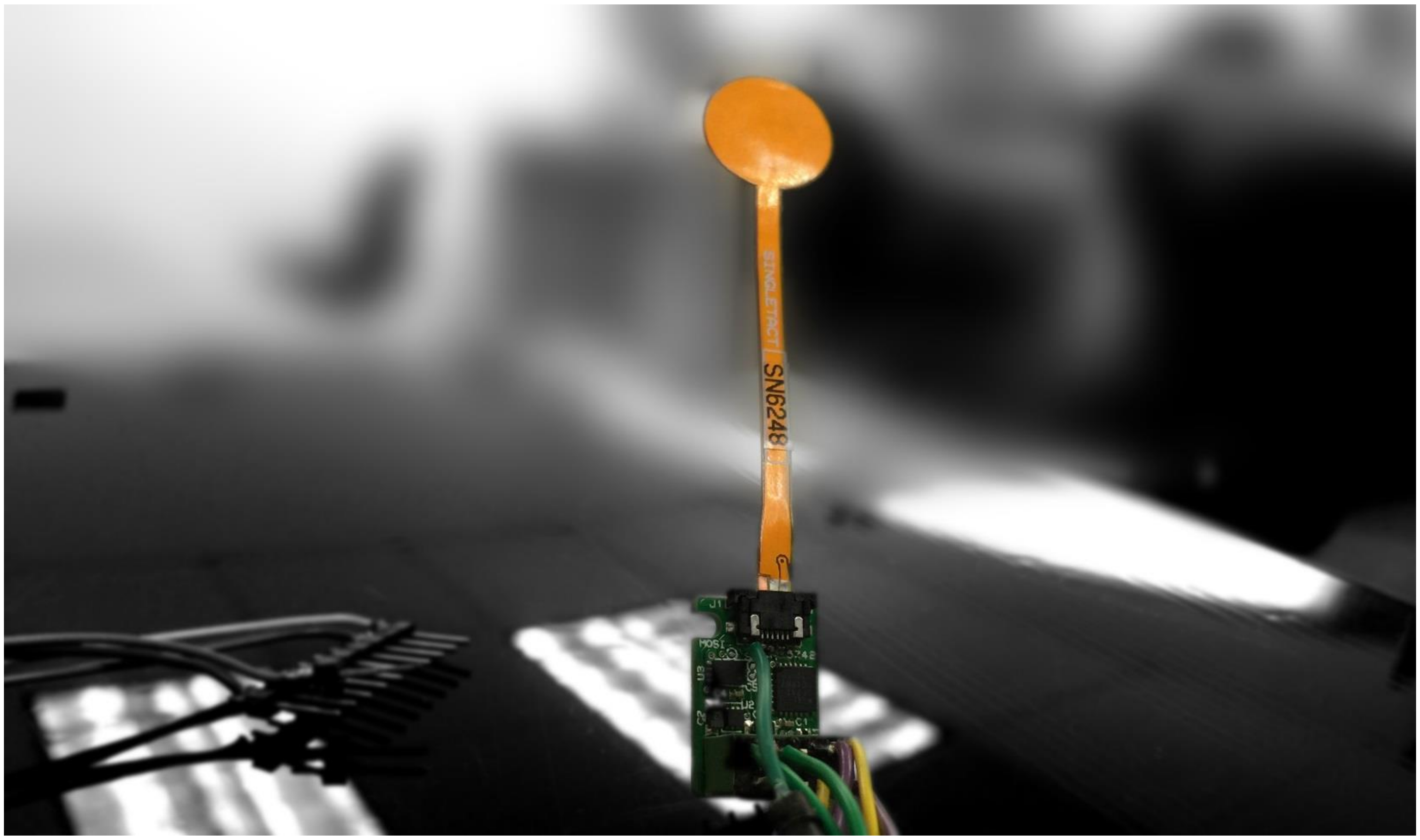
	Use Case	System Overview	
1	Force Measurement		Analogue Out. Connect to multitier, oscilloscope or data acquisition card
2	Evaluation/ Force Measurement[*1]		Arduino and PC data acquisition evaluation software
3	Customer Integration[*2]		I^{2}C Digital Output. Interface to user electronics
4	Customer Integration[*3]		User reads capacitance using their own circuitry

*1 – In addition to the data acquisition example, a .NET library is available to download for simple integration into a user's own software suite. See Example .NET API.

*2 – Supports over 100 SingleTact interface boards on a single I^{2}C bus. The interface board firmware can be modified to fit user's specific use cases – if required please contact PPS to discuss this option.

*3 – PPS maybe able to assist with this – use the contact links at http://www.singletact.com/contact/.

2 INTERFACE DESIGN



2.1 Connectivity

The sensor is plugged into the FFC connector on the green interface board (**with the sensor connector pads facing upward**). The connections are outlined in Figure 4.

Electrical parameters are outlined in Table 1.

Figure 3 Sensor Assembly

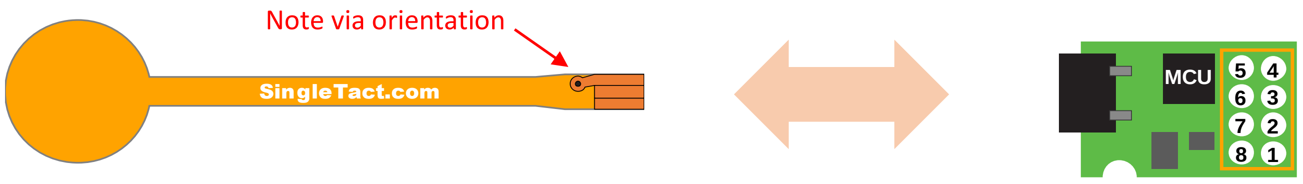


Figure 4 Interface board header connections

CONNECTION	PIN NUMBER		CONNECTION
Reserved	5	4	Reserved
I^2C Interface (SDA)	6	3	I^2C Interface (SCL)
Frame Sync	7	2	Analog Out
Ground	8	1	Vcc

Table 1 Electrical parameters

Parameter	Value
Supply Voltage, Vcc	3.7 – 5V
I^2C clock frequency	100KHz or 400KHz
I^2C bus level	3 – 5V
I^2C output range (sensor data)	10-bit (Operational FSR output 9-bit)
Analog output range	0 – 2V (Operational FSR output 0.5 – 1.5V)
Permitted analog output load	>5KΩ
Frame Sync level	3.3V CMOS output
Sensor update rate (I^2C or analog)	>140Hz (dependent on settings)

2.2 Analog Interface

The analog output swings from **0** to **2V**, with the valid working output ranging from **0.5V** to **1.5V** as shown in Figure 5.

As pressure increases beyond the full scale range (FSR) the output will increase to **2V** and then limit.

The sensor should be unloaded at power on to allow the sensor's baseline to be registered correctly.

NOTE: An output below **0.5V** may indicate negative pressures, which occur when the sensing area is under tension. This should be avoided since it can damage the internal structure of the sensor.

NOTE: Sensor over pressure should be limited to less than 3x FSR to avoid damaging the sensor.

Figure 5 Analog Output

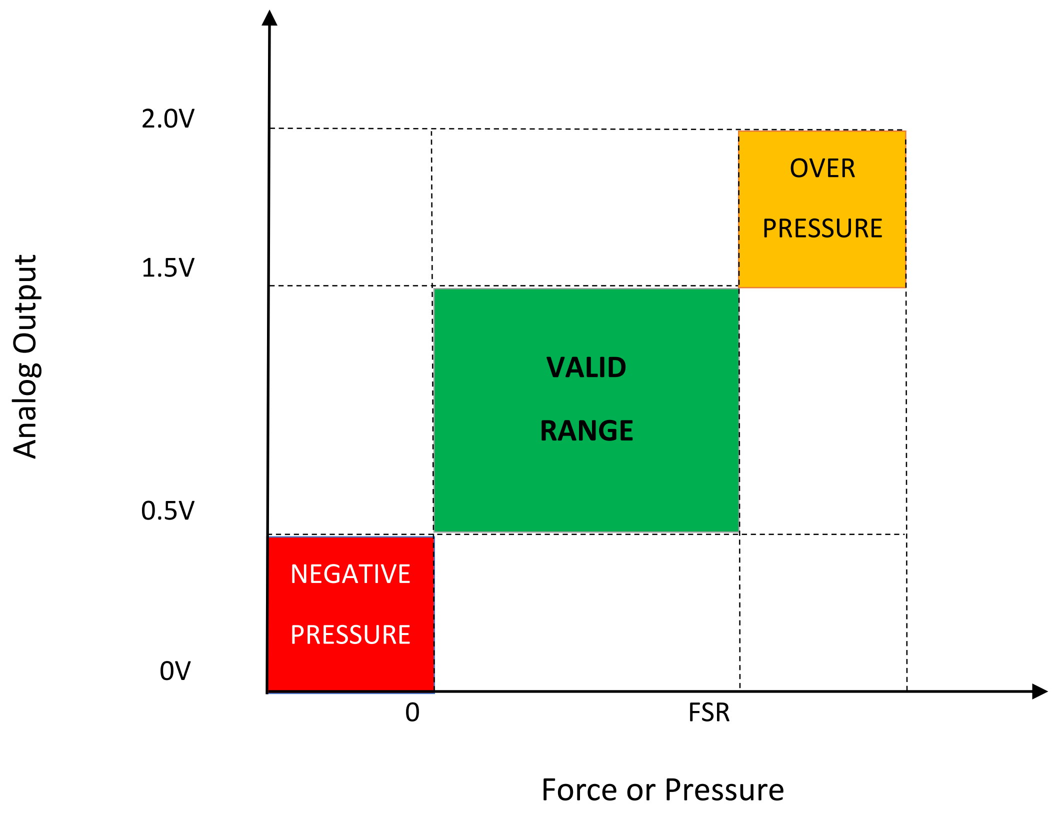


Figure 6 DAQ Connection Requirements

CONNECTION	PIN NUMBER		CONNECTION
No Connect	5	4	No Connect
No Connect	6	3	No Connect
No Connect	7	2	Analog Out
Ground	8	1	Vcc

2.3 I^{2}C Interface

The SingleTact I^{2}C interface supports the standard (**100 Kbits/s**) clock rate in **7-bit** address mode.

The SCL and SDA lines must be pulled up to the bus voltage which can be between **3V** and **5V**. Please refer to the I2C specification for bus protocol implementation & pull-up value considerations.

The interface board will always respond to two I^{2}C addresses: **0x04** ***and*** the address specified in flash (register address 0). As shipped the default flash address is also **0x04.**

Multiple sensor interfaces may be connected to a single I^{2}C bus. The bus address of individual sensor interfaces can be configured by writing desired address value (4 to 127) via the I^{2}C interface to register address 0 with an I2C Write Operation. Change of individual sensor I^{2}C addresses is supported by the PC and Arduino Example.

NOTE: As the interface board will always respond to address 0x04 then this address must be considered reserved for SingleTact. Where multiple SingleTact interfaces are to be connected to the same I^{2}C bus then address 0x04 must be considered invalid and in this use case the configurable address of all connected SingleTact nodes must be individually changed from the default value before each SingleTact is added to the multi-node bus.

The SingleTact software architecture is based on a **192 byte** register block – see Figure 7 and Table 2 for details.

All control registers are located in first **112** bytes and get written to NVM when modified (and are therefore persistent after a power cycle). Configuration registers on Calibrated sensors interfaces are protected from modification.

The sensor results are available from bytes **128 to 133.** As shipped, results are updated at >140Hz (this is dependent on capacitance sensor settings).

NOTE: It is the Users responsibility not to write to any Reserved locations.

Figure 7 Register Layout

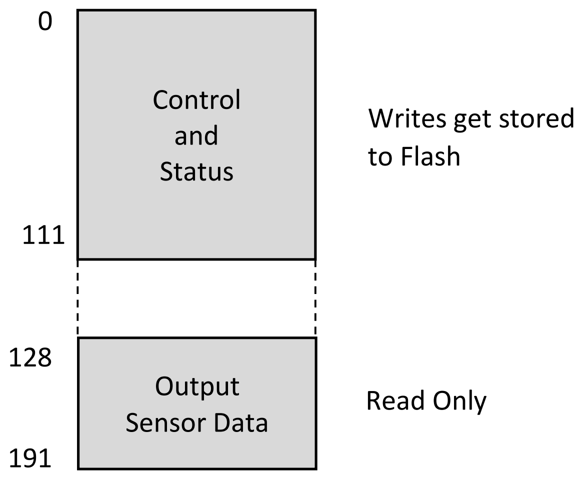


Table 2 Register Details

BYTE	SETTING
0	I^{2}C Address (4-127)
1	User configurable serial number MSB
2	User configurable serial number LSB
3	Reserved
4	Reserved
5	Capacitive Sense (Accumulator) Default 0x04 *1
6	Capacitive Sense (Reference Gain) Default 0x01 *1
7	Reserved
8	Capacitive Sense (Discharge Time) Default 0x03 *1

BYTE	SETTING
9	Capacitive Sense (Output Current) Default 0x00 [*1]
10	Output digital scaling MSB
11	Output digital scaling LSB
12	Number of elements, must be 1
13	Reserved
14	Delimiter – leave as 0xFF
15	First element to scan, set to 0
16-39	Reserved
40	Delimiter – leave as 0xFF
41	Sensor baseline MSB
42	Sensor baseline LSB
43-90	Reserved
91	Delimiter – leave as 0xFF
92-127	Reserved
128	Frame index MSB (increments on each new reading)
129	Frame index LSB (increments on each new reading)
130	Sensor Timestamp MSB (0.1ms increments) [*1]
131	Sensor Timestamp LSB (0.1ms increments) [*1]
132	Sensor output MSB

BYTE	SETTING
133	Sensor output LSB
134 - > 191	Reserved

**1 Should only be used as a coarse estimate as it is subject to drift.*

2.4 I^{2}C Operations

I^{2}C SingleTact supports three I^{2}C operations: Write, Read Request and Read.

2.4.1 I^{2}C Write Operation

I^{2}C bus Transfer: *Master Write to Slave*.

This command writes values to the register block. All writes also update the internal flash memory so settings are persistent through a power cycle.

Bytes **3** to **N-1** (where N is the packet length) contain the data to be written to consecutive registers.

Data may be written to the first **128** bytes (excluding reserved locations).

Writing outside of the valid range will fail.

Configuration registers on Calibrated sensors interfaces are protected from modification.

Table 3 Write Operation Data Packet Format

BYTE	TO SENSOR
0	0x02
1	Write offset in register block
2	Number of bytes to write (1 – 28)
3 -> (N-1)	Data to write (1 to 28 bytes)
N (max 31)	0xFF – end of packet delimiter

2.4.2 I^{2}C Read Request Operation

I^{2}C bus Transfer: *Master Write to Slave*.

This command sets the read location (register block offset) and read length for a following Read operation.

Table 4 Read Request Operation Data Packet Format

BYTE	TO SENSOR
0	0x01
1	Read offset in register block
2	Number or bytes to read (1 – 32)
3	0xFF – end of packet delimiter

2.4.3 I^{2}C Read Operation

I^{2}C bus Transfer: *Master Read from Slave*.

An I^{2}C *Master Read from Slave* transfer can be used to directly read the register set and sensor data.

In normal operation a read of the two Sensor Output byte registers returns a 10-bit output range from 0000 to 0x3FF, corresponding to the 0 to 2V analog output range.

The functional sensor FSR output range is 9-bits from x0100 to 0x2FF, corresponding to the 0.5V to 1.5V analog output range. The larger 10-bit total range allows for detection of negative values when the sensor is under tension and some level of over pressure detection.

The sensor should be unloaded at power on to allow the sensor's baseline to be registered correctly.

Where a Read operation is not preceded by a Read Request operation the read location defaults to **128** (the sensor output location) and consecutive reads will therefore simply read the default 32 bytes of the sensor data region.

Where a Read operating is preceded by a Read Request operation then the register offset and read length as set by the Read Request will be used.

Data can be read from anywhere in the register block (addresses **0 – 191**).

Reading outside of the valid range will fail.

I^{2}C slave Read operations simply return the register data values up to the number of requested bytes (32 max) in the data packet.

NOTE: A sensor output reading below **0x0100** may indicate negative pressures, which occur when the sensing area is under tension. This should be avoided since it can damage the internal structure of the sensor.

NOTE: Sensor over pressure should be limited to less than 3x FSR to avoid damaging the sensor.

Table 5 I^{2}C Master Read from Slave Data Packet Format

BYTE	FROM SENSOR
0 - 31*	Register Data Read Location-Read Location+31*

* number of bytes read can be modified by a preceding read-request command.

2.5 Conversion Detail

The SingleTact electronics interface measures the capacitive sensor with **16-bit** precision. This is scaled to a **10-bit** digital (**2V** analog) output using the following calculation:

$$\textit{SingleTact Output} = \frac{\textit{Raw capacitance} - \textit{Baseline capacitance}}{\textit{Digital scaling value}}$$

The digital scaling value is a 16-bit value stored at register locations 10 and 11 (see Table 2). For increased precision (within a given sensor's valid operating range) the digital scaling value can be adjusted in **0.01** increments. A value of **100** represents unity scaling (100 x 0.01).

The internal capacitance to digital converter (CDC) operates at **140** to **4000** Hz depending on the capacitance sensor settings (in particular the number of accumulations).

Each time the CDC completes a measurement

- the output register gets updated
- the frame index increases by one
- an active high pulse is produced on the frame synchronization output pin
- a timestamp is generated by SingleTact interface board (however as there is no crystal oscillator this should only be used as a coarse estimate).

The frame synchronization output (Frame Sync, Figure 4) goes high each time a new measurement is available. This can be used to synchronize the I^2C communications channel to the capacitance sensor.

Alternatively, the sensor might be polled as quickly as possible over I^2C. Since the frame index increments with each frame it can be used to identify duplicate or missing data readings.

2.6 Product Categories

Standard sensors will perform typically to their specified force range.

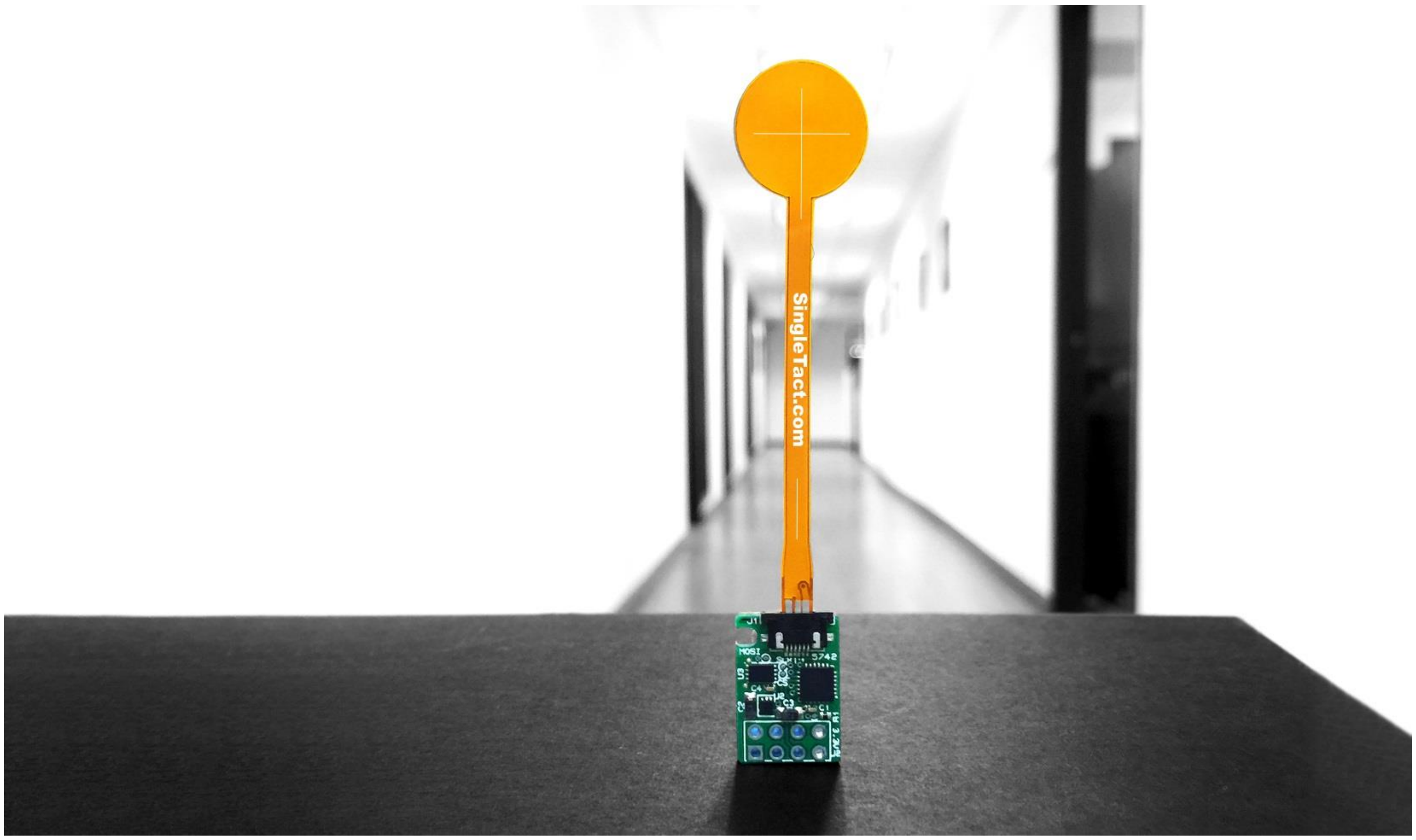
The separately available, generic electronics interface from PPS is suitable for use with the Standard sensors. With a standard sensor fitted and loaded to its specified maximum force (e.g. 45N for a 45N sensor) the scaling factor may be configured via the I^2C bus to generate an I^2C sensor value equal to 511 and equivalent analog output of 1.5V to ensure the sensor is operating within its valid range.

Calibrated sensors offer improved accuracy and linearity over the Standard sensors and come as a matched sensor plus electronics interface board providing a pre-configured, calibrated linear output for the specified sensor range, e.g. for an input force of 0 to 10N a 10N calibrated sensor will output a linear I^2C range of 0 to 511 and equivalent analog output of 0.5V to 1.5V.

To maintain calibration, calibrated electronics + sensor combinations as shipped should be maintained as matched pairs.

To implement multiple sensor solutions multiple sensor interface boards can be connected to a single I^2C bus as described in the I2C Interface section.

3 TROUBLESHOOTING SingleTact



3.1 Arduino UNO not detected by PC.

Arduino UNO requires to install a driver to communicate over USB port.

Follow the step-by-step instruction from https://www.arduino.cc/en/Guide/HomePage.

3.2 Invalid setting error on PC (Popup reports" Failed to set").

Likely reason:

- faulty pin connection.

3.3 No Analog output (remains at 0V).

Check wire connections and ensure that you are powering the sensor.

Likely reasons:

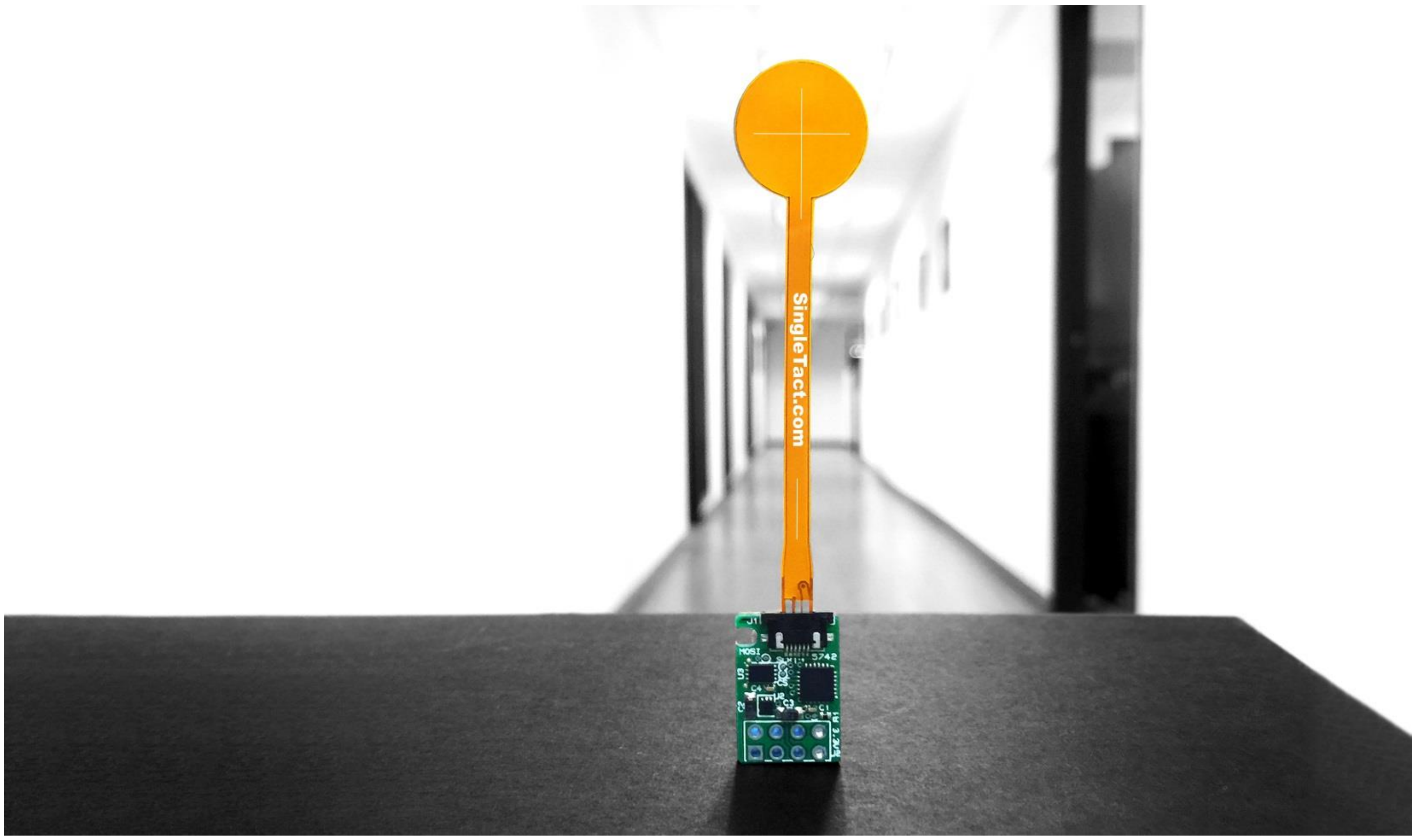
- Power, Ground or output connection in the wrong place.
- power supply is off or faulty.

3.4 Analog output stays at 0.5V.

Likely reason:

- Possible sensor fault.
    - Check sensor orientation (see Figure 3).
    - If in error analog output will stay at 0.49 - 0.5 V.
    - Digital output will remain on the baseline (0 counts).

4 EXAMPLE USE CASE



4.1 PC and Arduino Example

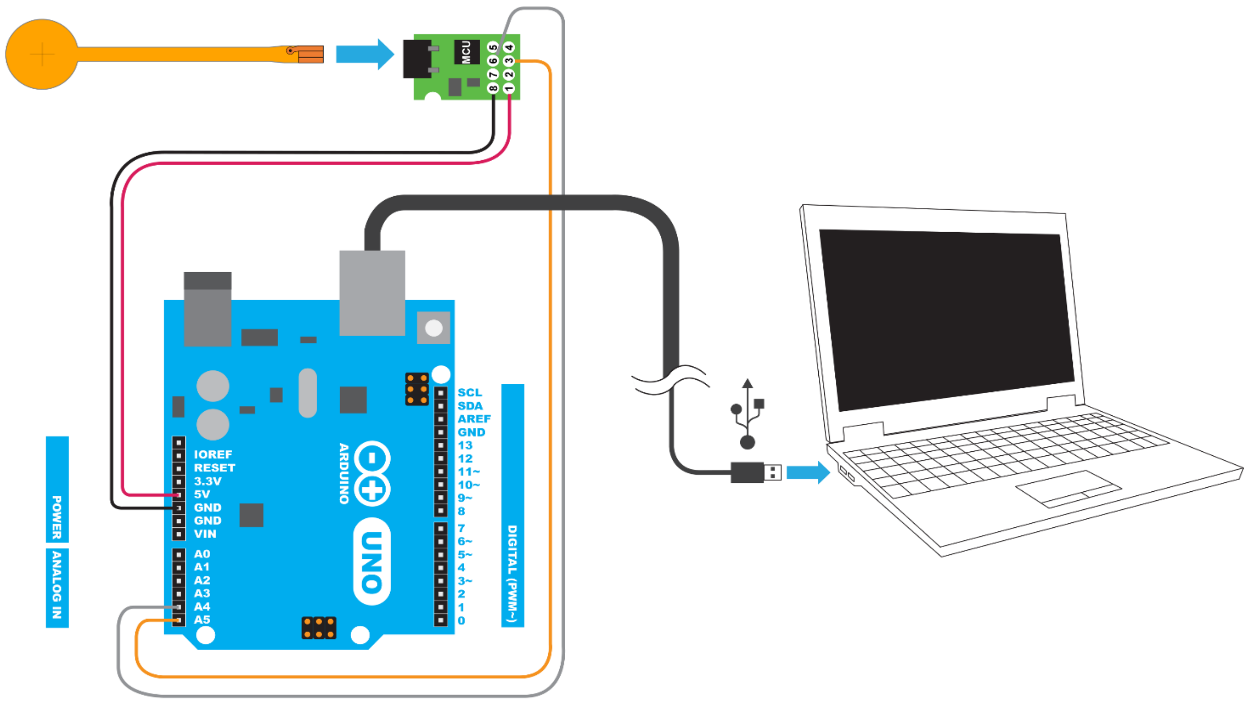
An Arduino UNO board can be used to implement a USB serial interface to SingleTact.

The code for an Arduino application (source) and an associated .NET based PC DAQ GUI application (both Windows executable and source) can be downloaded via www.singletact.com.

Once the Arduino board has been programmed with the SingleTact firmware (see Programming the Arduino UNO with SingleTact Example) the PC application can be run to visually observe the sensor results.

As the Arduino code is stored in flash the programming (or 'upload' in Arduino terms, only needs to be done once for a new board.

Figure 8 Arduino and SingleTact Assembly



Note: USB communication may need additional driver installation from Arduino software package. See https://www.arduino.cc/en/Guide/Windows#toc4 for further information.

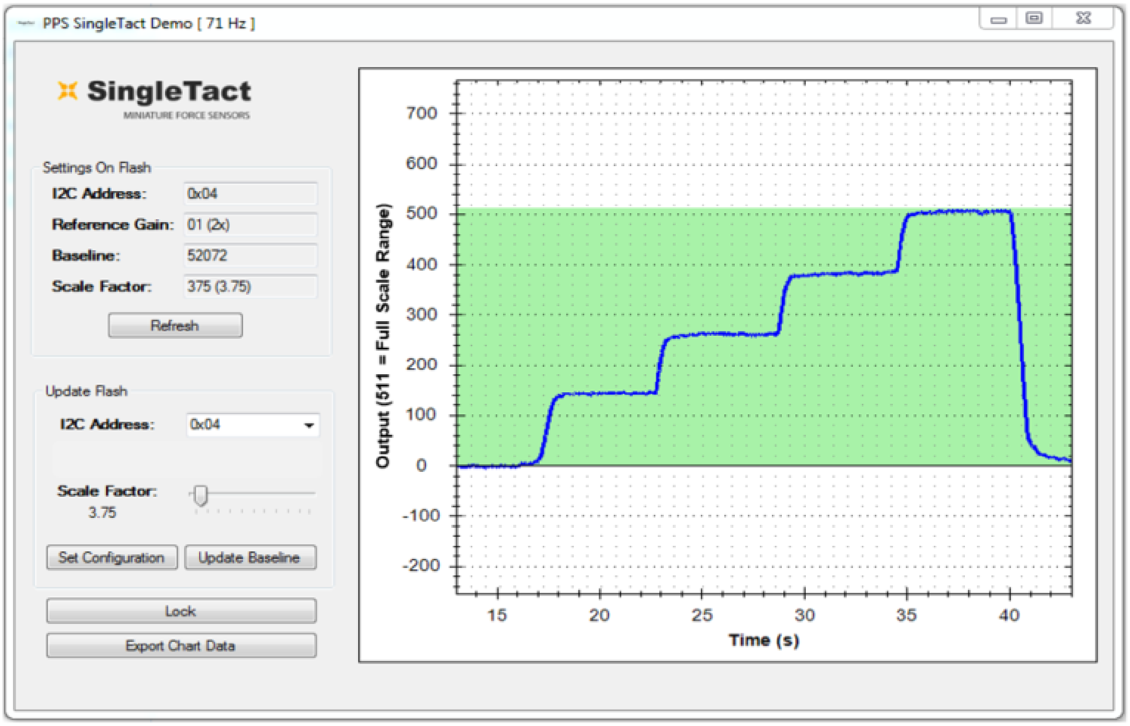
Figure 9 SingleTact and Arduino UNO Connection

CONNECTION	PIN NUMBER		CONNECTION
No Connect	5	4	No Connect
Arduino UNO pin A4	6	3	Arduino UNO pin A5
No Connect	7	2	No Connect
Arduino UNO GND pin	8	1	Arduino UNO 5V pin

To run the Windows GUI application:

- Open the PCExecutable folder.
- Run SingleTact Demo.exe to bring up the demonstration application.

Figure 10 Demo of PC DAQ software



Settings On Flash

I2C Address: 0x04
Reference Gain: 01 (2x)
Baseline: 52072
Scale Factor: 375 (3.75)

Update Flash

I2C Address: 0x04
Scale Factor: 3.75

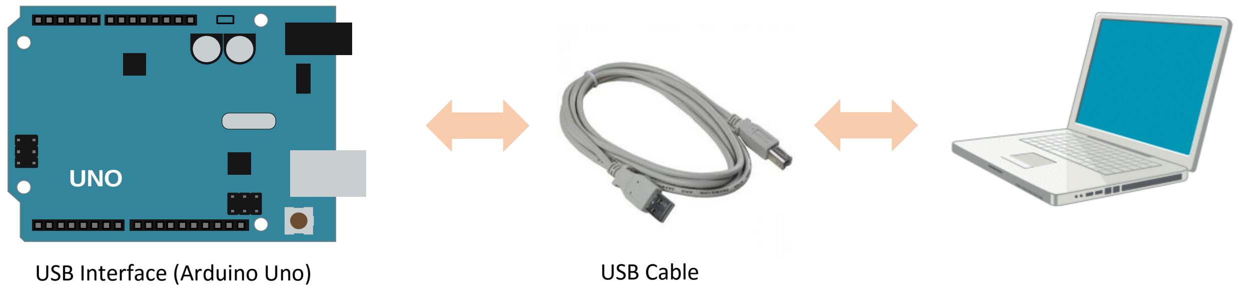
The PC application can be used to change the sensor’s I^{2}C address and to modify its output scaling. For more information on these settings please refer to the I2C Interface section.

4.2 Programming the Arduino UNO with SingleTact Example

This process outlines how to program the Arduino UNO with SingleTact example firmware.

1. Download and install the Arduino Software from: https://www.arduino.cc/en/Main/Software
2. Download the Arduino firmware (ExampleArduinoInterface) from: www.singletact.com
3. Connect the Arduino to the PC using the supplied USB cable.
4. Open the Arduino IDE software:

Figure 11 Arduino - PC connection



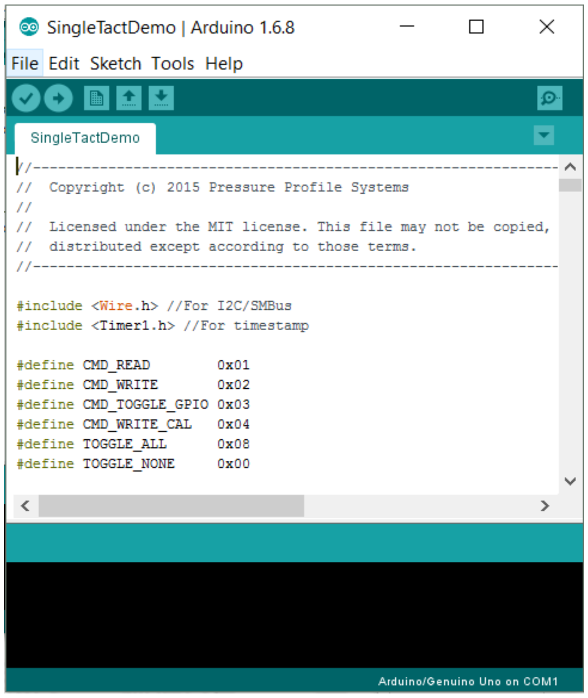
Note: USB communication may need additional driver installation from Arduino software package. See https://www.arduino.cc/en/Guide/Windows#toc4 for further information.

Follow the step-by-step instructions.

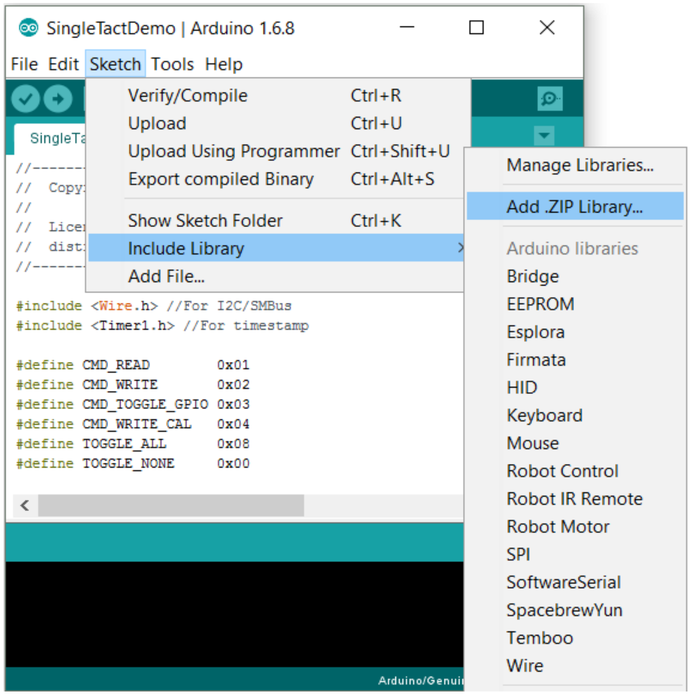
1. Go to File --->Open and open “SingleTactDemo.ino”
2. Go to Sketch --->Include Library --->Add .zip Library and select “Timer1.zip”
3. Go to Sketch --->Verify/Compile.
4. Go to Sketch --- > Upload*.

***Note:** If you receive an error on Upload make sure the Arduino is selected under Tools --->Port.

Figure 12 Arduino integrated development environment

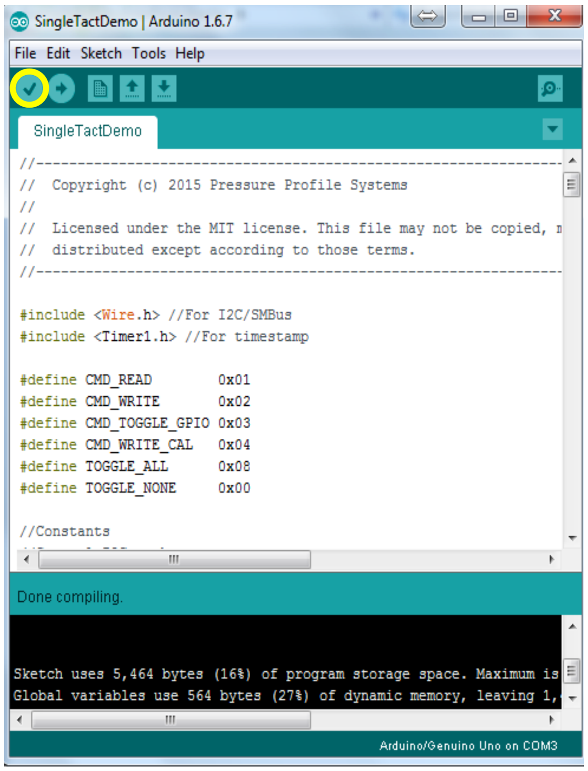

SingleTactDemo | Arduino 1.6.8
File Edit Sketch Tools Help
SingleTactDemo
Arduino/Genuino Uno on COM1

```
//------------------------------------------------------------------
//  Copyright (c) 2015 Pressure Profile Systems
//
//  Licensed under the MIT license. This file may not be copied,
//  distributed except according to those terms.
//------------------------------------------------------------------

#include <Wire.h> //For I2C/SMBus
#include <Timer1.h> //For timestamp

#define CMD_READ        0x01
#define CMD_WRITE       0x02
#define CMD_TOGGLE_GPIO 0x03
#define CMD_WRITE_CAL   0x04
#define TOGGLE_ALL      0x08
#define TOGGLE_NONE     0x00
```



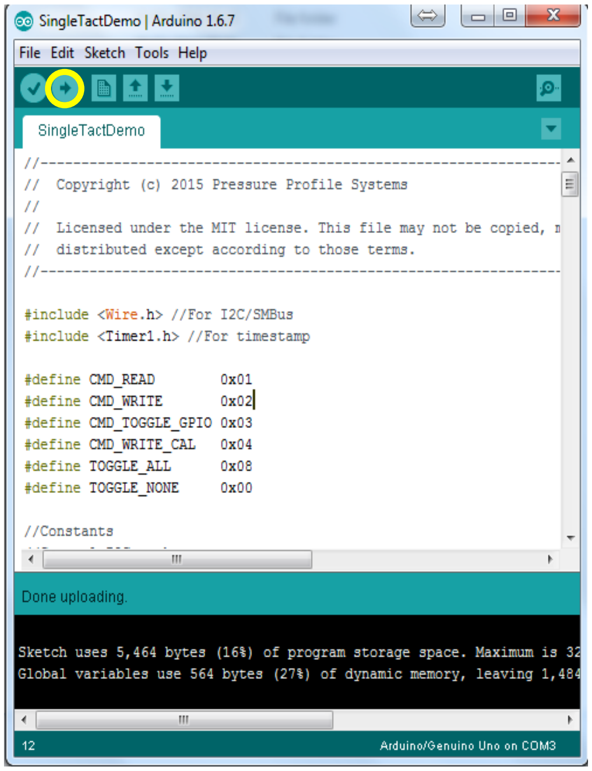
```
//------
//  Copy
//
//  Lice
//  dist
//------

#include <Wire.h> //For I2C/SMBus
#include <Timer1.h> //For timestamp

#define CMD_READ        0x01
#define CMD_WRITE       0x02
#define CMD_TOGGLE_GPIO 0x03
#define CMD_WRITE_CAL   0x04
#define TOGGLE_ALL      0x08
#define TOGGLE_NONE     0x00
```

Figure 13 Compiling and uploading the SingleTactDemo.ino file


SingleTactDemo | Arduino 1.6.7
File Edit Sketch Tools Help
SingleTactDemo
Arduino/Genuino Uno on COM3

```
//------------------------------------------------------------------
//  Copyright (c) 2015 Pressure Profile Systems
//
//  Licensed under the MIT license. This file may not be copied, n
//  distributed except according to those terms.
//------------------------------------------------------------------

#include <Wire.h> //For I2C/SMBus
#include <Timer1.h> //For timestamp

#define CMD_READ        0x01
#define CMD_WRITE       0x02
#define CMD_TOGGLE_GPIO 0x03
#define CMD_WRITE_CAL   0x04
#define TOGGLE_ALL      0x08
#define TOGGLE_NONE     0x00

//Constants
```

```
Done compiling.

Sketch uses 5,464 bytes (16%) of program storage space. Maximum is
Global variables use 564 bytes (27%) of dynamic memory, leaving 1,
```


SingleTactDemo | Arduino 1.6.7
File Edit Sketch Tools Help
SingleTactDemo
12
Arduino/Genuino Uno on COM3

```
//------------------------------------------------------------------
//  Copyright (c) 2015 Pressure Profile Systems
//
//  Licensed under the MIT license. This file may not be copied, n
//  distributed except according to those terms.
//------------------------------------------------------------------

#include <Wire.h> //For I2C/SMBus
#include <Timer1.h> //For timestamp

#define CMD_READ        0x01
#define CMD_WRITE       0x02
#define CMD_TOGGLE_GPIO 0x03
#define CMD_WRITE_CAL   0x04
#define TOGGLE_ALL      0x08
#define TOGGLE_NONE     0x00

//Constants
```

```
Done uploading.

Sketch uses 5,464 bytes (16%) of program storage space. Maximum is 32
Global variables use 564 bytes (27%) of dynamic memory, leaving 1,484
```

4.3 Arduino Demo Outline

The diagrams in this section provide an outline of the Arduino demo functionality as described in the previous section. In this case the PC to Arduino interface was setup to mirror the I^2C interface, keeping the Arduino code as simple as possible.

Figure 14 Arduino example - communication architecture

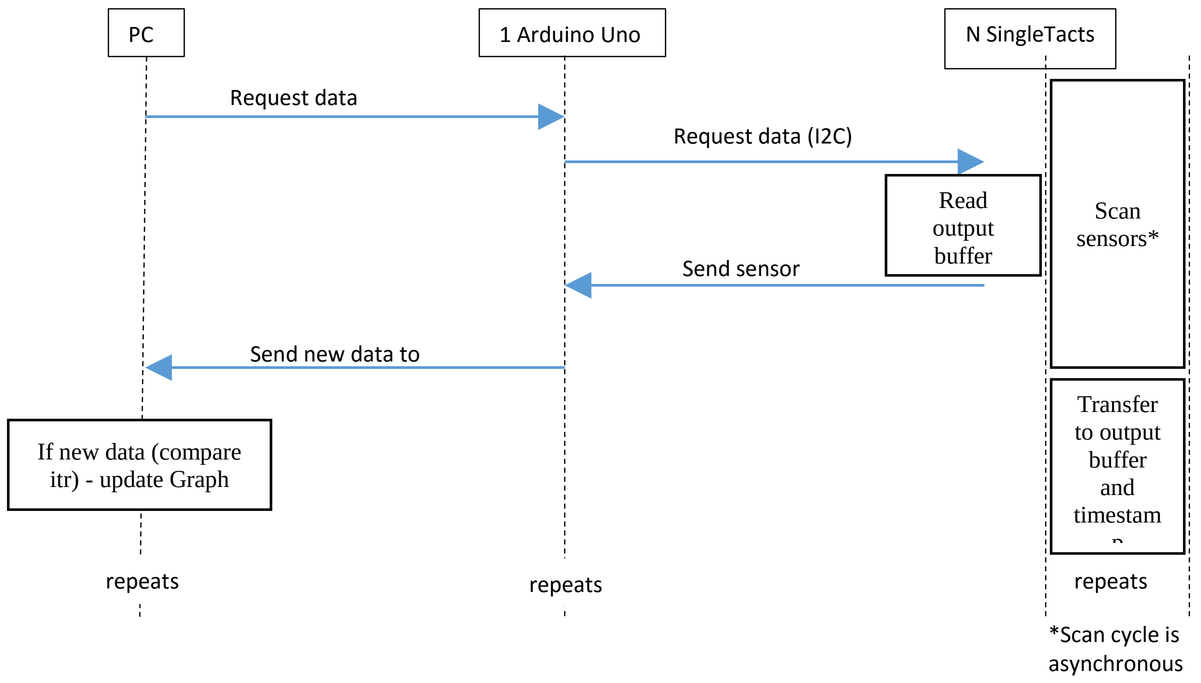
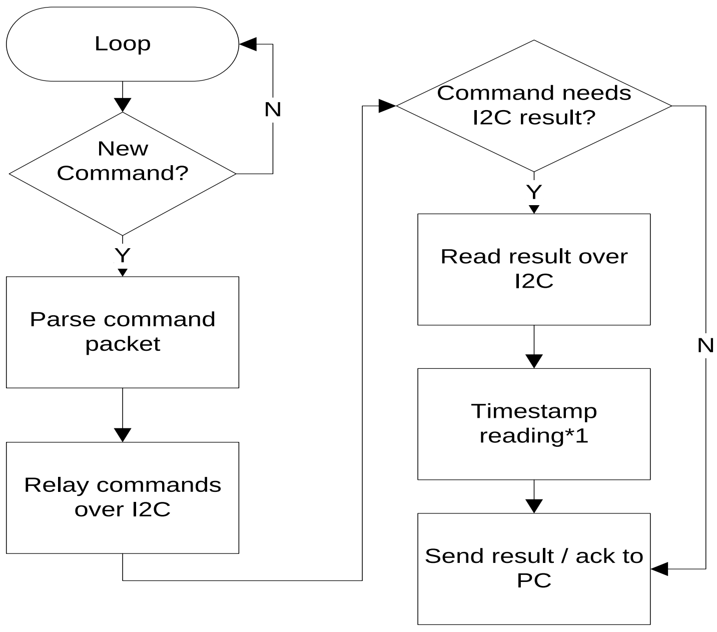


Figure 15 Arduino Application flow



(*1 Note: The Arduino contains a crystal oscillator so it is able to produce a more accurate time stamp than the SingleTact interface board.)

On the host, the Arduino appears as a virtual RS-232 serial device. Data is sent to/from the Arduino using a serial API, such as the one available in .NET.

The Arduino calculates a timestamp for each packet using the Arduino's crystal controlled oscillator. This can be used as the time for each sensor.

The serial commands, which mirror the raw I^{2}C commands (as shown in blue in Figure 16), are outlined in the following tables. Header and footer bytes are added to easily delimit serial packets. A timeout can be specified for I^{2}C transfers.

Figure 16 Serial packet structure (sent to Arduino)

BYTE	FROM PC TO ARDUINO
0	Header = 0xFF
1	Header = 0xFF
2	Header = 0xFF
3	Header = 0xFF
4	I^{2}C address of sensor
5	Timeout (in 100ms increments)
6	ID (echoed in reply)
7	Read (0x01) or Write (0x02)
8	Read/ write location
9	N bytes to read/ write (max 32)
10-> (10 + N-1)	Data to write 0 bytes for read request
11 + N	0xFF – signifies end of packet
12 + N	Footer = 0xFF
13 + N	Footer = 0xFF
14 + N	Footer = 0xFF
15 + N	Footer = 0xFF

Figure 17 Serial packet structure (sent from Arduino)

BYTE	FROM ARDUINO TO PC
0	Header = 0xFF
1	Header = 0xFF
2	Header = 0xFF
3	Header = 0xFF
5	1 if timeout exceeded
6	ID (echoed transmit ID)
7	Timestamp MSB
8	Timestamp
9	Timestamp
10	Timestamp LSB
11	N I^{2}C bytes to be sent (max 32)
12 -> 12+N	I^{2}C data
13+N	Footer = 0xFE
14+N	Footer = 0xFE
15+N	Footer = 0xFE
16+N	Footer = 0xFE

4.4 Example .NET API

This section provides some detail on the .NET API used to construct the PC GUI application.

Download the .NET Interface and demo application from www.singletact.com.

For convenience the low level PC interface is encapsulated in two .NET components.

1. ArduinoSingleTactDriver – The basic Arduino interface. The user must create one of these.
2. SingleTact – There can be multiple SingleTacts each with their own I^{2}C address.

Creating a SingleTact interface is as simple as:

```
arduinoSingleTactDriver.Initialise(COMport);                //Start Arduino driver
singleTact_.I2cAddressForCommunications = 0x04;             //Set I2C address
singleTact_.Initialise(arduinoSingleTactDriver);            //Start sensor
```

The sensor is read using the following:

```
SingleTactFrame newFrame = singleTact_.ReadSensorData(); //Get sensor data
  if (null != newFrame)                                     //If we have data
  {//Process result}
```

Settings can be pulled from the sensor using:

```
singleTact_.PullSettingsFromHardware();
```

and sent to the sensor using:

```
singleTact_.PushSettingsToHardware();
```

<u>**NOTE:**</u> The sensor settings can be modified using commands such as:

```
singleTact_.Settings.ReferenceGain = ###
```

5 Resources

SingleTact homepage
http://www.singletact.com/

I2C-bus specification and user manual VERSION 6, April 2014
http://www.nxp.com/documents/user_manual/UM10204.pdf

Arduino home
https://www.arduino.cc/

Microsoft .NET Framework
https://www.microsoft.com/net

6 Glossary

API	Application Program Interface
CDC	Capacitance to Digital Converter
DAQ	Data Acquisition
FFC	Flexible Flat Cable (connector)
FSR	Full Scale Range
I^2C	Inter IC bus
IDE	Integrated Development Environment
LSB	Least Significant Byte
MSB	Most Significant Byte
.NET	A Microsoft .NET software framework
NVM	Non-Volatile Memory
RS-232	A serial communications standard

7 Revision History

7.1 Revision 2.0

1) Removed Section 3 Updating the Interface Board.
2) Updated Table 1 accessibility to interface board design detail.

7.2 Revision 2.1

1) Added Revision History.
2) Section 1: Referenced Calibrated and Uncalibrated product options.
3) Figure 2: Fixed the link in footnote 3.
4) Table 1: Added I^{2}C Sensor Output range.
5) Table 2: Corrected the addresses of the following parameters:

40	Delimiter – leave as 0xFF
41	Sensor baseline MSB
42	Sensor baseline LSB

Were 39, 40, & 41 respectively.

6) Section 2.3 Removed reference to I^{2}C high speed mode. Updated detail on multiple sensor interfaces sharing a single I^{2}C bus
7) Section 2.4.3: Added detail on I^{2}C Sensor Output data values.
8) Section 2.5: corrected output scale resolution from 12-bit to 10-bit and clarified operating output values.
9) Added Section 2.6 Product Categories.
10) Figure 10: Removed Set Reference Gain control from the GUI image (the Reference Gain Setting is automatic in the current interface board design).