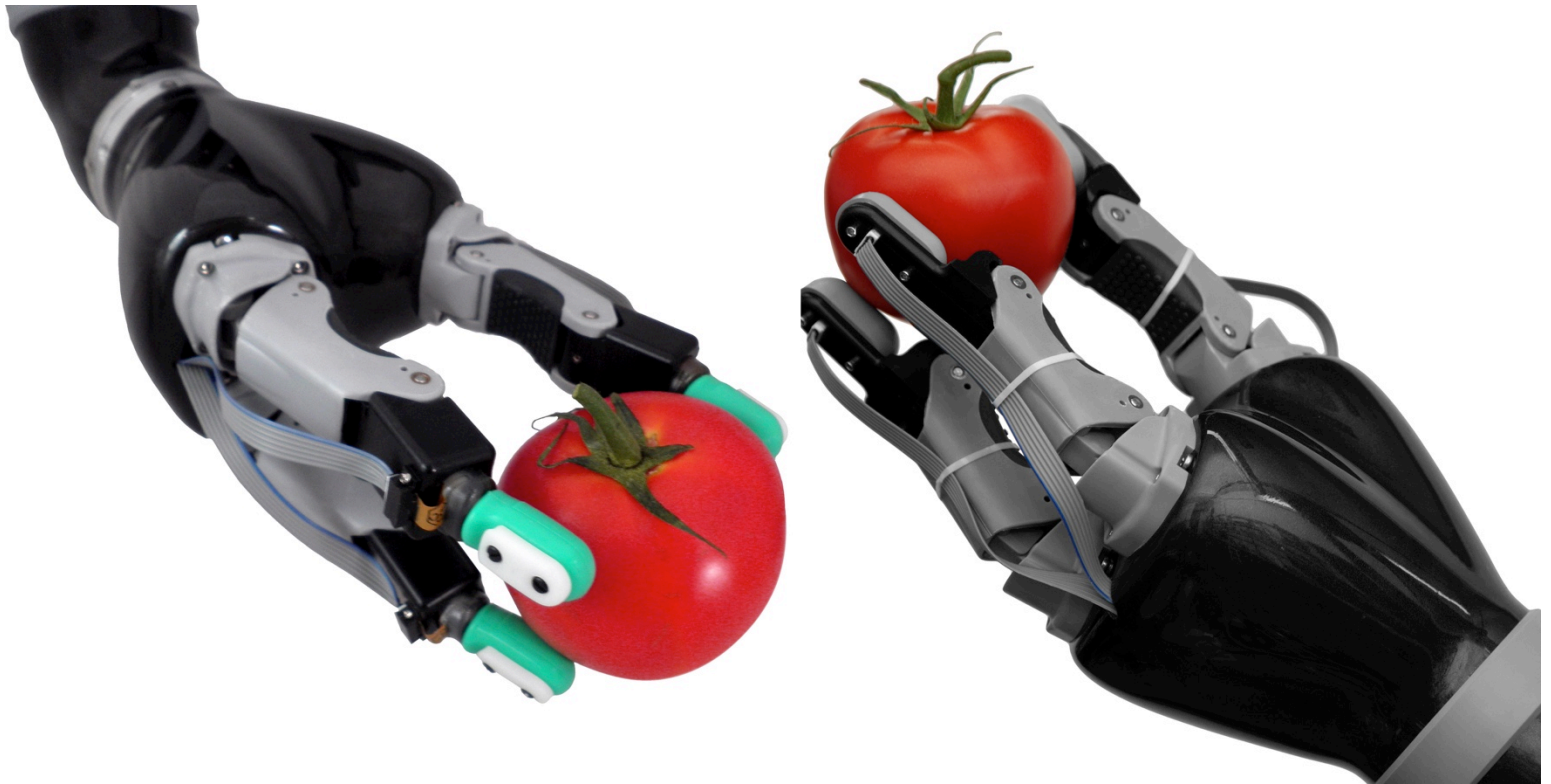


syntouch[®] LLC



BioTac and NumaTac Integration Manual for JACO2 and MICO

Updated: April 21, 2015

Authors: Gary Lin, Raymond Peck & Jeremy Fishel

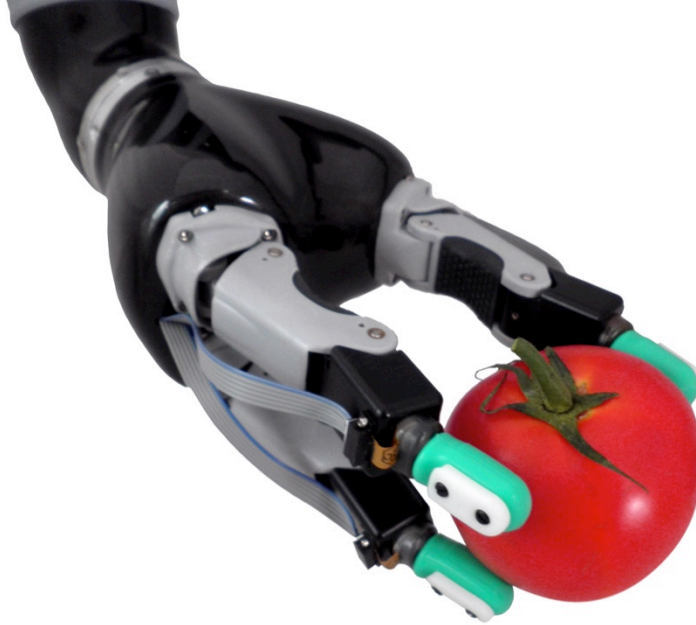
Table of Contents

1	BioTac and NumaTac Tactile Sensors.....	3
1.1	BioTac System for Kinova Robots.....	3
1.2	NumaTac System for Kinova Robots.....	4
2	CAN-USB Hardware	5
2.1	Customer Cable	5
2.2	Peak CAN USB.....	5
3	CAN-USB Software	6
3.1	Pre-request.....	6
3.2	Installation Guide.....	6
3.3	Software Description.....	8
3.4	Functions.....	8
3.4.1	Main function (main).....	8
3.4.2	Save data (bt_save_buffer_data)	9
3.4.3	BioTac main function (bt_main).....	10
3.4.4	Initialize JACO ARM and load its API (initialization).....	11
3.4.5	Go to home position (homePosition)	11
3.4.6	Command finger position (fingerPosition).....	12
3.4.7	Freeze the arm motion (stopMotion).....	12
3.4.8	SynTouch JACO main function (stjaco::main).....	12
4	Removal of JACO-BioTac Mechanical Adapter	14
4.1	Unplug the Connectors	14
4.2	Remove set screw and push pin out.....	14
4.3	Remove black plastic screw.....	15
4.4	Remove adapter.....	15
Appendix A: Troubleshooting Problems		16
Error #1		16
Compiling error:.....		16
Solution:.....		16
Error #2		17
Compiling error:.....		17
Solution:.....		17

1 BioTac and NumaTac Tactile Sensors

SynTouch, the world leader in advanced tactile sensing, in collaboration with Kinova has created two comprehensive sensory systems for the JACO2 and MICO robotic assistive hand. Each system comes with unique capabilities providing the customer options in choosing their sensory system.

1.1 BioTac System for Kinova Robots



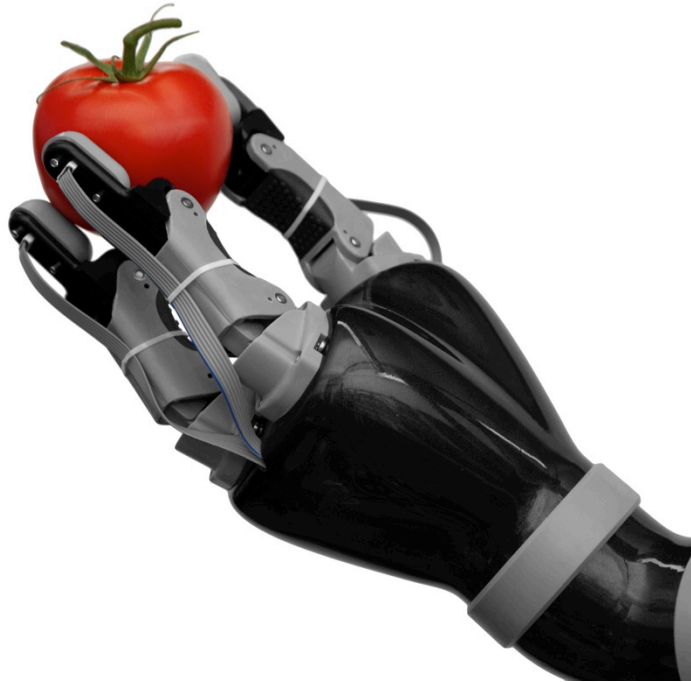
BioTac System for JACO

The BioTac is SynTouch's multimodal human-like tactile sensor. It supports 3 sensory modalities reflective of the full capabilities of human touch: deformation, vibration and temperature. This information can be used to extract shear and normal forces, point of contact, slip and texture, and thermal discrimination of objects. These fluid-filled sensors are exquisitely sensitive to contact and the skins can be easily replaced if worn or damaged.

More information on the BioTac sensor, including product manuals can be found on SynTouch's website:

<http://www.syntouchllc.com/Products/BioTac/>

1.2 NumaTac System for Kinova Robots



NumaTac System for JACO

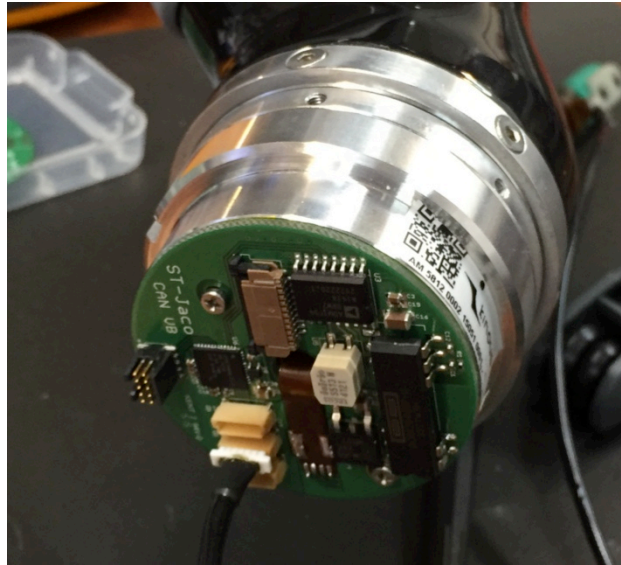
The NumaTac is SynTouch's low-cost foam-based sensor which is reduced in function from the BioTac supporting only the pressure sensing modality. This information can be used to sense initial contact with similar sensitivity to the BioTac and the compliant packaging makes it possible to produce delicate contact reflexes.

More information on the NumaTac sensor, including product manuals can be found on SynTouch's website:

<http://www.syntouchllc.com/Products/NumaTac/>

2 CAN-USB Hardware

SynTouch provides an integrated electronics board for all JACO2 and MICO hands located internally in the wrist of robot. These boards take advantage of the two user lines through the Kinova arm to transmit data via USB to the host computer and are permanently installed. Up to 3 BioTac or NumaTac sensors can be attached to this board.



SynTouch Internal Electronics Board

2.1 Customer Cable

A modified version of the original JACO cable is provided to attach to the base of the arm and receive these data lines with the following pin map:

Pin 2 CANL

Pin 3 GND

Pin 7 CANH

A 120 Ohm resistor was connected between CANH and CANL according to CAN Bus Protocol.

2.2 Peak CAN USB

To interpret CAN data in the host computer, SynTouch recommends the Peak CAN USB adapter.

Product information: <http://www.peak-system.com/PCAN-USB.199.0.html?&L=1>

3 CAN-USB Software

3.1 Pre-request

- **Peak CAN Driver**

Please download and install from

<http://www.peak-system.com/fileadmin/media/files/usb.zip>

Or from product information website

<http://www.peak-system.com/PCAN-USB.199.0.html?&L=1>

- **JACO Driver**

Please download JACO2 or MICO Research Edition SDK from either:

<http://kinovarobotics.com/downloads/JACO2-SDK.zip>

<http://kinovarobotics.com/downloads/MICO-SDK.zip>

<http://kinovarobotics.com/support/>

- **Microsoft Visual C++ 2010**

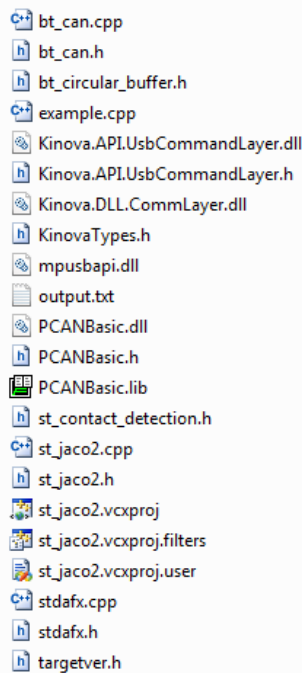
- **Boost C++ Library**

The BioTac Demo software needs boost C++ Library. Install the boost C++ library from

<http://www.boost.org/> (Version: 1.55.0)

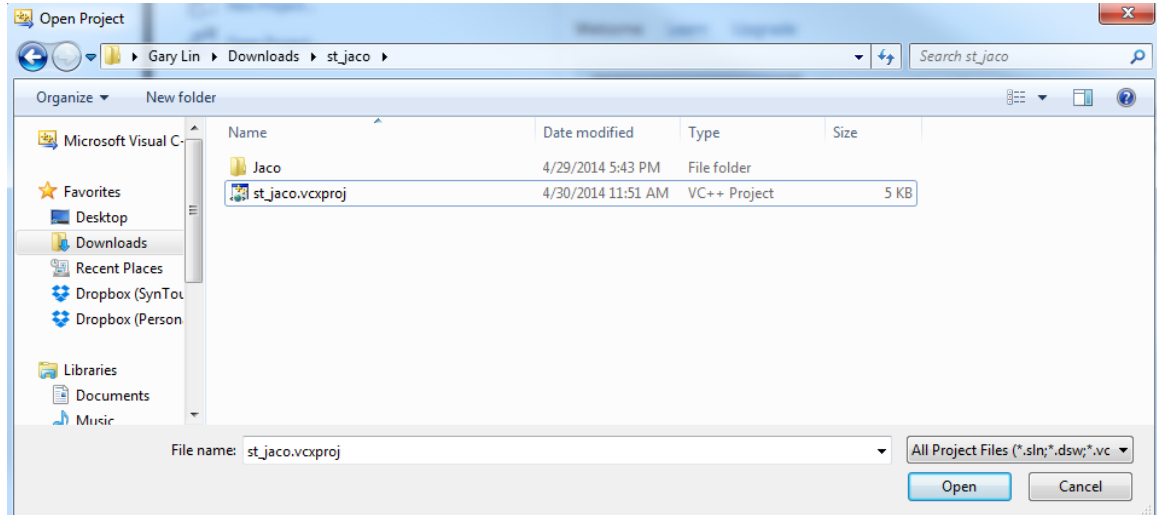
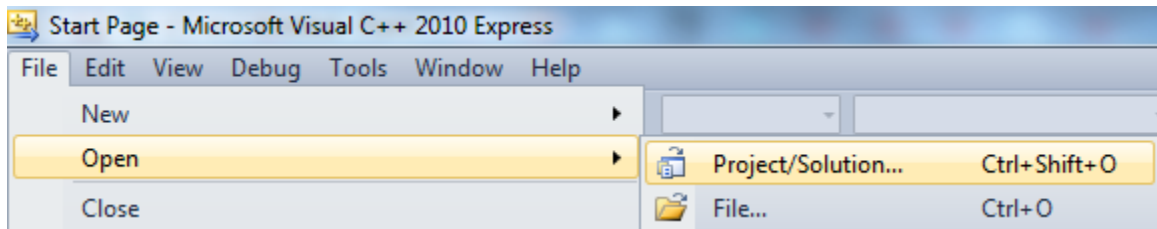
3.2 Installation Guide

1. Download and unzip the file “st_jaco2.zip” to desired location

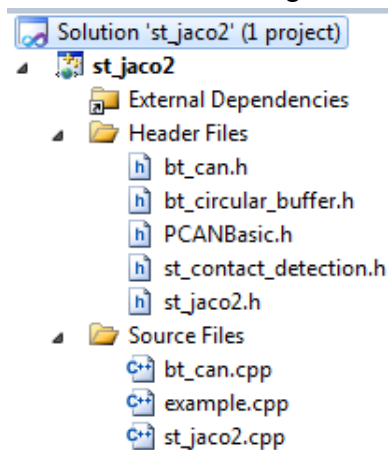


bt_can.cpp
bt_can.h
bt_circular_buffer.h
example.cpp
Kinova.API.UsbCommandLayer.dll
Kinova.API.UsbCommandLayer.h
Kinova.DLL.CommLayer.dll
KinovaTypes.h
mpusbapi.dll
output.txt
PCANBasic.dll
PCANBasic.h
PCANBasic.lib
st_contact_detection.h
st_jaco2.cpp
st_jaco2.h
st_jaco2.vcxproj
st_jaco2.vcxproj.filters
st_jaco2.vcxproj.user
stdafx.cpp
stdafx.h
targetver.h

2. Open Microsoft Visual C++ and open the object file “st_jaco2.vcxproj”



3. We will see following files in the project



4. Add boost library path into the project environment (see Appendix)
5. Compile and run example demo
6. Use joystick to move the hand to proper position and hit key “c” on keyboard to close the hand.

-
- **Please do NOT press any key while the arm is controlled by the joystick. According to JACO API documentation, the commands from joystick have higher priority than those sending from PC. So the example API program cannot control the arm when it is manipulated by joystick.**
-

7. Hit key “c” again to toggle the hand open/closing function to open the hand.
8. Hit key “r” to reset the position memory.
9. Use joystick to move the hand position to next desired position and key “c” to grasp another object.

3.3 Software Description

Key “c” is used to control finger open/close. The program will remember the position of the arm while key “c” is been pressed.

WARNING: After key “c” being entered, if the arm position has been modified by joystick

Key “q” save the circular buffer to the same folder “st_jaco” and quits the program

Key “r” resets the position memory

3.4 Functions

In example.cpp

3.4.1 Main function (main)

```
int main( int argc,  
          const char* argv[] );
```

Details

- In this function, it creates two threads for running the bt_main and stjaco::main programs in parallel.
- It listens to any new data in the shared memory pdc_shared_data and executes the contact detection algorithm.
- It listens to any value change from the keyboard. If the user presses any key, it passes the pressed key information to two shared memories (biotac_keyboard and jaco_keyboard).

In `bt_can.cpp`

3.4.2 Save data (`bt_save_buffer_data`)

```
void bt_save_buffer_data(  const char *file_name,
                           bt_circular_buffer<bt_data>*data_save,
                           int num_samples);
```

Save BioTac data which is in the circular buffer to file, including batch, frame, and channel ids, raw values, parity check, etc.

Arguments

<code>file_name</code>	pointer of the string of file name such as “output.txt”
<code>data_save</code>	pointer to access the circular buffer
<code>num_samples</code>	numbers of sampling data saved into the file

Return Value

This function does not return any value.

Details

This function saves data in a text file whose name is `file_name`. By default, a saved data format is as follows:

```
time,  batch_index,  frame_index,  channel_id,  value[0],
bt_parity[0],      value[1],  bt_parity[1],  value[2],
bt_parity[2]
```

Note:

`time`: the relative time to the very first sample after the power is on

`batch_index`: in this project a batch contains one frame, so the batch index is equal to the frame index

`frame_index`: in this project a frame contains one sample, so the frame index increases one for every sample

`channel_id`: is saved as integer (0-3, 15-35) by default

`value[0], value[1], value[2]`: BioTac data from three fingers

`bt_parity[0]`, `bt_parity[1]`, `bt_parity[2]`: The parity check for each BioTac data. Logic 0 means good parity check and data is valid. Logic 1 means bad parity and data is invalid.

3.4.3 BioTac main function (`bt_main`)

```
int bt_main(void);
```

Arguments

None

Return Value

Return 0 if it is successfully terminated

Return 1 if there is any error related to Peak CAN-USB

Details

- This is the main function to handle all the functions that initialize the settings related to Peak USB-CAN device and listen to BioTac samples and collect data in a circular buffer.
- In this `bt_main` function, it pushes the sensing data (by default, PDC channel) to a shared memory (`pdc_shared_data`) for contact detection purpose. The following is the code:

```
// Example of using PDC data for contact detection
if(data.channel_id == CD_CHANNEL)
{
    // Lock the data first to prevent data accessing conflicting
    boost::unique_lock<boost::mutex> pdc_lock(pdc_shared_data_mutex);
    pdc_shared_data.push(data);
}
```

The sensing channel can be modified through the variable `CD_CHANNEL` in `st_contact_detection.h` file. See the Appendix for all the possible channels.

- In this `bt_main` function, it listens to any changes on the keyboard. If it detects the key “q” pressed by user, it will save all the collected data in circular buffer to the file (please refer to the save data function), terminates the Peak CAN-USB connection, and exit from the `bt_main` function.

In st_jaco.cpp

3.4.4 Initialize JACO ARM and load its API (initialization)

```
bool stjaco::initialization(void);
```

Loads Kinova API and initialize function pointers.

Arguments

None

Return Value








Return true (1) if library loaded and pointers created without any problem

Return false (0) if there is any error loading the library or creating function pointers

Details

For detail information for the Kinova API, please refer to Kinova programming guide.

Please make sure the following files are under the subfolder "st_jaco2\Jaco2"

 Kinova.API.UsbCommandLayer.h	2/27/2014 4:48 PM	C/C++ Header	8 KB
 Kinova.DLL.CommLayer.dll	3/6/2014 11:04 AM	Application extens...	31 KB
 KinovaTypes.h	2/27/2014 4:30 PM	C/C++ Header	62 KB
 mpusbapi.dll	1/17/2013 1:12 PM	Application extens...	59 KB
 stdafx.cpp	12/2/2013 7:59 AM	C++ Source	1 KB
 stdafx.h	3/4/2014 2:15 PM	C/C++ Header	1 KB
 targetver.h	3/4/2014 1:46 PM	C/C++ Header	1 KB

3.4.5 Go to home position (homePosition)

```
void stjaco::homePosition(void);
```

Change the robot arm to position control and make it to go back to home position.

Arguments

None

Return Value

This function does not return any value.

3.4.6 Command finger position (fingerPosition)

```
void stjaco::fingerPosition(int finger_status);
```

Setup the goal position of the finger depending on the input argument.

Arguments

finger_status indicate the goal position of the finger is open or close. 0: finger open. 1: finger close

Return Value

This function does not return any value.

3.4.7 Freeze the arm motion (stopMotion)

```
void stjaco::stopMotion(void);
```

Stop the motion of the arm.

Arguments

None

Return Value

This function does not return any value.

Details

This function stops the motion of the arm, waits 200 millisecond for arm stabilization, and records the arm Cartesian position.

3.4.8 SynTouch JACO main function (stjaco::main)

```
int stjaco::main(void);
```

Initialize the JACO Arm related program and control its movement depending on the joystick and keyboard input.

Arguments

None

Details

- This function calls the initialization function for JACO Arm and load its API
- It checks the contact detection shared memory (cd_shared_data) and stops three finger movements individually if a contact event has happened to the finger.
- It monitors any change on the key board:
Key “c” is used to control finger open/close. The program will remember the position of the arm while key “c” is been pressed.
Key “r” resets the position memory
Key “q” stops the motion of the arm, close the Kinova API, and exit the main function.

WARNING: After key “c” is entered at time 1, if the arm position is modified by joystick and “c” is pressed again, the arm position will move back to the original position where “c” was pressed at time 1.

In bt_circular_buffer.h

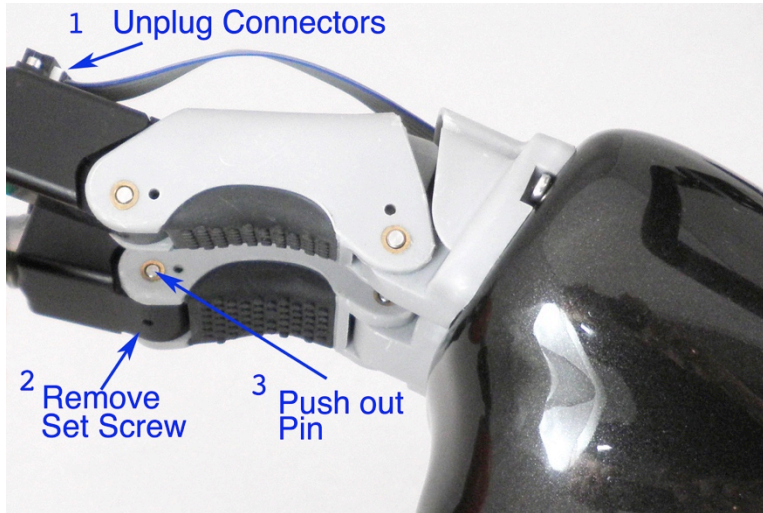
Please do not modify any code here except `BUFFER_SIZE` . The constant `BUFFER_SIZE` is used to setup the size of the circular buffer. By default, `BUFFER_SIZE = 132000` and can store about forty (40) seconds of data. If you prefer a bigger size of circular buffer, you can increase its size, there are about 3500 samples per second.

4 Removal of JACO-BioTac Mechanical Adapter

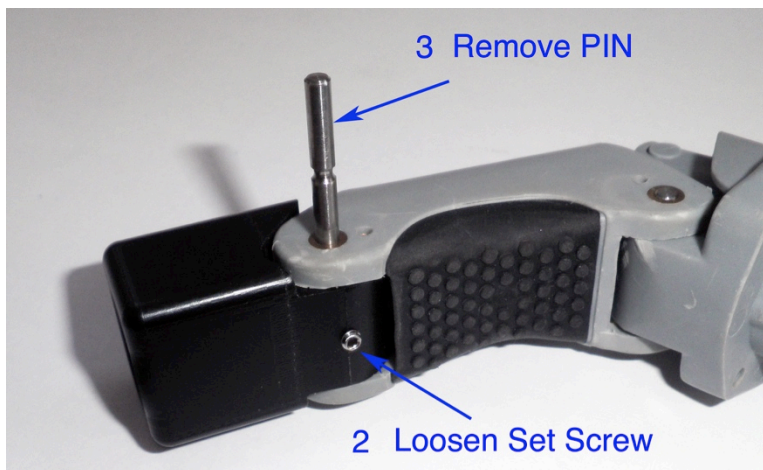
This section describes the steps needed to remove the JACO-BioTac Mechanical Adapter. The NumaTac adapter has a similar attachment procedure.

4.1 Unplug the Connectors

➤ *Warning: unplug connectors carefully to avoid damage to the BioTac*



4.2 Remove set screw and push pin out.

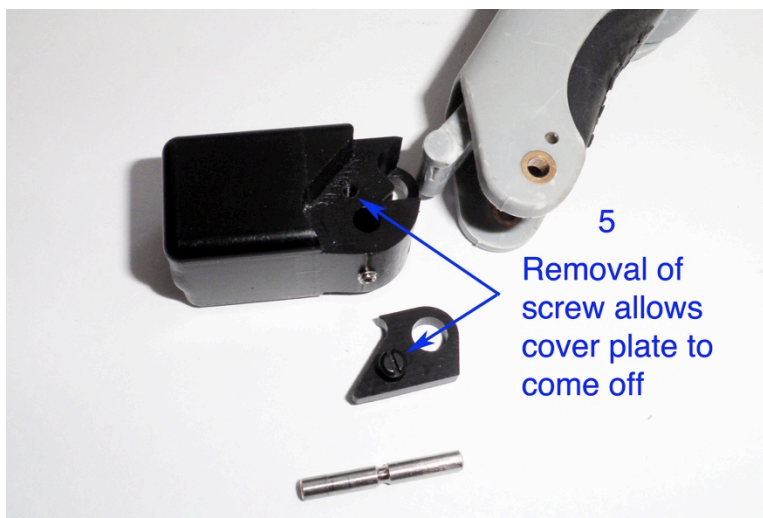


4.3 Remove black plastic screw



4.4 Remove adapter

Once screw is removed, cover plate will come off to allow adapter to separate



Appendix A: Troubleshooting Problems

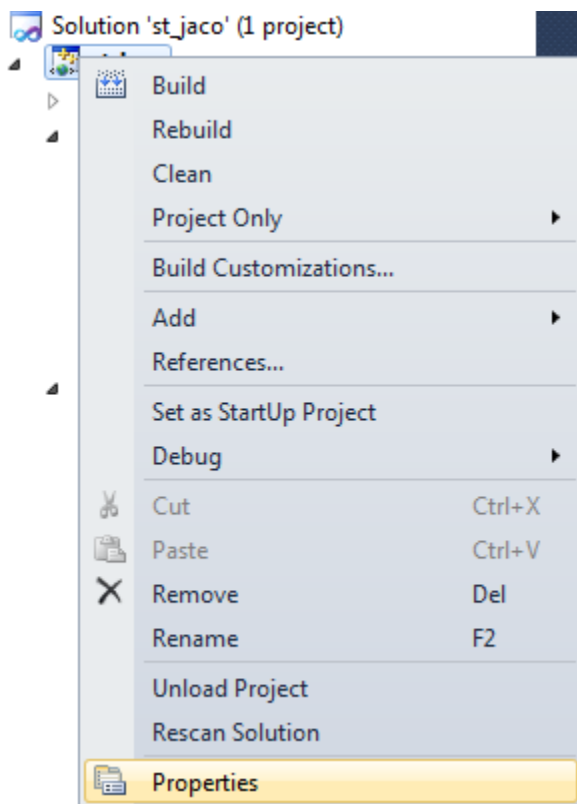
Error #1

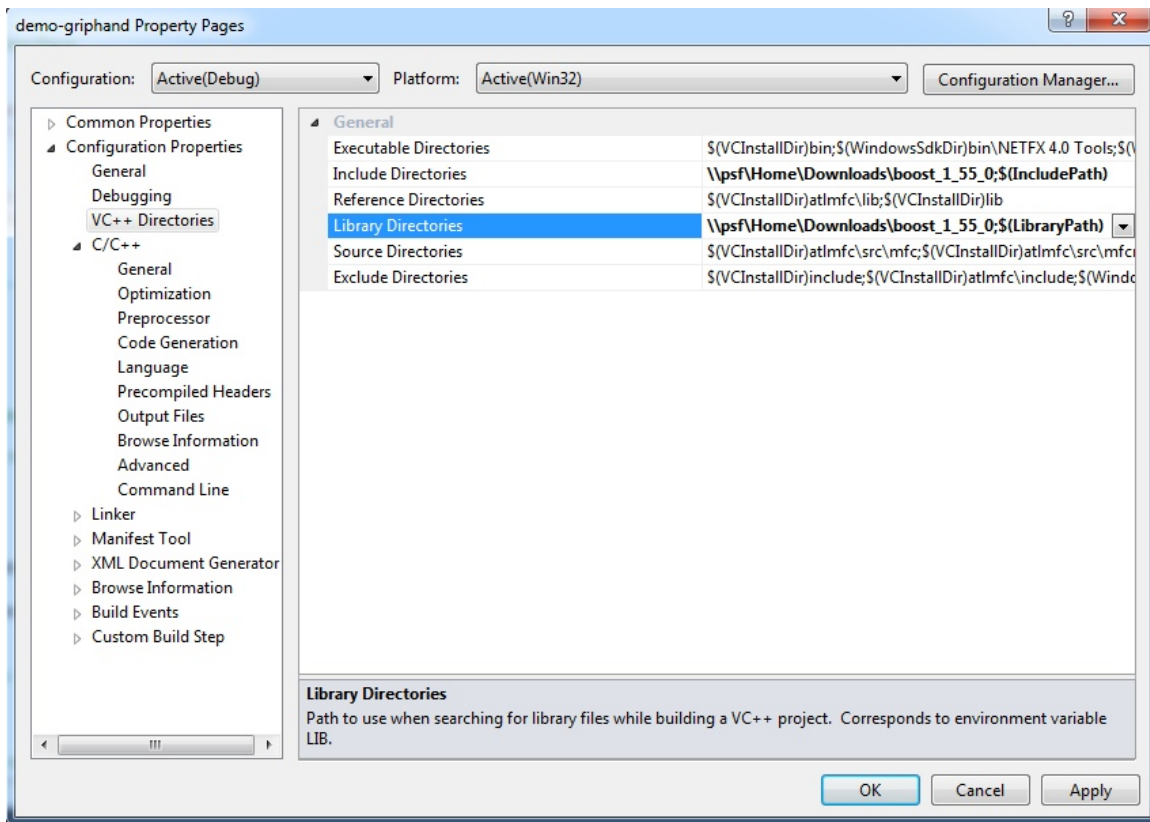
Compiling error:

fatal error C1083: Cannot open include file: 'boost/thread.hpp': No such file or directory

Solution:

Include the library path (\Downloads\boost_1_55_0) into the “VC++ Directories” under “project property”





Error #2

Compiling error:

fatal error LNK1104: cannot open file 'libboost_thread-vc100-mt-gd-1_55.lib'

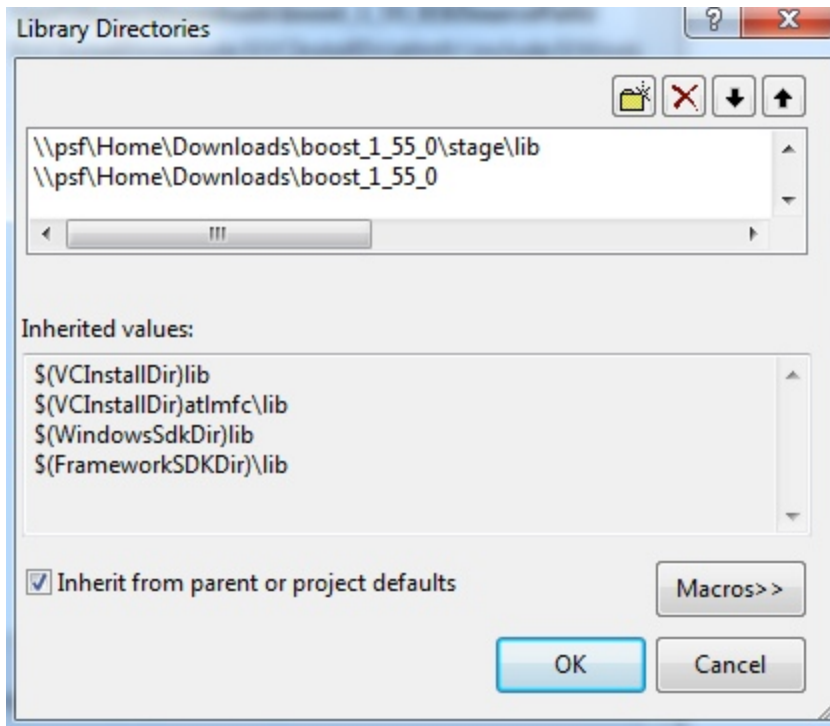
Solution:

Chapter 5.1 in http://www.boost.org/doc/libs/1_55_0/more/getting_started/windows.html

If you wish to build from source with Visual C++, you can use a simple build procedure described in this section. Open the command prompt and change your current directory to the Boost root directory. Then, type the following commands:

```
bootstrap
.\b2
```

The first command prepares the Boost.Build system for use. The second command invokes Boost.Build to build the separately-compiled Boost libraries. Please consult the [Boost.Build documentation](#) for a list of allowed options. This will compile to generate the **stage** folder



In *st_contact_detection.h* file, the magic number CD_CHANNEL can be modified to the following constant number:

```
BT_PAC_SAMPLING
BT_PDC_SAMPLING
BT_TAC_SAMPLING
BT_TDC_SAMPLING
BT_E01_SAMPLING
BT_E02_SAMPLING
BT_E03_SAMPLING
BT_E04_SAMPLING
BT_E05_SAMPLING
BT_E06_SAMPLING
BT_E07_SAMPLING
BT_E08_SAMPLING
BT_E09_SAMPLING
BT_E10_SAMPLING
BT_E11_SAMPLING
BT_E12_SAMPLING
BT_E13_SAMPLING
BT_E14_SAMPLING
BT_E15_SAMPLING
```

BT_E16_SAMPLING

BT_E17_SAMPLING

BT_E18_SAMPLING

BT_E19_SAMPLING

➤ **NumaTac sensors only support sampling of BT_PAC_SAMPLING and BT_PDC_SAMPLING, all other channels for the NumaTac will return a value of 2635, which is an error code for unsupported channel.**

CONTACT

SynTouch LLC
2222 S.Figueroa PH2
Los Angeles, CA 90007

213.973.4102
info@syntouchllc.com

www.syntouchllc.com