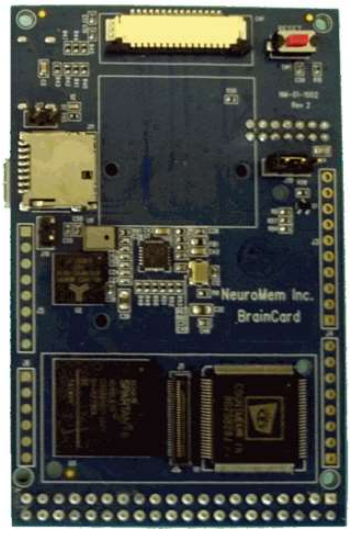
BrainCard,
Low-power, trainable pattern recognition for IoT



Version 1.3
Revised 08/08/2016

GENERAL VISION

BrainCard is a product of General Vision, Inc. (GV)

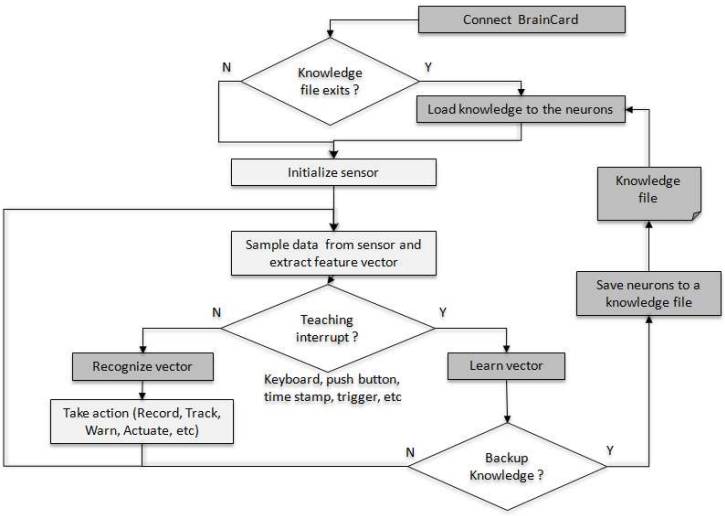
Contents

1 Introduction

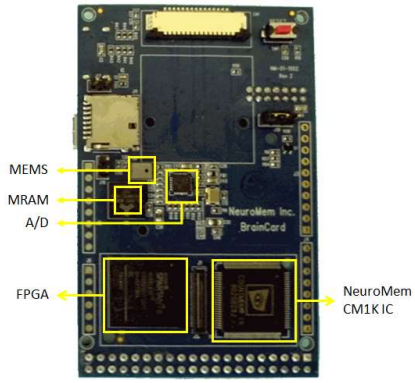
BrainCard is a trainable pattern recognition board for IoT and smart appliances featuring the NeuroMem CM1K chip with 1024 neurons and which can interface to Arduino, Raspberry Pi or Intel® Edison module.

Typical workflow

The data collected through the processor board (or through the FPGA of the Braincard if programmed adequately) can be broadcasted to the neurons for learning and recognition. Learning can be triggered by external user inputs, time stamps, but also the ability of the neurons to detect drifts from learned models. Learning can also be done “off line” on data previously collected and saved. Recognition can consist of using the neurons to classify input patterns or identifying novel or abnormal patterns. Depending on your application, the output of the neurons can control actuators, trigger a selective recording or transmission or else. Applications include identification, surveillance, tracking, adaptive control and more.



Hardware overview



The NeuroMem CM1K is a trainable neural network which can learn and recognize examples in real-time. For more information of the NeuroMem technology, please refer to the NeuroMem Technology Reference Guide.

The Field Programmable Gate Array of the BrainCard is a configurable switchbox which is programmed to act as the glue between an external processor like the Arduino or Raspberry PI and the neurons, or between a sensor and the neurons.

Other components:

- Audio MEMS
- A/D converter
- 16 MB SDRAM
- Flash (dedicated to the FPGA configuration)

Connectivity

- Arduino connectors. Only supports Arduino boards with 3.3v IOs.
- Raspberry Pi connector
- Intel® Edison connector (and dedicated USB connector)
- SD card slot
- Expansion connector to stack additional CM1K chips (up to 9,216 neurons per 1024 increment)
- Connector compatible with the RaspiCam camera module
- Mini HDMI connector

2 Getting Started

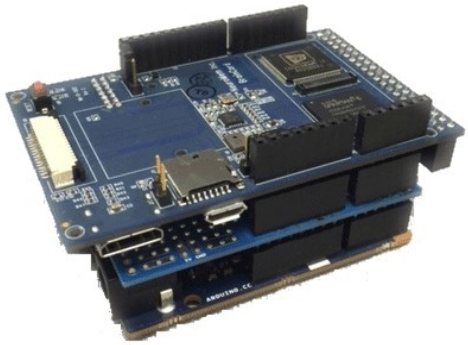
BrainCard as a neuromorphic Shield…

The Braincard is powered through the USB port or the Arduino power connector (J5). In term of IOs, the BrainCard only accepts 3.3V I/O on the Arduino connectors J 3 and J4.

We have tested the BrainCard as a shield with the following boards:

Hardware Platform	Arduino platform	Rapsberry PI platform
IDE Platform	Arduino IDE	Python
Boards	ADAFruit Metro Intel Arduino/Genuino 101 Kocoafab Orange board Intel Edison Arduino breakout board (1) Intel Edison module	Raspberry PI 2 and 2

Depending on the base board you select to interface to the BrainCard, you may have to use spacers or in-between shields to ensure that the BrainCard is properly plugged in and powered. In the example below, the heights of the BrainCard's PMOD and the Arduino101 USB connector type A make it impossible to properly assemble the 2 boards without spacers or a dummy proto shield.



(1) In this first revision of the BrainCard, it is not possible to plug the Intel Edison module directly into the J2 connector on the back of the BrainCard. We still have to fix some instability issues between the Edison and the FPGA of the BrainCard at the power up. However, the problem is non-existent if you plug into J2 the SparkFun block for Intel Edison base populated with the Intel Edison module. Note that depending on the presence or not of Arduino the connectors J5-J6 (or remaining solder balls), you may have to add an in-between block with pass-thru hirose connector to make sure the Edison base block is plugged properly.

BrainCard API

Go to http://www.general-vision.com/BrainCardBSP and retrieve the Board Support Package which contains libraries and examples for various programming IDE.

The functions of the BrainCardNeurons library are described in the technical manual TM_NeuroMem_API.pdf.

A series of short and academic test programs are supplied for the Arduino IDE and also for the Raspberry PI. In addition to the short descriptions below, detailed comments are included in the source code and print statements executed by the code.

Test_SimpleScript

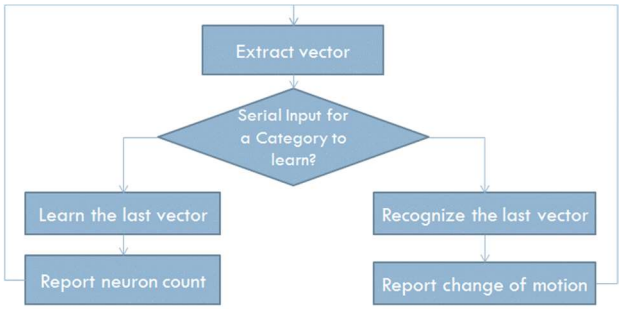
Simple script stimulating the neurons to learn and recognize patterns generated programmatically. Detailed description of this script is available at http://www.general-vision.com/documentation/TM_TestNeurons_SimpleScript.pdf.

Test_SimpleScript2

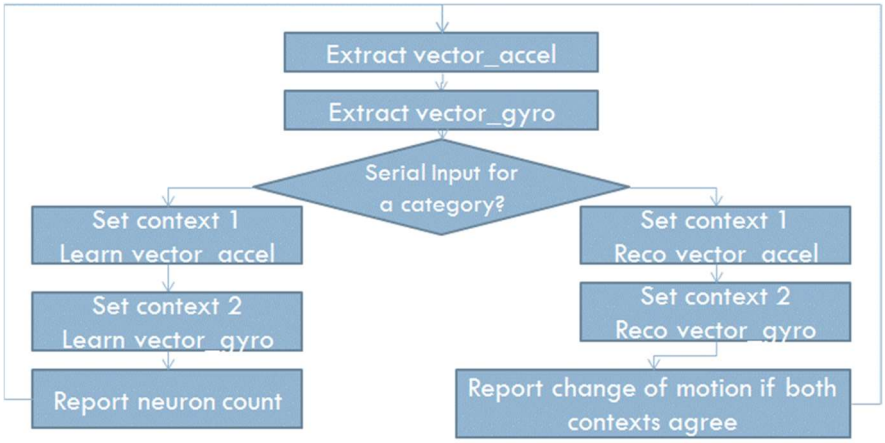
Simple script stimulating the neurons to learn and recognize patterns generated programmatically. Shows the difference between a recognition executed when the neurons are set to KNN mode versus RBF mode. KNN always gives a response for the closest match while RBF can report cases of "unknown" and cases of uncertainties. Detailed description of this script is available at http://www.general-vision.com/documentation/TM_TestNeurons_SimpleScript.pdf.

Test_Neuron_andIMU version 1 and 2

This script is intended for use with the BrainCard plugged on an Intel Arduino/Genuino101 to interface with its 6-axis IMU. The script assembles signals from the accelerometer and gyroscope into a simple feature vector broadcasted continuously to the neurons for recognition. Learning is performed by entering a category value through the serial input. Refer to the General Vision movie tutorial.



Test_Neurons_andIMU2 is similar except that it illustrates the ability of the NeuroMem neurons to handle multiple networks in a same chip for more robust decision making. The signals of the accelerometers and gyroscope are assembled into two separate feature vectors and associated to 2 different contexts. Learning a motion builds 2 decision spaces at once and consequently commits more neurons. The script displays a positive response only if both sub-networks agree with the classification, thus producing a more conservative but accurate response than in the script Test_Neurons_andIMU.



Remark1: This example is very academic and assemble a pattern which should be more sophisticated for real-life system taking a calibration into account, integrating a sampling rate adequate for the type of motion and profiling the waveforms more selectively using distances between peaks and zero crossing, etc.

Remark2: This example does not use the NeuroMem neurons available on the Intel Arduino/Genuino101, but solely the NeuroMem neurons of the CM1K chip. It could be modified to use both sets of neurons for different purposes.

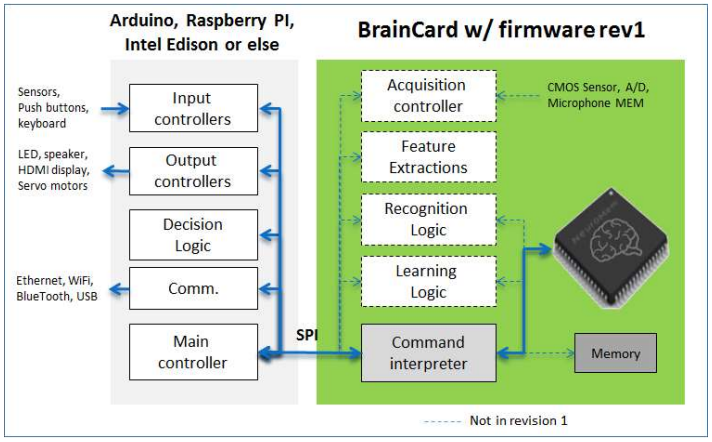
TestSPIComm

This script is supplied to help verify that board is properly initialized and that the SPI communication with the board is stable.

3 BrainCard Configuration

FPGA default configuration

Our initial firmware version 1 is limited to a simple communication protocol between your microcontroller and the CM1K. This enables the use of the NeuroMem neurons to learn and recognize patterns generated by or collected through a shield by your microcontroller. Examples supplied for the Arduino IDE manipulate numeric data as well as data coming from accelerometers. Check http://general-vision.com/braincardbsp/ for links to latest videos and tutorials.



Communication controller

Communication with the FPGA is presently established through an SPI protocol described in a next chapter. Functions performing the SPI_Read and SPI_Write between the BrainCard and the Arduino and Raspberry PI platforms are included in the BrainCard API.

Updating the FPGA configuration

At initialization of the BrainCard, the FPGA looks for a valid user configuration file in sector 8 of its Flash memory. If none is found, it loads the default configuration residing in sector 0.

Updating or upgrading the FPGA configuration is done very simply by copying your new configuration file (bc_top.bin) to an SD Card, running the FPGA_LoadConfig script and re-booting the Braincard.

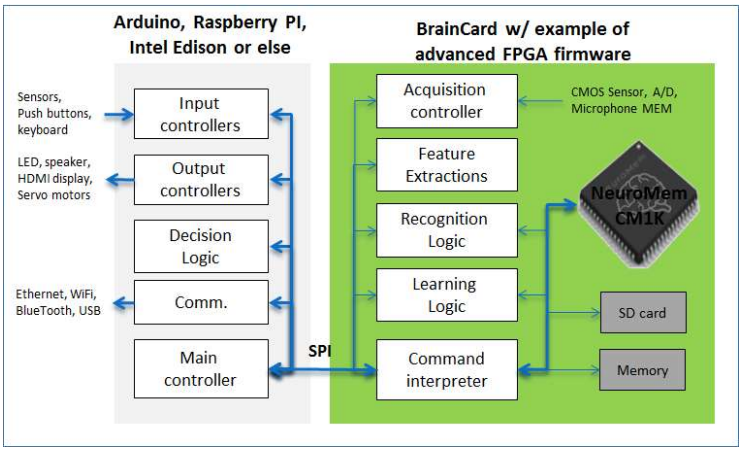
The Arduino examples contains two scripts to help you configure the FPGA of the BrainCard:

- FPGA_LoadConfig; to load a new configuration file from an SD card to the sector 8 of the Flash memory. The file must be saved on the SD card as "bc_top.bin".
- FPGA_ResetConfig; to erase any user configuration file starting at sector 8 of the Flash memory

Flash dedicated to the FPGA (Micron 25P80):

0x100000 0x080000	bc_top
0x7FFFFF 0x00000	bc_default Flag Dual boot or Single boot

Future revision



Communication controllers

- SPI controller
- I2C controller
- Etc.

Data acquisition examples

- Signal from A/D or microphone: Accumulate data in on-board memory with a user-defined sampling rate and buffer length.
- Images: Store single or multiple frames to on-board memory or SD card, control gain, shutter speed and trigger via I2C control command

Feature Extractions examples

- Multiple choices per input data type
- Images: color or monochrome subsamples, color and intensity histograms, SURFs, HOGs, etc.
- Signal: raw sampling, power spectrum, wavelet

Learning Logic examples

- Multi-modal (multiple sensor inputs sampled at the same time)
- Multi-scale, Multi-feature (extracted from the same input signal to build redundancy)
- Reinforced learning based on the recognition in previous samples

Recognition Logic examples

- Limited to the closest match
- Extended to the first N best match to build more robust decision
- Aggregate recognition derived from multiple sensors, multiple scales and multiple features at a same sampling time

Decision Logic examples

- Resizing and repositioning of the region or window of search based on the current recognition
- Switch sensor, switch feature extraction
- Rule based actuation/decision
- Pattern recognition based actuation/decision

4 SPI Protocol

Communication between the BrainCard and the Arduino and Raspberry PI platforms is presently established through an SPI protocol described below. Functions performing the SPI_Read and SPI_Write are included in the BrainCard API.

The SPI protocol is based on a 10-bytes control command described below and intended to access the various components of the board including the NeuroMem CM1K chip, the memory, SD card, A/D converter, CMOS sensor, and all.

Byte command sequence:

- Byte0 Reserved
- Byte 1-2-3-4 R/W bit + 31-bit address
- Byte 5-6-7 24-bit Data length expressed in number of words
- Byte 8-9 Minimum first 2 bytes of data
- Byte + Remaining data (up to Data Length)

Programming examples of the Read and Write functions are supplied in C++, Python and Arduino. We use the following convention in our examples, but it is optional:

- Read_Addr(long **Address**, long **Length**, int[] **Data**) if data length is greater than a single word
- Write_Addr(long **Address**, long **Length**, int[] **Data**) if data length is greater than a single word
- Read(byte **ModuleID**, byte **Register**, int **Data**)
- if data length is a single word, to read a single 16-bit register
- Write(byte **ModuleID**, byte **Register**, int **Data**) if data length is a single word, to write a single 16-bit register

The full memory map of the BrainCard will be published shortly, but is presently limited to access to the on-board CM1K chip or a chain of CM1K chips.

Address Range	**Module=** Address[30-24]	Functionality defined by registers = Address[23:8]
0x01000000 0x0100001F	NeuroMem CM1K 0x01 (d01)	Access to the registers of the CM1K chips whether limited to a single chip or a chain of N chips if NM2K extension modules are plugged on J1.
To be published	Board settings	Access to the settings of the board
To be published	SDRAM	Access to the bank of SDRAM of the board connected to the host.
To be published	CMOS	Access to the camera module plugged on CM1
To be published	AD	Access to the A/D converter SPI lines
To be published	Microphone	Access to the microphone
To be published	SD	Access to the SD card
To be published	PMOD	Access to the PMOD A and B

Write command

Reserved	Address[31:0]			Data length[23:0]	Data
0x00	Bit 31=1	Module[6:0]	Register[24:0]	Length in words	Input array
1 byte	1 byte		3 bytes	3 bytes	Data length * 2 bytes

Example of Single Write

Write 0x33AA to the MAXIF (register 7) of the CM1K (module 1) 0x00 81 00 00 07 00 00 01 33 AA

Multiple Write

Write 4 consecutive byte components [11,12,13,14] of a neuron 0x00 81 00 00 01 00 00 02 11 12 13 14

Read command

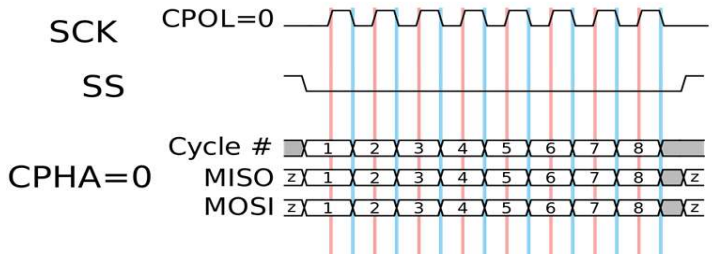
Reserved	Address[31:0]			Data length[23:0]	Data
0x00	Bit 31=0	Module[6:0]	Register[24:0]	Length in words	Output array
1 byte	1 byte		3 bytes	3 bytes	Data length *2 bytes

Single Read
Read the MINIF register 6 of the CM1K (module 1) 0x00 01 00 00 06 00 00 01
Data is returned into 2 bytes or a word

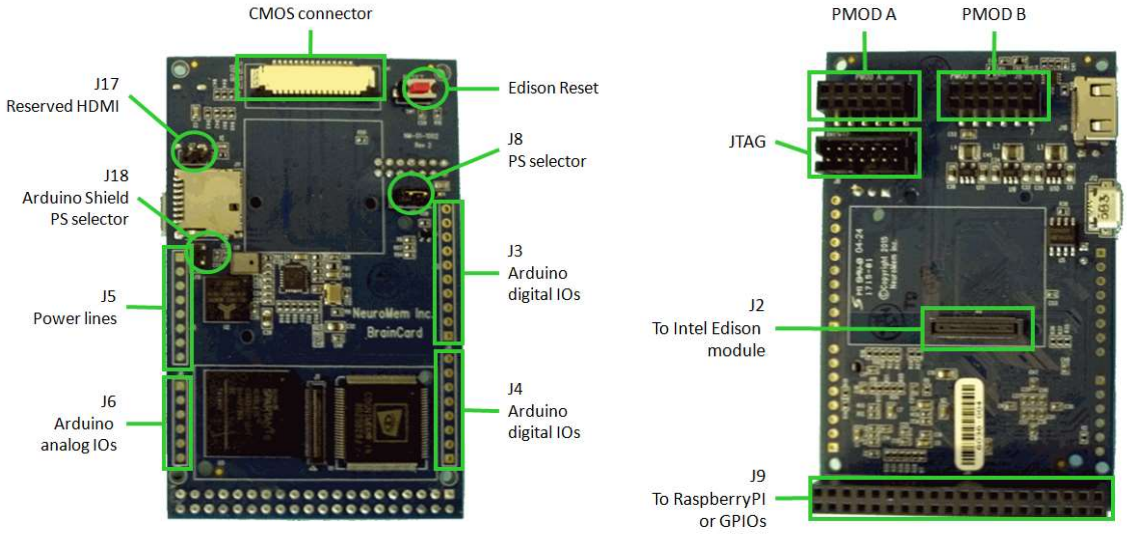
Multiple Read
Read 8 consecutive byte components of a neuron 0x00 01 00 00 01 00 00 04
Data is returned into 8 bytes.

Timing Specifications

CPOL= 0 -> clock idle a logic level 0
CPHA= 0 -> Data sampled on rising edge of SCK
Recommended SCK frequency <= 10 MHz
General CSn Data pin #10 .
FPGA reconfiguration CSn= data pin #8



5 Connectors and Pin out



Jumper J1
Connector for a stackable NM2K neuron extension module. Up to 4 NM2K can be stacked.

Jumper J8
Select to supply the power to the BrainCard through a USB connection (pin 1-2) or Arduino power connector (pin 2-3).

Jumper 17
Optional 5 Volts power supply to HDMI through the BrainCard. Default is OFF (no jumper).

Jumper 19
Optional 3.3 Volts power supply to an Arduino Shield if the 3.3V pin of the Arduino power connector is not already powered. This feature can be useful if the BrainCard is interfaced to an Intel Edison module or Raspberry PI.

Arduino Power (J5)

Pins	Signal
1	NC
2	NC
3	NC
4	NC/ 3.3V**
5	5V
6	GND
7	GND
8	Reserved

** disconnected by default. See J19 below

Arduino digital (J3)

Pins	Signal (3.3V max)
1	D8
2	D9
3	D10 / SPI_CS
4	D11 /MOSI
5	D12 / MISO
6	D13 / SPI_CLK
7	NC
8	AREF
9	SDA
10	SCL

Arduino digital (J4)

Pins	Signal (3.3V max)
1	D0 (alt RX)
2	D1 (alt TX)
3	D2
4	D3
5	D4
6	D5
7	D6
8	D7

Arduino analog (J6)

Pins	Signal
1	A0
2	A1
3	A2
4	A3
5	A4
6	A5

JTAG jumper			
1	NC	2	3.3 Volts
3	GND	4	TMS
5	GND	6	TCK
7	GND	8	TDO
9	GND	10	TDI
11	GND	12	NC
13	NC	14	NC

CMOS connector (CM1)	
Pins	Signal
1	GND
2	CAM1_DN0*
3	CAM1_DP0*
4	GND
5	CAM1_DN1
6	CAM1_DP1
7	GND
8	CAM1_CN
9	CAM1_CP
10	GND
11	CAM_GPIO
12	CAM_CLK
13	SCL0
14	SDA0
15	3.3 Volts
(*)Shared w/ PMODB	

PMOD A	
Pins	Signal
1	PMODA1P
2	PMODA1N
3	PMODA2P
4	PMODA2N
5	GND
6	3.3 Volts
7	PMODA3P
8	PMODA3N
9	PMODA4P
10	PMODA4N
11	GND
12	3.3 Volts

PMOD B	
Pins	Signal
1	PMODB1P
2	PMODB1N
3	PMODB2P*
4	PMODB2N*
5	GND
6	3.3 Volts
7	PMODB3P
8	PMODB3N
9	PMODB4P
10	PMODB4N
11	GND
12	3.3 Volts

(*)Shared w/ CM1

Raspberry Pi connector (J9)

Pins J9	Signal	Pins	Signal
1	NC	2	5 Volts In
3	I2C1SDA	4	5 Volts In
5	I2C1SCL	6	GND
7	GPIO4	8	UART_TXD
9	GND	10	UART_RXD
11	GPIO17	12	GPIO18
13	GPIO27	14	GND
15	GPIO22	16	GPIO23
17	NC	18	GPIO24
19	MOSI	20	GND
21	MISO	22	GPIO25
23	SCLK	24	GPIO8
25	GND	26	GPIO7
27	NC	28	NC
29	GPIO5	30	GND
31	GPIO6	32	GPIO12
33	GPIO13	34	GND
35	GPIO19	36	GPIO16
37	GPIO26	38	GPIO20
39	NC	40	GPIO21

Edison connector (J2)

In this first revision of the BrainCard, it is not possible to plug the Intel Edison module directly into the J2 connector on the back of the BrainCard (we still have to fix some instability issues between the Edison and the FPGA of the BrainCard at the power up). However, it is possible to plug into J2 the Edison base module available from SparkFun and populated with the Intel Edison module.

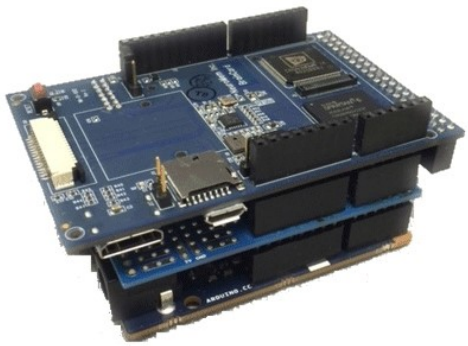
Pins J2	Signal	Pins	Signal
1	GND	2	3.3 Volts
3	USB_ID	4	3.3 Volts
5	GND	6	3.3 Volts
7	MSIC_SPI	8	NC
9	GND	10	NC
11	GND	12	NC
13	GND	14	3.3 Volts
15	GND	16	USB_DP
17	GPI_PWRBT	18	USB_DN
19	FAULT	20	NC
21	PSW	22	GP134
23	V_BAT_BAK	24	GP44
25	GP165	26	GP45
27	GP135	28	GP46
29	NC	30	GP47
31	RCVR_MODE	32	GP48
33	GP13	34	GP49
35	GP12	36	RESET_O#
37	GP182	38	NC
39	GP183	40	NC
41	GP19	42	GP15
43	GP20	44	GP84
45	GP27	46	GP131
47	GP28	48	GP14
49	NC	50	GP42
51	GP111	52	GP40
53	GP110	54	GP41
55	GP109	56	GP43
57	GP115	58	GP78
59	GP114	60	GP77
61	GP130	62	GP79
63	GP129	64	GP82
65	GP128	66	GP80
67	OSC_CLK_O	68	GP83
69	FW_RCVR	70	GP81

6 Troubleshooting SPI

The BrainCard does not power ON

- If the blue LED D1 is not lit, verify that the Jumper J18 is properly configured (between pin 2 and 3 for a USB power supply, pin 1 and 2 for Arduino power supply)
- Note that the LED ON is not sufficient to determine that the BrainCard gets enough power supply for stable operation. If experiencing instability in the behavior of the SPI communication, CM1K, and other components, try using external supply whenever possible over a cascaded power-over-USB coming from an Arduino board for example.

Depending on the base board you select to interface to the BrainCard, you may have to use spacers or in-between shields to ensure that the BrainCard is properly plugged in and powered. In the example below, the heights of the BrainCard's PMOD and the Arduino101 USB connector type A make it impossible to properly assemble the 2 boards without spacers or a dummy proto shield.



The BrainCard is not responding to SPI communications

- Verify that it is properly powered. The blue LED D1 should be lit
- Push the reset buttons of both the BrainCard and the processor card
- Run the Test_SPIComm and diagnose the type of the error (i.e. single Read or Write, multiple Read or Write, etc.)
- Test a different SPI speed (refer to your processor documentation)

- On Arduino:
    - Verify that the SPI Chip Select pin 10 of the Arduino connector is not used by another shield
    - Verify that the pin 10 to 13 of the Arduino connector are reserved for the SPI communication of the BrainCard and not assigned anywhere in your code
    - The default SPI clock divider is one-quarter the frequency of the system clock (4 Mhz for the boards at 16 MHz). Depending on your Arduino processor, you might want to increase it using the function SetClockDivider.
- On Raspberry PI:
    - Make sure the SPI port is enabled (sudo raspi-config/advanced options)
    - Speed should range from ~8kHz to 125MHz depending on the clock divider in use