NeuroMem library for Arduino IDE

Version 1.4
Revised 06/21/2018

GENERAL VISION

CONTENTS

The NeuroMem library is a product of General Vision, Inc. (GV)

For information about ownership, copyrights, warranties and liabilities, refer to the document Standard Terms And Conditions Of Sale or contact us at www.general-vision.com.

1   INTRODUCTION

The NeuroMem library for Arduino is intended to access a NeuroMem network through compatible Arduino boards.

Step 1: Microcontroller
Sensor data collected through Arduino shields and other plug-in modules can be assembled into feature vectors(*).
Their learning can be triggered by external user inputs, programmable time stamps, but also by the detection of novelties made by the neurons themselves.
The pattern vectors are then broadcasted to the NeuroMem neurons for either learning or recognition(*).

Step 2: NeuroMem library
The NeuroMem neurons handle the automatic learning of your examples and associated categories, the recognition of new patterns, the detection of uncertainty cases if any, the detection of drifts or decrease of confidence, the reporting of novelties and/or anomalies.
(*) can also be programmed in the FPGA to optimize speed performance

Step 3: Microcontroller
Finally, you must format the response of the neurons to convert it into a global decision or action for your application, without forgetting to save the knowledge built by the neurons for backup purposes or distribution to other systems(*).
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.
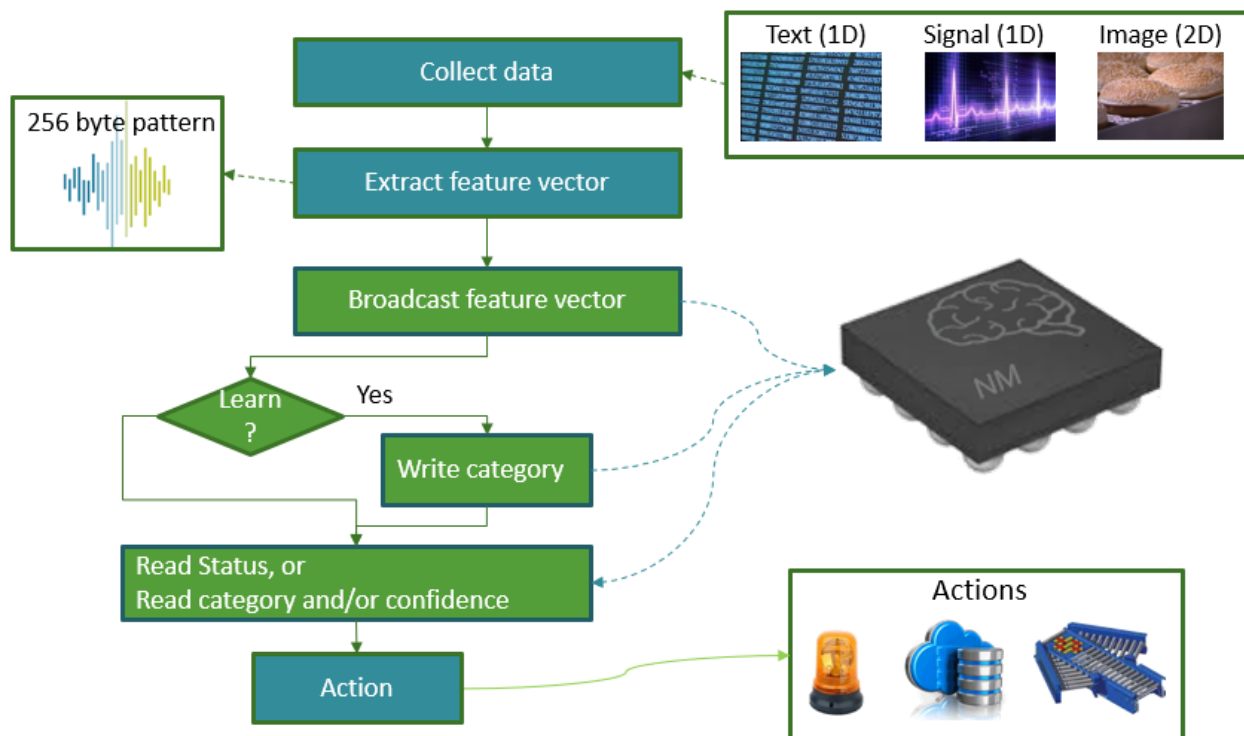
1.1   Installation

Copy the NeuroMem folder to MyDocuments\Arduino\libraries
Launch the Arduino IDE
Under Tools/Board Manager menu, select your Arduino processor board
Under File/Examples menu, scroll down to reach the Examples from Custom Libraries
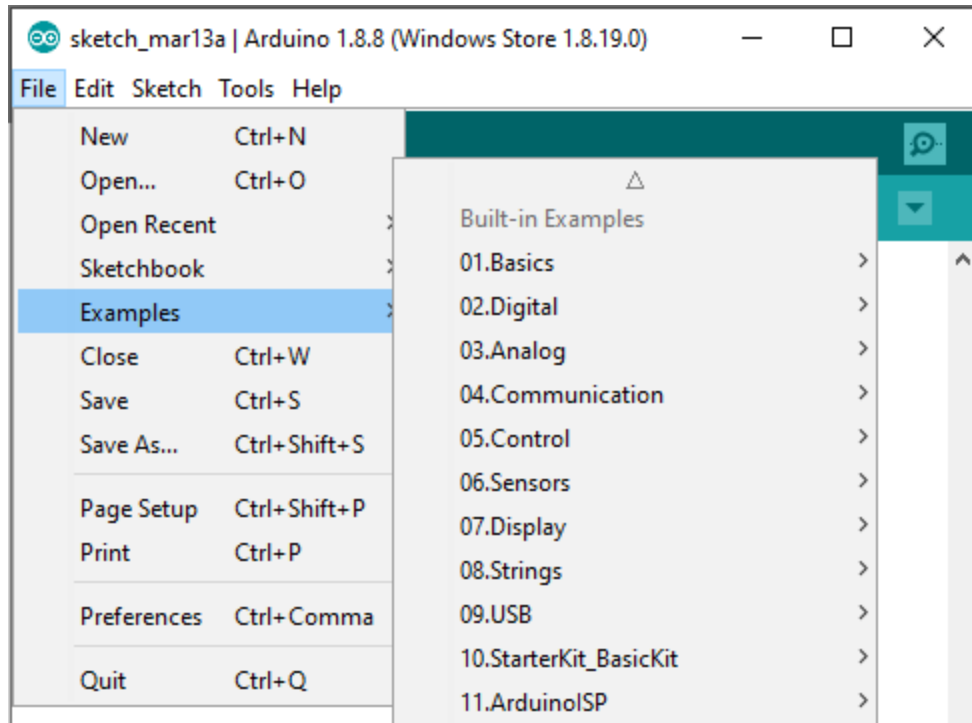
The NeuroMem examples are divided as follows:
- Generic examples for all NeuroMem-Smart devices (NeuroMem_xyz.ino)
- Hardware specific examples (NeuroShield_xyz.ino, etc)
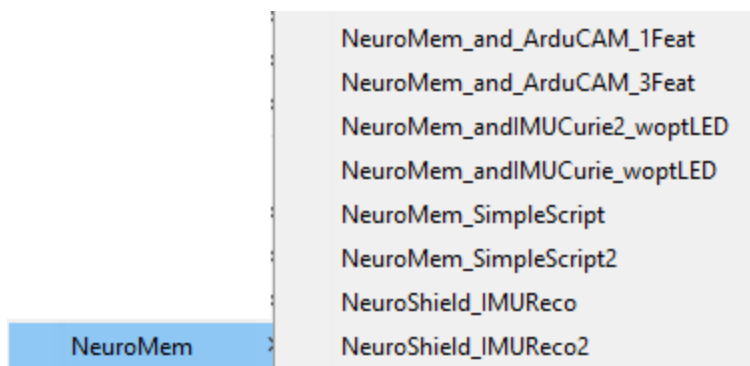
1.2   Typical application workflow

The data collected through the sensors 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.
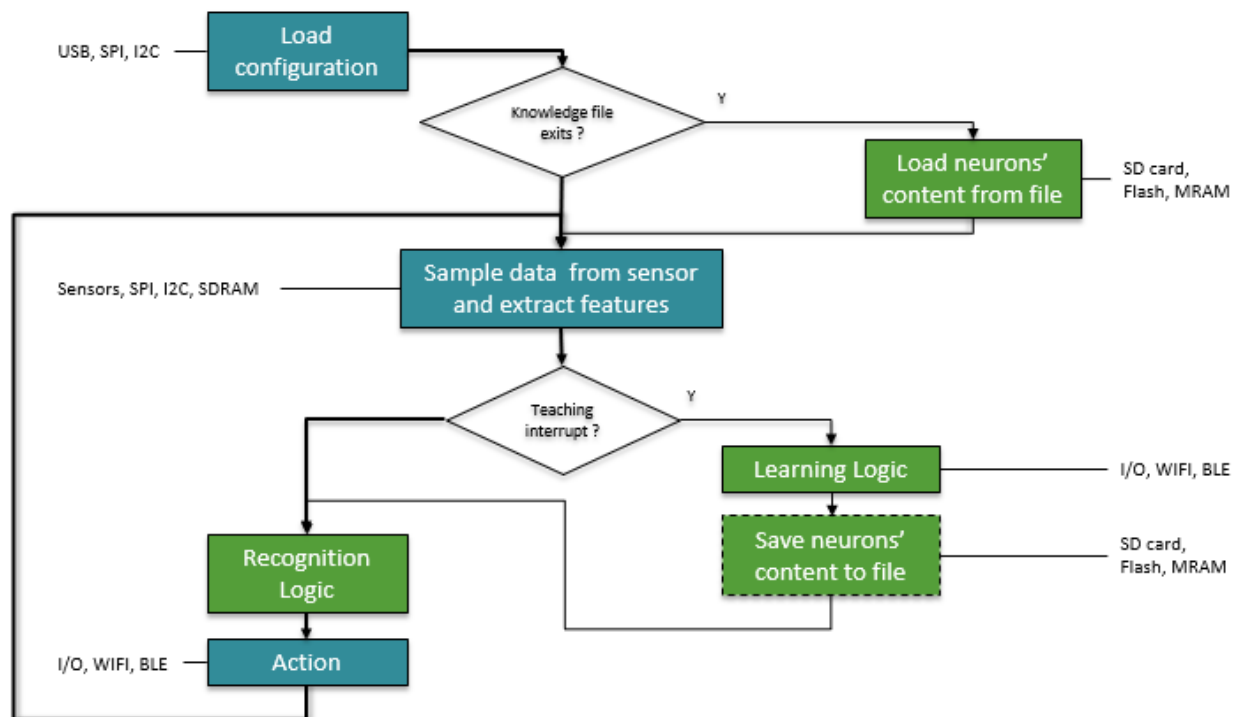

1.2.1   How do the neurons learn and recognize?

Our NeuroMem RBF tutorial is a simple application demonstrating how the neurons build a decision space autonomously by learning examples and how they recognize new vectors with ability to report cases of unknown and uncertainties too. For the sake of a simplicity, the decision space is limited to a 2D space but the mechanism is the same to recognize a (X1, X2) vector as well as an (X1, X2, … X127) vector which is the maximum length supported by the neurons of the CM1K.

For more information of the NeuroMem technology, please refer to the NeuroMem Technology Reference Guide.

1.2.2   What is a feature vector?

The neurons are agnostic to the data source which means that they ready to learn and recognize feature vectors extracted from text (parsing), discrete measurements, audio signal, video signal, digital images, etc. The only constraint is that the vector must be formatted as a byte array of maximum length 256.

In the case of images, a feature vector can be a subsampling of pixels within a region of interest, a histogram, some histogram of gradients within a patch of pixels, etc. In the case of an audio or biosensor signal, the feature vector can be a set of signal samples taken at a specific sampling rate, a histogram of peaks or zero crossing, or a power spectrum, etc.

2   THE NEUROMEM LIBRARY

2.1   NeuroMemAI_library

The functions of the NeuroMemAI library are described in the technical manual TM_NeuroMem_API.pdf. They have a dependency to the NeuroMemSPI library in order to access the hardware.

```
NeuroMemAI();
int begin(int Platform);
void forget();
void forget(int Maxif);
void clearNeurons();
int countNeuronsAvailable();

void setContext(int context, int minif, int maxif);
void getContext(int* context, int* minif, int* maxif);
void setRBF();
void setKNN();

int broadcast(unsigned char vector[], int length);
int learn(unsigned char vector[], int length, int category);
int classify(unsigned char vector[], int length);
int classify(unsigned char vector[], int length, int* distance, int* category, int* nid);
int classify(unsigned char vector[], int length, int K, int distance[], int category[], int nid[]);

void readNeuron(int nid, unsigned char model[], int* context, int* aif, int* category);
void readNeuron(int nid, unsigned char neuron[]);
int readNeurons(unsigned char neurons[]);
int writeNeurons(unsigned char neurons[]);
```

2.2   NeuroMemSPI_library

The NeuroMem Smart protocol is described http://www.general-vision.com/documentation/TM_NeuroMem_Smart_protocol.pdf

```
NeuroMemSPI();
int connect(int Platform);
int FPGArev();
int read(unsigned char reg);
void write(unsigned char reg, int data);
void writeAddr(long addr, int length, int data[]);
void readAddr(long addr, int length, int data[]);
```

3   ACADEMIC EXAMPLES

3.1   Test_SimpleScript

Simple script stimulating the neurons to learn and recognize patterns generated programmatically.

Teach the neurons to learn a set of predefined vectors and practice with cases of recognition demonstrating how the neurons can be queried to report positive identification, or on the contrary uncertain identification, or unknown identification.

Detailed description of this script is available at http://www.general-vision.com/documentation/TM_TestNeurons_SimpleScript.pdf.

3.2   Test_SimpleScript2

Extension of the Test_SimpleScript demonstrating additional capabilities of the neurons, including the use of the KNN, the use of the LSup/L1 norms for the calculation of distances, and the use of multiple contexts to learn and classify vectors representing different dimensions or data types within the same NN.

For more information about the RBF/KNN classifiers, the definition of Contexts, the use of different Norms, please refer to the NeuroMem Technology Reference Guide.
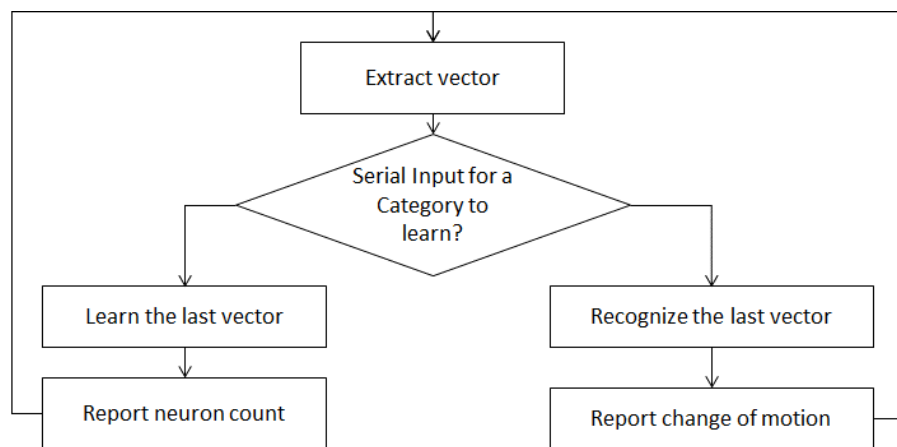
4   EXAMPLES OF MOTION LEARNING AND RECOGNITION

4.1   NeuroMem_andCurieIMU1

View Step-By-Step Tutorial

Hardware requirements
-    Arduino 101

This script requires an Arduino/Genuino101 to use the 6-axis IMU of the Intel Curie module. It 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.

In the Arduino code, the LOOP continuously reads the IMU signals and extracts a simple feature vector which is a normalized subsampling of the 6 axes. The recognition of this vector starts automatically as soon as the neurons have some knowledge. The neurons build the knowledge as soon as you start teaching examples.
When you enter a category through the serial port, the program associates it to the last feature vector. When teaching a vertical motion for example, you want to teach more than once to make sure the neurons learn an example of a vertical up-to-down and vertical down-to-up, obviously at selected speed and amplitudes.


4.2   NeuroMem_andCurieIMU2

Hardware requirements
-    Arduino 101

Test_Neurons_andIMU2 is similar to the Test_Neurons_andIMU1 and illustrates the ability of the NeuroMem neurons to handle multiple networks in a same chip for more robust decision making. This script requires an Arduino/Genuino101 to use the 6-axis IMU of the Intel Curie module.

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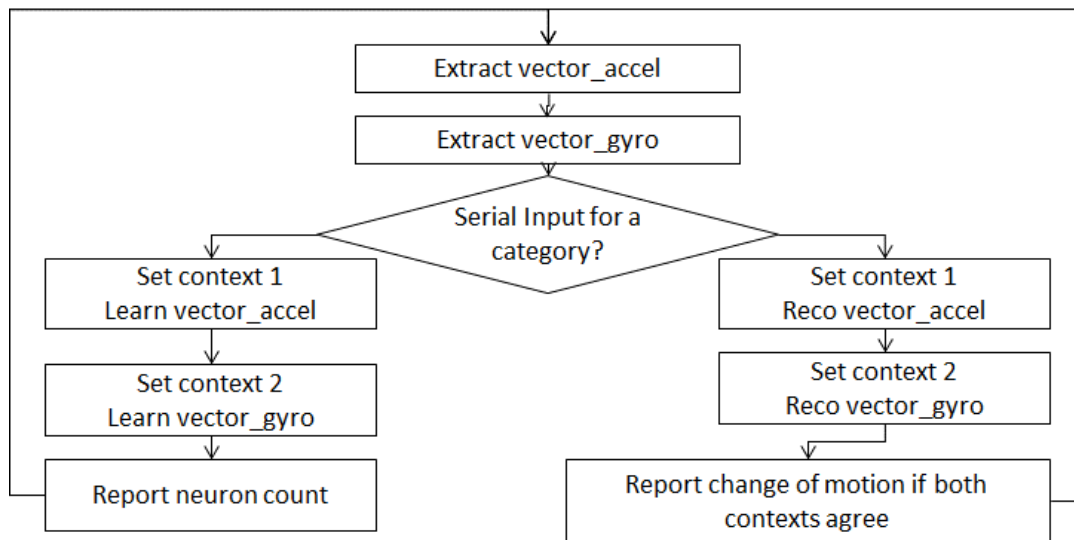
Remark1: This example is very academic and assemble a pattern which should be more sophisticated for real-life system taking a calibration into account, integrating a sampling rate adequate for the type of motion and profiling the waveforms more selectively using distances between peaks and zero crossing,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

4.3 NeuroShield andIMU

View Step-By-Step Tutorial

Hardware requirements
- NeuroShield
-
 Library requirements
- https://playground.arduino.cc/Main/MPU-6050
- https://github.com/jrowberg/i2cdevlib/tree/master/Arduino/MPU6050
- https://github.com/jrowberg/i2cdevlib/tree/master/Arduino/I2Cdev

This script uses the Invensense 6-axis IMU of the NeuroShield. It 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.

In the Arduino code, the LOOP continuously reads the IMU signals and extracts a simple feature vector which is a normalized subsampling of the 6 axes. The recognition of this vector starts automatically as soon as the neurons have some knowledge. The neurons build the knowledge as soon as you start teaching examples.
When you enter a category through the serial port, the program associates it to the last feature vector. When teaching a vertical motion for example, you want to teach more than once to make sure the neurons learn an example of a vertical up-to-down and vertical down-to-up, obviously at selected speed and amplitudes.

5   EXAMPLES FOR OBJECT LEARNING AND RECOGNITION

View Step-By-Step Tutorial

Hardware requirements
- ArduCam Shield V2 with camera module
- Note that this shield uses the ICSP connector as SPI access and unfortunately the NeuroShield does not offer a pass-thru for this connector.
- You will need to connect the 3x2 SPI standard connector (ICSP) from the Arduino microcontroller board to the ArduCam board as follows:
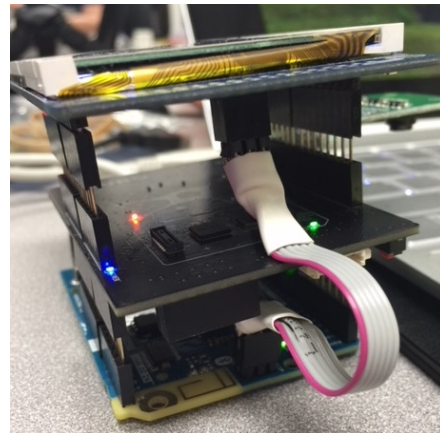
Option #1: Soldering on the ArduCAM shield

Option #2: Custom cabling
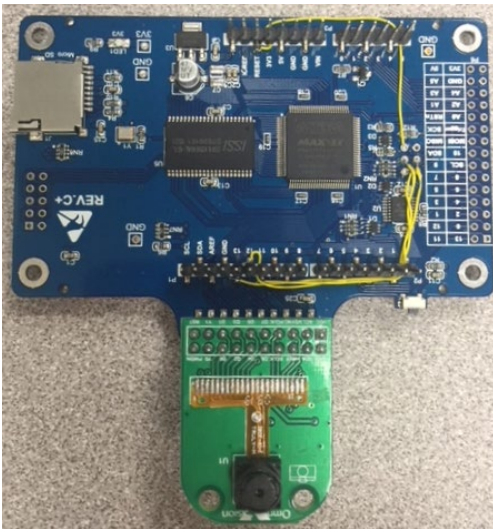
Wire the signals
MOSI, MISO and Clock to pin 11,12,13 of the P1 connector.

Spacers and flex cable between ICSP from Arduino board to ICSP of ArduCam shield

Signal	ICSP	P1
MOSI	Pin4	D11
MISO	Pin1 (upper-right in front view)	D12
SCK	Pin3	D13

Library requirements
- ArduCAM at https://github.com/ArduCAM/Arduino
- UTFT4ArduCAM_SPI at https://github.com/ArduCAM/Arduino

5.1   NeuroMem_andArduCam_ObjectReco

View Step-By-Step Tutorial

Description
- Display video frame and the region of interest being monitored on the LCD
- Recognize continuously the region of interest
- When shutter button is depressed…
  - o   < 2 seconds ==> learn a new category
  - o   > 2 seconds ==> learn a Background category to correct erroneous classification

5.2   NeuroMem_andArduCam_ObjectReco_3Feat

Description
Same as the Neuromem_andArduCam1 except that it extracts three different feature vectors to learn and classify the objects: a subsample, a histogram of the colors and a profile of the lines and columns.

Recognition based on multiple features or contexts
The 3 features are calculated at the same time during the readout of the pixel values from the FIFO.
- Feature 1= pixel subsampling ➔ used to train neurons assigned to Context 1
- Feature 2= RGB histogram ➔ used to train neurons assigned to Context 2
- Feature 3= Composite profile➔ used to train neurons assigned to Context 3

Decision rule between contexts
The script reports a category on the screen only if the neurons of 2 out of the 3 contexts (at the minimum) agree on the same category. Otherwise, the script reports the object as "Unknown".

5.3   NeuroMem_andArduCam_TargetTracking

View Step-By-Step Tutorial

This script illustrates how you can learn a target and train the neurons to correct the position of the camera so the target always appears at the center of the field of view.
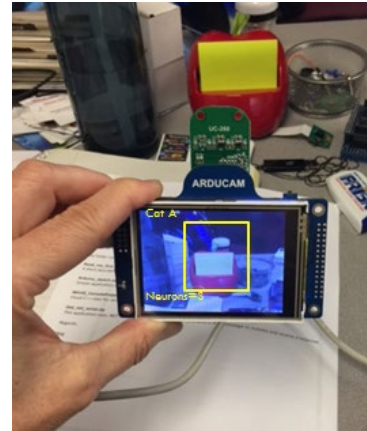
6 TROUBLESHOOTING SPI

6.1 The Arduino board cannot be programmed properly

- Verify that it is properly powered
- Verify that the selected COM port is correct
- Unplug all shield boards before programming, including the shield with the NeuroMem network

6.2 The Arduino board is not responding to SPI communication

- Verify that it is properly powered
- Push the reset buttons
- Run the Test_SPIComm and diagnose the type of the error (single Read/Write, multiple Read/Write)
- Test a different SPI speed (refer to your processor documentation)
- Test powering the platform with an external power supply
- Verify that the SPI Chip Select pin is not used by another shield
- 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.