As part of the existing framework there is a remote control object. I originally designed this to take inputs from controllers like joysticks so they could inject currents into neurons. However, it works for a wide variety of sensor systems. It has a generic interface that converts incoming signals into currents for a user specified neuron. I will probably use that as a base and expand it so it can also handle generic outputs. The user will be able to pick a property from any of the objects in the simulation. That property value will be converted using a gain function and then transmitted to the receiver. So you could pick a joint position, a neuron conductance, or in your case the membrane voltage. This will make configuring the system easy and application agnostic. On top of this I will create a new simple serial controller type that will open a serial port and perform the actual serial IO. I suggest the data packets below. Please let me know if you feel it needs modifications. The same packet structure would be used to send data from animatlab to the RIO and back from that to animatlab. Within AnimatLab the user will assign a given data ID to each of the items that are to be transmitted or received. Within the RIO code it would need to have a corresponding mapping. Data will be included in a packet only if it has changed since the last time an update has been sent, and a packet will only be sent if at least one data item must be sent. If there is an error in the transmission where the checksum is incorrect then the receiver will send a Data Error packet. When the original transmitter receives this packet type it will resend all data values regardless of whether they have changed. Data packets will be streaming continuously between AnimatLab and RIO, so I do not believe that an acknowledge packet will be required for each receipt. Let me know if this sounds okay, or if you see any issues. I am trying to wrap something up that I am working on right now, so it will probably be a few weeks before I can get to this. Is that okay?

David

Data Send Packet

|  |  |  |
| --- | --- | --- |
| **Byte[s]** | **Description** | **Value** |
| 0-1 | Header Byte String | 0xFF, 0xFF |
| 2 | *Data Send Message ID* | 0x01 |
| 3-4 | *Total size of message including header, message id, data id's, data value's, checksum, and footer* |  |
| 5-6 | Data ID | ID associated with a given part in the GUI editor. This is user configurable. The receiving system will need to know what each ID corresponds to. |
| 7-11 | Data Value (Floating point value) | Data value corresponding to the preceding Data ID |
| ..... | Repeat Data ID and Data Value pairs as required for all data that will be transmitted in this packet |  |
| 12 | Checksum |  |

Data Error Packet

|  |  |  |
| --- | --- | --- |
| **Byte[s]** | **Description** | **Value** |
| 0-1 | Header Byte String | 0xFF, 0xFF |
| 2 | Data Error Message ID | 0x02 |
| 3-4 | Total size of message including header, message id, checksum, and footer |  |
|  |  |  |
| 5 | Checksum |  |
|  |  |  |

https://ssl.gstatic.com/ui/v1/icons/mail/images/cleardot.gif

**ANIMAT SERIAL http://animatlab.com/Portals/_default/Containers/DarkKnight/Images/Title-BG-Blue.png**

The Animat Serial interface implements a simple serial protocol to allow the simulation to communicate data between external hardware and the simulation components. This will allow you to do things like sending the membrane voltage of a neuron out to your hardware, or importing in data values and using that to set the current on a neuron. You are not limited to just interacting with the neural systems though. You can use this for virtually all components in the simulation. For example, you could send out the joint angle of hinge joint as data that your hardware could see. AnimatSerial is built on top of the RemotControl device and uses the standard RemoteControl Linkage idea. To use it you will need to

* add a remote control and choose Animat Serial. Then you will need to
* set the name, port (example: COM10 on Windows), baud rate, and then press the edit button next to the Links property (shown as ...).
  + This will open the editor for adding, removing, and editing your linkages. You can add input links from the hardware to the simulation, or output links from the simulation back to the hardware.
  + You can also choose to use a pass through linkage or a pulsed linkage. Pass through linkages should be used when you have a continuous stream of data coming in. Pulsed links should be used when you discrete data events that occur. One difference between the linkages for the serial system and the XBee controller is that if you create an input linkage then you can simply type in the name of the source property instead of choosing it from a drop down. You can set this to whatever you like. Similarly, if you choose an output link then you can type in the target property. A second difference is that for all serial linkages you **must**specify a unique Data ID. This is the ID that it uses to sync up with the hardware on the serial protocol. So the ID that is listed here must match the ID that the hardware knows about. For example, if you want to send the membrane voltage from a neuron to your hardware then you would set it's data ID to be some value, say 0. Any time that the membrane voltage changes then this will be transmitted to the hardware over the serial link and be associated with an ID of 0. Each time the hardware sees data come in with an ID of 0 it needs to understand that this is the membrane voltage from neuron X. Essentially, you must make sure the data between the hardware and simulation are in synch. Here is an attachment that describes the serial communications protocol so you can implement it with your hardware:[AnimatSerial ICD.](http://animatlab.com/Portals/0/Media/AnimatSerial_ICD.docx)

I have also included a library and test sketch for an **Arduino** to demonstrate how to use the AnimatSerial communications. It also includes a test project file. You can download this here: [AnimatSerialTest.zip](http://animatlab.com/Portals/0/Media/AnimatSerialTest.zip).

* Basically, you create an AnimatSerial object and pass it a pointer to the Serial line you want it to use.
* You also need to specify the maximum input and output ID values you will be expecting.
* Then you call begin with the baud rate you want to use.
* After that you can call animat.readMsgs() to read any incoming messages.
* You check the animat.isChanged() property to see if any data values have changed in the last read.
  + If they are then you can loop through them to get the new value.
* To send data you need to call animat.addData(ID, value). You use this to buffer up all your changes for this run of the update loop, then you call animat.writeMsgs() to send them all to the simulation.
* Here is the code for the sketch for you to look over. It sends a sine wave of data continuously on ID 0, and sends a spike of data each time the sine wave completes on ID 1.

#include "AnimatSerial.h"

AnimatSerial animat(&Serial1, 10, 10);

#define PI 3.14159

#define PI2 2\*3.14159

float x=0;

int counter=0;

int counterror=0;

AnimatData data;

void setup() {

Serial.begin(57600);

while(!Serial);

Serial.println("Starting setup");

animat.begin(38400); // 115200

}

void loop() {

animat.readMsgs();

if(animat.isChanged())

{

for(int i=0; i<animat.getindatatotal(); if(counter="=" if(counterror="=100)" counterror="0;" val1="10\*sin(x)," val2="0;" +="0.01;"> PI2)

{

val2=1;

//Serial.println("Pulsing Val2");

}

//Serial.print("Adding data: ");

//Serial.println(val1);

animat.addData(0, val1);

animat.addData(1, val2);

//Serial.println("Writing data");

animat.writeMsgs();

if(x > PI2)

x = 0;

counter = 0;

}

}

</animat.getindatatotal();>