JaCa-Arduino - A tutorial

Introduction

The JaCa-Arduino project is an attempt to provide a means for the CArtAgO platform to work with Arduino based applications. By this integration we aim to allow the creation of artifacts that directly control (read or set values) the different sensors and actuators that can be attached to an Arduino board.

This tutorial will present a simple example of an agent application that can turn a LED on and off.
Even though the application is quite easy, it suits the purpose of showing the workings of this integration attempt.

The tutorial will show the hardware setup used, the way in which to write your configuration files and the code examples for the Arduino, CArtAgO Artifact and Jason agent.

We assume that the reader is familiar with the Jason and CArtAgO projects. For more details please refer to the links below:

- http://jason.sourceforge.net/Jason/Jason.html - Jason
- http://cartago.sourceforge.net/ - CArtAgO

Hardware setup

The current project was tested on an Arduino Duemilanove board (with the ATMega368 microcontroller) that also uses an Arduino Ethernet Shield based on the Wiznet W5100 Ethernet chip. The functioning is not reduced only to the above setup. Basically, you can use any type of Arduino Board (e.g. UNO, Mega, Mega2560, Diecimila) as long as you can find an appropiate Ethernet or Wireless shield.
Additionally, you will need an LED (any type will be good) and an appropiate resistor to protect the LED from burning (e.g. 1K). Connect the shorter (negative) leg of the LED to the GND of the Arduino Board and the longer one (positive), via the resistor, to digital pin 7 from the Arduino Board. An ethernet cable will also be required to establish the connection between your computer and the board.
Below are three pictures: two for the boards used and one showing the whole assembly.

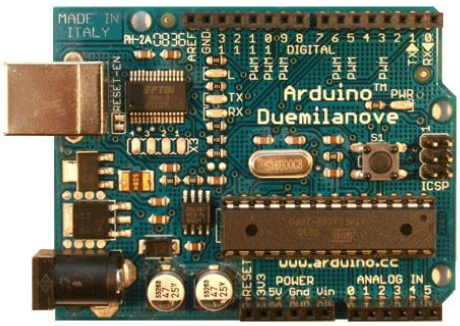

Fig. 1 Arduino Duemilanove Board

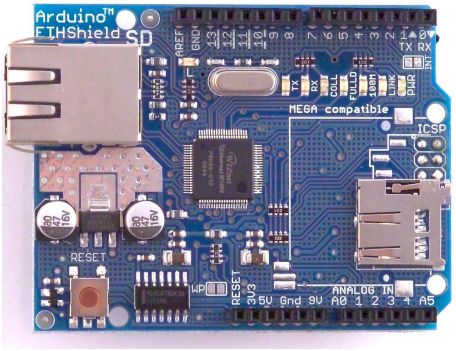

Fig. 2 Arduino Ethernet Shield (Wiznet W5100 chip)

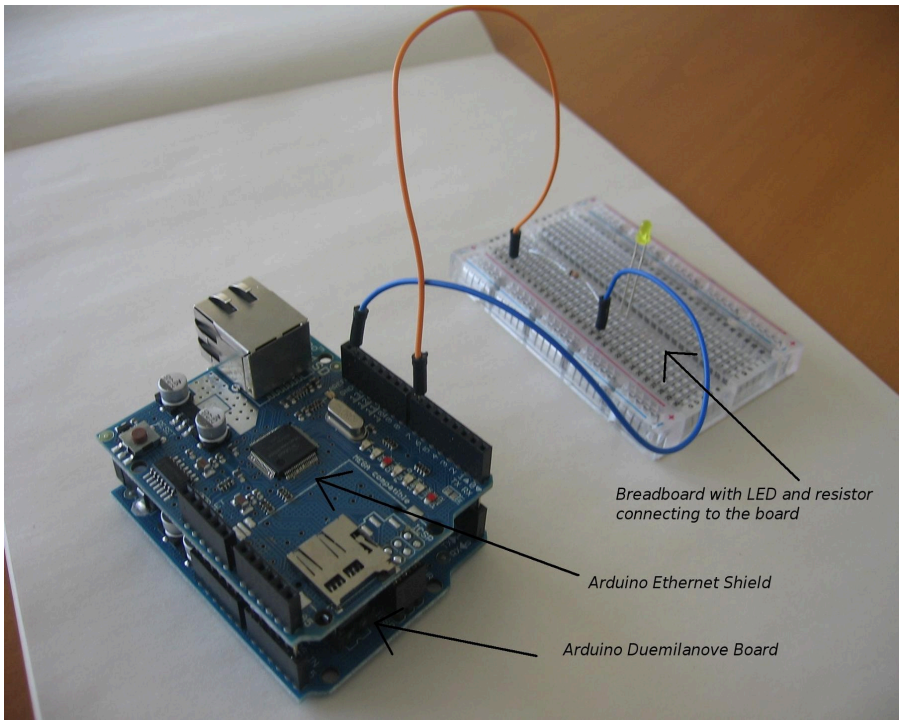

Fig.3 Arduino + LED assembly

Functioning principal and configuration files

The idea behind the project is simple and nice. Run a tiny web server implementation (this is why you need an Ethernet shield) on the Arduino that accepts GET and POST requests.
Write handlers for each type of request, depending on the URL, and use GET requests to retrieve values from a sensor and POST requests to set values for an actuator. In this way it is easy to write a CArtAgO artifact that manipulates the reads and writes of values from and to the board.

To make things even easier, we have developed a generic Artifact that handles this and you are only required to write the following types of files:

- an XML file that describes the sensors you have on the board and the operations that you can do on them
- a PDE file which will be loaded on to the Arduino and which implements handlers for the different operations (requests) you have (how to read/write values to sensors/actuators connected to the board)

Examples for these files will be given next.

XML file

The XML file is used for the configuration of the CArtAgO Artifact that will be linked to the Arduino Board and will control the sensor and actuators on it. It is used to define names of sensors, some simple presets and the operations that can be performed on the sensors.
Here is an example snippet from the LED project:

```
<sensor name="LED">
   <sensoroptions readfreq="1000" />
   <operation name="queryLED" type="GET" valuename="onstate" cardinality="1" valuetype="int" />
   <operation name="setLED" type="SET" valuename="onstate" cardinality="1" valuetype="int" />
</sensor>
```

From this we can deduce the following information:

- the LED will be queried for its state every 1000ms
- the *queryLED* operation will match a GET request on the URL “/LED/onstate”
- the *setLEd* operation will match a POST request on the ULR “/LED/onstate”

Operations can be made to get/set the values of multiple parameters at once. For instance:

```
<operation name="getDummy" type="GET" valuename="dummyvals" cardinality="2" valuetype="float int" />
```

gets the value name *dummyvals* as a tuple of two variables, one of type *float* and one of type *int*.
Accepted valuetypes are *int, float* and *boolean.*

PDE files

PDE files are actually C code that gets compiled to the appropriate Arduino Board that you have at your disposal. To edit and compile such files, use the dedicated Arduino IDE.
The following link directs you to a getting started with Arduino web page: http://arduino.cc/en/Guide/HomePage.

As mentioned, PDE files contain the setup and handler functions that allow you to retrieve and set values from the sensors attached to the Arduion Board.

This snippet is an example of a handler function that helps retrieve the state of the LED (on or off):

// ##### request handler functions #####

```
boolean led_get_onstate_handler(TinyWebServer& web_server) {
    web_server.send_error_code(200);
    web_server.send_content_type("text/plain");
    web_server.end_headers();
    Client& client = web_server.get_client();
    client.println(getLedState(), DEC);

    return true;
}
```

Once you have written your setup and handler functions, all you have to do is include their definitions in the ***WebServerSensorMgr.h*** file and bind them to URL patterns in the ***SensorMgrInit.pde*** file and you are set. Please bare in mind that the URL pattern has to match the **sensor** and **value** names in the operations you have defined in the XML file for the CArtAgO artifact configuration. It is by this naming mechanism that the appropriate values are read/written by the correct mechanism.

E.g.: binding handler functions to URL patterns

```
TinyWebServer::PathHandler handlers[] = {
    {"/", TinyWebServer::GET, &index_handler },
    {"/LED/onstate", TinyWebServer::GET, &led_get_onstate_handler },
    {"/LED/onstate", TinyWebServer::POST, &led_set_onstate_handler },
    {NULL},
};
```

“/LED/onstate” matches sensor name “LED” and valuename “onstate”.

E.g.: the array of

```
sensorSetupFunc setupFuncArr[] = {
    &setupLEDSensor,
    NULL
};
```

Project Quick Install and Setup Guide

The following is a step by step description of the supplied project files and of the way to set them up.

Preconditions:

- installation of Arduino IDE version 0.22
- installation of Eclipse or jEdit
- installation of Jason plugin either for Eclipse (recommended) or jEdit

Steps:

- Download the **JaCa-Arduino-Demo.zip** file and dearchive it.
- Copy the folders ***TinyWebServer*** and ***WebServerSensorMgr*** to the ***libraries*** folder of your Arduino IDE installation (it is a workaround needed to acomodate for the way in which the IDE looks for files when trying to compile and link them)
- use the Arduino IDE to open the files in the ***arduino-webserver (LEDSensor.pde, SensorMgrInit.pde, WebServerSensorMgr.pde)*** folder and compile them
- use a serial cable to connect to the Arduino board and load the compiled code on to the board via the IDE
- create a new Jason project using the ***asl***, ***lib*** folders and the .**mas2j** file from the archive
- connect an Ethernet cable from your computer to the Arduino Ethernet Shield. Set up your computer's network interface to be in the same network as the Arduino Board (default IP for Arduino board is given in **config_LED.xml** and is **192.168.1.1** so you have to set your NIC to an IP from the 192.168.1.0/8 network)
- Run the project and enjoy the simple example.