Was ist nötig, um einen neuen Block zu generieren?

Hier am Beispiel des http-Get

Installation der Eclipse Entwicklungsumgebung
http://blog.ardublock.com/2012/06/11/setup-ardublock-development-in-eclipse/
Unser aktueller Source befindet sich im Zip-File, Ordner "Quellen".

1. Beschreibung des Blocks in der Datei ardublock.xml

```
<BlockGenus name="sendHTTPGet" kind="data" color="30 30 255"
          initlabel="bg.sendHTTPGet" editable-label="no" is-label-value="no">

          <BlockConnectors>
                 <BlockConnector connector-type="string"
                        connector-kind="socket" label="bc.host">
                        <DefaultArg genus-name="host" label="Server IP" />
                 </BlockConnector>
                 <BlockConnector connector-type="number"
                        connector-kind="socket" label="bc.port">
                        <DefaultArg genus-name="number" label="80" />
                 </BlockConnector>
                 <BlockConnector connector-type="string"
                        connector-kind="socket" label="bc.URI">
                        <DefaultArg genus-name="URI" label="Server GET " />
                 </BlockConnector>
                 <BlockConnector connector-type="string" connector-kind="plug" />
          </BlockConnectors>
    </BlockGenus>
```

2. Und einhängen in einen Baukasten auf der linken Seite der Ardublockoberfläche (am Ende der Datei ardublock.xml).

```
<BlockDrawer button-color="50 50 255" name="bd.HTTP_communication" type="factory">
    <BlockGenusMember>sendHTTPGet</BlockGenusMember>
</BlockDrawer>
```

3. Beschriftung in der Landessprache: ardublock.properties (International) bzw. ardublock_de.properties (Deutsch). Definiert den angezeigten Text im Block.

```
bg.sendHTTPGet=http GET
bg.sendHTTPGet.description=REST Protokoll GET
bg.URI=URI
bc.URI=URI
bg.URI.description=Kommando
bg.port=Port
bc.port=Port
bg.port.description=Port Nummer (80, 8080)
```

4. Codegenerator (Java) erstellen (möglichst unter Eclipse per Cut&Paste aus seiner ähnlichen Vorlage holen, dann passen alle Bezüge).
   Hier ein Ausschnitt aus der Datei IoTHTTPGet.java
   Am einfachsten vorher getestete C-Files aus dem Arduino per Cut&Paste in den Java-Quelltext übernehmen. Wenn im Quelltext vorher mit `String BlaBla =“` ein String begonnen wird, vervollständigt Eclipse die notwendigen Steuerzeichen \n ,\“ usw. selbständig. Hier am Beispiel der httpGET Funktion:

```
String httpGET ="//------------------------------------- HTTP-Get\n"
                + "int httpGET(String host, String cmd, String &antwort,int Port) {\n"
                + Client
                + " String text = \"GET http://\"+ host + cmd + \" HTTP/1.1\\r\\n\";\n"
                + " text = text + \"Host:\" + host + \"\\r\\n\";\n"
                + " text = text + \"Connection:close\\r\\n\\r\\n\";\n"
                + Check
                + " if (ok) { // Netzwerkzugang vorhanden \n"
                + "  ok = client.connect(host.c_str(),Port);// verbinde mit Client\n"
                + "  if (ok) {\n"
                + "   client.print(text);                 // sende an Client \n"
                + "   for (int tout=1000;tout>0 && client.available()==0; tout--)\n"
                + "    delay(10);                         // und warte auf Antwort\n"
                + "   if (client.available() > 0)         // Anwort gesehen \n"
                + "    while (client.available())         // und ausgewertet\n"
                +"      antwort = antwort + client.readStringUntil('\\r');\n"
                +"    else ok = 0;\n"
                +"    client.stop(); \n"
                +"    Serial.println(antwort);\n"
                + "  } \n"
                + " } \n"
                + " if (!ok) Serial.print(\" no connection\"); // Fehlermeldung\n"
                + " return ok;\n"
                + "}\n";
```

Der folgende Befehl fügt den Code im fertigen Arduino-Programm dann als Funktionsdefinition ein. Sollte der httpGET von mehreren Blöcken genutzt werden, ist zwingend immer genau derselbe Quelltext erforderlich. Nur dann erkennt der Codegenerator die Dublette und erzeugt die Funktionsdefinition auch nur einmal. Unser httpGET wird z.B. auch vom Thingspeak und yahoo-Block verwendet. Änderungen müssen also immer an mehreren Stellen nachgeführt werde – die man kennen muss… Das ist die größte Hürde für ein verteiltes Arbeiten mit Ardublock.

```
translator.addDefinitionCommand(httpGET);
```

Variable Informationen aus dem Ardublock werden über die folgenden Befehlszeilen aus den Eingängen im Ardublock geholt. Diese ändern sich ja jedesmal entsprechend der Programmierung durch den Anwender:

```
String host,cmd,port;
TranslatorBlock translatorBlock = this.getRequiredTranslatorBlockAtSocket(0);
host = translatorBlock.toCode();

translatorBlock = this.getRequiredTranslatorBlockAtSocket(1);
port = translatorBlock.toCode();

translatorBlock = this.getRequiredTranslatorBlockAtSocket(2);
cmd = translatorBlock.toCode();
```

Damit haben wir jetzt alle Informationen, um im Arduinoprogramm unseren Block an der richtigen Stelle einzubauen:

```
String ret = "httpRESTGET("+host+","+cmd+","+port+")";
return codePrefix + ret + codeSuffix;
```

Als letztes noch in der Datei block-mapping.properties den Bezug zwischen Ardublock-Puzzleteil und dem java Quelltextgenerator herstellen:

```
sendHTTPGet=com.ardublock.translator.block.IoT.IoTHTTPGet
```

Fertig!