Move a toy car with your mind

Build a telekinetic application with Bluemix and Cloudant

Software Developer
Big Data University

Software Developer
Big Data University

Senior Program Manager – Big Data and Cloud
IBM

08 May 2015
(First published 13 November 2014)

Learn how to extract brain wave data from the Muse wearable device to perform an action in the real world: in this case, move a toy car. The article explains how to work with the Arduino and the toy car remote control from the hardware side; and from the software side, how to work with Bluemix to create a Node.js web application to see real-time brain wave activity. The article also shows how to use the Bluemix Cloudant NoSQL database service to store all the data collected from the device and how to use R inside the dashDB service to plot a graph, based on the data collected. This article demonstrates how a simple and cheap solution can help you move physical objects with the power of your mind!

Telekinesis, or the ability to move an object with your thoughts, is no longer a far-fetched reality. Using the Muse™, a wearable device that collects brain wave signals, an Arduino device, and a Python program, this article demonstrates how you can move a toy car using the power of your mind. In reality, the toy car is not moved by analyzing your thoughts, but actually by determining the type of brain waves that are being most active at a given point in time.

Moving a toy car may not be a meaningful application. However, imagine the possibilities for people with disabilities who cannot walk or talk—using their brain waves, they can now potentially move an object. Moreover, thanks to the internet, you may be able to move an object with your brain waves in another country or continent! Follow the steps outlined in this tutorial to get your
first object (a toy car) to move. This tutorial also describes how a Node.js web application running on IBM Bluemix was developed to store and show the brain activity collected using the Muse.


What you'll need to build your application

- A toy car with remote control (the New Bright 1:24 Scale Radio Control Sports Car model is used in this tutorial).
- An Arduino device.
- A Muse headband.
- A Bluemix account, so you can use the Node.js runtime, the Cloudant NoSQL database service, and the dashDB service.
- A DevOps Services account to get and fork the sample code below.
- Familiarity with Python programming language.
- Familiarity with Cloudant (noSQL database).

The Muse connects to the laptop using Bluetooth. The laptop is connected to an Arduino Uno device, which is connected to the remote controller that comes with the toy car.

The laptop runs a local Python application with logic that analyzes the type of brain wave received by the Muse. When it determines that ‘alpha’ waves are the strongest (associated with relaxation), it will execute a small program running on the Arduino Uno device to move the car forward.
At the same time, a Bluemix application using Node.js, Cloudant NoSQL database, and dashDB is running. This is a web application that will display in real time the Muse sensor's activity, the brain activity, and the data being passed and stored in a Cloudant NoSQL database. The Cloudant and dashDB Bluemix services are configured so that the data is synchronized from Cloudant to dashDB. Using R within dashDB, we show a simple visualization of the data collected.

Run the app
Get the code

Part I. Hardware - Toy car remote and Arduino

Step 1. Disassemble toy car remote

You first need to set up all the hardware. For this tutorial, we used a specific car model (New Bright 1:24 Scale Radio Control Sports Car) that allowed us to solder some wires to specific points and have full control of the car. Although it's not strictly necessary, we recommend that you use a similar model to make things easier.
This particular car has forward/backward and left/right buttons in the remote control. After opening up the remote, you can look for the corresponding test point (TP) spots on the circuit board that lead to their respective buttons. These spots on the circuit board are used for testing in the manufacture process. We can use them to send HIGH and LOW signals to control the car without removing the buttons. Also, since the remote control is powered by two AA batteries (3V total), we can power the circuit board by using the 3.3V output in the Arduino.

**TP spots to solder wires:**

- Backward - TP80
- Forward - TP81
- Left - TP82
- Right - TP83
Note: This schema is specific to the remote control we're using. If you are using another model of toy car and different remote control, you need to look for the specific test points of the circuit board.

After soldering wires on the TP spots (forward/backward and left/right) and on the power spots, the remote control is ready to be connected to the Arduino.

Step 2. Connect the Arduino to the toy car remote

After soldering the controller on the right places, it's time to connect the Arduino. Connect the colored wire as follows:

- Blue (backward TP80) - Pin 4
- White (forward TP81) - Pin 2
- Green (left TP82) - Pin 8
- Yellow (right TP83) - Pin 12
- Red - 3.3V
- Orange - GND
**Note:** You can use any colors you want for the cables. Just remember to wire the cables to the right pins.

Let's see how the fully assembled setup looks.
Part II. Software

Step 1. Extract the data from the Muse headband

The Muse headband includes an SDK and sample applications. This first describes how you can extract the different types of brain waves from the Muse.

1. Pair the Muse to your computer using Bluetooth (details are described on the device's manual).
2. Install the Muse SDK on your computer.
   **Note:** Make sure to install the pyliblo and liblo (version 0.27) libraries.
3. Run muse-io:
   ```
   $ ./muse-io
   ```
4. Run the following command to see all the OSC messages coming from the Muse:
   ```
   $ oscdump 5000
   ```

Step 2. Develop a web app using Node.js on Bluemix to show brain states

Now that you successfully extracted some data from the Muse, let's develop the web application to show all this data.

We suggest you fork our repository and get the code for the app.
1. Change the code to send data to a Cloudant database.
   a. Navigate to nodejs-app/app.js.

   ```javascript
   /*
    * CLOUDANT DATABASE SETTINGS
    */
   var cradle = require('cradle'); //Must have Cradle library installed
   var account = new(cradle.Connection)({
     host: 'username.cloudant.com', //Cloudant URL
     port: 443,
     secure: true,
     auth:{
       username: "username", //Cloudant username
       password: "password" //Cloudant password
     }
   });
   var db = account.database('databasename'); //database name
   
   arrayJson.push(dataCloudantSave);
   if(arrayJson.length==100){
     db.save(arrayJson,function(err,res){
       console.log(err);
       console.log(res);
     });
     arrayJson = [];
   }
   ```

   b. Look for the following code and change the variables with the information for your Cloudant database:

   ```javascript
   if (startSession){
     //Change the variation in which the chart should change color, e.g 0.2, 0.35...
     drawChartCanvas(att["C0"],att["C1"],att["C2"],att["C3"],0.1);
     //The last attribute is the "VARIATION" in which the chart should change the color
   }
   ```

   2. Set the color variation of the bar chart.
      b. Change the last attribute in the `drawChartCanvas` method (variation 0-1).

   ```javascript
   if (startSession){
     drawChartCanvas(att["C0"],att["C1"],att["C2"],att["C3"],0.1);
     //The last attribute is the "VARIATION" in which the chart should change the color
   }
   ```

   3. Prepare the app for deployment.
a. Navigate to nodejs-app/manifest.yml.

```
applications:
  - disk_quota: 256M
    host: movetoycar
    name: movetoycar
    path: .
    domain: mybluemix.net
    instances: 1
    memory: 128M
```

b. Change the file with information about your app.

**Note:** You can create a Node.js application on the Bluemix dashboard and then change the manifest.yml with the information of the app (provided when the app is created).

**READ:** Creating a Node.js application on Bluemix

4. Deploy the application to Bluemix.

**Note:** Before deploying the application, you must have installed the Cloud Foundry command-line interface (CLI).

a. Open a command prompt.
b. Go to the application directory (nodejs-app).
c. Run the following `cf` commands:
   - Connect to Bluemix:
     ```bash
     $ cf api https://api.ng.bluemix.net
     ```
   - Log into Bluemix (enter email and password when requested):
     ```bash
     $ cf login
     ```
   - Push the application:
     ```bash
     $ cf push <appname>
     ```

**Tip:** When deploying an application using the `cf` tool, open another command prompt and tail the logs by using `$ cf logs <appname>`.

**READ:** Deploying a Node.js application to Bluemix

**Step 3. Develop the code to control your RC Car from Arduino**

1. If you don't have the Arduino IDE installed, [download it](https://www.arduino.cc/en/Main/Download).
2. Open the file called serialArduino.ino on the repository.
3. Upload the code to your Arduino.

That's it. Your arduino is set.

Now let's take a look at the code:

Here, we set the pins that will send commands to the controller:

```cpp
#define FORWARD 2
#define BACKWARD 4
#define LEFT 8
#define RIGHT 12
#define LED 13

String inputString = ""; // a string to hold incoming data
boolean stringComplete = false, goodString = false; // whether the string is complete
int forward=HIGH, backward=HIGH, left=HIGH, right=HIGH;

void setup() {
    // initialize serial:
    Serial.begin(115200);
    // reserve 10 bytes for the inputString:
    inputString.reserve(10);
    pinMode(FORWARD, OUTPUT);
    pinMode(BACKWARD, OUTPUT);
    pinMode(LEFT, OUTPUT);
    pinMode(RIGHT, OUTPUT);
    pinMode(LED, OUTPUT);

    digitalWrite(FORWARD, HIGH);
    digitalWrite(BACKWARD, HIGH);
    digitalWrite(LEFT, HIGH);
    digitalWrite(RIGHT, HIGH);
}

On the main loop, we check for the serial input and set the pins accordingly:

```cpp
void loop() {
    if (stringComplete) {
        //Serial.println(inputString);
        goodString = false;

        if(inputString == "F\n"){
            digitalWrite(LED, HIGH);
            forward = LOW;
            backward = HIGH;
            goodString = true;
        }

        else if(inputString == "B\n"){
            forward = HIGH;
            backward = LOW;
            goodString = true;
        }

        else if(inputString == "L\n"){
            left = LOW;
            right = HIGH;
            goodString = true;
        }

        else if(inputString == "R\n"){
            left = HIGH;
        }
    }

    // Here is where the code begins to send commands
    // to the controller based on the inputString.
}
```
Step 4. Redirect the data from the Muse using a Python program

The last step is reading the data from the Muse and sending, at the same time, the right command to the Arduino and the correct data to be displayed on your web application.

1. Before you run the Python application, you need to connect to the Muse using the software they provided. This software will receive the data through Bluetooth and redirect it to an OSC port.
   a. Go to the SDK folder.
   b. Run the following command:
      ```
      $ ./muse-io --dsp --osc osc.udp://localhost:5000
      ```
   c. Change the Python script (moveRCToyCar.py) to use your customized URL provided by Bluemix (change the variable called `webAppURL`) and the port your Arduino is connected to (arduinoPort).
      If you don't know which port your Arduino is connected to, just open the Arduino IDE, click on the **Tools** menu, and look for the checked option on **Serial Port**.

2. Once you have finished changing the code, it's time to run it!
   `$ python moveRCToyCar.py -t 0.12`

   **Note:** You can use the `-t` flag to set the level of "relaxation" you have to accomplish to move the car. You can also omit it and it will default to 0.35. (You can also change this in the code.) If everything went well, you should expect to see something like the following:
Setting threshold to 0.12
Connecting to arduino on /dev/ttyACM0 at 115200 baud
Connected!
Message to Arduino: <STOP> 0
Message to Arduino: <STOP> 0
Message to Arduino: <STOP> 0
Message to Arduino: <STOP> 0
Message to Arduino: <STOP> 0

3. When you want to stop, press Enter before you disconnect the Muse or the Arduino. You will see the following:

Closing everything...
Waiting for threads to close...
Message to Arduino: <STOP> 0
Arduino thread closed!
Local server thread closed!
Waiting for OSC server to close...
OSC server closed!

**Note:** If you forget to press Enter before you disconnect the Muse or the Arduino, you will probably have to kill the Python process. If you are using a Unix-like system (such as OSX or Ubuntu), just run the following command and it will close any Python applications running at the moment:

$ sudo killall python

**Step 5. Use R to analyze the data stored on Cloudant and dashDB**

**Note:** For this part, you need a Cloudant account and the dashDB service on Bluemix.

1. Sync data from Cloudant.
   a. Enter the information for your Cloudant account and database.
   b. Choose a name for the table to be created.
   c. Select **Start Sync**.

2. To run the R script, go to **Analyze/Develop R Scripts** in dashDB's menu, where you have the options to create and import script.
   The following R script, which can be found in the repository, fetches the data from a table named SAMPLES and plots a line chart using one of the fields.
library(bluR)
mycon <- bluConnect("BLUDB", ",", ",")
bluAnalyticsInit(mycon)

datamuse <- as.data.frame(blu.data.frame("BLU11196"."SAMPLES")[, c("C0","C1","C2","C3","F","S0","S1","S2","S3","TIMESTAMP","_ID","_REV")])

plot(datamuse[1:nrow(datamuse),1],type="l")

For this scenario, BLU11196 is the database's user, and SAMPLES is the name of the table.

Take a look at the plotted graph of the Muse data using R:
Conclusion

In this tutorial, you have learned how to extract brain wave signals from the Muse headband. You’ve also learned how to interpret this data using a Python program, which has the logic to make a toy car move forward or not. The tutorial has also shown how you can create a web application in Node.js on Bluemix to display in real time the brainwave activity, as well as the threshold that has been coded to make the toy car move. You may argue this is not really a Telekinesis demo since you are not really making the toy car move by thinking about it. Instead, the toy car is being moved by how relaxed you get, because we are detecting your alpha brain waves. In spite of this argument, one thing is a fact—your brain waves are making the car move!

The Cloudant NoSQL service provides access to a fully managed NoSQL JSON data layer that’s always on. The dashDB service helps you move your data into
a next-generation columnar in-memory database, run complex analytical queries with in-database algorithms, and integrate with analytic and business intelligence tools. The SDK for Node.js runtime helps you develop, deploy, and scale server-side JavaScript apps with ease.
About the authors

Thiago Domingues
Follow me on G+

Lucas Lima
Follow me on G+

Raul F. Chong
@raulchong

© Copyright IBM Corporation 2014, 2015
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)