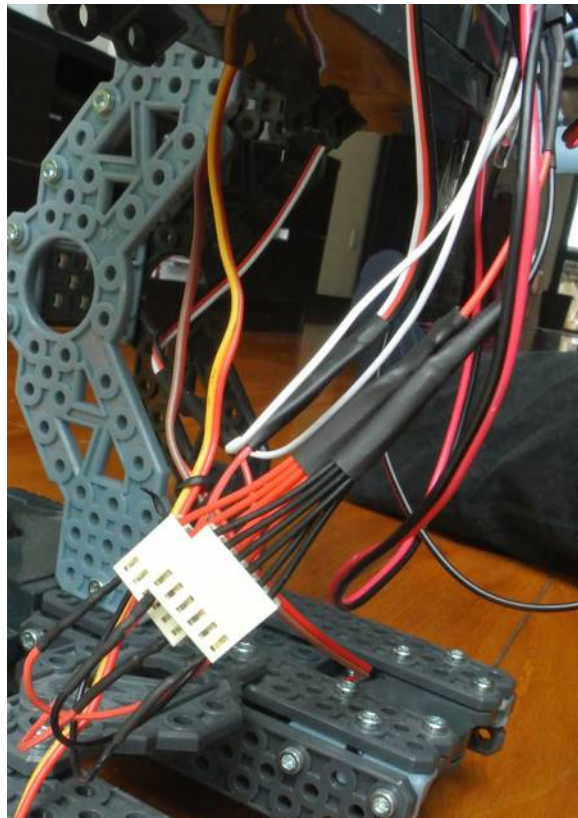
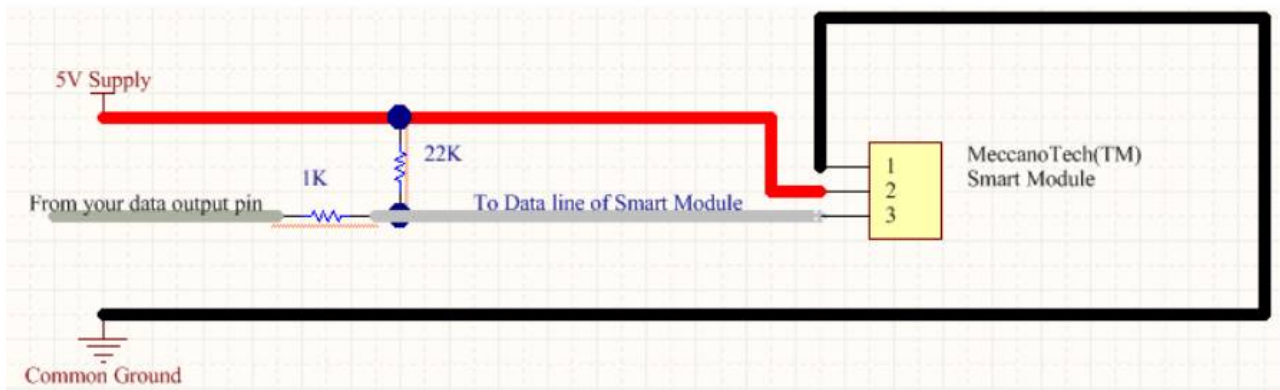


## Wiring and coding to control the Meccanoid's servos and LEDs.

Wiring and coding is different from T. Volkova's code because the G15 has less servos and I chose to control them differently, using the wires from MECCONTROL. 1 type of wire is used to duplicate the ground and 1 type of wire has the resistors integrated as advised by the Meccanoid servo library.



Wire duplication (for ground, in black on the picture) Duplication for later projects

To effectively control the Meccanoid servos from the motorshield V1, the documentation is very helpful to make the right connections.

## ***What pins are not used on the motor shield?***

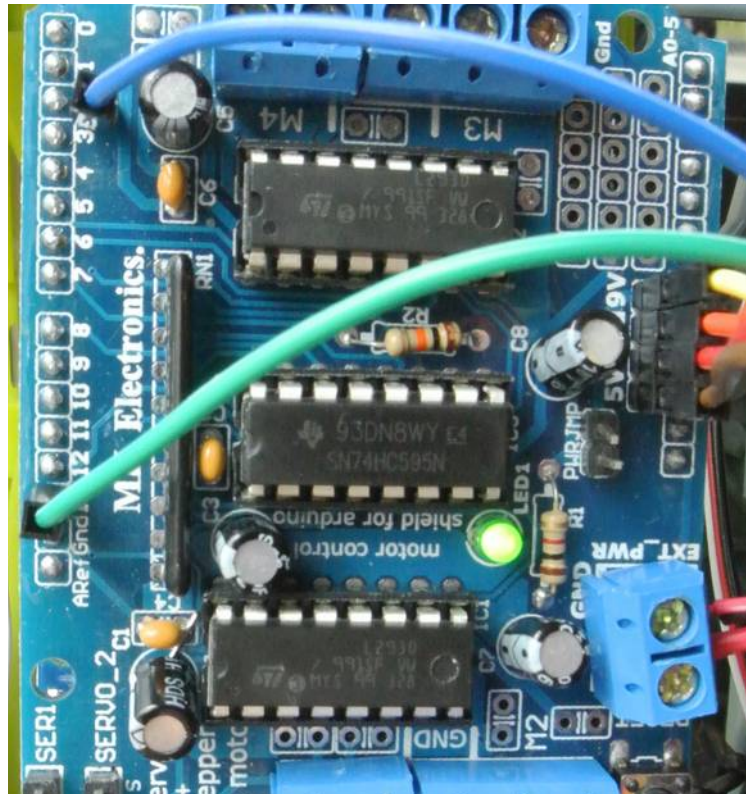
***All 6 analog input pins are available. They can also be used as digital pins (pins #14 thru 19)***

***Digital pin 2, and 13 are not used.***

So, Digital pin 2 and 13 would be used to control the servos and LEDs of the Meccanoid from the motorshield. At least, that's my understanding.

Pin 2 (in blue)

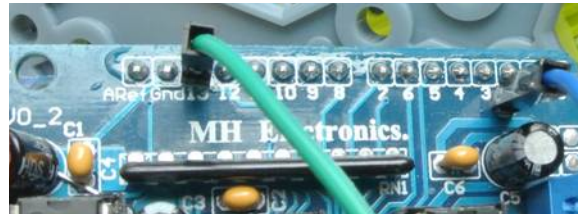
Pin 13 (in green)



+5V and ground for the Meccanoid's servos (only 2 cables effectively used)

```
const int chainPin1 = 13; // chainPin1 used for Left Arm and Head LEDs: LEDs @the end of the daisy chain. Pin 13 used with Motor Shield V1 Clone
const int chainPin3 = 2; // Pin 2 used with Motor Shield V1 Clone
```

The Green cable is directly welded on the Pin 13 of the motorshield board. The green cable lead to the Left Arm wire with resistors from MECCONTROL, then an extension to connect to the LEDs in the head.



The Blue cable is welded on a free hole of the board, connect to the Pin 2 on the PCB. It will be connected to a wire including resistors from MECCONTROL to control the right arm.

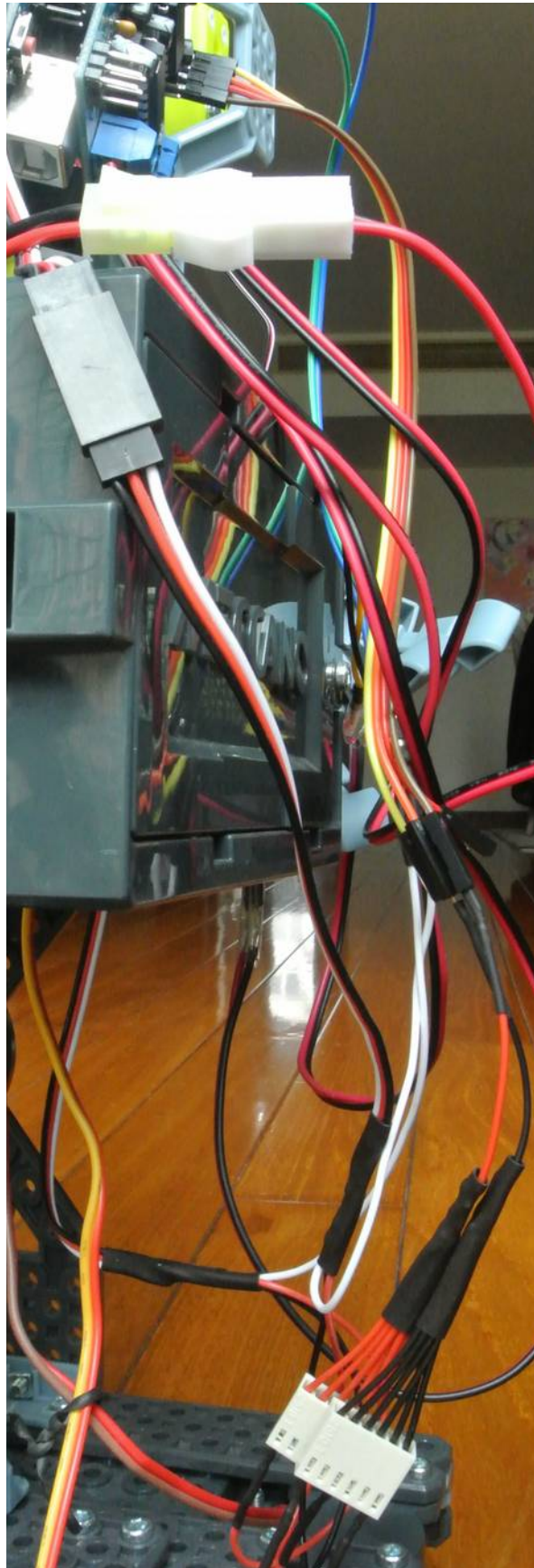


Above: original Adafruit Motorshield with Pi2 “wired/connected/ to hole  
Below: Clone with blue cable welded on the hole



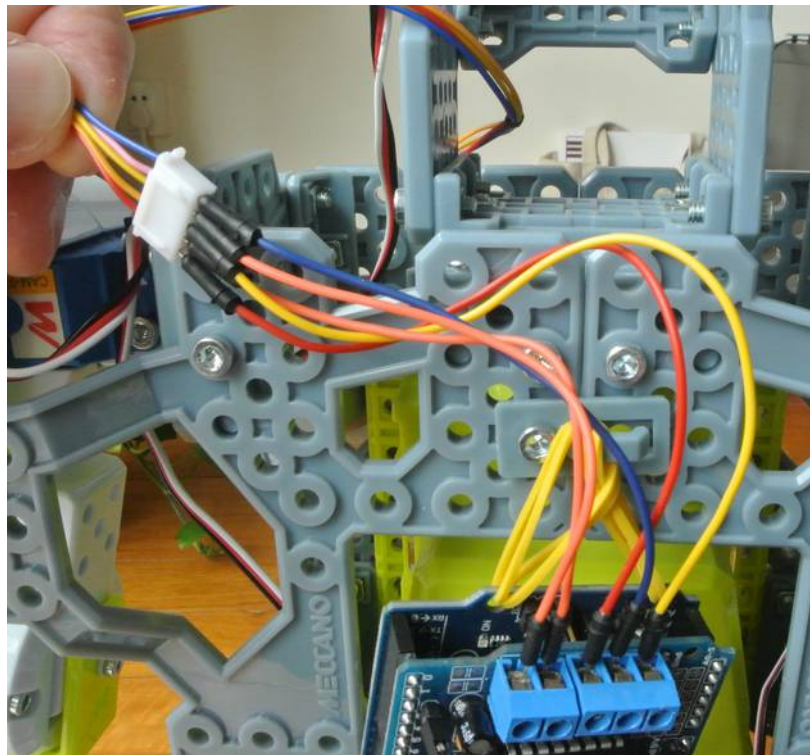
The brown cable will power the servos in the arms with +5V. It is connected to a jumper wire to 6-way socket from MECCONTROL (only 2 are used. I don't know whether it makes sense in terms of power distribution). The orange cable is the ground. It is connected to a jumper wire to 6-way socket, used to share the ground across the different servos.

Blue & Green wires going down, connected to 6way jumper to control the servos and LEDs. Brown wire connected to 6way jumper to give +5V. Orange wired connected to 6way jumper to give ground. C battery box from Meccanoid connected to Motorshield power pin with Tamiya connectors.



## Wiring and coding to control the stepper motor.

The stepper motor was added just because it could! It doesn't do anything interesting yet for the robot, but the connection were available on the motorshield and I had in with the Arduino kit. I will make something out of it later. It is a "28BYJ-48, 5volts, DC". Electric wiring is a bit of hit and miss since I didn't go through the various reverse-engineering method described on the Internet. Colors of the connection cables could have been in line with the motor cables color, I just did had the full set.



The stepper motor uses the same control library as the DC motors, already declared. So, it is just a matter of initializing it. Useful information is available at <http://robocraft.ru/files/datasheet/28BYJ-48.pdf>

```
// Declare additionnal stepper motor, from Arduino box, 28BYJ-48 5V DC
AF_Stepper step_motor(48,2); // create motor connected on M3 and M4, so port
2,GROUND=RED. Not such if that's correct for the time being; It is a 64step motor 360/5.6

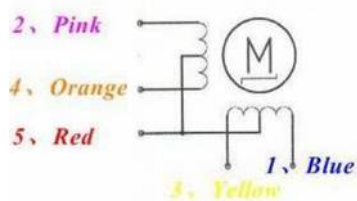
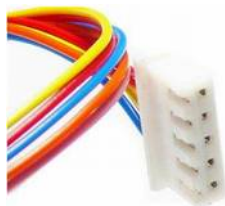
// Stepper motor
step_motor.setSpeed(500); // rpm test

step_motor.step(1000, FORWARD, DOUBLE);
step_motor.release();
delay(1000);

// START OF STEPPER MOTOR

// step_motor.step(500, FORWARD, SINGLE);
// step_motor.step(500, BACKWARD, SINGLE);
// delay(100);

step_motor.step(500, FORWARD, DOUBLE);
step_motor.step(500, BACKWARD, DOUBLE);
delay(100);
```



## **That's All!**

That's all for the time being. I hope the explanations and the posted code will be enough for others people to understand and recreate/improve on this modification of the Meccanoid G15 robot. As it is, it doesn't do much except reinterpreting things already available on the Internet. It is enough to start playing!

In any case, thanks for the original code posted by T. Volokova that does most of the useful control. Also, again, using a different motorshield that would be stackable would allow to increase easily the possibilities of this build.

# Appendix

An image of the folder containing the libraries



AFMotor comes from “Adafruit-Motor-Shield-library-master.zip”, available from Adafruit, as library from the Motorshield V1. Meccanoid-library comes from “meccanoid-library.zip”, available at the Meccano website.

The code (in case you get this document, but not the code file)

```
#include <MeccaBrain.h>

#include <AFMotor.h> // Added by Laurent Malod-Panisset, to use a Chinese clone of AdaFruit MotorShield V1. Use
Adafruit Lib. Next time, buy the real AdaF V2!

//

//Originally from TatyanaVolkava @GitHub, https://github.com/robotscity/meccanoid\_arduino\_demo
//Simplified for myself, Laurent Malod-Panisset. Work from TatyanaVolkava put as comments when not used
//

// From Tatyana Volkava
//To try this example you'll need the following:
//0) Meccanoid G15 in its human form
//1) Any Arduino
//2) Motor shield or motor driver, to drive robot's wheels. I tried the one based on L293D and it was quite decent
//3) Tamiya Mini female connector with wires - to connect to the robots' battery
//4) Male connectors: CWF-2 to connect one motor and PLS-2 to connect another
//5) 3 resistors of 22 kOhm and 3 resistors of 1 kOhm - to make pullup resistors (according to the manual)
//6) Download the library: http://cdn.meccano.com/open-source/meccanoid-library.zip
//7) You can also check out the reference material, to understand, how do the Smart Modules work
```



```
// http://cdn.meccano.com/open-source/Meccano\_SmartModuleProtocols\_2015.pdf

// Currently, Laurent Malod:

// Motor shield is a chinese clone from MH Electronics, MotorShield V1 clone. Library is the old AdaFruit Motorshield V1 one.

// Tamiya Mini female connector is used, bought at HobbyKing.com

// Male connectors not implemented, using Arduino prototype wire.

// Using servo cables extenders from MECCONTROL

// Using servo cables with resistors from MECCONTROL

// Pins to connect Meccanoids' servos, where chain 1 is left arm, chain 2 is head and chain 3 is right arm

// pins can be any digital pins, not necessary PWM

const int chainPin1 = 13; // chainPin1 used for Left Arm and Head LEDs: LEDS @the end of the daisy chain. Pin 13 used with Motor Shield V1 Clone

const int chainPin3 = 2; // Pin 2 used with Motor Shield V1 Clone

// Declare the motors, using AdaFruit example for V1 version

AF_DCMotor left_motor(1, MOTOR12_1KHZ); // create motor #1, 1KHz pwm, LEFT

AF_DCMotor right_motor(2, MOTOR12_1KHZ); // create motor #2, 1KHz pwm, RIGHT

// Declare additional stepper motor, from Arduino box, 28BYJ-48 5V DC

AF_Stepper step_motor(48,2); // create motor connected on M3 and M4, so port 2, GROUND=RED. Not such if that's correct for the time being; It is a 64step motor 360/5.6

// Using the motor shield to power the driver, so no need to use Pin 10, 11, 12, 13 to control them like T. Volkova.
Removed from the program

MeccaBrain chain1(chainPin1); //each chain allows to plug up to 4 smart modules

//MeccaBrain chain2(chainPin2); // NO HEAD FOR G15 Meccanoid, HEAD LIGHT daisy chained on the Left Arm.

MeccaBrain chain3(chainPin3);

// Commands to drive the robot's motor, with MotorShield V1
```

```
void rotateLEFT(int speed)
{
    left_motor.setSpeed(speed); // Set the speed, max 255
    right_motor.setSpeed(speed); // Set the speed, max 255
    left_motor.run(BACKWARD);
    right_motor.run(FORWARD);
}
```

```
void turnLEFT(int speed)
{
    left_motor.setSpeed(speed); // Set the speed, max 255
    right_motor.setSpeed(speed); // Set the speed, max 255
    left_motor.run(RELEASE);
    right_motor.run(FORWARD);
}
```

```
void rotateRIGHT(int speed)
{
    left_motor.setSpeed(speed); // Set the speed, max 255
    right_motor.setSpeed(speed); // Set the speed, max 255
    left_motor.run(FORWARD);
    right_motor.run(BACKWARD);
}
```

```
void turnRIGHT(int speed)
{
    left_motor.setSpeed(speed); // Set the speed, max 255
    right_motor.setSpeed(speed); // Set the speed, max 255
    left_motor.run(FORWARD);
    right_motor.run(BACKWARD);
}
```

```
void goFORWARD(int speed)
```

```

{
  left_motor.setSpeed(speed); // Set the speed, max 255
  right_motor.setSpeed(speed); // Set the speed, max 255
  left_motor.run(FORWARD);
  right_motor.run(FORWARD);
}

void goBACKWARD(int speed)
{
  left_motor.setSpeed(speed); // Set the speed, max 255
  right_motor.setSpeed(speed); // Set the speed, max 255
  left_motor.run(BACKWARD);
  right_motor.run(BACKWARD);
}

void relax()
{
  left_motor.run(RELEASE);
  right_motor.run(RELEASE);
  delay(10);
}

//Joints mapping. This is from T. Volkova, modified
//Chain 1 - Left Arm. 1.0 is Arm Pitch, 1.1 is Arm Roll, 1.2 is Elbow // CHANGED FROM T.VOLKOVA: CHAIN 1
ARMS -> 1.0 Left Arm Pitch, 1.1 Left Arm Roll, 1.2, Right Arm Pitch, 1.3 Right Arm Roll
//Chain 2 - Head. 2.0 is Head Yaw, 2.1 is Head Roll, 2.2 is LEDs // CHANGED: NO HEAD on my model
//Chain 3 - Right Arm. 3.0 is Arm Pitch, 3.1 is Arm Roll 3.2 is Elbow // CHANGED CHAIN

//const byte LEFT_ARM_PITCH=0; NO ARM_PITCH FOR THE SMALL G15 MECCANOID
const byte LEFT_ARM_ROLL=1;
const byte LEFT_ARM_ELBOW=2;
const byte RIGHT_ARM_ROLL=6;
const byte RIGHT_ARM_ELBOW=7;

```

```
// T. Volkova's code, simplified because my robot doesn't have the mobile head
```

```
//jointName is LEFT_ARM_ROLL etc
```

```
//pos is 0...255
```

```
void setJoint(byte jointName, byte pos)
```

```
{  
  switch(jointName){  
    case LEFT_ARM_ROLL:  
      //chain1.setServoPosition(1, pos);  
      chain1.setServoPosition(0, pos);  
      chain1.communicate();  
      break;  
    case LEFT_ARM_ELBOW:  
      //chain1.setServoPosition(2, pos);  
      chain1.setServoPosition(1, pos);  
      chain1.communicate();  
      break;  
    case RIGHT_ARM_ROLL:  
      //chain3.setServoPosition(1, pos);  
      chain3.setServoPosition(0, pos);  
      chain3.communicate();  
      break;  
    case RIGHT_ARM_ELBOW:  
      //chain3.setServoPosition(2, pos);  
      chain3.setServoPosition(1, pos);  
      chain3.communicate();  
      break;  
  }  
}
```

```
// T. VOLKOVA's code
```

```
//Set the color of eye LEDs. red, green and blue are from 0 to 7 (0 - no color, 7 - max color).
```

```
//fadetime is from 0 to 7 and means the speed of color change (0 - immediate change, 7 - longest change)
```

```
//example: setColor(7,0,0,3) means change color to red with average speed
```

```
void setEyesColor(byte red, byte green, byte blue, byte fadetime)
```

```
{  
    chain1.setLEDColor(red, green, blue, fadetime);  
    chain1.communicate();  
}
```

```
//Servo colors
```

```
const byte JOINT_BLACK=0xF0;
```

```
const byte JOINT_RED=0xF1;
```

```
const byte JOINT_GREEN=0xF2;
```

```
const byte JOINT_BROWN=0xF3;
```

```
const byte JOINT_BLUE=0xF4;
```

```
const byte JOINT_VIOLET=0xF5;
```

```
const byte JOINT_SEA=0xF6;
```

```
const byte JOINT_WHITE=0xF7;
```

```
// T. VOLKOVA's code. Some parts removed due to different robot
```

```
//set the servo color
```

```
//for example, setJointColor(RIGHT_ARM_ELBOW, JOINT_VIOLET)
```

```
void setJointColor(byte jointName, byte color)
```

```
{  
    switch(jointName){  
  
        case LEFT_ARM_ROLL:  
            chain1.setServoColor(0, color);  
            chain1.communicate();  
            break;  
  
        case LEFT_ARM_ELBOW:  
            chain1.setServoColor(1, color);  
            chain1.communicate();  
            break;  
  
        case RIGHT_ARM_ROLL:  
            chain3.setServoColor(0, color);
```

```
    chain3.communicate();
    break;
case RIGHT_ARM_ELBOW:
    chain3.setServoColor(1, color);
    chain3.communicate();
    break;
}
}

// Arduino initialization & check

void setup() {
    pinMode(chainPin1, OUTPUT);
    pinMode(chainPin3, OUTPUT);

    Serial.begin(9600); // Set Up the library for communication at 9600bps

    // DC motors
    //left_motor.setSpeed(200); // Set the speed to 200 out of 255 (200/255)
    //right_motor.setSpeed(200); // Set the speed to 200/255, same for the right motor. Could be modified later on for
    different purposes.

    Serial.print("Tick");
    //left_motor.run(FORWARD);
    //right_motor.run(FORWARD);
    goFORWARD(200);
    delay(1000);

    Serial.print("Tock");
    //left_motor.run(BACKWARD);
    //right_motor.run(BACKWARD);
    goBACKWARD(200);
    delay(1000);
```

```

Serial.print("Tack");

relax();

delay(1000);

// Stepper motor
step_motor.setSpeed(500); // rpm test

step_motor.step(1000, FORWARD, DOUBLE);
step_motor.release();
delay(1000);

// T. VOLKOVA's code
//"Discover" all the modules (make them blue-colored instead of green-colored)
//for some unknown reason, I have to repeat it from time to time
for (int i = 0; i < 50; i++)
{
    chain1.communicate();
    chain3.communicate();
}

//delay to be sure that all modules are ready

//if some module is "not discovered" than it will remain green and later this module will behave strangely
delay(2000);
}

void loop() {

// From T. Volkova
//make every joint of the robot a different color just for fun
//so that we cycle through all the possible colors and joints
for(int i=0; i<7; i++)
{
    setJointColor(i, JOINT_BLACK+i);
}
}

```

```
}
```

```
//bring both arms down. They are inverted, so that left arm mirrors right arm  
//setJoint(LEFT_ARM_ROLL, 255); Meccanoid was a bit violent with himself,  
//setJoint(RIGHT_ARM_ROLL, 0); so i changed the values a bit
```

```
setJoint(LEFT_ARM_ROLL, 205);  
setJoint(RIGHT_ARM_ROLL, 50);  
delay(2000);
```

```
//set both elbows in middle position  
setJoint(LEFT_ARM_ELBOW, 127);  
setJoint(RIGHT_ARM_ELBOW, 127);  
delay(2000);
```

```
// Changed also from 0 and 255 to 20 and 235  
setJoint(LEFT_ARM_ROLL, 20);  
setJoint(RIGHT_ARM_ROLL, 235);  
delay(3000);
```

```
//bring both arms to the middle  
setJoint(LEFT_ARM_ROLL, 127);  
setJoint(RIGHT_ARM_ROLL, 127);  
delay(2000);
```

```
//move elbows as if he was fighting  
setJoint(LEFT_ARM_ELBOW, 127);  
delay(2000);
```

```
setJoint(LEFT_ARM_ELBOW, 50);
```



```
setJoint(RIGHT_ARM_ELBOW, 127);
delay(2000);

setJoint(LEFT_ARM_ELBOW, 127);
setJoint(RIGHT_ARM_ELBOW, 205);
delay(2000);

setJoint(LEFT_ARM_ELBOW, 50);
setJoint(RIGHT_ARM_ELBOW, 127);
delay(2000);

//set both elbows in middle position
setJoint(LEFT_ARM_ELBOW, 127);
setJoint(RIGHT_ARM_ELBOW, 127);
delay(1000);

//bring both arms down
setJoint(LEFT_ARM_ROLL, 205);
setJoint(RIGHT_ARM_ROLL, 50);
delay(1000);

// USE DRIVER MOTORS
rotateRIGHT(200);
delay(2000);
rotateLEFT(200);
delay(4000);
rotateRIGHT(200);
delay(2000);

relax();

// START OF STEPPER MOTOR
```

```
// step_motor.step(500, FORWARD, SINGLE);
// step_motor.step(500, BACKWARD, SINGLE);
// delay(100);

step_motor.step(500, FORWARD, DOUBLE);
step_motor.step(500, BACKWARD, DOUBLE);
delay(100);

step_motor.step(500, FORWARD, INTERLEAVE);
step_motor.step(500, BACKWARD, INTERLEAVE);
delay(100);

step_motor.step(500, FORWARD, MICROSTEP);
step_motor.step(500, BACKWARD, MICROSTEP);
delay(100);

// From T. Volkova
//loop through all the possible eyes colors
//We have 3 bits per channel
for(int j=0; j<128; j++)
{
    byte red = (j >> 4) & 0x07;
    byte green = (j >> 2) & 0x07;
    byte blue = j & 0x07;
    setEyesColor(red, green, blue, 0);
}

}
```