SUPERNOVA

Robotics And Design Laboratory

Samuele Altruda
Matteo Scarpello
Isabel Hernandez
Stefano Cerri
Rossy Yang

POLITECNICO DI MILANO

Scuola di Ingegneria Industriale e dell'Informazione

Computer Science and Engineering

SUPERNOVA

Engineers: Altruda Samuele, Cerri Stefano, Matteo Scarpello
Designers: Hernanez Isabel, Yang Rossy

Academic year 2015 - 2016

Contents

List of Figures

Chapter 1

Introduction

1.1 Inspiration

During the first week of the course we attended 5 lessons in which we understood the problems and the techniques related to project and build a robot. During the lessons our team was submitted some hard test to deepen the field of robotics toy and to understand the phases of building a robot. We tried to move a 4 DOF small robot arm, we disassembled a robotic toy, and finally we was ready to face a new challenge.

We were asked to build a robot, a dancing robot. We had the possibility to choose a style of dancing and a related kind of music. We choose house music as we know some artists and we take inspiration from their concert choreography. We noticed that during the concert the environment and all the atmosphere is conveyed to create an alienation of the audience with light and loud music. The Dj is a small figure, a little and dark shape on the stage, the leading factor is the music and the light itself. There are stunning and amazing games of light that flash the stage and the audience, that dance following the rhythm given by a deafening music basses. The main movements of the crowd is an arm shaking that resemble the VU meter bass vibration.

We tried to create a robot infusing in it the features mentioned above, creating a small Dj. The main objective was to make a show constituted by light, music and laser. We choose an alien figure to reveal the real soul of the music, which is capable to paint an addictive world of joy. The robot must be a small shape, which lead the audience using his unique extraordinary power: a rhythmic arm.

After we finished the work, we decided to give a name to our creation: SUPERNOVA.

A supernova is an astronomical event that occurs during the last stellar evolutionary stages of a massive star's life, whose dramatic and catastrophic destruction is marked by one final titanic explosion. For a short time, this causes the sudden appearance of a 'new' bright star, before slowly fading from sight over several weeks or months. Our robot spreads lights as the 'new' star in all direction and it is surrounded by loud music as several explosion.

Supernovae are more energetic than novae. In Latin, Nova means "new", referring astronomically to what appears to be a temporary new bright star. Adding the prefix "super-" distinguishes supernovae from ordinary novae, which are far less luminous. As supernovae, our robot is new, and it is brighter than other ordinary robots.

Chapter 2

Building report

In this section we illustrate the building process, for the electronic boards and the structure.

2.1 Electronics building report

First we designed the PCB layout on Cadsoft Eagle [1] . After a careful check of all connection on the schematic we proceed with the routing of the board. We decided to use manual route to avoid some problem which automatic routing creates and to keep some constraints that results in better signal on the board.

After that we printed by laser printer the circuit on a gloss paper and with the iron we attach the circuit on the copper board. We etch the circuit using hydrogen peroxide and muriatic acid to reduce the cost of PCB producing[2] , leaving the copper board for 20 minute in the solution at 50°C to accelerate the process.

Then we soldered all the component avoid overheating of the more susceptible components, keeping the solder temperature under 350°C. After a careful check of all the route we tested the board produced with a multimeter evaluating the resistance of the routes, and the connection.

Finally we assembled all the remaining component, like the MSGEQ7 on its socket, and the Molex connector, and we tested the system with the Arduino to check the firmware and the hardware together.

In the first part of the project development we used a bench power supply to provide the right voltage to the different components of the circuits. As final step we designed a power supply recycling a laptop one. We added a led and a switch. The laptop PSU has 65W power at 18.5V, so after that we built a step down module composed by 3 module to provide 6V, 5V, and 12V, enclosing them in a small box.

[1] We used the Eagle Cad 7.5 Professional

[2] Before there's much copper dissolved in the solution, Cu + 2 HCl + H2O2 -> CuCl2+ 2H2O is the dominant net reaction. That is, the extra oxygen in solution from the peroxide is oxidizing the copper metal, in presence of the acid, to make copper chloride. That's our starter etchant. The resulting CuCl2 should be a nice emerald green color.

After you've dissolved a lot of copper into the solution, and used up all the peroxide, the copper chloride does most of the etching for you: CuCl2 + Cu -> 2 CuCl. That's the end etchant.

Eventually you etch so much that you convert all the CuCl2 into CuCl, which doesn't dissolve copper (and is a yucky brown color). As long as you've got enough acid in the solution, you can simply add more oxygen to re-oxidize the copper, making more copper chloride and water: 2 CuCl + 2 HCl + O -> 2 CuCl2 + H2O. And then you can etch again.

2.2 Structure building report

2.2.1 Concept

We start with the concept of a robot taking in account the music we have chosen. After that we analysed the kind of music, we noticed what elements the artist have in common, we selected some features like headphones, lights and anonymous face in to the crowd. We analyse the movements of the Djs, how they motivate people, how is the reaction of the audience. We observe the people loose in the music and starting dance in a free way, with these elements we created the movements and the mechanism of the robot, like the heart, the hands, the arms and the expression of the eyes. We understood how the lights are a important part of a Dj show. We were looking for a clear, simple and minimal design that resemble a Dj. The image of the Dj needs to be clean, but at the same time he need to draw the attention of the crowd. For this reason we choose the black as base color for the robot leaving all expression to lights and color.

2.2.2 Building

The robot is composed by a head, 2 arms, 2 soft cover pieces, 2 hard shell pieces, a “heartbeat” piece, an internal frame structure and electronic components.

The internal structure part was mainly built by a MDF panel. Its shape was defined by the curve of lateral view of robot’s external body. The function of the panel is a frame structure, that helps all the electronic components to be fixed in the proper positions while lets the robot remain to a fine look. The panel also connects the joints of the head and the 2 arms.

The external part was created by two main parts: the upper soft part and the lower hard part. The soft part allows the robot’s head and arm to move freely and let the technician free to adjust the internal components so as to have an easy maintenance. Lower hard part which offers the function of protection and supporting.It defines the good shape of the design, and also makes the “heart beat” piece have a place to hold and to work.

The head and the hard lower external parts were produced by PET plastic thermal transform process. They both were separated into two parts because of the process of the prototype, and then they were glued together to make a close space to hold the components.

The soft upper external part and the two arms were produced by EVA plastic thermal transform process. As for the hard part, they also were separated into two parts because of the process of the prototype, and then depends on the properties of the material, we chose the way of sewing and to use the velcro to create a fine close space.This also allows technician to maintain the robot easy to modify.

Figure 2.1: Robot

Chapter 3

Project specification

In this section we want to make a detailed explanation of the technical specification of the robot in terms of electronics, Arduino's firmware and structure.

3.1 Electronics Technical Specifications

3.1.1 Light board

The first board that we built was the Arduino shield which control the lights show (Fig. 3.1).

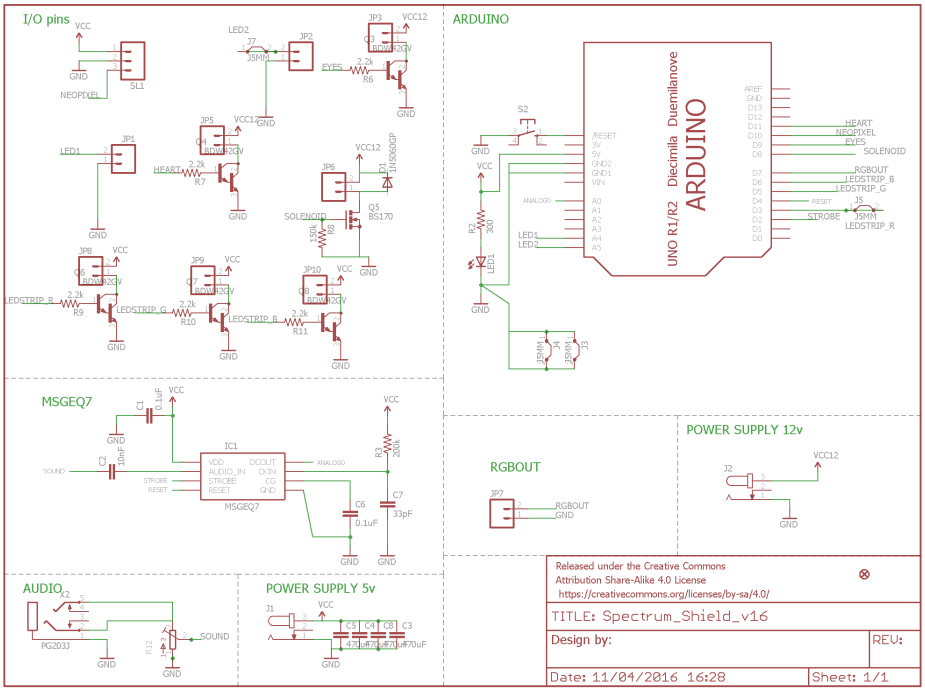
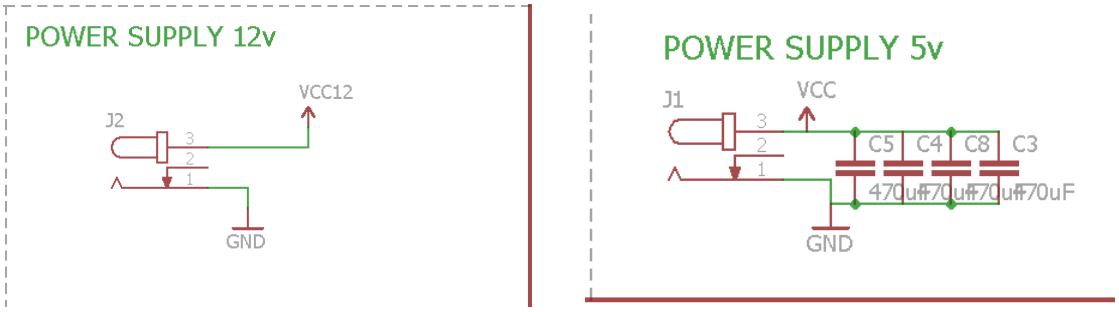


Figure 3.1: Schematic of all system

The system is characterized by 7 blocks that are illustrated in the following pages.

3.1.1.1 Power supply

The first and most important block is the power supply 3.2. We decided to use DC jack as usual standard, as said before. The 5 volt were more critical as they will supply the MSGEQ7, which suffer from voltage oscillation, so we choose to insert 4 capacitor: 2 of 470μF as charge reserve for high current element as the Neopixel matrix. Then one capacitor of 220μF and ones of 10μF, to cut off higher frequency noise.



(a) First figure

(b) Second figure

Figure 3.2: Left: 12V Power supply. Right: 5V Power supply

3.1.1.2 MSGEQ7

In this part of the schematic (Fig 3.3) we designed the conditioning circuit to let the MSGEQ7 IC working properly following the instruction provided in its datasheet [1].

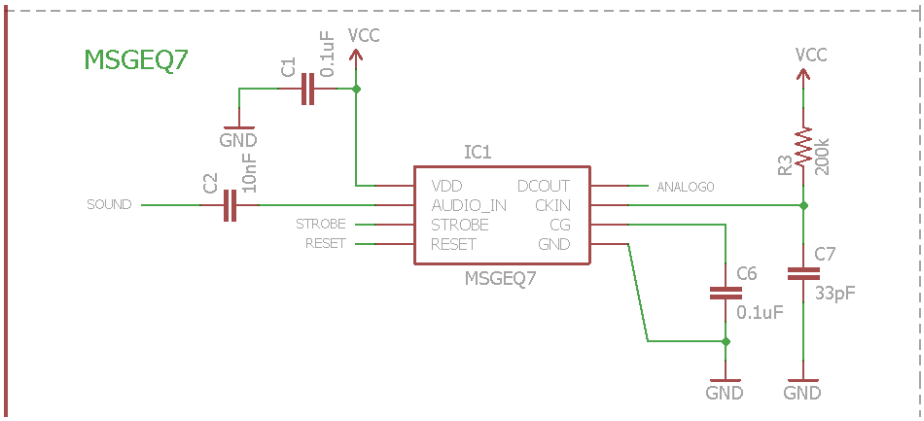


Figure 3.3: MSGEQ7 conditioning circuit

The capacitor between VCC and GND cut off the high frequency of noise affecting the voltage. The 10nF capacitor decoupling the audio signal from DC. The 200K and 33pF assure a perfect RC oscillating branch. The MSGEQ7 is connected to digital pin 4 (reset), digital pin 2(strobe), analog pin A0.

[1] https://www.sparkfun.com/datasheets/Components/General/MSGEQ7.pdf

3.1.1.3 Audio

We use a audio jack (Fig 3.4) to provide audio signal to the board and we use a simple 10K trimmer to regulate the power of the signal. The audio signal is directly provided to the MSGEQ7 module, which analyse the sound and give raw data of spectral density to Arduino.

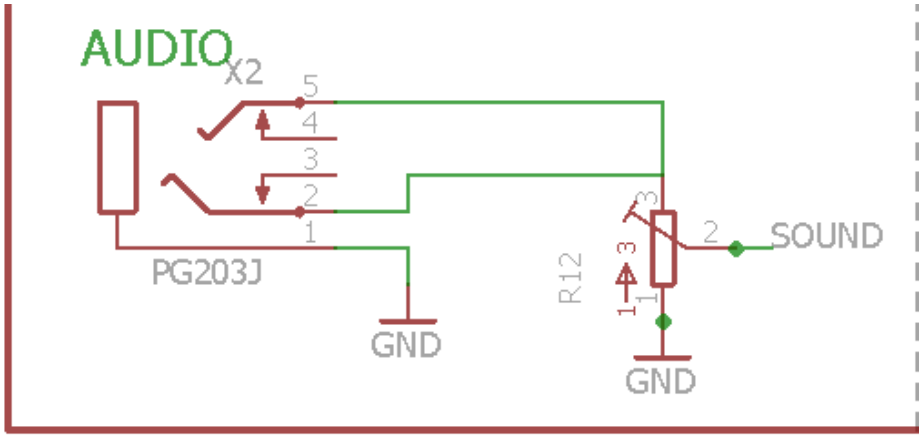


Figure 3.4: Audio path

3.1.1.4 Signal out

This Molex male connector is used to provide a simple digital connection for external devices (Fig 3.5), which could operate synchronously with the system. The signal is generated by Arduino taking in account the value of spectral density of bass frequency (around 60Hz). The signal is high over a threshold, and low under that. The signal out is provided by digital pin 7.

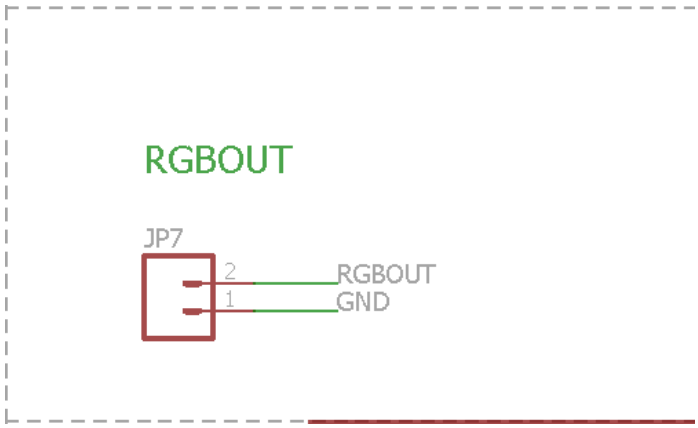


Figure 3.5: Molex connector for bass signal

3.1.1.5 I/O Pins

The digital pin 10 of Arduino is connected to the data input of the Neopixel matrix, which is composed by 64 SK6812 [2] leds. The matrix needs also ground and 5 volts connections.

There are also 2 pins dedicated to laser. In particular we choose analog pin 4 and 5, using them as digital output. They provide the digital signal via a molex connection.

The solenoid is connected via molex connector between 12 volts and the drain of a BS170[3], putting in parallel a diode 1n4148[4] to prevent reverse high peak voltage, which can damage the circuit. The gate of BS170 is connected to digital pin 8.

The eyes and heart leds are connected as the previous via molex connector. They are put between 12 volts and the collector of a TIP120 darlington bjt[5]. A 2.2k resistor is connected between Arduino pins and the TIP120 base to prevent overcurrent damaging the Arduino ports. The two base are connected to digital pin 11 and 9, which let control the leds by PWM.

We designed also 3 molex connector to actuate also a RGB led strip, which is actually not connected. The RGB strip require as eyes and heart leds a TIP120 with a resistor of 2.2k. We reserved digital pins 1 5 6.

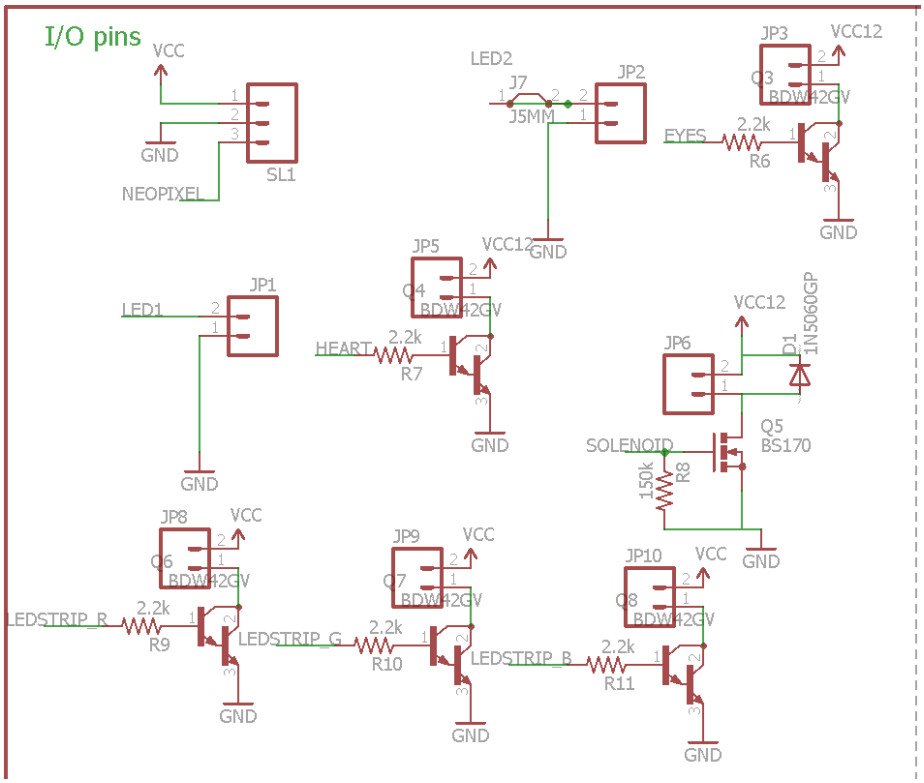


Figure 3.6: I/o pins schematic

[2] https://cdn-shop.adafruit.com/product-files/1138/SK6812+LED+datasheet+.pdf

[3] *https://www.fairchildsemi.com/datasheets/BS/BS170.pdf*

[4] *http://www.nxp.com/documents/data_sheet/1N4148_1N4448.pdf*

[5] https://www.fairchildsemi.com/datasheets/TI/TIP120.pdf

3.1.1.6 Arduino

We decided to design also a circuit for resetting the Arduino and a led to check the presence of the voltage.(Fig 3.7,Fig 3.8)

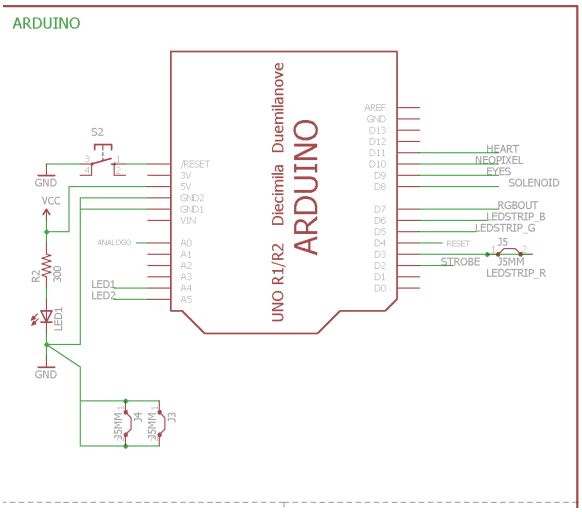


Figure 3.7: Arduino circuit

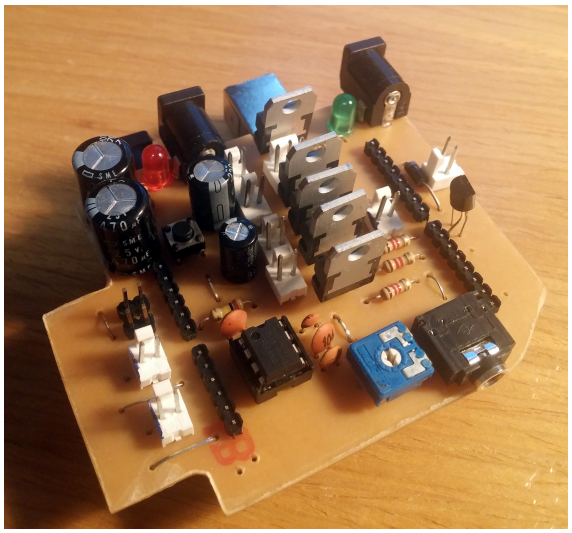


Figure 3.8: Final implementation of the circuit

In the next table we show all the Arduino ports, with the relative connections.

Arduino pin	Function
Reset	reset pushbutton
5v	5 VCC provided by external power supply
GND	GND provided by external power supply
A0	pin 3 MSGEQ7
A4	laser 1 anode
A5	laser 2 anode
D2	pin 4 MSGEQ7
D3	ledstrip pin R
D4	pin 7 MSGEQ7
D5	ledstrip pin G
D6	ledstrip pin B
D7	digital signal out
D8	gate BS170 driving solenoid
D9	base TIP120 driving led eyes
D10	Neopixel data out
D11	base TIP120 driving led heart

3.1.2 Sound board

The second board that we built was the Arduino shield which control the beat detection and servos movements. (Fig. 3.9).

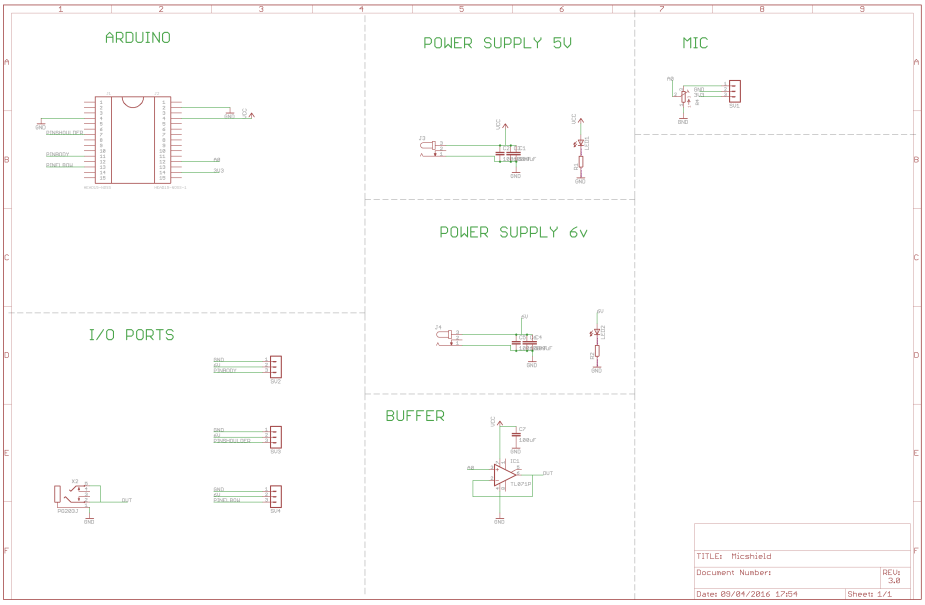
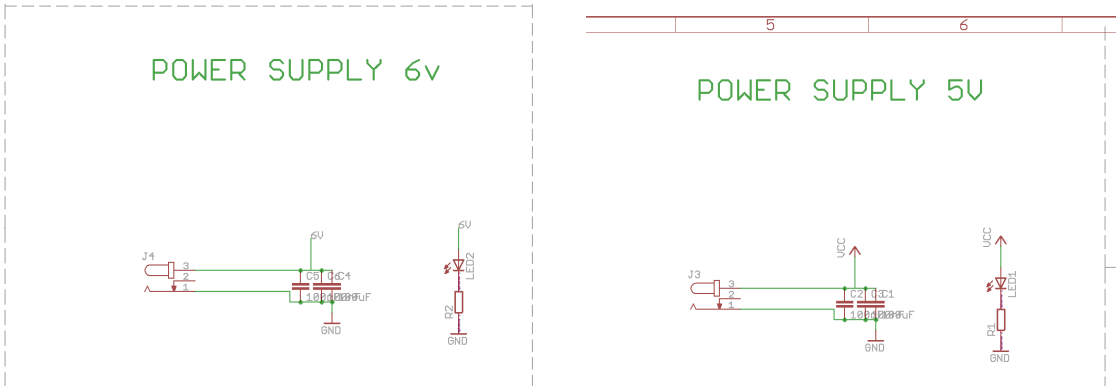


Figure 3.9: Schematic of all system

3.1.2.1 Power Supply

The first block is the power supply (Fig 3.10). We decided to use DC jack as usual standard for DC voltage. We decided to provide 2 voltages to the board: 6 volts to power the servos, and 5 volts for the Arduino. We put per each servo a 1000μF capacitor as charge reserve for high peak current request. For the 5 volts branch we put 3 capacitor of 100μF, 10μF, 100nF to cut off noise at every frequency. These capacitor were very important because the microphone connected to the Arduino suffer from power noise. We put a led in each branch of power lines to check the voltage.



(a) First figure

(b) Second figure

Figure 3.10: Left: 6V Power supply. Right: 5V Power supply

3.1.2.2 Microphone and Buffer

We use an electret microphone to capture the audio signal of music[6]. This kind of microphone is low noise and high performing. It is a module constituted by an on board amplification by a MAX4466[7]. The microphone results to be good at high music volume as characterized by low sensitivity.

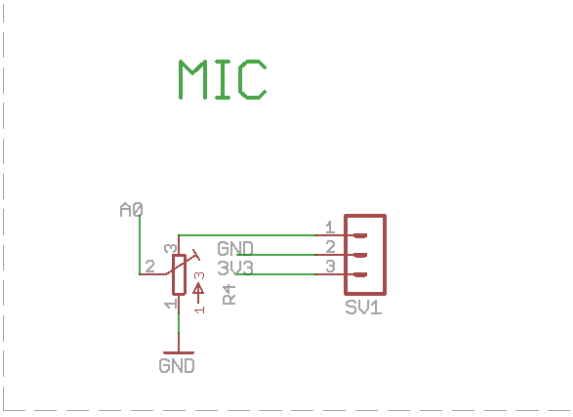


Figure 3.11: Microphone

[6]https://www.adafruit.com/product/1063

[7]https://www.adafruit.com/datasheets/MAX4465-MAX4469.pdf

We decided to put a 10k trimmer to regulate the power of the signal entering the boards. The audio signal is provided to the A1 analog pin of Arduino and also to the non inverting pin of the TL071[8]. The buffer is capable to keep the audio signal good, while it is travelling along long wire.

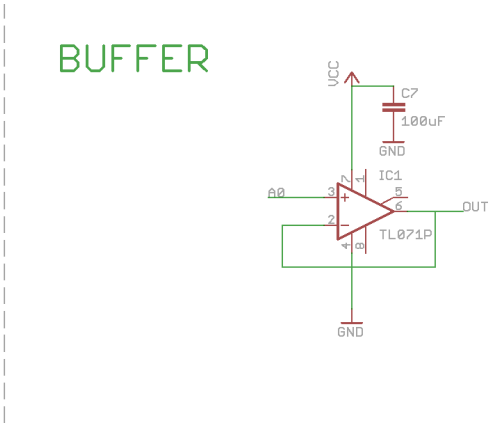


Figure 3.12: TL071 in non inverting buffer configuration

3.1.2.3 I/O ports

We use a audio jack (Fig 3.13) to provide audio signal to the lights board. We decided to use pin header for the connection of the servos to the board.

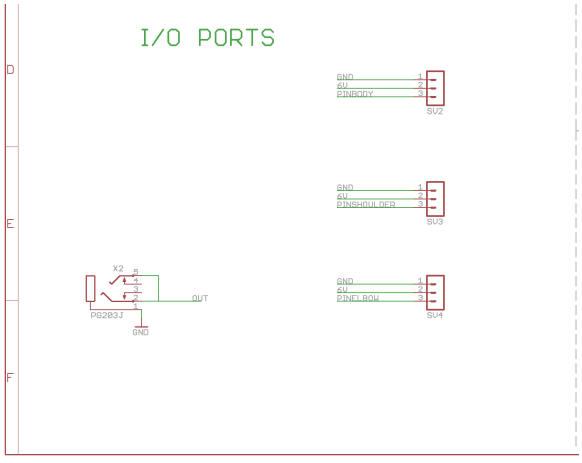


Figure 3.13: Audio input on the left. Pin header for servos on the right.

[8] http://www.ti.com/lit/ds/symlink/tl072.pdf

3.1.2.4 Arduino

The A0 analog pin is connected to the signal of the microphone. The 3.3V pin provide a constant low noise voltage to it. The digital pin 10 provides digital signal for the servo of the robot elbow, the pin 8 for the body servo and the digital pin 4 for the shoulder.

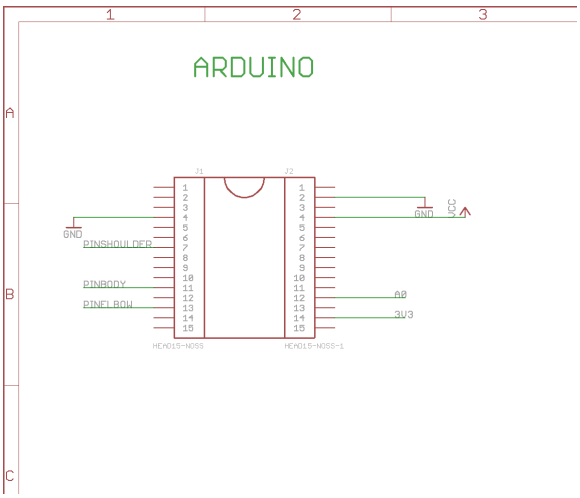


Figure 3.14: Arduino schematic.

Figure 3.15: Final implementation of the circuit

In the next table we show all the Arduino ports, with the relative connections.

Arduino pin	Function
5v	5 VCC provided by external power supply
GND	GND provided by external power supply
A1	audio signal in
D4	Shoulder servo
D8	Body servo
D10	Elbow servo

3.1.3 PSU

We designed a power supply system taking in account the fact that we have a total current consumption of max 3A. We decided to recycle a laptop charger to reduce cost and also for its specifications that fitted our needs perfectly (PSU visible in figure 3.16).

Figure 3.16: Laptop charger used. 65W per 3.5A max.

After the laptop charger we decided to split the 18.5V provided by the PSU to generate the 3 voltage needed. In particular we need 12V for the solenoid and all the led of the eyes and the heart, 5V to power the 2 Arduino and the MSGEQ7, and 6V to correctly make the servos work.

Figure 3.17: LM2596 module. It is a Simple High-Efficiency Step-Down (Buck) regulator.

We used 3 module of LM2596[9] (visible in figure 3.17). These module are simple to use, as they provide a multirotation potentiometer to reach fine voltage. Between them and the PSU we put a fuse to protect the system from short circuit. We choose the LM2596 module becacuse we wanted to limit the power dissipation derived from voltage regulation; typical components for voltage regulation abelonging to LM78** family are characterized by low efficiency as they implement linear regulation dissipating the overvoltage by heating. Our module is composed by a Buck system that using switching technology reduces the dissipation. The connection from the power box to the different board are made using DC jack, the standard for DC voltage.

3.2 Coding Technical Specifications

3.2.1 Sound board code

3.2.1.1 Beatdetection

The beat detection and the movement of the servos are in the sketch called 'beat_det.ino'. In order to keep track of the beat of the music we first use a DSP (Digital Signal Process) algorithm. It consists in four stages:

1. The signal is bandpass filtered to the range of 20-200hz
2. The absolute magnitude of the signal is calculated and it is passed through a low pass filter at 10hz
3. After detecting envelope is passed through a third bandpass filter tuned to 1.7-3.0hz
4. We rejected constant low frequency components, leaving only the drum kicks, the range 1.7-3.0hz corresponds to 100-180bpm, common tempos for electronic music

We implemented this approach but we saw that the algorithm missed some beats and, of course, it did not detect the beats when there is no bass frequencies. So we tried another algorithm[10] that instead using filters it consists of:

1. Detect sound energy variations by computing the average sound energy of the signal and comparing it to the instant sound energy.The average energy is not be computed on the entire song because some songs have both intense passages and more calm parts
2. The instant energy is compared to the nearby average energy, for example if a song has an intense ending, the energy contained in this ending shouldn't influence the beat detection at the beginning
3. We detect a beat only when the energy is superior to a local energy average. Thus we will compute the average energy on say: 44032 samples which is about 1 second, that is to say we will assume that the hearing system only remembers of 1 second of song to detect beat. This 1 second time (44032 samples) is what we could call the human ear energy persistence model. It is a compromise between being to big and taking into account too far away energies, and being too small and becoming to close to the instant energy to make a valuable comparison.

[9] http://www.ti.com/lit/ds/symlink/lm2596.pdf

[10] http://www.flipcode.com/misc/BeatDetectionAlgorithms.pdf

Then we improved it by keeping the energy values computed on 1024 samples in history instead of the samples themselves. With this approach we saw that the algorithm keeps track of the beats of the song better than the first approach.

3.2.1.2 Use of ATMEGA328 register for ADC

Instead of using the analogRead function we decided to use directly the registers of the ADC of AtMEGA328 in order to have a faster sampling for the beat detection algorithm. We use these registers:

- ADMUX |= (1 « ADLAR) to left align the ADC value so we can read highest 8 bits from ADCH register only. This, of course, decreases our resolution but at the same time increases the performance because we read only one register instead of two (ADCL is not read)
- ADCSRA |= (1 « ADPS2) | (1 « ADPS0) to set ADC clock with 32 prescaler 16 MHz/32 = 500kHz
- ADCSRA |= (1 « ADATE) to enabble auto trigger
- ADCSRA |= (1 « ADIE) to enable interrupts when measurement complete
- ADCSRA |= (1 « ADEN) to enable ADC
- ADCSRA |= (1 « ADSC) t0 start ADC measurements
- sei() to enable interrupts

We used the method ISR(ADC_vect), the Interrupt Service Routine, to sample the audio signal from the microphone and to detect the beat. Of course the algorithm should be very efficient and fast because otherwise a new ISR is called before the older one is not yet finished.

```
/* This code reads music from an analog device and retrieve beats.
 * When a beat is found it moves servos to create a rhythmic movement
 */

#include <Servo.h>

//Audio out with 38.5kHz sampling rate and interrupts
#define NUM_SAMPLE 1024
#define ENPERMIN 43
//threshold for beat detection
#define T 1.5
#define pinLed 12 //pin heart
#define pinBody 8 //pin body
#define pinShoulder 4 //pin shoulder
#define pinElbow 10 //pin elbow
#define microphoneBias 80 //bias of the microphone
#define beatDelay 12 //minimum delay between two beats (beatDelay/4*100[ms])
//define start and end rotation of the servos
#define bodyRotation 0
```

```
#define bodyZero 25
#define shoulderRotation 60
#define shoulderZero 120
#define elbowRotation 75
#define elbowZero 0

//define servos
Servo body;
Servo shoulder;
Servo elbow;

//if true servos are running to zero position (e.g.bodyZero) otherwise servos are
    runnning to the beat movement position (e.g. bodyRotation)
bool up = true;

int cont = 0;
int i = 0;
int j = 0;
long e = 0;
long E [ENPERMIN];
long sum = 0;

void setup(){

 body.attach(pinBody);
 body.write(bodyZero);
 shoulder.attach(pinShoulder);
 shoulder.write(shoulderZero);
 elbow.attach(pinElbow);
 elbow.write(elbowZero);

 cli();//diable interrupts
 //set up continuous sampling of analog pin 0
 //clear ADCSRA and ADCSRB registers
 ADCSRA = 0;
 ADCSRB = 0;
 ADMUX |= (1 << REFS0); //set reference voltage
 ADMUX |= (1 << ADLAR); //left align the ADC value- so we can read highest 8 bits
     from ADCH register only
 ADMUX |= (1 << MUX0); // set pin A1
 ADCSRA |= (1 << ADPS2) | (1 << ADPS0); //set ADC clock with 32 prescaler-
     16mHz/32=500kHz
 ADCSRA |= (1 << ADATE); //enabble auto trigger
 ADCSRA |= (1 << ADIE); //enable interrupts when measurement complete
 ADCSRA |= (1 << ADEN); //enable ADC
 ADCSRA |= (1 << ADSC); //start ADC measurements
 sei();//enable interrupts

}

ISR(ADC_vect) {//when new ADC value ready
  if (i < NUM_SAMPLE) {
```

```
    e = e + (ADCH-microphoneBias) * (ADCH-microphoneBias);//update the variable
        incomingAudio with new value from A0 (between 0 and 255)
    i ++;
  }
  else { //new 1024 sample
    sum = sum - E[j] + e;
    E [j] = e;
    j++;
    if (j == ENPERMIN) {
      j = 0;
    }
    if (e > ((sum/ENPERMIN)*T)) { //beat found
      if(cont > beatDelay){
        cont = 0;
        beat();
        analogWrite(pinLed, 200);
      }

    }
    else {
      analogWrite(pinLed, 0);
    }
    i = 0;
    e = 0;
    cont ++;
    if(cont == beatDelay){
      beat();
    }
  }
}

//function that moves the servo
void beat(){

  if( up == true ){
    body.write(bodyRotation);
    shoulder.write(shoulderRotation);
    elbow.write(elbowRotation);
    up = false;
  }
  else{
    body.write(bodyZero);
    shoulder.write(shoulderZero);
    elbow.write(elbowZero);
    up = true;
  }
}

void loop(){

}
```

3.2.2 Light board code

In the other Arduino we use a different sketch, MSGQE7_final.ino. The code receives the input from the function MSGEQ7.read, which gives the value of all 7 spectrum bands. The value of each band is used to create a Vu Meter on the NeoPixel matrix and use leds depending on the amplitude of the signal of some frequencies. We use lower frequency amplitude to trigger laser, solenoid and heart leds, medium frequencies amplitude to drive eyes leds.

All the threshold for move the solenoid and to write to the matrix and leds can be easily modified by changing the values of some constant on the top of the sketch.

```
/*
 *  This code takes in input an audio signal and use the MSGEQ7 chip to write a
     VU meter on a RGB Matrix.
 *  Before doing that it shows on the RGB Matrix a String that is contained into
     the variable matrixText.
 *  It also activates with respect to signal amplitude and frequency :
 *  - a solenoid
 *  - lasers
 *  - RGB strips
 *  - led of the heart
 *  - led of the eyes
 */

//include the libraries
#include <Adafruit_GFX.h>
#include <Adafruit_NeoMatrix.h>
#include <Adafruit_NeoPixel.h>
#include <FastLED.h>
#include <MSGEQ7.h>


//Debug mode 1 => activated 0 => not activated
#define DEBUG 0

#define NUM_LEDS 64 //number of leds of the matrix
#define MATRIX_PIN 10 //pin Neopixel
#define BASS_THRESHOLD 100 //bass threshold
#define TREBLE_THRESHOLD 90 //treble threshold
#define SOLENOID_THRESHOLD 80 // solenoid threshold
#define BRIGHTNESS 190 // brightness of the neopixel matrix

// MSGEQ7 settings
#define MSGEQ7_INTERVAL ReadsPerSecond(60) //number of read per second of the
    MSGEQ7
#define MSGEQ7_SMOOTH 100 // Range: 0-255
#define FLOOR 30 // increase this if you have noise when no signal input
#define MAX_IN 150 // change this depending on the signal level input.

// MSGEQ7 pinS
#define PIN_ANALOG A0
#define PIN_RESET 4
```

```
#define PIN_STROBE 2

// Heart, Eyes, Solenoid, Laser pin, RGBexit, ledRGBstrip, TextMessage
#define PIN_HEART 11 //pin heart
#define PIN_EYES 9 //pin eyes
#define PIN_SOLENOID 8 //pin solenoid pump
#define PIN_LASER1 A5 // pin laser1
#define PIN_LASER2 A4 // pin laser2
#define RGB_EXIT 7 // uscita video out pulse
#define LED_RGB_STRIP_R 3// ledRGBstrip
#define LED_RGB_STRIP_G 5// ledRGBstrip
#define LED_RGB_STRIP_B 6// ledRGBstrip


// The leds
CRGB leds[NUM_LEDS];
//MSGEQ7 settings
CMSGEQ7<MSGEQ7_SMOOTH, PIN_RESET, PIN_STROBE, PIN_ANALOG> MSGEQ7;

//Variables
long bassColor; //bass color
long colorTreble; //treble color
char temp;
char temp2;
String matrixText = "HI! LET'S PARTY!!:-)";
void colorSelectTreble (void);
void colorSelectBass(void);


// Neopixel setup
Adafruit_NeoMatrix matrix = Adafruit_NeoMatrix(8, 8, MATRIX_PIN,
  NEO_MATRIX_BOTTOM + NEO_MATRIX_LEFT +
  NEO_MATRIX_COLUMNS + NEO_MATRIX_PROGRESSIVE,
  NEO_GRB           + NEO_KHZ800);

void setup() {
  delay(2000);
  text(matrixText); //write initial words
  delay(2000);
  LEDS.addLeds<WS2812,10,GRB>(leds,NUM_LEDS);
  LEDS.clear();
  blackOut(); // set all leds to CRGB::Black
  LEDS.show();
  MSGEQ7.begin();
  #ifdef DEBUG
    Serial.begin(115200);
  #endif


}
```

```
void loop() {

  // analyze without delay
  bool newReading = MSGEQ7.read(MSGEQ7_INTERVAL);

  // Led strip output
  if (newReading) {

    int led6 = map(MSGEQ7.get(MSGEQ7_6), FLOOR, MAX_IN, 0, 5);
    leds[9+led6] = CRGB::Red;
    for (int i = 0; i < led6; i++) {leds[9+i] = colorTreble;}

    int led5 = map(MSGEQ7.get(MSGEQ7_5), FLOOR, MAX_IN, 0,5);
    leds[17+led5] = CRGB::Red;
    for (int i = 0; i < led5; i++) {leds[17+i] = colorTreble;}

    int led4 = map(MSGEQ7.get(MSGEQ7_4), FLOOR, MAX_IN, 0,5);
    leds[25+led4] = CRGB::Red;
    for (int i = 0; i < led4; i++) { leds[25+i] = colorTreble;}

    int led3 = map(MSGEQ7.get(MSGEQ7_3), FLOOR, MAX_IN, 0, 5);
    leds[33+led3] = CRGB::Red;
    for (int i = 0; i < led3; i++) {leds[33+i] = colorTreble;}

    int led2 = map(MSGEQ7.get(MSGEQ7_2), FLOOR, MAX_IN, 0, 5);
    leds[41+led2] = CRGB::Red;
    for (int i = 0; i < led2; i++) {leds[41+i] = colorTreble;}

    int led1 = map(MSGEQ7.get(MSGEQ7_1), FLOOR, MAX_IN-20, 0, 7);
    leds[49+led1] = CRGB::Red;
    for (int i = 0; i < led1; i++) {leds[49+i] = colorTreble;}

    int led0 = map(MSGEQ7.get(MSGEQ7_0), FLOOR, MAX_IN, 50, 255);//bass
    for (int i = 0; i < 8; i++) {leds[0+i]= bassColor; }
    for (int i = 0; i < 8; i++) {leds[56+i] = bassColor; }
    for (int i = 0; i < 8; i++) {leds[7+i*8] = bassColor; }
    for (int i = 0; i < 8; i++) {leds[0+i*8] = bassColor; }

    LEDS.setBrightness(BRIGHTNESS);
    LEDS.show();
    blackOut();

    //turn up lights over threshold
    if(MSGEQ7.get(MSGEQ7_0)>BASS_THRESHOLD ){
      LEDS.setBrightness(255);
      colorSelectBass();
    }

    //select treble color over threshold
    if(MSGEQ7.get(MSGEQ7_4)>TREBLE_THRESHOLD ){
      colorSelectTreble();
    }
```

```
    //Light of heart
    uint16_t input = map(MSGEQ7.get(MSGEQ7_BASS), FLOOR, MAX_IN, 0,20);
    analogWrite(PIN_HEART, exp (input) );

    //Light os eyes
    uint16_t input2 = map((MSGEQ7.get(MSGEQ7_4)), FLOOR, MAX_IN, 0,20);
    analogWrite(PIN_EYES, exp(input2) );

    //Move the solenoid turn on RGBExit signal
    if(MSGEQ7.get(MSGEQ7_0)>SOLENOID_THRESHOLD){
     analogWrite(PIN_SOLENOID, exp (input) );
      digitalWrite(RGB_EXIT,HIGH);
      }
    else{
      digitalWrite(PIN_SOLENOID,LOW);
      digitalWrite(RGB_EXIT,LOW);
      }

    //Laser1 and Laser2 on
    analogWrite(PIN_LASER1, exp (input) );
    analogWrite(PIN_LASER2, exp (input) );


    // activate ledRGBstrip
    analogWrite(LED_RGB_STRIP_R, exp(map(MSGEQ7.get(MSGEQ7_0), FLOOR, MAX_IN,
        0,20))); //ledRGBstrip_R
    analogWrite(LED_RGB_STRIP_G, exp(map(MSGEQ7.get(MSGEQ7_3), FLOOR, MAX_IN,
        0,20))); //ledRGBstrip_G
    analogWrite(LED_RGB_STRIP_B, exp(map(MSGEQ7.get(MSGEQ7_6), FLOOR, MAX_IN,
        0,20))); //ledRGBstrip_B

    //Debug
    #ifdef DEBUG
      Serial.println(MSGEQ7.get(MSGEQ7_0));
      Serial.println(exp (input));
      Serial.println(MSGEQ7.get(MSGEQ7_1));
      Serial.println(MSGEQ7.get(MSGEQ7_2));
      Serial.println(MSGEQ7.get(MSGEQ7_3));
      Serial.println(MSGEQ7.get(MSGEQ7_4));
      Serial.println(MSGEQ7.get(MSGEQ7_5));
      Serial.println(MSGEQ7.get(MSGEQ7_6));
    #endif

 }
}


////////////////////////////////////////////////////////
//Functions
////////////////////////////////////////////////////////
```

```
//set all the leds to CRGB::Black
void blackOut(void) {
  fill_solid(leds, NUM_LEDS, CRGB::Black);
}

//select color for the bass
void colorSelectBass(void) {
  temp++;
  if(temp==9){temp=0;};
  switch(temp){
          case 0:
            bassColor = CRGB::Amethyst;
            break;
          case 1:
            bassColor = CRGB::Chartreuse;
            break;
          case 2:
            bassColor = CRGB::DodgerBlue;
            break;
          case 3:
            bassColor = CRGB::Fuchsia;
            break;
          case 4:
            bassColor = CRGB::Gold;
            break;
          case 5:
            bassColor = CRGB::Aqua;
            break;
          case 6:
            bassColor = CRGB::LawnGreen;
            break;
          case 7:
            bassColor = CRGB::MediumBlue;
            break;
          case 8:
            bassColor = CRGB::Orange;
            break;
          case 9:
            bassColor = CRGB::Yellow;
            break;
       }
}

//select color for the treble
void colorSelectTreble (void) {
  temp2++;
  if(temp2==9){temp2=0;};
  switch(temp2){
          case 0:
            colorTreble = CRGB::DodgerBlue;
            break;
```

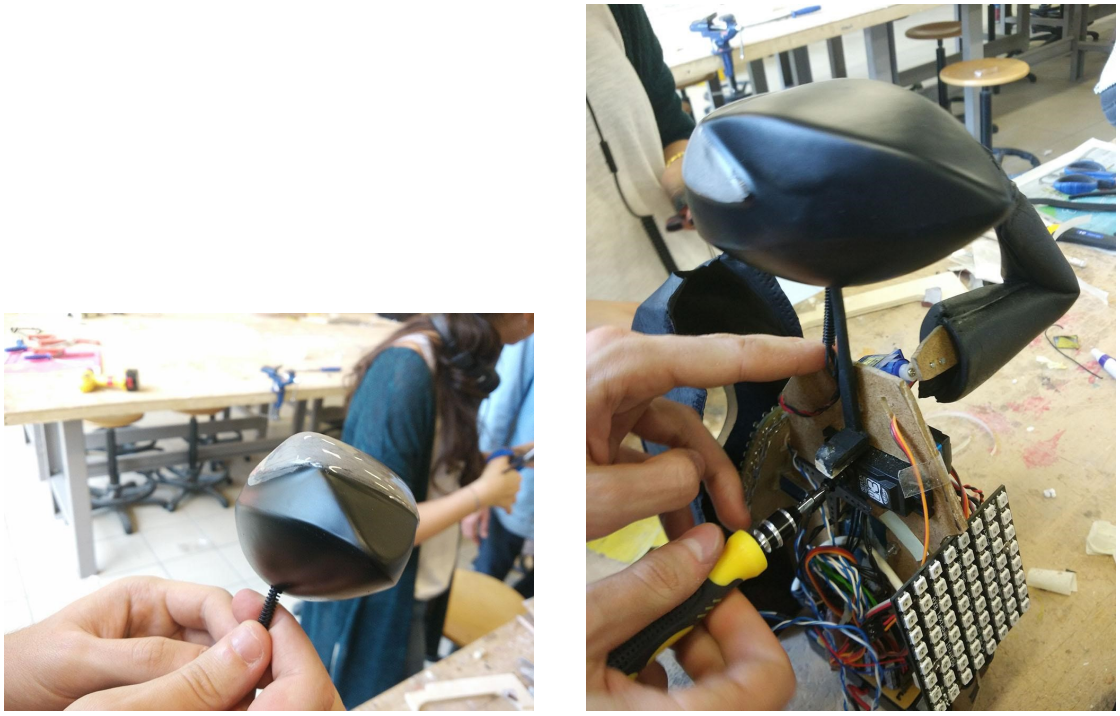
```
          case 1:
            colorTreble = CRGB::Aqua;
            break;
          case 2:
            colorTreble = CRGB::Yellow;
            break;
          case 3:
            colorTreble = CRGB::LawnGreen;
            break;
          case 4:
            colorTreble = CRGB::Orange;
            break;
          case 5:
            colorTreble = CRGB::Chartreuse;
            break;
          case 6:
            colorTreble = CRGB::Chartreuse;
            break;
          case 7:
            colorTreble = CRGB::Gold;
            break;
          case 8:
            colorTreble = CRGB::Fuchsia;
            break;
          case 9:
            colorTreble = CRGB::MediumBlue;
            break;
      }
}

//shows in the matrix the String message
void text(String matrixText){
  matrix.begin();
  matrix.setTextWrap(false);
  matrix.setBrightness(255);
  matrix.Color(255, 0, 0);
  int x = matrix.width();
  int pass = 0;
  int tmp=millis();
  while(millis()-tmp<10400){
    matrix.fillScreen(0);
    matrix.setCursor(x, 0);
    matrix.print(matrixText);
    if(--x < -117) {
     x = matrix.width();
    }
    matrix.show();
    delay(80);
  }
}
```

3.3 Structure Technical Specifications

3.3.1 Head

The head is composed by an internal part and an external shell. The internal part is composed by a small LED board and a frame for supporting. Both parts were fixed inside the head shell; only the connecting electric wire was left outside. The shell is formed by two half parts, and then they were sealed by glue as a closed container.

The shell was produced by thermal transform in PET plastic, and in the lower part has a hole to make the connecting-bar-like structure (a long stick) that can easily access the inner part of the head, and connect them to the servo on the main structure frame, to make the head move naturally we also add a spring to simulate the real neck structure.



(a) First figure (b) Second figure

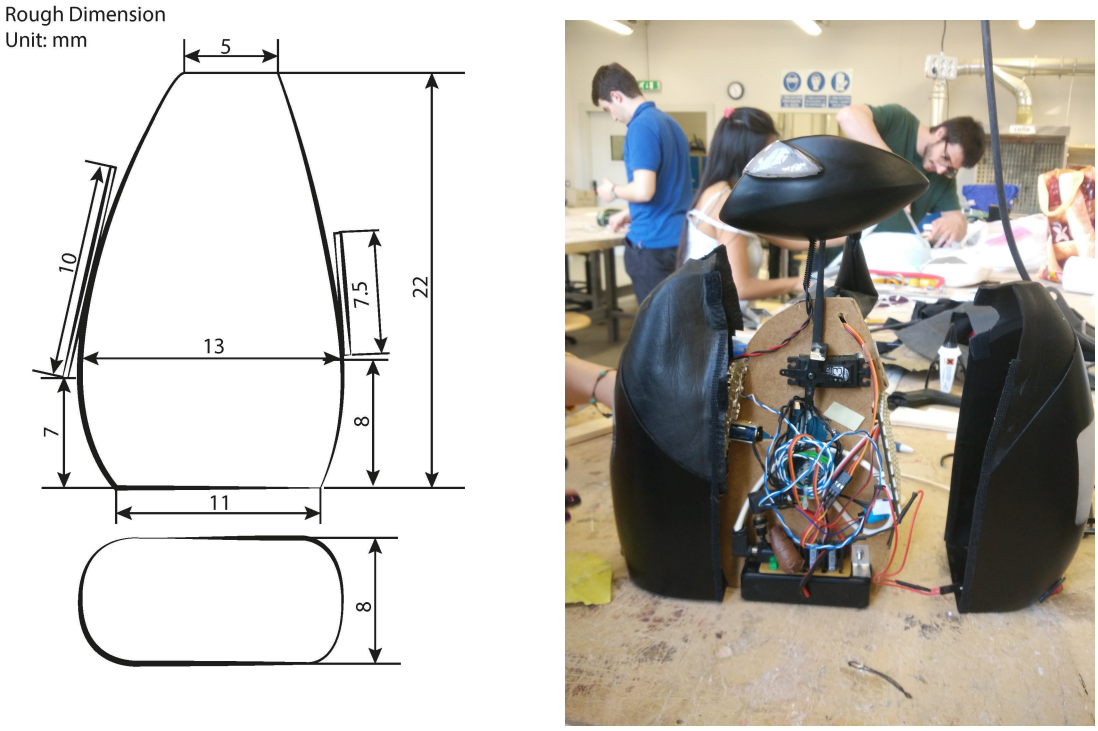
Figure 3.18: Robot head

3.3.2 Body

The body is holding the main structure of the robot. It includes an internal structure, and an external shell (cover).

The internal part is composed by a frame which was cut manually by MDF board, and its shape was defined by the lateral view of robot. The electronic components are interlocked on the slots of the panel frame or on the side part of the frame. For reinforcing the stability some delicate key points were fixed with screws.

The supply box is in the bottom of the frame to make it more stable and the LED matrix was



(a) First figure

(b) Second figure

Figure 3.19: Robot body

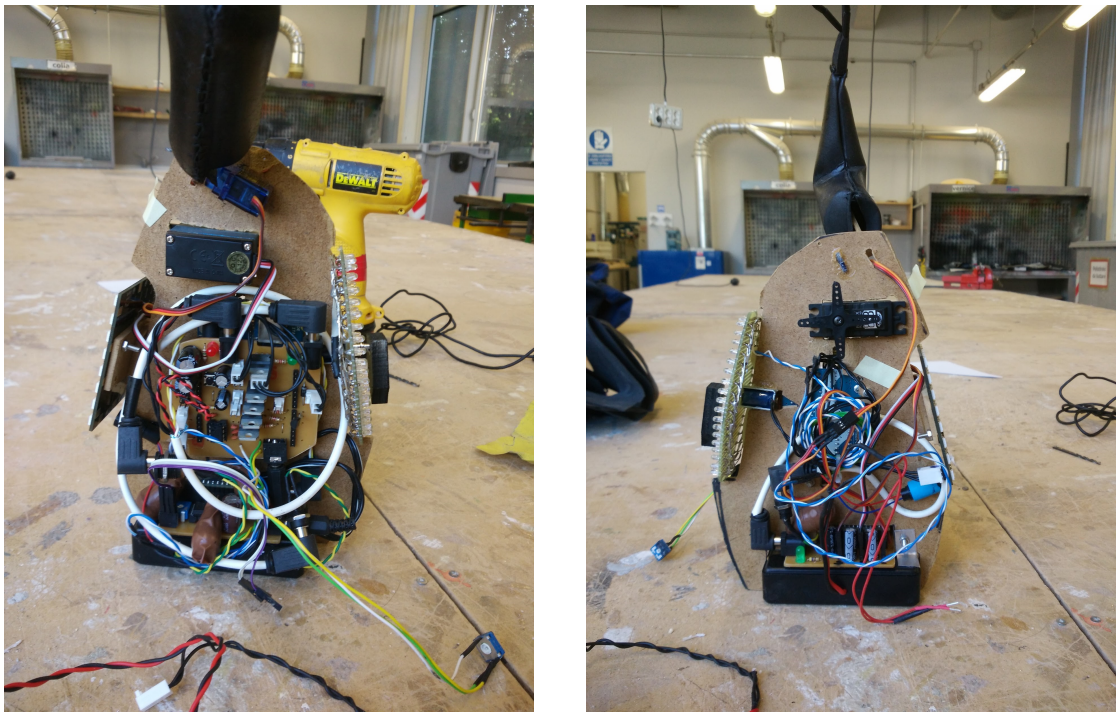
insert in the back slot to guarantee its right position and to create the light effect. Below it, two lasers light were put to create special effect.

The external cover is combined by two half pieces.The two pieces can be well closed in order to form a closed space (like a pure vessel). Each piece is made up by two material, the upper part is EVA plastic and the lower part is PET. They were glued together to have a complete form. Both parts are produced by the thermal transform process. The EVA plastic part is soft, which lets the arms and the head move easily without blocking by the hard body and, at the same time, we attached velcro on the upper part to guarantee an easily opening and closing when necessary.

The lower hard parts control the main form of the robot. The front cover is with an oval opening and an oval part is for the “heartbeat” where the front light come out from the inner structure. A small hole in the down part is for holding the microphone, which can sense the signal from outside. In the lower part a potentiometer was fixed, which allow people to easily adjust the lights intensity of the robot.

The “heartbeat” effect is created by a solenoid which is fixed in the center of the LED board, we used a small block as weight to make sure that when it bounces it will be more stable and without rotation.

The back cover’s mainly function is to show the LED matrix’s effects. A square shape part was remained opaque for creating the light effect, and close to the edge of the cover there are two slots to insert the switch and cable jack.



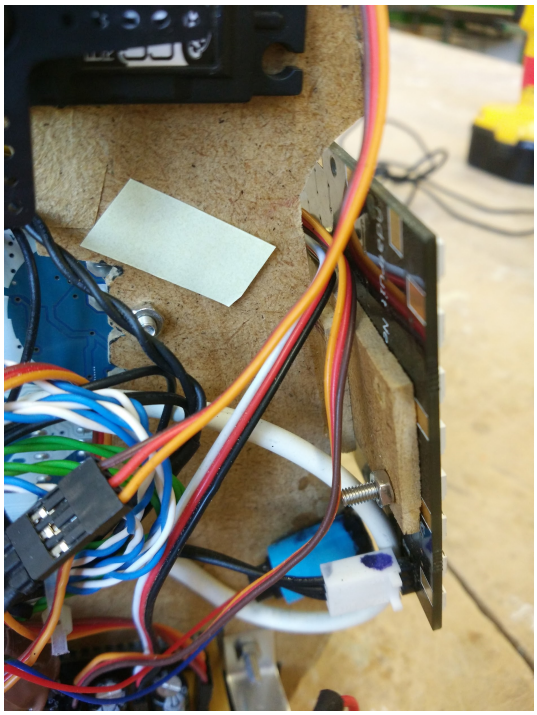
(a) First figure

(b) Second figure

Figure 3.20: Robot body opened

3.3.3 Arms and hands

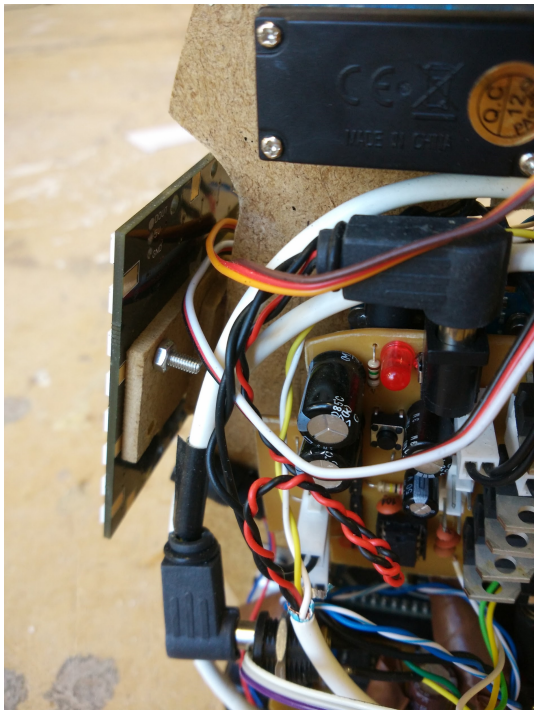
There are two arms for our robot, one is fixed static on one side, just as aesthetic function, the other one is connected with two sticks and two servos which simulate the DJ movements on the stage. Both arms are made of the EVA plastic with thermal transform process, and for the moveable one, the soft material make it bends in different ways and directions, and the light weights also make the servos easier to make the movements. The two hands were made by polystyrene and iron manually.



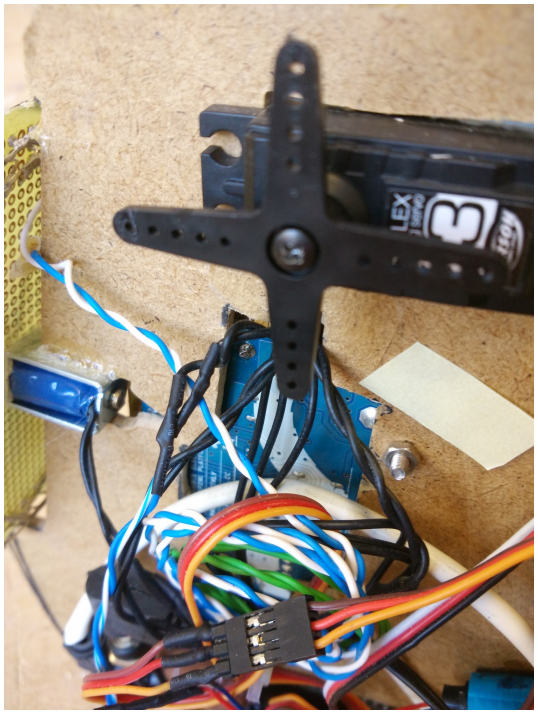
(a) First figure

(b) Second figure

Figure 3.21: Robot body details

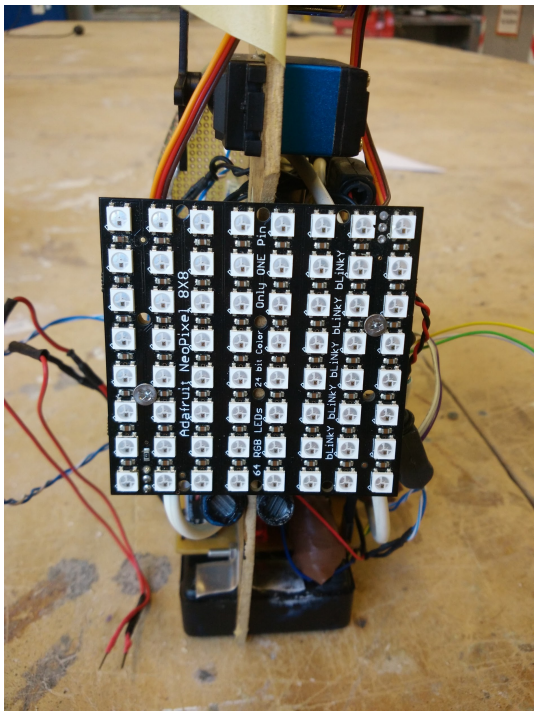


(a) First figure

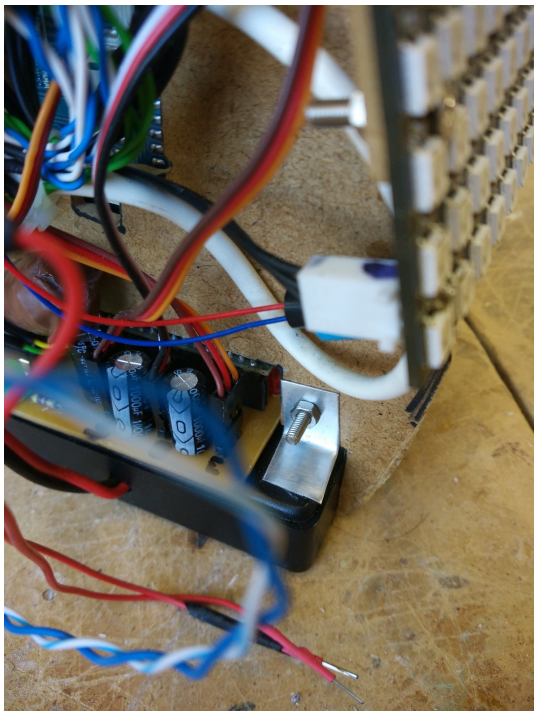


(b) Second figure

Figure 3.22: Robot body details



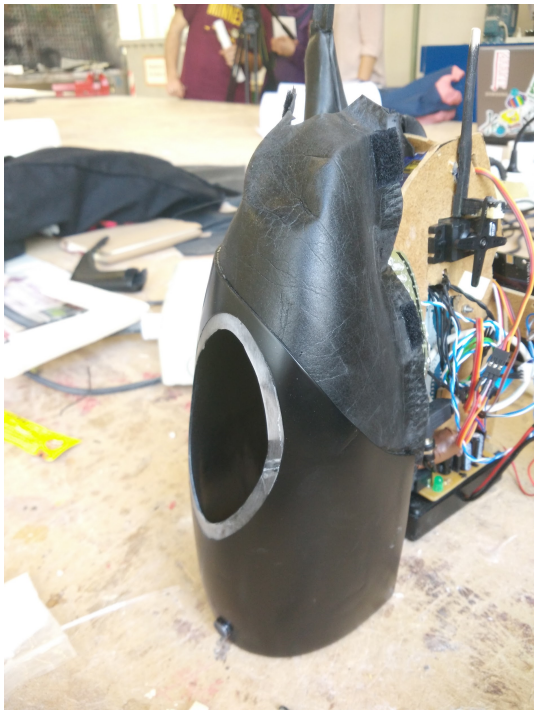
(a) First figure



(b) Second figure

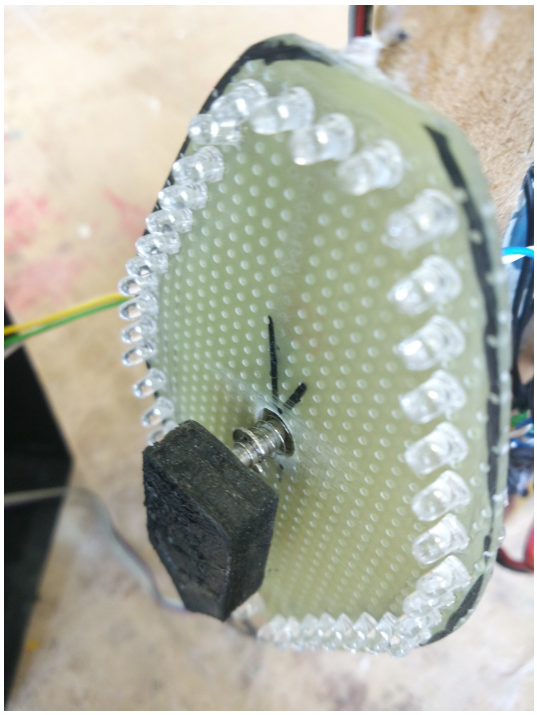
Figure 3.23: Robot body opened

(a) First figure



(b) Second figure

Figure 3.24: Robot cover parts



(a) First figure

(b) Second figure

Figure 3.25: Robot back and anterior part

(a) First figure

(b) Second figure

Figure 3.26: Robot back arms

Chapter 4

Accessories (Laser game generator)

4.1 Building report

We decide also to build an accessory, in particular a laser game generator. This type of devices are largely used during the Dj shows. They create the right atmosphere.

The box is built using wood, for its simplicity to cut and glue. The black color is chosen to make contrast with the red light source coming from the internal part.

4.2 Project specification

In this section we illustrate the electronics and coding specification about the laser game generator.

4.2.1 Electronics Technical Specification

The device is based on a ATtiny85, a microcontroller similar and with the same architecture of Atmega328 (Fig 4.1).

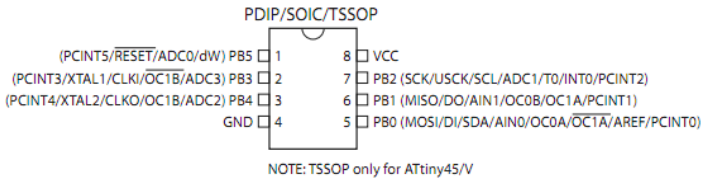
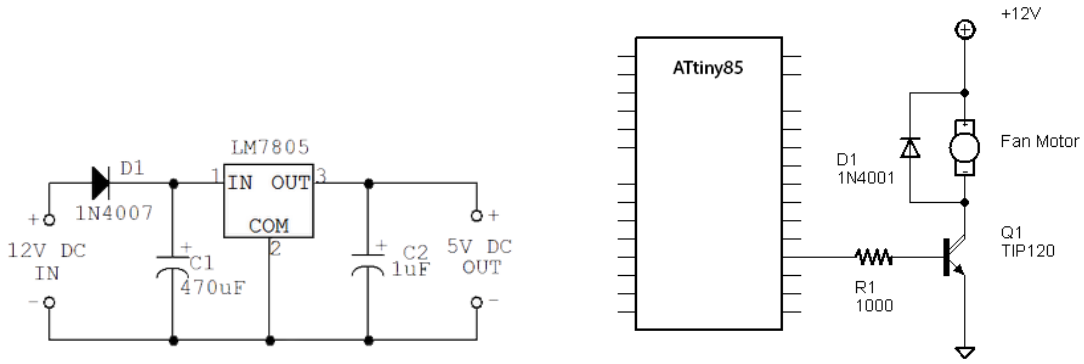


Figure 4.1: ATtiny pinout.

The power supply is provided by an external power supply, which generate the 12V DC necessary to activate the fans. The ATtiny is powered by 5V regulated using a Lm7805 between the 12V and the ATtiny (Fig 4.2).

We connect 2 TIP120 to port 0 and port 1 as in the schematic 4.2, to provide the 12V the right time and we put a diode to prevent reverse voltage cause by inductive phenomena (Fig 4.2).

The 2 laser diode are powered using the 5V provided by LM7805. One is located behind a fan to create a stroboscopic effect and one is located near a fan to be always visible.



(a) First figure (b) Second figure

Figure 4.2: Left: 12V to 5V using LM7805. Right: Fan connection.

The device is based on the concept of persistence of vision[1], based on the maximum eyes refresh rate, which is lower than the laser movements created by reflecting on a mirror. Infact the device incorporate two rotating mirrors fixed on the two fans. At the starting the laser point a space of the mirror, but then rotating the mirror the laser point moves. Adding a smoke generetar the effect is greater[2].

Figure 4.3: Laser vortex example. Created using green laser and rotating mirror.

[1] https://en.wikipedia.org/wiki/Persistence$_of_vision, http://www.vision.org/visionmedia/philosophy-and-ideas/persistence-of-vision/136.aspx$

[2] https://www.youtube.com/watch?v=0uZfa9UdFyU

4.2.2 Coding Technical Specification

We programmed the ATtiny using an arduino uno and the same IDE of arduino. The code activate alternatively two port to switch on the 2 fan in 2 different time.

```
#define pinfan1 0 //define of pin of fan 1
#define pinfan2 1 //define of pin of fan 2

int value=1;

void setup() {

 pinMode(pinfan1,OUTPUT); // set the pin as output
 pinMode(pinfan2,OUTPUT); // set the pin as output


}

void loop() {

  digitalWrite(pinfan1,LOW);
  digitalWrite(pinfan2,HIGH);
  delay(250);
  digitalWrite(pinfan1,HIGH);
  delay(250);
  digitalWrite(pinfan1,LOW);
  digitalWrite(pinfan2,LOW);
  delay(200);
  digitalWrite(pinfan1,HIGH);
  delay(200);
}

 // the pins are set alternatively high or low to create different light game.
```

Chapter 5

Cost Table

In this chapter we want to put in evidence the cost afforded for the realization of the robot.

5.1 Light Board Cost Table

Component	Cost	Seller
pcb	1€	RS component
Arduino uno	10€	Amazon
Woofer driver solenoid	9€	Amazon
BS170	0.20€	RS component
Woofer lights	4€	Ebay
TIP120	0.40€	RS component
Eyes lights	2€	Ebay
TIP120	0.40€	RS component
Crimped wire	2.8€	RS component
Pin header	0.7€	RS component
Led matrix	40€	Robot Italy
Laser	2€	Amazon
MSGEQ7	5.88€	Robot Italy
trimmer	0.5€	RS component
audio jack	1€	RS component
DC jack male	1€	RS component
DC jack female	1€	RS component
Total	**82.18€**	

5.2 Audio Board Cost Table

Component	Cost	Seller
pcb	1€	RS component
Arduino mini	2.8€	Amazon.it
motor elbow	3.3€	Robot Italy
motor shoulder	3.3€	Robot Italy
motor body CARSON	14€	Amazon.it
trimmer	0.5€	RS component
audio jack	1€	RS component
DC jack male	1€	RS component
DC jack female	1€	RS component
MIC	8€	Robot Italy
Tl071	0.4€	RS component
Total	**36.3€**	

5.3 Laser game generator

Component	Cost	Seller
ATtiny	1.46€	RS component
TIP120	0.40€	RS component
TIP120	0.40€	RS component
fan	4€	RS component
fan	4€	RS component
laser	1€	Amazon
laser	1€	Amazon
Total	**12.26€**	

5.4 Structure Cost Table

Component	N.	Cost
glue	1	8,2
PET sheet	1(90x120)	9,00
MDF panel	1(30x50)	1,5
Spring	1	0,5
Small Component	1	1,6
Small ball	2	0,8
Total	**13.6€**	

If you can dream it, you can do it. (Walt Dysney)