Dancing Robot

EECS 129B – Professor Kelfstad

Have you ever wished that your toys interacted with you? This robot will react to music and sounds generated by you while playing an instrument. It will be able to twist, rotate its head, and move its arms to the beat of your sound. A new, fun and interactive way to learning an instrument.

Group Members (Group P)
Roderick Buenviaje
Ken Urata

Table of Contents

Description

The main purpose of this project is to create a learning tool for young children. Wanting to create a useful and fun way to learn how to use an instrument was the main goal. As young children are easily discouraged when trying to learn something new, this project was meant to curve that discouragement and make learning more fun. Make it so that the child would want to learn how to use an instrument.

This project is to create a robot that reacts to sound, but primarily musical sounds. The robot, through the use of a computer, receives sounds and translates the sounds into movements. The head, torso, and each arm are able to move together or individually. The head and arms are able to rotate a full 360º while the body has a more limited rotation. Due to the head being completely separate from the body's motion, a mount is needed and that mount restricts the amount of movement of the torso. The robot will have a preset sequence of sounds that it will perceive as correct and will move accordingly, however if the "wrong" sound is encountered, the computer will display that a wrong note has been hit. Ideally, the preset sequence would be a musical piece that a person is learning, however it could be anything.

After everything is set up, the computer, with the aid of an internal or external microphone accepts sounds. As the player progresses through a preset song, the robot will move (dance) through the song making it more of a game to get further in the dance. Though a dance can be vague, through the use of the computer, it can be shown that the robot is functioning as it is supposed to, and there will be an error read out stating the player's performance as well.

The robot uses two DC motors and two stepper motors to control the various parts of the body. The arms are controlled using stepper motors since we wanted very precise control over these parts while the head and body utilize DC motors. The body is also very precise in its movements due to a gear box. The head is a bit more unmanageable, but through the use of short

voltage pulses, we can control the head to a certain extent. The robot is connected to the computer through a USB cable which allows us to power the robot and also send it data. The computer will process the sounds and send the corresponding signal to the Arduino board which activates certain motors causing movement.

This project is primarily aimed at younger children, however, it may also be used by adults as well. Adults generally have more patience with learning new things as compared to children so it would just be more of a game, rather than a source of motivation. If the learning experience is more difficult than enjoyable, the will to learn is also diminished. The hope is that this project will be able to make children want to learn how to play an instrument, and give children a greater appreciation for music.

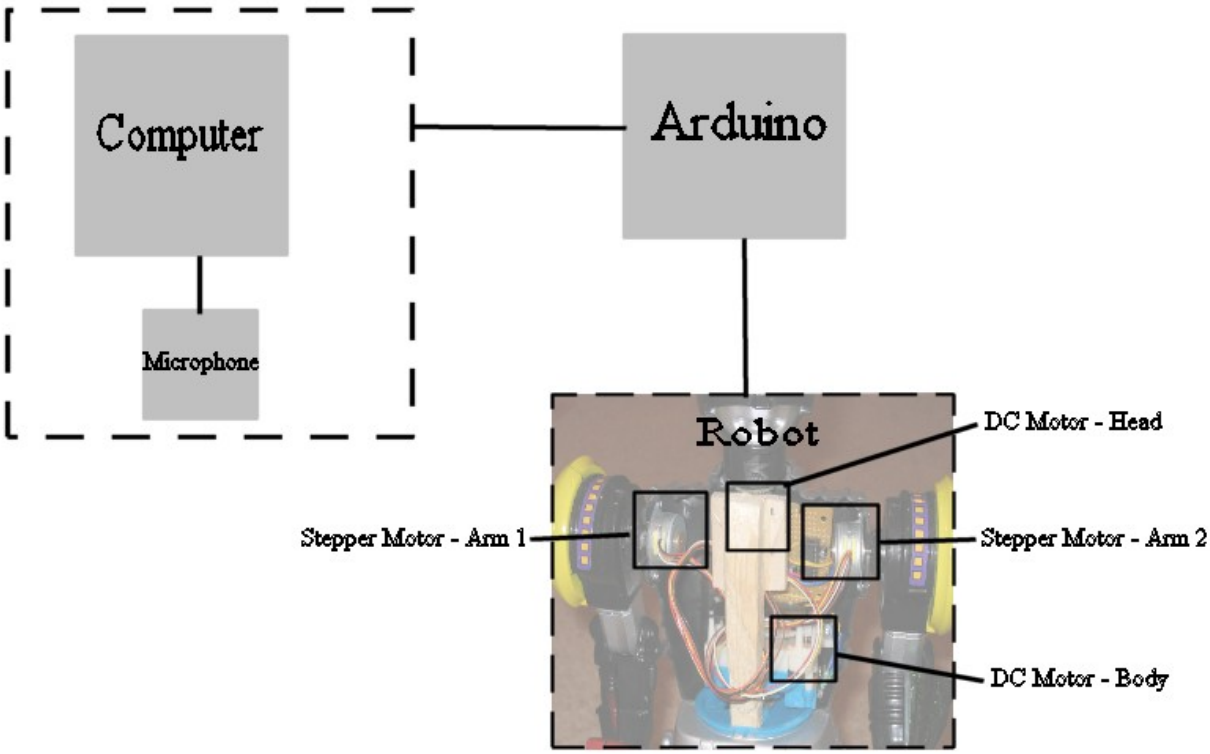
System Level Block Diagram

The project is basically separated into three components: the computer (signal processor), the Arduino, and the robot itself.

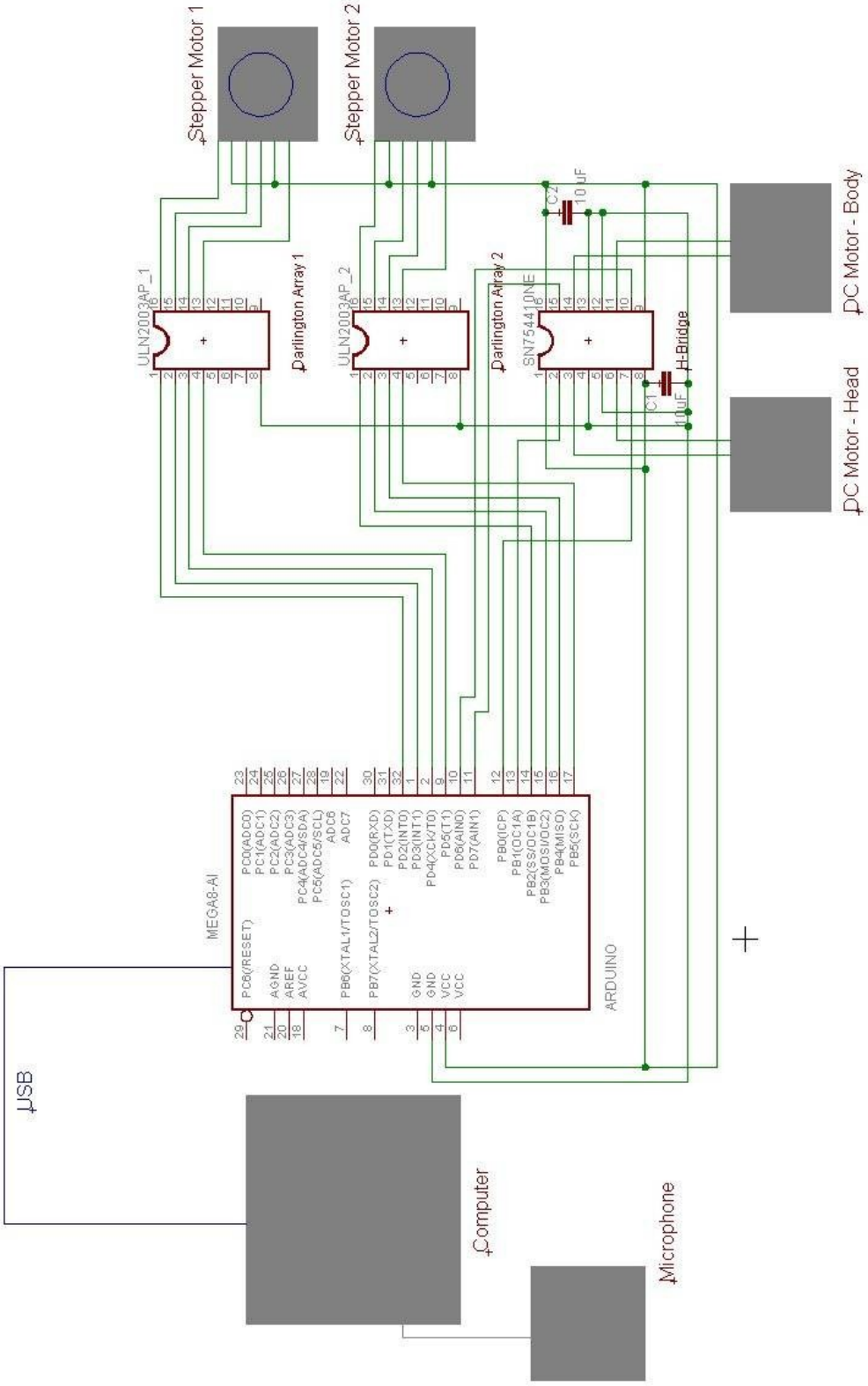
The computer handles all the signal processing and determines what needs to be sent to the Arduino. The computer also acts as our power source for this project, greatly simplifying matters by leaving out an external source. Included in the computer portion is the microphone, which allows the computer to receive outside sounds.

The Arduino is the component that connects the robot to the computer. It allows us to send voltages to certain pins, which in turn activates specific motors.

The robot itself is just the motors that allow the robot to move. Without the input voltage and input signals from the Arduino, the robot is useless. It would be unable to move without both components.



Circuit Level Diagram



IC Details

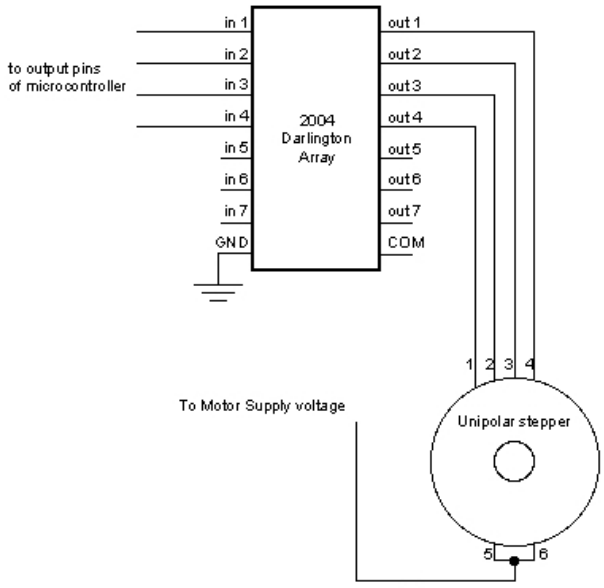


Figure 1: Darlington Array Connected to Stepper Motor[1]

In this image, it shows exactly how the stepper motor is connected to the Darlington Array. The in 1 corresponds to the out 1 and so on. Pins 5 and 6 in this image are the power pins for the motor and are the second and fifth pins in reality.

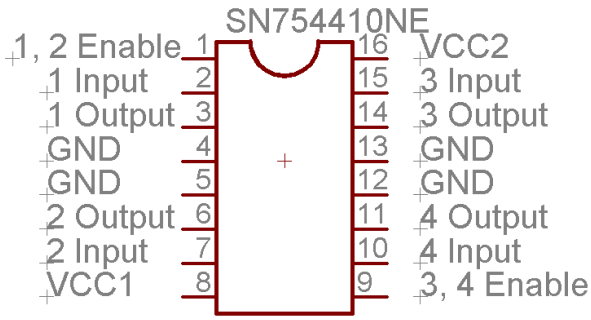


Figure 2: H-Bridge

This image is of the H-Bridge showing what each pin does. If 1, 2 Enable is high, then the left side of the IC will operate, if not then even if one of the inputs is high, the motor will not operate. The same is valid for the 3,4 Enable corresponding to the right side. The VCC's are the motor supply voltages and the motors connect to the outputs. The inputs connect to the microcontroller.

[1] Original image: http://www.tigoe.net/pcomp/code/archives/picbasic_pro/000245.shtml

Software Description

Program loaded onto Arduino:

Input: Takes in serial input from the computer, a character ranging from the ASCII values '*' to 'Z' (42 to 122) to specify how the robot shall move, and the characters '{' and '}' (123 and 125) to decrease or increase the time between each movement.

Output: N/A

The program controls two stepper motors and two DC motors built into the robot. When the serial input is received, the robot immediately performs the movement corresponding to the assigned ASCII value. Since there are a total of 4 possible simultaneous movement (arm x2, head, torso), and 3 possible actions (not moving, moving forwards, moving backwards), there are a total of 81 possible inputs (42-122). The program is designed so the stepper motors move at a steady pace within the time given for a beat, while the burst of voltage sent to DC motors is always the same magnitude.

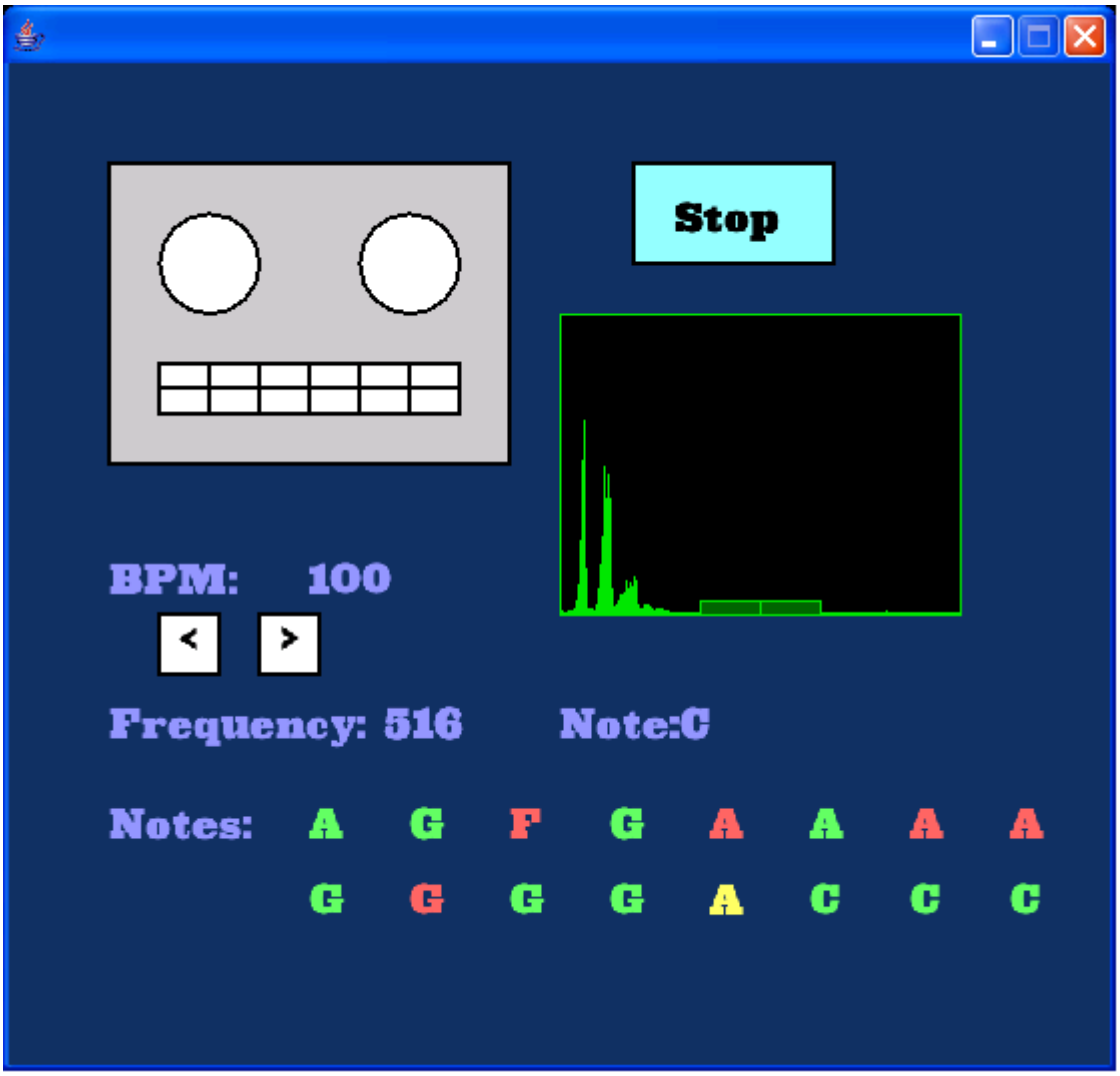
Program on computer:

Input: Sound signal through the microphone

Output: Depending on the frequency that is read from the microphone, the program sends a character from the values 42 to 122 to specify how the robot should move.

The program provides a user interface to activate the robot, specify the set of pitches to be played, and to control the tempo. Also, a Fast Fourier Transform spectrum is formed and outputted, based on the inputs from the microphone. When the robot is activated by a start button, the robot shown on the screen will flash its eyes, signaling when one should be hitting the correct notes. When the correct note is not hit within the given time, the program will not send a signal to the robot to make its move.

Screenshot of program on computer:



System Test Plan

Testing the stepper/DC motors, both individually and simultaneously

Use the Serial input feature included in the Arduino programming environment. Send inputs to the Arduino to see if the robot does the correct corresponding movement based on the character(s) it receives.

Testing the accuracy of sound signal processing

Let the microphone receive a note in which we already know the frequency to. An example would be the A440 (440 Hz) used for the standard tuning for musical instruments, so it should be compared to the value outputted on the computer screen. The Fast Fourier Transform spectrum that is outputted can be also used as a visual reference.

Testing the functionality of the program on computer (w/o sending signals to Arduino)

When pressing the start button, see if the robot eyes flash and serve as a metronome (a tool used to keep a steady beat) at the specified BPM (beats per minute). Then, increase/decrease the BPM using the '<' and '>' buttons to see if the flashing eyes change in speed accurately. Also, observe if the program reads in the set of notes to be played from a file is read correctly.

Testing the computer program and program loaded on Arduino in conjunction

See if the robot remains still in an absence of a clear and loud pitch while the programs are running. Then, run the programs again, but this time inputting the correct set of notes and in accordance with the beat to see if the robot is moving properly.

Cost Analysis

Quantity	Manufacturer	Item Description	Part Number	Total Price (not including shipping and tax)
1	Arduino	Arduino USB Board		$31.95
1	Kool Toyz	Light ‘N Sound Cosmic Robot		$15.00
2	Jameco Valuepro	Stepper Motors (5V/.5A 7.5deg)	25BY4801	$11.90
2		DC Motors (1.5-3V)		$5.00
2	Toshiba	Darlington Driver – 7Ch	ULN20003AP	$1.50
1	Texas Instruments	H-Bridge	SN754410NE	$3.25
2		Capacitors (10μF, 25V)		$0.50
1		A to B USB cable		$2.95
3 Colors		Wire		$6.59
1		Soldering Board		$0.75
1 bag		Plastic Gears		$4.95
		Wood		$2.00

Total cost of all parts: $86.34

Summary

The overall project was much more difficult than previously imagined when we first began. There are two major parts to this project: creating the hardware design and creating the program to interact with the hardware. That is for the hardware, the robot being able to move its head, body, and both arms and for the software, being able to send signals to all the parts and take in sound as an input in determining what to do.

As for the hardware, we were able to accomplish all our goals and even more so than when we first began. The original plan was to get only two simultaneous movements at the same time, but through the use of different motors and careful use of the output pins on the Arduino, we were able to produce four simultaneous movements in the robot. Though we were able to accomplish our goals, it took quite a lot of time to determine what was needed for the movements we wanted. When we first began, we believed that we would be able to use DC motors to accomplish what we wanted, however that soon fell through. A lot of the development was in determining what needed to be used to have the motors do what we wanted. Trying to create h-bridges and other arrays using transistors and diodes became very problematic and caused a lot of delay and additional cost. The small size of all the components also caused trouble as we were not able to buy larger motors that would produce the movements we wanted. Since everything needed to be contained within the robots body, there were size limitations to the parts we used. We also had to carefully plan how to use the space provided or certain things would not have fit well.

Programming was overall very difficult and time consuming to do. However, we were able to meet the basic requirements that we set for the project. Any additions would resemble icing on a cake. Creating the user interface proved to be one of the most time-consuming processes. Also, minor programming errors lead to many unnecessary halts in progress.

Figuring out the programming in Processing and the sound library Sonia was not as trouble-free as expected. To a degree, Sonia made the sound signal processing easy. However, since we are not experts in digital signal processing, using the library still proved to be somewhat difficult. Another difficulty in programming was timing everything to work. There were many factors to consider in timing (BPM, delay, user error margin, etc).