Kevin Frederick
Damon McCoy
Jacob Melvin
Dan Perez

Dancing Automaton

1.0 Abstract

This dancing automaton is meant to entertain and draw laughs from all who watch him in action. With his cowboy hat, overgrown belt buckle, Clock necklace, and Glittery silver glove he would stick out on any dance floor. His body, arms, head, and foot move to the beat of the music most of the time. To produce his somewhat jerky movement we used a simple set of belts and pulleys driven by motors. His great sense of rhythm comes from sound sensors and crickets that control the motors. The Crickets activate the motors with varying degrees of power based on the noise level reported from the sound sensor. The sound sensors add randomness to the dancing as they have a fixed sampling rate that sometimes misses beats and sometimes picks up background sounds guaranteeing that each dance will be unique.

2.0 Design

As we began working on the project it soon became evident that our initial design was quite flawed. We eventually threw out our entire cam design and moved to an easier and more efficient pulley design. We also decided to make our project true to the fashion of the Cabaret Mechanical Theatre by turning around the robot and allowing users to see exactly how everything worked.

Most of the robot and base were made of basswood that was laser cut from designs created with xfig. The Lego parts and motors are all hidden inside the base.

2.1 Foot

The feet where made from basswood original laser cut for use as cams. To provide the motion of the foot we had to scrap our initial cam design. Instead we used a mechanism similar to that used in our "Bone Ranger" project. We found a piece of Lego that had a rod pivoting on a gear. This allowed the rod to move up and down as the gear turned. A problem arose, as the foot did not merely want to go up and down, it wanted to go side to side also. We fixed this problem by carving a grove into the foot for the rod to be fixed into. However this did not fix the problem entirely. We attached metal wire to bottom of the foot and ran the wire into the base. The wire was on the left and right of the foot and kept the foot from slipping side to side. Only the upward movement we wanted occurred.

2.2 Legs

The legs are made of laser cut basswood. The legs were merely designed for a visual effect. However, the legs were redesigned more than any other part of our project. Whenever we altered the design of any piece of our project, we had to design new legs. For instance when we had to widen the chest we had to widen the legs and when we had to make notches for the foot, we had to design a new leg. All in all we had five leg designs. The use of the laser cutter made the construction of new legs much easier.

2.3 Arms

Once again, we scrapped our initial cam design to provide the motion for the arms. Instead we used a pulley system. We used two Lego motors that were attached two wooden pulleys we created with belts to provide the motion.

The creation of the actual arms is true to our initial design. We created two chunks of wood. Carving, with a handsaw and drill press, a grove into one and a piece to fit into the grove in the other. Holes were drilled through both parts and a simple piece of fishing line to hold the pieces together. We hoped this would provide natural movement as the pulleys moved the upper arm.

However, we did encounter one obstacle when creating the arms. A certain person, we will only refer to as Dan, was not paying attention when he assembled the arms and accidentally created two right arms. Try as we might to use the second right arm, we eventually had to scrap most of the assembled arm and build a left one.

2.4 Chest

Our initial under ambitious design did not take chest movement into account. After we began work on the project it became evident that we could add chest movement using the same mechanism we had already designed for the arm. The chest design, made again from laser cut basswood, went through three incarnations. The first design ended up being too slim. We could not fit all the pulleys and the arms behind the chest as we had hoped. Therefore we made a second chest. This design was much larger and wider. This design was also flawed. When the arms would move they would crash into the chest. Our last design took this into account by having slits along the waste that would allow the arms to swing freely as the chest turned side to side. By this time we had become great at the reuse of discarded parts. We used unused pulleys into washers. Actually we only used one unused pulley as we doubled the pulley that would turn the rod as a washer. In addition we disassembled one pulley and used the larger wheel to screw on the chest. After being painted this wheel became known as the "Flavor Flav" clock. Another problem arose with the spinning of the chest. The chest would spin upside down and then shake back and forth staying upside down. We soon discovered it was because we designed the top heavier then the bottom and the weight would keep the top of the chest down. To fix this problem we attached a gem to its stomach. This weighed the chest down, and provided the appropriate motion. After some clever touching up, it became his belt buckle.

2.5 Head

The mechanism for the head was quite simple; we glued one end of a spring inside a pre cut oval block of wood and the other end inside the support beam for the chest. Initially the spring was too loose and we had to shorten it and place a stabilizing wire inside the spring to keep the head from going limp. We relied solely upon the motion of the body and arms to provide motion for the head. This initially worked quite well. To make the head a bit more visually appealing we added a cowboy hat, two "wacky eyes", and drew a face upon it. Eventually through wear and tear the head became limp and would only sagged downward to its left. We added gems to the right side of its hate to weight it down. In the end this merely kept the head looking in an upward direction with minimal bobbing.

2.6 Cricket/Sound Sensors

This section will describe the code we used to power the motors for various dancing movements. We will describe the code in the hope that future projects will find it useful.

We used 4 motors and three crickets. One motor/cricket pair powered each arm, while one cricket controlled two motors driving the foot and chest separately.

The code mirrors the physical arrangement, so that we have the same code in the two arm crickets but different code in the foot-and-chest cricket.

The foot and chest simply response once volume level returned by the sound sensor is high enough. The response is constant, i.e., the response does not change as a function of the volume -- the volume merely turns the response on and off.

The arm movement is much more interesting. Arm movement is meant to scale as a function of the volume level returns by the crickets. This is challenging since our crickets and motors are limited by the discretization of the drive and power settings allowed: increments of 0.1 second for drive duration, and a "setpower" feature with levels 0 (off) to 8. The sound sensor output also presents difficulties, since it changes when the battery is replaced. There is a potentiometer on the sound sensor that allows some tuning; in effect it usually just adds another variable to prediction problems. However, the final effect is remarkably good -- since all we need to do is develop some "dancing" movements, the irregularity introduced in all of these variables is often as helpful as harmful.

More details of specific implementation details can be found in the Appendix, with comments in the code itself.

The key logo constructs utilized in this code are:

Global variables: declared by the 'global [var]' block at the start of the code. These are then set with 'setvar' -- that is, the variable name ('var' in this case) is appended to the 'set' keyword.

motor commands:

a, sets subsequent command to refer to motor a

thisway rotates current motor in given direction (depends on polarity of wire when you plug the connection wire into the cricket)

thatway the inverse of above

setpower n sets current motor power to n, between 0(off) and 8(full)

rd reverse current motor direction

brake apply brake to current motor

onfor ngive power to motor for n tenths-of-second

on turn motor on

off turn motor off (will still coast)

subroutines: start with 'to' keyword, end with 'end'. Preferred infinite loop method is via tail recursion in subroutine (see code). Return value with 'output' statement.

when [cond] [statements]: multitasking feature. Triggers code block 'statements' when 'cond' is met. Works [well] for smaller 'statements' blocks -- for example, we intended to drive both arms with a single cricket, but our arm code didn't pre-empt well -- it would drive one arm for a very long time before switching back to other task/arm.

bus control: The coding here is somewhat opaque, but here's what works: start with a 'bsend $h' where h is a hex number representing (i think) the bus address. Our sound sensors were labeled 67, 68, and 69, so we used '$h' values $167, $168, and $169. then, for the sounds sensor, your retrieve the output with 'bsr $01' or "bsr $02'. Address (?) '$01' returns a clap yes/no results, while '$02'returns and integer for 0 (off) to 255 (loud) for volume. Our usually read around 50 with just ambient noise -- the 'test' procedure will allow you to see in Microworld what kind of values it's returning. The sound sensors output also depends on the setting of the potentiometer (the little gray dial, clockwise is less sensitive) on the sensor itself, as well as the battery voltage (amperage? --

whichever) in the cricket! Your readings *will* change when you change the battery (in our experience).

if/ifelse: these flow control statement behave as you expect, though the syntax is delicate (see the code).

3.0 Evaluation

In our original layout for the dancing automaton, we wanted the foot to tap. Strings were going to be used to go through the ankle, which would pull from the back end so that the foot could move up and down in a tapping motion. The shoulders would have been on a cam system, this would have caused an up and down motion. Fishing line would have been used as the string for the cam system. We wanted the arms to have loose joints, so it would dangle and make it’s own moving motions. The chest would have moved in a side-to-side motion using a pulley system. We wanted to use the crickets to power the motors inside of the base, one of them would power the chest and foot and the other would power both shoulders. Our plan for it was to look like a wooden Pinocchio.
The final layout changed the original plan to a large degree. Instead of it being a 3D structure, three vertical support beams were used. Flat body parts were mounted on them to give it a 2D/3D feel and look. The side edges were angled so that the arms would not knock into the sides when the chest is turning. The chest stuck out away from the rest of the body. Instead of the cam system, a pulley system was used to get a twisting motion. Instead of the strings, black elastic belts were used to produce enough torque. On the head, a little metal rod was inserted inside of the spring to make it stable so the head would stand up straight. The dancing automaton ended up having a cowboy hat, silver glove, clock for necklace, and a red jewel for a belt buckle and a waist belt. The base was colored with red and silver glitter. We ended up using three crickets, one for the chest and foot, and one for each shoulder. Lego’s were used to make the floor of the base, in order to easily install the motors into their proper positions.

The dancing automaton was extremely entertaining; it brought laughter to everyone who laid eyes on it. It danced along with the music, and was on beat with the songs therefore it brought entrainment for hours. It was funny watching it move in random motions with the different songs. The computation value was tremendously high because it allowed us to provide the dancing robot with a sense of randomness. Every time it danced the movements were never the same. You would never get bored watching it dance.

When the chest was being placed it had to be secured with washers to make sure that it moved only side to side, not back and forth. Since there was a big circle on his chest, we decided to uses it as a decoration, and it remind us of an old school rapper by the name of Flava Flav, who was a member of a legendary rap group called Public Enemy. Flava Flav was known for wearing an actual sized clock as necklace. It gave our dancing guy robot a hip-hop feel. One hand is silver and glittery in representation of Michael Jackson, giving it a pop feel, and we used a cowboy hat to give him a country feel.

4.0 Education

To the skeptic, our dancing automaton is little more than a glorified dancing flower. However, on closer examination we can see several useful educational lessons -- because our automaton incorporates everything from simple Lego push rods to microprocessor-controlled motors with sound sensors, there is a variety of educational levels to examine the automaton at. And, while the motion in our automaton is simple to "figure out," it does embody many interesting mechanical and physical concepts.

We'll focus first on simply viewing the automaton in action on what one might learn from a passive observation of the automaton. From the front, we can see little of how the automaton works: we see a glimpse of a simple push rod moving his tapping foot; the head is obviously driven passively since it is apparently only connected by a spring; and the arms don't appear to have any internal mechanism but instead just seem to rotate on the shoulders. Yet, the movement produced is quite striking -- the effectiveness of the motion produced adds an extra dimension to the enjoyment of investigating the simple mechanism. We'll discuss this aspect below.

Turning to the back of the automaton, the details of its operation are presented: elementary belt drives move the chest and arms, and a loose push rod taps the foot. Examining the foot shows a simple translation of motion: the regular unidirectional rotation of the motor is converted to a bi-directional up-and-down movement by the push rod, which, due to it's loose coupling with the foot, produces a nice, gravity-assisted, tapping effect. The educational value should be apparent: there's an easy case study on motion translation. One could ask a student: where have you seen a similar mechanism? (Trains, engines, etc.) Or, how does this compare to the movement produced by a person pedaling a bicycle?

The belt drive of the arms and chest can be similarly examined for mechanical learning opportunities: here, the tighter coupling of the chest pulley with the motor pulley means that the chest more closely follows the motion of the motor. If the motor starts spinning wildly, the chest will echo this motion and thus look biologically unrealistic. What does this imply about how we need to move the motor to produce realistic chest movement? (We need to make small motor movements that return to the original position.)

Examining the arms, we see that they use the same belt design, but exploit it to different effect: while the upper arms might respond the same way as the chest, the lower arm's hinge design gives it a different motion. Also, because we're now simulating a shoulder joint, a wider range of movement appears plausible: for one, we can perform complete rotations without completely losing touch with reality. We can ask great physiological questions of this design: is the shoulder joint really the way humans move? (No, our automaton uses an axle; humans use a ball-and-socket with limited rotational movement.) How does the elbow joint mimic a real elbow? (Both bend in only one direction!)

So far, we've just focused on the mechanical aspects of the automaton. Can anything be learned from the crickets, motors, and sound sensors? This area turns out to be a bit subtler. One can certainly begin by experimenting with different sounds: is the motion driven by harmonies or rhythms? (Mostly rhythm, since the sound sensors send amplitude/volume.) Does the same sound always produce the same motion (is it deterministic)? Does the movement really look like a "dance"? Does the movement change in proportion to volume, or does it seem to change less for increases at higher volumes? What effect does adjusting the dials (potentiometers) on the sound sensors have? How can one cricket control two motors and produce distinct movements on each?

On a higher level yet, we can examine the code used for the crickets: it's probably most instructive here to try and program the motion on one's own, and then compare it to the code we used. For example, a problem we faced was that the cricket tended to want to turn the chest more in one direction that the other, even though, semantically, the code appeared to perfectly return the chest every time. Why is this? (We suspect it's because of the relative effectiveness of attempting to brake the rotation versus switching directions.) This disparity suggests another line of investigation: how does the deterministic nature of the computer program interact with the "less" deterministic nature of the physical world? In our case, the semantics of the cricket code didn't really reflect the capabilities of the motors.

The list of potential lessons for the seemingly simple movement in this device is undoubtedly endless, especially for a teacher with more creativity than us! Nevertheless, it shows that even from a passive examination of the automaton in action we can ask useful educational questions. And, though the automaton at first appears very simple and easy to "figure out," there are still many questions to be asked of the design.

We mentioned before that the simplicity of the automaton coupled with the delightful motion it exhibits is an added benefit. One of best successes of our automaton is the fact that we combined lots of simple parts to produces an automaton that elicits "wow" responses. The response to the motion is enhanced when one investigates the mechanism responsible for the motion and sees that is it actually quite simple. The combination of these two serves to demystify our automaton -- in our minds, this is extremely positive! In other words, on one hand we have a wondrous dancing figure, and on the other hand we have a collection of simple devices that produce the wonderful behavior. By seeing how simple mechanisms can produce striking results, an observer will hopefully conclude that many seemingly magical devices are based on simple motions in combination. If we're lucky, this could encourage further exploration of the physical world that may in turn demystify other objects -- for example, after seeing our simple push rod (converting rotation motion in to linear motion), the motion of engine rods (our push-rod in reverse, converting linear motion into rotational motion) might be more accessible.

Thus, if our automaton -- first through the delightful motion, and then through a investigate examination -- can elicit investigation of the world around us, and inspire a curious and questioning mind, then we have certainly succeeded in creating an object with educational value.

Appendix

Part A

```
;;;;;;;;;;;;;;;;;;;;;
; Foot and chest movement cricket code
;
; This file contains commands to read from a sound
; sensor and drive two motors (differently) based
; on the sound sensor results.
;;;;;;;;;;;;;;;;;;;;;

; Global variables
; sval holds the output from the sound sensor
global [sval]

; Foot-and-chest mover routine
; The neat thing here is the multitasking via
; the 'when' construct.
; One value from the sound sensor drives motor a (foot)
; while other drives motor b (chest), and both are
; polling sound sensor.
; Notice that the foot only rotates one direction because
; of our push-rod design, while the chest ('when' block
; rotates both. Resist the temptation to brake the chest
; motor -- in this case doing so will cause the motor to
; favor one direction.
to footandchest
    setsval clap-get-volume
    if sval > 70 [a, thisway setpower 3 onfor 2 off brake]
    when [ clap-get-volume > 80 ] [b, thisway setpower 1 onfor 1
thatway onfor 1 ]
    footandchest
end

; Simple function to query the sound sensor (bus device)
; Not much to this, just read the volume ('$02'),
; versus the boolean clap ('$01').
to clap-get-volume
    ; Be sure to change the '$169' to the specific
    ; bus address of the sound sensor you're using.
    bsend $169
    output bsr $02
end
```

Part B

```
;;;;;;;;;;;;;;;;;;;;;
; Arm movement cricket code
;
; This file contains commands to read from a sound
```

```
; sensor and drive one motor, producing different
; motions based on the values returned by the sound
; sensor.
;;;;;;;;;;;;;;;;;;;;;

; Global variables
; sval is the volume output from the sound sensor.
; oncea is a tweak to help keep our arm from favoring
;    one direction of rotation.
; onceb same as oncea, but for motor b.
global [sval oncea onceb]

; Test procedure to view sound sensor output on computer
to test
    setsval clap-get-volume
    send sval
    test
end

; Arm mover routine
; Works primarily with cascading ifelse tree to produce different
; motion amplitudes based on volume level.
; It's tricky to keep the motor from favoring one direction of
rotation!
; To keep it from favoring one direction, we move one direction, then
; brake, then rotate the other directon (once) when the volume is low
enough.
; Hard to explain -- try to read the code -- this was developed
; through trial and error testing.
; Key improvements were:
; (1) rotating a bit when the volume is low, but only once.
; (2) having equal number of 'thisway'/'thatway' 'ifelse' bodies
; (3) playing with when to rake, and when not
to groove
    setsval clap-get-volume
    if sval < 70 [if oncea = 0 [setoncea 1 a, rd onfor 1] a, off]
    if sval > 70 [setoncea 0]
    ifelse sval > 130 [a, thisway setpower 8 onfor 1 brake ]
          [ifelse sval > 110 [a, thatway setpower 6 onfor 1 brake ]
          [ifelse sval > 100 [a, thisway setpower 4 onfor 1 brake ]
          [ifelse sval > 90 [a, thatway setpower 3 onfor 1 brake ]
          [ifelse sval > 80 [a, thisway setpower 2 onfor 1 brake ]
          [if sval > 70 [a, thatway setpower 1 onfor 1 brake] ] ] ] ] ]
    brake
    groove
end

; Simple function to query the sound sensor (bus device)
; Not much to this, just read the volume ('$02'),
; versus the boolean clap ('$01').
to clap-get-volume
    ; Be sure to change the '$167' to the specific
    ; bus address of the sound sensor you're using.
    bsend $167
    output bsr $02
end
```

```
---------------

global [sval]

; foot-and-chest mover routine
; the neat thing here is the multitasking via
; the 'when' construct
; one value from the sound sensor drives motor a (foot)
; while other drives motor b (chest), and both are
; polling sound sensor
; notice that the foot only rotates one direction because
; of our push-rod design, while the chest ('when' block
; rotates both. Resist the temptation to brake the chest
; motor -- in this case doing so will cause the motor to
; favor one direction.
to footandchest
    setsval clap-get-volume
    if sval > 70 [a, thisway setpower 3 onfor 2 off brake]
    when [ clap-get-volume > 80 ] [b, thisway setpower 1 onfor 1
thatway onfor 1 ]
    footandchest
end

to clap-get-volume
    bsend $169
    output bsr $02
end
```