THIS = THEN = THAT
Artifact report

CART 360 – Physical Computing and Tangible Media
Concordia University

Adèle Bédard
December 4th, 2019

Github repository:
https://github.com/adelebedard/cart360

➔ Insights & Future Development

The project's intent is to create an object that already has this childish or playful connotation and encourage adults to try and play with it. In that sense, I wanted to encourage people to try things with it. That's why I decided to use mainly the data from the gyrometer to decide the frequency of the tone. In fact, the most obvious solution would have been to have the note higher when it goes faster, but I did not want it to be that direct, for people to be able to experiment and try things. I maybe could have done more with the Mozzi library, but I had a hard time with the compatibility, so I decided to just use the tone function, for this demonstration. The sounds generated act like a melody, since I used only real notes frequencies, so it isn't too dissonant. The form also feels playful or childish. It almost looks like a DIY for kids, with the felt and the big apparent seams. I wanted it to feel soft at the touch and not rigid, so that people would have the tendency to play nice with it, and just give it small throws, rather than throwing full force. The main idea is to use it with other people. Throwing the ball and seeing what melody is created, depending on the way we throw. We can also do it with multiple people, throwing it around a circle for example. One thing that could be interesting to add with only one ball would be that a different sound occurs when the ball is thrown two times in the same direction. In adding that, it could detect that either people are running to keep throwing in the same direction, or there are at least three people, and either is encouraged! It is also possible to use it alone, since for now my ball doesn't recognize directions. One of the applications that I find very interesting with that is juggling. I feel like juggling is similar to music in a way, since it creates a rhythm. By reinforcing this rhythm with the musical ball, it brings out a lot more the meditative aspect of both juggling and music. It also makes the juggler experiment more things with the rotation of the balls, which can be very interesting as well as challenging.

For sure, this ball is a very simple example of the concept that I would like to implement: it could be expanded in so many ways. One of the ways that would be very interesting is to have many balls, let say three, that would be able to communicate to each other. Therefore, each ball could have its uniqueness, but also be interacting with the other balls, their movement influencing its sound. It could create a more complex soundscape, that would need each ball's movement to be complete. It could also play on absence and highlighting the moments where there are less people together, as something more intimate. The idea behind this is to create a community where play and creation is at the center of it. It would bring people together to create something, a music or soundscape. Music is also an interesting aspect of it: I feel like music is often something that helps gather people. And in that case, my ball does too.

- **Research**

Materials

I researched a lot of materials, did some tests, with a lot of different ideas.

My idea was to put everything inside a ball and then cover it. I looked at four kinds of balls. First, for my proposal I tried with this one:

It needed to be big enough to contain my materials, so it worked fine. However, that was not how I wanted the final product to be. I then looked at three other types of balls.

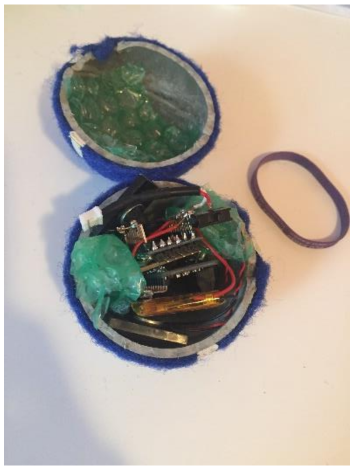
I went for the blue tennis ball, since it is a bit bigger than the normal tennis ball, since it's one of those tennis balls used to put under chairs or other furniture to attenuate the sound. It doesn't bounce like the real tennis ball, but I didn't need that for my project. And I kept the orange ball as plan B if my electronics didn't fit in the smaller tennis ball.

Then, I looked at how to cover my ball. First, I found two main method: like a juggling ball or a baseball/tennis ball.

I opted for the baseball pattern, because it would be easier to do and I found a tutorial online with mesurements, so I had more chances of doing it right (https://feltmagnet.com/textiles-sewing/How-to-Make-a-Cloth-Ball). Also, I want to still be able to open my ball, so I don't want to sew everything up to the top on one go. Therefore, the baseball kind of covering allows me to let the top be uncovered at sew it only at the end.

I looked at a few covering materials, like balloons or leather, but quickly targeted felt to be the best option, because of its softness.

Then, I looked at a way to fill the ball. Both inside it, where electronics are, and outside it, between the ball and the cover. For the inside, I needed something that would not get stuck with the electronics, so cotton or polyesther filling couldn't work. I opted for bubble wrap, often with the bubbles popped, to put between electronics for them to fit better, and to put around it on loose parts so that the ball closes tight and the electronics don't move inside. For the outside, I tried with cotton balls, but keeping them whole would have made my ball be too big, so I decided to break them and make them unravel to become thinner.



Electronics

Speakers and Audio Amplifier

I did a lot of research to understand how the relationship between the speakers and the audio amplifier works. I wanted to be sure of what I should buy. I also wasn't sure if it would work with one speaker, since the audio amp that I got was stereo, so I researched on that as well. I ended up buying a speaker that was too big, so Elio lent me smaller speakers, that I have been using ever since. They fit great, however, the wires are so thin they break very often at the soldering point. Therefore, I let that part be exposed in my ball to be able to still resolder them.

Batteries

I started with simple AA batteries, since it would give me almost the 5V (4.5V) I needed to make the amplifier work at its maximum. However, this was too big to fit in a ball, so I researched on LiPo batteries. I wasn't sure how these JST connectors worked, so I asked Elio. I ended up buying it anyways, since I had to have them quicker, but I found later that I should have just added JST connectors to my order, which where 1$ each. It would probably have saved me a lot of trouble. But I didn't want to make a new order with just that, since it would have cost me 10$ of shipping for it. Then, Elio suggested that cut the jst connector of the battery and solder a new one as well as other wires to be able to recharge it and connect it to the micro controller. However, I realized that it would mean that my microcontroller would be on all the time, since the battery would be soldered directly to it, so he suggested that I insert a three-way switch, so that it either recharges or activates the microcontroller, not both.

Microcontroller

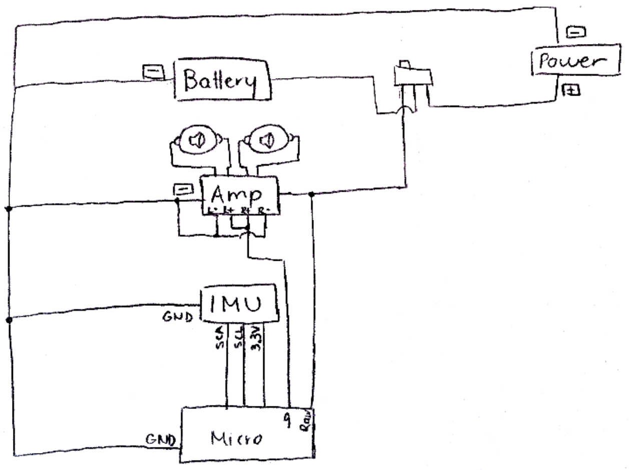
I looked a bit at microcontrollers, but ended up quickly deciding to buy the Arduino Pro Mini 3.3V, since my IMU had to be on 3.3V, and I had to have something small. It also allowed the I2C communication. I also did some research on how to program it, first using Arduino Uno, then using a FTDI. Elio lent me some, but they were all 5V, so I bought a 3.3V one. In the meantime, I used a LD33V, which convert the VCC signal to 3.3V, but I was still afraid to break my IMU, because the signal going from the SCL and SCA pins was greater than 3.3V.

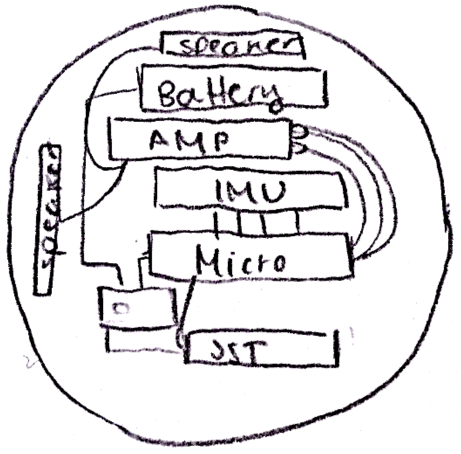
IMU

At one point, I used my Arduino Uno, with power coming from my battery to test my IMU. However, I didn't know that the battery had to be connected to the V_{in} rather than the 3.3V, so I burnt my 3.3V pin on my Uno and the IMU. I had to look a bit online to confirm what happened and for it not to happen again. I also did some research on how to connect with I2C, since we hadn't seen that in class at that point.

Circuit

My final circuit is: (see GitHub repository for other circuit drawings)





Code

Mozzi

To make the sound, I first looked at many projects online using sounds, and tutorials on how to make music with Arduino. Many of them used the tone function to create a melody, and other more complicated projects used their own customed libraries. Then, Elio suggested me to try Mozzi, and it was a very good idea. I started with the FMsynth example, but I couldn't figure out how to modify it and the progression from low volume to high volume of the sound wasn't really something I wanted. I therefore tried and compared three other ones: the sinewave, the vibrato and the oscil_wash_control examples. They all seemed to be similar. However, when I tested them, my speakers were not soldered properly so I tested it with only a piezo buzzer, since it was the weekend and I had not access to the sensor lab. Then, when I tested them with my actual speakers, they didn't sound as great, and I didn't really explore further, since I found out that Mozzi and the IMU had a compatibility problem (see section below).

IMU alone

I used only the first example of the Sparkfun library for this IMU, which uses I2C, because it corresponded to my needs. I was a tutor in mechanical physics, so I was very confident that I could figure out how to process the data from the IMU, particularly the accelerometer. I quickly realized that there was something weird with the way it was calculating acceleration. It was saying that when the ball is on the ground, not moving, it had an acceleration of 1g, which doesn't make sense, since the ball isn't moving. Therefore, I assumed that it was not counting the gravity. In the air, the acceleration of an is 9.8m/s^2, equivalent to the gravity force. However, since it was not counting gravity, I assumed that my IMU would say that there is zero acceleration in the air. And I was right, in the air, it calculates an acceleration around zero. I put a threshold at 0.2, since the values are not calculated often, so I wanted it to make a sound quicker. Therefore, it starts to make a sound even just before it is in the air if the value is collected at that time. Then, I realized that the gyrometer seemed to work really well for my needs. In fact, I thought it would calculate the orientation of the ball, but it rather calculated the way it is rotating. Therefore, when it is rotating at that instant its value is high, while when it is steady its value is close to zero. It gave me exactly what I needed directly to modulate the frequency of my sound.

Incompatibility research and explanation

I found out that there was a problem of compatibility with Mozzi and the IMU libraries. First, it gave me the error: Error uploading to Arduino Pro/Arduino Pro Mini. I tried with my Arduino Uno board, but it didn’t work either. I searched online and it said something on how maybe the IDE was not up to date or something in that vein. I didn’t believe it was the case, so I searched in the error message further. I found that the error was given just after "multiple definition of `__vector_24'". I researched that online and found a couple of forums with people having this problem with servos and other libraries. They explained it was a timer problem, two libraries using the same timers or timer interrupts. The error message also gave me the first place where this was declared, which was in a file in the Mozzi library. I searched that file but couldn’t find anything related to timers, as the examples in the forum gave. I switched the library declarations, putting the IMU declarations on top and found the other file where it was declared, this time in the Wire.h library, which I don’t know how to access. Therefore, I searched about Mozzi and Wire.h compatibility online and found this website (https://github.com/sensorium/Mozzi/issues/46) that suggested one of the Mozzi examples that used I2C, but I couldn’t figure out the example. I then asked Elio about this, and he told me the day after that there was a solution, but I preferred to have my ball complete and making sound, rather than to spend time on this issue. For further development, resolving this bug and using Mozzi would be a great idea.

IMU + tone

I ended up using the tone function to create sounds with the IMU. I have decided to only use real notes, and found a website outlining the frequencies of each note, from B0 to DS8.

https://maker.pro/arduino/projects/arduino-speaker

I recopied these frequencies in order in an array, so that each time the ball is in the air, it goes to find the value in the array corresponding to the note that must be played. It is a simple cross product with the hypothetical maximum of the gyrometer value, the current value and the number of elements in the array, 89.