4/25/2018

Banana Genocide Machine Report

Cullen Diebold
CSCI-4957 Embedded Systems

Banana User Manual:

1) The way the Banana Genocide Machine is crafted is with a removable top and bottom. This gives access to all the boards and controls if a problem should ever arise.
2) Be careful when handling the machine. NEVER pick up the banana machine from the yellow banana slicer itself. ALWAYS pick up the banana machine from the base or container to ensure no eviscerations.
3) Attach the three provided alligator clips to the banana machine in the following sequence.
   - You will be provided two yellow alligator clips and one black alligator clip. Connect the black alligator clip to any of the bottom ground sections on the front panels Makey-Makey interface. Now connect one of the yellow alligator clips to the front panel clip named 'CLICK'. Connect the last yellow alligator clip to the front panel clip named 'SPACE'. It should look like the below image.

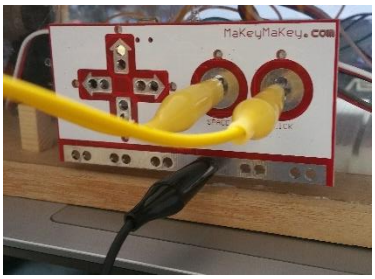

Figure 1: Makey-Makey connections

4) Connect each one of the ends of the alligator clips onto a banana or any conductive object.
5) Once the banana machine is placed in your local banana slicing spot, connect the banana machine to power by using the provided 5v power supply into the back right of the machine. While on the back of the machine, you may notice a micro-USB connector towards the middle of the product. Connect a USB power supply to the micro-USB port to provide power to the main processor and the banana-user-interface (Makey-Makey interface).
6) When the user is ready to slice a banana follow these steps.
7) The Makey-Makey interface alligator cable named 'CLICK' is the label that causes the servo motors to spin counter clockwise, making the banana slicer move upwards, while the label named 'SPACE' makes the servo motors spin clockwise, which in turn slices the banana.
8) To slice a banana by using the banana machine, the user is first required to unpeel a banana. Once the banana is unpeeled, the user shall place the naked fruit on the banana lunette (chopping block).
9) The user must now grab the 'SPACE' connected banana, along with the ground connected banana (black alligator wire).
10) Be amazed as your breakfast is being made for you before your very eyes!
11) Once the banana machine has sliced through roughly 40% of the fruit, the servo motors will slow down. At this point, the user needs to interact with the machine by applying pressure to the middle segment of the banana slicer. Make note to push on the yellow banana slicer frame, and not the razor blades to avoid any injury.

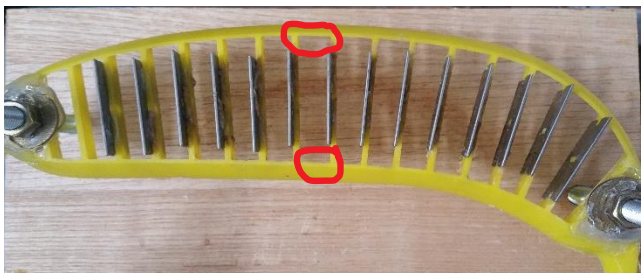

Figure 2: Pressure points on the banana slicer for continued slicing action.

12) Now that your banana is roughly 80% cut, that is good enough for us! It's time to remove the slicer and enjoy your freshly cut, equally distributed, banana.
13) To enjoy your banana keep holding the ground banana (along the black alligator clip wire) and grab the banana with the yellow alligator wire named 'CLICK'. This will cause the banana machine to rotate the servo motors away from the banana causing the blades to rise.
14) After the banana (which should still be within the slicer) is at a comfortable distance from the banana board, insert a small object (end of provided toothbrush) and push the banana segments out of the slicer.
15) Enjoy your sliced banana!
16) Actually, don't. The razor blades attached to the banana slicer, have been coated with a thin layer of water and oil, making them possibly dangers.
17) Still waiting on FDA approval.

Engineering Notebook Transcribed:

March 13 – 2018:

Conception of Idea. This idea was a created on the back of a Mellow-Mushroom napkin, when my other computer science friends and I bantered about what would be the most 'useless, but enjoyable' kitchen utensil to use.

March 14 – 2018:

Online Shopping. On this day, I surfed the internet to find high-powered servos, and any additional parts/boards needed to make this monstrosity a working creation.

March 19 – 2018:

Programming started. The first few lines of code were written and derived from the Servo lab provided by my Embedded Systems instructor. I could control only one of my high-torque servos, but ended up running into a power issue, which I didn't realize was a power issue at that time.

March 24 – 2018:

Complexities with programming. Originally, I planned to copy most of the code used in our servo lab, but instead I found that using the previous project was a better starting point. I attempted to create a finite state machine (FSM) with my code, which ended up not working as planned. I scrapped most of the ideas here (as seen by the documented and commented code) and decided to try again later.

March 26 – 2018:

Change programming. I changed the programming to the current design seen, with a few exceptions. The original program only worked with one servo and did not read a value from the Makey-Makey interface. My next challenge was to read input from the Makey-Makey interface. Once I could read proper/correct values from the alligator clips, I was ready to create a crude prototype.

April 2 – 2018:

Buying hardware. On this day, I traveled to two different hardware stores in search of finding the perfect attachment to my servo. I found that most hardware stores carry a type of bolt, called lag bolts, to use in the implementation process. After getting home, I realized the previously bought bolts were too long which made the prototype extremely top heavy, meaning I had to go buy a set of smaller bolts for the machine.

April 8 – 2018:

Too much glue. I found the perfect bolts to use for the machine, I used CA (super) glue and clamps to attach the banana slicer to the prototyped frame. This caused the nuts to stay onto the banana slicer, but also caused the bolts to attach to the nuts, making the bolts unable to spin.

April 9 – 2018:

Destruction. On this day I scrapped the plexiglass frame and tested the banana slicers current cutting potential. Since the bolts were glued to the nuts, I was required to destroy the previous frame to free up the banana slicer. Once the banana slicer was free, I attempted to cut the banana, peel and all by applying a 5-lbs weight to the top of the slicer. I was unable cut through the peel and fruit itself, which caused me to explore other ideas on cutting up the banana. I found razor blades in my workshop, which I then attached to the banana slicer, making it cut through peels and bananas with relative ease.

April 16 – 2018:

Power supply solved. This day was as free lab day where we could bring in our projects and work on them independently with help from our professor. During this lab session I prompted the professor with the power supply issues I was having. He stated to try to power the Makey-Makey interface with the TI-MSP432 launchpad board. I had to do a little research to find that the launchpad pushed 500 mAh, and the Makey-Makey board accepted up to 500 mAh, but it worked perfectly. This solved all my problems, and made the servos spin with as much power as needed.

April 18 – 2018:

Version III. Since I now had a properly functional hardware and software base, I decided to create a more permanent structure for my project. I decided to use a wooden frame that broke into two separate pieces in case any electrical problems would arise.

April 19 – 2018:

Completed case. On this day, I attached all the electrical components to the inside of the case, but I found out quickly that I did not anticipate the power supply connections. I had to drill into the already created case and file out the back end of the plexiglass to make both power ports accessible.

April 22 – 2018:

Final Testing. This day was the final day of testing. I had completed the case, and attached all the components to their correct locations within the case. On this day, I did a quick test-run of how the machine would slice a banana, and it worked as well as the servos could possibly work. I am super proud of this process, and this banana killing machine will always have a home on my bookshelf.

Schematics:

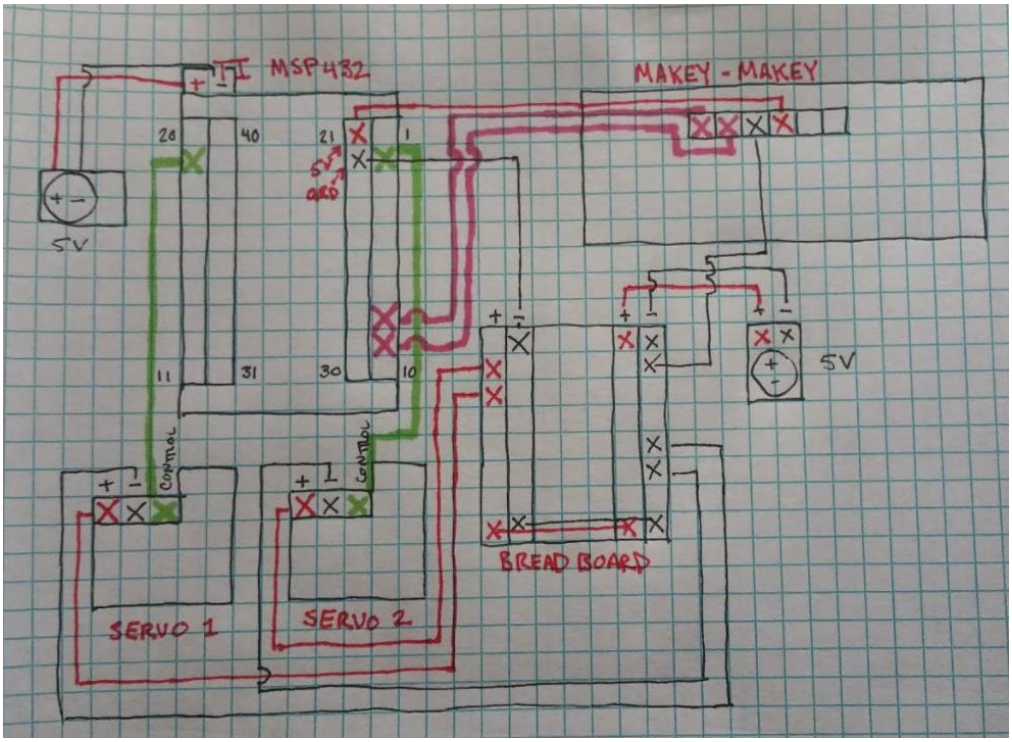

Figure 1: Banana Machine Schematic

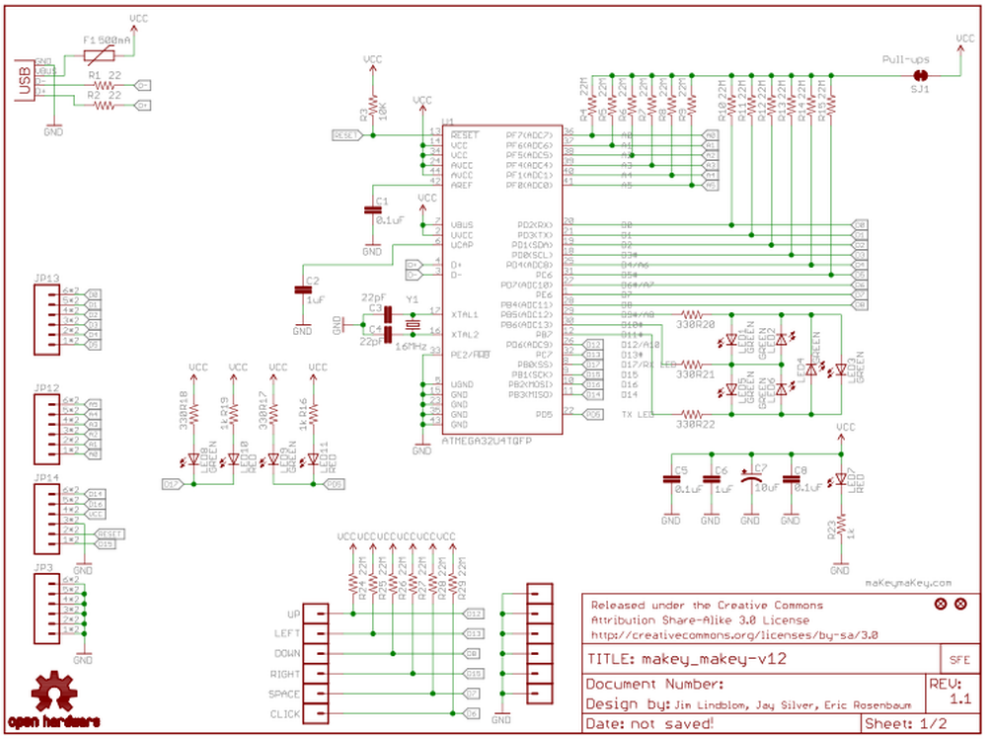

Figure 2: Makey-Makey Schematic

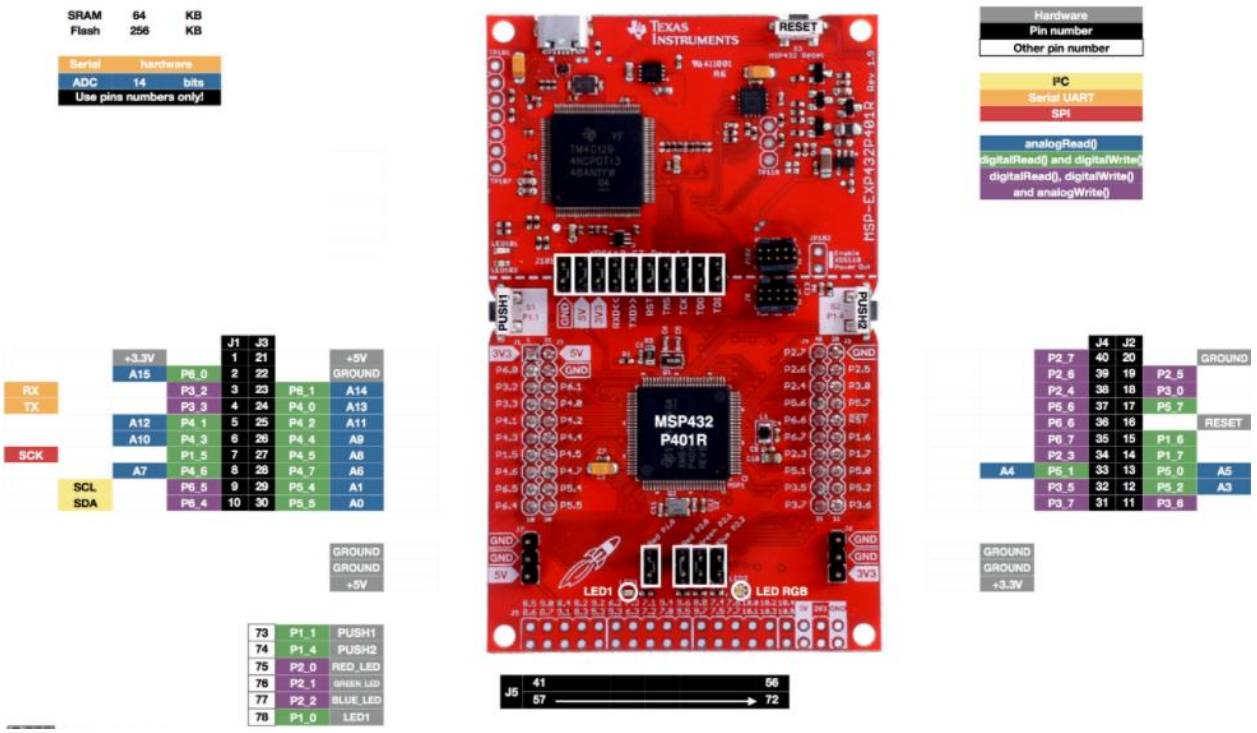

Figure 3: launchpad Schematic

Project Description:

What does the project do?

The project that I worked on is patented by the name of "The Banana Genocide Machine", and it does exactly what you would think it does. It uses bananas to kill other bananas, and since it's robotically controlled, it could wipe out the entire banana population with ease. Simply stated, this machine takes an unsliced banana, and uses an electrical charge that is conducted by picking up other bananas to send a signal to the micro-controller to either destroy the banana, or to raise the banana slicer to save the banana from the impending death. The banana genocide machine uses other bananas as switches, and when the circuit is closed something happens, either the banana slicer moves upwards or downwards.

How does the project work?

The Banana Genocide Machine works by reading the values given off by the bananas being touched. The values being read are 'not-touched', 'touched-clockwise', and 'touched-counterclockwise'. When the pin that is connected to the clockwise banana is being touched, and the ground is being held by the other hand, there is code to tell the servo motors to spin in a clockwise direction, making the banana slicer move closer to the banana cutting platform. When no banana is being touched the output is of a 'LOW' value, and when the low value is read, the micro-controller tells the servos to not turn by sending a stable value. One of the most important elements of this build is how the nuts are attached to the banana slicer itself. If the nuts were not solidly attached to the banana slicer, then none of this process would work. The stationary nuts, allow the bolts to spin freely, which dictates the direction of the slicer. When the bananas are being held, the microcontroller treats the bananas as a closed circuit which causes an action to happen on the servos, which then creates the up-down motion seen in the demonstration.

Challenges and how I overcame them.

The project displayed during presentation was a collection of difficult challenges that needed to be examined and contemplated to create what is seen today. These are a few of the main challenges I came across while developing this product: Power supply issues, Interfacing issues, Construction issues, and Programming issues.

When dealing with power supplies, it is always wise to double check information about your circuits. If there is too much power being pushed into a board, then that board may end up fried and dead, while on the other hand, if there is too little power being applied then no work will be done and nothing will happen. During this build I used a 5-volt power supply to power two micro-controllers and two 5-volt servo motors. I thought that there would be enough power being sent to all the electrical components, but I was sorely mistaken. The main issue this caused, was not enough power to power every component within this machine. That problem became solved when I decided to power the board components separately from the servo motors. Once the motors were powered by the external 5-volt power supply, and the boards were powered by another external 5-volt power supply, the machine worked as intended.

Connecting two separate boards is a difficult process, because the inventor needs to know how the two boards are going to communicate. I spent around two afternoons studying the type of interface

that a Makey-Makey board uses. I did not find any information about these boards and their interfaces. Luckily, I found an article describing how a Makey-Makey interface is similar to adding external buttons to a micro-controller. Once I found this out, I attached jumper wires from the Makey-Makey interface, to the MSP432, and read the pin values associated with their 'buttons'. This allowed me to read if the pin values being read were 'LOW' or 'HIGH'.

There will always be issues when inventing a new product. The main construction issue I came across, stated before, was with glue. I found myself gluing parts together that were supposed to be movable parts. This meant that I was forced to destroy the entire project and rebuild it from scratch due to how it was designed. As for other construction issues, when I created the case for the components, I did not consider the external power supplies used to power this device. I was once again forced to re-think my architecture to design a case that would be able to provide power to what was necessary.

Lastly, all programs have bugs in them. I had a difficult time finding the correct way to program the micro-controller so that it did exactly what I wanted it to do. The main issue that I was having programmatically, was that I was trying to create a finite state machine (FSM) to control my servo motors, when all I needed to do was create more simplistic code and add more power to the machine. At first, I thought my code was erroneous, because when I would attempt to turn the servo motors in a certain direction, they would stutter and not respond when I let go of the switches. I later found out that I was not providing enough power to all components which was causing the Makey-Makey interface to completely shut itself off, which left the servo motors in a constantly running state. Once power was supplied properly, my program and machine ran as expected.

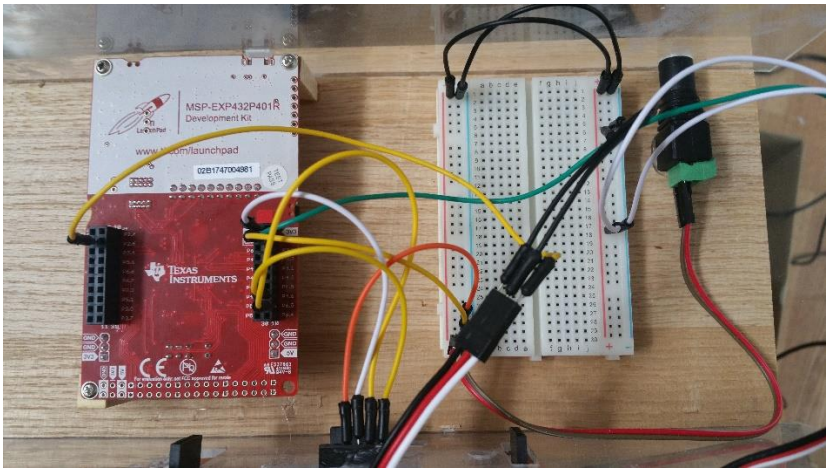

Figure 1: internal components and wiring of the banana machine

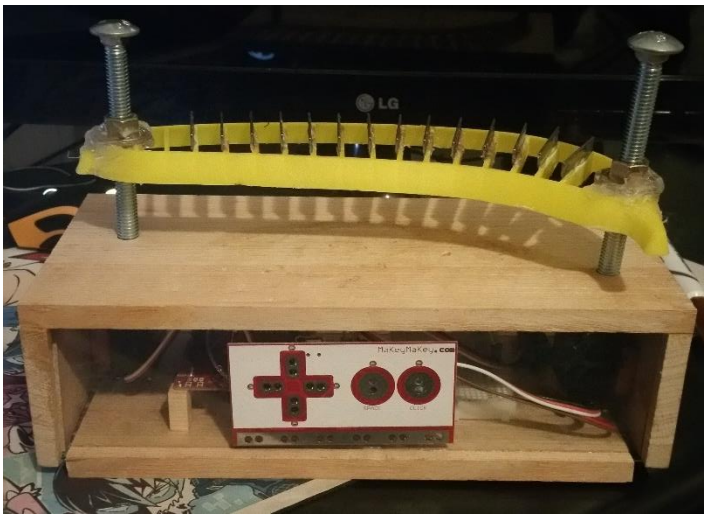

Figure 2: completed Banana Killing Machine

```
void loop() {

  if (digitalRead(9) == LOW)
  {
    //Serial.print("\r\n");
    //Serial.print("Banana NOT Here");

    myServo.writeMicroseconds(1500);       //Makes the servos 'stop' at 1500.
    servo2.writeMicroseconds(1500);
  }

  while (digitalRead(9) == HIGH)
  {
    //Serial.print("Banana Here");
    ///////CALL A SERVO METHOD BELOW INSTEAD OF DIRECTLY SETTING MICROSECONDS//////DID NOT WORK//////
    //clockwise();
    ////////////////////////////////////////////////////////////////////////////////////////////////

    myServo.writeMicroseconds(200);
    servo2.writeMicroseconds(200);
  }

  while(digitalRead(8) == HIGH)
  {
    //Serial.print("\r\n");
    //Serial.print("SAVE THE BANANA!");

    myServo.writeMicroseconds(2200);
    servo2.writeMicroseconds(2200);
  }
```

Figure 3: majority of code for the Banana Killing Machine