J_Ray

Cell Battery Brick

Purpose:

Working in the service industry mean that you are constantly making change for both cash and coin change. Currently it takes too long to have to either get change from the safe or search a bag of coins to make the correct change. Now it can be said that one could just become more organized and it may make the change making process much quicker, but who would want to just give organize when we could automate the process.

Specifications:

- Needs to be compact and light enough to stay in a backpack all day
- Battery operated
- Needs to be able to dispense coins for 8+ hours on a single charge (A hundred transactions is a good starting point)
- Needs to find the minimal number or coins for specific change
- Easily debuggable incase a failure occurs during a shift for low downtime
- Easily refillable

Parts/tools:

- Arduino Micro
- 2 servos
- 4 X 4 matrix keypad
- 9V battery
- Breadboard (will make creating the circuit easy)

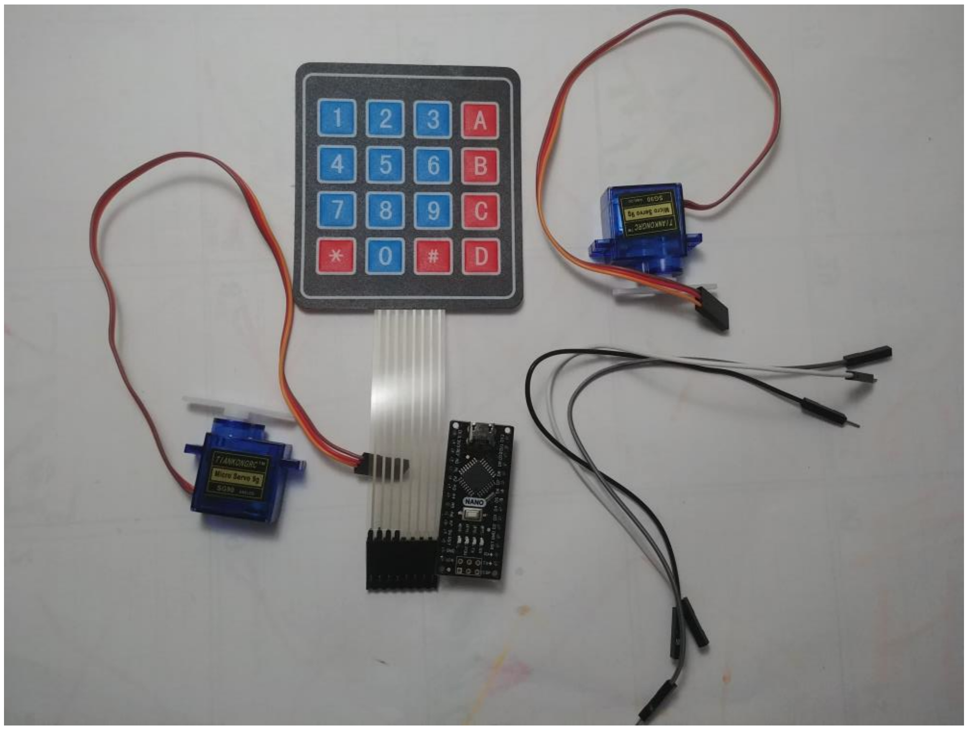
Current project to do list / additions:

- Low power mode
- 3d printable enclosure

Build:

Step 1:

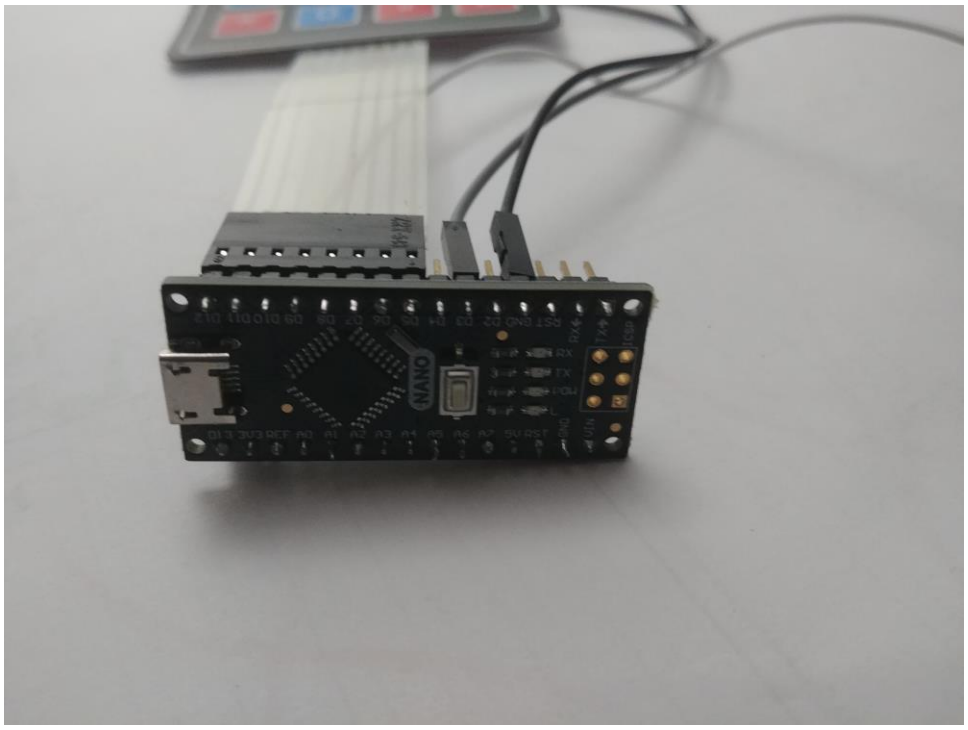
As we always do, we will start this project off by ordering all our components so that we will need. While waiting on these to arrive we have a perfect amount of time to search for libraries we can add to the standard Arduino libraries to make coding this project achievable in a short period of time. The libraries chosen for this project are **Keypad**, and **Servo**.



Step 2:

As the components arrive, we can begin to figure out how to power each and what pins need to be connected the Arduino and which need to be connected to the power and ground bus. While doing this we can begin to define which pins on the Arduino will be used for each component.

One pin on the other side for D13 as well:



Step 3:

This step may seem simple, but many people skip it and end up with code that needs hours and hours of debugging because they were just too lazy to download a sample code. It is not fun to write the “perfect code” for an hour and run it only to figure out you have no idea what most of the code you just wrote means, let alone know how to debug it. To avoid this just play around with sample code for each library maybe even write some small code to make sure you really get it, then you can begin to put the code together. In this project the best uses of your time would be to make sure that if you tell a servo to do turn and then return without having to copy paste code. If it will do what you expect and when reading a keypad, it can always read and display the correct value then you should be good to continue.

Step 4:

We now need to start meshing these two new libraries we have learned into one. Best way I have found is to write independent functions for each to start and then stich the different functions together in the main program. This can be especially useful considering you are normally calling these functions multiple times and it is easiest and best practice to put something you are reusing into a function. Secondly, make sure you define all the values you will be using for functions and globally. Getting this done now will keep you from forgetting to add them later and then having to type a value into an equation directly or just completely forgetting to put them in at all.

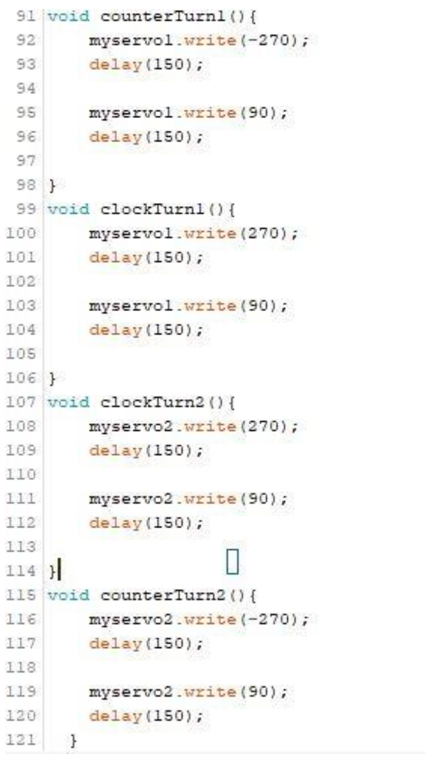
Step 5:

Once we are here, we need to think about the layout of the case because the layout will determine the direction and how much the servos must turn. In the code you can see that I have chosen to make a clockwise turn on servo1 a quarter and a counterclockwise turn on servo1 a dime. This choice is mostly arbitrary, but I liked the idea of the enclosure being somewhat symmetrical. Therefore, I chose to put a larger coin on each side.

Step 6:

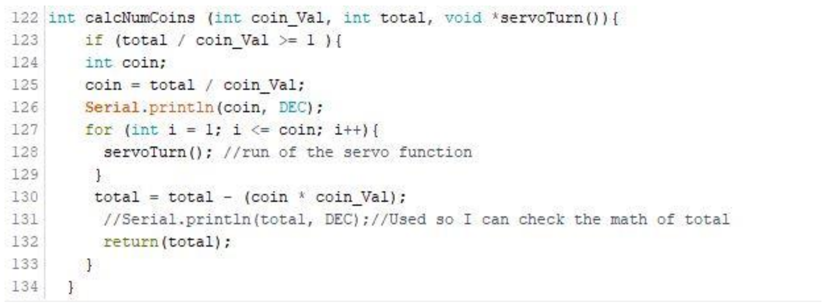
The functions for the servos will be very easy if you learned how to properly use them from step 3so I will not spend much time on them other than to say you will probably have to adjust the delay time and the angles they are turning. Do not forget one of them should be a negative. Now the minimum coins counter function is not bad, but you must understand what each variable being pased into it means. The first value is just the value of the coin and it is dividing whatever the value plugged in was by that and rounding down to the nearest whole number. This is its basic functionality the only read addition I made to it is passing the correct servo function all to the function using a pointer variable.

Relay Functions:



```
void counterTurn1(){
    myservo1.write(-270);
    delay(150);

    myservo1.write(90);
    delay(150);

}
void clockTurn1(){
    myservo1.write(270);
    delay(150);

    myservo1.write(90);
    delay(150);

}
void clockTurn2(){
    myservo2.write(270);
    delay(150);

    myservo2.write(90);
    delay(150);

}
void counterTurn2(){
    myservo2.write(-270);
    delay(150);

    myservo2.write(90);
    delay(150);
  }
```

Number of each coin function:



```
int calcNumCoins (int coin_Val, int total, void *servoTurn()){
    if (total / coin_Val >= 1 ){
    int coin;
    coin = total / coin_Val;
    Serial.println(coin, DEC);
    for (int i = 1; i <= coin; i++){
      servoTurn(); //run of the servo function
     }
     total = total - (coin * coin_Val);
      //Serial.println(total, DEC);//Used so I can check the math of total
      return(total);
    }
  }
```

Step 7:

Only things left to do with the code and circuit is to test it and make sure to hammer out any minor bugs as the main code should work. My code could be cleaned up but has been left with extra lines commented out to make it easy for me to debug in the future if any problems come up.

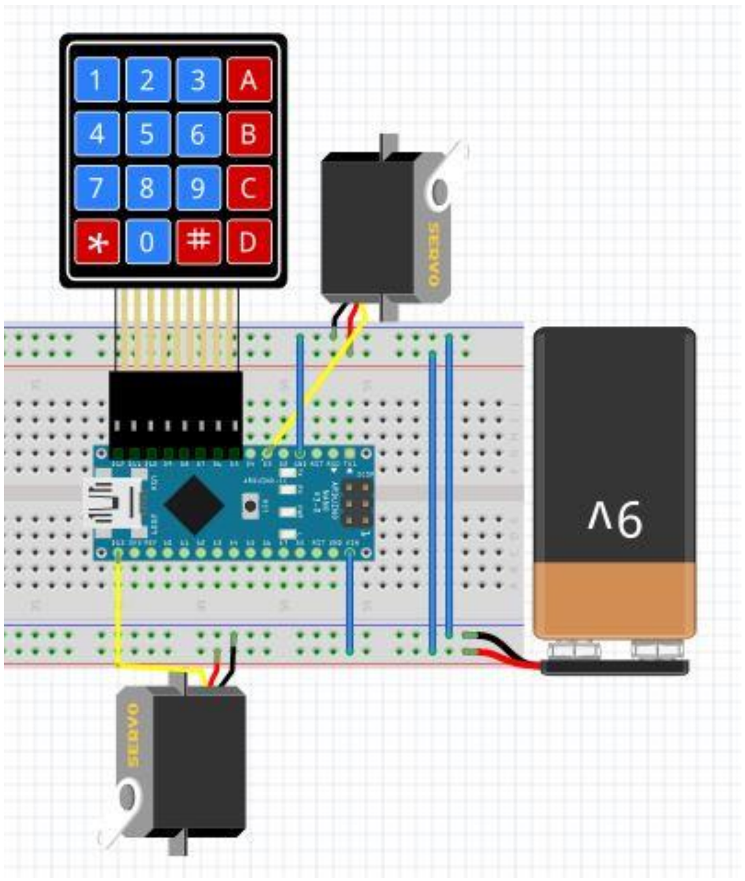
To Do Still Steps:

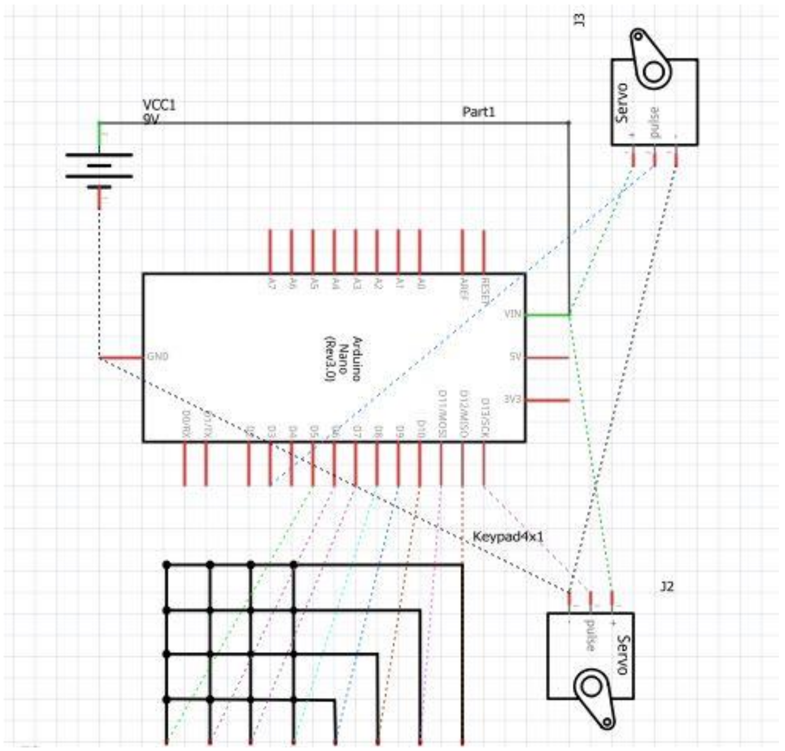
Step 8:

Build in a low power mode as currently the Arduino is either on and listening for a key press or completely off. This may be as good as it will get for me. I am not really concerned with the amount of power it draws while waiting for a press, but it is something that I will be researching at the very least to use on future projects.

Step 9:

Build an encloser. It is all mapped out now just pull out Fusion 360 and model it to be printed.


1
2
3
A
4
5
6
B
7
8
9
C
*
0
#
D
SERVO
SERVO
9V


J3
Servo
pulse
VCC1
9V
Part1
A7
A6
A5
A4
A3
A2
A1
A0
AREF
RESET
VIN
5V
3V3
GND
Arduino
Nano
(Rev3.0)
D0/RX
D1/TX
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11/MOSI
D12/MISO
D13/SCK
Keypad4x1
J2
Servo
pulse