Preface:

This project is intended for the Boy Scout Youth to get a understanding of Robotics but is a good project for anyone learning Arduino stuff. We will be building a Useless box. It satisfies the BSA requirements for Robotics requirement of having at least two axis of motion (door and switch) and must be autonomous. This is a great post PineWood derby Merit Badge that the youth should be able to build on their own with minimal help from their parents or other adults. Creativity and Problem solving are key things you will face. Keeping a record and documentation are also important.

What we are building:

We are building an Arduino powered Useless box like the one seen in this video:

https://www.youtube.com/watch?v=Edn1RIT7E8s

We are taking an Arduino, which is a Microcontroller development kit designed to make it easy to teach all ages how to use microcontrollers and how to program them. It uses the Arduino IDE which can use C programing language but adds some flexibility and gives more feedback for errors then others may. The Arduino is placed on the Breadboard. The breadboard has back side connections between pins. the numbered rows are connected. For Example: row #1 is connected on pins a,b,c,d,e. Also it is connected on f,g,h,i,j. The Red and Blue Rows are used as power and ground bus. They are connected all the way across.

Plug the Arduino in using a gentle pressure placing it where there is good room on all sides but closer to the edge on the side with the USB connector. The ProMicro will have it's 12 pins start from row 3 going to row 14. More hookup info is below in the hookup section.

Definitions:

IDE- Integrated Development Environment. This is the computer program we enter human understandable language in. The IDE will output assembly language, the stuff the processor understands. Essentially the IDE is the translator for us to talk to the microcontroller.

PWM- Pulse Width Modulate, Digital version of analog. Example is if we can only turn a light switch on or off, if we switch it very fast with equal parts on and off, it will appear to our eyes that the light is constant and on only half as bright.

GPIO- General Purpose Input Output. These are Digital input or output. For input you may need to be careful and provide an pullup or pulldown resistor to prevent a float situation. Since the inputs are very high resistance, they can float between on and off states and wreak havoc on your program.

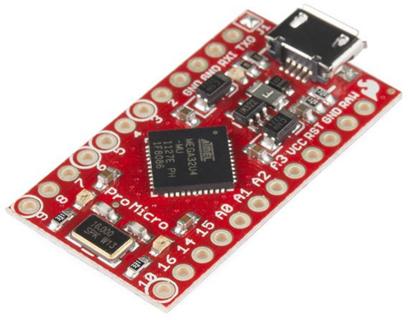
ADC- Analog to Digital Converter, This is how we convert an Analog signal to a digital one that the computer can understand. There are a number of ways to achieve this, But the important take away is they spit out a digital value equal to the analog value.

Getting Started:

First things first, You will need an IDE to program this thing. We'll be using the Arduino IDE. You can get it here http://arduino.cc/en/Main/Software. It is free, open sourced and runs on Windows, Mac and Linux. After you download the software you will want to go to the Getting started page http://arduino.cc/en/Guide/HomePage and follow the steps provided. Since the Arduino ProMicro is Arduino Compatible, but not an official Arduino product we'll also need the add-on files from https://www.sparkfun.com/products/12640. Follow the directions there to install the hardware files so the IDE knows how to talk to the ProMicro. Pay attention to the hookup page as it has special info about how to use the ProMicro, https://learn.sparkfun.com/tutorials/pro-micro--fio-v3-hookup-guide. Specifically about how to use the reset pin to restart the device in bootloader mode for flashing a new program to it.

Hook It Up:

Features of ProMicro:



ATmega32U4 running at 5V/16MHz

Supported under Arduino IDE v1.0.1

On-Board micro-USB connector for programming

4 x 10-bit ADC pins

12 x Digital I/Os (5 are PWM capable)

Rx and Tx Hardware Serial Connections

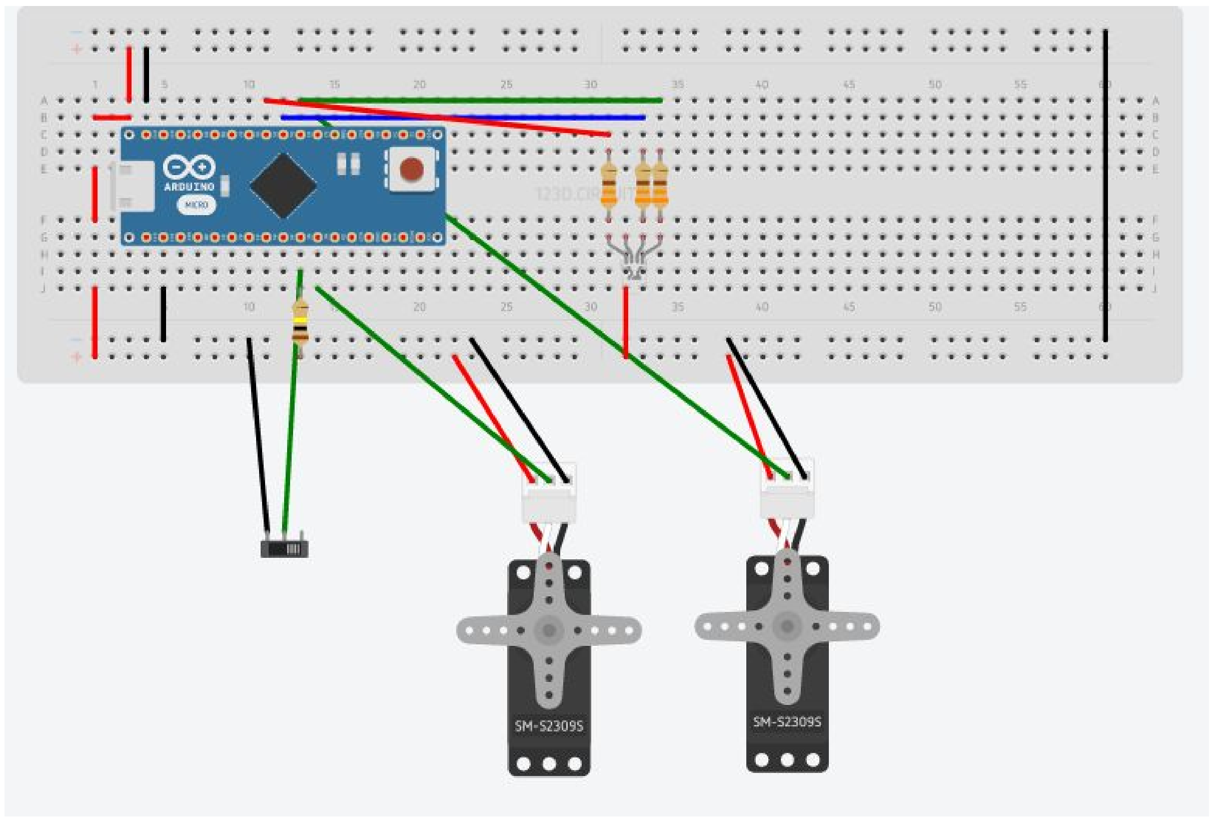
Our Smallest Arduino-Compatible Board Yet!

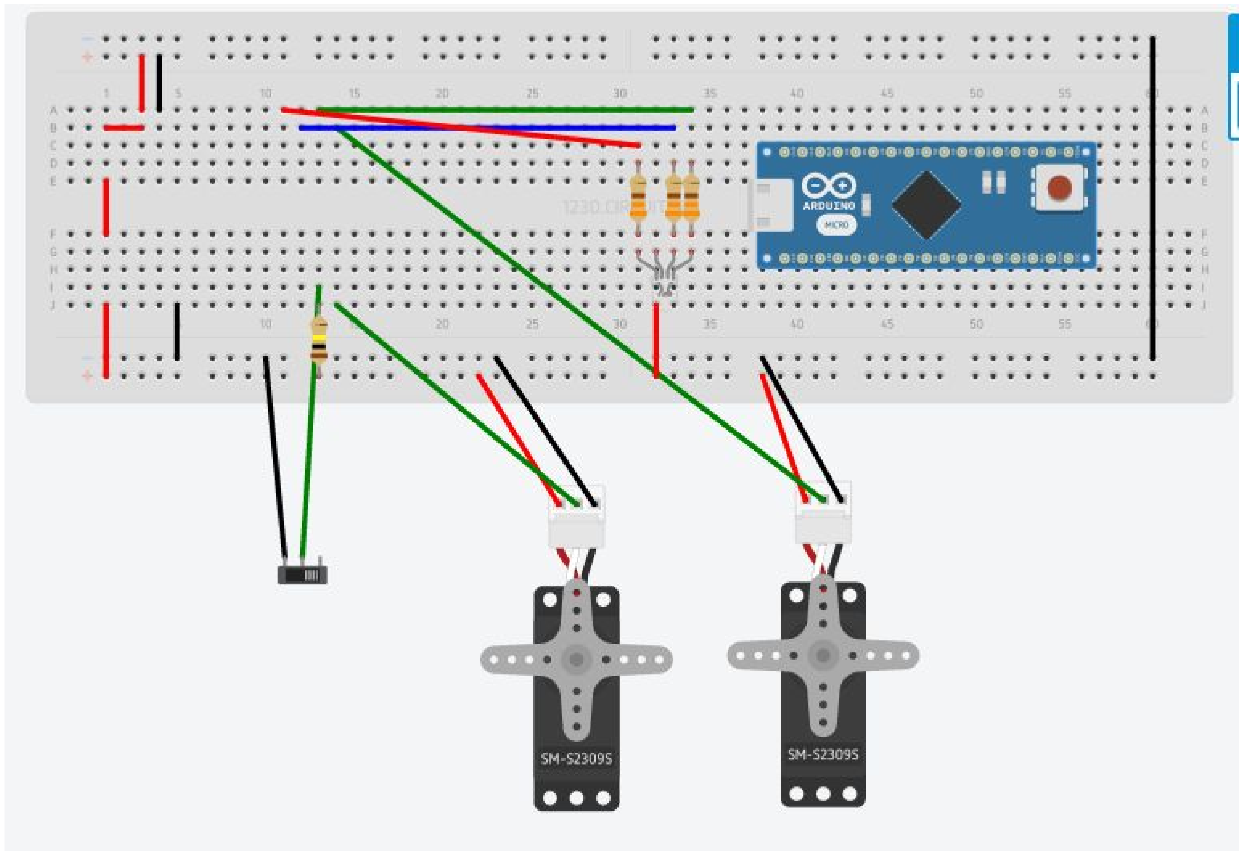
Dimensions: 1.3x0.7"

This tiny little board does all of the neat-o Arduino tricks that you're familiar with: 4 channels of 10-bit ADC, 5 PWM pins, 12 DIOs as well as hardware serial connections Rx and Tx. Running at 16MHz and 5V, this board will remind you a lot of your other favorite Arduino-compatible boards but this little guy can go just about anywhere. There is a voltage regulator on board so it can accept voltage up to 12VDC. If you're supplying unregulated power to the board, be sure to connect to the "RAW" pin on not VCC.

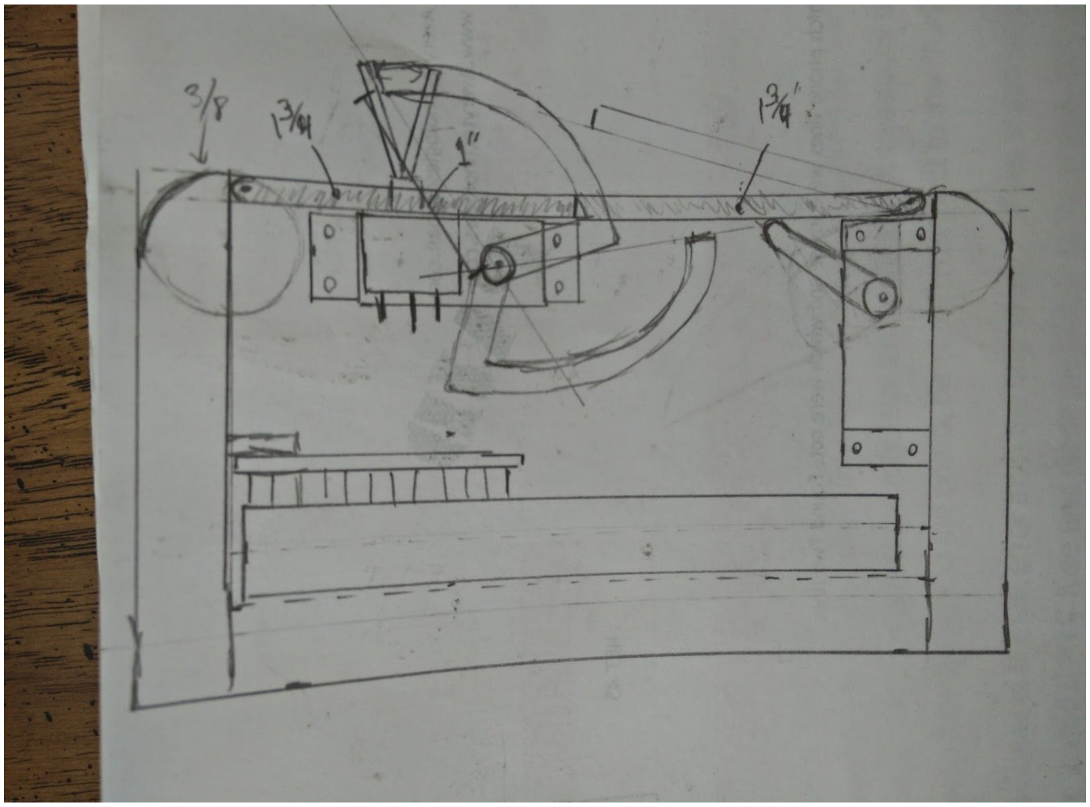
This latest revision corrects the silk error from the last version of the board so that pin 14 is correctly labeled. We've also added a PTC fuse and diode protection to the power circuit and corrected the RX and TX LED circuit.

The program I used to layout the Bread board Does not have the ProMicro we are using. Instead I used the right locations on the breadboard. In the second photo I moved the ProMicro to show the wire locations better. The ProMicro should be placed pins 3 through 14. I like to use Red for power (5V), Black for Ground and Green for signal wires.
The Anode on the RGB LED is the second one in (pin 2). It's the longest one. The Red LED is the single one to the left (Pin 1). The other Center one is the Blue LED (Pin3). The Green lead is the last one (Pin4).
The Switch is pretty simple. Center pin is common. If there are three pins, then either out side pin can be used. This is called an On-On switch. If there is only two pins then use the other pin. This is an On-Off. By only using one of the two outside pins, we use it like an On-Off.





Build Box:



I chose to build a smallish box 4” x 3” inside and about 3” tall inside. The main thing is that you have room for the servos to operate and for the breadboard. If you want to have LED’s and Speakers, go for it. What you build the box out of is up to you. I chose ⅝ “ plywood. This is a chance to get creative. For the placement of things, I recommend drawing it up like I have, but you may prefer to mock it up once you can see and feel the box.

Parts List: (you may need to order replacement parts)

Pro Micro from Spark Fun An Uno, Micro or any other can be used, but you'll need to figure out the hookup.
2 Micro Servos (DX.com link http://www.dx.com/p/towerpro-sg90-9g-mini-servo-with-accessories-12859#.Vi76alUVhBc) Any standard servo/micro servo of 9g or more should suffice.
RGB LED Common Anode
Switch
Switch pull up resistor (1x 100k Ohm)
LED Resistors (3x 330 Ohm)

Run a program:

Now you can burn your first program. Goto http://arduino.cc/en/Tutorial/HomePage and look at Blink. You can copy and paste the code or type it in. You'll have to change some things. First off, the ProMicro doesn't provide an onboard LED for testing. Also We'll want to be sure that we don't use one of the programing pins needed to communicate with the computer for the test program. Make sure you've read the sparkfun hookup guide, as it explains what pins to be aware of.

Flash the Skeleton:

Well you now have used the IDE. We have lots and lots of Example code we can review at the Tutorial page above. Now all that is left is to install the Skeleton. Keep in mind, you'll have to make some changes to get things to work for your box! If you have things plugged in differently or have other additional items, you'll need to make those changes. You will definitely have to adjust the servo parameters to set the start and stop positions for both.

What now?

Have fun. You can flash new programs on this thing thousands of times. The real magic is in you!
Here is a Video of it opperating at this point: https://www.youtube.com/watch?edit=vd&v=Edn1RIT7E8s

You get to get to make the magic happen. Change the delays. Change the LED colors and flash. Make the Servos move slower. The arduino.cc website has tons of examples. Look at some of the LED and servo ones for ideas. Read the code and copy parts, edit, see what works. What I've provided in the skeleton is the bare minimum to make it work.

Skeleton:

```
#include <Servo.h>

int ledPinRed = 16;  // LED we'll turn on while program runs. Helps with troubleshooting.
int tiggerSwitch = 8;  // The switch to activate the machine
int posDoorUp = 120;  // Variable storage for servo positioning, used for Door opening
int posSwitchUp = 180; // Variable storaged for servo, used for swing arm up.
int posSwitchDn = 1; // Variable storage for servo, used for swing arm down.
int posDoorDn = 80; // Variable storage for servo, used for Door Closing.
```

```
Servo servoDoor; // Create Object for Door Servo.
Servo servoSwitch; //Create object for Switch servo.
int ledPinGreen = 14;
int ledPinBlue = 15;



void  setup () {

servoSwitch.attach(10);  //setup Switch servo on pin 10
servoDoor.attach(9);  //setup Door servo on pin 9.
pinMode(ledPinGreen,OUTPUT);
digitalWrite(ledPinGreen, HIGH);
pinMode(ledPinBlue, OUTPUT);
digitalWrite(ledPinBlue, HIGH);
pinMode(ledPinRed, OUTPUT); // setup pin 8 as an output.
digitalWrite(ledPinRed, HIGH);
pinMode(tiggerSwitch, INPUT); //setup pin 14 as an input for the Switch.
servoSwitch.write(posSwitchDn);
servoDoor.write(posDoorDn);

}

void loop() {

digitalWrite (ledPinRed, HIGH);  //Turn on LED so we know things are running

if( digitalRead(tiggerSwitch) == LOW ) {  //test if switch has been flipped, if so do next

servoDoor.write(posDoorUp);
delay(300);
servoSwitch.write(posSwitchUp);
delay(500);
servoSwitch.write(posSwitchDn);
delay(1000);
servoDoor.write(posDoorDn);

}
else {
  digitalWrite (ledPinGreen, HIGH);
  delay(500);
  digitalWrite (ledPinGreen, LOW);
  delay(500);
}
delay(500);
digitalWrite (ledPinRed, LOW); //Turn off LED to cause a Blinking action. if LED is on longer then off
delay(500);  //if LED stays on longer then off, it ran through code
}
```

REF:

https://www.arduino.cc/en/Reference/else
https://www.arduino.cc/en/Reference/for
https://www.arduino.cc/en/Reference/while
https://www.arduino.cc/en/Tutorial/BlinkWithoutDelay
https://www.arduino.cc/en/reference/random#.Uy8NPXW9gqn
https://www.arduino.cc/en/Tutorial/Sweep
https://www.arduino.cc/en/Reference/HomePage#.Uy8NOnW9gqn