MrBaddeley
Arduino, Servos and others. Basics…. instructions

Version 0.1 (Draft)

https://www.patreon.com/user?u=4294285
for other parts and instructions

Basics…

I’m not by any stretch of the imagination a programmer or electronics expert, but I wanted to put together a basic overview which has enabled me to add a few animations and servos to Dan’s Padawan 360 sketch. The internet is full of “how to” guides so I’m just going to talk about the basics and how to get started, along with my experiences. I am pretty much going to focus solely on Servos / LEDs controlled by the Adafruit 16 Channel Servo driver, because it’s what I used and it’s easy.

Anyhow… Firstly terminology and bits I used.

Arduino: A small computer which comes in many forms (size and capacity), basically it can have inputs and outputs. It’s made to read sensors and drive “stuff”.

Sketch: the name of the program you (or others) write for the Arduino.

Library: A pre written bit of Arduino programme which makes it easy to control hardware (like a PC driver kinda).

LEDs: small diodes which emit light ☺ They have a positive and a negative leg, don’t get them wrong. They also need a resistor otherwise you’ll burn them out. (google it).

Servos: Small, strong motors which have three connections, a “ground”, 5v (or more) and a “PWM (again google it)” basically a voltage which can range from 0 to 5 volts. The more the voltage, the further the servo turns, the less, it goes the other way.

Adafruit 16-Channel 12-bit PWM/Servo Driver - I2C interface A cool board which you can connect 16 servos or LEDS to and it controls them all via the Arduino. It only needs power and 2 wires to control all the servos. You can also chain them to control hundreds of Servos. I’ve used two in my R2, one in the dome for the Holoprojectors and one in the body for the panels etc.

Hooking it up..

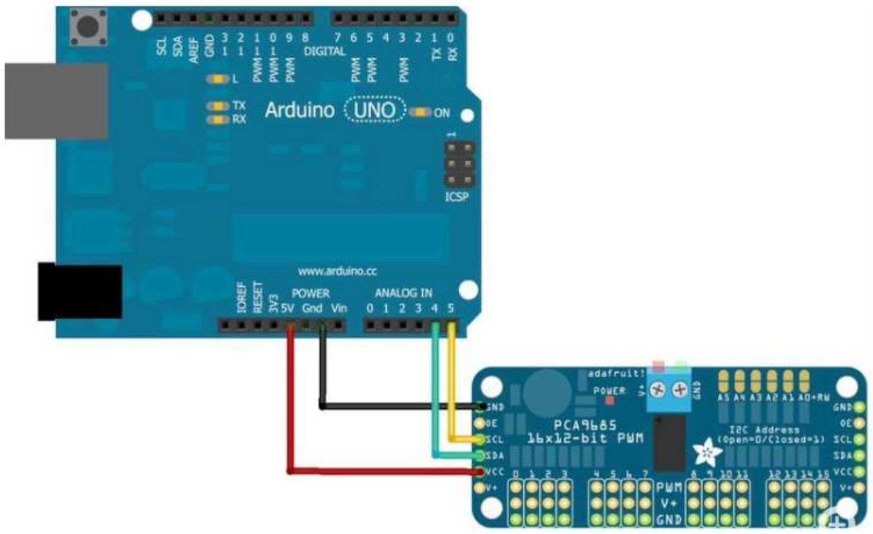
To play with a couple of servos and get used to it, you will need an Arduino (I used a mega, but an uno is equally good to test), a Adafruit Servo Board, a couple of servos (I like the SG90s, cheap from ebay or Amazon), a reasonable 5v power supply (2 or 3 amp is fine) and some jump cables.

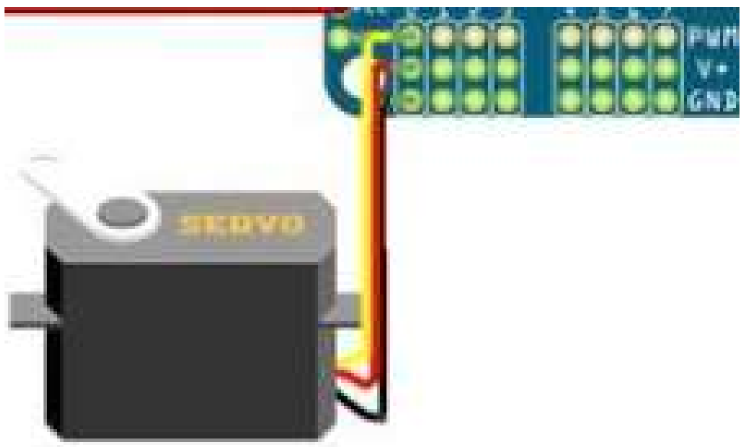
Here's a link to the Adafruit site which has a very easy tutorial showing how to solder it together and hook it up.

Basically I followed the site and hooked it up as shown. I aslo powered the Adafruit board from the same Power supply as the Arduino. Then I simply plugged a couple of servos into the Adafruit. The "Brown" or Black wire goes closest the edge of the board (GND).

So you've got three wires on the Servo, the GND (or negative), the V+ which I used 5v (red), these power the servo's movements.

The last Wire (yellow) is the signal wire, the Adafruit basically sends a voltage down this one which can range from 0 to 5 volts and depending on the voltage, depends on the angle the Servo turns to. In the software you send a number to the Servo and this dictates the angle.





The basic code…

It’s well worth learning the basics of the Adafruit code and how to control multiple board and servos. Once you get the basics you can add no end of toys and control to pretty much anything. As I said, there’s loads of coding help and best practice, so I’m just going to go through the basics I did.

The core code is shown across and it’s split into three sections, this is how the Arduino works, you have a first bit, then a setup and finally the loop keeps running. This is an excerpt from my code which opens and closes the gripper. As it starts, this would only vibrate your servo if you just run this as the open and close command would have no time to actually complete but I wanted to go through these bits and explain how I understand them.

The first two are the “#include” lines, these just load up the libraries (like drivers). Each only needs loading once even though you may have multiple servo driver boards.

Wire is the i2c driver, this is effectively how we serially communicate to the boards.

Adafruit_PWMServoDriver is obviously the driver for the board. This code sits at the beginning as we can’t do much without loading these. For how to load libraries and the Arduino software, loads of stuff on the net.

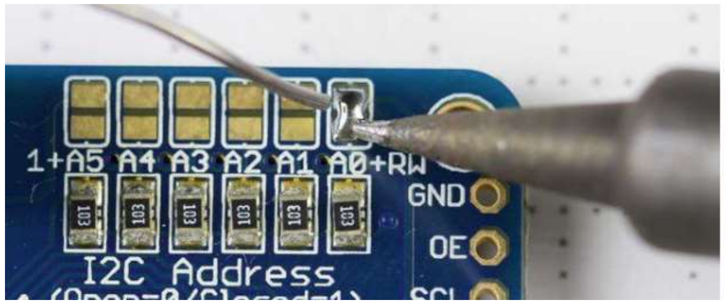
```
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>
Adafruit_PWMServoDriver pwm1 = Adafruit_PWMServoDriver(0x40);
Adafruit_PWMServoDriver pwm2 = Adafruit_PWMServoDriver(0x41);
int GripperOpen=270;
int GripperClose=325;
void setup()
 pwm1.begin();
  pwm1.setPWMFreq(60);  // Analog servos run at ~60 Hz updates
  pwm2.begin();
  pwm2.setPWMFreq(60);  // Analog servos run at ~60 Hz updates
 Wire.begin();
Void loop()
pwm2.setPWM(3,0,GripperClose);
 pwm2.setPWM(3,0,GripperOpen);
```

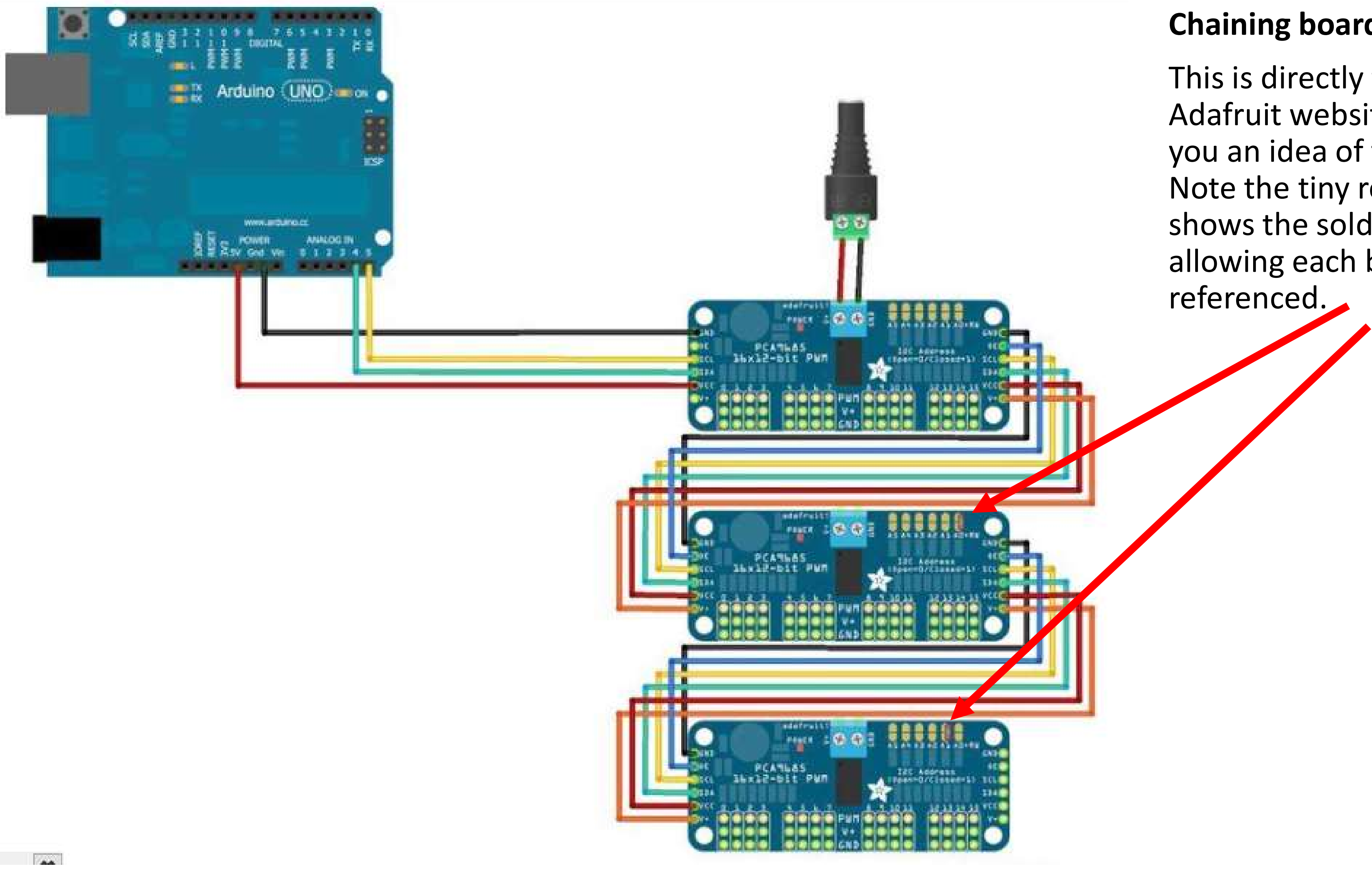
Setting them up

The next bit then sets up two “things”, pwm1 and pwm2. These are the reference for the two boards I have my R2. So the Adafruit_PWMServoDriver calls the library, pwm1 is our name for the device (you could call it Fred if you want) and the “=“ tells it the address. A new untouched board is 0x40. To add another onto the same bus you have to solder across two plates in the driver board (again documented on the website, pic below). Each plate is effectively binary so you can add loads on addressable boards. 0x41 is just the first one soldered across. So we now have two boards referenced by pwm1 and pwm2, each can have 16 Servos (or LEDs).

Next we set the variable for opening the Gripper and closing the Gripper. These are slightly difference per Servo so what I did was use a basic sketch (not unlike this one) and start at 300, reducing down till I got to the limit of the Servo, this gives me the lower limit, then the same going up. I then have two values for upper and lower. Generally it appears that these work across all servos of that model. On this sketch this isn’t the limits of the servo but the open and close position. We use variables so it’s meaningful in the code.

```
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>
Adafruit_PWMServoDriver pwm1 = Adafruit_PWMServoDriver(0x40);
Adafruit_PWMServoDriver pwm2 = Adafruit_PWMServoDriver(0x41);
int GripperOpen=270;
int GripperClose=325;
void setup()
 pwm1.begin();
  pwm1.setPWMFreq(60);  // Analog servos run at ~60 Hz updates
  pwm2.begin();
  pwm2.setPWMFreq(60);  // Analog servos run at ~60 Hz updates
 Wire.begin();
Void loop()
pwm2.setPWM(3,0,GripperClose);
 pwm2.setPWM(3,0,GripperOpen);
```





Chaining boards…

This is directly from the Adafruit website but gives you an idea of the chaining. Note the tiny red line which shows the solder "address" allowing each board to be referenced.

Setting them up

Next we move into the "Setup" bit, this fires everything up. Wire.begin starts the i2c (serial) communications, pwm1&2.begin fires up the two boards and pwm1&2. setPWMFreq(60) sets the frequency. Not exactly sure what this does (other than the obvious description) but it's always 60 and it works for both Servos and LEDs. After this we move into the "loop" where effectively the main program runs. This fires as a loop constantly after all the setup is done and is where the main body of the program runs.

It is really this simple. I single line of code sends the move request to the adafruit board and it does the rest! Pwm2 (or 1) references the Board. The setPWM sends the position, the first number (3) is the servo reference on the board, so this is pwm2-fourth Servo (as they're addressed 0,1,2,3). The next two number are "on and off" numbers. If you read the website it may make sense (doesn't really to me) but I simply use 0 for the first number and the position for the second, works perfectly so I keep it simple. The problem with this sketch alone is that it would try to move the servo back and forth with no time for the physical movement so the servo would just vibrate but it gives you the basics.

```
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>
Adafruit_PWMServoDriver pwm1 = Adafruit_PWMServoDriver(0x40);
Adafruit_PWMServoDriver pwm2 = Adafruit_PWMServoDriver(0x41);
int GripperOpen=270;
int GripperClose=325;
void setup()
 Wire.begin();
pwm1.begin();
  pwm1.setPWMFreq(60);  // Analog servos run at ~60 Hz updates
  pwm2.begin();
  pwm2.setPWMFreq(60);  // Analog servos run at ~60 Hz updates
Void loop()
pwm2.setPWM(3,0,GripperClose);
 pwm2.setPWM(3,0,GripperOpen);
```

Controlled “animations” or loops…

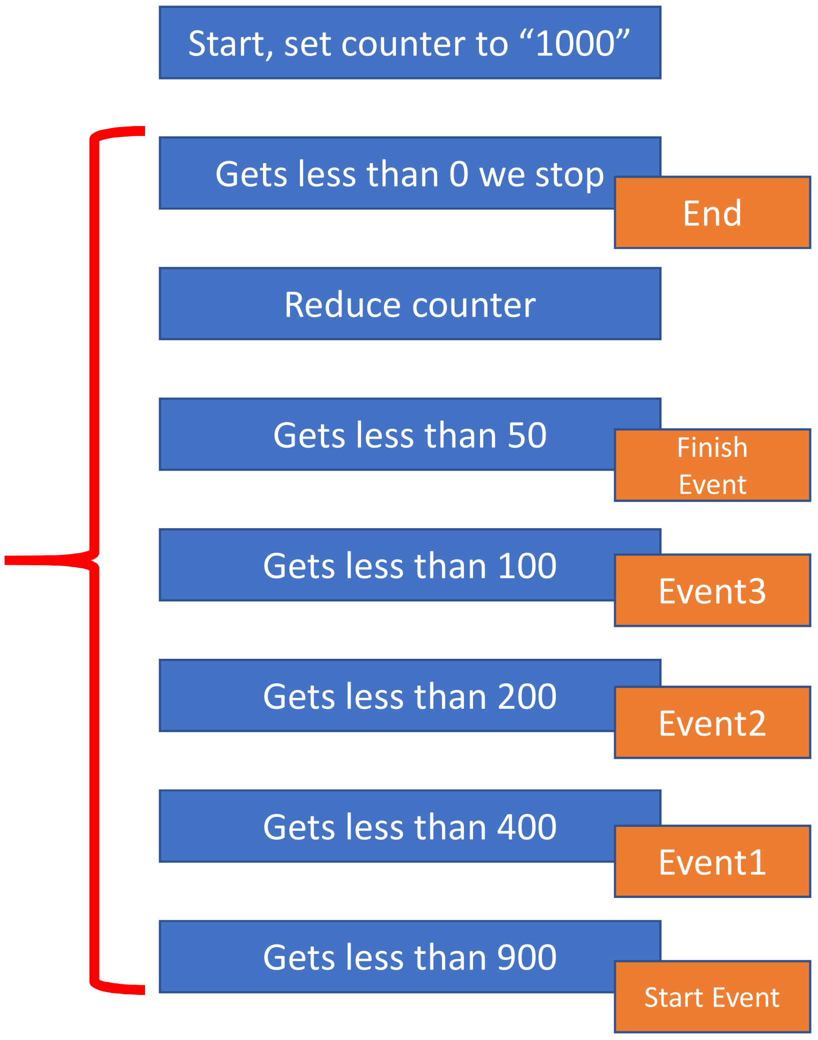
So now we’ve got the basics, let me share how I do animation loops (again I’m not a programmer but it will get you started).

This is how I do animations (for example, door opens, Arm pops out, gripper opens, it closes, opens, closes, door closes).

We have to do it this way for multi-tasking (so it doesn’t stop the rest of the programme.)

Basically when I want to start the animation, I simply set a counter to 1000 (obviously other numbers will give longer or shorter animation loops). It then counts down and as it hits certain values it creates an event (for example opening a door). So in the example shown, we set to 1000, when it gets less than 900 the start event (door opens for example) and through to 50 where we finish the event (door closes). In between we can push arms out, open grippers etc. etc. This works well and doesn’t take much processing time as when the counter is 0 it ignores the rest of the code.

Hope that makes sense, on the next page I’ll show you the code, it’s mainly uses If and Else statements make decisions. The reason we do the statement in reverse order is that once an event is triggered it jumps out of the routine, so less than 100 doesn’t trigger less that 200, 400 or 900.



Controlled “animations” or loops the code!

Here’s the code. The way it works is that if the Gripanicount is 0 then nothing happen, however if its over 0 then it checks the If / else if statements. The Gripanicount--; just reduces the count by 1 everytime the routine is ran.

When it gets below 950, but above 750 it opens the door (Servo 2). Then below 750 it pushes the arm out (Servo 1). At below 500 the Gripper is opened. Below 250 closes the gripper and pulls the Arm in. (Note you can do multiple actions within one else if statement.)

Finally below 10 the door closes and when it gets to 0 it ignores the routine.

Obviously you can have as many as these routines under different counters and use a trigger to set the “1000”. I’ll cover how we do this on the Padawan 360 sketch on the next page.

The Padawan used the XBOXRECV library to scan the control button on an xbox 360 controller. I’ll not cover any of that coding as it works and already in the sketch.

```
if(Gripanicount > 0) //Check to see if the animation loop has started
{
 Gripanicount --;
if(Gripanicount <10) //Return to home Finish event
{
 pwm1.setPWM(2,0,DoorClose);
  else if (Gripanicount <250) // Event 4
   {
 pwm1.setPWM(3,0,GripperClose);
pwm1.setPWM(1, 0, ArmIn);
   }
   else if (Gripanicount <500) //Event 3
   {
pwm1.setPWM(3,0,GripperOpen);
   }
   else if (Gripanicount <750) //Event 2
   {
 pwm1.setPWM(1, 0, ArmOut);
   }
   else if (Gripanicount <950) //Event 1
   {
   pwm1.setPWM(2,0,DoorOpen);
```

Combining with the Padawan X360

This is actually very easy, the call “Xbox.getButtonPress” checks the status of the buttons and the brackets define the button. In the example here, it check for the RIGHT button and then checks that L1 is pressed. This is to use combo keypresses to start events. Combos allow a lot more combinations than just a single key press. In this example if Right and L1 is pressed it starts the Gripper Animation.

Hopefully this gives you a start on using Arduinos and Servos. Note LED can also be connected to the Adafruit Servo board, Grd and PWM with the PWM being + and Grd being – Then you send the signal using 0 – 4095 for brightness. With this board you do not need any resistors for LED as they’re built it. This lets you use a combination of Servos and LEDS ran by the same Adafruit board. (As I did for the Holoprojectors.

For connections, I connected the Arduino mega (my main Arduino) to the Body Adafruit, then the body adafruit to the dome adafruit using a 6 wire slipring. I run Grd, 5v and the two signal wires. If you’re using 5v to run the servos, this can also be used to power the Adafruit board so you can get away with 4 wires. Play and test, the best way to learn!

```
if (Xbox.getButtonPress(RIGHT, 0))
{

          if (Xbox.getButtonPress(L1, 0)) {

           Gripanicount=1000;

            }
}
```

Supported and tested by Sean Lavigne, Jay Williams, Steven Elford, Robert Gusek, Rob Dinniwell, Joseph Masci, Sam D. Fenimore, LarryJ, tevans, Rick Davis, Brendan Faulkner, Nicolas Carré, Ben Langley, Mathieu Saint-marc, Chistopher Edwards, Mark Oram, Tim Parr, Jon Haag, John Gardener, Ryan Roehitch, Oiva Ranta, Wes Thierry, Robert Bean, Mitchell Young, Jake Danible, Simon Ruel, William Meyer, Brian Bishop, Danny Olsson and Brian.