Controlling a Servo with a Potentiometer on Arduino

Summary

This document outlines some of the logic for how to implement servo control with a potentiometer on an Arduino. This does not make use of the servo library. This document starts very basic, and then quickly gets the meat of servo control.

Reading Potentiometer on Arduino

Set up the basic Arduino program:

```cpp
void setup() {
  Serial.begin(9600); // set up the serial connection for printing
}

void loop() {
  Serial.println(444); // print out the value 444
}
```

Open the Arduino serial monitor to see the output from the board. When board is powered and connected to the computer.

Place the potentiometer on the breadboard. We want a variable value, not binary, so we’ll read values on the Arduino from the analog input pins. Hook up a wire from the center leg of the potentiometer to analog pin 0. In the program we’ll represent this like:

```cpp
int ANALOGINPUTPIN = 0;
```

Power the potentiometer from the 5V pin on the Arduino, this will be the red column on the bread board, and hook up the ground, the green column.
Make a variable to hold output value:

```cpp
int analogValue = 0;
```

Print the new analog value with:

```cpp
Serial.println(analogValue);
```

To read value coming from pot, use function `analogRead()`. In the loop, read in the value with:

```cpp
analogValue = analogRead(ANALOGINPUTPIN);  // read the analog input
```

and print with:

```cpp
Serial.println(analogValue);
```

Sketch should look like this:

```cpp
int ANALOGINPUTPIN = 0;
int analogValue = 0;

void setup() {
  Serial.begin(9600);  // set up the serial connection for printing
}

void loop() {
  analogValue = analogRead(ANALOGINPUTPIN);
  Serial.println(analogValue);  // print out the value 444
}
```

**Functions Introduced:**


Controlling a Servo with Arduino

Servo has three wires, one ground, one positive power, and one data. Data is usually yellow. Red usually positive, and black is ground. Communicate over a digital pin, like 2. Attach 5V(red) and ground(green or black) wires. Servo need high voltage on the data for between 500 and 2500 microseconds in a 50 Hz cycle, ideally between 1milli and 2milli.

Add pin and timings in code:

```cpp
int SERVOPULSEPIN = 2; // Control pin for servo motor, usually yellow wire to the servo
int minPulseTime = 500; // minimum servo pulse time
int maxPulseTime = 2400; // maximum servo pulse time
int pulseTime = 0; // Amount of time to pulse the servo
```

To read from the digital pin, it needs to configured for output in setup:

```cpp
pinMode(SERVOPULSEPIN, OUTPUT); // Set servo pin as an output pin
```

Convert the analogValue to a one between min and max using the map function:

```cpp
pulseTime = map(analogValue,0,1023,minPulseTime,maxPulseTime); // convert the analog value
```

Write this out to Serial to check the values:

```cpp
Serial.println(pulseTime);
```

Only pulse once per 20 ms interval, so add two more variables to control timing:

```cpp
long lastPulse = 0; // the time in milliseconds of the last pulse
int refreshTime = 20; // the time needed in between pulses in milliseconds
```
The function millis() can be used to get the time sketch has been running, so the loop can be structured like this:

```c
if (millis() - lastPulse >= refreshTime) {
}
```

The digital pins can be instructed to output voltage or not using digitalWrite() with pin and value HIGH or LOW. Use delayMicroSeconds() with pulseTime for sending pulse, and then reset the lastPulse:

```c
if (millis() - lastPulse >= refreshTime) {
    digitalWrite(SERVOPULSEPIN, HIGH);       // turn on pulse to the servo
    delayMicroseconds(pulseTime);                  // length of the pulse sets the motor position
    digitalWrite(SERVOPULSEPIN, LOW);            // stop pulse to the servo
    lastPulse = millis();                        // save the time of the last pulse
}
```

The final program looks like so:

```c
int SERVOPULSEPIN = 2;     // Control pin for servo motor, usually yellow wire to the servo
int minPulseTime = 500;        // minimum servo pulse time
int maxPulseTime = 2400;       // maximum servo pulse time
int pulseTime = 0;             // Amount of time to pulse the servo
long lastPulse = 0;        // the time in milliseconds of the last pulse
int refreshTime = 20;      // the time needed in between pulses in milliseconds
                          // servos have 50Hz rate, 1sec/50 = 20 milliseconds
int analogValue = 0;       // the value returned from the analog sensor, between 0 and 1023
int ANALOGINPUTPIN = 0;    // the analog pin that the sensor is on

void setup() {
    pinMode(SERVOPULSEPIN, OUTPUT);  // Set servo pin as an output pin
    pulseTime = minPulseTime;        // Set the motor position value to the minimum
    Serial.begin(9600);              // Set up the serial connection for printing
}

void loop() {

    analogValue = analogRead(ANALOGINPUTPIN);                  // read the analog input
    pulseTime = map(analogValue, 0, 1023, minPulseTime, maxPulseTime); // convert the analog value
                          // to a range between minPulse
                          // and maxPulse.

    // pulse the servo again if the refresh time (20 ms) have passed:
    if (millis() - lastPulse >= refreshTime) {
        digitalWrite(SERVOPULSEPIN, HIGH);       // turn on pulse to the servo
        delayMicroseconds(pulseTime);                  // length of the pulse sets the motor position
        digitalWrite(SERVOPULSEPIN, LOW);            // stop pulse to the servo
        lastPulse = millis();                        // save the time of the last pulse
    }
    Serial.println(analogValue);
    Serial.println(pulseTime);
}
```

Functions Introduced:

