```
1: /*
2:  * Entry Sensor Light: with Sharp Distance Sensor and PWM Output
3:  * - Use a distance sensor (Sharp# GP2Y0A710K0F) to ramp (fade) a PWM output
4:  *    up when the sensed distance changes. Then after a time out, ramp the PWM
5:  *    output down at a (potentially) different rate.
6:  *
7:  * By Chris "Sembazuru" Elliott, SembazuruCDE (at) GMail.com
8:  * 2013/07/19
9:  *
10:  * To the extent possible under law, Chris "Sembazuru" Elliott has waived all
11:  *  copyright and related or neighboring rights to SharpDistanceSensorPWMout.ino
12:  *  See http://creativecommons.org/publicdomain/zero/1.0/ for details.
13:  *  (Attribution would be nice, but in no way compulsory.)
14:  *
15:  * Hardware configuration on Arduino UNO:
16:  * - Analog output of sensor to A0
17:  * - LED13 used to show trigger status
18:  * - PWM on pin 3 used for the ramping output (pin 4 used as ground reference for
19:  *    testing with 0.1" 5V LED)
20:  * - A5 pin used to disable serial diagnostics (A4 pin used as ground reference
21:  *    for testing with 0.1" shorting pins)
22:  */
23:
24: // Template version. See http://playground.arduino.cc/Main/runningMedian for details and downloads.
25: #include <RunningMedian.h>
26:
27: // Pair of pins to enable serial diagnostics. Short these two pins (or just the diagnosticsPin to ground) to
    silence serial diagnostics.
28: const byte diagnosticsPin = A5;
29: const byte diagnosticsGnd = A4;
30:
31: // Constants and variables for data collection.
32: const byte sensorPin = A0;
33: unsigned int currentADC;
34: unsigned int currentMedian;
35: const unsigned int medianArraySize = 16; // number of data points to collect to calculate the median
36: const unsigned long readingInterval = 17;
37: unsigned long readingStart = millis();
38: RunningMedian<unsigned int, medianArraySize> myMedian;
39:
40: // Constants and variables for triggering.
41: unsigned int lastMedian;
42: const unsigned int triggerThreshold = 7; // +/- value for medians
43: const byte triggerLED = 13; // Indicator LED for trigger
44: const unsigned long triggerDelay = 10000; // How long the trigger will last (minimum) in milliseconds
45: unsigned long triggerStart; // To allow capturing the time at the start of a trigger
46: boolean triggered = false;
47:
48: // Constants and variables for ramping the output up and down.
49: boolean ramping = false; // true == need to ramp, false == done ramping.
50: boolean rampDir = false; // true == up, false == down.
51: const byte rampPin = 3; // PWM pin to ramp up and down.
52: const byte rampGnd = 4; // Adjacent ground provided for use of 0.1" spacing 5V LED.
53: const int rampMin = 0;
54: const int rampMax = 255;
55: const byte rampStep = 4; // How much to change the rampValue by each iteration.
56: const byte rampUDRatio = 2; // Amount to divide the rampStep by when ramping down to change the ramp up to
    ramp down rate ratio.
57: const unsigned long rampTime = 3000; // How many milliseconds (base) to take for the ramp up.
58: const unsigned long rampStepInterval = rampTime / ((rampMax - rampMin) / rampStep); // Calculate how long
    between ramp steps. Do this calculation once here.
59: unsigned long rampStepStart = millis(); // Sensor readings take 16.5ms +/- 3.7ms.
60: int rampValue = rampMin;
61:
62: void setup()
63: {
64:   Serial.begin(115200);
65:   while (!Serial); // Wait for serial port to connect. Needed for Leonardo only.
```

```
66:   delay(1000); // Simply to allow time for the ERW versions of the IDE time to automagically open the Serial
    Monitor. 1 second chosen arbitrarily.
67:
68:   // Setup pins for enabling/disabling serial diagnostics.
69:   pinMode(diagnosticsPin, INPUT_PULLUP);
70:   pinMode(diagnosticsGnd, OUTPUT);
71:   digitalWrite(diagnosticsGnd, LOW);
72:
73:   // Setup pins for using the onboard LED to indicate triggering.
74:   pinMode(triggerLED, OUTPUT);
75:   digitalWrite(triggerLED, LOW);
76:
77:   // Setup pins for the ramping (fading) PWM pin.
78:   analogWrite(rampPin, rampValue);
79:   pinMode(rampGnd, OUTPUT);
80:   digitalWrite(rampGnd, LOW);
81: }
82:
83: void loop()
84: {
85:   if ((millis() - readingStart) > readingInterval)
86:   {
87:     getData();
88:     triggerCheck();
89:     if (digitalRead(diagnosticsPin))
90:     {
91:       diagnostics(readingStart);
92:     }
93:   }
94:   if (ramping)
95:   {
96:     if ((millis() - rampStepStart) > rampStepInterval)
97:     {
98:       rampValue = rampOutput(rampValue, rampStep, rampDir);
99:       if (digitalRead(diagnosticsPin))
100:      {
101:        diagnostics(rampStepStart);
102:      }
103:    }
104:  }
105: }
106:
107: void getData()
108: {
109:   readingStart = millis();
110:   currentADC = analogRead(sensorPin);
111:   myMedian.add(currentADC);
112:   lastMedian = currentMedian;
113:   myMedian.getMedian(currentMedian);
114: }
115:
116: void triggerCheck()
117: {
118:   // Trigger if the median drops by more than the threshold or raises by more than the threshold.
119:   if ((currentMedian < (lastMedian - triggerThreshold)) || (currentMedian > (lastMedian + triggerThreshold)))
120:   {
121:     triggerStart = millis();
122:     digitalWrite(triggerLED, HIGH);
123:     triggered = true;
124:     ramping = true;
125:     rampDir = true;
126:   }
127:   // If currently triggered, check to see if enough time has passed since the last trigger event to turn off
    the trigger.
128:   if (triggered)
129:   {
130:     if ((millis() - triggerStart) > triggerDelay)
131:     {
```

```
132:       digitalWrite(triggerLED, LOW);
133:       triggered = false;
134:       ramping = true;
135:       rampDir = false;
136:     }
137:   }
138: }
139:
140: int rampOutput(int _value, int _step, boolean _dir)
141: {
142:   rampStepStart = millis();
143:   if (!_dir)
144:   {
145:     _step = -_step / rampUDRatio;
146:   }
147:   _value += _step;
148:   if (_value < rampMin)
149:   {
150:     _value = rampMin;
151:     ramping = false;
152:   }
153:   if (_value > rampMax)
154:   {
155:     _value = rampMax;
156:     ramping = false;
157:   }
158:   analogWrite(rampPin, _value);
159:   return _value;
160: }
161:
162: void diagnostics(unsigned long _time)
163: {
164:   // Use the F() macro throughout to not waste valueable SRAM on diagnostic messages.
165:   Serial.print(_time);
166:   Serial.print(F(" :"));
167:   Serial.print(F(" Array Size = "));
168:   Serial.print(myMedian.getCount());
169:   Serial.print(F(" Current = "));
170:   Serial.print(currentADC);
171:   Serial.print(F(" "));
172:   Serial.print(F(" Current Median = "));
173:   Serial.print(currentMedian);
174:   Serial.print(F(" Last Median = "));
175:   Serial.print(lastMedian);
176:   Serial.print(F(" Median Delta = "));
177:   Serial.print(((int) currentMedian) - ((int) lastMedian)); // Casting to signed values to allow a negative
    result.
178:   Serial.print(F(" Triggered = "));
179:   Serial.print(triggered);
180:   Serial.print(F(" Ramping = "));
181:   Serial.print(ramping);
182:   Serial.print(F(" Ramp Direction = "));
183:   if (rampDir)
184:   {
185:     Serial.print(F(" UP "));
186:   }
187:   else
188:   {
189:     Serial.print(F("DOWN"));
190:   }
191:   Serial.print(F(" Ramp Value = "));
192:   Serial.print(rampValue);
193:   Serial.println();
194: }
195:
196:
```