```cpp
/*
Name:       RadioReciever.ino
Created:    12/22/2017 9:51 PM
Author: Michael Langford
*/

#include "RadioReciever.h"

//pulse widths
volatile int time[CHANNELS];

//current offset into array, used by interrupt method
volatile int offset = 0;

//percentage point output
float percents[CHANNELS];

//last micros() read by interrupt method
volatile int last_time;

//smoothing banks
float a[CHANNELS];
float b[CHANNELS];
float c[CHANNELS];
float d[CHANNELS];
float e[CHANNELS];
float f[CHANNELS];
float g[CHANNELS];

//initialization code
void init_RadioReciever() {
    pinMode(RADIO_RX, INPUT);

    int delta = 0;

    while (true)
    {
        delta = micros();
        wait_for_pulse();
        delta = micros() - delta;
        if (delta > TIME_MINIMUM)
            break;
    }

    attachInterrupt(digitalPinToInterrupt(RADIO_RX), interrupt_method, RISING);

}

//returns percentage from -100.0f to 100.0f for specified channel
float GetChannel(int channel)
{
    return percents[channel];
```

```
53  }
54
55  //update reciever
56  void update_RadioReciever() {
57      for (int i = 0; i < CHANNELS; i++)
58      {
59          //this strategy smooths out integer changes in the channels values. These changes,
60          //if left unfiltered, can jack up the derivative part of the PID loop
61          percents[i] += (get_percent(time[i]) - percents[i])*RADIO_DELTA_SPEED;
62      }
63  }
64
65  //wait for a single pulse to occur
66  void wait_for_pulse() {
67      while (true)
68      {
69          if (digitalRead(RADIO_RX) == HIGH) break;
70      }
71      while (true)
72      {
73          if (digitalRead(RADIO_RX) == LOW) break;
74      }
75  }
76
77  //convert pulse width into a percentage value
78  float get_percent(int pulse)
79  {
80      pulse -= CENTER_PW;
81      float new_percent = (float)pulse / DELTA_PW;
82      new_percent = max(-100.0, new_percent);
83      new_percent = min(100.0, new_percent);
84
85      return new_percent;
86  }
87
88  //interrupt method, called each time a pulse occurs
89  void interrupt_method()
90  {
91      time[offset] = micros() - last_time;
92      last_time = micros();
93
94      offset++;
95
96      //roll over
97      if (offset == CHANNELS)
98          offset = 0;
99  }
```