```
/*
 * ////////////////////////////////////////////////
 * //making sense of the Parallax PIR sensor's output
 * ////////////////////////////////////////////////
 *
 * Switches a LED according to the state of the sensors output pin.
 * Determines the beginning and end of continuous motion sequences.
 *
 * @author: Kristian Gohlke / krigoo (_) gmail (_) com / http://krx.at
 * @date:   3. September 2006
 *
 * kr1 (cleft) 2006
 * released under a creative commons "Attribution-NonCommercial-ShareAlike 2.0"
license
 * http://creativecommons.org/licenses/by-nc-sa/2.0/de/
 *
 *
 * The Parallax PIR Sensor is an easy to use digital infrared motion sensor module.
 * (http://www.parallax.com/detail.asp?product_id=555-28027)
 *
 * The sensor's output pin goes to HIGH if motion is present.
 * However, even if motion is present it goes to LOW from time to time,
 * which might give the impression no motion is present.
 * This program deals with this issue by ignoring LOW-phases shorter than a given
time,
 * assuming continuous motion is present during these phases.
 *
 */

////////////////////////////
//VARS
//the time we give the sensor to calibrate (10-60 secs according to the datasheet)
int calibrationTime = 10;

//the time when the sensor outputs a low impulse
long unsigned int lowIn;

//the amount of milliseconds the sensor has to be low
//before we assume all motion has stopped
long unsigned int pause = 5000;

boolean lockLow = true;
boolean takeLowTime;

int pirFwd = 8;    //the digital pin connected to the PIR sensor's output
int pirBwd = 11;
int pirLft = 9;
int pirRgt = 10;
int ledPin = 13;
```

```
/* Connect the wires from the RC remote to the following pins */

const int fwdPin  = 2; // Forward button
const int bckPin  = 3; // Back button
const int lftPin  = 5; // Left button
const int rgtPin  = 4; // Right button

/* Each direction gets a number */
const int NEUTRAL = 0;
const int FORWARD = 1;
const int REVERSE = 2;
const int LEFT    = 3;
const int RIGHT   = 4;


////////////////////////////
//SETUP
void setup(){
  Serial.begin(9600);
  pinMode(pirFwd, INPUT);
  pinMode(pirBwd, INPUT);
  pinMode(pirLft, INPUT);
  pinMode(pirRgt, INPUT);
  pinMode(ledPin, OUTPUT);

  pinMode( fwdPin, OUTPUT );
  pinMode( bckPin, OUTPUT );
  pinMode( lftPin, OUTPUT );
  pinMode( rgtPin, OUTPUT );

  digitalWrite( fwdPin, HIGH );
  digitalWrite( bckPin, HIGH );
  digitalWrite( lftPin, HIGH );
  digitalWrite( rgtPin, HIGH );

/*
  digitalWrite(pirFwd, LOW);
  digitalWrite(pirBwd, LOW);
  digitalWrite(pirLft, LOW);
  digitalWrite(pirRgt, LOW);
*/

  //give the sensor some time to calibrate
  Serial.print("calibrating sensor ");
    for(int i = 0; i < calibrationTime; i++){
      Serial.print(".");
      delay(500);
      }
    Serial.println(" done");
    Serial.println("SENSOR ACTIVE");
```

```
    delay(50);

  Serial.begin(9600);

  }

////////////////////////////
//LOOP
void loop(){

  if(digitalRead(pirFwd) == HIGH){
    Serial.println("Motion forward detected");
    digitalWrite(ledPin, HIGH);   //the led visualizes the sensors output pin state
    forward(1000);
    delay(2000);
  }

  if(digitalRead(pirBwd) == HIGH){
    Serial.println("Motion backward detected");
    digitalWrite(ledPin, HIGH);   //the led visualizes the sensors output pin state
    back(1000);
    delay(2000);
  }

  if(digitalRead(pirLft) == HIGH){
    Serial.println("Motion left detected");
    digitalWrite(ledPin, HIGH);   //the led visualizes the sensors output pin state
    forward_left(500);
    forward(1000);
    delay(2000);
  }

  if(digitalRead(pirRgt) == HIGH){
    Serial.println("Motion right detected");
    digitalWrite(ledPin, HIGH);   //the led visualizes the sensors output pin state
    forward_right(500);
    forward(1000);
    delay(2000);
  }

}

 void forward( int duration ){
  gas( FORWARD );
  steer( NEUTRAL );
  delay( duration );
  stop();
}
```

```
void forward_left( int duration ){
  gas( FORWARD );
  steer( LEFT );
  delay( duration );
  stop();
}


void forward_right( int duration ){
  gas( FORWARD );
  steer( RIGHT );
  delay( duration );
  stop();
}


void back( int duration ){
  gas( REVERSE );
  steer( NEUTRAL );
  delay( duration );
  stop();
}


void back_left( int duration ){
  gas( REVERSE );
  steer( LEFT );
  delay( duration );
  stop();
}


void back_right( int duration ){
  gas( REVERSE );
  steer( RIGHT );
  delay( duration );
  stop();
}

/**
* Push the forward or back button
* @param state (int) Direction, use direction constant
**/
void gas( int state ){
  if( state == FORWARD ){
    switchPins( fwdPin, bckPin, fwdPin );
  } else if( state == REVERSE ){
    switchPins( fwdPin, bckPin, bckPin );
  } else {
    switchPins( fwdPin, bckPin, 0 );
```

```
  }
  delay( 5 );
}

/**
* Push the left or right button
* @param state (int) Direction to steer, use direction constant
**/
void steer( int state ){
  if( state == LEFT ){
    switchPins( lftPin, rgtPin, lftPin );
  } else if( state == RIGHT ){
    switchPins( lftPin, rgtPin, rgtPin );
  } else {
    switchPins( lftPin, rgtPin, 0 );
  }
  delay( 5 );
}

/**
* Unpress all buttons
**/
void stop(){
  gas( NEUTRAL );
  steer( NEUTRAL );
}

/**
* We always exclusively use either forward OR back or left OR right,
* so this function toggles one to LOW (pushes the button) while the other
* or both (when 'which' does not match any pin) get set to an
* INPUT (button not pressed / high impedance).
* @param pin1 (int) First pin (use pin constant)
* @param pin2 (int) Second pin (use pin constant)
* @param which (int) If this matches pin1 or pin2 that pin (IE button) will be toggled
low.
        if there is no match both pins will be inputs and no button will be pressed
*/
void switchPins( int pin1, int pin2, int which ){
  if( which == pin1 ){
    pinMode( pin1, OUTPUT );
    digitalWrite( pin1, LOW );
    pinMode( pin2, INPUT );
  } else if ( which == pin2 ) {
    pinMode( pin1, INPUT );
    pinMode( pin2, OUTPUT );
    digitalWrite( pin1, LOW );
  } else {
    pinMode( pin1, INPUT );
    pinMode( pin2, INPUT );
```

```
  }
}
```