Programm 2

**Another example**

This function will read a sensor five times with analogRead() and calculate the average of five readings. It then scales the

data to 8 bits (0-255), and inverts it, returning the inverted result.

int ReadSens\_and\_Condition(){

int i;

int sval;

for (i = 0; i < 5; i++){

sval = sval + analogRead(0); // sensor on analog pin 0

}

sval = sval / 5; // average

sval = sval / 4; // scale to 8 bits (0 - 255)

sval = 255 - sval; // invert output

return

**Another example**

FOR

// Dim an LED using a PWM pin

int PWMpin = 10; // LED in series with 1k resistor on pin 10

void setup()

{

// no setup needed

}

void loop()

{

for (int i=0; i <= 255; i++){

analogWrite(PWMpin, i);

delay(10);

}

}

3

for (x = 0; x < 255; x ++)

{

digitalWrite(PWMpin, x);

sens = analogRead(sensorPin);

if (sens > threshold){ // bail out on sensor detect

x = 0;

break;

}

delay(50);

}

Reference

4

A function to compare a sensor input to a threshold

int checkSensor(){

if (analogRead(0) > 400) {

return 1;

else{

return 0;

}

}

The return keyword

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

RedBot Library\_Test

Created 30 Jul 2013 by Mike Hord @ SparkFun Electronics.

This code is beerware- feel free to make use of it, with or without

attribution, in your own projects. If you find it helpful, buy me a beer

the next time you see me at the local- or better yet, shop SparkFun!

This is a simple hardware/library use demo for the RedBot. All it does

is print accelerometer and sensor data over the serial port, while

waiting for a tap on the accelerometer. When the accelerometer is tapped,

it drives forward a few inches.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Include the libraries. We make a provision for using the Xbee header

// via software serial to report values, but that's not really used in

// the code anywhere.

#include <RedBot.h>

#include <SoftwareSerial.h>

// Instantiate the accelerometer. It can only be connected to pins A4

// and A5, since those are the I2C pins for this board.

RedBotAccel xl;

// Instantiate the motor control class. This only needs to be done once

// and indeed SHOULD only be done once!

RedBotMotor motor;

// Instantiate the sensors. Sensors can only be created for analog input

// pins; the Xbee software serial uses pins A0 and A1 and the

// accelerometer uses pins A4 and A5.

RedBotSensor lSen = RedBotSensor(2);

RedBotSensor cSen = RedBotSensor(3);

RedBotSensor rSen = RedBotSensor(6);

// Create a software serial connection. See the Arduino documentation

// for more information about this. The pins used here are the hard

// wired pins the Xbee header connects to.

SoftwareSerial xbee(15, 14);

void setup()

{

Serial.begin(57600);

xbee.begin(57600);

Serial.println("Hello world!");

xbee.println("Hello world!");

// Enable bump detection. Once a bump occurs, xl.checkBump() can be

// used to detect it. We'll use that to start moving.

xl.enableBump();

}

void loop()

{

// xl.read() causes the accelerometer values to be read into the

// xl.x, xl.y, and xl.z variables for easy access.

xl.read();

xbee.print("X: "); xbee.println(xl.x);

xbee.print("Y: "); xbee.println(xl.y);

xbee.print("Z: "); xbee.println(xl.z);

// sensor.read() returns the current value of the analog sensor.

xbee.print("L sen: "); xbee.println(lSen.read());

xbee.print("C sen: "); xbee.println(cSen.read());

xbee.print("R sen: "); xbee.println(rSen.read());

// checkBump() looks for a tap input. Tap input events are stored,

// and cleared on checkBump(). You may want to call checkBump()

// before using it to make sure you only detect events that occur

// AFTER it is called!

if (xl.checkBump())

{

motor.drive(255); // drive a bit

delay(500); // wait a bit

motor.brake(); // stop

delay(200); // wait for stop to finish

xl.checkBump();

}

delay(500);

}