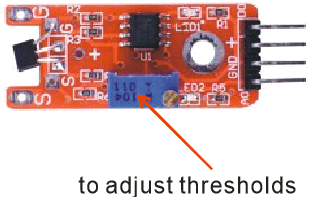
Linear Hall Sensor



Overview

Hall effects are magnetic sensors, and vary their voltage output in relation to a detected magnetic field. They are used to detect proximity, position, speed, and current. Analog Hall sensors report a signal proportional to the magnetic field strength (an analog quantity) rather than act as a discrete (digital) switch indicating a magnet’s “presence” or “absence.” The module used in this experiment can provide both an analog reading of proximity and a digital “switch” value determining whether the reading exceeds some proximity threshold (set by the blue onboard potentiometer). This experiment uses the Raspberry Pi to measure both signals of the linear Hall sensor, and drives a blinking LED based on whether the captured analog signal exceeds a software-specified threshold. (Thus the sample code recreates, in software, the same decision logic mapping a sensed analog value to switched digital effect that the onboard comparator and potentiometer accomplish in hardware.)

Experimental Materials

Raspberry Pi x1

Breadboard x1

Linear Hall sensor x1

ADC0832 x1

LED (3 pin) x1

Dupont jumper wires

Any magnet (you provide)

Experimental Procedure

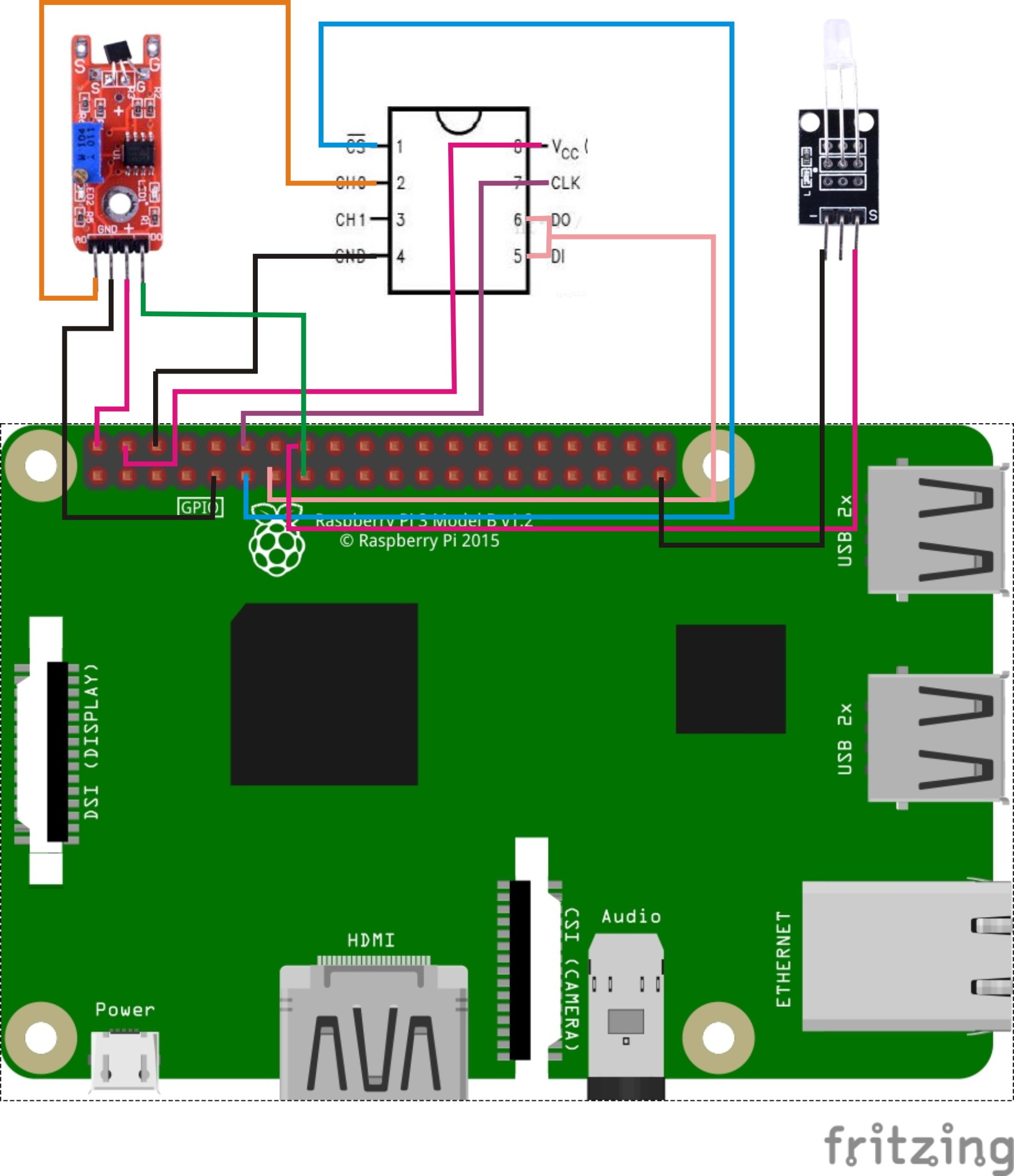
1. If you have not done so already, prepare your development system by installing the Python interpreter, RPi.GIO library, and wiringPi library as described in READ\_ME.TXT.
2. Install the ADC0832 analog/digital converter IC, linear Hall effect sensor, and three-pin LED on your breadboard, and use Dupont jumper wires to connect them to each other and your Raspberry Pi as illustrated in the Wiring Diagram below. (The three-pin LED provided in this kit includes onboard series resistors, so no additional resistors are needed.)
3. Execute the sample stored in this experiment’s subfolder.

If using C, compile and execute the C code:  
cd Code/C  
gcc linearHall.c -o linearHall.out –lwiringPi  
./linearHall.out

If using Python, launch the Python script:  
cd Code/Python  
python linearHall.py

1. Make experimental observations.  
   When you hold your magnet vertically close to the sensor, the Hall effect generates an (analog) voltage, which the ADC converts to a (digital) signal readable by the Raspberry Pi. The sample code then turns on the LED if that voltage exceeds a certain threshold.

Wiring Diagram



AD0382 pin position:

CS ↔ Raspberry Pi Pin 11

CLK ↔ Raspberry Pi Pin 12

DI ↔ Raspberry Pi Pin 13

D0 ↔ Raspberry Pi Pin 13

CH0 ↔ Linear Hall Sensor Pin A0

VCC ↔ Raspberry Pi +5V

GND ↔ Raspberry Pi GND

Linear Hall pin position:

A0 ↔ ADC0382 Pin CH0

D0 ↔ Raspberry Pi Pin 15

GND ↔ Raspberry Pi GND

"+" ↔ Raspberry Pi +5V

LED pin position:

"S" ↔ Raspberry Pi Pin 16

"-" ↔ Raspberry Pi GND

Sample Code  
(These are listings of the files in this experiment’s Code subfolder.)

Python Code

#!/usr/bin/env python

import RPi.GPIO as GPIO

import ADC0832

import time

Hall\_DO\_PIN = 15

LedPin = 16

thresholdVal = 100

def init():

GPIO.setmode(GPIO.BOARD)

GPIO.setup(Hall\_DO\_PIN,GPIO.IN, pull\_up\_down=GPIO.PUD\_UP)

GPIO.setup(LedPin, GPIO.OUT)

ADC0832.setup()

def loop():

while True:

global digitalVal

digitalVal = GPIO.input(Hall\_DO\_PIN)

if(digitalVal == 0):

print 'DO is %d' % digitalVal

analogVal = ADC0832.getResult(0)

print 'Current analog value is %d'% analogVal

if(analogVal > thresholdVal):

GPIO.output(LedPin, GPIO.HIGH)

time.sleep(0.2)

else:

GPIO.output(LedPin, GPIO.LOW)

if \_\_name\_\_ == '\_\_main\_\_':

init()

try:

loop()

except KeyboardInterrupt:

ADC0832.destroy()

print 'The end !'

C Code

#include <wiringPi.h>

#include <stdio.h>

#include <string.h>

#include <errno.h>

#include <stdlib.h>

#define ADC\_CS 0

#define ADC\_CLK 1

#define ADC\_DIO 2

#define Hall\_DO\_Pin 3

#define LedPin 4

#define thresholdVal 100

typedef unsigned char uchar;

typedef unsigned int uint;

uchar get\_ADC\_Result(void)

{

uchar i;

uchar dat1=0, dat2=0;

digitalWrite(ADC\_CS, 0);

digitalWrite(ADC\_CLK,0);

digitalWrite(ADC\_DIO,1); delayMicroseconds(2);

digitalWrite(ADC\_CLK,1); delayMicroseconds(2);

digitalWrite(ADC\_CLK,0);

digitalWrite(ADC\_DIO,1); delayMicroseconds(2);

digitalWrite(ADC\_CLK,1); delayMicroseconds(2);

digitalWrite(ADC\_CLK,0);

digitalWrite(ADC\_DIO,0); delayMicroseconds(2);

digitalWrite(ADC\_CLK,1);

digitalWrite(ADC\_DIO,1); delayMicroseconds(2);

digitalWrite(ADC\_CLK,0);

digitalWrite(ADC\_DIO,1); delayMicroseconds(2);

for(i=0;i<8;i++)

{

digitalWrite(ADC\_CLK,1); delayMicroseconds(2);

digitalWrite(ADC\_CLK,0); delayMicroseconds(2);

pinMode(ADC\_DIO, INPUT);

dat1=dat1<<1 | digitalRead(ADC\_DIO);

}

for(i=0;i<8;i++)

{

dat2 = dat2 | ((uchar)(digitalRead(ADC\_DIO))<<i);

digitalWrite(ADC\_CLK,1); delayMicroseconds(2);

digitalWrite(ADC\_CLK,0); delayMicroseconds(2);

}

digitalWrite(ADC\_CS,1);

pinMode(ADC\_DIO, OUTPUT);

return(dat1==dat2) ? dat1 : 0;

}

int main(void)

{

uchar digitalVal = 1;

uchar analogVal = 0;

if(wiringPiSetup() == -1)

{

printf("setup wiringPi failed !\n");

return -1;

}

pinMode(ADC\_CS, OUTPUT);

pinMode(ADC\_CLK, OUTPUT);

pinMode(Hall\_DO\_Pin, INPUT);

pullUpDnControl(Hall\_DO\_Pin, PUD\_UP);

pinMode(LedPin, OUTPUT);

while(1)

{

if((digitalVal = digitalRead(Hall\_DO\_Pin)))

{

printf("Do is %d.\n", digitalVal);

analogVal = get\_ADC\_Result();

printf("Current analog value is %d.\n", analogVal);

if(analogVal > thresholdVal)

{

digitalWrite(LedPin, HIGH);

}

delay(200);

}

else

{

digitalWrite(LedPin, LOW);

}

}

return 0;

}