Two-axis Joystick



Overview

Joysticks are positional controllers commonly found not only inmany video games, but also in navigational controllers for a wide variety of vehicle, such as airplanes, construction equipment, military vehicles, and remote control cars, as well as other positionable objects or parts of objects (wings, wheels, flaps, etc.).

A two-axis controller reports the position of its control stick separately in horizontal (left/right) and vertical (up/down) dimensions. The two-axis joystick module features a mechanical cross-rocker switch made of two perpendicular bidirectional resistors, which change resistance as the rocker moves with the stick. Since the module uses 5V power, in the middle or neutral state, both resistors read about 2.5V volts. As the stick is moved to one end of its range, voltage sinks to 0V; at the other end, it measures 5V. The sensor reports these two voltages—horizontal or x-axis position, and vertical or y-axis position—as analog outputs. In addition, the joystick contains a simple pushbutton switch, which is reported as a third, digital output. The pushbutton is not related electronically or mechanically to the joystick’s positional controller, but is frequently a valuable additional input to applications that can require a joystick’s position.

In this experiment, you’ll use the Raspberry Pi to monitor the joystick and report its three outputs—X position, Y position, and pushbutton—to the command line.

Materials Needed

Raspberry Pi x1

Breadboard x1

Joystick x1

ADC0832 x1

Dupont jumper wires

Experimental Procedure

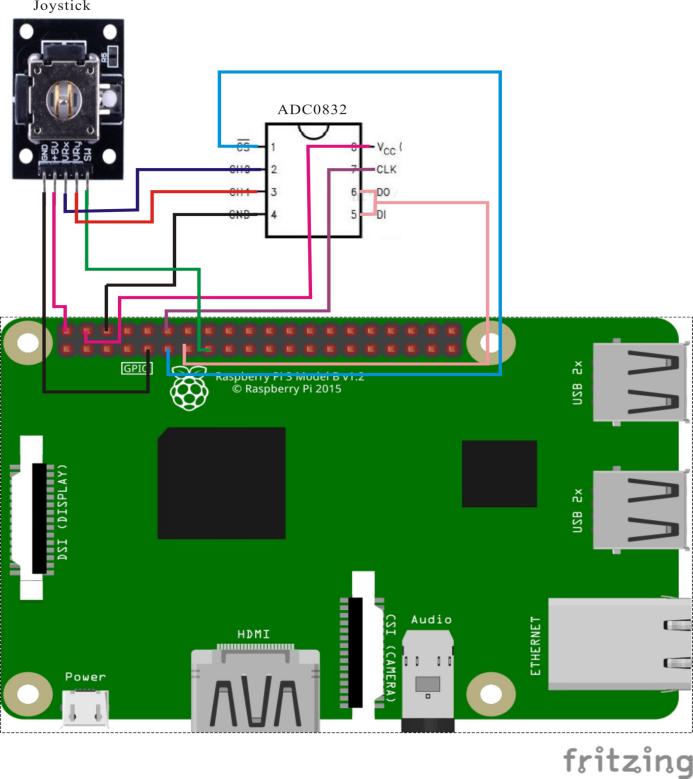
1. If you have not done so already, prepare your development system by installing the Python interpreter, RPi.GIO library, and wiringPi library as described in READ\_ME.TXT.
2. Install the ADC0832 analog/digital converter IC and joystick on your breadboard, and use Dupont jumper wires to connect them to each other and your Raspberry Pi as illustrated in the Wiring Diagram below.
3. Execute the sample stored in this experiment’s subfolder.

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

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

1. Make experimental observations as you move the joystick and press the onboard button. The Raspberry Pi’s command line output reports the position of the controller and status of the button through periodic text messages.

Wiring Diagram



ADC0382 pin position:

CS ↔ Raspberry Pi pin 11

CLK ↔ Raspberry Pi pin 12

DI ↔ Raspberry Pi pin 13

DO ↔ Raspberry Pi pin 13

CH0 ↔ Joystick pin VRx

CH1 ↔ Joystick pin VRy

VCC ↔ Raspberry Pi +5V

GND ↔ Raspberry Pi GND

Joystick pin position:

sw ↔ Raspberry Pin pin 15

VRx ↔ ADC0382 pin CH0

VRy ↔ ADC0382 pin CH1

GND ↔ Raspberry Pi GND

+5V ↔ Raspberry Pi +5V

Sample Code

Python Code

#!/usr/bin/env python

#

# This is a program for JoystickPS2 Module.

# This program depend on ADC0832 ADC chip.

#

import ADC0832

import RPi.GPIO as GPIO

import time

btn = 15

xFlag = 0

yFlag = 0

def setup():

ADC0832.setup() # Setup ADC0832

GPIO.setmode(GPIO.BOARD) # Numbers GPIOs by physical location

GPIO.setup(btn, GPIO.IN, pull\_up\_down=GPIO.PUD\_UP) # Setup button pin as input an pull it up

def getResult():

#get joystick result

global xFlag, yFlag

if ADC0832.getResult(1) == 0:

xFlag = 1 #up

if ADC0832.getResult(1) == 255:

xFlag = 2 #down

if ADC0832.getResult(0) == 0:

yFlag = 1 #left

if ADC0832.getResult(0) == 255:

yFlag = 2 #right

if GPIO.input(btn) == 0:

print 'Button is pressed!' # Button pressed

def loop():

while True:

getResult()

if xFlag == 1:

print 'up'

elif xFlag == 2:

print 'down'

if yFlag == 1:

print 'left'

elif yFlag == 2:

print 'right'

def destroy ():

GPIO.cleanup() # Release resource

if \_\_name\_\_ == '\_\_main\_\_': # Program start from here

setup()

try:

loop()

except KeyboardInterrupt:

destroy()

C Code

#include <wiringPi.h>

#include <stdio.h>

typedef unsigned char uchar;

typedef unsigned int uint;

#define ADC\_CS 0

#define ADC\_CLK 1

#define ADC\_DIO 2

#define JoyStick\_Button 3

#define UP 1

#define DOWN 2

#define LEFT 1

#define RIGHT 2

uchar get\_ADC\_Result(uchar xyVal)

{

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);

if(xyVal == 'x'){

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

}

if(xyVal == 'y'){

digitalWrite(ADC\_DIO,1); 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 xFlag, yFlag;

uchar xVal = 0, yVal = 0, bVal = 0;

if(wiringPiSetup() == -1)

{

printf("setup wiringPi failed !");

return -1;

}

pinMode(ADC\_CS, OUTPUT);

pinMode(ADC\_CLK, OUTPUT);

pinMode(JoyStick\_Button, INPUT);

pullUpDnControl(JoyStick\_Button, PUD\_UP);

while(1)

{

xFlag = 0;

yFlag = 0;

xVal = get\_ADC\_Result('x');

if(xVal == 0)

{

xFlag = UP; //up

}

if(xVal == 255)

{

xFlag = DOWN; //down

}

yVal = get\_ADC\_Result('y');

if(yVal == 0)

{

yFlag = LEFT; //left

}

if(yVal == 255)

{

yFlag = RIGHT; //right

}

bVal = digitalRead(JoyStick\_Button);

if(bVal == 0)

{

printf("Button is pressed !\n");

}

switch(xFlag)

{

case UP:

printf("up\n");

break;

case DOWN:

printf("down\n");

break;

default:

break;

}

switch(yFlag)

{

case LEFT:

printf("left\n");

break;

case RIGHT:

printf("right\n");

break;

default:

break;

}

delay(200);

}

return 0;

}