KUMAN
www.kumantech.com

Knock Switch

Overview

A knock switch is a simple switch that detects when a knock, shock or jolt is registered. (Unlike a shock switch, it detects impact rather than changes in position.) In this experiment, you'll make your Raspberry Pi turn on an LED light whenever you "knock" the knock switch.

Experimental Materials

```
Raspberry Pi      x1
Breadboard        x1
Knock Switch      x1
LED (3 pin)       x1
Resistor(330Ω)    x1
Dupont jumper wires
```

Experimental Procedure

1. If you have not done so already, prepare your development system by installing the Python interpreter, RPi.GPIO library, and wiringPi library as described in READ_ME_FIRST.TXT.

2. Install the shock switch and three-pin LED on your breadboard, and use the resistor and Dupont jumper wires as illustrated in the Wiring Diagram below. Note you will connect only two of the three pins on the LED.

3. Execute the sample stored in this experiment's subfolder.
   If using C, compile and execute the C code:

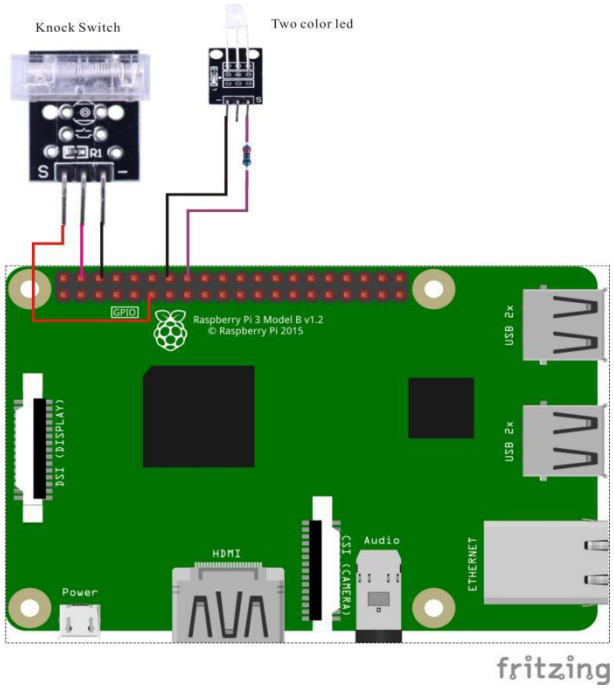
   ```
   cd Code/C
   gcc knockSwitch.c -o knockSwitch.out -lwiringPi
   ./knockSwitch.out
   ```

If using Python, launch the Python script:

```
cd Code/Python
python knockSwitch.py
```

4. Make experimental observations. Each knock changes the current state of the LED. Knock once to turn the light on. Knock a second time to turn it off.

Wiring Diagram



Knock Switch pin position:

S ↔ Raspberry Pi pin 11

"+" ↔ Raspberry Pi +5V

"-" ↔ Raspberry Pi GND

LED pin position:

"S" ↔ Raspberry Pi pin 16 (through resistor)

"-" ↔ Raspberry Pi GND

Sample Code

Python Code

```
#!/usr/bin/env python
import RPi.GPIO as GPIO

KnockPin = 11
LedPin = 16

Led_status = 0

def setup():
    GPIO.setmode(GPIO.BOARD)  # Numbers GPIOs by physical location
    GPIO.setup(LedPin, GPIO.OUT)   # Set LedPin's mode is output
    GPIO.setup(KnockPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)

def swLed(ev=None):
    global Led_status
    Led_status = not Led_status
    GPIO.output(LedPin, Led_status) # switch led status(on-->off;
off-->on)
    print "LED: " + ("on" if Led_status else "off")

def loop():
    GPIO.add_event_detect(KnockPin, GPIO.FALLING, callback=swLed,
bouncetime=200) # wait for falling
    while True:
        pass   # Don't do anything

def destroy():
    GPIO.output(LedPin, GPIO.LOW)     # led off
    GPIO.cleanup()                    # Release resource

if __name__ == '__main__':     # Program start from here
    setup()
    try:
        loop()
    except KeyboardInterrupt:  # When 'Ctrl+C' is pressed, the child
program destroy() will be  executed.
        destroy()
```

C Code

```
#include <wiringPi.h>
#include <stdio.h>

#define KnockPin     0
#define LedPin       4

int knockPinValue = -1;

int main(void)
{
    int knockValue = -1;
    if(wiringPiSetup() == -1)
    {
        printf("setup wiringPi failed !");
        return 1;
    }
    pinMode(KnockPin, INPUT);
    pinMode(LedPin,  OUTPUT);

    while(1)
    {
        knockValue = digitalRead(knockPin);
        knockPinValue = knockValue;
        delay(6);

        knockValue = digitalRead(knockPin);
        if(knockPinValue != knockValue)
        {
            printf("Detected knocking!\n");
            digitalWrite(LedPin, !digitalRead(LedPin));
        }
    }
    return 0;
}
```

Technical Background

The device is a normally-open switch held high with a 10KΩ pull-up resistor connected to +5V, that closes to ground when a knock is detected. The switch remains closed only momentarily, which the sample code detects by polling (C code) or by associating an event-trigger with the state transition (Python code).