



Raspberry Pi Starter kit # K0063

User Guide

Rev 1.0 Jan 2017

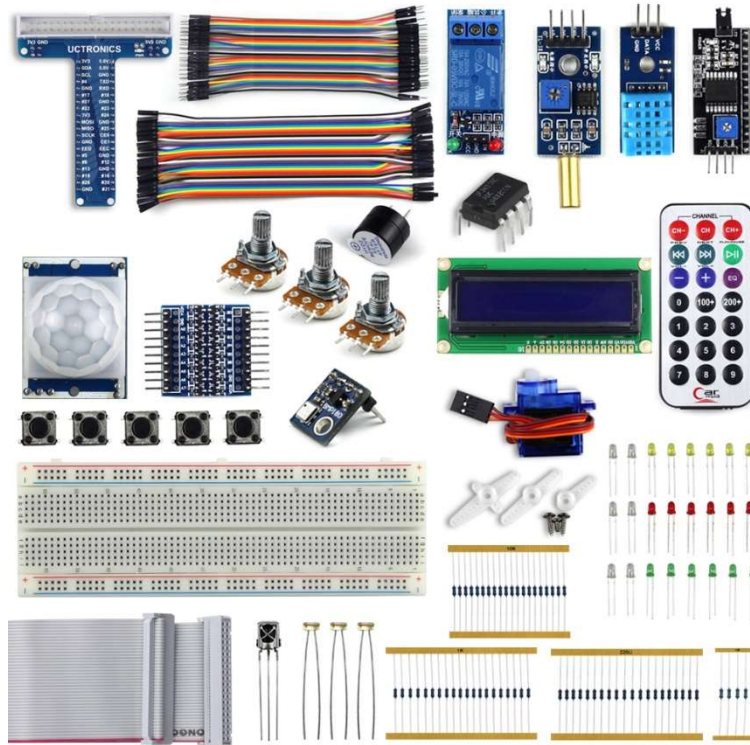
Table of Contents

1. Kit Introduction.....	5
2. Components and projects listing	6
3. Project details.....	7
3.1 Solderless Prototype Breadboard.....	7
3.2. 40pin ribbon cable	7
3.3 T-type GPIO Extension Board	8
3.3.1 Introduction.....	8
3.3.2 Hardware required.....	8
3.3.3 Circuit graph:	9
3.3.4 Prerequisite:.....	9
3.3.5 Program and Code	9
3.4 8 channel logic level converter	10
3.5 Push buttons	10
3.5.1 Introduction.....	10
3.5.2 Hardware required.....	11
3.5.3 Circuit graph	11
3.5.4 Program and code.....	12
3.6 40pin M/M jumper wires	12
3.7 40pin F/F jumper wires.....	13
3.8 1602 serial LCD module	13
3.8.1 Introduction:.....	13
3.8.2 Hardware required.....	14
3.8.3 Circuit Connection Graph:.....	14
3.8.4 Program and code	14
3.9 I2C Serial Interface Adapter Module.....	16
3.9.1 Introduction.....	16
3.10 SG90 Servo Motor	17
3.10.1 Introduction	17

3.10.2 Hardware required	18
3.10.3 Prerequisite:	18
3.10.4 Circuit Graph:	19
3.10.5 Program and code.....	19
3.11 5V 1-Channel Relay Module	19
3.11.1 Introduction	19
3.11.2 Hardware required	20
3.11.3 Circuit connection graph.....	21
3.11.4 Software Installation:.....	21
3.12 DHT11 Temp & Humi Sensor Module	22
3.12.1 Introduction	22
3.12.2 Hardware required	23
3.12.3 Circuit Graph:	23
3.12.4 Program and code.....	23
3.13 BMP180 Barometric Pressure Sensor Module.....	26
3.13.1 Introduction:	26
3.13.2 Hardware required	27
3.13.3 Circuit Graph:	27
3.13.4 Software Installation:.....	27
3.14 HC-SR501 Infrared Motion Sensor Module.....	28
3.14.1 Introduction	28
3.14.2 Hardware required	29
3.14.3 Circuit Graph	30
3.14.4 Program and code.....	30
3.15 Analog to Digital Converter.....	32
3.15.1 Introduction	32
3.15.2 Hardware required	33
3.15.3 Circuit graph.....	34
3.15.4 Program and code:.....	34
3.16 Tilt Sensor Module.....	37

3.16.1 Introduction	37
3.16.2 Hardware required	37
3.16.3 Prerequisite:	38
3.16.4 Circuit graph:	38
3.16.5 Program and code	39
3.17 3x Potentiometer (10kilohm adjustable resistor)	39
3.18 Piezo Buzzer	40
3.18.1 Introduction	40
3.18.2 Hardware required	41
3.18.3 Circuit Graph:	41
3.18.4 Program and code	42
3.19 GL5516 Photo resistor (Light Sensor)	43
3.19.1 Introduction	43
3.19.2 Hardware required	44
3.19.3 Circuit Connection Graph	45
3.19.4 Install software:	45
3.20 LED (6 x White, 6 x Red, 6 x Yellow, 6 x Green)	46
3.21 Infrared Remote Controller	47
3.21.1 Introduction	47
3.21.2 Hardware required	48
3.21.3 Circuit Graph	48
3.21.4 Program and code	48
3.22 Infrared Receiver (VS1838B)	52
3.23 How to read resistor color code	54
Appendix	55
How to read Raspberry Pi I/O pin diagram (GPIO pin graph)	55
Thanks	56

1. Kit Introduction



UCTRONICS designs, manufactures and provides technical service for open source (Raspberry pi and Arduino) hardware and software. Dedicated to applying the Internet and the latest industrial technology in open source area, we strive to provide best hardware support and software service for general makers and electronic enthusiasts around the world. We aim to create infinite possibilities with sharing. No matter what field you are in, we can lead you into the electronic world and bring your ideas into reality.

This is an entry-level learning kit for Arduino. Some common electronic components and sensors are included. Through the learning, you will get a better understanding of Arduino, and be able to make fascinating works based on Arduino.

If you have any problems for learning, please contact us at sales@uctronics.com, our engineer will solve the problem with you as soon as possible.

If you need to buy any electronic components, please feel free to view www.uctronics.com.

NOTE:

Before learning the next courses, please copy the source code we provided to your Raspberry Pi's /home/ directory, or you can get the source code directly from our GitHub repository:

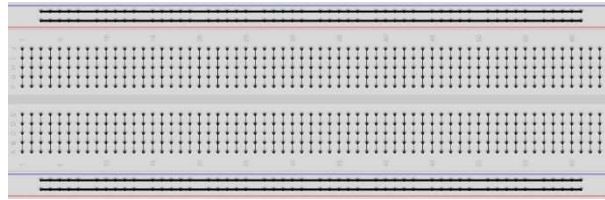
\$ [git clone https://github.com/UCTRONICS/Arducam_RPi_Code.git](https://github.com/UCTRONICS/Arducam_RPi_Code.git)

2. Components and projects listing

No	Components	Projects
1	1x Solderless Prototype Breadboard	Specific introduction
2	1x 40 Pin Ribbon Cable	Wring RPi and the RPi GPIO
3	1x T-type GPIO Extension Board	Output signal from Raspberry Pi GPIO pin to LED
4	1x 8 Channel Logic Level Converter	Specific introduction
5	5x Push Buttons	Controlling an LED with a button
6	1x 40 Pin Jumper Wires (15cm, M to M)	Specific introduction
7	1x 40 Pin Jumper Wires (20cm, F to F)	Specific introduction
8	1x 1602 Serial LCD Module	Drive I2C 16×2 LCD with Raspberry Pi
9	1x I2C Serial Interface Adapter Module	Specific introduction
10	1x SG90 Servo Motor	Drive Servo Motor with Raspberry Pi
11	1x 5V 1-Channel Relay Module	Using Raspberry Pi to drive relay
12	1x DHT11 Temp & Humi Sensor Module	Drive DHT11 Temperature Sensor with Raspberry Pi
13	1x BMP180 Barometric Pressure Sensor	Using Raspberry Pi to drive BMP180
14	1x HC-SR501 Infrared Motion Sensor	Drive motion sensor and turn on LED
15	1x I/P Analog To Digital Converter	make a MP3 music player
16	1x Tilt Switch Sensor Module	Tilt switch work with Raspberry Pi
17	3x Potentiometer	make a MP3 music player
18	1x Piezo Buzzer	Using Raspberry Pi to drive buzzer
19	3x GL5516 Photo resistor (Light Sensor)	Get light strength data with photo resistor
20	24x LED	Make a LED blinking
21	1x Infrared Receiver (VS1838B)	Use Raspberry pi to get IR remote code
22	1x Infrared Remote Controller	Use Raspberry pi to get IR remote code
23	20x Resistors (10Kohm) 20x Resistors (1Kohm) 20x Resistors (220ohm) 5x Resistors (1Mohm)	How to read resistor color code

3. Project details

3.1 Solderless Prototype Breadboard



There are total 830 pins on the Solderless Prototype Breadboard, 12 rows of pins as the above picture. The pins on each black line output and input the same signal. You can connect the breadboard to Raspberry pi to transmit signals. With this board, you can extensive more projects about Raspberry pi.

Specification of the breadboard:

- Material: ABS plastic material, Phosphor bronze nickel plated spring clips
- Accepts a variety of wire sizes (20-29 AWG)
- Dimensions: 16.5 x 5.4 x 0.9 cm
- Colored coordinates for easy component placement.

3.2. 40pin ribbon cable

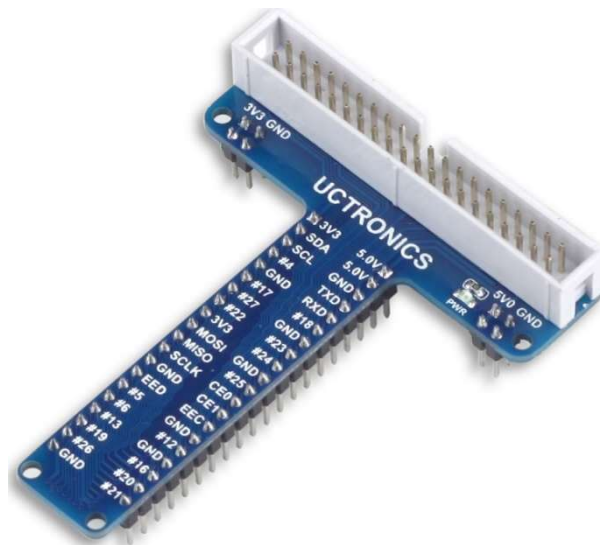


As you can see in the following picture, we can use the 40pin ribbon cable to connect Raspberry pi and GPIO extensive board.

It's pretty easy to use both the 40pin ribbon cable and GPIO extensive board in your project.

3.3 T-type GPIO Extension Board

3.3.1 Introduction



This board can break out all those tasty power, GPIO, I2C and SPI pins from the 40-pin header onto a solderless breadboard. Each order comes with a 40 pin ribbon cable and assembled T Type plus GPIO Expansion Board. You can plug the 40-pin GPIO cable between the Pi computer and the T Type plus GPIO expansion board.

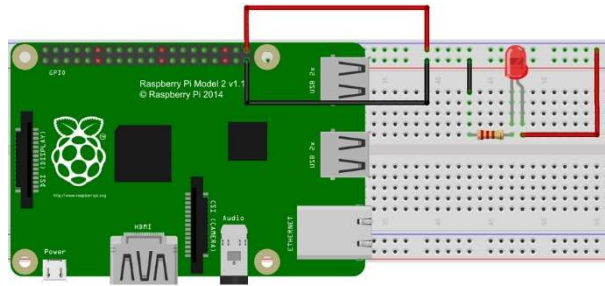
In this project (Output signal from Raspberry Pi GPIO pin to LED), we will use GPIO port 21 (Pin 40) to output signals to LED and make it flashing ten times.

If you don't know what GPIO layout is, check the appendix (How to read Raspberry Pi I/O pin diagram).

3.3.2 Hardware required

- 1x Raspberry Pi 2/3/zero
- 1x 8GB MicroSD memory card preinstalled Raspbian OS.
- 1x LED
- 1x 220 Ω resistance
- 1x Breadboard

3.3.3 Circuit graph:



3.3.4 Prerequisite:

1) Raspbian should be upgraded to latest version in order to support RPI.GPIO module

Please run following commands in shell:

- `sudo apt-get update`
- `sudo apt-get upgrade`

2) Enable I2C and SPI protocol to enable the protocol, run shell command

- `sudo raspi-config`

Then select Advance Options and enable I2C and SPI

You need to reboot to effect the configuration

3.3.5 Program and Code

1) Run following command in shell window:

- `sudo nano testgpio.py`

2) Then copy python code and paste the code into testgpio.py

Then Type "Ctrl" + "X" and "Yes" to save the file and press enter

3) If you don't want use Nano to write code, you can also download above python file from our server by typing following shell command:

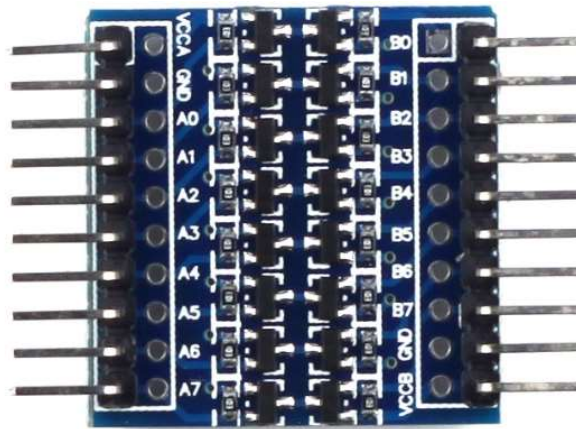
- `cd testgpio`

Finally, run following command in shell window:

- `sudo python testgpio.py`

You will see the LED flashing 10 times and then the python program stops automatically.

3.4. 8 channel logic level converter



Specification of the 8 channel logic level converter

- 8-channel high voltage and low voltage logic bidirectional logic conversion
- Two-way conversion between Ax and Bx
- Conversion level range: 1.8V-6V
- Module size: 28mm X 27.5mm
- Instructions: for example :(3.3V to 5V conversion)
- VCCA connect to 3.3V power supply
- VCCB connect to 5V power supply
- GND connect to negative, two power supply common ground
- Ax input 3.3V TTL level, Bx output 5V TTL level
- Bx input 5V TTL level, Ax will output 3.3V TTL level

3.5. Push buttons

3.5.1 Introduction



Specification of the buttons:

- Standard, Flat Plunger Type (without Ground Terminal)
- Contact material: Silver plated
- Operating force:
 - 0.98 N {100 gf}
 - 1.47 N {150 gf}
 - 2.55 N {260 gf}
 - 4.9 N {500 gf}

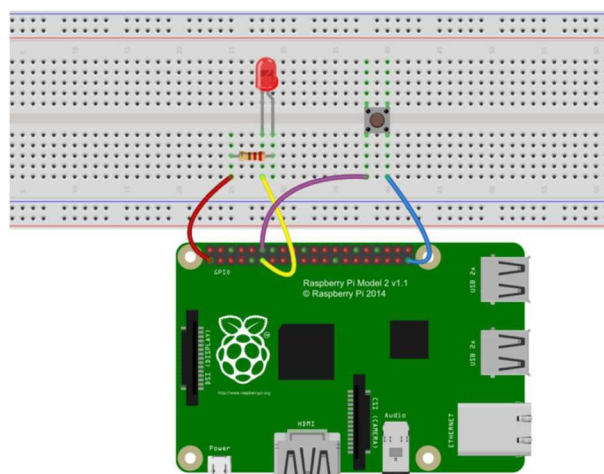
In this project (Controlling an LED with a button), we will learn how to detect the state of a button, and then toggle the state of LED based on the state of the button.

Note: Be sure to read the safety precautions common to all Tactile Switches for correct use.

3.5.2 Hardware required

- 1x Arduino MEGA 2560
- 1x USB Cable
- 1x Button
- 1x LED
- 1x 10KΩ Resistor
- 1x 220Ω Resistor
- 1x Breadboard
- 4x Jumper Wires

3.5.3 Circuit graph



3.5.4 Program and code

1) Run following command in shell window

- `cd btnAndLed`

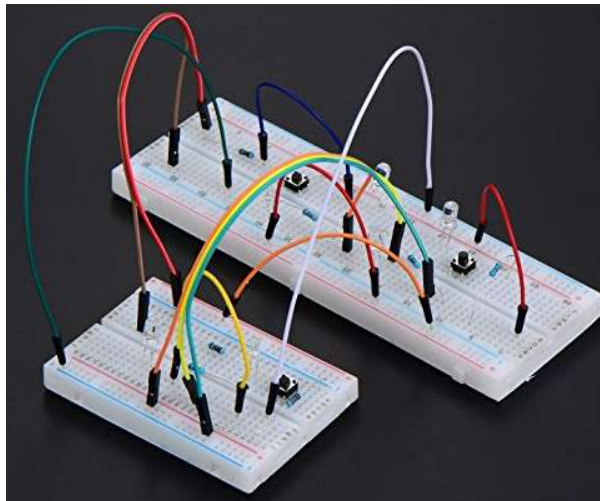
2) Run the program

- `sudo python btnAndLed.py`

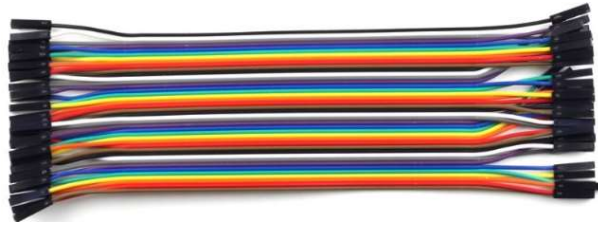
Now, when you press the button, you can see the state of the LED will be toggled.
(ON->OFF, OFF->ON).

3.6. 40pin M/M jumper wires

The 40 pin jumper wires in the kit are male to male, the length is 15cm. You will need them in many projects about Raspberry pi and other electronic projects.



3.7. 40pin F/F jumper wires



The 2 x 8 pin jumper wires in the kit are female to female, the length is 20cm. There are three kinds of connection cable available in the kit totally. You can choose the one you need in your project.

3.8. 1602 serial LCD module

3.8.1 Introduction:



LCD1602 is a kind of character LCD display. The LCD has a parallel interface, meaning that the microcontroller has to manipulate several interface pins at once to control the display. The interface consists of the following pins:

- 1) A register select (RS) pin that controls where in the LCD's memory you're writing data to. You can select either the data register, which holds what goes on the screen, or an instruction register, which is where the LCD's controller looks for instructions on what to do next.
- 2) A Read/Write (R/W) pin that selects reading mode or writing mode
- 3) An Enable pin that enables writing to the registers
- 4) 8 data pins (D0-D7). The status of these pins (high or low) are the bits that you're writing to a register when you write, or the values when you read.
- 5) There's also a display contrast pin (Vo), power supply pins (+5V and GND) and LED Backlight (Bkl+ and Bkl-) pins that you can use to power the LCD, control the display contrast, and turn on or off the LED backlight respectively.

In this project (Drive I2C 16×2 LCD with Raspberry Pi), we will connect the Pi with an I2C

enabled LCD screen which only has 4 pins as shown below.

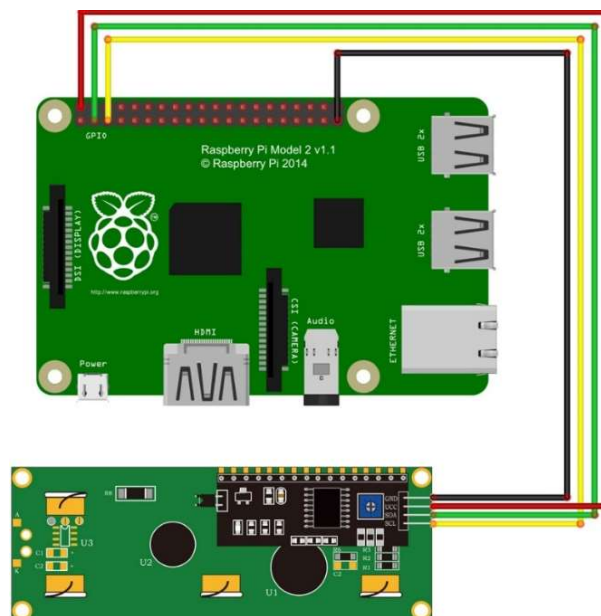
If you don't know what GPIO layout is, check our appendix: How to read Raspberry Pi I/O pin diagram (GPIO pin graph).

Note: To make sure this project works properly, you should use a MicroSD card with fresh-installed Raspbian OS.

3.8.2 Hardware required

- 1x Raspberry Pi
- 1x 1602 LCD module
- 1x I2C serial adapter module
- 4x Jumper wires

3.8.3 Circuit Connection Graph:



3.8.4 Program and code

1) Enable i2c using raspi-config utility

In terminal, type the following command:

- `sudo raspi-config`

Select Advanced Option -> I2C -> Enable I2C -> Finish

The Pi will reboot after you click the Finish Button

After rebooting the Pi, we need to modify the module's config file. Type the following

command in terminal:

- `sudo nano /etc/modules`

Add following two lines in modules file if they do not exist:

- `i2c-bcm2708`
- `i2c-dev`

Then Type "Ctrl" + "X" and "Yes" to save the file.

2) Install smbus and i2c python library

Type following command in terminal:

- `sudo apt-get update`
- `sudo apt-get install -y python-smbus i2c-tools`
- `sudo reboot`

After rebooting the system, type the following command in order to check software installation:

- `lsmod | grep i2c_`

You should see i2c_bcm2708 in a list, this means the library has been installed successfully.

Otherwise you might need to find another Raspbian OS MicroSD card and repeat Step 1 and 2.

3) Testing Hardware

Depending on your Raspberry Pi version, please run one of following commands in terminal:

- `sudo i2cdetect -y 1` or `sudo i2cdetect -y 0`

You should see something as follows:

```
0 1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  -----
10:  -----
20:  -----
```

30: --- -- 27

40: ---

50: ---

60: ---

70: ---

If you can only see "--- --" sign in the list without any numbers, it means either your circuit connection is wrong or your software is not properly installed.

4) Download Python Code and run the project:

In terminal window, type the following commands:

- `cd i2clcd`
- `sudo python i2clcd.py`

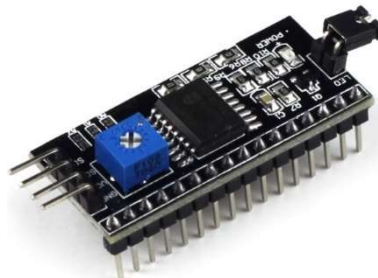
Now your LCD should display the following messages:

- *Created by*
- *Lee*
- *> Tutorial Url:*
- *> <http://uctronics.com>*

If your Pi does not show any runtime error but LCD still does not display any messages, you can use a screw driver to adjust the contrast screw on the back of the LCD until you see the message.

3.9 I2C Serial Interface Adapter Module

3.9.1 Introduction



This I2C Serial Interface Adapter Module can be used with 1602 LCD module, then you can

only connect the 1602 LCD module to Raspberry pi with 4 pins. It'll be easier for you to develop some innovative project about the 1602 LCD.

Interface of the adapter module:

- VCC: 3.3-5V
- GND: Ground
- SCL: I2C Serial Clock (A2)
- SDA: I2C Serial Data (A3)

The application of the I2C Serial Interface Adapter Module is in project 8 (Drive I2C 1602 LCD with Raspberry Pi), please check the project 8 for your reference.

3.10. SG90 Servo Motor

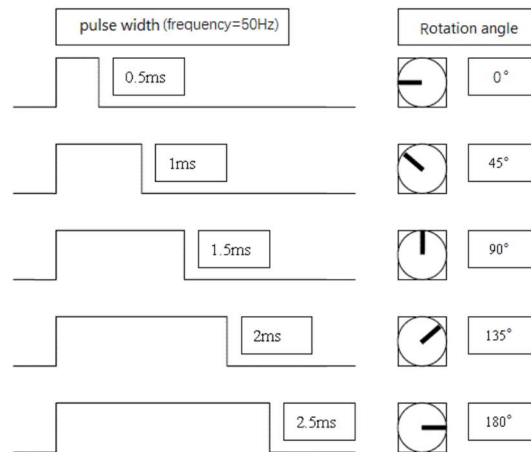
3.10.1 Introduction



Servo is a type of geared motor that can only rotate 180 degrees. It is controlled by sending pulses signal from your microcontroller. These pulses tell the servo what position it should move to.

Servo consists of shell, circuit board, non-core motor, gear and location detection. Its working principle is as follow: Raspberry Pi sends PWM signal to servo motor, and then this signal is processed by IC on circuit board to calculate rotation direction to drive motor, and then this driving power is transferred to swing arm by reduction gear. At the same time, position detector returns location signal to judge whether set location is reached or not.

The relationship between the rotation angle of the servo and pulse width as shown below:



In this project (Drive Servo Motor with Raspberry Pi), we will use Raspberry Pi to send command through GPIO to Servo Motor and control its rotation action.

If you don't know what GPIO layout is, check our appendix: How to read Raspberry Pi I/O pin diagram (GPIO pin graph)

3.10.2 Hardware required

- 1x Raspberry Pi 2/3/zero
- 1x 8GB MicroSD memory card preinstalled Raspbian OS.
- 1x Servo Motor
- 1x Breadboard
- 1x GPIO breakout kit (optional)

3.10.3 Prerequisite:

1) Raspbian should be upgraded to latest version in order to support RPI.GPIO module

Please run following commands in shell:

- `sudo apt-get update`
- `sudo apt-get upgrade`

2) Enable I2C and SPI protocol

To enable the protocol, run shell command

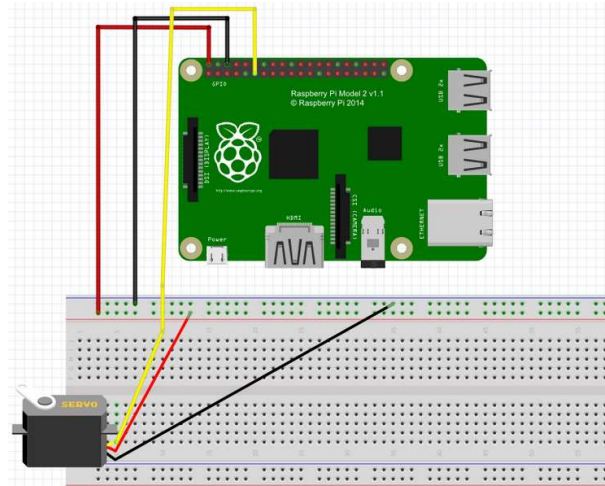
- `sudo raspi-config`

Then select Interfacing Options and enable I2C and SPI

You need to reboot to effect the configuration.

Note: Unlike Arduino board 5V input voltage, Raspberry GPIO pin accept only 3 Volt. Wrong voltage input might damage the Pi board. Please be very careful!

3.10.4 Circuit Graph:



3.10.5 Program and code

Run following command in shell window:

- `cd pi-servo`
- `sudo python pi-servo.py`

You will see following manual in terminal window:

- `l = move to the left`
- `r = move to the right`
- `m = move to the middle`
- `t = test sequence`
- `q = stop and exit`

You can type above 5 options (l, r, m, t, q) and control the servo motion accordingly.

3.11 5V 1-Channel Relay Module

3.11.1 Introduction



A relay is an electrically operated switch. It is generally used in automatic control circuit. Actually, it is an "automatic switch" which uses low current to control high current. It plays a role of automatic regulation, security protection and circuit switch. When an electric current is passed through the coil it generates a magnetic field that activates the armature, and the consequent movement of the movable contact(s) either makes or breaks (depending upon construction) a connection with a fixed contact. If the set of contacts was closed when the relay was de-energized, then the movement opens the contacts and breaks the connection, and vice versa if the contacts were open.

When the current to the coil is switched off, the armature is returned by a force, approximately half as strong as the magnetic force, to its relaxed position. Usually this force is provided by a spring, but gravity is also used commonly in industrial motor starters. Most relays are manufactured to operate quickly. In a low-voltage application this reduces noise; in a high voltage or current application it reduces arcing.

When the coil is energized with direct current, a diode is often placed across the coil to dissipate the energy from the collapsing magnetic field at deactivation, which would otherwise generate a voltage spike dangerous to semiconductor circuit components.

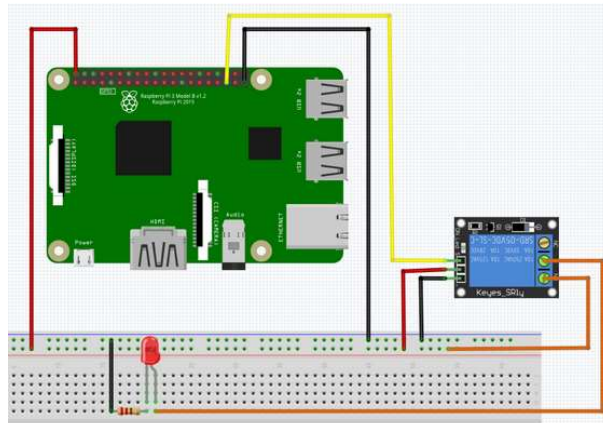
In this project (Using Raspberry Pi to drive relay), we will use Raspberry pi to drive relay and control the on/off of a LED light. After running the code from Pi, the LED will turn on and off LED every one second.

3.11.2 Hardware required

- 1x Raspberry Pi3
- 1x Raspberry Pi T style extension
- 1x Relay module

- 1x LED
- 1x USB mouse and keyboard
- 1x HDMI cable and monitor (or TV)
- 1x 1k ohm resistor
- 1x breadboard
- 8x jump wires

3.11.3 Circuit connection graph



3.11.4 Software Installation:

1) Install git core (if you have already installed git core, please skip this step). Please run following shell command in Pi terminal:

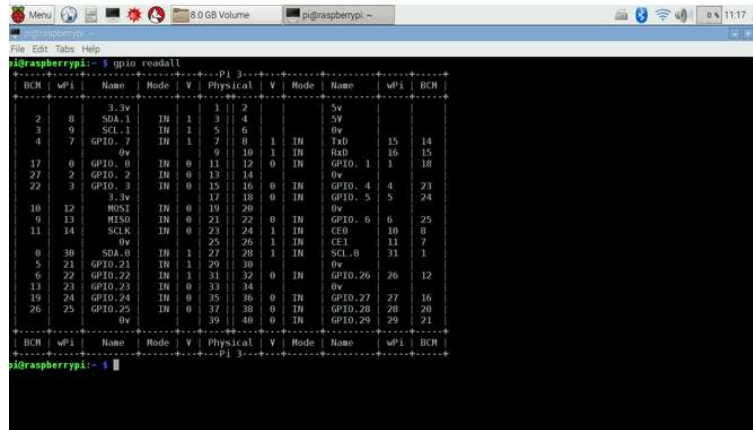
- `sudo apt-get install git-core`
- `sudo apt-get update`
- `sudo apt-get upgrade`

2) Install wiringPi library (if you have installed wiringPi, please skip this step). Please run following command in Pi terminal:

- `git clone git://git.drogon.net/wiringPi`
- `cd wiringPi`
- `./build`

Note: In relay.c file, the relay is connected to port no.24. However, connection graph shows

the relay is connected to GPIO 19. This is because relay.c includes wiringPi library whose port number does not match GPIO number port 24 is actually GPIO 19. You can use GPIO readall to check GPIO port mapping (result as per following graph)



BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM
2	8	3.3v			3	2		5v		
3	9	SDA.1	IN	1	5	6		0v		
4	7	SCL.1	IN	1	7	8	1	I2D	15	14
		0v			9	10	1	RxD	16	15
17	0	GPIO.0	IN	0	11	12	0	IN	GPIO.1	1
27	2	GPIO.2	IN	0	13	14		0v		
22	3	GPIO.3	IN	0	15	16	0	IN	GPIO.4	4
		3.3v			17	18	0	IN	GPIO.5	5
10	12	REST	IN	0	19	20		0v		
0	13	RTS0	IN	0	21	22	0	IN	GPIO.6	6
11	14	SCLK	IN	0	23	24	1	IN	CE0	10
0	30	SDA.0	IN	1	27	28	1	IN	CE1	11
5	21	GPIO.21	IN	1	29	30		0v		
6	22	GPIO.22	IN	1	31	32	0	IN	GPIO.26	26
13	23	GPIO.23	IN	0	33	34		0v		
19	24	GPIO.24	IN	0	35	36	0	IN	GPIO.27	27
26	25	GPIO.25	IN	0	37	38	0	IN	GPIO.28	28
		0v			39	40	0	IN	GPIO.29	29

Compile and running the code by typing following commands:

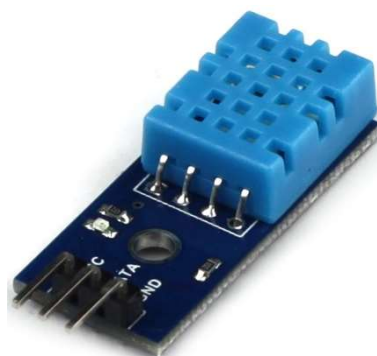
- `cd relay`
- `gcc -Wall -o relay relay.c -lwiringPi`
- `sudo ./relay`

You will see LED flash every one second

Type "Ctrl" + "c" to stop the program

3.12 DHT11 Temp & Humi Sensor Module

3.12.1 Introduction



This DHT-11 temperature & humidity sensor features a temperature & humidity sensor complex with a calibrated digital signal output. By using the exclusive digital-signal-acquisition technique and temperature & humidity sensing technology, it ensures high

reliability and excellent long-term stability. This sensor includes a resistive-type humidity measurement component and an NTC temperature measurement component, and connects to a high-performance 8-bit microcontroller, offering excellent quality, fast response, anti-interference ability and cost-effectiveness.

In this project (Drive DHT11 Temperature Sensor with Raspberry Pi), we will get temperature and humid data from DHT11 and send it to Raspberry Pi, then display the temperature and humid on 16x2 LCD screen.

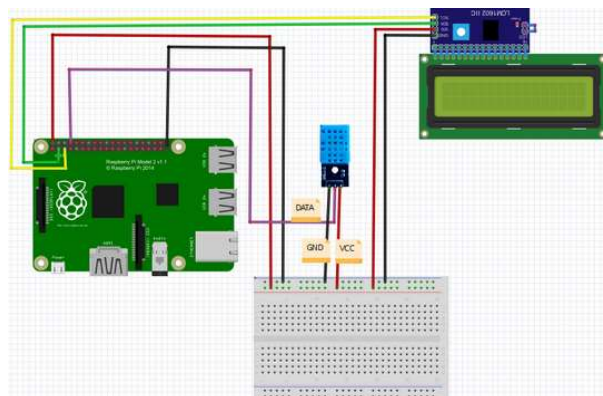
If you don't know what GPIO layout is, check the appendix: How to read Raspberry Pi I/O pin diagram (GPIO pin graph)

Note: Unlike Arduino board 5V input voltage, Raspberry GPIO pin accept only 3 Volt. Wrong voltage input might damage the Pi board. Please be very careful!

3.12.2 Hardware required

- 1x Raspberry Pi 2/3/zero
- 1x 8GB MicroSD memory card x 1preinstalled Raspbian OS.
- 1x DHT11 Temperature/Humid Sensor
- 1x I2C 1602 LCD screen
- 1x Breadboard
- 1x GPIO breakout kit (optional)

3.12.3 Circuit Graph:



3.12.4 Program and code

1) Raspbian should be upgraded to latest version in order to support RPI.GPIO module

Please run following commands in shell:

- `sudo apt-get update`
- `sudo apt-get upgrade`

2) Enable I2C and SPI protocol

To enable the protocol, run shell command

- `sudo raspi-config`

Then select Advance Options and enable I2C and SPI

After rebooting the Pi, we need to modify the module's config file. Type the following command in terminal:

- `sudo nano /etc/modules`

Add following two lines in modules file if they do not exist:

- `i2c-bcm2708`
- `i2c-dev`

Then Type "Ctrl" + "X" and "Yes" to save the file.

3) Install smbus and i2c python library

Type following command in terminal:

- `sudo apt-get update`
- `sudo apt-get install -y python-smbus i2c-tools`
- `sudo reboot`

After rebooting the system, type the following command in order to check software installation:

- `lsmod | grep i2c`

You should see i2c_bcm2708 in a list, this means the library has been installed successfully. Otherwise you might need to find another Raspbian OS MicroSD card and repeat Step 2 and 3.

4) Raspberry Pi, I2C 1602 LCD and DHT11 sensor pin connection

LCD Pin	Description	Pi Function	RPi Pin
GND	GND	GND	pin 6 or pin 39
VCC	+5V/3.3v	+3.3V	pin 2
SDA	SDA	GPIO 02	pin 3
SCL	SCL	GPIO 03	pin 5
DHT11 DATA pin		GPIO 14	pin 8
DHT11 +VCC pin		+3V	pin 1
DHT11 GND(-)		GND	pin 6

5) Testing Hardware

Depending on your Raspberry Pi version, please run one of following commands in terminal:

- `sudo i2cdetect -y 1` or `sudo i2cdetect -y 0`

You should see something as follows:

```

0 1 2 3 4 5 6 7 8 9 a b c d e f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  3f
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --

```

If you can only see "-- -- --" sign in the list without any numbers, it means either your circuit connection is wrong or your software is not properly installed.

Please take special attention on the "3f" in above result, this is the i2c address of your LCD, it might be other value such as 27. If the value is not 3f, you need change line 18 of file "pi-dht11-i2clcd.py" (you need download this file in Step 6 with wget command)

6) If you get other number in the step 5(not 3F), please type following command in shell window

- `sudo nano pi-dht11-i2clcd.py`

Then copy python code and paste the code into pi-dht11-ic2lcd.py.

You need change line 18 of file "pi-dht11-i2clcd.py" (I2C_ADDR = 0x3f, use your number to replace 3f)

Then Type "Ctrl" + "X" and "Yes" to save the file and press enter

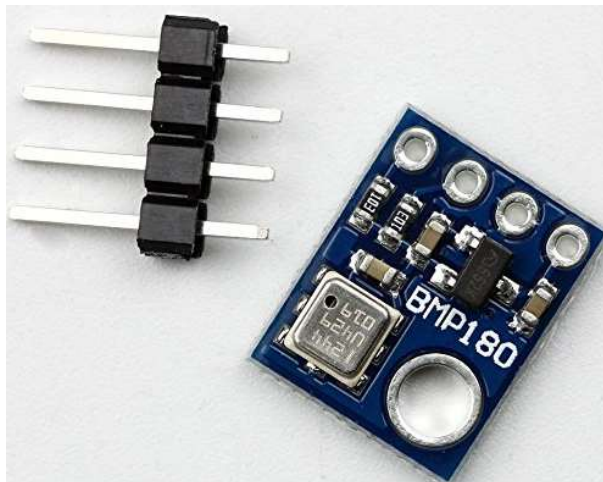
Finally, run following command in shell window:

- `cd dht11`
- `sudo python pi-dht11-i2clcd.py`

If you cannot see the text clearly in LCD, use a screw driver to adjust the brightness screw (on the back of I2C serial interface adapter module) until the display is OK.

3.13 BMP180 Barometric Pressure Sensor Module

3.13.1 Introduction:



The BMP180 is the function compatible successor of the BMP085, a new generation of high precision digital pressure sensors for consumer applications.

- The I2C bus is used to control the sensor, SDA (serial data) and SCL (serial clock) have open-drain outputs
- Pressure range: 300 ~ 1100hPa (+9000m ~ -500m relating to sea level)
- Supply voltage: 1.8 ~ 3.6V (VDD), 1.62V ~ 3.6V (VDDIO)
- Low power: 5uA at 1 sample / sec. in standard mode
- Low noise: 0.06hPa (0.5m) in ultra-low power mode, 0.02hPa (0.17m) advanced

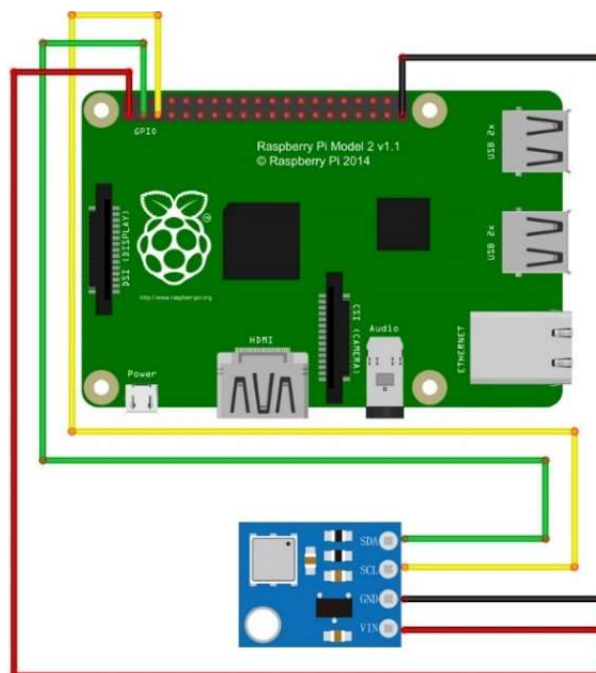
resolution mode

In this project (Using Raspberry Pi to drive BMP180), we will use Raspberry pi to get environment temperature, pressure from BMP180 sensor and calculate attitude, then display the result to monitor.

3.13.2 Hardware required

- 1x Raspberry Pi 3
- 1x GPIO to Breadboard 40-pin breakout interface
- 1x BMP180
- 1x USB mouse and keyboard
- 1x HDMI cable and monitor(or HDMI TV)
- 1x Breadboard
- 4x jump wires

3.13.3 Circuit Graph:



3.13.4 Software Installation:

1) Install BMP Python Library

- `cd Adafruit_Python_BMP`
- `sudo python setup.py install`

2) Test BMP180 program

- `cd examples`
- `sudo python simpletest.py`

3) Result

After running code, you will see result as per following picture:

```
pi@raspberrypi:~/Arducam_RPi_Code/Adafruit_Python_BMP $ cd examples/  
pi@raspberrypi:~/Arducam_RPi_Code/Adafruit_Python_BMP/examples $ sudo python simpletest.py  
Temp = 26.40 *C  
Pressure = 101558.00 Pa  
Altitude = -19.30 m  
Sealevel Pressure = 101566.00 Pa
```

3.14. HC-SR501 Infrared Motion Sensor Module

3.14.1 Introduction

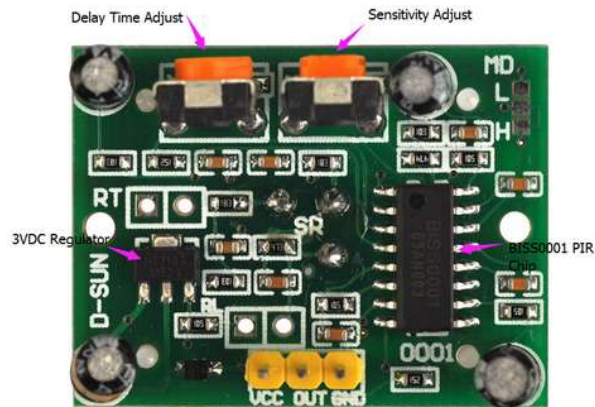


Generally speaking, motion sensor is made of thermoelectric sensor, and it can detect body motion. The motion sensor module generally uses the BISS0001 ("Micro Power PIR Motion Detector IC") to convert the analog signal from the sensor to digital output.

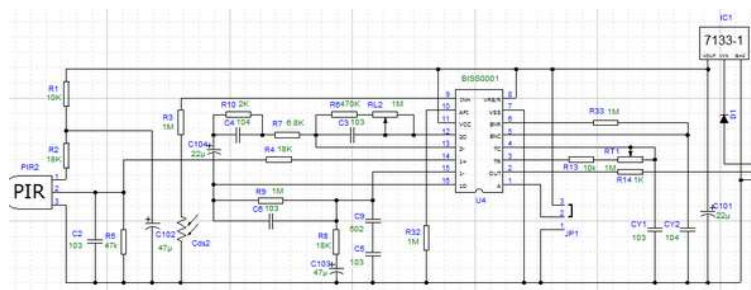
Experimental principle of this project is that, when the detection of the human body motion, GPIO output get high level and light is on; else, GPIO output would get low level and LED would be off.

In this project (Drive motion sensor and turn on LED), Python Language will help us achieve this experiment. Enjoy Raspberry Pi Python programming tour with us.

Motion sensor Interface Layout



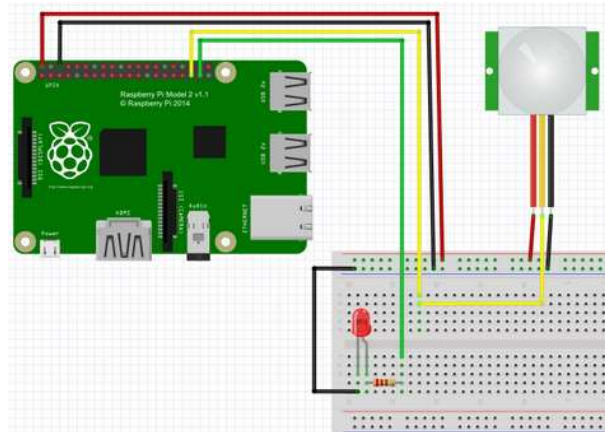
Motion sensor Schematic



3.14.2 Hardware required

- 1x Raspberry Pi board
- 1x Motion Sensor
- 1x LED
- 8x Jumper wires
- 1x Breadboard
- 1x 1KΩ resistor
- 1x T-type GPIO Extension Board

3.14.3 Circuit Graph



3.14.4 Program and code

Before Python programming, Install GPIO library and Python library file in Raspberry pi: open terminal, update the apt-get software installation package list (**Note:** Network must be connected), and then enter installation command to install the Raspberry gpio-python package.

Follow the next commands to complete installation, after entering each commands in terminal, press "enter" to complete:

1) Update software list command:

- `sudo apt-get update`

2) Install python command:

- `sudo apt-get install python-dev`

3) Install python-pip command (python-pip is a manager for python software package):

- `sudo apt-get install python-pip`

4) Install rpi.gpio with pip command:

- `sudo pip install rpi.gpio`

5) Testing Command:

- `sudo python`

After all steps, you will see the following photo. It means it is successful to install GPIO library and Python library file. You can begin to program this project via Python

```
pi@raspberrypi ~$ sudo python
Python 2.7.3 (default, Jan 13 2013, 11:20:46)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import RPi.GPIO as GPIO
>>>
```

Programming

You can compile code not only through connecting monitor with your Pi, also through SSH.

You can choose two methods to compile the project code.

A. File editor to compile code:

a) Create a new file "led.py" in any direction (such as "/home/pi/"): enter the command **sudo touch led.py** and then press enter

b) Open file "led.py": enter the command **sudo nano led.py** and then press enter

c) Paste the following code in the file "led.py"

```
#turns on and off a light emitting diode(LED) depending on motion sensor

import RPi.GPIO as GPIO          #importing the RPi.GPIO module
import time                      #importing the time module
GPIO.cleanup()                  #to clean up at the end of your script
led_pin = 37                    #select the pin for the LED
motion_pin = 35                 #select the pin for the motion sensor
def init():
    GPIO.setmode(GPIO.BOARD)     #to specify which pin numbering system
    GPIO.setwarnings(False)
    GPIO.setup(led_pin,GPIO.OUT)  #declare the led_pin as an output
    GPIO.setup(motion_pin,GPIO.IN,pull_up_down=GPIO.PUD_UP) #declare the motion_pin as an
input
print("_____")

def main():
    while True:
        value=GPIO.input(motion_pin)
        if value!=0:              #to read the value of a GPIO pin
            GPIO.output(led_pin,GPIO.HIGH) #turn on led
            time.sleep(2)          #delay 2ms
            print "LED on"         #print information
        else:
            GPIO.output(led_pin,GPIO.LOW)  #turn off led
            time.sleep(2)          #delay 2ms
            print "LED off"        #print information

init()
main()
GPIO.cleanup()
```

d) After compile the project, press "Ctrl" + "X" to save the code, enter "Y" to confirm saving, and press "enter" to exit the file editor.

```

File Edit View SCP Settings Help
GNU nano 2.2.6 File: led.py

import RPi.GPIO as GPIO                                #importing the RPi.GPIO module
import time                                             #importing the time module
GPIO.cleanup()                                         #to clean up at the end of your script
led_pin = 37                                           #select the pin for the LED
motion_pin = 35                                        #select the pin for the motion sensor

def init():
    GPIO.setmode(GPIO.BOARD)                          #to specify which pin numbering system
    GPIO.setwarnings(False)
    GPIO.setup(led_pin,GPIO.OUT)                      #declare the led pin as an output
    GPIO.setup(motion_pin,GPIO.IN,pull_up_down=GPIO.PUD_UP) #declare the motion pin as an input
    print("-----")

def main():
    while True:
        value=GPIO.input(motion_pin)
        if value!=0:
            GPIO.output(led_pin,GPIO.HIGH)            #to read the value of a GPIO pin
            time.sleep(2)                               #turn on led
            print "LED on"                             #delay 2s
            #print information
        else:
            GPIO.output(led_pin,GPIO.LOW)              #turn off led
            time.sleep(2)                               #delay 2s
            print "LED off"                             #print information

init()
main()
GPIO.cleanup()

```

Experiment Result:

Enter the following command:

- `cd led`
- `sudo python. ./led.py`

Press "enter" to run the project.

When detection body motion, the LED turns on, else, the LED turns off, and the monitor shows as the photo:

```

$ cd ~/$ pip install --user RPi.GPIO & sudo python led.py
led.py:5: RuntimeWarning: No channels have been set up yet - nothing to clean up! Try cleaning up at the end of your program instead
GPIO.cleanup()                                     #to clean up at the end of your script
-----
LED off
LED off
LED off
LED off
LED on
LED off
LED off
LED off
LED off
LED on
LED off
LED off

```

3.15 Analog to Digital Converter

3.15.1 Introduction



The ADC0832 chip is 8-bit successive approximation A/D converter with a serial I/O and

configurable input multiplexers with up to 8 channels. The serial I/O is configured to comply with the NSC MICROWIRE™ serial data exchange standard for easy interface to the COPS™ family of processors, and can interface with standard shift registers or μ Ps. The 8-channel multiplexer is software configured for differential inputs as well as channel assignment.

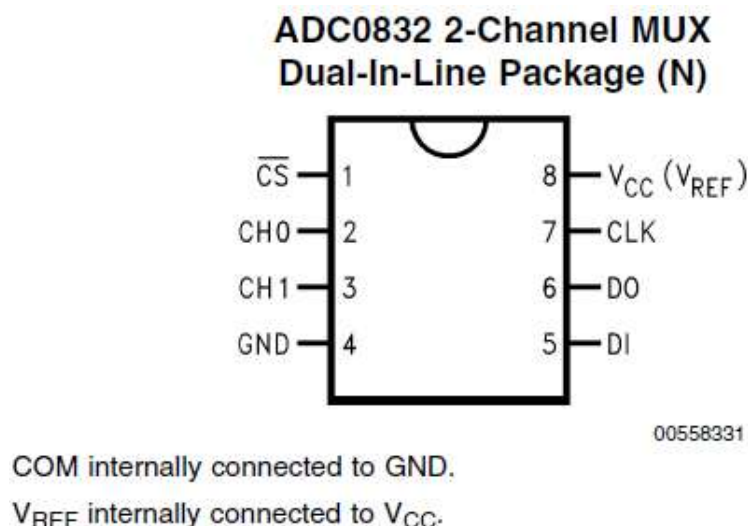
The differential analog voltage input allows increasing the common-mode rejection and offsetting the analog zero input voltage value. The voltage reference input can be adjusted to allow encoding any smaller analog voltage span to the full 8 bits of resolution.

Specification of ADC0832

- Resolution: 8 Bits
- Total Unadjusted Error: $\pm 1 / 2$ LSB and ± 1 LSB
- Single Supply: 5 VDC
- Low Power: 15 MW
- Conversion Time: 32 μ s

In this project (Use Raspberry and A/D converter to make a MP3 music player), we will use potentiometer to control the volume of music player. As Raspberry pi cannot accept analog signal from potentiometer, we need to use ADC module to convert analog signal to digital signal before sending it to Raspberry Pi.

A/D converter pin graph:

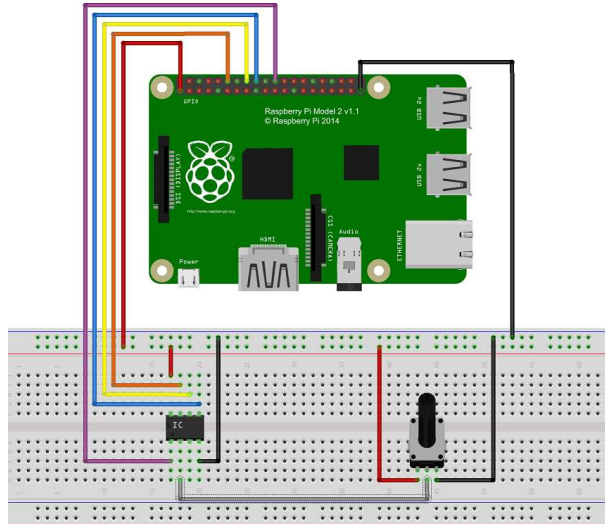


3.15.2 Hardware required

- 1x Raspberry pi
- 1x A/D converter

- 1x Potentiometer
- 11x Jumper wires
- 1x Breadboard

3.15.3 Circuit graph



3.15.4 Program and code:

1) Install python on Raspberry Pi by running following terminal commands:

- `sudo apt-get update`
- `sudo apt-get install python-dev`

```
File Edit Tabs Help
pi@raspberrypi:~$ sudo apt-get update
Get:1 http://archive.raspberrypi.org/ jessie InRelease [13.2 kB]
Get:2 http://mirrordirector.raspbian.org/ jessie InRelease [14.9 kB]
Get:3 http://mirrordirector.raspbian.org/ jessie/main armhf Packages [8,961 kB]
Get:4 http://archive.raspberrypi.org/ jessie/main armhf Packages [144 kB]
Get:5 http://archive.raspberrypi.org/ jessie/ui armhf Packages [14.2 kB]
Ign http://archive.raspberrypi.org/ jessie/main Translation-en_GB
Ign http://archive.raspberrypi.org/ jessie/main Translation-en
Ign http://archive.raspberrypi.org/ jessie/ui Translation-en_GB
Ign http://archive.raspberrypi.org/ jessie/ui Translation-en
Get:6 http://mirrordirector.raspbian.org/ jessie/contrib armhf Packages [37.5 kB]
Get:7 http://mirrordirector.raspbian.org/ jessie/non-free armhf Packages [70.3 kB]
Get:8 http://mirrordirector.raspbian.org/ jessie/rpi armhf Packages [1,356 B]
Ign http://mirrordirector.raspbian.org/ jessie/contrib Translation-en_GB
Ign http://mirrordirector.raspbian.org/ jessie/contrib Translation-en
Ign http://mirrordirector.raspbian.org/ jessie/main Translation-en_GB
Ign http://mirrordirector.raspbian.org/ jessie/main Translation-en
Ign http://mirrordirector.raspbian.org/ jessie/non-free Translation-en_GB
Ign http://mirrordirector.raspbian.org/ jessie/non-free Translation-en
Ign http://mirrordirector.raspbian.org/ jessie/rpi Translation-en_GB
Ign http://mirrordirector.raspbian.org/ jessie/rpi Translation-en
Fetched 9,276 kB in 4min 35s (33.7 kB/s)
Reading package lists... Done
pi@raspberrypi:~$ sudo apt-get install python-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libpython2.7-dev python2.7-dev
The following NEW packages will be installed:
  libpython-dev libpython2.7-dev python-dev python2.7-dev
0 upgraded, 4 newly installed, 0 to remove and 39 not upgraded.
Need to get 18.2 MB of archives.
After this operation, 25.7 MB of additional disk space will be used.
```

2) Install RPi.GPIO module by running following commands:

- `sudo apt-get install python-setuptools`

- `sudo easy_install rpi.gpio`

```
pi@raspberrypi:~$ sudo apt-get install python-setuptools
Reading package lists... Done
Building dependency tree
Reading state information... Done
python-setuptools is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 39 not upgraded.
pi@raspberrypi:~$ sudo apt-get install rpi.gpio
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'python3-rpi.gpio' for regex 'rpi.gpio'
Note, selecting 'python-rpi.gpio' for regex 'rpi.gpio'
python-rpi.gpio is already the newest version.
python3-rpi.gpio is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 39 not upgraded.
```

3) Install ALSA sound utilities and MP3 player by running following commands:

- `sudo apt-get install alsa-utils`
- `sudo apt-get install mpg321`

```
pi@raspberrypi:~$ sudo apt-get install alsa-utils
Reading package lists... Done
Building dependency tree
Reading state information... Done
alsa-utils is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 39 not upgraded.
pi@raspberrypi:~$ sudo apt-get install mpg321
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libao-common libao4 libaudio-scrobbler-perl libauthen-sasl-perl libconfig-inifiles-perl lib
  libfont-afm-perl libhtml-form-perl libhtml-format-perl libhtml-parser-perl libhtml-tagset-
  libhttp-cookies-perl libhttp-daemon-perl libhttp-date-perl libhttp-message-perl libhttp-ne
  libio-socket-ssl-perl liblist-moreutils-perl liblwp-mediatypes-perl liblwp-protocol-https-
  libnet-smtp-ssl-perl libnet-ssleay-perl liburi-perl libwww-perl libwww-robotrules-perl
Suggested packages:
  libdigest-hmac-perl libgssapi-perl libidn-perl libnet-dump-perl libcrypt-ssleay-perl libauthen-ntlm-p
The following NEW packages will be installed:
  libao-common libao4 libaudio-scrobbler-perl libauthen-sasl-perl libconfig-inifiles-perl lib
  libfont-afm-perl libhtml-form-perl libhtml-format-perl libhtml-parser-perl libhtml-tagset-
  libhttp-cookies-perl libhttp-daemon-perl libhttp-date-perl libhttp-message-perl libhttp-ne
  libio-socket-ssl-perl liblist-moreutils-perl liblwp-mediatypes-perl liblwp-protocol-https-
  libnet-smtp-ssl-perl libnet-ssleay-perl liburi-perl libwww-perl libwww-robotrules-perl mpg
0 upgraded, 31 newly installed, 0 to remove and 39 not upgraded.
Need to get 1,744 kB of archives.
After this operation, 5,314 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://mirrordirector.raspbian.org/raspbian/ jessie/main libao-common armhf 1.1.0-3 [1
Get:2 http://mirrordirector.raspbian.org/raspbian/ jessie/main libao4 armhf 1.1.0-3 [32.1 kB
Get:3 http://mirrordirector.raspbian.org/raspbian/ jessie/main liblist-moreutils-perl armhf
```

4) Download python code for this project:

- `wget http://osoyoo.com/driver/raspi-adc-pot.py`
- `chmod 777 raspi-adc-pot.py`

```
pi@raspberrypi:~$ wget http://osoyoo.com/driver/raspi-adc-pot.py
2016-10-08 17:24:53 (72.3 MB/s) - 'raspi-adc-pot.py' saved [3625/3625]
pi@raspberrypi:~$ ls
Desktop  Makefile  pic  python_games  RPi.GPIO-0.2.0.tar.gz  update6python-dev.png  Videos
Downloads  Documents  Music  Pictures  raspi-adc-pot.py  setup6gpio.png  usr  wiringPi
demo.py  Downloads  Downloads  public  RPi.GPIO-0.2.0  Templates  util6s6gpio.png
pi@raspberrypi:~$ chmod 777 raspi-adc-pot.py
pi@raspberrypi:~$
```

Run the project:

1) Upload sound module:

- `sudo modprobe snd-bcm2835`
- `sudo amixer cset numid=3 1`

```
pi@raspberrypi:~$ sudo modprobe snd-bcm2835
pi@raspberrypi:~$ sudo amixer cset numid=3 1
numid=3,iface=MIXER,name='PCM Playback Route'
; type=INTEGER,access=rw-----,values=1,min=0,max=2,step=0
; values=1
tombas@raspberrypi:~$
```

2) Download sample MP3 music files

- `wget http://osoyoo.com/driver/long_time_no_see.mp3`
- `mpg321 long_time_no_see.mp3`

```
pi@raspberrypi:~$ wget http://osoyoo.com/driver/long_time_no_see.mp3
2016-10-08 18:44:44 -- http://osoyoo.com/driver/long_time_no_see.mp3
Resolving osoyoo.com (osoyoo.com)... 158.69.116.108
Connecting to osoyoo.com (osoyoo.com)[158.69.116.108]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10023907 (9.6M) [audio/mpeg]
Saving to: 'long_time_no_see.mp3'

long_time_no_see.mp3
long_time_no_see.mp3
27%[=====] 2.57M 8.36KB/s eta 26m 55s
100%[=====] 9.56M 35.8KB/s in 30m 40s

2016-10-08 18:55:27 (5.32 KB/s) - 'long_time_no_see.mp3' saved [10023907/10023907]

pi@raspberrypi:~$ ls
default Python BMP dseio.py Downloads Makefile Pictures raspi-adc-pot.py run.png usr
lsd777.png Desktop long_time_no_see.mp3 Music Public RPI.GPIO-0.2.0 raspi-adc-pot.py snd.png Videos
pi@raspberrypi:~$ mpg321 long_time_no_see.mp3
```

3) Open another terminal window and run following command:

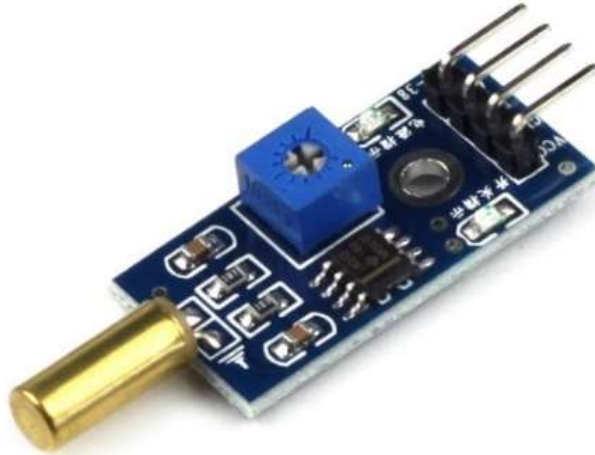
- `sudo python raspi-adc-pot.py`

```
pi@raspberrypi:~$ sudo python raspi-adc-pot.py
raspi-adc-pot.py:56: RuntimeWarning: This channel is already in use, continuing
anyway. Use GPIO.setwarnings(False) to disable warnings.
GPIO.setup(SPI0MOSI, GPIO.OUT)
raspi-adc-pot.py:58: RuntimeWarning: This channel is already in use, continuing
anyway. Use GPIO.setwarnings(False) to disable warnings.
GPIO.setup(SPI0CLK, GPIO.OUT)
raspi-adc-pot.py:59: RuntimeWarning: This channel is already in use, continuing
anyway. Use GPIO.setwarnings(False) to disable warnings.
GPIO.setup(SPI0CS, GPIO.OUT)
trim_pot: 912
pot_adjust: 912
last_read 0
trim_pot changed True
Volume = 89%
set_volume 89
tri_pot_changed 89
trim_pot: 897
pot_adjust: 15
last_read 912
trim_pot changed True
Volume = 88%
set_volume 88
tri_pot_changed 88
trim_pot: 909
pot_adjust: 12
```

4) Connect your earphone to Raspberry Pi, change volume by adjusting potentiometer and enjoy your music.

3.16 Tilt Sensor Module

3.16.1 Introduction



Specification of the tilt switch

- Operating voltage: 3.3V-5V
- Output: digital output (0 and 1)
- With fixed bolt hole for easy installation
- VCC: 3.3V-5V
- GND: GND
- OUT: digital output (0 and 1)
- Sensitivity adjustable
- Applies to various angle sensors, tilt sensors etc.

In this project (Tilt switch work with Raspberry Pi), we will read tilt sensor signal from GPIO port 20 (Pin 38) and control the status of LED. When Pin 38 get high voltage (True), we turn on LED, otherwise turn off LED. The input signal value (1 or 0) will be displayed to screen every 0.1 second.

If you don't know what GPIO layout is, check the appendix: How to read Raspberry Pi I/O pin diagram (GPIO pin graph).

3.16.2 Hardware required

- 1x Raspberry Pi 2/3/zero
- 1x 8GB MicroSD memory card preinstalled Raspbian OS.
- 1x LED

- 1x Tilt Sensor Module
- 1x 220 Ω resistance
- 1x Breadboard
- 7x Jumper wires
- 1x GPIO breakout kit(optional)

3.16.3 Prerequisite:

1) Raspbian should be upgraded to latest version in order to support RPI.GPIO module.

Please run following commands in shell:

- `sudo apt-get update`
- `sudo apt-get upgrade`

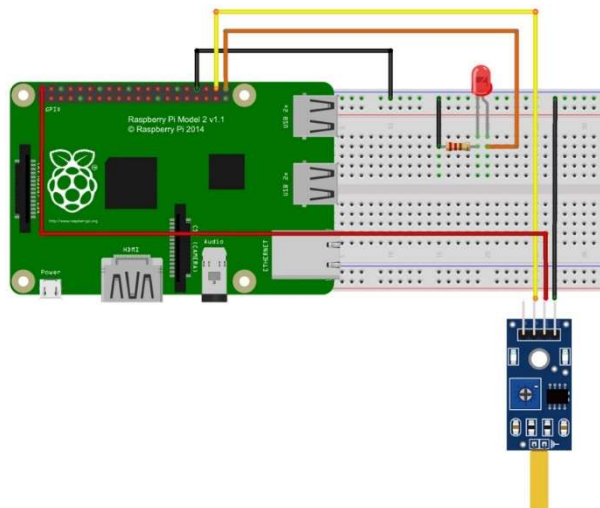
2) Enable I2C and SPI protocol to enable the protocol, run shell command

- `sudo raspi-config`

Then select Advance Options and enable I2C and SPI

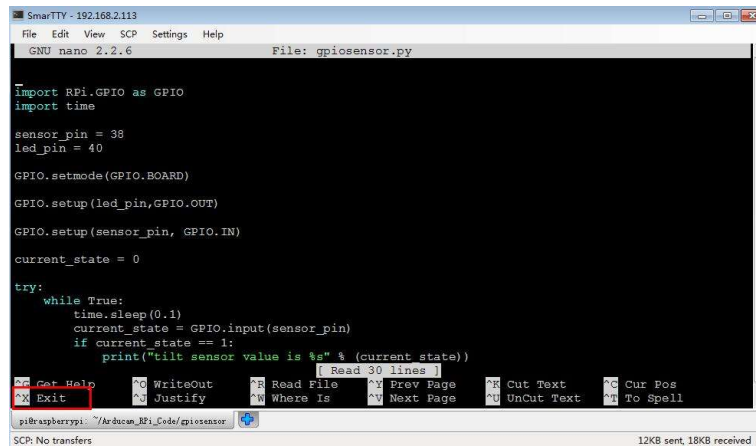
You need to reboot to effect the configuration

3.16.4 Circuit graph:



3.16.5 Program and code

You have two ways to write python code:



```

SmarTTY - 192.168.2.113
File Edit View SCP Settings Help
GNU nano 2.2.6 File: gpiosensor.py

import RPi.GPIO as GPIO
import time

sensor_pin = 38
led_pin = 40

GPIO.setmode(GPIO.BOARD)
GPIO.setup(led_pin,GPIO.OUT)
GPIO.setup(sensor_pin, GPIO.IN)

current_state = 0

try:
    while True:
        time.sleep(0.1)
        current_state = GPIO.input(sensor_pin)
        if current_state == 1:
            print("tilt sensor value is %s" % (current_state))

[Read 30 lines]
Exit Help WriteOut Read File Prev Page Cut Text Cur Pos
Exit Justify Where Is Next Page UnCut Text To Spell
pi@raspberrypi: ~/Arducon_RPi_Code/gpiosensor
SCP: No transfers 12KB sent, 18KB received

```

Run following command in shell window:

- `sudo nano gpiosensor.py`

Then copy python code and paste the code into gpiosensor.py

Then Type "Ctrl" + "X" and "Yes" to save the file. and press "enter" to exit the file editor.

Run following command in shell window:

- `sudo cd gpiosensor`
- `sudo python gpiosensor.py`

If you turn the sensor board into vertical position, the pi will turn on LED. If you put the board into horizon position, the LED will be off.

3.17 3x Potentiometer (10kilohm adjustable resistor)



Potentiometers, sometimes called pots, are relatively simple devices. One terminal of the potentiometer is connected to a power source, and another is hooked up to a ground — a

point with no voltage or resistance and which serves as a neutral reference point. The third terminal slides across a strip of resistive material. This resistive strip generally has a low resistance at one end, and its resistance gradually increases to a maximum resistance at the other end. The third terminal serves as the connection between the power source and ground, and it usually is operated by the user through the use of a knob or lever.

The user can adjust the position of the third terminal along the resistive strip to manually increase or decrease resistance. The amount of resistance determines how much current flows through a circuit. When used to regulate current, the potentiometer is limited by the maximum resistivity of the strip.

Potentiometers are commonly used to control electrical devices such as volume controls on audio equipment. Potentiometers operated by a mechanism can be used as position transducers, for example, in a joystick. Potentiometers are rarely used to directly control significant power (more than a watt), since the power dissipated in the potentiometer would be comparable to the power in the controlled load.

Specification of the potentiometer:

- Resistance: 10K Ohm
- Wheel Diameter: 16mm
- Wheel Thickness: 5.5mm
- Handle length: 15mm

The application of potentiometer is in project 14(Use Raspberry pi and A/D converter to make a MP3 music player), please check the project 14 for your reference.

3.18 Piezo Buzzer

3.18.1 Introduction



A buzzer or beeper is an audio signaling device. As a type of electronic buzzer with integrated structure, which use DC power supply, are widely used in computers, printers, photocopiers, alarms, electronic toys, automotive electronic equipment, telephones, timers and other electronic products for voice devices. Buzzers can be categorized as active and passive buzzers (See the following pictures).

When you place the pins of buzzers upward, you can see that two buzzers are different, the buzzer that green circuit board exposed is the passive buzzer.

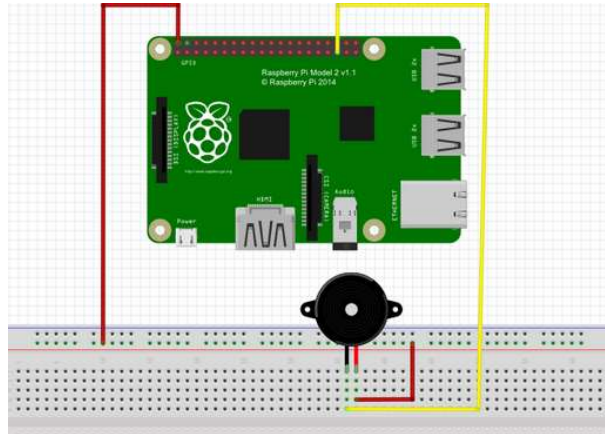
In this study, the buzzer we used is active buzzer. Active buzzer will sound as long as the power supply. We can program to make the Arduino output alternating high and low level, so that the buzzer sounds.

In this project (Using Raspberry Pi to drive buzzer), we will learn to use Raspberry Pi to drive buzzer and make alert sound repeatedly.

3.18.2 Hardware required

- 1x Raspberry Pi 3
- 1x Raspberry Pi T-style extension
- 1x Buzzer
- 1x Mouse and keyboard
- 1x HDMI cable and HDMI monitor(or LCD TV)
- 1x Breadboard

3.18.3 Circuit Graph:



3.18.4 Program and code

1) Install git core (if you have already installed git core, please skip this step). Please run following shell command in Pi terminal:

- `sudo apt-get install git-core`
- `sudo apt-get update`
- `sudo apt-get upgrade`

2) Install wiringPi library (if you have installed wiringPi, please skip this step). Please run following command in Pi terminal:

- `cd buzzer_rpi`
- `git clone git://git.drogon.net/wiringPi`
- `cd wiringPi`
- `./build`

Note: In buzzer_rpi.c file, the buzzer is connected to port no.24. However, connection graph shows the buzzer is connected to GPIO 19. This is because buzzer_rpi.c includes wiringPi library whose port number does not match GPIO number. Port 24 is actually GPIO 19. You can use GPIO readall to check GPIO port mapping (result as per following graph)

```

pi@raspberrypi:~$ gpio readall
-----PI 3-----
BCM  WP1  Name  Mode  V  Physical  V  Mode  Name  WP1  BCM
-----
2  8  SDA.1  IN    1  3  4  5V
3  9  SCL.1  IN    1  5  6  0V
4  7  GPiO. 7  IN    1  7  8  1  IN  Ta0  15  14
          0V
17  0  GPiO. 0  IN    0  11  12  0  IN  GPiO. 1  1  18
27  2  GPiO. 2  IN    0  13  14  0  IN  GPiO. 4  4  23
22  3  GPiO. 3  IN    0  15  16  0  IN  GPiO. 5  5  24
          5.3v
10  12  MOSI  IN    0  19  20  0  IN  GPiO. 6  6  25
9  13  MISO  IN    0  21  22  0  IN  GPiO. 6  6  25
11  14  SCLK  IN    0  23  24  1  IN  CE0  10  8
          0V
0  30  SDA.0  IN    1  27  28  1  IN  SCL.0  31  1
5  21  GPiO. 21  IN    1  29  30  0V
6  22  GPiO. 22  IN    1  31  32  0  IN  GPiO. 26  26  12
13  23  GPiO. 23  IN    0  33  34  0V
19  24  GPiO. 24  IN    0  35  36  0  IN  GPiO. 27  27  16
26  25  GPiO. 25  IN    0  37  38  0  IN  GPiO. 28  28  20
          0V
          39  40  0  IN  GPiO. 29  29  21
-----
BCM  WP1  Name  Mode  V  Physical  V  Mode  Name  WP1  BCM
pi@raspberrypi:~$

```

4) Compile and run the code by typing following commands:

- `gcc -Wall -o buzzer buzzer_rpi.c -lwiringPi`
- `sudo ./buzzer`

You will hear buzzer sound.

Use “Ctrl” + “c” to stop the program

3.19 GL5516 Photo resistor (Light Sensor)

3.19.1 Introduction



A photoresistor is a light-controlled variable resistor. The resistance of a photoresistor decreases with the increasing incident light intensity; in other words, it exhibits photoconductivity. A photoresistor can be applied in light-sensitive detector circuits.

A photoresistor is made of a high resistance semiconductor. In the dark, a photoresistor can have a resistance as high as a few megohms (MΩ), while in the light, a photoresistor can have a resistance as low as a few hundred ohms. If incident light on a photoresistor exceeds a certain frequency, photons absorbed by the semiconductor give bound electrons enough energy to jump into the conduction band. The resulting free electrons (and their hole partners) conduct electricity, thereby lowering resistance.

The resistance range and sensitivity of a photoresistor can substantially differ among

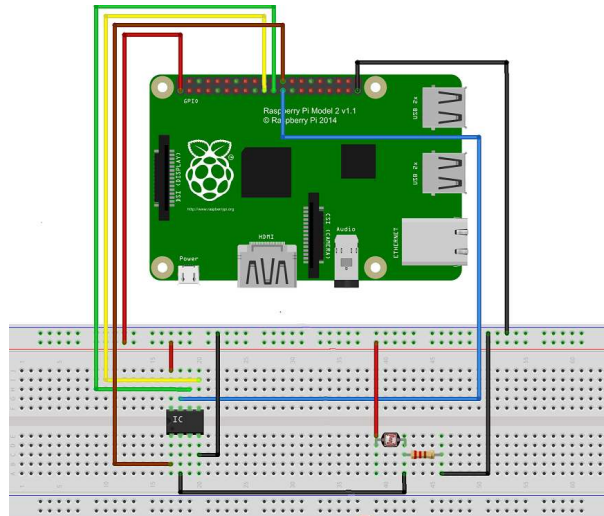
dissimilar devices. Moreover, unique photoresistor may react substantially differently to photons within certain wavelength bands.

In this project (Get light strength data with photo resistor), we will use Raspberry Pi to get light strength data with photo resistor. As Raspberry pi cannot handle analog input, we need an A/D adapter to convert analog voltage into digital signal.

3.19.2 Hardware required

- 1x Raspberry pi2/3
- 1x 8GB MicroSD memory card preinstalled Raspbian OS.
- 1x GPIO breakout kit(optional)
- 1x A/D adapter
- 1x Photo resistor
- 1x 10K ohm resistor
- 1x Breadboard
- 11x Jumper wires

3.19.3 Circuit Connection Graph



3.19.4 Install software:

1) Connect your Pi to internet and then install GIT with following terminal command:

- `sudo apt-get install git-core`

If there is any error during git installation, run following commands first, then do GIT installation again.

- `sudo apt-get update`
- `sudo apt-get upgrade`

2) Download wiringPi library installation package with GIT

- `git clone git://git.drogon.net/wiringPi`

If you need updated wiringPi, run following commands:

- `cd wiringPi`
- `git pull origin`

3) Build wiringPi library

- `cd wiringPi`
- `./build`

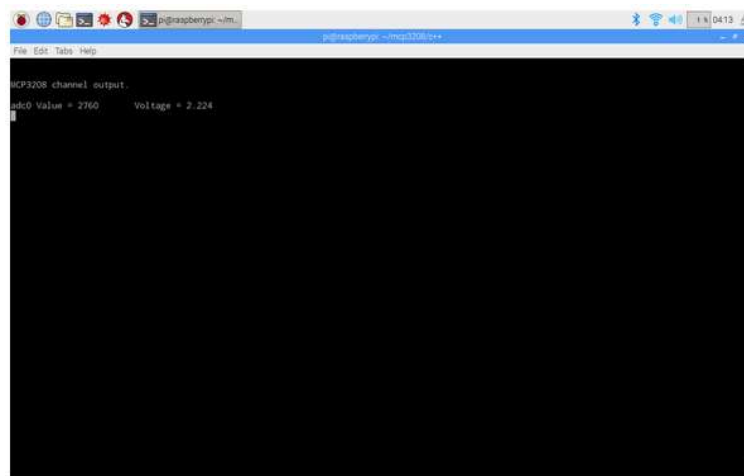
4) Compile sample code with following command:

- `cd raspi photoresistor`
- `gcc -Wall -o app raspi_photoresistor.c -lwiringPi`

5) Run the code:

- `sudo ./app`

After running the sample code, your pi terminal will show light value as following:



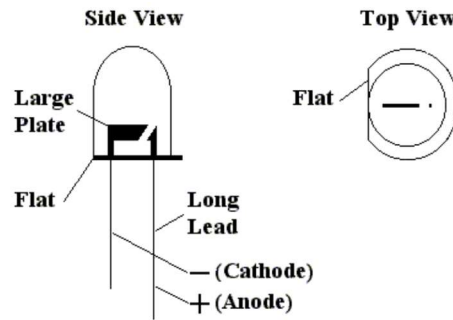
3.20. LED (6 x White, 6 x Red, 6 x Yellow, 6 x Green)



The LED is the abbreviation of light emitting diode. It is usually made of gallium arsenide, gallium phosphide semiconductor materials. The LED has two electrodes, a positive electrode and a negative electrode, it will light only when a forward current pass, and it can be red, blue, green or yellow light, etc. The color of light depends on the materials it was made.

In general, the drive current for LED is 5-20mA. Therefore, in reality it usually needs an extra

resistor for current limitation so as to protect the LED.



The application of the LED is in project 13 (Using Raspberry Pi to drive motion sensor and turn on LED), please check the project 13 for your reference.

3.21 Infrared Remote Controller

3.21.1 Introduction



Specification of the IR controller:

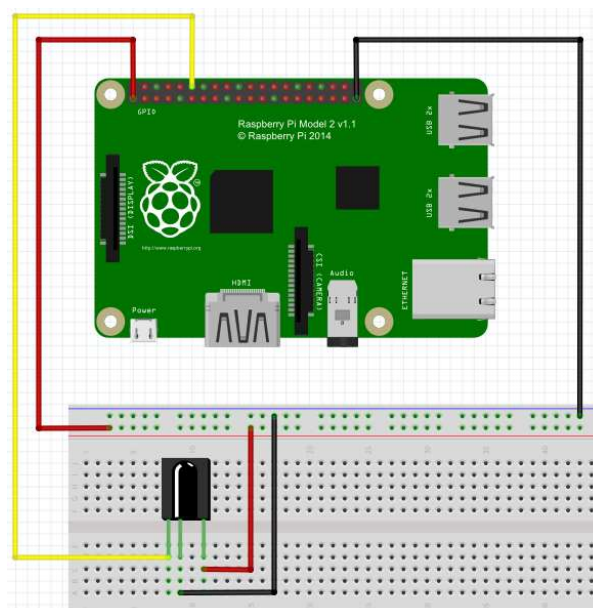
- Super three decoding -WAV + WMA + MP3
- Support TF card (cell phone memory card inside), U disk playback mode.
- Onboard 2W power amplifier sound super loud with PCB terminals
- Gold-plated headphone output interface
- Gold-plated AUX input interface
- With a button interface, microcontroller interface, making the secondary development possible.
- Micro USB 5V power supply interface, mobile power supply is simple and convenient
- Support breakpoint memory save function

In this project (Use Raspberry pi to get IR remote code), we will connect an Infrared remote control receiver to Raspberry Pi. After an IR remote controller key is pushed, Raspberry Pi will decode the signal and display the key code on the terminal.

3.21.2 Hardware required

- 1x Raspberry Pi 2/3/zero
- 1x 8GB MicroSD memory card x 1preinstalled Raspbian OS
- 1x VS1838B
- 1x Infrared Remote Controller
- 1x Breadboard
- 1x GPIO breakout kit(optional)
- 5x Jumper Wires

3.21.3 Circuit Graph



3.21.4 Program and code

1) LIRC (Linux Infrared remote control) is an open source library which allow Linux to decode IR signal

- `sudo apt-get install lirc`

Change configuration with nano command

- `sudo nano /etc/lirc/hardware.conf`

Amend following contents in /etc/lirc/hardware.conf file

- `LIRCD_ATGS=""`
- `DRIVER="default"`
- `DEVICE="/dev/lirc0"`
- `MODULES="lirc-rpi"`

Edit module file with nano

- `sudo nano /etc/modules`

Press "Ctrl" + "X" to save and exit.

2) Compile the following commands:

- `lirc-dev`
- `lirc-rpi gpio_in_pin_18 gpio_out_pin_17`

GPIO 18 pin will get input data from IR receiver

3) Restart the Raspberry pi to enable configuration:

- `reboot`

Restart lirc by running following command

- `sudo /etc/init.d/lirc restart`

Test IR receiver

1) Stop lirc by running:

- `sudo /etc/init.d/lirc stop`

2) run following code

- `mode2 -d /dev/lirc0`

After running above command, terminal might show following errors:

```

pi@raspberrypi ~$ sudo /etc/init.d/lirc stop
pi@raspberrypi ~$ mode2 -d /dev/lirc0
mode2: could not get file information for /dev/lirc0: mode2_default_initr: No such file or directory
pi@raspberrypi ~$

```

This means /lirc0 module has not be installed under /dev diretory.

To fix it, you need uncomment "#dtoverlay=lirc-rpi" in /boot/config.txt file

- `sudo nano /boot/config.txt`

find #dtoverlay=lirc-rpi and remove the "#" sign, press "Ctrl" + "X" to save and exit.

Reboot and test again with following two commands:

- `reboot`
- `mode2 -d /dev/lirc0`

If no error shows up after running above commands, you can press keys in the IR remoter control. You will see following result:

```

pi@raspberrypi ~$ mode2 -d /dev/lirc0
space 178446
pulse 9084
space 4605
pulse 610
space 526
pulse 612
space 536
pulse 612
space 529
pulse 609
space 530
pulse 607
space 528
pulse 610
space 529
pulse 611
space 526
pulse 609
space 530
pulse 584
space 1658
pulse 584
space 1667
pulse 610
space 1645
pulse 589
space 1669
pulse 590
space 1661
pulse 584
space 1669
pulse 583
space 1672
pulse 608
space 1634
pulse 618
space 528

```

Press "Ctrl" + "c" to exit

Record IR code running by following two commands:

- `sudo /etc/init.d/lirc stop`

- *irrecord -list-namespace*

1) Write down the key names as following:

- *KEY_CHANNELDOWN*
- *KEY_CHANNELUP*
- *KEY_CHANNEL*
- *KEY_PREVIOUS*
- *KEY_NEXT KEY_PLAY*
- *KEY_VOLUMEDOWN*
- *KEY_VOLUMEUP*
- *KEY_EQUAL*
- *KEY_NUMERIC_0 ~ KEY_NUMERIC_9*

2) Run IR code record command:

- *irrecord -d /dev/lirc0 ~/lircd.conf*

3) Keep press RETURN key until you see

- *"Press RETURN now to start recording"*

4) Press RETURN again, then press each key quickly as per the instruction in the screen.

Each press will show up a dot sign in the screen. After the dots fill two lines, screen will prompt:

- *"Please enter the name for the next button (press to finish recording)"*

Then press the key you want to record.

Example:

1) Enter KEY_PLAY and return, screen prompt:

- *Now hold down button "KEY_PLAY"*

2) Press and hold PLAY key in your IR controller, PI will record the key code and prompt:

- "Please enter the name for the next button (press to finish recording)"

3) Repeat the procedure until all your keys are recorded, save the config file to lircd.conf

4) Replace /etc/lirc/lircd.conf file with your lircd.conf file:

- `sudo cp ~/lircd.conf /etc/lirc/lircd.conf`

5) Restart lirc and test IR keys with following 2 commands:

- `sudo /etc/init.d/lirc start`
- `irw`

After you press keys in the IR controller, Pi will display key codes as following:

```
pi@raspberrypi:~$ sudo cp ~/lircd.conf /etc/lirc/lircd.conf
pi@raspberrypi:~$ sudo /etc/init.d/lirc start
[ ok ] Starting lirc (via systemctl): lirc.service.
pi@raspberrypi:~$ irw
000000000000000001 00 KEY_CHANNELDOWN /home/pi/lircd.conf
000000000000000003 00 KEY_CHANNEL /home/pi/lircd.conf
000000000000000002 00 KEY_CHANNELUP /home/pi/lircd.conf
000000000000000004 00 KEY_PREVIOUS /home/pi/lircd.conf
000000000000000005 00 KEY_NEXT /home/pi/lircd.conf
000000000000000006 00 KEY_PLAY /home/pi/lircd.conf
000000000000000007 00 KEY_VOLUMEDOWN /home/pi/lircd.conf
000000000000000008 00 KEY_VOLUMEUP /home/pi/lircd.conf
000000000000000009 00 KEY_EQUAL /home/pi/lircd.conf
00000000000000000a 00 KEY_NUMERIC_0 /home/pi/lircd.conf
00000000000000000b 00 KEY_NUMERIC_1 /home/pi/lircd.conf
00000000000000000c 00 KEY_NUMERIC_2 /home/pi/lircd.conf
00000000000000000d 00 KEY_NUMERIC_3 /home/pi/lircd.conf
00000000000000000e 00 KEY_NUMERIC_4 /home/pi/lircd.conf
00000000000000000f 00 KEY_NUMERIC_5 /home/pi/lircd.conf
000000000000000010 00 KEY_NUMERIC_6 /home/pi/lircd.conf
000000000000000011 00 KEY_NUMERIC_7 /home/pi/lircd.conf
000000000000000012 00 KEY_NUMERIC_8 /home/pi/lircd.conf
000000000000000013 00 KEY_NUMERIC_9 /home/pi/lircd.conf
```

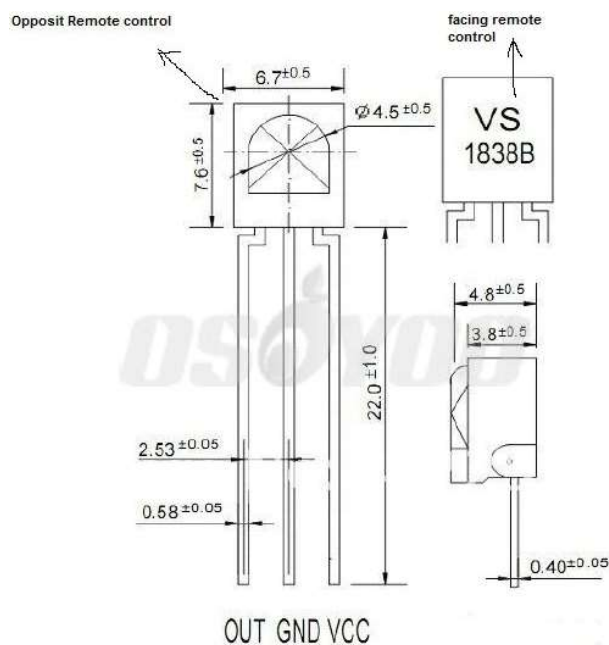
3.22 Infrared Receiver (VS1838B)



The IR receiver is used with the IR remote controller. It has high-speed high-sensitivity PIN photodiode and a low -power, high-gain preamplifier IC, plus the use of epoxy resin dry outer shield anti- scratch design.

Specification of the IR receiver:

- Low power consumption
- Wide-angle and long distance reception
- Strong anti-interference ability, able to withstand environmental disturbances
- Outside interference shielding design
- Ultra- compact design package with outer shield anti-jamming capability,
- Built-in CMOS IC
- Voltage: 2.7-5.5V
- Receiving Distance: 13-15M
- Size: pin length: 21mm

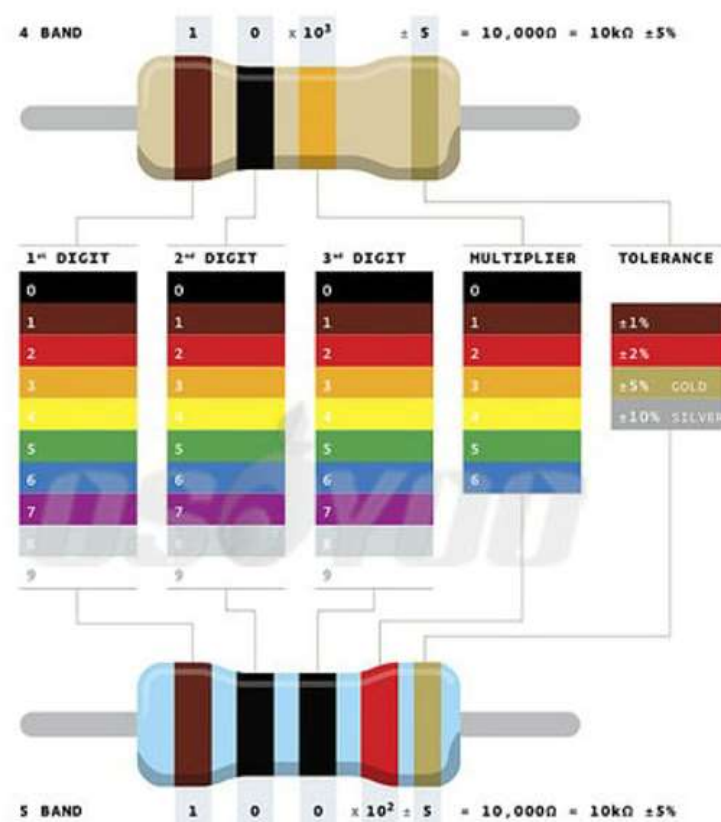


The application of the IR receiver is in project 20 (Use Raspberry pi to get IR remote code), please check the project 20 for your reference.

3.23 How to read resistor color code

Resistors values are marked using colored bands, according to a code developed in the 1920s, when it was too difficult to write numbers on such objects.

Each color corresponds to a number, like you see in the table below. Each resistors has either 4 or 5 bands. In the 4-band type, the first two bands indicate the first two digits of the value while the third one indicates the number of zeros that follow (technically it represents the power of ten). The last band specifies the tolerance: in the example below, gold indicates that the resistor value can be 10k ohm plus or minus 5%



RESISTORS INCLUDED IN THE STARTER KIT

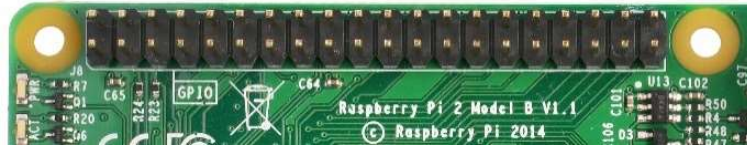
You'll find either a 4 band or
a 5 band version

			5 BAND
			4 BAND
	220Ω	560Ω	4.7kΩ
			5 BAND
			4 BAND
1kΩ	10kΩ	1MΩ	10MΩ

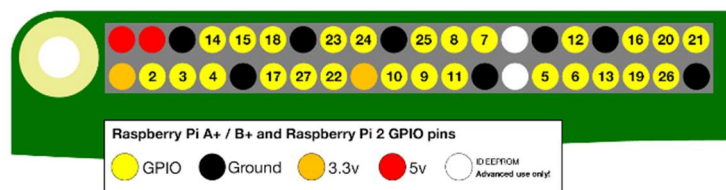
Appendix

How to read Raspberry Pi I/O pin diagram (GPIO pin graph)

Raspberry Pi I/p pins are located in the upper left corner of board, see following picture:



These pins are combination of Voltage supplies, Grounds and GPIO (general purpose input/output) pins. You can distinguish them from following graph:



Understand those pins functions are fundamental to design projects which allow Raspberry Pi to communicate with sensors and many other devices.

Detail explanation of pins:

wiringPi	BCM	Function	Physical Pin		Function	BCM	wiringPi
		3.3V	1	2	5V		
8	2	SDA.1	3	4	5V		
9	3	SCL.1	5	6	GND		
7	4	GPIO.7	7	8	TXD	14	15
		GND	9	10	RXD	15	16
0	17	GPIO.0	11	12	GPIO.1	18	1
2	27	GPIO.2	13	14	GND		
3	22	GPIO.3	15	16	GPIO.4	23	4
		3.3V	17	18	GPIO.5	24	5
12	10	MOSI	19	20	GND		
13	9	MISO	21	22	GPIO.6	25	6
14	11	SCLK	23	24	CE0	8	10
		GND	25	26	CE1	7	11
30	0	SDA.0	27	28	SCL.0	1	31
21	5	GPIO.21	29	30	GND		
22	6	GPIO.22	31	32	GPIO.26	12	26
23	13	GPIO.23	33	34	GND		
24	19	GPIO.24	35	36	GPIO.27	16	27
25	26	GPIO.25	37	38	GPIO.28	20	28
		GND	39	40	GPIO.29	21	29

- **GPIO** are your standard pins that simply be used to turn devices on and off. For example, a LED.
- **I2C** (Inter-Integrated Circuit) pins allow you to connect and talk to hardware modules that support this protocol (I2C Protocol). This will typically take up 2 pins.
- **SPI** (Serial Peripheral Interface Bus) pins can be used to connect and talk to SPI devices. Pretty much the same as I2C but makes use of a different protocol.
- **UART** (Universal asynchronous receiver/transmitter) are the serial pins used to communicate with other devices.
- **DNC** stands for do not connect, this is pretty self-explanatory. The power pins pull power directly from the Raspberry Pi.
- **GND** are the pins you use to ground your devices. It doesn't matter which pin you use as they are all connected to the same line.

Thanks

This is all of the introduction and user guide of the Raspberry Pi starter kit. Thanks for purchasing or interest in our products. Hoping the information above will be helpful to you.

We are always striving to provide our customers with the best products and services.

If you have any problem of our products or question about electronics, or you want to learn more innovative projects, please feel free to contact us. We will try our best to support you.

Hoping we can keep good business relationships. Have a nice day, friends!

Website: <http://www.uctronics.com>

Email: sales@uctronics.com

Tel: +86 025 84271192