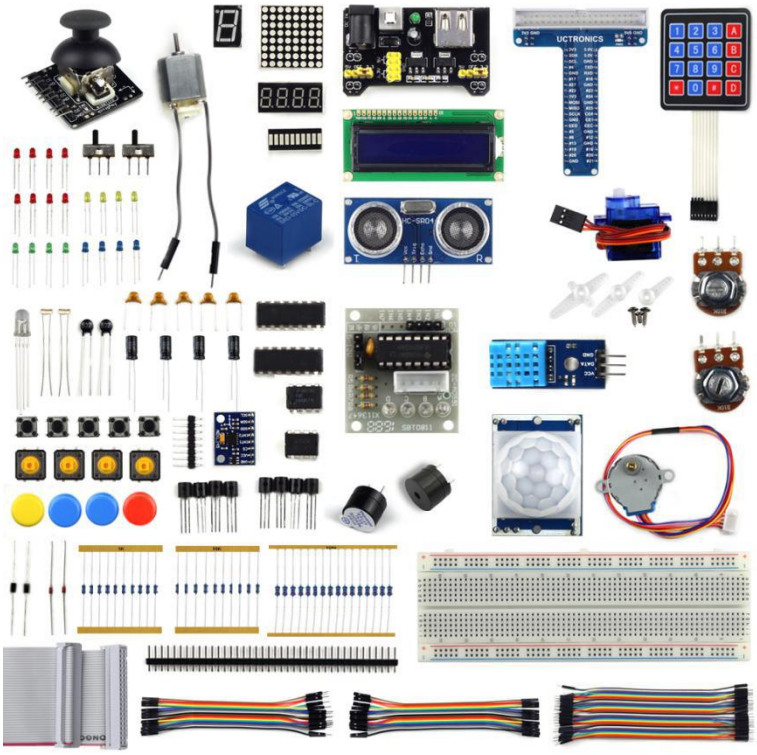
UCTRONICS Ultimate Starter Kit for Raspberry Pi

#K0064

User Guide

Table of Contents

1. Introduction



UCTRONICS designs, manufactures and provides technical service for open source (Arduino and Raspberry Pi) hardware and software. We are dedicated to applying the Internet and the latest industrial technology in open source area, we strive to provide hardware support and software service for general makers and electronic enthusiasts around the world.

This is an Ultimate Starter Leaning Kit for Raspberry Pi, the kit contains more than 50 kinds of different electronic components, more than 180 components are included.

With this kit, you are able to do more experiment, getting more idea into real action without the restriction of hardware and software. What' s more, we have carefully prepared a guide book (guide book of PDF) and supporting experimental code routine which includes a total of 26 experiments.

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.

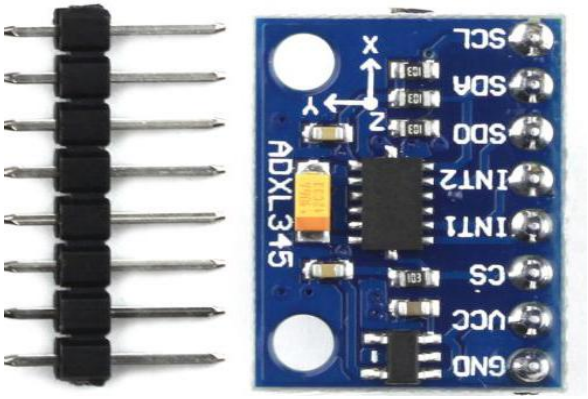
If you need to buy any electronic components, please feel free to view www.uctronics.com.

2. Kit contents

Item	Qty	Item	Qty	Item	Qty
ADXL345 Triaxial Accelerometer Sensor Module	1	74HC595	2	Resistor (10kΩ)	10
DHT-11 Digital Temperature & Humidity Sensor	1	L9110 motor driver	1	Photoresistor	2
HC-SR04 Ultrasonic Distance Sensor Module	1	DC Motor	1	Thermistor	2
Pyroelectric Infrared PIR Motion Sensor Module	1	4-bit 7-segment Display	1	1N4148 Diode	2
PS2 Joystick Module	1	LED Bar Graph Display	1	1N4001 Diode	2
LCD1602	1	Dot-matrix display	1	PNP Transistor (8550)	4
ULN2003 Stepper Motor Driver Board	1	RGB LED	1	NPN Transistor (8050)	4
SG90 micro small servo motor	1	Red LED	8	Capacitor (104)	5
DC 5V 4 Phase Step Stepper Motor	1	Green LED	4	Capacitor (10uF)	4
4x4 Matrix Array 16 Key Keyboard	1	Yellow LED	4	M to M Jumper Wires	40
T Type 40pin GPIO Board	1	Blue LED	4	M to F Jumper Wires	20
20cm FC40 40pin GPIO cable	1	Button (large)	4	F to F Jumper Wires	20
830 Tie Point Solderless Breadboard	1	Button (small)	5	Header (40pin)	1
MB102 Solderless Breadboard Power Supply	1	Switches	2	Relay	1
B10K Potentiometer	2	Button cap (red)	1	Resistor (220Ω)	16
Active Buzzer	1	Button cap (yellow)	1	Resistor (1kΩ)	10
Passive Buzzer	1	Button cap (blue)	2		

3. Assembly

3.1 ADXL345 Triaxial Accelerometer Sensor Module



ADXL345 is a compact, slim, low-power triaxial accelerometer that allows high-resolution (13-bit) measurements for accelerations up to 16 g. The digital output data is a 16-bit twos complement format and can be accessed via a SPI (3-wire or 4-wire) or i2c digital interface.

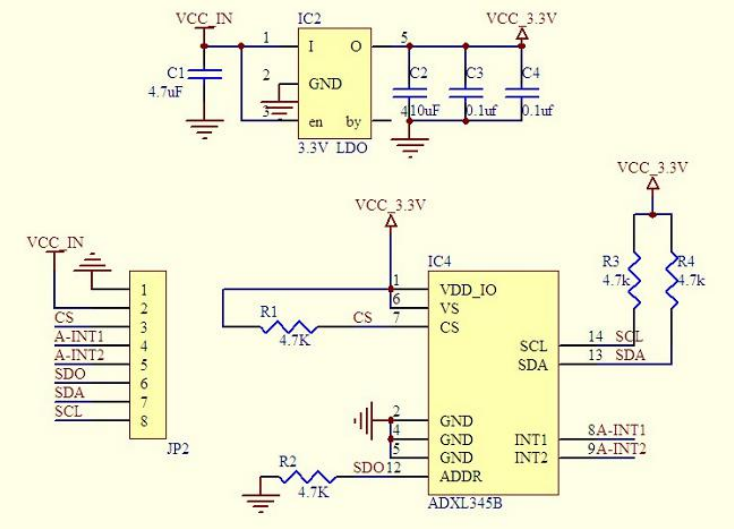
3.1.1 Specification

- Communication: I2C/SPI
- Measuring Ranging: ±2g±16g
- Digital Interface: SPI / I2C
- Power supply: 5V/3V
- Size: 3x5×1 mm
- Low Power Consumption
- Compact accelerometer

3.1.2 Application

- Mobile Device
- The measurement of the static acceleration of gravity in tilt-sensing applications
- The measurement of dynamic acceleration resulting from motion or shock

3.1.3 Schematic



3.2 DHT-11 Digital Temperature & Humidity Sensor

This DHT11 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 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 micro-controller, offering high quality, fast response, anti interference ability and cost-effectiveness.

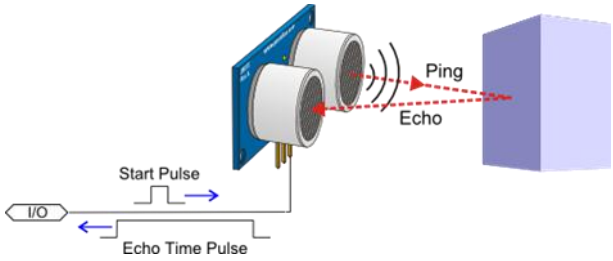
3.2.1 Specification

- VCC: 3.3V~5V
- GND: Ground

- DATA: MCU IO port
- Digital signal output
- Tips: Do not reverse the VCC and GND, otherwise it will burn

3.3 HC-SR04 Ultrasonic Distance Sensor Module

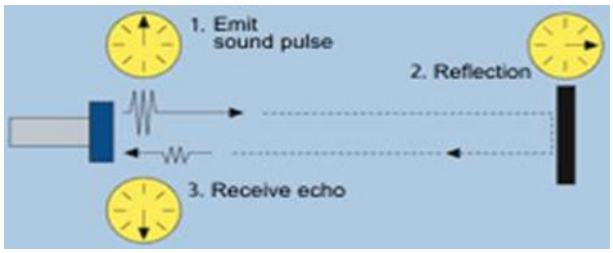
The HC-SR04 ultrasonic sensor module for Arduino is used for obstacle detection. Ultrasonic sensor transmits the ultrasonic waves from its sensor head and again receives the ultrasonic waves reflected from an object.



Ultrasonic sensor general diagram

3.3.1 Working Principle

The ultrasonic sensor emits the short and high frequency signal. These propagate in the air at the velocity of sound. If they hit any object, then they reflect back echo signal to the sensor. The ultrasonic sensor consists of a multi vibrator, fixed to the base. The multi vibrator is combination of a resonator and vibrator. The resonator delivers ultrasonic wave generated by the vibration. The ultrasonic sensor actually consists of two parts; the emitter which produces a 40kHz sound wave and detector detects 40 kHz sound wave and sends electrical signal back to the micro-controller.



Ultrasonic working principle

The ultrasonic sensor enables the robot to virtually see and recognize object, avoid obstacles, measure distance. The operating range of ultrasonic sensor is 2cm to 450cm.

3.3.2 Specification

- Working Voltage : 5V (DC)
- Static current: < 2mA
- Output signal: Electric frequency signal
- Output Voltage: 0~5V
- Sensor angle: <= 15 degrees.
- Detection distance: 2~450cm
- High precision: Up to 0.3cm
- Input trigger signal: 10us TTL impulse
- Echo signal : output TTL PWL signal
- Mode of connection: 1. VCC 2. trig(T) 3. for echo(R) 4. GND
- Using method:

  (1) Supply module with 5V

  (2) The output will be 5V while obstacle in range, otherwise it will be 0V

3.3.3 Application

- Automatic change over’ s of traffic signals
- Intruder alarm system
- Counting instruments access switches parking meters
- Back sonar of automobiles

3.4 Pyroelectric Infrared PIR Motion Sensor Module

PIR sensors respond to heat and can be triggered by animals such as cats and dogs, as well as people and other heat sources. The ‘output’ pin of Passive Infrared (PIR) sensor will go HIGH when the motion has been detected.

3.4.1 Specification

- Detector range: <=120° of vertebral angle within 7 meters
- Detector angle: 100°
- Working voltage range: 5V~20V DC
- Working temperature: -15~+70℃
- Trigger Mode: TTL Voltage output
- Level Output: high 3V/ low 0V
- Level Holding Time: 3s
- Quiescent current: <50μA
- PCB Dimension: 32x24mm
- Trigger: L cannot be repeated trigger/H can be repeated trigger (Default repeated trigger).

3.4.2 Application

- Security Products
- Human body sensors toys
- It can automatically and quickly open various types of incandescent, fluorescent lamps,

buzzer, automatic doors, electric fans, automatic washing machine and dryer Machines and other devices.

3.5 PS2 Joystick Module

A joystick is an input device consisting of a stick that pivots on a base and reports its angle or direction to the device it is controlling. A joystick, also known as the control column, is the principal control device in the cockpit of many civilian and military aircraft, either as a center stick or side-stick. It often has supplementary switches to control various aspects of the aircraft's flight.

3.5.1 Specification

- Support both the output digital value and the output analog value
- Port Definition:

  (1) DO for digital output

  (2) AO for analog output
- Output:

  (1) Two channels simulated output

  (2) one digital output
- X, Y output are separately for two potentiometer which can read out the twist Angle through the AD transform

3.5.2 Application

- Video games controlling
- Machines controlling such as cranes, trucks, surveillance cameras
- Mobile device

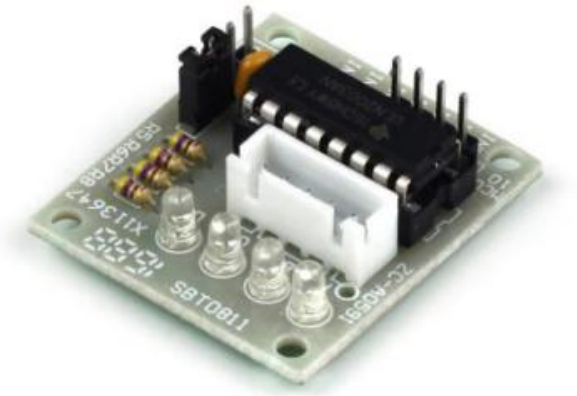
3.6. LCD1602

LCD1602 is a kind of character LCD display. The LCD display has a parallel interface, meaning that the micro-controller has to manipulate several interface pins at once to control the display.

3.6.1 Specification

- Voltage: 5V DC
- LCD display type: Characters
- Dimension: 80x36x9mm
- LCD display module with blue blacklight
- Wide viewing angle and high contrast
- Built-in industry standard HD44780 equivalent LCD controller
- 2-lines X 16-characters display

3.7 ULN2003 Stepper Motor Driver Board

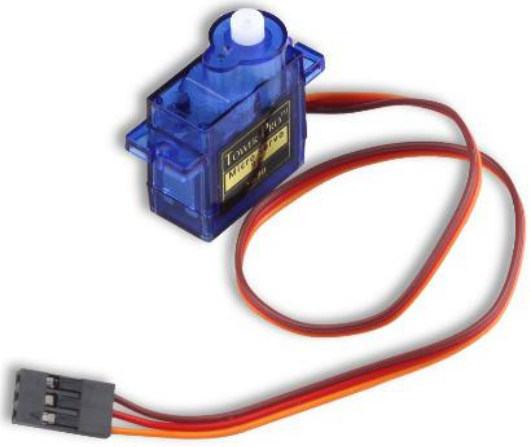


There are four LEDs on the module. The white socket in the middle is for connecting a stepper motor. IN1, IN2, IN3, IN4 are used to connected with the Raspberry Pi.

3.7.1 Specification

- White socket in the middle is for connecting a stepper motor. IN1, IN2, IN3, IN4 are used to connected with the Raspberry Pi.
- ULN2003 stepper motor driver chip with high-power
- A, B, C, D four-phase LED indicates the working status of the stepper motor
- Stepper motor with a standard interface, when used directly pluggable

3.8 SG90 micro small servo motor

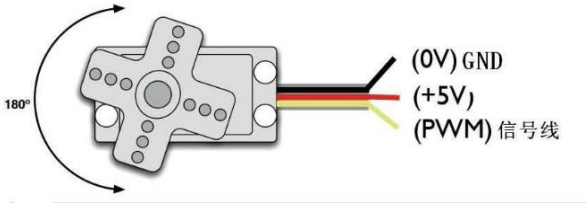


SG 90 micro small Servo motor is a type of geared motor that can only rotate 180 degrees.It consists of shell, circuit board, non-core motor, gear and location detection. It is controlled by sending pulses signal from your microcontroller. These pulses tell the servo what position

it should move to.

3.8.1 Specification

- Torsional moment: 1.5kg/cm
- Working voltage: 4.2~6V
- Temperature range : 0℃~-55℃
- Operating speed: 0.1 seconds /60°
- Dead band width: 10 microseconds
- Size: 23x12.2x29 mm
- Weight: 9g



3.9 DC 5V 4 Phase Step Stepper Motor

Stepper motors, due to their unique design, can be controlled to a high degree of accuracy without any feedback mechanisms. The shaft of a stepper, mounted with a series of magnets, is controlled by a series of electromagnetic coils that are charged positively and negatively in a specific sequence, precisely moving it forward or backward in small "steps".

3.9.1 Specification

- Phase : 4

- Current : 92mA
- Resistance : 130Ω
- Voltage : 5V DC
- No-Load Pull-Out Frequency : 800bps
- No-Load Pull-In Frequency : 500bps
- Pull-In Torque : ≥ 78.4mN.m
- Wiring Instruction : A (Blue), B (Pink), C (Yellow), D (Orange), E (Red, Mid-Point)

3.10 4x4 Matrix Array 16 Key Keyboard

This 16-button keypad provides a useful human interface component for micro-controller projects. The convenient adhesive backing provides a simple way to mount the keypad in varies applications.

3.10.1 Specification

- Connector:

  (1) Dupont 8 pins

  (2) 0.1" (2.54mm) Pitch
- Mount Style: Self-Adherence
- Max Circuit Rating:

  (1) 35V DC

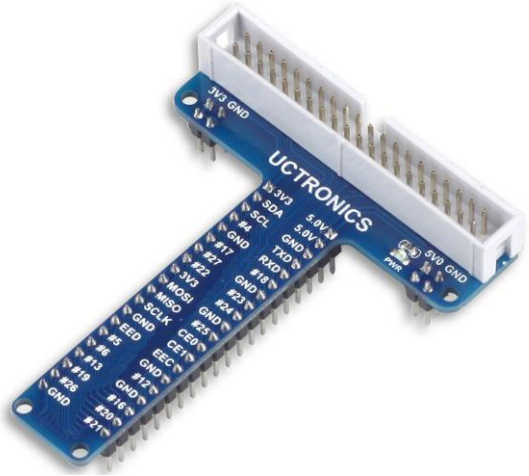
  (2) 100mA
- Insulation :

  (1) 100M Ohm

(2) 100V

- Dielectric Withstand: 250VRms (60Hz, 1min)
- Contact Bounce: <=5ms
- Life Expectancy: 1 million closures
- Operation Temperature: -20~ +40℃
- Pad Size: 68x7x0.8mm
- Cable Length: 85 mm (include connector)

3.11 T Type 40pin GPIO Board



T Type 40pin GPIO Board is a small board that connects to the 40-pin GPIO connector on the Raspberry Pi and breaks the pins out to breadboard-friendly arrangement and spacing. It is in a fancy T-shape, which is not as compact, but is much easier to read the labels.

3.11.1 Specification

- Break out all those tasty power, GPIO, I2C and SPI pins from the 40-pin header onto a solderless breadboard
- This GPIO expansion board is in a fancy T-shape, which is not as compact, but it is a little easier to read the labels

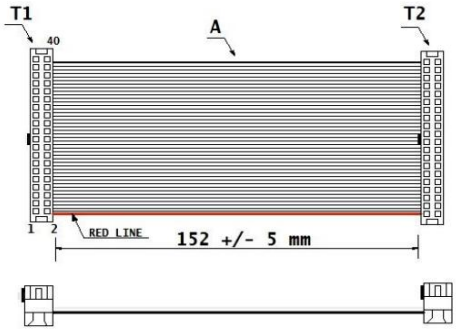
3.12 20cm FC40 40pin GPIO cable

This 2x20 flat ribbon-cable fits the GPIO headers on the Raspberry Pi 3, 2, and B+, so you can easily connect to the low-level peripherals.

You can plug the 40-pin GPIO cable between the Pi computer and the T Type Plus GPIO expansion board.

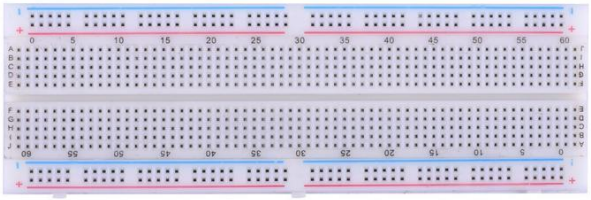
3.12.1 Specification

- Cable Length: 20cm
- 40Pin FC cable
- Female to Female
- Dimension:



ITEM	DESCRIPTION	QUANTITY
A	L/W PVC UL2651 #28 40P PITCH 2.54mm	1
T1	IDC F-C2-40 2.54mm	1
T2	IDC F-C2-40 2.54mm	1
W	152±5mm	

3.13 830 Tie Point Solderless Breadboard



This is true full size solderless breadboard! It has 2 split power buses, 10 columns, and 63 rows - with a total of 830 tie in points. All pins are spaced by a standard 0.1". The two sets of five columns are separated by about 0.3", perfect for straddling a DIP package over. The board accepts wire sizes in the range of 20-29AWG.

3.13.1 Specification

- Voltage: 300V
- Current: 3A~5A
- ABS Housing
- Solderless Breadboard with 830 Tie-points
- Material: ABS plastic material, Phosphor bronze nickel plated spring clips
- Accepts a variety of wire sizes (20~29 AWG)
- Dimension: 16.5x5.4x0.9cm
- Colored coordinates for easy component placement

3.14 MB102 Solderless Breadboard Power Supply Module

3.14.1 Specifiction

- Power supply: 5V, 3.3V

- Input voltage: 6.5~12 V (DC) USB power supply
- Output voltage: 3.3V/5V
- Maximum output current: <700 mA
- Fluctuation two road independent control, can switch to 0V, 3.3V, 5V
- On-board two groups of 3.3V,5V DC output plug pin

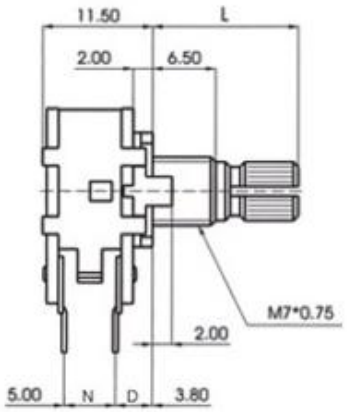
3.15 B10K Potentiometer

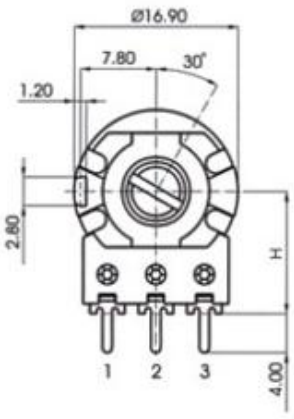
A potentiometer, informally a pot, is a three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider. The potentiometer is essentially a voltage divider used for measuring electric potential (voltage).

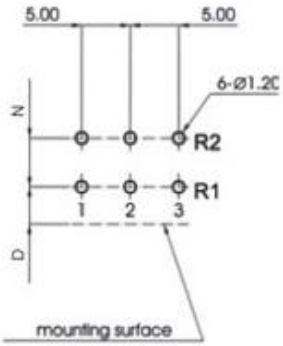
3.15.1 Specification

- Resistance value : 10K Ohm
- Adjustment Type : top adjustment
- Shaft Length: 15mm

3.15.2 Dimension







	PH1	PH2	PH3
H	12.5	18	22
N	5.0	6.0	6.0
D	3.8	3.5	3.5

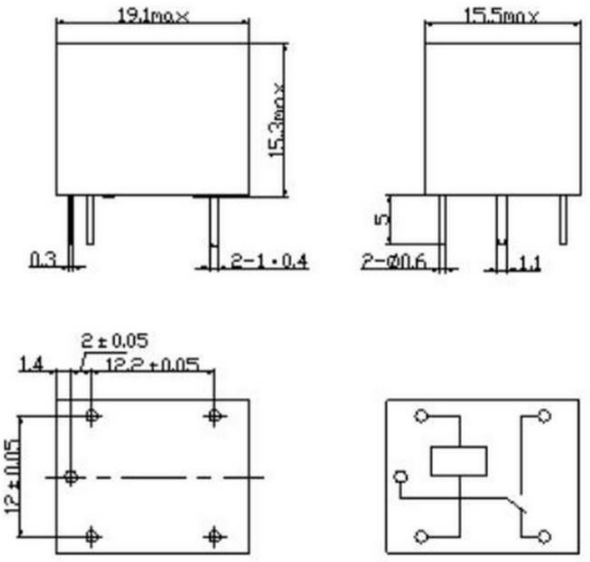
3.16 Relay

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.

3.16.1 Specification

- Coil voltage: 5V
- Coil resistance: 70~80ohm
- Pull current: 43~46mA
- Release current: 15~18mA
- Coil pin: 3,5
- Normally open contact: 24
- Normally closed contact: 14

3.16.2 Pin Definition:



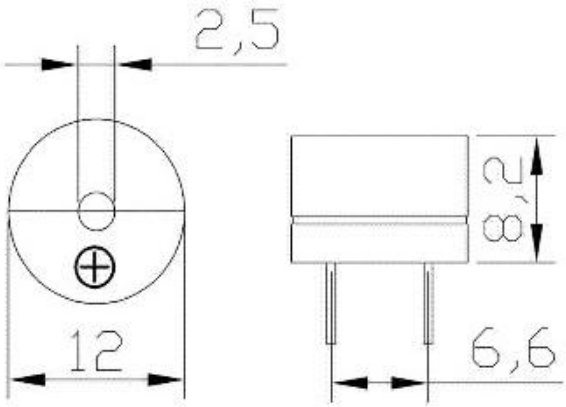
3.17 Buzzer

A buzzer or beeper is an audio signaling device. An active buzzer will generate a tone using an internal oscillator, so all that is needed is a DC voltage. A passive buzzer requires an AC signal to make a sound. It is like an electromagnetic speaker, where a changing input signal produces the sound, rather than producing a tone automatically.

3.17.1 Passive buzzer

Specification:

- Resistance: 16Ω
- Dimension:



3.17.2 Active Buzzer

Specification:

- Voltage: 4V~8V DC
- Maximum current: 30MA/5V DC
- Minimum sound pressure: 85db/10cm
- Resonant frequency: 2300+~-300HZ
- Working temperature: -20~70℃

3.18 Chips

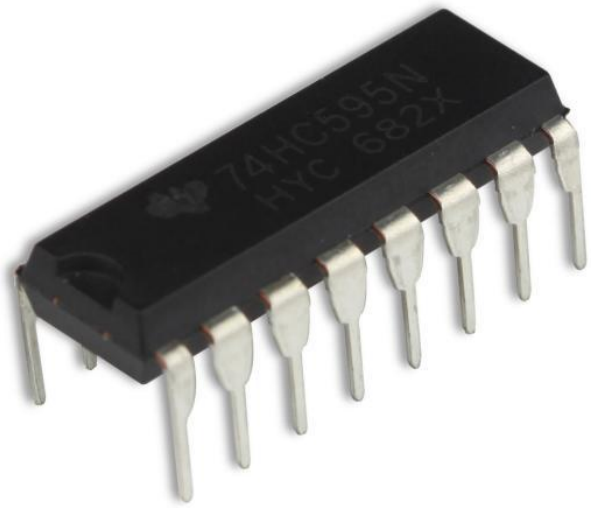
3.18.1 ADC0832

Specification:

- Resolution: 8Bits

- Total Unadjusted Error: ±0.5LSB and ±1LSB
- Single Supply: 5V DC
- Low Power: 15mW
- Conversion Time: 32μs

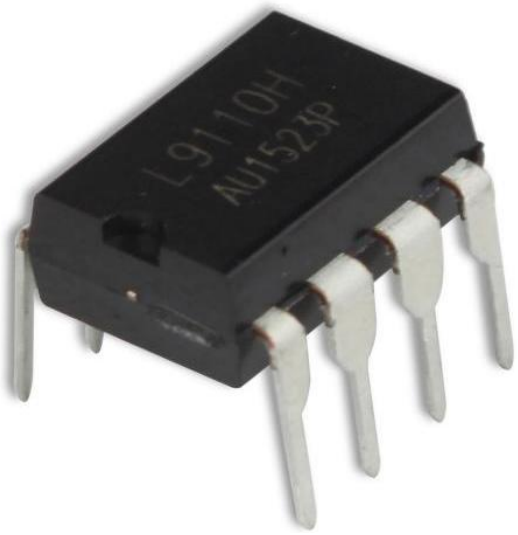
3.18.2 74HC595



Specification:

- Supply voltage: 2V~6V
- Pin number: 16
- Working temperature sensitivity: -40℃
- Operating temperature: 85℃
- Chip label: 74HC595
- Maximum output current: 7.8mA
- Logic chip function: 8bit Shift Register (3-State)

3.18.3 L9110 motor driver



Specification：

- Supply voltage: 2.5V~12V
- Output capability: 800mA continuous current per channel
- Operating temperature: 0℃~80℃
- TTL / CMOS output level compatible, and can be directly connected to the CPU
- Output built-in clamp diodes for inductive load
- Integrated control and drive into a monolithic IC
- With pin high-voltage protection function

3.19 DC Motor

A DC motor is any of a class of electrical machines that converts direct current electrical power into mechanical power.

3.19.1 Specification

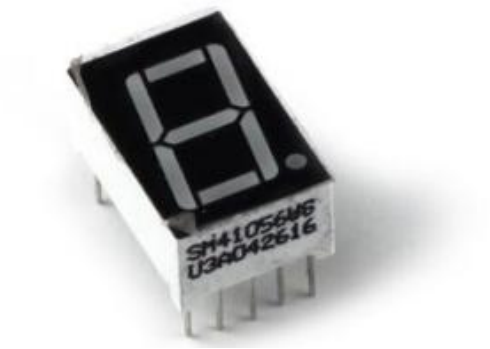
- Length: 25mm
- Diameter: 21mm
- Axis diameter: 2mm
- Axis length: 9mm
- Voltage: 3V~6V

3.20 Displays

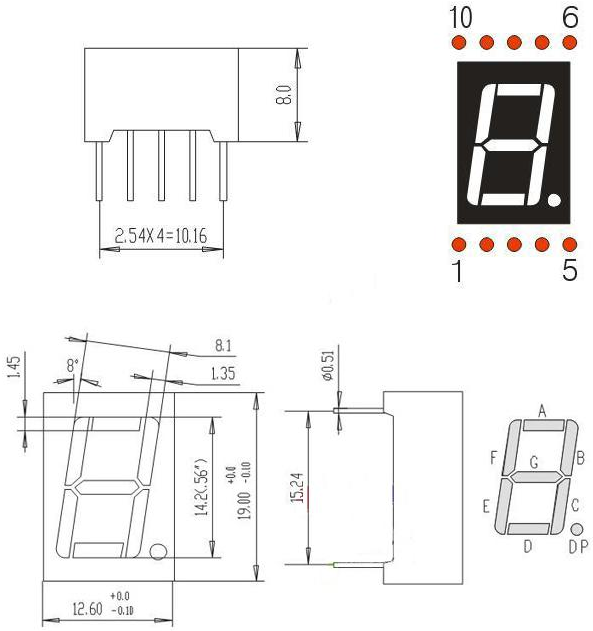
3.20.1 Segment Display

The segment display is a form of electronic display device for displaying decimal numerals that is an alternative to the more complex dot matrix displays.

(1) 7-segment display

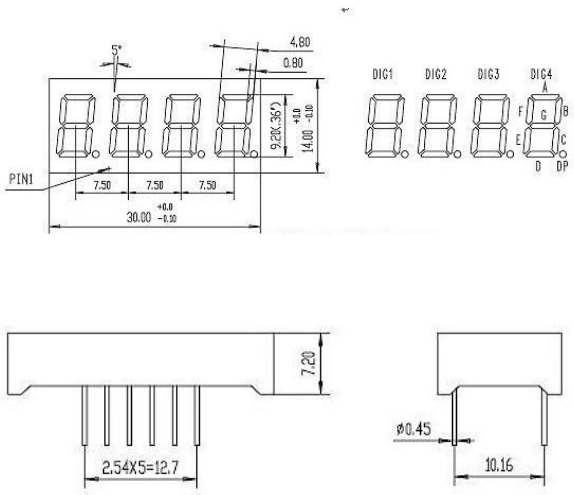


Dimension:

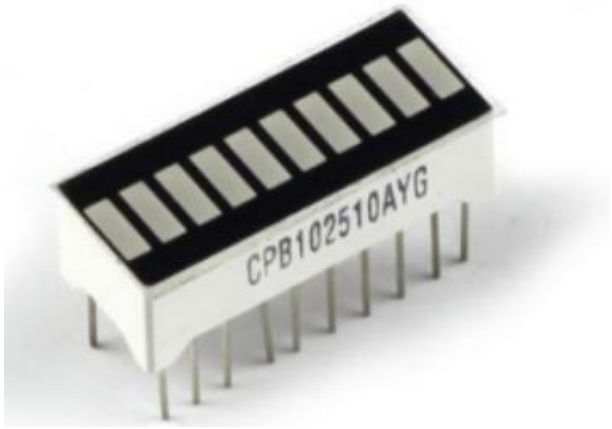


(2) 4-bit 7-segment Display

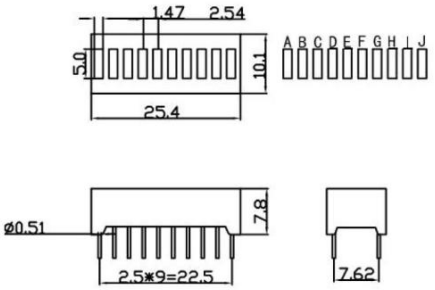
Dimension:



3.20.2 LED Bar Graph Display

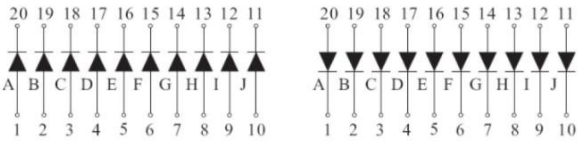


Dimension:



UNIT: MM(INCH) TOLERANCE: ±0.25(0.01")

Internal Circuit Diagram



1025101A **1025101B**

3.20.3 Dot-matrix Display

The display consists of a dot matrix of lights or mechanical indicators arranged in a rectangular configuration. A dot matrix controller converts instructions from a processor into signals which turns on or off lights in the matrix so that the required display is produced. It is a display device used to display information on machines, clocks, railway departure indicators and many other devices requiring a simple display device of limited resolution.

Specification:

- Emitted Color: Red
- Type: Common anode
- Size: 32x32x13mm
- Pin length: 8mm
- Led Size: 3mm

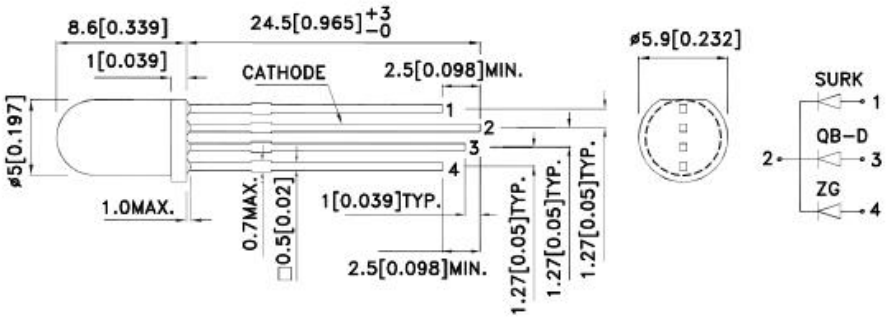
3.21 LEDs

3.21.1 RGB LED

The Hyper Red source color devices are made with AlGaInP on GaAs substrate Light Emitting Diode. The Blue source color devices are made with InGaNLight Emitting Diode.

The Green source color devices are made with InGaN on Sapphire Light Emitting Diode. Static electricity and surge damage the LEDS.

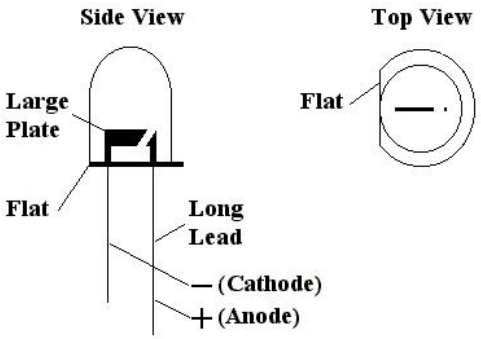
It is recommended to use a wrist band or anti-electrostatic glove when handling the LEDs. All devices, equipment and machinery must be electrically grounded.



3.21.2 Red, Green, Yellow, Blue LEDs

Light Emitting Diode (LED) LEDs are solid state devices that emit light when electricity passes through them. They are directional and need to go in the circuit the right way around.

Once a certain "on" threshold voltage is reached (about 2V for a red LED) the current through an LED rises very quickly with the voltage. In most applications, a resistor is needed to protect the LED from being overloaded due to this effect.

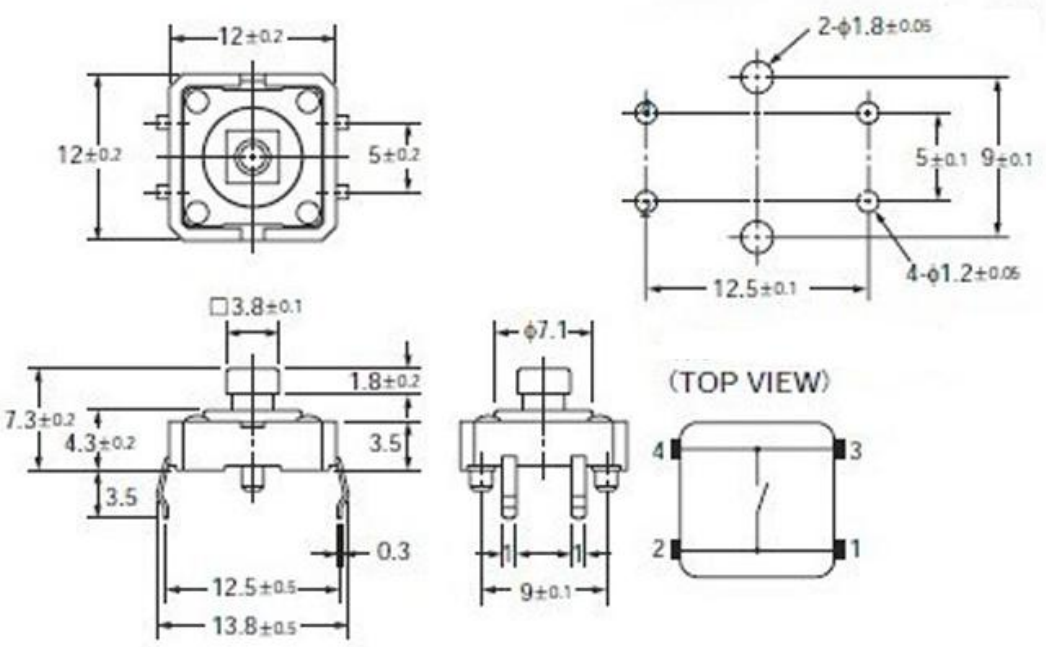


3.22 Push Button, Switches and Caps

3.22.1 Button (large)

Buttons are a common component used to control electronic devices. They are usually used as switches to connect or disconnect circuits. Although buttons come in a variety of sizes and shapes.

Dimension:



3.22.2 Button (small)

Specification

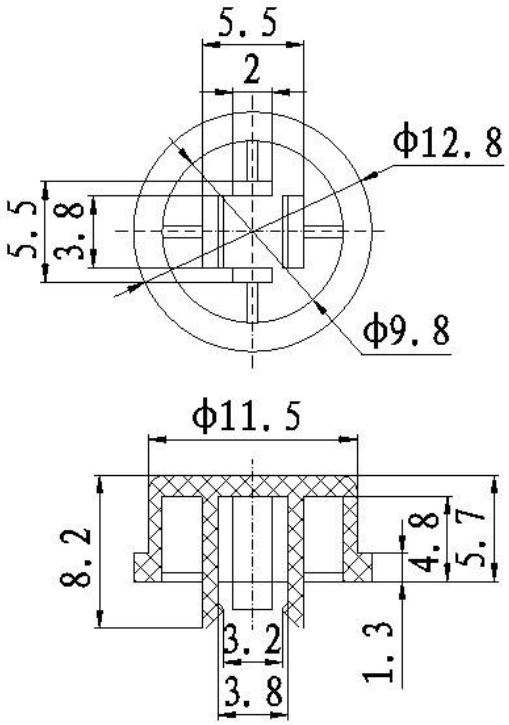
- Size: 6x6x5mm
- color: Black
- Pin number: 4
- Temperature：-25°~+-85C°
- Rated Load：DC 12V, 0.1A
- Contact Resistance：<=0.03Ω
- Withstand Voltage：AC250 V (50Hz) /MIN
- Actuation Force：1.3+-0.5N
- Lnsulation Resistance：>=100MΩ
- Life：100000 times

3.22.3 Button cap

Specification:

- Diameter: 9.58mm
- Height: 5.1mm

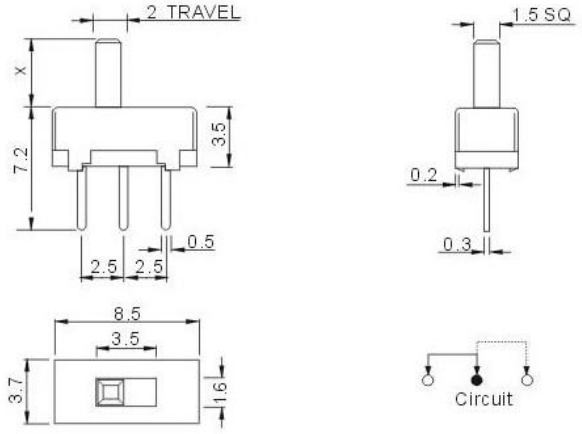
- Compatible with switch Size: 12x12x7.3mm
- Dimension:



3.23.4 Switch

Specification:

- Rate: DC 50V/0.5A
- Contact Resistance: 30moh
- Operating Life: 10,000cycles
- Dimension:



3.23 220 , 1K and 10K Ohm Resistors

Resistors limit the flow of electricity through part of the circuit. This can be used to control timing circuits, divide voltages in to smaller portions or protect devices that are sensitive to too much current.

Resistance is measured in Ohms (Ω), kilohms (1,000Ω=1kΩ) or Megohms (1,000,000Ω =1MΩ)

Resistors can go in the circuit either way round.

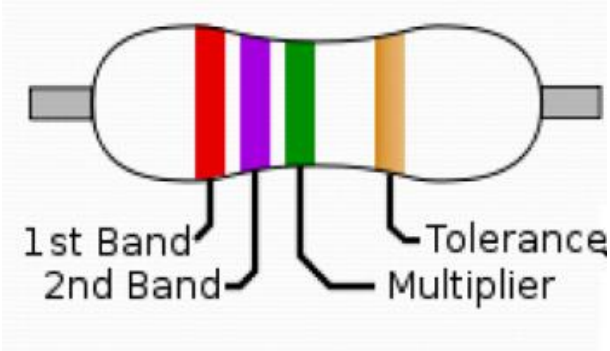
The value is marked on the device with a color code (see right), or it can be measured with almost any multimeter.

A shorthand is often used to write the value of a resistor. For example: "100R" =100Ω "10k" =10kΩ "4k7" =4.7kΩ

Some resistors used in these experiments: 220R – Red, Red, Brown 1k – Brown, Black, Red 10k – Brown, Black, Orange 100k – Brown, Black, Yellow.

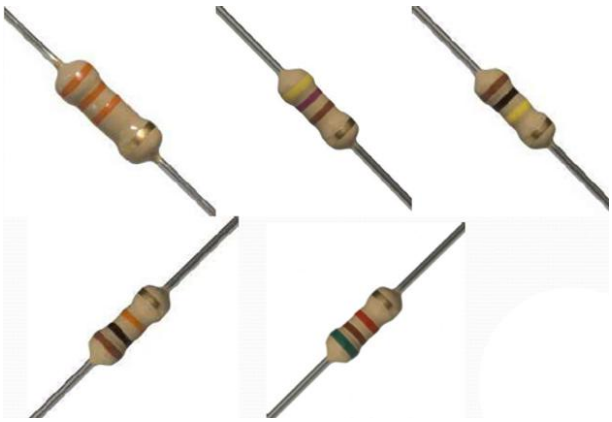
3.23.1 Specification:

- Resistor color codes:
- 1st band = 1st number
- 2nd band = 2nd number
- 3rd band = 3rd number
- 4th band = #of zeros/multiplier
- 5th band = tolerance



3.23.2 Tolerance:

- Gold = within 5%
- Black: 0
- Brown: 1
- Red: 2
- Orange: 3
- Yellow:4
- Green: 5
- Blue: 6
- Violet: 7



- Gray: 8
- White: 9

3.24 Light Sensor (Photoresistor)

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.

3.24.1 Specification:

- Maximum Voltage: 150V DC
- Maximum Wattage: 100mW
- Operating Temperature: -30~+70℃
- Spectral Peak: 540nm
- Bright Resistance (10Lux): 10~20 KΩ
- Dark resistance: 1 KΩ
- Response time:

  (1) Rise: 20ms

  (2) Down: 30ms

3.25 Analog Temperature Sensor (Thermistor)

A thermistor is a type of resistor whose resistance varies significantly with temperature, When the temperature increases, the thermistor resistance decreases; when the temperature decreases, the thermistor resistance increases. It can detect surrounding temperature changes in real time.

3.25.1 Specification:

- Resistance: 5Kohm + / -1%
- Standard tolerances: ±5%, ±10%, ±20%
- B Value tolerance: ±5%
- Measuring power: ≤0.5mW
- Dissipation Constant: ≥6.0mW/℃
- Time Constant: ≤30 seconds
- Rated Power: 0.5W
- Temperature Range: -55℃~125℃

3.26 Diode

A diode is a two-terminal electronic component that conducts primarily in one direction (asymmetric conductance); it has low (ideally zero) resistance to the current in one direction, and high (ideally infinite) resistance in the other.

3.26.1 1N4148 Diode

3.26.1.1 Specification

- Normal Forward Current: 150mA
- Maximum Forward Current: 500mA
- Maximum Repetitive Peak Current: 450mA
- Maximum Repetitive Peak Voltage: 100V
- Maximum Continuous Reverse Voltage: 75V
- Maximum Forward Voltage: 1V
- Maximum Reverse Recovery Time: 4ns
- Maximum Power: 500mW
- Diameter: 1.85mm
- External Length (height): 4.25mm
- Maximum Junction Temperature: 200℃

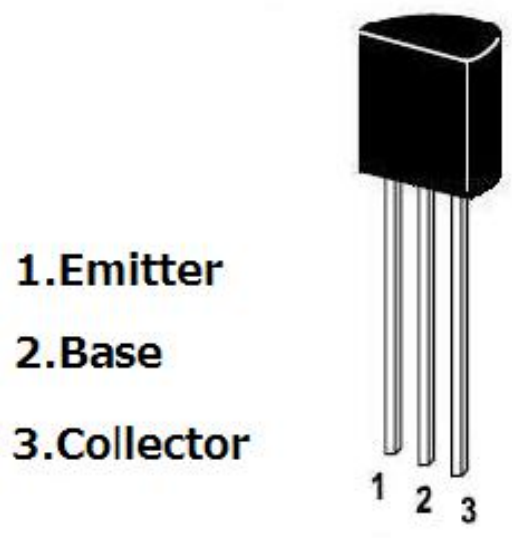
3.26.2 1N4001 Diode

3.26.2.1 Specification

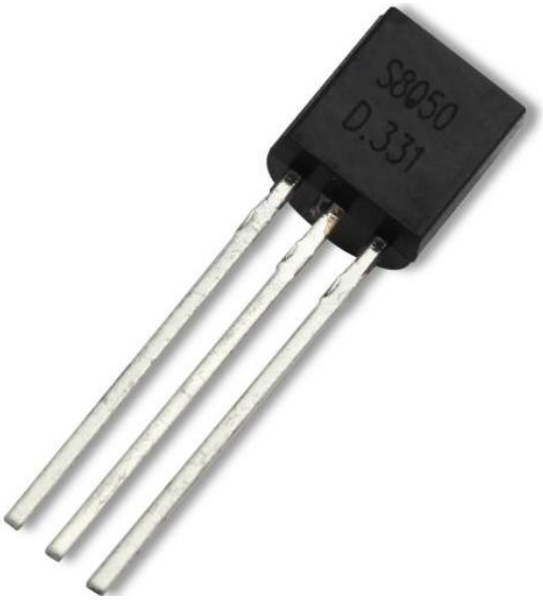
- Maximum Peak Repetitive Reverse Voltage: 50V
- Maximum Working Peak Reverse Voltage: 50V
- Maximum DC Blocking Voltage: 35V
- Average rectified output Current: 1V
- Maximum Typical junction Capacitance: 15pF
- Maximum Typical Thermal Resistance Junction to Ambient: 100 K/W
- Maximum DC Blocking Voltage temperature: +150C°
- Operating and Storage Temperature Range: -65~150C°
- Weight: 0.30 g

3.27 Transistor

A transistor is a semiconductor device used to amplify or switch electronic signals and electrical power. It is composed of semiconductor material usually with at least three terminals for connection to an external circuit. A voltage or current applied to one pair of the transistor's terminals controls the current through another pair of terminals. Because the controlled (output) power can be higher than the controlling (input) power, a transistor can amplify a signal. And the transistor is divided into two kinds, one is NPN, for instance, such as S8050, another one is PNP transistor such as the S8550.



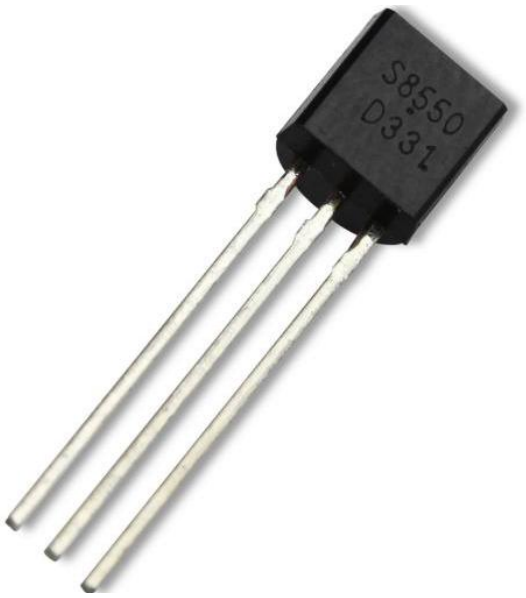
3.27.1 NPN Transistor (8050)



3.27.1.1 Specification

- Maximum Collector-base Voltage: 40V
- Maximum Collector-Emitter Voltage: 25V
- Maximum Emitter-Base Voltage: 6V
- Maximum Collector Current: 1.5A
- Maximum Junction Temperature: 150C°
- Maximum Storege Temperature: -65~150C°

3.27.2 PNP Transistor (8550)



3.27.2 Specification

- Maximum Collector-base Voltage: -40V
- Maximum Collector-Emitter Voltage: -25V
- Maximum Emitter-Base Voltage: -6V
- Maximum Collector Current: -1.5A
- Maximum Junction Temperature: 150C°
- Maximum Storege Temperature: -65~150C°

3.28 Capacitor

A capacitor is a passive two-terminal electrical component that stores electrical energy in an electric field. The effect of a capacitor is known as capacitance. While capacitance exists between any two electrical conductors of a circuit in sufficiently close proximity, a capacitor is specifically designed to provide and enhance this effect for a variety of practical applications by consideration of size, shape, and positioning of closely spaced conductors, and the intervening dielectric material.

3.28.1 Capacitor (104)

3.28.1.1 Specification

- Operating Temperature: -25℃~+85℃
- Capacitance: 104PF
- Rated Voltage: 50V

3.28.2 Capacitor (10uF)

3.28.2 Specification

- Rated Voltage: 50V
- Capacitance: 10μF
- Capacitance Tolerance: ±20%
- Dimensions (without pins): 12x5mm

3.29 Jumper Wires

A jump wire (also known as jumper, jumper wire, jumper cable, DuPont wire, or DuPont cable – named for one manufacturer of them) is an electrical wire or group of them in a cable with a connector or pin at each end (or sometimes without them – simply "tinned"), which is normally used to interconnect the components of a breadboard or other prototype or test circuit, internally or with other equipment or components, without soldering.

3.29.1 40x Male to Male Jumper Wires

3.29.2 20x Male to Female Jumper Wires

3.29.3 20x Female to Female Jumper Wires

3.30 Header (40pin)

A pin header (often abbreviated as PH, or simply header) is a form of electrical connector.

4. Raspberry Pi Pin Number Introduction

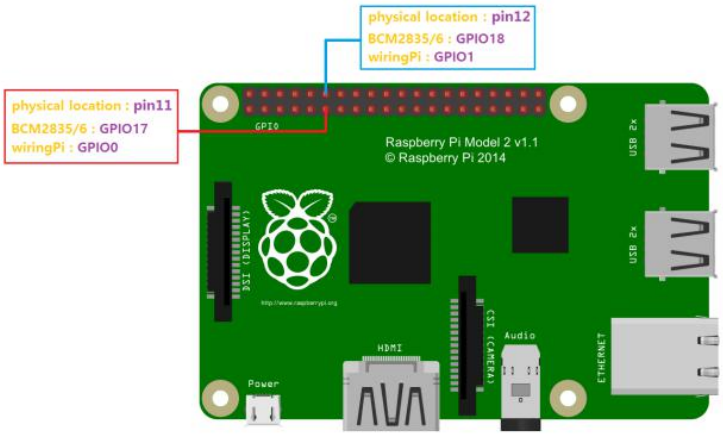
WiringPi Pin	BCM GPIO	Name	Header	Name	BCM GPIO	WiringPi Pin
–	–	3.3v	1 | 2	5v	–	–
8	R1:0/R2:2	SDA1	3 | 4	5v	–	–
9	R1:1/R2:3	SCL1	5 | 6	0V	–	–
7	4	GPIO7	7 | 8	TXD	14	15
–	–	0V	9 | 10	RXD	15	16
0	17	GPIO0	11 | 12	GPIO1	18	1
2	R1:21/R2:27	GPIO2	13 | 14	0V	–	–
3	22	GPIO3	15 | 16	GPIO4	23	4
–	–	3.3v	17 | 18	GPIO5	24	5
12	10	MOSI	19 | 20	0V	–	–
13	9	MISO	21 | 22	GPIO6	25	6
14	11	SCLK	23 | 24	CE0	8	10
–	–	0V	25 | 26	CE1	7	11
30	0	SDA0	27 | 28	SCL0	1	31
21	5	GPIO21	29 | 30	0V	–	–
22	6	GPIO22	31 | 32	GPIO26	12	26
23	13	GPIO23	33 | 34	0V	–	–
24	19	GPIO24	35 | 36	GPIO27	16	27
25	26	GPIO25	37 | 38	GPIO28	20	28
		0V	39 | 40	GPIO29	21	29
WiringPi Pin	BCM GPIO	Name	Header	Name	BCM GPIO	WiringPi Pin

There are three methods for numbering Raspberry Pi' s GPIO:

1. Numbering according to the physical location of the pins, from left to right, top to bottom, the left is odd, the right is even.

2. Numbering according the GPIO registers of BCM2835/2836 SOC.

3. Numbering according the GPIO library wiringPi.



5. Raspberry Pi GPIO Library Introduction

Currently, there are two major GPIO libraries for Raspberry Pi, RPi.GPIO and wiring Pi.

RPi.GPIO:

RPi.GPIO is a python module to control Raspberry Pi GPIO channels. For more information about RPi.GPIO, please visit:

https://pypi.python.org/pypi/RPi.GPIO/

For examples and documentation, please visit:

http://sourceforge.net/p/raspberry-gpio-python/wiki/Home/

The RPi.GPIO module is pre-installed in the official Raspbian operating system, you can use it directly.

wiringPi:

The wiringPi is a GPIO access library written in C for the BCM2835/6 SOC used in the Raspberry Pi. It' s released under the GNU LGPLv3 license and is usable from C and C++ and many other languages with suitable wrappers. It' s designed to be familiar to people who have used the Raspberry Pi "wiring" system. For more information about wiringPi, please visit : http://wiringpi.com/

Install wiringPi:

Step 1 : Get the source code

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

Step 2 : Compile and install

```
$ cd wiringPi
$ git pull origin
$ sudo ./build
```

Press Enter, the script "build" will automatically compile wiringPi source code and then install it to the Raspberry Pi.

Next, verify whether the wiringPi is installed successfully or not:

wiringPi includes a command-line utility gpio which can be used to program and setup the GPIO pins. You can use this to read and write the pins and even use it to control them from shell scripts.

You can verify whether the wiringPi is installed successfully or not by the following commands:

$ sudo gpio -v

```
pi@raspberrypi ~ $ sudo gpio -v
gpio version: 2.26
Copyright (c) 2012-2015 Gordon Henderson
This is free software with ABSOLUTELY NO WARRANTY.
For details type: gpio -warranty

Raspberry Pi Details:
  Type: Model 2, Revision: 1.1, Memory: 1024MB, Maker: Sony
pi@raspberrypi ~ $
```

$ sudo gpio readall

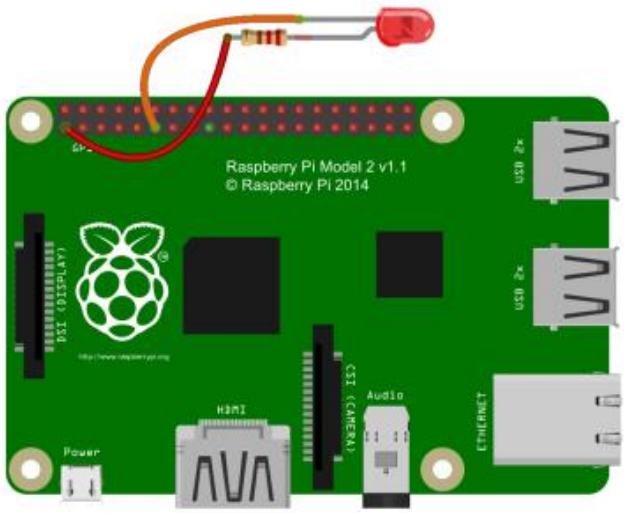
```
pi@raspberrypi ~ $ sudo gpio readall
 +-----+-----+---------+------+---+---Pi 2---+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
 |   2 |   8 |   SDA.1 | ALT0 | 1 |  3 || 4  |   |      | 5V      |     |     |
 |   3 |   9 |   SCL.1 | ALT0 | 1 |  5 || 6  |   |      | 0v      |     |     |
 |   4 |   7 | GPIO. 7 |   IN | 1 |  7 || 8  | 1 | ALT0 | TxD     | 15  | 14  |
 |     |     |      0v |      |   |  9 || 10 | 1 | ALT0 | RxD     | 16  | 15  |
 |  17 |   0 | GPIO. 0 |   IN | 0 | 11 || 12 | 0 | IN   | GPIO. 1 | 1   | 18  |
 |  27 |   2 | GPIO. 2 |   IN | 0 | 13 || 14 |   |      | 0v      |     |     |
 |  22 |   3 | GPIO. 3 |   IN | 0 | 15 || 16 | 0 | IN   | GPIO. 4 | 4   | 23  |
 |     |     |    3.3v |      |   | 17 || 18 | 0 | IN   | GPIO. 5 | 5   | 24  |
 |  10 |  12 |    MOSI | ALT0 | 0 | 19 || 20 |   |      | 0v      |     |     |
 |   9 |  13 |    MISO | ALT0 | 0 | 21 || 22 | 0 | IN   | GPIO. 6 | 6   | 25  |
 |  11 |  14 |    SCLK | ALT0 | 0 | 23 || 24 | 1 | ALT0 | CE0     | 10  | 8   |
 |     |     |      0v |      |   | 25 || 26 | 1 | ALT0 | CE1     | 11  | 7   |
 |   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 | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+---Pi 2---+---+------+---------+-----+-----+
pi@raspberrypi ~ $
```

If you can see the information shown above, it indicates that the wiringPi has been installed successfully.

6. How to use the wiringPi and the RPi.GPIO

Here we take a blinking LED for example to illustrate how to use the wiringPi C library and the RPi.GPIO Python module.

Step 1 : Build the circuit according to the following schematic diagram



Note : Resistance=220Ω

For Python user:

Step 2 : Create a file named led.py

$ sudo touch led.py

```
pi@raspberrypi /home $ ls
pi
pi@raspberrypi /home $ sudo touch led.py
pi@raspberrypi /home $ ls
led.py  pi
pi@raspberrypi /home $
```

Step 3 : Open the file led.py with vim or nano

$ sudo vim led.py

Write down the following source code, then save and exit.

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

Led = 11    # pin11

def setup():
    GPIO.setmode(GPIO.BOARD)    # Numbering according to the physical location
    GPIO.setup(Led, GPIO.OUT)   # Set pin mode as output
    GPIO.output(Led, GPIO.HIGH) # Output high level(+3.3V) to off the led

def loop():
    while True:
        print '...led on'
        GPIO.output(Led, GPIO.LOW)  # led on
        time.sleep(0.5)
        print 'led off...'
        GPIO.output(Led, GPIO.HIGH) # led off
        time.sleep(0.5)

def destroy():
    GPIO.output(Led, GPIO.HIGH)     # led off
    GPIO.cleanup()                  # Release resource

if __name__ == '__main__':     # Program start from here
    setup()
    try:
        loop()
    except KeyboardInterrupt:  # Press 'Ctrl+C' to end the program
        destroy()
```

Step 4 : Run the program

$ sudo python led.py

```
pi@raspberrypi /home $ ls
led.py  pi
pi@raspberrypi /home $ sudo python led.py
...led on
led off...
...led on
led off...
...led on
led off...
```

Press Enter, you should see that the LED is blinking. Press ' Ctrl+C' , the program execution will be terminated.

Resources :

http://sourceforge.net/p/raspberry-gpio-python/wiki/Examples/

http://wiringpi.com/reference/

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: https://github.com/UCTRONICS/Arducam_Starter_Kit_Python_Code_for_RPi.git

7. Project contents

NO.	Project	NO.	Project
1	Blinking LED	14	Measure the distance
2	Active Buzzer	15	Temperature&Humidity Sensor—DHT-11
3	Passive Buzzer	16	Dot-matrix display
4	Controlling an LED with a button	17	Photoresistor
5	Relay	18	Thermistor
6	LED Flowing Lights	19	LED Bar Graph
7	Breathing LED	20	Controlling an LED through LAN
8	Controlling a RGB LED by PWM	21	Movement Detection Based on PIR
9	7-segment display	22	DC Motor
10	4-digit 7-segment display	23	How to control a servo
11	LCD1602	24	How to control a stepper motor
12	A Simple Voltmeter	25	How to use the acceleration sensor ADXL345
13	Matrix Keyboard	26	PS2 Joystick

Table 2 Project contents

8. Project details

8.1 Project 1: LED blinking

8.1.1 Overview

In this tutorial, we will start the journey of learning Raspberry Pi. In the first lesson, we will learn how to control an LED.

8.1.2 Requirement

- Raspberry Pi ×1
- 220Ω Resistor ×1
- LED ×1
- Breadboard ×1
- Jumper Wires ×2

8.1.3 Principle

In this lesson, we will program the Raspberry Pi to output high(+3.3V) and low level(0V), and then make an LED which is connected to the Raspberry Pi GPIO flicker with a certain frequency.

1. What is LED ?

Please refer to Chapter 3.21.2 Red, Green, Yellow, Blue LEDs

2. What is the resistor ?

Please refer to Chapter 3.23 220 , 1K and 10K Ohm Resistors

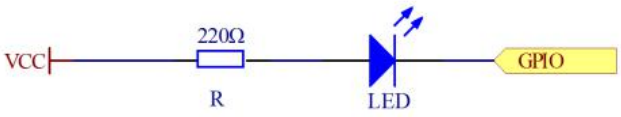
3. Working principle

The main function of the resistor is to limit current. In the circuit, the character 'R' represents resistor, and the unit of resistor is ohm(Ω).

The band resistor is used in this experiment. A band resistor is one whose surface is coated with some particular color through which the resistance can be identified directly.

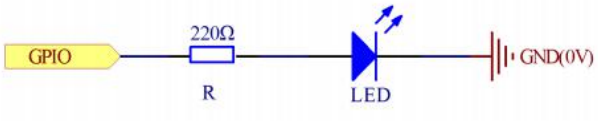
There are two methods for connecting an LED with Raspberry Pi GPIO:

①



As shown in the schematic diagram above, the anode of LED is connected to VCC(+3.3V), and the cathode of LED is connected to the ground (GND). When the GPIO output high level, the LED is on; when the GPIO output low level, the LED is off.

②



As shown in the schematic diagram above, the anode of LED is connected to Raspberry Pi GPIO via a resistor, and the cathode of LED is connected to the ground GND). When the GPIO output high level, the LED is on; when the GPIO output low evel, the LED is off.

The size of the current-limiting resistor is calculated as follows: 5~20mA current is required to make an LED on, and the output voltage of the Raspberry Pi GPIO 3.3V, so we can get the resistance:

R = U / I = 3.3V / (5~20mA) = 165Ω~660Ω

In this experiment, we choose a 220ohm resistor.

The experiment is based on method ①, we select pin 11 of Raspberry Pi to control an LED. When the pin 11 of Raspberry Pi is programmed to output low level, then the LED will be lit, next delay for the amount of time, and then programmed the pin 11 to high level to make the LED off. Continue to perform the above process, you can get a blinking LED.

4. Key functions:

Python user:

- GPIO.setmode(GPIO.BOARD)

There are two ways of numbering the IO pins on a Raspberry Pi within RPi.GPIO. The first is using the BOARD numbering system. This refers to the pin numbers on the P1 header of the Raspberry Pi board. The advantage of using this numbering system is that your hardware will always work, regardless of the board revision of the RPi. You will not need to rewire your

connector or change your code.

The second numbering system is the BCM(GPIO.BCM) numbers. This is a lower level way of working - it refers to the channel numbers on the Broadcom SOC. You have to always work with a diagram of which channel number goes to which pin on the RPi board. Your script could break between revisions of Raspberry Pi boards.

- GPIO.setup(channel, mode)

This sets every channel you are using as an input(GPIO.IN) or an output(GPIO.OUT).

- GPIO.output(channel, state)

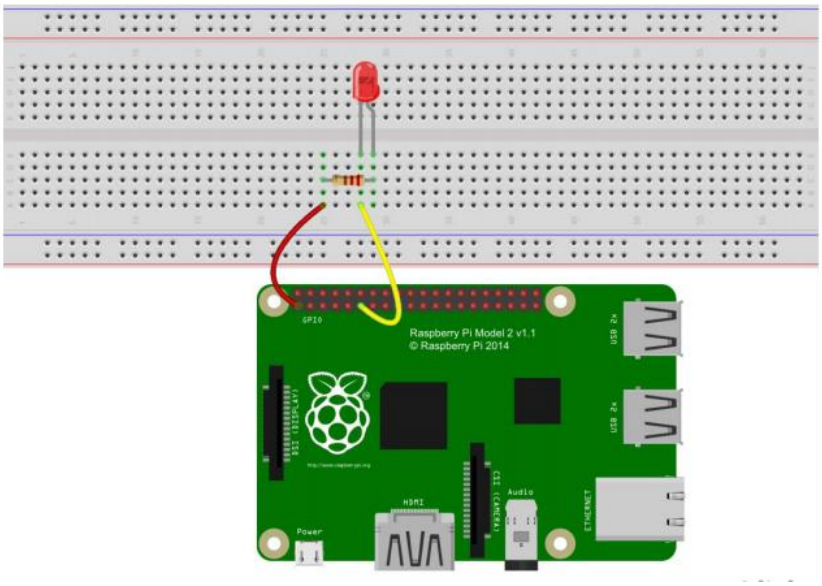
This sets the output state of a GPIO pin. Where channel is the channel number based on the numbering system you have specified (BOARD or BCM). State can be 0 / GPIO. LOW / False or 1 / GPIO. HIGH / True.

- GPIO.cleanup( )

At the end any program, it is good practice to clean up any resources you might have used. This is no different with RPi.GPIO. By returning all channels you have used back to inputs with no pull up/down, you can avoid accidental damage to your RPi by shorting out the pins. Note that this will only clean up GPIO channels that your script has used. Note that GPIO.cleanup () also clears the pin numbering system in use.

8.1.5 Procedures

1. Build the circuit



2. Program

Python user:

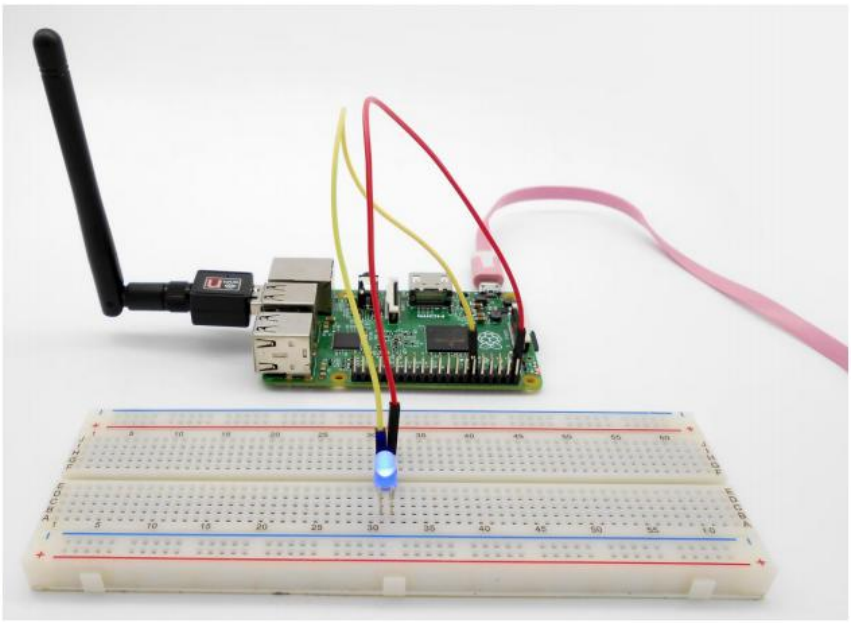
2.1 Edit and save the code with vim or nano.

(https://github.com/UCTRONICS/Arducam_Starter_Kit_Python_Code_for_RPi.git)

2.2 Run the program

```
$ sudo python 01_blinkingLed_1.py
```

Press Enter, and then you can see the LED is blinking.



8.2 Project2: Active Buzzer

8.2.1 Overview

In this lesson, we will learn how to program the Raspberry Pi to make an active buzzer sound.

8.2.2 Requirement

- Raspberry Pi ×1
- Active buzzer ×1
- 1 k Resistor ×1
- NPN Transistor (S8050) ×1
- Breadboard ×1
- Jumper wires

8.2.3 Principle

1. What's Buzzer ?

Please refer to chapter 3.17 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 Raspberry Pi output alternating high and low level, so that the buzzer sounds.

2. What's transistor ?

Please refer to chapter 3.27 Transistor

3. Working Principle

There are two driving circuit for the buzzer:

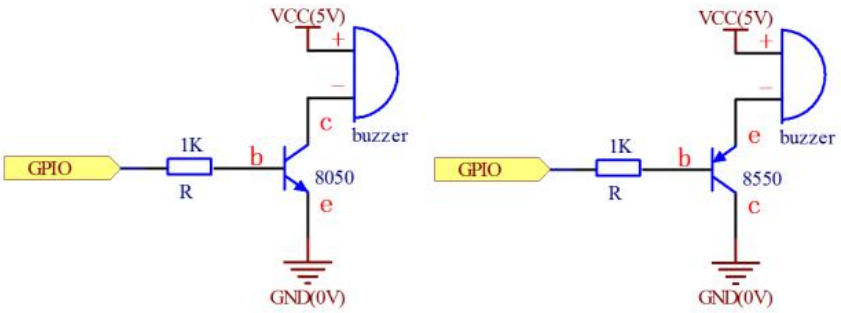


Figure1 Figure2

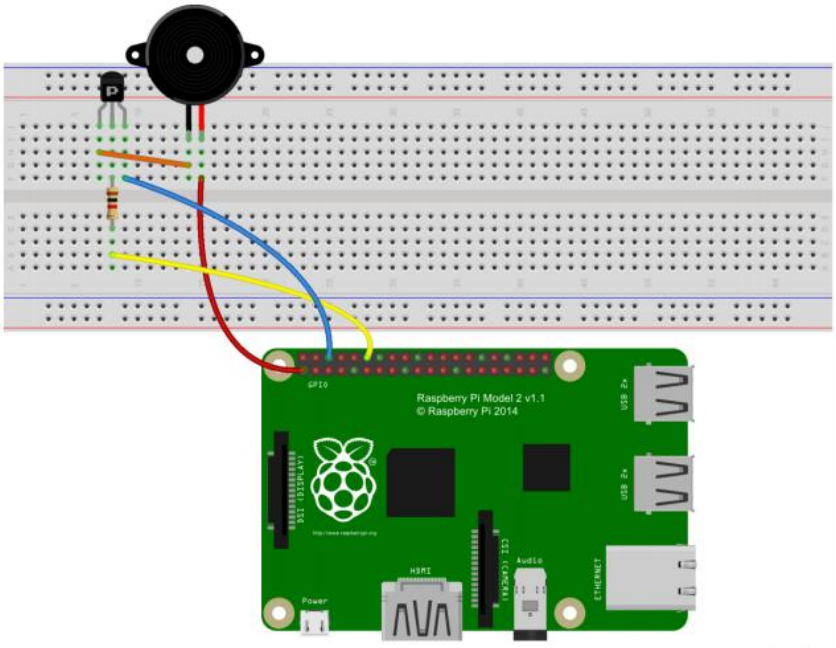
Figure 1: Set the Raspberry Pi GPIO as a high level, the transistor S8050 will conduct, and

then the buzzer will sound; set the Raspberry Pi GPIO as low level, the transistor S8050 will cut off, then the buzzer will stop.

Figure 2: Set the Raspberry Pi GPIO as low level, the transistor S8550 will conduct, and the buzzer will sound; set the Raspberry Pi GPIO as a high level, the transistor S8550 will cut off, then the buzzer will stop.

8.2.4 Procedures

1. Build the circuit



2. Program

Python user:

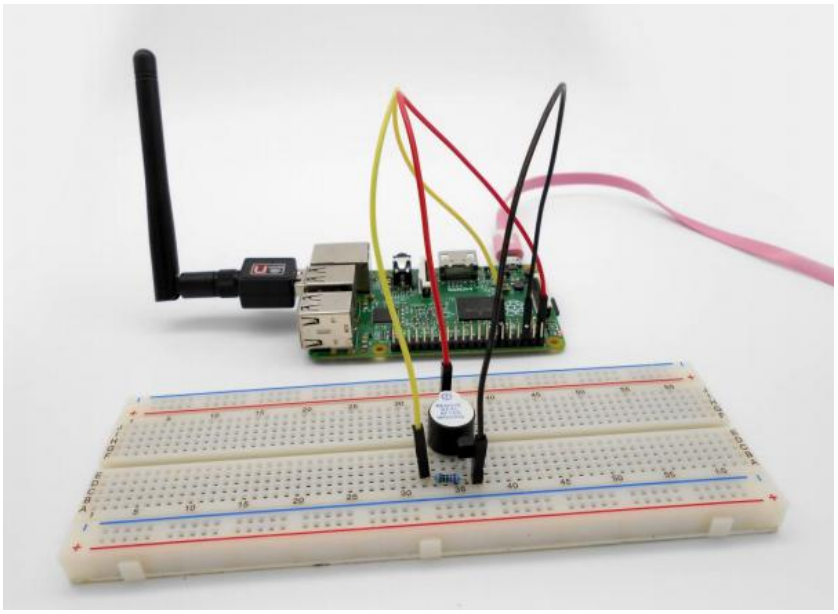
2.1 Edit and save the code with vim or nano.

(https://github.com/UCTRONICS/Arducam_Starter_Kit_Python_Code_for_RPi.git)

2.2 Run the program

```
$ sudo python 02_activeBuzzer.py
```

Now, you should be able to hear the sound of the buzzer.



8.2.5 Summary

By learning this lesson, we have mastered the basic principle of the buzzer and the transistor. We also learned how to program the Raspberry Pi and then control the buzzer. I hope you can use what you have learned in this lesson to do some interesting things.

8.3 Project 3: Passive Buzzer

8.3.1 Overview

In this lesson, we will learn how to program the Raspberry Pi to make a passive buzzer sound with different frequency.

8.3.2 Requirement

- Raspberry Pi ×1
- Passive buzzer ×1
- 1 kΩ Resistor ×1
- NPN Transistor (S8050) ×1
- Breadboard ×1
- Several Jumper wires

8.3.3 Principle

1. What's Active Buzzer ?

Please refer to chapter 3.17 Buzzer

2. Key functions

Python user:

- GPIO.cleanup( )

At the end any program, it is good practice to clean up any resources you might have used. This is no different with RPi.GPIO. By returning all channels you have used back to inputs with no pull up/down, you can avoid accidental damage to your RPi by shorting out the pins. Note that this will only clean up GPIO channels that your script has used. Note that GPIO.cleanup() also clears the pin numbering system in use.

- p = GPIO.PWM(channel, frequency)

To create a PWM instance

- p.start(dc)

To start PWM.

- p.ChangeFrequency(freq)

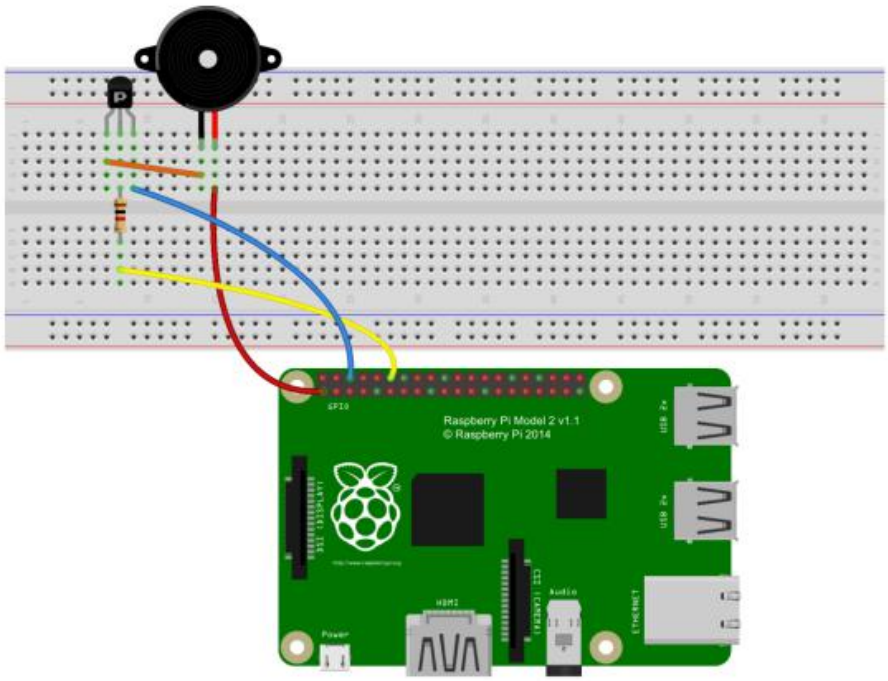
To change the frequency.

- p.stop()

To stop PWM.

8.3.4 Procedure

1. Build the circuit



2. Program

Python user:

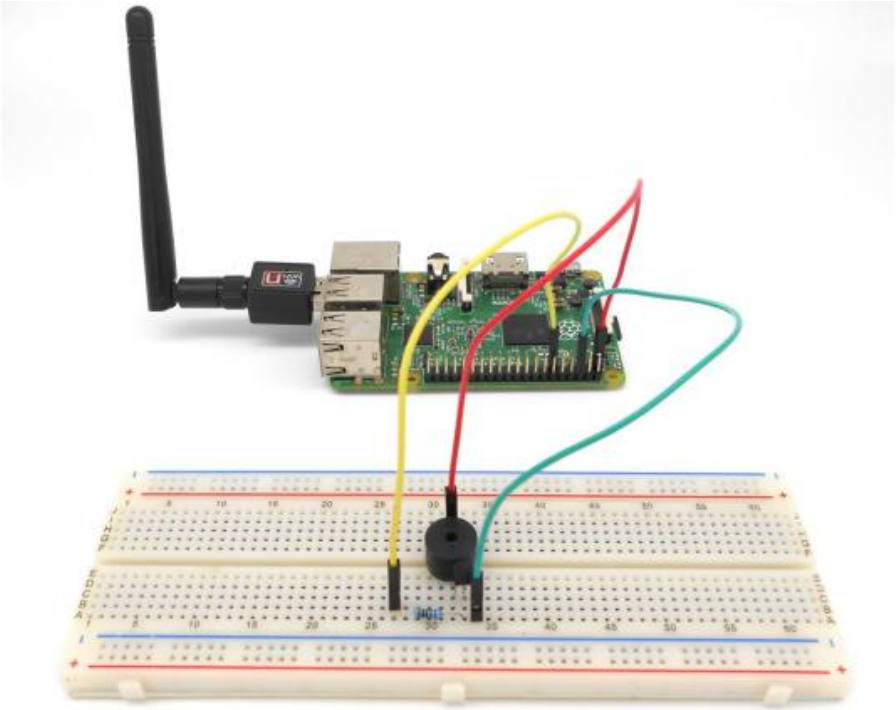
2.1 Edit and save the code with vim or nano.

(https://github.com/UCTRONICS/Arducam_Starter_Kit_Python_Code_for_RPi.git)

2.2 Run the program

```
$ sudo python 03_passiveBuzzer.py
```

Now, you should be able to hear the sound of the buzzer



8.4 Project 4: Controlling an LED with a button

8.4.1 Overview

In this lesson, we will learn how to detect the status of a button, and then toggle the status of LED based on the status of the button.

8.4.2 Requirement

- Raspberry Pi ×1
- Button ×1
- LED ×1
- 220Ω Resistor ×1
- Breadboard ×1
- Several Jumper wires

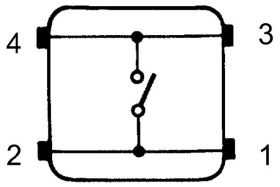
8.4.3 Principle

1. What's Button ?

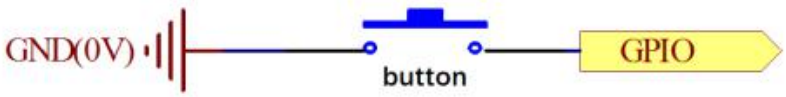
Please refer to chapter 3.22.1 Button

2. Working Principle

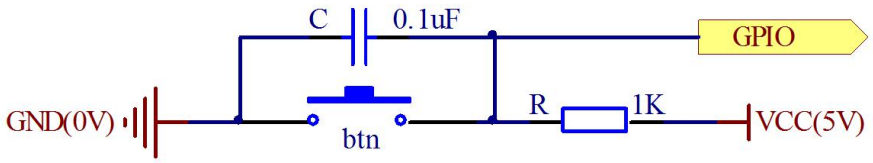
The button we used is a normally open type button. The two contacts of a button are in the off state under the normal conditions, only when the button is pressed they are closed.



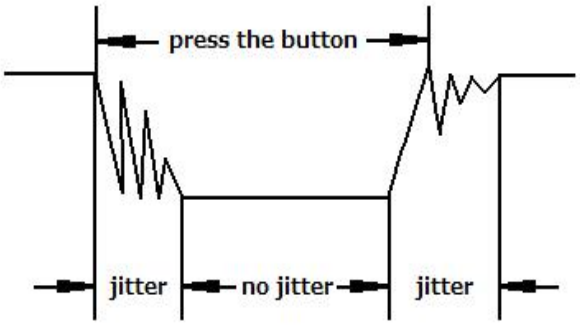
The schematic diagram we used is as follows:



The button jitter must be happened in the process of using. The jitter waveform is as the flowing picture:



Each time you press the button, the Raspberry Pi will think you have pressed the button many times due to the jitter of the button. We must to deal with the jitter of buttons before we use the button. We can through the software programming method to remove the jitter of buttons, and you can use a capacitance to remove the jitter of buttons. We introduce the software method. First, we detect whether the level of button interface is low level or high level. When the level we detected is low level, 5~10 MS delay is needed, and then detect whether the level of button interface is low or high. If the signal is low, we can confirm that the button is pressed once. You can also use a 0.1uF capacitance to clean up the jitter of buttons. The schematic diagram is shown in below:



3. Interrupt

Hardware interrupts were introduced as a way to reduce wasting the processor's valuable time in polling loops, waiting for external events. They may be implemented in hardware as a distinct system with control lines, or they may be integrated into the memory subsystem.

4. Key functions:

Python user:

- GPIO.input(channel)

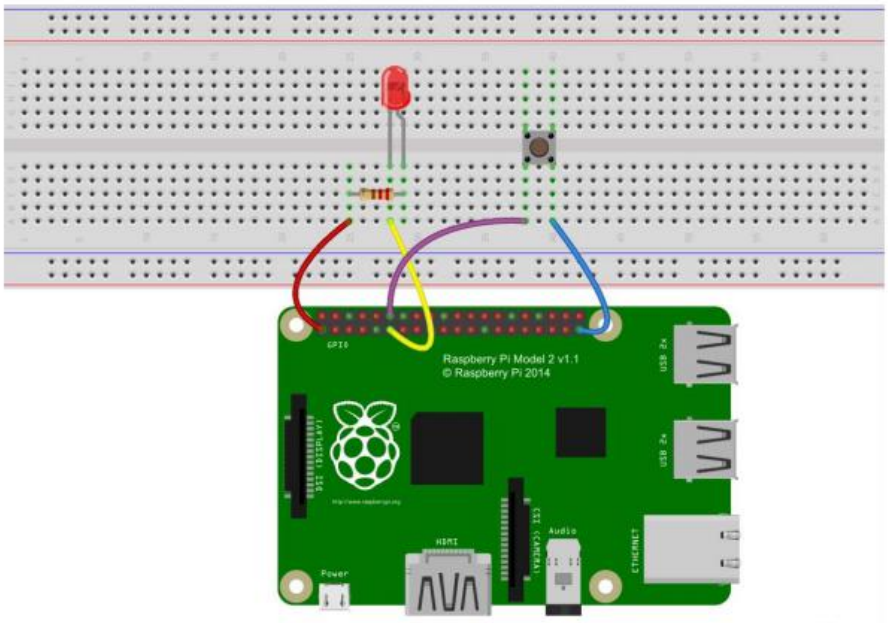
This is used for reading the value of a GPIO pin. Where channel is the channel number based on the numbering system you have specified (BOARD or BCM)). This will return either 0 / GPIO.LOW / False or 1 / GPIO.HIGH / True.

- GPIO.add_event_detect(channel, mode)

The event_detected( ) function is designed to be used in a loop with other things, but unlike polling it is not going to miss the change in state of an input while the CPU is busy working on other things. This could be useful when using something like Pygame or PyQt where there is a main loop listening and responding to GUI events in a timely basis.

8.4.4 Procedures

1. Build the circuit



2. Program

Python user:

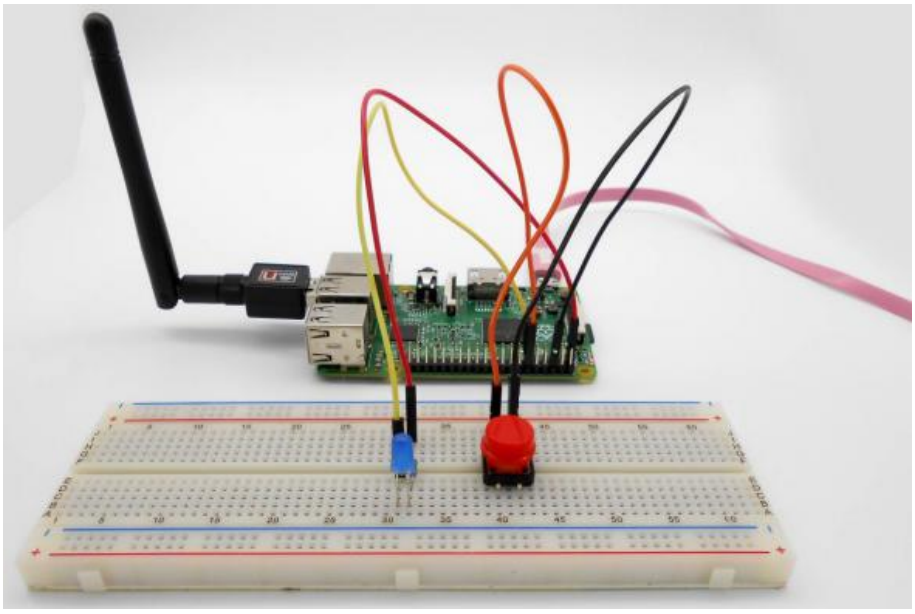
2.1 Edit and save the code with vim or nano.

(https://github.com/UCTRONICS/Arducam_Starter_Kit_Python_Code_for_RPi.git)

2.2 Run the program

```
$ sudo python 04_btnAndLed_1.py
```

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



8.4.5 Summary

Through this lesson, you should have learned how to use the Raspberry Pi detect an external button status, and then toggle the status of LED relying on the status of the button detected before.

8.5 Project 5: Relay

8.5.1 Overview

In this lesson, we will learn how to control a relay to cut off or connect a circuit.

8.5.2 Requirement

- Raspberry Pi ×1
- Relay ×1
- NPN Transistor (S8050) ×1
- Diode (1N4001) ×1
- 1KΩ Resistor ×1
- Breadboard ×1
- Several Jumper wires

8.5.3 Principle

1. What' s relay ?

Please refer to chapter 3.16 Relay

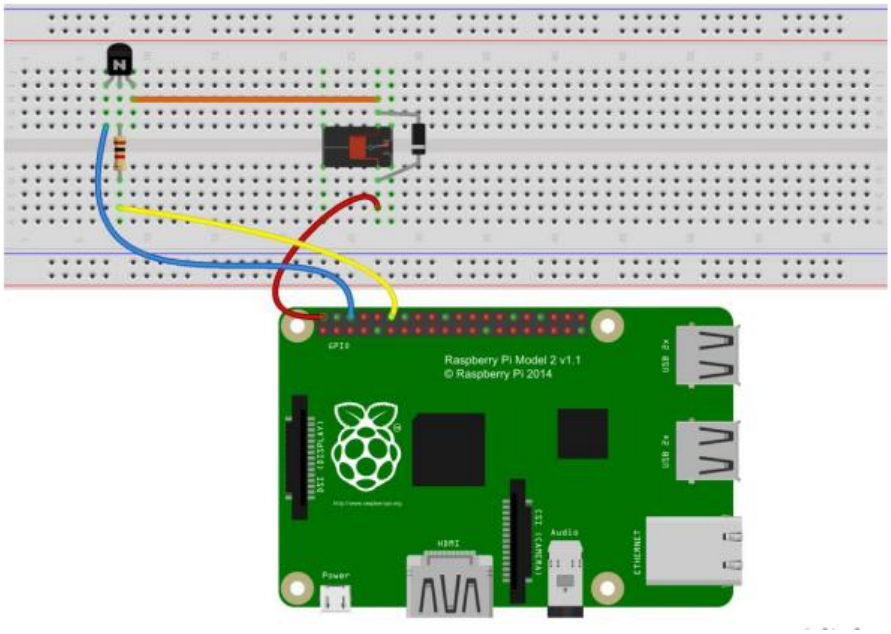
2. Working Principle

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.

8.5.4 Procedures

1. Build the circuit



2. Program

Python user:

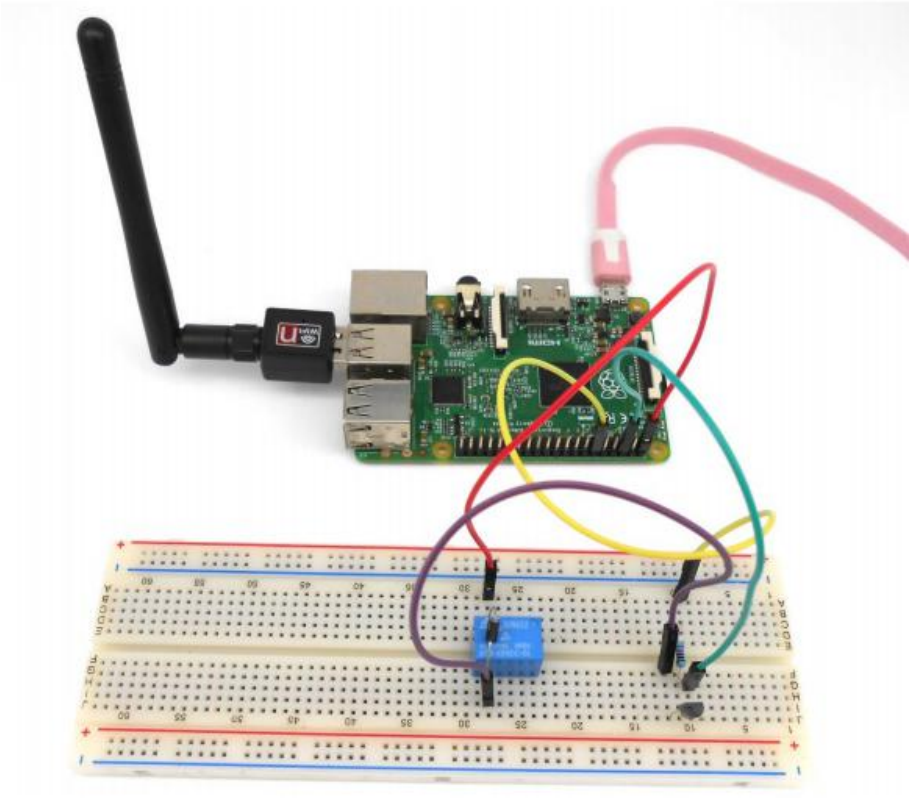
2.1 Edit and save the code with vim or nano.

(https://github.com/UCTRONICS/Arducam_Starter_Kit_Python_Code_for_RPi.git)

2.2 Run the program

```
$ sudo python 05_relay.py
```

Press Enter, you should be able to hear the sound of relay toggling.



8.6 Project 6: LED Flowing Lights

8.6.1 Overview

In the first class, we have learned how to make an LED blink by programming the Raspberry Pi. Today, we will use the Raspberry Pi to control 8 LEDs, so that 8 LEDs showing the result of flowing.

8.6.2 Requirement

- Raspberry Pi ×1
- LED ×8
- 220Ω Resistor ×8
- Breadboard ×1
- Several Jumper wires

8.6.3 Principle

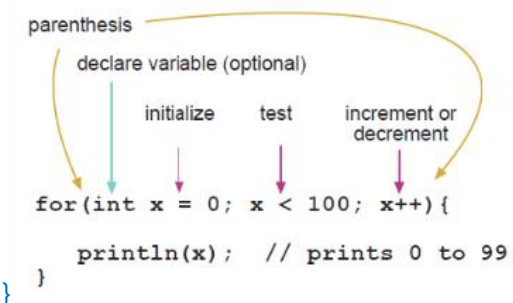
1. Key function:

- for statements

The for statement is used to repeat a block of statements enclosed in curly braces. An increment counter is usually used to increment and terminate the loop. The for statement is useful for any repetitive operation, and is often used in combination with arrays to operate on collections of data/pins.

There are three parts to the for loop header:

```
for (initialization; condition; increment) {
//statement(s);
```



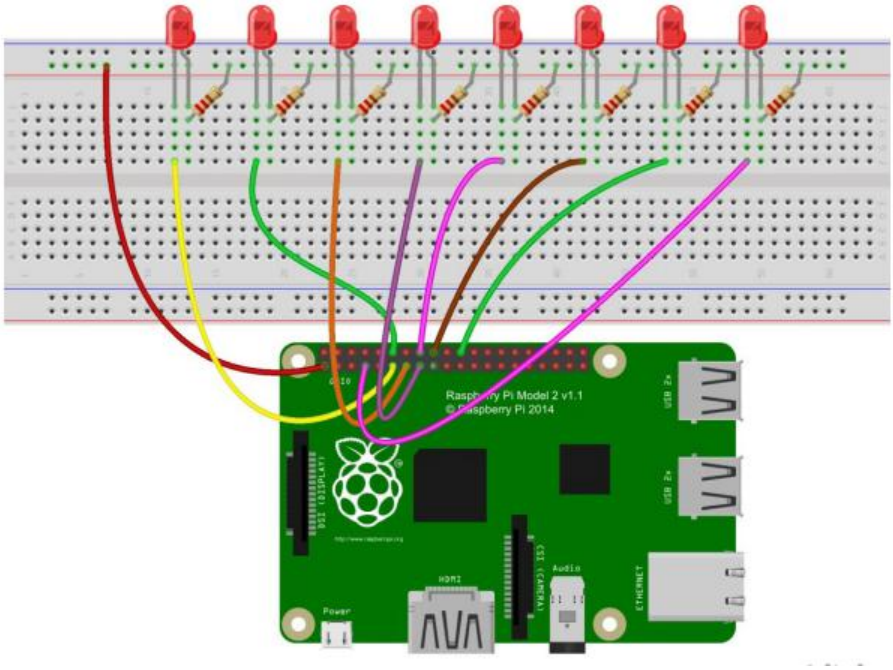

```
}
```

The initialization happens first and exactly once. Each time through the loop, the condition is

tested; if it's true, the statement block, and the increment is executed, then the condition is tested again. When the condition becomes false, the loop ends.

8.6.4 Procedures

1. Build the circuit



2. Program

Python user:

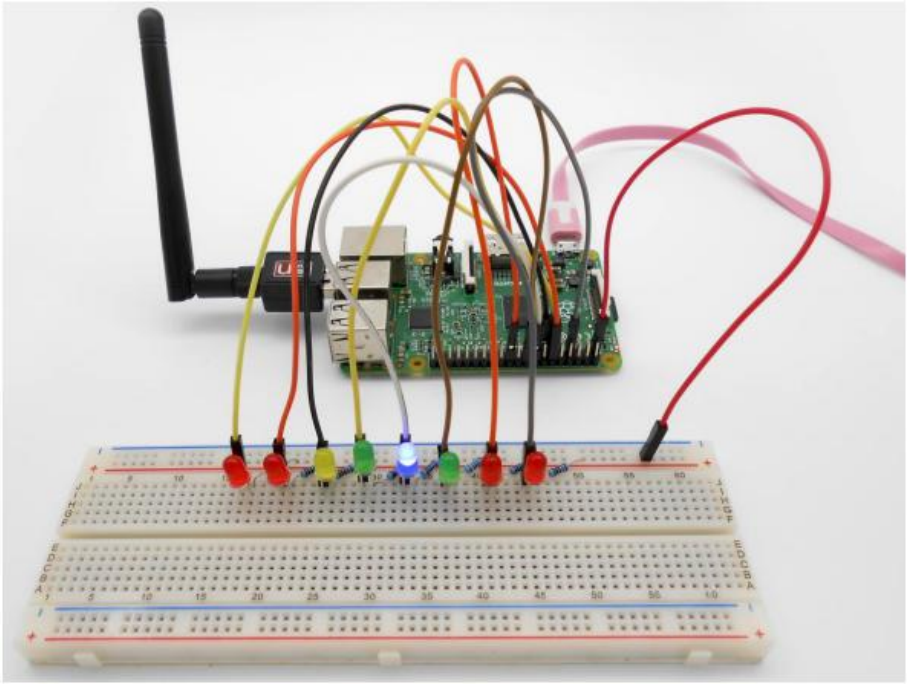
2.1 Edit and save the code with vim or nano.

(https://github.com/UCTRONICS/Arducam_Starter_Kit_Python_Code_for_RPi.git)

2.2 Run the program

```
$ sudo python 06_flowingLed.py
```

Now, you should see 8 LEDs are lit in sequence from the right red one to the left, next from the left to the right one. And then repeat the above phenomenon.



8.6.5 Summary

Through this simple and fun experiment, we have learned more skilled programming about the Raspberry Pi. In addition, you can also modify the circuit and code we provided to achieve even more dazzling effect.

8.7 Project 7: Breathing LED

8.7.1 Overview

In this lesson, we will learn how to program the Raspberry Pi to generate PWM signal. And use the PWM square-wave signal control an LED gradually becomes brighter and then gradually becomes dark like the animal's breath.

8.7.2 Requirement

- Raspberry Pi ×1
- LED ×1
- 220Ω Resistor ×1
- Breadboard ×1
- Several Jumper wires

8.7.3 Principle

1. What's PWM ?

Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off. This on-off pattern can simulate voltages in between full on (5 Volts) and off (0 Volts) by changing the portion of the time the signal spends on versus the time that the signal spends off. The duration of "on time" is called the pulse width. To get varying analog values, you change, or modulate, that pulse width. If you repeat this on-off pattern fast enough with an LED for example, the result is as if the signal is a steady voltage between 0 and 3.3v controlling the brightness of the LED. In the graphic below, the green lines represent a regular time period. This duration or period is the inverse of the PWM frequency. In other words, with Raspberry Pi's PWM frequency at about 500Hz, the green lines would measure 2 milliseconds each. A call to pwmWrite() is on a scale of 0-1023, such that pwmWrite(1023) requests a 100% duty cycle (always on), and pwmWrite(511) is a 50% duty cycle (on half the time) for example.

2. Key function

Python user:

- p = GPIO.PWM(channel, frequency)

This is used for creating a PWM.

- p.start(dc)

Start the pwm you have created.

- p.ChangeFrequency(freq)

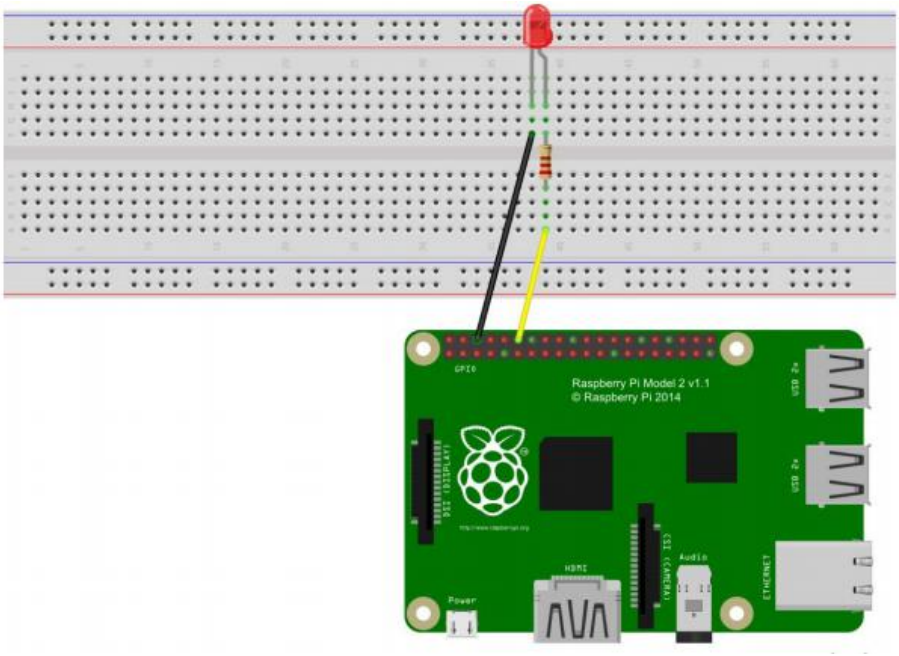
Change the frequency of pwm.

- p.stop( )

Stop the pwm.

8.7.4 Procedures

1. Build the circuit



2. Program

Python user:

2.1 Edit and save the code with vim or nano.

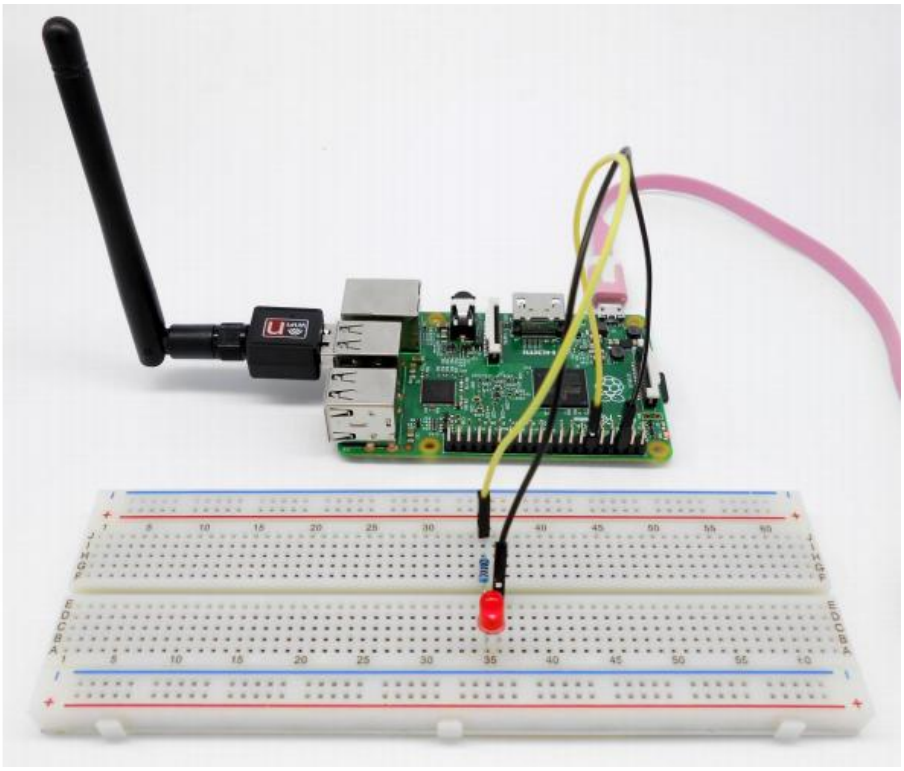
(https://github.com/UCTRONICS/Arducam_Starter_Kit_Python_Code_for_RPi.git)

2.2 Run the program

```
$ sudo python 07_breathingLed.py
```

Now, you should see the LED gradually from dark to brighter, and then from brighter to dark,

continuing to repeat the process, its rhythm like the animal's breathing.



8.7.5 Summary

By learning this lesson, I believe that you have understood the basic principles of the PWM, and mastered the PWM programming on the Raspberry Pi platform.

8.8 Project 8: Controlling a RGB LED with PWM

8.8.1 Overview

In this lesson, we will program the Raspberry Pi for RGB LED control, and make RGB LED emits a variety of colors of light.

8.8.2 Requirement

- Raspberry Pi ×1
- RGB LED ×1
- 220Ω Resistor ×3
- Breadboard ×1
- Several Jumper wires

8.8.3 Principle

1. What's RGB LEDs ?

Please refer to chapter 3.21.1 RGB LED

What we used in this experiment is the common anode RGB LED. The longest pin is the common anode of three LEDs. The pin is connected to the +5V pin of the Raspberry Pi, and the three remaining pins are connected to the Raspberry Pi' s D9, D10, D11 pins through a current limiting resistor.

In this way, we can control the color of RGB LED by 3-channel PWM signal.

2. Key function

- int softPwmCreate (int pin, int initialValue, int pwmRange)

This creates a software controlled PWM pin. You can use any GPIO pin and the pin numbering will be that of the wiringPiSetup() function you used. Use 100 for the pwmRange, then the value can be anything from 0 (off) to 100 (fully on) for the given pin. The return value is 0 for success. Anything else and you should check the global errno variable to see what went wrong.

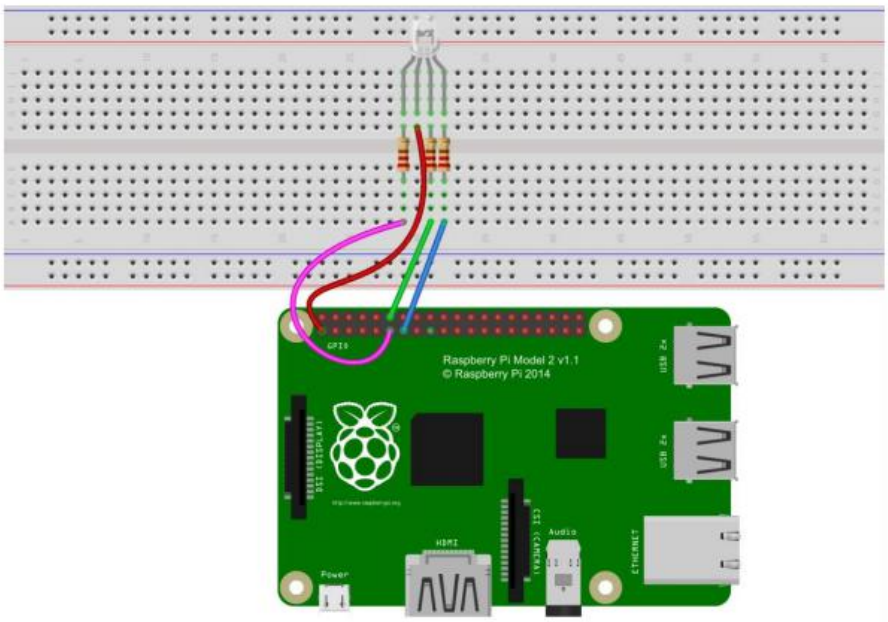
- void softPwmWrite (int pin, int value)

This updates the PWM value on the given pin. The value is checked to be in-range and pins

that haven't previously been initialised via softPwmCreate will be silently ignored.

8.8.4 Procedures

1. Build the circuit



2. Program

Python user:

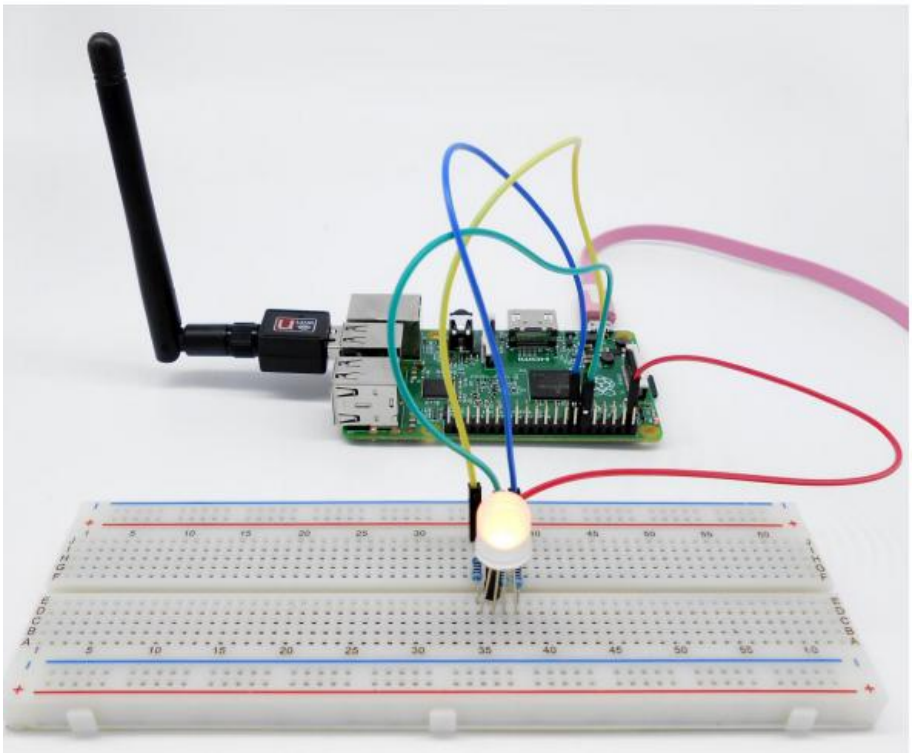
2.1 Edit and save the code with vim or nano

(https://github.com/UCTRONICS/Arducam_Starter_Kit_Python_Code_for_RPi.git)

2.2 Run the program

```
$ sudo python 08_rgbLedLed.py
```

Now, you can see that the RGB LED emitting red, green, blue, yellow, white and purple light, then the RGB LED will be off, each state continues 1s, after repeating the above procedure.



8.8.5 Summary

By learning this lesson, I believe that you have already known the principle and the programming of RGB LED. I hope you can use your imagination to achieve even more cool ideas based on this lesson.

8.9 Project 9: 7-segment display

8.9.1 Overview

In this lesson, we will program the Raspberry Pi to achieve the controlling of segment display.

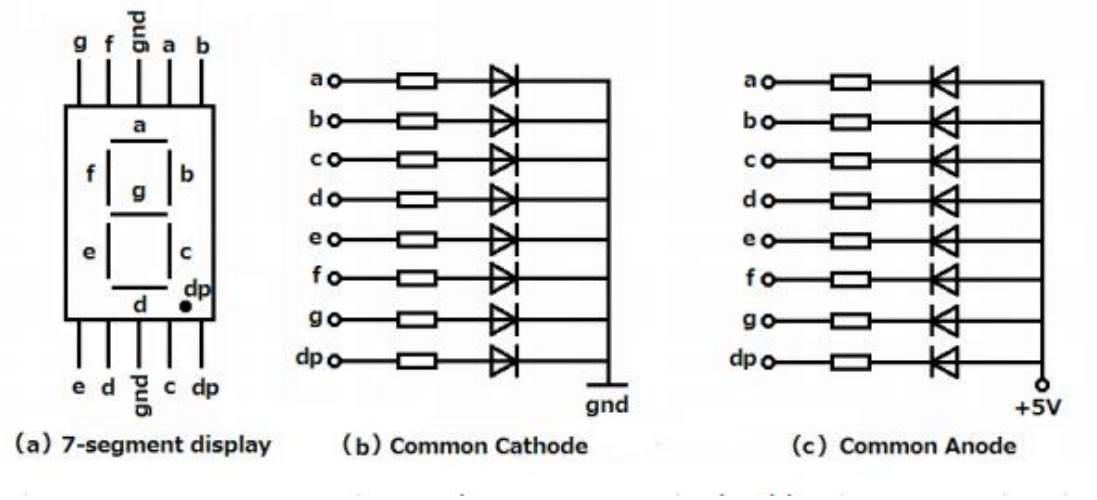
8.9.2 Requirement

- Raspberry Pi ×1
- 220Ω Resistor ×1
- 7-Segment display ×1
- Breadboard ×1
- Several Jumper wires

8.9.3 Principle

1. What's seven-segment display ?

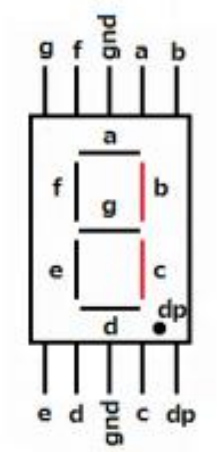
The seven-segment display is a form of electronic display device for displaying decimal numerals that is an alternative to the more complex dot matrix displays. Seven-segment displays are widely used in digital clocks, electronic meters, basic calculators, and other electronic devices that display numerical information. The seven-segment display is an 8-shaped LED display device composed of eight LEDs (including a decimal point), these segments respectively named a, b, c, d, e, f, g, dp. The segment display can be divided into common anode and common cathode segment display by internal connections.



(a) 7-segment display (b) Common Cathode (c) Common Anode

When using a common anode LED, the common anode should to be connected to the power supply (VCC); when using a common cathode LED, the common cathode should be connected to the ground (GND). Each segment of a segment display is composed of LED, so a resistor is needed for protecting the LED.

A 7-segment display has seven segments for displaying a figure and a segment for displaying a decimal point. If you want to display a number '1', you should only light the segment b and c.



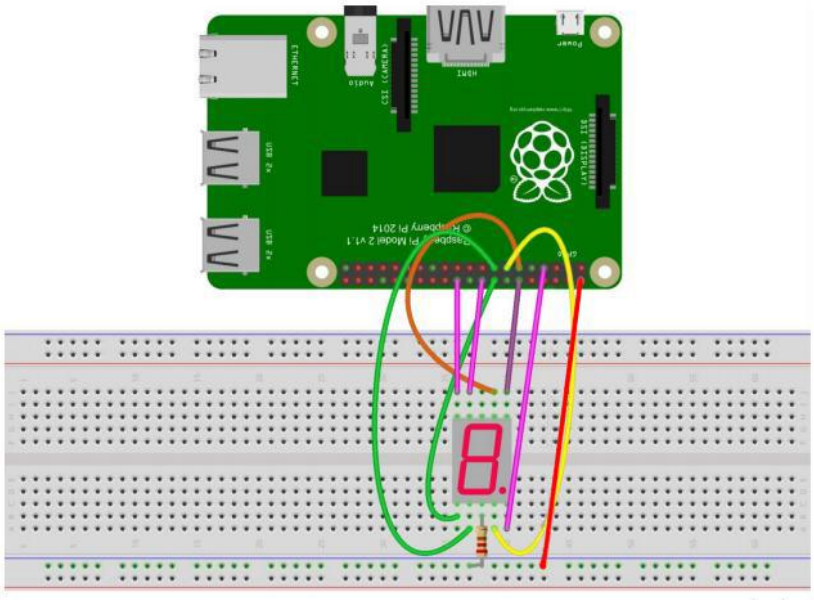
2. Key function

- void digitalWriteByte (int value)

This writes the 8-bit byte supplied to the first 8 GPIO pins. It's the fastest way to set all 8 bits at once to a particular value, although it still takes two write operations to the Pi's GPIO hardware.

8.9.4 Procedures

1. Build the circuit



2. Program

Python user:

2.1 Edit and save the code with vim or nano.

(https://github.com/UCTRONICS/Arducam_Starter_Kit_Python_Code_for_RPi.git)

2.2 Run the program

```
$ sudo python 09_segment.py
```

Now, you should see the number 0~9, a~F are displayed on the segment display.

8.9.4 Summary

Through this lesson, we have learned the principle and programming of segment display. I hope you can combine the former course to modify the code we provided in this lesson to achieve cooler originality.

8.10 Project 10 4-digit 7-segment display

8.10.1 Overview

In this lesson, we will program the Raspberry Pi to achieve the controlling of 4-digit 7-segment display.

8.10.2 Requirement

- Raspberry Pi ×1
- 220Ω Resistor ×4
- 4-digit 7-segment display ×1
- Breadboard ×1
- Several Jumper wires

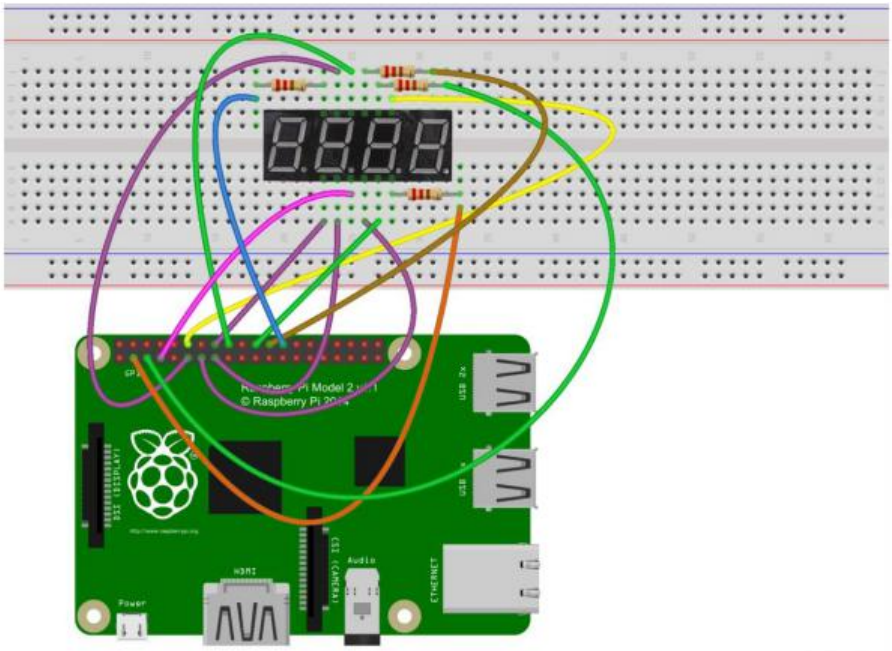
8.10.3 Principle

1. What's four-digit segment display ?

The four-digit segment display is a form of electronic display device for displaying decimal numerals that is an alternative to the more complex dot matrix displays. Four-digit segment displays are widely used in digital clocks, electronic meters, basic calculators, and other electronic devices that display numerical information. The four-digit segment display is an 4x8-shaped LED display device composed of 32 LEDs (including four decimal points), these segments respectively named a, b, c, d, e, f, g, h, dig1, dig2, dig3, dig4.

8.10.4 Procedures

1. Build the circuit



2. Program

Python user:

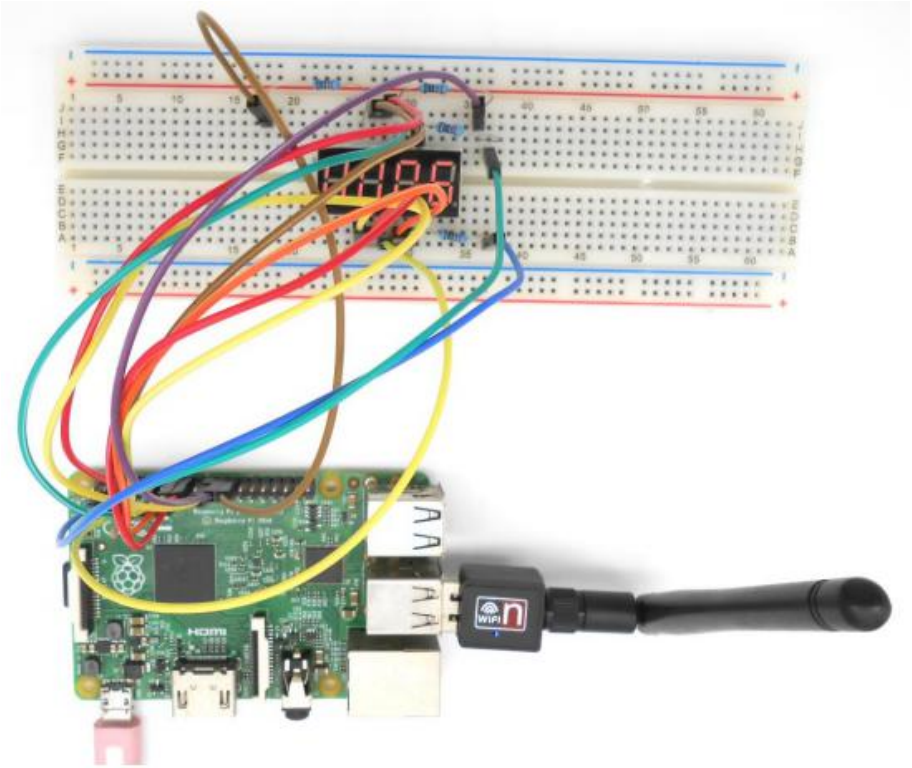
2.1 Edit and save the code with vim or nano.

(https://github.com/UCTRONICS/Arducam_Starter_Kit_Python_Code_for_RPi.git)

2.2 Run the program

```
$ sudo python 10_fourBitSegment.py
```

Now, you should see the number is displayed on the segment display.



8.11 Project 11: LCD1602

8.11.1 Overview

In this lesson, we will learn how to use a character display device — LCD1602 on the Raspberry Pi platform. First, we make the LCD1602 display a string "Hello Geeks!" scrolling, then display “ Uctronics” and “ www.uctronics.com” static.

8.11.2 Requirement

- Raspberry Pi x 1
- 1x LCD1602 x 1
- 1x 10KΩ Potentiometer x 1
- 1x Breadboard x 1
- Several Jumper wires

8.11.3 Principle

1. What’s LCD display ?

Please refer to chapter 3.6. LCD1602

2. The interface consists of the following pins

- 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.
- A Read/Write (R/W) pin that selects reading mode or writing mode
- An Enable pin that enables writing to the registers
- 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.
- There's also a display contrast pin (Vo), power supply pins (+5V and Gnd) and LED Backlight (Bklt+ and BKlt-) pins that you can use to power the LCD, control the display contrast, and turn on or off the LED backlight respectively.

The process of controlling the display involves putting the data that form the image of what you want to display into the data registers, then putting instructions in the instruction

register. The wiringPiDev Library simplifies this for you, so you don't need to know the low-level instructions.

The Hitachi-compatible LCDs can be controlled in two modes: 4-bit or 8-bit. The 4-bit mode requires six I/O pins from the Raspberry Pi, while the 8-bit mode requires 10pins. For displaying text on the screen, you can do most everything in 4-bit mode, so example shows how to control a 2x16 LCD in 4-bit mode.

A potentiometer, informally a pot, is a three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider. If only two terminals are used, one end and the wiper, it acts as a variable resistor or rheostat.

3. Key function

- int lcdInit (int rows, int cols, int bits, int rs, int strb, int d0, int d1, int d2, int d3, int d4, int d5, int d6, int d7)

This is the main initialisation function and must be called before you use any other LCD functions. Rows and cols are the rows and columns on the display (e.g. 2, 16 or 4,20). Bits is the number of bits wide on the interface (4 or 8). The rs and strb represent the pin numbers of the displays RS pin and Strobe (E) pin. The parameters d0 through d7 are the pin numbers of the 8 data pins connected from the Pi to the display. Only the first 4 are used if you are running the display in 4-bit mode. The return value is the 'handle' to be used for all subsequent calls to the lcd library when dealing with that LCD, or -1 to indicate a fault. (Usually incorrect parameters)

- lcdPosition (int handle, int x, int y)

Set the position of the cursor for subsequent text entry. x is the column and 0 is the left-most edge. y is the line and 0 is the top line.

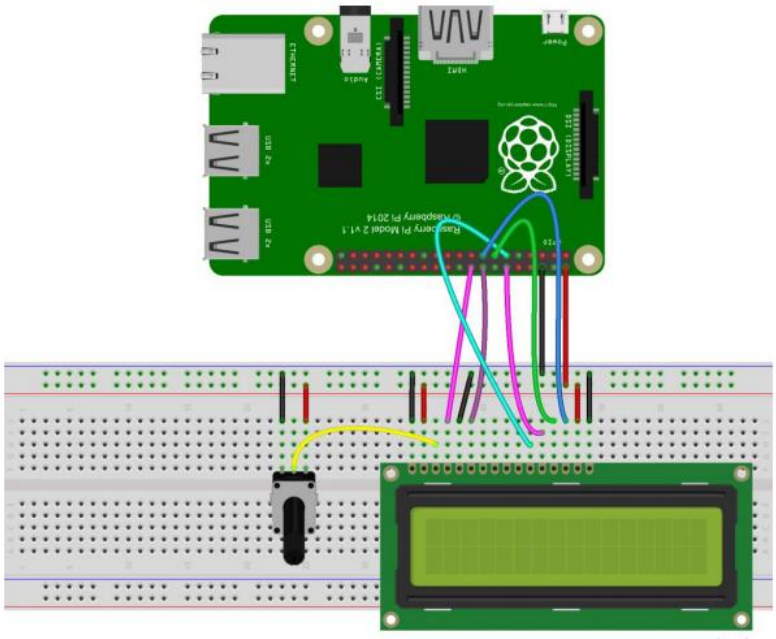
- lcdPuts (int handle, const char xstring)
- lcdPrintf (int handle, const char xmessage, ...)
- lcdPutchar (int handle, unsigned char data)

These output a single ASCII character, a string or a formatted string using the usual printf formatting commands.

At the moment, there is no clever scrolling of the screen, but long lines will wrap to the next line, if necessary.

8.11.4 Procedures

1. Build the circuit



2. Program

Python user:

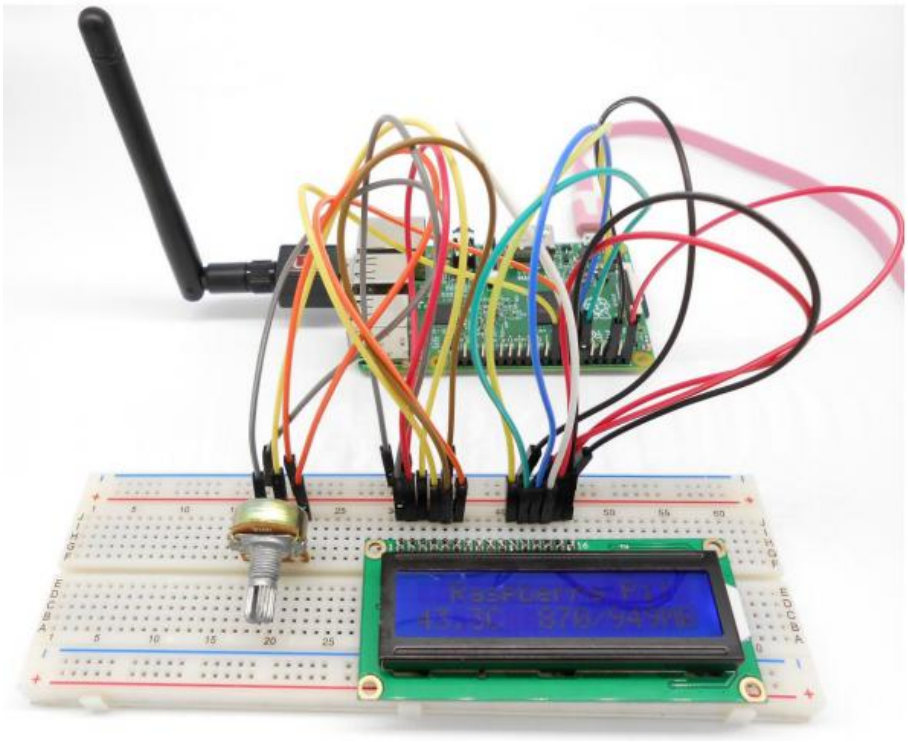
2.1 Edit and save the code with vim or nano.

(https://github.com/UCTRONICS/Arducam_Starter_Kit_Python_Code_for_RPi.git)

2.2 Run the program

```
$ sudo python 11_lcd1602.py
```

Now, you can see the string "Hello Geeks!" is shown on the LCD1602 scrolling, and then the string "UCTRONICS" and "www.uctronics.com" is displayed on the LCD1602 static.



8.11.5 Summary

I believe that you have already mastered the driver of LCD1602 through this lesson. I hope you can make something more interesting base on this lesson and the previous lesson learned.

8.12 Project 12: A Simple Voltmeter

8.12.1 Overview

In this lesson, we will make a simple voltmeter with Raspberry Pi and LCD1602, the range of this voltmeter is 0~5V. Then, we will measure the voltage of the potentiometer' s adjustment end with the simple voltmeter and display it on the LCD1602.

8.12.2 Requirement

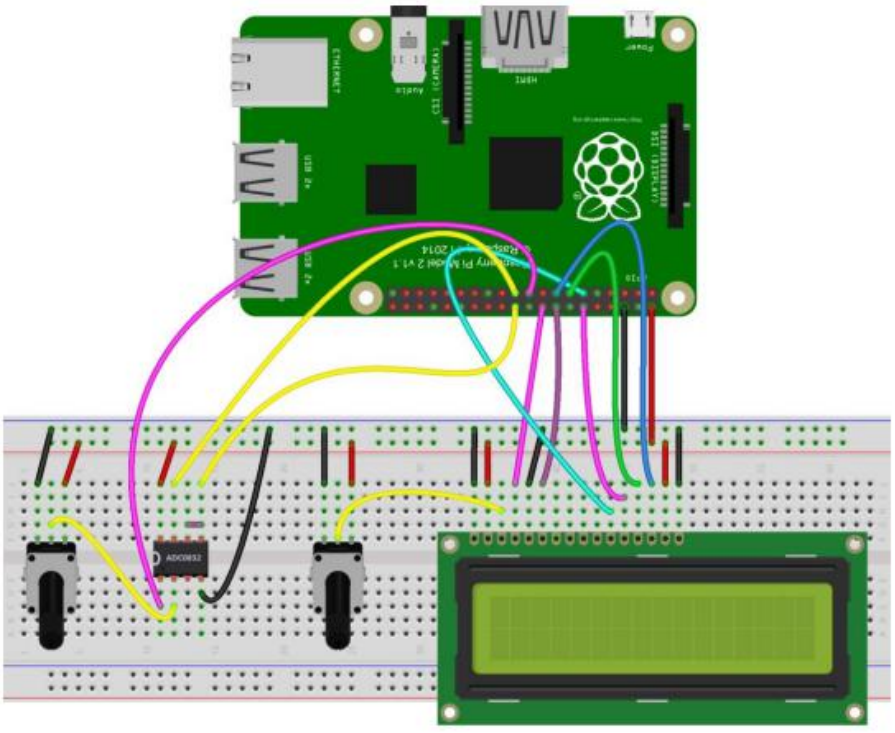
- Raspberry Pi ×1
- LCD1602 ×1
- Potentiometer ×2
- Breadboard ×1
- Several Jumper wires

8.12.3 Principle

The basic principle of this experiment: Converting the analog voltage that the ADC0832 collected to digital quantity through programming, then display the voltage on the LCD1602.

8.12.4 Procedures

1. Build the circuit



2. Program

Python user:

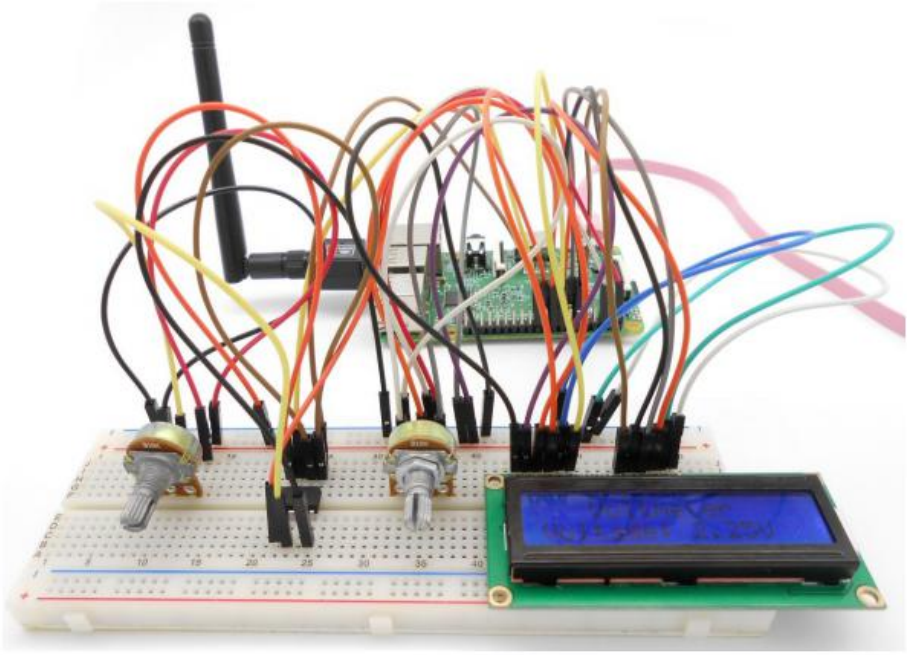
2.1 Edit and save the code with vim or nano.

(https://github.com/UCTRONICS/Arducam_Starter_Kit_Python_Code_for_RPi.git)

2.2 Run the program

```
$ sudo python 12_voltmeter.py
```

Now, when you turning the shaft of the potentiometer, you will see the voltage displayed on the LCD1602 will be changed.



8.12.5 Summary

The substance of voltmeter is reading analog voltage which input to ADC. Through this course, I believe that you have mastered how to read analog value and how to make a simple voltmeter with Raspberry Pi.

8.13 Project 13: Matrix Keyboard

8.13.1 Overview

In this lesson, we will learn how to use a matrix keyboard.

8.13.2 Requirement

- Raspberry Pi ×1
- 4x4 Matrix Keyboard×1
- Several Jumper wires

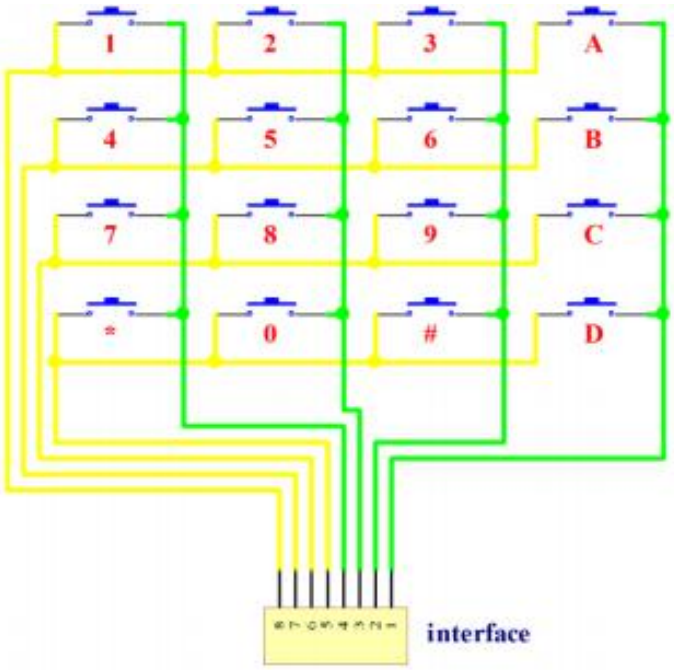
8.13.3 Principle

1. What's 4x4 matrix keyboard ?

Please refer to chapter 3.10 4x4 Matrix Array 16 Key Keyboard

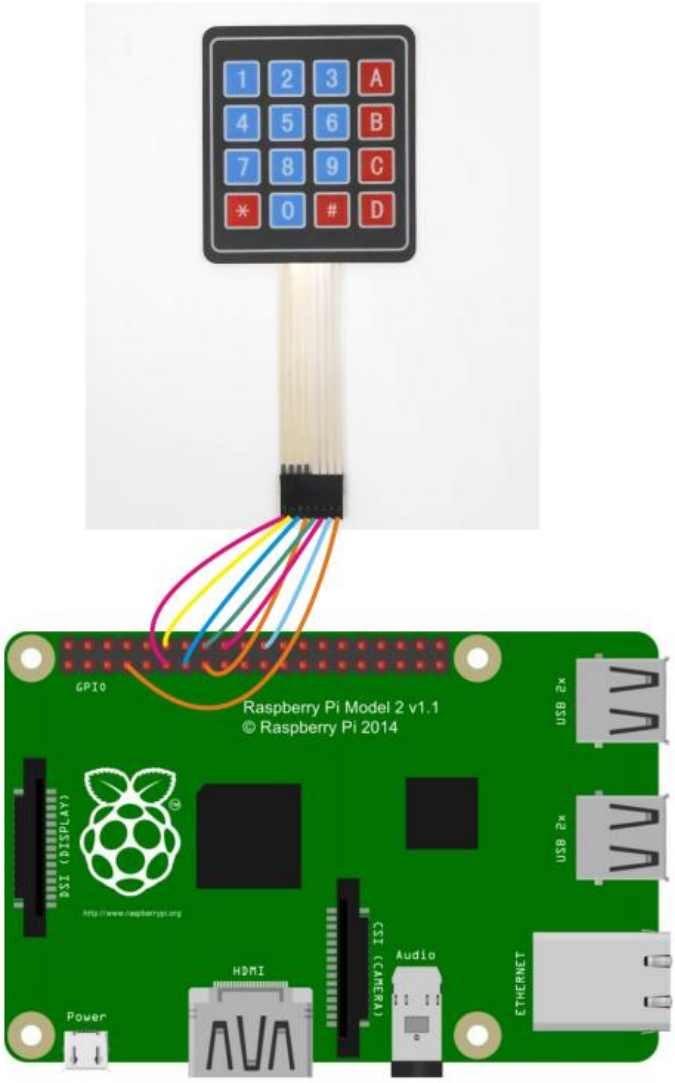
In order to save the resources of the microcontroller port, we usually connect the buttons in a matrix in an actual project.

The following is the schematics of 4x4 matrix keyboard:



8.13.4 Procedures

1. Build the circuit



2. Program

Python user:

2.1 Edit and save the code with vim or nano.

(https://github.com/UCTRONICS/Arducam_Starter_Kit_Python_Code_for_RPi.git)

2.2 Run the program

```
$ sudo python 13_matrixKeyboard.py
```

Now, when you press one of the button on the 4x4 matrix keyboard, you will see the corresponding key value will be displayed on the terminal.


A B C D
3 6 9 #
2 5 8 0
1 4 7 *

8.14 Project 14: Measure the distance

8.14.1 Overview

In this lesson, we will learn how to measure the distance by the ultrasonic distance sensor.

8.14.2 Requirement

- Raspberry Pi ×1
- Ultrasonic distance sensor ×1
- Breadboard ×1
- Several Jumper wires

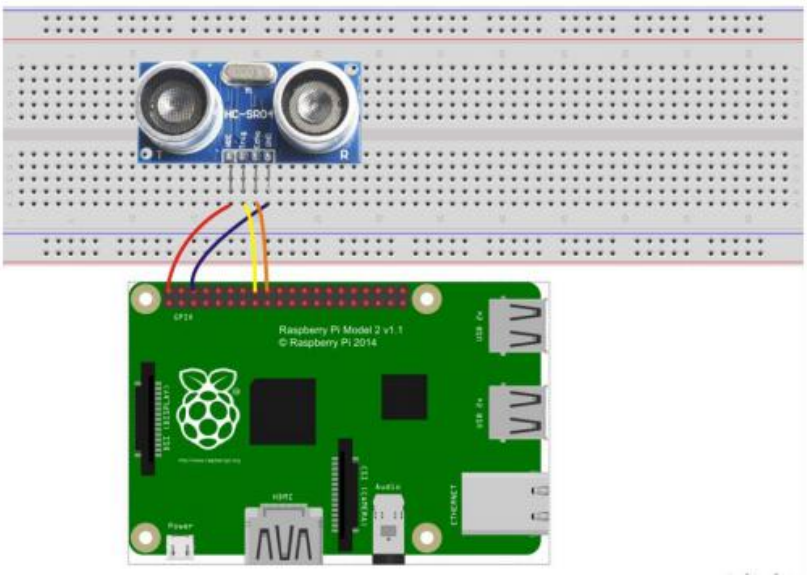
8.14.3 Principle

1. What's ultrasonic distance sensor ?

Please refer to chapter 3.3 HC-SR04 Ultrasonic Distance Sensor Module

8.14.4 Procedures

1. Build the circuit



2. Program

Python user:

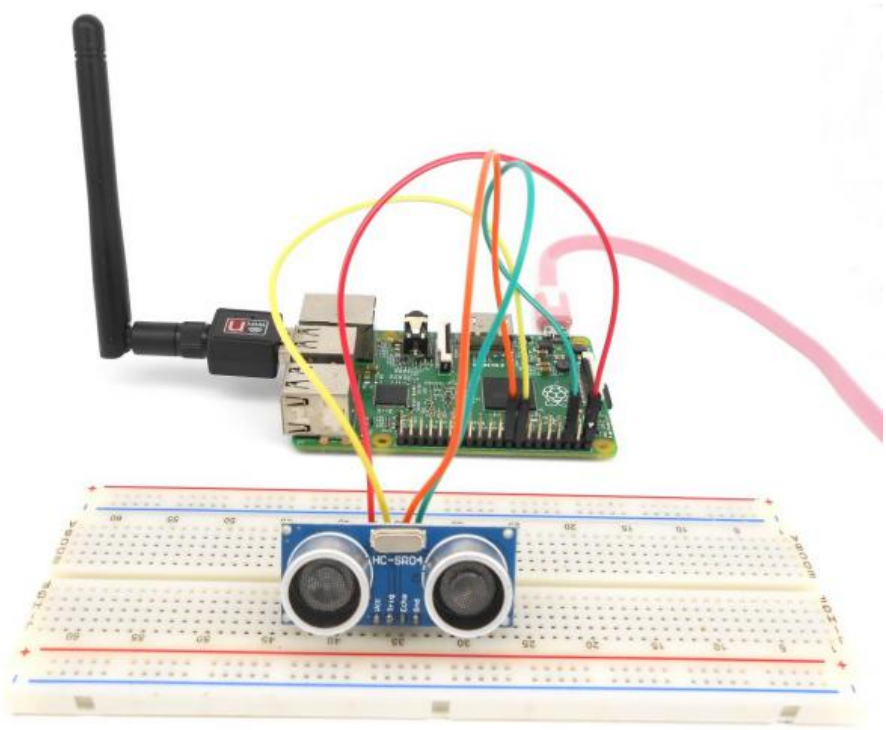
2.1 Edit and save the code with vim or nano.

(https://github.com/UCTRONICS/Arducam_Starter_Kit_Python_Code_for_RPi.git)

2.2 Run the program

```
$ sudo python 14_distance.py
```

Now, you will see the distance between the obstacle and the ultrasonic sensor display on the screen.



8.15 Project 15: Temperature & Humidity Sensor DHT-11

8.15.1 Overview

In this lesson, we will learn how to use DHT-11 to collect temperature and humidity data.

8.15.2 Requirement

- Raspberry Pi ×1
- DHT-11×1
- Breadboard ×1
- Several Jumper wires

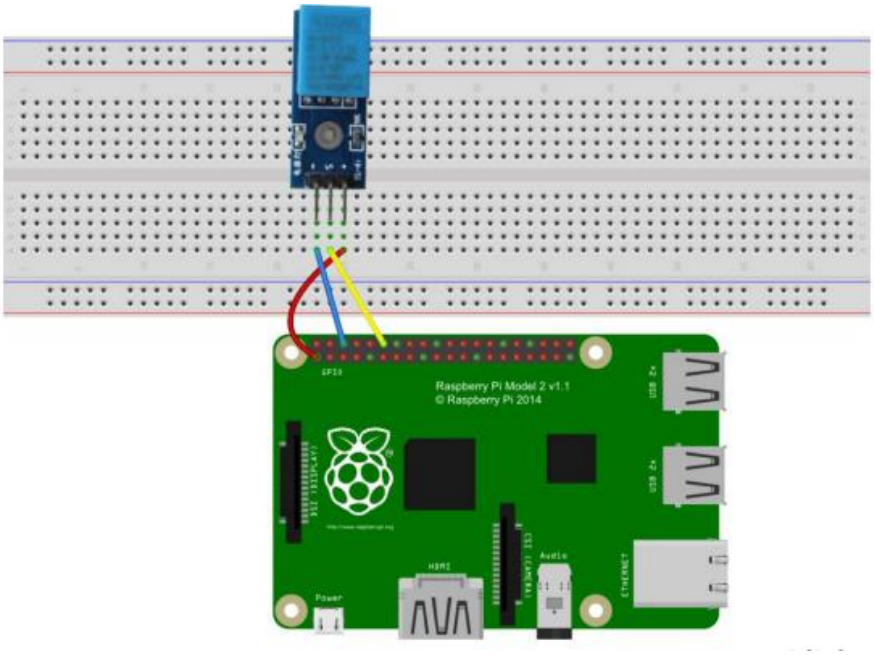
8.15.3 Principle

1. What's Temperature & Humidity Sensor DHT-11 ?

Please refer to chapter 3.2 DHT-11 Digital Temperature & Humidity Sensor

8.15.4 Procedures

1. Build the circuit



2. Program

Python user:

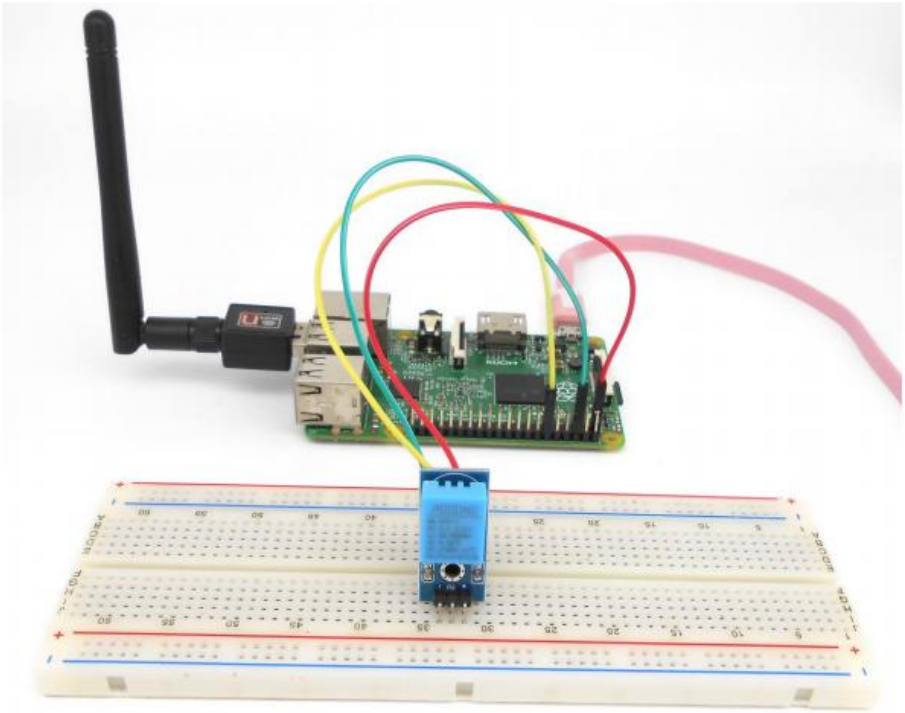
2.1 Edit and save the code with vim or nano.

(https://github.com/UCTRONICS/Arducam_Starter_Kit_Python_Code_for_RPi.git)

2.2 Run the program

```
$ sudo python 15_dht11.py
```

Now, you can see the temperature and humidity data displayed on the terminal.



8.16 Project 16: Dot-matrix display

8.16.1 Overview

In this lesson, we will program to control a 8x8 dot-matrix display to realize the display of graphical and digital we want.

8.16.2 Requirement

- Raspberry Pi x 1
- 8x8 Dot-matrix display x 1
- 74HC595 x 2
- Breadboard x 1
- Several Jumper wires

8.16.3 Principle

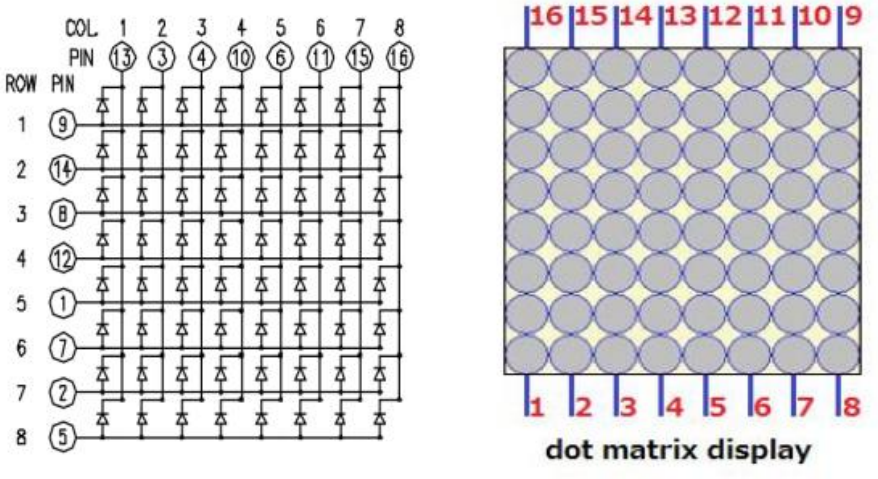
1. What's Dot-matrix display ?

Please refer to chapter 3.20.3 Dot-matrix Display

2. Working principle

The display consists of a dot-matrix of lights or mechanical indicators arranged in a rectangular configuration (other shapes are also possible, although not common) such that by switching on or off selected lights, text or graphics can be displayed. A dot-matrix controller converts instructions from a processor into signals which turns on or off lights in the matrix so that the required display is produced.

The internal structure and appearance of the dot-matrix display is as shown in below:



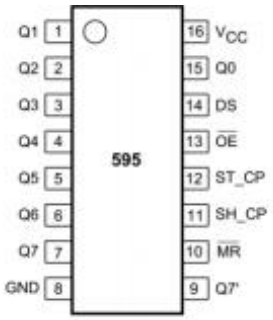
A 8x8 dot-matrix display consists of 64 LEDs, and each LED is placed at the intersection of the lines and columns. When the corresponding row is set as high level and the column is set as low level, then the LED will be lit.

A certain drive current is required for the dot-matrix display. In addition, more pins are needed for connecting dot-matrix display with controller. Thus, to save the Raspberry Pi's GPIO, driver IC 74HC595 is used in this experiment.

3. What's 74HC595 ?

The 74HC595 is an 8-stage serial shift register with a storage register and 3-state outputs. The shift register and storage register have separate clocks. Data is shifted on the positive-going transitions of the SH_CP input. The data in each register is transferred to the storage register on a positive-going transition of the ST_CP input. The shift register has a serial input (DS) and a serial standard output (Q7') for cascading. It is also provided with asynchronous reset (active LOW) for all 8 shift register stages. The storage register has 8 parallel 3-state bus driver outputs. Data in the storage register appears at the output whenever the output enable input (OE) is LOW.

In this experiment, only 3 pins of Raspberry Pi are used for controlling a dot-matrix display due to the existence of 74HC595.



The flowing is the function of each pin:

DS: Serial data input

Q0-Q7: 8-bit parallel data output

Q7: Series data output pin, always connected to DS pin of the next 74HC595

OE: Output enable pin, effective at low level, connected to the ground directly

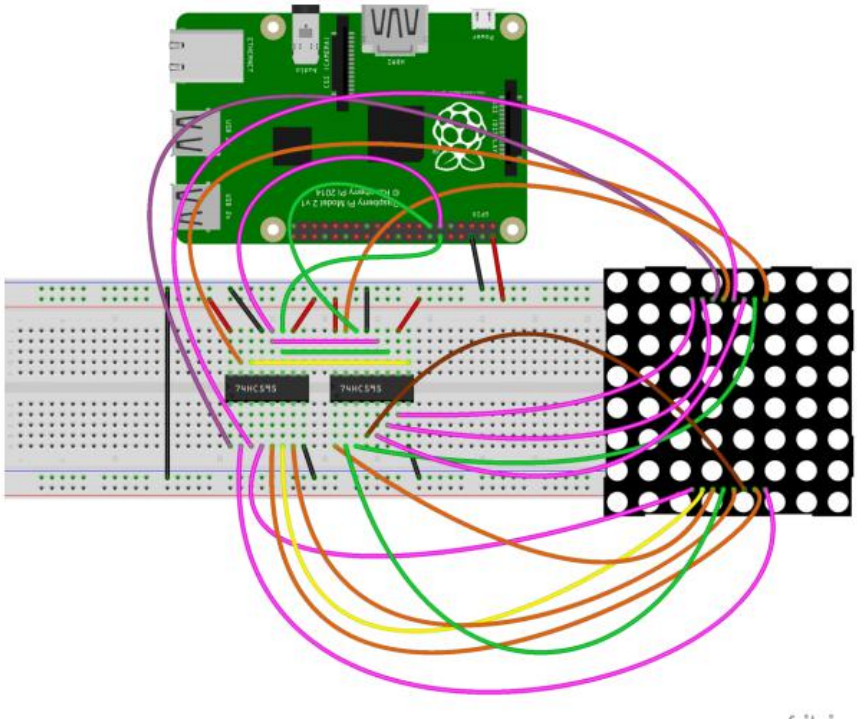
MR: Reset pin, effective at low level, directly connected to 5V high level in practical applications

SH_CP: Shift register clock input

ST_CP: storage register clock input

8.16.4 Procedures

1. Build the circuit (Make sure that the circuit connection is correct and then power, otherwise it may cause the chips to burn.)



2. Program

Python user:

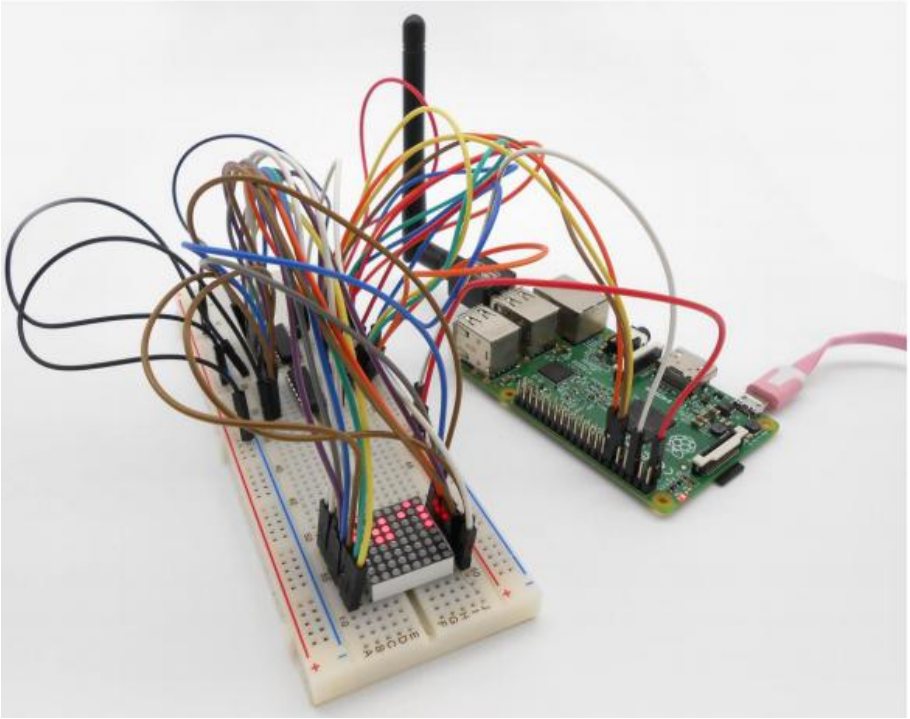
2.1 Edit and save the code with vim or nano.

(https://github.com/UCTRONICS/Arducam_Starter_Kit_Python_Code_for_RPi.git)

2.2 Run the program

```
$ sudo python 16_ledMatrix.py
```

Now, you can see a rolling "UCTRONICS" should be displayed on the dot-matrix display.



8.16.1 Summary

In this experiment, we have not only learned how to operate a dot-matrix display to display numbers and letters, but also learned the basic usage of 74HC595, then you can try operating the dot-matrix display to show other images.

8.17 Project 17: Photoresistor

8.17.1 Overview

In this lesson, we will learn how to measure the light intensity by photoresistor and make the measurement result displayed in the screen.

8.17.2 Requirement

- Raspberry Pi ×1
- ADC0832 ×1
- Photoresistor ×1
- 10KΩ Resistor ×1
- Breadboard ×1
- Several Jumper wires

8.17.3 Principle

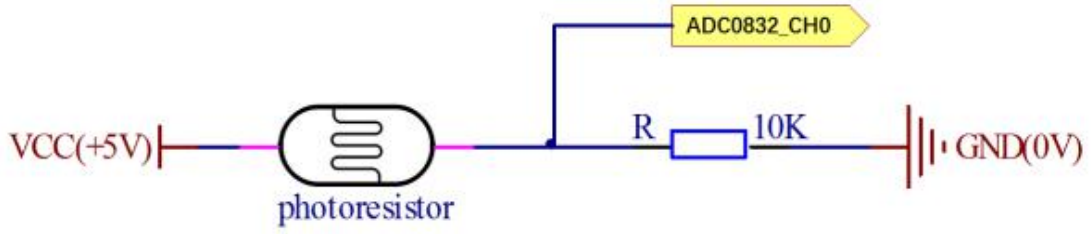
1. What's Photoresistor ?

Please refer to chapter 3.24 Light Sensor (Photoresistor)

2. Working Principle

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 photoresistors may react substantially differently to photons within certain wavelength bands.

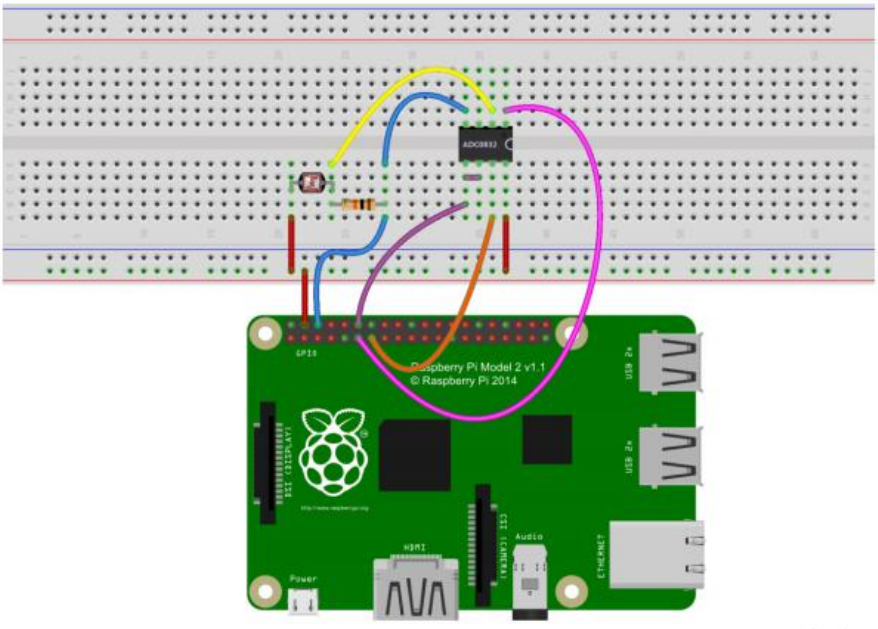
The schematic diagram of this experiment is shown below:



With the increase of the light intensity, the resistance of photoresistor will be decreased. The voltage of GPIO port in the above figure will become high.

8.17.4 Procedures

1. Build the circuit



2. Program

Python user:

2.1 Edit and save the code with vim or nano.

(https://github.com/UCTRONICS/Arducam_Starter_Kit_Python_Code_for_RPi.git)

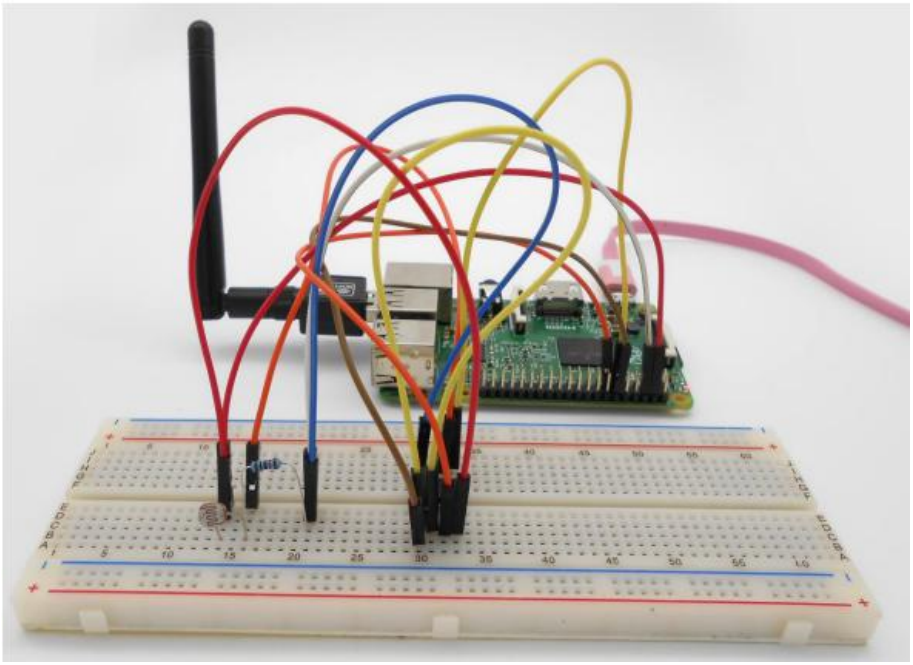
2.2 Run the program

```
$ sudo python 17_photoresistor.py
```

Now, when you try to block the light towards the photoresistor, you will find that the value displayed in the screen will be reduced. Otherwise, when you use a powerful light to irradiate the photoresistor, the value displayed will be increased.

8.17.5 Summary

By learning this lesson, we have learned how to detect surrounding light intensity with the photoresistor. You can play your own wisdom, and make more originality based on this experiment and the former experiment.



8.18 Project 18: Thermistor

8.18.1 Overview

In this lesson, we will learn how to use a thermistor to collect temperature by programming the Raspberry Pi and ADC0832.

8.18.2 Requirement

- Raspberry Pi ×1
- ADC0832 ×1
- Thermistor ×1
- 10KΩ Resistor ×1
- Breadboard ×1
- Several Jumper wires

8.18.3 Principle

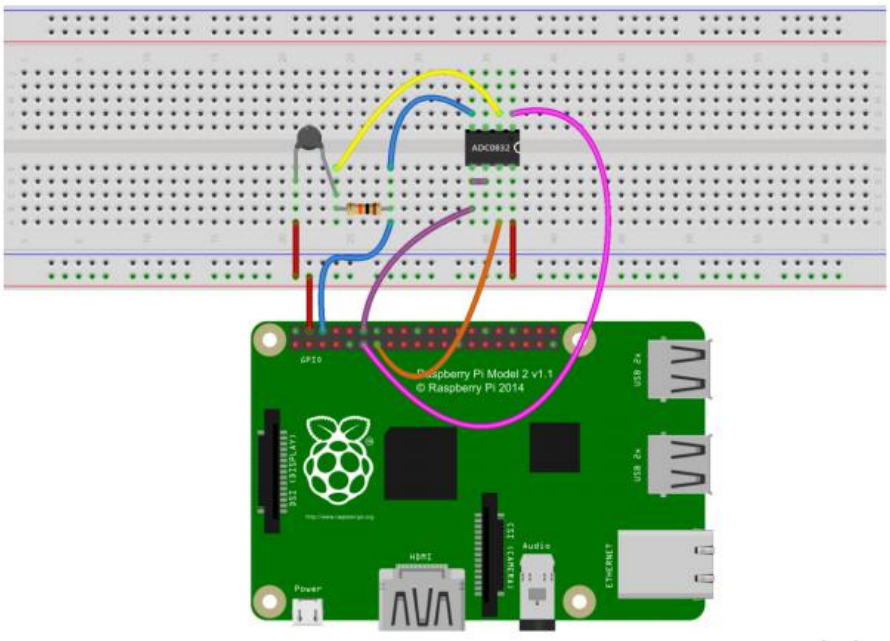
1. What's Thermistor ?

Please refer to chapter 3.25 Analog Temperature Sensor (Thermistor)

In the experiment, we need an analog-digital converter ADC0832 to convert analog signal into digital signal.

8.18.4 Procedures

1. Build the circuit



2. Program

Python user:

2.1 Edit and save the code with vim or nano.

(https://github.com/UCTRONICS/Arducam_Starter_Kit_Python_Code_for_RPi.git)

2.2 Run the program

```
$ sudo python 18_thermistor.py
```

Now, press Enter, if you touch the thermistor, you can see current temperature value displayed on the screen change accordingly.

8.19 Project 19: LED Bar Graph

8.19.1 Overview

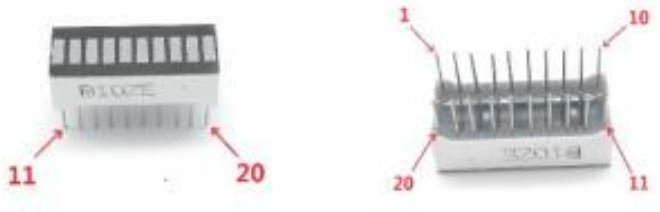
In this lesson, we will learn how to control an LED bar graph by programming the Raspberry Pi.

8.19.2 Requirement

- Raspberry Pi ×1
- ADC0832 ×1
- LED bar graph ×1
- 220Ω Resistor ×10
- 10KΩ Potentiometer ×1
- Breadboard ×1
- Several Jumper wires

8.19.3 Principle

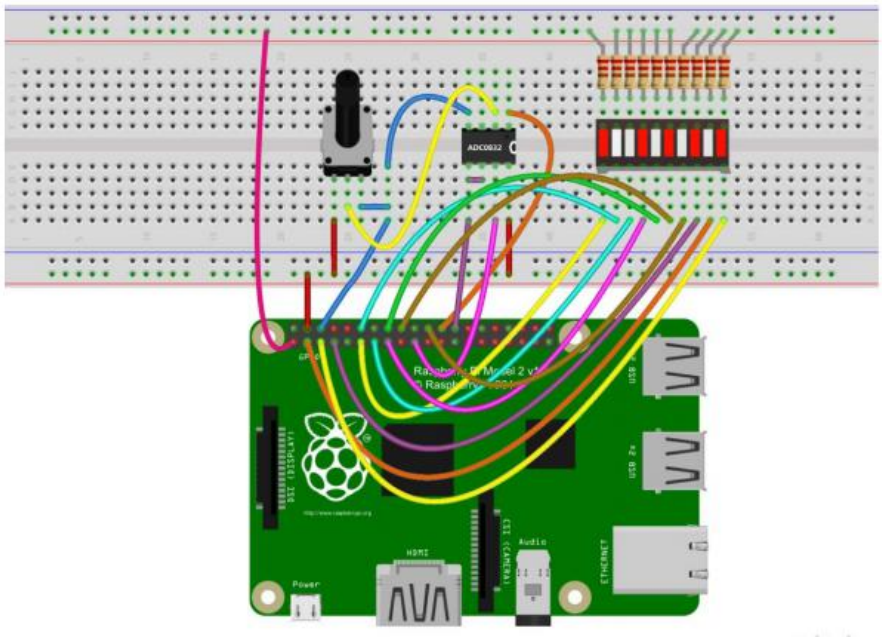
1. What's LED Bar Graph ?



The bar graph - a series of LEDs in a line, such as you see on an audio display - is a common hardware display for analog sensors. It's made up of a series of LEDs in a row, an analog input like a potentiometer, and a little code in between. You can buy multi-LED bar graph displays fairly cheaply. This tutorial demonstrates how to control a series of LEDs in a row, but can be applied to any series of digital outputs. This tutorial borrows from the For Loop and Arrays tutorial as well as the Analog Input tutorial. The sketch works like this: first you read the input. You map the input value to the output range, in this case ten LEDs. Then you set up a for loop to iterate over the outputs. If the output's number in the series is lower than the mapped input range, you turn it on. If not, you turn it off.

8.19.4 Procedures

1. Build the circuit



2.Program

Python user:

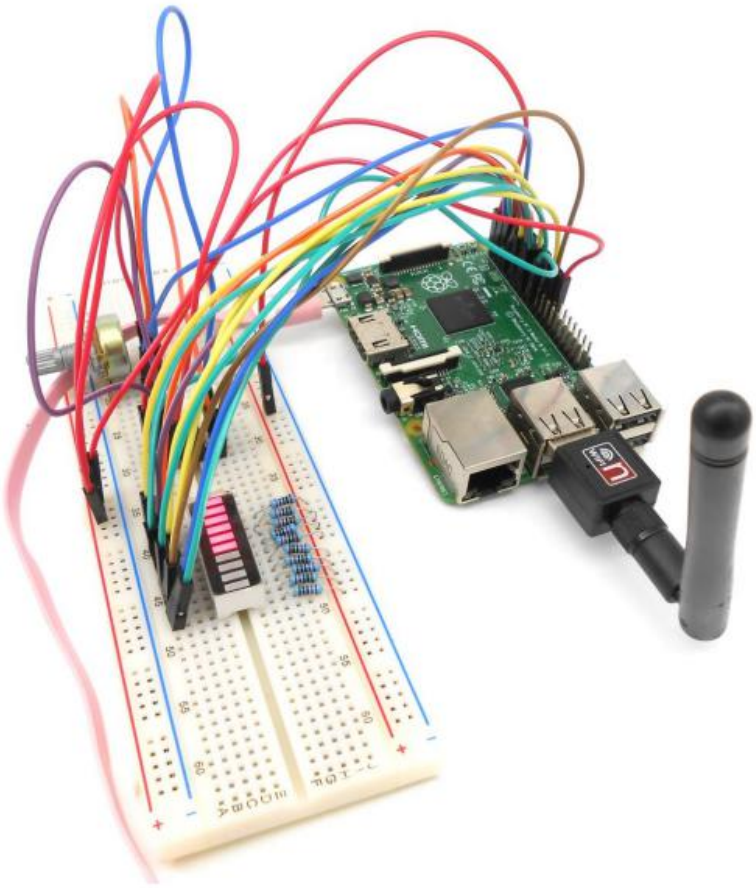
2.1 Edit and save the code with vim or nano.

(https://github.com/UCTRONICS/Arducam_Starter_Kit_Python_Code_for_RPi.git)

2.2 Run the program

```
$ sudo python ledBar.py
```

Now, when you turn the knob of the potentiometer, you will see that the number of LED in the LED bar graph will be changed.



8.20 Project 20: Controlling an LED through LAN

8.20.1 Overview

In this lesson, we will introduce TCP and socket, and then programming to control an LED through the local area network(LAN).

8.20.2 Requirement

- Raspberry Pi ×1
- LED ×1
- 220Ω Resistor ×1
- Breadboard×1
- Several Jumper wires

8.20.3 Principle

1. What's TCP ?

The Transmission Control Protocol (TCP) is a core protocol of the Internet Protocol Suite. It originated in the initial network implementation in which it complemented the Internet Protocol (IP). Therefore, the entire suite is commonly referred to as TCP/IP. TCP provides reliable, ordered, and error-checked delivery of a stream of octets between applications running on hosts communicating over an IP network. TCP is the protocol that major Internet applications such as the World Wide Web, email, remote administration and file transfer rely on. Applications that do not require reliable data stream service may use the User Datagram Protocol (UDP), which provides a connectionless datagram service that emphasizes reduced latency over reliability.

2. What's Socket ?

A network socket is an endpoint of an inter-process communication across a computer network. Today, most communication between computers is based on the Internet Protocol; therefore most network sockets are Internet sockets.

A socket API is an application programming interface (API), usually provided by the operating system, that allows application programs to control and use network sockets. Internet socket APIs are usually based on the Berkeley sockets standard.

A socket address is the combination of an IP address and a port number, much like one end of a telephone connection is the combination of a phone number and a particular extension. Based on this address, internet sockets deliver incoming data packets to the appropriate application process or thread.

Several Internet socket types are available:

1. Datagram sockets, also known as connectionless sockets, which use User Datagram Protocol (UDP).

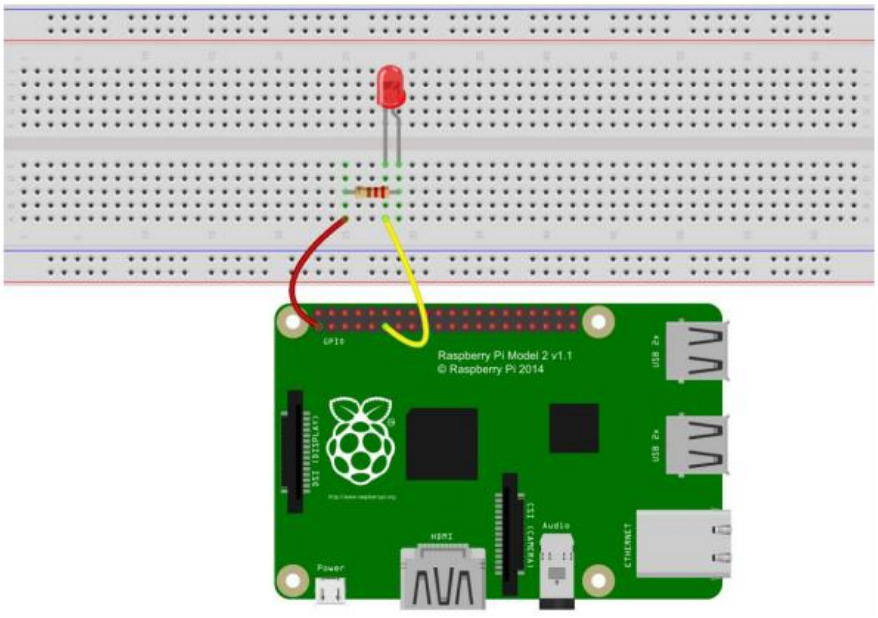
2. Stream sockets, also known as connection-oriented sockets, which use Transmission Control Protocol (TCP) or Stream Control Transmission Protocol (SCTP).

3. Raw sockets (or Raw IP sockets), typically available in routers and other network equipment. Here the transport layer is bypassed, and the packet headers are made accessible to the application.

In this experiment, our program is based on stream socket, and the program is divided into two parts, the client and the server. The server routine is run on the Raspberry Pi, and the client routine is run on the PC. So you can send command to the server through the client, and then control the LED connected to the Raspberry Pi.

8.20.4 Procedures

1. Build the circuit



2. Program

Python user:

2.1 Edit and save the server code with vim or nano on the Raspberry Pi.

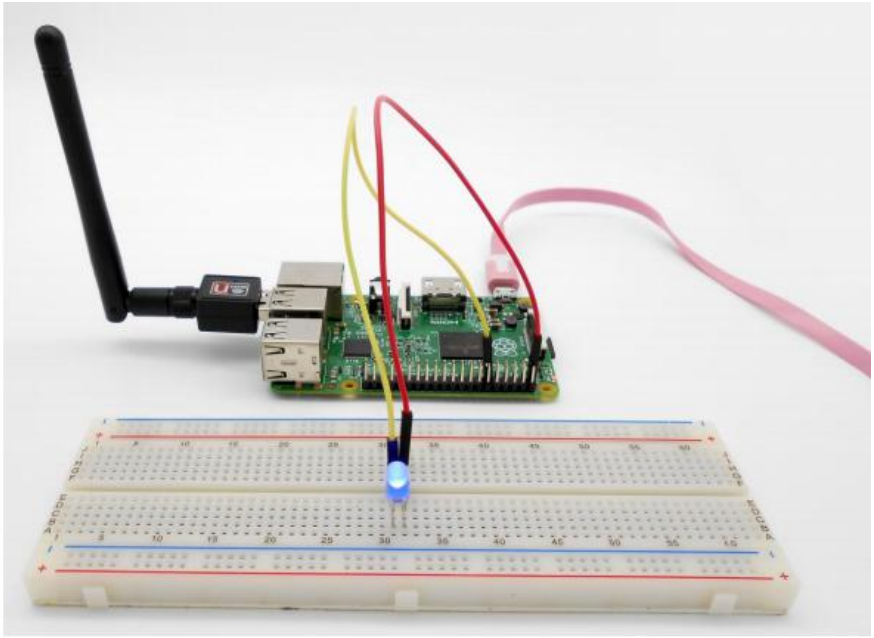
(https://github.com/UCTRONICS/Arducam_Starter_Kit_Python_Code_for_RPi.git)

2.2 Run the program

$ sudo python ledServer.py (On Raspberry Pi)

$ python ledClient.py (On PC)

Now, input "ON" in the terminal and then press Enter, you will find the LED connected to the Raspberry Pi is on, input "OFF", the LED is off.



8.20.5 Summary

By learning this lesson, you should have understood the basic principles of inter-computer communication. I hope this lesson will help you open the door to learn IoT(Internet of Things).

8.21 Project 21: Movement Detection Based on PIR

8.21.1 Overview

In this lesson, we will learn how to use the Passive Infrared (PIR) sensor to detect the movement nearby.

8.21.2 Requirement

- Raspberry Pi ×1
- PIR Movement Sensor ×1
- Several Jumper wires

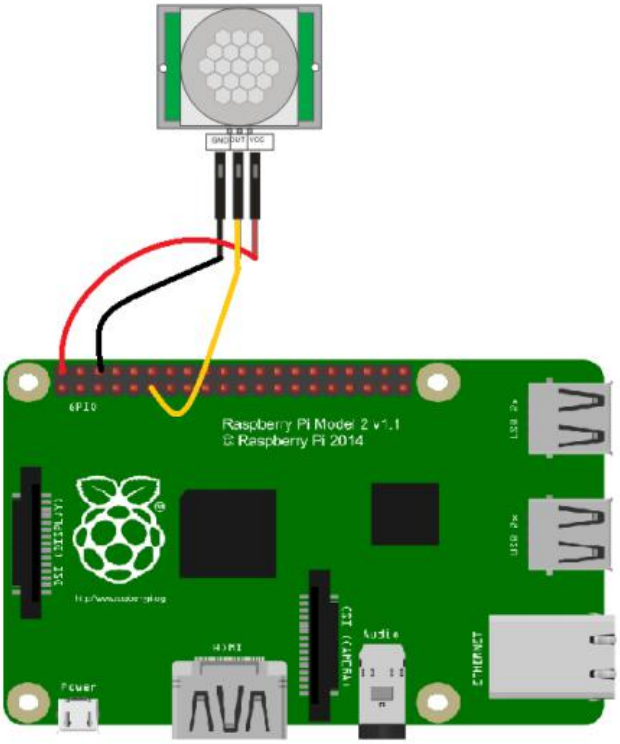
8.21.3 Principle

1. What's PIR sensor ?

Please refer to chapter 3.4 Pyroelectric Infrared PIR Motion Sensor Module

8.21.4 Procedures

1. Build the circuit



2. Program

Python user:

2.1 Edit and save the code with vim or nano.

(https://github.com/UCTRONICS/Arducam_Starter_Kit_Python_Code_for_RPi.git)

2.2 Run the program

```
$ sudo python 21_PIR.py
```

8.22 Project 22: DC Motor

8.22.1 Overview

In this comprehensive experiment, we will learn how to control the state of DC motor with Raspberry Pi. The state of DC motor includes its forward, reverse, acceleration, deceleration and stop.

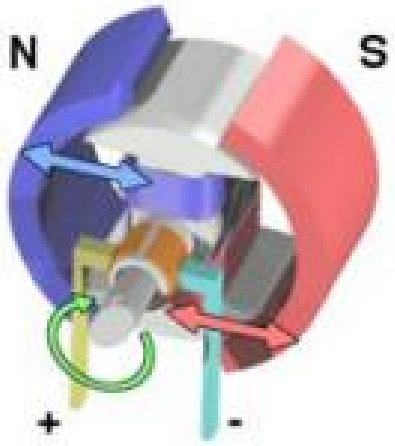
8.22.2 Requirement

- Raspberry Pi ×1
- L9110 DC Motor driver ×1
- DC motor ×1
- Button ×4
- LED ×1
- 220Ω Resistor ×1
- Capacitor(104, 0.1uF) ×1
- Breadboard ×1
- Several Jumper wires

8.22.3 Principle

1. What's DC motor ?

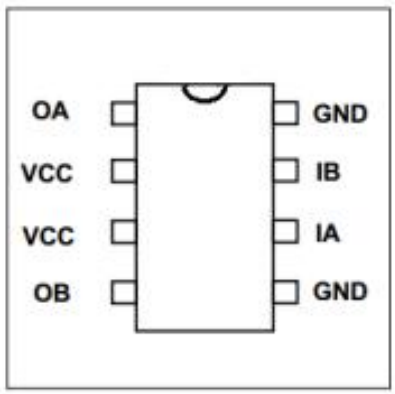
The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some internal mechanism, either electr mechanical or electronic, to periodically change the direction of current flow in part of the motor. Most types produce rotary motion; a linear motor directly produces force and motion in a straight line.



DC motors were the first type widely used, since they could be powered from existing direct-current lighting power distribution systems. A DC motor's speed can be controlled over a wide range, using either a variable supply voltage or by changing the strength of current in its field windings. Small DC motors are used in tools, toys, and appliances. The universal motor can operate on direct current but is a lightweight motor used for portable power tools and appliances.

2. What's L9110 ?

L9110 is a driver chip which is used to control and drive motor. The chip has two TTL/CMOS compatible input terminals, and possesses the property of anti-interference: it has high current driving capability, two output terminals that can directly drive DC motor, each output port can provide 750~800mA dynamic current, and its peak current can reach 1.5~2.0A; L9110 is widely applied to various motor drives, such as toy cars, stepper motor, power switches and other electric circuits.



OA, OB: These are used to connect the DC motor.

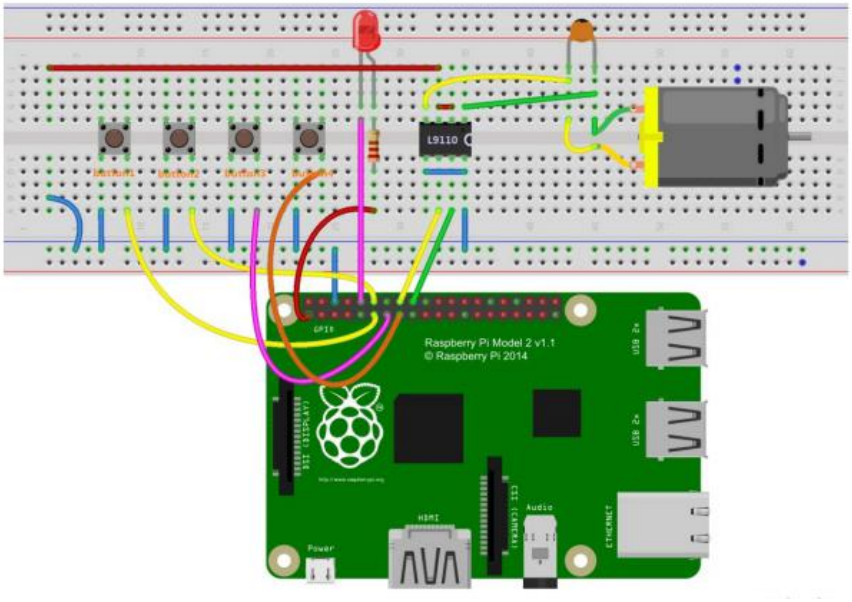
VCC: Power supply (+5V)

GND: The cathode of the power supply (Ground).

IA, IB: The input terminal of drive signal.

8.22.4 Procedures

1. Build the circuit



2. Program

Python user

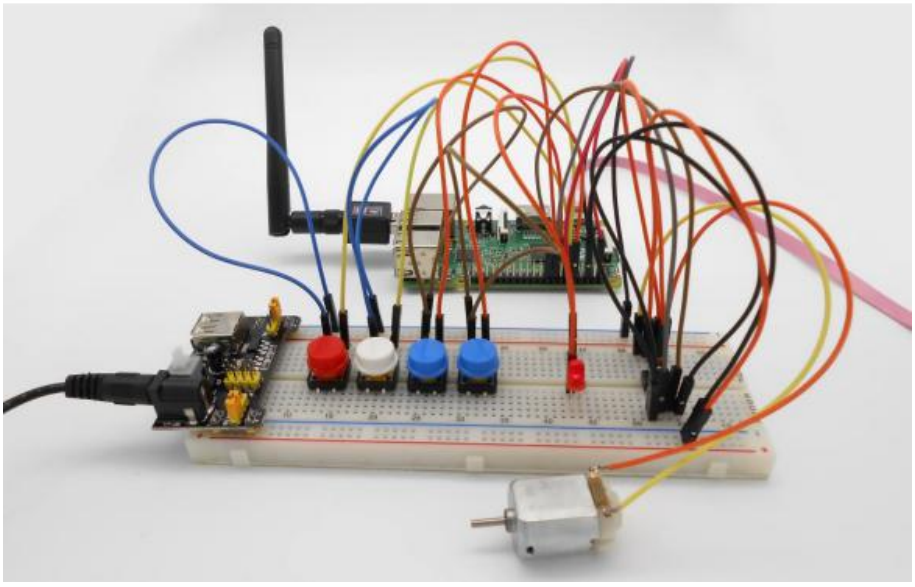
2.1 Edit and save the code with vim or nano.

(https://github.com/UCTRONICS/Arducam_Starter_Kit_Python_Code_for_RPi.git)

2.2 Run the program

```
$ sudo python 22_motor.py
```

Press the button 1 to stop or run the DC motor; press the button 2 to forward or reverse the DC motor; Press the button 3 to accelerate the DC motor; Press the button 4 to decelerate the DC motor. When the motor is running, the LED will be lit up. Otherwise, the LED will be extinguished.



8.22.5 Summary

I think you must have grasped the basic theory and programming of the DC motorafter studying this experiment. You not only can forward and reverse it, but also can regulate its speed. Besides, you can do some interesting applications with the combination of this course and your prior knowledge.

8.23 Project 23: How to control a servo

8.23. 1 Overview

In this lesson, we will introduce a new electronic device (Servo) to you, and tell you how to control it with Raspberry Pi.

8.23.2 Requirement

- Raspberry Pi ×1
- Servo ×1
- Breadboard ×1
- Several Jumper wires

8.23.3 Principle

1. What's Servo ?

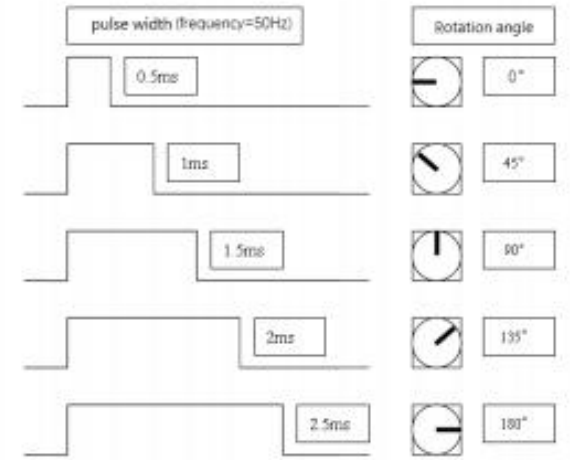
Please refer to chapter 3.8 SG90 micro small servo motor

2. Working principle

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.

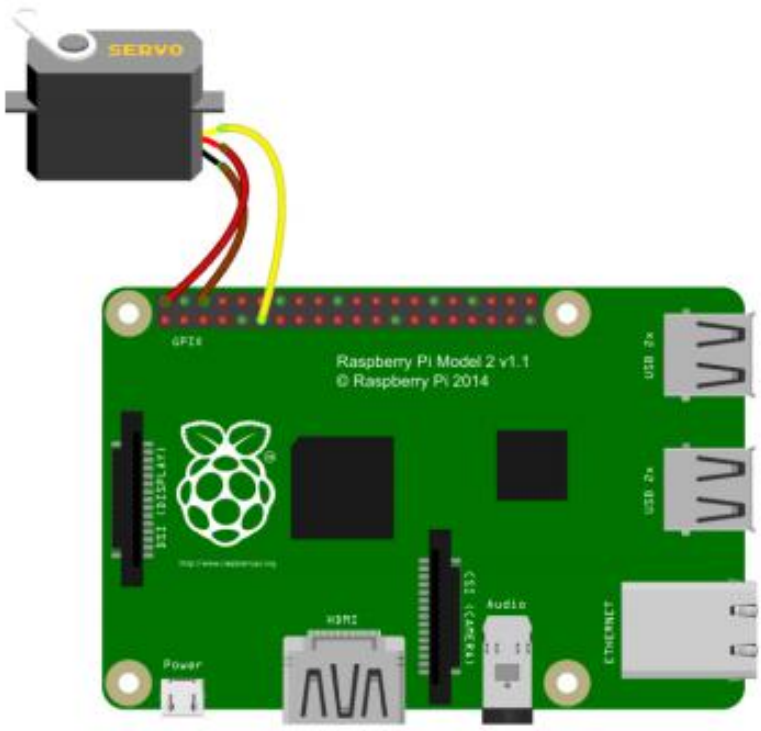
At the same time, position detector returns location signal to judge whether set location is reached or not.

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



8.23.4 Procedures

1. Build the circuit



2. Program

Python user:

2.1 Edit and save the code with vim or nano.

(https://github.com/UCTRONICS/Arducam_Starter_Kit_Python_Code_for_RPi.git)

2.2 Run the program

```
$ sudo python 23_servo.py
```

Press Enter, you should see the servo motor rotate 180 degrees. And then rotate in opposite direction.


TOWER PRO

8.24 Project 24: How to control a stepper motor

8.24.1 Overview

In this lesson, we will introduce a new electronic device — stepper motor to you, and tell you how to control it with Raspberry Pi.

8.24.2 Requirement

- Raspberry Pi ×1
- Stepper motor ×1
- ULN2003 stepper motor driver module ×1
- Several Jumper wires

8.24.3 Principle

1. What's Stepper motor

Please refer to chapter 3.9 DC 5V 4 Phase Step Stepper Motor

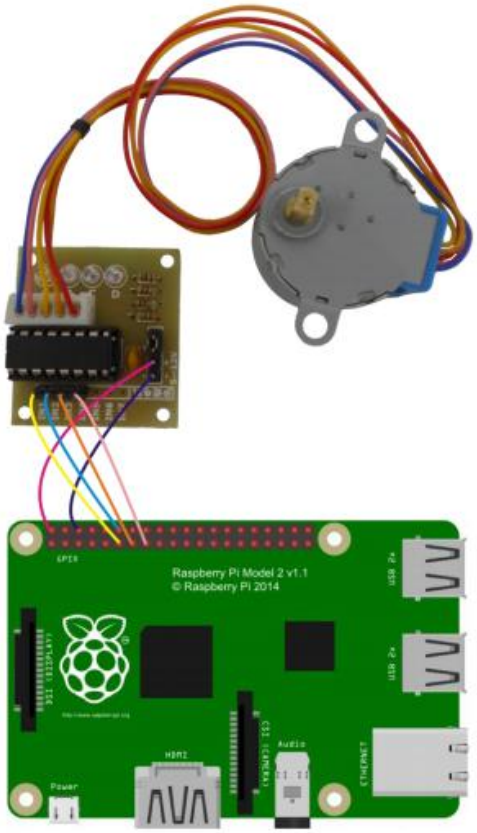
2. What's ULN2003 driver module

Please refer to chapter 3.7 ULN2003 Stepper Motor Driver Board

The Raspberry Pi's GPIO cannot directly drive a stepper motor due to the weak current. Therefore, a driver circuit is necessary for controlling a stepper motor. What we used in this experiment is a ULN2003-based driver module.

8.24.4 Procedures

1. Build the circuit



2. Program

Python user:

2.1 Edit and save the code with vim or nano.

(https://github.com/UCTRONICS/Arducam_Starter_Kit_Python_Code_for_RPi.git)

2.2 Run the program

```
$ sudo python 24_stepperMotor.py
```

Press Enter, you should see that the stepper motor is running.

8.25 Project: 25 How to use the acceleration sensor ADXL345

8.25.1 Overview

In this lesson, we will learn how to use an acceleration sensor ADXL345 to get acceleration data.

8.25.2 Requirement

- Raspberry Pi ×1
- ADXL345 module ×1
- Several Jumper wires

8.25.3 Principle

1. What's ADXL345 ?

Please refer to chapter 3.1 ADXL345 Triaxial Accelerometer Sensor Module

The ADXL345 is well suited for mobile device applications. It measures the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion or shock. Its high resolution (3.9 mg/LSB) enables measurement of inclination changes less than 1.0°.

Low power modes enable intelligent motion-based power management with threshold sensing and active acceleration measurement at extremely low power dissipation.

2. Key functions

- int wiringPiI2CSetup (int devId)

This initialises the I2C system with your given device identifier. The ID is the I2C number of the device and you can use the i2cdetect program to find this out. wiringPiI2CSetup() will work out which revision Raspberry Pi you have and open the appropriate device in /dev.

The return value is the standard Linux filehandle, or -1 if any error – in which case, you can consult errno as usual.

- int wiringPiI2CRead (int fd)

Simple device read. Some devices present data when you read them without having to do any register transactions.

- int wiringPiI2CWriteReg8 (int fd, int reg, int data)
- int wiringPiI2CWriteReg16 (int fd, int reg, int data)

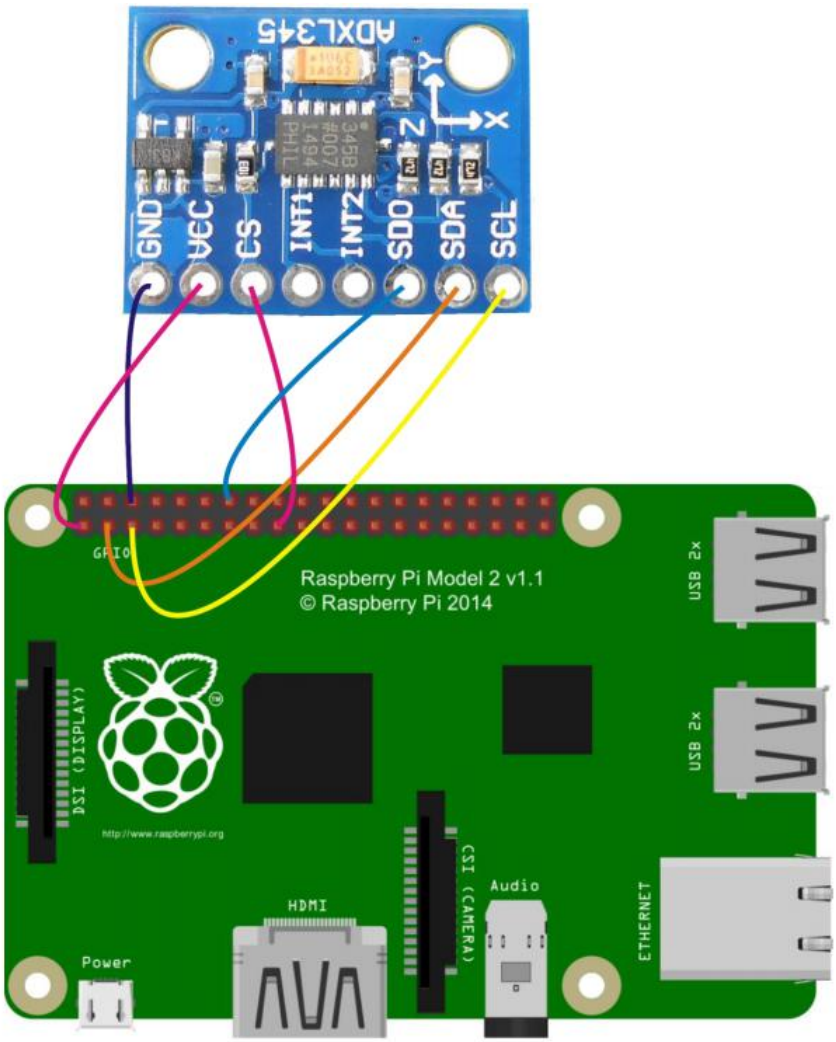
These write an 8 or 16-bit data value into the device register indicated.

- int wiringPiI2CReadReg8 (int fd, int reg)
- int wiringPiI2CReadReg16 (int fd, int reg)

These read an 8 or 16-bit value from the device register indicated.

8.25.4 Procedures

1. Build the circuit



2. Program

NOTE:

The following program use I2C interface. Before you run the program, please make sure the I2C driver module of Raspberry Pi has loaded normally.

Python user:

2.1 Edit and save the code with vim or nano.

(https://github.com/UCTRONICS/Arducam_Starter_Kit_Python_Code_for_RPi.git)

2.2 Run the program

```
$ sudo python Adafruit_ADXL345.py
```

Press Enter, you should see that the acceleration data will be displayed on the terminal.

8.26 Project 26: PS2 Joystick

8.26.1 Overview

In this lesson, we will learn the usage of joystick. We program the Raspberry Pi to detect the state of joystick.

8.26.2 Requirement

- Raspberry Pi ×1
- ADC0832 ×1
- PS2 Joystick ×1
- Breadboard ×1
- Several Jumper wires

8.26.3 Principle

1. What's Joysticks ?

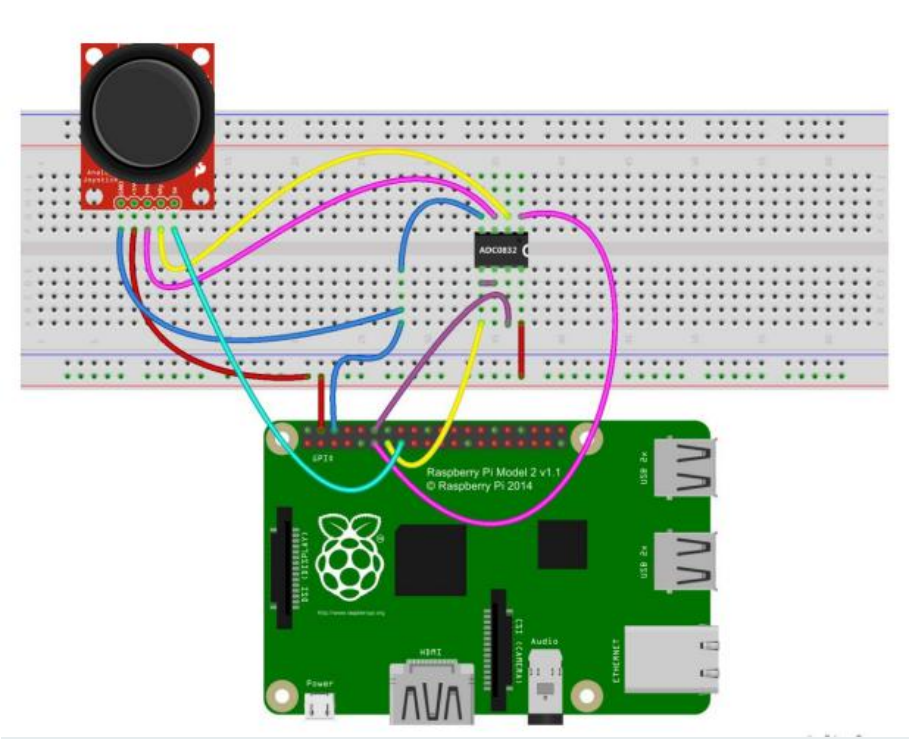
Please refer to chapter 3.5 PS2 Joystick Module

2. Application

Joysticks are often used to control video games, and usually have one or more push-buttons whose state an also be read by the computer. A popular variation of the joystick used on modern video game consoles is the analog stick. Joysticks are also used for controlling machines such as cranes, trucks, underwater unmanned vehicles, wheelchairs, surveillance cameras, and zero turning radius lawn mowers. Miniature finger-operated joysticks have been adopted as input devices for smaller electronic equipment such as mobile phones.

8.26.4 Procedures

1. Build the circuit



2. Program

Python user:

2.1 Edit and save the code with vim or nano.

(https://github.com/UCTRONICS/Arducam_Starter_Kit_Python_Code_for_RPi.git)

2.2 Run the program

```
$ sudo python 26_joystick.py
```

Press Enter, you should see that the joystick state information displayed on the terminal.