ELECROW

Raspberry Pi Pico

Advanced Kit

32 Advanced Projects to Play with Raspberry Pi Pico and Micro-Python

Contents

Raspberry Pi Pico Adavanced Kit

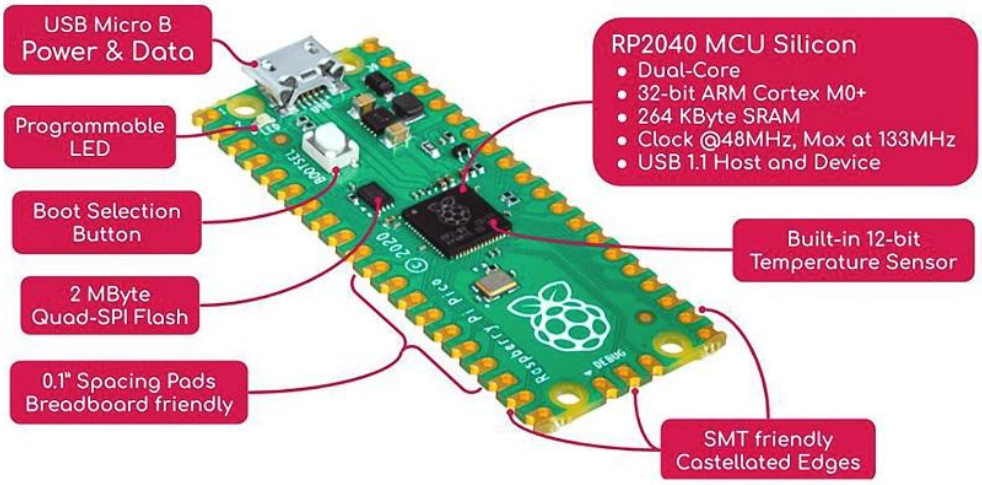
Lesson 1: Onboard LED Blinking Light

1.Introduction to Raspberry Pi Pico

1.1 Introduction

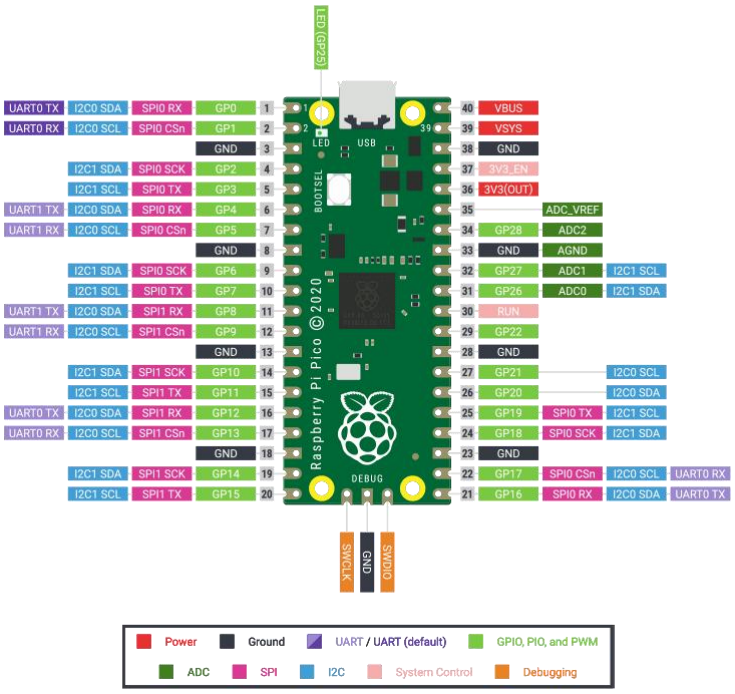
The Raspberry Pi Pico is a low-cost, high-performance microcontroller development board with a flexible digital interface. In terms of hardware, the RP2040 microcontroller chip, which is officially independently developed by Raspberry Pi, is equipped with an ARM Cortex M0 + dual-core processor, with a running frequency of up to 133MHz, built-in 264KB SRAM and 2MB memory, and up to 26 multi-function GPIO pins onboard. In terms of software, you can choose the C/C++ SDK provided by the Raspberry Pi, or use MicroPython for development, and there are complete development materials and tutorials, which can be easily developed and embedded in the product.

1.2 Configuration of Pico

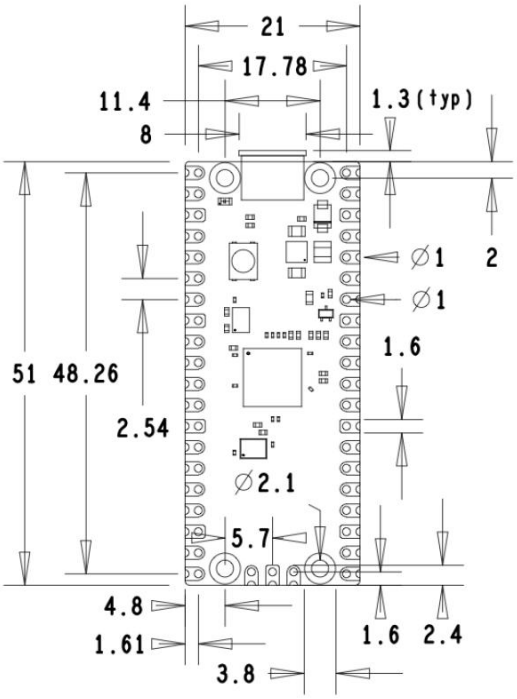


① Dual-core Arm Cortex-M0 + @ 133MHz;
② 2 UARTs, 2 SPI controllers and 2 I2C controllers;
③ The chip has built-in 264KB SRAM and 2MB onboard flash memory;
④ 16 PWM channels;
⑤ Support up to 16MB off-chip flash memory through dedicated QSPI bus;
⑥ USB 1.1 host and device support;
⑦ DMA controller;
⑧ 8 Raspberry Pi Programmable I/O (PIO) state machines for custom peripheral support;
⑨ 30 GPIO pins, 4 of which can be used as analog input;
⑩ Support USB mass storage boot mode of UF2 for drag-and-drop programming.

1.3 Pin Diagram



1.4 Size

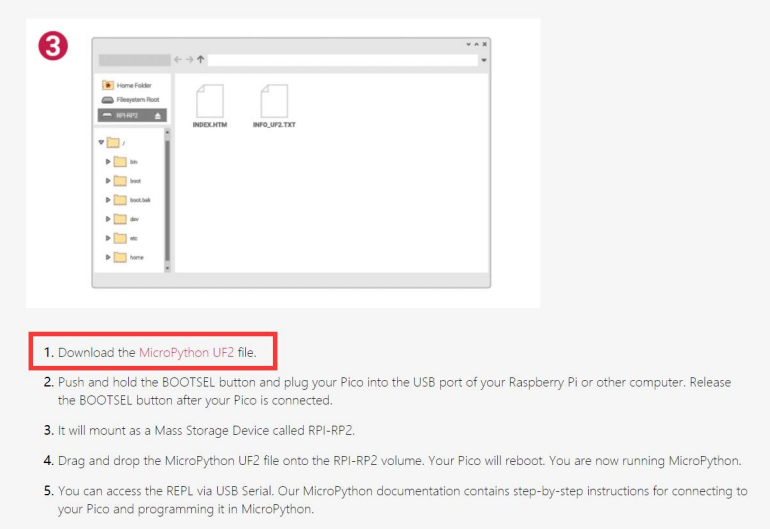


2.Programming environment

2.1 Burn firmware

Enter the official website and follow the instructions to download and burn the firmware:

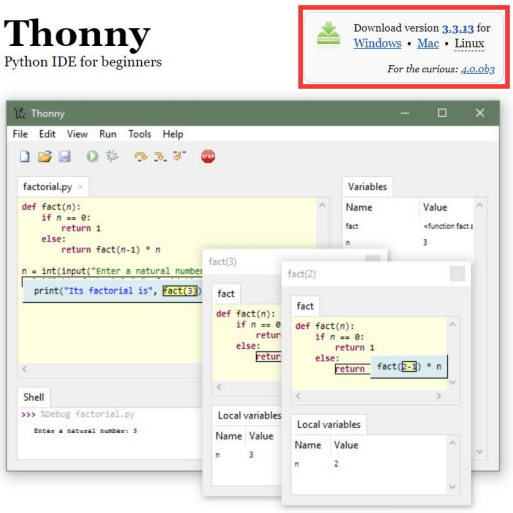
https://www.raspberrypi.com/documentation/microcontrollers/micropython.html



Operation instructions: Press and hold the BOOTSEL button, insert the USB cable connected to the Pico into the USB port of the computer, a new U disk folder will pop up on the computer, drag and drop the “UF2” file just downloaded to the folder, the Raspberry Pi Pico will restart automatically. In this way, the firmware burning is completed.

2.2 Install programming software

Enter the software official website **“https://thonny.org/”** to download the software, it is best to download the latest version, otherwise the Raspberry Pi Pico may not be supported;



Operation instructions: Install Thonny, open Thonny software after installation, open “Tools -> Settings -> Interpreter”, select MicroPython (Raspberry Pi Pico) interpreter, and select the serial port number of Raspberry Pi Pico at the serial port (If the board has been connected to the computer, the software will generally automatically detect the serial port number) and click OK, you can see that the files in the Raspberry Pi Pico are displayed at the bottom left of the software; if the file tree on the left is

not displayed, you can check the “view -> file”.

3.Run the program

3.1 Onboard LED Blinking light:

```
from machine import Pin
import utime
led = Pin(25, Pin.OUT)
if __name__ == '__main__':
    while True:
        led.value(1)
        utime.sleep_ms(100)
        led.value(0)
        utime.sleep_ms(100)
```

Operation instructions: Write a program, save the program to my computer, click the button or press the key **F5** to start the program; click the button or press the key **"Ctrl+F2"** to stop the program.

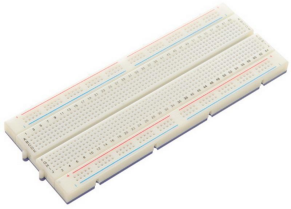
3.2 Run programs offline

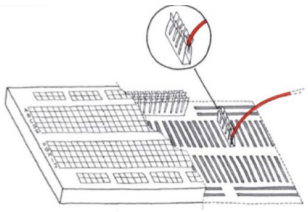
Create a new file, after writing the code, press the key “**ctrl+s”** to save the file on the Raspberry Pi Pico and name it **“main.py”** (must add the suffix **“.py”**), When the Pico is powered up again, the main.py program will run automatically.

Lesson 2: Breathing Light

1.Breadboard:

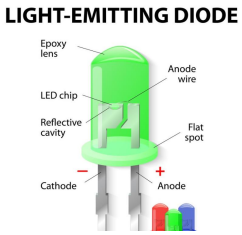
Breadboard is designed and manufactured for solderless experimentation of electronic circuits. Since there are many small jacks on the board, various electronic components can be inserted or pulled out at will, eliminating the need for soldering and saving the assembly time of the circuit, and the components can be reused, so it is very suitable for the assembly and debugging of electronic circuits.





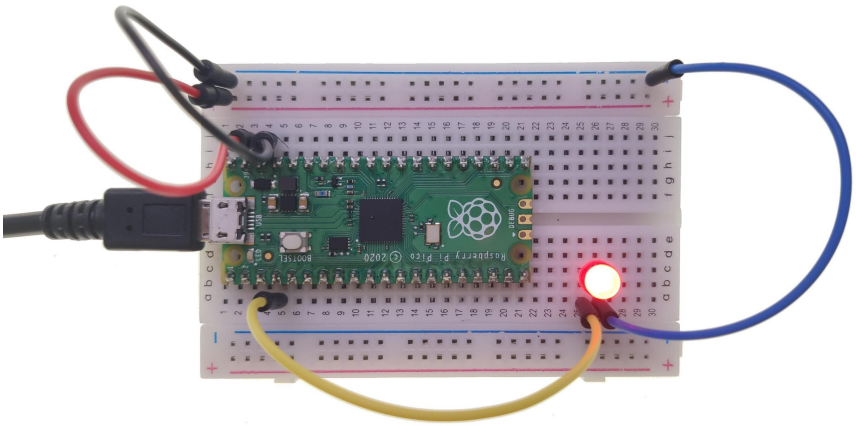
2.Light emitting diode:

Light emitting diode(short name:LED), is solid-state semiconductor devices that convert electricity into visible light. The core part of LED is a chip composed of P-type semiconductor and N-type semiconductor. When the current passes through the chip, the electrons in the N-type semiconductor and the holes in the P-type semiconductor collide and compound in the luminous layer to produce photons, and emit energy in the form of photons, namely visible light. Its long pin is positive, short pin is negative, the current is unidirectional from positive to negative.



3.Project Introduction:

Through PWM pulse width modulation, the LED light is controlled to light up gradually, and then gradually turn off, so as to form a breathing light effect in a cycle.



4.Circuit connection:

4.1 Material preparation: Raspberry Pi Pico*1, USB cable*1, Breadboard*1, LED*1, Dupont line;

4.2 Port connection:

Raspberry Pi Pico	LED
GP2	VCC (long pin)
GND	GND (short pin)

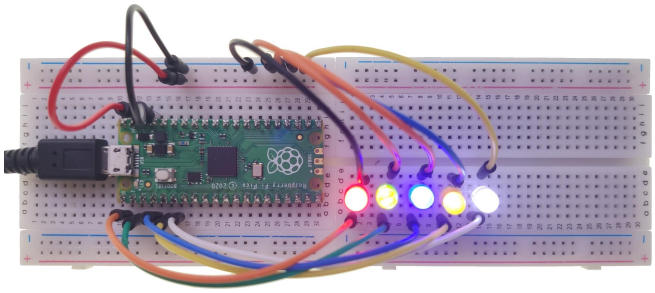
5.Program analysis: Breathing Light

```
from machine import Pin, PWM
import utime
led = PWM(Pin(2))
led.freq(1000)                          # Set the frequency value
led_value = 0                           #LED brightness initial value
led_speed = 5                           # Change brightness in increments of 5
if __name__ == '__main__':
    while True:
        led_value += led_speed
        led.duty_u16(int(led_value * 500))      # Set the duty cycle, between 0-65535
        utime.sleep_ms(100)
        if led_value >= 100:
            led_value = 100
            led_speed = -5
        elif led_value <= 0:
            led_value = 0
            led_speed = 5
```

Lesson 3: Colorful Flowing Light

1.Project Introduction:

Through the GP port of the Pico, the 5 LED lights are controlled to light up in turn, and then turn off in turn to form a Colorful flowing light.



2.Circuit connection:

2.1 Material preparation: Raspberry Pi Pico*1, USB cable*1, Breadboard*2, LED*5, Dupont line;

2.2 Port connection:

Raspberry Pi Pico	LED
GP0	Led1_VCC (+)
GP1	Led2_VCC (+)
GP2	Led3_VCC (+)
GP3	Led4_VCC (+)
GP4	Led5_VCC (+)
GND	GND (-)

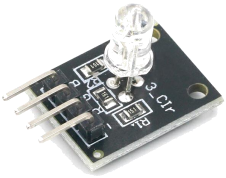
3.Program analysis: Colorful Flowing Light

```
from machine import Pin
import utime
leds = [Pin(i,Pin.OUT) for i in range(0,5)]
if __name__ == '__main__':
    while True:
        for n in range(0,5):
            leds[n].value(1)
            utime.sleep_ms(50)
        for n in range(0,5):
            leds[n].value(0)
            utime.sleep_ms(50)
```

Lesson 4: RGB Colorful Light

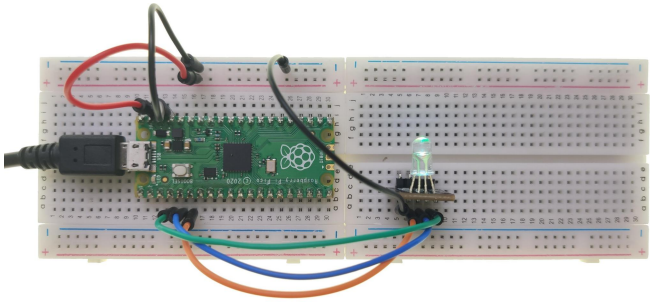
1.RGB LED:

The R,G, and B in RGB LED stand for red, green, and blue respectively. In theory, you can use some combination of these three colors to create any color; Through PWM voltage input can adjust the intensity of the three primary colors (red/green/blue), so as to achieve full color mixing effect.



2.Project Introduction:

Control the three primary colors of the RGB LED to display different brightness randomly, so as to realize the colorful light effect.



3.Circuit connection:

3.1 Material preparation: Raspberry Pi Pico*1, USB cable*1, Breadboard*2, RGB LED*1, Dupont line;

3.2Port connection:

Raspberry Pi Pico	RGB LED
GND	GND
GP2	R
GP3	G
GP4	B

4.Program analysis: RGB Colorful Light

```
from machine import Pin,PWM
import utime
import random
Led_R = PWM(Pin(2))
Led_G = PWM(Pin(3))
Led_B = PWM(Pin(4))
```

```
# Define the frequency
Led_R.freq(2000)
Led_G.freq(2000)
Led_B.freq(2000)
if __name__ == "__main__":
    while True:
        # range of random numbers
        R=random.randint(0,65535)
        G=random.randint(0,65535)
        B=random.randint(0,65535)
        print(R,G,B)
        Led_R.duty_u16(R)
        Led_G.duty_u16(G)
        Led_B.duty_u16(B)
        utime.sleep_ms(100)
```

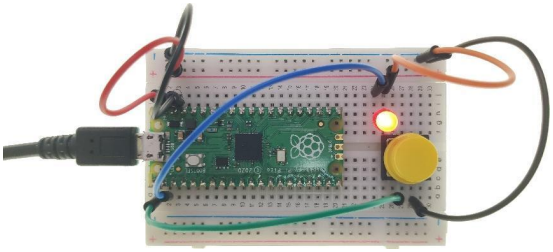
Lesson 5: Switch Light

1.Button:

Button, also known as tact switch, is a kind of electronic components. Its internal structure is connected and disconnected by the force change of metal shrapnel.When in use, press the switch, The switch is closed and the circuit is turned on. When released, the switch bounces and the circuit is disconnected.

2.Project Introduction:

The Raspberry Pi Pico detects the level change of the button to determine whether the button was pressed. Press the button to turn on the LED light for the first time, and press the button to turn off the LED light again, so as to realize the function of turning on and off the LED light.



3.Circuit connection:

3.1 Material preparation: Raspberry Pi Pico*1, USB cable*1, Breadboard*2, Button*1, LED*1, Dupont line;

3.2 Port connection:

Raspberry Pi Pico	Button
GND	Pin1(anyone)
GP0	Pin2(another one)

Raspberry Pi Pico	LED
GP1	VCC (+)
GND	GND (-)

4.Program analysis: Switch Light

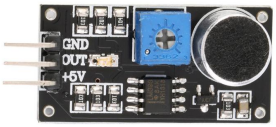
```
from machine import Pin
from utime import sleep_ms
button = Pin(0, Pin.IN, Pin.PULL_UP)      #Internal pull-up
led = Pin(1, Pin.OUT)
```

```
State=0                                  #0 means that the light is currently off
if __name__ == '__main__':
    while True:
        print(button.value())
        if button.value() == 0:          #key press
            if State==0:
                led.value(1)
                sleep_ms=100
                while button.value() == 0:
                    State=1
            else:
                led.value(0)
                sleep_ms=100
                while button.value()== 0:
                    State=0
```

Lesson 6: Voice-activated Light

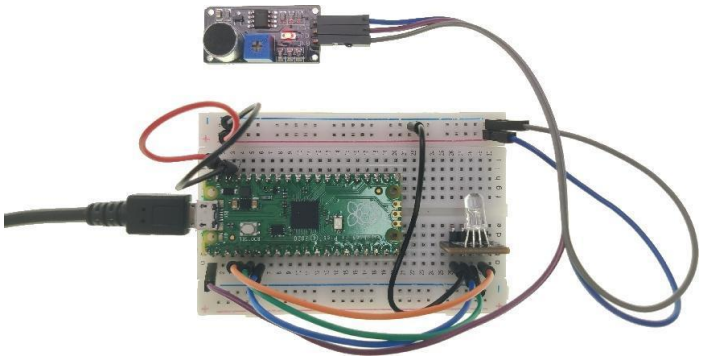
1.Sound sensor:

The sound sensor acts like a microphone , which is used to receive sound waves. The sensor has a built-in condenser electret microphone that is sensitive to sound. Sound waves make the electret film in the microphone vibrate, resulting in a change in capacitance and a small voltage corresponding to the change. This voltage is then converted into 0-5V voltage, which is received by the data collector after A/D conversion and transmitted to the microcontroller.



2.Project Introduction:

Detect the level change of the sound sensor through Raspberry Pi Pico (note: the potentiometer can adjust its sensitivity), when the sound loudness is greater than the threshold, the sensor is triggered, and then the RGB LED is controlled to light up, and then automatically turn off after waiting for 3 seconds.



3.Circuit connection:

3.1 Material preparation: Raspberry Pi Pico*1, USB cable*1, Breadboard*2, Sound sensor*1, RGB LED *1, Dupont line;

3.2Port connection:

Raspberry Pi Pico	Sound sensor
VSYS	+5V
GND	GND
GP0	OUT

Raspberry Pi Pico	RGB LED
GND	GND
GP2	R
GP3	G
GP4	B

4.Program analysis: Voice-activated Light

```
from machine import Pin,PWM
from utime import sleep_ms
sound = Pin(0, Pin.IN, Pin.PULL_DOWN)      # Port internal pull-down
Led_R = PWM(Pin(2))
Led_G = PWM(Pin(3))
Led_B = PWM(Pin(4))
Led_R.freq(2000)
Led_G.freq(2000)
Led_B.freq(2000)

if __name__ == '__main__':
    while True:
        print(sound.value())
        if sound.value() == 1:
            Led_R.duty_u16(65535)
            Led_G.duty_u16(65535)
            Led_B.duty_u16(65535)
            sleep_ms(2000)
        else:
            Led_R.duty_u16(0)
            Led_G.duty_u16(0)
            Led_B.duty_u16(0)
```

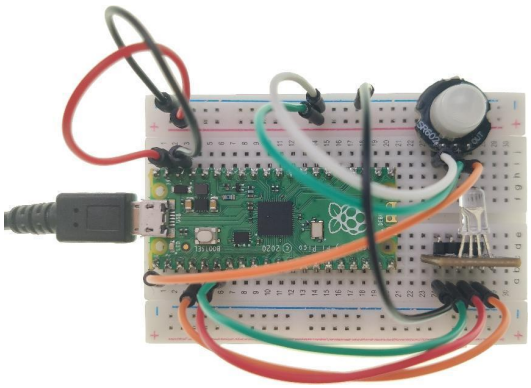
Lesson 7: Human Body Sensor Light

1.PIR sensor:

PIR sensor, also known as pyroelectric infrared sensor, is a sensor that uses infrared light for data processing.Since the human body has a constant body temperature, generally around 37 degrees, it emits infrared rays with a specific wavelength of about 10um.The infrared light emitted by the human body is enhanced by the Philippine filter and then concentrated on the infrared induction source.

2.Project Introduction:

Detect whether someone was present in the sensor detection area through the Raspberry Pi Pico. If a person is detected, the light will be turned on, and the light will be automatically turned off after a period of time.



3.Circuit connection:

3.1 Material preparation: Raspberry Pi Pico*1, USB cable*1, Breadboard*1, PIR sensor*1, RGB LED*1, Dupont line;

3.2 Port connection:

Raspberry Pi Pico	PIR Sensor
3V3	+
GND	-
GP0	OUT

Raspberry Pi Pico	RGB LED
GND	GND
GP2	R
GP3	G
GP4	B

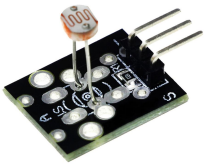
4.Program analysis: Human Body Sensor Light

```
from machine import Pin,PWM
from utime import sleep
PIR = Pin(0, Pin.IN, Pin.PULL_DOWN)
Led_R = PWM(Pin(2))
Led_G = PWM(Pin(3))
Led_B = PWM(Pin(4))
Led_R.freq(2000)
Led_G.freq(2000)
Led_B.freq(2000)
if __name__ == '__main__':
    while True:
        print(PIR.value())
        if PIR.value() == 1:
            Led_R.duty_u16(65535)
            Led_G.duty_u16(65535)
            Led_B.duty_u16(65535)
            sleep(3)
        else:
            Led_R.duty_u16(0)
            Led_G.duty_u16(0)
            Led_B.duty_u16(0)
```

Lesson 8: Smart Corridor Light

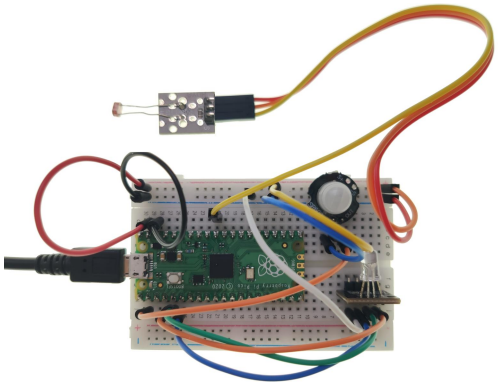
1.Light sensor:

The Light sensor is a sensor that uses a Light element to convert a light signal into an electrical signal. Its sensitive wavelength is near the wavelength of visible light, including infrared wavelengths and ultraviolet wavelengths. The light sensor is not only limited to the detection of light, it can also be used as a detection element to form other sensors to detect many non-electrical quantities, as long as these non-electrical quantities are converted into changes in optical signals.



2.Project Introduction:

According to the analog value fed back by the photosensitive sensor, it is judged whether it is day or night. When it is night, the PIR sensor is used to detect whether someone passes by, and if a human body is detected, the light is turned on, otherwise the light is turned off to achieve the effect of energy-saving lamps.



3.Circuit connection:

3.1 Material preparation: Raspberry Pi Pico*1, USB cable*1, Breadboard*2, Light sensor*1, PIR Sensor*1, RGB LED*1, Dupont line;

3.2 Port connection:

Raspberry Pi Pico	Light sensor
3V3	+
GND	-
GP26(ADC0)	S

Raspberry Pi Pico	PIR Sensor
3V3	+
GND	-
GP1	OUT

Raspberry Pi Pico	RGB LED
GND	GND (-)
GP2	R
GP3	G
GP4	B

4.Program analysis: Smart Corridor Light

```
from machine import Pin,PWM，ADC
light_sensor_pin = 0              # ADC0 multiplexing pin is GP26
PIR_pin = 1
Led_R_pin = 2
Led_G_Pin = 3
Led_B_pin = 4

def setup():
    global light_sensor_ADC
    global PIR
    global Led_R
    global Led_G
    global Led_B
    light_sensor_ADC = ADC(light_sensor_pin)
    PIR = Pin(PIR_pin, Pin.IN, Pin.PULL_DOWN)
    Led_R = PWM(Pin(Led_R_pin))
    Led_G = PWM(Pin(Led_G_Pin))
    Led_B = PWM(Pin(Led_B_pin))
    Led_R.freq(2000)
    Led_G.freq(2000)
    Led_B.freq(2000)
def loop():
    while True:
        print ('light_sensor Value: ', light_sensor_ADC.read_u16())
        Print ('PIR Value: ',PIR.value())
        if PIR.value() == 1 and light_sensor_ADC.read_u16() > 35000:
            Led_R.duty_u16(65535)
            Led_G.duty_u16(65535)
            Led_B.duty_u16(65535)
        else:
            Led_R.duty_u16(0)
            Led_G.duty_u16(0)
            Led_B.duty_u16(0)
if __name__ == '__main__':
    setup()
    loop()
```

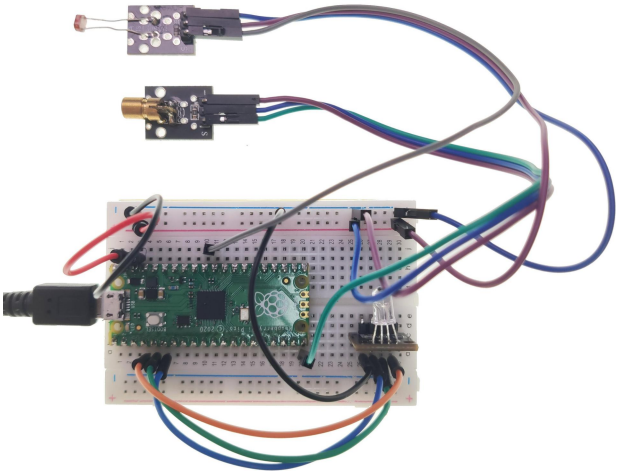
Lesson 9: Laser Sight

1.Red laser emitter:

A red laser emitter is a device capable of emitting laser light, which has the characteristics of high directivity, high monochromaticity and high brightness. It consists of a 650nm red laser diode head and a resistor. When powered up, it emits laser pulses.

2.Project Introduction:

Start the program, RGB LED initially displays green, use the red laser emitter to aim at the Light sensor and start timing, when it locks on the target (Light sensor), the RGB flashes red and blue, and stops timing.



3.Circuit connection:

3.1 Material preparation: Raspberry Pi Pico*1, USB cable*1, Breadboard*1, Red laser emitter*1, Light sensor*1, RGB LED*1, Dupont line;

3.2 Port connection:

Raspberry Pi Pico	Light sensor
3V3	VCC (+)
GND	GND (-)
GP26(ADC0)	S

Raspberry Pi Pico	RGB LED
GND	GND
GP11	R
GP12	G
GP13	B

Raspberry Pi Pico	Red laser emitter
GND	GND (-)
3V3	VCC (+)
GP15	S

4.Program analysis: Laser Sight

```
from machine import Pin,ADC,PWM
from time import sleep
photo = 0                       # ADC0 multiplexing pin is GP26
LaserPin = 15                   #Red laser transmitter
Led_R = PWM(Pin(11))
Led_G = PWM(Pin(12))
Led_B = PWM(Pin(13))
Led_R.freq(2000)
Led_G.freq(2000)
Led_B.freq(2000)
def setup():
    global Laser
    global photo_ADC
    photo_ADC = ADC(photo)
    Laser = Pin(LaserPin,machine.Pin.OUT)
    Laser.value(0)

def loop():
    aim_time = 0
    while True:
        Laser.value(1)
        Led_R.duty_u16(0)
        Led_G.duty_u16(65535)
        Led_B.duty_u16(0)
        print("time:  ",aim_time)
        sleep(0.1)
        aim_time += 0.1
        if photo_ADC.read_u16() < 15000:        # Hit by a red laser, the light flashes
            print("Aimed:",aim_time)
            aim_time = 0
            for i in range(10):
                Led_R.duty_u16(65535)           # red light
                Led_G.duty_u16(0)
                Led_B.duty_u16(0)
                sleep(0.1)
                Led_R.duty_u16(0)               # blue light
```

```
                    Led_G.duty_u16(0)
                    Led_B.duty_u16(65535)
                    sleep(0.1)
                sleep(2)
if __name__ == '__main__':
    setup()
    loop()
```

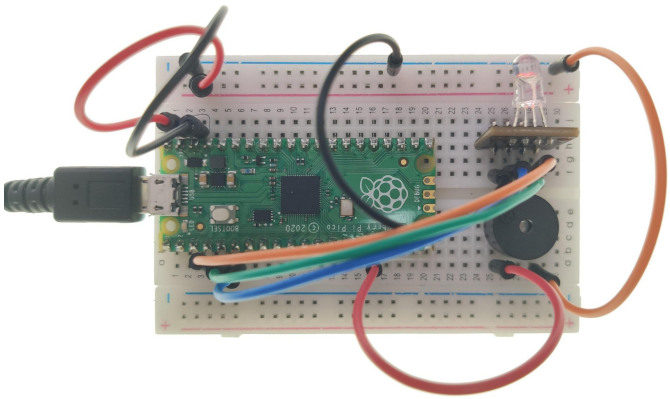
Lesson 10: Police Car Lights

1.Passive buzzer:

The passive buzzer uses the phenomenon of electromagnetic induction to attract or repel the electromagnet and the permanent magnet formed after the voice coil is connected to the alternating current to push the diaphragm to sound.When DC power is connected, the diaphragm can only be pushed continuously without sound, and sound can only be generated when it is connected or disconnected.

2.Project Introduction:

Through the buzzer and RGB LED lights alternately beeping and flashing to achieve the effect of police car lights.



3.Circuit connection:

3.1 Material preparation: Raspberry Pi Pico*1, USB cable*1, Breadboard*1, Buzzer*1, RGB LED*1, Dupont line;

3.2 Port connection:

Raspberry Pi Pico	Buzzer
GP12	VCC (+)
GND	GND (-)

Raspberry Pi Pico	RGB LED
GND	GND
GP4	R
GP3	G
GP2	B

4.Program analysis: Police Car Lights

```
import utime
from machine import Pin,PWM
Led_R = PWM(Pin(4))
Led_G = PWM(Pin(3))
Led_B = PWM(Pin(2))
buzzer = PWM(Pin(12))
Led_R.freq(2000)
Led_G.freq(2000)
Led_B.freq(2000)
buzzer.duty_u16(1000)
if __name__ == '__main__':
    while True:
        buzzer.freq(750)
        Led_R.duty_u16(0)
        Led_G.duty_u16(0)
        Led_B.duty_u16(65535)
        utime.sleep_ms(230)
        buzzer.freq(1550)
        Led_R.duty_u16(65535)
        Led_G.duty_u16(0)
        Led_B.duty_u16(0)
        utime.sleep_ms(100)
```

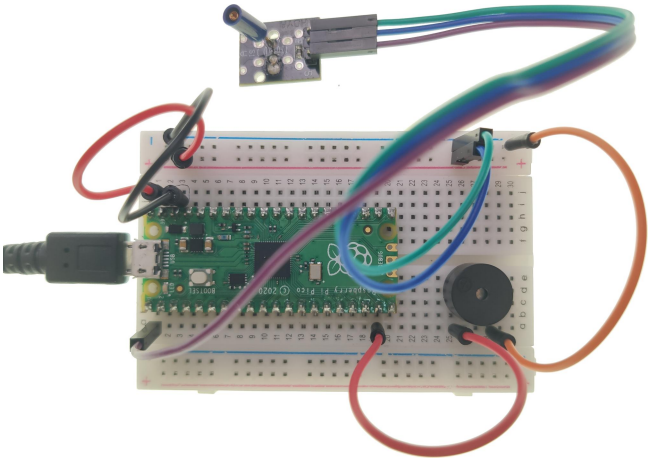
Lesson 11: Anti-theft Alarm

1.Vibration sensor:

A vibration sensor, also called a vibration switch, is a commonly used alarm detection sensor.The switch is in the open-circuit OFF state when it is at rest. When it is touched by an external force and reaches the corresponding vibration, the conductive pin will generate instantaneous conduction and become an instantaneous ON state. When the external force disappears, the open-circuit OFF state will be restored.

2.Project Introduction:

The level signal of the vibration sensor is detected by the Raspberry Pi Pico, and when vibration is detected, the buzzer is controlled to sound to achieve the effect of an anti-theft alarm.



3.Circuit connection:

3.1 Material preparation: Raspberry Pi Pico*1, USB cable*1, Breadboard*1, Vibration sensor*1, Buzzer*1, Dupont line;

3.2 port connection:

Raspberry Pi Pico	Buzzer
GP14	VCC (+)
GND	GND (-)

Raspberry Pi Pico	Vibration sensor
GND	GND (-)
VCC	VCC (+)
GP0	S

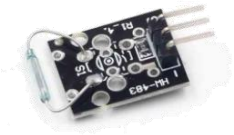
4.Program analysis: Anti-theft Alarm

```
from machine import Pin,PWM
import utime
vibrate = Pin(0, Pin.IN, Pin.PULL_UP)
buzzer = PWM(Pin(14))
def playtone(frequency):          # Buzzer play function
    buzzer.duty_u16(1000)
    buzzer.freq(frequency)

def bequiet():                    # stop play function
    buzzer.duty_u16(0)

if __name__ == '__main__':
    while True:
        If vibrate.value() == 0:
            for i in range(10) :
                playtone(555)
                utime.sleep_ms(50)
                bequiet()
                utime.sleep_ms(50)
        bequiet()
```

Lesson 12: Music Box

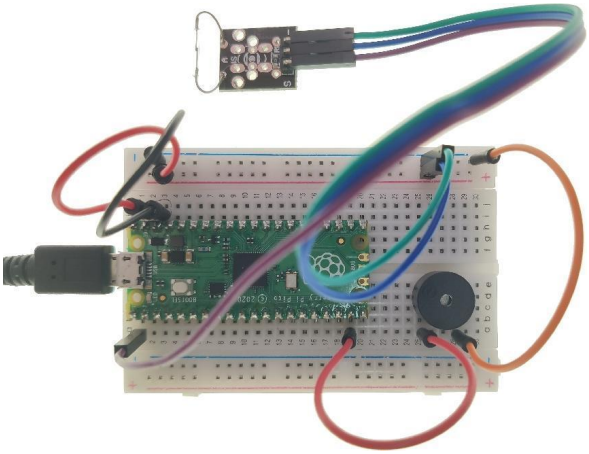
1.Reed switch:

Reed Switch is a special magnetically sensitive switch. Inside the reed switch, there are two magnetizable reeds overlapped at the ends, sealed in a glass tube, and the distance between the two reeds is only about a few microns. The glass tube is filled with high-purity inert gas. During operation, the two reeds are not in contact, and the external magnetic field causes different polarities near the end positions of the two reeds, and as a result, the two reeds with different polarities will attract and close each other.



2.Project Introduction:

Place the small magnet close to the reed switch, and then trigger the buzzer to play music by detecting the level signal change of the reed switch to achieve the effect of a music box.



3.circuit connection:

3.1 Material preparation: Raspberry Pi Pico*1, USB cable*1, Breadboard*1, Buzzer*1, Reed Switch*1, magnet*1, Dupont line;

3.2 Circuit connection:

Raspberry Pi Pico	Buzzer
GP14	+
GND	-

Raspberry Pi Pico	Reed Switch
GND	GND (-)
VCC	VCC (+)
GP0	S

4.Program analysis: Music Box

```
from machine import Pin,PWM
import utime

Reed_switch = Pin(0, Pin.IN, Pin.PULL_UP)
buzzer = PWM(Pin(14))

Tone_CL = [0, 131, 147, 165, 175, 196, 211, 248]          # low C note frequency
Tone_CM = [0, 262, 294, 330, 350, 393, 441, 495]          # middle C note frequency
Tone_CH = [0, 525, 589, 661, 700, 786, 882, 990]          # high C note frequency

# sheet music
song_1 = [ Tone_CM[3], Tone_CM[5], Tone_CM[6], Tone_CM[3],
           Tone_CM[2], Tone_CM[3], Tone_CM[5], Tone_CM[6],
           Tone_CH[1], Tone_CM[6], Tone_CM[5], Tone_CM[1],
           Tone_CM[3], Tone_CM[2], Tone_CM[2], Tone_CM[3],
           Tone_CM[5], Tone_CM[2], Tone_CM[3], Tone_CM[3],
           Tone_CL[6], Tone_CL[6], Tone_CL[6], Tone_CM[1],
           Tone_CM[2], Tone_CM[3], Tone_CM[2], Tone_CL[7],
           Tone_CL[6], Tone_CM[1], Tone_CL[5] ]
# The beat of the song
beat_1 = [ 1, 1, 3, 1, 1, 3, 1, 1,
           1, 1, 1, 1, 1, 1, 3, 1,
           1, 3, 1, 1, 1, 1, 1, 1,
           1, 2, 1, 1, 1, 1, 1, 1,
           1, 1, 3 ]
# Buzzer play function
def playtone(frequency):
    buzzer.duty_u16(1000)
    buzzer.freq(frequency)
# Stop play function
def bequiet():
    buzzer.duty_u16(0)

if __name__ == '__main__':
    while True:
        if Reed_switch.value() == 0:
            for i in range(1, len(song_1)):       # play song
                playtone(song_1[i])               # Set the frequency of the song notes
                sleep(beat_1[i] * 0.5)            # Delay a note by one beat * 0.5 seconds
        sleep(1)
        bequiet()
```

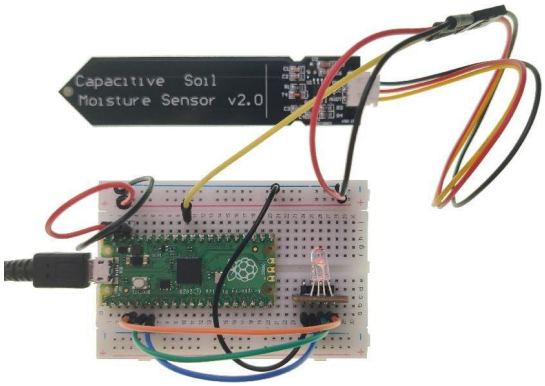
Lesson: 13 Plant Doctor

1.Soil Moisture Sensor:

The soil moisture sensor can be inserted into the soil to measure the relative soil moisture content, which is commonly used in soil moisture monitoring, agricultural irrigation and forestry protection. This capacitive soil moisture sensor is different from most resistive sensors on the market, avoiding the problem that resistive sensors are easily corroded and greatly extending its working life.

2.Project Introduction:

Real-time detection of soil humidity. Under normal soil humidity, the green light lights up and the buzzer does not sound; when the soil humidity is too low, the red light lights up and the buzzer sounds.



3.Circuit connection:

3.1 Material preparation: Raspberry Pi Pico*1, USB cable*1, Breadboard*1, Soil Moisture Sensor*1, Buzzer*1, RGB LED*1, Dupont Line;

3.2 Port connection:

Raspberry Pi Pico	Soil Moisture Sensor
3V3	VCC
GND	GND
GP26	AUOT

Raspberry Pi Pico	Buzzer
GND	-
GP14	+

Raspberry Pi Pico	RGB LED
GP11	R
GP12	G
GP13	B
GND	GND

4.Program analysis: Plant Doctor

```
from machine import Pin,ADC,PWM
from time import sleep
import math

Soil_moisture_pin = 0            # ADC0 multiplexing pin is GP26
buzzer = PWM(Pin(14))
Led_R = PWM(Pin(11))
Led_G = PWM(Pin(12))
Led_B = PWM(Pin(13))
Led_R.freq(2000)
Led_G.freq(2000)
Led_B.freq(2000)
def setup():
    global Moisture
    Moisture =   ADC(Soil_moisture_pin)
def playtone(frequency):
    buzzer.duty_u16(1000)
    buzzer.freq(frequency)
def bequiet():
    buzzer.duty_u16(0)
# Information about the Soil Moisture Sensor
def Print(x):
    if x > 20000:                          # soil water shortage
        playtone(330)
        Led_R.duty_u16(65535)
        Led_G.duty_u16(0)
        Led_B.duty_u16(0)
    If 15000 < x and x < 20000:            # proper soil moisture
        bequiet()
        Led_R.duty_u16(0)
        Led_G.duty_u16(65535)
        Led_B.duty_u16(0)
def loop():
    while True:
        Moist = Moisture.read_u16()
        print ('temperature = ', Moist)
        Print(Moist)
        sleep(0.2)

if __name__ == '__main__':
    setup()
    loop()
```

Lesson 14: Dimming Desk Light

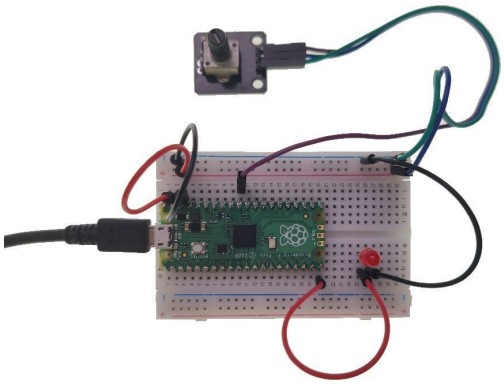
1.Potentiometer:

A potentiometer is a resistive element with three terminals whose resistance can be adjusted according to a certain changing law. A potentiometer usually consists of a resistive body and a movable brush. When the brush moves along the resistor body, the resistance value or voltage that has a certain relationship with the displacement is obtained at the output end.

Potentiometers can be used as both three-terminal components and two-terminal components. The latter can be regarded as a variable resistor, because its function in the circuit is to obtain an output voltage that has a certain relationship with the input voltage (applied voltage), so it is called a potentiometer.

2.Project Introduction:

Rotate the potentiometer to control the brightness of the lamp to achieve the effect of dimming the desk lamp.



3.Circuit connection:

3.1 Material preparation: Raspberry Pi Pico*1, USB cable*1, Breadboard*1, Potentiometer*1, LED*1, Dupont line;

3.2 Port connection:

Raspberry Pi Pico	Potentiometer
VSYS	VCC
GND	GND
GP26	OUT

Raspberry Pi Pico	LED
GND	GND (-)
GP15	VCC (+)

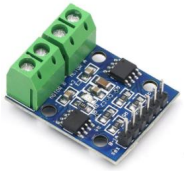
4.Program analysis: Dimming Desk Light

```
from machine import Pin,ADC,PWM
from time import sleep

Led_pin = 15
Potentiometer_pin = 0              # ADC0 multiplexing pin is GP26

def setup():
    global LED
    global Pot_ADC
    LED = PWM(Pin(Led_pin))
    LED.freq(2000)                 #Set the LED operating frequency to 2KHz
    Pot_ADC = ADC(Potentiometer_pin)

def loop():
    while True:
        print ('Potentiometer Value:', Pot_ADC.read_u16())
        Value = Pot_ADC.read_u16()
        LED.duty_u16(Value)
        sleep(0.2)

if __name__ == '__main__':
    setup()
    loop()
```

Lesson 15: Variable Speed Fan

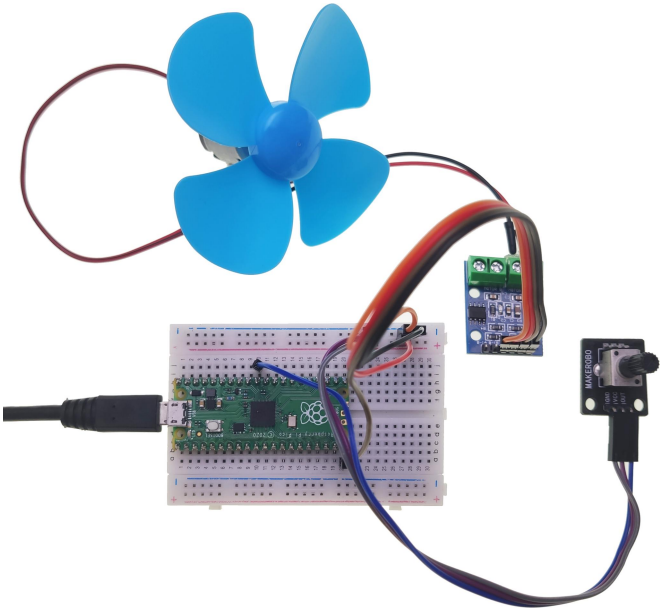
1. Motor Drive Module:

The motor drive module is mainly controlled by the L9110S chip. The L9110S is a two-channel push-pull power amplifier ASIC designed for controlling and driving motors. The two output ends of the drive module can directly drive the forward and reverse motion of the motor. It has a large current drive capability. Each channel can pass a continuous current of 800mA, and the peak current capability can reach 1.5A. L9110S is widely used in toy car motor drive, pulse solenoid valve drive, stepper motor drive and switching power tubes and other circuits.



2.Project Introduction:

Rotate the potentiometer to control the motor driver to output different voltages to achieve the effect of speed regulation fan.



3.Circuit connection:

3.1 Material preparation: Raspberry Pi Pico*1, USB cable*1, Breadboard*1, Potentiometer*1, L9110S motor drive module*1, Motor*1, Fan blade*1, Dupont line.

3.2 Port connection:

Raspberry Pi Pico	Potentiometer
3V3	VCC
GND	GND
GP26	OUT

Raspberry Pi Pico	motor drive module
GND	GND
3V3	VCC
GP15	A-1A
GND	A-1B

Motor drive module	Motor
MOTOR A	GND (-)
A	VCC (+)

4.Program analysis: Variable Speed Fan

```
from machine import Pin,ADC,PWM
from time import sleep

A_1A_pin = 15                    # Motor drive module
Pot_pin = 0                      # ADC0 multiplexing pin is GP26

def setup():
    global A_1A
    global pot_ADC

    A_1A = PWM(Pin(A_1A_pin))
    A_1A.freq(1000)              #Set the driver operating frequency to 1K
    pot_ADC = ADC(Pot_pin)

def loop():
    while True:
        print ('Potentiometer Value:', pot_ADC.read_u16())
        Value = pot_ADC.read_u16()
        A_1A.duty_u16(Value)         # control fan speed
        sleep(0.2)

if __name__ == '__main__':
    setup()
    loop()
```

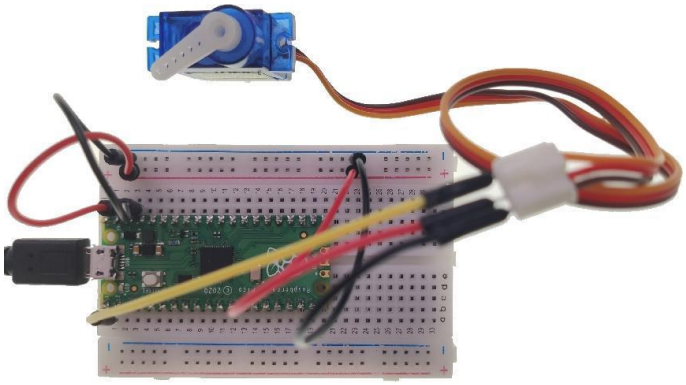
Lesson 16: Servo control

1.Servo:

The 9g small servo is a position (angle) servo driver, suitable for those control systems that require the angle to change continuously and can be maintained. Common in model aircraft, aircraft models, remote control robots and mechanical parts. In use, the accessories of the servo usually include a bracket that can fix the servo to the base and a rudder plate that can be sleeved on the drive shaft. Through the holes on the rudder plate, other objects can be connected to form a transmission model.

2.Project Introduction:

Control the servo to rotate back and forth through the Pico.



3.Circuit connection:

3.1 Material preparation: Raspberry Pi Pico*1, USB cable*1, Breadboard*1, Servo*1, Dupont line;

3.2 port connection:

Raspberry Pi Pico	Servo
GP0	S(yellow)
3V3	VCC (red)
GND	GND(brown)

4.Library file installation:

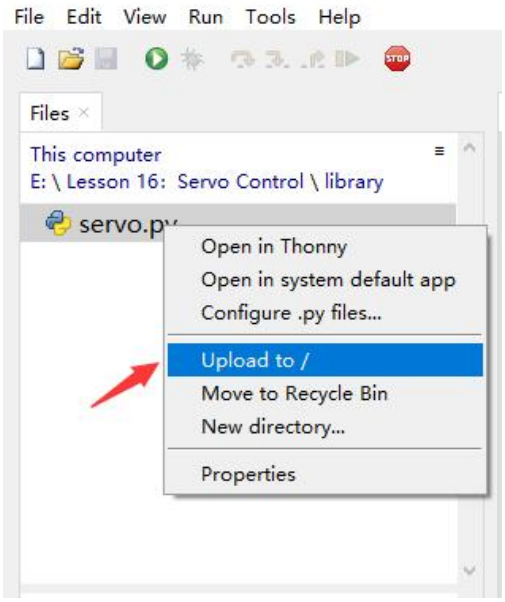
When using the servo, need to upload the **"servo.py"** library file first.The operation steps are as follows:

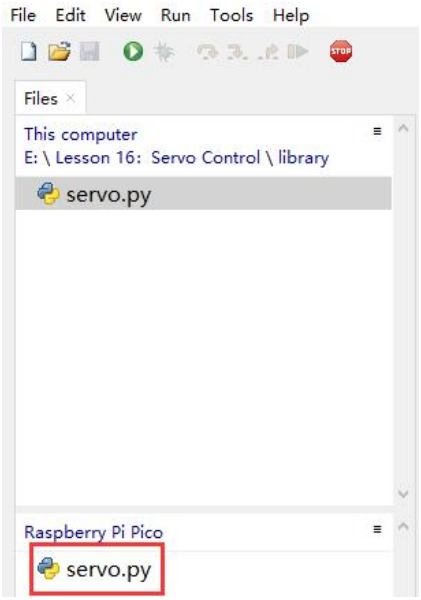
Step1: Connect the Pico to Thonny with a USB cable;

Step2: Find the **"servo.py"** library in the file window;

Step3: Right-click and select **"Upload to / "** to start uploading the library file;

Step4: "servo.py" appears at bottom left, indicating that the upload is successful, and you can start running the program.





5.Program analysis: Servo control

```
import utime
from servo import Servo

s1 = Servo(0)          # Servo pin is connected to GP0

def servo_Map(x, in_min, in_max, out_min, out_max):
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min
def servo_Angle(angle):
    if angle < 0:
        angle = 0
    if angle > 180:
        angle = 180
    s1.goto(round(servo_Map(angle,0,180,0,1024)))   # Convert range value to angle value

if __name__ == '__main__':
    while True:
        print("Turn left ...")
        for i in range(0,180,10):
            servo_Angle(i)
            utime.sleep(0.05)
        print("Turn right ...")
        for i in range(180,0,-10):
            servo_Angle(i)
            utime.sleep(0.05)
```

Lesson 17: Mechanical Arm

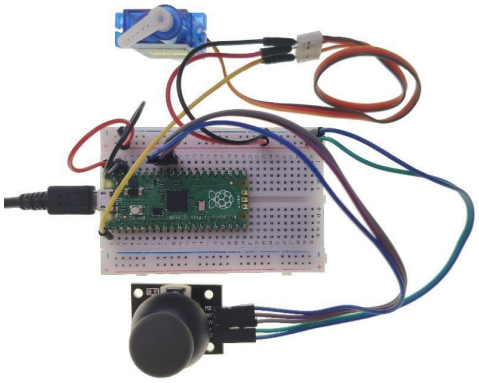
1.PS2 Joystick:

The PS2 joystick module is also called a dual-axis button rocker. It is mainly composed of two potentiometers and a button switch. The two potentiometers output the corresponding voltage values on the X and Y axes respectively with the twist angle of the rocker. Pressing the joystick in the Z-axis direction can trigger the touch button. Under the action of the supporting mechanical structure, in the initial state of the rocker without external force twisting, the two potentiometers are in the middle of the range.

Joysticks are generally widely used in drones, video games, remote control cars, PTZs and other devices in model aircraft. Many devices with screens also often use joysticks as input controls for menu selection.

2.Project Introduction:

By turning the PS2 joystick left (right), the servo is controlled to rotate clockwise (counterclockwise) , simulate the rotation effect of the mechanical arm.



3.Circuit connection:

3.1 Material preparation: Raspberry Pi Pico*1, USB data cable*1, Breadboard*2, PS2 Joystick*1, Servo*1, Dupont line;

3.2 Port connection:

Raspberry Pi Pico	PS2 Joystick
GND	GND
VSYS	+5V
GP26(ADC0)	VRX
GP27(ADC1)	VRY
GP28	SW

Raspberry Pi Pico	Servo
GP0	S(yellow)
3V3	VCC(red)
GND	GND(brown)

4.Library file installation:

Upload the **"servo.py"** library file to the Raspberry Pi Pico. For the specific operation steps, please refer to Lesson 16.

5.Program analysis: Mechanical Arm

```
from machine import Pin,ADC
from time import sleep
from servo import Servo

s1 = Servo(0)
VRX = ADC(0)                    # ADC0 multiplexing pin is GP26
VRY = ADC(1)                    # ADC1 multiplexing pin is GP27
SW = Pin(28, Pin.IN, Pin.PULL_UP)

def Map(x, in_min, in_max, out_min, out_max):
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min

def servo_Angle(angle):
    s1.goto(round(Map(angle,0,180,0,1024))) # Convert range value to angle value

def direction():
    global i
    i = 0
    adc_X = round(Map(VRX.read_u16(),0,65535,0,255))
    adc_Y = round(Map(VRY.read_u16(),0,65535,0,255))
    Switch = SW.value()
    if  adc_X <= 30:
        i = 1               # Define up direction
    elif adc_X >= 255:
        i = 2               # Define down direction
    elif adc_Y >= 255:
        i = 3               # Define left direction
    elif adc_Y <= 30:
        i = 4               # Define right direction
    elif Switch == 0:       # and adc_Y ==128:
        i = 5               # Define Button pressed
    elif adc_X - 125 < 15 and adc_X - 125 > -15 and adc_Y -125 < 15 and adc_Y -125 > -15 and
adc_SW == 255:
        i = 0               # Define home location
```

```
def loop():
    num = 90
    while True:
        direction()        # Call the direction judgment function
        servo_Angle(num)
        sleep(0.01)
        if i == 1:
            num = 0
        if i == 2:
            num = 180
        if i == 3:
            num = num - 1
            if num < 0:
                num = 0
        if i == 4:
            num = num + 1
            if num > 180:
                num = 180
        if i==5:
            num = 90

if __name__ == '__main__':
    loop()              # call the loop function
```

Lesson 18: Access Card

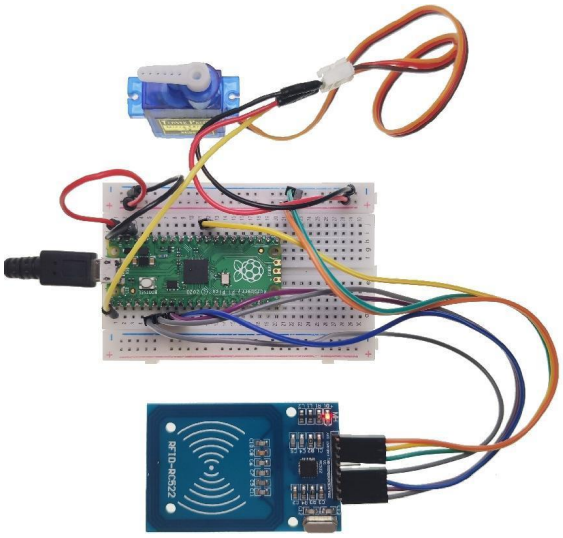
1.MFRC522 RFID module:

RFID (Radio Frequency Identification), also known as radio frequency identification, is a communication technology that can identify specific targets and read and write related data through radio signals without the need to establish mechanical or optical contact between the identification system and specific targets.

MFRC522 is a high-integration card reader chip used in 13.56MHz contactless communication. It is a low-voltage, low-cost, small-volume contactless card reader chip. This module uses MFRC522 original chip to design the card reading circuit, which is easy to use and low in cost. It is suitable for equipment development, card reader development or users who need to design/produce radio frequency card terminals.

2.Project Introduction:

The data (password) is bound to the RFID card through the "Data Write" program, and then the "Data Read" program is used to verify whether the data in the RFID card matches, and then the rotation of the steering gear is controlled to simulate the effect of the access control card opening the door.



3.Circuit connection:

3.1 Material preparation: Raspberry Pi Pico*1, USB cable*1, Breadboard*2, RFID module*1, RFID tag*2, Servo*1, Dupont line;

3.2 Port connection:

Raspberry Pi Pico	MFRC522 RFID module
GND	GND
3V3	3.3V
GP5	SDA
GP6	SCK
GP7	MOSI
GP4	MISO
--	IRQ
GPIO22	RST

Raspberry Pi Pico	Servo
GP0	S(yellow)
3V3	VCC(red)
GND	GND(brown)

4.Library file installation:

Upload the **"servo.py"** and **"mfrc522.py"** library files to the Raspberry Pi Pico. For the specific operation steps, please refer to Lesson 16.

5.Program analysis:

① Date Write

```
import mfrc522
from machine import Pin

sck = 6
mosi = 7
miso = 4
cs = 5        #SDA pin
rst = 22
def do_write():
    rdr = mfrc522.MFRC522(sck=sck, mosi=mosi, miso=miso, rst=rst, cs=cs)
    print("")
    print("Place card before reader to write address 0x08")
    print("")
    try:
        while True:
            (stat, tag_type) = rdr.request(rdr.REQIDL)
            if stat == rdr.OK:
                (stat, raw_uid) = rdr.anticoll()
                if stat == rdr.OK:
                    print("New card detected")
                    print("  - tag type: 0x%02x" % tag_type)
                    print("  - uid  : 0x%02x%02x%02x%02x" % (raw_uid[0], raw_uid[1],
raw_uid[2], raw_uid[3]))
```

```
                print("")
                if rdr.select_tag(raw_uid) == rdr.OK:
                    key = [0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF]
                    if rdr.auth(rdr.AUTHENT1A, 8, key, raw_uid) == rdr.OK:
                        stat = rdr.write(8, b"\ x00 \x01 \x02 \x03 \x04 \x05 \x06 \x07 \x08
\x09 \x0a \x0b \x0c \x0d \x0e \x0f")
                        rdr.stop_crypto1()
                        if stat == rdr.OK:
                            print("Data written to card")
                        else:
                            print("Failed to write data to card")
                    else:
                        print("Authentication error")
                else:
                    print("Failed to select tag")
        except KeyboardInterrupt:
            print("Bye")

if __name__ == '__main__':
    do_write()    # write [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
```

② Date Read

```
import mfrc522
from machine import Pin
from servo import Servo
import utime

s1 = Servo(0)
sck = 6
mosi = 7
miso = 4
cs = 5        #SDA pin
rst = 22

def servo_Angle(angle):
    if angle < 0:
        angle = 0
    if angle > 180:
        angle = 180
    s1.goto(round(angle * 1024 / 180))

def do_read():
    rdr = mfrc522.MFRC522(sck=sck, mosi=mosi, miso=miso, rst=rst, cs=cs)
    print("")
```

```
        print("Place card before reader to read from address 0x08")
        print("")
        try:
            while True:
                num = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]      //Access code
                (stat, tag_type) = rdr.request(rdr.REQIDL)
                if stat == rdr.OK:
                    (stat, raw_uid) = rdr.anticoll()
                    if stat == rdr.OK:
                        print("New card detected")
                        print("  - tag type: 0x%02x" % tag_type)
                        print("  - uid : 0x%02x%02x%02x%02x" % (raw_uid[0], raw_uid[1],
raw_uid[2], raw_uid[3]))
                        print("")
                        if rdr.select_tag(raw_uid) == rdr.OK:
                            key = [0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF]
                            if rdr.auth(rdr.AUTHENT1A, 8, key, raw_uid) == rdr.OK:
                                if rdr.read(8) == num:
                                    for i in range(0,180,10):
                                        servo_Angle(i)
                                        utime.sleep(0.05)
                                    utime.sleep(1)
                                    for i in range(180,0,-10):
                                        servo_Angle(i)
                                        utime.sleep(0.05)
                                else:
                                    servo_Angle(0)
                                rdr.stop_crypto1()
                            else:
                                print("Authentication error")
                        else:
                            print("Failed to select tag")

        except KeyboardInterrupt:
            print("Bye")

    if __name__ == '__main__':
        do_read()       # Read success, return [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
```

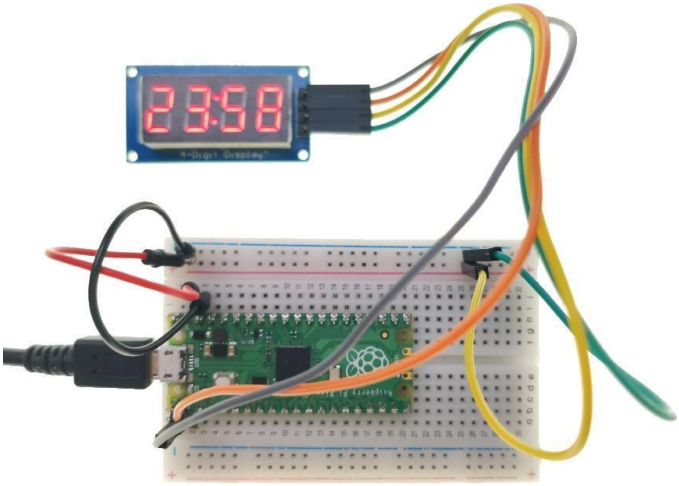
Lesson 19: Electronic Clock

1.TM1637 4-Bits Digital Tube:

TM1637 is a special circuit for LED drive control with keyboard scan interface, which integrates MCU digital interface, data latch, LED high voltage drive, keyboard scan and other circuits.The TM1637 module has four pins. Compared with 10 pins using a four-digit digital tube, using the TM1637 module can greatly save the number of pins.

2.Project Introduction:

The hour and minute data of the clock are displayed through the TM1637 4-Bits digital tube, and the clock point is controlled to flash circularly to make an electronic clock.



3.Circuit connection:

3.1 Material preparation: Raspberry Pi Pico*1, usb cable*1, Breadboard*2, TM1637 4-Bits digital Tube*1, Dupont line;

3.2 Port connection:

Raspberry Pi Pico	TM1637 4-Bits Digital Tube
VSYS	VCC
GND	GND
GP1	DIO
GP0	CLK

4.Library file installation:

Upload the **"tm1637.py"** library file to the Raspberry Pi Pico. For the specific operation steps, please refer to Lesson 16.

5.Program analysis: Electronic Clock

```
import tm1637
from machine import Pin
from utime import sleep
tm = tm1637.TM1637(clk=Pin(0), dio=Pin(1))
Sec = 55        //second
Min = 58        //minute
Hour = 23       //Hour
if __name__ == '__main__':
    while True:
        tm.numbers(Hour,Min,colon=True)
        sleep(0.5)
        tm.numbers(Hour,Min,colon=False)
        sleep(0.5)
        Sec = Sec + 1
        if Sec == 60:
            Min = Min + 1
            Sec = 0
            if Min == 60:
                Hour = Hour + 1
                Min = 0
                if Hour == 24:
                    Hour = 0
```

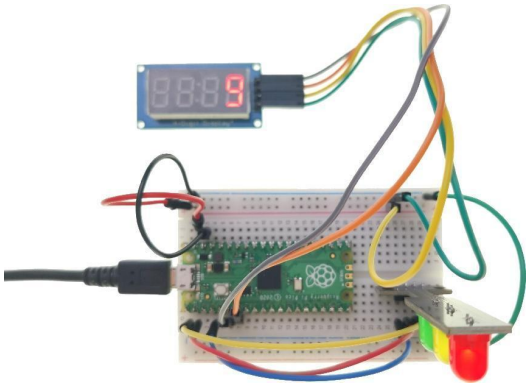
Lesson 20: Traffic Light

1.Traffic Light Module:

The traffic light module is to encapsulate the LED lights of three colors of red, yellow and green into one module, and share the cathode (GND) for control.

2.Project Introduction:

Control the red light to turn on, the red light goes off after the 4-Bits digital tube counts down for 30 seconds; then the yellow light flashes 5 times with a time interval of 0.3 seconds; finally the green light turns on for 10 seconds and then turns off, reciprocating, simulating the effect of traffic lights.



3.Circuit connection:

3.1 Material preparation: Raspberry Pi Pico*1, USB cable*1, Breadboard*1, Traffic Light Module*1, TM1637 4-Bits Digital Tube, Dupont Line;

3.2 Port connection:

Raspberry Pi Pico	Traffic Light Module
GND	GND
GP0	R
GP1	Y
GP2	B

Raspberry Pi Pico	TM1637 4-Bits Digital Tube
3V3	VCC
GND	GND
GP4	DIO
GP5	CLK

4.Library file installation:

Upload the **"tm1637.py"** library file to the R aspberry Pi Pico. For the specific operation steps,

please refer to Lesson 16.

5.Program analysis: Traffic Light

```
from machine import Pin
from time import sleep
import tm1637

tm = tm1637.TM1637(clk=Pin(4), dio=Pin(5))
Led_R = Pin(0, Pin.OUT)
Led_Y = Pin(1, Pin.OUT)
Led_G = Pin(2, Pin.OUT)

if __name__ == '__main__':
    while True:
        num = 30
        Led_R.value(1)
        for i in range(30):
            num=num-1
            tm.number(num)
            sleep(1)
        Led_R.value(0)
        for i in range(5):
            Led_Y.value(1)
            sleep(0.3)
            Led_Y.value(0)
            sleep(0.3)
        Led_G.value(1)
        sleep(10)
        Led_G.value(0)
```

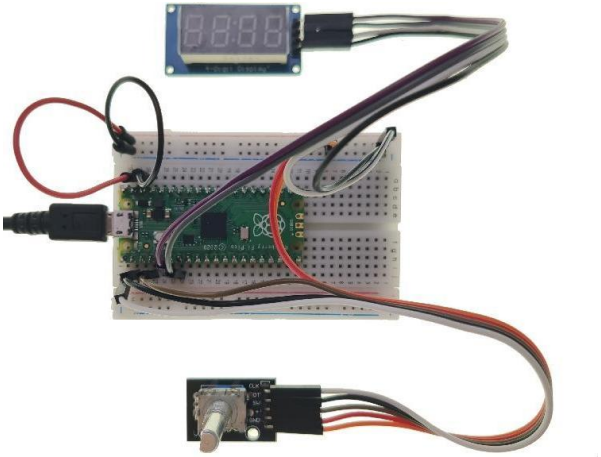
Lesson 21: Electronic Hourglass

1.Encoder:

The encoder can rotate and measure the number of pulses output when it rotates in the forward or reverse direction. Unlike the potentiometer, the rotation count is unlimited. With the buttons on the rotary encoder, it can be reset to the initial state. Which is to count from 0.

2.Project Introduction:

Turn the encoder to set the time, and it will be displayed on the TM1637 4-Bits digital tube in real time. Press the button and the electronic hourglass starts working.



3.Circuit connection:

3.1 Material preparation: Raspberry Pi Pico*1, USB Cable*1, Breadboard*1, Encoder*1, TM1637 4-Bits Digital Tube*1, Dupont line;

3.2 Port connection:

Raspberry Pi Pico	TM1637 4-Bits Digital Tube
3V3	VCC
GND	GND
GP4	DIO
GP5	CLK

Raspberry Pi Pico	Encoder
3V3	+
GND	GND
GP2	SW
GP1	DT
GP0	CLK

4.Library file installation:

Upload the **"tm1637.py"** library file to the Raspberry Pi Pico. For the specific operation steps, please refer to Lesson 16.

5.Program analysis: Electronic Hourglass

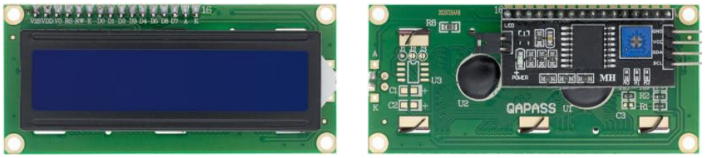
```
from machine import Pin
from time import sleep
import tm1637

tm = tm1637.TM1637(clk=Pin(4), dio=Pin(5))
RoA_Pin = 0                 # CLK
RoB_Pin = 1                 # DT
Btn_Pin = 2                 # SW

globalCounter = 0           # counter value
flag = 0                    # Whether the rotation flag occurs
Last_RoB_Status = 0         # DT state
Current_RoB_Status = 0      # CLK state

def setup():
    global clk_RoA
    global dt_RoB
    global sw_BtN

    clk_RoA =   Pin(RoA_Pin,Pin.IN)
    dt_RoB = Pin(RoB_Pin,Pin.IN)
    sw_BtN = Pin(Btn_Pin,Pin.IN, Pin.PULL_UP)
    # Initialize the interrupt function, when the SW pin is 0, the interrupt is enabled
    sw_BtN.irq(trigger=Pin.IRQ_FALLING,handler=btnISR)

# Rotation code direction bit judgment function
def rotaryDeal():
    global flag
    global Last_RoB_Status
    global Current_RoB_Status
    global globalCounter

    Last_RoB_Status = dt_RoB.value()
    # Judging the level change of the CLK pin to distinguish the direction
    while(not clk_RoA.value()):
        Current_RoB_Status = dt_RoB.value()
        flag = 1                    # Rotation mark occurs
    if flag == 1:                   # The flag bit is 1 and a rotation has occurred
        flag = 0                    # Reset flag bit
```

```
            if (Last_RoB_Status == 0) and (Current_RoB_Status == 1):
                globalCounter = globalCounter + 1       # counterclockwise, positive
            if (Last_RoB_Status == 1) and (Current_RoB_Status == 0):
                globalCounter = globalCounter - 1       # Clockwise, negative

# Interrupt function, when the SW pin is 0, the interrupt is enabled
def btnISR(chn):
    global globalCounter
    globalCounter = 0
    print ('globalCounter = %d' %globalCounter)
    while True:
    # Define a counter that changes every 1 second
        tm.number(globalCounter)
        globalCounter = globalCounter - 1
        sleep(1)
        if globalCounter == 0:
            break

def loop():
    global globalCounter
    tmp = 0
    while True:
        rotaryDeal()
        if etmp != globalCounter:
            print ('globalCounter = %d' % globalCounter)
            tmp = globalCounter
            tm.number(globalCounter)

if __name__ == '__main__':
    setup()
    loop()
```

Lesson 22: Billboard

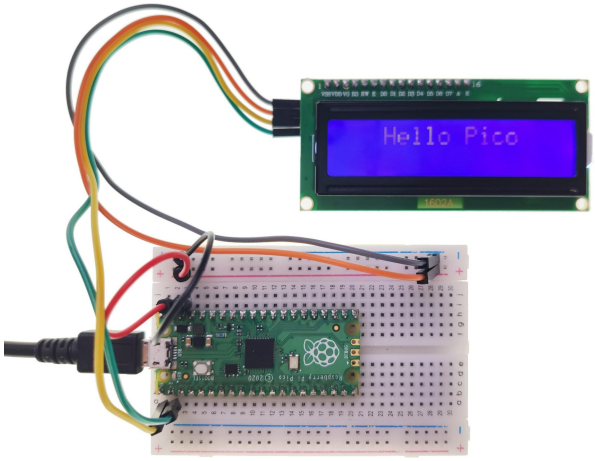
1.LCD1602:

LCD1602, or 1602 character-type liquid crystal display, is a kind of dot matrix module to show letters, numbers, and characters and so on. It's composed of 5x7 dot matrix positions; each position can display one character. The model 1602 means it displays 2 lines of 16 characters. The principle of LCD1602 liquid crystal display is to use the physical properties of liquid crystal to control its display area through voltage, and then display characters. (Note: The potentiometer on the back can adjust the clarity of the display)



2.Project Introduction:

The first line of LCD1602 scrolls from right to left to display the string of "Hello Pico", simulating the effect of billboard scrolling.



3.Circuit connection:

3.1 Material preparation: Raspberry Pi Pico*1, USB able*1, Breadboard*1, LCD1602*1, Dupont line;

3.2 Port connection:

Raspberry Pi Pico	LCD1602
GP0	SDA
GP1	SCL
VSYS	VCC (+)
GND	GND (-)

4.Library file installation:

Upload the **“lcd_api.py”** and **“i2c_lcd.py”** library file to the Raspberry Pi Pico. For the specific operation steps, please refer to Lesson 16.

5.Program analysis: Billboard

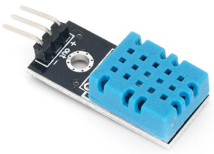
```
from machine import I2C, Pin
from i2c_lcd import I2cLcd
from utime import sleep

DEFAULT_I2C_ADDR = 0x3F                          # LCD 1602 I2C address
i2c = I2C(0,sda=Pin(0),scl=Pin(1),freq=400000)
lcd = I2cLcd(i2c, DEFAULT_I2C_ADDR, 2, 26)       # Initialize(device address, cursor settings)
text = '                    Hello Pico'          # Show scrolling information

if __name__ == '__main__':
    while True:
        tmp = text                               # Get the display information
        for i in range(0, len(text)):
            lcd.move_to(len(text),1)             # Position cursor
            lcd.putstr(tmp)                      # Display one by one
            tmp = tmp[1:]
            sleep(0.8)
            lcd.clear()                          # Clear display
```

Lesson 23: Mini Weather Station

1.DHT11 Temperature & Humidity Sensor:

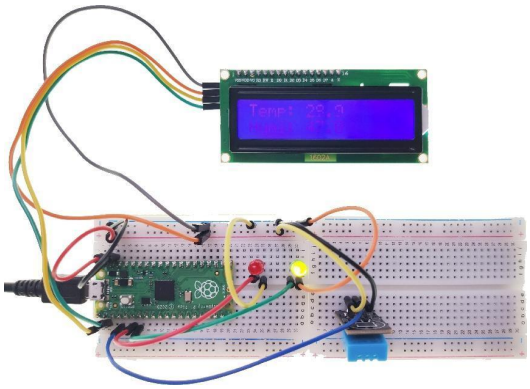
DHT11 Temperature & Humidity sensor is a temperature and humidity composite sensor with calibrated digital signal output, it includes a resistive humidity sensing element and an NTC temperature measuring element, and is connected with a high-performance 8-bit microcontroller.

Its application-specific digital module acquisition technology and temperature and humidity sensing technology ensure that the product has extremely high reliability and excellent long-term stability.



2.Project Introduction:

The ambient temperature value and humidity value are monitored in real time through the DHT11 Temperature & Humidity sensor, and displayed on the LCD1602 module synchronously. When the temperature is too high or the humidity is too low, it will trigger the LED light to turn on and off to remind.



3.Circuit connection:

3.1 Material preparation: Raspberry Pi Pico*1, USB cable*1, Breadboard*1, DHT11 Temperature & Humidity sensor*1, LCD1602*1, LED(R)*1, LED(G)*1, Dupont line;

3.2 Port connection:

Raspberry Pi Pico	DHT11 Temperature & Humidity sensor
VSYS	+
GND	-
GP2	out

Raspberry Pi Pico	LCD1602
GP0	SDA
GP1	SCL
VSYS	VCC (+)
GND	GND (-)

Raspberry Pi Pico	LED(R)
GND	- (short)
GP4	+ (long)

Raspberry Pi Pico	LED(G)
GND	- (short)
GP5	+ (long)

4.Library file installation:

Upload the **"lcd_api.py"**, **"i2c_lcd.py"** and **"dht.py"** library file to the Raspberry Pi Pico. For the specific operation steps, please refer to Lesson 16.

5.Program analysis: Mini Weather Station

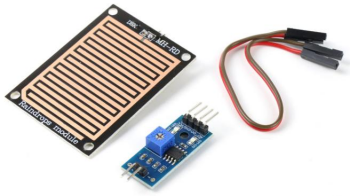
```
from machine import I2C, Pin
from i2c_lcd import I2cLcd
from utime import sleep
from dht import DHT11, InvalidChecksum

DEFAULT_I2C_ADDR = 0x3F                    # LCD 1602 I2C address
led_red = Pin(4,Pin.OUT)
led_green = Pin(5,Pin.OUT)
pin = Pin(2, Pin.OUT, Pin.PULL_DOWN)
dht11 = DHT11(pin)

def setup():
    global lcd
    i2c = I2C(0,sda=Pin(0),scl=Pin(1),freq=400000)
    lcd = I2cLcd(i2c, DEFAULT_I2C_ADDR, 2, 16)   # Initialize(device address, cursor settings)

def loop():
    try:
        while True:
            lcd.move_to(0,0)
            lcd.putstr("Temp: {}".format(dht11.temperature))
            lcd.move_to(0,1)
            lcd.putstr("Humi: {}".format(dht11.humidity))
            if dht11.temperature > 35 or dht11.humidity < 10:
                led_red.value(1)
                led_green.value(0)
                sleep(0.5)
                led_red.value(0)
                sleep(0.5)
            else:
```

```
                led_red.value(0)
                led_green.value(1)
            sleep(1)
            lcd.clear()
        except InvalidChecksum:
            print("Checksum from the sensor was invalid")
if __name__ == '__main__':
    setup()
    loop()
```

Lesson 24: Flood Warning

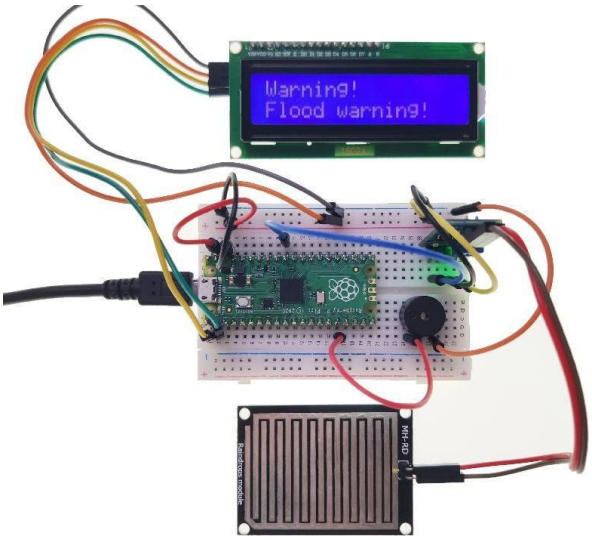
1.Raindrop sensor:

Raindrop sensor is a kind of sensing device, which is mainly used to detect whether it rains and the amount of rain, and is widely used in automobile automatic wiper system, intelligent lighting system and intelligent sunroof system. It consists of two parts, one is a sensor panel for detecting rainwater, and the other is a control module that converts the detected signal into an electrical signal.



2.Project Introduction:

It simulates the flood control alarm system, judges whether there is a flood disaster through the amount of rain detected by the sensor panel of the raindrop sensor, and then controls the buzzer to emit a siren sound and displays the alarm information on the LCD1602 LCD screen.



3.Circuit connection:

3.1 Material preparation: Raspberry Pi Pico*1, USB cable*1, Breadboard*1, Raindrop sensor*1, LCD1602*1, Buzzer*1, Dupont line;

3.2 Port connection:

Raspberry Pi Pico	Raindrop sensor
GND	GND
VSYS	VCC
--	DO
GP26(ADC0)	AO

Raspberry Pi Pico	LCD1602
GP0	SDA
GP1	SCL
VSYS	VCC (+)
GND	GND (-)

Raspberry Pi Pico	Buzzer
GND	GND
GP12	VCC (+)

4.Library file installation:

Upload the **"i2c_lcd.py"** and **"i2c_api.py"** library file to the Raspberry Pi Pico. For the specific operation steps, please refer to Lesson 16.

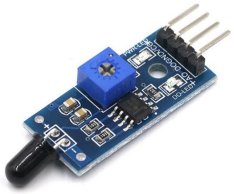
5.Program analysis: Flood Warning

```
from machine import Pin,ADC,PWM,I2C
from i2c_lcd import I2cLcd
from time import sleep
DEFAULT_I2C_ADDR = 0x3F        # LCD 1602 I2C address
Raindrop_AO = ADC(0)           # ADC0 multiplexing pin is GP26
Buzzer = 12
buzzer = PWM(Pin(Buzzer))
def setup():
    global lcd
    i2c = I2C(0,sda=Pin(0),scl=Pin(1),freq=400000)
    lcd = I2cLcd(i2c, DEFAULT_I2C_ADDR, 2, 16)
def loop():
    while True:
        text = 'Warning!\nFlood warning!'          # show alert information
        adc_Raindrop =Raindrop_AO.read_u16()
        if   adc_Raindrop < 30000:
            lcd.putstr(text)
            buzzer.duty_u16(1000)
            buzzer.freq(294)
            sleep(0.5)
            lcd.clear()
            buzzer.freq(495)
            sleep(0.5)
        else:
            buzzer.duty_u16(0)

if __name__ == '__main__':
    setup()
    loop()
```

Lesson 25: Alarm of Fire

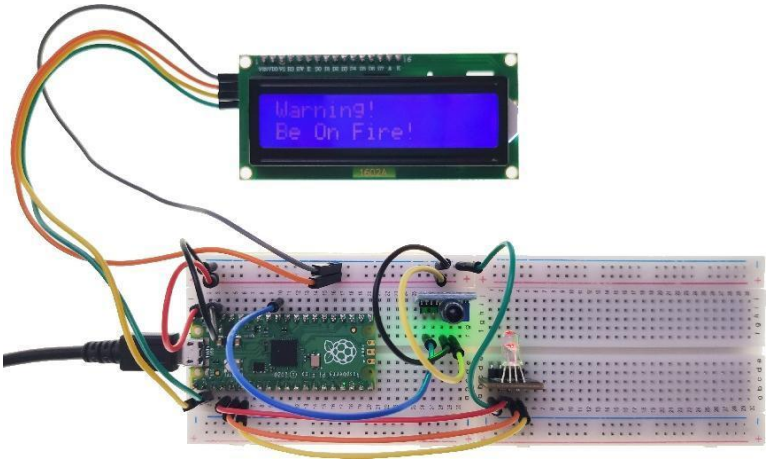
1.Flame sensor:

Flame sensor is generally divided into far-infrared flame sensor and ultraviolet flame sensor, and its working principle is to detect the heat radiation of fire source of specific wavelength. The far-infrared flame sensor can detect infrared light with a wavelength in the range of 700 nanometers to 1000 nanometers, and the detection angle is 60. When the infrared light wavelength is around 880 nanometers, its sensitivity reaches the maximum. The far-infrared flame probe converts the change of the intensity of the external infrared light into the change of the current, which is reflected as the change of the value in the range of 0~65535 through the A/D converter.



2.Project Introduction:

Simulate a fire alarm system, judge whether a fire occurs by judging whether the induction probe of the flame sensor detects the fire source, control the RGB lights to flash red and blue, and display the alarm information on the LCD1602 liquid crystal screen.



3.Circuit connection:

3.1 Material preparation: Raspberry Pi Pico*1, USB cable*1, Breadboard*2, Flame sensor*1, LCD1602*1, RGB LED*1, Dupont line;

3.2 Port connection:

Raspberry Pi Pico	Flame sensor
GND	GND
VSYS	VCC
--	DO
GP26(ADC0)	AO

Raspberry Pi Pico	LCD1602
GP0	SDA
GP1	SCL
VSYS	VCC (+)
GND	GND (-)

Raspberry Pi Pico	RGB LED
GND	GND
GP4	R
GP3	G
GP2	B

4.Library file installation:

Upload the **"i2c_lcd.py"** and **"i2c_api.py"** library file to the Raspberry Pi Pico. For the specific operation steps, please refer to Lesson 16.

5.Program analysis: Alarm of Fire

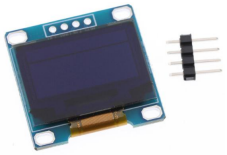
```
from machine import Pin,ADC,PWM,I2C
from i2c_lcd import I2cLcd
from time import sleep

DEFAULT_I2C_ADDR = 0x3F      # LCD 1602 I2C address
Flame_AO = ADC(0)            # ADC0 multiplexing pin is GP26

Led_R = PWM(Pin(4))
Led_G = PWM(Pin(3))
Led_B = PWM(Pin(2))
Led_R.freq(2000)             # Set the frequency to 2KHz
Led_G.freq(2000)
Led_B.freq(2000)

i2c = I2C(0,sda=Pin(0),scl=Pin(1),freq=400000)
lcd = I2cLcd(i2c, DEFAULT_I2C_ADDR, 2, 16)

If __name__ == '__main__':
    while True:
        text = 'Warning!\nBe On Fire!'          # show alert information
        Flame_value = Flame_AO.read_u16()      # Get the analog value of the flame sensor
        if Flame_value < 30000:
            lcd.putstr(text)
            Led_R.duty_u16(65535)
            Led_G.duty_u16(0)
            Led_B.duty_u16(0)
            sleep(0.5)
```

```
            lcd.clear()
            Led_R.duty_u16(0)
            Led_G.duty_u16(0)
            Led_B.duty_u16(65535)
            sleep(0.5)
        else:
            Led_R.duty_u16(0)
            Led_G.duty_u16(65535)
            Led_B.duty_u16(0)
```

Lesson 26: Electronic Wall Calendar

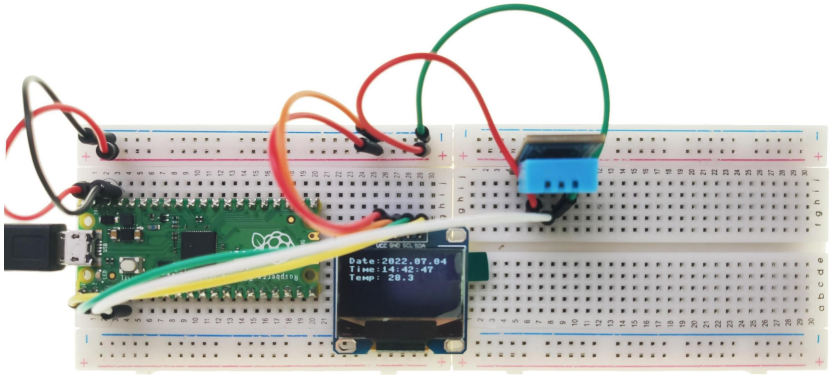
1.SSD1306 OLED:

OLED (Organic Light Emitting Diode) is a self-luminous display screen without backlight and liquid crystal, with excellent color saturation, contrast and response speed. Because the material is lighter and thinner, transparent and flexible, OLED can realize a variety of designs and is widely used in mobile phones, digital cameras, notebook computers, car audio and TVs.



2.Project Introduction:

Real-time display of current date, time and ambient temperature on the OLED.



3.Circuit connection:

3.1 Material preparation: Raspberry Pi Pico*1, USB cable*1, Breadboard*1, SSD1306 OLED*1, DHT11 Temperature & Humidity sensor*1, Dupont line;

3.2 port connection:

Raspberry Pi Pico	SSD1306 OLED
GP0	SDA
GP1	SCL
VSYS	VCC
GND	GND

Raspberry Pi Pico	DHT11 Temperature & Humidity sensor
VSYS	+
GND	-
GP2	out

4.Library file installation:

Upload the **“ssd1306.py”** and **“dht.py”** library file to the Raspberry Pi Pico. For the specific operation steps, please refer to Lesson 16.

5.Program analysis: Electronic Wall Calendar

```
from machine import Pin, I2C
from ssd1306 import SSD1306_I2C
from dht import DHT11, InvalidChecksum
import time

list = [2022, 06, 29, 10, 01, 25]
mon_max = [1,3,5,7,8,10,12]
mon_min = [4,6,9,11]

i2c=I2C(0,sda=Pin(0), scl=Pin(1), freq=400000)
oled = SSD1306_I2C(128, 64, i2c)

DHT_pin = Pin(2, Pin.OUT, Pin.PULL_DOWN)
dht11= DHT11(DHT_pin)

def set_time():
    global text1
    if list[5] > 9:
        if list[4] > 9:
            text1 = 'Time:%d:%d:%d'%(list[3],list[4],list[5])
        else:
            text1 = 'Time:%d:0%d:%d'%(list[3],list[4],list[5])
    else:
        if list[4] > 9:
            text1 = 'Time:%d:%d:0%d'%(list[3],list[4],list[5])
        else:
            text1 = 'Time:%d:0%d:0%d'%(list[3],list[4],list[5])

def set_date():
    global text2
    if list[2] > 9:
        if list[1] > 9:
            text2 = 'Date:%d.%d.%d'%(list[0],list[1],list[2])
        else:
            text2 = 'Date:%d.0%d.%d'%(list[0],list[1],list[2])
    else:
        if list[1] > 9:
            text2 = 'Date:%d.%d.0%d'%(list[0],list[1],list[2])
        else:
            text2 = 'Date:%d.0%d.0%d'%(list[0],list[1],list[2])

def date_change():
    list[2] = 1
```

```
        list[1] += 1
        if list[1] > 12:
            list[1] = 1
            list[0] += 1

    def time_change():
        list[5] += 1
        if list[5] > 59:
            list[5] = 0
            list[4] += 1
            if list[4] > 59:
                list[4] = 0
                list[3] += 1
                if list[3] > 23:
                    list[3] = 0
                    list[2] += 1
                    if list[1] in mon_max:
                        if list[2] > 31:
                            date_change()
                    elif list[1] in mon_min:
                        if list[2] > 30:
                            date_change()
                    elif list[1] == 2:
                        if (list[0] % 4 == 0 and list[0] % 100 != 0) or list[0] % 400 == 0:
                            if list[2] > 29:
                                date_change()
                            elif list[2] > 28:
                                date_change()

    if __name__ == '__main__':
        while True:
            set_date()
            oled.text(text2, 0, 0)
            set_time()
            oled.text(text1, 0, 10)
            oled.text("Temp: {}".format(dht11.temperature),0,20)
            oled.show()
            time_change()
            time.sleep(1)
            oled.fill(0)
```

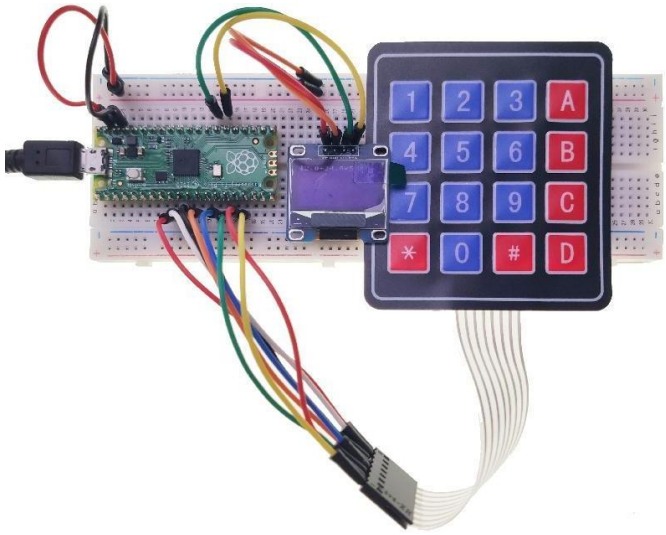
Lesson 27: Simple Calculator

1.4*4 Matrix Membrane Keyboard:

The 4*4 matrix membrane keyboard consists of four parts: panel, upper circuit, isolation layer and lower circuit. When the panel is not pressed down, the upper and lower contacts are disconnected, and the isolation layer plays an isolation role on the upper and lower lines; When the panel is pressed, the contact of the upper circuit deforms downward, and remerges with the lower circuit to make the circuit turn on, and the level signal changes; When the finger is released, the upper circuit contact bounces back, the circuit is disconnected, and the level signal is in its initial state.

2.Project Introduction:

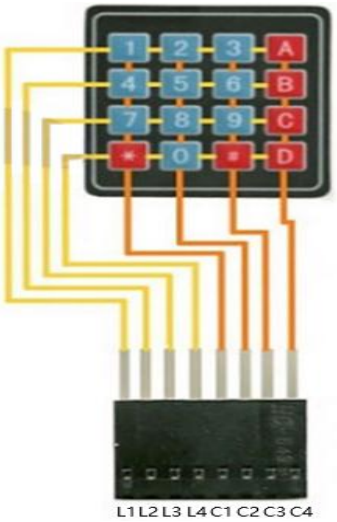
Detect the key feedback information of the 4*4 matrix membrane keyboard through the Raspberry Pi Pico, set the A, B, C, D keys as addition, subtraction, multiplication and division functions respectively, perform arithmetic operations, and print the operation process and results on the OLED display to realize the function of a simple calculator.



3.Circuit connection:

3.1 Material preparation: Raspberry Pi Pico*1, USB cable*1, Breadboard*1, 4*4 matrix membrane keyboard*1, SSD1306 OLED*1, Dupont line;

3.2 Port connection:



Raspberry Pi Pico	4*4 matrix membrane keyboard
GP13	L1
GP12	L2
GP11	L3
GP10	L4
GP9	C1
GP8	C2
GP7	C3
GP6	C4

Raspberry Pi Pico	SSD1306 OLED
GP20	SDA
GP21	SCL
VSYS	VCC
GND	GND

4.Library file installation:

Upload the **"ssd1306.py"** library file to the Raspberry Pi Pico. For the specific operation steps, please refer to Lesson 16.

5.Program analysis: Simple Calculator

```
from machine import Pin,Timer,I2C
import utime
from ssd1306 import SSD1306_I2C
import framebuf

debug=True

i2c = I2C(0, scl=Pin(21), sda=Pin(20), freq=40000)
oled = SSD1306_I2C(128, 64, i2c)
```

```
    keyName = [['1','2','3','+'],
               ['4','5','6','-'],
               ['7','8','9','*'],
               ['c','0','=','/']]
    keypadRowPins = [13,12,11,10]
    keypadColPins = [9,8,7,6]

    row = []
    col = []
    keypadState = [];
    for i in keypadRowPins:
        row.append(Pin(i,Pin.IN,Pin.PULL_UP))
        keypadState.append([0,0,0,0])
    for i in keypadColPins:
        col.append(Pin(i,Pin.OUT))

    def solve(oprt, oprdA, oprdB):
        if(oprt == "+"):
            return oprdA + oprdB
        elif(oprt == "-"):
            return oprdA - oprdB
        elif(oprt == "*"):
            return oprdA * oprdB
        elif(oprt == "/"):
            return round(oprdA / oprdB , 6)

    def calc(lst):
        operand = []
        operator = []
        for i in lst:
            if(debug):
                print(i)
            if(i=='+'):
                while (len(operator)!=0 and (operator[-1] == '*' or operator[-1] == '/' or operator[-1] ==
'-' or operator[-1] == '+')):
                    b = operand.pop(-1)
                    a = operand.pop(-1)
                    c = operator.pop(-1)
                    operand.append(solve(c,a,b))
                operator.append(i)
            elif(i=='-'):
                while (len(operator)!=0 and (operator[-1] == '*' or operator[-1] == '/' or operator[-1] ==
'-' or operator[-1] == '+')):
                    b = operand.pop(-1)
```

```
                    a = operand.pop(-1)
                    c = operator.pop(-1)
                    operand.append(solve(c,a,b))
                operator.append(i)
            elif(i=='*'):
                while (len(operator)!=0 and (operator[-1] == '*' or operator[-1] == '/')):
                    b = operand.pop(-1)
                    a = operand.pop(-1)
                    c = operator.pop(-1)
                    operand.append(solve(c,a,b))
                operator.append(i)
            elif(i=='/'):
                while (len(operator)!=0 and (operator[-1] == '*' or operator[-1] == '/')):
                    b = operand.pop(-1)
                    a = operand.pop(-1)
                    c = operator.pop(-1)
                    operand.append(solve(c,a,b))
                operator.append(i)
            elif(i=='('):
                operator.append(i)
            elif(i==')'):
                while(operator[-1] !='('):
                    b = operand.pop(-1)
                    a = operand.pop(-1)
                    c = operator.pop(-1)
                    operand.append(solve(c,a,b))
                operator.pop(-1)
            else:
                operand.append(i)
        while(len(operator) != 0):
            b = operand.pop(-1)
            a = operand.pop(-1)
            c = operator.pop(-1)
            operand.append(solve(c,a,b))
        return operand[0]

    def keypadRead():
        global row
        j_ifPressed = -1
        i_ifPressed = -1
        for i in range(0,len(col)):
            col[i].low()
            utime.sleep(0.005)                                  #settling time
            for j in range(0,len(row)):
```

```
                pressed = not row[j].value()
                if(pressed and (keypadState[j][i] != pressed)):          #state changed to high
                    keypadState[j][i] = pressed
                elif(not pressed and (keypadState[j][i] != pressed)):   # state changed to low
                    keypadState[j][i] = pressed
                    j_ifPressed = j
                    i_ifPressed = i
            col[i].high()
        if(j_ifPressed != -1 and i_ifPressed != -1):
            return keyName[j_ifPressed][i_ifPressed]
        else:
            return -1

def printOled(lst):
    oledPos = {
                "x" : 0,
                "y" : 0
              }
    oled.fill(0)
    string = ''
    for i in lst:
        string += str(i)
    l = 0
    while(l<len(string)):
        oled.text(string[l:l+16],oledPos["x"], oledPos["y"])
        oledPos["y"] =oledPos["y"] + 10
        l = l+16
    oled.show()

shiftFlag = False
signFlag = False
inputList = ['']

oled.show()
oled.fill(0)
oled.show()
oled.text("Elecrow",30,9,1)
oled.text("Simple",35,28,1)
oled.text("calculator",18,38,1)
oled.show()

if __name__ == '__main__':
    while True:
        key = keypadRead()
```

```
            if(key != -1):
                if((key <= '9' and key >='0') or key == '.'):
                    inputList[-1] = inputList[-1] + key
                elif(key == '+' or key == '-' or key == '*' or key == '/'):
                    if(inputList != ['']):
                        if(inputList[-1] == '' and (inputList[-2] == '+' or inputList[-2] == '-' or
inputList[-2] == '*' or inputList[-2] == '/')):
                            inputList[-2] = key
                        elif(inputList[-1]==''):
                            inputList[-1]=key
                            inputList.append('')
                        else:
                            inputList[-1] = float(inputList[-1])
                            inputList.append(key)
                            inputList.append('')
                elif(key == 's'):
                    shiftFlag = not shiftFlag
                elif(key == 'a'):
                    if(shiftFlag):
                        if(inputList[-1] != ''):
                            inputList[-1] = float(inputList[-1])
                            inputList.append(')')
                            inputList.append('')
                        else:
                            inputList[-1] = ')'
                            inputList.append('')
                        shiftFlag = False
                    else:
                        signFlag = not signFlag
                        if(inputList[-1] == ''):
                            inputList[-1] = '-'
                        else:
                            if(inputList[-1][0] == '-'):
                                inputList[-1]   = inputList[-1][1:]
                            else:
                                inputList[-1] = '-' + inputList[-1]
                elif(key == 'b'):
                    if(shiftFlag):
                        if(inputList[-1] == ''):
                            inputList[-1] = '('
                        else:
                            inputList.append('(')
                        inputList.append('')
                        shiftFlag = False
```

```
            else:
                if(inputList[-1] == ''):
                    inputList[-1] = 3.14
                else:
                    inputList.append(3.14)
                inputList.append('')
        elif(key == 'c'):
            if(shiftFlag):
                inputList = ['']
                shiftFlag = False
            else:
                if(inputList == ["error"]):
                    inputList = ['']
                if(inputList != ['']):
                    if(inputList[-1] == ''):
                        inputList.pop()
                        inputList[-1] = str(inputList[-1])[:-1]
                    else:
                        inputList[-1] = str(inputList[-1])[:-1]
        elif(key == '='):
            if(inputList[-1] == ''):
                inputList.pop(-1)
            elif(inputList[-1] != ')'):
                inputList[-1] = float(inputList[-1])
            try:
                ans = calc(inputList)
                inputList = [str(ans)]
            except:
                ans = ''
                inputList = []
                inputList.append("error")

        printOled(inputList)
        print(inputList)
```

Lesson28: Dc Reduction Motor

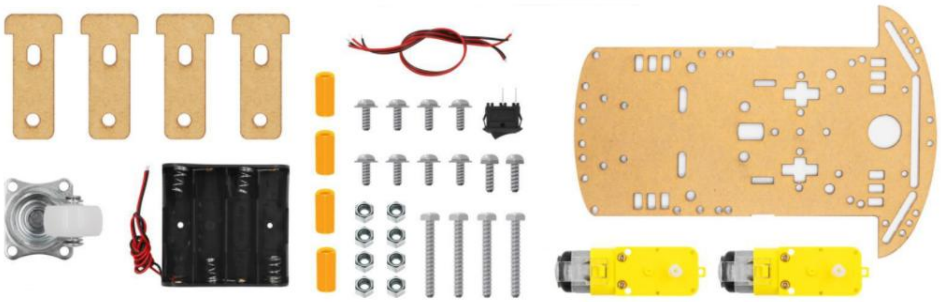
1.Dc reduction motor:

DC reduction motor, that is, gear reduction motor, which adds a gear reduction structure on the basis of ordinary DC motor to provide lower speed and larger torque. In addition, different reduction ratios can provide different speeds and torques.

2.Project Introduction:

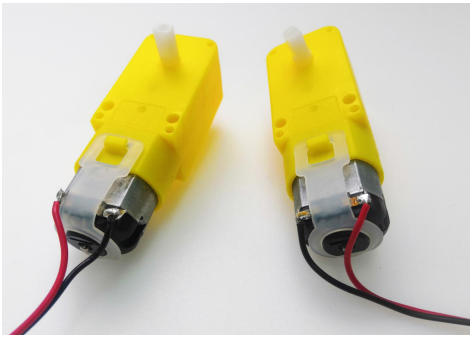
Assemble the trolley according to the following steps, and control the trolley to move forward, backward, left, and right through the program.

3.Assembly steps:

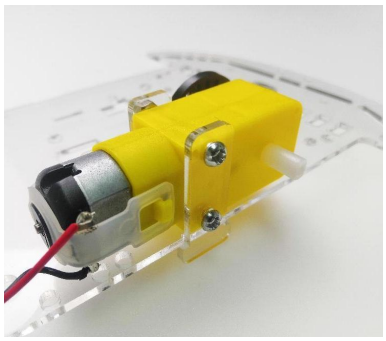
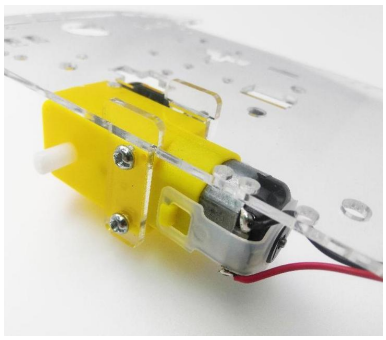


3.1 Tear off the protective film on the surface of all acrylic devices;

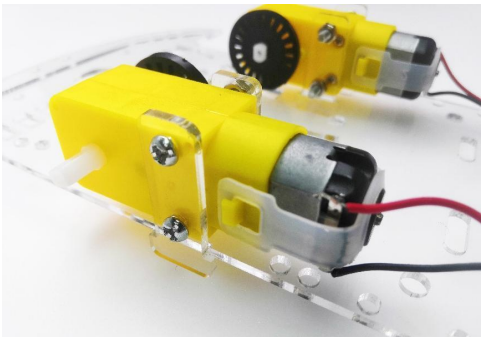
3.2 Connect the red and black wires to the two ports of the motor according to the connection method shown in the figure below; (It can be fixed by welding, or directly through the holes on the port to screw and fix)



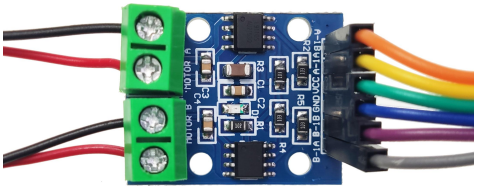
3.3 As shown in the figure below, fix the motor on the acrylic panel car body with acrylic fasteners. (The installation position of the red and black wires determines the direction of rotation of the motor.)

3.4 In the same way, fix another motor in the symmetrical position of the acrylic board;

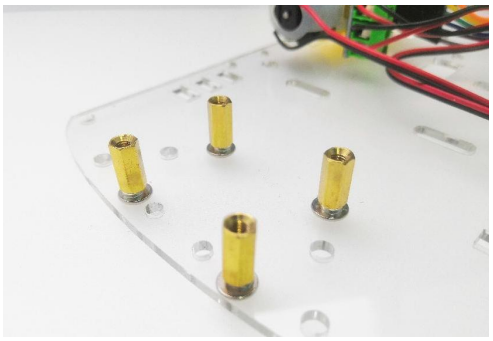


3.5 As shown in the figure below, connect the red and black wires of the two motors to the L9110s motor drive module ports respectively, and insert 8 DuPont lines to the 8 pin headers on the other side;

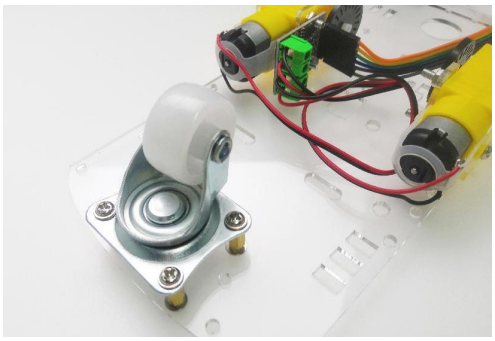


3.6 Fix the L9110s motor drive module;

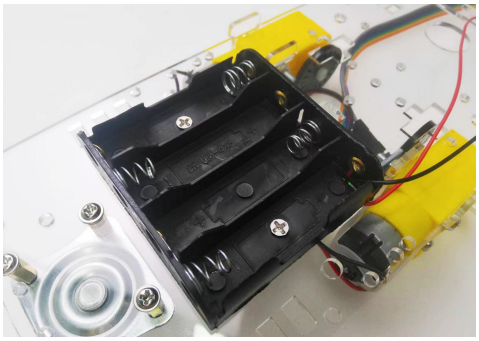
3.7 Fix the double-pass copper column on the acrylic board;



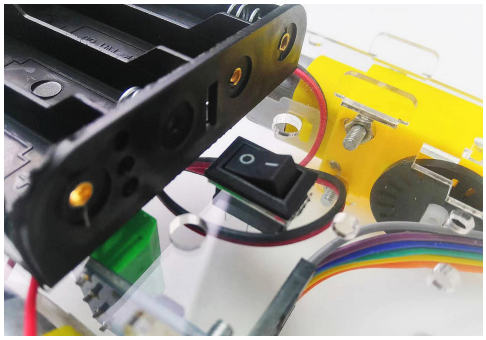
3.8 Fix the universal wheel;



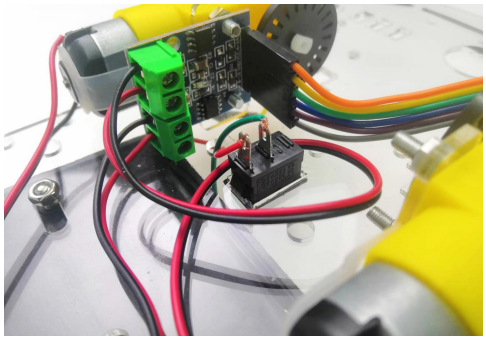
3.9 Fix the battery box;



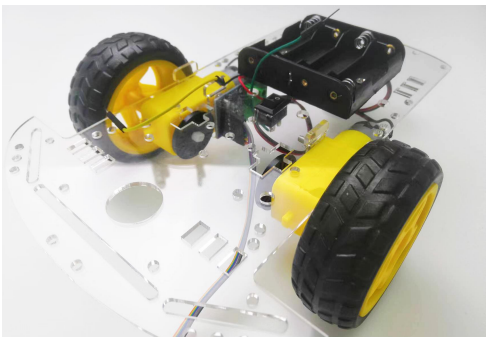
3.10 Fix the botton;



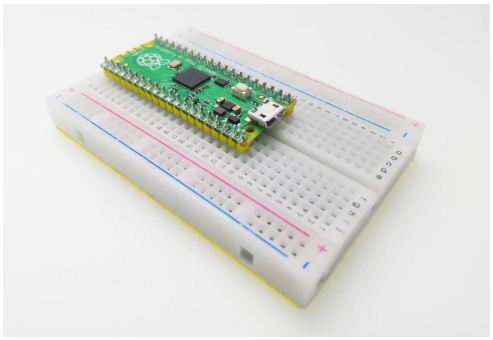
3.11 Connect the botton to the battery box（One end is connected to the positive electrode of the battery (red wire), and the other end is connected to the VBUS of the Pico.)



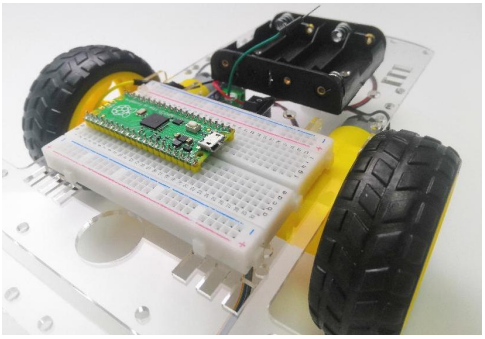
3.12 Install the wheels;



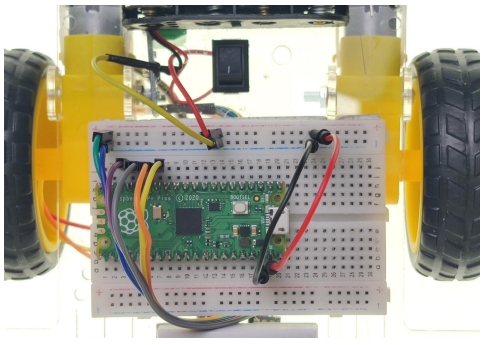
3.13 Connect the Raspberry Pi Pico and the breadboard;



3.14 Peel off the bottom film and stick the breadboard;



3.15 Insert the dupont line to the breadboard. (For details, please refer to: 4. Circuit Connection)



4.Circuit connection:

4.1 Material preparation: Car chassis Kit*1, Raspberry Pi Pico*1, USB cable*1, Breadboard*1, L9110S motor drive module*1, Dc reduction motor*2, AA battery*4 (purchased by yourself), battery box*1, Dupont line;

4.2 Port connection:

Raspberry Pi Pico	L9110S motor drive module
GP12	B-1A
GP13	B-1B
GND	GND
VSYS	VCC
GP10	A-1A
GP11	A-1B

5.Program analysis: Dc Reduction Motor

```
from machine import Pin
from utime import sleep
MotorPinA_1A = 10
MotorPinA_1B = 11
MotorPinB_1A = 12
MotorPinB_1B = 13
def setup():
    global motorA1
    global motorA2
    global motorB1
    global motorB2
    motorA1 = Pin(MotorPinA_1A,Pin.OUT)
    motorA2 = Pin(MotorPinA_1B,Pin.OUT)
    motorB1 = Pin(MotorPinB_1A,Pin.OUT)
    motorB2 = Pin(MotorPinB_1B,Pin.OUT)
def motor(A1,A2,B1,B2):
    motorA1.value(A1)
    motorA2.value(A2)
    motorB1.value(B1)
    motorB2.value(B2)
def loop():
    while True:
        motor(1,0,1,0)
        sleep(2)
        motor(0,1,0,1)
        sleep(2)
        motor(0,1,1,0)
        sleep(2)
        motor(1,0,0,1)
        sleep(2)
        motor(0,0,0,0)
        sleep(1)
if __name__ == '__main__':
    setup()
    loop()
```

Lesson 29: Bumper Cars

1.Collision sensor:

The collision sensor, also known as the collision switch, is equivalent to a key switch, relying on the internal mechanical structure to complete the conduction and disconnection of the circuit. When the outer detection arm of the collision sensor is impacted or pressed down by force, the inner reed is toggled, and the circuit is turned on.

2.Project Introduction:

By judging whether the collision sensors on the left and right sides of the front end of the car collide, the car is controlled to perform the corresponding steering action.

①When the collision sensor on the left collides, the trolley performs backward and right turn actions;

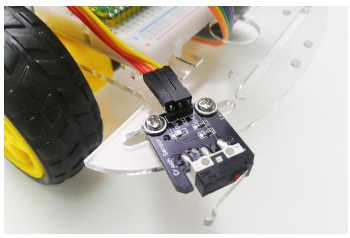
②When the collision sensor on the right collides, the trolley performs backward and left turn actions;

③When the collision sensors on the left and right sides collide at the same time, the car performs backward and U-turn actions;

④When the collision sensors on the left and right sides do not collide, the car keeps moving forward.

3.Assembly steps:

3.1 Place the collision sensor on the right front of the front of the car, and fix the collision sensor on the car with screws and nuts;



3.2 In the same way, fix the other Collision sensor on the other side of the front of the car.

4.Circuit connection:

4.1 Material preparation: Smart car (motor already assembled)*1, Collision sensor*2, Dupont line;

4.2 Port connection:

Raspberry Pi Pico	Collision sensor(left)
VSYS	VCC
GP17	OUT
GND	GND

Raspberry Pi Pico	Collision sensor(right)
VSYS	VCC
GP18	OUT
GND	GND

5.Program analysis: Bumper Cars

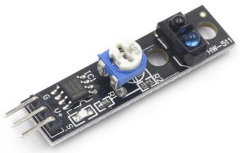
```
from machine import Pin,ADC,PWM
from utime import sleep

CollisionPin_L = 17          # Collision sensor(left)
CollisionPin_R = 18          # Collision sensor(right)
MotorPinA_1A = 10
MotorPinA_1B = 11
MotorPinB_1A = 12
MotorPinB_1B = 13

def setup():
    global motorA1
    global motorA2
    global motorB1
    global motorB2
    global Collision_L
    global Collision_R
    motorA1 = PWM(Pin(MotorPinA_1A))
    motorA2 = PWM(Pin(MotorPinA_1B))
    motorB1 = PWM(Pin(MotorPinB_1A))
    motorB2 = PWM(Pin(MotorPinB_1B))
    Collision_L = Pin(CollisionPin_L,Pin.IN,Pin.PULL_UP)
    Collision_R = Pin(CollisionPin_R,Pin.IN,Pin.PULL_UP)

def motor(A1,A2,B1,B2):
    motorA1.duty_u16(A1)
    motorA2.duty_u16(A2)
    motorB1.duty_u16(B1)
    motorB2.duty_u16(B2)

def loop():
    while True:
        Sum = Collision_L.value() * 2 + Collision_R.value()
        print(Sum)
        speed = 50000
        sleep(0.01)
        if Sum == 0:                    # No collision on the left and right
            motor(0,speed,0,speed)      # forward
```

```
            if Sum == 1:                    # Collision on the right
                motor(speed,0,speed,0)      # backward
                sleep(2)
                motor(speed,0,0,speed)      # Turn left
                sleep(2)
            if Sum == 2:                     # Collision on the left
                motor(speed,0,speed,0)       # backward
                sleep(2)
                motor(0,speed,speed,0)      # Turn right
                sleep(2)
            if Sum == 3:                    # Collision on the left and right, U-turn
                motor(speed,0,speed,0)      # backward
                sleep(2)
                motor(speed,0,0,speed)      # Turn left
                sleep(2)

if __name__ == '__main__':
    setup()
    loop()
```

Lesson30: Tracking car

1.IR tracking sensor:

The IR tracking sensor is essentially an IR reflective sensor. The module has one black and one blue diode for transmitting and receiving, the blue diode is used to transmit infrared rays, and the black diode is used to receive the reflected infrared rays. If the infrared ray encounters a white or brighter object, it will be reflected back, and if it encounters a black or darker object, the infrared ray will be absorbed and hardly reflected back. The current position of the sensor can be judged by the change of the electrical signal.



2.Project Introduction:

By judging whether the two IR tracking sensors at the front of the smart car detect the black line, the car is then controlled to perform the corresponding steering action.

① When the infrared tracking sensors on the left and right sides detect the black line, the car moves forward.

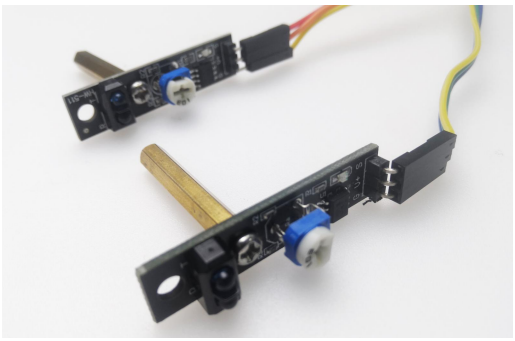
② When the infrared tracking sensor on the left leaves the black line, the car turns right;

③ When the infrared tracking sensor on the right leaves the black line, the car turns left;

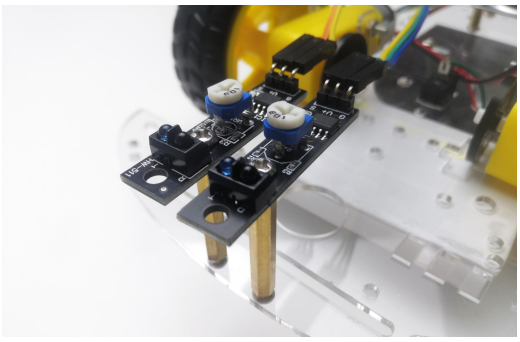
④ When the infrared tracking sensors on the left and right sides do not detect the black line, the car stops.

3.Assembly steps:

3.1 Fix two IR tracking sensors to the hexagonal copper pillars of M2.5*30 respectively;



3.2 Use screws to fix two IR tracking sensors under the front of the car.



4.Circuit connection:

4.1 Material preparation: Smart car (motor already assembled)*1, Tracking map*1, IR tracking sensor*2, Dupont line;

4.2 Port connection:

Raspberry Pi Pico	IR tracking sensor(left)
VSYS	V+
GP17	S
GND	G

Raspberry Pi Pico	IR tracking sensor(right)
VSYS	V+
GP18	S
GND	G

5.Program analysis: Patrol Line Vehicle

```
from machine import Pin,ADC,PWM
from utime import sleep

TrackingPin_L = 17        # IR tracking sensor(left)
TrackingPin_R = 18        # IR tracking sensor(right)
MotorPinA_1A = 10
MotorPinA_1B = 11
MotorPinB_1A = 12
MotorPinB_1B = 13

def setup():
    global motorA1
    global motorA2
    global motorB1
    global motorB2
    global Track_L
    global Track_R
    motorA1 = PWM(Pin(MotorPinA_1A))
    motorA2 = PWM(Pin(MotorPinA_1B))
    motorB1 = PWM(Pin(MotorPinB_1A))
    motorB2 = PWM(Pin(MotorPinB_1B))
    Track_L = Pin(TrackingPin_L,Pin.IN,Pin.PULL_UP)
    Track_R = Pin(TrackingPin_R,Pin.IN,Pin.PULL_UP)

def motor(A1,A2,B1,B2):
    motorA1.duty_u16(A1)
    motorA2.duty_u16(A2)
    motorB1.duty_u16(B1)
```

```
    motorB2.duty_u16(B2)
def loop():
    while True:
        Track = Track_L.value() * 2 + Track_R.value()
        print(Track)
        speed = 50000
        sleep(0.01)
        if Track == 0:                  # No black lines detected on the left and right
            motor(0,0,0,0)              # Stop
        if Track == 1:                  # Only the black line is detected on the right side
            motor(0,speed,speed,0)      # Turn right
        if Track == 2:                  # Only the black line is detected on the left side
            motor(speed,0,0,speed)      # Turn left
        if Track == 3:                  # Black lines are detected on both the left and right
            motor(0,speed,0,speed)      # Go forward

if __name__ == '__main__':
    setup()
    loop()
```

Lesson 31: Obstacle Avoidance Car

1.Ultrasonic ranging sensor:

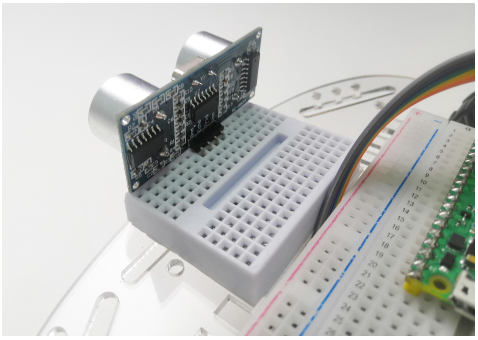
Ultrasonic ranging sensor is sensor that convert ultrasonic signals into electrical signals. The ultrasonic ranging sensor has two pins, one is TRIG (transmitting end) and the other is ECHO (receiving end). It applies the principle of ultrasonic echo ranging and uses accurate time difference measurement technology to detect the distance between the sensor and the target.

2.Project Introduction:

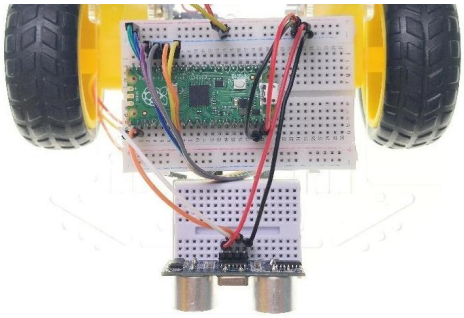
Real-time detection of the distance measurement value of the ultrasonic ranging sensor, when the distance value is less than the preset value, the car will be controlled to turn left to avoid obstacles, so as to achieve the effect of the obstacle avoidance car.

3.Assembly steps:

3.1 Paste the breadboard (small) to the front of the car, and insert the ultrasonic ranging sensor into the breadboard with the two probes facing forward;



3.2 Use Dupont wire to connect the circuit as detailed in the table below.



4.Circuit connection:

4.1 Material preparation: Smart car (motor already assembled)*1, Ultrasonic ranging sensor)*1, Dupont line;

4.2 Port connection:

Raspberry Pi Pico	Ultrasonic ranging sensor
GP1	Echo
GP0	Trig
VSYS	VCC
GND	GND

5.Program analysis: Obstacle Avoidance Car

```
from machine import Pin,PWM
import utime

MotorPinA_1A = 10
MotorPinA_1B = 11
MotorPinB_1A = 12
MotorPinB_1B = 13
motorA1 = PWM(Pin(MotorPinA_1A))
motorA2 = PWM(Pin(MotorPinA_1B))
motorB1 = PWM(Pin(MotorPinB_1A))
motorB2 = PWM(Pin(MotorPinB_1B))
speed = 50000

trig= Pin(0, Pin.OUT)
echo = Pin(1, Pin.IN)

def motor(A1,A2,B1,B2):
        motorA1.duty_u16(A1)
        motorA2.duty_u16(A2)
        motorB1.duty_u16(B1)
        motorB2.duty_u16(B2)

def getDistance(trig, echo):
    trig.low()                  # Generate 10us square wave
    utime.sleep_us(2)
    trig.high()
    utime.sleep_us(10)
    trig.low()

    while echo.value() == 0:
        start = utime.ticks_us()
    while echo.value() == 1:
        end = utime.ticks_us()
    d = (end - start) * 0.0343 / 2
    return d

def loop():
```

```
    while True:
        distance = getDistance(trig, echo)     # Get ultrasonic calculation distance
        print("distance:  {:.2f} cm".format(distance))
        utime.sleep(0.1)
        if distance < 30:
            motor(speed,0,0,speed)              # Turn left
            utime.sleep(0.3)
        else:
            motor(0,speed,0,speed)              # Go forward

if __name__ == "__main__":
    loop()
```

Lesson32: Remote Control Car

1.IR remote control & IR receiver module:

The IR remote control is used to generate the remote control code pulse, drive the Infrared emission tube to output the infrared remote control signal, and the IR receiver module completes the amplification, detection, shaping and demodulation of the remote control code pulse for the remote control signal.

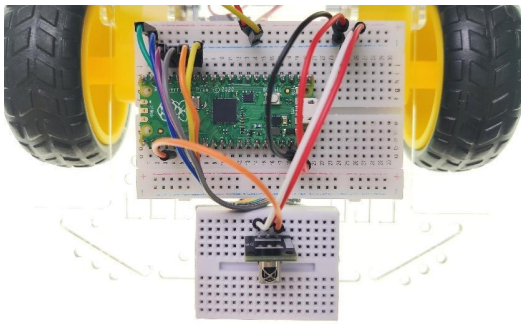
Infrared remote control is a wireless, non-contact control technology, with strong anti-interference ability, reliable information transmission, low power consumption, low cost, easy implementation, etc.It is widely used by many electronic devices, especially household appliances, and is increasingly applied to computer systems.

2.Project Introduction:

Send a movement command signal to the IR receiver module through the IR remote control, and control the car to perform actions such as forward, backward, left turn and right turn.

3.Assembly steps:

Remove the ultrasonic ranging sensor, insert the IR receiver module into the breadboard, and use the dupont lines to connect the IR receiver module to the Raspberry Pi Pico for communication.



4.Circuit connection:

4.1 Material preparation: Smart car (motor already assembled)*1, IR receiver module*1, IR remote control*1, Dupont line;

4.2 Port connection:

Raspberry Pi Pico	IR receiver module
GP2	S
VSYS	VCC (+)
GND	GND (-)

5.Program analysis: Remote Control Car

```
from machine import Pin,PWM
import utime

PIN = Pin(2,Pin.IN,Pin.PULL_UP)
MotorPinA_1A = 10
MotorPinA_1B = 11
MotorPinB_1A = 12
MotorPinB_1B = 13
motorA1 = PWM(Pin(MotorPinA_1A))
motorA2 = PWM(Pin(MotorPinA_1B))
motorB1 = PWM(Pin(MotorPinB_1A))
motorB2 = PWM(Pin(MotorPinB_1B))
speed = 50000

def motor(A1,A2,B1,B2):
    motorA1.duty_u16(A1)
    motorA2.duty_u16(A2)
    motorB1.duty_u16(B1)
    motorB2.duty_u16(B2)
N = 0

def exec_cmd(key_val):
    if(key_val == 0x18):
#         print("Button ^")
        motor(0,speed,0,speed)   # Go forward
    elif(key_val == 0x08):
#         print("Button <")
        motor(speed,0,0,speed)   # Turn left
    elif(key_val == 0x5a):
#         print("Button >")
        motor(0,speed,speed,0)   # Turn right
    elif(key_val == 0x52):
#         print("Button V")
        motor(speed,0,speed,0)   # Go back
    else:
        motor(0,0,0,0)   # Stop
#         print("STOP")

if __name__ == '__main__':
    while True:
        if PIN.value() == 0:
            count = 0
```

```
        while PIN.value() == 0 and count < 200:
            count += 1
            utime.sleep_us(60)
        count = 0
        while PIN.value() == 1 and count < 80:
            count += 1
            utime.sleep_us(60)
        idx = 0
        cnt = 0
        data = [0,0,0,0]
        for i in range(0,32):
            count = 0
            while PIN.value() == 0 and count < 15:
                count += 1
                utime.sleep_us(60)
            count = 0
            while PIN.value() == 1 and count < 40:
                count += 1
                utime.sleep_us(60)
            if count > 8:
                data[idx] |= 1<<cnt
            if cnt == 7:
                cnt = 0
                idx += 1
            else:
                cnt += 1
        if data[0]+data[1] == 0xFF and data[2]+data[3] == 0xFF:
            print("Retrieve key: 0x%02x" %data[2])
            N=data[2]
    if PIN.value() == 1:
        motor(0,0,0,0)   # Stop
    else:
        exec_cmd(N)
```

www.elecrow.com

All rights reserved, the right infringement must investigate!