# Quadcopter Control

指導教授：薛智文
組員：趙守浩 李怡慧 蘇威倫 林慶苗

# Introduction

A quadcopter, also called a quadrotor helicopter or quadrotor, is a multirotor helicopter that is lifted and propelled by four rotors. Quadcopters are classified as rotorcraft, as opposed to fixed-wing aircraft, because their lift is generated by a set of rotors (vertically oriented propellers).

There are some cases use quadcopter to make their life greater, such as Aerial Farming With GoPro and DJI Phantom 2.

## Aerial Farming With GoPro and DJI Phantom 2

Phantom 2 Vision (Photo credit: echeng)

I've been talking about unmanned aerial vehicles (UAVs) for several weeks now, but now I finally have my very own DJI Phantom 2. I'm writing this post on a Saturday, and I've had the UAV and related gear in my possession since this past Tuesday. I've never been great with RC stuff, and flying is the worst. But I'm already flying pretty well with my Phantom because of all the tech built into the UAV. With a compass and GPS on board this thing is super stable. If I get in trouble I just take my hands off the controls and it sits still wherever it may be. If I have trouble getting it back, have a low battery, or lose connection it will come right back and land from wherever it launched because the GPS locks in the starting position. I can also just flip a fail safe switch or turn of the radio and the ship will come home. And with live video feed being transmitted from my GoPro Hero3+ Black Edition I can see whatever the camera is seeing which allows me to fly far beyond line of sight. Trust me, line of sight is not very far! This feature also allows me to pick up flying easily because it's like a video game for me, and I've spent quite a few hours with a controller in my hands.

## Hardware

But how about the hardware design?

What can we improve to make QUADCOPTER more user friendly, more efficiency…etc. So we think about:

- Improve PID control ?

  Which can make the QUADCOPTER fly more influently

- Extend The Area of Remote Control?

  (Bluetooth 20M ,Antenna 300M)

  Which can control far more than before, and QUADCOPTER can be used on more cases which need bigger area

- Reduce Response Time?

  Which can make the QUADCOPTER achieve more action, not just turning left, turning right.

- Extend The Fly Time?(25 minutes)

  Now PHANTOM 2 can fly for 25 minutes, but it's still need to improve.

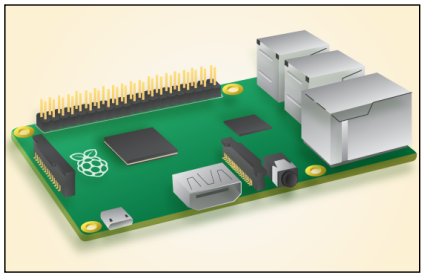
We decide to pick up the

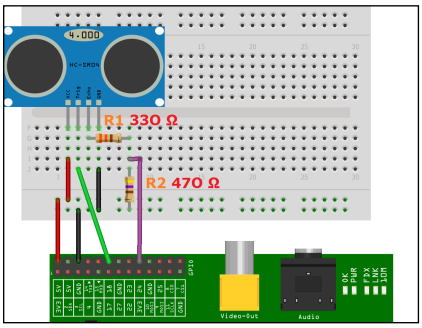
- Extend The Area of Remote Control?

  (Bluetooth 20M ,Antenna 300M)

Because if it solved, or we can extend a little area of remote control, it could be truly useful.

## HERE'S OUR TOOLS FOR PROJECT

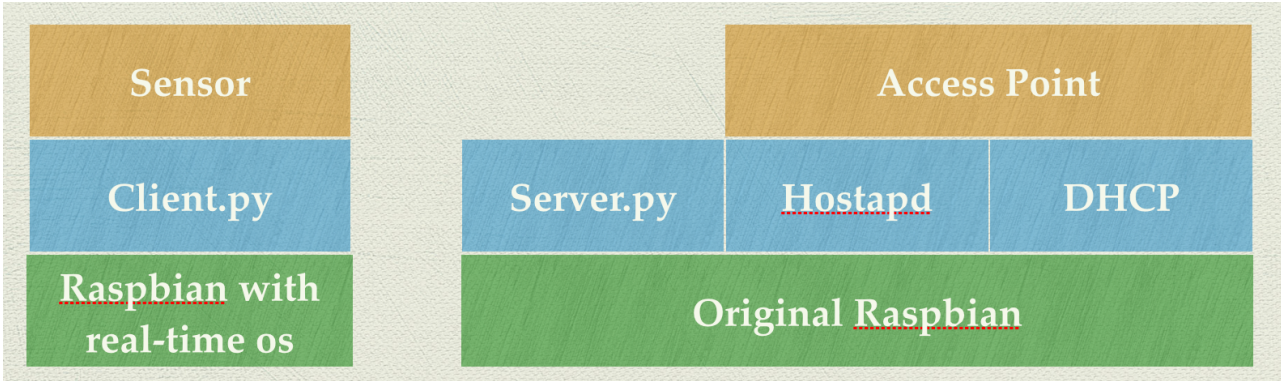





## TO DO

At first we try to apply the sdk of PHANTOM 2, but the version is too old, so we must try another plan.

Here's our new plan :




We use 2 Raspberry PI, one for server and another for client, we want to build the server and client ourselves to solve the sdk problem, at the end we actually done it.

## Actually Do

Connection

Interaction

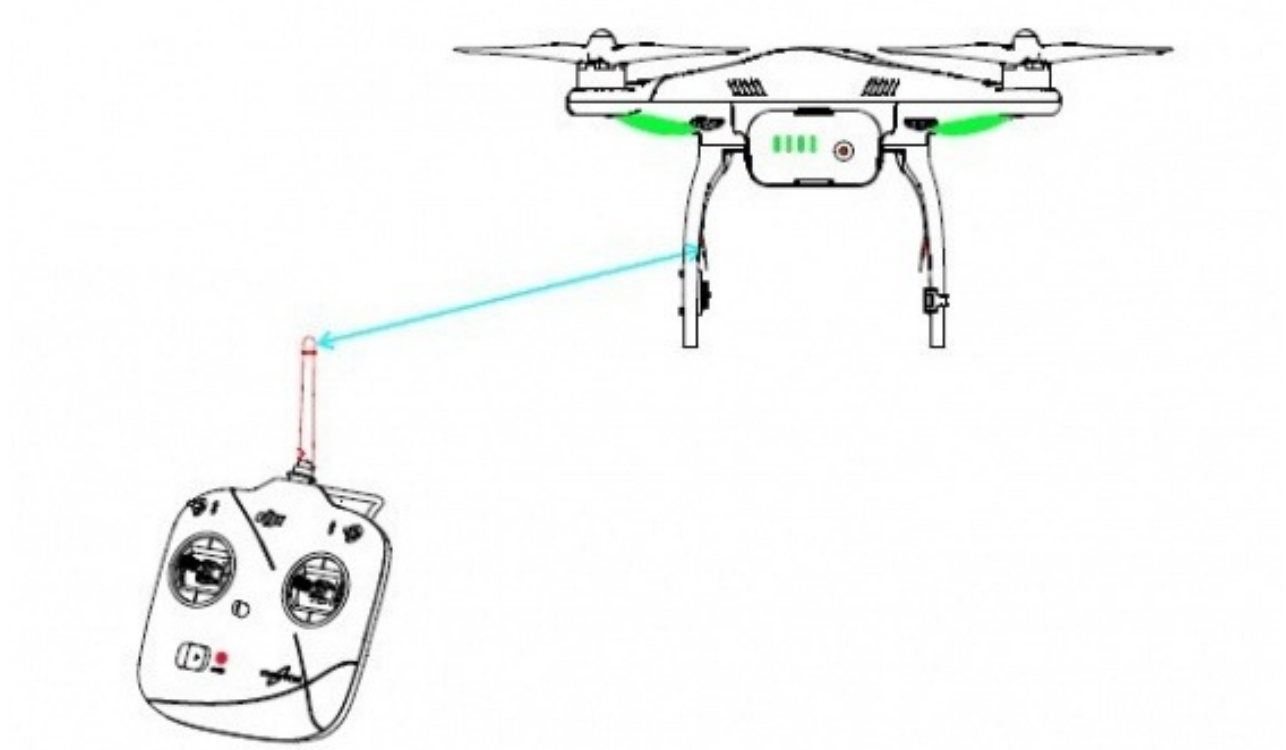
Sensor of Distance

GPS

### Connection

**BEFORE—>Antenna Orientation**

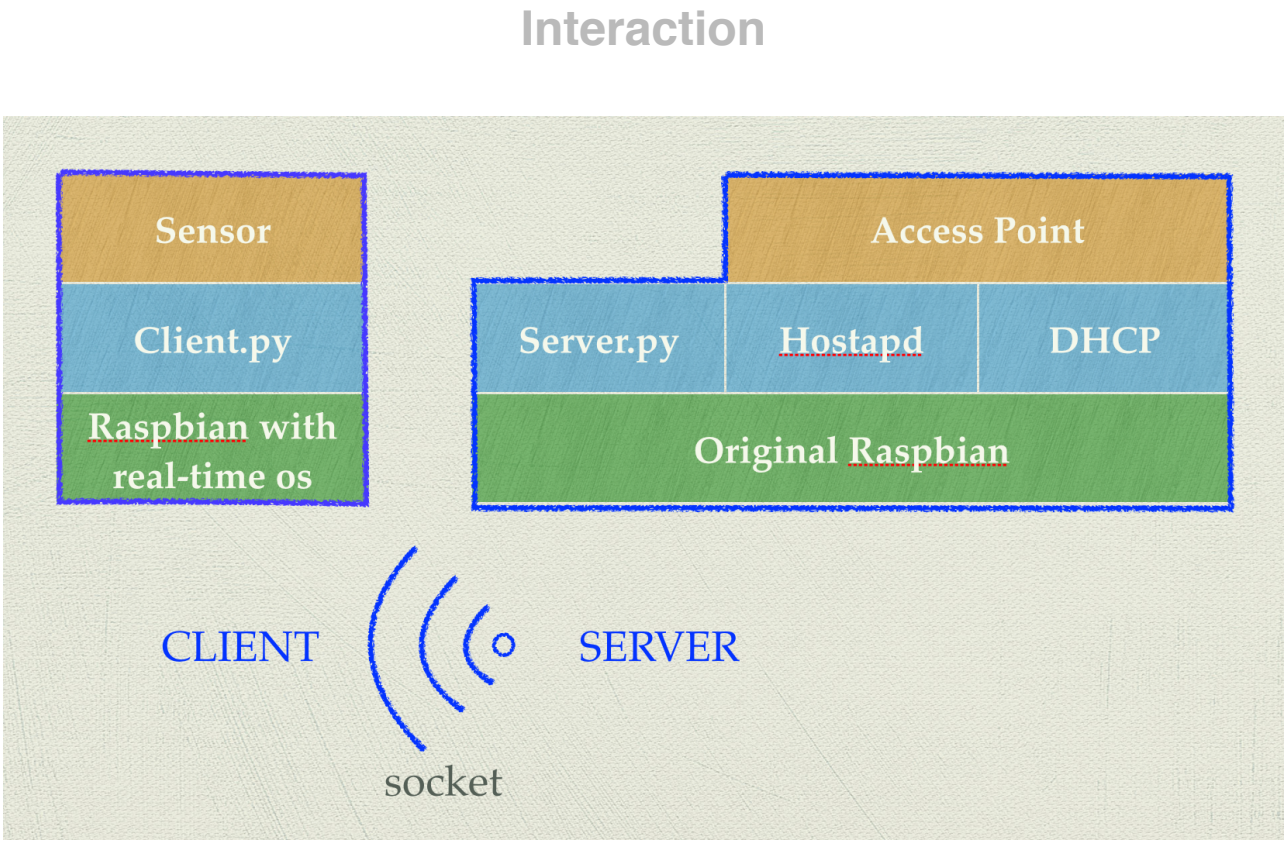
The remote controller's antenna should point skywards without obstructions for maximum communication range during flight.



**AFTER—>802.11 n**

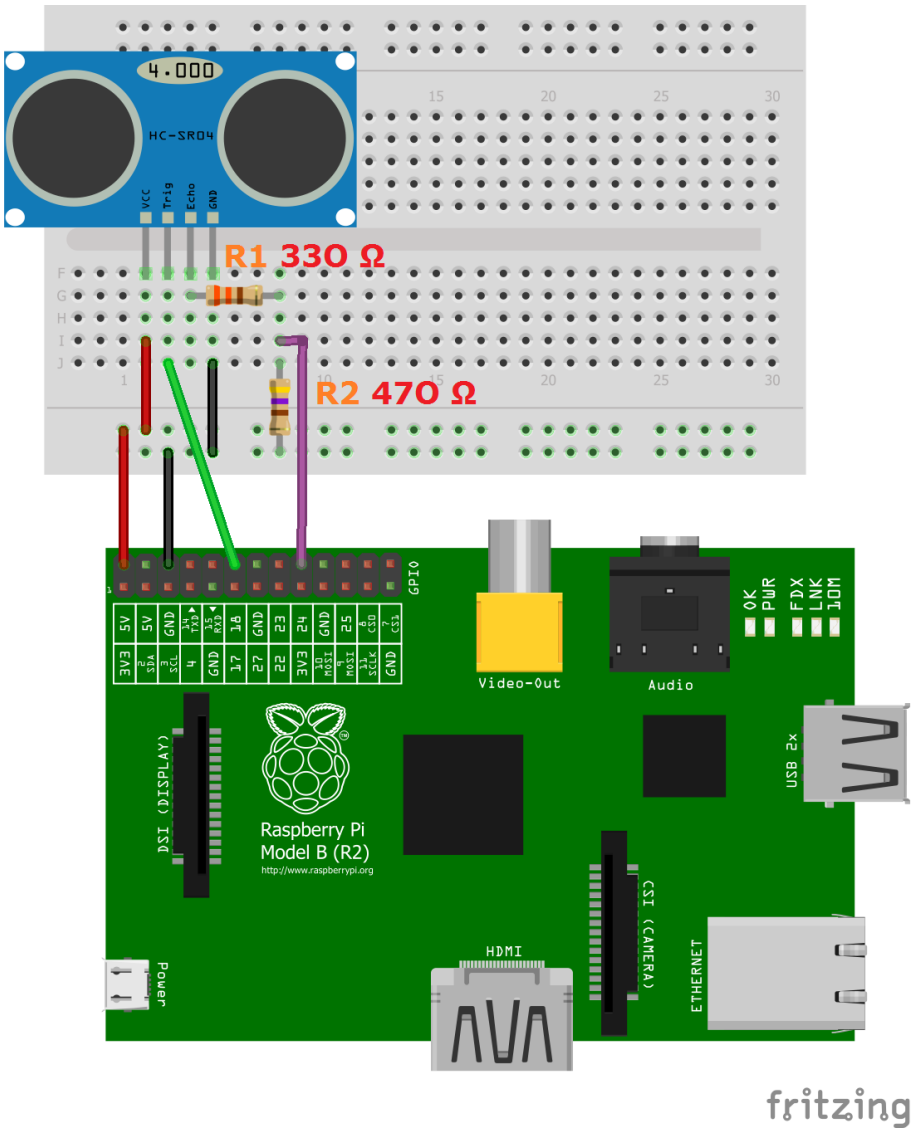
We use wifi for our remote control, and we can approximate approach the outdoor range for 250 m / 820 ft

| 802.11 Protocol | Freq (GHz) | Bandwidth (MHz) | Approximate indoor range | Approximate outdoor range |
|---|---|---|---|---|
| - | 2.4 | 20 | 20 m / 66 ft | 100 m / 330 ft |
| a | 3.7 - 5 | 20 | 35 m / 115 ft | 120 m / 390 ft |
| b | 2.4 | 20 | 35 m / 115 ft | 140 m / 460 ft |
| g | 2.4 | 20 | 38 m / 125 ft | 140 m / 460 ft |
| n | 2.4 - 5 | 20 - 40 | 70 m / 230 ft | 250 m / 820 ft |




We build Access Point on server end. And Sensor end and Server end can connect each other.

## Sensor of Distance



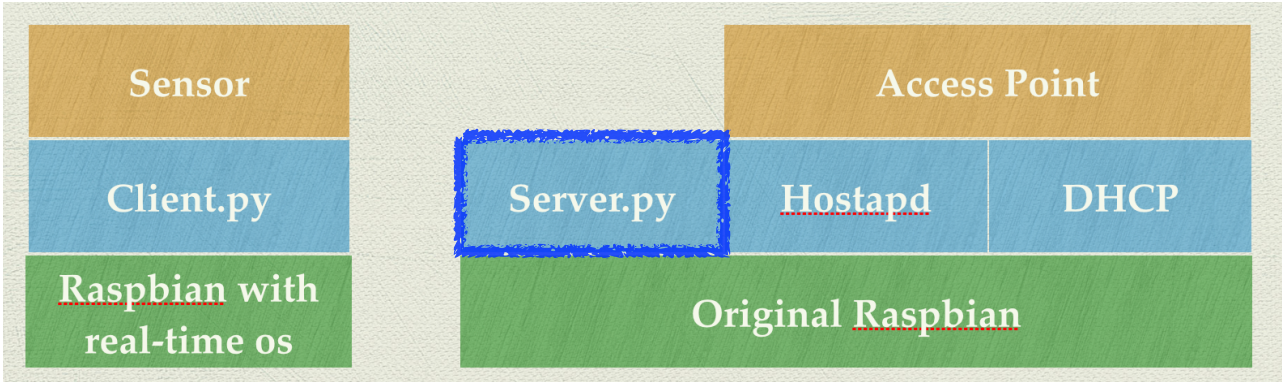

- sensor name : hc sr04
- pin name : echo
- usage : detect distance

We use this sensor for safety.

## GPS

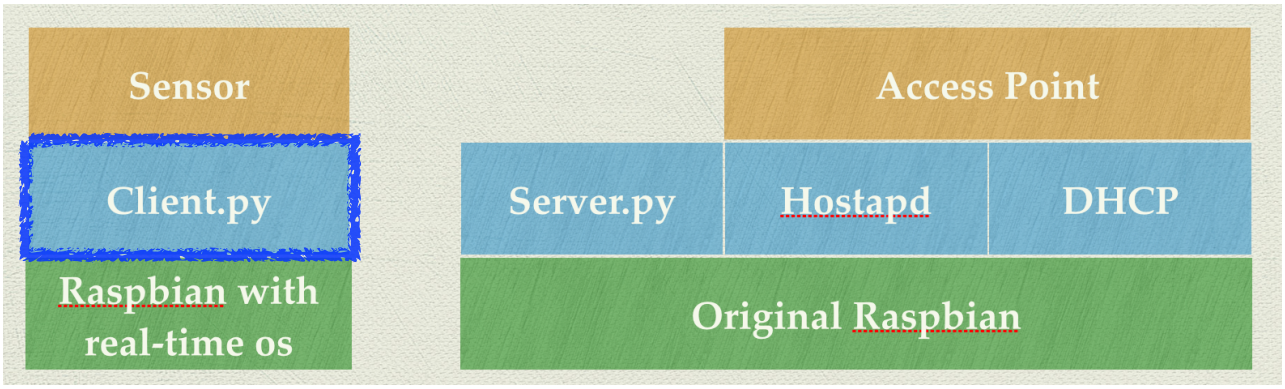
We add GPS, for the future can automatic-flying system.

## Source Code



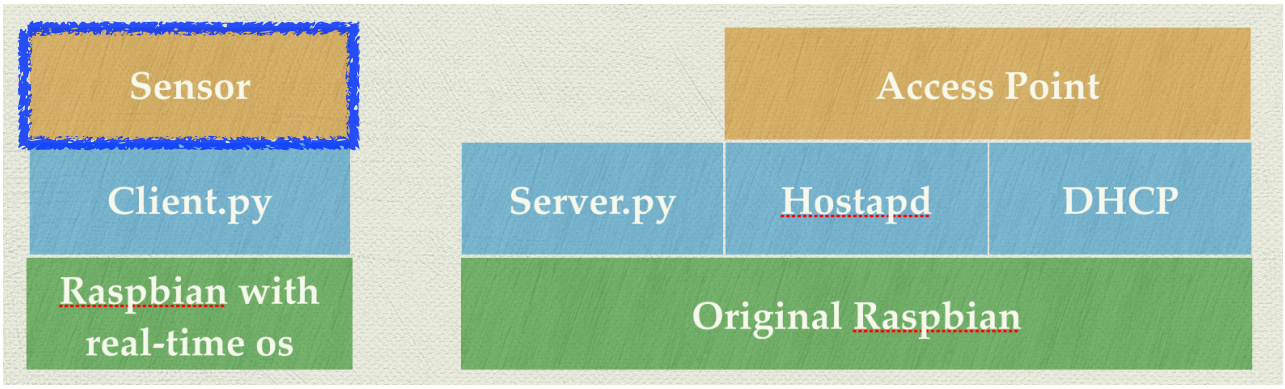

server.py

```
import socket, sys, time

try:
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
except socket.error, msg:
    sys.stderr.write("[ERROR] %s\n" % msg[1])
    sys.exit(1)

sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1) #reuse tcp
sock.bind(('localhost', 54321))
sock.listen(5)
cmd = ''
#sock.settimeout(10)
while True:
    (csock, adr) = sock.accept()
    print "Client Info: ", csock, adr
    print "Plz enter A)launching B)Landing:"
    while cmd != 'A':
        print "Plz enter A)launching B)Landing:"
        cmd = raw_input()
    while True:
        msg = csock.recv(1024)
        if not msg:
            print "ERROR:Lose connection with device!!"
            break
        else:
            print "Client send: " + msg
            if msg == "unstable":
                csock.send("Reset to the least stable state")
            elif msg == "rightShift":
                csock.send("Add power to the right side")
            elif msg == "leftShift":
                csock.send("Add power to the left side")
            elif msg == "hinder":
                csock.send("Turn right/left")
            elif msg == "lowPower":
                csock.send("Return voyage")
            elif msg == "unknown":
                csock.send("Reset to the least stable state")
            elif msg == "outOfRange":
                csock.send("Check gps")
            else:
                csock.send("good job")
#csock.close()
```




## client.py

```
import socket, sys
import distanceope
import gps
import time
import random

situate =
["unstable",”RightShift”,”LeftShift”,”Hinder”,”LowPower","unknown”,”OutOfRange”,”Stable!”,”Stable!”
,”Stable!”,”Stable!”,”Stable!"]
startGPS = getdata(initialise()) #Get GPS data of Starting point

#Initial
try:
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
except socket.error, msg:
    sys.stderr.write("[ERROR] %s\n" % msg[1])
    sys.exit(1)

try:
    sock.connect((’192.168.0.11’, 54321))
except socket.error, msg:
    sys.stderr.write("[ERROR] %s\n" % msg[1])
    exit(1)



while True:
        if csock.recv(1024) == "Launching":
                start = time.clock()

            #avoid of traffic accident
            if get_distance() < 20:
                sock.send("Obstacle")
                while get_distance() < 50:
                    sock.send("Go Back")
                    #to do: control motor to go back
                sock.send("Maintain safe distance")
            if time.clock() - start >=2: #Report in every 2 sec
                num = random.randint(0,11)
                msg = situate[num]
                sock.send(msg)
                print sock.recv(1024)
                start = time.clock()
#sock.close()
```




## distance.py

```
import RPi.GPIO as GPIO
import time
trigger_pin = 23
echo_pin = 24

GPIO.setmode(GPIO.BCM)
GPIO.setup(trigger_pin, GPIO.OUT)
GPIO.setup(echo_pin, GPIO.IN)

def send_trigger_pulse():
    GPIO.output(trigger_pin, True)
    time.sleep(0.001)
    GPIO.output(trigger_pin, False)

def wait_for_echo(value, timeout):
    count = timeout
    while GPIO.input(echo_pin) != value and count > 0:
        count = count - 1

def get_distance():
    send_trigger_pulse()
    wait_for_echo(True, 5000)
    start = time.time()
    wait_for_echo(False, 5000)
    finish = time.time()
    pulse_len = finish - start
    distance_cm = pulse_len * 340 *100 /2
    distance_in = distance_cm / 2.5
    return (distance_cm, distance_in)
```

# Future

- Present to Quadcopter