Guardian Angel

Member list:

- 李晨宇 1063514
  - Online research 、Python coding、Circuit deployment
- 陳亮瑜 1063523
  - Raspberry implementation

Idea of Final Project:

As a student, in our career every time when we have computer class we always try to suffering some secret website or play games during lecture time. An automated desktop switching system comes to our mind. We use US-015 ultrasonic sensor to help us implement the Guardian Angel. The Guardian Angel will automatically switch window when detecting an object that passes through the sensor. However, the function will not be triggered every time. The sensor will consistently return the distance between the sensor and the object. When the distance over a threshold, the functionality of switching window will later be triggered. By fulfilling this function, we need Raspberry to communication with the OS we are using, which is Window 10, we first connect Raspberry by using SSH via Putty, while executing the program we connect laptop and Raspberry by using a cable to transmit the message which program output, so the program on Raspberry and Windows 10 can run simultaneously. By solving the communication phase, the program on the laptop can receive this data and use it as the flag to control the switching on Windows 10.
We want to use this device like a watchdog to help us check whether the teacher comes or not. If do so, we act like a good kid who is fully focused on the course and stay cool on our game.

Operations and Process:

Raspberry pi 3:

```
sudo apt-get install python-serial
```

```
sudo apt-get install python-pip
```

```
sudo apt-get install hcsr04sensor
```

Check Serial port of device manager
python hc_sr04.py (sometime needs sudo~)

PC:

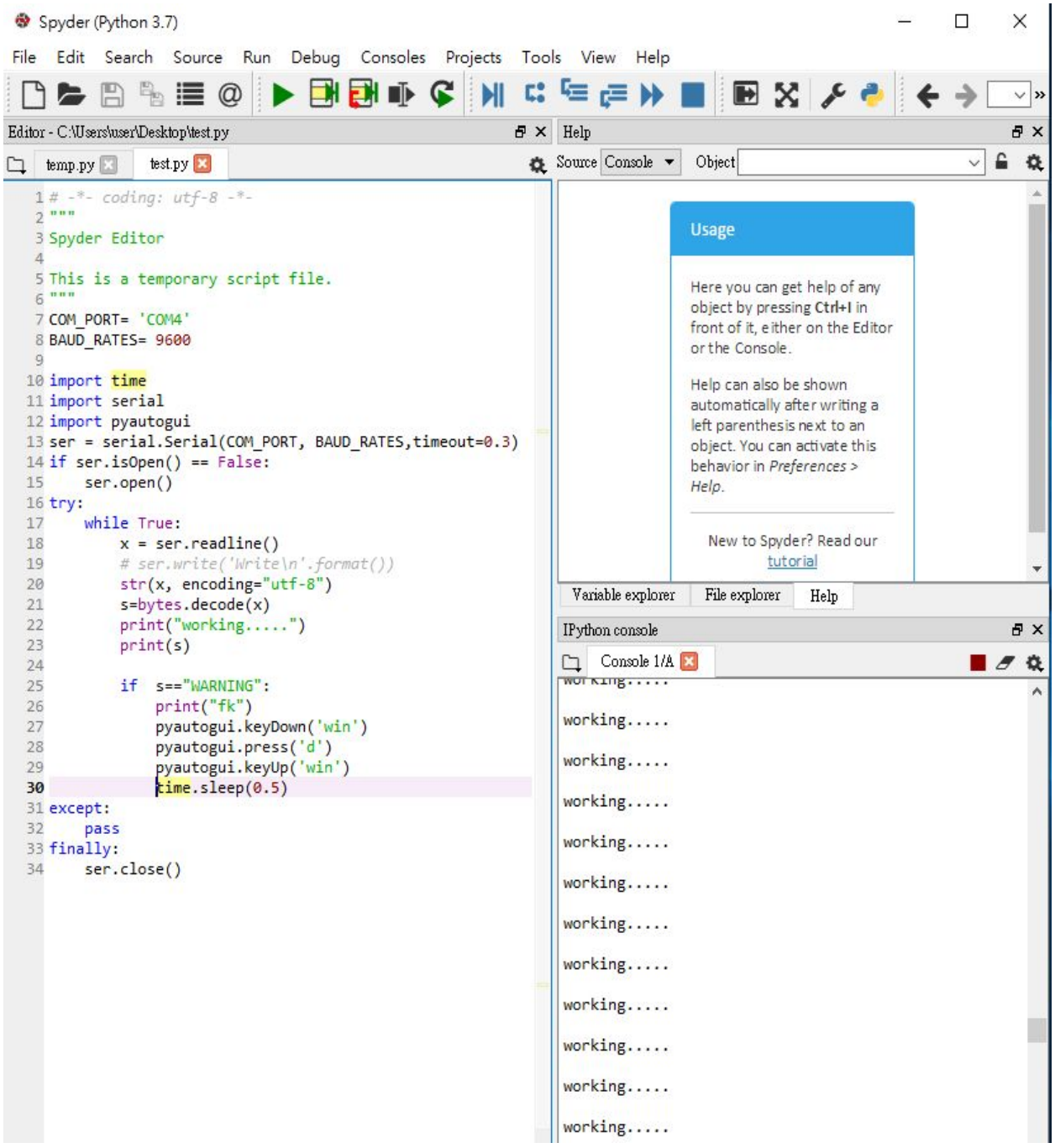
```
pip install pyserial
```

```
pip install pyautogui
```

```
pip install time
```

RUN Test.py

Computer receiver

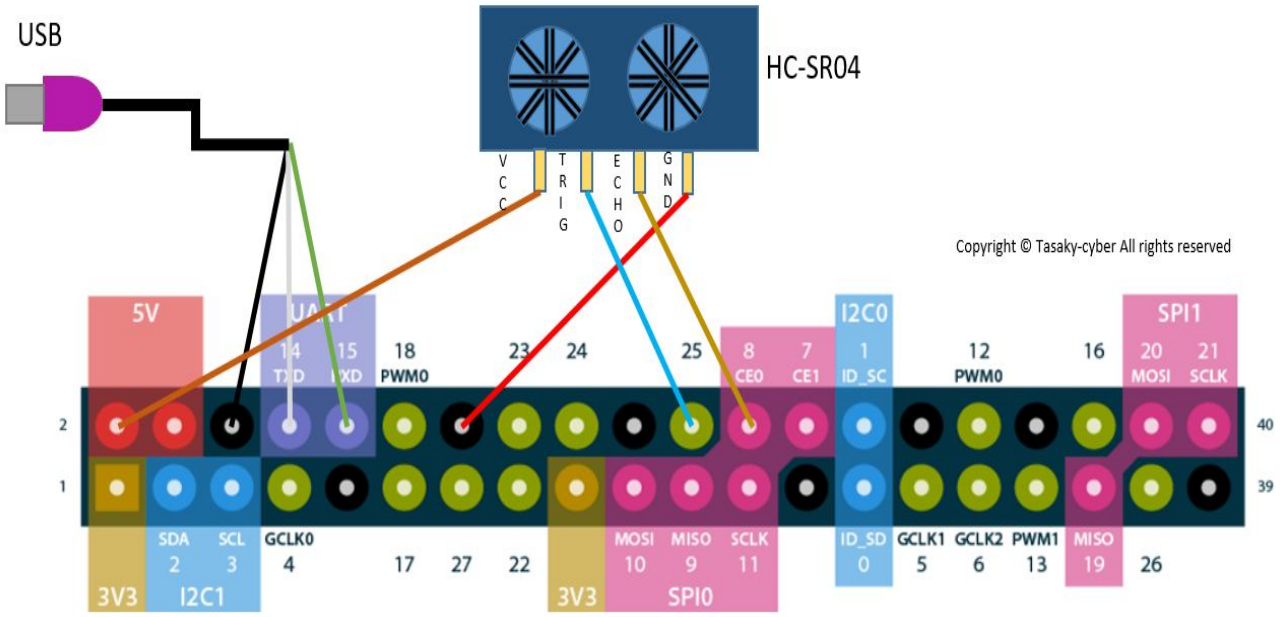


Raspberry Pi 3

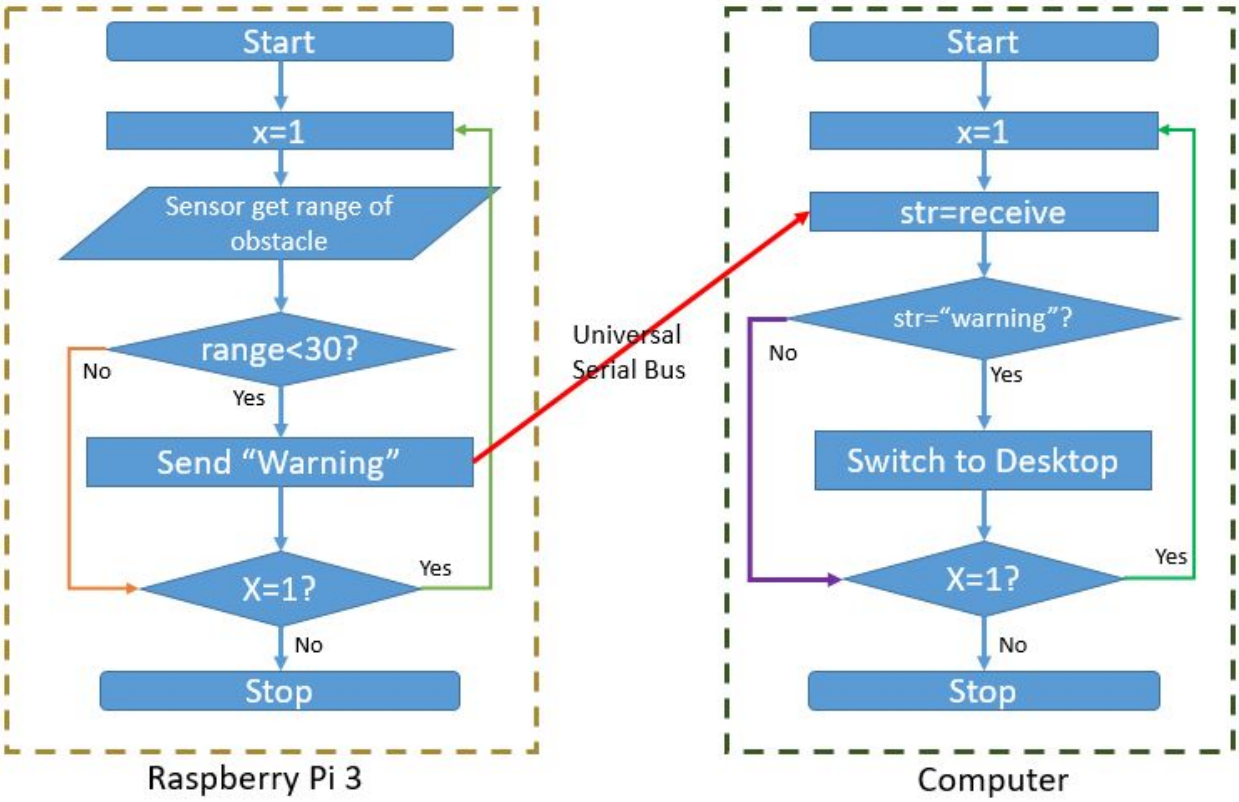
```
pi@raspberrypi: ~
  File "/usr/local/lib/python2.7/dist-packages/hcsr
04sensor/sensor.py", line 86, in raw_distance
    raise SystemError("Echo pulse was not received"
)
SystemError: Echo pulse was not received
pi@raspberrypi:~ $ sudo python hc_sr04.py
按下 Ctrl-C 可以中斷程式
Range 168.7 cm
Range 168.8 cm
Range 168.7 cm
Range 168.7 cm
Range 168.7 cm
Range 168.5 cm
Range 168.6 cm
Range 168.5 cm
Range 167.9 cm
Range 163.0 cm
Range 17.9 cm
Range 25.8 cm
Range 29.6 cm
Range 163.4 cm
Range 163.4 cm
Range 163.4 cm
Range 163.1 cm
Range 163.5 cm
Range 163.4 cm
Range 164.2 cm
Range 163.0 cm
Range 162.6 cm
```

```
pi@raspberrypi: ~
 GNU nano 3.2          hc_sr04.py

# -*- coding: utf-8 -*-
from hcsr04sensor import sensor
import time
import serial

TRIGGER_PIN = 25
ECHO_PIN = 8

ser = serial.Serial(
       port='/dev/ttyAMA0',
       baudrate = 9600,
       parity=serial.PARITY_NONE,
       stopbits=serial.STOPBITS_ONE,
       bytesize=serial.EIGHTBITS,
       timeout=1
)
if ser.isOpen() == False:
    ser.open()

try:
    print('按下 Ctrl-C 可以中斷程式')
    while True:
        sr04 = sensor.Measurement(TRIGGER_PIN, ECH$
        raw_measurement = sr04.raw_distance()
        distance = sr04.distance_metric(raw_measur$
        print('Range {:.1f} cm'.format(distance))
        time.sleep(0.05)
        if (distance < 30):
            ser.write("WARNING")
except KeyboardInterrupt:
    print('關閉程式 ')
```

Diagram of Circuit:



Program Flowchart



Take experimental curriculum as a reference

Lecture 01 - Introduction_Connect wires to your RPI, RPI Environment Setting
Lecture 02 - Basics_Wi-Fi Setting
Lecture 03 - GPIOs_GPIO, RPI Pinout, Python

The most time-consuming parts of Final Project

Communicate with Raspberry Pi 3 & Computer

At first, we have done a lot of research on how to make computer being able to communicate with Raspberry. However, most of the article just simply provide a simple approach to simulate the communication between Raspberry and other devices by launching multiple terminals. We almost finished other minor issues on programs, but if we can't transmit the data from Raspberry to the computer then these programs are just meaningless. Also, a good practice has to point out is that the version of Raspberry does matter the name of the Universal Asynchronous Receiver/Transmitter. In the beginning we use the wrong transmitter, which we use the same serial port as receiver and transmitter at the same time. And because of lacking relative knowledge, we debug

the problem that nothing is transmitted for a long time until the professor points out this mistake we have done.

Finding methods to simulate the keyboard

In this case we are considering the simulation of the keyboard should like an event triggered by user(shortcut key), or toggling certain windows by manipulating OS process. After some research, we found out that the package Pyautogui perfectly support our idea and functions are intuitive enough for the developer to call it. Pyautogui can perform multiple actions as user input, here we use function keydown() and press() to simulating the shortcut/hotkey done by user. Even though it looks very simple as I described it, in fact, it took us couple of times to achieve this simple purpose without other minor error.

The message received from Raspberry Pi 3 is not a string, "Warning" in bytes

str(Warning) can't directly change the bytes to string, but you print(str(Warning)). It will show the "Warning" on the shell, so the programmer won't detect the problem. In fact, the str(Warning) doesn't change it at all.

```
str(x, encoding = ”utf-8”)
s=bytes.decode(x)
```

The hcsr_04 sensor has Innate physical limitations

The range can detect 2M in theorem,but the receiver on the hcsr_04 is not sensitive. You need to get a large area of reflection, so sometime it will miss it. The whole process of emitting ultrasound and receiving reflection takes approximately 0.5 sec. If you run fast go through the hcsr_04 sensor, it will get the chance to miss the detection.

Youtube:

version1:
https://www.youtube.com/watch?v=NQ7_eZtwEm4

version2:
https://www.youtube.com/watch?v=K0-DYZfPu8Y