

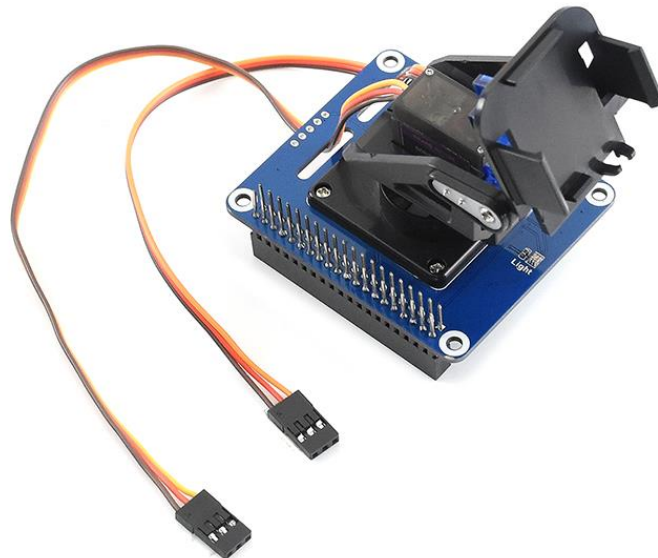


Pan-Tilt HAT

User Manual

OVERVIEW

This is a 2-DOF pan-tilt kit designed for Raspberry Pi. With onboard PCA9685 PWM chip and TSL2581 ambient light sensor, it allows the Pi to control camera movement and sense light intensity through I2C interface.



SPECIFICATION

Operating voltage:	3.3V/5V
PWM driver:	PCA9685
Working voltage:	3.3V
Interface:	I2C
Dimension:	56.6X65(mm)

CONTENT

Overview.....	1
Specification	1
Hardware.....	3
Controller.....	3
Communication protocol.....	3
I2C Write data.....	3
I2C read data.....	4
I2C adress.....	4
How to use	5
Download examples	5
Examples	6
Download examples to Raspberry Pi	6
Install libraries.....	6
Assembly	8
Servo and Light sensor.....	10
Camera	10
Web_Control	12
Expected result.....	16
FAQ.....	17

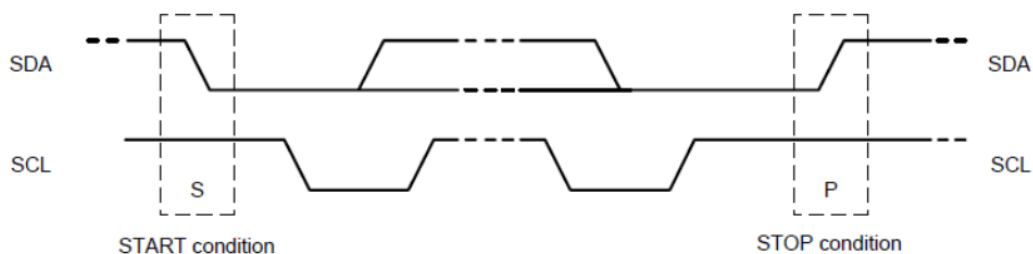
HARDWARE

CONTROLLER

The PWN driver is PCA9685, an I2C-bus controlled 26-channel LED controller, 12-bit resolution PWM output. Pan-Tilt HAT integrate TSL2581 on board. TSL2581 is a light sensor, can be used to detect light and work with camera. TSL2581 use I2C interface as well.

COMMUNICATION PROTOCOL

I2C-bus has one data line(SDA) and one clock line(SCL). When communicating, three kinds of signals are product: Start signal, Stop signal and Answer signal.

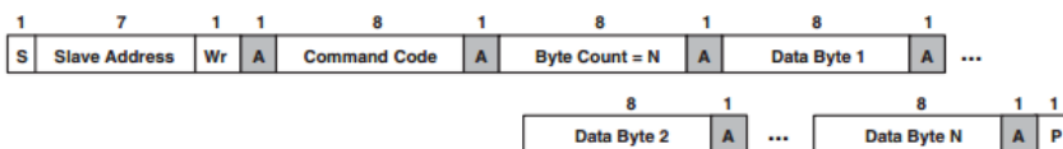


Start signal: SCL is High, SDA changes from High to Low, start to transmit data

Stop signal: SCL is High, SDA changes from Low to High, stop transmitting

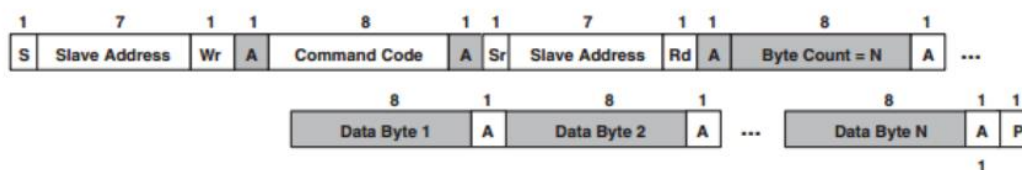
Answer signal: The receiver will answer a Low plus to sender after receiving 8-Bit data as ACK.

I2C WRITE DATA



When working, Raspberry Pi (hereafter named as Master) will first send a Start signal, then send a byte to TSL2581 (hereafter named as Slaver), whose first 7bits are address of Slaver and 1 bit write bit. Slave response with Answer signal every time it receives any data. Master send command register address to Slaver, then data of command register. Stop signals is sent to slave to stop communicating.

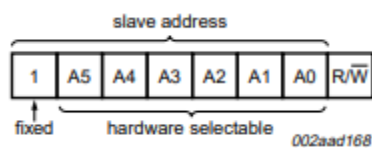
I2C READ DATA



When working, Master will first send a Start signal, then send a byte to Slaver, whose first 7bits are address of Slaver and 1 bit write bit. Slave response with Answer signal every time it receives any data. Master send command register address to Slave. After that, Mater will send a Start signal again, and then send a byte (7bits address and 1bit read bit) to Slaver. Slaver response and send data of the register to Master, master answer as well. Stop signals will be sent to stop communicating.

I2C ADRESS

I2C address of PCA9685:



PCA9685 Datasheet Page 7

I2C address of TSL2581:

Address SEL Terminal Level	Slave Address	SMB Alert Address
GND	0101001	0001100
Float	0111001	0001100
VDD	1001001	0001100

TSL2581 Datasheet Page 13

Note: The default I2C address pins are set as A5=A4=A3=A2=A1=0, address is 0x40.

I2C address pins are set as Float and its I2C address is 0x39 by default. If you use the module with other development board, please add R/W bit to Low bit

HOW TO USE

This part shows you how to use the module based on demo codes provided on wiki


DOWNLOAD EXAMPLES

Search with key word "Pan-Tilt HAT" on Waveshare Wiki, open the wiki page and download examples:

Resources

- [User Manual](#)
- [Demo code](#)
- [Schematic](#)

Uncompressing the 7z, you can get files as below:

 Light Sensor	2019/1/8 10:57	文件夹
 Servo Driver	2019/1/8 10:57	文件夹
 test	2019/1/8 10:57	文件夹
 web_Python	2019/1/8 10:40	文件夹

Servo Driver: Examples which test servos (BCM2835, WiringPi and Python)

Light Sensor: Ambient light sensing examples (BCM2835, WiringPi and Python)

test: test codes, used before assembling

web_Python: Remote control example

EXAMPLES

DOWNLOAD EXAMPLES TO RASPBERRY PI

You can download the demo code from wiki, uncompressing and copy to Raspberry Pi, or directly clone it from github:

Open terminal of Raspberry Pi and download it:

```
git clone https://github.com/waveshare/Pan-Tilt-HAT
```

Change it execute permission and enter the folder:

```
sudo chmod 777 -R Pan-Tilt_HAT  
  
cd Pan-Tilt_HAT
```

INSTALL LIBRARIES

To run the examples, you need to install related libraries first (wiringPi, bcm2835 and python), otherwise, examples cannot work properly.

BCM2835 libraries:

<http://www.airspayce.com/mikem/bcm2835/>

Download the library from bcm2835 libraries and install:

```
wget http://www.airspayce.com/mikem/bcm2835/bcm2835-1.58.tar.gz  
  
sudo tar zxvf bcm2835-1.xx.tar.gz  
  
cd bcm2835-1.xx  
  
sudo ./configure  
  
make
```

```
sudo make check  
  
sudo make install
```

Note: The xx is the version number you download, for example, if the version you download is bcm2835-1.52. then the command you should execute is `sudo tar zxvf bcm2835-1.52.tar.gz`

wiringPi libraries:

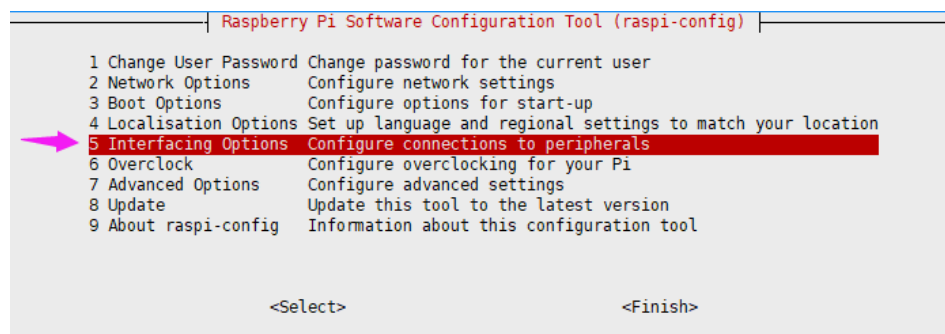
```
sudo apt-get install git  
  
sudo git clone git://git.drogon.net/wiringPi  
  
cd wiringPi  
  
sudo ./build
```

Python libraries:

```
sudo apt-get install python-pip  
  
sudo pip install RPi.GPIO  
  
sudo pip install spidev  
  
sudo apt-get install python-imaging  
  
sudo apt-get install python-smbus
```

Enable I2C interface:

```
sudo raspi-config
```



```
Raspberry Pi Software Configuration Tool (raspi-config)  
  
1 Change User Password Change password for the current user  
2 Network Options Configure network settings  
3 Boot Options Configure options for start-up  
4 Localisation Options Set up language and regional settings to match your location  
5 Interfacing Options Configure connections to peripherals  
6 Overclock Configure overclocking for your Pi  
7 Advanced Options Configure advanced settings  
8 Update Update this tool to the latest version  
9 About raspi-config Information about this configuration tool  
  
<Select> <Finish>
```

```

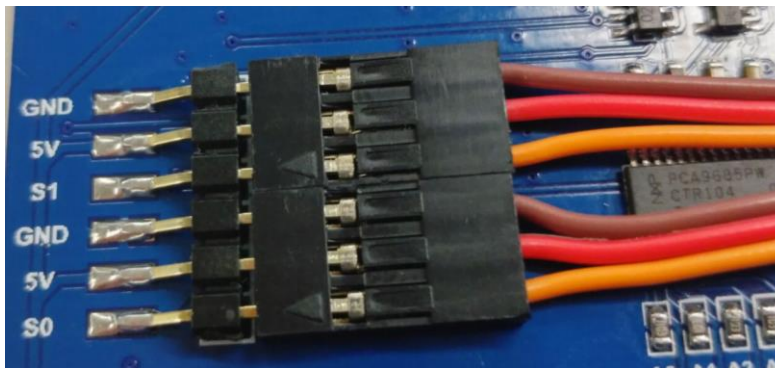
Raspberry Pi Software Configuration Tool (raspi-config)

P1 Camera      Enable/Disable connection to the Raspberry Pi Camera
P2 SSH         Enable/Disable remote command line access to your Pi using SSH
P3 VNC         Enable/Disable graphical remote access to your Pi using RealVNC
P4 SPI         Enable/Disable automatic loading of SPI kernel module
P5 I2C         Enable/Disable automatic loading of I2C kernel module
P6 Serial      Enable/Disable shell and kernel messages on the serial connection
P7 1-Wire      Enable/Disable one-wire interface
P8 Remote GPIO Enable/Disable remote access to GPIO pins
  
```

ASSEMBLY

Note: Before you assemble servos to Pan-Tilt HAT, please test the servo with test codes to avoid of servo stuck when rotate

1. Connect servos to Pan-Tilt HAT



Brown wire	GND
Red wire	5V
Yellow wire	S1/S0

You should connect the Pan servo (close to HAT board) to S1, and connect Tilt servo (close to camera) to S0. Please adjust the angle of the servo when assembling by using the test codes, avoiding of damaging.

2. test code

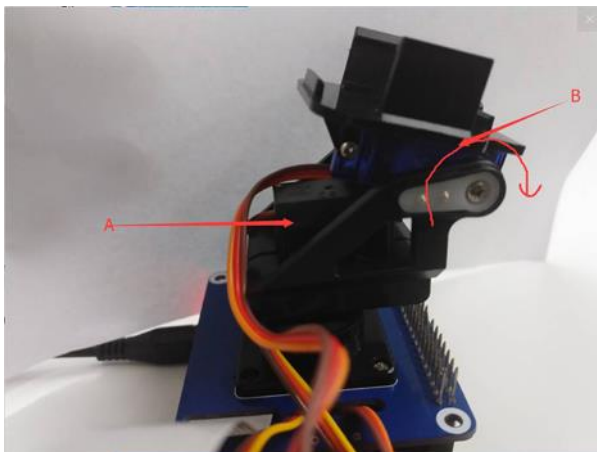
 Light Sensor	2019/1/8 10:57	文件夹
 Servo Driver	2019/1/8 10:57	文件夹
 test	2019/1/8 10:57	文件夹
 web_Python	2019/1/8 10:40	文件夹

3. Run the test code

```
sudo make  
  
sudo ./main
```

4. After running, both servos will rotate to 0-degree place (The starting place). Then power off and assemble the servo as assemble guide. (Do not rotate the servo when assembling)

Assemble guide: <https://www.waveshare.com/img/devkit/accBoard/Pan-Tilt-HAT/Pan-Tilt-HAT-assemble.jpg>



A: Tilt servo

B: Pan servo

The starting status of servos are as image above, and the direction of arrow are the rotate direction of servo.

SERVO AND LIGHT SENSOR

To run the Servo Driver and Light Sensor examples. Please enter the folder and using following commands to execute programs:

BCM2835 examples:

```
cd bcm2835  
sudo ./main
```

wiringPi examples:

```
cd wiringpi  
sudo ./main
```

Python examples:

```
sudo python main.py
```

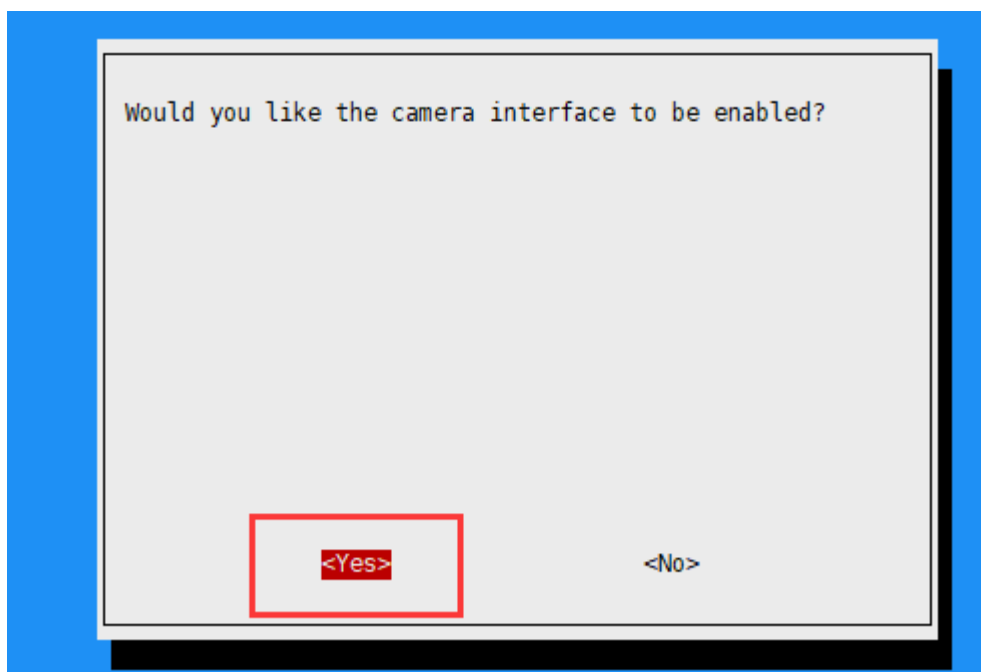
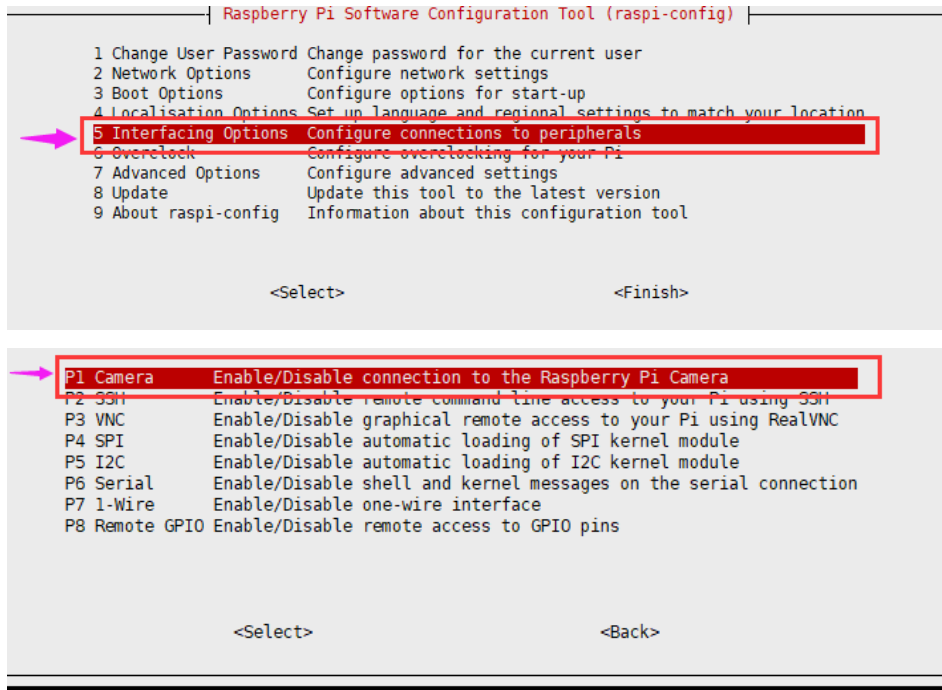
Note: If it prompt that files is not exist when running bcm2835 or wiringpi codes, please first execute command **make** and try again.

CAMERA

To use camera, you need to fist do settings

1. Connect Camera to Raspberry Pi
2. Enable Camera

```
sudo raspi-config
```



3. Reboot Raspberry Pi

```
sudo reboot
```

4. Test camera

You can test the camera with command: `raspistill -o image.jpg`

About details about the `raspistill` command, you can type `raspistill -hell` on

Terminal

Note that you should connect a display to Raspberry Pi for previewing camera.

5. Video recording

To record video via camera, you can use command: `raspivid -o video.h264 -t 1000`

For details about camera command, you can visit Raspberry Pi website.

WEB_CONTROL

1. Enable Camera by following the last chapter

2. Modify the modules file:

```
sudo nano /etc/modules
```

add statements `bcm2835-v4l2` to end of file (Note that `4l2` is the lowercase "l" instead of number 1)

reboot Raspberry Pi then you find a `video0` is appear in `/dev` folder

```
pi@raspberrypi:~$ ls /dev/
autofs          initctl         queue           ram5            tty11           tty29           tty46           tty63           vcs6
block           input           net             ram6            tty12           tty3             tty47           tty7            vcs7
btrfs-control  kmsg           network_latency ram7            tty13           tty30           tty48           tty8            vcsa
bus             log            network_throughput ram8            tty14           tty31           tty49           tty9            vcsa1
cachefiles     loop0          null           ram9            tty15           tty32           tty5             ttyAMA0         vcsa2
char           loop1          ppp            random          tty16           tty33           tty50           ttyprintk       vcsa3
console        loop2          ptmx           raw             tty17           tty34           tty51           ttyS0           vcsa4
cpu_dma_latency loop3          pts            rfcill          tty18           tty35           tty52           uhid            vcsa5
cuse           loop4          ram0           serial0         tty19           tty36           tty53           uinput          vcsa6
disk           loop5          ram1           serial1         tty2             tty37           tty54           urandom          vcsa7
fb0            loop6          ram10          snd             tty20           tty38           tty55           vchiq           vcsm
fd             loop7          ram11          snd             tty21           tty39           tty56           vcio            vhci
full           loop-control  ram12          stderr          tty22           tty4             tty57           vc-mem          video0
fuse           mapper         ram13          stdin           tty23           tty40           tty58           vcs             watchdog
gpiochip0     mem           ram14          stdout          tty24           tty41           tty59           vcs1            watchdog0
gpiochip1     memory_bandwidth ram15          tty            tty25           tty42           tty6             vcs2            zero
gpiochip2     mmcblk0       ram2           tty0            tty26           tty43           tty60           vcs3
gpiomem       mmcblk0p1     ram3           tty1            tty27           tty44           tty61           vcs4
hwrng         mmcblk0p2     ram4           tty10           tty28           tty45           tty62           vcs5
```

3. Install libraries

```
sudo apt-get install libv4l-dev libjpeg8-dev
```

```
sudo apt-get install subversion
```

4. Get information

Enter web_Python folder: `cd Pan-Tilt-HAT/web_Python`

```
pi@raspberrypi:~ $ cd Pan-Tilt-HAT/web_Python/
pi@raspberrypi:~/Pan-Tilt-HAT/web_Python $
```

Type `pwd` to get the current path information:

```
pi@raspberrypi:~/Pan-Tilt-HAT/web_Python $ pwd
/home/pi/Pan-Tilt-HAT/web_Python
pi@raspberrypi:~/Pan-Tilt-HAT/web_Python $
```

Type `ifconfig` command to get the IP information

```
pi@raspberrypi: ~/Pan-Tilt-HAT/web_Python
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.1.212 netmask 255.255.255.0 broadcast 192.168.1.255
inet6 fe80::5507:5edb:bd35:2824 prefixlen 64 scopeid 0x20<link>
ether b8:27:eb:ab:93:05 txqueuelen 1000 (Ethernet)
RX packets 6209 bytes 956955 (934.5 KiB)
RX errors 0 dropped 26 overruns 0 frame 0
TX packets 119 bytes 21634 (21.1 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Note: I connect the Raspberry Pi to WIFI, so the IP address is wlan0's, if you directly connect network jack with cable, the IP address should be eth0's

5. Modify main.py file: `sudo nano main.py`

Find `os.chdir(' ')`, change it to the path information you get above and add `/mjpg` to the end. for example:

```

pi@raspberrypi: ~/Pan-Tilt-HAT/web_Python
GNU nano 2.7.4 File: main.py

# -*- coding: UTF-8 -*-
import threading
import SocketServer
import RPi.GPIO as GPIO
from PCA9685 import PCA9685
from SocketServer import StreamRequestHandler as SRH
from time import ctime
import time

import thread
import os
import sys

o_path = os.getcwd()
sys.path.append(o_path)
os.chdir('/home/pi/Pan-Tilt-HAT/web_Python/mjpg')
from mjpg import camera

pwm = PCA9685()

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Linter ^_ Go To Line

```

Change the host to the IP address of your Raspberry Pi, for example:

```

pi@raspberrypi: ~/Pan-Tilt-HAT/web_Python
GNU nano 2.7.4 File: main.py

sys.path.append(o_path)
os.chdir('/home/pi/Pan-Tilt-HAT/web_Python/mjpg')
from mjpg import camera

pwm = PCA9685()
pwm.setPWMFreq(50)
pwm.setRotationAngle(0, 0)
pwm.setRotationAngle(1, 0)

host = '192.168.1.212'
port = 8000
addr = (host,port)

class Servers(SRH):
    def handle(self):
        global HStep,VStep,VPulse,HPulse
        print 'got connection from ',self.client_address
        self.wfile.write('connection %s:%s at %s succeed!' % (host,port,ctime())$
        VPulse = 0

[ Search Wrapped ]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Linter ^_ Go To Line

```

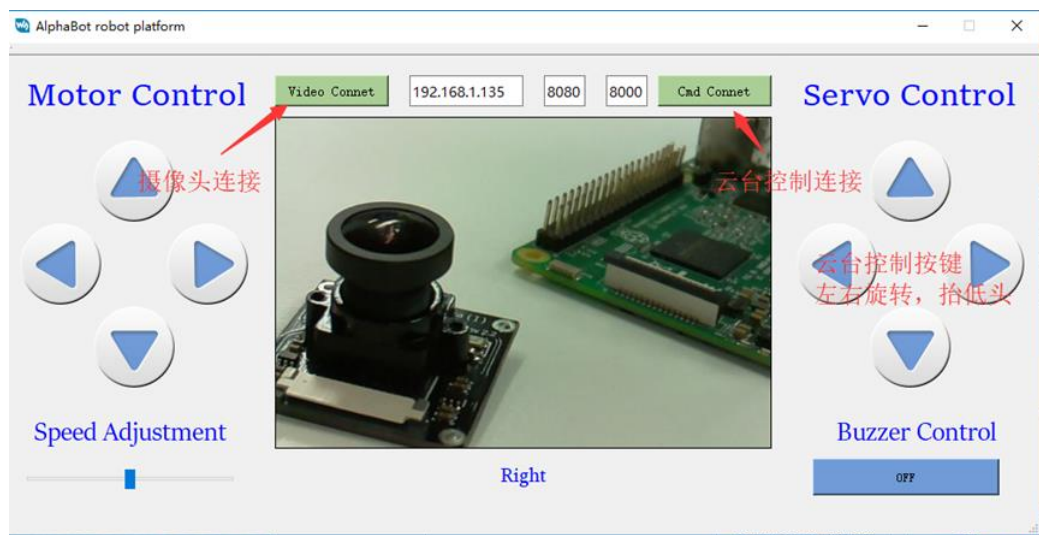
save and exit

6. Run the example: `sudo python main.py`:

```

pi@raspberrypi:~/RaspberryPi/web_Python $ sudo python main.py
server is running...
MJPG Streamer Version: svn rev: Unversioned directory
i: Using V4L2 device.: /dev/video0
i: Desired Resolution: 640 x 480
i: Frames Per Second.: 10
i: Format.....: YUV
i: JPEG Quality.....: 80
Adding control for Pan (relative)
UVC_IOCTL_ADD - Error: Inappropriate ioctl for device
Adding control for Tilt (relative)
UVC_IOCTL_ADD - Error: Inappropriate ioctl for device
Adding control for Pan Reset
UVC_IOCTL_ADD - Error: Inappropriate ioctl for device
Adding control for Tilt Reset
UVC_IOCTL_ADD - Error: Inappropriate ioctl for device
Adding control for Pan/tilt Reset
UVC_IOCTL_ADD - Error: Inappropriate ioctl for device
Adding control for Focus (absolute)
UVC_IOCTL_ADD - Error: Inappropriate ioctl for device
mapping control for Pan (relative)
UVC_IOCTL_MAP - Error: Inappropriate ioctl for device
mapping control for Tilt (relative)
UVC_IOCTL_MAP - Error: Inappropriate ioctl for device
mapping control for Pan Reset
UVC_IOCTL_MAP - Error: Inappropriate ioctl for device
mapping control for Tilt Reset
UVC_IOCTL_MAP - Error: Inappropriate ioctl for device
mapping control for Pan/tilt Reset
UVC_IOCTL_MAP - Error: Inappropriate ioctl for device
mapping control for Focus (absolute)
UVC_IOCTL_MAP - Error: Inappropriate ioctl for device
mapping control for LED1 Mode
UVC_IOCTL_MAP - Error: Inappropriate ioctl for device
mapping control for LED1 Frequency
UVC_IOCTL_MAP - Error: Inappropriate ioctl for device
mapping control for Disable video processing
UVC_IOCTL_MAP - Error: Inappropriate ioctl for device
mapping control for Raw bits per pixel
UVC_IOCTL_MAP - Error: Inappropriate ioctl for device
o: www-folder-path...: ./www/
o: HTTP TCP port.....: 8080
o: username:password.: disabled
o: commands.....: enabled
  
```

7. Download the [AlphaBot.exe](#) software from Waveshare wiki and open it. Type IP address of your Raspberry Pi to it and Click Video Connect and Cmd Connect to enable connection



Note: The software only support Windows PC and some of the functions are unavailable for Pan-Tilt HAT.

- To cancel the program, you need to disconnect Cmd and Camera first by pressing Video Connet and Cmd Connet buttons. The Ctrl+C to stop program.

EXPECTED RESULT

Servo Driver:

The servos will rotate the Pan servo and Tilt servo

Light Sensor:

Print device ID (it is not the I2C address) then output light intensity value

```
pi@raspberrypi:~/RaspberryPi/Light Sensor/bcm2835 $ sudo ./main
bcm2835 init success !!!
READ ID = 0x90
---- i2c sensor init ----
lux = 114
lux = 114
lux = 113
lux = 107
lux = 115
lux = 107
lux = 103
lux = 96
lux = 98
lux = 99
lux = 97
lux = 101
lux = 102
lux = 102
lux = 102
lux = 102
lux = 102
lux = 100
lux = 110
lux = 111
lux = 110
```


FAQ

1. Why the ID printed is 0xf0 or 0x00, and intensity data is 0 after running light sensor code?

A: Check I2C address first by command: `sudo i2cdetect -y 1`:

```
pi@raspberrypi:~$ sudo i2cdetect -y 1
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  39  --  --  --  --  --  --
40:  40  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
```

If the I2C addresses are incorrect in your Raspberry Pi, please check if you have modified I2C address, and check if you have enabled I2C interface. The default I2C device address of Light sensor is 0x39 and 0x40 is PCA9685' s.

2. Why I cannot run python or bcm2835 after running wiringpi example?

If you can run wiringpi example successfully but no the bcm2835 or python codes.

Please restart Raspberry Pi and test the two examples again.