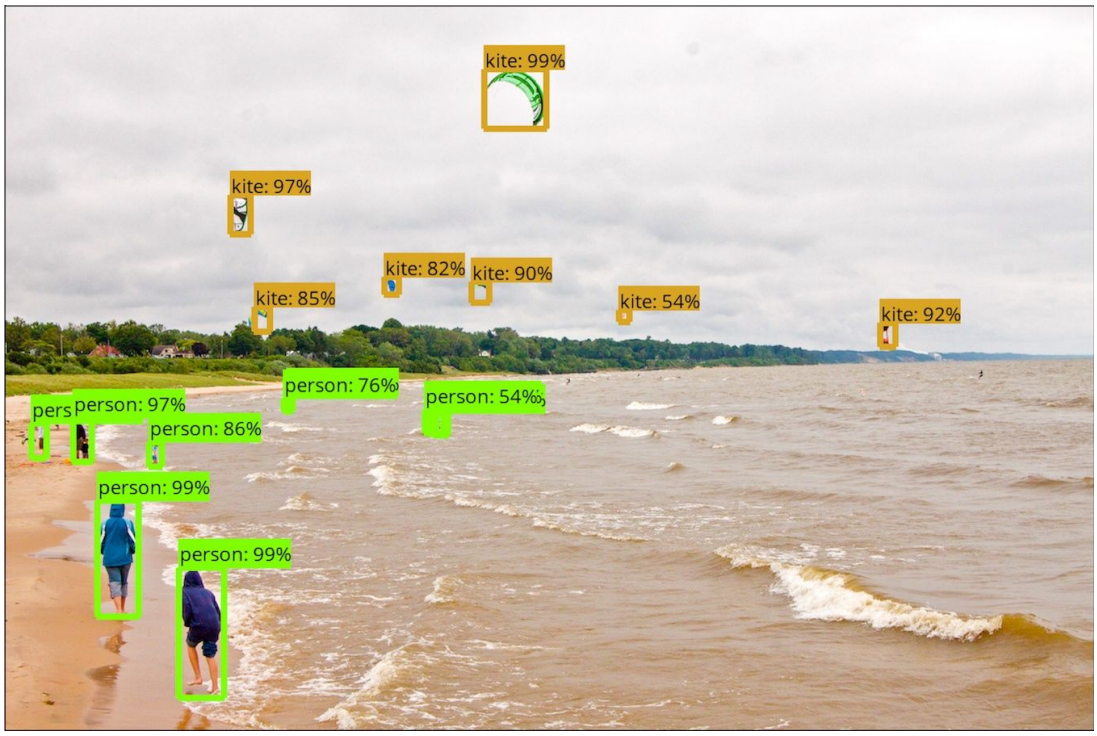
Chapter 3 Hello (Rpi hand waving) Human

Raspberry Pi can also access cameras which opens a whole new world of computer vision applications.

We have got the hardware (Rpi) that allows us to do mobile computations on real-world images and videos (it is just a sequence of images).

Google has a framework named *tensorflow* using which they have released APIs which help to identify objects.



And Raspberry Pi now supports *tensorflow*.

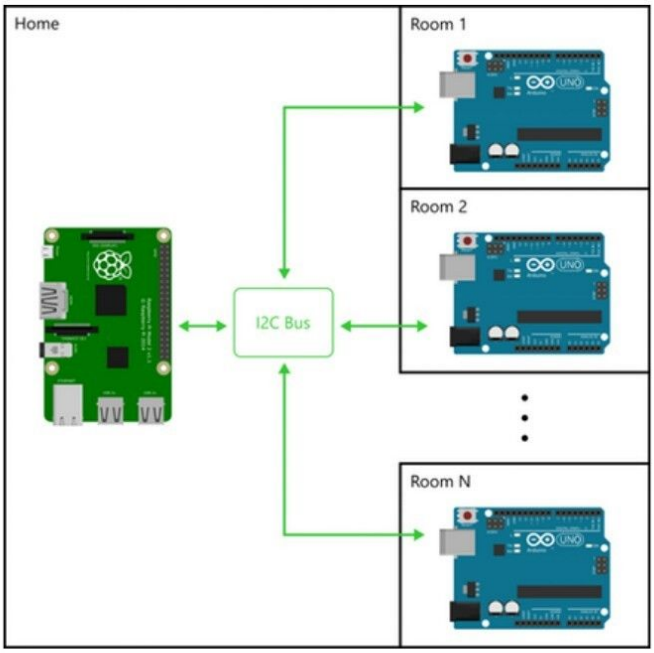
In this task, we will be using pi-cam on raspberry pi to identify a human and use a servo motor as a hand to wave at that human.

Ok, we have pi-cam + Rpi+ *tensorflow* to identify the images.
How do we move a motor using Rpi ? Use the Arduino!

Sure Rpi provides GPIO pins but it's main purpose is computation.

In real-world applications, systems use a master-slave model to ensure robustness and limiting the responsibilities of each component for reliable functioning.

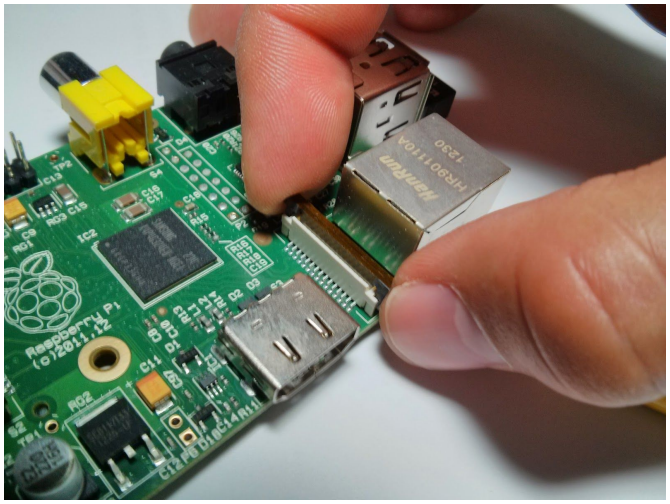
Example of master-slave application - home automation system



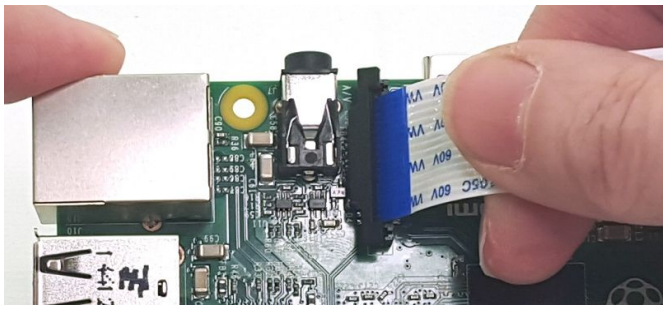
There are different networking protocols for master-slave and multi-device communication like i2c, SPI, CAN, MODBUS, Firmata etc. For the ease of application, we would just use one slave as the Arduino and communicate with it via USB using Firmata library.

Step 1 interface with the pi cam and observe the readings.

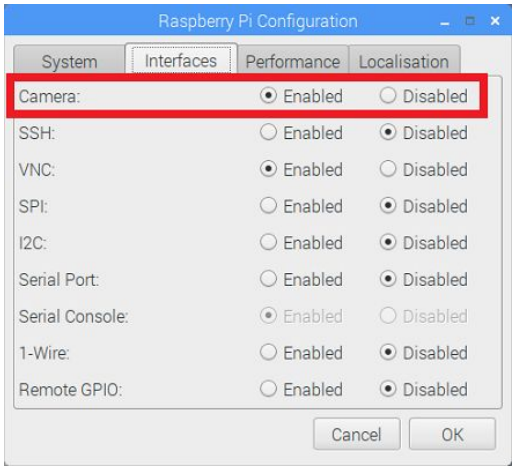
The pi-cam port is between the 3.5mm audio jack and the HDMI port. Lift the plastic tab to be able to insert the flex cable (the paper like thing attached to the pi-cam).



Then make sure that the metal leads on the flex cable are facing the HDMI port. Then push the plastic tab to lock the cable.



The pi-cam port on Rpi is disabled by default, which can be enabled with the following setting in the Rpi configuration menu. We have already enabled it for you.



Download the file from the below mentioned link and edit it to print the values that Rpi is seeing through the pi cam.
The values are being captured in a list(consider it as a bucket full of values/numbers) named *guesses.* Use the print() command to print the values from the *guesses*.
Download the code

Enter the code on line 163 inside the template as shown below. (Would highly recommend editing it in some code editor like sublime)

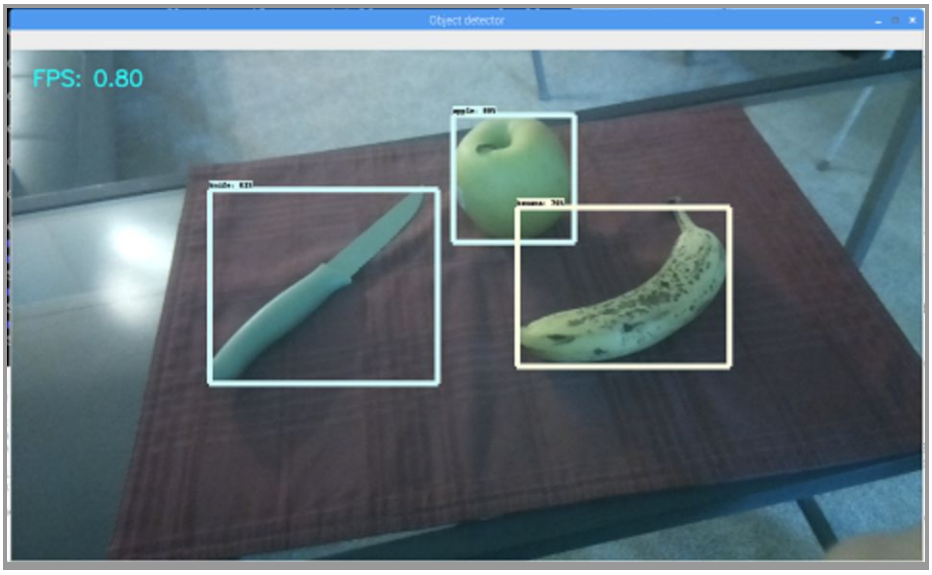
```
##################################edit starts#########################################


##################################edit ends###########################################
```

After editing this file on your computer move it to the Rpi using the following command:
scp *filename.py* pi@*your.ip*:**tensorflow1/models/research/object_detection/**
We will be moving the script in the highlighted path because the tensorflow environment is setup in that path.
Make sure your file is on the following path on your pi:
tensorflow1/models/research/object_detection/

On RPi you will need to navigate to the directory where you transferred the file:
You do this by the *cd* command.
cd **tensorflow1/models/research/object_detection/**

Run this file on Rpi while connected to a screen using the command python3 wave_read.py
Note: This will work only when you are running the script on Rpi locally and not via ssh.

One you run the script, be patient: Wait ~20 second for the red light to turn on and then ~20 more seconds for pi to start printing values.



Run it using the command xvfb-run python3 wave_read.py when not connected to a monitor or a screen on your laptop.
You can run this command from your remote computer and see the values in your laptop.
Again you need to be in that directory:
cd **tensorflow1/models/research/object_detection/**

The xvfb-run part is to inform the system that there is no display connected right now.

```
[pi@raspberrypi:~/tensorflow1/models/research/object_detection $ xvfb-run py
thon3 wave_read.py
/usr/lib/python3.5/importlib/_bootstrap.py:222: RuntimeWarning: compiletime
 version 3.4 of module 'tensorflow.python.framework.fast_tensor_util' does
not match runtime version 3.5
  return f(*args, **kwds)
/usr/lib/python3.5/importlib/_bootstrap.py:222: RuntimeWarning: builtins.ty
pe size changed, may indicate binary incompatibility. Expected 432, got 412
  return f(*args, **kwds)
['person 0.0']
['person 0.71550524', 'person 0.0']
['person 0.79382336', 'person 0.0']
['person 0.0']
['person 0.4886051', 'person 0.0']
['person 0.42387286', 'person 0.0']
['person 0.42529145', 'person 0.0']
```

Take some time to read through different guesses and understand how well the object identification model works.

The values you see beside each label is called the confidence score or prediction probability. The number signifies how sure the system is about its prediction. The numbers will be easy to interpret when converted to percentages.
Task: name 3 other objects it can identify.

Kill the current program by pressing ctrl+c

Step 2 Identify the threshold to identify a human

Download the code

You might have noticed a certain minimum range of score below which it falsely identifies a human which in simple language means it is not sure. This minimum level above which we decide a certain event to happen is called as a threshold value.

Decide a threshold value to identify a human and use the is_human variable in the code to compare it with your threshold value. Print “it’s a human” if the score qualifies as a human according to your threshold value.
The syntax for if statement is:

```
>>> if x < 0:
...     x = 0
...     print('Negative changed to zero')
... elif x == 0:
...     print('Zero')
... elif x == 1:
...     print('Single')
... else:
...     print('More')
...
```

This is just a syntax for if else, do not copy the above code.

Edit the code at line 172 in the template as shown below:
Be careful to edit the code where it falls in line with the line above *is_human = int(float(val))*
Use the value *is_human* and print “it’s a human” if the confidence score is above the threshold.

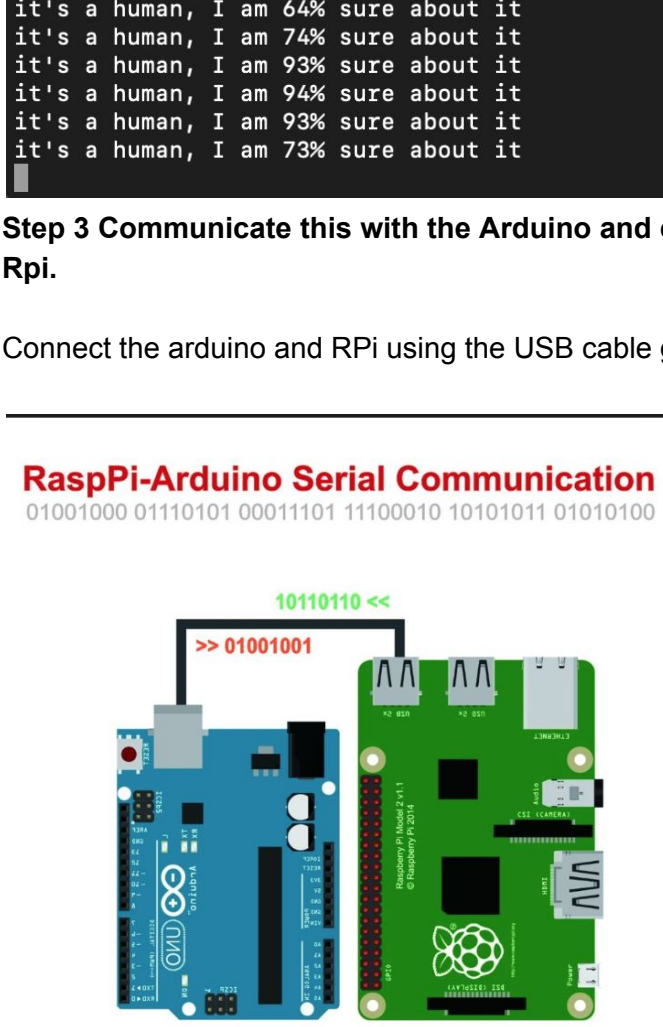
```
#use the value is_human and print "it's a human if the confidence score is above the threshold you decide"
################################edit starts#########################################


################################edit ends###########################################
```

Remember to move your file to the Rpi to the correct path.

```
pi@raspberrypi:~/tensorflow1/models/research/object_detection $ xvfb-run py
thon3 wave_identify_human.py
/usr/lib/python3.5/importlib/_bootstrap.py:222: RuntimeWarning: compiletime
 version 3.4 of module 'tensorflow.python.framework.fast_tensor_util' does
not match runtime version 3.5
  return f(*args, **kwds)
/usr/lib/python3.5/importlib/_bootstrap.py:222: RuntimeWarning: builtins.ty
pe size changed, may indicate binary incompatibility. Expected 432, got 412
  return f(*args, **kwds)
it's a human, I am 64% sure about it
it's a human, I am 74% sure about it
it's a human, I am 93% sure about it
it's a human, I am 94% sure about it
it's a human, I am 93% sure about it
it's a human, I am 73% sure about it
```

Step 3 Communicate this with the Arduino and control the led on Arduino with code from Rpi.

Connect the arduino and RPi using the USB cable given with arduino.



[Download the code]

On Arduino, load the Firmata code, named “StandardFirmata” from the examples.

Now connect the Arduino to a USB port on the Rpi
We first find the device port on Rpi where Arduino is connected using the command ls /dev/tty*
We are asking the system to show all the names in the devices folder starting with the tty.
Arduino will be named as /dev/ttyACM followed by a number.

```
[pi@raspberrypi:~/tensorflow1/models/research/object_detection $ ls /dev/tty*
/dev/tty    /dev/tty17  /dev/tty26  /dev/tty35  /dev/tty44  /dev/tty53  /dev/tty62
/dev/tty0   /dev/tty18  /dev/tty27  /dev/tty36  /dev/tty45  /dev/tty54  /dev/tty63
/dev/tty1   /dev/tty19  /dev/tty28  /dev/tty37  /dev/tty46  /dev/tty55  /dev/tty7
/dev/tty10  /dev/tty2   /dev/tty29  /dev/tty38  /dev/tty47  /dev/tty56  /dev/tty8
/dev/tty11  /dev/tty20  /dev/tty3   /dev/tty39  /dev/tty48  /dev/tty57  /dev/tty9
/dev/tty12  /dev/tty21  /dev/tty30  /dev/tty4   /dev/tty49  /dev/tty58  /dev/ttyACM0
/dev/tty13  /dev/tty22  /dev/tty31  /dev/tty40  /dev/tty5   /dev/tty59  /dev/ttyAMA0
/dev/tty14  /dev/tty23  /dev/tty32  /dev/tty41  /dev/tty50  /dev/tty6   /dev/ttyprintk
/dev/tty15  /dev/tty24  /dev/tty33  /dev/tty42  /dev/tty51  /dev/tty60
/dev/tty16  /dev/tty25  /dev/tty34  /dev/tty43  /dev/tty52  /dev/tty61
```

On Rpi, we will use the pyfirmata library (we have loaded it on your Rpi, on a new Rpi you install it using pip install pyfirmata) to communicate with Arduino and the sleep function for delays when blinking the led. Also, notice in the snapshot below that we have mentioned /dev/ttyACM0 as our board.

```
from pyfirmata import Arduino, util
from time import sleep


board = Arduino('/dev/ttyACM0')
```

Edit the

If is_human > :

print("it's a human")

code from Step 2 to blink the led on the Arduino using the following syntax:

To turn on/off: board.digital[***arduino_pin_number***].write(***1/0***) (1 for on, 0 for off)

For delay use the sleep command

Note: The onboard led on the arduino is wired on pin 18 by default.

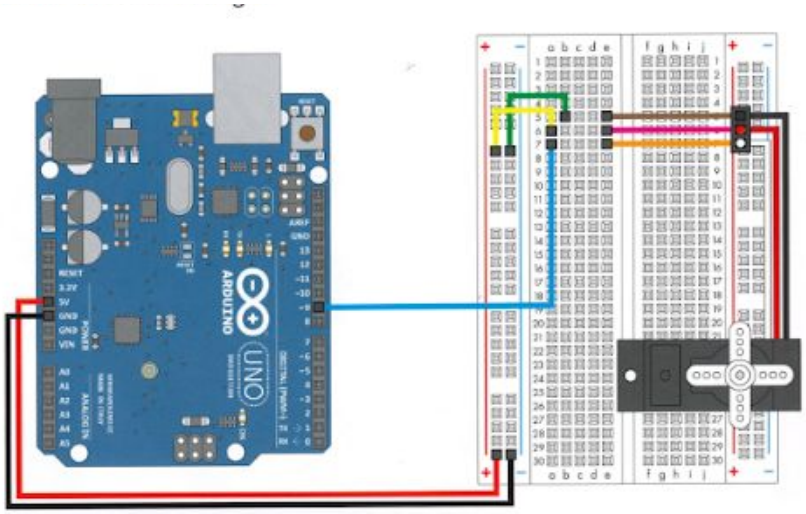
Edit the code starting line 174 from the downloaded file.

Blink the led if the picam sees a human

```
#use the value is_human and print "it's a human if the confidence score is above the threshold you decide"
################################edit starts#########################################



################################edit ends#########################################
```

Remember to move your file to the Rpi to the correct path.

The final step: controlling the servo



Download the code

We have already communicated with the board, now we need to initiate any of the pwm pins to initiate the servo control. We do this using the get_pin function.

```
board = Arduino('/dev/ttyACM0')
servo = board.get_pin('d:3:s')
```

The get_pin function returns the activated pin given by the pin definition.

Syntax for get_pin: get_pin(*x:y:z*)
where:
x : a for analog and d for digital, in our case we will use digital to control the servo.
y : pin number
z : s is for servo

We will use servo.write(*degrees*) function to rotate the servo.

Now, edit the blinky code to rotate the servo using the servo.write function.
Edit the code starting line 175 from the code you downloaded.

```
#use the value is_human and print "it's a human if the confidence score is above the threshold you decide"
################################edit starts########################################




################################edit ends##########################################
```

Remember to move your file to the Rpi to the correct path.
Have fun - Try the code with some other object.