

EX - 12 Touch Sensor

Problem

You want to make a buzzing sound with the Raspberry Pi using touch sensor.

Solution

These example programs show you how to use the capacitive touch sensors with Python and the Raspberry Pi's GPIO. It is relatively straightforward to adapt the programs to do different things. Just change the lines with print "pressed".

Two versions of the code are provided. One will continuously print output to the terminal while the touch sensor is pressed. The other will only print output once, irrelevant of how long the pad is pressed. Both pieces of code will work with the momentary and toggle boards.

Continuous Output:

This version of the code will continuously print output while the pad is pressed.

```
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)

padPin = 23 GPIO.setup(padPin,
GPIO.IN)

while True:
    padPressed = GPIO.input(padPin)

    if padPressed: print
        "pressed"

    time.sleep(0.1)
```

Single Output:

This version of the Python code will only print an output once each time the sensor detects a touch.

```
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)

padPin = 23 GPIO.setup(padPin,
GPIO.IN)
```

```
alreadyPressed = False
```

```
while True:
```

```
    padPressed = GPIO.input(padPin)
```

```
    if padPressed and not alreadyPressed: print "pressed"
```

```
    alreadyPressed = padPressed
    time.sleep(0.1)
```

Running the Code:

Open a terminal. Move to the directory that you saved the code and type in the following command (change the file name to whatever you called your Python file):

```
$ sudo python touchsensor.py
```

EX - 13

Send SMS from a Python Script on the Raspberry Pi

Problem

This exercise can be used to send text messages to either individual numbers or entire contact groups.

Solution

```
pi@raspberrypi ~ $ nano mysms.py
#!/usr/bin/env python
import urllib.request import
urllib.parse
uname = 'bhuvaneswaran@rajalakshmi.edu.in'
hashCode = '9890db2b86d4bf292adfc44fe0e19fc417e6f6a7' sender =
'TXTLCL'
numbers = ('919791519152','919994940914')
message = 'Test message sent from my Raspberry Pi'
def sendSMS(uname, hashCode, numbers, sender, message):
    data = urllib.parse.urlencode({'username': uname, 'hash': hashCode, 'numbers': numbers, 'message'
: message, 'sender': sender})
    data = data.encode('utf-8')
    request = urllib.request.Request("http://api.textlocal.in/send/?") f =
urllib.request.urlopen(request, data)
    fr = f.read()
    return(fr)
resp = sendSMS(uname, hashCode, numbers, sender, message) print (resp)
```

Running the python script: pi@raspberrypi ~ \$

python3 mysms.py

Discussion

| | |
|----------------|---|
| sender | If you have a promotional account, do not send this parameter. If you have a transactional account, use this field to specify the sender name for your message. Sender name must be 6 alpha characters and should be pre-approved by Textlocal. In the absence of approved sender names, use the default 'TXTLCL' |
| message | The message content. This parameter should be no longer than 765 characters. See Helpful Information for message length details. The message also must be URL Encoded to support symbols like &. |

| Login Parameters | |
|----------------------|--|
| username | The email address used to log into Textlocal. |
| hash | Your secure hash can be found within Messenger in the main navigation under " Help->All Documentation ". Alternatively you can use the password parameter instead and use your Textlocal password in plain text. |
| apiKey | This can be used instead of the username/hash/password. You can create them in your Messenger Control Panel (click here) for each application, and limit the usage of them by host IP Address. |
| Optional Parameters | |
| numbers | Comma-delimited list of mobile numbers in international format (i.e. 918123456789). Maximum of 10,000 numbers and error code 33 will be returned if exceeded. |
| group_id | This parameter can be used in place of the numbers parameter in order to send to an entire contact group. This parameter should contain the ID of the relevant group, which can found either within Messenger (in the "Reports" - "Advanced Reports" - "List of Group ID's" section) or by running the get_groups command. Additionally group 5 contains "contacts" and group 6 contains "opt-outs". |
| schedule_time | This parameter can be used to specify a schedule date/time for your message, which should be provided in Unix timestamp format. Times should be provided in GMT. |
| receipt_url | Use this field to specify an alternative URL to which the delivery receipt(s) will be sent. See handling receipts documentation. |

custom

This value will be set against the message batch and

| | |
|-----------------|---|
| | will passed back in the delivery receipts. This allows you to match delivery receipts to their corresponding messages. |
| optouts | Can be set to true in order to check against your own opt-outs list and Textlocal's global opt-outs database. Your message will not be sent to numbers within these lists. If not provided defaults to false . |
| validity | Can be set, up to 72 hours in advance, to say after which time, you don't want the message delivered. This should be in a <u>Unix timestamp</u> format. |
| unicode | Set this value to true to specify that your message body will contain unicode characters. See <u>Encoding/Decoding Unicode</u> Documentation |
| test | Set this field to true to enable test mode, no messages will be sent and your credit balance will be unaffected. If not provided defaults to false |

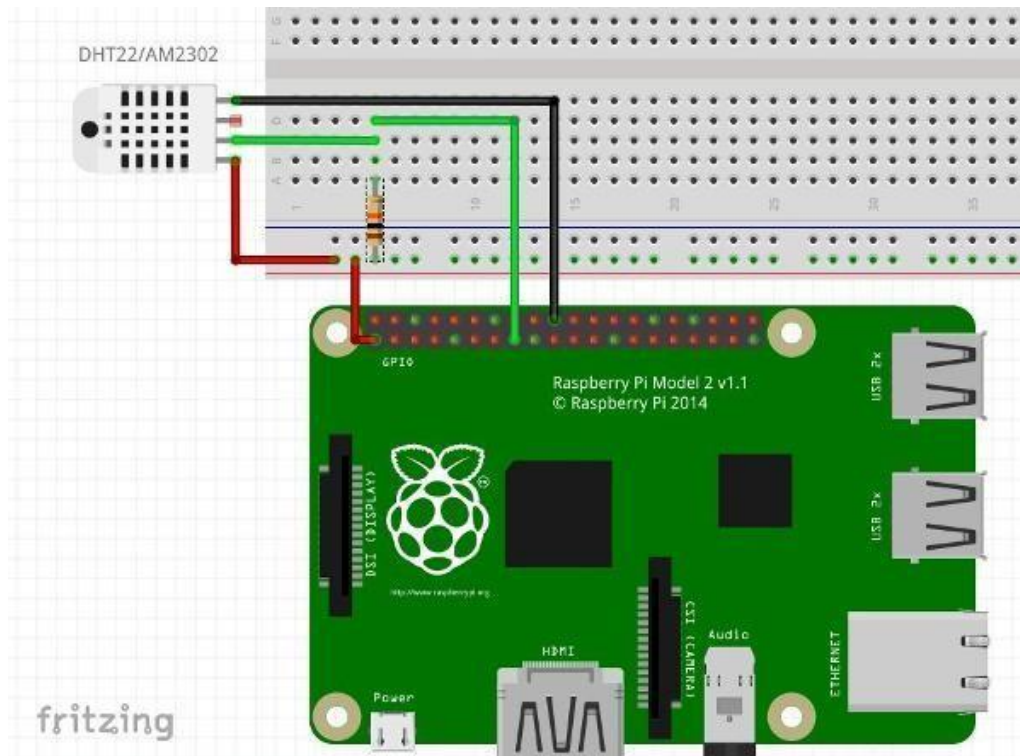
Ex - 14

Measure temperature and humidity with the sensor DHT22/AM2302

DHT.AM2302 is a sensor to measure temperature and humidity and Adafruit has implemented a Raspberry Pi driver for the sensor.

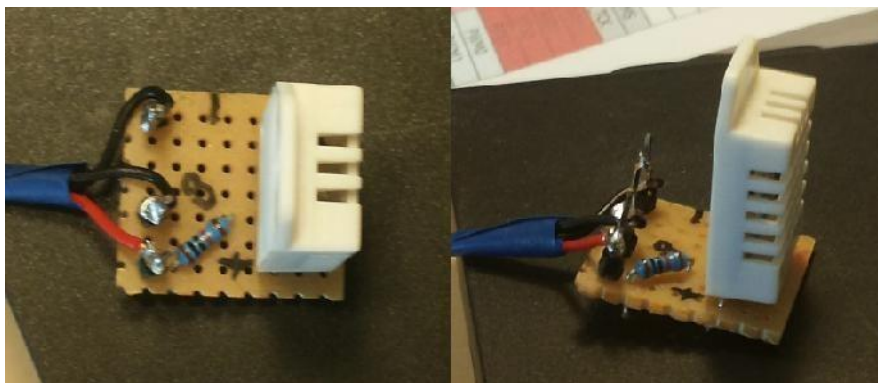
Breadboard circuit

The circuit is very simple. Just attach the leftmost sensor pin to the Pin 1 for 3.3V VCC, the second leftmost sensor pin to Pin 15 for data transmission and the rightmost sensor pin to a ground pin like Pin 20 of the Raspberry Pi (all in Board numbering):



Breadboard circuit for DHT22/Am2302

The assembled DHT22/AM.2302 sensor



The assembled circuit: temperature and humidity sensor dht22/AM2302

Install sensor drivers

```
$ git clone https://github.com/adafruit/Adafruit_Python_DHT
$ sudo apt-get install build-essential python-dev
$ cd Adafruit_Python_DHT
$ sudo python setup.py install
$ cd ..
```

Test the installation in python

Open python with

```
$ sudo python
```

and enter

```
import Adafruit_DHT
# the sensors output pin is connect to board pin number 15 # which is GPIO
# numbering pin 22.
Adafruit_DHT.read_retry(Adafruit_DHT.AM2302, '22')
```

If everything is fine, this outputs the current temperature and humidity with the first value being the humidity.

Update: Python 3

Execute the following commands for the Python3-version of the Adafruit driver:

```
$ git clone https://github.com/JoBergs/Adafruit_Python_DHT
$ sudo apt-get install build-essential python3-dev
$ cd Adafruit_Python_DHT
$ sudo python3 setup.py install
$ cd ..
```


Ex - 15 Detecting Movement

Problem

You want to trigger some action in Python when movement is detected.

Solution

Use a passive infrared (PIR) motion detector module. To make this recipe, you will need:

- Female-to-female jumper wires
- PIR motion detector module

Figure shows how the sensor module is wired. This module expects a power supply of 5V and has an output of 3.3V, making it ideal for use with a Raspberry Pi.

WARNING

Make sure that the PIR module you use has a 3.3V output. If it has 5V output, you will need to use a pair of resistors to reduce this to 3.3V

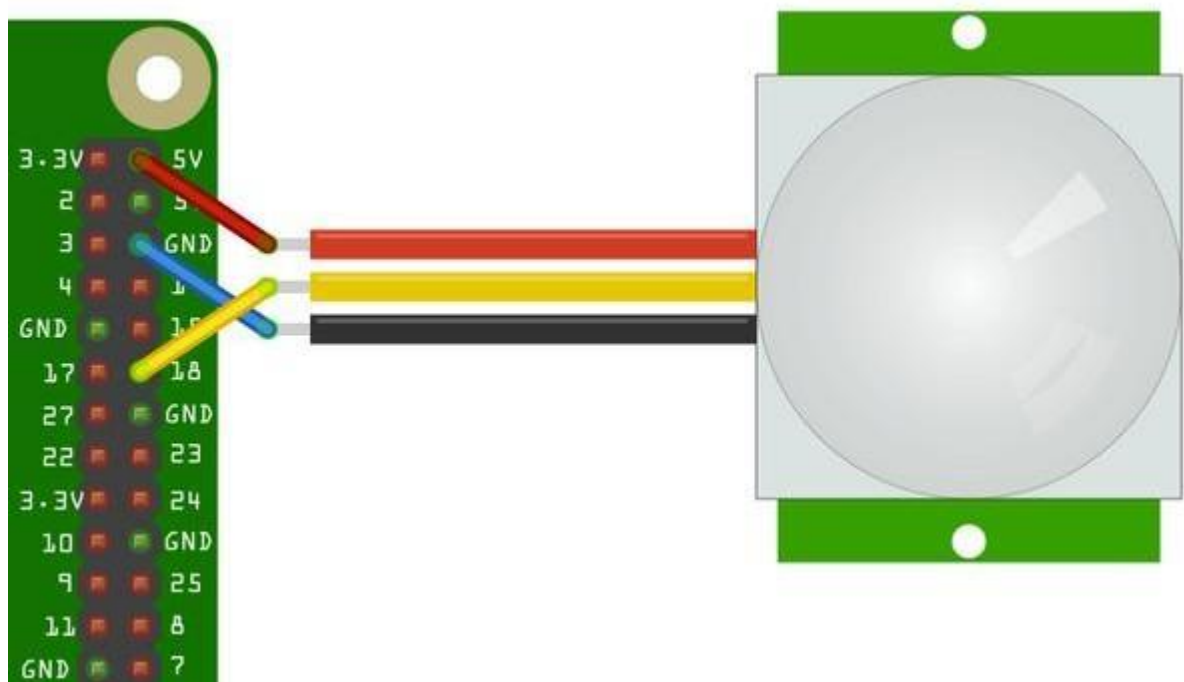


Figure. Wiring a PIR motion detector

Open an editor (nano or IDLE) and type in the following code
pir.py.

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

GPIO.setup(18, GPIO.IN)

while True:
    input_state = GPIO.input(18)
    if input_state == True:
        print('Motion Detected')
        time.sleep(1)
```

The program simply prints out the state of the GPIO input 18.

```
$ sudo python pir.py
Motion Detected Motion
Detected
```

Discussion

Once triggered, the output of the PIR sensor will stay high for a little while. You can adjust this using one of the trimpots on its circuit board. The second trimpot (if present) will set the threshold of light level that will disable the sensor. This is useful when the sensor is being used to control a light, turning it on when detecting movement, but only when its dark.

Ex - 16

Send Email from a Linux Terminal on the Raspberry Pi

Problem

You want to send an email message from a Linux terminal.

Solution

Linux has a package for the Simple Mail Transfer Protocol (SMTP) that you can use to send emails:

Discussion

Install the following two packages:

```
pi@raspberrypi ~ $ sudo apt-get install ssmtp mailutils
```

 Edit the following file:

```
pi@raspberrypi ~ $ sudo nano /etc/ssmtp/ssmtp.conf
```

 Set the mailhub:

```
mailhub=smtp.gmail.com:587
AuthUser=YourEmail@rajalakshmi.edu.in
AuthPass=YourPassword
UseSTARTTLS=Y
ES UseTLS=YES
```

Using Linux terminal type the following:

```
pi@raspberrypi ~ $ echo "Hello inbox" | mail -s "Test" bhuvangates@gmail.com
```

Ex – 17

Sending Email from Python

Problem

You want to send an email message from a Python program.

Solution

Python has a library for the Simple Mail Transfer Protocol (SMTP) that you can use to send emails:

```
pi@raspberrypi ~ $ nano myemail.py
import
smtplib
GMAIL_USER = 'your_name@gmail.com'
GMAIL_PASS = 'your_password'
SMTP_SERVER = 'smtp.gmail.com'
SMTP_PORT = 587
def send_email(recipient, subject, text):
    smtpserver = smtplib.SMTP(SMTP_SERVER, SMTP_PORT)
    smtpserver.ehlo()
    smtpserver.starttls()
    smtpserver.ehlo
    smtpserver.login(GMAIL_USER, GMAIL_PASS)
    header = 'To:' + recipient + '\n' + 'From: ' + GMAIL_USER
    header = header + '\n' + 'Subject:' + subject + '\n'
    msg = header + '\n' + text + '\n\n'
    smtpserver.sendmail(GMAIL_USER, recipient, msg)
    smtpserver.close()
send_email('destination_email_address', 'sub', 'this is text')
```

To use this example to send an email to an address of your choice, first change the variables GMAIL_USER and GMAIL_PASS to match your email credentials. If you are not using Gmail, then you will also need to change the values of the SMTP_SERVER and possibly SMTP_PORT.

You also need to change the destination email address in the last line.

Discussion

The send_email message simplifies the use of the smtplib library into a single function that you can reuse in your projects.

Being able to send emails from Python opens up all sorts of project opportunities. For example, you could use a sensor such as the PIR sensor to send an email when movement is detected.

Running python script: pi@raspberrypi ~ \$ python

myemail.py

Ex - 18

Assigning Static IP Address

Problem

This tutorial will show you how to set a static IP address on your Pi with the release of Raspbian Jessie. At the time of writing, the latest release date of Raspbian Jessie is 18-03-2016 with Kernel version 4.1.

Solution

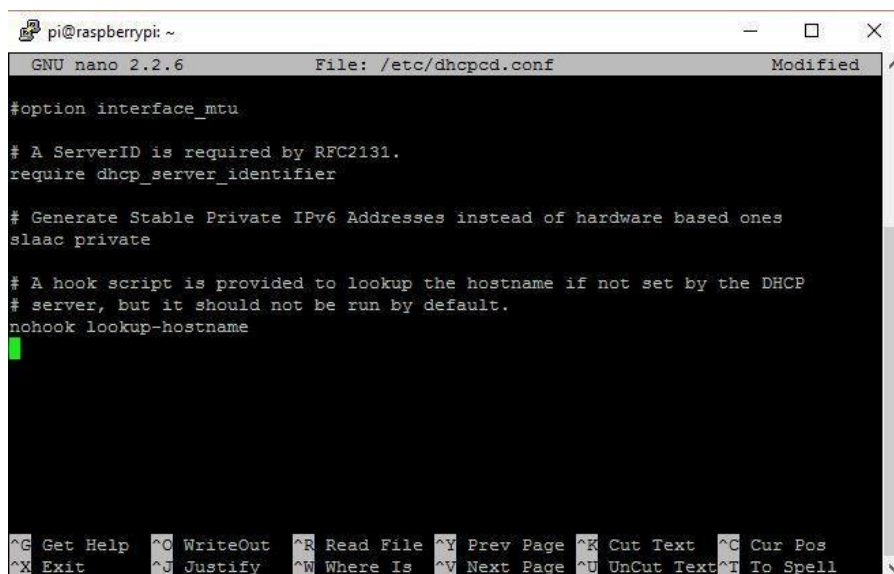
I recommend doing this on a fresh install, however if you have attempted to set a static IP address already, you may have found yourself editing the interfaces file (/etc/network/interfaces). I hope you made a backup, because you'll need to remove any edits you have made, and revert it back to its original state!

Discussion

The following is done over SSH, but you could just as well plug your Pi into a monitor, hook up a keyboard and mouse, and use the Terminal instead.

Start by editing the dhcpd.conf file `sudo nano`

`/etc/dhcpd.conf`



```
pi@raspberrypi: ~
GNU nano 2.2.6 File: /etc/dhcpd.conf Modified
#option interface_mtu

# A ServerID is required by RFC2131.
require dhcp_server_identifier

# Generate Stable Private IPv6 Addresses instead of hardware based ones
slaac private

# A hook script is provided to lookup the hostname if not set by the DHCP
# server, but it should not be run by default.
nohook lookup-hostname

^G Get Help ^C WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

Scroll all the way to the bottom of the file and add one, or both of the following snippets. Depending on whether you want to set a static IP address for a wired connection or a wireless connection `eth0` = wired, `wlan0` = wireless.

You'll need to edit the numbers in the snippet so they match your network configuration.

CODE

```
interface eth0
```

```
static ip_address=192.168.0.10/24 static
```

```
routers=192.168.0.1
```

```
static domain_name_servers=192.168.0.1 interface wlan0
```

```
static ip_address=192.168.0.200/24 static
```

```
routers=192.168.0.1
```

```
static domain_name_servers=192.168.0.1
```

interface = This defines which network interface you are setting the configuration for.

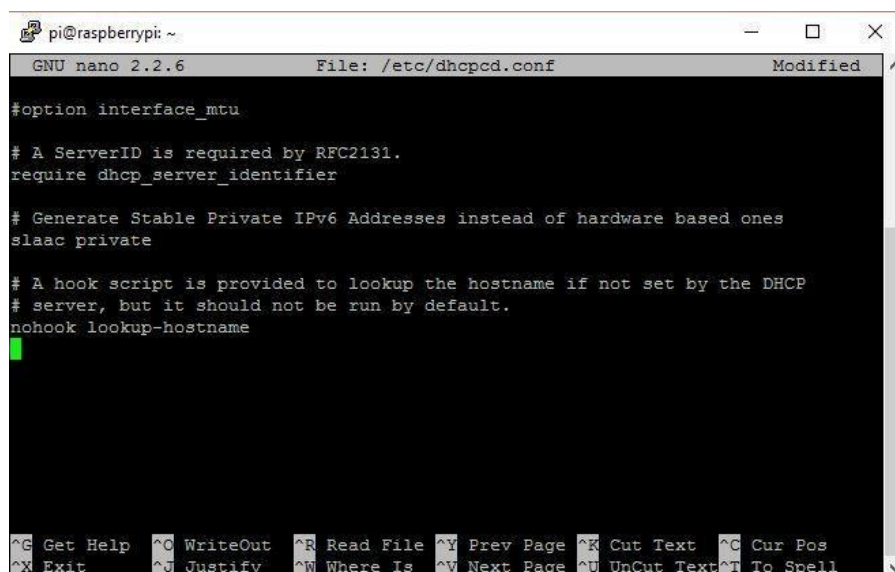
static ip_address = This is the IP address that you want to set your device to. (Make sure you leave the /24 at the end)

static routers = This is the IP address of your gateway (probably the IP address of your router)

static domain_name_servers = This is the IP address of your DNS (probably the IP address of your router). You can add multiple IP addresses here separated with a single space.

To exit the editor, press ctrl+x

To save your changes press the letter “Y” then hit enter



```
pi@raspberrypi ~  
GNU nano 2.2.6 File: /etc/dhcpd.conf Modified  
#option interface_mtu  
  
# A ServerID is required by RFC2131.  
require dhcp_server_identifier  
  
# Generate Stable Private IPv6 Addresses instead of hardware based ones  
slaac private  
  
# A hook script is provided to lookup the hostname if not set by the DHCP  
# server, but it should not be run by default.  
nohook lookup-hostname  
  
^G Get Help ^C WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos  
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

Now all you need to do is reboot, and everything should be set! reboot

You can double check by typing ifconfig

And checking the interfaces IP address

```
pi@raspberrypi ~  
Last login: Wed Apr 20 17:56:08 2016  
pi@raspberrypi:~$ sudo nano /etc/dhcpd.conf  
pi@raspberrypi:~$ sudo nano /etc/dhcpd.conf  
pi@raspberrypi:~$ ifconfig  
eth0      Link encap:Ethernet  HWaddr b8:27:eb:f3:63:7e  
          inet6 addr: fe80::214a:1361:51f:c4b4/64 Scope:Link  
          UP BROADCAST MULTICAST  MTU:1500  Metric:1  
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1000  
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)  
  
lo        Link encap:Local Loopback  
          inet addr: 127.0.0.1  Mask:255.0.0.0  
          inet6 addr: ::1/128 Scope:Host  
          UP LOOPBACK RUNNING  MTU:65536  Metric:1  
          RX packets:300 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:388 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1  
          RX bytes:33600 (32.0 KiB)  TX bytes:33600 (32.0 KiB)  
  
wlan0     Link encap:Ethernet  HWaddr d4:7b:b0:14:8a:ce  
          inet addr: 192.168.0.200  Bcast:192.168.0.255  Mask:255.255.255.0  
          inet6 addr: fe80::d47b:b0ff:fe14:88ce/64 Scope:Link  
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1  
          RX packets:1395 errors:0 dropped:430 overruns:0 frame:0  
          TX packets:390 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1000  
          RX bytes:290071 (280.2 KiB)  TX bytes:56694 (55.3 KiB)  
pi@raspberrypi:~$
```

Ex - 19

Running a Program while Booting

Problem

Having a program start by itself when you boot your Raspberry Pi can be very useful. There are a few ways of running a command at start - up, but in this tutorial we will create a script within `/etc/init.d` so that when the system boots the program will start/stop automatically on boot/shut-down.

Solution

For the purpose of this tutorial we will show you the method for an arbitrary program created in python 'example.py' with its location `/home/pi/example.py`. The following code will work for any script, just replace 'example' with name of your program and replace `/home/pi/example.py` to wherever you are actually keeping your script.

Discussion

First we need to create a script within `/etc/init.d/`. Type the following command in the terminal but replace 'example' with the name of your own program:

```
sudo nano /etc/init.d/example
```

This will open a text editor. Paste the following code into here and replace the name and location of the program with your own. The parts in black are the ones you'll need to change:

```
#!/bin/sh
# /etc/init.d/example

### BEGIN INIT INFO
# Provides:      example
# Required-Start: $remote_fs $syslog # Required-
Stop:  $remote_fs $syslog # Default-Start: 2 3 4 5
# Default-Stop:  0 1 6
# Short-Description: start a program from boot
# Description:   A simple script which will start a program from boot and stop upon shut-
down
### END INIT INFO
# Put any commands you always want to run here. case "$1" in
start)
echo "Starting example"
# run the program you want to start
```



```
/home/pi/example.py
;;
stop)
echo "Stopping example"
# end the program you want to stop killall
example.py
;;
*)
    echo "Usage: /etc/init.d/example {start|stop}" exit 1
    ;;
esac
```

Once you've copied the above code and replaced the names with that of your own exit and save with Ctrl+X.

Next you need to make the program executable:

```
sudo chmod +x /etc/init.d/example
```

Check the program is working correctly from etc/init.d/example by test starting it:

```
sudo /etc/init.d/example start
```

And the same again test stop the program:

```
sudo /etc/init.d/example stop
```

Next we need to register the program with the system so it knows to run/stop the program at boot/shutdown.

```
sudo update-rc.d example default
```

And that's it! Now you can reboot your Raspberry Pi and the program 'example' should start up automatically.

If you ever wish to stop the program running from boot type the following command:

```
sudo update-rc.d -f example remove
```