Wake up on LAN

with a Raspberry Pi

http://www.fnc.co.uk

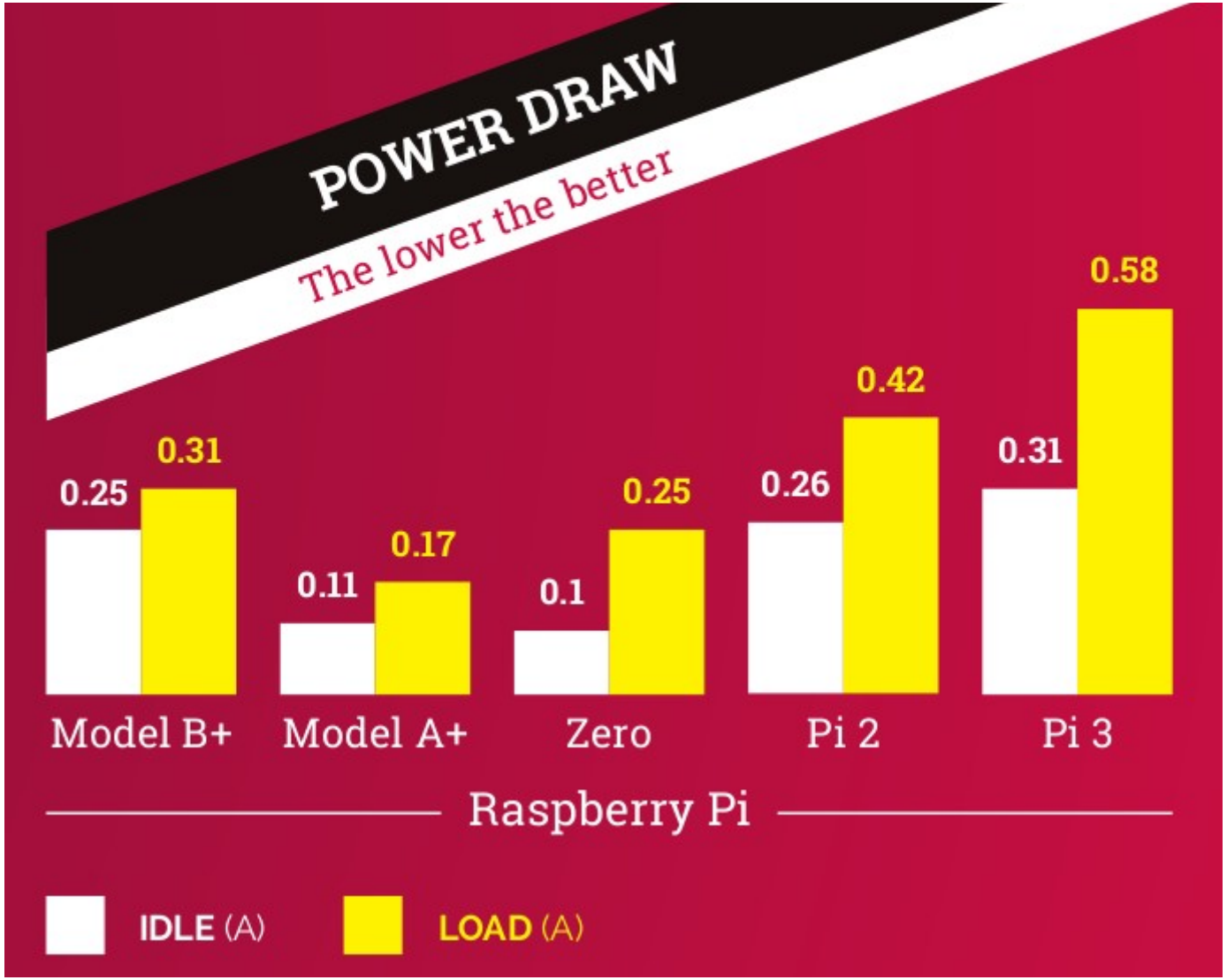
Raspberry Pi Jam
The Mitchell Library
23/09/2017

Motivation

- Several people need to connect to a remote Windows PC.
  - The PC does not need to be on all of the time.
- Need to deploy a low power wake-up-on-LAN service.
  - A Raspberry Pi is ideal for this, since it uses very little power.

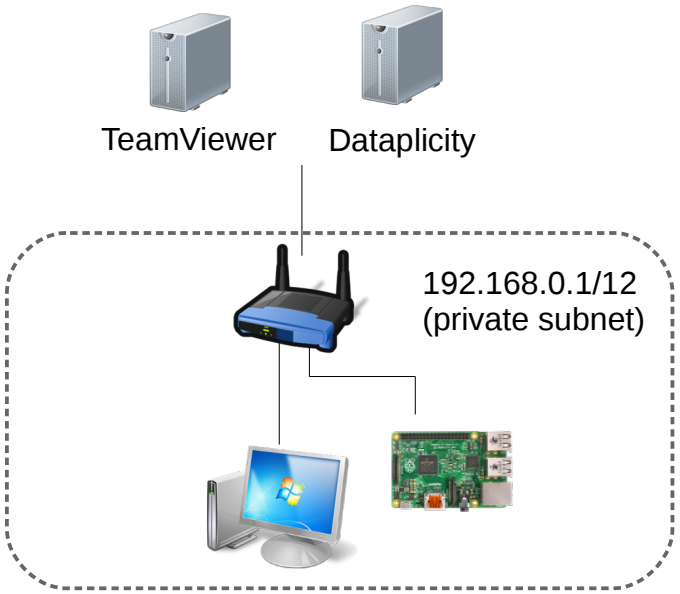
Electrical current

The MagPi



Raspberry Pi
Input voltage 5V
(P = I x V)

Network and tools

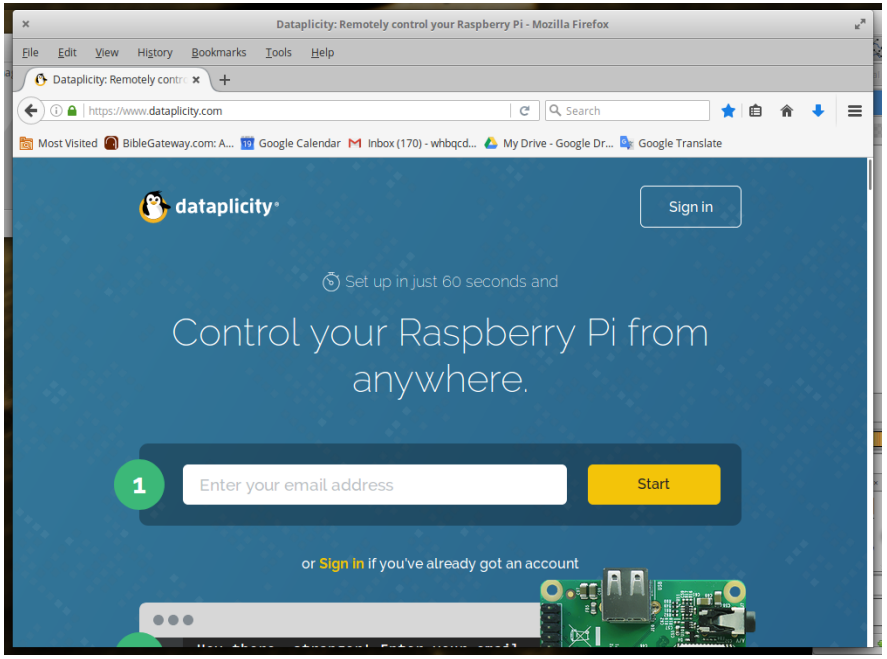


- Raspberry Pi on same private network as Windows PC.
- Configure Windows PC to start TeamViewer when it boots.
- Add Raspberry Pi to Dataplicity.
- Enable wake up on LAN on Windows PC.

Dataplicity

https://www.dataplicity.com/

- Dataplicity provides a console and can provide a wormhole to a Raspberry Pi web server.
  - Installation instructions on their web site are easy to follow.



Waking another PC

- Install etherwake on the Raspberry Pi.

```
apt-get install -y etherwake
```

- Then wake up PC, using the hardware (MAC) address of the Ethernet card.

```
etherwake -i eth0 13:10:54:3a:34:67
```

- After a few sections the Windows PC powers on and boots.

Waiting for the PC to boot

- Need to check if the PC has booted.
  - Windows firewall settings rejected ICMP packets from ping.
  - Windows PC is given a dynamic IP address.
- Solution:
  - Install arping, since Link Layer frames are not blocked by Windows firewall.
  - Use /proc/net/arp to find the IP address.
  - Set the sticky bit to allow etherwake and arping to run as root:

```
chmod u+s /usr/sbin/etherwake
chmod u+s /usr/sbin/arping
```

Wake Bash script (1/3)

```
#!/bin/bash

if [[ -z $1 ]]; then
  echo "usage $0 <mac address>"
  exit 1
fi

mac_address=$1
ip=""

get_ip_from_mac_address() {
  # Read the IP address from the ARP cache
  ip=$(cat /proc/net/arp | grep $mac_address | awk '{print$1}')

  if [[ -n $ip ]]; then  # An IP address has been found

    # The ARP record can be present, even though the machine is off.
    arping -c 1 $ip &> /dev/null

    # If the return code is 0, then the machine could be reached.
    ret_code=$?
    if [[ $ret_code != 0 ]]; then
      ip=""
    fi
  fi
}
```

Wake Bash script (2/3)

```
# Check if the machine is up and has an IP address
echo " >> Checking for PC with $mac_address on the local network."

# Try to get the IP address by calling the Bash function.
get_ip_from_mac_address

# Check if the ip variable is defined.
if [[ -n $ip ]]; then
  echo  " >> The PC with $mac_address is already up with IP $ip"
  exit 0  # Exit the script, returning success.
fi

echo " >> Waking up PC with hardware address $mac_address"
etherwake -i eth0 "$mac_address" # Now send the etherwake command

wait_count=300  # Set a maximum wait time of 5 minutes

# Echo without a linefeed to allow arping update dots to be printed.
echo -n " >> Waiting PC with hardware address $mac_address to boot "
```

Wake Bash script (3/3)

```
# Wait until the IP address comes up or the wait count reaches zero.
while [[ $wait_count > 0 ]]; do
  echo  -n "." # Inform the user that another ping step has occurred.

  get_ip_from_mac_address  # Try to get the IP address.

  if [[ -n $ip ]]; then  # If the IP address has been found.
    echo ""
    echo " >> PC with $mac_address is now up with IP address $ip"
    exit 0  # Exit the script, returning success.
  fi

  sleep 1  # Wait for a second.
  let wait_count--  # Decrement the wait count.
Done

# Print a warning message if needed.
if [[ $wait_count == 0 ]]; then
  echo " >> Could not wake up PC with hardware address $mac_address"
  exit 1  # Exit the script, returning an error state.
fi
```

Add an alias in ~dataplicity/.bashrc

```
alias wake_windows_pc="$HOME/bin/wake 18:23:45:2b:94:10"
```

Then start PC using alias `wake_windows_pc`