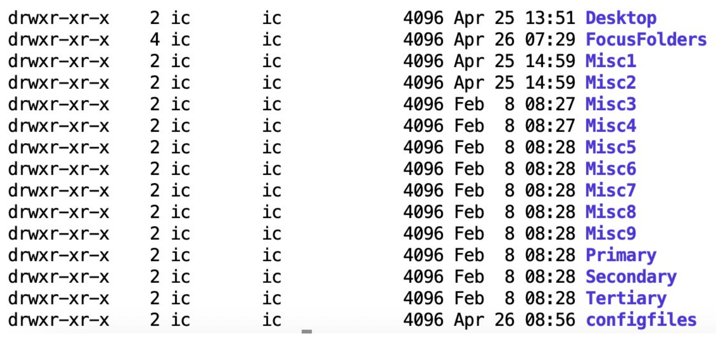
A Raspberry Pi Picture Frame

April 27, 2024

My goal was to create a picture frame which I could easily update from my computer. I tried a couple of the commercial picture frames and found them sorely lacking, so I thought, "Why not use a pi for this!" I decided that I wanted to be able to organize, filter, edit, rename and whatever else on my PC and then slurp the result over to the pi seamlessly. I decided to use rsync for this purpose as it is easy to implement on both the pi and the Mac. I'm very comfortable with this being driven by scripts from the command line since a command-line interface is where God intended us to be. :)

I therefore created the following directory structure on my computer and on the pi in a folder called ~/PictureFrame/PictureFrame_LR (since this unit lives in my Living Room):



```
drwxr-xr-x    2 ic       ic           4096 Apr 25 13:51 Desktop
drwxr-xr-x    4 ic       ic           4096 Apr 26 07:29 FocusFolders
drwxr-xr-x    2 ic       ic           4096 Apr 25 14:59 Misc1
drwxr-xr-x    2 ic       ic           4096 Apr 25 14:59 Misc2
drwxr-xr-x    2 ic       ic           4096 Feb  8 08:27 Misc3
drwxr-xr-x    2 ic       ic           4096 Feb  8 08:27 Misc4
drwxr-xr-x    2 ic       ic           4096 Feb  8 08:28 Misc5
drwxr-xr-x    2 ic       ic           4096 Feb  8 08:28 Misc6
drwxr-xr-x    2 ic       ic           4096 Feb  8 08:28 Misc7
drwxr-xr-x    2 ic       ic           4096 Feb  8 08:28 Misc8
drwxr-xr-x    2 ic       ic           4096 Feb  8 08:28 Misc9
drwxr-xr-x    2 ic       ic           4096 Feb  8 08:28 Primary
drwxr-xr-x    2 ic       ic           4096 Feb  8 08:28 Secondary
drwxr-xr-x    2 ic       ic           4096 Feb  8 08:28 Tertiary
drwxr-xr-x    2 ic       ic           4096 Apr 26 08:56 configfiles
```

This approach allows me to easily have multiple different PictureFrames, each with its own specific set of managed photos. For example, I could have PictureFrame_LR, PictureFrame_Office and PictureFrame_GuestRoom by simply defining the above directory structure for each and populating them appropriately.

Each of the above folders (except Desktop, FocusFolders and configfiles) is intended to have an arbitrary number of zero or more pictures in it. This allows me to display photos with a user-defined frequency.

"Primary" pictures are those which I want to see most often, while "Secondary" pictures are shown less often than "Primary" but potentially for longer than "Miscx" pictures. Finally, "Tertiary" pictures are those which I feel obligated to show, but choose to show only rarely.

Having nine "Misc" sets allows me to become more atomic in the presentation in the future if I choose to do so without having to rearrange everything. It also allows me to easily insert sets of pictures which are only relevant for a particular period of time, or for a particular audience.

There is just one file in Desktop which becomes the computer's desktop, and it is briefly shown whenever there is a transition from one set of pictures to the next.

The FocusFolders folder contains zero or more folders which in turn contain sets of photos which I want to temporarily "focus" the frame on displaying. This is done by adjusting the configuration file (doframe.cfg).

The configfiles folder contains the configuration files for the doframe.sh and BigBen.sh scripts (which are described later).

The block logic is as follows:

```
If there is a "Focus" folder specified, then show that folder only.
Otherwise perform:
    Sequence 1:
        Show Primary folder with specified timing
        Show Misc1 to Misc3 folder with specified timing
        Show Primary folder with specified timing
        Show Misc4 to Misc6 folder with specified timing
        Show Primary folder with specified timing
        Show Secondary folder with specified timing
        Show Misc7 to Misc9 folder with specified timing

    repeat Sequence 1 three times, then show the Tertiary folder (with specified timing),
then restart the entire process.
```

Note: In the following, all scripts and configuration files appear in parentheses - e.g. (doframe.sh) - and are listed by name at the end of this document. All script files are located in the home directory unless otherwise specified. Config files are in ~/PictureFrame/configfiles.

Process
I started by creating an Alpine image including XFCE using Craig's excellent tutorial (Thanks Craig!) which can be found here (IPv6 only):

www.makiki.ca/Pi/alpine_linux_installing_gui.html.

I manually rcp'd a few jpeg files onto the pi for testing and development and then created a script (doframe.sh) to implement the block logic. It uses feh [https://feh.finalrewind.org] as the display application. I loaded feh by executing:

```
    sudo apk update
    sudo apk add feh
```

With that working, I needed to get rsync going in order to populate the photo folders. On the pi I just had to:

```
    sudo apk add rsync
```

Fortunately, rsync is a standard app on MacOS, but it is available for Windows and Linux as well.

I then wrote the synchronization script (SyncUnit.sh) which lives on my Mac (but could easily live on a Linux box or - with some modification for batch or PowerShell - on a Windows box) to drive the synchronization process. Running this script will add and delete files on the pi so that the pictures and configuration files are identical on the Pi PictureFrame and on the source computer.

With this all working I decided to add a clock chime. I love the Westminster Chimes, so I found mp3 files for the quarter hour, half hour, three-quarter hour and hourly Westminster chimes. But I ALSO wanted to have the time announced, and the espeak voice just doesn't cut it for me. So, I wrote a script (maketime.sh) which runs on a Mac and uses the MacOS say utility to create AIFF formatted sound files for the times. However, I was using mocp on the pi, and it needs MP3 formatted sound files. So, I wrote a script (AIFFtoMP3.sh) to address that issue. Finally, I wanted to have the file names uniformly formatted to make scripting easier, so I wrote a script (fixtime.sh) to address that.

With the sound files in place I wrote a script (BigBen.sh) to drive the Westminster chimes and time announcement, and added a single line to the user's crontab (by calling cron -e):

```
*/15 * * * * /home/ic/BigBen.sh
```

I added support for configuration files to both doframe.sh and BigBen.sh and added them (doframe.cfg) and (BigBen.cfg) to the configfiles folder so that they could be easily maintained from the source computer. The pi could now be run without a keyboard or mouse (so, "armless"?).

I am using my Home Assistant configuration to control the power to the screen, so it is only on during certain hours, but the pi is always on.

To make the PictureFrame run automatically at boot, I first had to modify /etc/lightdm/lightdm.conf and add the line:

```
autologin-user=ic
```

in the [Seat:*] section to cause my user [which is ic] to be automatically logged in on boot. Then I created two files: one (zz-pictureframe.desktop) auto-runs the doframe.sh script and the other (xfce4-power-manager.desktop) removes the menu from the desktop. These both live in the ~/.config/autostart folder which xfce reads and executes when it starts up.

Finally, I decided that I wanted a way to reboot the PictureFrame without having to power it off or log into it, so I added a RESTART=YES parameter to the doframe.cfg file and then wrote a script (reload.sh) which processes that variable and uses sudo reboot to reboot the PictureFrame. I had to apk add sudo and create the file /etc/sudoers.d/ic with the single line entry:

```
            ic ALL=(root) NOPASSWD: /sbin/reboot
```

and added this line to the crontab using crontab -e to check once per minute for a reboot request:

```
            * * * * * /home/ic/reload.sh
```

I then reworked the SyncUnit.sh script to reset the RESTART=YES flag.

Et voila!

The future:
1) I want to add proximity sensing so that the screen will only be on when there is someone present in the room, timing out and turning off the screen after one hour of non-presence.
2) I want to add a Pi camera which will look out my front window and create a continuously updated webpage which I can browse from my computer. The chair in which I most often I sit faces into the room, and it's an understandably unacceptable burden to turn my head to see out the front window when I can have a Pi camera do it... :)
3) I want to run two monitors, each with an independent PictureFrame configuration on one Pi 4.

doframe.sh

```
#/bin/sh
# doframe.sh - create a Pi Picture Frame
#################################################################
### SET DIR to the appropriate PictureFrame directory on your pi ###
#################################################################
DIR=/home/ic/PictureFrame
#################################################################

CONFIGFILE=${DIR}/configfiles/doframe.cfg
FLAG=/tmp/doframe.flag

AWK=/usr/bin/awk
DATE=/bin/date
FEH="/usr/bin/feh -Z -x -F -Y -B "black" -q --on-last-slide quit"
GREP=/bin/grep

#Timeouts
PRIMARYTIMEOUT=`$GREP ^PRIMARYTIMEOUT $CONFIGFILE | $AWK
-f= '{print $2}'`
SECONDARYTIMEOUT=`$GREP ^SECONDARYTIMEOUT $CONFIGFILE |
$AWK -f= '{print $2}'`
TERTIARYTIMEOUT=`$GREP ^TERTIARYTIMEOUT $CONFIGFILE |
$AWK -f= '{print $2}'`
MISCTIMEOUT=`$GREP ^MISCTIMEOUT $CONFIGFILE | $AWK -f= '{print
$2}'`
FOCUSTIMEOUT=`$GREP ^FOCUSTIMEOUT $CONFIGFILE | $AWK -f=
'{print $2}'`
if [ .$PRIMARYTIMEOUT == . ]; then
        PRIMARYTIMEOUT=5
fi
if [ .$SECONDARYTIMEOUT == . ]; then
        SECONDARYTIMEOUT=5
fi
if [ .$TERTIARYTIMEOUT == . ]; then
        TERTIARYTIMEOUT=5
fi
if [ .$MISCTIMEOUT == . ]; then
        MISCTIMEOUT=5
fi
if [ .$FOCUSTIMEOUT == . ]; then
        FOCUSTIMEOUT=10
fi

D=`$DATE +"%Y%m%d.%H%M%S"`
echo $D Program Start -=-=-=- > $FLAG
echo CONFIGFILE=$CONFIGFILE >> $FLAG

#ITERATION is how many times the entire set has been displayed
ITERATION=1
#REPEAT is how many completed sets of 50 iterations have been displayed
REPEAT=0

FOCUS=`$GREP ^FOCUSFOLDER= $CONFIGFILE | $AWK -F= '{print $2}'`
if [ .$FOCUS = . ]; then
  FOCUSFOLDER=NULL
  echo $D Focus Folder is currently UNDEFINED >> $FLAG
else
  FOCUSFOLDER=$DIR/FocusFolders/$FOCUS
  echo $D Focus Folder is currently defined as: $FOCUS >> $FLAG
fi
if [ ! -d $FOCUSFOLDER ]; then
  echo Primary >> $FLAG
  $FEH -D $PRIMARYTIMEOUT $DIR/Primary 2>> $FLAG
fi

# When TERT=3 then Tertiary set is shown
  TERT=0

while :
do
  D=`$DATE +"%Y%m%d.%H%M%S"`
  FOCUS=`$GREP ^FOCUSFOLDER= $CONFIGFILE | $AWK -F=
'{print $2}'`
  if [ .$FOCUS = . ]; then
    FOCUSFOLDER=NULL
  else
    FOCUSFOLDER=$DIR/FocusFolders/$FOCUS
    echo $D Focus Folder is currently defined as: $FOCUS >> $FLAG
  fi
  if [ -d $FOCUSFOLDER ]; then
    echo FocusFolder $FOCUSFOLDER selected >> $FLAG
    $FEH -D $FOCUSTIMEOUT $FOCUSFOLDER 2>> $FLAG
  else

    for x in $(seq 1 3); do echo Misc$x >> $FLAG;$FEH -D
$MISCTIMEOUT $DIR/Misc$x; done
    echo Primary >> $FLAG
    $FEH -D $MISCTIMEOUT $DIR/Primary 2>> $FLAG

    for x in $(seq 4 6); do echo Misc$x >> $FLAG;$FEH -D
$MISCTIMEOUT $DIR/Misc$x; done
    echo Primary >> $FLAG
    $FEH -D $PRIMARYTIMEOUT $DIR/Primary 2>> $FLAG
    echo Secondary >> $FLAG
    $FEH -D $SECONDARYTIMEOUT $DIR/Secondary 2>> $FLAG

    for x in $(seq 7 9); do echo Misc$x >> $FLAG;$FEH -D
$MISCTIMEOUT $DIR/Misc$x; done
    echo Primary >> $FLAG
    $FEH -D $PRIMARYTIMEOUT $DIR/Primary 2>> $FLAG

    D=`$DATE +"%Y%m%d.%H%M%S"`
    echo $D Iteration $ITERATION.$REPEAT >> $FLAG
    ITERATION=$((ITERATION+1))
    TERT=$((TERT+1))

    if [ $TERT = 3 ]; then
       echo Tertiary >> $FLAG
       $FEH -D $TERTIARYTIMEOUT $DIR/Tertiary 2>> $FLAG
       TERT=0
    fi
fi
D=`$DATE +"%Y%m%d.%H%M%S"`
echo $D Iteration $ITERATION >> $Flag
if [ $ITERATION -gt 49 ]; then
       ITERATION=1
       REPEAT=$((REPEAT+1))
       echo $D $REPEAT iterations complete -=-=- > $FLAG
fi
done
```

doframe.cfg

```
##########################################################################
## THESE ARE THE PARAMETERS FOR doframe.sh                              ##
## All timeouts are expressed in seconds                                ##
## Include RESTART=YES as a parameter to force the PictureFrame to reboot ##
## Include RESTART=EXIT as a parameter to force doframe to exit         ##
##########################################################################
PRIMARYTIMEOUT=10
SECONDARYTIMEOUT=5
TERTIARYTIMEOUT=5
MISCTIMEOUT=5
#FOCUSFOLDER=MattChristmas
FOCUSTIMEOUT=10
#RESTART=YES
```

Times are in seconds.

The FOCUSFOLDER named here must exist in ~/PictureFrame/FocusFolders

zz-pictureframe.desktop

```
[Desktop Entry]
Hidden=false
Version=1.0
Type=Application
Name=IC Picture Frame
TryExec=/home/ic/doframe.sh
Exec=/home/ic/doframe.sh
```

xfce4-power-manager.desktop

```
[Desktop Entry]
Hidden=true
```

These live in ~/.config/autostart

SyncUnit.sh

Note that this file MUST live on the source computer, NOT on the pi PictureFrame

```
#!/bin/bash
UNIT=PictureFrame_$1
WHICHHOST=$2

THEDIR=/Users/ic/Data/Sync/SyncPictures/##PictureFrame/$UNIT
CONFIGFILE=$THEDIR/configfiles/doframe.cfg

GREP=/usr/bin/grep
MV=/bin/mv

if [ .$WHICHHOST == . ]; then
        WHICHHOST=ic@pictureframe.local
fi

cd $THEDIR 2> /dev/null

cat << EOF

                ***** SyncUnit.sh *****
    SyncUnit.sh is used to synchronize a PictureFrame from this PC.
    It assumes that you have the proper directory structure which
    will be reflected onto the PictureFrame unit which you specify.
                ***********************

EOF

case $PWD in
  ${THEDIR})
        echo Synchronizing the PictureFrame unit called $1 with the files on this computer
in the directory
        echo $THEDIR
        rsync -azP --delete . ${WHICHHOST}:~/PictureFrame/
     # Reset RESTART=YES flag if set in doframe.cfg
     $GREP -v -i ^RESTART=YES $CONFIGFILE > ${CONFIGFILE}.tmp
     $MV ${CONFIGFILE}.tmp $CONFIGFILE
     echo "#RESTART=YES" >> $CONFIGFILE
        ;;
  *)    CAT << EOF
   ERROR:  $THEDIR  does not exist

   Usage:  SyncUnit UnitID Who@WhichHost
   Where:  UnitID is the suffix of the PictureFrame_ directory to use for this synchronization
        Who@WhichHost is the user name followed by the @ symbol followed by the
           name or IP address of the picture frame to be synchronized.

EOF

esac
```

reload.sh

```
#!/bin/ash

# This will force doframe to exit or PictureFrame to reboot
based on the existence of a flag in the doframe.cfg file or
EXIT parameter

AWK=/usr/bin/awk
DATE=/bin/date
GREP=/bin/grep
KILL=/bin/kill
MV=/bin/mv
PS=/bin/ps
REBOOT=/sbin/reboot
SUDO=/usr/bin/sudo

CONFIGFILE=/home/ic/PictureFrame/configfiles/
doframe.cfg

D=`$DATE +"%Y%m%d.%H%M%S"`

HUPFLAG=`$GREP -i ^RESTART=YES $CONFIGFILE`
EXITFLAG=`$GREP -i ^RESTART=EXIT $CONFIGFILE`
if [ .$1 = .EXIT ]; then
        EXITFLAG=RESTART=EXIT
fi

$GREP -v -i ^RESTART= $CONFIGFILE > $
{CONFIGFILE}.tmp
$MV ${CONFIGFILE}.tmp $CONFIGFILE

if [ .$EXITFLAG = .RESTART=EXIT ]; then
        THEPID=`$PS ax | $GREP doframe.sh | $GREP -v
grep | $AWK '{print $1}'`
        THEOTHERPID=`$PS ax | $GREP /usr/bin/feh |
$GREP -v grep | $AWK '{print $1}'`
        $KILL -9 $THEPID $THEOTHERPID
        echo $D Exiting doframe.sh
        exit
fi

if [ .$HUPFLAG = . ]; then
        echo $D Reload=No >> /tmp/doframe.flag
        exit
fi

echo $D Rebooting this PictureFrame >> /tmp/doframe.flag
$SUDO $REBOOT
```

maketime.sh

```
#!/bin/sh
# This must be run on a Mac
# This creates a complete set of time files in timefiles which can then be
moved to the Pi and used with mocp.

for x in  0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 ;
do
  for y in  hundred-hours 15 30 45 ; do
        echo "The time is $x $y"
        say -v Daniel "The time is $x $y" -o timefiles/TimeIs$x$y.aiff
  done
done
```

AIFFtoMP3.sh

```
#!/bin/sh

for x in  0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 ;
do
  for y in  hundred-hours 15 30 45 ; do
        echo "The time is $x $y"
        lame -m m timefiles/TimeIs$x$y.aiff timefiles/TimeIs$x$y.mp3
  done
done
```

fixtime.sh

```
#/bin/ash

for x in 0 1 2 3 4 5 6 7 8 9;
do
  mv TimeIs${x}hundred-hours.mp3 TimeIs0${x}hundred-hours.mp3
  mv TimeIs${x}15.mp3 TimeIs0${x}15.mp3
  mv TimeIs${x}30.mp3 TimeIs0${x}30.mp3
  mv TimeIs${x}45.mp3 TimeIs0${x}45.mp3
done
```

maketar.sh

```
#!/bin/ash

DATESTAMP=`/bin/date +"%Y%m%d.%H%M%S"`
cd ~/
/bin/tar --exclude *.tgz --exclude *.jpg --exclude *.JPG --exclude *.HEIF --exclude */.cache  -cvpzf ~/Dev/PictureFrame_and_BigBen.$
{DATESTAMP}.tgz .
echo Created ~/Dev/PictureFrame_and_BigBen.${DATESTAMP}.tgz
```

```
#!/bin/ash
###################
#BigBen.sh  - play Westminster Chimes (Palace of Westminster,
London)
#Robert Taylor 20181024 - Revised for Alpine Pictureframe 20240206
#use mocp --help for details on "Chime" items
#Driven by $CFGDIR/BigBen.cfg file containing these entries
#PLAY=:QUARTER:HALF:THREEQUARTER:HOUR:    <- When to
play chimes
#ChimeVol=20                              <- Volume to play chimes
(mocp -v)
#VoiceVol=175                             <- Volume to play voice
(mocp -v)
#ACTIVE=:06:07:08:09:10:11:12:13:14:15:16:17:18:19:20:  <- Active
hours
#CHIME=[YES|NO]                           <- Provide chimes during
active hours
#SPEAK=[YES|NO]                           <- Announce time during
active hours
#MUTE=[YES|NO]
###################

echo RUNNING BigBen.sh

DIR=/home/ic
CFGDIR=$DIR/PictureFrame/configfiles

AWK=/usr/bin/awk
DATE=/bin/date
MOCP=/usr/bin/mocp
GREP=/bin/grep
SLEEP=/bin/sleep

if [ `$GREP -i ^MUTE= $CFGDIR/BigBen.cfg | $GREP -i YES` ]; then
        echo BigBen is muted.  Exiting.
        exit
fi

# Start the mocp server (it will just error out if it's already running)
$MOCP --server

THEDATE=`$DATE +"%Y%m%d.%H%M"`
THEHOUR=`$DATE +"%H"`
THEMINUTE=`$DATE +"%M"`

if [ "$1" = "" ]; then
        MIN=`$DATE +":%M:"`
else
        MIN=:$1:
        THEMINUTE=$1
fi

if [ "$2" = "" ]; then
        HOUR=`$DATE +":%H:"`
else
        HOUR=:$2:
        THEHOUR=$2
fi

TIME=$THEHOUR:$THEMINUTE
```

BigBen.sh

```
if [ "$MIN" = ":00:" ]; then
        THEMINUTE=hundred-hours
        TIME=`$DATE +"%H hundred hours"`
fi

echo Hour=[$HOUR]  Min=$MIN  TheHour=[$THEHOUR]
TheMinute=$THEMINUTE

if [ $MIN = ":15:" ];then
        WHICH=QUARTER
        DELAY=9
elif [ $MIN = ":30:" ]; then
        WHICH=HALF
        DELAY=12
elif [ $MIN = ":45:" ]; then
        WHICH=THREEQUARTER
        DELAY=17
elif [ $MIN = ":00:" ]; then
        WHICH=HOUR
        DELAY=19
fi

if [ "$WHICH" = "" ]; then
        echo Nothing to do.  Exiting.
        exit
fi

CHIME=`$GREP -i ^CHIME= $CFGDIR/BigBen.cfg | $AWK -F= '{print
$2}'`
SPEAK=`$GREP -i ^SPEAK= $CFGDIR/BigBen.cfg | $AWK -F= '{print
$2}'`

echo WHICH=$WHICH  CHIME=$CHIME   SPEAK=$SPEAK
$GREP -i ^ACTIVE= $CFGDIR/BigBen.cfg

if [ `$GREP -i ^ACTIVE= $CFGDIR/BigBen.cfg | $GREP $HOUR` ]; then
        echo CHIMING
        if [ `$GREP -i ^PLAY= $CFGDIR/BigBen.cfg | $GREP :$WHICH:`
]; then
          CHIMEVOL=`$GREP -i ^CHIMEVOL $CFGDIR/BigBen.cfg |
$AWK -F= '{print $2}'`
          if [ "$CHIMEVOL" = "" ]; then
              CHIMEVOL=30
          fi
echo Chime=$CHIME ChimeVol=$CHIMEVOL
          if [ "$CHIME" = "YES" ]; then
            $MOCP -v $CHIMEVOL -l $DIR/BigBenFiles/
BigBen_$WHICH.mp3
          $SLEEP $DELAY
          fi
          VOICEVOL=`$GREP -i ^VOICEVOL $CFGDIR/BigBen.cfg |
$AWK -F= '{print $2}'`
echo It is $TIME
          if [ "$SPEAK" = "YES" ]; then
              $MOCP -v $VOICEVOL -l $DIR/BigBenFiles/TimeIs$
{THEHOUR}${THEMINUTE}.mp3
          fi
        fi
fi
echo $THEDATE "BigBen.sh run" > $DIR/BigBen.lastrun
```

BigBen.cfg

```
PLAY=:QUARTER:HALF:THREEQUARTER:HOUR:
ChimeVol=20
VoiceVol=175
ACTIVE=:06:07:08:09:10:11:12:13:14:15:16:17:18:19:20:
CHIME=YES
SPEAK=YES
MUTE=NO
```

PLAY specifies which of the four possible chimes will be played
ChimeVol specifies the volume of the chimes if played
VoiceVol specifies the volume of the voiced time if used
ACTIVE specifies the hours (in 24 hour format) when sound will be made
CHIME specifies if Chimes will be played
SPEAK specifies if the time will be announced
MUTE allows for a silencing of the system without making other changes.

This is an additional utility which I created at the last minute.

whatismyip.start

This causes the Pi to announce its name, purpose and IP address at boot. This lives in /etc/local.d You have to execute rc-update add local default once this file is in place.

```
#!/bin/ash

MYNAME="SD_B, like BRAVO"
MYCREATION="2024 02 08"
MYPURPOSE="Picture Frame."


AWK=/usr/bin/awk
ESPEAK=/usr/bin/espeak
GREP=/bin/grep
IFCONFIG=/sbin/ifconfig
SLEEP=/bin/sleep

MYIP=`$IFCONFIG wlan0 | $GREP "inet addr:" | $AWK -F: '{print $2}' | $AWK '{print $1}'`
$ESPEAK -a 200 -g 15 "My network is now available so listen up!"
$SLEEP 1
$ESPEAK -a 200 -g 15 "My name is $MYNAME"
$SLEEP 1
$ESPEAK -a 200 -g 15 "My purpose in life is to be a $MYPURPOSE"
$SLEEP 1
$ESPEAK -a 200 -g 15 "I know it's not much, but it's all I have"
$SLEEP 1
$ESPEAK -a 200 -g 15 "I connect to the network on WLAN 0"
$SLEEP 1
$ESPEAK --punct -g 20 -a 200 "My IP address is $MYIP"
$SLEEP 2
$ESPEAK --punct -g 20 -a 200  "Again $MYIP"
$SLEEP 1
$ESPEAK -a 200 -g 15 "I will now carry on with being the best $MYPURPOSE I can be"
```

https://thepihut.com/blogs/raspberry-pi-tutorials/running-two-monitors-with-a-raspberry-pi-4

https://retropie.org.uk/forum/topic/30856/solved-using-feh-to-display-image-on-second-monitor-runcommand_onstart/3?lang=en-US

https://retropie.org.uk/forum/topic/30856/solved-using-feh-to-display-image-on-second-monitor-runcommand_onstart/2?lang=en-US

https://gist.github.com/okanon/d8469d76079782501b09c67e8d5ee04b (WiFi Config)