

Digital Signage for Churches and other Establishments

System documentation

An easy to use Digital Signage system for use in a Church or other facility.

Author: Gert Bakker, Coleshill, UK
File: Digital_Signage_System_Documentation.odt
Date: 16 February 2024
Issue: 1

Table of contents

Introduction.....	2
Copyright and licence.....	3
How it works.....	4
Development environment.....	5
The Digital Signage project.....	6
How the presentation works.....	7
How the System Administration works.....	9
How the emails work.....	10
Specifications.....	11
Files and Folders.....	12
Email commands.....	20
Appendix A – Presentation methods.....	22
Appendix B – Bill of materials.....	24
Appendix C – Log messages.....	26
Appendix D – Installation instructions.....	27

Index of Tables

Table 1 - Terminology.....	2
Table 2 - Email commands for Presentations.....	20
Table 3 – Email commands for System Management.....	21
Table 4 - Hardware Bill of Materials.....	24
Table 5 - Software Bill of Materials.....	25
Table 6 - Hardware purchases.....	27
Table 7 - Optional hardware items.....	27
Table 8 - Summary of the installation steps.....	28
Table 9- Scripts to be run by CRON.....	34

Introduction

This document describes a **Digital Signage** facility for a Church or other establishment where this may be of use.

Definition

A **Digital Signage** facility is one that displays a slide presentation on one or more screens in a facility. The presentation can be created using various methods and is managed using emails. It is designed to be simple to set-up, maintain and create presentations for.

Target audience

This is a technical document and meant for those wanting to install and use the software for this project. This document needs to be read together with the "User Manual".

Terminology

The following are the meaning and/or explanation of the words or phrases used in this document.

Word / Expression	Meaning and explanation
Device or Installation	The set of equipment and software that makes up the entire digital signage facility, including the display devices.
System	The processor (computer), the operating system (O/S) and the programs and scripts written specifically for this project.
dotenv	This is a special file that contains the secrets such as the email account name and password. The file name is by default ".env".
Application	Software programs, shell scripts, etc. that perform a specific task required for this project.
CRON	Deamon in the O/S that executes scheduled commands.
BASH	The Bourne Again SHell is the command-line interpreter that can execute single commands or scripts containing a set of commands.
Folder	A directory or folder in the file system of the device

Table 1 - Terminology

Project Goals

1. Keep the overall cost to an absolute minimum by utilising low cost equipment, free software, free email services, etc.
2. Keep maintaining and updating the system and the presentations as simple as possible.
3. Use internet and email to deliver and manage presentations and/or updates to the system.
4. System Administrator to receive a daily status message with any errors encountered by the system. This allows problems to be resolved easily.
5. Provide a level of security whereby only authorised email addresses can send instructions to the system.

Copyright and licence

Copyright

The Digital Signage project is Copyright © 2023, Gert Bakker, Coleshill, UK

GNU licence

This document is part of the <Digital Signage> project.

Digital Signage is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, version 3.

Digital Signage is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <<https://www.gnu.org/licenses/>>.

All other copyright and trademarks mentioned in this document and all files making up this Digital Signage project are acknowledged.

Originator email

Gert Bakker <gmjbakker@yahoo.co.uk>

How it works

The following is a brief explanation on how the “**Digital Signage**” works. More detailed information is further in this document.

1. Almost all interactions with the installation are done using emailed instructions called **commands**. These commands instruct the system what presentation to show and what system administration tasks to perform.
2. As the main purpose of this system is to display a presentation, the system can show them from different sources. With this issue, presentations generated using “Google Slides”, “LibreOffice Impress” and “Microsoft PowerPoint” have been implemented. A discussion on the pro and cons of various presentation methods are in Appendix A.
3. A presentation starts, restarts or stops (a) automatically after the system boots up, (b) when an email request is received or (c) via timed CRON events. All this is dependent on the actual installation.
4. Presentations are *transient*, that is, they are forgotten once another presentation is started. However, there is a facility to save the last presentation and to select it again using emailed commands.
5. Emails are retrieved at regular intervals. The system will process the emails and decides what actions need to be taken. It then executes these actions.
6. System Administration tasks are performed on a daily basis and a status message emailed to the administrator. Most admin tasks can be done via email messages. Only occasionally a System Administrator will need direct access to the system.
7. Timing of presentations, system admin tasks, etc. are installation dependent. These can be done using auto-start at boot-up, CRON entries or by sending an email.

Development environment

This chapter describes the development and target devices and software that was used to write and test this Digital Signage project.

Development device.

This project was developed using a laptop running Linux Mint. The laptop is a Lenovo ideapad 310 with 12Gb memory and 1Tb disk space. The Linux Mint version is 21.3 (Virginia) and it is using Cinnamon as the desktop environment

All scripts, Python programs and some of the documentation were written using VSCodium version 1.85.2. The extensions that are used on VSCodium are "Python" and "Shellman".

Python version 3.10.12 was used for testing the Python programs. Bash 5.1.16 was used as the Linux Mint Shell for all shell script testing.

Copying the software to the target device uses Rsync. This copies only the files required and those that have changed since the last copy run.

Target device

The target computing device for running the Digital Signage project is a Raspberry Pi 4 Model B Rev 1.5 with 2Gb memory and 128Gb storage on a µSD card. The Raspberry Pi uses Raspbian GNU/Linux 11 (bullseye) with Kernel: 6.1.21-v8+. BASH version 5.1.4 is used to run the scripts and Python version 3.9.2 runs the Python programs.

Screen used with the Raspberry Pi is an Acer HDMI monitor. A Logitech USB keyboard and mouse were used when required.

ENERGENiE power sockets are used to turn the Digital Signage screen on or off as required. The interface card is designed for the Raspberry Pi and works up to a respectable distance.

Some of the programs and scripts have code that detects the device it is running on and runs the code that works on that device only. This is in particular for the "html2text" utility and the "chromium" web browser. These bits of code may need to be changed when testing and running the project on a different O/S for the development and target devices.

The DOTENV Python module is used to store the secret information. See the DOTENV documentation for more information.

CRON is used to run timed events and AUTOSTART files are used for automatic starting of the presentations and email facilities.

Rsync from the development device will have copied the files on to the target device and into a working folder. Any changes and testing takes place in the working folder. When ready for production, a Makefile is used to install the software into the production folder.

The Digital Signage project

The Digital Signage project consists of four parts; (1) hardware, (2) presentation, (3) email control and (4) System Administration.

(1) Hardware

The hardware consists of the processor, the power source and the display screens. In addition there may be remote controlled power sockets to turn screens on and off. The hardware bill of materials used during the development is in appendix B. The actual hardware used is implementation dependent.

(2) Presentation

The main function of the installation is to present a set of slides to one or more screens. To accomplish this, various presentation methods have been considered. Appendix A has the list of the presentation methods that were considered for this implementation with their advantages and disadvantages.

After trials and considering the advantages and disadvantages, "Google Slides" was seen as the preferred method with "PowerPoint" and "LibreOffice Impress" as secondary ones. These three are used in this implementation.

(3) Email control

The control and management of the presentations and other functions of the installation are done using email as much as practicable.

This simplifies the operation of the system and will not require anyone to physically operate the system. Of course there will be occasions that this is required, but that will need to be done by a System Administrator who understands the system and the software.

A specific email address is used to address the installation and this is used to receive and send emails by the installation. The received emails contain requests (called **commands**) to be performed. File attachments can be part of the email and these are processed as specified by the command.

For security, only authorized email addresses can send instructions to the system. A list of authorized email addresses is maintained by the installation. Emails that were not authorised are saved into a separate folder and reported in the daily status message that is emailed to the System Administrator.

The email command format is kept as simple as possible. A single line of text is used for the action to perform of which the first word is the "**command**" and the rest of the line is the data that the command requires. There can be more than one action lines in an email, although this is not recommended.

(4) System administration

This is kept to a minimum as it involves knowledge of the processor, it's operating system and the programs, scripts and instructions making up this project.

To assist with the system administration, a daily status email is send to provide management data and if any problems were encountered. Any remedial action can then be taken by the System Administrator.

How the presentation works

The presentation is controlled by a single BASH shell script called: **start_presentation.sh**. This is the only script that starts and stops a presentation. This script can be called by other scripts and will run the presentation in background mode (using the & in Linux). The script can also be executed at boot-up so that the presentation starts when the system starts up.

The presentation controlling files

The presentation is controlled using three files, called "**trigger**" files. These files manage what presentation to start, restart or stop. The files are in a format that can be sourced-in into a BASH script and sets specific shell script variables. [See BASH documentation on how variables and sourced-in scripts work.](#)

The three presentation trigger files are:

running_now	This file holds the presentation that is running now or that is going to run. It is created from the running_next file.
running_next	This file holds the info for the next presentation to start or stop. Its presence makes a change to the presentation. It is created from a command in an email, but other scripts or programs can generate this file too.
running_last	This file holds the previous presentation. It is used when a "stop" command has been requested to restore the running_now file.

The trigger files contain two BASH variables that indicate the command to execute and the data that is associated with it. The variables are as follows and they are "exported" so Python programs can use them also.

RUN_CMND	This is the command to execute. These commands are usually from emails send to the system. More explanation of the commands are later in this chapter.
RUN_DATA	This variable holds the presentation data that the command requires. This can be the URL of the presentation or the name of the presentation file to run. Some commands do not require additional data.

The shell script uses the trigger files as follows:

- First the currently running presentation is stopped. This is done regardless of the contents of the trigger files.
- If the running_next file is present, the running_now file is copied to the running_last file and the running_next file is moved to the running_now file. The running_next file therefore is deleted. This is intentional. The new command to execute is now known.
- If the running_next file is not present, the running_now file is left as is. This ensures that when this script is started, the running_now file is used again.
- The running_now file is sourced-in so that the script that manages the presentation receives the two BASH variables and can perform the requested action.

The presentation is started in background and the script terminates.

The presentation commands

The commands that are used to manage the presentation are the following:

Google	Run a Google Slides presentation. The data provided is the URL for the presentation. This information is normally supplied via email.
PowerPoint	Run a PowerPoint slide presentation (.ppsx) using LibreOffice Impress as the program to run.
Impress	Run a LibreOffice Impress presentation using the Impress program. This presentation must have the "autostart" feature set up. See the User Manual for more info on this feature.
Stop Halt	Terminate the current presentation. Because this command is in the running_now file, copy the running_last file to the running_now file. That way a restart request will run the last presentation.

More commands may be added in the future, depending on what presentation format or service is going to be used.

The command can be in any letter-case as it will be converted to lower-case before being used by the script.

How the System Administration works

The System Administration works the same as the presentation control. The emailed commands generate a trigger file called "**manage_next**" and this starts the system admin script which will then execute the commands given.

The trigger file "manage_next" has three variables in it and these are as follows. Just as for the presentation control, the variables are defined with "export" in front of them.

RUN_CMND	This is the command to execute. These commands are emailed to the system. More explanation of the commands implemented is later in this document.
RUN_DATA	This has the additional information required by the command. See later in this document for what data is to be expected.
RUN_ATCH	The name of the attachment file that has the new script, program or other data. As a precaution the file is converted to LINUX format using the "dos2unix" utility before it is used.

The following commands are implemented for this project.

crontab	Update the CRONTAB with the contents of the attached file.
delete	Delete files from the "permanent" folder.
download	Download any production file and send as an attachment to the System Administrator. If the data is "crontab", the current CRONTAB is emailed as an attachment.
list	List contents of one or more folders. Add the folder name(s) on the command line separated by space(s).
reboot	This will reboot the system.
save_now	Save the current presentation (running_now) to the "permanent" folder using the data supplied with the command as the filename for the saved presentation. If the trigger file points to an attachment, this is also saved to the folder.
screen	Turn an ENERGENiE power socket that controls a screen on or off.
shutdown	Do a proper shutdown of the system.
update	Update scripts and programs by sending the replacement file as an attachment. The command line has the name of the existing file while the attachment can be any name.

The trigger file is sourced-in and the command executed.

See the chapter on the email commands for more information on what data is expected.

How the emails work

The presentation and system administration are all done using email messages send to the device by *authorized* email senders.

It works as follows:

1. Emails are created and send to the Digital Signage system containing commands in the body of the message and with any required attachments.
2. A Python program, running in background, checks at regular intervals if there are any messages in the INBOX. If so, the program will read and save the messages and any attachments. It then starts the script to processes the contents of the retrieved emails. When this script finishes, the program continues to check for messages. This Python program is started and stopped by CRON entries and at boot-up.
3. The scrip that was started by the email checking and reading program, runs a Python program that will read each saved message and finds out what the commands are. These commands then generate trigger files for doing system administration and/or presentations. All messages are processed before doing the System Administration tasks and the presentations.
4. If a System Administrator trigger file is present, the script that processes the System Administrator command will now perform these tasks.
5. If a Presentation trigger file is present the script that starts a presentation is run.
6. Error and Information messages generated by the scripts and the Python programs are written to the common log file.
7. To aid in the proper running of the Digital Signage system, a status email is sent to the System Administrator on a daily basis. The information in this email are the messages from the common log file, the status of the system, etc. The System Administrator can then check that everything is OK or that System Admin tasks need to be performed.

Specifications

The following are the specifications that were used for the target device of this project. The actual implementation may be different as the situation demands.

Hardware

The device used for the main processor is a "*Raspberry Pi* ®" version 4 with 4 cores and 2Gb RAM.

Disk storage is a 128Gb µSD card. Minimum for a Raspberry Pi is 8Gb disk storage.

The system requires internet access since commands, presentations and files are emailed to the system and as some presentations are played from an internet based source (Google Slides for example).

Existing screen(s) and video facilities are used to display the presentation. Audio is not intended to be used, but may be a consideration in the future.

Email

An email address is required for the installation so that instructions can be emailed to it. An Outlook email address was used during the development of this project.

Software

The operating system used during development is Raspbian version 11 (Bullseye).

Additional utilities and Python modules were required. These need to be installed separately as they do not come as standard with the version of O/S.

The Python programs, BASH shell scripts and other required files that have been developed for this project are copied into a working folder. Changes to the configuration files can now be made. When ready the programs, scripts and files are then installed in the production folder. The production folder is where everything is run from. That way any changes are made to the working folder files and tested before being committed to production.

"CRON" is used for timed events such as starting and stopping of a presentation, emails retrieval and sending and other admin tasks of the installation.

Startup, as defined for the LXDE desktop environment, is used for automatic start of the current presentation after a boot or restart of the system. Other auto startup settings may be required for, e.g. the "unclutter" utility, etc.

Files and Folders

The files and folders making up this project are listed below with a brief explanation on their usage. More information can be found in the comments in the scripts, configuration and program files.

Folders

./	Home folder of the project
./attachments	Email attachments
./messages	Email message text files
./other_msg	Unauthorized email messages
./permanent	Saved presentations
./queue	Email message waiting to be send
./unqueue	Email messages that have been sent

Python programs

check_read_email.py	Check for email messages and save them
process_messages.py	Process the email messages and its contents
send_email.py	Send email messages with attachments if needed
sysadmin_save_now.py	Save the running now info to permanent folder
turn_on_off.py	Turn the ENERGENiE power socket on or off

Configuration and sourced-in files

.env	The dotenv file with the secret info for emailing
logging.func	Logging function for use in scripts
send_email.func	Functions for sending an email
signage.conf	Main configuration constants

BASH Shell scripts

check_read_email.sh	Start the check_read_email.py program
daily_status.sh	Send daily status email to the System Administrator
issue_command.sh	Issue a presentation command
kill_check_read_email.sh	Terminate the check_read_email.py program
process_messages.sh	Process the saved email messages
send_email.sh	Send email messages directly or from a queue
start_presentation.sh	Main script to start the presentation
sysadmin_download.sh	Email a file by attaching it to an email
sysadmin_list.sh	Process a System Admin command to list folders
sysadmin_tasks.sh	Main script to process the System Admin tasks
sysadmin_update.sh	Process a System Admin command to update a file

Other data files

manage_next	System Admin command to be performed
pi_signage.log	File with the "debug", "info" and "error" messages
running_last	Presentation control file
running_now	Presentation control file
running_next	Presentation control file

More detailed description of the above folders and files is in the next chapter.

Note! Some names of the files and folders are configurable in the main configuration file. Make sure that the Makefile is changed to suit the installation.

Folders

The following explains each folder and what they are used for.

./

This is the home folder where the programs, scripts and data files are stored and where the other folders are relative to.

During development this is: `~/Documents/pi_signage`

When in full production this is: `~/Production/pi_signage`

This may need to be changed to suit the installation.

./attachments

This is where the attachments from an email message are stored. Each attachment filename has the format: **msgNN_filename** where:

msgNN The same as the filename of the email message.
 NN is a unique message counter.

filename The filename of the attachment in full.

The attachment filenames are cleaned-up by removing all non-alphanumeric characters and replace them with an underscore (_).

The attachments are processed depending on the instruction in the corresponding email. The files are deleted after a configurable number of days if kept in this folder. Each time an attachment is used for a presentation and is in this folder, the modified date/time is reset to when it was used. That way only aged attachments are deleted after a configurable number of days.

./messages

This is where the email messages are stored before their contents is processed. Each email gets a unique number which is added to the filename of the email message. The format is "**msgNN**", where "**msg**" is a prefix and **NN** is the message counter. This counter is incremented for each email message and resets to 1 after it reaches 9990. The files in this folder are processed and then deleted.

./queue

./unqueue

These two are used for sending emails. The 'queue' folder has the email messages waiting to be send and the 'unqueue' folder has the message files that have been emailed. Messages are put in the queue folder before the Python program sends the message. If a message failed to be send, they remain in the queue folder and will be retried next time the send script runs. The messages in the 'unqueue' folder are removed after a configurable number of days. This allows for diagnostics when a message failed and may need checking.

./other_msg

Any emails that were not from an authorized email address are stored in this folder. Also, when a message could not be sent, it is placed here too. The whole email message is stored, so it will need a separate reader to extract the message text and any attachments. A python program is planned for these emails to be read and any attachments saved. *More details to be worked out as this will need direct access to the system itself. This is a TODO!*

./permanent

This folder holds all the 'permanent' presentations. This is where slide shows that are to be shown regularly are stored, including attachment files. Each file is a copy of the `running_now` trigger file. The name of the file must be in lowercase and no spaces. Use underscore (`_`) instead of a space. See also the command to save the current presentation to this folder.

Configuration and common function files

This project relies on configuration and function definition files to allow changeable constants and common functions. The configuration files allow for adapting this project to the intended installation. **Note** that the Python programs use default values if it cannot find an environment variable, which happens when the program is not run in a BASH shell that sourced-in the common configuration files.

More detailed information is given in the "man" (manual) pages for each of the following files. These man pages are available in the "manpage" folder in the working directory.

./env

This is the "**dotenv**" file used by all the Python programs and the shell function to log messages. It contains the *secret* information. As the filename starts with a 'dot', it is not displayed when listing the folder, hence the name "dotenv".

The data stored in this file is as follows:

DEBUG='False'

DEBUG flag. Set to 'True' to write debug messages to the logging file. During development and, if required during production, the debug logging level messages can be useful to help in identifying problems. The DEBUG logging level depends on this flag being "True", or "False". The DEBUG flag can be set or reset by an email request.

USERNAME='user@email.com'

Email address for this installation. Used for logging into the email server to retrieve or send email messages.

PASSWORD='password'

This is the password to use with the username to logon to the email server. This is preferably the "apps" password that avoids two factor authentication. See the relevant email website on how to generate these apps passwords.

IMAP_SERVER='imap-mail.outlook.com'

The email server address for reading the emails, called an IMAP server. This needs to be set for whichever email server the username is registered to. For IMAP and SMTP server info, [check on this website](#).

IMAP_PORT=993

The port number to use for the IMAP server connection.

SMTP_SERVER='smtp-mail.outlook.com'

The email server address for sending the emails, called an SMTP server. This needs to be set for whichever email server the username is registered to.

SMTP_PORT=587

The port address to use for connecting to the SMTP server of the email system to be accessed.

EMAIL_AUTH='anyname@example.com,anothername@example.com'

This is the list of email addresses that are allowed to send instructions to the installation. If the sender is not in this list, the email will be ignored but saved into the "other_msg" folder.

MESG_COUNT='1'

This is the counter that gets added to the filename prefix of the stored email messages. It is incremented for each email read and is reset to 1 after the counter reaches 9990.

./signage.conf

This file has the common configurable constants for all the scripts and programs of this project. It is sourced-in by all BASH scripts. Because the Python programs are using these constants also, they have to be defined with **export** to ensure the Python program can use them. During testing of a Python program, if the common constants are not in the environment when it is executing, it uses a default value. *For more information about all the configurable constants, see the contents of this file.*

./logging.func

This file defines the BASH version of the Python "logging" module. All the scripts use this to write messages to the common log file. *See ./pi_signage.log file explanation below.*

./send_email.func

This file defines the BASH shell functions for queuing an email message or to send an email immediately. It formats the message using the RFC 5322 specification before being send.

Data files

The following are the trigger and other data files used by this project. Some of these are sourced-in by Bash scripts and Python programs while others are for storing the logging messages, etc.

./running_last

./running_next

./running_now

These are the trigger files that control the operation of the presentation. In each file are two variables that are sourced-in by the script that starts a presentation or any script that uses the data. The two variables are:

export **RUN_CMND**=

This is the command that starts or stops a presentation. See the explanation of the commands for further information.

export **RUN_DATA**=

The data to be used by the presentation. This is either a URL or a filename. See later in this document what data is expected.

The "running_now" file is the currently running presentation. The "running_next" file holds the next presentation data. When encountered by the "start presentation" script, the "running_now" file contents is saved to the

"running_last" file and then it overwrites the "running_now" file and the new presentation gets started.

./manage_next

This is the trigger file to perform System Management tasks. It is used in the same way as the "Running" files mentioned above.

The trigger file contains three variables which are:

export **RUN_CMND=**

Same usage as for the "running" files above. The possible commands are discussed later in this document.

export **RUN_DATA=**

This contains the data that was given on the command line. What data is required is explained for each RUN_CMND later in this document.

export **RUN_ATCH=**

This is the name as the file that was attached to the email sending the management update. The file is located in the attachments folder.

./pi_signage.log

This is the file that gets the logging messages from the Python programs and shell scripts. If the DEBUG flag is True, then debug messages are also written to this file. Normally only INFO and ERROR messages are written to it. See the Python "logging" module for more info about this feature.

This log file is emailed to the System Administrator on a daily basis to show what error and info messages were logged. It is then cleared for the next day's log messages.

Python programs

The following are the programs, written in Python, used in this project. For more information, see the comments and "docstrings" in the Python program. Use the "pydoc3" utility to display the comments of any program.

./check_read_email.py

This Python program runs in background and at regular intervals checks if there are any email messages. If any found, it reads and saves the messages and any attachments. It then executes the script to process the email messages and that script starts the SysAdmin tasks and the presentation. Upon termination of the script, this program continues the email checking process. This program is terminated with the "Interrupt" signal (SIGINT) by a separate script when required.

The retrieval of email messages and any attachments is as follows:

Each message is stored in the "messages" folder using the filename format:

msgNN.extn

Where:

msg	Filename prefix
NN	Counter value as per the "MESG_COUNT"
.extn	The extension .txt or .html

The file extension is based on the MIME format indicator that was found in the email message. The message file name (without the extension) is also used as

the prefix for the attachments. That way it is known which attachment belongs to which message. Attachments are stored in the "attachments" folder.

The read messages are deleted from the server. Any messages that are not from an authorised sender are stored in a separate folder (called "other_msg"). These will need to be read manually and will be deleted after a configurable number of days.

./process_messages.py

This Python program processes the email messages that are stored in the "messages" folder. It will read each message and create a trigger file depending on the command found. The attachment filename is found if required. Messages that have been processed are deleted when the program finishes. This program will process all email messages before terminating.

./send_email.py

This program sends one or more emails using RFC 5322 formatted messages. Messages can have attachments. When the message has been sent, it is moved to the "unqueue" folder where it will be deleted after a configurable number of days. Messages that do not conform to the correct format or have incorrect email addresses are copied to the "other_msg" folder.

./turn_on_off.py

This program, which is optional, will control ENERGENiE Power sockets. This is used to turn screen(s) on or off. Arguments passed are the socket number and to set the socket ON or OFF. The controller in the Raspberry Pi can control up to 4 power sockets and 0 means all sockets.

./sysadmin_save_now.py

This program saves the running_now trigger file to the permanent folder with a filename given by the email command. It will also save the attachment to the permanent file if the running_now file indicates this.

BASH shell scripts

This section describes the BASH shell scripts. The scripts are grouped by their function. For more information, see the comments in the scripts and the "man" pages created for each script.

Scripts managing a presentation.

./start_presentation.sh

This BASH script is the only script that starts and/or stops a presentation. It is executed after the system booted up and the X-Windows interface has started. In order for the network to be up and running, there is a special argument that will cause the script to sleep for 10 seconds. This should be enough time for the network to be available. This script is also run by other scripts and by CRON timed entries. *This script is not intended to be run from the command line.*

./issue_command.sh

This script executes **commands** passed to it as arguments. This script is intended to be used from the command line, by other scripts and/or by CRON entries. It provides a means of issuing commands to the system when using the

command line when accessing the system directly. For a list of possible commands, use the argument "help" from the command line.

Scripts dealing with the email messages

./check_read_email.sh

This script starts the Python program, running in background, that checks if there are any emails waiting. If this Python program is already active, this script terminates with exit code 80. Therefore it can be used to restart the Python program if, for any reason, it was no longer active.

This script is intended to be run after the system boots up and at specific times of the day via a CRON entry. By using regular running of this script it ensures that the Python program is kept running for the hours it is supposed to. See the example of CRON entries later in this document.

./process_messages.sh

This script runs the python program that processes the commands in the saved messages and then executes other scripts to action those commands. *This script is executed by the "check_read_email.py" program only!*

After processing all emails, if there is a "manage_next" trigger file, it runs the script that performs the System Administrator tasks. Then, if there is a "running_next" trigger file, it runs the script to start a presentation.

./send_email.sh

Script to send email messages. Email message files are either found in the "queue" folder or are passed as argument(s) to this script. This allows the sending of emails that are started by a timing process (such as CRON) or to send it immediately.

Email messages are text files that use the RFC 5322 format. The To and Subject headers are required. The Attachment header is required only when attaching a file to the email. If any of these headers are missing or incorrect, the message is ignored. It is moved to the "other_msg" folder.

Successfully send email messages are moved to the "unqueue" folder and are deleted from there after a number of days (configurable). This provides a facility in which sent messages can be checked.

./kill_check_email.sh

This script terminates the email checking program in a proper way. It is intended to be used by CRON in order to terminate the email checking process.

Scripts for System Administration tasks.

./daily_status.sh

This script does the daily cleaning up and email the System Administrator with the current system status. First it deletes any files that are over a certain number of days old. Then it collects some health information. It extracts the syslog entries of interest (if the script to do this is installed) and then adds the log file contents. After the message has been emailed, the logging message file is cleared and a message left in it to show that the daily status was completed. That way the log file is restarted each day.

./sysadmin_tasks.sh

This script processes the System Administration tasks that have been requested via email messages.

./sysadmin_update.sh

Script to perform the System administration **update** request where a file can be overwritten by a new version.

./sysadmin_list.sh

Script to perform the System Administrator **list** request. This results in an email to be sent to the System Administrator with the list of files in the folders requested in the email.

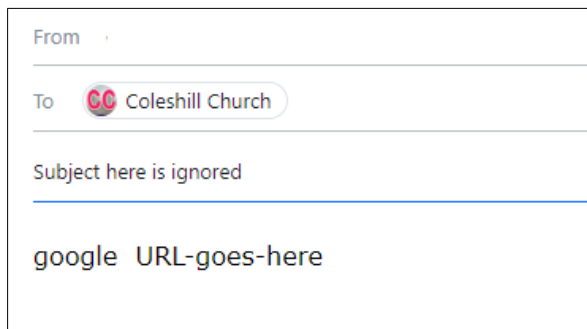
./sysadmin_download.sh

Script to perform the System Administrator **download** request. This will email any file in the production and subfolders to the System Administrator email address. There is a special command that emails the contents of the CRON table by sending the email command "download crontab".

Email commands

This chapter explains the commands that can be emailed to the Digital Signage system. Each command and it's associated data are given on a **single** line in the emailed message.

This is an example of an emailed command to start a Google Slides presentation:



The screenshot shows an email interface with the following fields:

- From:** (empty)
- To:** Coleshill Church (with a red circular icon containing 'CC')
- Subject:** here is ignored
- Body:** google URL-goes-here

"URL-goes-here" is the web link provided by the Google Slides presentation.

There can be more than one command line in a message and there can be multiple emails, but only the last command that created a trigger file will be the one that is acted upon.

Commands can be in any case.

Presentation commands

The following are the commands that affect a presentation.

Command	Data to be supplied	Explanation
google	URL of the Google Slides	Displays a presentation created using Google Slides.
powerpoint	No data, but attach the .ppsx PowerPoint slide show file.	PowerPoint slide show that starts automatically. File must be 20Mb or less as that is the maximum an email can send.
impress	No data, but attach the .odp presentation with the ImpressRunner autostart enabled	LibreOffice Impress presentation that with ImpressRunner will start automatically. Maximum file size is determined by the email system used and is generally 20Mb.
restart start	No data	Restarts or starts the last presentation. For example if changes to the currently running Google slides have been made, and it is still has the same URL.
run	Name of the file in the permanent folder.	Start a running file stored in the "permanent" folder.
save_now	Name of the file to store the data in.	Save the running_now file to the permanent folder where it will be available for running again.
stop halt	No data	Stops the current presentation.

Table 2 - Email commands for Presentations

System Management commands

The following are the commands to do System Management for the Digital Signage installation.

Command	Data to be supplied	Explanation
crontab	None. Add the new crontab file as an attachment	Replace the CRONTAB with a new one given as the attachment. This allows easy updating of the CRON settings.
debug	On or Off	Sets the debug flag to off or on. This will result in DEBUG messages being written to the log file.
delete	Name of file(s) to delete	This command deletes files in the "permanent" folder. Only give the filename. More than one filename can be given.
download	Name of file to be emailed as an attachment	Emails the file as an attachment to the System Administrator email address. Data given must be an existing file with folder name relative to the production folder
email_auth	List of authorized email addresses	Replaces the list of authorized email addresses that can send emails to control the system.
list	folder name(s)	Sends an email with the list of files in the folder(s) given in the command line.
reboot	No data	Reboots the system and starts the last presentation.
save_now	Name of the file to store the data in.	Save the running_now file to the permanent folder where it will be available for running again.
screen	Screen no (0-4) , on or off	To set the ENERGENiE power socket on or off. Give the socket number and on/off.
shutdown	No data	Shutdown the system completely. Use with care.
update	Name of the file to be updated. New file added as attachment	Update scripts and python programs, etc. with a new version. New file is attached.

Table 3 – Email commands for System Management

Appendix A – Presentation methods

This appendix list the presentation methods that were considered for this project together with their advantages and disadvantages. Use this information to consider which method will be preferred and what alternatives are also possible.

Of course this chapter can be expanded as and when other methods have been considered.

Google Slides

This is a free service for anyone with a Google account. Slides can be easily produced and when ready for publishing, giving the system the URL of the exported slides is enough.

Advantages

- Google Slides is free and is done using a web browser when signed in with a Google account.
- The slide show is easily emailed to the installation by exporting it to “web” and sending the URL generated by Google Slides in an email to the system.
- The Slides URL is unique for the presentation and does not change when updating the presentation. It can be saved for use at different times.
- No need to send the sideshow itself to the installation, reducing local storage requirements on the target device.
- Easily used by anyone familiar with PowerPoint.

Disadvantages

- The functionality of Google Slides is not as good as for PowerPoint. However, this may not be a problem as slides need to be simple and effective, not showy.

PowerPoint

Microsoft PowerPoint produces very sophisticated slide shows. It has many more features than Google Slides. This method requires that the presentation is attached to the email starting the presentation.

Advantages

- Powerful and feature rich presentations are possible.
- Very common solution and anyone with a bit of MS Office experience can create a slide presentation.

Disadvantages

- In order for the presentation to be emailed to the system, the maximum file size has to be 20Mb or less.
- PowerPoint presentation must be saved as an auto-start slide show so that no manual intervention is required to start the show.
- The Linux LibreOffice Impress software can display PowerPoint presentations and can also auto-start it, but it does not have all the capabilities of PowerPoint and some functionality may not work properly.

LibreOffice Impress

LibreOffice is free “Office” software and runs on Linux, Windows and Apple MAC devices. It has a full set of office functions including a PowerPoint equivalent called “Impress”.

This software can also run PowerPoint slides and export presentations to PowerPoint format.

Advantages

- Free software and runs on many operating systems.
- Can be developed on a Windows based machine that has LibreOffice installed.
- As easy to use as PowerPoint.
- Can only be auto started by using the LibreOffice extension "[ImpressRunner](#)". This provides the ability to auto-start the presentation. It is required that this extension is installed on all devices that are producing Impress presentations and on the actual device itself.

Disadvantages

- Emailing the presentation will require the file to be less than 20Mb.
- Presentation management limited to person who created the Impress presentation. More difficult to share with other creators. Similar to PowerPoint presentations.
- The presentation works slightly different on a Raspberry Pi and on Linux or Windows devices. It may need some experimenting with creating the presentation on a Raspberry Pi to test it out.
- During testing the Impress presentation stopped and caused the whole system to be unresponsive. This may be resolved in future upgrades.

PiPresents

PiPresents is a Raspberry Pi based digital signage system primarily aimed at museums and exhibitions. It is a Python based implementation. As can be seen from the list of "Advantages" and "Disadvantages", this solution is under consideration, but will require a lot of learning.

Advantages

- Sophisticated facilities for many types of presentations, including interactive and animation using the GPIO pins on the Raspberry Pi.
- Free to use

Disadvantages

- Requires an earlier version of the Raspbian operating system. This may cause some instabilities with other software, such as LibreOffice.
- Requires a steep learning curve and is not as easy to manage as other methods.
- Emailing the presentation will involve a lot of files. These can be delivered with multiple emails, but will need a level of expertise that may be prohibitive.
- Testing the latest version called "Beep" there were a number of errors and it failed to operate on the device used for this project.

Considering the above disadvantages, this software package will need a lot more investigation and testing to ensure that it works and is easy to use and update.

Other web based services

There are numerous web based digital signage creators and sellers. Most of these companies charge for the service. Some provide a free service but for a limited number of screens. However, as the main purpose of this project was to do everything at lowest cost, these solutions are less attractive.

Appendix B – Bill of materials

This appendix lists the hardware and software items that were used in the development of this project.

Hardware

Item	Specification and other information
Processor	Make: Raspberry Pi version 4 (4 cores and 2GB RAM)
Disk storage	µSD card with 128Gb of storage.
Screen	Any decent screen with HDMI, good size and resolution.
Keyboard and Mouse	A standard wireless keyboard and mouse is available, but is not required for the normal running of the system.
Power	Power is supplied by a standard Raspberry Pi power supply with USB-C connector.
Mounting	Mounting of the system is to be decided at final installation.
HDMI cable	An HDMI adaptor cable is required to connect the Raspberry Pi micro HDMI to normal HDMI connector.
ENERGENiE Power socket (Optional, only if required)	A Raspberry Pi controlled mains power switch that turns a socket on or off. This is intended to be used to turn screens off when not needed or when they need to be blanked. See also the Screen Blanking section below. Model purchased from Amazon UK: "ENERGENiE ENER002-2PI remote control sockets with Raspberry Pi controller board – white"

Table 4 - Hardware Bill of Materials

Software

The following table shows the software and Python modules that are required for this installation to work.

Software item	Specification and other information
Operating System	Raspbian Version 11 – Bullseye (or higher). This is the O/S that runs the system.
LibreOffice	The open Office software. Normally comes with the version of Raspberry O/S. If not, install this set of Office programs.
ImpressRunner	This is a LibreOffice extension and must be installed. This can be installed from the ImpressRunner website.
unclutter	Utility to remove the mouse pointer on the screen. Timing is set to remove the mouse pointer after 10 seconds of non-movement. Used to remove the mouse pointer during the display of the slides.
dotenv	Python module to process the data in a ".env" file. This file contains the email information and credentials and some other configuration data that needs to be kept secret. Used by all the Python programs.
logging	Python module to log messages to a file.
html2text	Utility to translate HTML file to text. Used in the message processing phase to extract the text from an HTML encoded email message.
syslogquick.sh	Shell script to extract the syslog messages of interest. Together with the configuration file: ~/bin/src/syslogcheck.conf

Software item	Specification and other information
dos2unix	Utility to convert DOS/MAC text files to UNIX (LINUX) end-of-line format. Used with the "update" and "crontab" commands to ensure the file emailed has the proper text format for UNIX/LINUX operating systems.
Scripts and programs	These are the scripts, programs, configuration files, folders, etc. as defined in the chapter "Files and folders" above.
ENERGENiE	Python module for ENERGENiE Power socket control if this is going to be used for this implementation.

Table 5 - Software Bill of Materials

Screen blanking

There is a problem with the Raspberry Pi model 4 initially, in that the setting for the display overlay module does not allow the usual vcgenclmd to turn the screen off and on. However by changing the /boot/config.txt file as shown below, the vcgenclmd works again.

Change the /boot/config.txt file

The following needs to be set in the /boot/config.txt file:

```
# Enable DRM VC4 V3D driver
dtoverlay=vc4-fkms-v3d
max_framebuffers=2
```

Screen blanking commands

Issue the following commands to turn the screen off and on:

```
vcgenclmd display_power 0 # Turns off screen
vcgenclmd display_power 1 # Turns on screen
```

This now works.

NOTE. It turns out that the screens currently in use do not go blank but display a screen saver set of pictures. This is unacceptable in a church environment. To get the screens to blank it is therefore necessary to turn the screens off. In our implementation this is done with ENERGENiE power sockets.

Appendix C – Log messages

The Python programs and BASH scripts all generate messages that are written to the main log file. See the Python module "[logging](#)" for more information.

To aid translation of the error and information messages into other languages, the texts for these have been put in a list which is defined at the start of the program. Debug messages are not put into a language list as they are only used during testing.

Error and Information messages

Normally there are two levels of log messages: Error and Information messages.

Error messages are generated when an error has been detected. The System Administrator upon receiving the log messages in the daily status email may need to take action. See the programs and scripts for what messages are possible.

Information messages are generated when there is a need to log an action. The System Administrator only needs to be aware of them so that it knows what has been going on in the system.

The log messages are emailed to the System Administrator every night and the log file is cleared for the next day of activities.

Log messages generated by the Python programs have a prefix to the message that indicates which program generated them.

Log messages generated by BASH scripts have the script name as the prefix to each message.

DEBUG messages

The python programs and bash scripts also generate a lot of debug messages, but only if the debug flag is set to true in the DOTENV file. These messages are used during testing or to diagnose problems during production. The DEBUG flag can be changed by sending an email command.

NOTE:

Changing this DEBUG flag to a different value during production will require all currently running scripts and programs to be restarted as the DEBUG flag is read only once by each program and script.

Appendix D – Installation instructions

This appendix list the steps to install this project.

Hardware purchases

The following is a list of the items to be purchased or obtained for free. The suppliers and costs (approximately) are those from the time this project was started.

Hardware item	Usage	Supplier	Cost £
Raspberry Pi model 4 with 2GB memory	Main processor of the system	Pimoroni	45.00
µSD card 128GB	Disk storage for the Raspberry Pi	Amazon	10.00
Case for Raspberry Pi model 4	To mount the processor in	Pimoroni	5.40
Power supply for Raspberry Pi model 4	Official power supply 15W USB-C	Pimoroni	9.00
HDMI converter cable	Converter cable to connect a HDMI cable to the micro HDMI port on the Pi	Pimoroni	4.00

Table 6 - Hardware purchases

Optional items

The following items are optional.

Hardware item	Usage	Supplier	Cost £
ENERGENIE Power Socket for Raspberry Pi	Pi controller power socket to provide remote power on and off. Mainly for the Screen(s)	Amazon Model: ENER002-2Pi	20.00
Keyboard and mouse	For use during installation and development. Also for System Admin	Amazon Model: Logitech	20.00
Computer screen	Used during the set-up and installation of the software etc.	Already available.	100.00

Table 7 - Optional hardware items

Summary of installation steps

The following table is a quick guide to the installation steps. Use these steps in the order listed. Further explanation and what to do is explained after the table.

It is recommended that the software is installed in a working folder. During development this was "~/Documents/pi_signage". Do any configuration or changes to the files in these working folders. Then use the "make install" command to put everything in the production folder.

Item	Action	Activity
1	SETUP	Raspberry Pi Operating System and initial O/S setup. Correct the vcgencmd for turning screens on or off.

Item	Action	Activity
2	INSTALL	Install the required software packages and modules. Packages: unclutter , dos2unix , html2text , ImpressRunner , LibreOffice Python modules: python-dotenv and energenie
3	SETUP	Create an Email account and get the IMAP and SMTP server addresses and ports.
4	CREATE	Create the working folder and the sub-folders. Create the production folder.
5	COPY	Copy the scripts, programs, configuration and data files to the working folder.
6	CONFIG	Configure the unclutter utility.
7	CONFIG	Configure the dotenv file, named ".env", with the secret data such as the email credentials.
8	CONFIG	Edit the common configuration file with the constants suitable for the intended installation. Also check some of the specific O/S issues in the Python and Shell scripts.
9	CONFIG	Change the Makefile to ensure the correct folders are used and all entries are correct.
10	TEST	Tests may be conducted with the scripts and programs to ensure they are all working and no errors found. Suggest that the DEBUG flag in the dotenv file is set to "True". Ensure it is set to "False" before doing the production install.
11	INSTALL	Run the command "make install" to install the files to the production folder.
12	EDIT	/etc/xdg/lxsession/LXDE-pi/autostart Update the file to start the script that will run the presentation.
13	CRON	Set the CRON entries for the starting and stopping of the presentations, reading the email messages, sending status messages, etc. These are very specific to the installation and will need careful working out.
14	FINISH	Full production is now possible. Ensure that the device can be accessed via SSH or VNC in order to manage the device. If the ENERGENiE sockets are used, ensure the aerial is fitted and the signal can be received by the sockets.

Table 8 - Summary of the installation steps

(1) Setup the Raspberry Pi

Instructions on how to setup the Raspberry Pi with an operating system and how to power it up for the first time, are on the official Raspberry Pi website. Follow the instructions on this website to have the Raspberry Pi up and running properly.

<https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up/0>

You will need access to a screen, keyboard and mouse for the installation. These can be items that are already present, or are purchased specifically for this project.

Once the Raspberry Pi operating system is installed and working and the network is up and access to the internet has been established, the Raspberry Pi is ready for the rest of the installation. Make sure that SSH and VNC are enabled in the Raspberry Pi configuration.

(2) Install required software packages

There are a number of software packages that will need to be installed as these are required for the proper operation of this project.

unclutter

Unclutter removes the mouse pointer on the screen after a number of seconds of inactivity. In this project it is recommended to use an inactivity of 10 seconds.

Install using the command: `sudo apt install unclutter`

dotenv

This is the Python module that provides access to the secrets file. This file is by default ".env" and is thus hidden when doing a folder list.

Install using the command: `sudo pip3 install python-dotenv`

html2text

This utility converts HTML formatted message text to normal text. It is used to find the commands and data in an email message.

There is a difference when running this utility on Linux Mint or Raspbian. The Python "Process messages" program checks what the current processor is and selects the correct command to execute.

Install using the command: `sudo apt install html2text`

LibreOffice

This software may need to be installed if it is not already included with the Raspberry O/S. Install by selecting it from the software installation program that comes with the Raspberry O/S.

ImpressRunner

This is an extension to the LibreOffice Impress program that allows for an automatic start of a presentation when the file with the presentation is executed. Visit the website for [ImpressRunner](#). Follow the instructions on the website on how to use it.

energenie

This is the Python module to manage the ENERGENiE power switches. Together with the daughter board this controls the powering on and off of screens if the screen blanking is not working. Only install if this facility is going to be used.

Install using the command: `sudo pip3 install energenie`

dos2unix

This is a utility that converts text files from the DOS format to UNIX format. This relates to the line terminating characters used in DOS (cr/lf) to the UNIX (lf) characters. Used to convert the files when updating the the scripts/programs using the "update" command.

Install using the command: `sudo apt install dos2unix`

logging

This is the Python module that writes messages to a log file. The log messages have a log level. See the documentation for this module for more information. All programs and scripts use this feature and a similar layout of the message is used as in the SYSLOG messages.

Install using the command: `sudo pip3 install logging`

syslogquick.sh

This script displays syslog messages of interest. The messages are found and it will only take the last two days into consideration. It is used in the daily email message to list the syslog messages. It is not required, but can be obtained from the author if needed.

(3) Email account

Setup a suitable email account on any free email service provider. For this project the email service selected was "outlook.com" which is a Microsoft service. It allows an easy access method by using account name and an "apps" password.

Make sure that a specific "apps" password is obtained as this avoids the two-factor authentication required for almost all email services. See the relevant website for the email service chosen for information on how to generate one of these "apps" passwords.

Once the email account has been set up collect the following information that will need to be saved in the "dotenv" file for this project.

USERNAME

The username or email address for the account. Used to login to the IMAP and SMTP servers.

PASSWORD

The "apps" password for the email address of the account. Used to login to the IMAP and SMTP servers.

IMAP_SERVER

The IMAP server address for reading the emails. Use your email provider's IMAP server. Look for your provider's IMAP server on Google or check this page: <https://www.systoolsgroup.com/imap/>.

IMAP_PORT

The IMAP server port address to use. For Outlook this is port 993. See the webpage suggested for the IMAP_SERVER.

SMTP_SERVER

The SMTP server address for sending email messages. As for the IMAP_SERVER, check the webpage for the SMTP server name and port number.

SMTP_PORT

The SMTP server port address to use. For Outlook this is port 587. See the IMAP_SERVER suggested webpage to check for the correct information for your email account.

The above details are to be kept as secret as possible as they are relevant to the specific project installation.

(4) Create the folders

Before the installation of the scripts, programs, configuration and data files, the relevant folders must be created. The following are recommendations as these were used during the development of this project.

Decide where the scripts are to be stored for (1) the development and configuration of the installation and (2) for the production running of the project.

The production files need to be stored separately from the development and configuration area to ensure that once a file is in production the changes can be monitored and managed.

The following folders were used:

For development and configuration the following working folder is recommended:

`~/Documents/pi_signage`

For full production of the installation create this folder:

`~/Production/pi_signage`

NOTE: The working folder is where the project files are maintained. The production folder is where the project files are when running the Digital Signage installation. Never update anything in the production folder. Instead do all changes in the working folder. Test them and when fully working, use the "make install" command to copy the files to the production folder.

The sub-folders for the working folder will need to be created manually. Use the folder information in this document for the folders to be created.

The sub-folders that are used in the production folder will be created by the [Makefile](#) when installing the software to the production folder using the 'make' command.

(5) Copy the files

All the files for this project are contained in a TAR.GZ file. This includes a folder with the PDF documentation.

Copy the TAR.GZ file to a suitable folder and use the normal utility or the file-manager to extract the files. It is assumed that whomever is installing the software has enough knowledge of the relevant commands.

Extract all the files and store them into the working folder. Check that they are all present using the list of files in this document.

Ensure that all executable programs and scripts have the correct file permissions. It is suggested to only allow executable for the "owner" (i.e. `chmod 744`). Data files just need the read/write permission for the "owner" (i.e. `chmod 644`).

(6) Configure the UNCLUTTER utility

Install the "unclutter" utility as described above. Then configure the file:

`/etc/default/unclutter`

The contents of the file should be amended as follows:

```
# /etc/default/unclutter - configuration file for unclutter

# Set this option to 'true' if you want to start unclutter
# automagically after X has been started for a user.
# Otherwise, set it to 'false'.
START_UNCLUTTER="true"

# Options passed to unclutter
EXTRA_OPTS="-idle 10 -root"
```


Change the "-idle" parameter (highlighted above) to 10.

This file is read by the following file when the X11 desktop starts:

```
/etc/X11/Xsession.d/90unclutter
```

There is no need to modify this file. This is for information only.

(7) dotenv configuration

The dotenv file is generated by copying the file ".env_change" to the ".env" file. Update the ".env" file to suit the implementation.

When the email account has been obtained, including the special "apps" password, update the "dotenv" file (.env) with the email data.

Update the IMAP and SNMTP data with the ones for the email account. See the website pointed to in the .env file for names and port numbers to use for the chosen email account.

Make sure that the list of authorised email addresses is set-up otherwise the email feature will not work.

Set the DEBUG flag to "False" when installing the files to the production folder. Otherwise the data in the logging file will be quite large.

Set the "MESG_COUNT" variable to 1. This counter is used to make sure that the email messages have a unique filename and have this counter prefixing the attachments so it knows which email message has what attachment.

(8) Common system configuration

The common configuration file is called "**signage.conf**". This file contains all the configurable constants that are used by all the BASH scripts and the Python programs.

Copy the file "**signage.conf.change**" to the proper name of "**signage.conf**". The configuration in the latter can now be updated to suit the implementation.

Follow the comments in the file and make the necessary changes. Be aware that the Python programs have defaults for these settings when executed outside a BASH script.

Make sure that all variables in this file have "export" in front of them. Otherwise they may not be seen by the Python programs.

Special issues to consider

There are some issues to be aware of. The HTML2TEXT utility works differently on the Linux Mint device and the Raspberry Pi. Similar caution is applied to the execution of the chromium command to start the web based presentations. As much as possible has been done to mitigate these difficulties, but attention to these are needed.

The program and scripts that are affected are (the line numbers are approximate):

```
process_messages.py (line 286)
start_presentation.sh (line 56)
```

(9) Update the Makefile

Make sure that the directory names for the working and production folders are changed to the folders to be used for this installation. Correct any other commands as necessary.

(10) Test the programs

Before committing the project to production, it is best to test all the scripts and programs in the working folder with the DEBUG flag in the dotenv file set to "True" so that there is debug information in the log file.

Once everything is working OK, reset the DEBUG flag to "False". As all scripts and programs use this DEBUG flag, setting this flag to False affects all.

(11) Install for production

If everything is now configured and tested, execute the command to install the files to the production folder.

There is a "Makefile" that has all the instructions in it to install the files.

Before using this Makefile, check that it is going to use the correct folders for working and production.

Execute the command: `make install`

This will create folders, copy the scripts, programs and data files to the production folder. For more info, see the "Makefile".

(12) Presentation AutoStart

In order for the presentation to start automatically after a boot-up of the system, the file for the X-Windows AutoStart needs to have the proper command added.

The file in question is: `/etc/xdg/lxsession/LXDE-pi/autostart`

The contents of this file should look something like this:

```
@lxpanel --profile LXDE-pi
@pcmanfm --desktop --profile LXDE-pi
@xscreensaver -no-splash

/home/pi/Production/pi_signage/start_presentation.sh on wait
```

Make sure that the "@" at the start of the presentation command is not added, otherwise it will be impossible to stop the presentation. The "@" means that the command is attempted again if the process has been killed. The argument "wait" is a special argument for the boot-up start as it waits 10 seconds before starting the presentation to allow time for the network to stabilize. The argument "on" (or "off") can be added to turn the ENERGENiE sockets off. This is of course only if these sockets are being used.

(13) CRONTAB settings

Setting the CRONTAB entries for managing and controlling the Digital Signage project will be solely dependent on where and how this installation is going to be used. There will be many considerations as to what is displayed, at what time, at what day, etc.

Make sure that there are entries for the System Admin functions and for reading and processing the emails. Those are essential to the working of the system.

The following script are designed to be started using CRON.

Script to run	Possible timings
check_read_email.sh	Starts the program to check for emails and then starts the script to read and process these emails. This script keeps running until terminated by a SIGINT interrupt signal. The interval between checking for emails is configurable in the main configuration file.
daily_status.sh	Run this once a day after the "delete_files.sh" script. It emails a status message to the SysAdmin person.
issue_command.sh	This can be run to start/stop/restart, a presentation and other functions. It is intended to be used both in scripts, CRON and from the command prompt.
kill_check_email.sh	Run this script at a time when the Digital Signage system is not in use. It terminates the "check email" program in an orderly fashion.
send_email.sh	At regular intervals execute this to email queued messages.

Table 9- Scripts to be run by CRON

Example of a CRON table as used during testing:

```
# Edit this file to introduce tasks to be run by cron.
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# For more information see the manual pages of crontab(5) and cron(8)
# # Example of job definition:
# .----- minute (0 - 59)
# | .----- hour (0 - 23)
# | | .----- day of month (1 - 31)
# | | | .----- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .---- day of week (0 - 6)
# | | | | | (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# * * * * * command to be executed

SHELL=/bin/bash
PATH=~:/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

# EMAIL control
# Start and regularly check the check_email program
*/10 7-20 * * * ~/Production/pi_signage/check_read_email.sh
# Terminate the check_email program running in background
10 21 * * * ~/Production/pi_signage/kill_check_email.sh

# Send emails from the queue folder
1-59/6 7-18 * * * ~/Production/pi_signage/send_email.sh

# PRESENTATION control
0 21 * * * ~/Production/pi_signage/issue_command.sh stop
0 7 * * * ~/Production/pi_signage/issue_command.sh start

# SYSTEM ADMINISTRATION entries
# Email the daily status message
0 1 * * * ~/Production/pi_signage/daily_status.sh

# Turn screen off on Wednesdays after Coffee and chat
0 12 * * wed ~/Production/pi_signage/issue_command.sh turn_off 0
```

```
# START at BOOT-UP
# Start the email checking program at boot up but delay it by 60 secs.
@reboot sleep 60 && /home/pi/Production/pi_signage/check_read_email.sh
```

(14) Full production

The installation should now be ready to be brought into full production.

If the ENERGENiE power sockets are used, make sure that the device can transmit the codes to the sockets and that the sockets are receiving the signal to turn them on and off. This may mean that the Raspberry Pi daughter board for these sockets has the aerial installed and it is visible.

Enjoy the project and please make any changes to suit your environment.

» End of document «