

CSCE 483 FALL 2020 UAS PILOT DATA COLLECTION Programmer Manual

Christopher Wray
Jordan Griffin
Samuel Sells
Venkata Dubagunta

TABLE OF CONTENTS

| | |
|---|-----------|
| Introduction | 3 |
| About This Manual | 3 |
| Code Access | 3 |
| Sections Covered | 3 |
| System Specifications | 3 |
| Operating System | 3 |
| Pin out Diagram | 4 |
| System Setup | 4 |
| Installing Required Packages | 4 |
| Running on boot | 4 |
| Formatting SD & USB for paths | 5 |
| Code Structure | 5 |
| Code Location | 5 |
| Module Structure | 5 |
| Data Collection | 6 |
| System Checks | 6 |
| Recording | 8 |
| Synchronization & Export | 10 |
| Processing | 10 |
| Error Handling & Troubleshooting | 13 |
| Error Handling | 13 |
| Error Code Key | 14 |
| Button & LED Configuration | 15 |
| LED Setup | 15 |
| Button Setup | 15 |
| Data Storage | 15 |
| File Structure | 15 |
| File Locations | 16 |
| Playback Requirements | 16 |
| Future Work | 16 |
| Hardware | 16 |
| Software | 16 |

| | |
|---------------------------------|-----------|
| UAS Pilot Data Collection | 3 |
| Appendix | 17 |
| Code Location | 17 |
| Figures | 17 |
| IDEFO | 17 |
| Sources & References | 24 |
| Acknowledgements | 24 |

Introduction

About This Manual

The purpose of this manual is intended for the developers to understand the elaborate details of code developed and used to run the UAS Pilot Data Collection Tool. Additionally it provides steps to set up the system on a new processor and how to troubleshoot the system.

About This Tool

The UAS Pilot Data Collection tool is used to record and synchronize the facial expressions and tablet interactions of a drone pilot. The device consists of a raspberry pi 4, rechargeable battery pack, PiCam 2.1, USB Webcam, and USB storage device embedded in a standard drone pilot harness. This manual is intended to help users understand how to set up and operate the system as well as provide useful information related to data storage and common error troubleshooting.

Code Access

The code developed is available on Github.

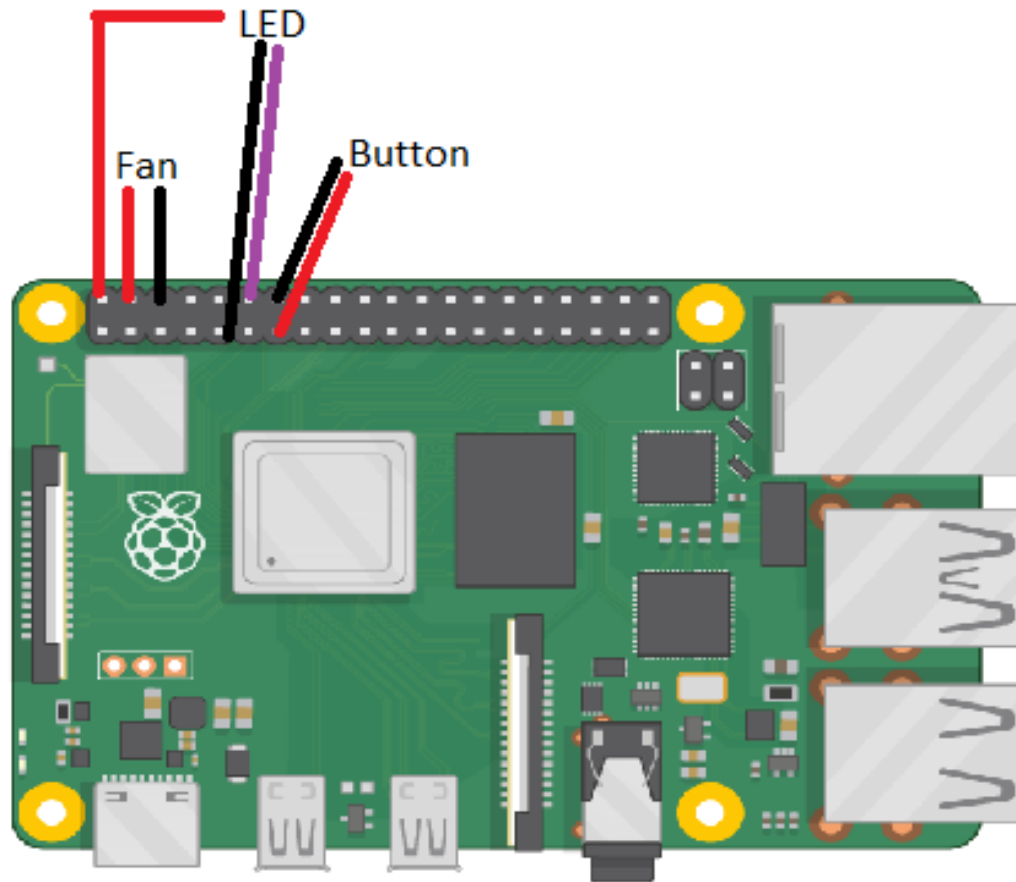
[Click here to access the repository](#)

System Specifications

Operating System

- Raspberry Pi 4 running Raspbian OS - Raspbian Buster, Linux 10
 - Installation and setup
- Coded in Python 3

Pin out Diagram



System Setup

Installing Required Packages

Run `install.sh` available on github that uses `pip3` to install the following packages

- openCV and dependencies
- vidGear and dependencies
- Ffmpeg and dependencies
- Picamera

Running on boot

- **To add code at startup follow the steps below:**
 1. Open the terminal on the raspberry pi
 2. Open the bashrc file using the following command :
sudo nano /home/pi/.bashrc
 3. At the end of the file enter the following command :
cd [to location of driver / required script]
python3 [name of script].py
 4. Reboot the pi :
sudo reboot
- **To remove code at startup follow the steps below:**
 1. Open script that runs on startup using an editor
 2. Comment out the entire code
 3. Reboot the pi :
sudo reboot
 4. Now , the code will not run on startup
 5. Open the bashrc file using the following command :
sudo nano /home/pi/.bashrc
 6. At the end of the file remove the following command :
cd [to location of driver / required script]
python3 [name of script].py
 7. Reboot the pi :
sudo reboot
 8. Now you are free to work on the pi without a background code running.

Formatting SD & USB for paths

- Format any USB desired using FAT32 and name the device “VIDEOS” - Note formatting the USB will erase all files on the device
- **On Windows :**
 1. Plug in the USB to a Windows PC
 2. Right click the USB in File Explorer
 3. Select “Format”
 4. Select “FAT32” under the File System dropdown
 5. Name the volume “VIDEOS” in Volume Label
 6. Click Start
 7. Let the process run, then eject the USB
- **On MacOS :**
 1. Plug in the USB to a MacOS PC

2. Open Disk Utility (Applications > Utilities)
3. Select the USB in the sidebar, then click Erase, Ensure you have selected the USB drive and not a single partition.
4. Name the USB “VIDEOS”
5. Select “MS-DOS (FAT)” under the File System dropdown
6. Select “GUID Partition Map” as the Scheme
7. Click “Erase”

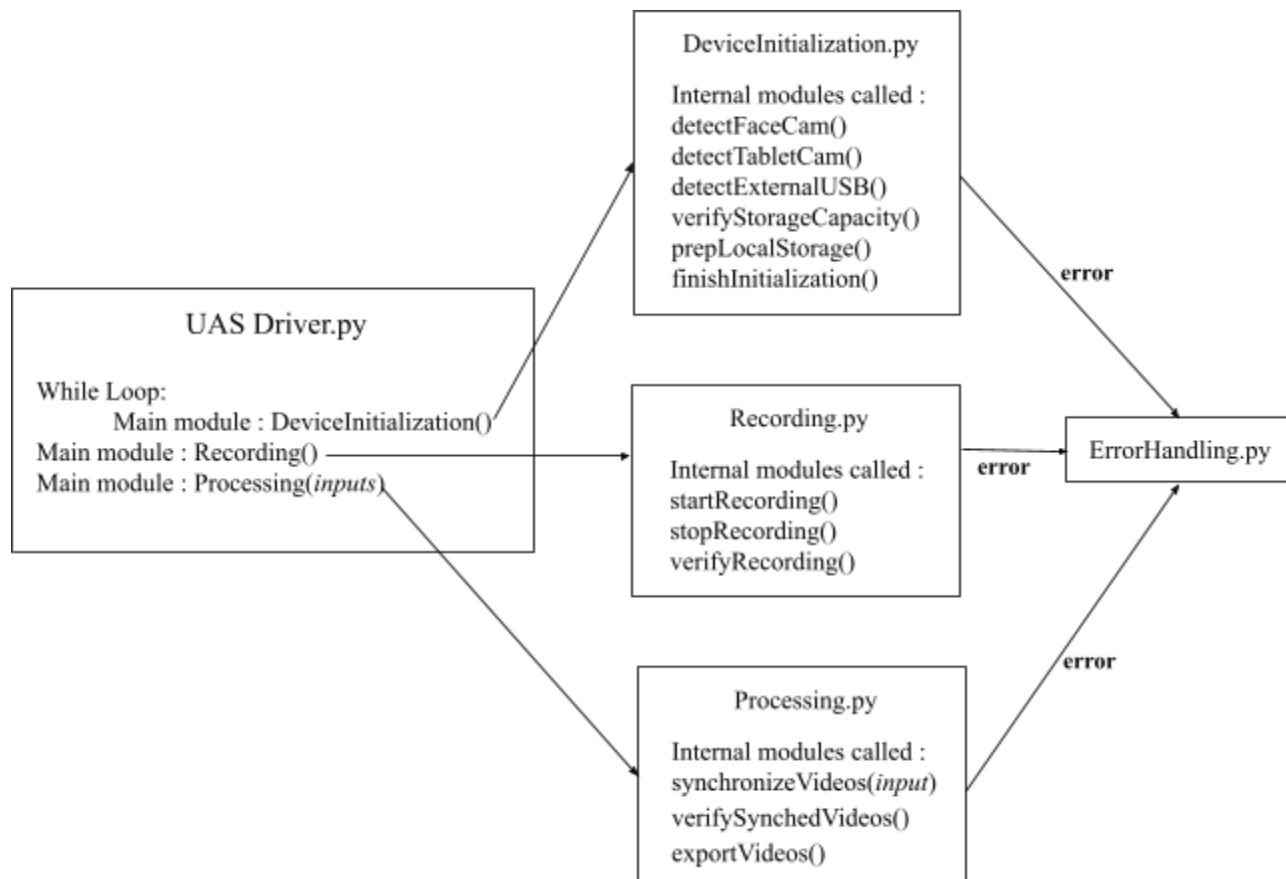
Code Structure

Code Location

- The files in the github repository are copied into : ~/Desktop/Codebase
- The main driver is added onto the bashrc file to run on startup.

Module Structure

- Each main and internal module shown below is described in the next few sections.



Data Collection

System Checks

Script reference : DeviceInitialization.py

IDEFO Available in Appendix

Main Module

Module Name : DeviceInitialization()

Function : Perform hardware detections, storage verification, prepare local storage

Input :

- Hardware : Waits for power supply to turn on
- Software : None

Output :

- Hardware : None
- Software : Integer object
 - 0 if all system checks happen successfully
 - -1 if even on system check fails

Description / Algorithm :

- In the main driver, this module is called in a while loop until all system checks are successfully met and module returns 0.
- In case any error occurs in system checks, -1 is returned to the driver while the internal modules hand over control to ErrorHandling with corresponding error codes.

Internal Module 1

Module Name : detectFaceCam()

Function : Determines if a valid stream can be obtained from the USB webcam focussed on the face in source 0.

Input : None

Output :

- Hardware : None
- Software : Boolean object
 - *True* if stream established successfully
 - *False* if stream not established and control given to Error Handling by calling errorFaceCam()

Description / Algorithm : Use Video Gear to set up stream in a try - except block

Internal Module 2

Module Name : detectTabletCam()

Function : Determines if a valid stream can be obtained from the Pi Cam ,
focused on the tablet in source 1 / source 2.

Input : None

Output :

- Hardware : None
- Software : Boolean object
 - *True* if stream established successfully
 - *False* if stream not established and control given to Error Handling
by calling errorTabletCam()

Description / Algorithm : Use Video Gear to set up stream in a try - except block

*****Note**** : Sources for Cameras are susceptible to change due to the processor's internal configuration. Currently **Tablet camera source change is handled in the code** using a try except block with source 1 / source 2. It is recommended that future developers check the source array if any misreadings occur to avoid interchange of camera output positions in the final merged video.*

Internal Module 3

Module Name : detectExternalUSB()

Function : Determines if a required external file system is mounted successfully
with the USB path : ~/media/pi/VIDEOS

Input : None

Output :

- Hardware : None
- Software : Boolean object
 - *True* if USB mounted
 - *False* if USB is not mounted and control given to Error Handling
by calling errorUSBDetect()

Description / Algorithm : Use Linux *statvfs* to check if USB is detected.

Internal Module 4

Module Name : verifyStorageCapacity()

Function : Determines if a required external file system is mounted successfully
and has more than 330MB

Input : None

Output :

- Hardware : None

- Software : Boolean object
 - *True* if USB has adequate storage capacity
 - *False* if USB does not have adequate storage capacity and control given to Error Handling by calling `errorUSBStorage()`

Description / Algorithm :

- Use Linux `statvfs` to check if USB is detected
- Available space calculated in MB as followed and checked : $(disk.f_bfree * disk.f_bsize / 1024 / 1024)$

Internal Module 5

Module Name : `prepLocalStorage()`

Function : Verifies existence if local SD card path : `~/Documents/localVids` to save video files and removes files from previous recording sessions.

Input : None

Output :

- Hardware : None
- Software : Boolean value
 - *True* if preparation done successfully
 - *False* if preparation fails

Description / Algorithm : Linux commands used : *exists* and *mkdir*

Internal Module 6

Module Name : `finishInitialization()`

Function : If all checks pass, called by driver to turn LED green

Input : None

Output :

- Hardware : LED turns green
- Software : None

Description / Algorithm : Call `LEDControl.turnGreen()`, control to driver

Recording

Script reference : Recording.py

IDEFO Available in the Appendix

Main Module

Module Name : `Recording()`

Function :

Input :

- Hardware : Waits for button press

- Software : None

Output :

- Hardware : None
- Software :
 - Float object : Duration (of recording)
 - List object : FileNameList (containing 2 file names : FaceCamVide, TabletCamVideos)

Description / Algorithm :

- In the main driver, this module is called after detection of a button press and waits until button is released
- Streams for both cameras initiated by VideoGear and appended to a list
- Writers for both streams initiated by WriteGear to save data frame by frame
- Recordings started, stopped and verified and control is returned to main driver

Internal Module 1

Module Name : startRecording()

Function : Indicates state transition and starts collecting framewise data from 2 cameras

Input : None

Output :

- Hardware : LED turns red
- Software :
 - List object : FileNameList (containing 2 file names : FaceCamVide, TabletCamVideos)
 - Float object : Duration (of recording)

Description / Algorithm :

- After obtaining streams and writer objects, both FaceCam and TabletCam streams are started using [stream].start()
- Start time is noted.
- After this, a while loop is established to read 2 frames separately using Videogear's [stream].read() and copied into corresponding frames to each camera.
- If any of the frames are not valid (return None) or a button press is detected, control exits the loop after noting Stop time
- Duration is calculated as : *Stop Time - Start Time* and stored in global variable duration.
- If any error occurs while recording, control given to Error Handling by calling errorRecording()

Internal Module 2**Module Name :** stopRecording()**Function :** Close all open objects securely and obtain mp4 files**Input :**

- Hardware : waits for button press in previous internal module
- Software: None

Output : None**Description / Algorithm :**

- First, both the camera streams are accessed and closed using Videogear's [stream].close()
- Next, writers are closed using Writergear's [writer].close(). This saves the videos as mp4 in the writer's location
- Files in the local Codebase moved to the SD access :
~/Documents/localVids
- If any error occurs while recording, control given to Error Handling by calling errorRecording()

Internal Module 3**Module Name :** verifyRecording()**Function :** Make sure the recordings are available on the SD Card**Input :** None**Output :**

- Hardware : None
- Software : Boolean object
 - *True* if all files in filename list available on ~/Documents/localVids
 - *False* if even one file in filename list not available on ~/Documents/localVids and control given to Error Handling by calling errorBadFile()

Description / Algorithm : Linux commands used : *exists*

Synchronization & Export

Processing

*Script reference : Processing.py**IDEFO Available in the Appendix***Main Module****Module Name :** Processing(fileNameList,duration)

Function :**Input :**

- Hardware : None
- Software :
 - Float object : Duration (of recording)
 - List object : FileNameList (containing 2 file names : FaceCamVide, TabletCamVideos)

Output :

- Hardware : turns LED Green on successful processing
- Software : Boolean object
 - *True* if all files in filename list available on ~/Documents/localVids
 - *False* if even one file in filename list not available on ~/Documents/localVids

Description / Algorithm :

- In the main driver this module is called after recording is done and LED is changed to blue by calling LEDControl.turnBlue()
- After establishing a common timestamp to append to video file names, it synchronizes videos, verifies merged video location and exports them to the USB drive with path : ~/media/pi/VIDEOS.

Internal Module 1

Module Name : synchronizeVideos(duration)

Function : Merge 2 camera perspectives and sync frame by frame

Input :

- Hardware: None
- Software :
 - Float object : Duration (of recording)

Output : None

Description / Algorithm :

- This module works in 2 phases :
- **Phase 1 :**
 - Stack 2 camera perspectives next to each other using ffmpeg filter_complex & overlay with appropriate padding and equal dimensions
 - This produces a fast version of the merged as frames are compressed and stores it on the SD Card for recovery if Phase 2 encounters any issues

- In case any of the files cannot be accessed by the ffmpeg library or it encounters any errors, the control is given to Error Handling by calling `errorBadFile()`
- **Phase 2 :**
 - Presentation timestamp of the fast merged video is set by calculating a slowing factor that would give the correct merged video of correct length.
 - Slowing factor = *(duration of recording) / (length of fast merged video)*
 - Ffmpeg `setpts` is used for this purpose and resultant merged video is stored on the SD Card `localVids` folder.
 - In case any of the files cannot be accessed by the ffmpeg library or it encounters any errors, the control is given to Error Handling by calling `errorBadSynch()`

Internal Module 2

Module Name : `verifySynchedVideos()`

Function : Make sure the merged file along with individual camera recordings are available on the SD Card

Input : None

Output :

- Hardware : None
- Software : Boolean object
 - *True* if all files in filename list available on `~/Documents/localVids`
 - *False* if even one file in filename list not available on `~/Documents/localVids` and control given to Error Handling by calling `errorBadSynch()`

Description / Algorithm : Linux commands used : *exists*

Internal Module 3

Module Name : `exportVideos()`

Function : Copy all files in the global `fileNameList` (3 files) from the SD card to the formatted USB Drive

Input : None

Output :

- Hardware : None
- Software : Boolean object
 - *True* if all files in filename list are copied to USB successfully
 - *False* if files not copied due to absence of USB and control given to Error Handling by calling `errorUSBStorage()`

Description / Algorithm :

- Linux Commands used : *exists*
- Python package *shutil.copy* used to copy without root access

Error Handling & Troubleshooting

Error Handling

Script reference : ErrorHandling.py

IDEFO Available in the Appendix

This module contains individual internal modules with a customized LED Blinking pattern for each error described below. For recoverable errors, you may troubleshoot and system checks will restart. For non recoverable errors the system must be turned off to either recover what data is available, or be restarted to start a new recording session.

1. Recoverable Errors :

- **errorFaceCam()** : Face camera (Logitech USB Camera) not detected
- **errorTabletCam()** : Tablet camera (Pi cam v2.1) not detected
- **errorUSBDetect()** : USB drive not detected / not plugged into pi
- **errorUSBStorage()** : USB exceeds storage capacity threshold / no space on USB

2. Non Recoverable Errors :

- **errorRecording()** : Error encountered while recording videos
- **errorBadSynch()** : Error encountered during Synchronization of videos in phase 2 of processing or while verifying
- **errorBadFile()** : Files not detected by libraries like ffmpeg or corrupted or wrong filenames passed to library functions

The structure of the internal module for all errors is similar and shown in the following section along with internal module description for updating the error log.

Internal Module for Errors

Module Name : error[*specific name*]()

Function : Show corresponding LED blinking pattern and write message to error Log

Input : None

Output :

- Hardware : LED blinking pattern in morse code for error

- Software : None

Description / Algorithm :

- LED is turned on and off in specific intervals for 3 loops using python time library command sleep
- Dot : sleep time = .1s
- Dash : sleep time = .4s
- Time between each loop = 1s
- Refer to error key for LED pattern displayed

Internal Module for Error Log

Module Name : updateErrorLog(errorName)

Function : Write error message with timestamp to errorLog.txt and copy it onto USB drive

Input :

- Hardware : None
- Software :
 - String object : errorName / error message

Output : None

Description / Algorithm :

- Timestamp of error occurrence calculated and appended to input string containing error message
- This new string is written to the errorLog.txt located in ~/Documents/localVids
- The same is copied onto / replaced in the USB drive to ensure existence of latest error log

Error Code Key

| ErrorCode (string) | Error Description (string) | Pattern | Recovery (Yes/No) |
|--------------------|----------------------------|------------------------|-------------------|
| errorFaceCam | No Face Camera | Dot dot dot ● ● ● | Yes |
| errorTabletCam | No Tablet Camera | Dot dot dash ● ● □ | Yes |
| errorUSBDetect | No USB | Dot dash dot ● □ ● | Yes |
| errorUSBStorage | USB Full | Dot dash dash ● □ □ | Yes |

| | | | |
|----------------|---------------------------------------|------------------------|----|
| | | ● □ □ | |
| errorBadFile | Raw Video Not Found | Dash dot dot □ ● ● | No |
| errorBadSync | Merged Video Not Synchronized / Found | Dash dot dash □ ● □ | No |
| errorRecording | Problem During Recording | Dash dash dot □ □ ● | No |

Button & LED Configuration

LED Setup

Script Reference : LEDControl.py

- **GPIO Pin connection : 18**
- **Python Integration :**
 - Use python neopixel library
 - Use python board to configure pi integration
 - **[LED Object] = neopixel.NeoPixel(board.D[*gpio pin number*], 1)**
- **Event Handler :**
 - **[LED Object][0] = (rgb value tuple)**

Button Setup

- **GPIO Pin connection : 27**
- **Python Integration :**
 - Button from python gpiozero package used
 - **[button object] = Button(*gpio pin number*)**
- **Event Handler :**
 - Check boolean output of **[button object].is_pressed()**

Data Storage

File Structure & Locations

- **Format :** The videos recorded using the data collection tool will be saved in MP4 format once processing of the videos is complete.
- **List of Available Files :**
 - *[timestamp]*FaceCamVideo.mp4
 - *[timestamp]*TabletCamVideo.mp4
 - *[timestamp]*fastMerged.mp4
 - *[timestamp]*merged.mp4
- **Location :** Raspberry Pi's SD card and the USB drive plugged into the system
 - SD Card Path : ~/home/pi/Documents/localVids
 - USB Path : ~/media/pi/VIDEOS
- **Past videos:** deleted from the Pi's memory upon startup, but any saved on the USB drive will remain, allowing the user to record multiple videos in one session.

Playback Requirements

- **Video viewing :** VLC on Windows 7+
- The video files are saved as “.mp4” and have been successfully played and viewed using VLC media player on Windows 7+ systems.

Future Work

Hardware

- Redesign the mounting of the face-cam to obtain a better view of the drone pilot and tablet.
- Upgrade Raspberry Pi to allow higher fps on recordings and reduce processing time of videos during the synchronization phase.

Software

- Enable WIFI and/or Bluetooth
 - Manual verification of face and tablet camera views
 - Allow app development to show preview of video to allow for adjustment of cameras before recording
- Enable additional data recovery

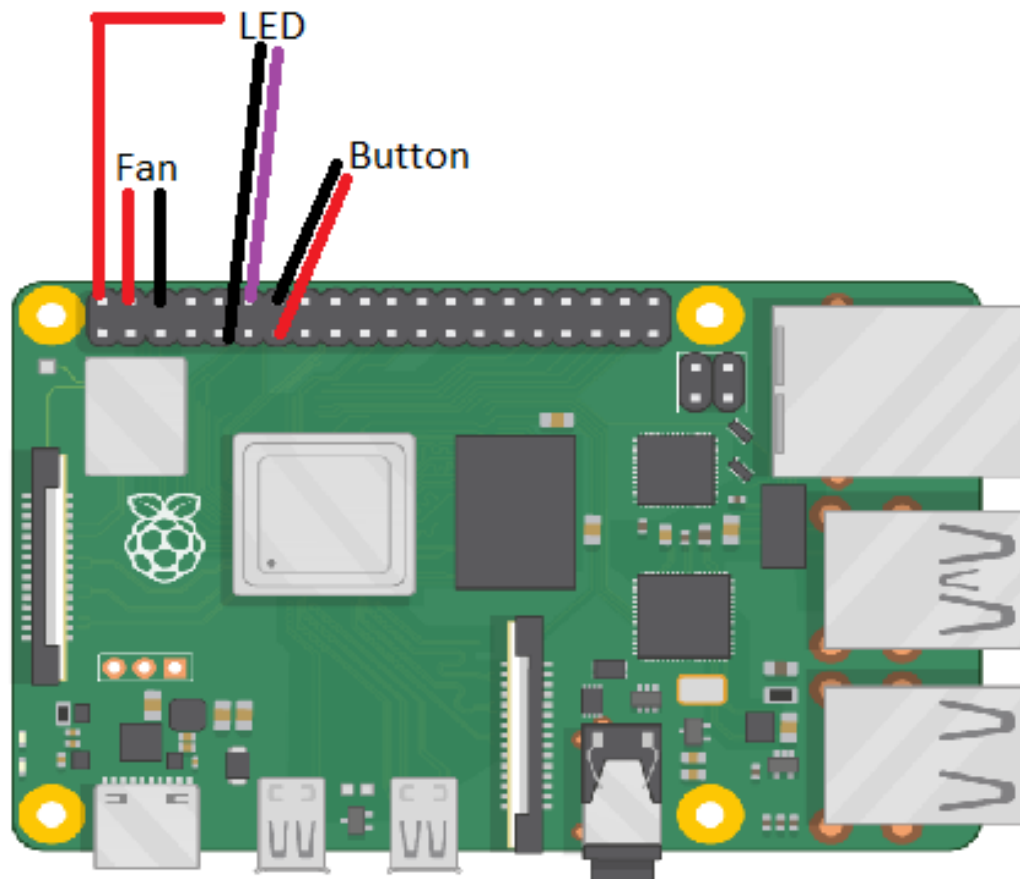
- Pair to another device and remotely transfer data
- Additional backup on device (long term, rather than short term)
- Additional button to try exporting local backup to connected storage device

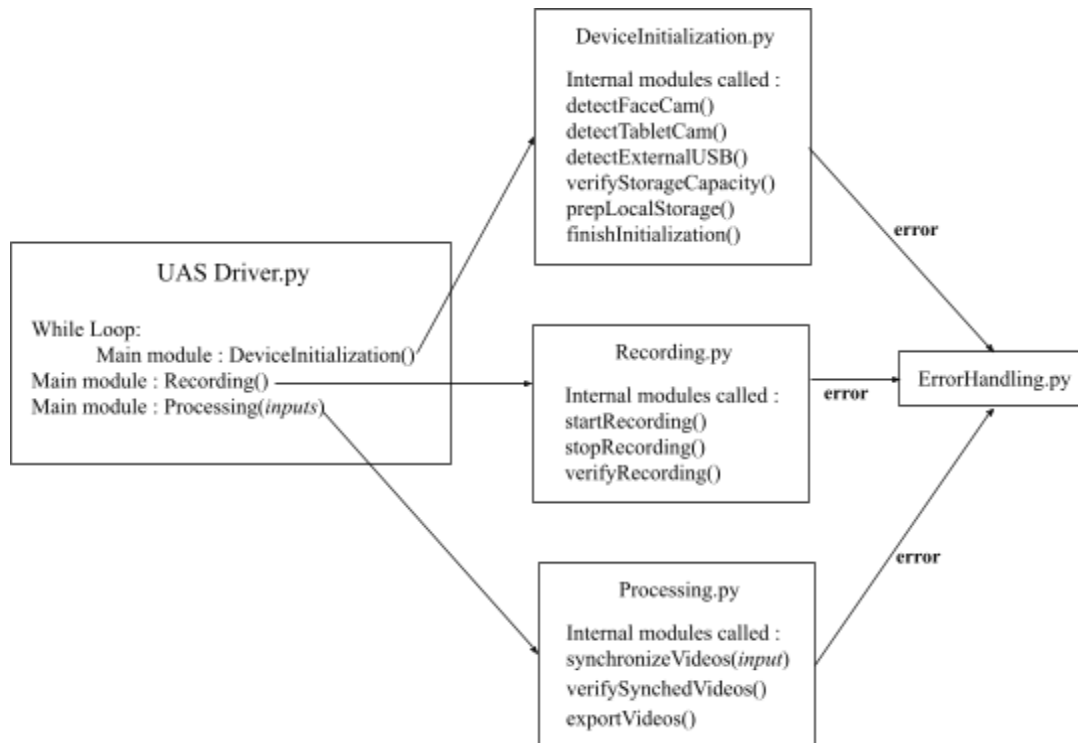
Appendix

Code Repository

- The code developed is available on Github : [Click here to access the repository](#)

Figures

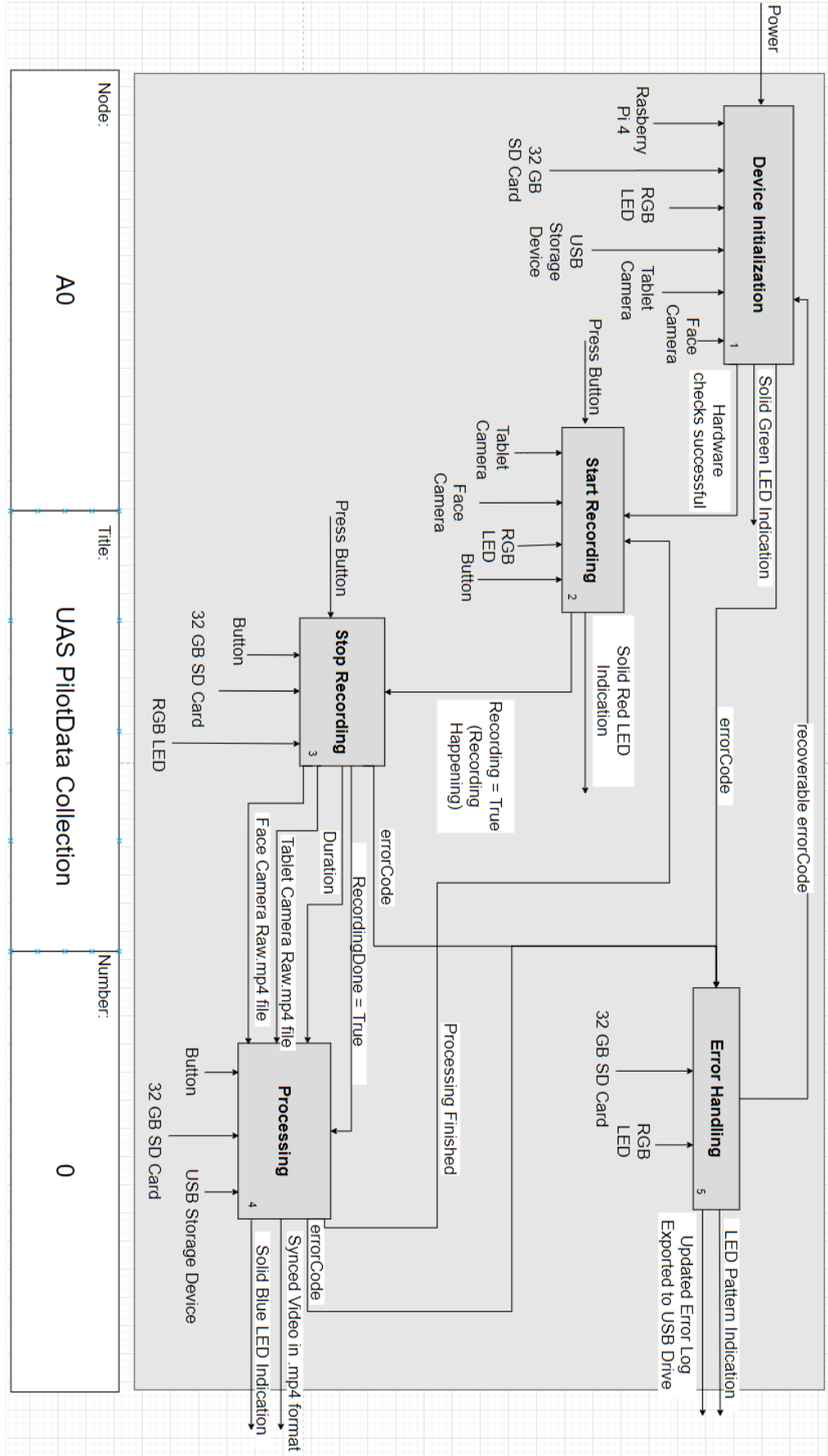


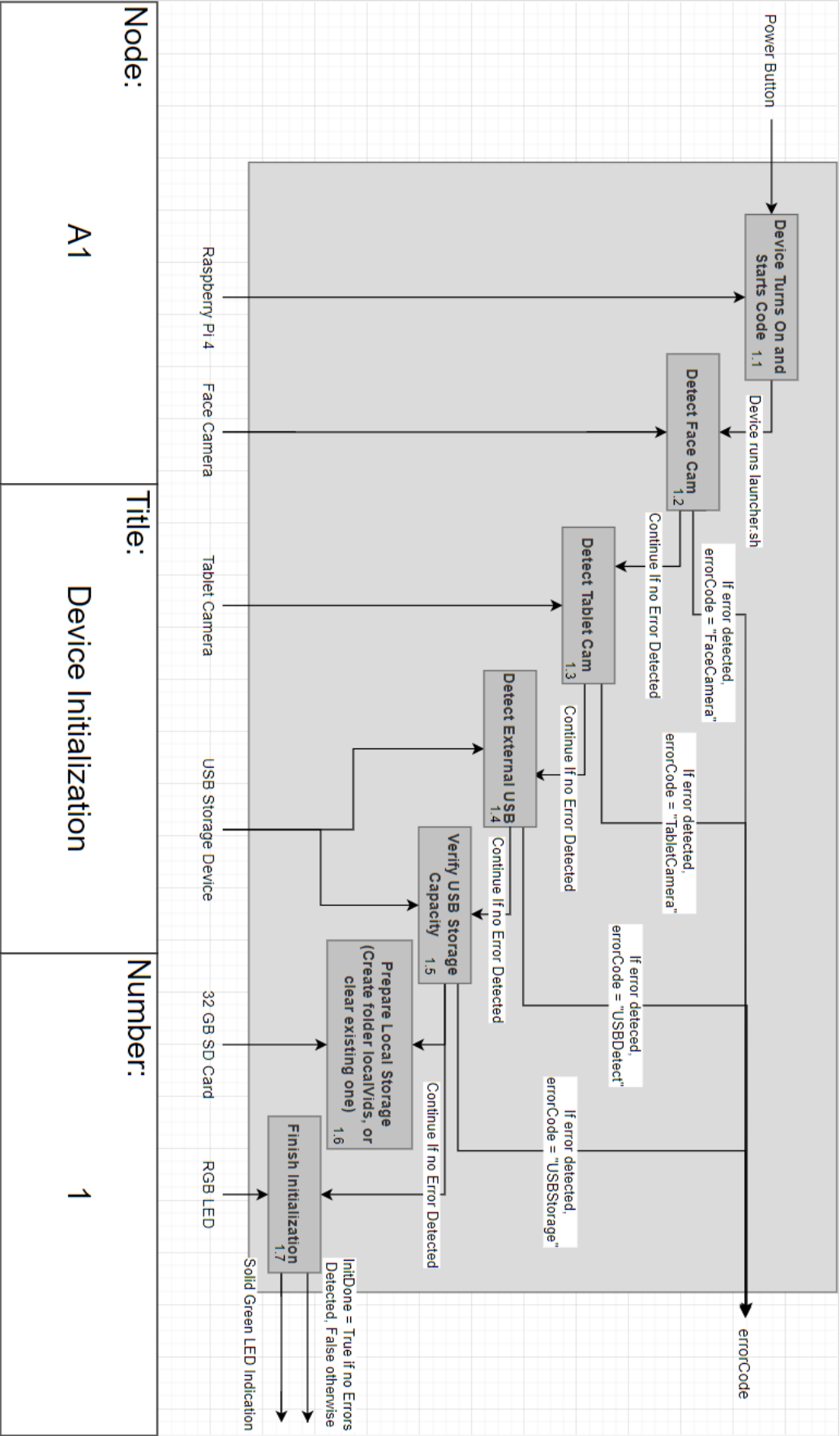


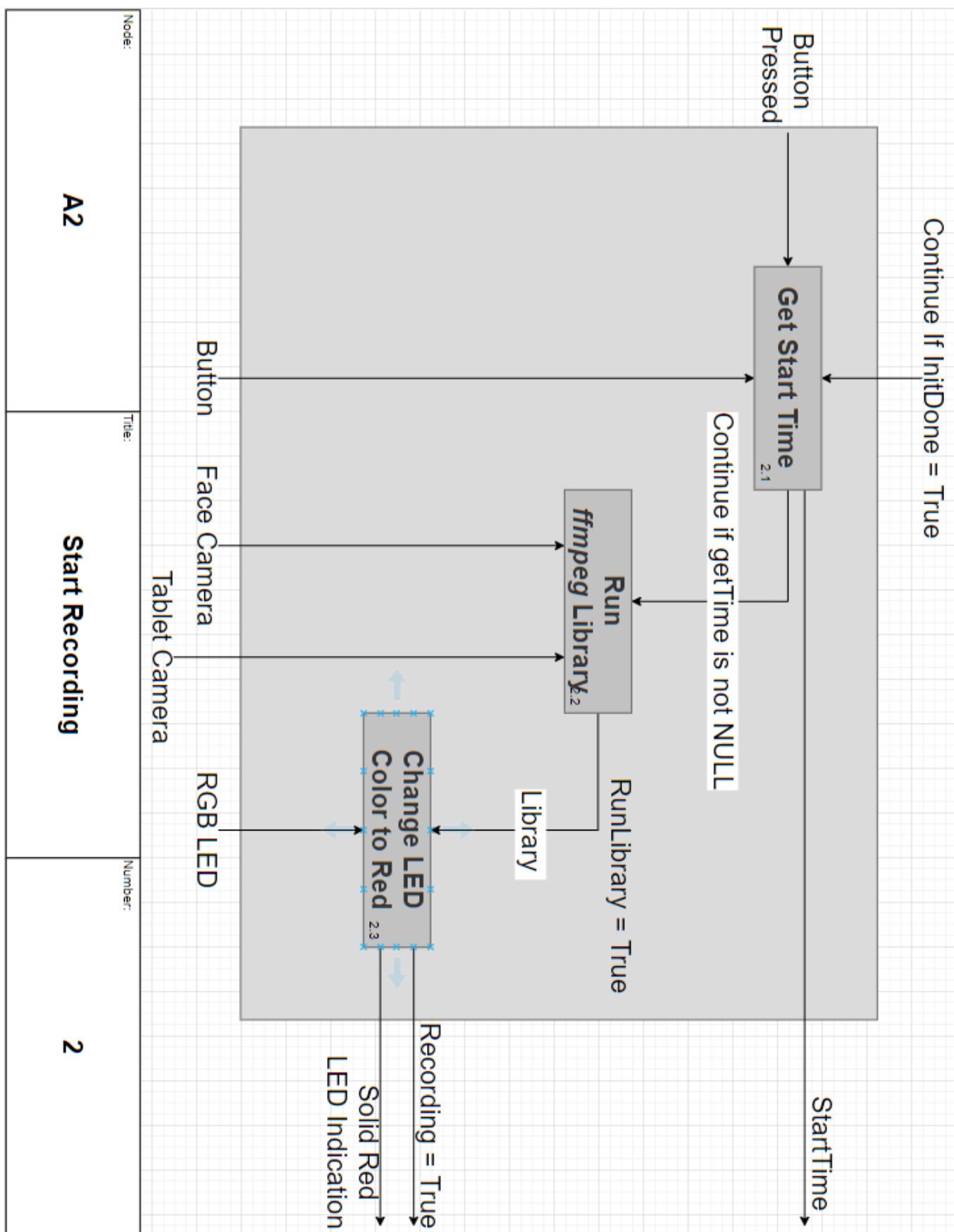
| ErrorCode (string) | Error Description (string) | Pattern | Recovery (Yes/No) |
|--------------------|---------------------------------------|------------------------|-------------------|
| errorFaceCam | No Face Camera | Dot dot dot ● ● ● | Yes |
| errorTabletCam | No Tablet Camera | Dot dot dash ● ● □ | Yes |
| errorUSBDetect | No USB | Dot dash dot ● □ ● | Yes |
| errorUSBStorage | USB Full | Dot dash dash ● □ □ | Yes |
| errorBadFile | Raw Video Not Found | Dash dot dot □ ● ● | No |
| errorBadSync | Merged Video Not Synchronized / Found | Dash dot dash □ ● □ | No |
| errorRecording | Problem During Recording | Dash dash dot □ □ ● | No |

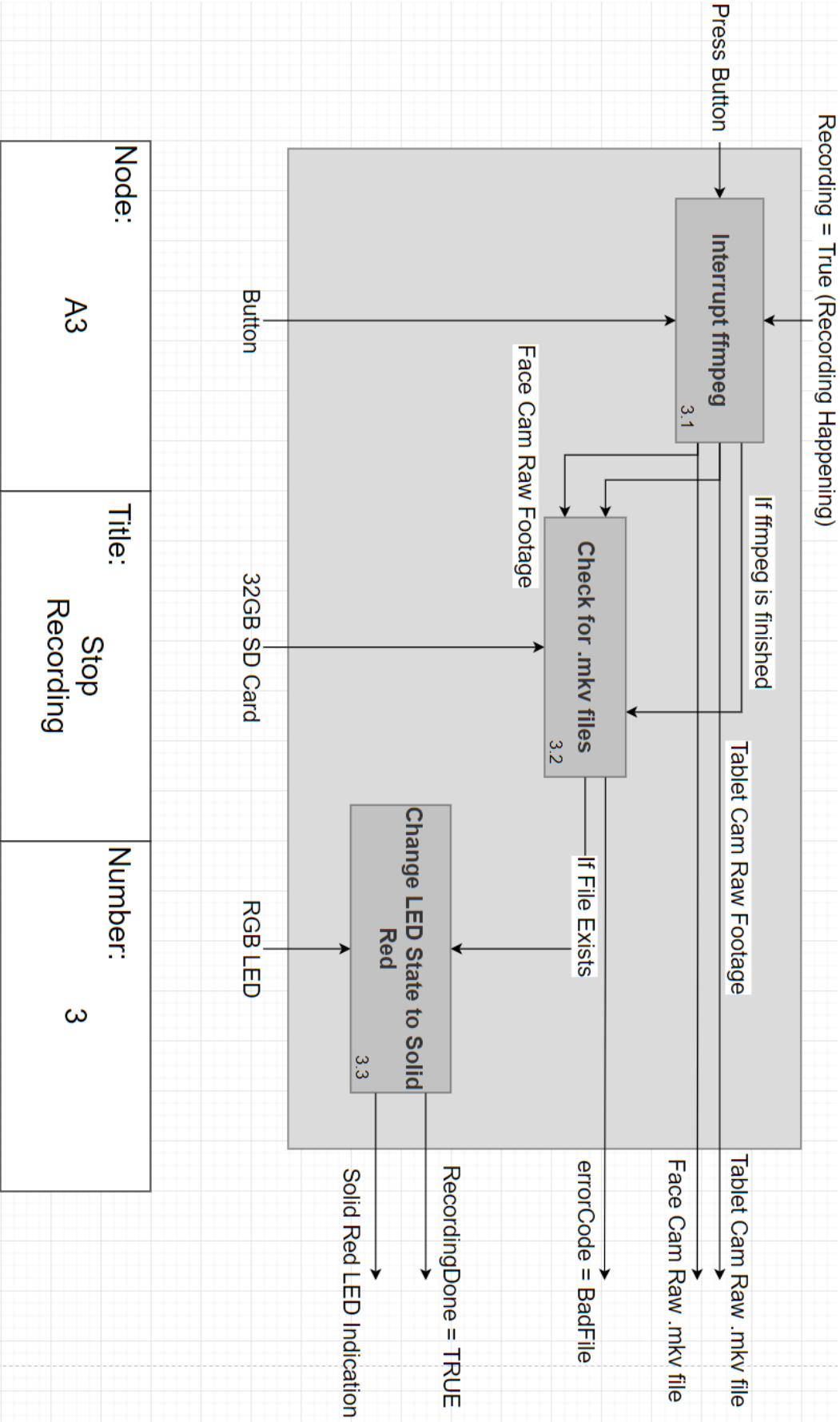
IDEF0

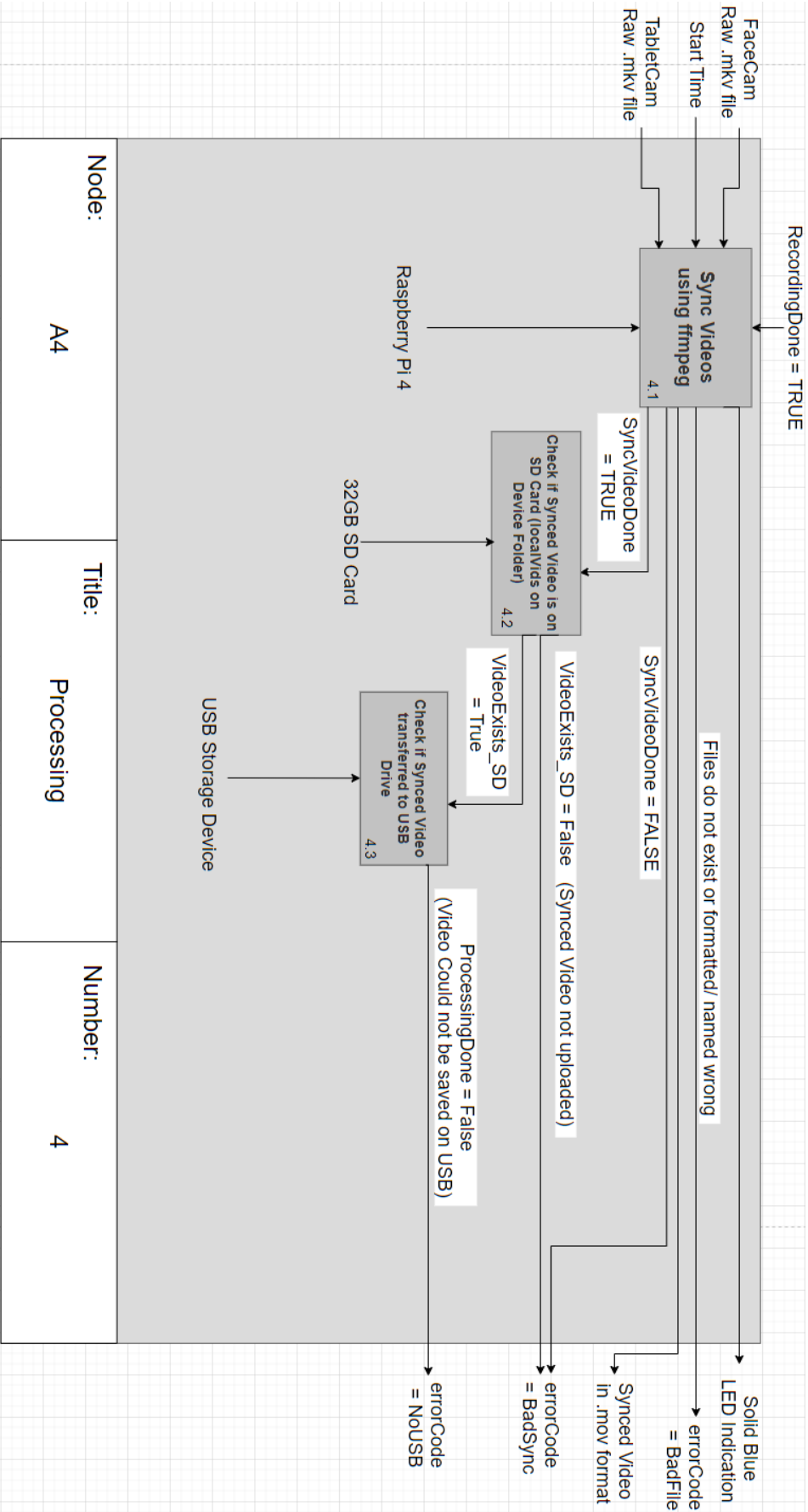
The IDEF0 for the system is on the following pages. It shows the design and structure of the system.

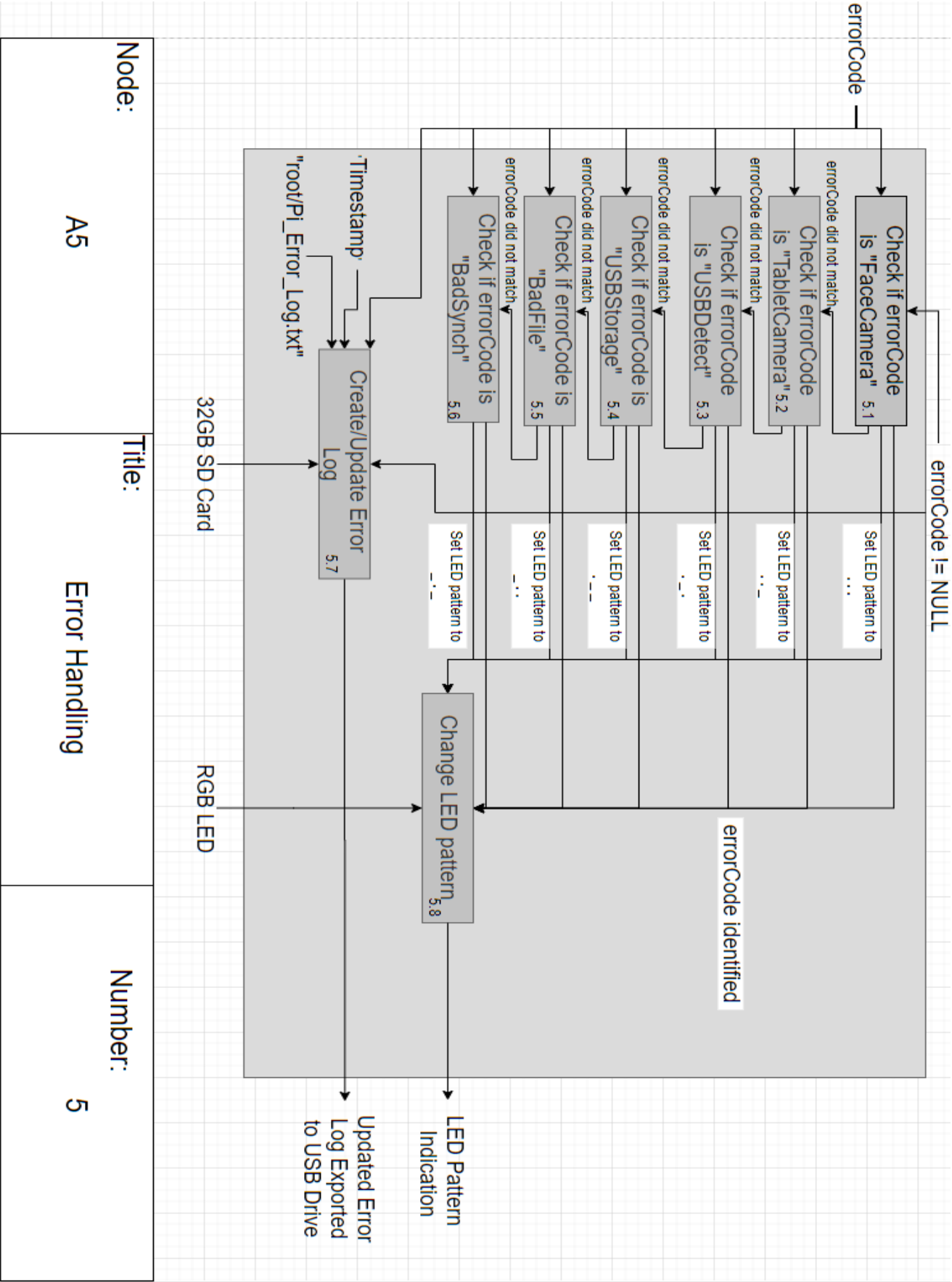












Sources & References

- Base Raspberry Pi case <https://www.thingiverse.com/thing:3749466>
- 40 mm Fan Mount <https://www.thingiverse.com/thing:1930251>
- Pi Cam Case: <https://www.thingiverse.com/thing:3829572>
- Pi Cam Mount: <https://www.thingiverse.com/thing:2886101>
- Illustration Source:
<http://dronereview.com/2016/08/28/flying-drones-is-becoming-a-marketable-skill/>
- Pin out Diagram
https://socialcompare.com/u/1906/raspberry-pi-4_1ab68d4ed26ceabb30086f7027af7e80.png

Acknowledgements


Letter of Acknowledgement

We, the members of the CSCE 483 Fall 2020 team 5: UAS Pilot Data Collection, hereby acknowledge that we have read, reviewed and verified this Programmer Manual, to submit as part of our project documentation.

Date: 11/29/2020

Signatures:

1. Jordan Griffin



2. Venkata Dubayunta



3. Christopher Wiley



4. Samuel Sells

