UNIVERSITY OF SOUTHAMPTON
Faculty of Engineering and Physical Sciences
School of Electronics and Computer Science

A project report submitted for the award of
MEng Computer Science

Supervisor: Professor Kirk Martinez
Examiner: Doctor Richard Gomer

Automated Remote Photography for Icelandic Glaciers

by Niall Morrison

May 3, 2023

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND PHYSICAL SCIENCES
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

A project report submitted for the award of MEng Computer Science

by Niall Morrison

Automated Remote Photography (ARP) enables photographers to remotely capture and download images without direct physical access to the camera. This project evaluates existing ARP solutions to develop a customisable ARP system suitable for monitoring glaciers in Iceland, where the ARP has limited battery, network signal and computational resources. Our novel ARP system automatically captures photos, uploads them to the cloud, and conserves energy by performing scheduled power cycles using a Raspberry Pi, an external power cycle timer (WittyPi), and a 4G-LTE module for remote access via mobile networks. Furthermore, cloud technologies are leveraged to enable remote monitoring and control via a web interface. The system was first prototyped, and then power and mobile data usage were optimised using image compression, operating system streamlining, camera modes, hardware substitution and network configuration. This dissertation demonstrates that our solution presents a viable ARP system for monitoring glaciers in Iceland, with the potential for usage in similar use cases.

Statement of Originality

1. I (Niall Morrison) have acknowledged all sources, and identified any content taken from elsewhere.
2. I have not used any resources produced by anyone else.
3. I did all the work myself, or with my allocated group, and have not helped anyone else.
4. The material in the report is genuine, and I have included all my data/-code/designs.
5. I have not submitted any part of this work for another assessment.
6. My work did not involve human participants, their cells or data, or animals.

Acknowledgements

A massive thank you to all my family members, as their 21 years of support, guidance and inspiration have helped get me where I am today.

Thank you to all of my friends that have helped me proofread this paper, especially Mark Towers, who helped a great deal with improving the formatting of this report!

Finally, a big thank you to Professor Kirk Martinez, whose guidance not only kept this project on the right track but also made the project thoroughly enjoyable.

Contents

Overleaf TexCount Word Count = 8798

Chapter 1

Introduction - Project Description

The original brief can be found in Appendix A.2

1.1 Problem

Automated/remote photography is a powerful tool that allows photographers to capture subjects and receive the image data without requiring immediate, direct physical access to the camera. However, design challenges arise from remote camera systems that are a significant distance from the data endpoint and a plentiful power source. Existing solutions are costly, lack customisation and mostly require immediate access to provide feedback to the operator.

1.2 Goals

This project aims to investigate the possible hardware and software techniques that enable effective remote/automated photography and implement a capable system.

1. Prototype an automated system that can take photos and transmit them.
2. From a system remote to the camera, access the image data sent by the camera.
3. Only have the system on when it needs to be on to preserve power.
4. Allow for the system to be configured for different photography cycles.
5. Provide means to facilitate the camera's set-up (such as checking where the camera is pointing).
6. Explore the possible ways to improve upon the prototype.
7. Test the system in the field.

1.3 Scope

1. **Hardware**
   (a) Components used will be 'off the shelf'.
   (b) Parts will be chosen such that the system can operate remotely.
   (c) Hardware choice will be limited to what is available within a given budget and time frame.
2. **Software**
   (a) Software will be the main way to combine the hardware elements to achieve the desired functionality.
   (b) A software solution will be required on the **receiving** end of the photos. The purpose of this will be to manage the data the camera produces and to control the camera.
   (c) As hardware choices are more rigid, improvements to the system's performance will be primarily explored through software.

3. **Other**

   (a) Should time permit for field testing, considerations will be made to make the system suitable for the environment it's placed in.

Chapter 2

Background

2.1 Time-lapse Photography

Time-lapse photography is a technique that captures a subject at defined intervals. It is widely used in various fields, particularly in documenting natural phenomena.

2.1.1 Physical Environments

Time-lapse photography can be used to study the earth's physical geography. For example, James *et al.* used it to monitor lava and glacial flow [18] [17].

Typical systems like these often use Digital Single Lens Reflex (DSLR) cameras. These cameras are connected to an external timer that triggers images to be captured [17]. Most will also run on an independent power supply suitable for a remote environment. For example, James' lava flow system used a 12V battery and a solar panel.

Trigger frequency is crucial in time-lapse systems, particularly when environmental conditions are uncertain. This is exemplified by the glacier study, where higher trigger frequency was deliberately implemented to account for uncertainty in image quality. The uncertainty in image quality was primarily caused by changing weather conditions around the glacier. This demonstrates how the surrounding environment influences the setup.

In addition, the glacial study highlighted the potential for the physical movement of time-lapse systems, which can negatively impact photo quality [18]. This can

lead to missed observations and wasted time, as the operator may not be aware of the movement until they next interact with the equipment.

Finally, James' glacial paper also acknowledges other potential environmental disruptions, such as tourists.

2.1.2 Wildlife

Time-lapse photography is also commonly used to study living organisms like plants.

The documentary by David Attenborough *et al.*, 'The Private Life of Plants', is a helpful example as the description of tools used and challenges faced is well documented. [33]

Similar to the physical environment examples, an SLR is triggered by an external computer with a timer. This setup also includes a powered rail for automatic position adjustment, allowing moving subjects to be kept within the image frame.

Furthermore, another similarity to the physical examples is that a camera's proper operation and setup depend on the surrounding environment. For example, lighting, rain and temperature were the main challenges mentioned.

Finally, an important limitation of typical automatic time-lapse systems is their inability to detect failure. As demonstrated by Kirby's experience in the documentary, system failure may not be discovered until long after its occurrence. Without immediate access, there was no way for Kirby to know about the malfunction.

2.1.3 Other Applications

Schoener presents a time-lapse photography example from an engineering perspective, using a system to measure water flow for water quality and supply analysis [35]. This system uses "low-tech" and "low-cost" equipment, differing from the previously mentioned environmental examples.

The decision to use time-lapse photography was motivated by the limitations of motion-triggered photography. Schoener found that having the camera start at regular intervals was more reliable than using water flow (motion) to trigger the camera.

Following the 2-year study, cost and portability were noted as key advantages of the time-lapse system compared to other, more advanced, permanent water flow monitoring systems. Additionally, visual inspection of data through time-lapse photography offers a unique benefit not provided by different sensors (e.g. Water flow meters).

Additionally, the system's reliance on an SD Card for storage limits the amount of data that can be collected in one go. With a capacity to store 180 days of photography, the device has a limited time frame for continuous operation without manual data collection. Using an SD Card also presents the risk of significant data loss due to failure.

Finally, the deployment of time-lapse systems in public locations may be subject to theft, requiring provisions to be made.

2.2 Camera Traps

Camera trap photography dates back to the early 20th century, with the idea that the camera system can detect stimuli to trigger itself. As described by Carey, early examples were as simple as using string for animals to tug on and activate a camera. [8].

As previously mentioned, motion-triggered photography may be unreliable when capturing subjects with difficult-to-detect motion [35]. Despite this, it is commonly used for capturing moving animals. Therefore, for suitable image subjects (i.e. easy to sense), motion-triggered photography may collect more relevant data than a time-lapse system.

Trolliet *et al.* [39] provide a detailed review of camera trap technology for wildlife study, covering both consumer and research contexts.

Like automated time-lapse photography, the method allows for remote photo capture without manual intervention [8]. However, these remote locations may be dangerous or difficult to access frequently [39], raising the question of how an operator would charge the trap's battery or retrieve its recorded data.

Moreover, the review demonstrates how human presence may not be optimal. For example, human presence may frighten an animal, or that animal's active hours could be at an inconvenient time in the day, like the middle of the night. Another feature that time-lapse photography could also provide.

In addition, the review highlights potential drawbacks of human presence, such as disrupting animal subjects or the desired image capture times being at inconvenient times (e.g. night). Camera trap photography can provide solutions to these issues.

Finally, the review discusses various configurations for camera trap systems, highlighting the need for careful consideration of hardware and setup based on the subject the system is meant to capture. This decision-making process is essential for trap systems, as the operator must design a reliable sensor system for their specific application.

2.3 Low Cost/Hobby-Grade Hardware

Many of the previously mentioned examples utilise costly camera equipment, with customisation limited by the options provided by manufacturers.

Grindstaff *et al.* [14] present a low-cost alternative photography system using a Raspberry Pi Model 3B+ and Raspberry Pi Camera Board v2 to monitor plant growth conditions remotely and improve research quality. This project was successful in achieving its aim of improving research quality.

The Raspberry Pi can provide both processing power for image processing and the flexibility of allowing the user to reconfigure the system by programming their custom behaviour. These are features that researchers design bespoke experiments value.

The Raspberry Pi offers both processing power for image processing and the flexibility of custom-programmed behaviour, which researchers may value for unique/specific experiments.

However, a significant limitation of a system like Grindstaff *et al.*'s [14] is reduced image processing fidelity compared to consumer-grade DSLRs. Despite this, fidelity was still sufficient for Grindstaff's experiments.

Though the scope of *et al.*'s [14] study is limited, it is argued that the system's flexibility allows for additional features, such as computer vision, to be appended to the system.

et al. [14] proposes that systems involving Raspberry Pis may offer a cost-effective alternative to more expensive traditional methods for researchers and industries.

2.4 Camera Networks

Networking can greatly enhance remote photography, as demonstrated by the study of Wireless Visual Sensor Networks (WVSNs) for surveillance by Abas *et al.* [2].

Abas *et al.* [2] highlight the power limitations of deploying networked devices with independent power sources. They recommend selective, intentional handling of image information to conserve power.

The report suggests several methods for reducing power consumption:

- Pre-processing and archiving of images to reduce data transmission and conserve power.
- Increased self-operation of the camera system to reduce the need for constant transmission to an external operator, conserving power.
- Dividing workloads between local camera systems and the cloud.
- Selection of power-efficient hardware.

Furthermore, the choice of communication methods is also discussed, including the impact on bandwidth, resolution, and fidelity of photos, as well as power

consumption. Abas *et al.* [2]. also introduce the idea of secure communication as a factor in the decision-making process.

Finally, Abas *et al.* [2] discuss the challenges of developing a general-purpose camera system due to the diverse hardware requirements of different settings (e.g. habitat monitoring vs. a controlled living room environment).

Chapter 3

Focus on Use-case and Specification

After a discussion with Kirk (the project supervisor), a specific use case for this project was identified. Like the examples in the project background review, the aim will be to use this project's automated camera to take photos of glacier landscapes in Iceland along with the research Kirk is participating in.

3.1 Glacial Photography

3.1.1 Subjects

The main photography subjects in Iceland that this project is concerned with are geological formations of rocks and the surrounding ice. This kind of environment is subject to significant seasonal change. The variations in this environment are such that they can be captured with **time-lapse photography**. There isn't a need for motion-triggered behaviour or anything of the like.

FIGURE 3.1: Glacsweb time-lapse [25].

FIGURE 3.2: Moving rocks and glacial ice [6].

Studying these time-lapses leads to a better understanding of the change in these glacial environments and, importantly, the changing climate's impact on the landscape. This section introduces the considerations that this application brings to the project.

3.1.2 Challenges Introduced by Iceland

Operating in Iceland introduces various challenges that must be considered in the design process.

Physical Access

Firstly, in the context of this project, there won't be constant and immediate access to a camera system. Unless an alternative method is established to communicate with the camera system/automate it, tasks that would otherwise be done with physical access won't be possible:

- Configuration.
- Error monitoring.
- Frequent data retrieval (reducing the risk of lost/corrupted data).
- Recharging the battery.

Sunlight

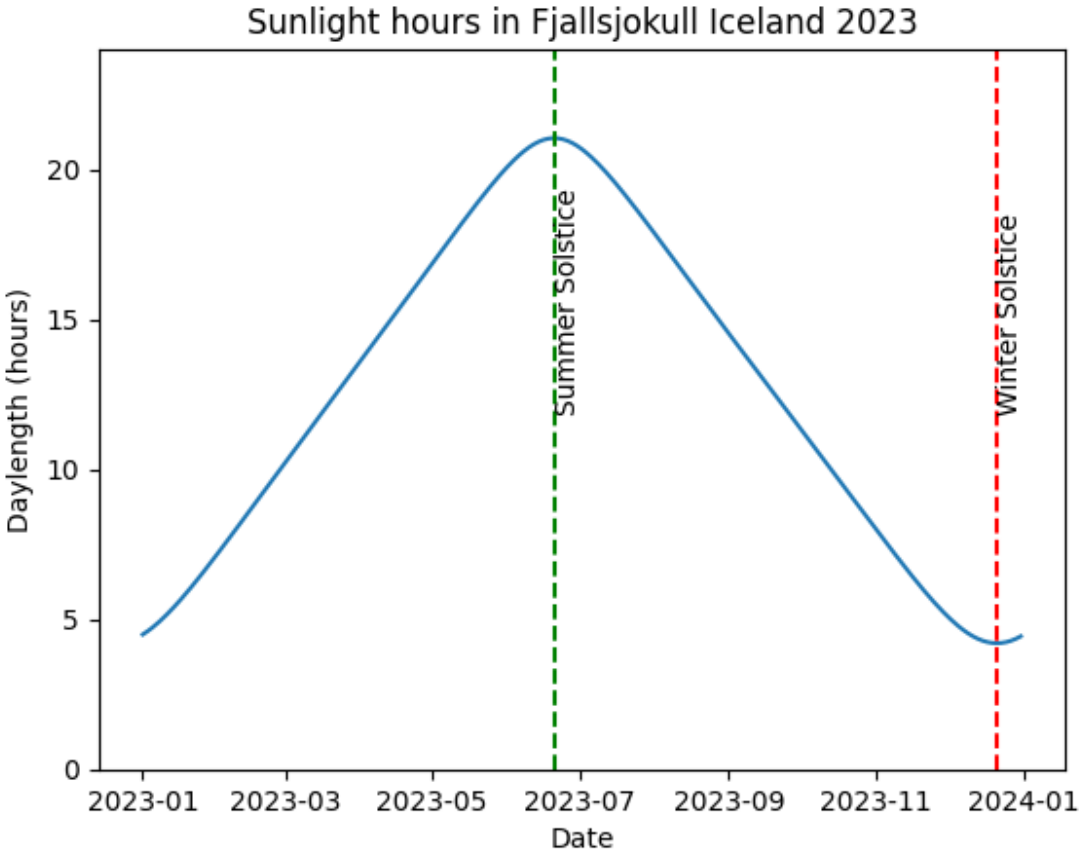


FIGURE 3.3: Example of sunlight hours in Iceland [23].

As demonstrated by Figure 3.3, before considering the impact of clouds, a typical Iceland summer will experience long days of up to 20 hours, but the winters will reach days as short as under 5 hours.

- A remote photography system could comfortably run on solar power during periods of long days. However, in the winter, a remote system will have to rely more on its battery power (especially if the daylight hours are interrupted by overcast skies).
- Shooting with a fixed schedule (e.g. Capture a photo at 7AM, 12PM, 5PM) may be fine during daylight-rich periods, but capture times may fall outside of daylight as the winter approaches. This is problematic if you rely on daylight to capture sufficient detail in your photos.

Weather

FIGURE 3.4: Example of 'unusable' image due to weather [25].

- **Rain and Snow** can cover photos and render them unusable. Little can be done to prevent these photos, but if the automated camera cannot tell which photos are usable, there should be a way to prevent these images from wasting limited resources (i.e. avoid wasting mobile data on a blurry photo).
- **Cloud** coverage can significantly reduce solar power performance.
- **Temperature** would mainly impact battery choice as certain variants may perform better in a colder environment.

3.2 More on Time-lapse Cameras

As time-lapse photography has been identified as this project's focus, they can be looked at in more detail.

3.2.1 Comparison Table

In addition to the custom time-lapse camera systems discussed earlier, existing consumer-grade time-lapse cameras can serve as inspiration. They can demonstrate features that they lack, and also suggest features that this project could implement.

	Camera			
Feature	*Brinno TLC200 (+Pro) [5]*	*Browning HP4 [27]*	*Odin 4K Time-Lapse Camera [9]*	*Tikee 3 + (Pro) [12]*
Price (GBP)	105-180	160	749	1040 - 1570
Battery Life*	3 Months	18 Months	42 Days	24 Days
Solar	Compatible	Compatible	Compatible	Yes
Customisable	Scheduling, Capture Settings	Scheduling, Capture settings	Scheduling, Capture settings	Scheduling, Capture, Upload settings
Remote Features				
Photo Uploads	No	No	Wifi Only	4G & Wifi
Monitoring	No	No	Wifi Only	Yes
Configuration	No	No	Wifi Only	Yes

Table 3.1: Comparing consumer-grade time-lapse cameras.

** These are the quoted maximum values. Enabling additional settings like increasing time-lapse frequency will shorten battery life significantly. Adding other power sources, such as solar power can increase these values.*

Summary

To summarise some of the observations from reviewing these cameras:

- Solar power compatibility is the default method to enable the long-term operation of the cameras.

- Consumer-grade cameras can get quite expensive. They don't provide many extra features for significant price increases. The base prices usually don't include solar panels either.
- The cameras that do network don't offer many options for saving data/power when uploading photos to the cloud.
- The scheduling on the explored models is static (e.g. shoot at noon every 24 hours). With this scheduling, you cannot say, "Take a photo at 12:00 and 15:00 but nothing at night, then repeat 24hrs later".
- All the cameras provide a way to be set up and preview what they will capture.
- Most of the time-lapse cameras have battery life in the order of months.

3.2.2 Limited Resources

When operating a remote time-lapse camera system, some finite resources/conveniences limit the operating conditions of the system, most notably:

- Battery Life.
- Storage.
- Mobile Data (for a networked camera, more data = more money spent).
- Access Time (i.e. time spent communicating with the device).

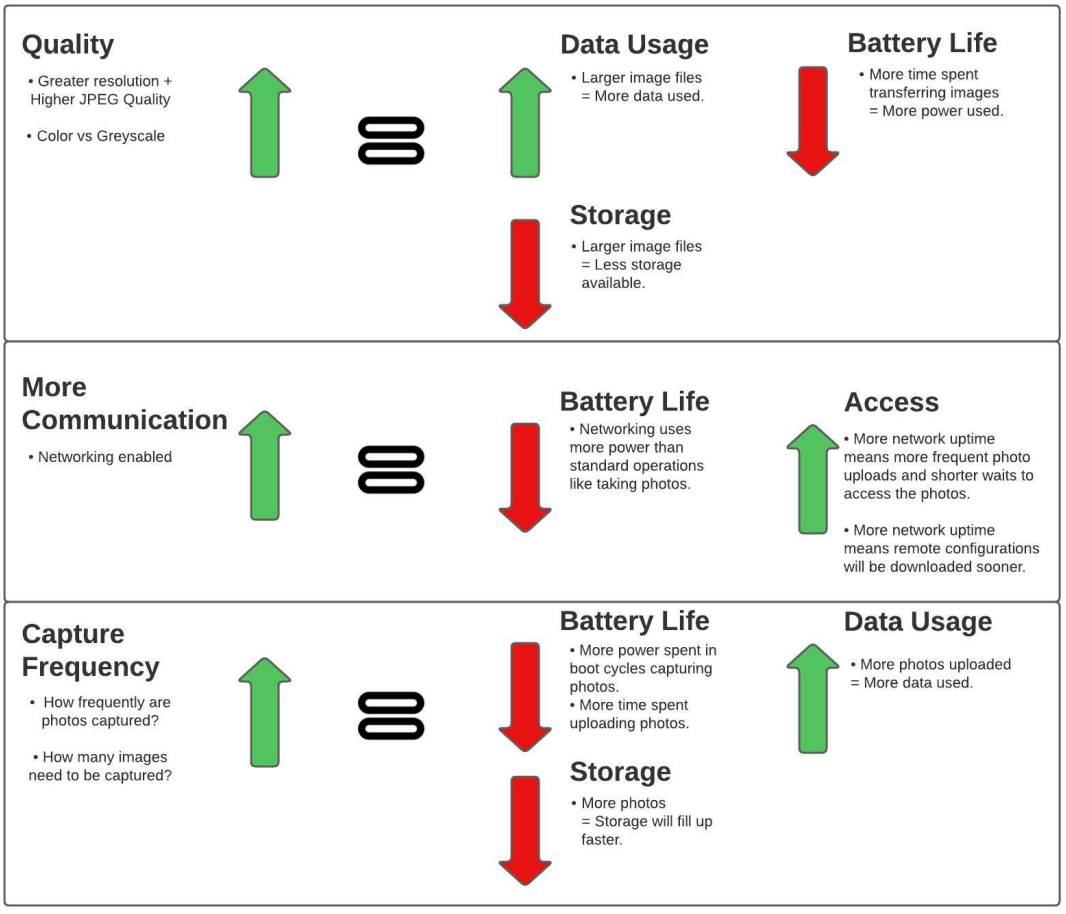


FIGURE 3.5: A summary of the trade-offs of managing the limited resources.

3.3 Requirements

Using the background research in combination with the project's goals, a list of project requirements can be produced to satisfy the demands of a remote automated photography system and improve upon the shortcomings of its predecessors.

Each requirement is also assigned a priority:

- Must: Requirement is essential for the project's completion.
- Should: Requirement should be prioritised after the must requirements are met.

Below is the project requirements table:

ID	Requirement	Priority	Justification
1	The camera system is automated.	Must	In a remote environment, there will be no manual operation. **(Section 3.1.2)**
2	The camera's data can be accessed remotely.	Must	The operator will not have to wait until the next manual access to retrieve the data. **(Section 2.2)**
3	The camera is only on when it needs to be on.	Must	The power source is limited. The camera will be wasting power if it stays on. **(Section 3.1.2 & 3.2.2)**
4	The camera can be configured to run on custom schedules.	Must	A standard feature of time-lapse cameras. **(Section 3.2.1)**

5	The camera can be configured from a remote location.	Must	Limitations of existing systems when put in a remote environment. **(Section 2)**
6	The camera can be connected to for setup.	Must	Impractical to attach a screen & a feature that most consumer-grade cameras have. **(Section 3.2.1)**
7	The camera can run independently in a remote environment off battery/solar power.	Must	See the remote location section **(Section 3.1)**.
8	The camera can run for a long period without access to solar power.	Must	See the weather section **(Section 3.1.2)**.
9	The camera implements measures to use less power.	Should	Limited Resources. **(Section 3.2.2)**
10	The camera implements measures to minimise mobile data usage.	Should	Limited Resources. **(Section 3.2.2)**
11	Camera has means to manage storage.	Should	Limited Resources. **(Section 3.2.2)**
12	Camera operator can detect failure from a remote location.	Should	Lack of remote failure detection an improvement identified in background research. **(Section 2.1.2)**

Chapter 4

Design and Justifications

This section describes the original design choices made during the project. These ideas are developed further in **Section 6.3**.

4.1 Hardware

4.1.1 Processor

The background research for this project suggests certain properties that the processor should/must have:

1. There needs to be an interface to connect a camera to the processor.
2. Power consumption should be low as the camera system will have a limited power supply.
3. The ability to run a full operating system will provide access to more tools to program custom behaviour for the camera.
4. A more compact processor is preferable as the camera system will have to be transported over long distances.
5. The processor should be relatively cheap.

A comparison can be made between similar compact processors that could be used:

Product	Price (GBP)	Operating Current*
Rock 5B RK3588 [30]	161.70	840mA - 2260 mA [19]
Rock Pi 4 B 4GB [34]	112.00	300mA [10]
ASUS Tinker Board [4]	70.99	700mA - 1000mA [4]
NanoPi M4 [26]	40.00	350mA - 1972mA [15]
Raspberry Pi 3B+ [32]	40.00	400mA [20]
Raspberry Pi Zero W [31]	15.00	120mA [3]

TABLE 4.1: A table comparing processors with similar features. **Idle Current with WiFi enabled (Higher Current = Higher Power Consumption). If a value is not found, then a range is specified.*

As shown by the table, the two Raspberry Pi models share a good balance between low cost and operating current and appear as better options than their counterparts.

The Raspberry Pis are also highly customisable in terms of software and peripheral compatibility. Another benefit of the Raspberry Pis is that hardware and software can usually be transferred between different models (e.g. If it works on 3B+, it also works on a Zero W).

At the time of writing this report, a Raspberry Pi 3B+ can easily be obtained through the university. Therefore, the **Raspberry Pi 3B+** is a sensible option for this project's prototype. The Raspberry Pi Zero can be explored later in the project when improving the prototype.

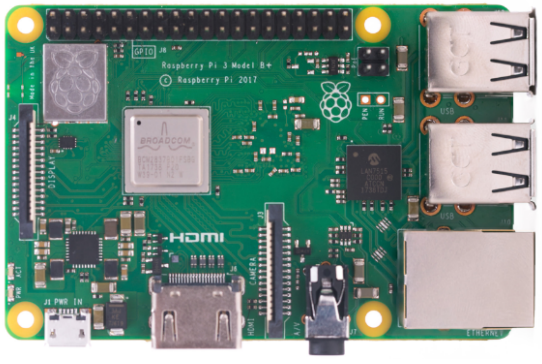


FIGURE 4.1: Raspberry Pi B3+ [32].

4.1.2 Camera Sensor

The Raspberry Pi has native support for the **Pi Camera Module v2**, making it the default choice. With an 8-megapixel Sony IMX219 image sensor, it provides

reasonable image quality. The module will only draw significant current when capturing photos in short bursts. Therefore, its power consumption is not a big concern.

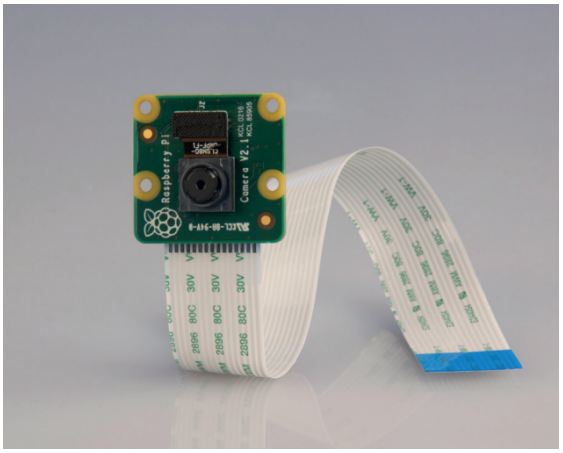


FIGURE 4.2: Pi Camera Module V2 [29].

4.1.3 Timer

As this process chose to focus on time-lapse photography, the camera system requires a method to schedule the activities that happen in the system.

4.1.3.1 WittyPi

The **WittyPi** board by UUGear lets you control the power cycles of your Raspberry Pi with an external timer [40]. It connects to a power supply and passes power to the Pi when powered on. Its timer, powered by a low-power coin cell battery, is synchronised with the current time on every boot. When the timer reaches a scheduled ON/OFF point, it will switch the Pi to the correct state if not already in it.

4.1.3.2 Scheduling Script

The WittyPi follows a script called 'schedule.wpi' to schedule its ON/OFF cycles. A schedule is structured as a loop with a defined start and stop.

The schedule script consists of a simple set of commands that can be generated with a custom script.

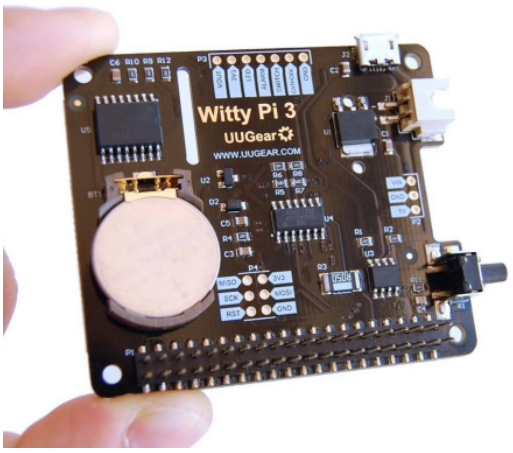


FIGURE 4.3: WittyPi [40].

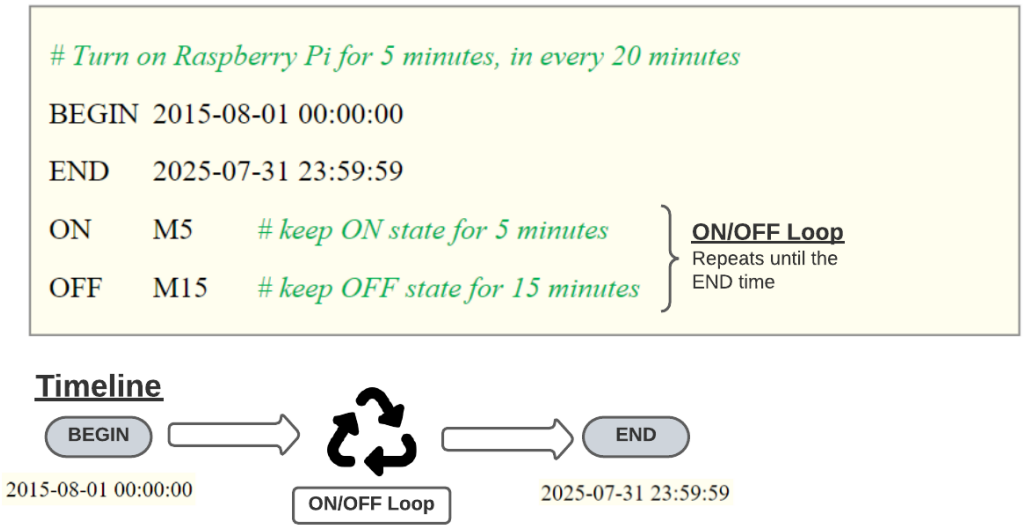


FIGURE 4.4: Demonstration of schedule script (adapted from [40]).

- BEGIN/END: start/end of loop.
- ON/OFF ⟨Length⟩: Turn on/off.
- DD HH MM (Days, Hours, Minutes).

4.1.3.3 Timer Routine

Every time the Raspberry Pi boots up, it will call a bash script called 'runScript.sh'. This synchronises the timer, tells the board when to next turn OFF/ON and runs another script called **'afterStartup.sh'**. The afterStartup.sh script instructs the Raspberry Pi what to do after completing a boot (e.g. Take a photo

after booting). Additionally, there is another script called **'beforeShutdown.sh'** that runs when the WittyPi triggers a shutdown.

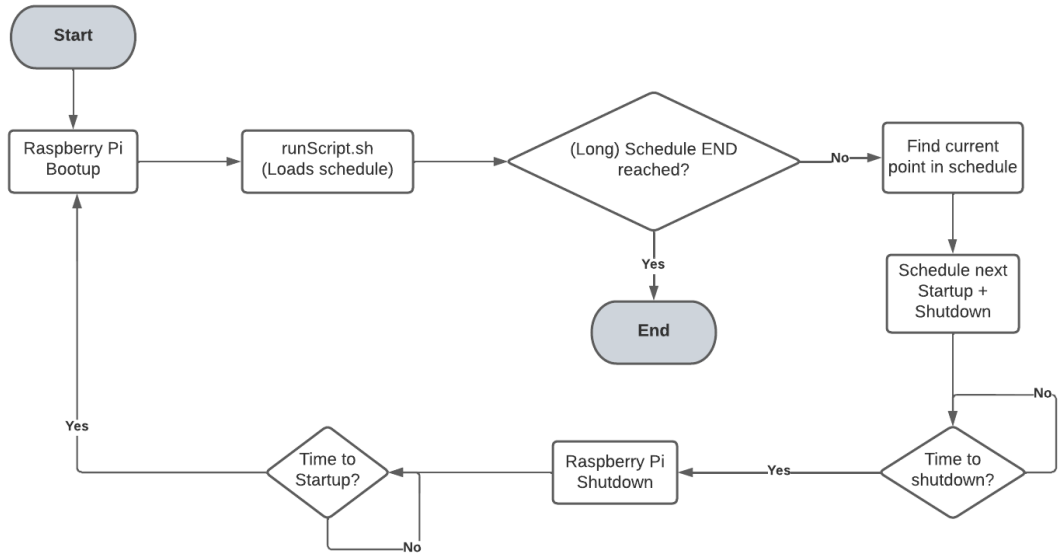


FIGURE 4.5: The timer routine flowchart (adapted from [40]).

4.1.4 Communication

Background research has shown that communication in a remote photography system:

- Allows for remote failure detection of the system.
- Direct access is not needed to the system to see results.
- You reduce the risk of losing data upon failure of storage.

4.1.4.1 Comparison of Communication Methods

Initial experimentation with a Raspberry Pi and Pi Camera Module v2 was conducted using the JPEG file format and the camera's maximum resolution of 3280×2464. This resulted in image sizes of $\sim$ 5MB, representing a worst-case scenario for comparing communication methods. Up-link speed was considered the primary network activity, as remote photography systems primarily involve uploading photos.

Method	**Peak Uplink**	**Daily Transfer Limit**	**Time to transfer 5MB (s)**
LTE	50 Mbit/s	N/A	0.8
LTE-M1	1 Mbit/s	N/A	40.0
Zigbee	250 kbit/s	N/A	160.0
NB-IoT	156 kbit/s	N/A	251.6
LoRa	21.9 kbit/s	82,125 bytes	1,826.5
SigFox	600 bit/s	1,680 bytes	66,667.0

Table 4.2: A table showing the specification of common IoT communication methods [1, 36, 22, 13].

Firstly, **these are theoretical maximum speeds, so realistically, speeds are lower**. Neither LoRa or SigFox are suitable for image transfer with strict daily transmission limits. NB-IoT and Zigbee both take over 2.6 minutes. A Raspberry Pi will have power overhead with the transmission. As the transfer speeds are relatively slow, a Raspberry Pi would waste significant power via overhead while waiting for transmission completion.

4G-LTE is the fastest of these communication methods. LTE-M1 is slower but offers power-saving modes and is within a reasonable transfer time. **It is worth investigating both for potential use.**

4.1.4.2 Choosing a Communication Method

4G-LTE and LTE-M operate on the same networks. Therefore, the choice is a question of power efficiency. The choice can be made by comparing communication hardware for the Raspberry Pi.

Hardware	**Type**	**Idle draw (mA)**	**Transmission Draw (mA)**
SIM7600E-H [41]	4G-LTE	150	300
ME910C1 [38]	LTE-M	12	190

Table 4.3: Comparison of 4G-LTE and LTE-M hardware.

Using these values, and the values from (Table 4.2), we can predict how 4G-LTE and LTE-M perform when transmitting 15MB. The predictions account for each component's power draw and transfer speeds, also assuming some idle time **(The time spent not transmitting)**.
Assumptions:

- Transmission happens at peak current draw.

- Both components operate at a constant 5V.
- $Energy(E) = Voltage \times Current \times Time$
- $E_{Total} = E_{transmitting} + E_{idle}$

Type	Upload Speed (kbps)	Time to send (s)	Transmission Draw (mA)	Total Time (s)	Upload Energy (J)	Idle draw (mA)
LTE	40000	0.017	300	0.0500	0.0750	150
LTE-M	375	107	190	320	304	12

FIGURE 4.6: Energy used for sending 15MB (3 Photos) [41].

Idle (s)	LTE Energy (J)	LTE-M Energy (J)
0	0	304
60	45	308
120	90	311
180	135	315
240	180	318
300	225	322
360	270	326
420	315	329
480	360	333
540	405	336
600	450	340
660	495	344
720	540	347

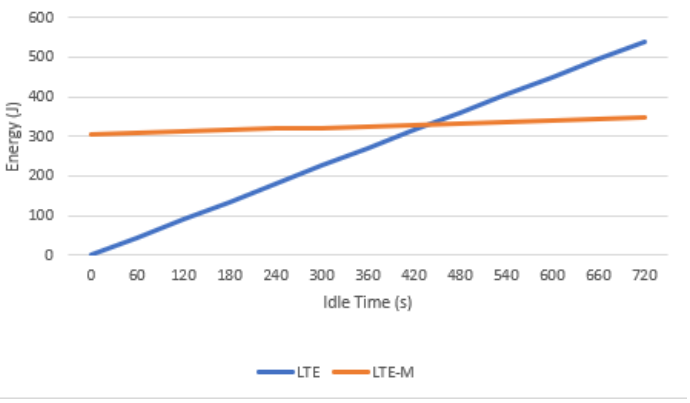


FIGURE 4.7: Graph showing how the energy consumption of 4G-LTE and LTE-M components changes with the amount of idle time.

As shown by Fig. 4.1.4.2, over longer periods of idle time, LTE-M will consume less energy due to its low idle power draw. However, as 4G-LTE uploads data much faster, it will consume less energy than LTE-M unless there is a great amount of idle time. The remote photography system should not have a lot of idle time, **so 4G-LTE is the most suitable option for the system.**

The system will use the **WaveShare SIM7600E-H 4G-HAT** [41] that will allow the Raspberry Pi this design is using to utilise 4G/LTE mobile networks.

FIGURE 4.8: SIM7600E-H 4G-HAT [37].

4.1.5 Power Source

A **lead acid** battery will be used. Lead acid batteries are suitable as they are high-capacity, physically robust, have a low self-discharge rate and aren't very sensitive to periods of being overcharged/undercharged. This makes them very suitable for long-term, unsupervised use in an outdoor environment.

FIGURE 4.9: A typical 12V 12Ah (144Wh) Lead Acid Battery [7].

There will be no opportunity to recharge the battery in a remote environment, so the camera would benefit by being able to recharge itself. Solar power is the default method, as seen in the background research. A solar charging controller will manage the power load between charging the camera's battery and powering the camera.

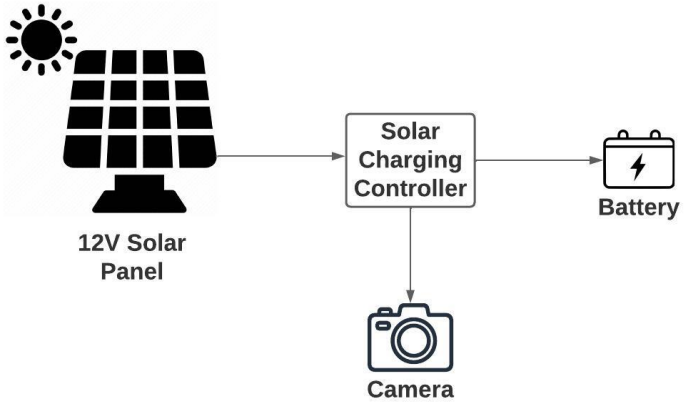


FIGURE 4.10: Solar panel configuration.

4.2 Software

4.2.1 Capturing and Uploading Photos

A software solution is needed to combine the functionality of the components.

The Raspbian OS (Raspberry Pi default) comes with the **'picamera' module already installed with Python**. Controlling the Pi-Camera Module v2, and capturing photos is trivial.

For uploading the photos, there needs to be an endpoint where the camera system uploads the photos. **AWS (Amazon Web Services)** is a good choice for this because:

- It provides access to free cloud storage.
- Its API libraries are compatible 32-Bit OSs (required to operate the PiCamera).
- AWS' 'S3' storage is flexible enough for an application to be built around it.

The 'picamera' python module would produce and store image files that the AWS libraries can upload to the cloud.

FIGURE 4.11: The planned process of capturing images and uploading them.

4.2.2 Viewing Photos

The AWS Cloud already provides an interface for viewing and downloading files that it hosts. This can be used to view the uploaded photos.

4.2.3 Camera Setup

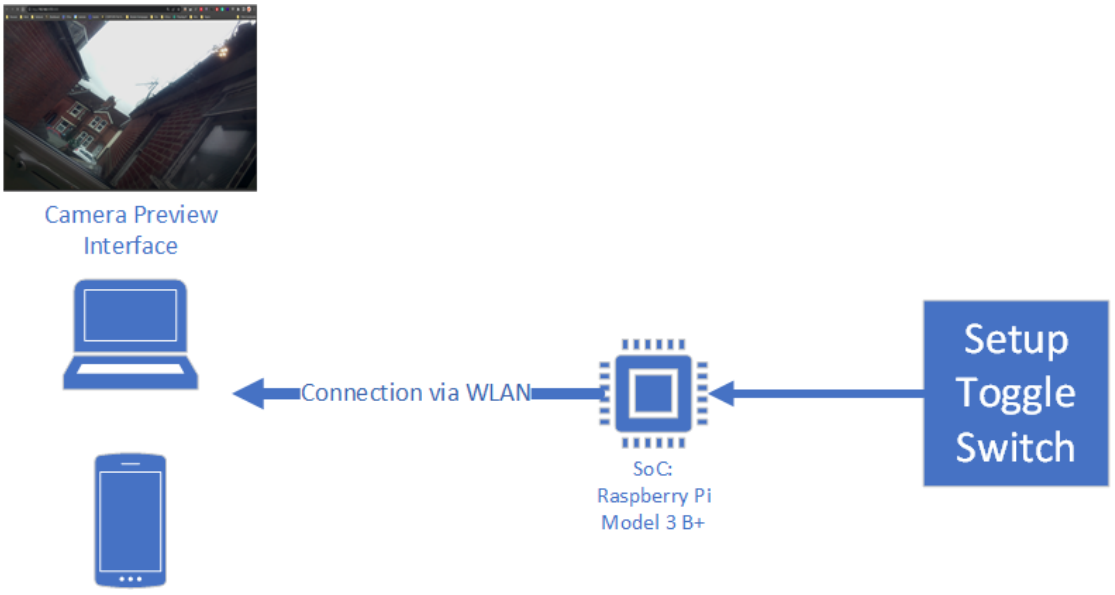


FIGURE 4.12: How an operator would view the camera output when setting it up.

For the camera setup, the operator's primary concern would be the direction the camera is pointing in. They need a way to check this.

Adding a preview screen to the hardware would increase costs and alternatives exist that do not require additional hardware.

The Raspberry Pi 3B+ has WLAN capabilities and can stream the camera's output to an operator's device [32]. **A toggle switch** would be added to toggle this camera stream, as once the camera is set up, it is no longer necessary to stream its output constantly.

4.2.4 Timer Configuration

Timer configuration will also utilise **AWS** so that an operator can change camera settings remotely. As discussed in Section 4.1.3.3, the timer that controls the photography system operates on a schedule file called **'schedule.wpi'**. The schedule file would be generated and stored on AWS through a web application that the operator uses. When the camera system is powered on, it can download updated timer schedules.

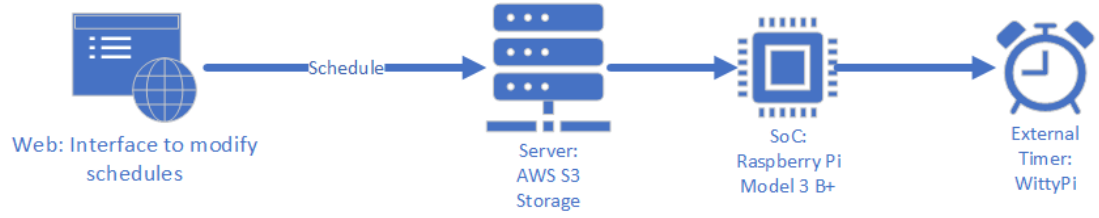


FIGURE 4.13: How an operator would change the photography timing schedule from a remote location.

4.3 System Diagrams

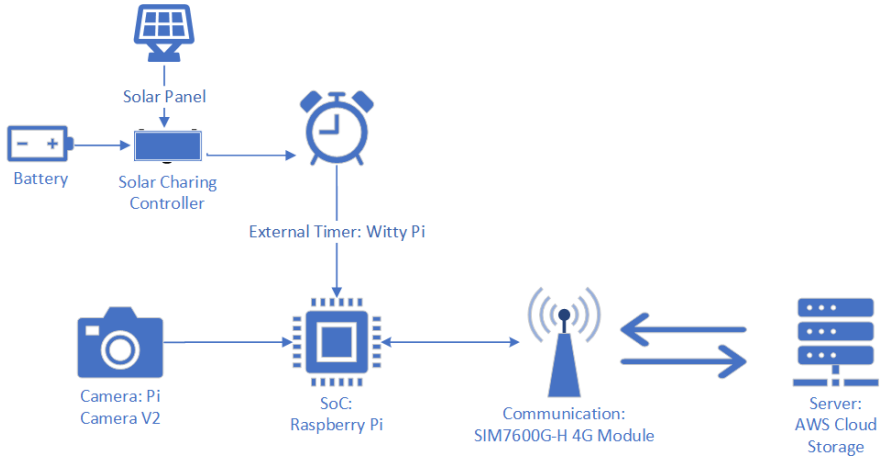


FIGURE 4.14: Diagram outlining system components.

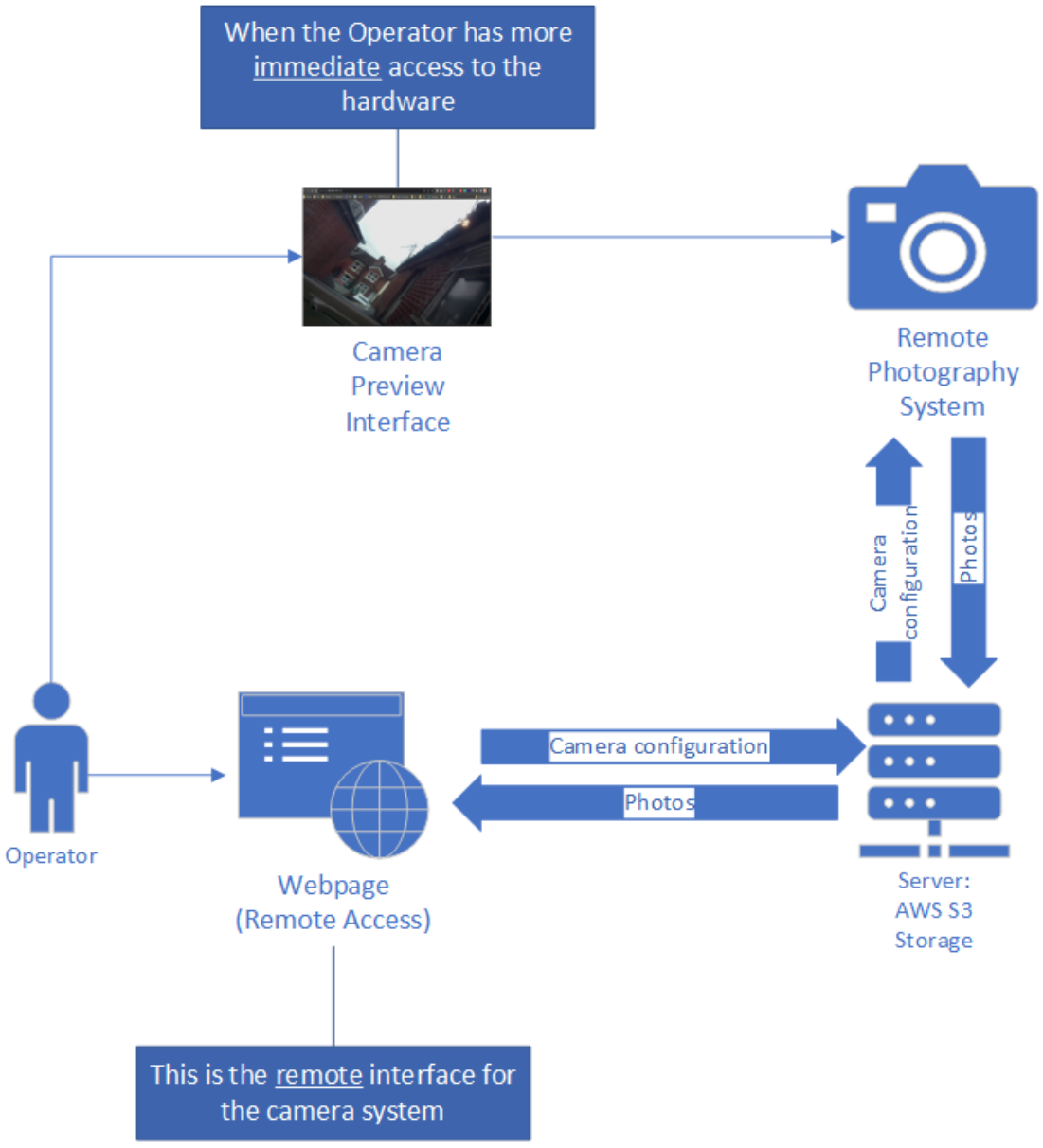


FIGURE 4.15: Diagram outlining the system data flow.

Chapter 5

Implementation

5.1 Prototype

At the time of the progress report, a prototype was produced that successfully implements goals 1, 2, 3, and 5 from the original project brief (See Table 5.1), providing a solid foundation for further development and improvement over the remainder of this project. **Section 6.1.2** details the testing and evaluation of the prototype and its influence on the project.

5.1.1 Work Completed

The goals set in the project brief can be used as a reference to assess how well the project has progressed at the checkpoint of the prototype.

Goal ID	Description	Status
1	Prototype an automated system that can take photos and transmit them.	Complete
2	From a system remote to the camera, access the image data sent by the camera.	Complete
3	Only have the system on when it needs to be on to preserve power	Complete
4	Allow for the system to be configured for different photography cycles	In progress

5	Provide means to facilitate the camera's set-up (such as checking where the camera is pointing)	Complete
6	Explore the possible ways to improve upon the prototype	In progress
7	To test the system in the field	Not started

Table 5.1: Account of current work completed using goals defined in the project brief.

The incomplete goals are worked on during the remainder of this project.

Pictures

The following are pictures of the prototype:

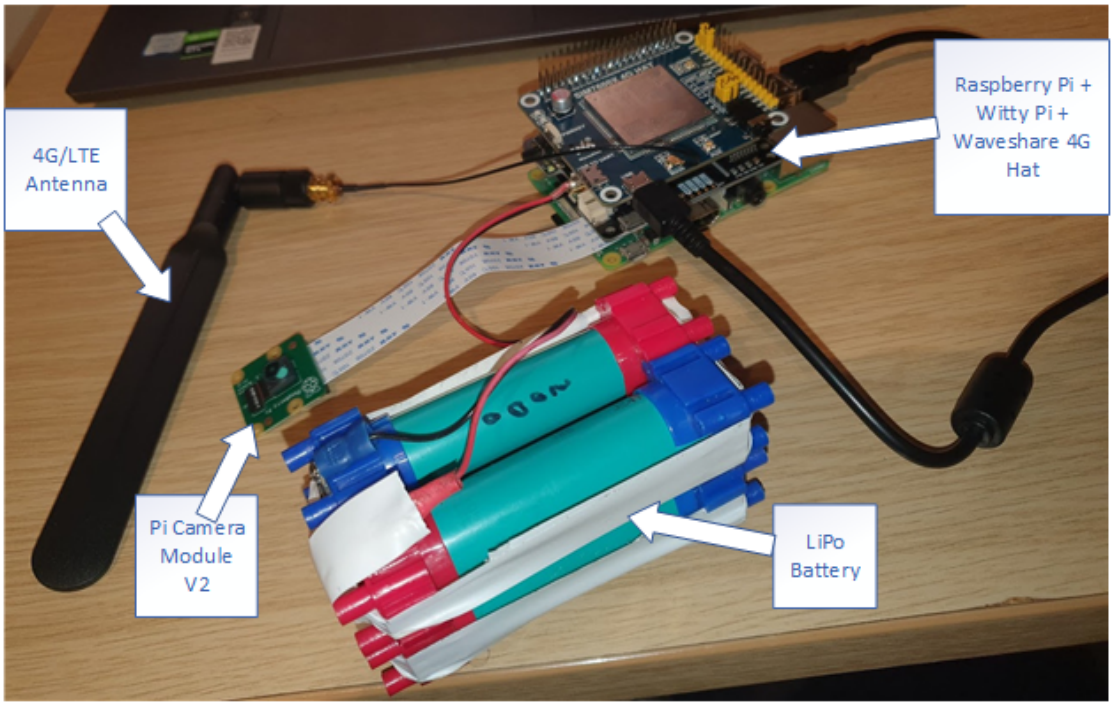


FIGURE 5.1: A demonstration of all the hardware connected up.

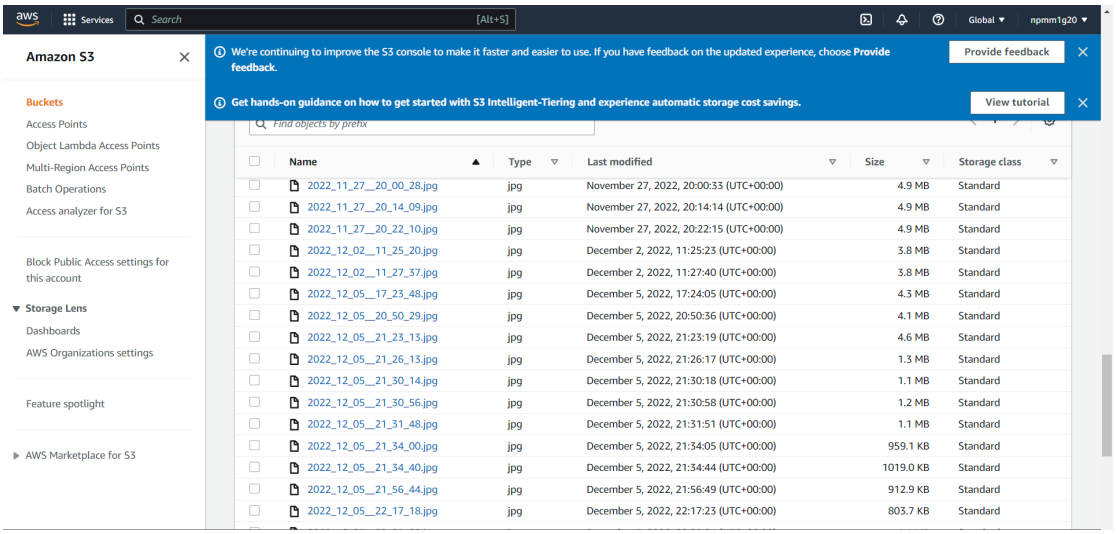


FIGURE 5.2: A photo showing the AWS dashboard showing photos that the prototype has uploaded.

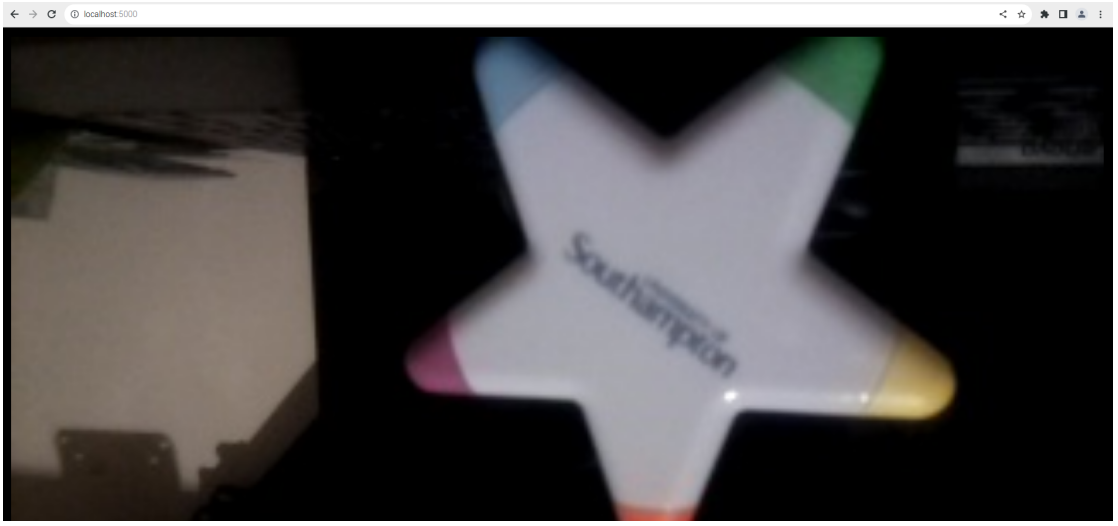


FIGURE 5.3: The camera output viewed through a web interface (for camera setup).

5.2 Refinement

This section describes the next iteration of this project's implementation.

5.2.1 Optimisations

5.2.1.1 Raspberry Pi Zero

One of the benefits of using Raspberry Pi processors is that the same software and peripherals can be transferred to other Raspberry Pi models with minimal effort.

This allows this project to consider beneficial hardware substitutions easily.

The Raspberry Pi Zero W model is an alternative model to the 3B+ that uses less power but provides the same functionality. The Zero offers less processing power. However, this project does not involve any heavy computation, meaning it may be good to trim away the overhead of the 3B+.

On a fresh install of the Raspbian Operating System, the power consumption of the two models was measured and compared during a boot and idle experiment.

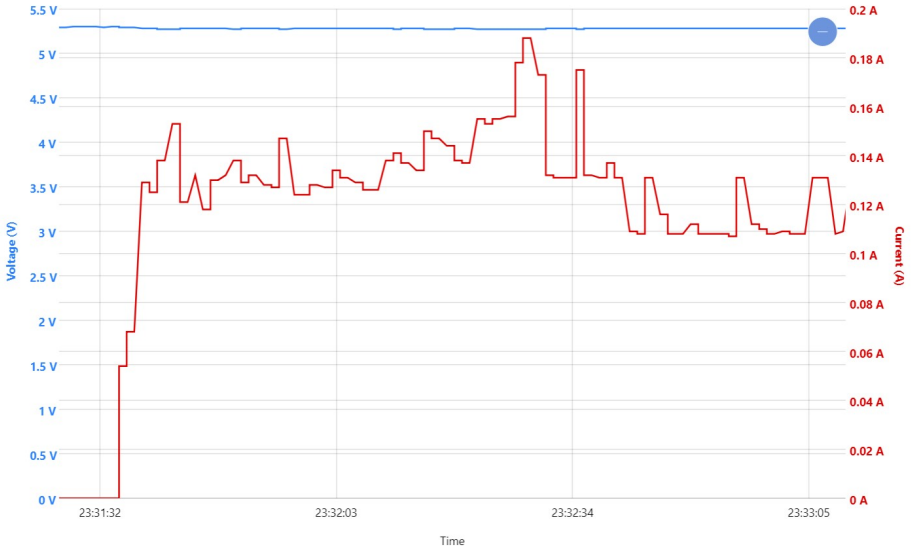


FIGURE 5.4: Raspberry Pi Zero W Power Usage.

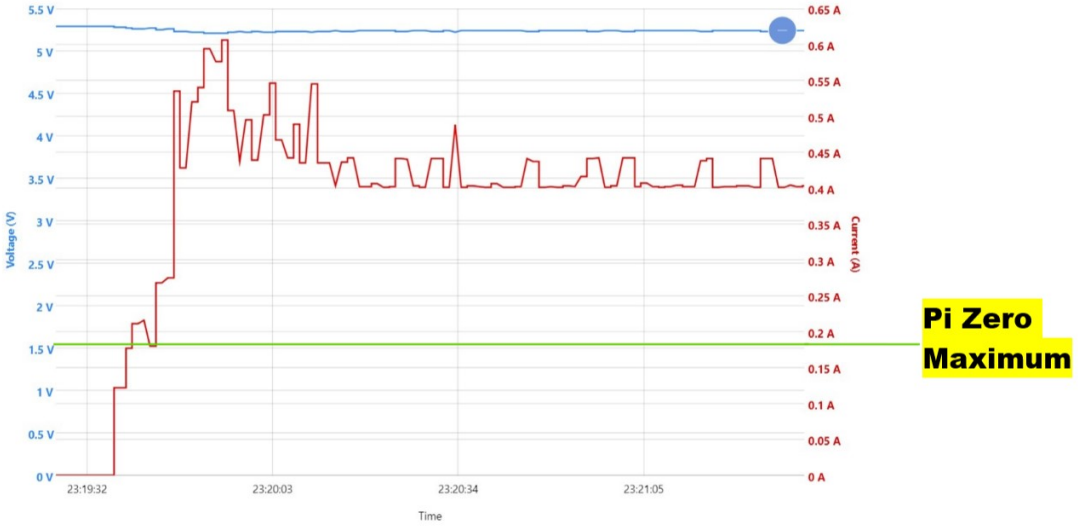


FIGURE 5.5: Raspberry Pi 3B+ Power Usage with comparison to the peak current draw for the Pi Zero.

As $Power\ Consumption = Voltage \times Current \times Time$, greater current will lead to greater power consumption. In this test, the average current of the Pi Zero W

is 129mA which is a very significant 68.9% decrease from the 3B+'s average of 416mA. This hardware substitution is clearly necessary.

5.2.1.2 Thumbnails

As implemented by the prototype, uploading every full-size JPEG captured is not resource efficient for two reasons:

- Not every photo is worth spending **mobile data** on to upload - e.g. a photo could be blurry **(see 3.1.2)**.
- Uploading unusable photos will waste **more power**.

Instead of uploading every photo in its original form, the photos can be compressed. By removing colour, altering the JPEG image quality, and reducing the resolution, the original can be transformed into a **thumbnail**. Conveniently, these thumbnails typically preserve sufficient detail to preview the original version of the photo.

Libvips is an image-processing library that is characterised by being particularly fast and resource-efficient when compared to similar libraries [21]. This is an excellent choice for producing the thumbnails as it's known for performing well on resource-constrained systems.

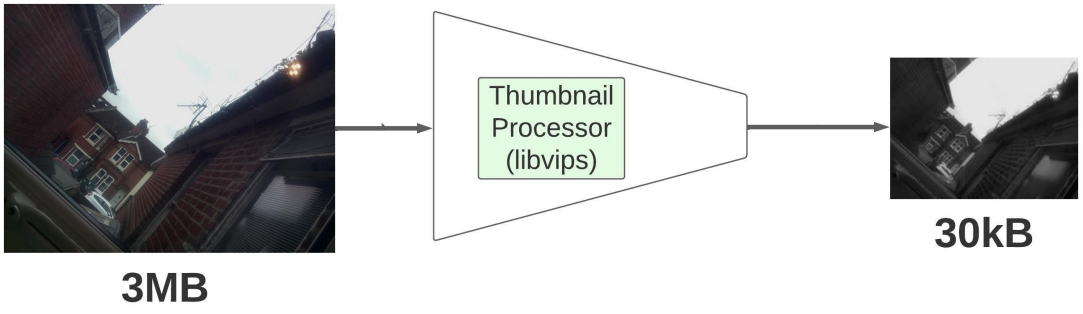


FIGURE 5.6: Reducing image file size with libvips [21].

These thumbnails allow an operator to select photos from a collection of thumbnails and then request the original version of the images from the camera (as demonstrated in Figure 5.6). This gives the operator more control over using limited resources efficiently.

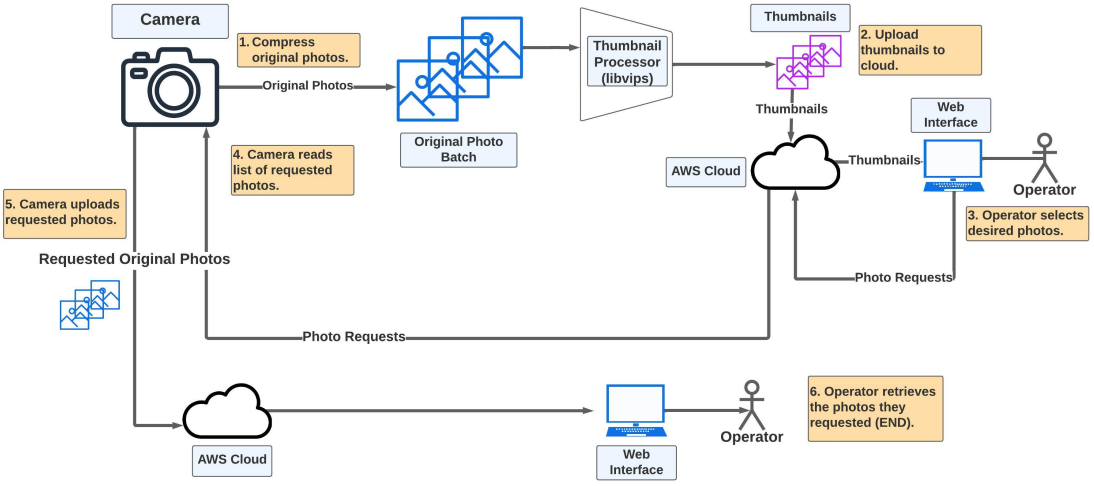


FIGURE 5.7: Diagram of the thumbnail management system.

5.2.1.3 Upload Optimisation

Initially, a naive approach to photo uploads was followed. After every photo capture, the camera system uploaded that photo. However, through a process of experimentation and comparison, batch photo uploads were determined to be more efficient.

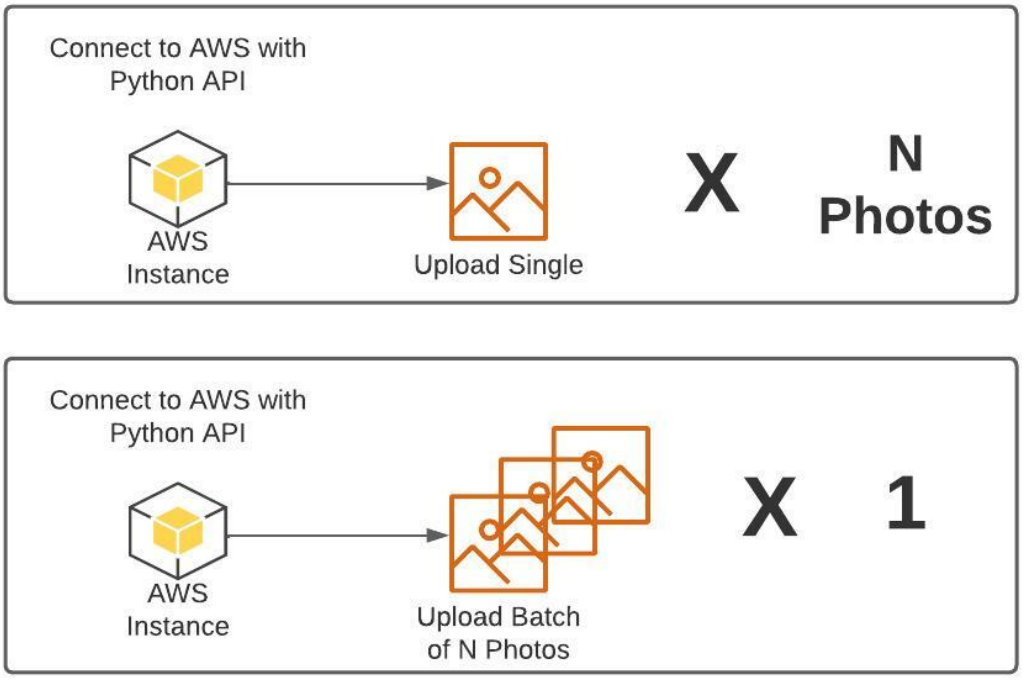


FIGURE 5.8: Comparison of single and batch photo uploads.

To compare single and batch photo uploads:

1. Produce a selection of 8 photos.
2. Measure the energy consumed and time taken to upload the 8 photos individually.
3. Measure the energy consumed and time taken to upload the 8 photos as a batch.
4. Repeat and calculate averages.

	Time (s)				Energy (mWh)			
File	**Run 1**	**Run 2**	**Run 3**	**Avg**	**Run 1**	**Run 2**	**Run 3**	**Avg**
1	2.05	2.09	1.93	2.02	3	2	2	2
2	2.07	1.89	1.94	1.97	3	2	2	2
3	2.58	1.84	1.83	2.06	2	2	1	2
4	1.80	2.53	1.78	2.01	1	3	1	1
5	2.21	2.05	2.13	2.13	2	2	2	2
6	2.24	2.91	2.10	2.39	2	3	2	2
7	2.83	1.96	1.86	2.18	2	2	2	2
8	2.06	1.84	1.84	1.91	3	2	1	2
			Total	16.66				16

TABLE 5.2: Time taken and energy consumed sending **single photos**.

Run	**Time Taken (s)**	**Energy (mWh)**
1	5.67	5
2	5.64	5
3	5.80	5
4	5.89	5
5	5.77	5
Avg	5.75	5

TABLE 5.3: Time taken and energy consumed sending photos in **8 photo batches**.

The experiments show a 65% decrease in time taken and a 69% decrease in the energy consumed by uploading the eight photos as batches instead of one by one.

- A power/time cost is associated with connecting to the AWS cloud services. This cost is repeated for each photo when uploading photos individually, but multiple photos share this cost when uploading photos in batches. Hence, uploading in batches will consume less power.

- In addition, if photos are only uploaded after every n photos that are captured, then the camera system will not have to network and use more power for $n - 1$ power cycles. This will lead to more power savings.

5.2.1.4 Modes

The camera prototype started with a basic execution model where the same task of capturing and then uploading was performed every power cycle of the camera. (See Figure 4.11)

With the optimisations of batch uploads and thumbnails, the camera will need a way of not performing the same tasks every power cycle.

At the start of a power cycle, the camera will automatically determine the required tasks in that cycle and then perform them.

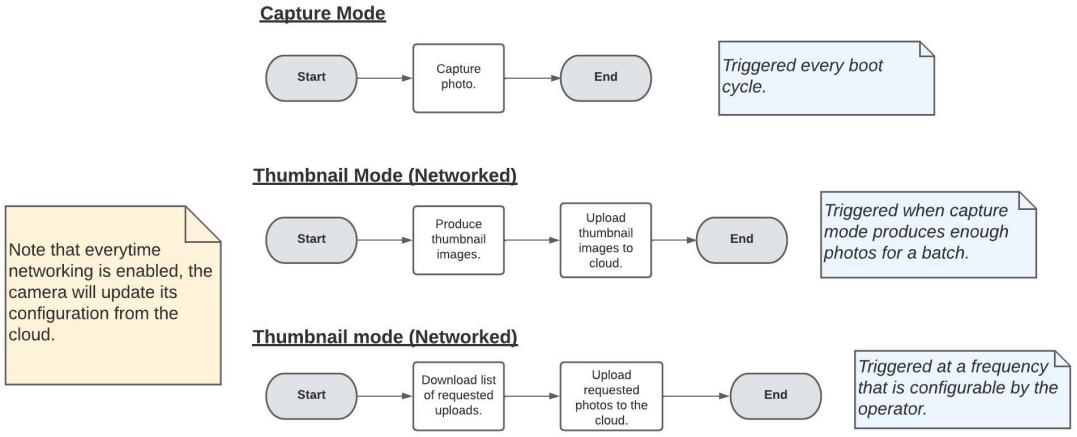


FIGURE 5.9: The different modes for the camera.

5.2.1.5 Operating System

By default, a Raspberry Pi operating system will run a large group of background tasks (**services**).

Most of these tasks will increase boot-up time and idle load on the Raspberry Pi's processor. Long boot times and more significant idle processor load mean greater power usage. Power inefficiencies will accumulate over several power cycles.

Section 5.2.1.4 discussed how the camera doesn't perform the same operations every time and, consequently, the operating system will not need the same services every time (e.g. It won't need anything to do with networking when it's only capturing a photo).

Service	Purpose
dhcpcd.service	Network Configuration
ssh.service	SSH Setup
keyboard-setup.service	Keyboard Configuration

TABLE 5.4: A few examples of Linux OS services.

We can analyse the boot time impact of each service by using the Linux **'systemd-analyse' and 'systemd-analyse blame'** commands.

```
pi@raspberrypi:~ $ systemd-analyze blame
30.384s dhcpcd.service
13.021s dev-mmcblk0p2.device
 9.863s hciuart.service
 6.135s raspi-config.service
 5.938s rng-tools-debian.service
 5.924s avahi-daemon.service
 5.592s systemd-logind.service
 5.425s systemd-udev-trigger.service
 5.332s polkit.service
 4.985s keyboard-setup.service
 4.203s systemd-rfkill.service
 4.085s dphys-swapfile.service
 3.986s systemd-journald.service
 3.761s ModemManager.service
 3.515s rc-local.service
 3.288s e2scrub_reap.service
 3.199s ssh.service
```

FIGURE 5.10: 'sytemd-analyze' command breakdown of how much boot time each service uses.

These services can be **enabled/disabled to implement the different camera modes**. The mode can then be automatically selected to suit the demands of the camera.

Implementing the camera modes on an operating system is an effective optimisation. A streamlined OS mode for capturing photos contributes to 44% boot time savings and 43% power savings compared to an OS mode that enables networking for photo uploads.

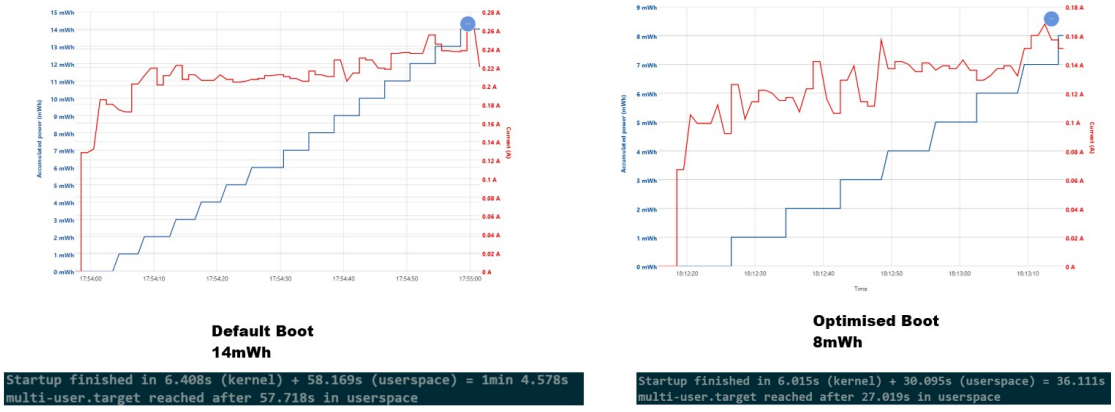


FIGURE 5.11: Comparison of boots before and after optimisation.

Additionally, the idle current draw is taken down from 110mA to 92mA (16% decrease) when we consider the time period where the boot has finished and the current stabilises.

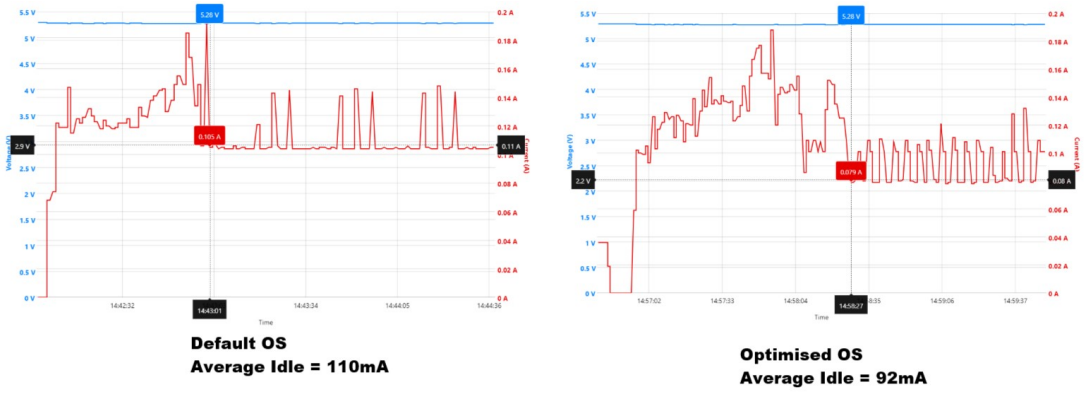


FIGURE 5.12: Comparison of idle current draws before and after optimisation.

A camera capture happens every cycle, and networking is only typically needed every few cycles, so that means these savings are made across the majority of the camera's power cycles.

5.2.1.6 Network Configuration

For this project, we are mainly concerned with the network's upload speed because the camera will be uploading several photos and won't be downloading anything

other than the small messages it uses to operate. The better the upload speed, the less time the camera will have to spend uploading images and the less time it will have to spend consuming power.

The Waveshare 4G Module (see Section 4.1.4.2) has different network operating modes that can be changed and investigated via a text-based serial interface.

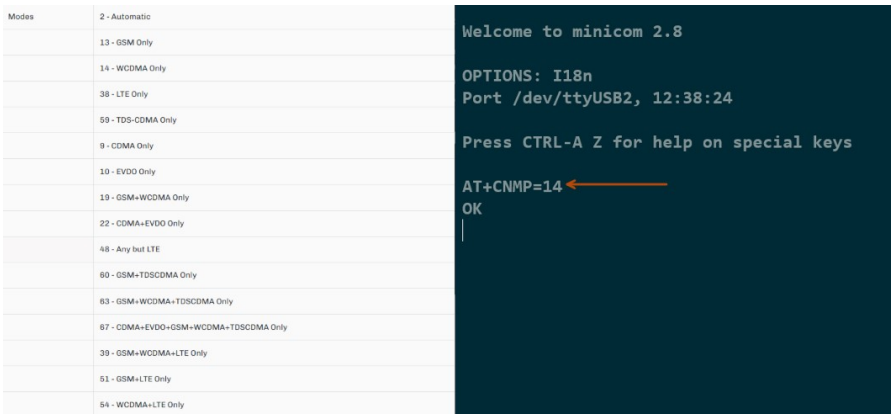


FIGURE 5.13: The left-hand side shows the different networking modes (each mode is given a number) [41]. The right-hand side demonstrates changing the mode using a serial interface.

The **speedtest-cli** tool can measure network speeds and performance to compare the different modes.

- Due to the inconsistent nature of networking, repeats and averages were taken.
- The external server that speedtest-cli uses for the test was fixed for a fair test.

	Upload Speed (Mbit/s)						
Mode	**1**	**2**	**3**	**4**	**5**	**Avg**	**Range**
LTE (38)	0.79	0.81	0.78	0.88	0.78	0.81	0.10
WMDA (14)	4.56	4.98	4.29	4.20	4.57	4.52	0.78
Auto (2)	4.97	7.57	5.15	4.65	2.68	5.00	4.89

TABLE 5.5: Upload speeds with different network modes.

The observations from this experiment are:

- Upload speeds are unstable for Automatic (Mode 2) and have a larger range than other modes.

- LTE appears to be optimised for download traffic - i.e. download speeds are high but upload speeds are not (see Figure 5.14).
- **WMDA (Wideband Code Division Multiple Access) is the best configuration** as upload speeds are more consistently high compared to its counterparts.

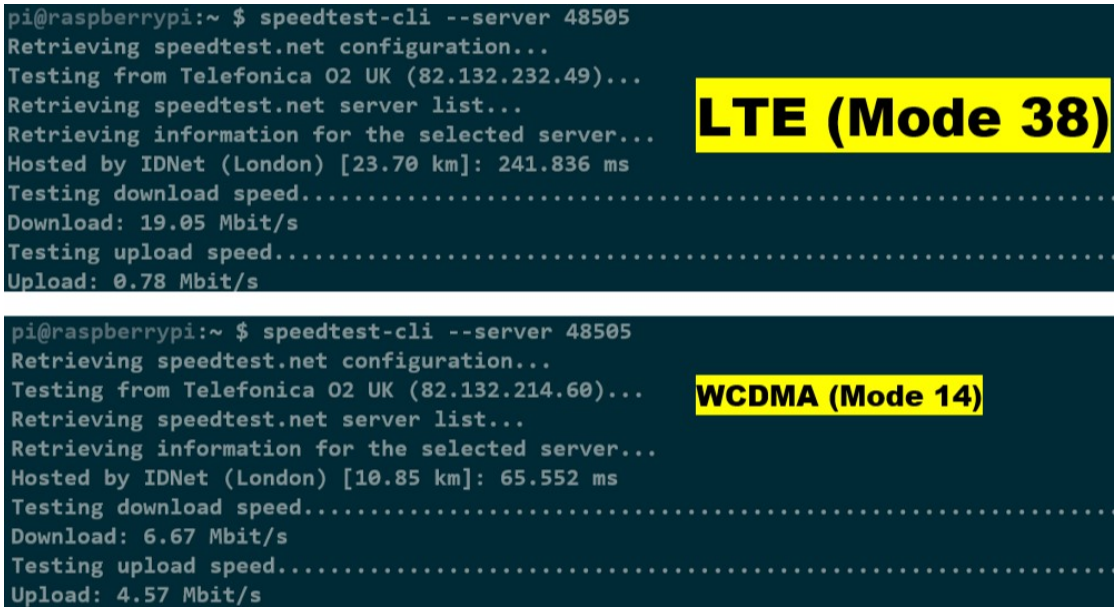


FIGURE 5.14: Examples of internet speed tests using speedtest-cli.

5.2.2 Deployment

For setting up this camera, the camera is connected to via Wifi and the output of the camera is streamed to a separate device like a phone or laptop. Eben Kouao's code facilitates this by streaming a Raspberry Pi's camera feed through a flask server [11].

In addition to the camera modes in Section 5.2.1.4, there is a setup mode that enables the camera stream server. This setup mode is simply triggered by toggling a physical switch attached to the camera system.

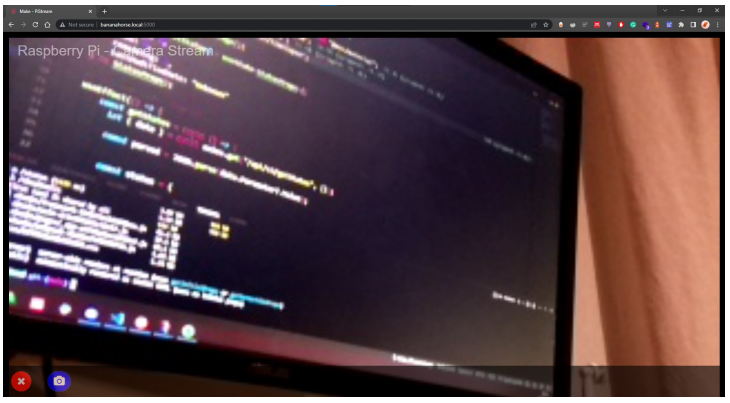


FIGURE 5.15: The streamed output of the camera.

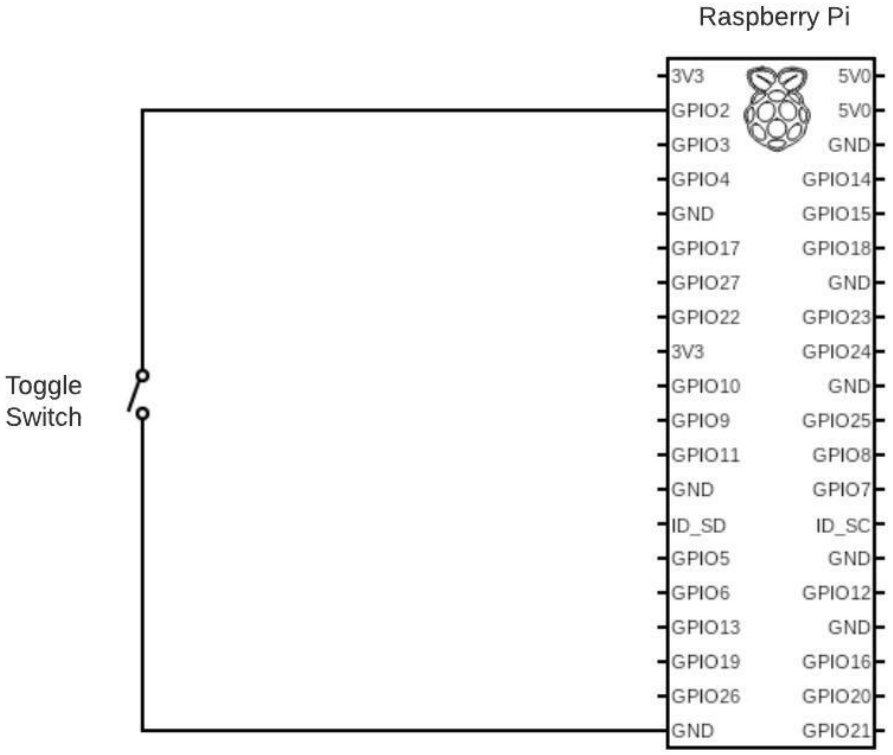


FIGURE 5.16: How the setup toggle switch is connected to the camera system.

5.2.3 Web Interface

To interface with the camera system remotely, a NextJS web application was built using TypeScript [28]. AWS cloud services bridge the connection between the web application and the camera system's hardware. (See Figure 4.15)

5.2.3.1 Schedule Customisation

The scheduling page allows the operator to customise the on/off cycles of the camera with a resolution of seconds. The camera synchronizes with this schedule when it next has networking enabled.

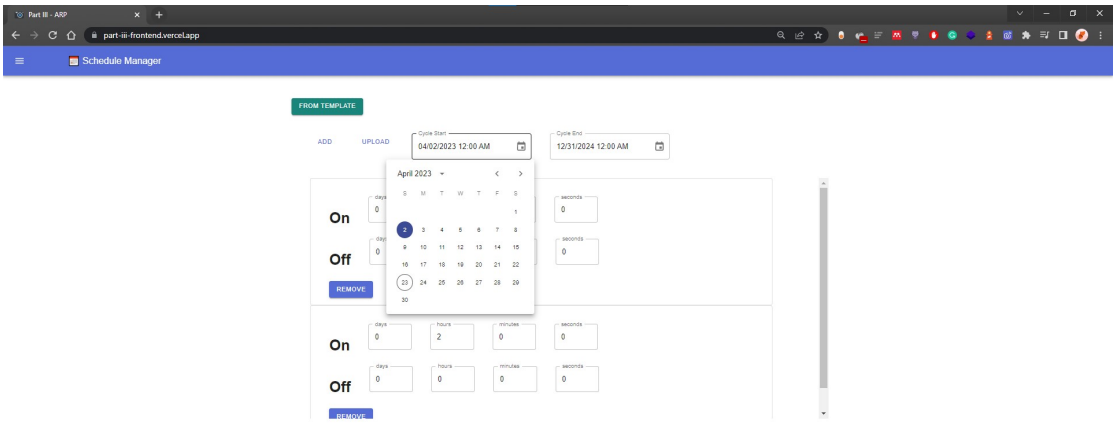


FIGURE 5.17: Interface that generates the schedules that control the on/off cycles of the camera (see Section 4.1.3).

By default, the interface allows the operator to build the schedule in the WittyPi's schedule format. However, to make operation more natural, the interface also provides a way to easily create a daily schedule (Section 4.1.3).

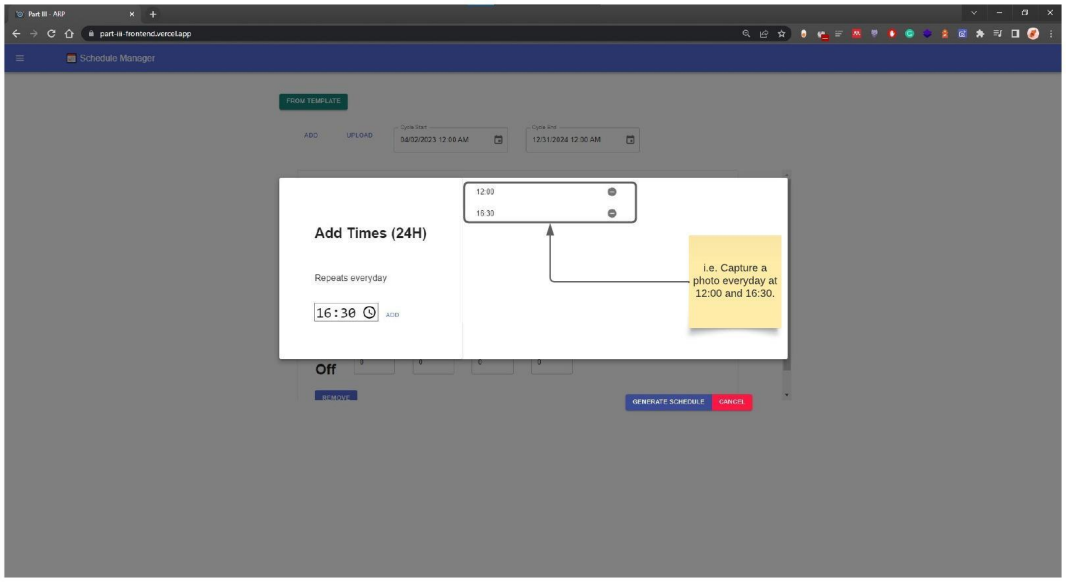


FIGURE 5.18: An interface that simplifies the creation of schedules.

5.2.3.2 Camera Monitoring

The status page allows the camera operator to remotely obtain operation data about the camera:

- The input voltage (e.g. a low voltage would indicate a low battery).
- The storage space left on the camera.
- The date and time the camera last communicated with the cloud.

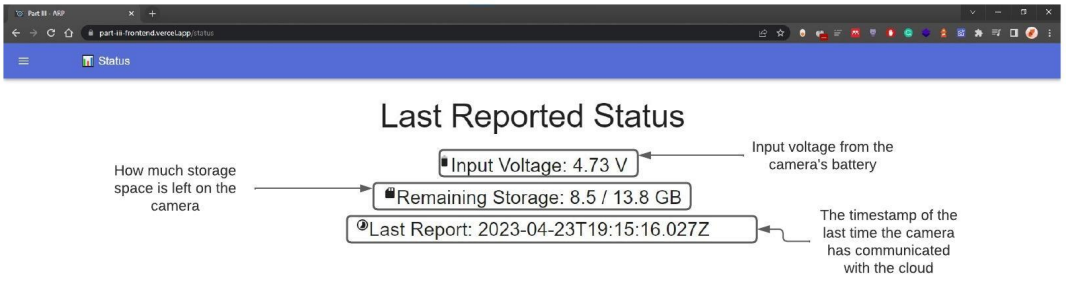


FIGURE 5.19: A page showing the camera's operating conditions.

5.2.3.3 Thumbnail Management

This page allows the operator to view image thumbnails, request full-quality images and also manage the storage on the camera (explained in Section 5.2.1.2).

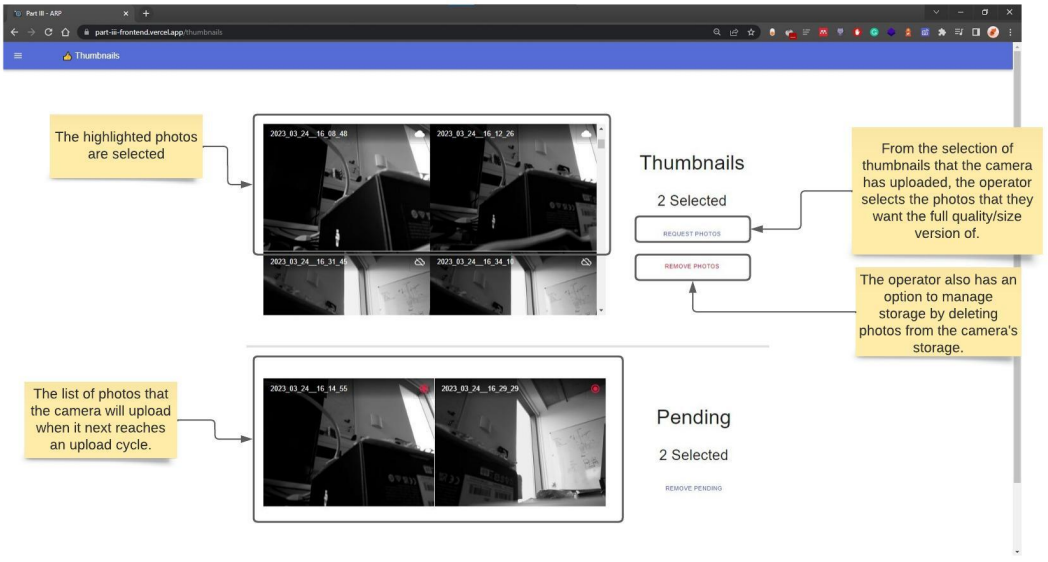


FIGURE 5.20: The thumbnail management interface.

5.2.3.4 Browsing Images

This page allows the operator to browse the images that the camera uploads.

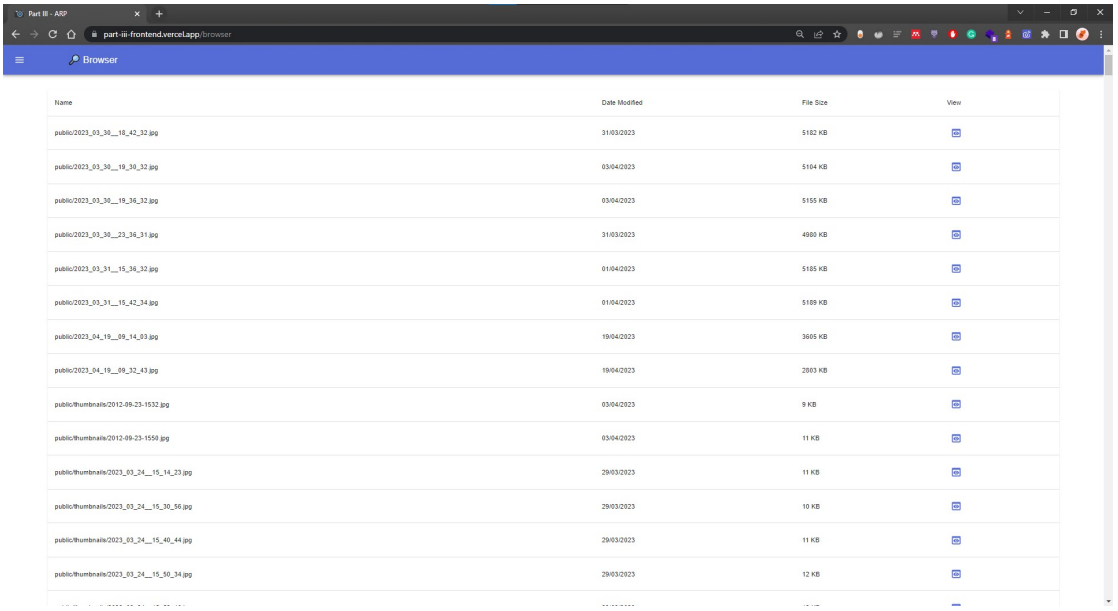


FIGURE 5.21: The image browser.

5.2.4 Photos of Refined Implementation

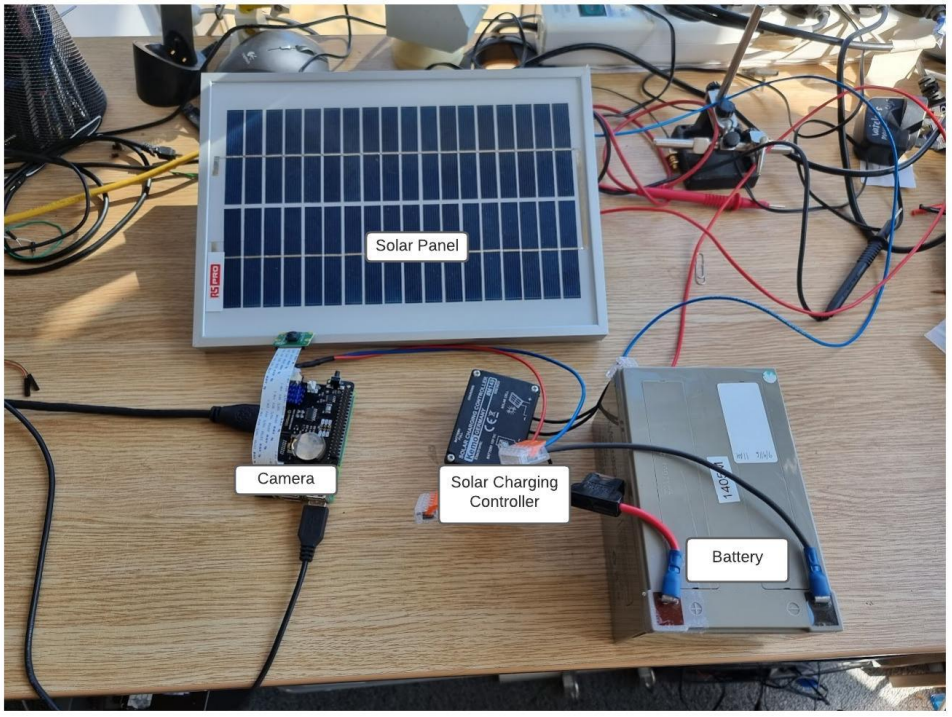


FIGURE 5.22: A picture of the camera system connected to solar power.

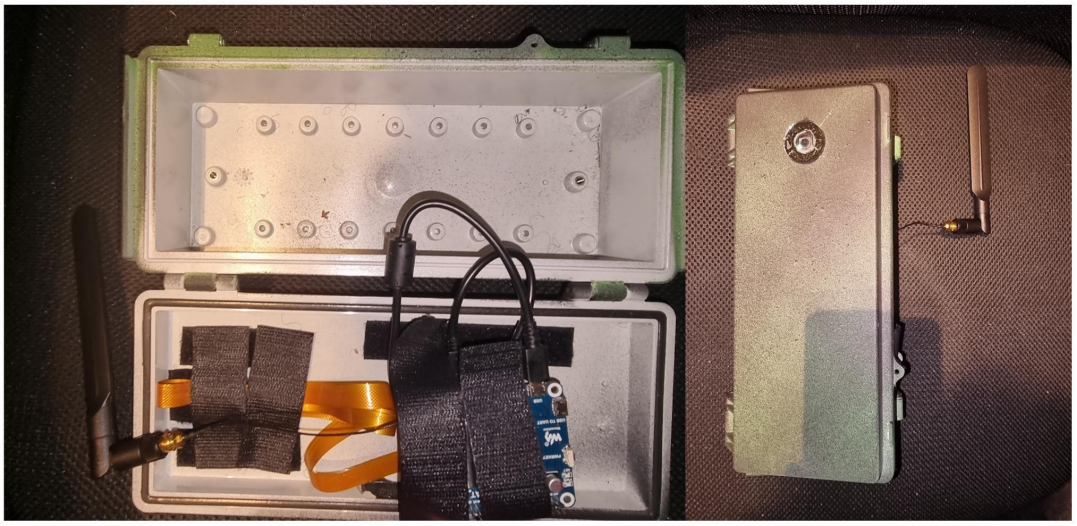


FIGURE 5.23: A picture of the camera system within a protective enclosure.

Chapter 6

Testing and Evaluation

6.1 Power Analysis

This remote photography system will run on battery power and not rely on mains power. Through background research, it is clear that power consumption must be carefully managed to ensure optimal operation. To achieve this, the power consumption of various system tasks will be measured and optimised.

These power measurements:

- Provide a quantitative measure for the success of this project.
- Highlight parts of the camera system that can be optimised.
- Demonstrate the limitations of this project's implementation.
- Suggest a power budget, informing choices such as required battery size.

6.1.1 Method

Using a power tester, I can measure the current, voltage and power that is passed into the system. The tester can be connected between the Raspberry Pi's power supply and the Raspberry Pi.

Power Supply → Power Tester → Raspberry Pi

FIGURE 6.1: Innovateking-EU A3/A3-B Power Tester (reproduced from [16]).

To get a general idea of consumption, the prototype's activities were divided into 6 different tasks that were run 3 times to obtain an average consumption.

- Measure the Energy/Capacity used in mWh ('run' column in Table 6.1.2).
- To calculate the energy consumed:

$$\textbf{Energy}_{\textbf{Consumed}} = \textbf{Voltage} \times \textbf{Current} \times \textbf{Time}$$

- For the prototype a full cycle consumption is a sum:

$$\textbf{Energy}_{\textbf{Full Cycle}} = E_{\textbf{Boot}} + E_{\textbf{Idle}} + E_{\textbf{Capture and Upload}}$$

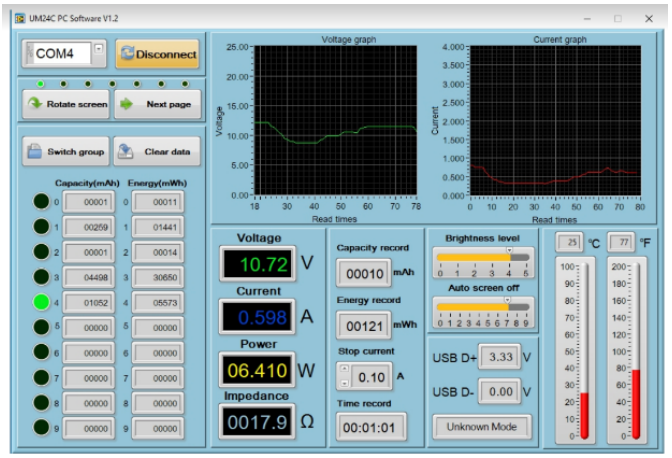


FIGURE 6.2: The software used to monitor power usage (reproduced from [16]).

6.1.2 Prototype Measurements

The results of the measurement

		Run (mWh)		
Activity	**1**	**2**	**3**	**Mean**
Boot	13	14	13	13.3
Idle (30s)	15	15	15	15.0
Capture	5	4	4	4.3
Capture and Upload	52	50	45	49.0
Camera Setup (30s)	29	29	28	28.7
Full Cycle	90	70	84	81.3

TABLE 6.1: Power consumption of the different tasks the prototype can perform. A full cycle does not include camera setup.

Discussion

The key takeaways from this power analysis are:

- According to Geerling's benchmark [20], the current consumption of a Raspberry Pi B3+ at idle should be 1.9W. Idling for 30s comes to roughly 16mWh, which is comparable to Geerling's results. Idle consumption could be reduced by controlling background programs on the Raspberry Pi. It might be worth testing other Pi models that have a lower current draw.
- The boot consumption is comparable to idling for 30s. Reducing the boot time could be a good way to reduce power consumption.
- Image capture does not consume a lot of power. Image upload contributes significantly to power consumption, making it a key area for power optimisation. This could be achieved with techniques such as compression, batch image uploads and pre-processing (e.g. rejecting over-exposed images).
- The power used for uploading photos varies a lot. The number of upload periods should be reduced to reduce the chance of potentially costly uploads.

6.1.3 Refined Implementation Measurements

Using the same methods as described in Section 6.1.2, the power consumption of the refined camera implementation was measured.

For the prototype, the focus was on measuring sub-activities to find out the specific parts of the camera needing power optimisation. This iteration focuses more on consolidating these sub-activities into measurements that can be used to estimate a power budget for when this camera is deployed.

			Run (mWh)			
Activity	**1**	**2**	**3**	**4**	**5**	**Mean**
Setup 30s	2	2	2	2	2	2.0
Capture Cycle	13	13	14	14	14	13.6
Thumbnail Cycle (1 Capture + Produce & Upload 5 Thumbnails)	44	47	47	47	46	46.2
Upload (2 Photos ∼8MB)	38	38	55	39	38	41.6

TABLE 6.2: The power measurements for the refined implementation of the project.

6.1.4 Power Budgets

A power budget can be determined for the camera system using the power measurements. **Note that these values assume 100% efficiency. Some power will be lost during operation leading to greater power consumption in reality.**

	Cost (mWh)	**Times a week**	**Total Cost (mWh)**
Capture and Upload	81.3	6	487.8

Total:	488	***per week***
	x	
	61	***per year***
	=	
	29756	**mWh**

TABLE 6.3: A power budget derived from the prototype's power measurements (**see Section 6.1.2**).

The only operation the prototype could run is capturing and uploading a photo immediately, so a sequence of 6 captures (1 per day) is considered.

	Cost (mWh)	Times a week	Total Cost (mWh)	
Capture Cycle	13.6	4	54.4	
Thumbnail Cycle (5 Thumbnails)	46.2	1	46.2	
Upload Cycle (2 Photos)	41.6	1	41.6	
	Total:	142		*per week*
		x		
		61		*per year*
		=		
		8674		**mWh**

TABLE 6.4: A power budget derived from the refined implementation's power measurements (**see Section 6.1.3**).

For comparison's sake, a 6 photo sequence over 6 days is considered for the refined implementation. This includes a thumbnail cycle and an upload cycle (With 2 selected photos).

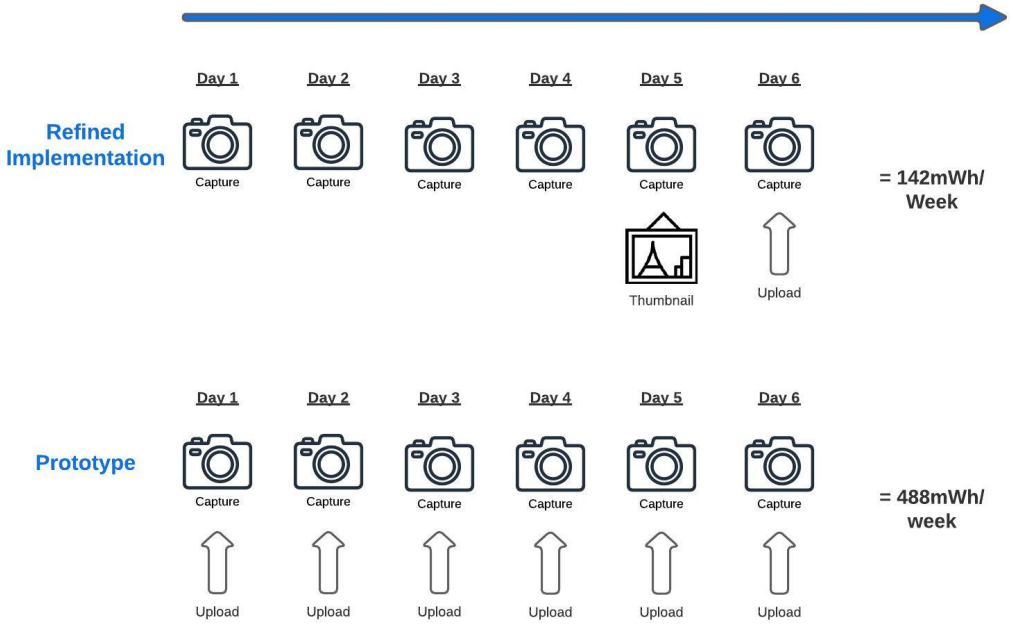


FIGURE 6.3: The two sequences visualised.

6.1.5 Power Evaluation

Firstly, the refined implementation consumes about 8700mWh = 8.7Wh (assuming 100% efficiency). Therefore, a typical battery can easily provide for this with a significant amount of padding (see the 144Wh battery in Figure 4.9). This also means that in terms of battery life, this project's implementation meets the standards set by consumer-grade time-lapse cameras (see Section 3.2.1).

Secondly, comparing the prototype to the refined implementation over 366 days shows a **71%** decrease in power consumption going from the prototype to the refinement. This indicates that the power efficiency improvements made during the project have been successful.

Additionally, the initial prototype power analysis directly motivated several key design choices (Section 5.2.1):

1. Changing to the Raspberry Pi Zero W.
2. Decreasing boot times.
3. Trying different upload methods (i.e. different upload frequency, batches).
4. Utilising image compression.

This shows that this prototyping strategy was effective overall.

Finally, the prototype's power cost is quite expensive before considering increasing demands such as increasing the frequency of the photography. This would make the system reliant on having a large battery and reasonable access to solar power - with access to solar energy being limited during the winter (see Figure 3.3). The improved implementation significantly reduces power consumption and provides more freedom to increase the camera system's demands by increasing the frequency of capturing and uploading photos.

6.2 Long Test

The motivation for the extended test is to ensure the camera system can run correctly for a significant amount of time without manual operation. The camera was programmed to perform a power cycle every 20 minutes.

To monitor the long test:

- Logging was added to the code that runs on the camera. Both regular events and error events were recorded and labelled so that they were easy to identify.
- The web interface for the camera was regularly checked to check that the camera could still communicate with the cloud.

```
2023-04-01 17:00:33,291 INFO: Internet connection established
2023-04-01 17:00:34,667 INFO: Capture mode
2023-04-01 17:00:36,193 INFO: Captured image: /home/pi/Github/indep_proj/in/2023_04_01__17_00_34.jpg
2023-04-01 17:00:36,194 INFO: Update mode
2023-04-01 17:00:36,526 INFO: Uploaded Full Images: []
2023-04-01 17:20:32,073 INFO: Starting main script
2023-04-01 17:20:32,100 INFO: Internet connection established
2023-04-01 17:20:33,433 INFO: Capture mode
2023-04-01 17:20:34,887 INFO: Captured image: /home/pi/Github/indep_proj/in/2023_04_01__17_20_33.jpg
2023-04-01 17:20:34,888 INFO: Thumbnail mode
2023-04-01 17:20:41,018 INFO: Uploaded Thumbnails: ['out/2023_04_01__16_40_31.jpg', 'out/2023_04_01__16_00_31.jpg',
2023-04-01 17:40:49,139 INFO: Starting main script
2023-04-01 17:40:49,158 ERROR: No internet connection, exiting
```

FIGURE 6.4: An extract from the camera logs.

Evaluation

In ideal operating conditions, the camera behaved as it was designed to.

However, when the camera could not obtain a network connection, the camera was not initially programmed to handle the issue. To fix this, two patches were added to the code.

- Thumbnails that could not be uploaded have their upload delayed until the next thumbnail cycle.
- Full photo upload requests that could not be completed are now ignored and will remain as requests. The camera system will naturally resolve these requests when it can.

Overall, it was shown that this testing strategy helps identify the unexpected behaviour of the camera. Ideally, this test should be run longer to force more unexpected behaviour so that solutions are found before deploying the camera.

6.3 Requirement Testing

To assess the completeness of the implementation, the requirements in **Section 3.3** can be referred to.

Test Table

Table 6.5: The list of requirements, their completion status and references to evidence.

ID	Requirement	Category	Met	Evidence
1	Camera system is automated.	Must	Yes	**(Section 5)**
2	Camera's data can be accessed remotely.	Must	Yes	**(Section 5.2.3)**
3	Camera is only on when it needs to be on to preserve power.	Must	Yes	**(Section 5.2.1.4)**
4	Camera can be configured to run on custom schedules.	Must	Yes	**(Section 5.2.3)**
5	Camera can be configured from a remote location.	Must	Yes	**(Section 5.2.3)**
6	Camera can be connected to for setup.	Must	Yes	**(Figure 5.3)**
7	Camera can run independently in a remote environment off battery/solar power.	Must	Yes	**(Figure 5.22)**

8	Camera can run for a long period without access to solar power.	Must	Yes	**(Section 6.1.4)**
9	Camera implements measures to use less power.	Should	Yes	**(Section 5.2.1)**
10	Camera implements measures to minimise mobile data usage.	Should	Yes	**(Section 5.2.1)**
11	Camera has means to manage storage.	Should	Yes	**(Section 5.2.3)**
12	Camera operator can detect failure from a remote location.	Should	Yes	**(Section 5.2.3)**

Evaluation

The requirements derived from this project's goals and background research have all been completed. This indicates the success of the project with regard to solving the main problem.

6.4 Cost Evaluation

When comparing the total cost of this project's implementation to the consumer-grade solutions in Section 3.2.1, we can see that the price is comparable to the cheaper solutions. This implies that the solution proposed by this project is reasonably affordable.

Item	**Price (GBP)**
SIM7600E-H 4G-HAT	70.00
Raspberry Pi Zero W	13.58
Pi Camera v2	23.75
Witty Pi	20.83
Lead Acid Battery (12V * 1.2Ah = 14.4Wh)	15.16
12V Solar Panel	12.23
Solar Charging Controller	9.99
Total:	165.54

Table 6.6: A cost breakdown for the hardware used for the project.
For the items supplied by the ECS department, a reference price is provided.

Chapter 7

Conclusion

7.1 Summary

Firstly, this report reviewed the current landscape of automated photography and identified the strengths and weaknesses of current solutions. This review included time-lapse cameras, camera traps, low-cost/hobby-grade cameras and camera networks.

Based on the background research conducted, it was established that a targeted approach towards a specific use case would be optimal in designing a camera system. After consulting the project supervisor, the project scope was subsequently refined to concentrate on developing an automated camera system capable of remotely monitoring glaciers in Iceland. To reflect this choice, the project requirements cover the essential features necessary for a camera to function effectively in Iceland's remote conditions while incorporating the recommended enhancements previously identified in the research.

Subsequently, the report detailed the design choices and justifications for the hardware and software components of the project. The hardware selection described solutions for capturing and processing photos, communication, system timing, and power management. Power consumption considerations heavily influenced the decision-making process. Furthermore, the software design described the camera control system and the interface between the camera and its operator.

In the next section, an iterative implementation process is described going from a rudimentary prototype to a refined implementation. The prototype satisfied the

basic functionality in the design and is a first attempt at adding some of the networked behaviour of the camera by implementing photo uploads. Crucially, the prototype highlighted the properties of the camera system that needed improvement in the next iteration.

In the subsequent refinement, the prototype is enhanced through multiple avenues, as outlined below:

- Transitioning to the Raspberry Pi Zero W model, which substantially minimises power consumption.
- Adding a thumbnail image system to efficiently use limited mobile data and conserve battery power.
- Altering the camera system's operating system to consume less energy compared to the default configuration.
- Reducing photo upload time by sending photos in batches rather than individually.
- Identifying the most performant network configuration for photo uploads.

The second iteration also included the implementation of the web interfaces used to control and monitor the camera system.

Moreover, this report demonstrates the testing and evaluation of this project's implementation. Firstly, power consumption testing showed that the final implementation could operate independently on battery power for an extended period, fulfilling a critical project requirement. The considerable power consumption reductions from the prototype are also demonstrated, further demonstrating the camera's ability to use power efficiently. Secondly, the implementation was tested against the project's requirements, demonstrating the completion of the tasks planned for this project. In addition, the proposed solution was shown to be reasonably affordable compared to consumer-grade time-lapse cameras. Lastly, a 'long test' was conducted to validate the camera's ability to operate continuously and automatically without interruption.

The project's objective of continuously developing and testing an automated remote photography solution was achieved, yielding an implementation that:

- Is energy-efficient, allowing it to operate for extended periods without an abundant power source.

- Is network-enabled, allowing operators to monitor, configure, and collect data remotely without immediate physical access, a crucial feature for automated remote photography.
- Provides the means to manage the resource limitations encountered when running a camera in a remote environment.
- Incorporates hardware and software design choices that facilitate easy customisation.
- Is relatively affordable to implement.

In summary, this project offers a viable solution to the challenge of performing automated photography in remote locations, featuring limited resource management, customizability, and remote feedback provision, which conventional photography solutions lack. The project also provides a robust foundation for adapting the system to similar use cases.

7.2 Future Work

Firstly, The result of this project is a highly customisable system that could be used in many remote automated photography applications. However, there remains an opportunity to enhance the accessibility of this camera's customisation to enable researchers to use the system effectively in their research, catering to varying requirements distinct from those encountered in glacier photography in Iceland.

Possible customisation options may include:

- Making it easier to program custom behaviour into the camera's regular schedule (e.g. adding custom scripts).
- Facilitate easier hardware changes (e.g. infrared camera sensors, more powerful processors, 5G modems).
- Add more control over power saving/data saving options (e.g. the operator may have access to cheaper mobile data and can afford to upload more detailed photos to the cloud).

Furthermore, the current project primarily focuses on operating a single camera. A natural extension would be to scale the system and develop a network capable of managing multiple cameras simultaneously. Integrating additional cameras would facilitate more extensive data collection across various locations, therefore enhancing research endeavours that rely on high-volume data sets. Marchowski's study presents a photography-based method for automated bird counting in population surveys [24]. The research illustrates that automating bird counts with machine learning significantly accelerates the process compared to traditional manual counting methods. However, the speed remains constrained by the time required to acquire and prepare the extensive image data needed for the automated counting process. Similar projects could greatly benefit from the ability to quickly gather substantial quantities of information, particularly when conducted in remote locations.

Lastly, while this project only leverages cloud computing for storing photos and controlling the camera system, many more cloud technologies could be incorporated into this system. For example, the processing capabilities of this project's camera system are inherently limited; however, once the data is stored in the cloud, there are fewer constraints. Marchowski's research demonstrates how automatically interpreting image data may require processing power much greater than that of a Raspberry Pi [24]. However, if cloud computing is leveraged to do the image processing, that could automatically give the camera system's operator a more timely interpretation of the data they are collecting without putting more strain on the resource-limited camera.

7.3 Gantt Chart

Below is the Gantt chart produced in Microsoft Project. This shows the planned and actual progress of the project. Overall, the Gantt chart was an effective tool that ensured this project effectively used its allocated time.

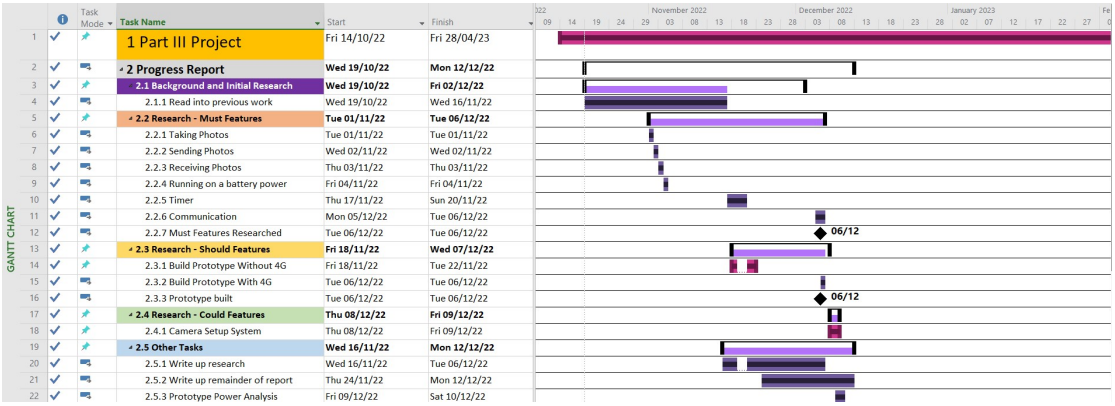


FIGURE 7.1: Gantt chart part 1.

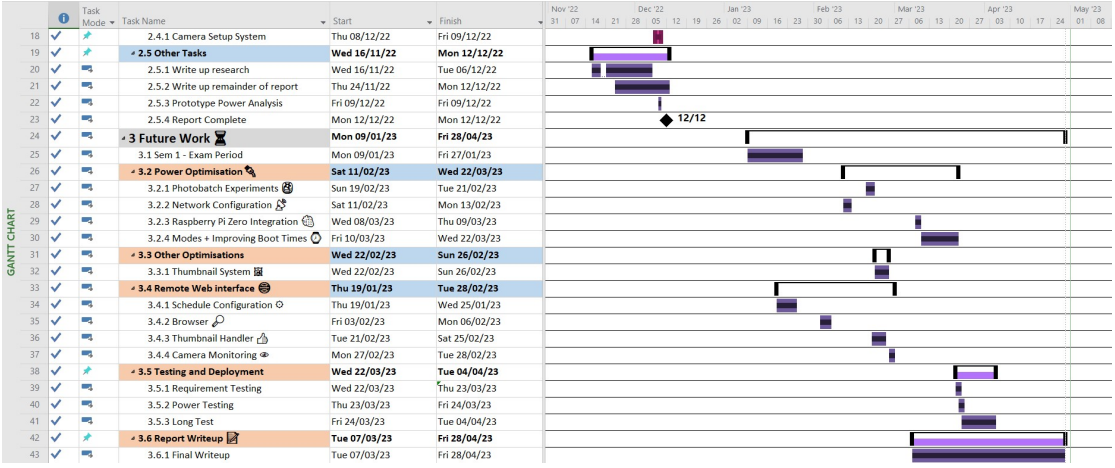


FIGURE 7.2: Gantt chart part 2.

Appendix A

Additional Documents

A.1 Risk Analysis

A.1.1 Project Risk Table

This table summarises the risk analysis for this project. For each problem, a probability, severity and mitigation is given. Risk is calculated as $Risk = Probability \times Severity$

Problem	**Probability**	**Severity**	**Risk**	**Mitigation**
Hardware damaged/fails	2	5	10	Organise replacement options from reliable suppliers so that the absence of hardware does not halt progress.
Failure/Inaccessibility to cloud storage that the project relies on	1	3	3	Organise alternative cloud hosting solutions (e.g Microsoft Azure as an alternative to AWS)

Problem	**Probability**	**Severity**	**Risk**	**Mitigation**
Loss of code	1	2	2	Consistent use of version control so that changes to code can easily be tracked and the amount of lost code is minimised.
Lack of good documentation for software and/or hardware used in the project	2	2	4	Make use of forums and/or discussion boards where people talk about the problematic piece of hardware/software. Contact manufacturers/designers where possible.
Falling off the project schedule	3	5	15	Have well-defined tasks with priorities. Tasks that are not as important to complete can be excluded should time restrictions become too tight.
Illness disrupts the ability to develop the project	3	2	6	Add provisions for time delays/slippage in the project planning.

Problem	**Probability**	**Severity**	**Risk**	**Mitigation**
Illness of supervisor	3	2	6	Maintain a consistent and well-established communication channel with the supervisor. Incorporate methods which do not require face-to-face contact.
Other university work gets in the way of the project	4	2	8	Be prepared to minimise non-essential parts of personal schedule, such as socialising, to allocate more time to work on the project.
Insufficient skills to progress part of the project	5	2	10	Adapt elements of the project to align with my personal skill set where appropriate. Additionally, know where to find the essential learning materials and do not hesitate to contact the project supervisor.
A major personal event that prevents project progress.	1	5	5	At this point, seek help through the university and consider applying for deadline extensions.

Table A.1: Project risk table

A.2 Original Brief

A.2.1 Problem

Automated/remote photography is a powerful tool that allows us to capture subjects and for the image data to reach the photographer without requiring immediate, direct physical access. However, design challenges arise from remote camera systems that are a significant distance from the data endpoint and a plentiful power source.

A.2.2 Goals

This project aims to investigate the possible hardware and software techniques that enable effective remote/automated photography and implement a capable system.

1. Prototype an automated system that can take photos and transmit them.
2. From a system remote to the camera, access the image data sent by the camera.
3. Only have the system on when it needs to be on to preserve power.
4. Allow for the system to be configured for different photography cycles.
5. Provide means to facilitate the camera's set-up (such as checking where the camera is pointing).
6. Explore the possible ways to improve upon the prototype.
7. Test the system in the field.

A.2.3 Scope

The priority will be to implement a prototype that achieves the core functionality of remote photography. Once completed, the project can direct its focus towards improving the prototype. Finally, time permitting, the project can include testing the system in the field.

1. **Hardware**

   (a) Components used will be 'off the shelf'.

   (b) Parts will be chosen such that the system can operate remotely.

   (c) Hardware choice will be limited to what is available within a given budget and time frame.

2. **Software**

   (a) Software will be the main way to combine the hardware elements to achieve the desired functionality.

   (b) A software solution will be required on the **receiving** end of the photos. However, not too much focus should be directed to this feature. Focus will be on the **sending** end's software.

   (c) As hardware choices are more rigid, improvements to the system's performance will be primarily explored through software.

3. **Other**

   (a) Should time permit for field testing, considerations will be made to make the system suitable for the environment it's placed in.

A.3 Data Archive Contents

Operating System

This folder contains an image of the SD card with this project setup.

- README.txt
- arpimage.img

Camera Scripts

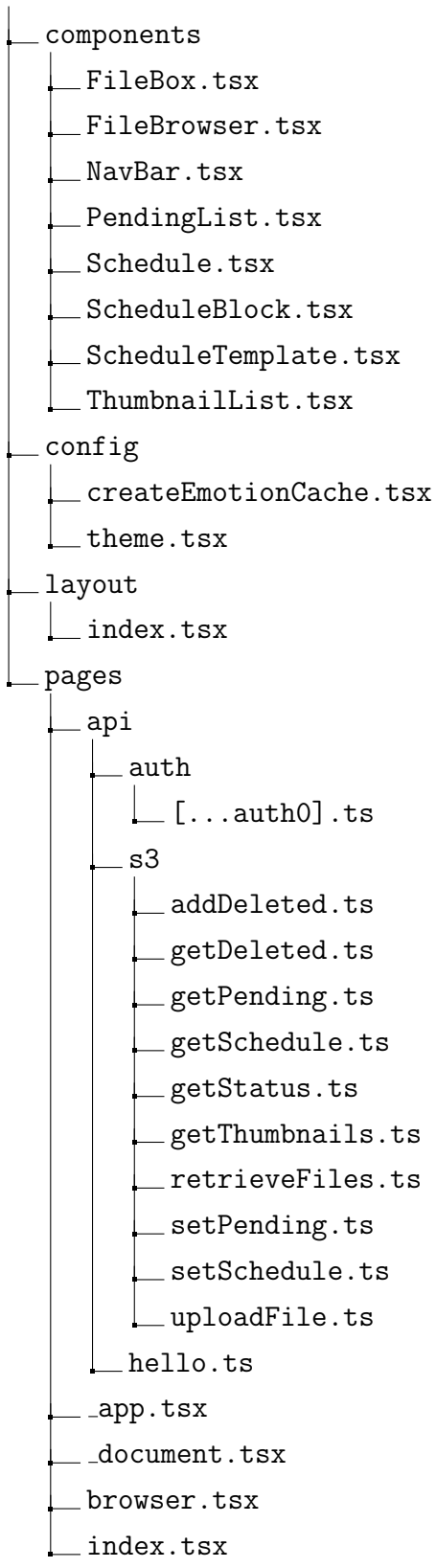
Also contained in the operating system image, these are the raw scripts for the camera hardware.

- boot_profiles
  - etc
    - systemd
      - system
        - camera-control.service
        - log-boot-target.service
        - select-target.service
  - lib
    - systemd
      - system
        - ModemManager.service
        - avahi-daemon.service
        - dhcpcd.service
        - keyboard-setup.service
        - networking.service
        - wpa_supplicant.service
  - usr
    - local
      - bin
        - capture_mode.sh
        - log-boot-target.sh
        - mode_config.conf
        - network_mode.sh

```
        ├── select-target.sh
        └── setup_mode.sh
    ├── README
├── scripts
    ├── __pycache__
    │   ├── batchPhotos.cpython-310.pyc
    │   ├── capture.cpython-310.pyc
    │   ├── capture.cpython-38.pyc
    │   ├── network.cpython-310.pyc
    │   ├── thumbnail.cpython-310.pyc
    │   ├── thumbnail.cpython-38.pyc
    │   ├── upload.cpython-310.pyc
    │   └── upload.cpython-38.pyc
    ├── atcontrol.py
    ├── batchPhotos.py
    ├── capture.py
    ├── main.py
    ├── main.sh
    ├── network.py
    ├── setup.sh
    ├── status.py
    ├── syncSchedule.py
    ├── testUpload.py
    ├── thumbnail.py
    └── upload.py
├── witty_scripts
    ├── input_voltage.sh
    └── utilities.sh
├── apt_packages.txt
├── camera.log
├── cameratree
├── mode_config.conf
├── requirements.txt
└── setup_guide
```

Web Application

These file make up the web application for this project.



```
    ├── status.tsx
    └── thumbnails.tsx
├── public
    ├── favicon.ico
    ├── next.svg
    ├── thirteen.svg
    └── vercel.svg
├── styles
    ├── Home.module.css
    └── globals.css
├── types
    ├── BlockUpdate.ts
    ├── Duration.ts
    ├── DurationUpdate.ts
    ├── FileBoxProps.ts
    ├── ImageRecord.ts
    ├── PendingRecord.ts
    ├── RemoveBlock.ts
    ├── ScheduleBlockProps.ts
    ├── StatusProps.ts
    ├── ThumbnailRecord.ts
    └── UploadType.ts
├── README.md
├── next.config.js
├── package-lock.json
├── package.json
└── tsconfig.json
```

A.4 Cost Breakdown

Item	**Cost GBP (without VAT)**
Waveshare SIM7600E-H 4G-HAT	70.00
Raspberry Pi Zero W	13.58
Total:	83.58

Table A.2: The list of items purchased for the project in addition to the items obtained from the ECS department.

Bibliography

[1] *3GPP Release 8 and 13*. URL: https://www.roundsolutions.com/en/3gpp-release-8-and-13/.

[2] Kevin Abas, Caio Porto, and Katia Obraczka. "Wireless smart camera networks for the surveillance of public spaces". In: *Computer* 47.5 (2014), pp. 37–44. ISSN: 00189162. DOI: 10.1109/MC.2014.140.

[3] Alex and RasPi.TV. *How much power does Pi Zero W use?* Mar. 2017. URL: http://raspi.tv/2017/how-much-power-does-pi-zero-w-use.

[4] ASUS. *ASUS Tinker Board S R2.0*. URL: https://tinker-board.asus.com/product/tinker-board-s-r2.html.

[5] Brinno. *Brinno TLC200 Time Lapse Camera*. URL: https://brinno.com/pages/product-tlc200.

[6] British Geological Survey. *ICELAND GLACIER OBSERVATORY: 2 Year Icefall timelapse by Dr Jez Everest - YouTube*. May 2013. URL: https://www.youtube.com/watch?v=qSV2mIFMj9k.

[7] CAMDENBOSS. *AGM Battery Datasheet*. URL: https://www.camdenboss.com/products/datasheet/bel_series_D.pdf.

[8] Henry R. Carey. "Camera-Trapping: A Novel Device for Wild Animal Photography". In: *Journal of Mammalogy* 7.4 (Nov. 1926), p. 278. ISSN: 00222372. DOI: 10.2307/1373576.

[9] Chronosys. *Odin 4K time-lapse camera*. URL: https://www.chronosys.co.uk/.

[10] Dante4. *Rock Pi 4 Tests and benchmarks - Radxa Forum*. Jan. 2019. URL: https://forum.radxa.com/t/tests-and-benchmarks/235/8.

[11] Eben Kouao. *EbenKouao/pi-camera-stream-flask: Create your own live camera stream using a Raspberry Pi 4*. URL: https://github.com/EbenKouao/pi-camera-stream-flask.

[12] Enlaps. *Tikee 3 Pro*. URL: https://enlaps.io/en/tikee-3-pro-plus-solar-panel.html.

[13] Shahin Farahani. "ZigBee/IEEE 802.15.4 Networking Examples". In: *ZigBee Wireless Networks and Transceivers* (2008), pp. 25–32. DOI: 10.1016/B978-0-7506-8393-7.00002-9.

[14] Brandin Grindstaff et al. "Grindstaff et al., p.1 Affordable Remote Monitoring of Plant Growth and Facilities using Raspberry Pi Computers 1". In: *bioRxiv* (2019). DOI: 10.1101/586776. URL: https://doi.org/10.1101/586776.

[15] hjc. *NanoPi M4 performance and consumption review - Reviews, Tutorials, Hardware hacks - Armbian Community Forums*. Sept. 2018. URL: https://forum.armbian.com/topic/8097-nanopi-m4-performance-and-consumption-review/.

[16] Innovateking-EU. *USB Color Tester Instructions Model A3 A3-B*. Tech. rep. 2018. URL: https://files.banggood.com/2018/05/A3&A3-B.pdf.

[17] M. R. James and S. Robson. "Sequential digital elevation models of active lava flows from ground-based stereo time-lapse imagery". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 97 (Nov. 2014), pp. 160–170. ISSN: 09242716. DOI: 10.1016/j.isprsjprs.2014.08.011.

[18] Mike R. James, Penelope How, and Peter M. Wynn. "Pointcatcher software: Analysis of glacial time-lapse photography and integration with multitemporal digital elevation models". In: *Journal of Glaciology* 62.231 (Feb. 2016), pp. 159–169. ISSN: 00221430. DOI: 10.1017/jog.2016.27.

[19] JEAN-LUC AUFRANC. *Rock 5B Benchmark*. URL: https://www.cnx-software.com/2022/07/20/rock-5b-rk3588-sbc-preview-what-works-what-doesnt-in-debian-11/.

[20] Jeff Geerling. *Raspberry Pi Power Consumption Benchmarks*. URL: https://www.pidramble.com/wiki/benchmarks/power-consumption.

[21] *libvips*. URL: https://www.libvips.org/API/current/.

[22] *LoRa — LoRa documentation*. URL: https://lora.readthedocs.io/en/latest/.

[23] MAPLOGS.COM. *Fjallsárlón, Iceland Sunrise Sunset Times*. URL: https://sunrise.maplogs.com/fjalls_rl_n_iceland.159296.html.

[24] Dominik Marchowski. "Drones, automatic counting tools, and artificial neural networks in wildlife population censusing". In: *Ecology and Evolution* 11.22 (Nov. 2021), pp. 16214–16227. ISSN: 2045-7758. DOI: 10.1002/ECE3.8302. URL: https://onlinelibrary.wiley.com/doi/full/10.1002/ece3.8302.

[25] Kirk Martinez. *Fjallsárlón timelapse 2018 - YouTube*. Aug. 2019. URL: https://www.youtube.com/watch?v=PKgbatZHnvc.

[26] *NanoPi M4 - FriendlyELEC WiKi*. URL: https://wiki.friendlyelec.com/wiki/index.php/NanoPi_M4.

[27] NatureSpy. *Browning Recon Force Elite HP4*. URL: https://shop.naturespy.org/products/browning-recon-force-elite-hp4-wildlife-camera-btc-7e-hp4.

[28] *NextJS Documentation*. URL: https://nextjs.org/docs.

[29] RS-Online. *Raspberry Pi Camera Module v2 Specifications*. URL: https://cdn.sparkfun.com/datasheets/Dev/RaspberryPi/RPiCamMod2.pdf.

[30] Radxa. *Rock5 - Radxa Wiki*. URL: https://wiki.radxa.com/Rock5.

[31] Raspberry Pi. *Raspberry Pi Zero W*. URL: https://www.raspberrypi.com/products/raspberry-pi-zero-w/.

[32] Raspberry Pi Foundation. *Raspberry Pi 3 Model B+*. URL: www.raspberrypi.org/products/raspberry.

[33] Richard Kirby and David Attenborough. *David Attenbourough: Interview & Behind-the-Scenes*. 1995. URL: https://www.youtube.com/watch?v=Whs0yJnjTHI&ab_channel=NigelVerney.

[34] *Rock Pi 4 - the next generation RPI*. URL: https://rockpi.org/rockpi4.

[35] Gerhard Schoener. "Time-Lapse Photography: Low-Cost, Low-Tech Alternative for Monitoring Flow Depth". In: (2017). DOI: 10.1061/(ASCE)HE.1943. URL: https://orcid.org/0000-0002-1183-0419..

[36] SIGFOX. "Sigfox Technical Overview". In: (2017). URL: https://www.avnet.com/wps/wcm/connect/onesite/03aebfe2-98f7-4c28-be5f-90638c898009/sigfox-technical-overview.pdf.

[37] *SIM7600E-H 4G HAT - Waveshare*. URL: https://www.waveshare.com/wiki/SIM7600E-H_4G_HAT.

[38] Telit. *Power Consumption - Telit Wireless Solutions ME910C1 Series User Manual [Page 26] — ManualsLib*. URL: https://www.manualslib.com/manual/1494005/Telit-Wireless-Solutions-Me910c1-Series.html?page=26.

[39] Franck Trolliet et al. *B A Use of camera traps for wildlife studies. A review*. Tech. rep. 3. 2014, pp. 446–454.

[40] UUGear. *Witty Pi 3 Realtime Clock and Power Management for Raspberry Pi User Manual (revision 1.06)*. Tech. rep. 2020. URL: http://www.uugear.com/doc/WittyPi3_UserManual.pdf.

[41] Simcom Wireless and Solutions Limited. *SIM7500 and SIM7600 Series AT Command Manual LTE Module*. 2021. URL: www.simcom.com.