ECE4094
Project A

Design Specification

	First Name	Last Name	ID	Email
Supervisor	Bill	Corcoran	-	bill.corcoran@monash.edu
Student 1	Anthony	Baudinette	26949415	ajbau1@student.monash.edu
Student 2	Jarred	Paola	25958674	jpao1@student.monash.edu

Introduction **2**

Project Design Overview **3**

Functional Specifications **4**

Server specification 4

Install requirements 4

Modules and Commands 4

Client specification 5

Install requirements 5

Modules and Commands 6

TCP keyword strings 6

Non-Functional Specifications **7**

Appendix A - Gantt Chart (Initial) **8**

Appendix B - Gantt Chart (Revised) **9**

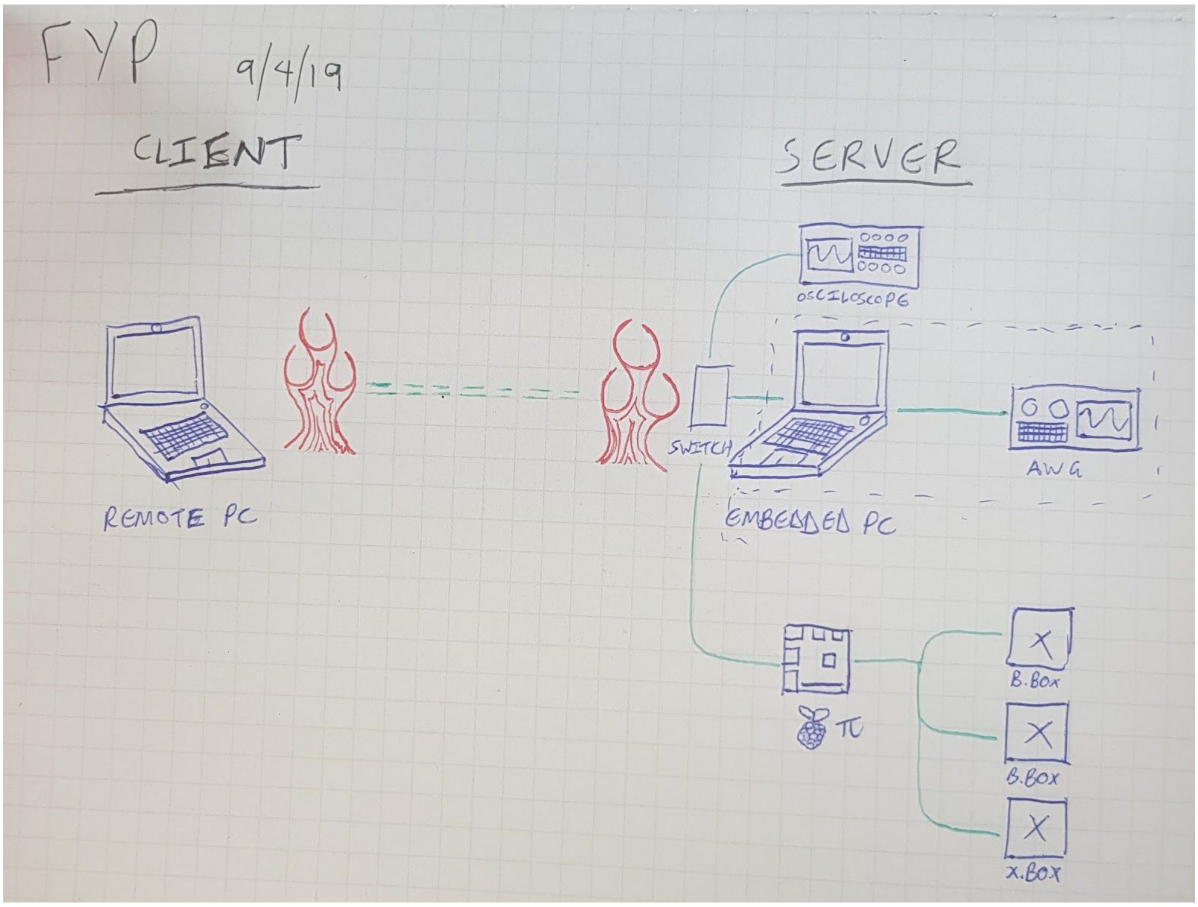
Introduction

This document aims to set out the reasoning behind our decisions in the project so far, and where we wish to focus our direction as we move forward. In this document, we will show the various considerations we have made as we encounter roadblocks and new information as a part of our research and application of the project, including but not limited to the decisions, specifications, and requirements of us as a consequence of these choices.

We will reflect and observe choices we have made regarding the functionality of the project, what we need and want to perform in the final product, and what we can and can not control or choose with regards to the design, functionality, and timeline as we build towards our end goal.

We are constantly learning, not only in terms of the knowledge we can apply to the project itself, but also the ways in which we learn and operation, individually and as a team, and it is with this in mind that we take the time to reconsider our progress to date in order to optimise our final design.

Project Design Overview



Our project consists of a ‘client’ and ‘server’ connection between a remote computer that is operated by the user, and an embedded computer that is connected to various devices. The remote computer will ideally only require interaction and use with Matlab coding, with our project capabilities enacting and using Python functionality that is not required of the user to understand. Through this, the user will be able to call and listen to devices on the embedded computer, using them without being physically connected.

At the beginning of the year, we made a Gantt Chart (see Appendix A) to timeline our aims and aspirations for the project, what we hoped to achieve throughout the year and when. Now that we have reached the middle of the year, we have developed a revised Gantt Chart (Appendix B) with consideration to our newfound knowledge, project familiarity, progress, and remaining options. These considerations have been divided into functional specifications and non-functional specifications.

Functional Specifications

Server specification

Install requirements

Python and pyVisa library:
The tunnel component of the project is implemented via a TCP connection in python 3. The program also requires pyVisa with version 1.10.0 being used to implement our program

Keysight soft front panel:
This requirement is specific to the implementation of the project the soft front panel must be running either on the same device with AWG-ip set to “localhost” or on the embedded PC with the server connected to the embedded PC by LAN . General visa operations with non-PXI instruments can operate without the soft front panel running.

Processor performance requirements:
As the server of the project is designed to run on an embedded PC, the performance of the application does not appear to be a limiting factor. However in the interest of security any PC operating as the server and thus internet connected needs to have supported OS and be up to date with security patches.

Alternatively, a router may be desired as a part of the network, in order for the client to interact with several or more devices. This is an implementation extension that we wish to explore and develop more specifications around as we further explore the capabilities of our current functionality and the ease of extension moving forward.

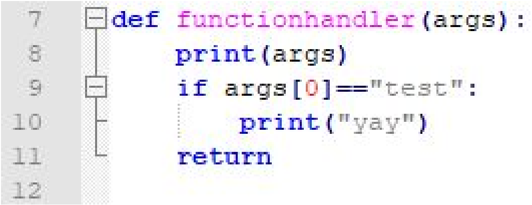
Modules and Commands

Configuration file:
On startup the server program reads from config.txt to set the options for the other modules. This task is performed by the run.py script which also calls the initialization of the visa module.
The settings stored in the config.txt file are:

- TCP port
- TCP IP
- TCP buffer size
- AWG-VisaAddress
- txtFileLocation

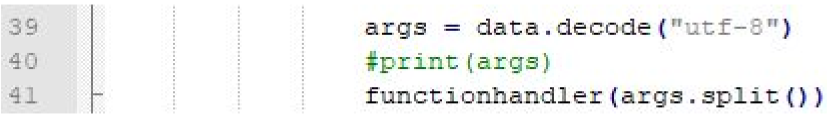
Function handler:

The TCP module of the device interprets all incoming messages and splits them into arguments routing them to the appropriate module to perform the requested function, the exact commands and arguments for this section is described by the TCP keyword strings section. The code below shows (1) reading the first element of the ASCII string and comparing it against a set of keywords, and (2) the python function decoding ASCII and splitting into arguments.



```
def functionhandler(args):
    print(args)
    if args[0]=="test":
        print("yay")
    return

```



```
                args = data.decode("utf-8")
                #print(args)
                functionhandler(args.split())
```

Visa interface:

This module facilitates two way communication with instruments with its primary functions being read and write commands, but for usability the module will also include functionality to list all connected devices (although it should be noted due to limitations of TCP on pyVisa the module cannot seek instruments on TCP without an address). In the case of the AWG pyVisa cannot directly interact with the instrument so the program communicates through the pyVisa interface to the soft front panel which has its own visa driver to communicate with the AWG.

File transfer:

This functionality of this module is to upload and save a txt file of a set length onto the server storage space. All files uploaded will be put into a folder as set by txtFileLocation in config.txt. A second instruction from the client calls a visa write to upload the file to the AWG.

Client specification

Install requirements

Matlab:
The operator will generally interface with Matlab as the main means of use. Matlab will be able to call and use python functions and modules developed, initiating the TCP Python tunnel to interact with the devices. This also allows the inclusion of a GUI for the user for ease of use rather than only coding, allowing for a wider target audience. It is possible to call the python scripts directly outside of MATLAB but for the purpose of ease of use, our project our example code for the end user will be run within MATLAB.

Python:
The tunnel component of the project is implemented via a TCP connection in python 3. As TCP communication is included by default with python no other libraries are required.

Modules and Commands

The functionality of the client generally mirrors that of the server as it send the instructions to run each module defined in the above server functionality section. The client python script contains a function which with a few exceptions simply restructure the function arguments to transmit over TCP.

Each function will start a TCP connection to the server when it starts, send its instruction, wait for response, and close the TCP connection before returning the result to MATLAB. The only complexity in the client side program will be in segmenting and normalising any txt files in preparation of transmitting to the server.

For written commands with no return string/value, the return is a success flag with an optional with string result. A further explanation of the instruction strings transmitted is in the following section.

TCP keyword strings

Each TCP communication transmits a string of ASCII characters keywords in the instructions are separated by spaces. The current list of instructions is shown in the table below: (*Italic terms represent data not keywords)* in the case of strings in arg[3] re-add spaces

tx->rx	arg[0]	arg[1]	arg[2]	arg[3]	Instruction description
client-> server	ping				Debugging code to check if server is available
server-> client	pong				Server response to ping
client-> server	visa	listDevices			Prompts server to return all connected devices
server-> client	visa	availiableDevices	*Instrument ID array*		Response to listDevices
client-> server	visa	read	*instID*		
server-> client	visa	readResult	*Success flag*	*result*	Response to visa read
client-> server	visa	write	*instID*	*command*	
server-> client	visa	writeResult	*Success flag*	*Error description (optional)*	Response to visa write
client-> server	visa	query	*instID*	*command*	Interpreted by server and write followed by read, server responds with write and read responses
client-> server	file	txRequest	*filename*	*Number of segments to send*	
server-> client	file	sendNextSegment			Will execute as many times as defined by txRequest
client-> server	file	upload	*filename*		
server-> client	file	uploadResult	*Success flag*	*Error description (optional)*	

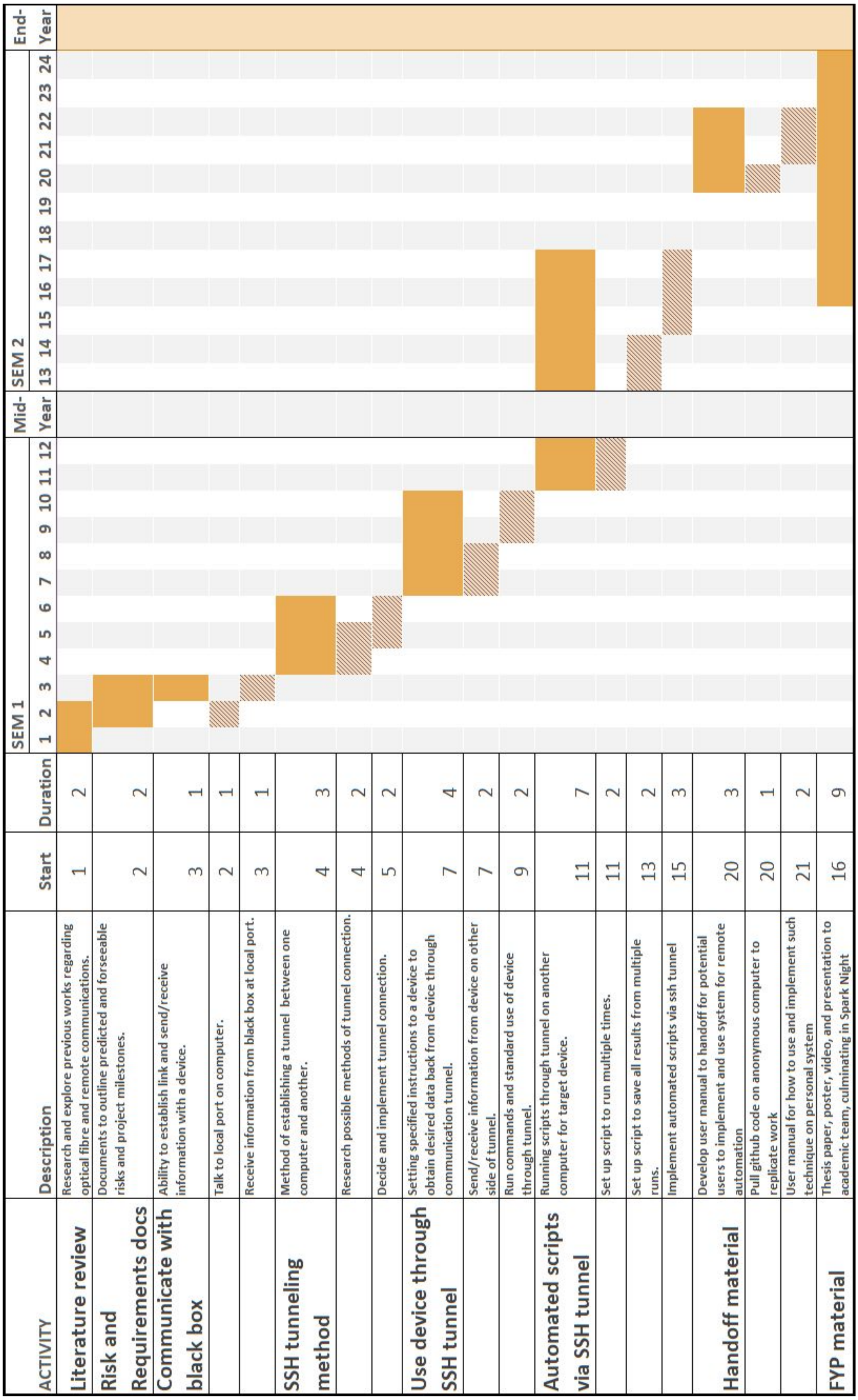
Non-Functional Specifications

When trying to interact with programs and devices, there are some legal considerations that we need to make. Since we are only running a software solution, then we are not taking apart any devices, and so do not run the risk of voiding warranties. Our biggest hurdle then is tapping into the operation of the software on the devices. Since the visa commands we are using and sending to the device are the same as the device and computer interaction already used, we are only specifying the instructions manually rather than through another program, and therefore should have no problems with obstructing or tampering with the devices, so long as we only use commands and instructions provided from relevant user manuals and programmer guides.

In terms of digital space, an easy to transfer, small package of files would be desirable. Something that can be copied and pasted into a desired directory, and be ready to go. As for physical space, we are not taking any space with the project itself as it is only programming.

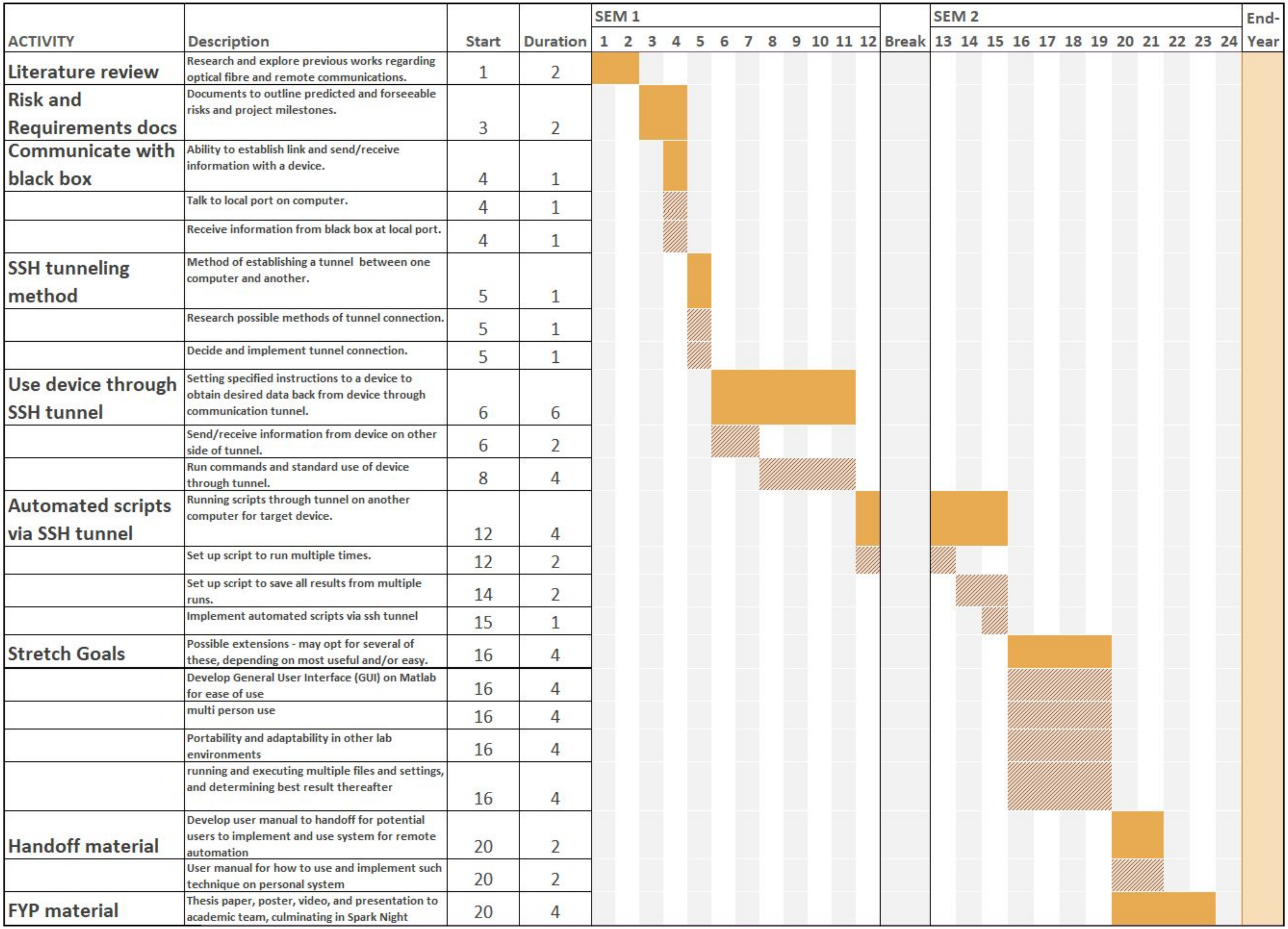
For the use case of the project, at least two computers with one connected to a device such as a function generator or a oscilloscope would need to be set up to show basic functionality. For development, however, this is not a regular requirement of specific set up, and thus does not necessitate storage or consideration on top of the project itself.

Appendix A - Gantt Chart (Initial)



ACTIVITY	Description	Start	Duration	SEM 1												Mid-Year	SEM 2												End-Year
				1	2	3	4	5	6	7	8	9	10	11	12		13	14	15	16	17	18	19	20	21	22	23	24	
Literature review	Research and explore previous works regarding optical fibre and remote communications.	1	2	■	■																								
Risk and Requirements docs	Documents to outline predicted and forseeable risks and project milestones.	2	2		■	■																							
Communicate with black box	Ability to establish link and send/receive information with a device.	3	1			■																							
	Talk to local port on computer.	2	1		▨																								
	Receive information from black box at local port.	3	1			▨																							
SSH tunneling method	Method of establishing a tunnel between one computer and another.	4	3				■	■	■																				
	Research possible methods of tunnel connection.	4	2				▨	▨																					
	Decide and implement tunnel connection.	5	2					▨	▨																				
Use device through SSH tunnel	Setting specified instructions to a device to obtain desired data back from device through communication tunnel.	7	4							■	■	■	■																
	Send/receive information from device on other side of tunnel.	7	2							▨	▨																		
	Run commands and standard use of device through tunnel.	9	2									▨	▨																
Automated scripts via SSH tunnel	Running scripts through tunnel on another computer for target device.	11	7											■	■		■	■	■	■	■								
	Set up script to run multiple times.	11	2											▨	▨														
	Set up script to save all results from multiple runs.	13	2														▨	▨											
	Implement automated scripts via ssh tunnel	15	3																▨	▨	▨								
Handoff material	Develop user manual to handoff for potential users to implement and use system for remote automation	20	3																					■	■	■			
	Pull github code on anonymous computer to replicate work	20	1																					▨					
	User manual for how to use and implement such technique on personal system	21	2																						▨	▨			
FYP material	Thesis paper, poster, video, and presentation to academic team, culminating in Spark Night	16	9																	■	■	■	■	■	■	■	■	■	

Appendix B - Gantt Chart (Revised)



ACTIVITY	Description	Start	Duration
Literature review	Research and explore previous works regarding optical fibre and remote communications.	1	2
Risk and Requirements docs	Documents to outline predicted and forseeable risks and project milestones.	3	2
Communicate with black box	Ability to establish link and send/receive information with a device.	4	1
	Talk to local port on computer.	4	1
	Receive information from black box at local port.	4	1
SSH tunneling method	Method of establishing a tunnel between one computer and another.	5	1
	Research possible methods of tunnel connection.	5	1
	Decide and implement tunnel connection.	5	1
Use device through SSH tunnel	Setting specified instructions to a device to obtain desired data back from device through communication tunnel.	6	6
	Send/receive information from device on other side of tunnel.	6	2
	Run commands and standard use of device through tunnel.	8	4
Automated scripts via SSH tunnel	Running scripts through tunnel on another computer for target device.	12	4
	Set up script to run multiple times.	12	2
	Set up script to save all results from multiple runs.	14	2
	Implement automated scripts via ssh tunnel	15	1
Stretch Goals	Possible extensions - may opt for several of these, depending on most useful and/or easy.	16	4
	Develop General User Interface (GUI) on Matlab for ease of use	16	4
	multi person use	16	4
	Portability and adaptability in other lab environments	16	4
	running and executing multiple files and settings, and determining best result thereafter	16	4
Handoff material	Develop user manual to handoff for potential users to implement and use system for remote automation	20	2
	User manual for how to use and implement such technique on personal system	20	2
FYP material	Thesis paper, poster, video, and presentation to academic team, culminating in Spark Night	20	4