# SETUP

* Ensure Power is connected
* Disable Wifi
* Set Up Screen
  + Plug in HDMI Dongle
  + Set Screen Resolution to 1080p
  + Mirror screen to projector
* Shut down all non-essential applications
  + Ensure all VMs are shutdown
* Launch VS Code
  + Load FireOne Folder
  + Set VS Code to Light Mode
  + Zoom In (1)
  + Upgrade Font Size (22)
  + Remove Status Bar
  + Remove Minimap
  + Open Invoke-Show.ps1
  + Open Invoke-Explosion.ps1
  + Open Bit Manipulation.ps1
  + Set to Invoke-Show.ps1 to start
* Lauch PowerPoint
  + Presenter View Disable
* Plug in and turn on presenter remote
  + Plug into Laptop
  + Set up Presenter remote with 30 mins on it
  + Pause Timer
* Phone on Silent

# Count down

* Ensure presentation has focus
* Start Presenter Remote Timer
* 1 Click to automate countdown

# Title Slide

* Good Afternoon
* Thanks for coming

# Sponsors

* Say Thanks

# BIO

* Spent a lot of time building and automating solutions for clients
* Now working as a cloud solutions architect
* Other Pashion: Fireworks
* Been setting them up professionally for over 17 years
* (1 Click) for photos

# Intro to Talk

* Noticed that the Firing System had a serial port on it
* Knew that serial is a very basic protocol
* Figured ‘how hard can this be’
* Managed to figure it out
* Build a PowerShell Module
* Journey around real world impact

# Aims

* PowerShell can and does have a real world impact
* Share my love of Fireworks
* Talk about how I turned this work into a module
* (This is a PowerShell conference after all)
* Touch on some of the reverse engineering techniques I used

# Disclaimer

* Usual Disclaimers about these being my personal views
* Do try this stuff at home
* Just not with fireworks
* (Click)
* Fireworks have Immense Power
* (Click)
* And are highly unpredictable

# (Take a Drink/Pause)

# Video

* Starting with a demo of the final PowerShell module being used to set off fireworks
* Unfortunately no live demo of this bit
* (Play Video)
* (Next Slide)
* Before getting into the code
* Go over how professional fireworks are typically connected to firing systems

# Explainer – Fireworks

* Fireworks used in the video
* Electrical igniters (Black Wires)
* Dodgy resistor

# Explainer – Modules

* Fireworks plug into a module board
* Connects to the module control board by Centronics cable
* Each module is numbered so you can address fireworks kind of like an IP address
* Example shows LEDs and wire loops simulating fireworks for testing

# Explainer – Control

* Modules connect to Control Module
* Normally these are pre-loaded with the programs via USB
* Can fire without any additional input

# Explainer – Laptop

* Most important: Nice cup of English Breakfast Tea
* Laptop connected via Serial Cable
* In theory could have used USB to USB but Serial is FAR easier to reverse Engineer

# Explainer – Excel

* What you didn’t see was the Excel Spreadsheet being used to create the show with timings

# Invoke-Show Script

* Read into PowerShell using the awesome ImportExcel Module (Shout out to Doug Finke)
* Orders the fireworks into time order
* Checks the data is valid
* Runs a test against the FireOne Panel to ensure all Cues exist
* Main Loop
  + Sets up a high accuracy timer
  + As there is an ordered list we can simply check if the current firework needs to fire
  + Works fast enough for simultaneous shots
  + When ready calls the Panel to ignite the firework

# (Take a Drink/Pause)

# Reverse Engineering

* Used a yoctopuce RS232 with Serial Adapter
* Yoctopuce is designed for industrial automation
* Enables snooping of Serial connections without interfering
* Other physical and virtual options were considered but primarily where more expensive
* Sends data to its own application for visualisation
* Can do 1000 other things but didn’t need them
* Reverse Engineering
  + Used the official software to perform actions
  + Sends a hexadecimal request
  + Receives a steam of hexadecimal back
  + Looking for known numbers / increments primarily
  + Eventually managed to figure out all the main commands and expected responses through a lot of trial and error

# Requirements

* Easy to Iterate and Extend
* Private and Public Functions
* Easy to Pick up and Use

# Module Design

* ModuleBuilder enables Public/Private Functions with an easy and simple compilation process
  + Shoutout to the ModuleBuilder team – Joel Bennett primarily but others have contributed
* PSFramework enables setting persistence between load, very useful for COM management
  + Also provides a really nice logging framework which works well with individual function testing and compiles nicely with ModuleBuilder
  + Shoutout to Friedrich Weinmann
* Wanted to show an implementation of the module but didn’t want to overly restrict the format
  + Maintaining as a script enables people to swap out data sources
  + Avoids a core module dependency on ImportExcel
  + Consistent with module vs script convention
  + Timing logic very Dumb and not optimised or accurate

# Invoke-Explosion Cmdlet

* Each command is in a different file
* Commands are primarily made up of 12 Byte ‘packets’
* Module is Hex encoded decimal value
* Cue is bit flag on 1 of the bytes 4-8 (right to left)
* (Show Bit Minipulation.ps1 to visualise how the bits move)
* Firing Process
  + Mention Fire and Test modes on the Panel
  + Send Data to COM
  + Follow immediately with second command to fire

# Send-RawData Script

* Start by identifying and opening the COM port
* Making use of PSFramework for logging and COM port identification
* Creates a checksum and appends it to the byte array
* Checksums are used to ensure that data wasn’t corrupted during transit
* Serial typically uses Binary Exclusive OR for validation
* Result packets are always the same length as the sending packet
* Send the packet
* Wait for Response unless told not to
* Loop through reading all data from the port until we’re received all expected data or we time out
* Validate response by doing a checksum calculation and comparing against the received checksum

# Demo (Recap)

* How Module Builder makes development modular
* How PSFramework allows logging and preferences to be set outside of the module
* How to manipulate binary data to send a request via COM

# (Take a Drink/Pause)

# Takeaways

* Most PowerShell has kinetic impact
* May not be explosive
  + Break medical machine – life critical consequences
  + Break CEOs email – financial consequences
* Modular Modules makes development easier
  + Ease collaboration and change requests
  + Easier to get started
* Build on the shoulders of giants
  + Invest time in what provides value
* Have fun learning!
  + Roller coaster journey
  + Overall fun project
  + Got me to this fun conference
* Thank you

# Video Again

* (Start Q&A)

# Questions

# Compiling the Module

* Run through the update-LocalModule.ps1 Script
* Reads the configuration from Build.psd1 – ModuleBuilder inputs
* Deletes any old versions to avoid issues with the compilation process
* Increments the build number for easy module identification
* Builds the module
* Creates a ZIP version for future automated deployment to the PS Gallery
* Ensures that there isn’t a valid COM connection as otherwise this connection would be orphaned when the module is reloaded – would require a Environment restart
* Either loads the module directly from the build path
* Or – Copies the module to the user’s module directory and loads automatically from there – enables persistent usage in other sessions

# Ignitors

* What an electrical igniter looks like
* How they are inserted into a firework
* Professional fireworks are often consumer fireworks with the mandatory safety fuse cut off and an electrical igniter added instead

# Setting up

* How the fireworks arrive onsite
* Boxed in UN marked, fire retardant boxes
* Each firework has a label identifying its location and wiring position

# Finished Setup

* What a site will look like after setup
* Modules come in various shapes and sizes