Raspberry Pi Setup Tutorial and Notes:

- Hardware:
    - Raspberry Pi 4 Model B (4gb)
        - Make sure it has Wi-Fi
        - Probably passable with less Ram, but cost should be coming down significantly soon
    - Rasp Pi Power Supply
    - Rasp Pi Case
    - Rasp Pi Terminal Block attachment
    - Rasp Pi Small Pinout Cable extension
    - Break Beam Sensors
    - Small Gauge Wire (18 gauge)
        - Along with wire connectors and tools to strip and make new connections
    -

- Raspberry Pi Initial Settings and Setup Commands
    - Enable SSH Access
        - Command: *sudo raspi-config*
        - Go to Interface Options and enable SSH connections
    - Enable cron
        - Command: *sudo crontab -e*
        - Select “nano” as the editor since it’s the easiest
        - Cron Details:
        - Cron is installed by default
        - Link to Info: How to Schedule Python Scripts With Cron — The Only Guide You’ll Ever Need | by Dario Radečić | Towards Data Science

Raspberry Pi Bash Commands and Descriptions:

- Commands to Run:
    1) *sudo apt install python3-pip*
        - Allows pip install for python-based packages.
        - Link: How to install pip on the Raspberry Pi - Pi My Life Up
    2) *sudo adduser “username”*
        - Username must be lower case, leave out quotes in actual command.
        - Example: *sudo adduser test*
        - Hit enter for all the information to leave blank.
        - Use the command below to check users on the system.
            - ls /home
    3) *sudo rpi-update*
        - will allow adding users to the RPi-GPIO group and make the GPIO pins available for them.

4) *sudo adduser "username" gpio*
    - adds a user to the GPIO group and lets them access the pins.
5) *sudo apt-get install freetds-dev freetds-bin unixodbc-dev tdsodbc*
    - Installs FreeTDS which are the drivers used for ODBC Connections
    - The 'bin' bit of the command specifies where the file will be located.
6) *pip install pyodbc sqlalchemy*
    - This may not actually be necessary, basically we just want pyodbc.
7) *sudo dpkg-reconfigure tdsodbc*
    - This is the way to do it after you install of the above
    - It automatically sets FreeTDS as the ODBC driver instead of having to manually do it and find all of the correct file paths.
    - Not necessary but good to have:
        - *sudo nano /bin/odbcinst.ini*
            - Opens the file like a text file and allows you editing of it.
            - Enter this Text (driver and setup paths will likely be different for each install)

              [FreeTDS]
              Description=FreeTDS Driver
              Driver=/usr/lib/odbc/libtdsodbc.so
              Setup=/usr/lib/odbc/libtdsS.so

How to make a script (.py) executable in terminal:

- There are two things, and I am not sure if one works without the other. So here goes.
- *#!/usr/bin/env python*
    - Add this to the top of whatever script you are writing. This makes it so when you execute it, the system knows where to find Python
- Command: *chmod +x /filepath.py*
- *(chmod +x /home/user/Documents/WifiCheck_Prod.sh)*
    - Idk if this one must be done to make the file executable or if just the above is enough. Find out soon enough.
    - Make sure you do this for 'bash' scripts to become executable.
        - Bash scripts can't be executed from the terminal without this step.
- Setting up Cron Tasks
    - CRON Commands:
        - Edit crontab tasks: crontab -e
        - View crontab tasks: crontab -l
    - Crontab Tasks Setup:
        - ** * * * python /home/user/Documents/TestScript1.py >> /home/user/Documents/Test_Log1/txt*
        - **/5 * * * /home/user/Documents/WifiCheck.sh > /home/user/Documents/Test_Log1.txt*

        - **/5 * * * * /home/user/Documents/WifiCheck_Prod.sh > /home/user/Documents/Test_Log_Prod_1.txt*
            - This works without the path the bash because it's a bash command. Can set the logs to overwrite or append either way I don't care.
    - CRON Extra Tips:
        - > = write to file, overwriting what was already there
        - >> = append to file, create if it doesn't exist
        - In CRON Specify 'python' before you type out the executable file path
- Systemd Commands, Start Service, Stop, Enable, and Re-Load:
    - View active services:
        - *systemctl --type=service*
    - Reload systemctl daemon after making changes to unit file:
        - *sudo systemctl daemon-reload*
    - Enable a service:
        - *sudo systemctl enable servicename.service*
    - Start a service:
        - *sudo systemctl start servicename.service*
    - Stop a service:
        - systemctl stop servicename.service
    - Disable a service:
        - *systemctl disable Script2.service*

How to setup Systemd file:

- To create or edit an existing Service file:
    - *sudo nano /etc/systemd/system/Servicename.service*
    - *sudo nano /etc/systemd/system/Script1.service*

Systemd BreakDown

- [**Unit**] : *This section is what sets up when your service starts*
- **Description**=Whatever you want the name to be : *Description can legit be whatever you want*
- **After**=network-online.target : *Service to start after the network is online. Can also use network.target but idk how that works tbh*
- **Wants**=network-online.target : *When using an 'after' you must include either a 'needs' or 'wants' specification*

- [**Service**]
- **Type**=simple : *There are others like 'One-Shot' and 'Idle' but allegedly this is the best for scripts that are supposed to run all the time.*
- **User**=user : *Make this whatever username you want to use. Lowercase is required.*
- **Restart**=always : *This can be set to 'on-failure' but this is allegedly the best way to make a script keep running*
- **RestartSec**=3 or 5 or 10: *Might play around with this and see if a different value makes the date come through better.*

- **ExecStartPre**=/bin/sleep 30 : *For whatever reason it seems like this was the key to getting the script to run properly after network connection was established. Without this bit it never seemed to run properly.*
- **ExecStart**=/usr/bin/python3 /home/user/Documents/Rasp_to_SQL.py : *This is the file*

- [**Install**]
- WantedBy=multi-user.target

Eric's Notes: After lots of trial and error, here are some of the findings.

- Initial Setup List:
  - Install 'pip'
  - Install newest Python package
- On New Raspberry Pi install after changing hostname
  - For whatever reason when I log in I no longer have to use "sudo" before all of the commands.
  - Idk if it's because I finally changed the generic hostname or what, but just know that it takes longer to use 'sudo command' than it is to just use the command. Plus you get a weird error using the 'sudo' stuff. Whatever, as long as it works.
- First start off with the sudo nano /etc/system/systemctl/ServiceName.service
  - Obvi replace the "ServiceName" bit with the actual service you want to create.
  - Include the path to the script interpreter before the path to the actual file you want to run or else it won't work, even if you make the script an executable, couldn't get it to work like that.
  - If it's a bash script or .sh file, you likely don't need to add the "/usr/bin/python3" bit before the file path. Just make sure that the bash/.sh file is executable.

Random other commands or settings

- Grep commands to find executable PID.
  - *ps aux | grep /home/user/Documents/Rasp_to_SQL.py*
    - Replace with actual file path for whatever program you are running.
- To 'Fix' Systemd ongoing file from timing out the SQL Server Connection
  - Fixed the python script to open and close a new connection each time the break beam changes state.
  - Edit the file /etc/ssl/openssl.cnf
  - *sudo nano /etc/ssl/openssl.cnf*
  - Change the MinProtocol and CipherString to 1.0 and 1 respectively.
  - Link to website listed below.
  - Note: I don't actually know how useful this is for new setups.
- Miscellaneous information:
  - Hostname / client name defaults to: raspberrypi
    - Command to call is: hostname

- To get Systemd Service to stop and restart on network loss, I have added in a couple of lines to stop and start the service (in this case its sudo systemctl stop Script1.service and then sudo systemctl start Script1.service) and this seems to be the best way to make sure that the service is stopped and then starts if the internet ever goes down.

Useful Links or Tips:

- Connect to MSSQL using FreeTDS / ODBC in Python. · GitHub
    - (Comments are the most useful on this one)
- How to install pip on the Raspberry Pi - Pi My Life Up
    - Pip install RPi
- Access GPIO pins without root. No access to /dev/mem. Try running as root! - Raspberry Pi Stack Exchange
    - Add user to RPi group
- Understanding and administering systemd :: Fedora Docs
    - Systemd information
- pyodbc.OperationalError: ('08001', '[08001] [Microsoft][ODBC Driver 17 for SQL Server]TCP Provider: Error code 0x2746 (10054) (SQLDriverConnect)') in mkleehammer pyodbc - Lightrun. How to fix weird timeout issue for Posting data to SQL
- excel - Delete a duplicated line (only if it's the next one!) - Stack Overflow
    - Create a filter for consecutive data
- Automatically connect a Raspberry Pi to a Wifi network | We Work We Play
    - creating a static IP (note remove the # words or else wifi won't work at all)
- Auto Check and Reconnect WiFi for a Raspberry Pi
    - Check and auto-reconnect to Wifi. Need to do some tweaks, but remember to get the location of bash and set the ping ip address line correctly.
- How Do I Set a Static IP Address on Raspberry Pi?
    - configure static IP using DHCP.
    - THIS IS THE ONE THAT ACTUALLY WORKS
- How to auto-connect your Raspberry Pi to a hidden SSID wifi network – RasPi.TV
    - Auto-Connect to hidden networks
- Sudo: unable to resolve host explained - Globo.Tech
    - Fix the changed hostname problems
- Reboot automacticly at midnight - Raspberry Pi Forums
    - Setup midnight reboot using crontab
- Convert sql output to string in python - Stack Overflow
    - Convert SQL query to a string (data type pyodbc to string)