Firmware / gSender

grblHAL and gSender will need to go through a couple major stages of development in order to get them ready for prototype use, initial board release, and fulfilling longer-term goals of the project.

Prototype Use

Only the minimum viable features for testing are needed at this stage to validate the board works:

- [x] ~~Initial firmware pin routing and any other baseline considerations to support the STM32F4 chip for typical IO~~
  - [x] ~~Limit switches~~
  - [x] ~~Spindle or Laser control (En, PWM/0-10V analog, Dir)~~
  - [x] ~~4th axis~~
- [x] ~~Baseline HAL plugin support for TMC2660 motor drivers to communicate over SPI~~

Board Release

- [ ] All circuitry and hardware tested internally and proven to be reliable
  - [ ] Power management (DC in, E-stop, toggle switch)
  - [ ] USB & Ethernet connection
  - [ ] Motors
  - [ ] Limit switches (JST and AUX)
  - [ ] Probe
  - [ ] Flood
  - [ ] Laser
  - [ ] 4th axis signals & limit switch
  - [ ] Spindle (power, digital, analog, RS485 for both connectors)
  - [ ] Tool Length Sensor
  - [ ] AUX outputs
  - [ ] SD Card
  - [ ] CANBUS (Pendant)
  - [ ] RGB LED (power in, both plugs out)
  - [ ] Analog in
  - [ ] Door
  - [ ] Aux comms (SPI, UART, programming pins, IRQ)
  - [ ] Probe points for later QA testing
- [ ] Motor drivers tuned for reasonably optimized CNC performance
  - [ ] Standby current reduction to help thermals
  - [ ] Address the divide between current tune and results seen so far with testers
- [ ] Defined EEPROM defaults for Sienci CNCs - copied from existing GRBL source
- [x] ~~Ability to enter built-in DFU mode from grblHAL for firmware update~~

- [ ] Handle typical machine startup and states (handling the larger range of error and alarm states grblHAL has, connection, homing)
    - [ ] For example, currently gSender isn't able to handle initial machine startup for Error 7 (somehow interpreted as Alarm 11 and freezes up without allowing the settings to be modified), whereas IOsender still allows it to happen
- [ ] Handle typical machine uses (jogging, set zero, gotos, probing, file attributes, run job, pause, stop, PWM/spindle)
    - [ ] Probing macro tweaks to better suit HAL? (touch off seems noticeably faster with the new MCU, the probing sequence could be made to run much faster?)
    - [ ] Jog time estimation needs to be changed since the HAL buffer is so much larger it'll say the job is done far before it's actually done
- [ ] 3 programmable macro buttons
    - [x] ~~Store and Edit macros in MCU EEPROM~~
    - [x] ~~Ability to select realtime characters to send~~
    - [ ] Send longer macros by directing to file path on SD card
- [ ] Ability to perform easy user firmware updating/flashing
- [ ] Handle more advanced tools (diagnostics, surfacing, squaring, tuning, firmware)
    - [x] ~~Generative and reliable EEPROM handling for HAL machines~~
- [ ] Access SD card
    - [ ] Smart enough to choose between Zmodem or FTP for file uploading to SD based on a USB or Ethernet connection
- [ ] Move compile-time settings to EEPROM for typical user tweaks or hardware changes to their machine (defines the CNCs physical setup plus some extras), what are the bit masks, the data structure, the EEPROM value we want to use
    - [ ] Motion system speeds and limits
    - [ ] Limits/Homing & attributes
    - [ ] Dual Y-axis homing - This is difficult to make runtime configurable
        - [ ] Talk to Terje to make homing config more modular
    - [ ] Spindle & attributes/comm style
    - [ ] Laser
    - [ ] Coolant/Mist/Vacuum - Store attributes about what is connected. Return capabilities upon prompting.
    - [ ] 4th axis & attributes
- [ ] Control spindle and laser via two separate PWMs
    - [ ] Could allocate laser as a reserved tool to store its X, Y, and Z offsets and send spindle control values differently when it's loaded
    - [ ] Need to store different min/max values
- [ ] RGB Status LED

- [ ] No connection:orange, Idle:white, Cycle/Running:green, Jogging:green, Hold:yellow, Door:yellow, Homing:blue, Check:blue, Alarm:red, E-stop:red, Sleep:gray, Tool Change:purple
- [ ] Fine to not do colour changes while motion in order to avoid stuttering

- [ ] MCU communication over Ethernet
- [ ] gS communication over Ethernet
- [ ] Internal tests of stall homing at machine limits and motor torque-out during cutting (use a variety of scenarios so we're confident with sense resistor selection)

Long-term

- [ ] Interface with motor driver settings directly for per-user tuning (power, currents, microsteps, max motion values, etc.)
- [ ] RS485 spindle control
- [ ] Stall homing for rough machine positioning
  - [ ] Use detections to do an initial machine self-check (are systems good to go?)
  - [ ] Detect aspects about the machine to update it's own default settings (e.g. machine limits)
- [ ] Detect stepper torque-out during job run to alert user
  - [ ] Can pause and move out of the way to stop from breaking things
- [ ] Hand-off more gS functionality to store on the SD or firmware
  - [ ] History of alarms & errors
  - [ ] Touch plate & style/macro (receive bit diameter)
  - [ ] Spindle/laser delay after command
  - [ ] CNC stats and maintenance
  - [ ] Program start/pause/resume/stop macros
  - [ ] TLS & tool change macros
  - [ ] ATC & macros
- [ ] Handle handing off typical settings to come from the CNC firmware instead of locally
- [ ] MCU job running over SD card
- [ ] Support HAL single-step mode
- [ ] Z 'baby-stepping' override (step the Z-axis up or down to tune the start of a job, or step it up slightly when job is paused to stop the bit from burning into the material)
- [ ] Independent override for plunging (i.e. Z-axis only moves)
- [ ] Spindle at speed checks
- [ ] CANBUS Pendant
  - [ ] Pendant enabled overrides
- [ ] Door input
- [ ] Adaptive feed rate overrides based on spindle load feedback
- [ ] S-curve support
- [ ] Toggle acceleration values between fast and precise? (3D carve vs. typical cutting)