Nose Gear Controller System for Canard Aircraft

System Documentation and Build Instructions

Author: Trevor Howard (trevor@borialis.com)

Date: 2024-02-05

Revision: 1.0

Nose Gear Controller System for Canard Aircraft

Introduction

The nose gear control system used on my Cozy C-GTCZ is an implementation of Marc Zeitlin's Auto-extend gear controller circuit first published (AFAIK) on the CozyBuilders mailing list, and available at this DropBox location: Nose Gear AutoExtend Circuit. The major innovation with this design was the incorporation of a LIDAR range-finder which (in conjunction with a pair of airspeed sensors and a throttle position sensor), permitted the auto-extend feature to trigger when expected – ie, during the landing phase – and not at other times such as slow flight, low passes, etc.

Marc's circuit was very cool but he only provided the circuit schematic and a Bill of Materials. The actual construction details were left to the builder. While relatively straight-forward, building with discrete components generally led to a rat's nest of relays and wires and typically a not-inconsiderable amount of post-build trouble-shooting.

I did in fact build one of these circuits and it worked well, but after a while I wanted to incorporate a couple of changes. Since I was playing around with CAD-based PCB design software at the time, I naturally gravitated to a PCB-based design.

System Basics

My implementation starts with Marc's design and then adds some additional elements.
The basic component is the Main Relay Board (MRB), and this is all that needs to be constructed to implement the original circuit. In addition, there are two optional boards that can be built to add additional features.

Main Relay Board (required)

- Implements the core gear control and latching auto-extend functionality
- Control point for the gear actuator/limit switches and gear control panel
- Provision for optional gear controller power switch or hard-wired (always-on) gear control power
- Optionally provides an on-board +5V regulated supply (needed for some LIDARs)
- Optionally provides on-board circuit protection via self-resetting PolyFuses
- Optionally adds +5V gear-up and gear-down indications for driving EFIS contact inputs

Arduino Custom Shield (optional)

- Replaces the hardware-based determination of the auto-extend trigger signal with a software-determined signal

- Provision for an optional LCD display module showing AGL altitude and airspeed
- Provision to interface with several different EFIS's for airspeed input
- Provision to interface with several different LIDARs for AGL altitude input
- Provision for audio-alerts and altitude call-outs

Control Panel Board (optional)

- Simplifies the wiring for the gear control panel and allows individual components to be replaced easily since they plug into the board
- Provision to automatically power off the gear controller after time delay when master power is off (still in development)

Here's a conceptual view of the system:

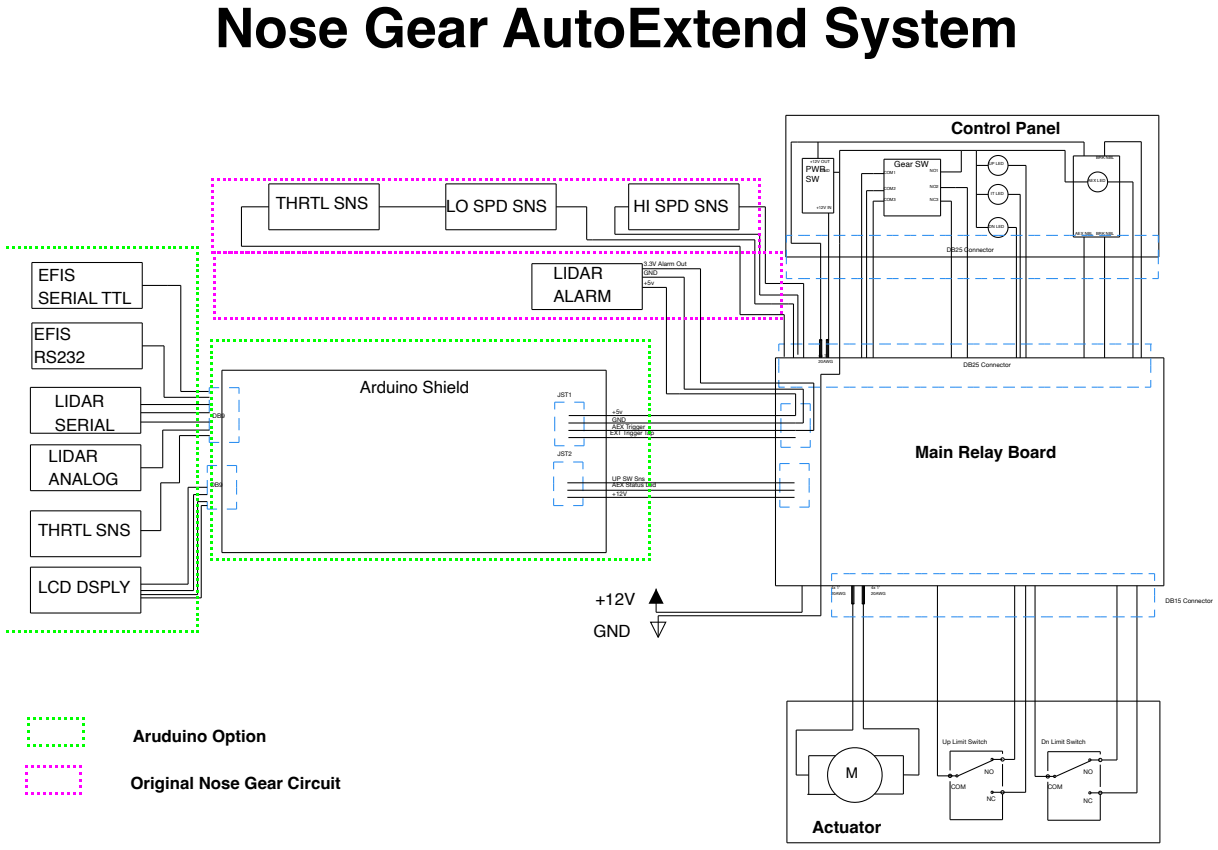

Figure 1 - Nose Gear AutoExtend System Conceptual Diagram

Main Relay Board Assembly Instructions

These instructions assume that you are proficient with normal through-hole soldering techniques. Assembly is very straight forward and as long as attention is paid to the correct orientation of the parts you should have a working main controller board in a couple of hours. All components except for one diode are mounted on the top side of the board.

This board is somewhat configurable depending on what you want to do. The main options are:

+5 regulated supply

A buck-style voltage regulator that provides up to 1A of +5V power. Needed by some LIDARS (Lightware) as a power supply. (also for the Arduino, if going that route).

Self-resetting circuit protection

Two polyfuses may optionally be installed that will trigger when the downstream power draw is > 10A (on the high-power side of the circuit) or > 1A (on the control side of the circuit). The 10A polyfuse is mainly intended to prevent damage in the event of a gear jam or micro-switch failure but it will also protect against shorts in the actuator wiring, etc. The 1A polyfuse is mainly to guard against shorts in the control side wiring.

Note that this is meant to AUGMENT existing circuit protection in the NG circuit, NOT REPLACE IT! You should still have a standard 15A fuse or breaker on the NG circuit supply line.

EFIS-friendly gear-up and gear-down signals

Many EFIS's provide a means to input the gear state which can be displayed on the EFIS and/or give audio warnings. However, these are typically required to be sensor voltages ~ 5V. This option provides negative logic (+5V = FALSE, GND = TRUE) gear signals which can safely be used with an EFIS or EMS .

AEX trigger options

The default build assumes that the AEX triggering inputs will be a positive signal from a Lightwave LIDAR configured as an altitude alarm switch and a grounding connection from three serially-connected switches (throttle position microswitch and two speed switches, per Marc's original design). If this is what you are doing, it is simply a matter of connecting the LIDAR switch output and the two sides of the serial-switch outputs to the appropriate interface points on the board and you are good to go.

If using an Arduino to determine speed/altitude and provide the AEX trigger, then the serially-connected airspeed pressure switch stuff needs to be bypassed by installing a jumper on JP2

If using some other means of external AEX triggering then the JP2 jumper is again used to bypass the speed/throttle switches and the AEX function is triggered when the AEX trigger input is raised HIGH (+5V). Negative logic AEX triggering (where the AEX trigger is grounded to initiate the AEX function) is also supported – in this case transistor Q2 is not used, the base and collector pins of Q2 are shorted together via a wire or jumper, and the 1K resistor R1 is replaced with a 0-ohm resistor or wire jumper.

Always-on Power option

The default build assumes that the gear controller will have an on-off gear controller power switch built into the gear control panel. This allows you to power up the gear controller and raise&lower the gear without needing the master power on. On the other hand, some people like to have the gear "always on" by being powered directly from the battery – this gives the capability to raise&lower the gear by using the gear position select switch only. The relay board provides for either option.

Note that if using the Arduino or other microcontroller, there should be some means of powering off the gear controller because otherwise the microcontroller and other active components will present a small but ever-present current draw that will eventually kill the battery. The required shutoff could be done either from the afore-mentioned gear power switch or from the master power switch if you have the gear powered from master power.

The always-on power option is enabled by installing a jumper on the underside of the board which essentially shorts together the pins on the control panel connector that would normally go to either side of the gear power switch. Be advised that if this option is enabled, the following components/options CANNOT also be installed:

- UP LED
- +5V Supply
- Arduino
- EFIS-friendly gear signals
- Power LED

These represent active current draws if they are installed on (or have connections to) the relay board.

OK, with these options in mind and without further ado let's get on to the actual assembly instructions:

The Board:

Here's a picture of the board showing where the components go:

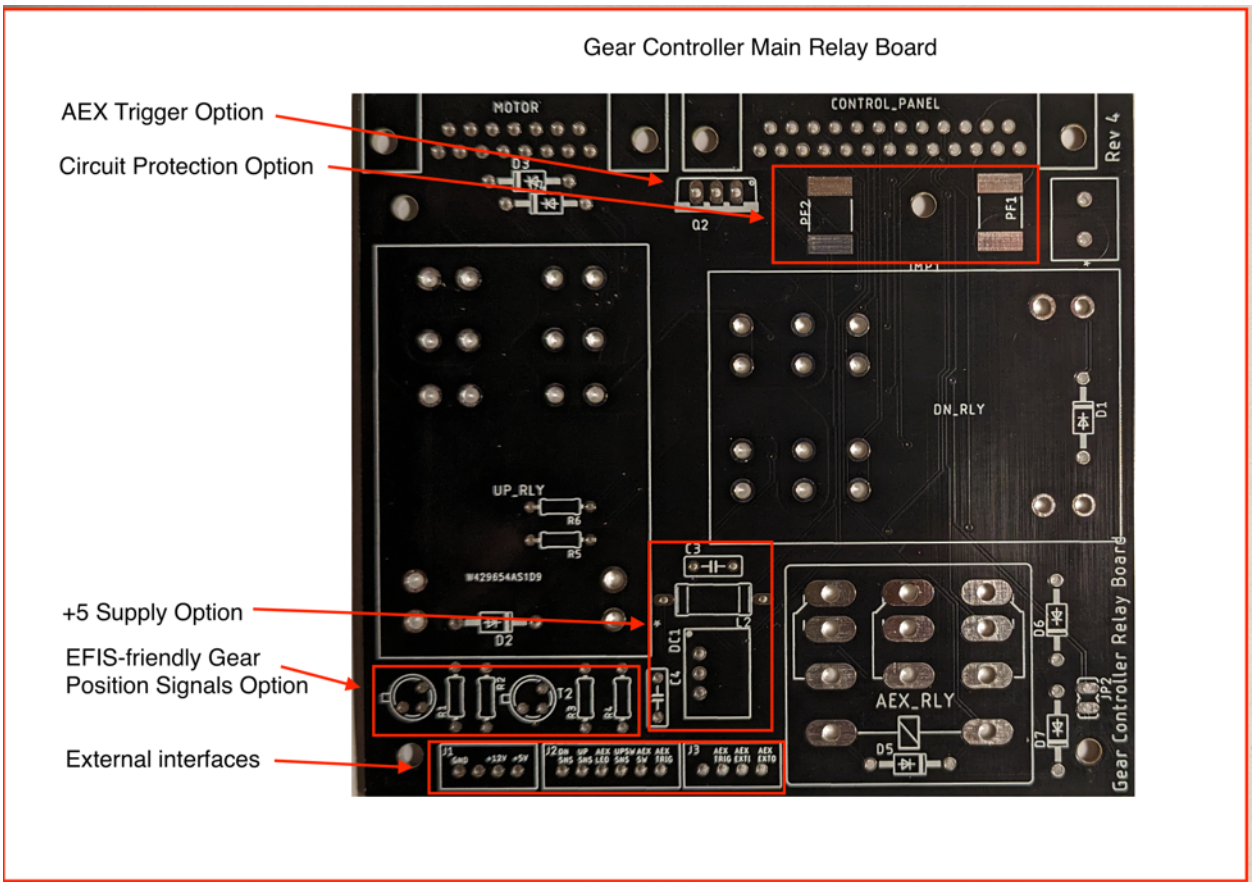

Figure 2 - Main Relay Board

The Parts:

These are all the parts you will need for the Main Relay Board(Figure 3). Depending on what options you choose, some may not be required. Complete BOM's are given below

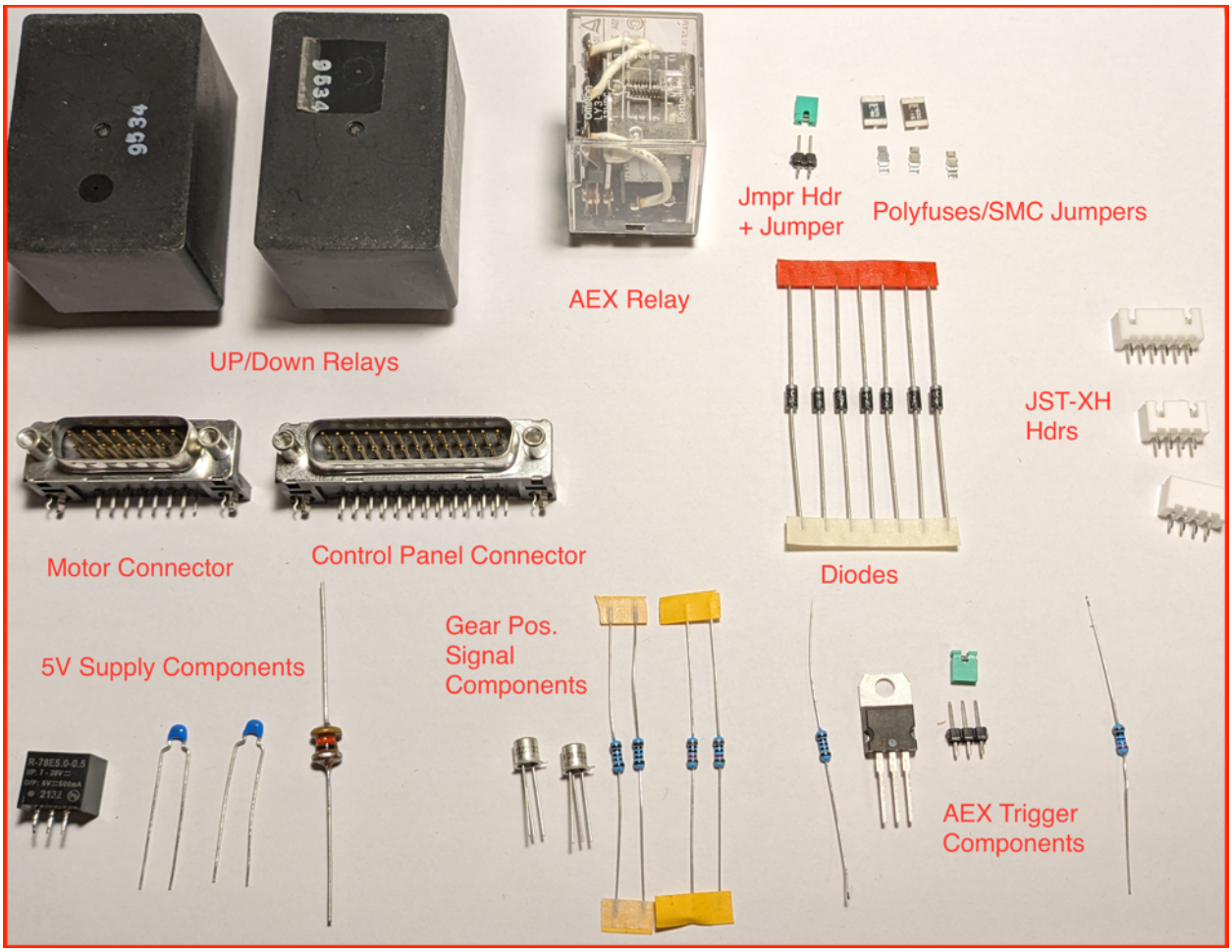

Figure 3 - Main Relay Board Parts

Note: one part is missing from this picture, namely the two-terminal screw-in power connector. It IS included in the basic board BOM though.

Bills of Materials (BOM):

The board can be configured in a number of different ways, so depending on your requirements you may need only some of the parts shown above (and listed below).

Most if not all of the parts are available from Digikey, so I have provided Digikey lists for each different option. You can use these lists to order directly from Digikey or just download them to get the part numbers you can use to order from your preferred supplier.

Basic Board: https://www.digikey.com/en/mylists/list/AF6GQ96Q92
5V Regulated Supply: https://www.digikey.com/en/mylists/list/KT60I4XPJO
Self-resetting Circuit Protection: https://www.digikey.com/en/mylists/list/MVLB1ONKQR
EFIS-friendly Gear Position Signals: https://www.digikey.com/en/mylists/list/UNIHME86OV

Other miscellaneous items like resistors, JST connector kits, stand-offs, mounting hardware and the like are not included in the BOMs. These can be acquired from vendors such as Digikey and Mouser but it is cheaper to get them from Amazon, Robotshop, Sparkfun, Adafruit, etc.

Board Assembly

Generally, we'll install all the small components (resistors, diodes, etc) first since getting to the mounting positions is more difficult once the relays are on. Note that for a full Arduino install, all the optional Relay board components are required. It's helpful to print out a copy of the schematic (Appendix A) as a reference to guide you during board assembly.

1) Install all the 1N4007 diodes(Figure 4). **NOTE: all the diodes are mounted to the top of the board, but NOT diode D5 (the AEX relay flyback diode). D5 must be mounted on the bottom of the board. If D5 is mounted on top then the AEX relay will not sit correctly on the board.** Pay close attention to orientation of the diodes. Here's the board with the diodes placed:

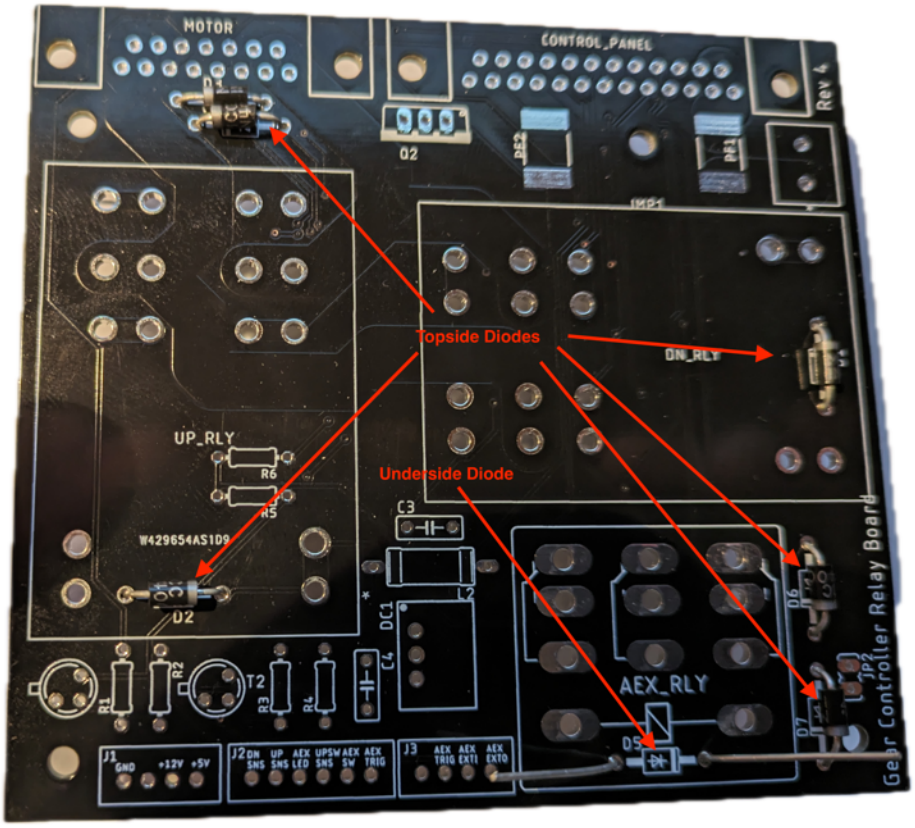

Figure 4 - Main Relay Board, Diodes Placed

2) (Optional) If using the circuit protection, install the two polyfuses PF1 (LF600.16) and PF2 (LF.075.6). Note these are surface mount components, so put some flux on the pads so that solder will flow underneath them

3) If NOT using the circuit protection, solder two SMT jumpers across the pads where PF1 and PF2 would go (Figure 5):

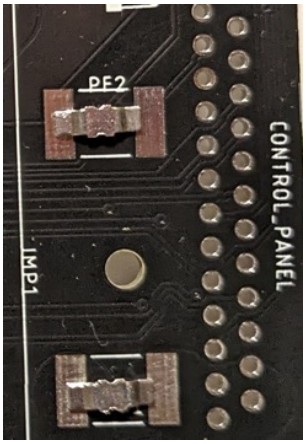

Figure 5 - Main Relay Board, No Circuit Protection

Two segments of #14 bare copper house wiring 8mm long would also work.

4) **If NOT using a separate gear power switch (ie, gear is always powered from the battery or is powered from the master switch)** short the two sides of the DB25 control panel connector together with an SMT jumper using the pads provided on the underside of the board(Figure 6):

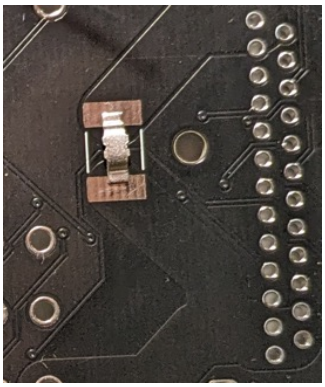

Figure 6 - Main Relay Board - underside, power switch bypass jumper installed

Two segments of #14 bare copper house wiring 8mm long would also work.

5) Install jumper header JP2.
6) Install the 2-port screw-terminal power connector on the right top of the board (just to the left of the revision call-out). Ensure the ports (where the wires are inserted) are oriented to the outside of the board!
7) Inspect the DB15 and DB25 connectors for bent pins and straighten, if necessary. Then install, making sure ALL the pins extend the same distance through the underside of the board. Then solder in place.(Figure 7)

8) (Optional) Install the JST-XH headers. You could always just hardwire the external connections by soldering wires directly to the board, but I recommend using the JST connectors for ease of assembly. Make sure to install the headers so they line up with the component outline (the slots on the headers will point to the outside of the board)(Figure 7)
9) (Optional) If using the +5V supply, install capacitors C1 and C4 and inductor L2. These can be placed in any orientation. Install voltage regulator DC1, ensuring that it is in the correct orientation (matches the outline on the board) (Figure 7).
10) (Optional) If using the gear-up/gear-down signals, install T1, T2, R1, R3, R5 and R6. Note that use of the gear-up/gear-down signals also requires that the +5V supply option be installed as well. Pay attention to the orientation of the 2N2222 transistors – the tab should be pointing in the same direction as the outline on the board (Figure 7).
11) (Optional) If using standard AEX positive +5V triggering install the power transistor Q2 and resistor R2(Figure 7)

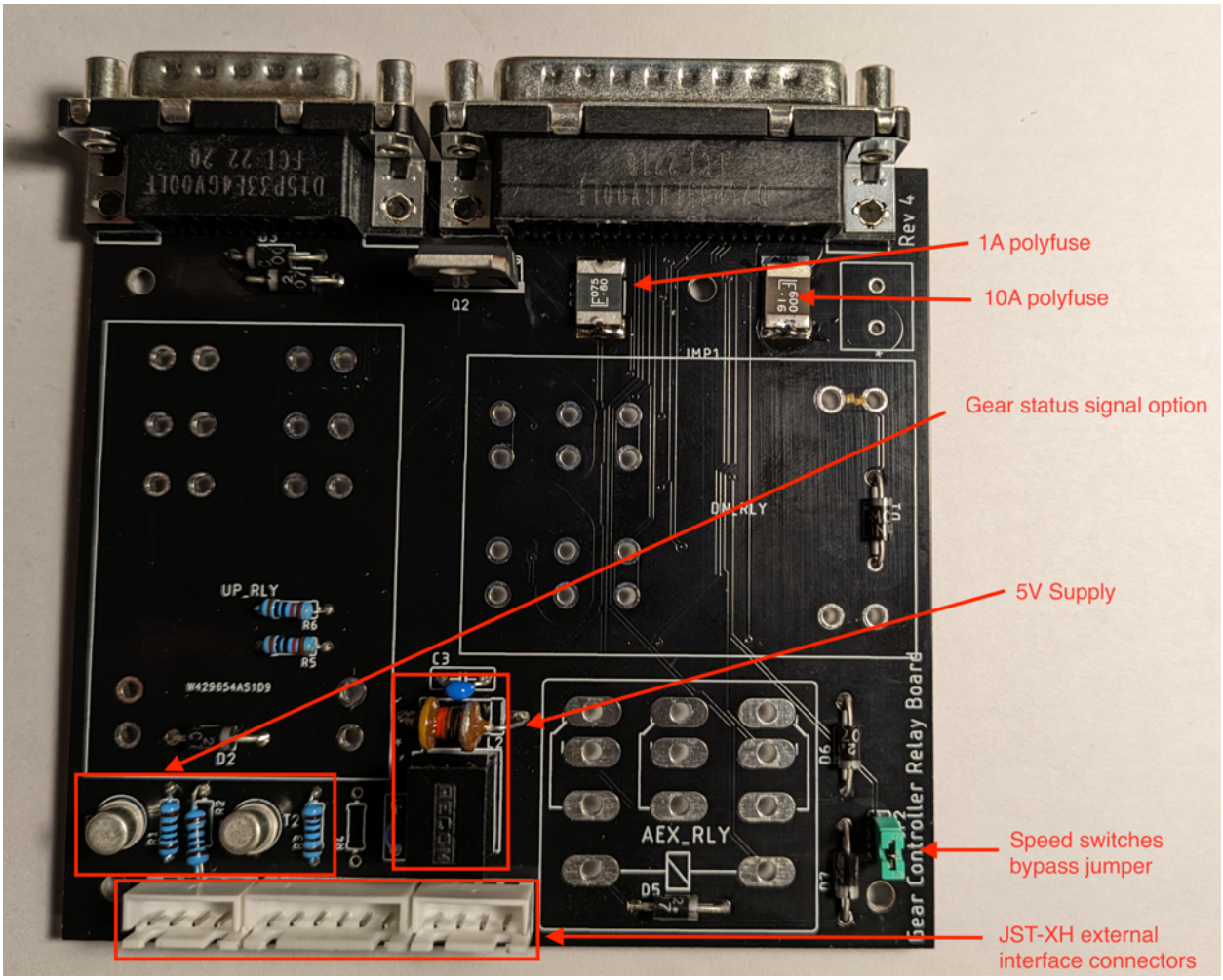

Figure 7 -Main Relay Board, all options installed

12) (Optional, external AEX trigger reversed logic). Insert a jumper on the jumper header taking the place of Q2. The two pins bridged by the jumper should be the two closest to the main power connector (Figure 8).

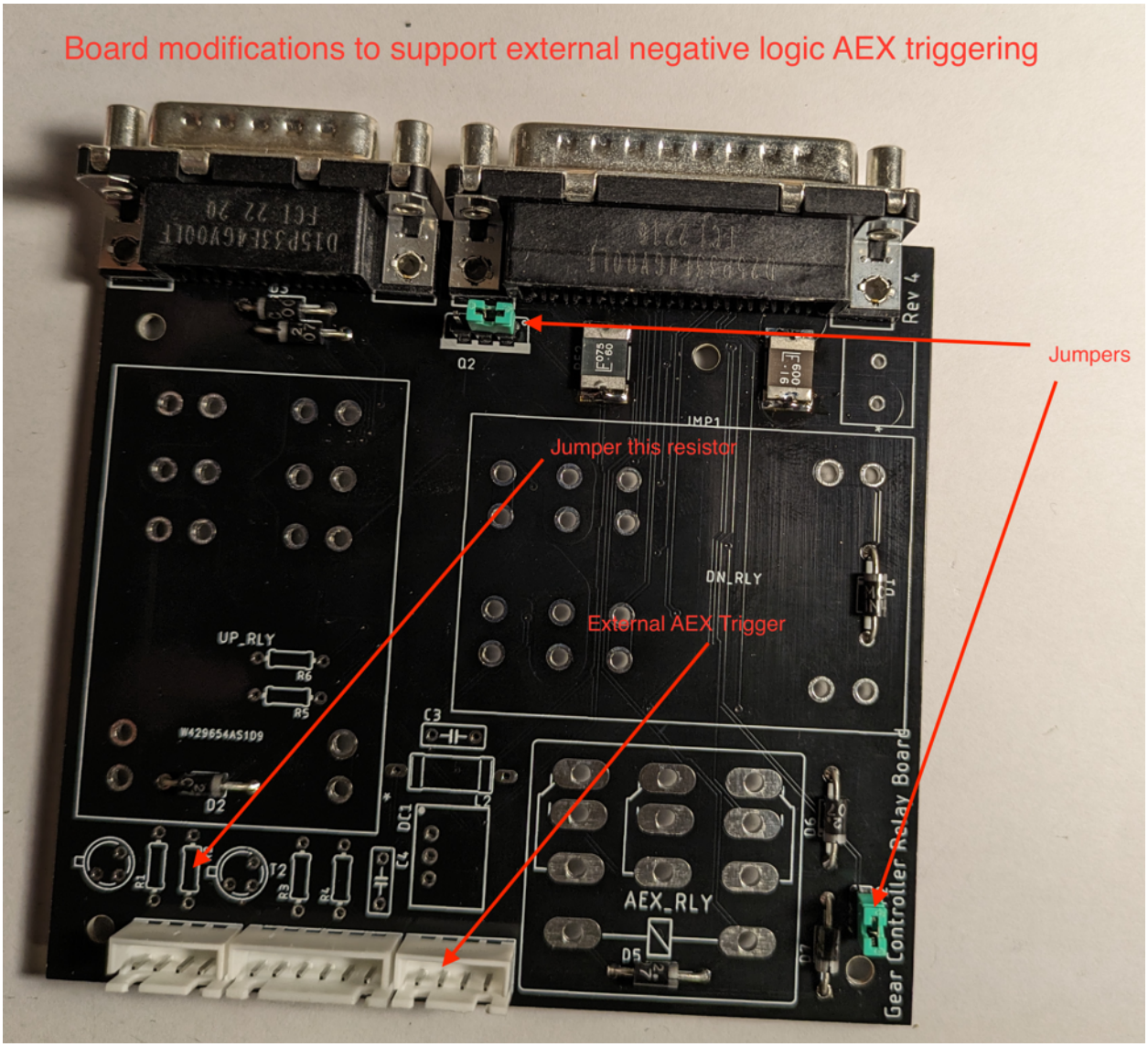

Figure 8 - AEX External Trigger w/Reversed Logic

13) If using the LED-illuminated AEX enable switch called out in the BOM, install current limiting resistor R4

14) Lastly, install the three relays. Note that due to the large hole sizes, these will suck up a lot of solder so make sure you get the hole completely filled in with solder. Adding additional flux to the relay pins and through-hole pads wouldn't be a bad idea. The finished board should look something like this (Figure 9):

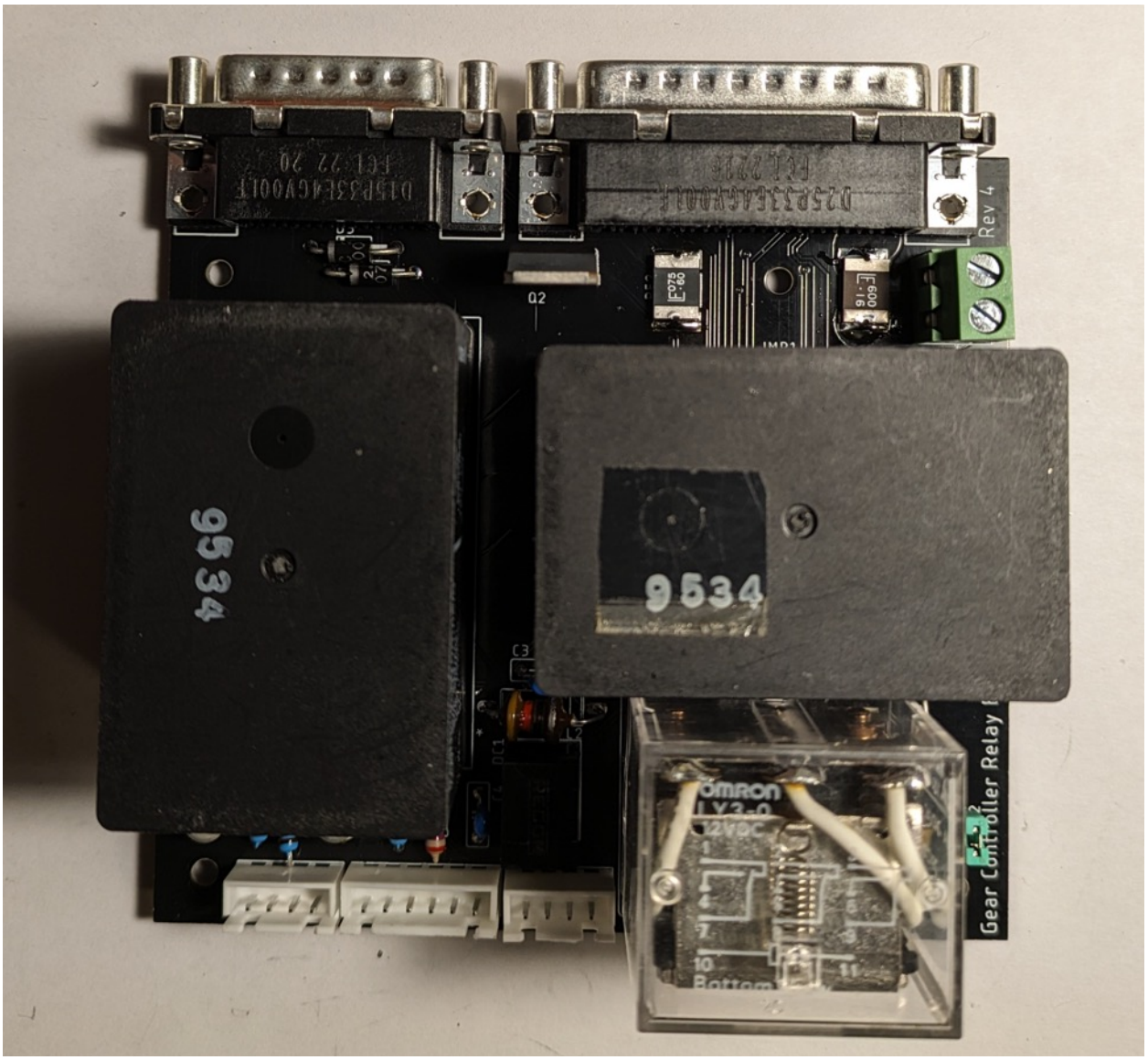

Figure 9 - Main Relay Board complete

15) If using hardware speed switches as per Marc's OG circuit, all you need to do now is construct the DB15 relay interface and DB25 control panel wire harnesses and wire up the switch and trigger interfaces. See Appendix 2 (Wire harnesses) for instructions on that. If using the Arduino, see the next section.

Arduino Custom Shield Assembly Instructions

"Arduino" is the generic name for a family of customizable microcontroller boards that have become very popular in the maker community. They come in a variety of flavours, but for my gear controller implementation I have chosen the "Uno" variant as it is cheap (~ $10) and plenty powerful enough for the task at hand.

The Uno (and indeed, most of the Arduino variants) allows you to add additional hardware components through the use of a "shield", which is essentially a daughter-card that plugs into the top of the board using a standardized set of headers. Multiple shields may be stacked on top of one another, but in our case, we will just be using one custom-built shield.

The Arduino+shield combo:

- Handles serial communications with a variety of LIDARs to obtain altitude data
- Handles serial communications with a variety of EFISs to obtain airspeed data
- Allows for a LIDAR configured to output an analog signal for altitude to obtain altitude data
- Generates an AEX trigger signal based on speed, altitude and RPM (or throttle position) inputs
- Provides AEX status information to the pilot via the status LED on the AEX enable switch
- Optionally, provides AGL altitude and airspeed information to the pilot via an LCD display
- Optionally, provides voice callouts of gear status and AGL altitude directly to the pilot's headset or aircraft intercom

As with the Main Relay board, the shield is somewhat customizable depending on what you are using to provide altitude and airspeed data and what options you choose to implement.

The Board

Here is what the (unpopulated) board looks like, along with the Arduino Uno that it will sit on top of (Figure 10):

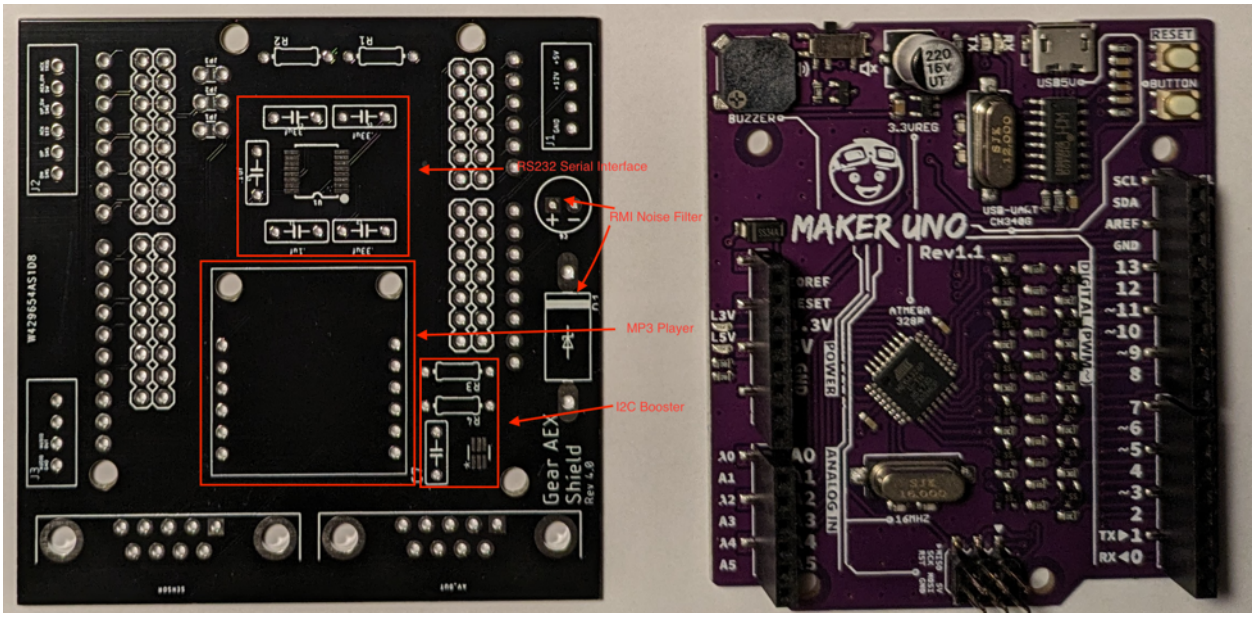

Figure 10 - Arduino Shield Board and Arduino Master Board

The Parts

Here are all the parts you will need. Depending on the options you choose, some will not be required(Figure 11):

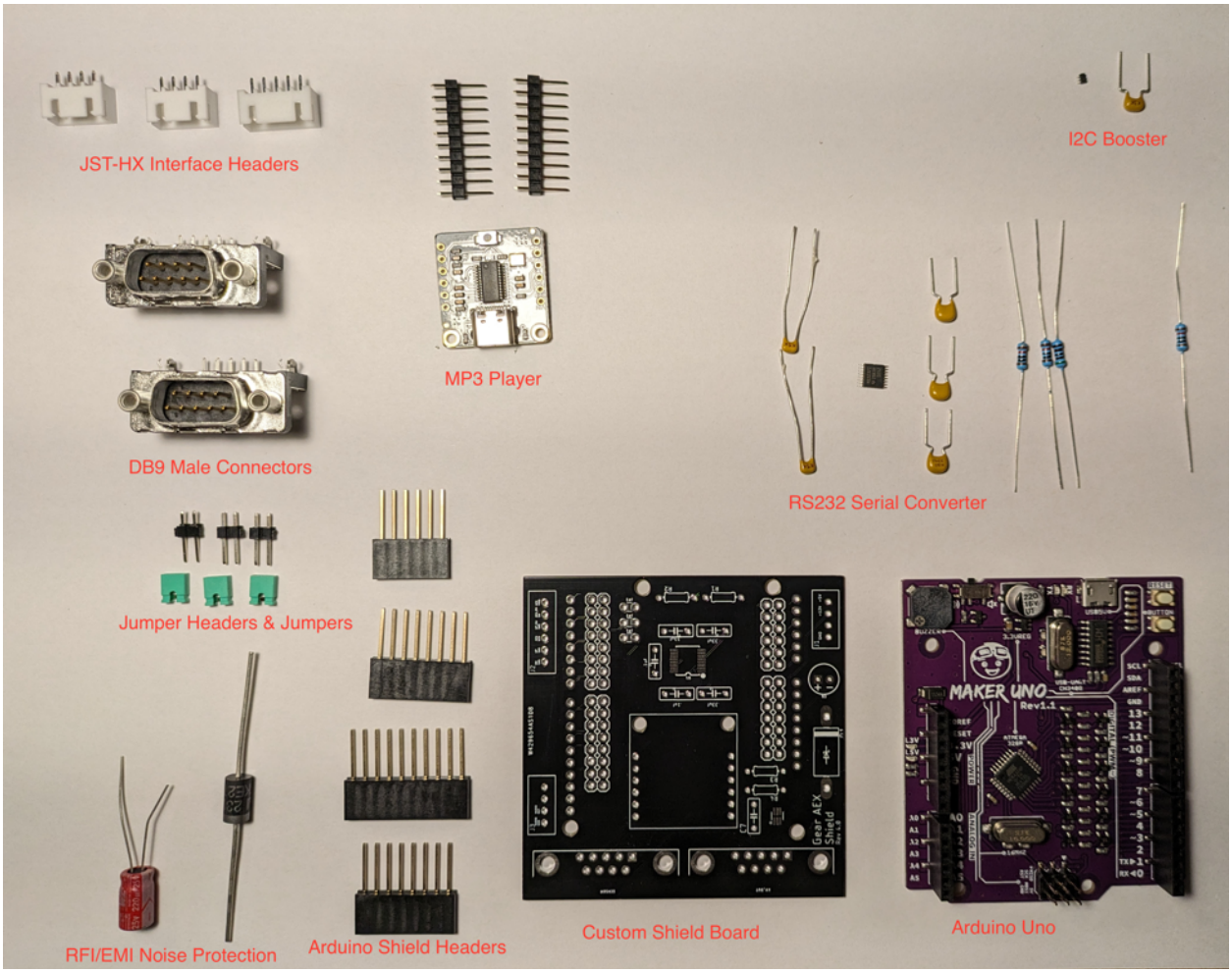

Figure 11 - Arduino Shield Components

Bills of Materials (BOM):

The board can be configured in a number of different ways, so depending on your requirements you may need only some of the parts shown above (and listed below).

Most if not all of the parts are available from Digikey, so I have provided Digikey lists for each different option. You can use these lists to order directly from Digikey or just download them to get the part numbers you can use to order from your preferred supplier.

Basic Custom Shield Board: https://www.digikey.com/en/mylists/list/J6HDY1LBND
LCD Display : https://www.digikey.com/en/mylists/list/G15YV3GI4F

Audio Alerts: https://www.digikey.com/en/mylists/list/GKSGMX4RWC

Other miscellaneous items like resistors, JST connector kits, stand-offs, mounting hardware and the like are not included in the BOMs. These can be acquired from vendors such as Digikey and Mouser but it is cheaper to get them from Amazon, Robotshop, Sparkfun, Adafruit, etc.

Board Assembly

As with the main relay board, all components are installed on the top side of the board.

Unlike the main relay board, we have some SMT (surface mount technology) components to mount. Despite what many Youtube videos claim, these can be tough to hand-solder. You need an appropriately sized (ie very small) soldering tip, a steady hand, and (at least for me), some means of magnification to see what you are doing. If you have never done SMT soldering before, I would suggest getting an SMT practice kit before attempting to solder these components. The main reason for choosing SMT rather than through-hole components was cost and availability (for instance, the DIP version of the MAX3232 used for the RS232 interface is $15 vs $1 for the SMT version).

Okay, here we go:

1. Mount the MAX3232 chip. Since this is an SMT chip, flood the pads with some flux, add a small dab of solder on one of the corner pads, position the chip and tack-weld it down. Move to the other side of the chip, get a small blob of solder on the iron and draw it across the line of pins. You want the solder to flow underneath the pins. When the side opposite the tacked down pin is done, move to the other side. You will most likely get solder bridges, these can be fixed by repeatedly drawing the tip across the line of pins, or if necessary, de-soldering wick can be used to soak up unwanted solder. When done, you it should look something like this:

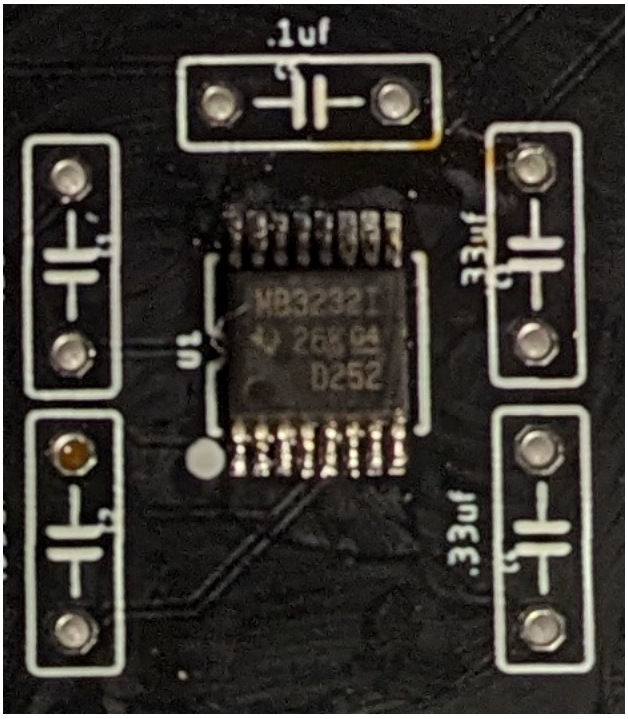

Figure 12 - MAX3232 chip installed

2. Confirm that there are no solder bridges by carefully inspecting with a magnifier and/or testing adjacent pins using a multimeter
3. Install capacitors C1(.1uf), C2(.33uf), C3(.33uf), C4(.33uf) and C5(.1uf). Orientation doesn't matter.
4. (Optional) If using the LCD display, install the SMT I2C booster chip LTC4311. Use the technique outlined in Step 1 to solder this very tiny component. Install capacitor C7(.01uf).

   Editor's note: it may not be necessary to do this step. The main reason for the booster is to protect against noise on the I2C bus which is used to talk to the remote display. I have had good luck using shielded wire for the bus lines. My advice would be to skip this step initially and only add the LTC4311 chip later if the display seems flakey.
   - tdh
5. Install resistors R1 (10K), R2(5K), R3 (10K) and R4 (10K).
6. Install the Arduino shield headers. Take care to get them perpendicular to the board or it will be difficult to insert the shield onto the Arduino. (If you have a solderless breadboard, a good way to do it is to insert some (male) jumper headers into the breadboard, plug the female side of the Arduino headers into those and then invert the shield onto the headers so you can solder on the back side).
7. Install the JST-XH external interface headers J1, J2, J3. Make sure to orient them so that the slots on the headers point outwards.
8. Install the DB9 connectors.
9. Install the transorb D1 and capacitor C6 (100uf). These components make the shield less susceptible to RMI/EMI and noise spikes from the power supply line (say, from engine startup). Note that orientation DOES matter for these components(Figure 13):

Figure 13 - Arduino Shield Transorb and Power Supply Filter Capacitor Installed

   For the electrolytic cap, the lighter stripe with a minus sign (-) on it should be to the outside of the board.

10. (Optional) If using the audio alerts option, install the supplied headers onto the MP3 daughter card. These may require shortening. If so, trim to the appropriate number of

pins using diagonal cutters or a utility knife. Then install the headers first onto the MP3 daughter-card, and then solder the MP3 daughter-card to the shield board.

11. Install the three jumper headers J1, J2, and J3
12. If your EFIS outputs TTL-level serial signals (unlikely), then install a jumper on JP1. Otherwise, install jumpers on JP2 and JP3. This enables the MAX3232 RS232-serial conversion chip. The only EFIS that I'm aware of that outputs TTL-level serial data is the MGL ASX-1; all others (Dynon, Garmin, GRT) use the RS232 serial protocol.

13. This completes the assembly of the shield, and the completed board should look something like this(Figure 14):

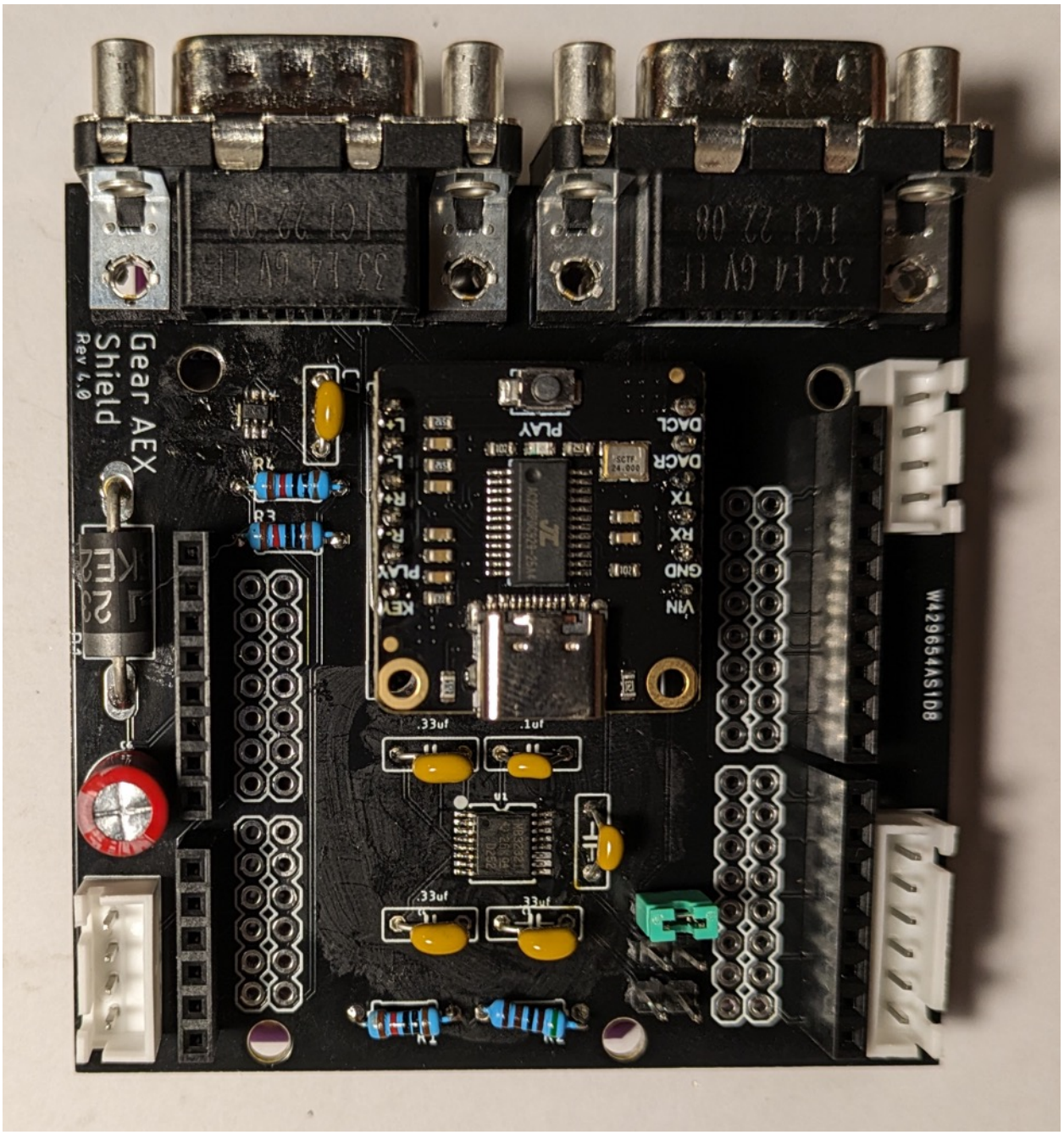

Figure 14 - Arduino Shield Assembly Completed

While the hardware assembly is done, the Arduino+shield combo is useless in this state – to make it useful, we need to configure and load the microcontroller firmware. Instructions for this are in the next section.

Configuring and loading the Arduino firmware

Ok, we've built the hardware, now we need to make it work. The Arduino ecosystem is well developed and documented, so I think it would be redundant and pointless to go into any depth here. Head on over to Arduino.cc and download the Arduino IDE for your preferred platform. Try out some tutorials, make sure you are able to upload a program (or "sketch", as they are called in Arduino-land), and once you are comfortable with the IDE and are able to upload a tutorial sketch onto your board then proceed.

First, prepare your IDE with the libraries used by the GearAutoExtend firmware.
Simply copy the contents of the "Libraries" folder from wherever you downloaded the GearAutoExtend project to the "Libraries" folder in the Arduino home directory (where you installed the Arduino IDE). You can also use the IDE to install the latest version of the required libraries, using the "Tools->Manage Libraries" menu, but if you just copy the supplied libraries then you know have versions that will be compatible with the code that uses them.

Copy the whole folder "GearAutoExtend" from wherever you downloaded it from into the Arduino home folder (typically just called "Arduino"). Fire up the IDE and navigate to the folder GearAutoExtend, expand it and open the sketch file "GearAutoExtend.ino". You should see something like this (Figure 15):

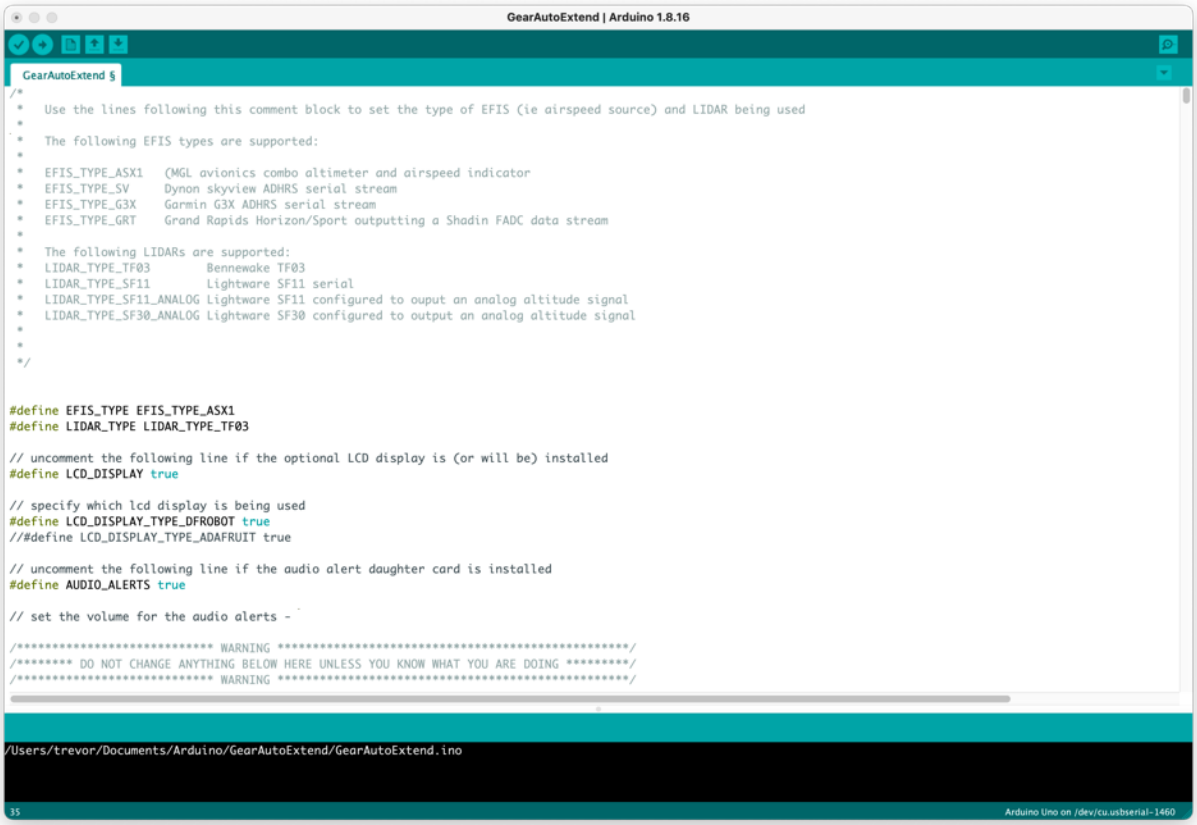

Figure 15 - Arduino IDE

Examine the comment block at the top - it lists the supported EFIS and LIDAR types. If you see your EFIS and LIDAR there, then congratulations, life is easy. (If you don't, well then welcome to the world of Arduino software development ;)).

Assuming your LIDAR and EFIS are supported, then configuring the software is a matter of setting the two #define statements at the top to the correct values for your hardware (cut and paste from the comment block recommended for this).

Eg, if you have a SF11 LIDAR which is configured to provide an analog voltage as an altitude signal, and you have a Garmin G3X EFIS, then you would change the two #define statements to:

#define EFIS_TYPE **EFIS_TYPE_G3X**
#define LIDAR_TYPE **LIDAR_TYPE_SF11_ANALOG**

By default, the LCD display support is included. It can be removed by changing this line:

#define LCD_DISPLAY

to

//#define LCD_DISPLAY

(i.e., change it into a comment). Deleting the line altogether also works. BTW, nothing bad will happen if you leave the LCD screen support enabled but don't install the LCD screen module.

Next, the audio alerts are enabled by default but you can disable them by changing the following line:

#define AUDIO_ALERTS

to

//#define AUDIO_ALERTS

Again, nothing bad will happen if you leave audio alerts enabled but don't have the MP3 daughter card installed. However, since the initialization code looks for the audio card if enabled, it will be slower to initialize as it tries to talk to a card that doesn't exist.

That's it! Now, ensure that the code compiles by clicking on the compile button at the top left of the code window(Figure 16):

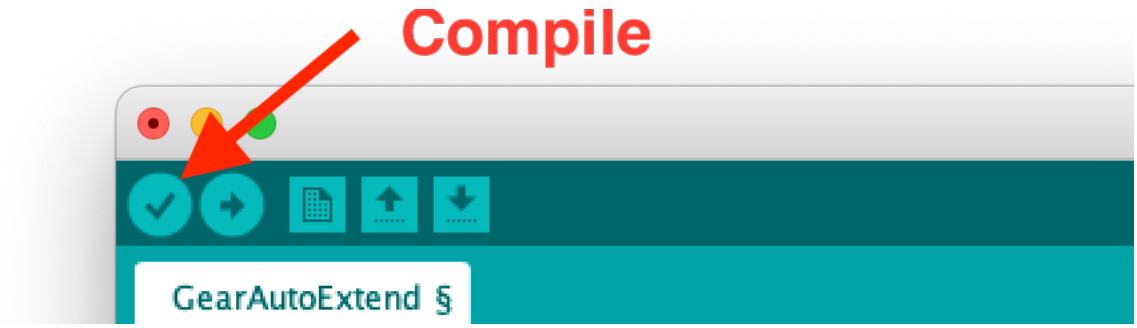

Figure 16 - Arduino IDE Compile Check

If no error messages occur, then upload the code to the board(Figure 17):

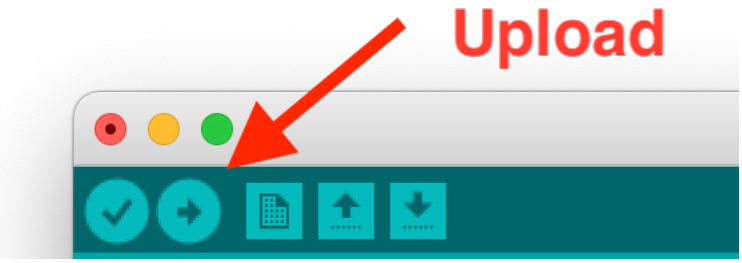

Figure 17 - Arduino IDE Upload Firmware to Board

Note that the serial port used by the LIDAR uses the same GPIO pins (D0 and D1) used to upload firmware to the board so ensure that an active LIDAR is not plugged it while the firmware is being updated (you'll likely get board timeout messages while uploading the firmware if a LIDAR module is plugged in and powered up)

Setting up the Audio Card

The DFROBOT DFR0768 daughter-card is a complete MP3 player with 128MB of onboard flash memory for mp3 audio files. The Arduino talks to it using a software serial connection on GPIO pins D8/D9. Before it can be used, we need to load the audio files onto the flash memory and configure some parameters.

Loading the Sounds

This is easy – use your micro-USB cable to connect the DFR0768 to your computer. It will show up as a USB flash drive called NONAME. Just copy the files in the Sounds directory of the GearAutoExtend folder to the root directory of the NONAME flash drive. Make sure to EJECT the flash drive before disconnecting the USB cable.

Configuring the Card

There are some persistent defaults we need to override, namely the startup mode, startup sound and serial communications baud rate (the default baud rate is too fast for an Arduino's software serial connection). A special-purpose one-time sketch has been provided for this called mp3_player_init.

First, plug your computer into the Arduino using the USB-micro cable.
Copy the mp3_player_init folder into your Arduino home directory, fire up the Arduino IDE and open sketch "mp3_player_init"". Open up a serial console using "Tools->Serial Monitor". Set the baud rate of the serial monitor to 115200.

Now upload the sketch to the Arduino.

The following messages should display on the serial monitor:

```
Trying to start at default baud rate of 115200…
Success! Setting Gear Auto-extend defaults...
VOL: 20
Playmode:1
MP3 Player setup is complete.
Restart required
```

Now, unplug the Arduino, then plug it back in.

If the MP3 player initialization was successful, the following messages should appear:

```
Unable to communicate at 115200 baud, trying 19200...
Success!
VOL: 20
Playmode:3
```

This indicates that the reset was successful and the module is now ready to be used for GearAutoExtend audio alerts.

Control Panel Board Assembly Instructions

I'm not convinced that having a separate board for the control panel saves you a whole lot of time vs just wiring up the discrete components, but it does make it a little more convenient since it can be done on the bench instead of trying to butt-splice a ton of wires together whilst huddled in the cockpit or needing to route a long snake with a DB25 connector on the end. But I needed the board for another (as yet unreleased) feature, so WTH, here is it.

First, here's the control panel(Figure 18):

Figure 18 - Nose Gear Control Panel

Since I'm using the Arduino microcontroller I need a power switch, then the standard 3-position gear switch, the 3 status LED's, and an AEX enable switch. My gear selector switch is a wimpy miniature toggle, which I prefer over the honking big standard toggle switch, but each to his/her own. The current flows through this switch are small (< 1A) so the current rating of a bigger switch is not required, and 4PDT (or 3PDT) normal-size toggle switches take up a ton of real-estate behind the 3"x 1.5" panel, thus the small toggle. The AEX enable switch is independently illuminated and communicates AEX status via blink codes, in the absence of the optional LCD panel.

The Board

Pretty straightforward, basically just a bunch of connectors. Note that all the connectors are currently JST-XH (25mm pitch) connectors but a future revision will change the LED connectors to JST-PH, since the -XH connectors don't fit through the .25" hole need for the panel LEDs (Figure 19)

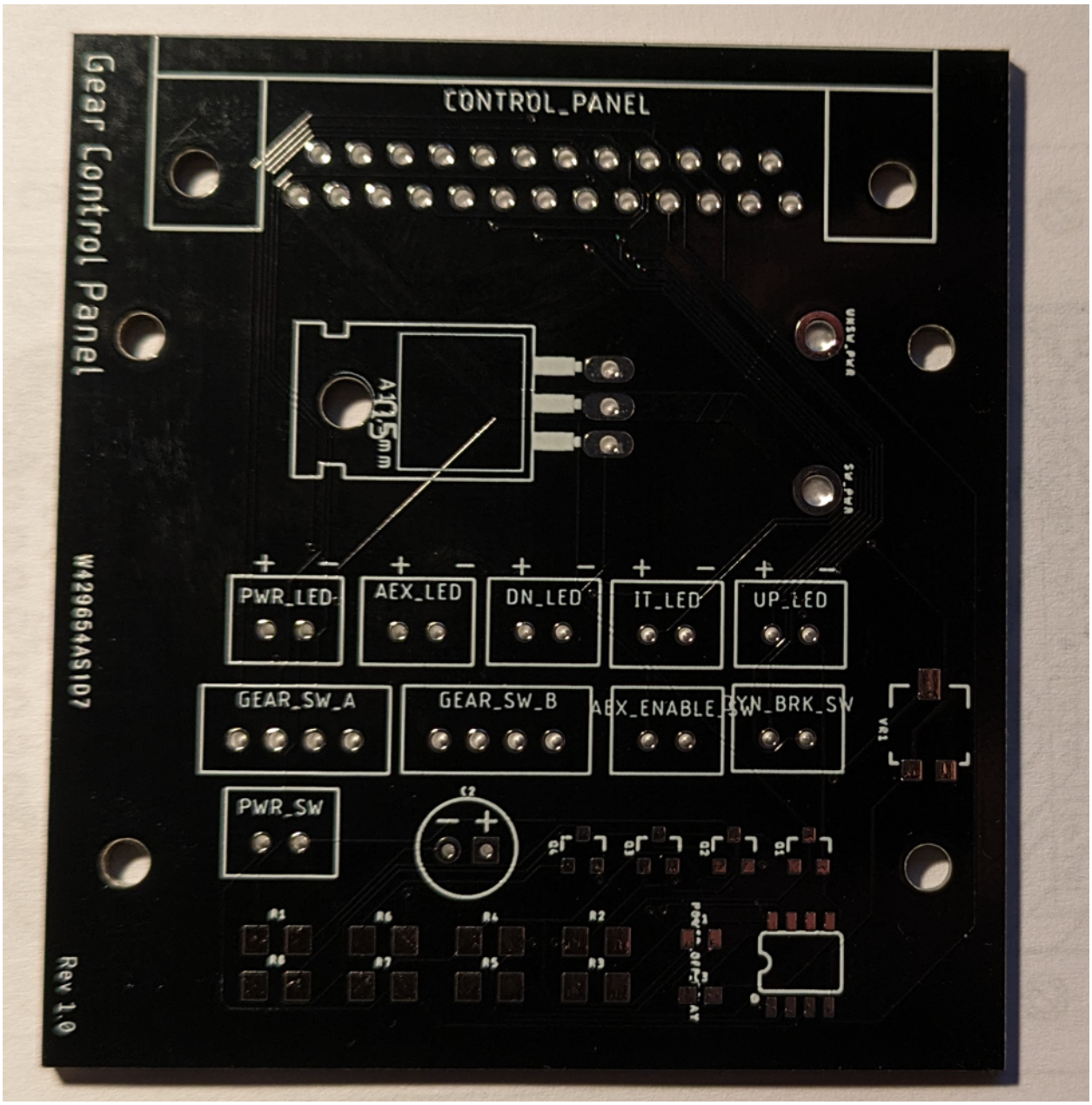

Figure 19 - Control Panel Board

The Parts

Switches, connectors, LEDs (Figure 20):

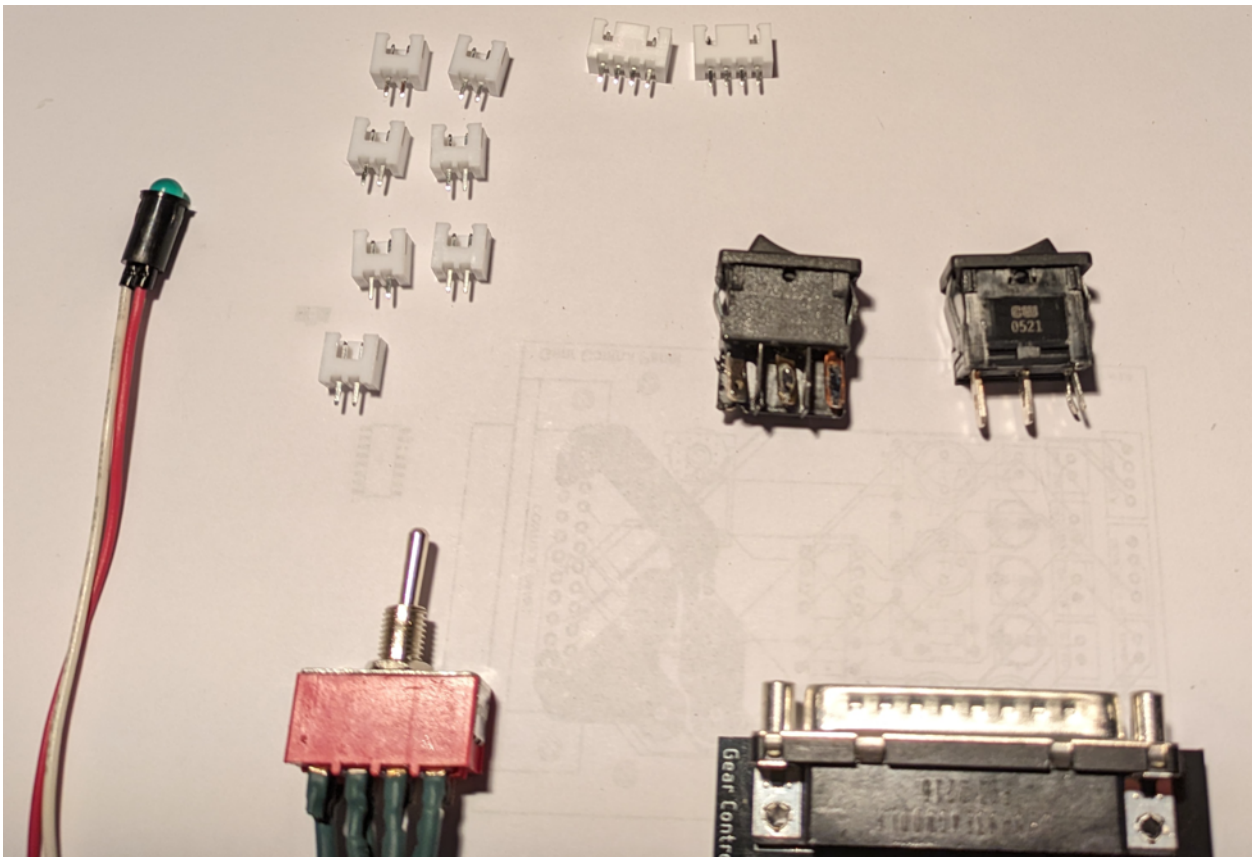

Figure 20 - Control Panel Components

BOMs:

https://www.digikey.com/en/mylists/list/U84KF1U2JM

Skip the SPST power switch if you have always-on power.
Change the 4PDT toggle to a 3PDT toggle if not using the Arduino

Board/Control Panel Assembly

Note that to make use of the JST-XH connector headers on the board means you will need the appropriate matching JST-XH plugs, push-in terminals and crimper. JST-XH kits with crimper included are ~$30 from Amazon, OR sets of pre-crimped JST-XH 2-pin and 4-pin cables are readily available on Amazon for ~$10 (just search for "JST-XH"). Alternatively, the components can just be wired directly to the board which is easier but means the control panel is permanently attached to the board because of the front-mount components (LEDs). Other combinations are possible, for instance the rocker switches spec'ed out in the BOM accept quick-disconnect terminals so the leads could be hard-wired to the board and the switch would still be detachable via the quick-disconnects.

The following instructions assume that the JST connectors are being used throughout.

So:

1. Mount the DB25 connector.
2. Mount the 7 JST connector headers. Orient the headers so that the open side is toward the front of the board (control-panel side). **NOTE: the 2-pin connector labelled "PWR_SW" is NOT used.**
3. Assemble (or obtain) 7 2-wire cables with JST-XH 2-pin plugs, approximately 8cm long
4. Assemble (or obtain) 2 4-wire cables with JST-XH 4-pin plugs, approximately 8cm long
5. (Prep Power sw leads) Solder a 10cm length of min. #16 wire (#14 preferred) to the pads labelled SW_PWR and UNSW_PWR. Then crimp a blue .187 PIDG connector to each free end. These will be used to attach the power switch to the board.
6. (Prep Power sw LED leads). Strip and tin the ends from one of the 2-wire cables, slip some shrink wrap on each one, then solder to the LED tabs on the power switch. Take care to get the correct orientation – the lead on the "+" terminal of the power switch should go to the "+" terminal of the JST connector header labelled PWR_LED.
7. (Prep AEX sw leads) Crimp the red .110 PIDG connectors to the bare ends of three of the 2-wire cables. These will be used to connect the AEX switch to the board.
8. (Power switch LED current-limiting resistor) Mount the 3.3K SMD resistor between the pads labelled "R1"

9. (Attach 4-wire JST cables to the toggle switch solder tabs). Using a sharpie, label one of the 4-wire cables “A” and the other one “B”. Also, with the switch actuator facing away from you label each switch column (i.e. pole) 1-2-3-4 from left-to right on the top of the switch. Before beginning to solder the wires on the switch solder lugs, strip and tin about 1cm of wire on each end and put a piece of shrink wrap on each of the eight wires. Attach the two JST four-wire cables as per the following diagram (Figure 21):

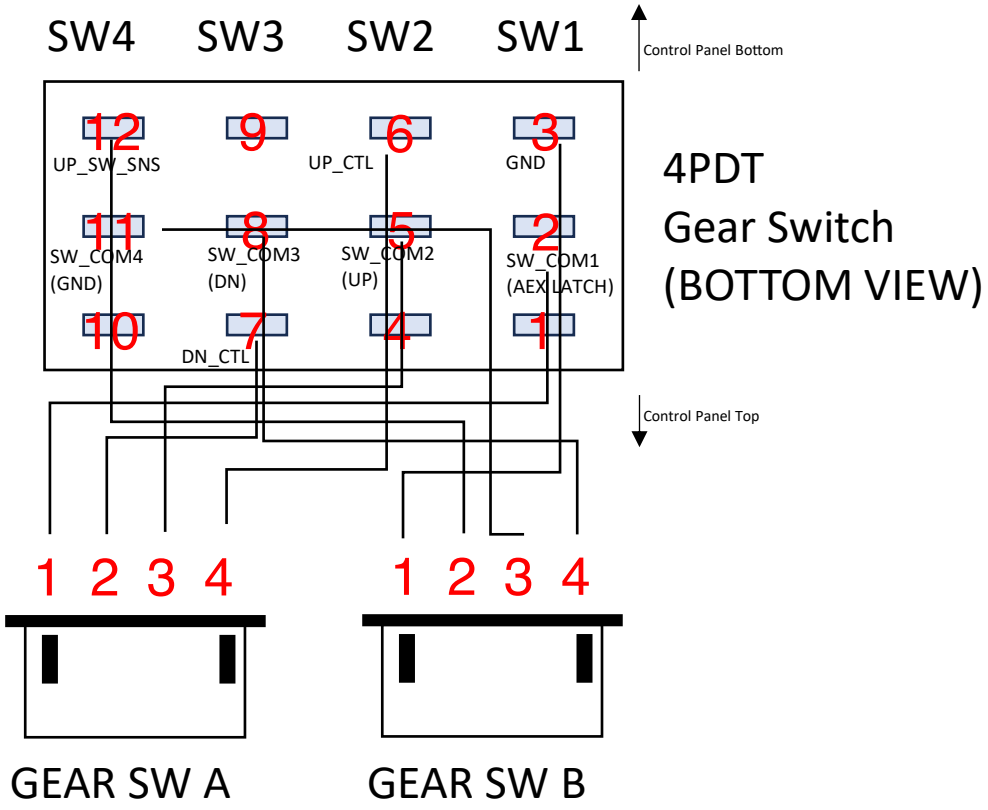

Figure 21 - Control Panel Gear Position Select Switch Wiring

When soldering the solder tabs on the switch, bend the tinned end into a u-shape and hook it through the hole in the solder tap, crimp the “hook” tight onto the tab and then solder in place. Slip the shrink-wrap down over the solder tab and heat-shrink everything in place.

10. (Prepare LEDs) Trim the flying leads on the panel LED’s to 8cm. Insert the three panel LEDs into the control panel (note: the JST-XH plugs will NOT fit through the .25” hole required for the panel-mount LED’s. A future board revision will use smaller connectors). Strip and tin the ends, then slip some shrink-wrap on each one. Identify the ground pin (“-“) on the appropriate board connector and solder the ground line from the plug to the ground (black) lead of the LED. Then do the same for the supply lead (red or white).

Shrink-wrap the exposed inline-connections.

11. (Connect components to board). Mount the switches to the control panel. The lighted switches should be oriented so the that the LED window on the switch is at the top. Plug in all the JST connectors (make sure to get the two 4-pin SW-A and SW-B connectors into their correct headers). Attach the soldered-in power-supply lines to the power switch using the PIDG quick-disconnects. Attach the AEX switch LED 2-wire cable to the top two terminals on the switch, such that the left side terminal (looking down from the top) goes to the "+" pin on the board connector, and the right side terminal goes to the "-" pin on board. Connect another two-wire cable from the "AEX_ENABLE" connector to the left-side terminals on the switch and the other cable from the "DYN_BRAKE" connector to the right-side terminals on the switch.

    The completed control panel board should look something like this (Figure 22):

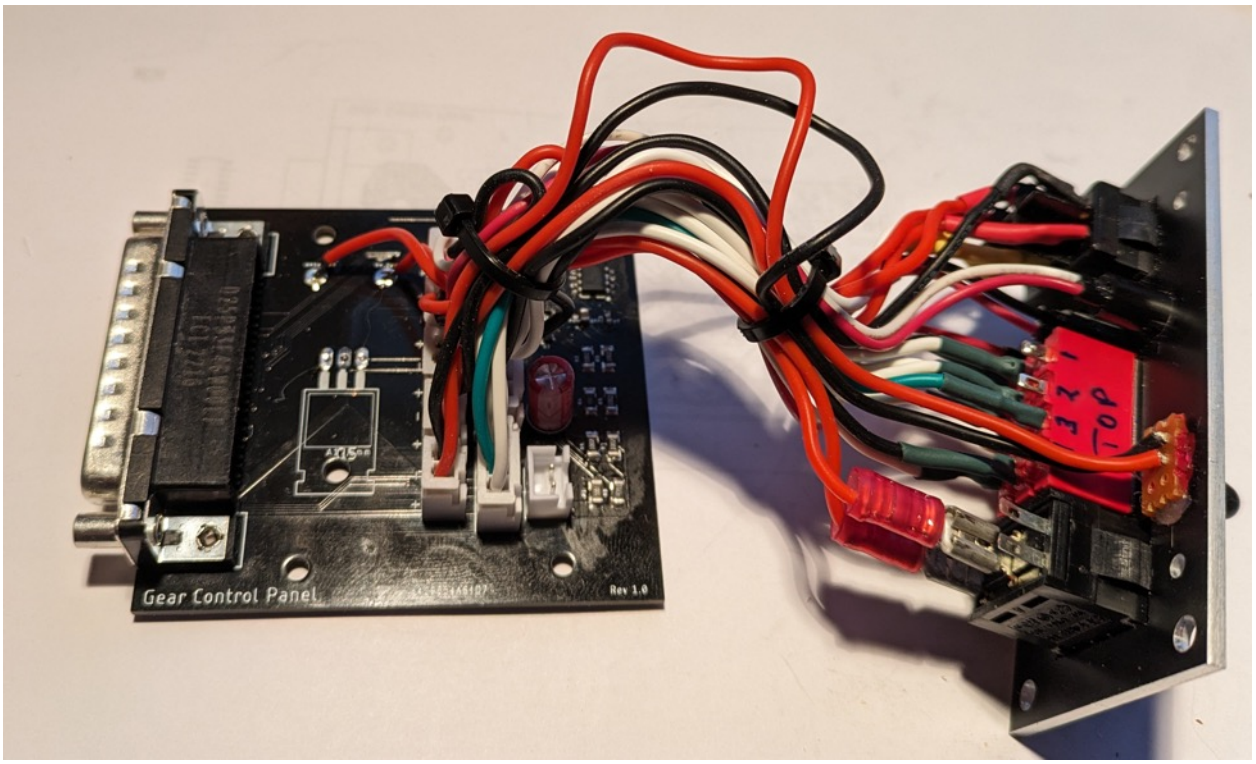

Figure 22 - Control Panel Board Completed

As a QC check it's not a bad idea to pull out your multimeter and confirm that all the pins on the DB25 connector are going to the correct places. Turn all the switches OFF (and gear switch to middle position) before you do this test.

PIN	Function	Control Panel Location
1	UNSW_PWR	Power switch top terminal
2	UNSW_PWR	Power switch top terminal
3	UP_CTL	Gear switch SW2 bottom row
4	SW_COM3	Gear switch SW3 middle row
5	DN_CTL	Gear switch SW3 top row
6	UP_LED	UP (top) LED power lead
7	DN_LED	DN (bottom) LED power lead
8	IT_LED	In-transit (middle) LED power lead
9	AEX_ENABLE_SW	AEX enable switch, left pole, bottom terminal
10	AEX_LED	AEX enable switch LED, left, top position
11	GND	All LED GND leads, power switch LED top right terminal, AEX enable switch top right terminal, Gear switch SW4 middle row, gear switch SW1 bottom row
12	SW_PWR	Power switch bottom terminal
13	SW_PWR	Power switch bottom terminal
14	UNSW_PWR	Power switch top terminal
15	UNSW_PWR	Power switch top terminal
16	UP_SW_SNS	Gear switch SW4, bottom row
17	SW_COM2	Gear switch SW2 middle row
18	n/c	
19	SW_COM1	Gear switch SW1 middle row
20	DYN_BRAKE_OUT	AEX enable switch, right pole, bottom terminal
21	DYN_BRAKE_IN	AEX Enable switch, right pole, middle terminal
22	n/c	
23	n/c	
24	SW_PWR	Power switch bottom terminal
25	SW_PWR	Power switch bottom terminal, AEX switch left pole middle terminal

Appendix 1 – Schematics

The Eagle schematic (.sch) and board layout (.brd) files are part of the download package, but just for reference I have included them here. If there are any discrepancies between these printed versions and the Eagle files, then the Eagle files should be considered definitive.

Main Relay Board (Figure 23):

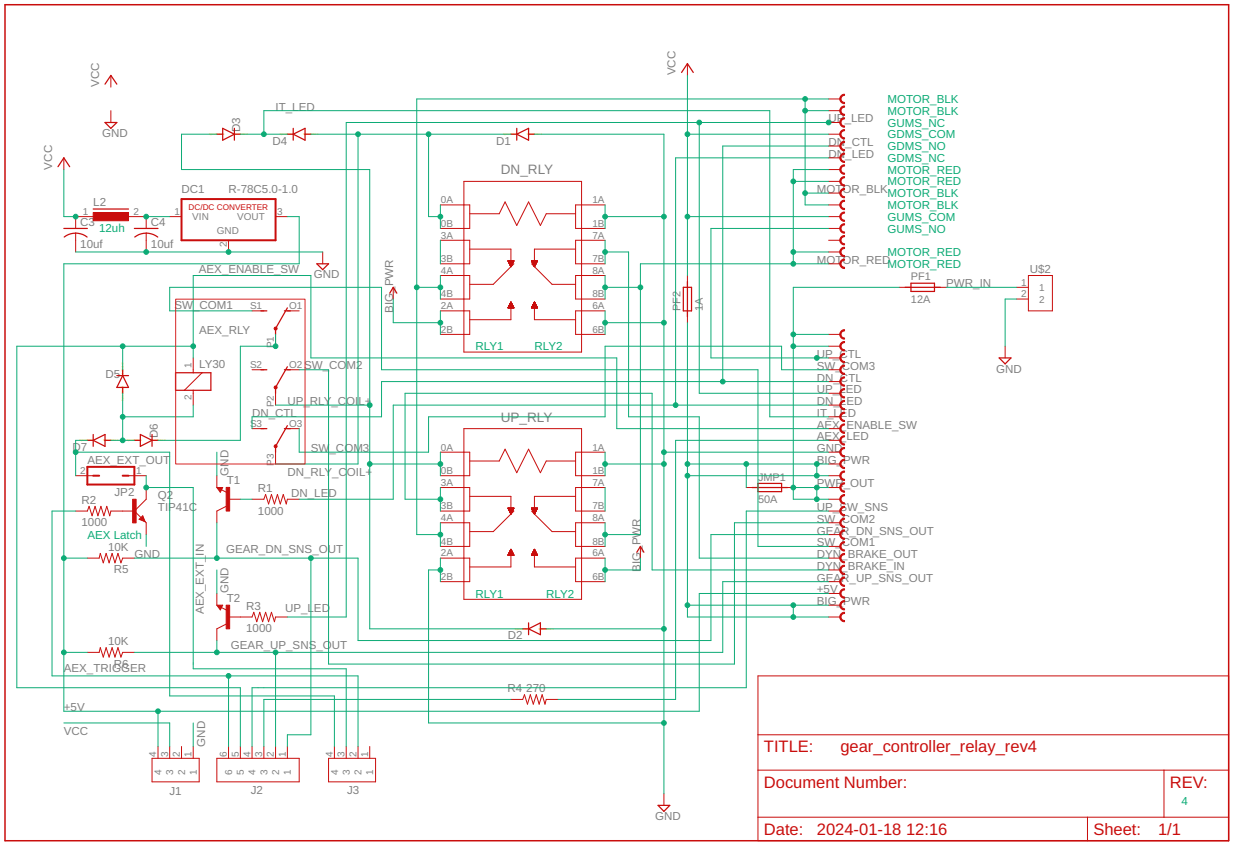

Figure 23 - Main Relay Board Schematic

Arduino Shield (Figure 24):

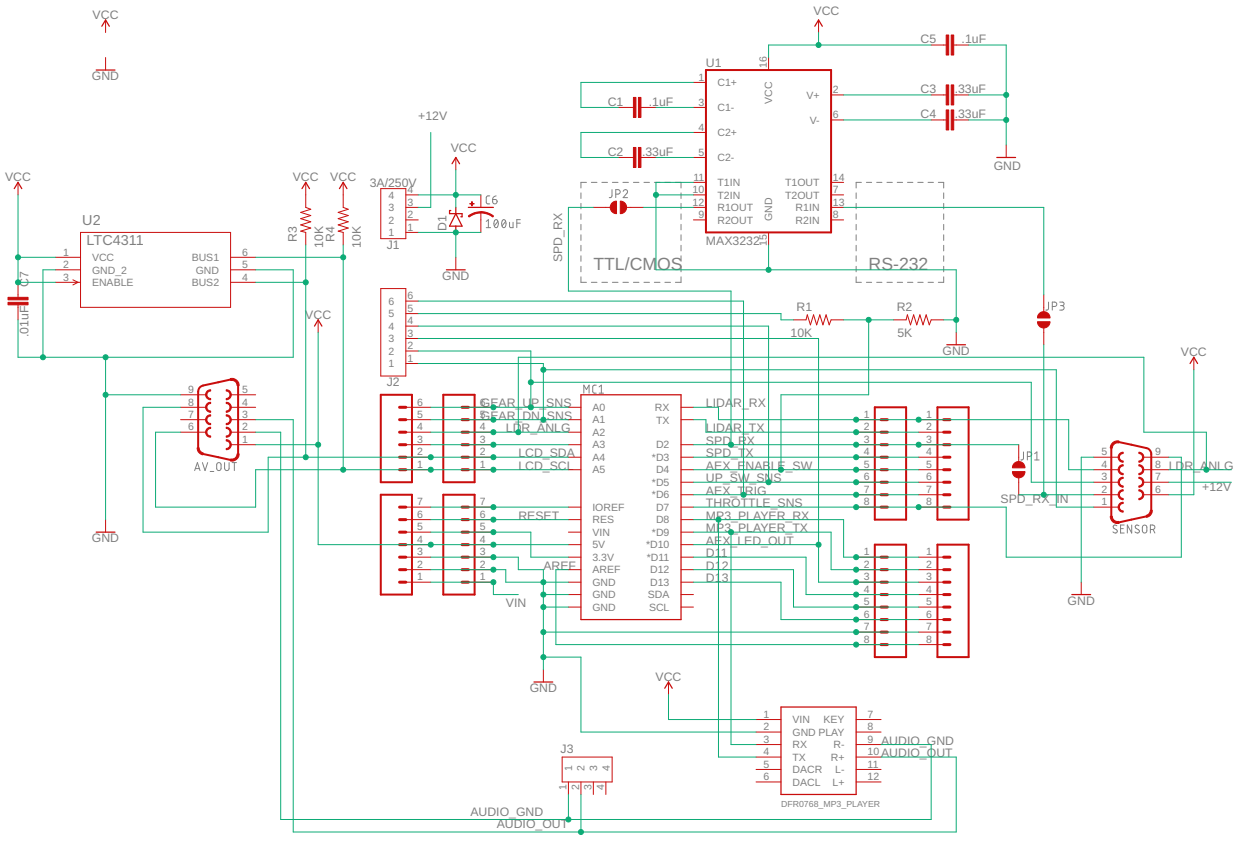

Figure 24 - Arduino Shield Board Schematic

Control Panel Board (Figure 25)

Note that the top section of this circuit with the MOSFET and 555 timer implements the soft power on/off with auto-off delay feature. This is still in development, and will likely change in the future, so ignore it for now.

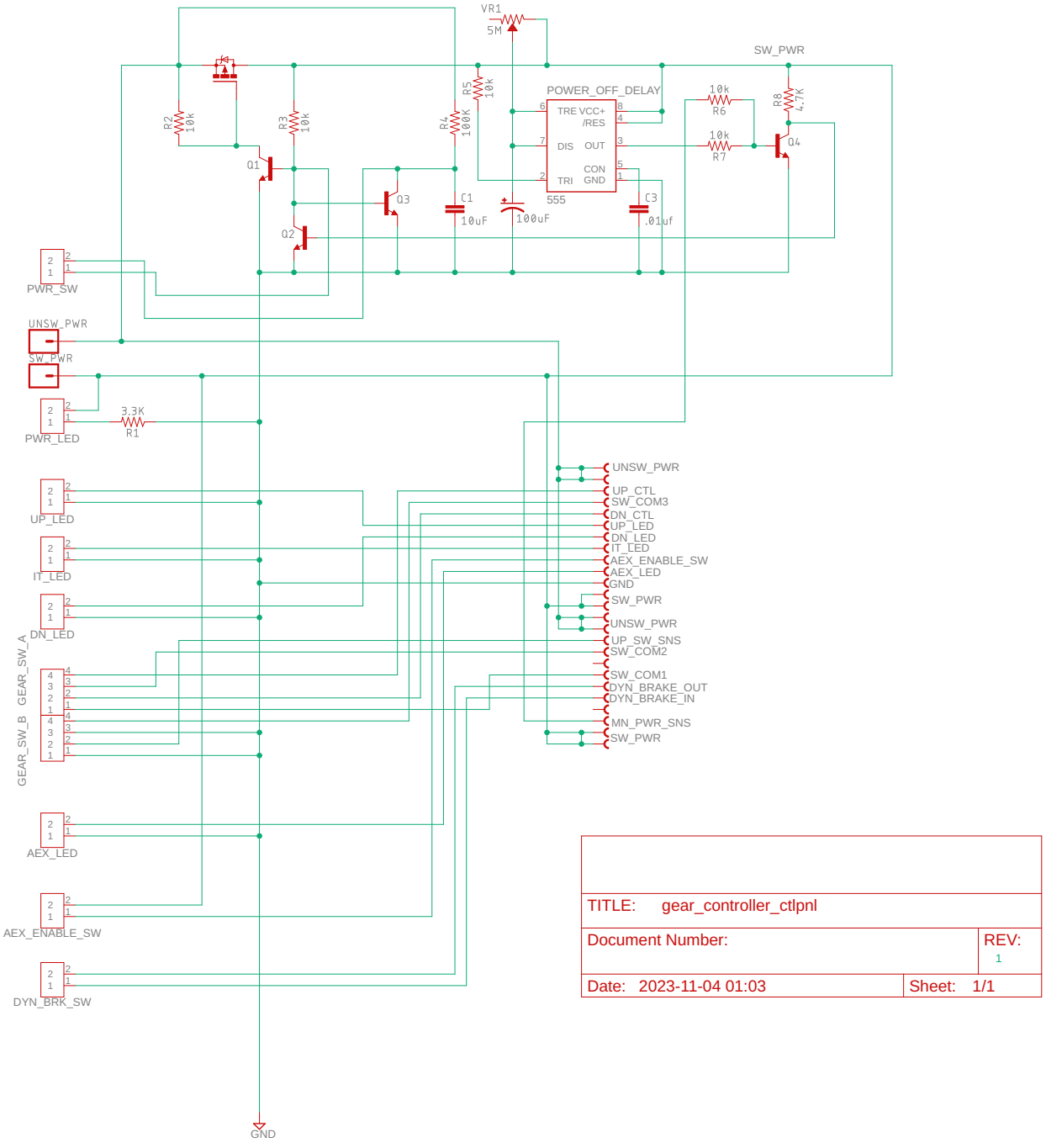

Figure 25 - Gear Control Panel Board Schematic

Appendix 2 – Wire Harnesses

Probably the most tedious part of the build is to construct the wire harnesses. I picked D-SUB connectors for the external harnesses because the PCB-mating connectors are cheap and easily obtainable, and I assumed that most Cozy/EZ builders would own or have access to a D-SUB pin crimp tool. I picked JST-XH connectors for the internal (board-to-board) harnesses because they are also fairly ubiquitous and have a locking feature. Most people will NOT have the requisite crimp tool for these but the crimp tools are relatively cheap ($35), or, if you prefer, pre-crimped wire kits are available from Amazon for about $15, if you're not fussy about needing to use Tefzel wire. Note: if you're using machined D-SUB pins (my preference), these pins are pretty expensive if you get them through Digikey or AS&S (~ .80/pin at AS&S) but can often be found on ebay for a lot less. Using D-SUB connectors with solder cups is much cheaper of course.

Regarding the use of D-SUB pins (which have a max current rating of 5A) for the high-current 15A lines, refer to this article by Bob Nuckolls (of Aerolectric Connection fame) which talks about this exact thing: http://www.aeroelectric.com/articles/High_Currents_thru_D-Subs/High_Current_D-Subs.html Essentially you just gang together however many you need to get up to your max expected current. In this case we group four DSUB pins for each high-current line, which gives a max theoretical current rating of 20A. BUT (and this is key), it is essential to have a 30cm pigtail on each pin, (ideally 15cm on each side of the pin) to normalize the spread-out current load (read the article to find out why).

Depending on your setup you'll need some or all of the following external wire harnesses:

- Main relay board to actuator & microswitches (DB15).
- Main relay board to control panel (DB25)
- Main relay board to external sensors (JST-XH)
- Arduino shield to EFIS and sensors (DB9)
- Arduino shield to LCD display and audio jack (DB9)

And the following internal wire harnesses:

- Arduino power/gnd (JST-XH)
- Arduino control panel sensors&trigger (JST-XH)

See below for pinout/construction details on all of these.

Main Relay Board to Actuator Harness

Before constructing this harness, verify the action of your nose gear limit switches as it appears there may be some variants out in the field. The standard configuration is that switches are actuated (COM = NO) when the gear is not at the limit of travel, and when the limit (either up or down) is reached the switch turns off (COM = NC). If your setup is such that the microswitch is actuated when the limit is reached, then you will need to reverse the NC and NO pins at the DB15 connector.

If (like me) you have an EZ-Noselift but removed the original harness with its molex connector then the required harness goes thusly (Figure 26):

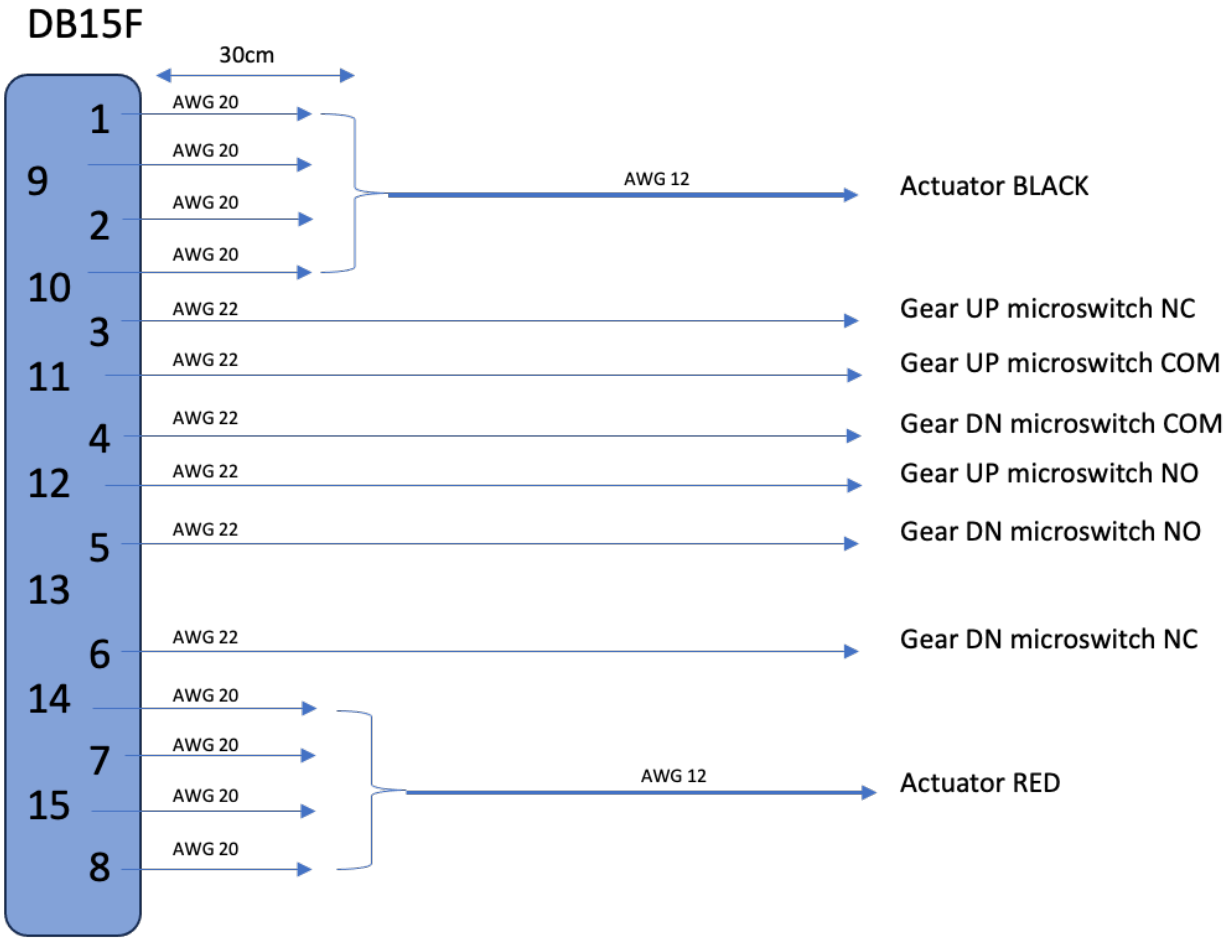

Figure 26 - Main Relay Board to Actuator Wire Harness

The #20 wire pigtails coming from 1,2,9,10 and 7,8,14,15 extend from the connector about 30cm (12”), then are soldered or (better) crimped together and spliced to the large #12 wires

coming from the actuator. The microswitch wires can be either #20 or #22 but #22 makes for a more compact harness.

If you retained the P1 connector from the original EZ-Noselift harness then that connector can be re-used by either splicing onto the existing backshell wires or crimping on new Molex pins(Figure 27):

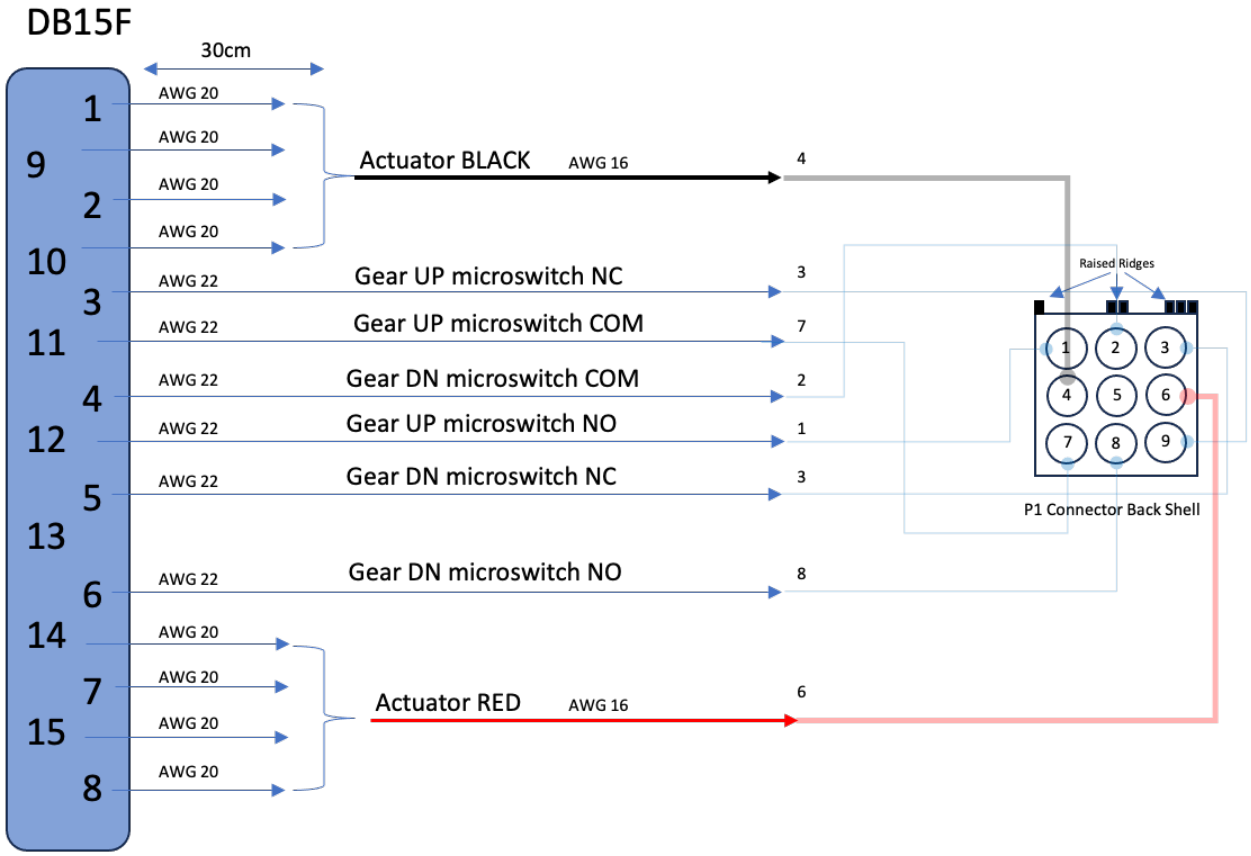

Figure 27 - Relay Board to Actuator Harness using existing EZNoseLift P1 Molex Connector

With the harness connected only to the motor and microswitches, confirm continuity from the DB15 connector:

Pin 1 -> motor BLACK lead
Pin 2 -> motor BLACK lead
Pin 3 -> Up Limit Switch NC terminal (opposite side from the microswitch pivot point)
Pin 4 -> Down Limit Switch COM terminal (underneath the microswitch pivot point)
Pin 5 -> Down Limit Switch NC terminal (opposite side from the microswitch pivot point)
Pin 6 -> Down Limit Switch NO terminal (middle)
Pin 7 -> motor RED lead
Pin 8 -> motor RED lead

Pin 9 -> motor BLACK lead
Pin 10-> motor BLACK lead
Pin 11 -> Up Limit Switch COM terminal (underneath the microswitch pivot point)
Pin 12 -> Up Limit Switch NO terminal (middle)
Pin 14-> motor RED lead
Pin 15 -> motor RED lead

The preceding test assumes that your limit switch setup is normal (ie, switches are actuated when the gear strut is not at the limits and spring open when it is fully extended/retracted). If you have a reversed setup for the UP limit switch, reverse pins 12 and 3. If you have a reversed setup for the DOWN limit switch, reverse pins 5 and 6.

Main Relay Board to Control Panel Harness

If you are using the control panel board then this is basically just a pin-to-pin mapping, with the exception of pins 18, 22, and 23 which on the Relay board side are gear position sense lines and a utility +5V output (if the +5V supply was installed). These pins are unused on the panel board side (Figure 28).

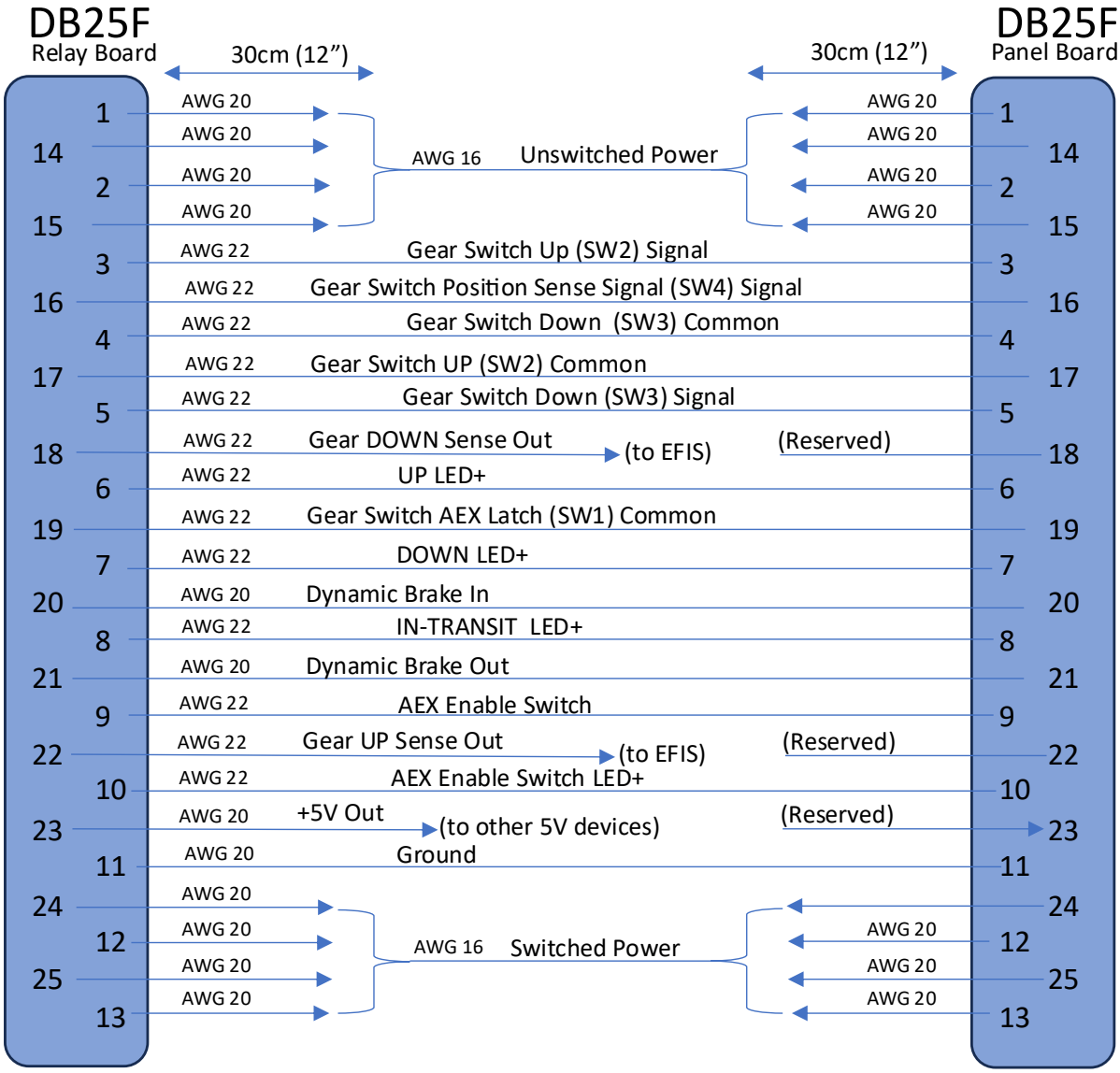

Figure 28 - Main Relay Board to Control Panel Board Harness

If not using the control panel board, the individual component wiring is a little more complicated (Figure 29):

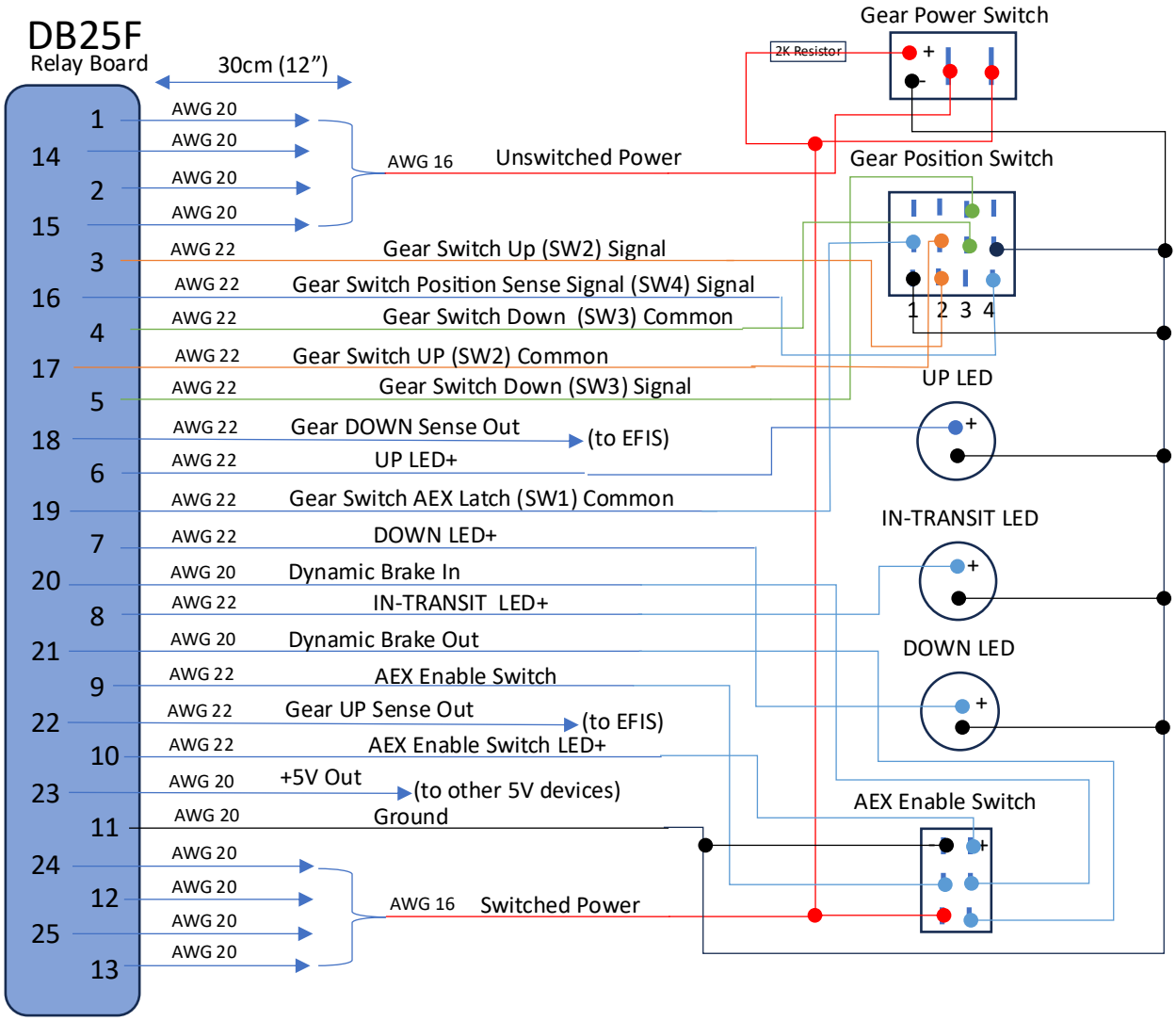

Figure 29 - Main Relay Board to Discretely Wired Control Panel Harness

Some notes:

- If the Arduino is NOT being used then the gear selector switch can be a 3PDT rather than a 4PDT; in that case, the SW4 pole wiring is not applicable.
- The Arduino drives the AEX status LED on the AEX enable switch, so if Arduino is not being used this wiring can be eliminated. Alternatively, you could wire it up to the AEX Enable line with a suitably sized in-line resistor (~1K) so it's illuminated when AEX is enabled.
- If a gear power switch is not used (gear controller is always on), there is still a requirement to bring +12V out to the control panel for the AEX enable switch to function

correctly. In this case, any one of pins 1,2,12,13,14,15,24 or 25 on the relay board connector can be used and would be connected directly to the AEX enable switch.

Main Relay Board to External Sensor Harness

This harness will primarily be used when the Arduino board is NOT being used and the AEX trigger is either being supplied from some other external source OR the pressure-switch speed sensors, throttle position switch and LIDAR altitude switch are being used as per Marc's original circuit. This harness plugs into jack J3 on the Main Relay Board (Figure 30).

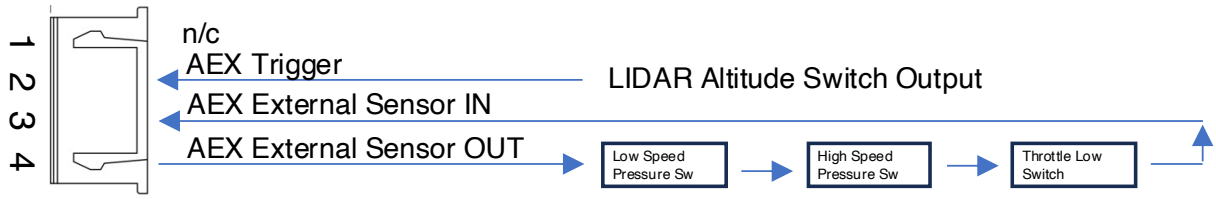

Figure 30 - Main Relay Board External Interface Harness

Board-to-Board Interconnects

The Arduino Shield interfaces with the Main Relay Board using two interconnect cables with JST-XH connectors on each end. Obviously if you aren't using an Arduino then these cables are not required.

Power Connector Cable

This 3-wire cable carries power (5V and 12V) and ground from the relay board socket J1 to the Arduino board socket J1(Figure 31):

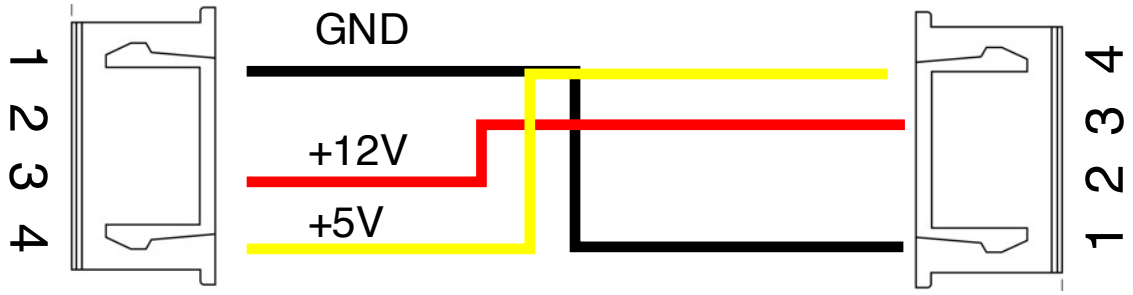

Figure 31 - Main Relay Board to Arduino Power Harness

Signal Connector Cable

This 6-wire cable carries signals and indicators between the matching J2 sockets on the relay and shield boards (Figure 32):

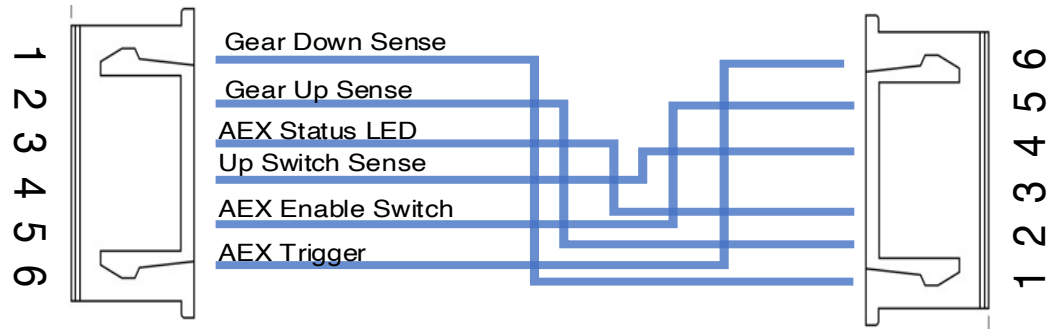

Figure 32 - Main Relay Board to Arduino Signal Harness

Arduino Shield External Interface Cables

There are two DB9 interface cables that originate from the Arduino shield. One carries the sensor signals for airspeed, throttle and altitude that enable the Arduino to make an AEX decision(Figure 33). The other carries the signals that drive the optional LCD display and audio outputs(Figure 34).

Sensor Cable

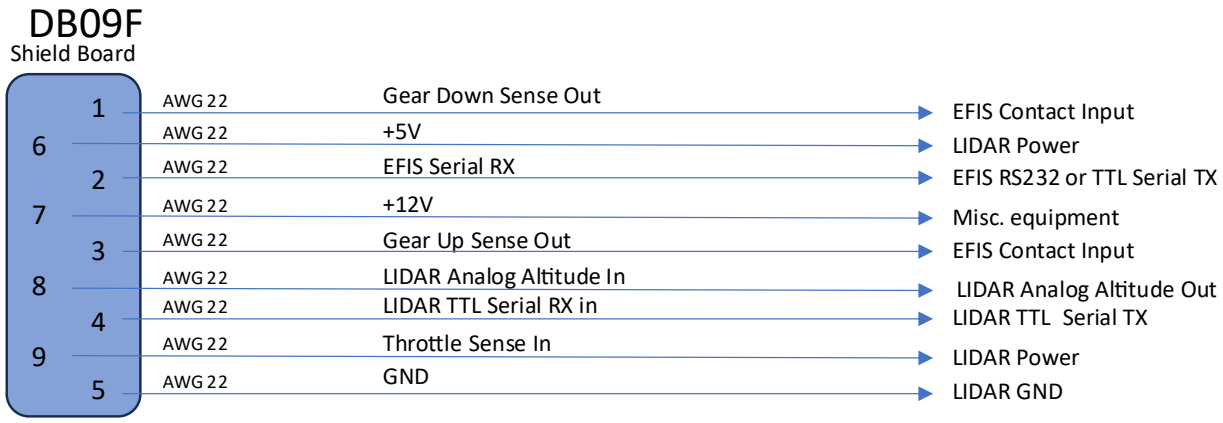

Figure 33 - Arduinio Shield Sensor Harness

AV Cable

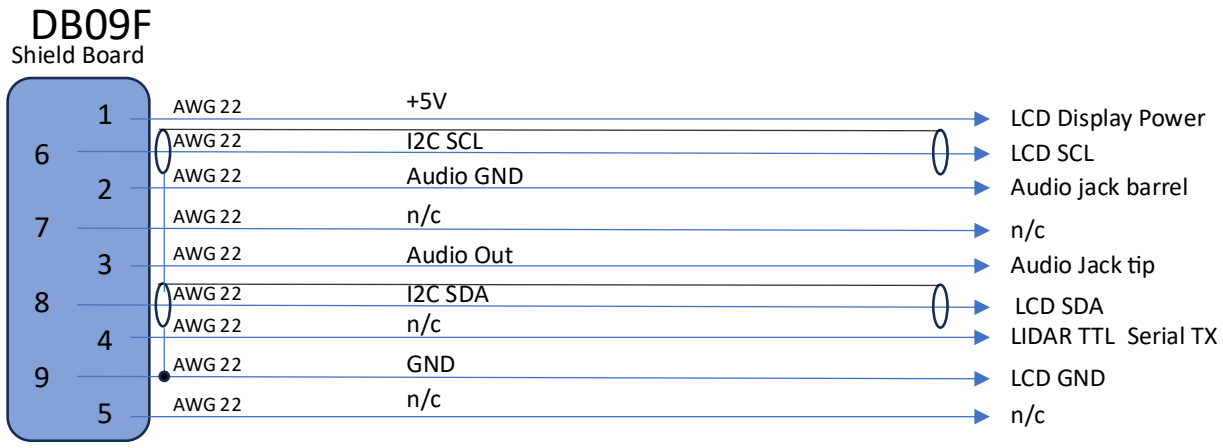

Figure 34 - Arduino Shield to Audio/Visual Outputs Harness

Additional Notes:

1. If using the LCD display, pins 6 (I2C SCL) and 8 (I2C SDA) should each be wired to a separate shielded wire, with the shields tied together and connected to pin 9 (GND), preferably within the DB9 back-shell. Leave the shields unconnected at the LCD display end.

Appendix 3 – Remote Display

The optional display is a small (16 char x 2 lines) LCD screen which continuously shows the airspeed and AGL altitude being received by the Arduino unit on the first line. Informational, warning or error messages are displayed on the second line(Figure 35). Eg:

Figure 35 - Arduino External Display

Assembly is simply a matter of soldering the power, ground, SCL and SDA from the Arduino shield AV Connector harness to the appropriate points on the display unit and mounting the display unit into the 3D-printed case.

Note:

This unit is driven via the I^2C bus from the Arduino. This bus is not meant for long cable runs or noisy environments, and in a Cozy IV cockpit we have both. Therefore, it is essential that shielded wire be used to connect the SDA and SCL lines. Note that each line requires its OWN shielded wire – don't use a 2-wire cable for this.

If you find that the display is blanking or freezing, this is likely due to the I^2C bus glitching from noise. Installing the I^2C booster chip on the shield may fix it, but I have had good luck with just using the wire shielding as discussed above.

Appendix 4 – Arduino Considerations

There are many flavours of Arduino to choose from (Nano, Uno, Mega, etc), but only a few will work with the custom shield. The Arduino hardware design is open-sourced so there are many variants from different vendors. For our purposes the the main consideration is that it should have at least as much dynamic RAM as an UNO (2K) and operate main restriction is the ability to handle a +5V supply voltage and +5V signal inputs – many types of Arduino are restricted to +3.3V supply and signal voltages.

The custom shield was designed to work with an Arduino UNO R3, but there are other styles of Arduino that will work as well. Here is a non-exhaustive list of Arduinos that support +5V and are pin-compatible with the custom shield

- Maker UNO

  The board I have used in my builds. I like it because it it's really cheap (~$10), uses all surface-mount components so can be mounted flat and has LEDs on the all the GPIO pins for easy diagnosis. It dispenses with an on-board voltage regulator, so an external +5V supply is required to power it. (https://www.cytron.io/p-maker-uno-simplifying-arduino-for-education)

- Arduino Mega

  A good choice if you want to do other things with the microcontroller than just act as a gear controller. More GPIO pins, more serial ports, more memory. More $$$. (https://store.arduino.cc/products/arduino-mega-2560-rev3)

- Arduino Uno Rev4

  Latest revision, pin compatible with Uno Rev3 and offers more memory and a faster processor (https://store-usa.arduino.cc/pages/uno-r4?gad_source=1&gclid=Cj0KCQiAn-2tBhDVARIsAGmStVlpxJXa2HUSwBKcJcfaq2u71DuJz5TjAEqegS2NQEJsV0ZpYVMsEakaAvulEALw_wcB&selectedStore=us)

- Ruggeduino-SE

  This is a special hardened version of the UNO, which TBH, I would probably have used in my build if I knew it existed at the time. It's definitely more suited to the cockpit environment of an airplane, but it's not particularly cheap (~$80). (https://www.rugged-circuits.com/microcontroller-boards/ruggeduino-se-special-edition)

Appendix 5- LIDAR Considerations

Using the AGL altitude as an additional criterion to the AEX trigger determination prevents nuisance deployments of the nose gear when the speed envelope and throttle position requirements are met (such as slow flight, stall demonstrations, etc). Thanks probably to all the interest around autonomous vehicles these days, reasonably-priced LIDARs with 100m+ ranges are now readily available. (Note: some EFIS systems DISPLAY an AGL altitude but AFAIK none of them output it as part of their serial datastreams). Most of these LIDARs boast range resolution under 10cm, so in addition to using the LIDAR for the AEX trigger determination, we can take advantage of this to add landing assist functionality using audio altitude callouts (Arduino-based installations only).

Choosing a LIDAR

The Arduino software (currently) supports the following LIDARS:

- Lightware SF11C* (100m)
- Lightware SF30C (100m)
- Bennewake TF03* (180m)
- Any LIDAR that can output an analog altitude signal < 5V* (SF11, SF30, others)

The LIDARs marked with an asterisk (*) have actually been bench tested, for the others, the support code is based on published specifications and should be regarded as untested software. As far as I am aware, the only LIDAR which has been flight tested with this setup is the Bennewake TF03, since this is the one I used in my build.

In theory, any LIDAR that can be configured to send a continuous datastream (eg, does not need to be polled) or outputs an analog signal < 5v should be readily adaptable to the existing code base.

IMO you should choose a LIDAR that has a minimum range of 100m (~300'). There are quite a few LIDARs that have ranges of 50m (~150') but given a standard descent profile of ~500'/min then at 150' you are only 20 seconds away from touchdown – about the exact amount of time required to extend the gear. Better to have a little margin – in fact, the more margin you have, the better off you are.

Configuring the LIDAR (serial or analog)

The supported LIDARs all output a TTL-level (NB: NOT RS232!) serial datastream. The LIDAR interface code expects the LIDAR to be connected to the Arduino's serial port (GPIO pins 0/1).

The LIDAR must be configured with the appropriate baud rate: 19200 for the SF11/SF30 and 115200 for the TF03. (115200 is about the max that the UART on the Arduino can handle). The LIDAR must be configured to output a continuous stream of altitude data packets at least once per second. It's beyond the scope of this document to detail how to configure all the different supported LIDARs – consult the manufacturer's installation guide for procedures on how to do this.

Some of the supported LIDARs have the option to be configured to output an analog altitude signal where at the maximum supported altitude they will output some constant voltage which then decreases linearly in relation to the AGL altitude. The 10-bit DAC in the Arduino gives about 1mv resolution, so a 0-3.3v range with a 100m LIDAR gives about 3cm (~1") resolution, plenty accurate enough for our purposes. Again, consult the manufacturer's installation guide for procedures on how to configure the LIDAR to do this. Note that the analog LIDAR support is based on the Lightware SF11 and so the conversion factor from voltage to altitude is specific to that unit – other analog LIDARs would likely need a different conversion factor. Determination of that factor is left as an exercise for the reader.

Appendix 6 – EFIS considerations

Arduino-based installations will need to use an EFIS serial datastream to get airspeed data in lieu of the airspeed pressure switches called out in Marc's OG circuit.

The following EFISs are supported:

- MGL ASX-1 (not an EFIS, but provides airspeed data in serial format) *
- Dynon Skyview*
- Garmin G3x
- GRT Horizon

The EFISs marked with an asterisk (*) have actually been bench tested, for the others, the support code is based on published specifications and should be regarded as untested software.

The supported EFISs all have a means of configuring one of their available serial ports to output a continuous stream of data, of which the airspeed is just one part. In theory, any EFIS capable of outputting a datastream containing the airspeed should be easily adaptable.

Connecting the EFIS to the Arduino Shield

(All EFISs)

Determine which serial port you are going to use, and break out the TX line (the RX line is not used). This should go to Pin 2 on the "Sensor" DB9 connector of the Arduino shield.

Ensure that jumpers are installed on jumper headers JP2 and JP3 of the Arduino shield, and that JP1 does NOT have a jumper on it. These jumpers route the incoming RS232 signal from the EFIS to the MAX3232 converter chip on the Arduino shield.

Note that if a jumper is installed on JP1 and the EFIS is connected and powered up, the high-voltage RS232 signal will likely fry the GPIO input on the Arduino.

Configuring the EFIS

It's beyond the scope of this document to detail the steps required to configure the EFIS – consult the manufacturers documentation for this. In general, you will need to determine the appropriate format for the serial data and set the baud rate to match what is set in the Arduino code. The data format is an EFIS-specific thing. For the serial communication speed, you want to keep this fairly low due to limitations on the software serial support of the Arduino. I have

had no issues running at 19200 but I wouldn't go higher than that.

- Skyview

  Serial data speed: 9600
  Serial data type: DYNON ADAHRS

- Garmin G3X

  Serial data speed: 19200
  Serial data type: Attitude Air Data

  Note that the G3X has the capability to output a "GPS" message which contains the AGL altitude. In theory this could be used in place of the LIDAR, however it is only accurate to 100'. This message also only has groundspeed, not airspeed. Implementation of this message to replace the LIDAR is left as an exercise for the reader.

- GRT Horizon

  Serial data speed: 19200
  Serial data type: Shadin type 'Z'

Appendix 7 – Indicators and Messages

Indicators

The AEX status is communicated to the pilot by the status light on the AEX enable switch:

- Off - AEX is not enabled
- Blinking slowly – AEX initializing, waiting for data, or operating in sensor degraded mode (not all sensors reporting but at minimum for AEX determination)
- Blinking quickly – AEX error (insufficient sensor input for AEX determination, unknown state)
- Steady On – AEX is enabled and armed

Messages

If the LCD display is installed, then AEX status is communicated to pilot via the message line (line 2 of the display).

- “AEX Enabled”

  AEX has been enabled via the AEX enable switch. Note this does not mean that AEX is armed, only that it has the potential to be armed.

- “AEX Disabled”

  AEX switch was turned off after initially being turned on

- “AEX Failed”

  An unrecoverable error occurred. The status light will be blinking rapidly. Resetting the controller by power-cycling may resolve this.

- “AEX Triggered”

  AEX has met the trigger conditions and will start extending the gear

- “AEX Armed”

  All sensors are reporting and AEX will deploy the gear when the trigger conditions are met.

- “AEX Armed – WARN”

  Not all sensors are reporting – AEX operating in degraded mode. AEX will trigger when the minimum trigger conditions are met.

- “Extending Gear”

  Gear is extending, either because it was triggered by AEX or manually via the selector switch

- “Gear Ext. Canceled”

  An AEX gear extension was cancelled by recycling the selector switch

- “spd data timeout”

  Timeout error detected on the serial airspeed data input. If Altitude data is still reporting, AEX will remain armed in degraded mode.

- “alt data timeout”

  Timeout error detected on the serial altitude data input. If airspeed data is still reporting, AEX will remain armed in degraded mode.

- “wt: spd/alt data”

  AEX is initializing – waiting for airspeed and/or altitude data streams to begin. Typically seen on systems with gear controller power switches when the gear controller is on but the master power is not.

- “WRN:Armed no spd”

  AEX is armed but no speed data is available

- “WRN:Armed no alt”

  AEX is armed but no altitude data is available

- “Er: no spd, no alt”

  AEX enabled but no airspeed and no altitude data is available. AEX status switch will be flashing quickly

- “Er: audioini fail”

  Unable to talk to the audio daughter card; audio alerts disabled

Appendix 8 – Bill of Materials

This list is compiled from the master lists on the Digikey site (hyperlinked at various points above). In the event of discrepancies from what is here vs the Digikey lists, the Digikey lists should be considered gospel.

Description	Digi-Key Part Number	Quantity	Unit Price	Extended Price
Main Relay Board				
RELAY GEN PURPOSE DPDT 30A 12V	PB351-ND	2	$25.51	$51.02
RELAY GEN PURPOSE 3PDT 10A 12V	Z11232-ND	1	$17.40	$17.40
CONN D-SUB PLUG 15POS R/A SLDR	609-1492-ND	1	$2.68	$2.68
CONN D-SUB PLUG 25POS R/A SLDR	609-1504-ND	1	$4.45	$4.45
DIODE GEN PURP 1KV 1A DO41	1N4007DICT-ND	10	$0.14	$1.41
CONN HEADER VERT 2POS 2.54MM	929450-01-02-ND	2	$0.71	$1.42
MICRO-MINIATURE SMT JUMPER	36-5102CT-ND	3	$0.34	$1.02
TERM BLK 2POS SIDE ENTRY 5MM PCB	ED1631-ND	1	$1.61	$1.61
5V Regulated Supply Option				
DC DC CONVERTER 5V 3W	945-1648-5-ND	1	$3.25	$3.25
CAP CER 10UF 25V X5R RADIAL	445-181284-1-ND	1	$0.52	$0.52
FIXED IND 12UH 2.4A 130 MOHM TH	495-6783-1-ND	1	$0.59	$0.59
EFIS-friendly Gear Position Signals Option				
TRANS NPN 30V 0.8A TO18	1514-2N2222PBFREE-ND	2	$2.36	$4.72
RESISTOR KIT - 1/4W (500 TOTAL)	1568-COM-10969-ND	1	$8.95	$8.95
Positive AEX Trigger Option				
TRANS NPN 60V 10A TO220	D44H8GOS-ND	1	$1.05	$1.05
Self-resetting Circuit Protection Option				
PTC RESET FUSE 60V 750MA 2920	F5633CT-ND	1	$0.74	$0.74
PTC RESET FUSE 16V 6.0A 2920	18-2920L600/16MRCT-ND	1	$0.86	$0.86
Arduino Basic				
MAKER UNO	3614-MAKER-UNO-ND	1	$7.18	$7.18
CONN D-SUB PLUG 9POS R/A SLDR	609-1480-ND	2	$2.54	$5.08
ARDUINO STACKABLE HEADER KIT - R	1568-1413-ND	1	$1.75	$1.75
CAP CER 0.1UF 50V X7R RADIAL	399-9870-1-ND	2	$0.22	$0.44
CAP CER 0.33UF 5% 50V X7R RADIAL	399-13993-ND	2	$0.55	$1.10
CAP ALUM 100UF 20% 25V RADIAL	732-8825-1-ND	1	$0.13	$0.13
CONN HEADER VERT 2POS 2.54MM	929450-01-02-ND	3	$0.71	$2.13
IC TRANSCEIVER FULL 2/2 16TSSOP	296-13096-1-ND	1	$1.00	$1.00
Audio Alerts Option				
DFROBOT DFPLAYER PRO - A MP3 PLA	1738-DFR0768-ND	1	$8.90	$8.90
CONN JACK STEREO 3.5MM	889-1822-ND	1	$1.58	$1.58
Remote Display Option				
GRAVITY: I2C 16X2 ARDUINO LCD WI	1738-1418-ND	1	$11.90	$11.90
IC ACCELERATOR I2C 1CH SC70-6	LTC4311ISC6#TRPBFCT-ND	1	$6.02	$6.02
CAP CER RAD 10NF 25V C0G 10%	C321C103K3G5TA-ND	1	$0.52	$0.52
Control Panel				
CONN D-SUB PLUG 25POS R/A SLDR	609-1504-ND	1	$4.45	$4.45
SWITCH ROCKER DPST 10A 125V	CKN2068-ND	1	$8.65	$8.65
SWITCH TOGGLE 4PDT 5A 120V	EG2429-ND	1	$10.82	$10.82
LED RED 1/4" HOLE 12V PANEL MNT	5102H1-12V-ND	1	$3.30	$3.30
LED GREEN 1/4" HOLE 12V PAN MNT	5102H5-12V-ND	1	$3.30	$3.30
LED YELLOW 1/4" HOLE PANEL MOUNT	L20697-ND	1	$7.20	$7.20
SWITCH ROCKER SPST 16A 125V	SW649-ND	1	$3.61	$3.61
CONN HEADER VERT 2POS 2.5MM	455-B2B-XH-A-ND	8	$0.15	$1.20
CONN HEADER VERT 4POS 2.5MM	455-B4B-XH-A-ND	2	$0.23	$0.46
CONN QC RCPT 14-16AWG 0.187	A0910CT-ND	2	$0.47	$0.94
CONN QC RCPT 18-22AWG 0.110	A27949CT-ND	6	$0.45	$2.70
RES 3.3K OHM 5% 1/4W 1206	RMCF1206JT3K30CT-ND	1	$0.10	$0.10

Figure 36 - Complete Bill of Materials (BOM)