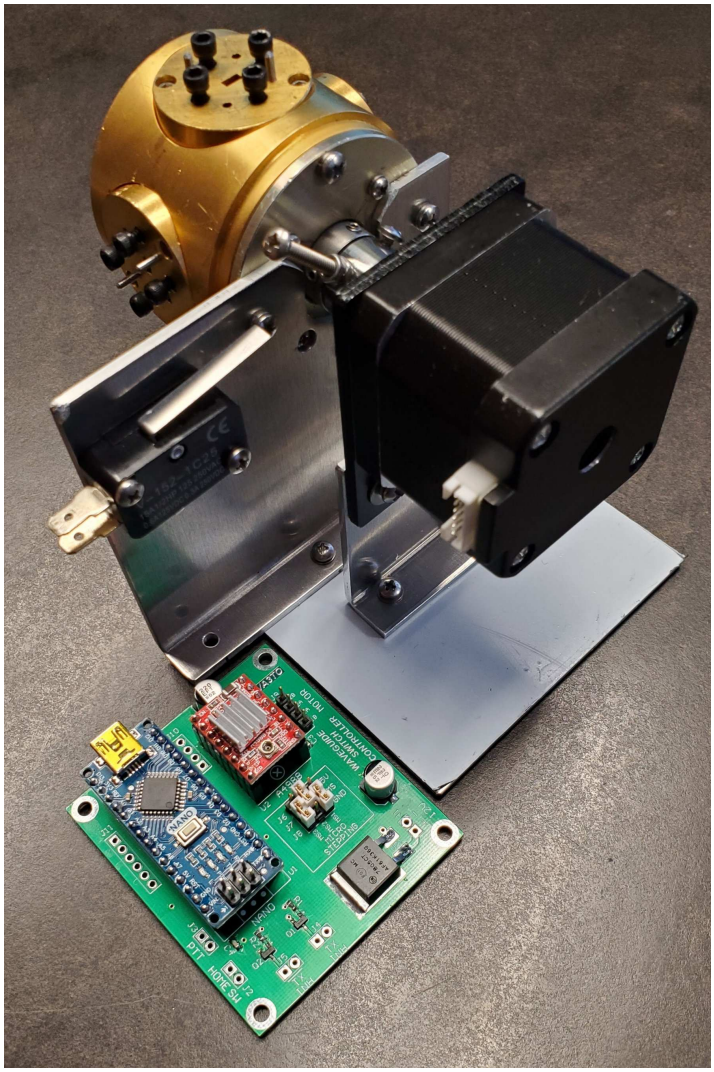
Motorizing a Manual Waveguide Switch using a Stepper Motor

Hugh Duff VA3TO Mar. 2023
radeohedca@hotmail.com

Affordable surplus motorized waveguide switches for the millimeter-wave bands are virtually unobtanium so I decided to try my hand at motorizing a manual WR-15 switch that I was lucky enough to acquire using a stepper motor. I used a standard 42 x 38mm NEMA 17 stepper motor with an Arduino NANO microcontroller and an A4988 stepper motor driver on a custom carrier board.

A video showing the assembly in operation can be viewed at this link:
https://www.youtube.com/watch?v=AOKYyWwhkv8



MECHANICAL DETAILS

Brackets made from 0.100” aluminum plate were fabricated to mount the WG switch and stepper motor based on the physical requirements of a Kuhne 76 GHz transverter with a short straight piece of waveguide connecting the TX port to the switch. The WG Switch bracket was made wide enough to mount the home position microswitch. Some of the bracket holes were slotted to allow the motor to be aligned to the WG switch with minimal rotational friction.

The manual WG switch normally uses a detent mechanism to position the rotor precisely but it requires a lot of torque to overcome. I was able to disengage the detent mechanism to let the precision of the stepper motor position the rotor, and it's a lot easier for the motor to turn.

The large knob used to manually rotate the waveguide switch position was removed and a 3/8" to 5mm shaft coupler was made from a piece of stainless steel rod to couple the waveguide switch to the stepper motor. Since the WG switch is fully manual with no motor or optical sensors, I use a microswitch that is triggered by a long 4-40 screw on the shaft coupler (idea borrowed from W5LUA) to establish a reference “Home position”. Any N.O. microswitch with a long activator arm can be used.

Since waveguide switches come in various shapes and sizes with different physical arrangements of shaft sizes & port orientation, and individual layout requirements will vary, mechanical mounting details and shaft coupler design are left to the builder.

STEPPER MOTOR & CONTROLLER BOARD

A custom carrier board was designed to mount the NANO microcontroller and A4988 stepper controller board, along with a 5V regulator. SIP female headers should be used for the NANO to facilitate programming as it needs to be programmed while it is off the carrier board. I also used them for the A4988. (A limited number of pc boards are available for sale. Contact the author for availability.)

The Arduino NANO (or compatible) board easily handles the task at hand. It contains an 8 bit ATmega328P microcontroller, a USB to serial converter I.C. and a 5V regulator, and comes pre-loaded with the Arduino bootloader firmware.
Beware of bad compatible boards when ordering a NANO and make sure it’s a 16 MHz 5V version with a 328P microcontroller. Some of the knock-offs use a 12 MHz oscillator and will not run properly with the code as written. Also make sure you don’t order a board that uses the less capable Tiny88 processor or some other NANO variant.

The A4988 stepper motor driver board simplifies the amount of programming and takes some processing load off the microcontroller by cycling the correct phasing sequence to rotate the motor. The interface is reduced to a Direction pin (Hi or Lo) and a Step pin which gets pulsed to turn the motor. It also has built-in FET drivers and manages the current to the stepper motor windings.

Both the NANO and A4988 boards are inexpensive and can be ordered from many online sources.

The Nema 17 stepper motor has a native step size of 1.8° but the A4988 stepper controller allows motor to microstep in 1/2, 1/4, 1/8 or 1/16 steps. Microstepping can be configured by strapping the MS1, MS2 and MS3 jumpers accordingly. The tradeoff of microstepping is speed so I chose to use 1/4 step microstepping so not to sacrifice too much speed. That provides 0.45° precision to position the rotor in the RX and TX positions. For this project we strap MS1 = GND, MS2 = +5V and MS3 = GND for 1/4 step microstepping.

There are a few variations of NEMA17 stepper motors offering different torque ratings. I tested 42 x 20mm, 42 x 38mm and 42 x 48mm models and found that the 42 x 38mm had a suitable amount of torque for this project. Most of these motors are coming out Asia so the specifications can be ambiguous. They also come specified to work at different voltages however it seems to be less important as they are current driven off the A4988 controller. The 42 x 38mm motor I chose is a model number 17HS4401, branded “SIMAX3D” that I ordered from Amazon.com.

The A4988 stepper motor controller has a current adjust pot that needs to be adjusted according to the specified current requirement of the chosen motor. There are a couple of different methods to accomplish this. A quick web search will turn up several sites and video links that describe the process. I followed one of the procedures but found that the motor was getting quite warm at idle, so I experimented a little to reduce the current while ensuring it still operates properly. The motor now operates cool to the touch with sufficient torque.

Active HI and LO TX Inhibit outputs are provided to holdoff the transmitter until the switch reaches the TX position, however this is a simulated output derived in software, not a true indication of the physical position of the WG rotor. The builder could add a second microswitch that can be activated when the switch is in the TX position if positive indication is desired.

SOFTWARE

The C code was written and programmed into the NANO microcontroller board using the Arduino IDE environment. It runs a Home/calibration procedure at power-up, rotating back until it triggers the microswitch then slowly forward until the switch is released (“Home” position), then offset to the RX position of the WG switch, which can be fine tuned in firmware for perfect alignment. The motor is slowed down for the homing procedure so that the trigger point of the microswitch is consistent with no recoil. In the worst case situation, if the unit was last powered down with the rotor is sitting in the TX position then it will only need to rotate a little over 90° to trigger the microswitch, so the complete cal procedure takes place within a second or two after power-up. Grounding the PTT pin moves the rotor of the WG switch 90° to the TX position.

Every build will have mechanical variations in the placement of the microswitch and possibly different rotational angle between RX and TX of the WG switch, so some values in the code will need to be altered to suit. The code is well documented with details of which values will need to be changed.

Programming the NANO is a simple task. Remove the NANO from the carrier board and connect it to a computer USB port using an appropriate cable. The board is sufficiently powered by the USB bus for programming so it does not need to be connected to a power supply.

Download and install the latest version of the Arduino IDE onto your computer.
Run the Arduino IDE and configure it as follows:

Tools - Port – COMx (use Device Manager to determine which port is being used).
Tools – Board – Arduino AVR Boards - Arduino Nano
Tools – Processor – ATmega328P. *

*Note that some Nano compatible boards come with an old version of the bootloader so you may need to select “ATmega328P (Old Bootloader)” if you encounter any errors when attempting to program the board.

A sketch containing the C code for this project can be downloaded from this link:
va3to.com/Articles & Documents.htm

Open the sketch in the Arduino IDE and upload it to the board using the round icon with the right arrow. The IDE should report “Done uploading” when programming is completed. Remove the programming cable and re-install the NANO into the carrier board.

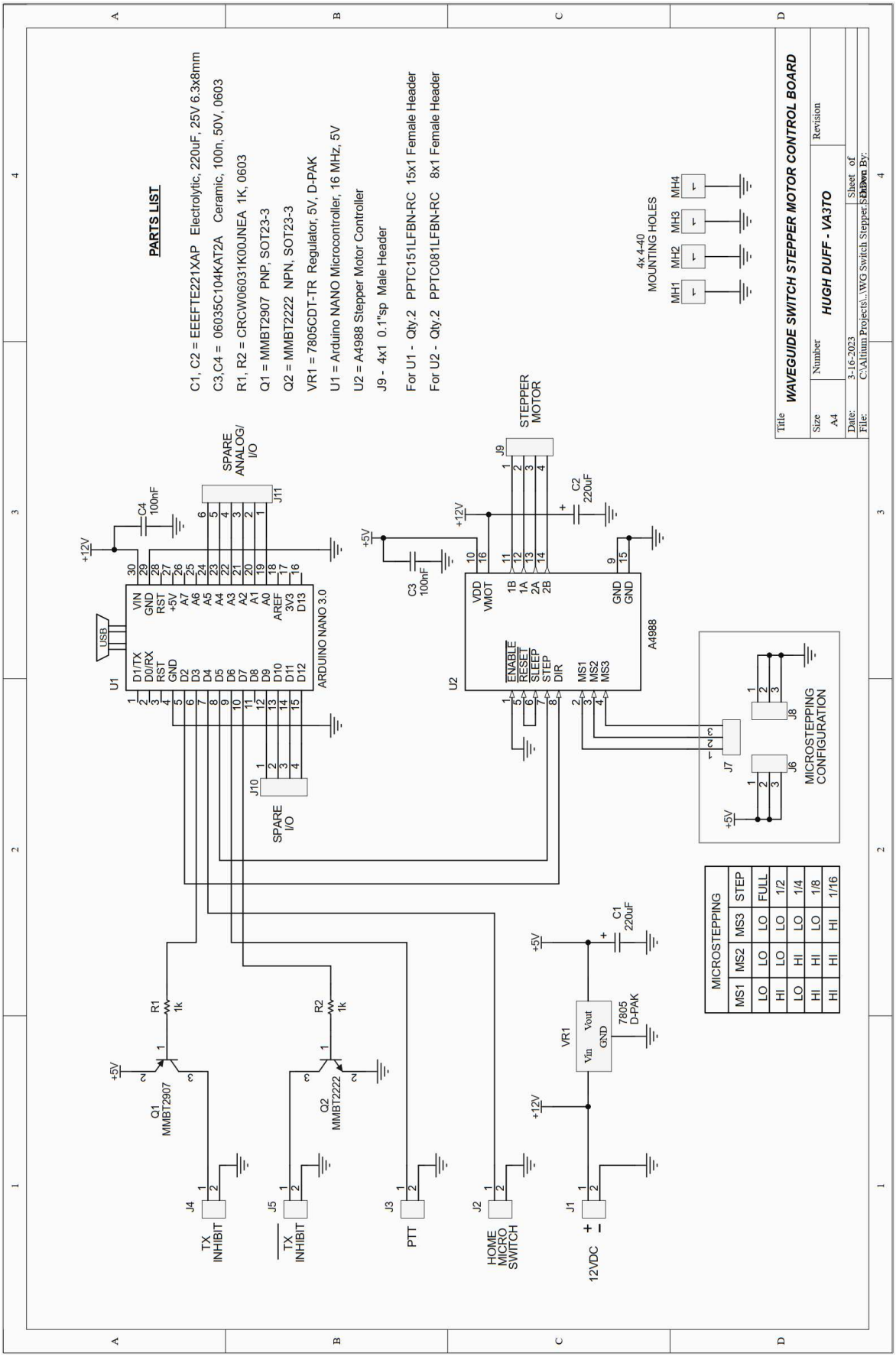
CALIBRATION

To calibrate the RX position, I temporarily altered the "RXoffset" constant in the software to move the switch roughly 180° from the reference point after power up so that I could sight the opening across the two horizontal ports (this WG switch also has "thru" positions). With a little trial and error by tweaking the offset variable, I was able to get the aperture perfectly aligned. Then I subtracted the number of steps that it takes to put the WG switch back at the RX position (- 135°). This new value is the number of steps (1/4 microsteps) it takes to move the rotor from the "home" reference point to the RX position.

FINAL WORDS

The WR-15 waveguide switch assembly constructed here will be installed into my new 76 GHz transverter but any manual rotary waveguide switch for other bands can be motorized in the same manner. I found the mechanical aspect of this project to be the most challenging part. Of course there are different ways to approach the physical requirements of mounting and coupling the stepper motor to the waveguide switch depending on the configuration and layout requirements. Hopefully the motorizing part as described here will make it more inviting for others to consider taking on such a project.

SCHEMATIC


PARTS LIST
C1, C2 = EEEFTE221XAP Electrolytic, 220uF, 25V 6.3x8mm
C3,C4 = 06035C104KAT2A Ceramic, 100n, 50V, 0603
R1, R2 = CRCW06031K00JNEA 1K, 0603
Q1 = MMBT2907 PNP, SOT23-3
Q2 = MMBT2222 NPN, SOT23-3
VR1 = 7805CDT-TR Regulator, 5V, D-PAK
U1 = Arduino NANO Microcontroller, 16 MHz, 5V
U2 = A4988 Stepper Motor Controller
J9 - 4x1 0.1"sp Male Header
For U1 - Qty.2 PPTC151LFBN-RC 15x1 Female Header
For U2 - Qty.2 PPTC081LFBN-RC 8x1 Female Header
4x 4-40 MOUNTING HOLES
MH1 MH2 MH3 MH4
Title WAVEGUIDE SWITCH STEPPER MOTOR CONTROL BOARD
Size A4
Number HUGH DUFF - VA3TO
Revision
Date: 3-16-2023
Sheet of
File: C:\Altium Projects\..\WG Switch Stepper.SchDoc
Drawn By:
U1 ARDUINO NANO 3.0
U2 A4988
STEPPER MOTOR
SPARE ANALOG/ I/O
SPARE I/O
MICROSTEPPING CONFIGURATION
MICROSTEPPING
MS1 MS2 MS3 STEP
LO LO LO FULL
HI LO LO 1/2
LO HI LO 1/4
HI HI LO 1/8
HI HI HI 1/16
TX INHIBIT
TX INHIBIT
PTT
HOME MICRO SWITCH
12VDC
Q1 MMBT2907
Q2 MMBT2222
7805 D-PAK