Arbitrary Waveform Generator

Attempting to find a black box solution that will allow the CRIO to control the input signal frequency and amplitude through some digital logic.

Definitions:

1. DDS, Direct Digital Synthesis.
2. RTOS, A real-time operating system (RTOS) is an operating system (OS) intended to serve real-time applications that process data as it comes in, typically without buffer delays.

Requirements:

1. An output achieving at least a -2V to 2V peak to peak output signal.
2. 0-20kHz frequency range.
3. Programmable via TTL or other logic.

Videos and Posts:

General STM Programming

- STM software development, Debugging, GPIO, Timers, ADC, Standby, UART, I2C, SPI. Videoe Playlist

STM32 PCB and Firmware Design

- STM Application note on the STM32F4 MCU hardware development. Application Note
- STM Kicad PCB design video. Video
- Schematics and build files for a simple development baord. Video Playlist
- Reference PCB design for an STM32F4 as well as firmware design. GitHub
- Youtube channel for STM32 PCB design and other useful PCB design knowledge. Youtube channel
- PCB manufacturer. Website

DAC implementation

- How to use the DAC STM32CubeIDE HAL. Video
- Using the DAC to make a sine wave. Video

Electronic circuits

- A few basic level shifting circuits. Post

FreeRTOS

- ST tutorial series on the STM32F4 FreeRTOS implementation. Video Playlist

- Digi-Key FreeRTOS tutorials. Video Playlist
- Tutorials and Blog on FreeRTOS. Video Playlist, Blog
- CMSIS-RTOS2 official page. Page

Communications

- Interfacing with and SD card using SPI. Blog
- Possible Modbus implementation. GitHub
- STM32Cube USB device library user manual (UM1734). Manual
- STM32Cube USB host library user manual (UM1720). Manual

Bootloader

- STM32 microcontroller system memory boot mode. Manual
- Talking to the on-board Bootloader. Video
- Bootloader. Video
- Upgrading STM32F4DISCOVERY board firmware using a USB key (AN3990). Manual

Programming via SWD using ST-Link v2

- ST-LINK/V2 in-circuit debugger/programmer for STM8 and STM32 (UM1075). Manual
- STM32, SWD, ST-Link Debugging on Custom Hardware Tutorial. Video
- How to Resolve Can not connect to target for the STM32 and ST-Link. Video

Machine Learning AI

- Digi-Key X-CUBE AI. Video

Embedded Linux

Software Tools

- Linux Based STM flash tool. Application
- STM32Programmer. Application

Waveform Generation Calculations:

$$PSC = \frac{\frac{Fclock}{Ns}}{Fsine*(Period+1)} - 1$$

$$Fsine = \frac{\frac{Fclock}{Ns}}{(Period+1)*(PSC+1)} - 1$$

$$Fsine = \frac{\frac{90Mhz}{100}}{(5+1)*(10+1)} = 13.636kHz$$

Notes:

- When using op-amps it is important to consider the frequency limitations caused by the limited slew rate and bandwidth.
- Managing to get a working sine wave in the range of 0Hz and +- 50kHz.

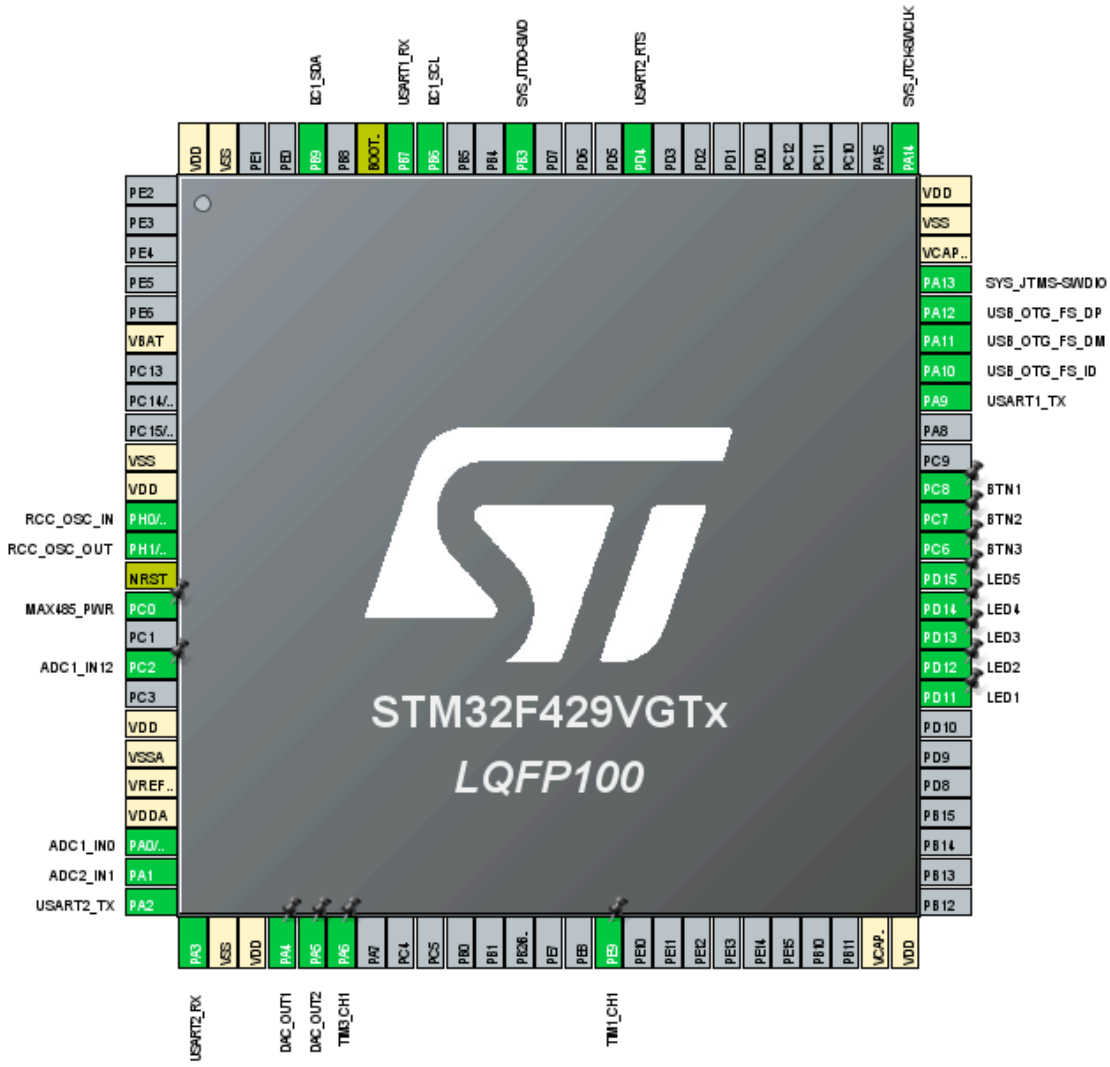
PCB considerations:

- USB.
- USB DFU.
- Switch between power circuitry and the rest of the board
- Level shifting circuitry
- DIP switch for boot mode settings
- DIP switch between power section and rest of circuit
- Fuses
- Reset buttons
- LEDs
- ESD protection
- Reverse polarity protection
- SW with trace for debugging
- Temp sensor
- MAX485for RS485
- Power supply for +-5V

PCB Testing Process:

1. Slowly ramp up voltage

Current MCU pin out:



SPRINT 1

01/03/2021

Goals:

1. Use the ST-Link v2 programmer to programme an MCU
2. Develop the UART message protocol for the communications between the CRIO and the Arbitrary Waveform Generator.
3. Develop the UART bootloader software

ST-Link v2 programmer

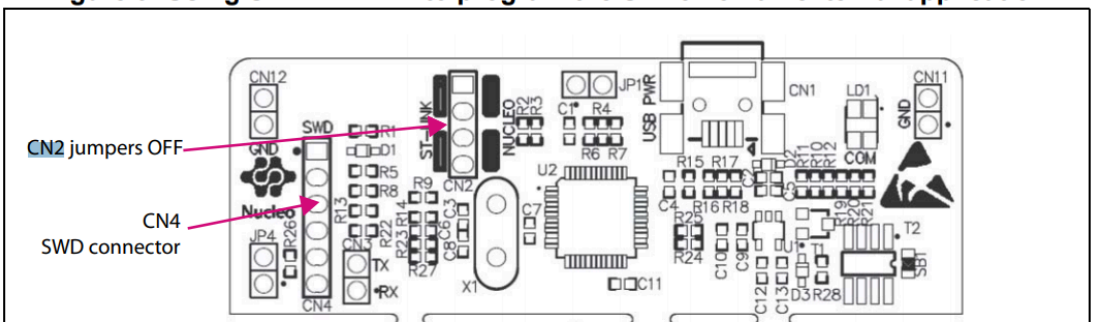
ST Link v2 programmer

STM32 Nucleo 144 SWD Pin out

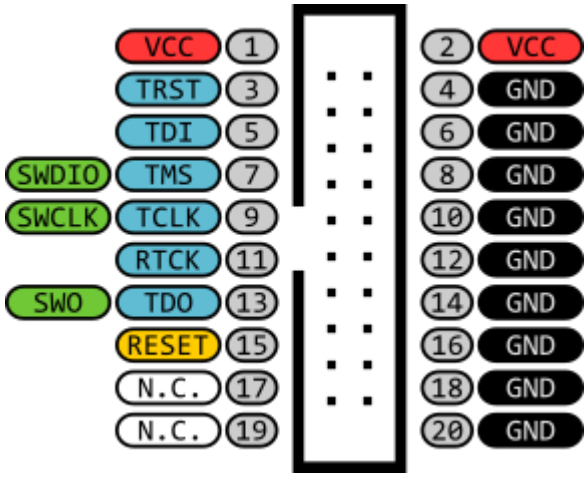
Using this pin out prooved difficult and I was not able to programme the development board this way.

Pin	CN4	Designation
1	VDD_TARGET	VDD from application
2	SWCLK	SWD clock
3	GND	ground
4	SWDIO	SWD data input/output
5	NRST	RESET of target STM32
6	SWO	Reserved

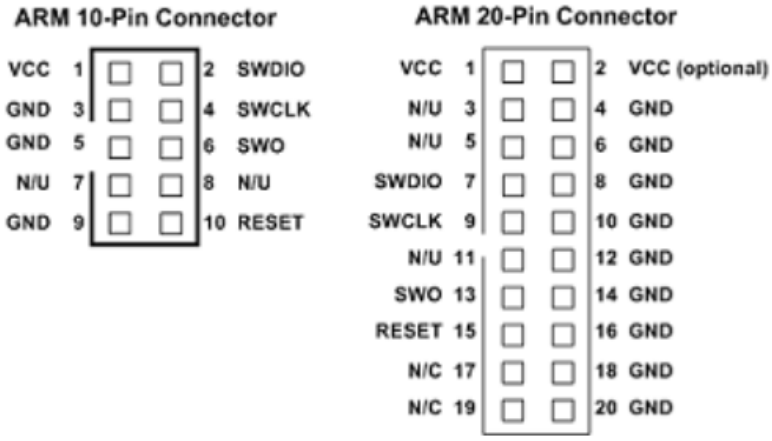
Figure 9. Using ST-LINK/V2-1 to program the STM32 on an external application



ST Link v2 programmer pin out



ARM JTAG 20 to 10 pin out



Directly soldering jumpers to the SWD pins of the STM prooved successful

The working connection based on the NUCLEO_F439ZI:

ARM 20-pin connector	STM32
VCC	5V pin
SWDIO	PA13(SWDIO)
SWCLK	PA14(SWCLK)
SWO	PB3(SWO)
RESET	RESET
GND	GND

UART message protocol

8 bytes message structure:

|<| : Start of message byte.
|ADDR| : Device Address byte.
|CMD| : Command byte.
|DATA1| : Data byte 1.
|DATA2| : Data byte 2.
|DATA3| : Data byte 3.
|DATA4| : Data byte 4.
|>| : End of message byte.

List of commands from CRIO to STM:

1. On/off command.
2. Change - the frequency of DAC channel 1.
3. Change - the frequency of DAC channel 2.
4. Change - the amplitude of DAC channel 1.
5. Change - the amplitude of DAC channel 2.
6. Request - Voltage and Current measurement of channel 1 output.
7. Request - Voltage and Current measurement of channel 2 output.
8. Request - Temperature sensor 1 and 2 output.
9. Acknowledge message received.
10. Bad message received.
11. Request current system state.

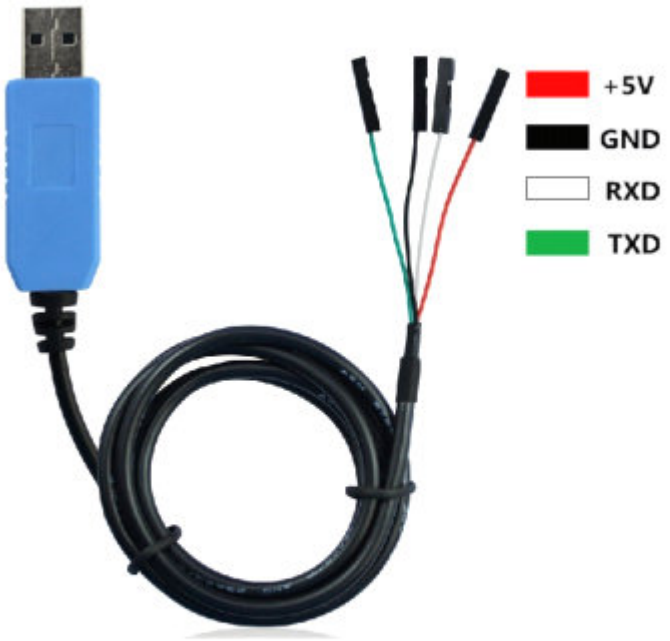
List of commands from STM to CRIO:

1. Return - the frequency of DAC channel 1.
2. Return - the frequency of DAC channel 2.
3. Return - the amplitude of DAC channel 1.
4. Return - the amplitude of DAC channel 2.
5. Return - Voltage and Current measurement of channel 1 output.
6. Return - Voltage and Current measurement of channel 2 output.
7. Return - Temperature sensors 1 and 2 output.
8. Acknowledge message received.
9. Bad message received.
10. Low power mode.

UART bootloader

Note that this has alll been done for the STM32F439ZI chip, each chip will have its own specific set up!

A USB to serial converter is needed to programme the MCU via a laptop.



- Red - VCC (5V)
- Black - GND
- Green - TXD - (3.3V TTL, connects to target RXD)
- White - RXD - (3.3V TTL, connects to target TXD)

Boot pin configurations:
Refer to the chip reference manual for the correct configuration of pins Boot0 and Boot1 in order to put the device into the desired bootloader mode.
For the STM32F439VGT6 the following pin configurations set the desired memory location to refer to for the boot mode.

BOOT0	BOOT1	Boot mode
0	0	Main Flash(Default)
0	1	System memory
1	0	
1	1	Embedded SRAM

USART bootloader protocol:
Refer to AN2026 to activate the specific peripheral bootloader pins and settings. For the USART1 bootloader protocol commands and process refer to AN3155.

Peripheral	State	Note
USART1	Enabled	OUSART1 configuration: 8 bits, even parity and 1 Stop bit
USART1_RX pin	Input	PA10 pin: USART1 in reception mode

Peripheral	State	Note
USART1_TX pin	Output	PA9 pin: USART1 in transmission mode

retrd

All communication from the programming tool to the device is verified by:

1. Checksum: received blocks of data bytes are XOR-ed. A byte containing the computed XOR of all previous bytes is added to the end of each communication (checksum byte). By XOR-ing all received bytes, data plus checksum, the result at the end of the packet must be 0x00.
2. For each command the host sends a byte and its complement (XOR = 0x00).
3. UART: 1 start bit, even parity and 1 stop bit.
4. Each packet is either accepted (ACK answer) or discarded (NACK answer):
    - ACK = 0x79
    - NACK = 0x1F

List of Bootloader commands
This is a snippet from AN3155. Each command that is sent through to the STM must be sent with it's compliment.

- The STM will return with a "79" to acknowledge a correct message.

Command[1]	Code	Description
Get[2]	0x00	Gets the version and the allowed commands supported by the current version of the bootloader.
Get Version & Read Protection Status[2]	0x01	Gets the bootloader version and the Read protection status of the Flash memory.
Get ID[2]	0x02	Gets the chip ID.
Read Memory[3]	0x11	Reads up to 256 bytes of memory starting from an address specified by the application.
Go[3]	0x21	Jumps to user application code located in the internal Flash memory or in the SRAM.
Write Memory[3]	0x31	Writes up to 256 bytes to the RAM or Flash memory starting from an address specified by the application.
Erase[3][4]	0x43	Erases from one to all the Flash memory pages.
Extended Erase[3][4]	0x44	Erases from one to all the Flash memory pages using two byte addressing mode (available only for v3.0 USART bootloader versions and above).
Write Protect	0x63	Enables the write protection for some sectors.
Write Unprotect	0x73	Disables the write protection for all Flash memory sectors.
Readout Protect	0x82	Enables the read protection.
Readout Unprotect[2]	0x92	Disables the read protection.
Get Checksum	0xA1	Computes a CRC value on a given memory area with a size multiple of 4 bytes.

1. If a denied command is received or an error occurs during the command execution, the bootloader sends NACK byte and goes back to command checking.
2. Read protection. When the RDP (Read protection) option is active, only this limited subset of commands is available. All other commands are NACK-ed and have no effect on the device. Once the RDP has been removed, the other commands become active.
3. Refer to STM32 product datasheets and to AN2606 to know the valid memory areas for these commands.
4. Erase (x043) and Extended Erase (0x44) are exclusive. A device can support either the Erase command or the Extended Erase command, but not both.

Once the system memory boot mode is entered and the STM32 has been configured:

1. The DFU command "7F 00",No compliment needed.
2. The GET command "00 FF"
3. The GO command "21 DE"
    - Send, the start address, this is usually 0x8000000, wait for ACK
    - Might require a check sum to be sent with this address, AN3155
4. The EXTENDED ERASE command "44 BB" (This seems to work inconsistantly).
    - First send the READ PROTECT command "82 7D"
    - Then send the READ UNPROTECT "92 6D"
    - The WRITE UNPROTECT command "73 8C"
    - Then send EXTENDED ERASE COMMAND "44 BB"
5. The WRITE MEMORY command "31 CE"

- Send, 31 CE, wait for ACK
- Send, 08 00 00 00 08, the start address which is usually 0x08000000 and the CRC, wait for ACK
- Send, the number of bytes to be written (1 byte), the data (N+1 bytes), the checksum, wait for ACK
  - Checksum byte: XOR (N, N+1 data bytes)
  - N+1 must be a multiple of 4

6. The WRITE PROTECT command "63 63"
7. The WRITE UNPROTECT command "73 8C"
8. The READ PROTECT command "82 7D"
9. The READ UNPROTECT command "92 6D"

Physical wires required between Waveform Generator and control enclosure

1. RS485_A
2. RS485_B
3. GND
4. Boot line
5. Reset

STM MPU MCU complementry system

STM32MP1:
The STM32MP157C/F devices are based on the high-performance dual-core Arm® Cortex®-A7 32-bit RISC core operating at up to 800 MHz. The STM32MP157C/F devices also embed a Cortex® -M4 32-bit RISC core operating at up to 209 MHz frequency.

Introduction:

- STM32MP1 web page. Website

Demonstration:

- PHYTEC phyCORE-STM32MP1 at Embedded World 2020. Video

Development:

- ST STM32MP1 workshop. Video Series
- Digikey tutorial for STM32MP1 development. Blog
- Azure IoT Edge and Remote FOTA on STM32MP1. Video

Development boards:

- Digikey, Development board, with secure boot. Product
- RS, Development board, without secure boot. Product

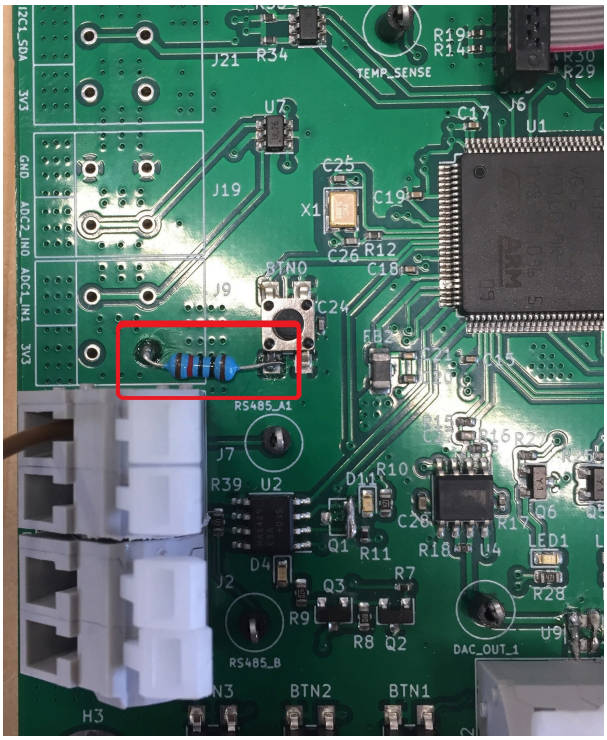
Goals:

1. Use the ST-Link v2 programmer to programme the manufactured PCBs.
2. Write the UART message handlers for the communications between the CRIO and the AWG.
3. Complete the Interface PCB for the Sledehammer amplifier.

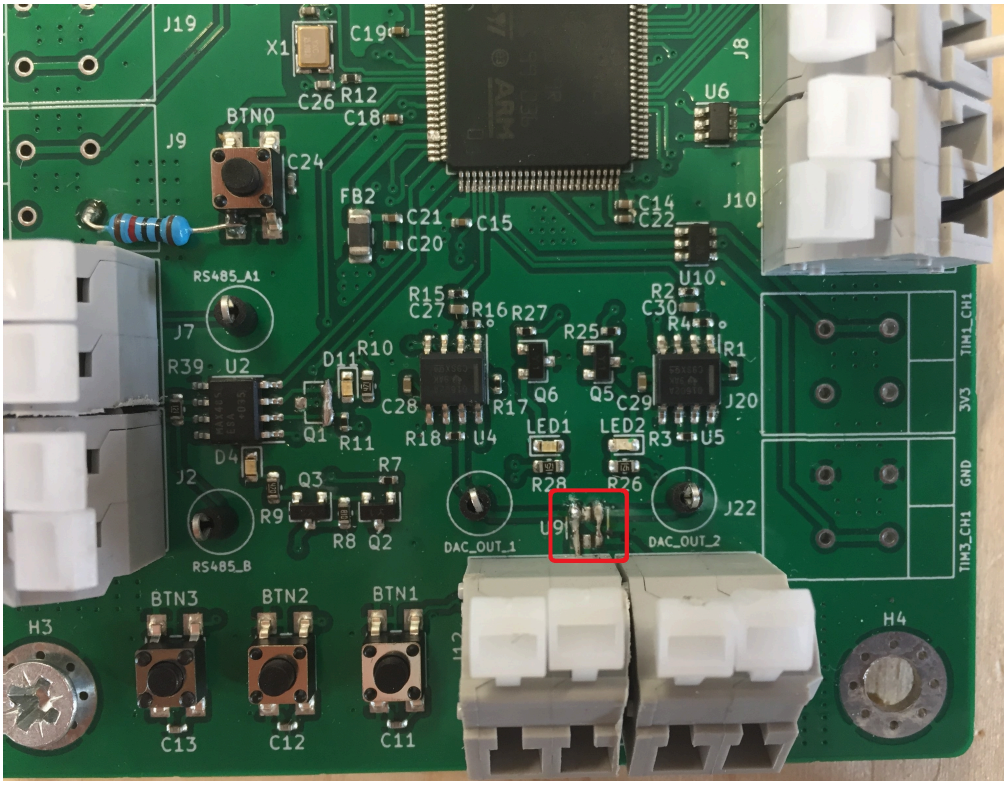
Setting up the AWG

1. 10kOhm resistor was added as a pull up resistor to bring the NRST pin to 3.3v
2. Removal of the ESD protection at the outputs of the DACs
3. Removal of the diode before thw MAX485 chip

The 10kOhm pull up resistor added can be seen in the figure below.

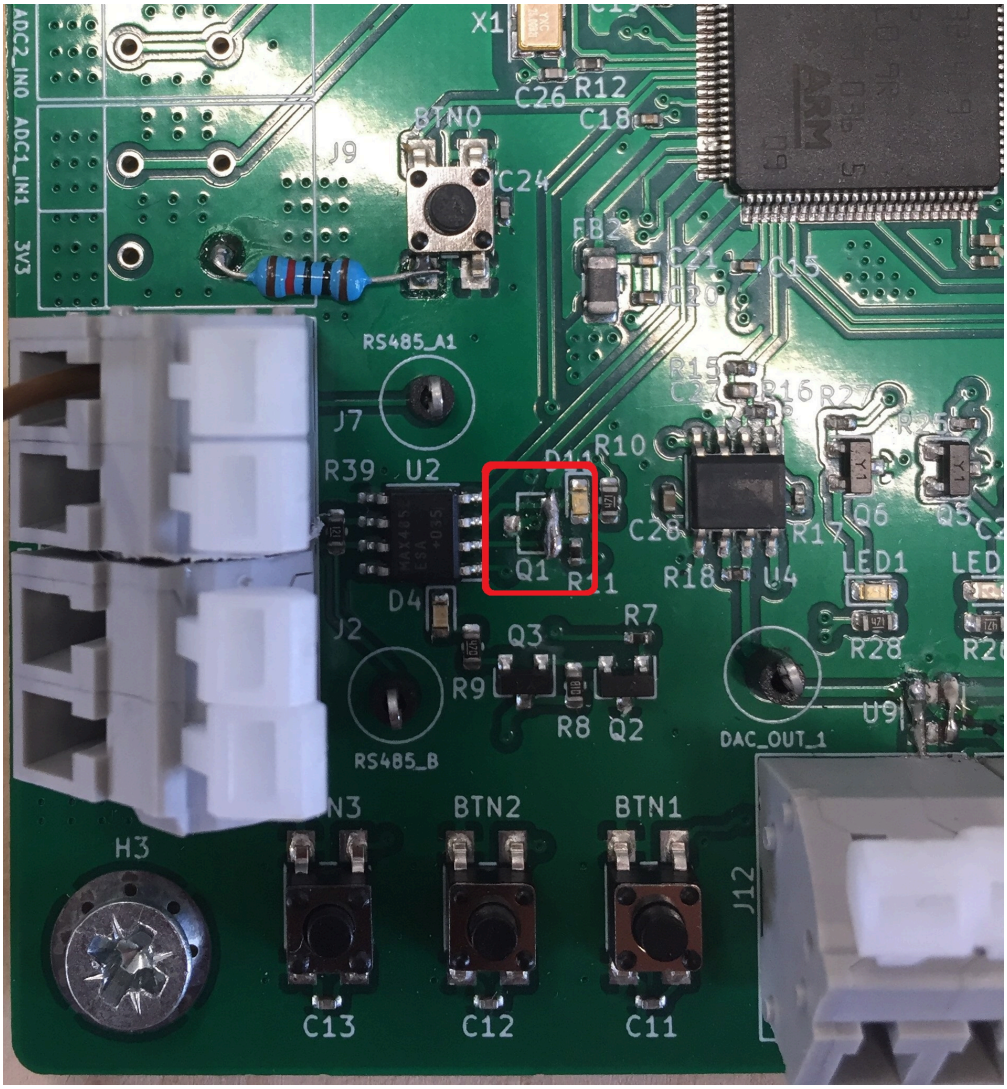


The ESD protection after the Op Amp circuits was removed as the diode to ground was clamping the circuit. Seen in the image below.



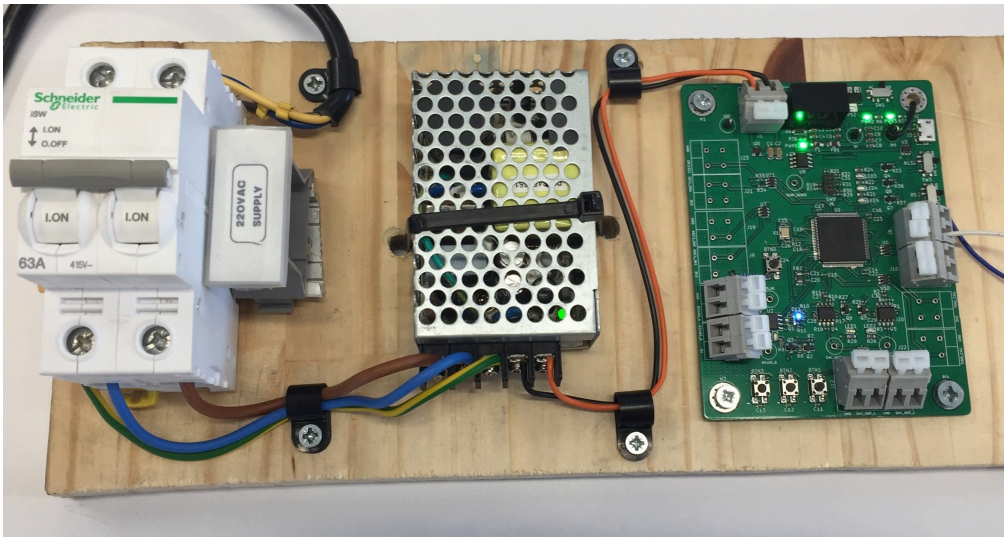
The inverting logic gate (The Npn diode) that was used to set the -- pin and -- pin which sets the MAX485 into transmit or recieve mode was removed. This logic is not necessary as the -- pin is inverted already so both pins recieve the same logic to switch between transmition and reception mode.

The removed diode location can be seen in the figure below.



Checking the Power Suppies

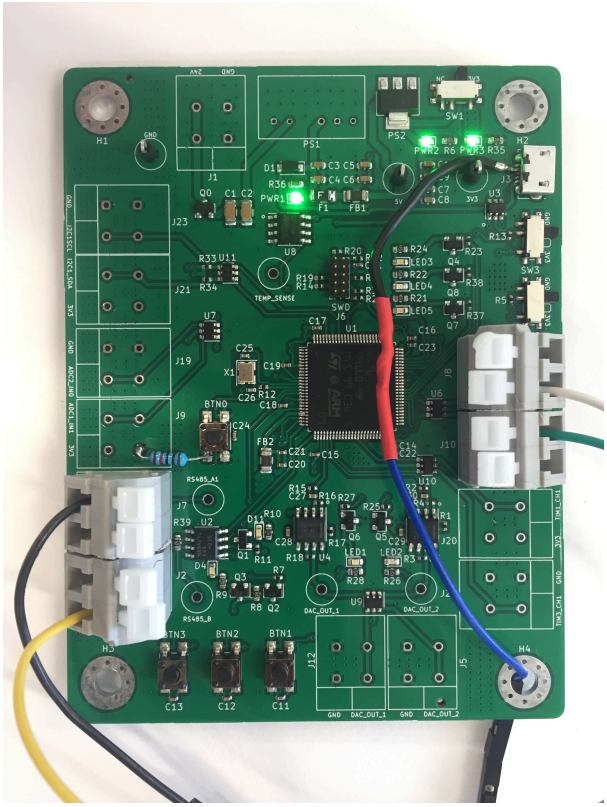
A 24V supply was connected to the 9V - 36V Input. +-5v and 3.3v was measured at the correct locations The STM MCU started up correctly and began to run the code.



UART Programming of the AWG

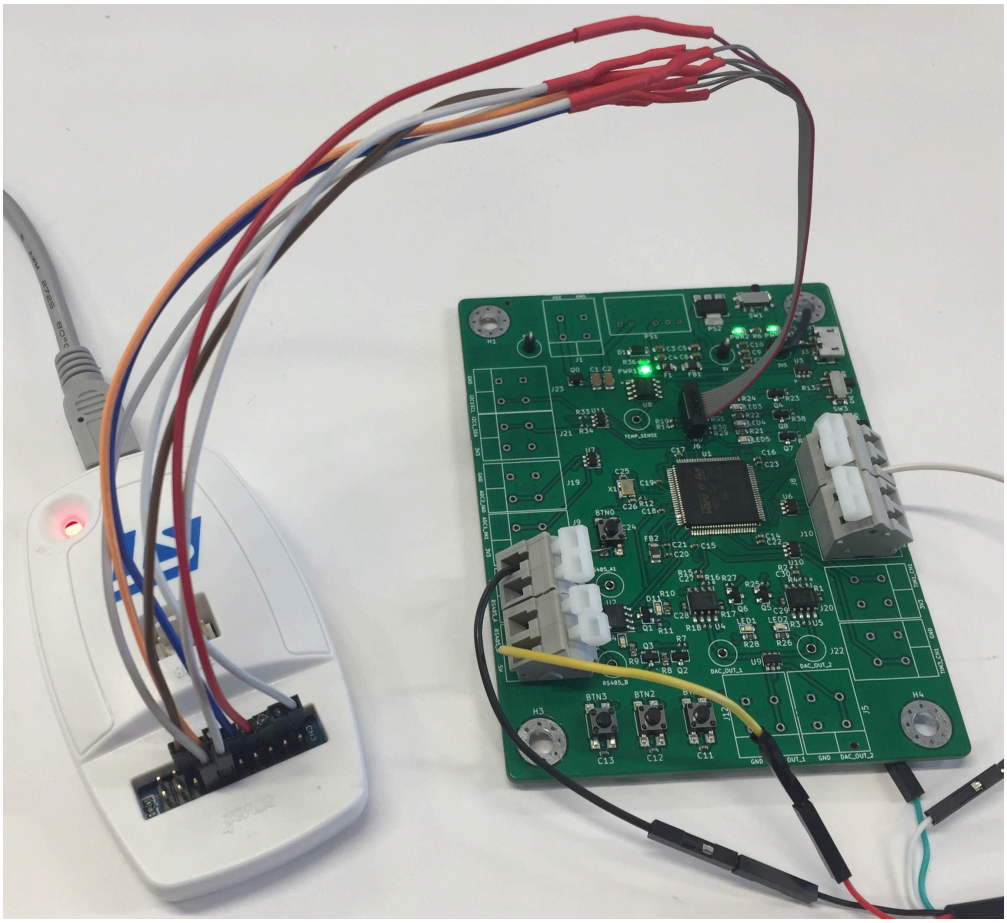
The AWG was first programmed via the USB to serial interface

1. An external wire was soldered to USB ID pin in order to access PA10
2. Boot0 switch(sw3) was set to 3v3 and the device was reset to enter bootloading mode.
3. Start STM32CubeProgrammer and select UART programming
4. Select erasing & programming.
5. Select the .elf file to be uploaded, this can be found in the Debug folder of the build files from the STM32CubeIDE.



ST Link V2 Programming of the AWG

1. Connect debugger and AWG PCB using the connector as seen in the figure below.
    - The footprint for the debugger on the AWG PCB is incorrected and inverted therefore the connector had to be manually wired to the correct orientation.
    - Add an extra wire from the ground of the debugger to the ground of the AWG PCB.
2. Start STM32CubeProgrammer and select ST-Link programming
3. Select erasing & programming.
4. Select the .elf file to be uploaded, this can be found in the Debug folder of the build files from the STM32CubeIDE.



Testing waveform generation

The max output frequency is dependent on Ns, the number of samples that form one full period of the sine wave. For lower output frequencies a higher Ns can be used to increase the resolution of the signal. THe following table indicates the apropriate ns value for a desired output frequency range. These are based on the STM32F429VGT6 used on the AWG.

Based on the calculation:

$$PSC = \frac{\frac{Fclock}{Ns}}{Fsine*(Period+1)} - 1$$

Ns	Max Output Frequency
40	<= 20kHz
75	<= 10kHz
180	<= 5kHz
240	<= 2.5kHz

There should be a function in the code that takes in the desired frequency and

selects the appropriate Ns.

Other requirements for AWG:

1. The output signal must be scaled by o.5.
2. The output signal must be offset by 500.
3. DAC resolution = 4096
4. Fclock = 90 MHz
5. Period = 1.
6. PSC > 50

In [ ]:

```
Running cells with 'Python 3.12.7' requires the ipykernel package.

Run the following command to install 'ipykernel' into the Python environme
nt.

Command: '/usr/local/bin/python3.12 -m pip install ipykernel -U --user --f
orce-reinstall'
```