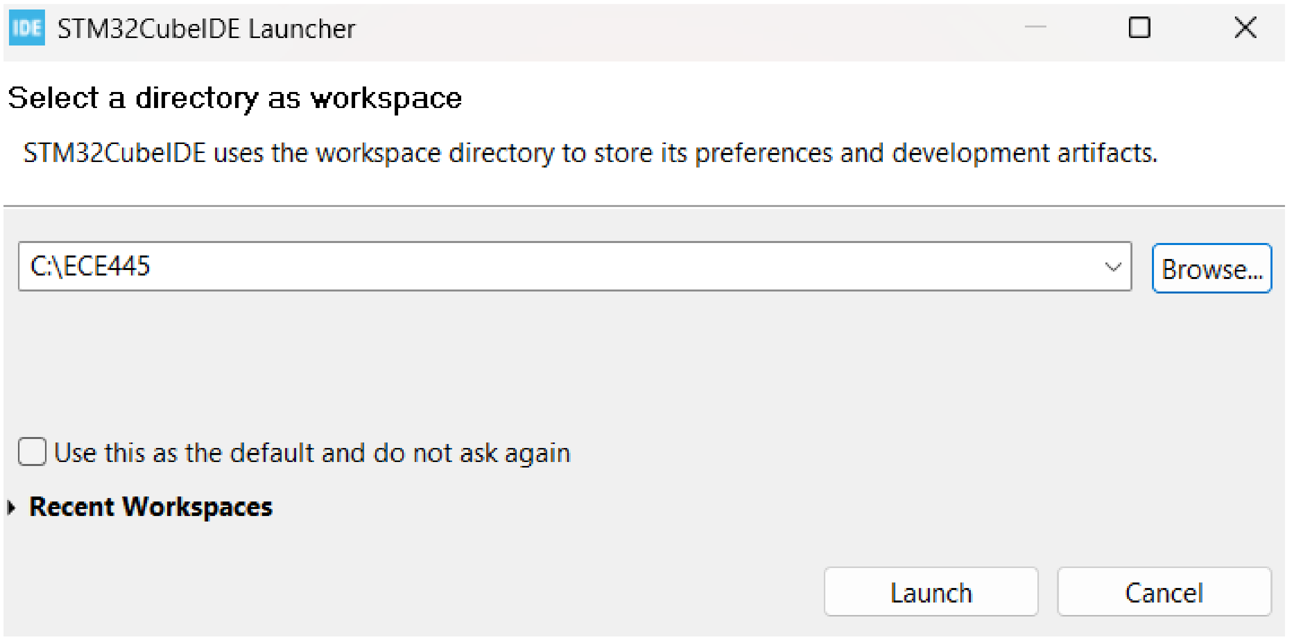
ECE 445 CAD Assignment

Setting up your microcontroller configuration

The first step of STM32 based development is to use the SMT32Cube IDE to

Workspace and project Setup

1. Open STM32CubeIDE


STM32CubeIDE Launcher

Select a directory as workspace

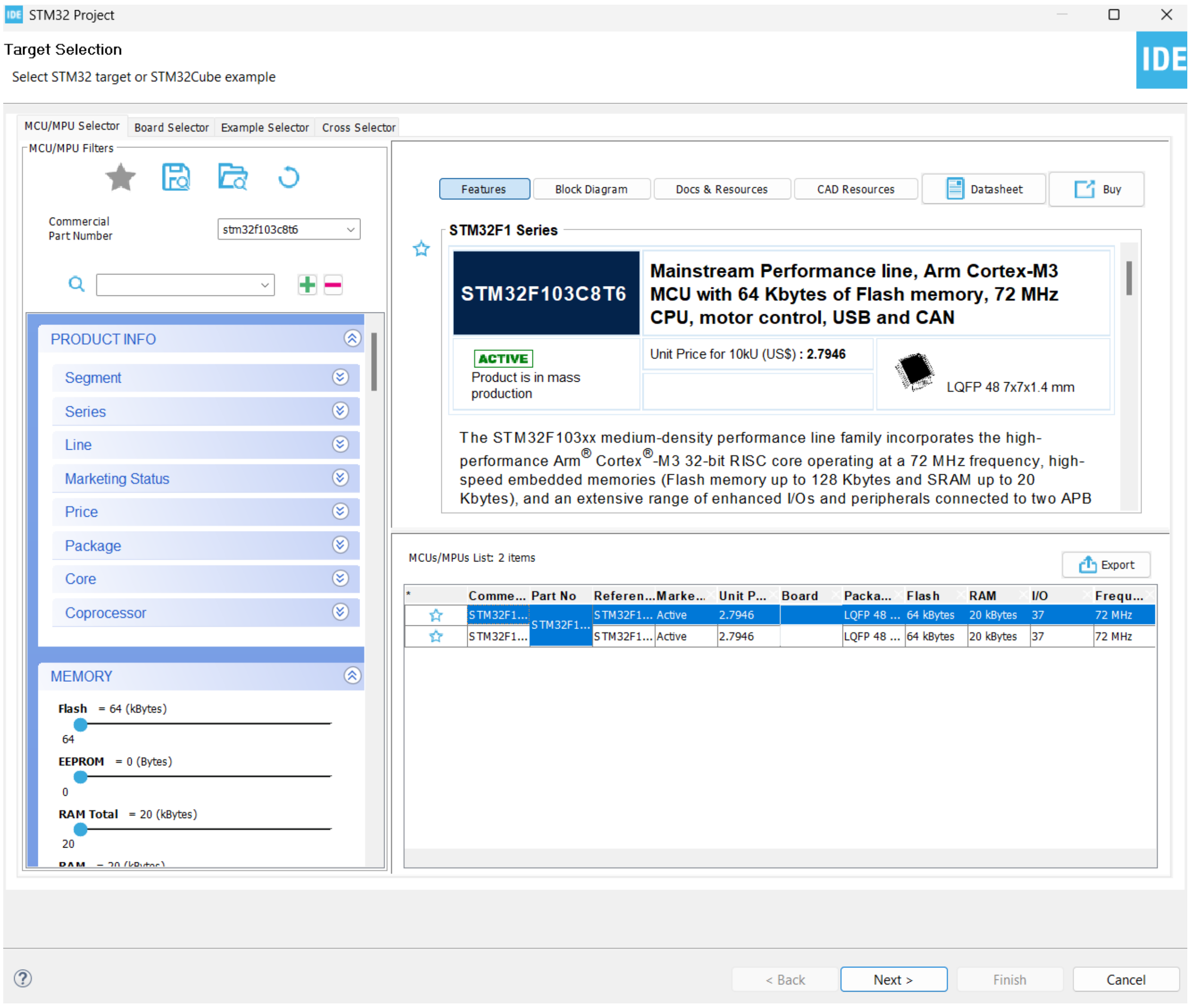
STM32CubeIDE uses the workspace directory to store its preferences and development artifacts.

C:\ECE445

Use this as the default and do not ask again

2. Select a workspace
3. "Start new project"

4. Select STM32F103C8T6



Target Selection

Select STM32 target or STM32Cube example

Commercial Part Number: stm32f103c8t6

Flash = 64 (kBytes)
64

EEPROM = 0 (Bytes)
0

RAM Total = 20 (kBytes)
20

RAM = 20 (kBytes)

STM32F1 Series

STM32F103C8T6

Mainstream Performance line, Arm Cortex-M3 MCU with 64 Kbytes of Flash memory, 72 MHz CPU, motor control, USB and CAN

ACTIVE
Product is in mass production

Unit Price for 10kU (US$) : **2.7946**

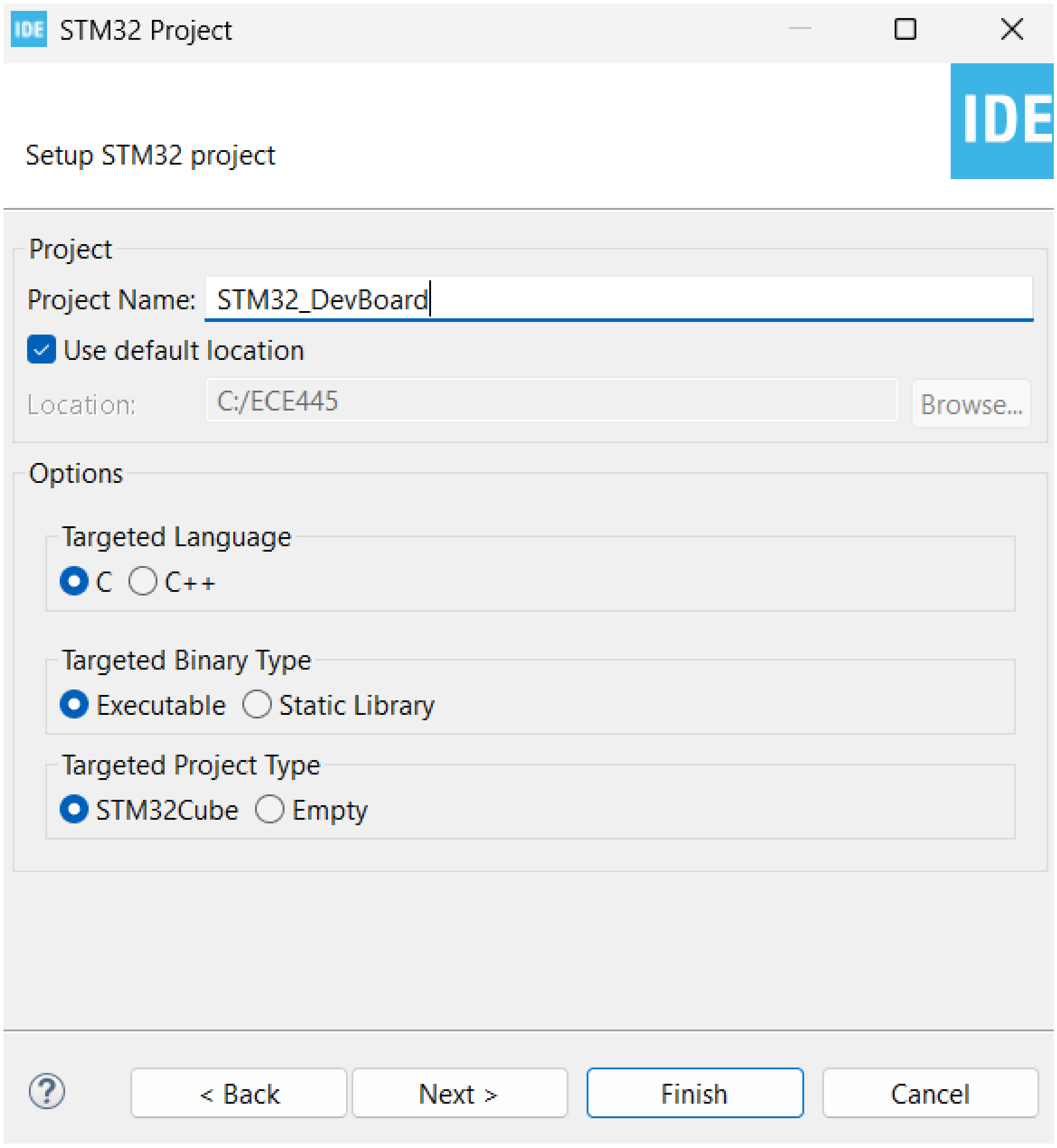
LQFP 48 7x7x1.4 mm

The STM32F103xx medium-density performance line family incorporates the high-performance Arm® Cortex®-M3 32-bit RISC core operating at a 72 MHz frequency, high-speed embedded memories (Flash memory up to 128 Kbytes and SRAM up to 20 Kbytes), and an extensive range of enhanced I/Os and peripherals connected to two APB

MCUs/MPUs List: 2 items

*	Comme...	Part No	Referen...	Marke...	Unit P...	Board	Packa...	Flash	RAM	I/O	Frequ...
☆	STM32F1...	STM32F1...	STM32F1...	Active	2.7946		LQFP 48 ...	64 kBytes	20 kBytes	37	72 MHz
☆	STM32F1...		STM32F1...	Active	2.7946		LQFP 48 ...	64 kBytes	20 kBytes	37	72 MHz

5. Name your project, click yes when asked to open in device configuration view


STM32 Project

Setup STM32 project

Project

Project Name: STM32_DevBoard

[✓] Use default location

Location: C:/ECE445

Options

Targeted Language

[✓ C] C++

Targeted Binary Type

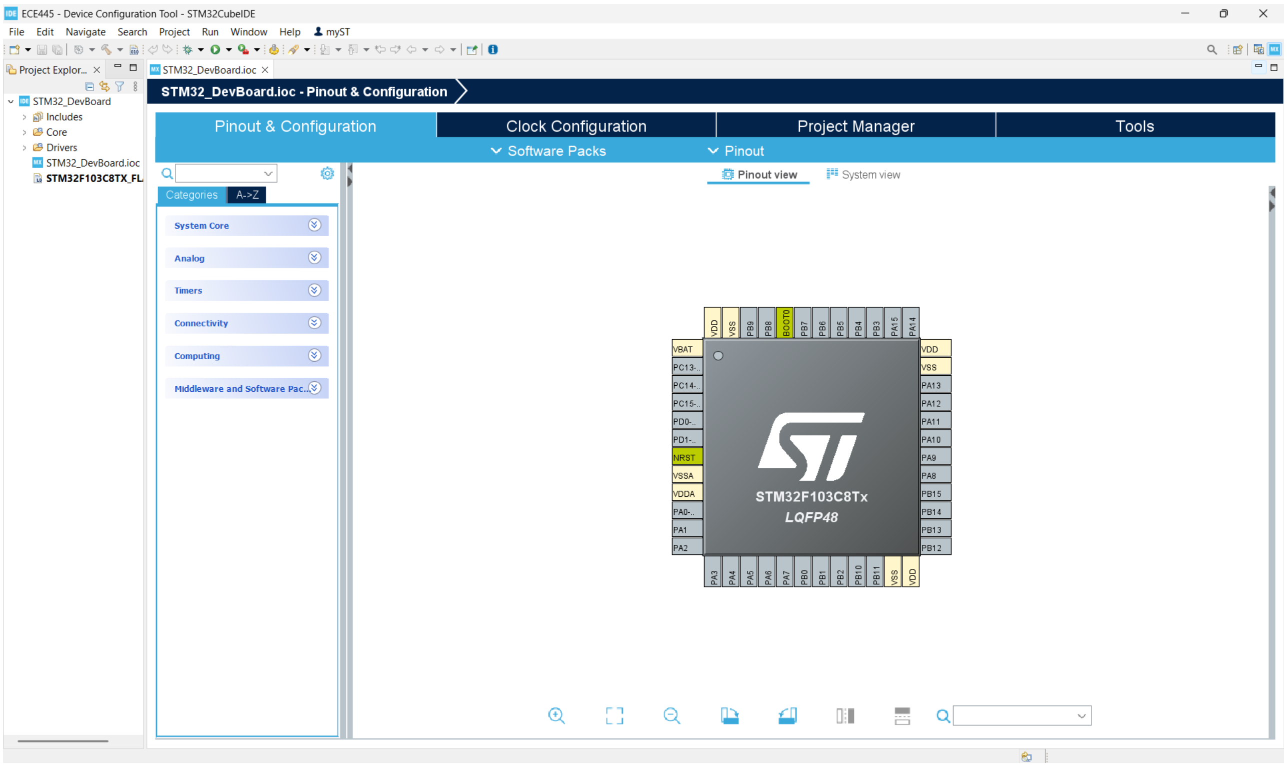
[✓ Executable] Static Library

Targeted Project Type

[✓ STM32Cube] Empty

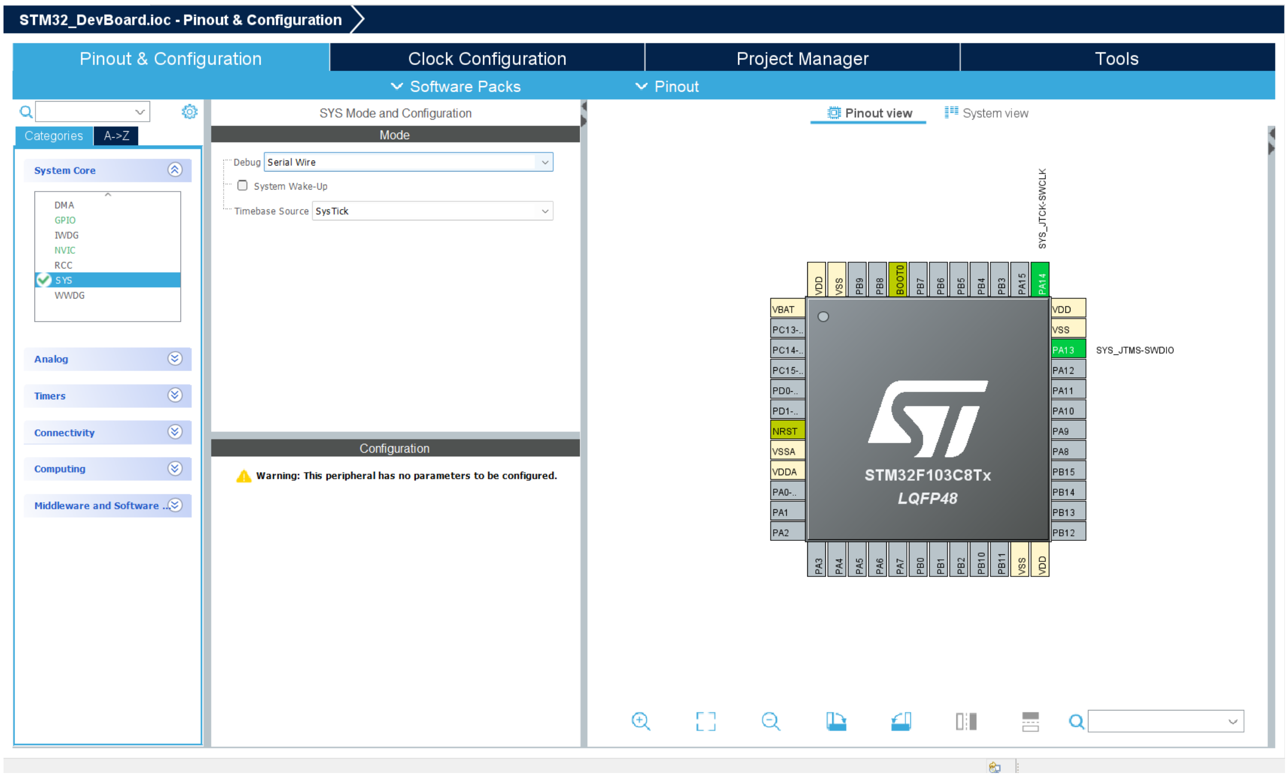
Configure microcontroller

- For this project, we will be adding Serial Wire Debug (SWD), USB CDC, I2C, SPI, UART, and several GPIO pins



Serial Wire Debug (SWD)

- `System Core` -> `SYS` -> `Serial Wire`



Debug: Serial Wire

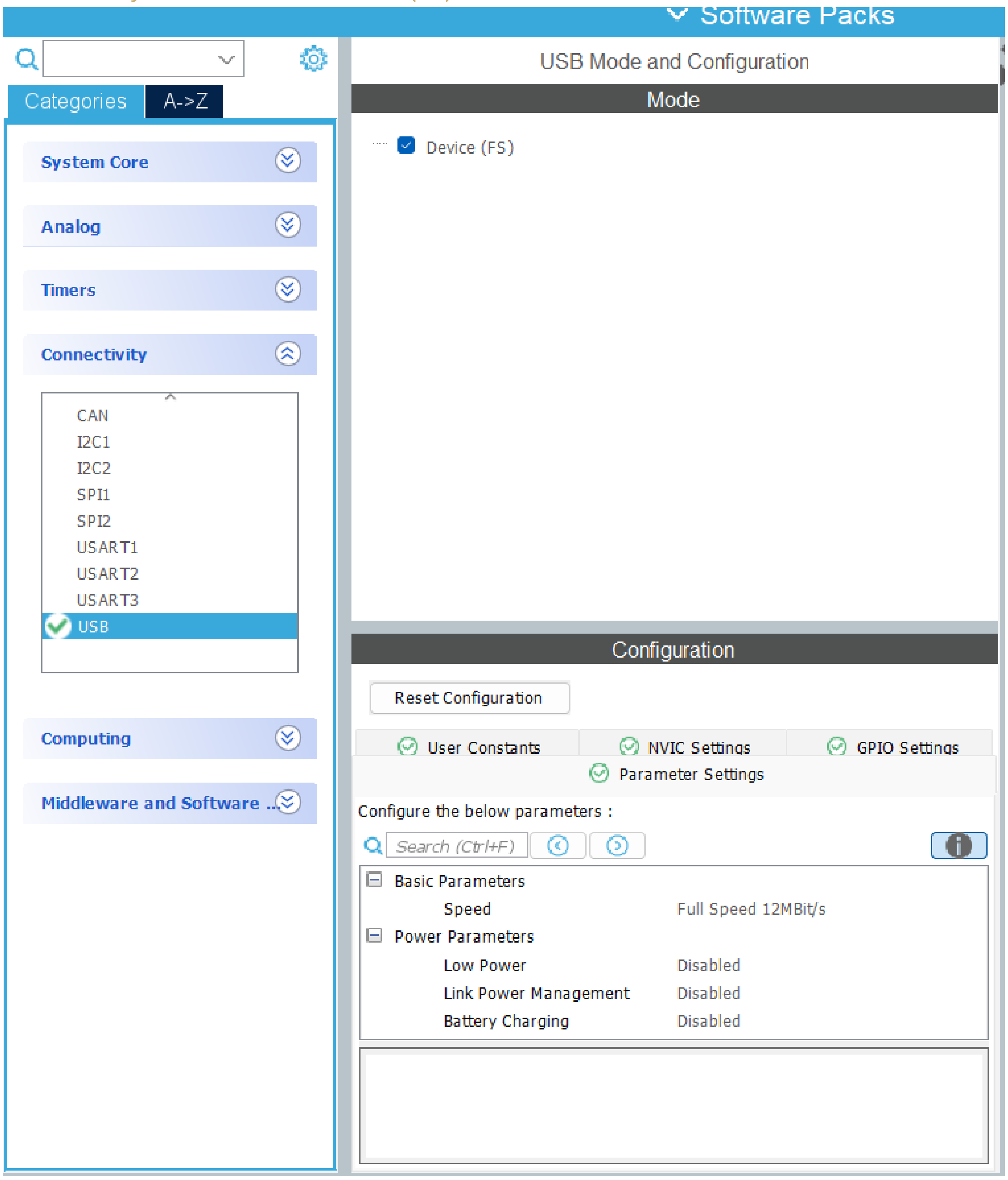
☐ System Wake-Up

Timebase Source: SysTick

Warning: This peripheral has no parameters to be configured.

USB Communication Device Class (USB CDC)

- `Connectivity` -> `USB` -> Enable `Device (FS)`



USB Mode and Configuration

Mode

☑ Device (FS)

Configuration

Configure the below parameters :

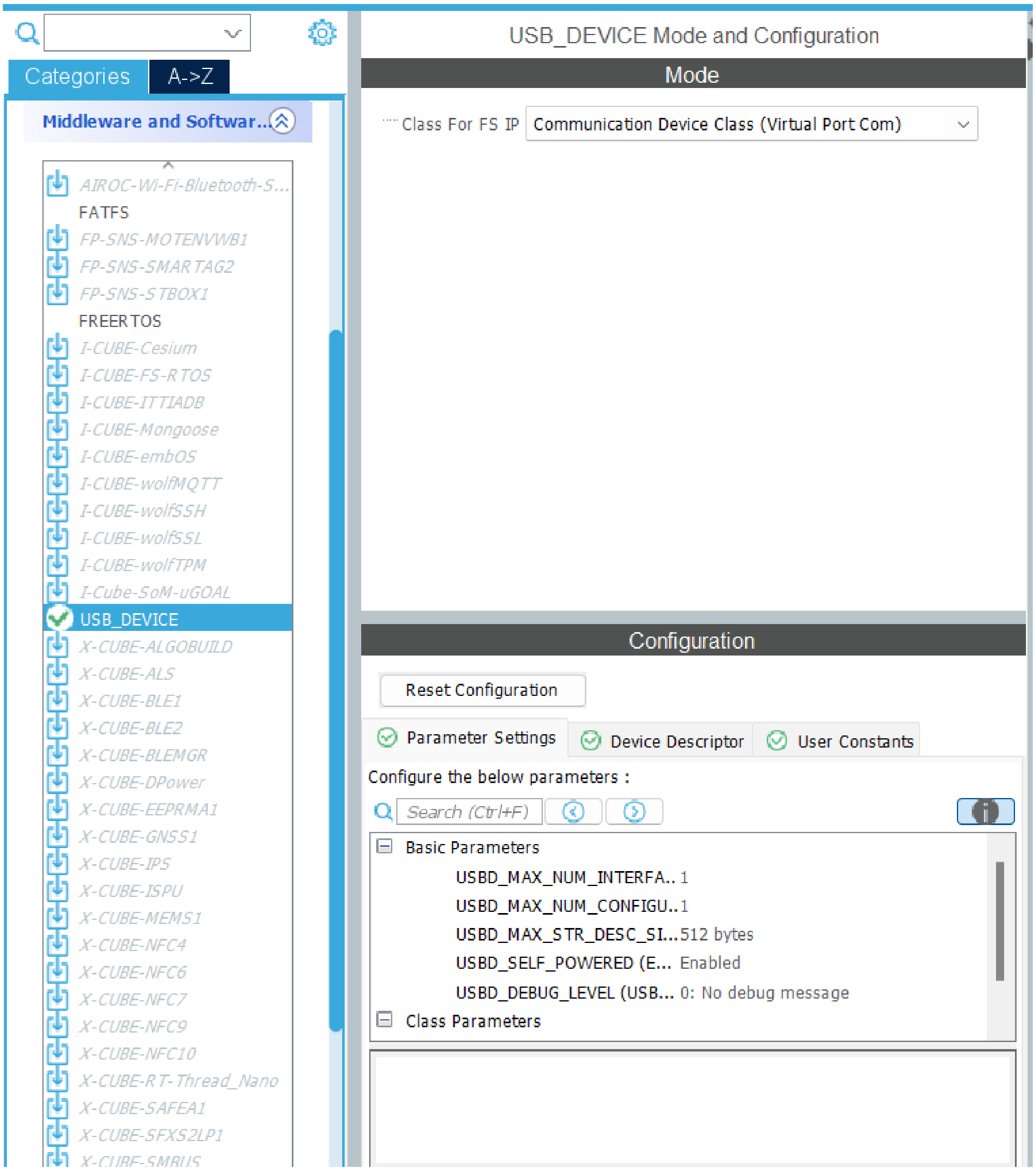
⊟ Basic Parameters

Speed	Full Speed 12MBit/s

⊟ Power Parameters

Low Power	Disabled
Link Power Management	Disabled
Battery Charging	Disabled

- `Middleware and Software` -> `USB_DEVICE` -> `Class for FS IP` -> `Communication Device Class (Virtual Port Com)`



USB_DEVICE Mode and Configuration

Class For FS IP: Communication Device Class (Virtual Port Com)

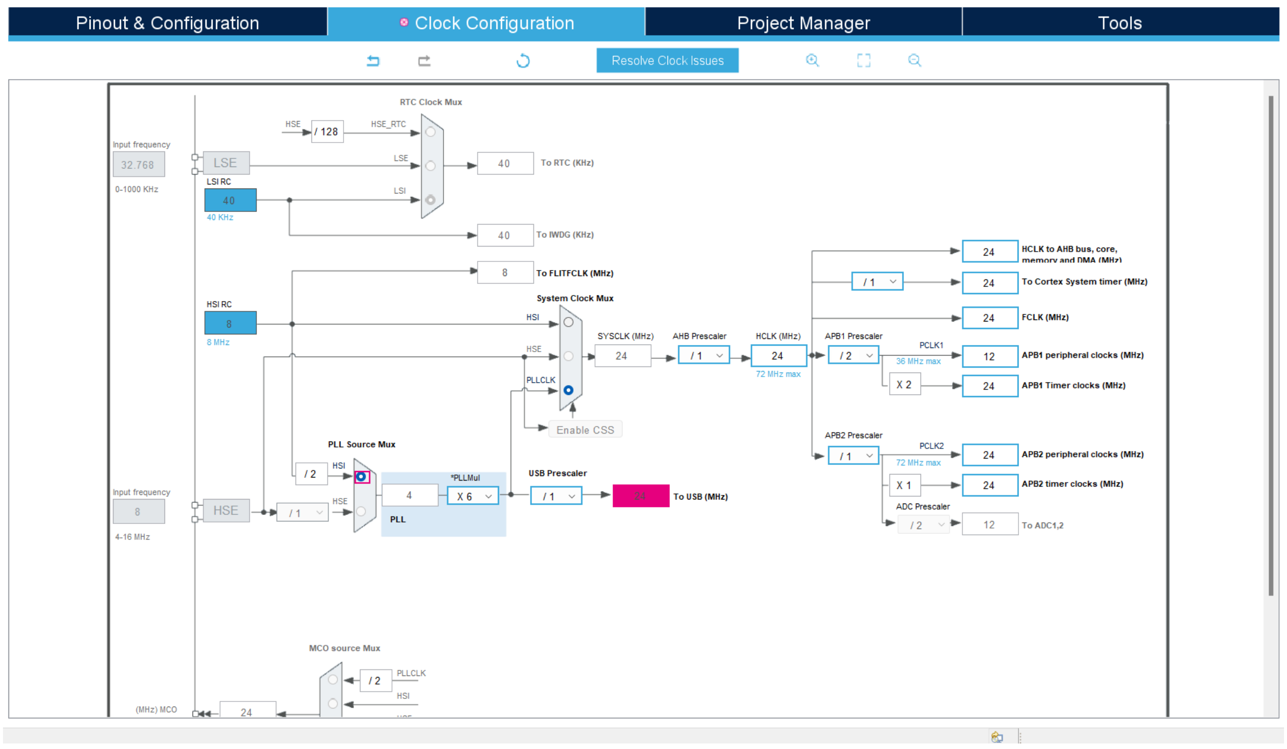
Configure the below parameters :

Basic Parameters
- USBD_MAX_NUM_INTERFA.. 1
- USBD_MAX_NUM_CONFIGU.. 1
- USBD_MAX_STR_DESC_SI... 512 bytes
- USBD_SELF_POWERED (E... Enabled
- USBD_DEBUG_LEVEL (USB... 0: No debug message

Class Parameters

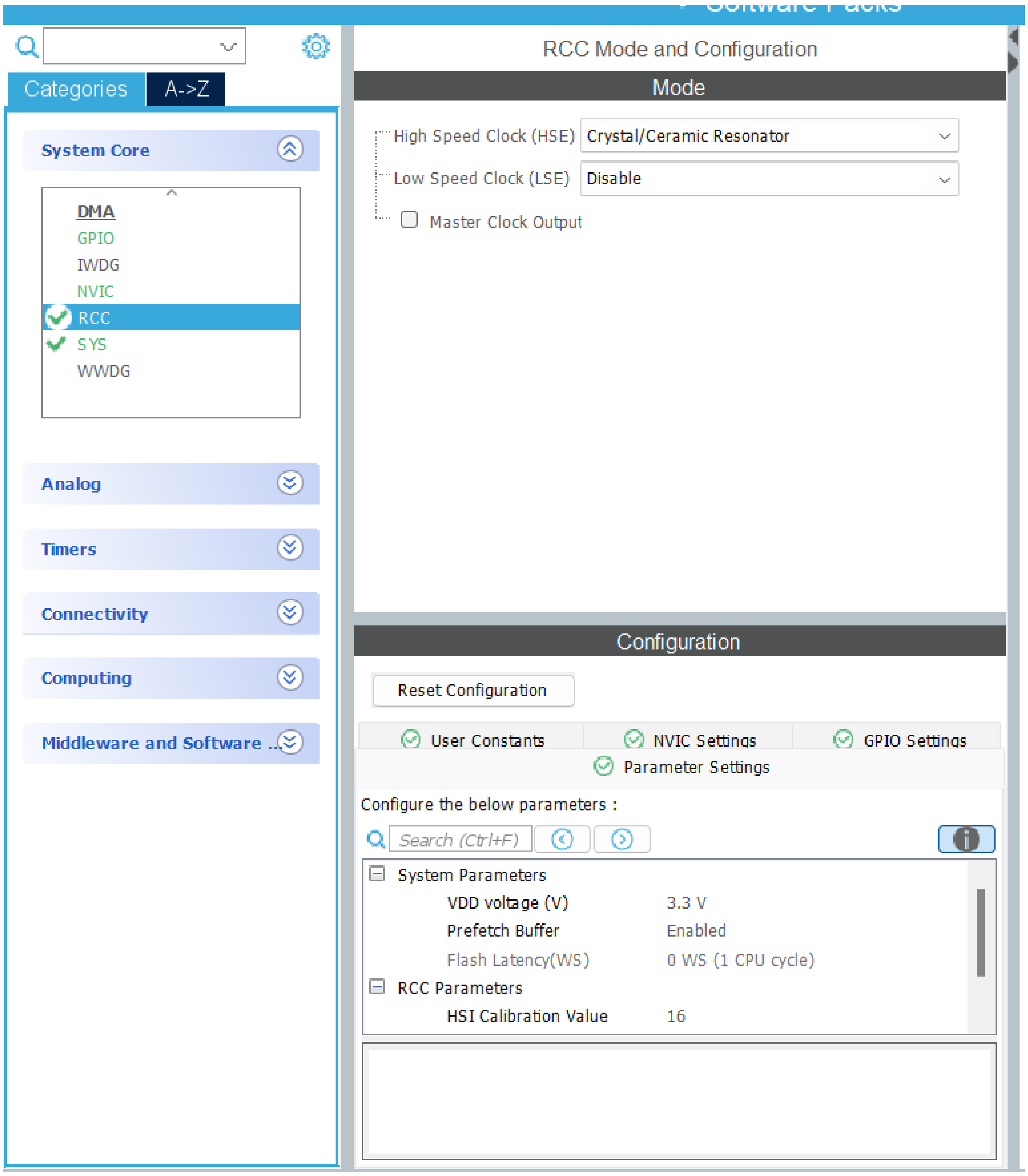
Clock Configuration pt 1

- Go to the Clock Configuration tab. Notice that when you click "Resolve Clock Issues" it cannot. We need to use an external clock source to get a higher frequency signal for the USB clock.



External Oscillator (for USB)

- `System Core` -> `RCC` -> `High Speed Clock` -> `Crystal/Ceramic Resonator`



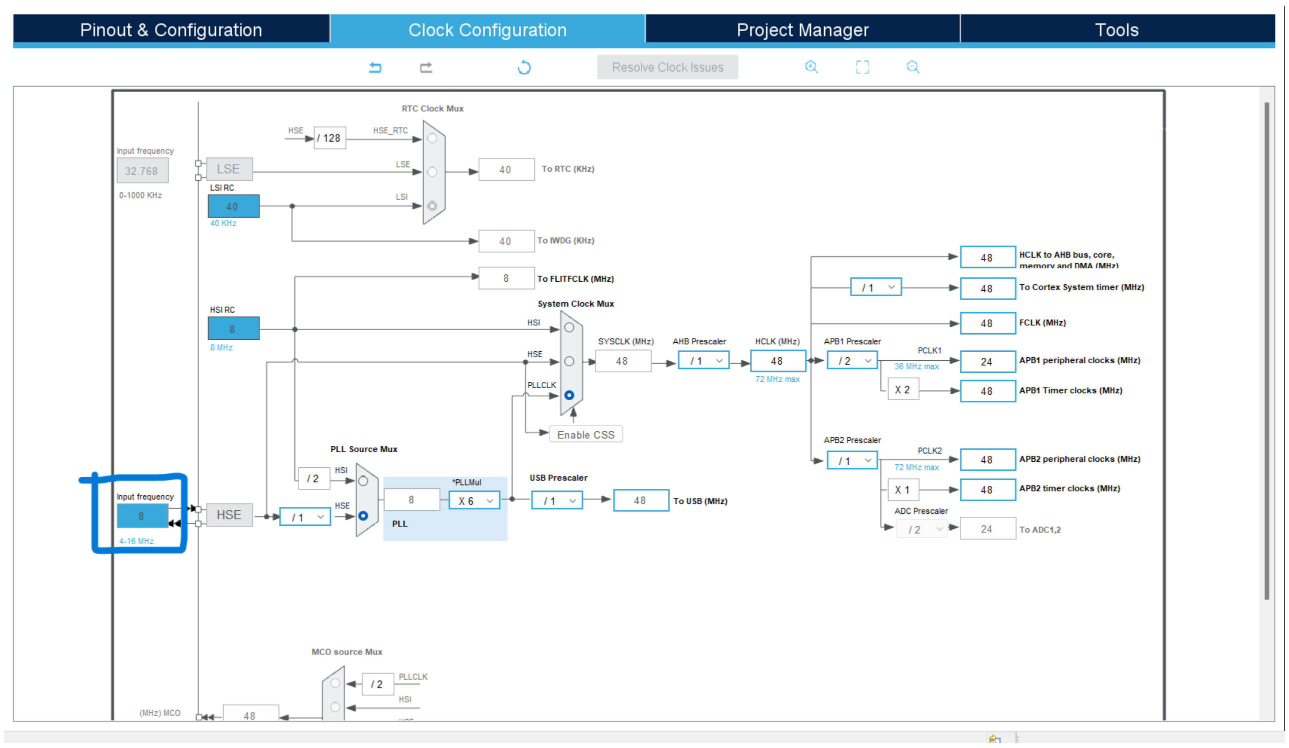
High Speed Clock (HSE): Crystal/Ceramic Resonator
Low Speed Clock (LSE): Disable
Master Clock Output

Configure the below parameters :

System Parameters
VDD voltage (V) 3.3 V
Prefetch Buffer Enabled
Flash Latency(WS) 0 WS (1 CPU cycle)
RCC Parameters
HSI Calibration Value 16

Clock Configuration pt 2

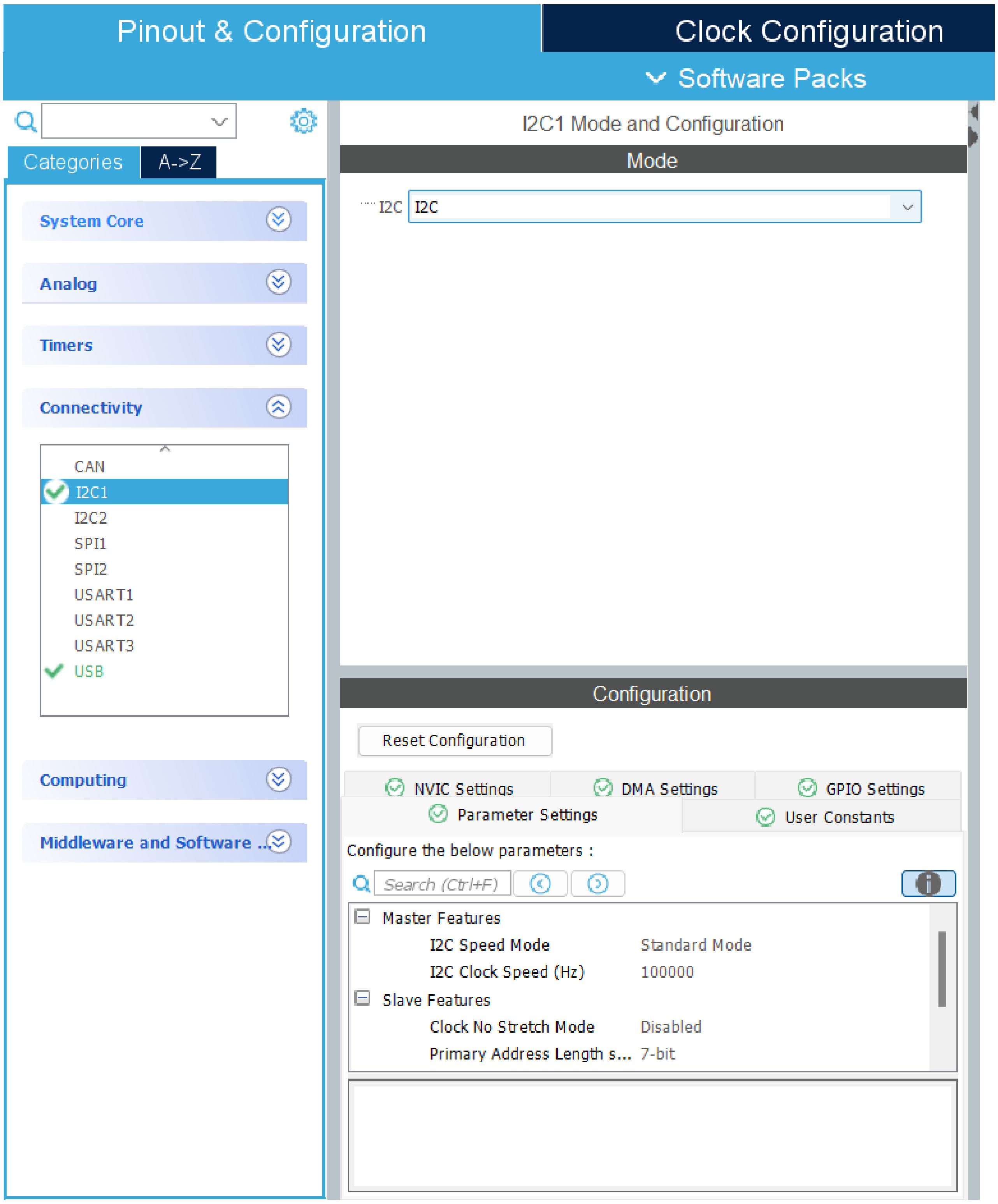
- Go to the Clock Configuration tab. Next to HSE make sure you enter 8 MHz. This will be the frequency of our crystal oscillator. Now run "Resolve Clock Issues". It will not give any errors.



Inter-Intergrated Circuit (I2C)

- I2C is a adress-based bus communication protocol. It uses 2 wires (Clock and Data) to connect to several devices. Every message sent has an address, and the device with that address stores the incoming message.

- `Connectivity` -> `I2C1` -> `I2C`



I2C1 Mode and Configuration

Mode

I2C: I2C

Configuration

Configure the below parameters :

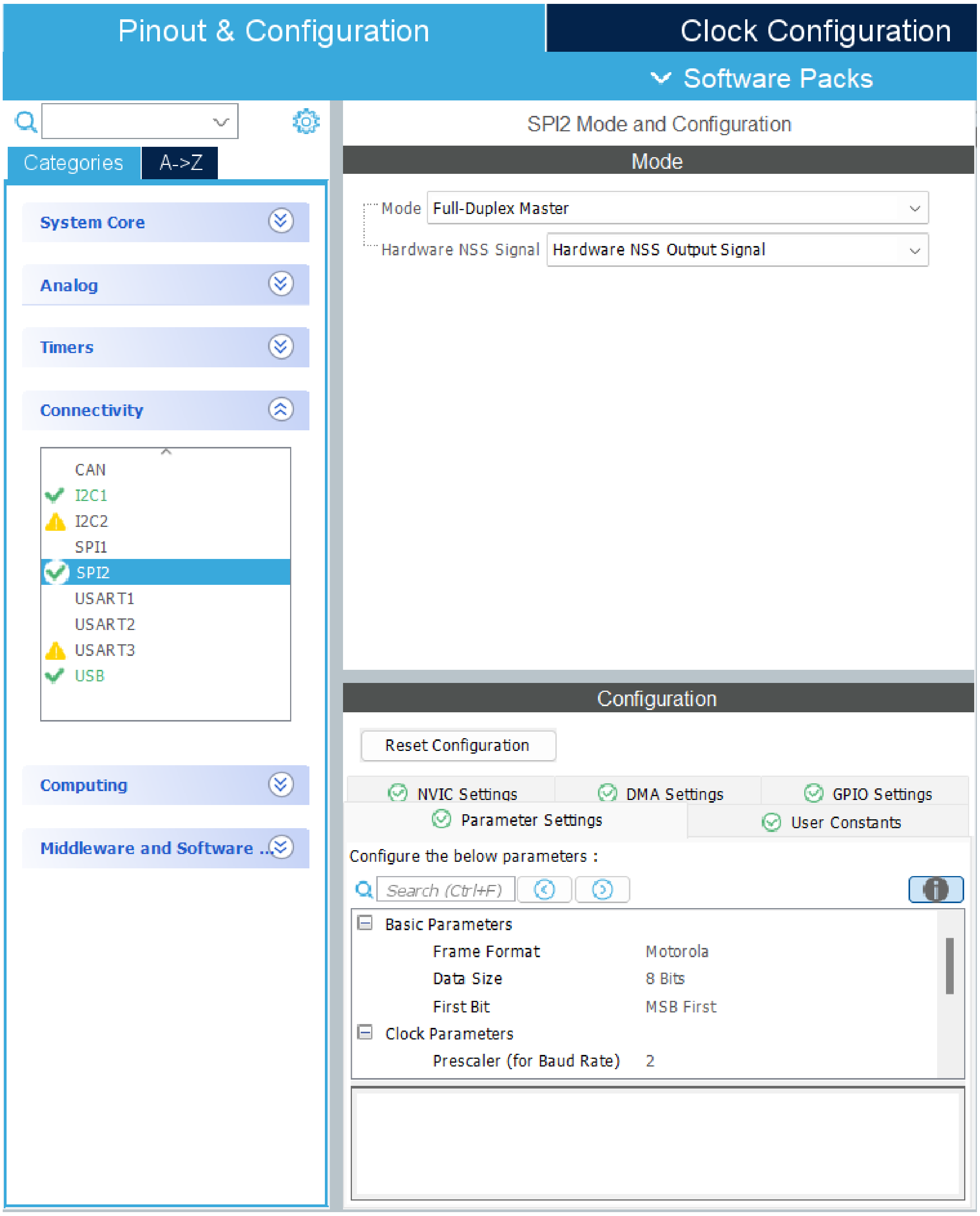
Master Features
- I2C Speed Mode: Standard Mode
- I2C Clock Speed (Hz): 100000

Slave Features
- Clock No Stretch Mode: Disabled
- Primary Address Length s...: 7-bit

Serial Peripheral Interface (SPI)

- SPI is a full duplex digital communication protocol which uses a select wire (typically CS or SS) to select which device to send signals to.

- `Connectivity` -> `SPI2` -> `Full Duplex Master` and `Hardware NSS Output Signal`



Mode: Full-Duplex Master
Hardware NSS Signal: Hardware NSS Output Signal

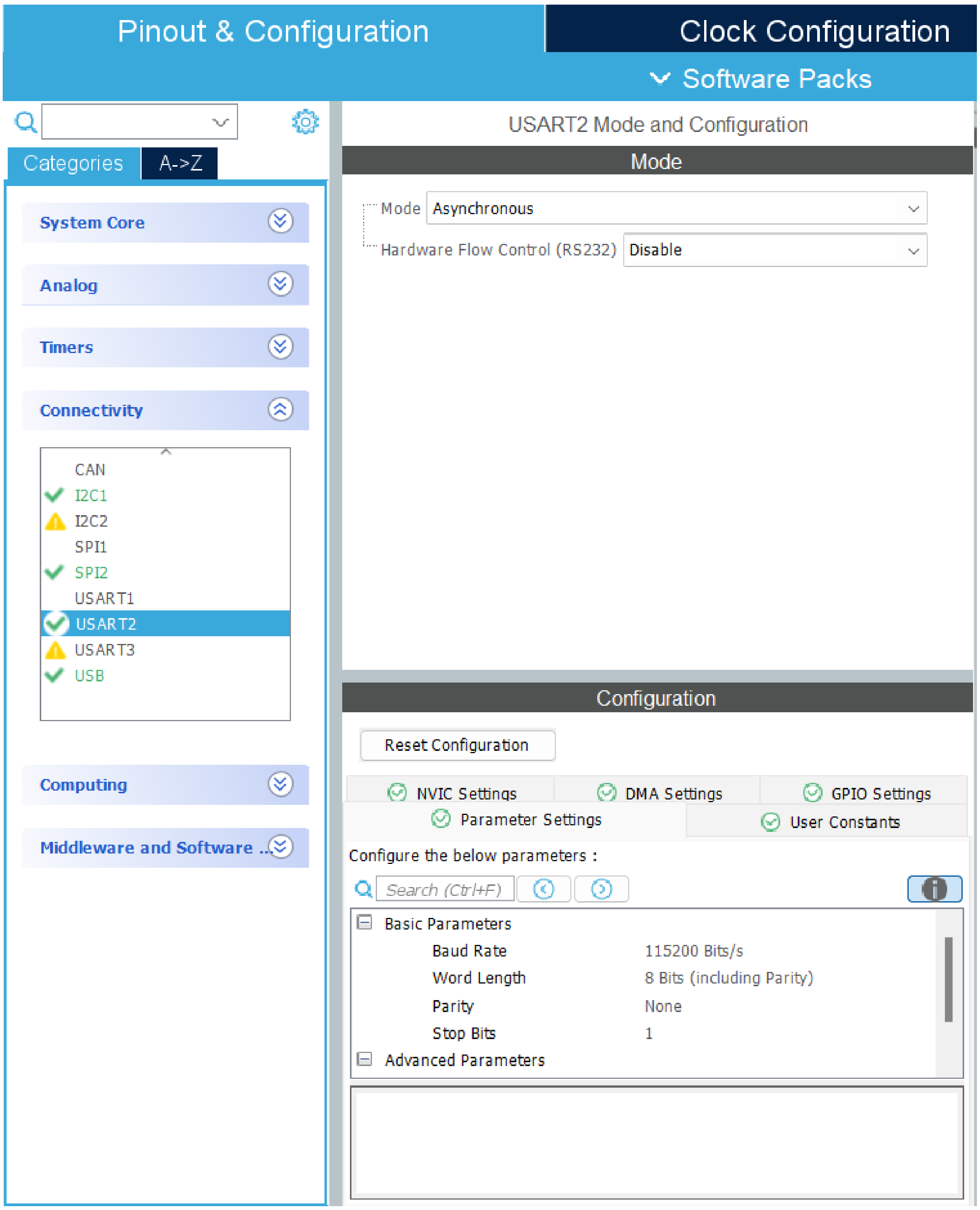
Configure the below parameters :

Basic Parameters
Frame Format Motorola
Data Size 8 Bits
First Bit MSB First
Clock Parameters
Prescaler (for Baud Rate) 2

Universal Synchronous Asynchronous Receiver/Transmitter (USART)

- USART is a full duplex, (typically) asynchronous digital communication protocol. The communicating devices must be set to have the same data rates and data can be received and transmitted simultenously(full duplex) on 2 different wires. TX (transmit) from one device should be attached to RX (receive) on the other device and vice versa.

- `Connectivity` -> `USART2`



USART2 Mode and Configuration

Mode

Mode Asynchronous

Hardware Flow Control (RS232) Disable

Configuration

Configure the below parameters :

Basic Parameters

Baud Rate	115200 Bits/s
Word Length	8 Bits (including Parity)
Parity	None
Stop Bits	1

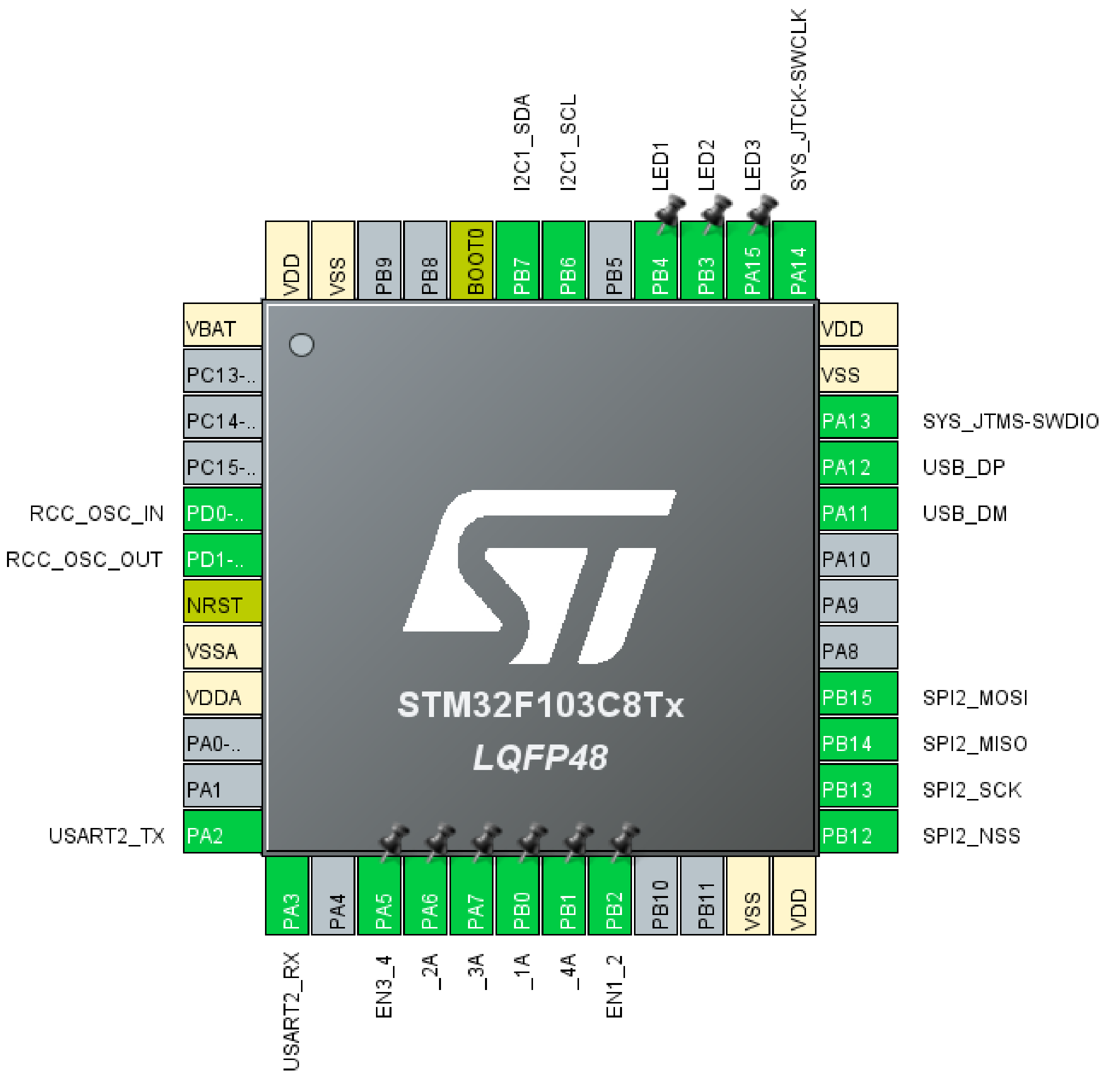
Advanced Parameters

General Purpose Input/Output (GPIO)

- GPIOs can be either analog (taking values between 0 and 3.3V) or digital (either 0 or 3.3V) and are used for controlling peripheral devices. In this case, an H-bridge motor driver (L293D).
- Left click on pins PA5, PA6, PA7, PB0, PB1, PB2, PB3, PB4, PA15 and select GPIO Output
- Right click on pins PA5, PA6, PA7, PB0, PB1, PB2, PB3, PB4, PA15 and Enter User Label. Name them as EN3_4, _2A, _3A, _1A, _4A, and EN1_2, LED2, LED1, LED3 respectively.

It is best to have an idea of how your peripheral ICs have their pins physically arranged, that way you can label your MCU pins to that they line up and do not have to criss-cross traces later on.

The IC should now look like:



Code Generation

Click the Device Configuration Tool Code Generation icon and click yes. This generates the starter code, we will return here later.

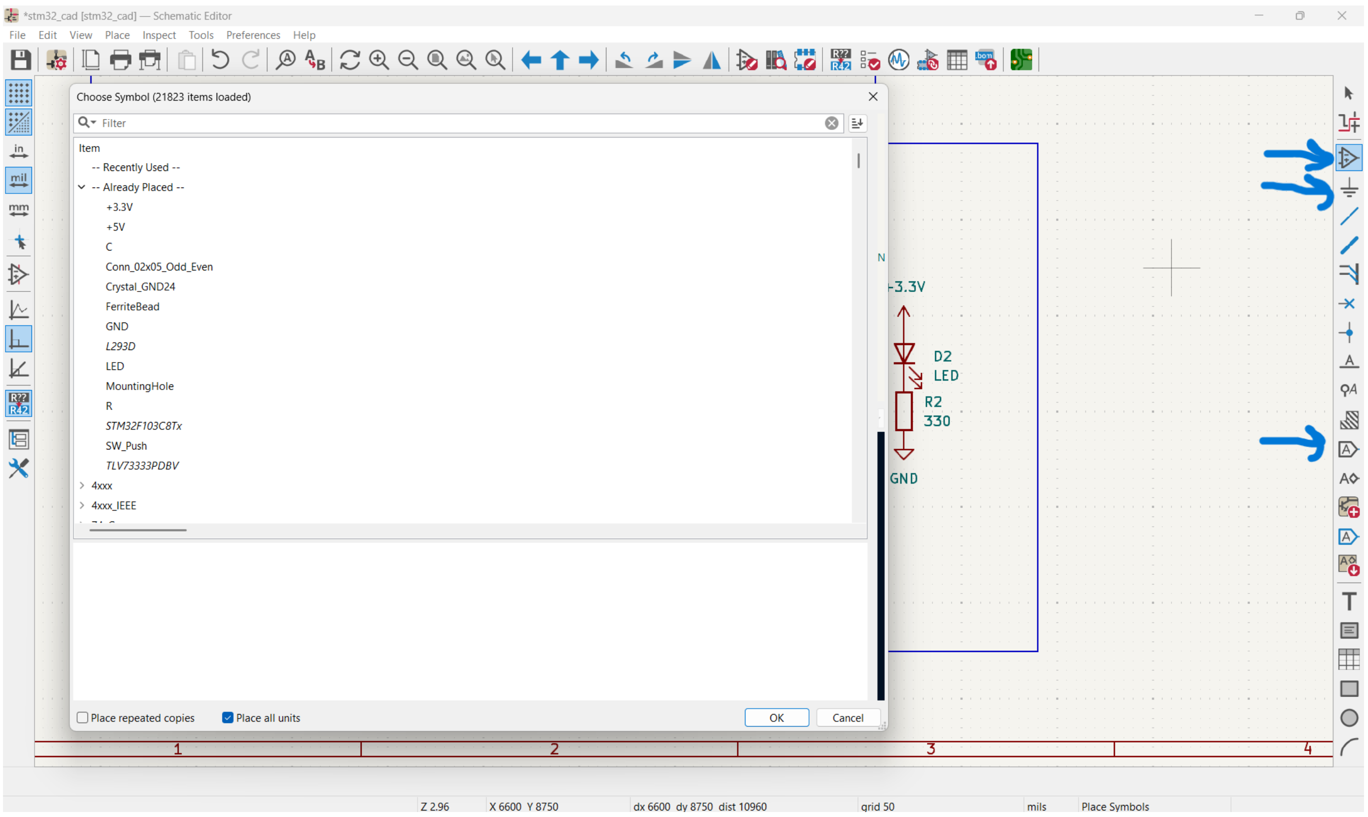
KiCAD Schematic

1. Download and open KiCAD 9.0
2. File -> New Project -> Select location and name it
3. Open the .kicad_sch file, this is the schematic
4. Go to File -> Page Settings -> Size: A3

The schematic is a way to visually represent connections in an electrical circuit. The "wires" or "nets" are labels that are attached to "pins" on "symbols"(components). Once you have instantiated all of the components and defined all of the "nets" connecting these components, you will assign physical models to the components and then do layout (physical arrangement).

Keyboard Shortcuts

- To draw a wire, press `W`
- To rotate a part, press `R`. This applies in layout and schematic.
- To change a value (Ohms, Farads, etc.), hover over the symbol and press `V`
- To place a symbol, either press `A` or click the Op-Amp button on the right panel (top arrow)
- To place a global label, either press `Ctrl+L` or global label button (bottom arrow)
- To place a power symbol press `P` or select the ground symbol button on the right panel (middle arrow)


*stm32_cad [stm32_cad] — Schematic Editor
File Edit View Place Inspect Tools Preferences Help
-3.3V
D2
LED
R2
330
GND
1
2
3
4
Z 2.96
X 6600 Y 8750
dx 6600 dy 8750 dist 10960
grid 50
mils
Place Symbols

Choose Symbol (21823 items loaded)

Filter

Item
-- Recently Used --
-- Already Placed --
+3.3V
+5V
C
Conn_02x05_Odd_Even
Crystal_GND24
FerriteBead
GND
L293D
LED
MountingHole
R
STM32F103C8Tx
SW_Push
TLV73333PDBV
4xxx
4xxx_IEEE

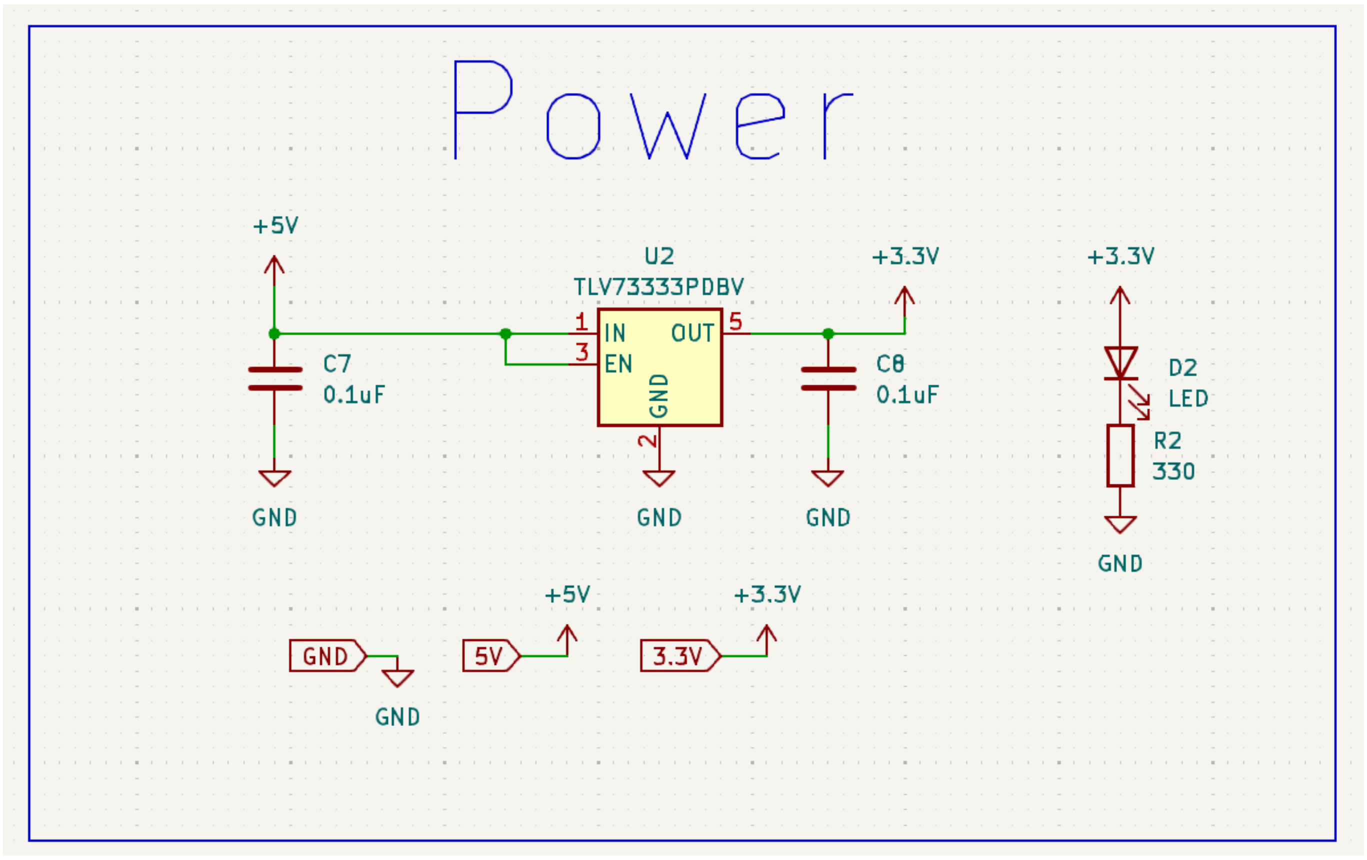
Place repeated copies [✓ Place all units]

OK Cancel

Power Subsystem

- We will be (primarily) powering the board via USB C cable when we are not programming the board.
- Place the following:
- 3.3V, 5V, and GND Power Symbols
- 3.3V, 5V, and GND Global Labels *We connect the power symbol and the labels together so we can use either throughout the design and have it mean the same thing*
- 3.3V Low Dropout Regulator: `TLV73333PDBV`
- 2x Generic Capacitor: `C`
- Generic Resistor: `R`
- Generic LED: `LED`
- Double click on the capacitor and set value to `0.1uF`
- Double click on the resistor and set value to `3.3V`

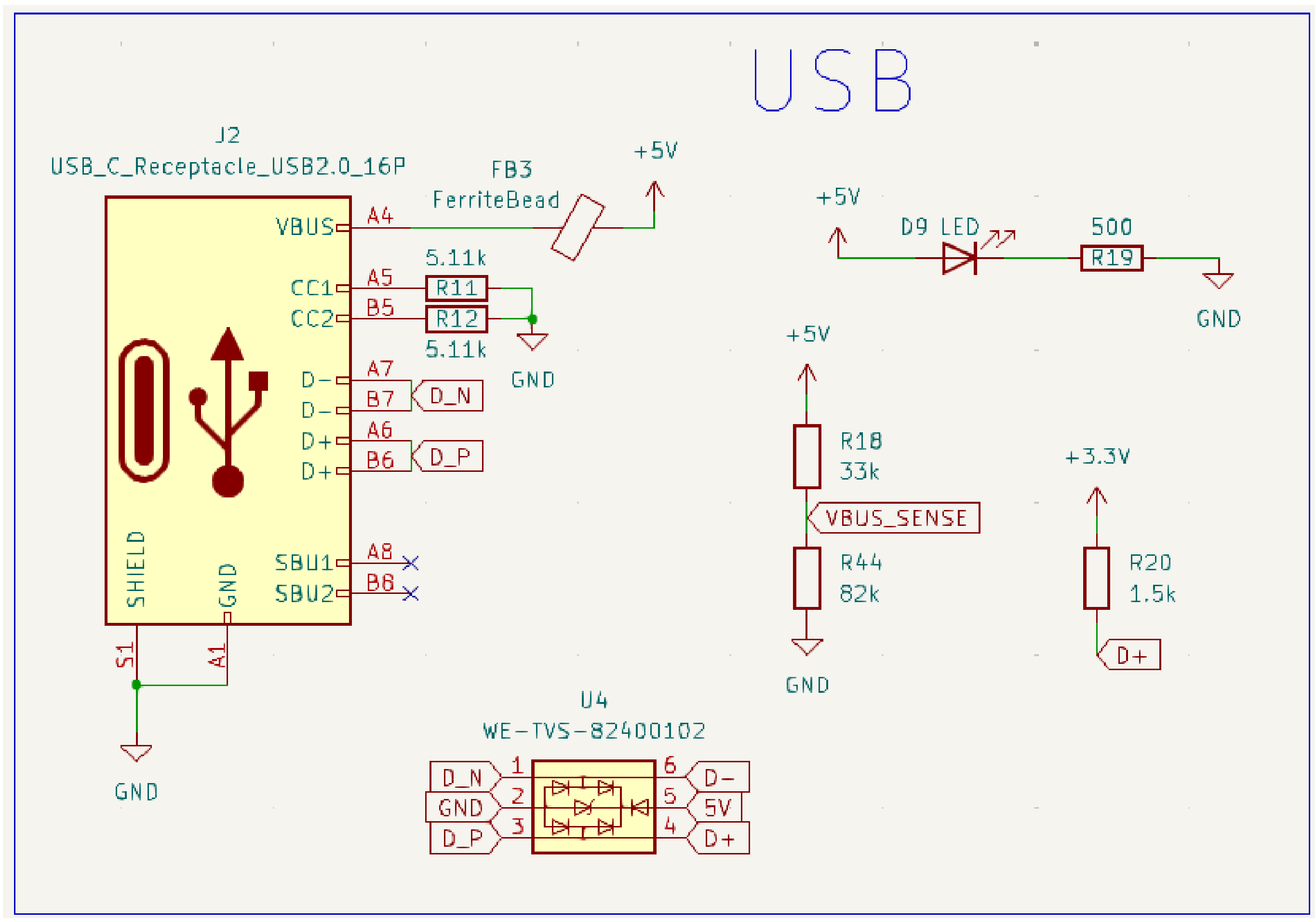
Make sure the values for passive components and labels reflect the below image.



USB Subsystem

- We will be using a USB C connector for power and also for Virtual COM Port serial communication. The ESD protection diode is used to protect the MCU from charge buildup from inserting and removing the USB cable. The ferrite bead is to filter electrical noise on the USB cable. The 1.5k Ohm resistor is used to pull up the D+ line, which allows for differential signaling, which is foundational to USB communication.
- Place the following:
- 3.3V, 5V, and GND Power Symbols
- 3.3V, 5V, and GND Global Labels
- USB-C Port: `USB_C_Receptacle_USB2.0_16P`
- Electrostatic Discharge Diode: `WE-TVS-82400102`
- 2x Generic Capacitors: `C`
- Generic Resistor: `R`
- Generic LED: `LED`
- Generic Ferrite Bead: `FerriteBead`

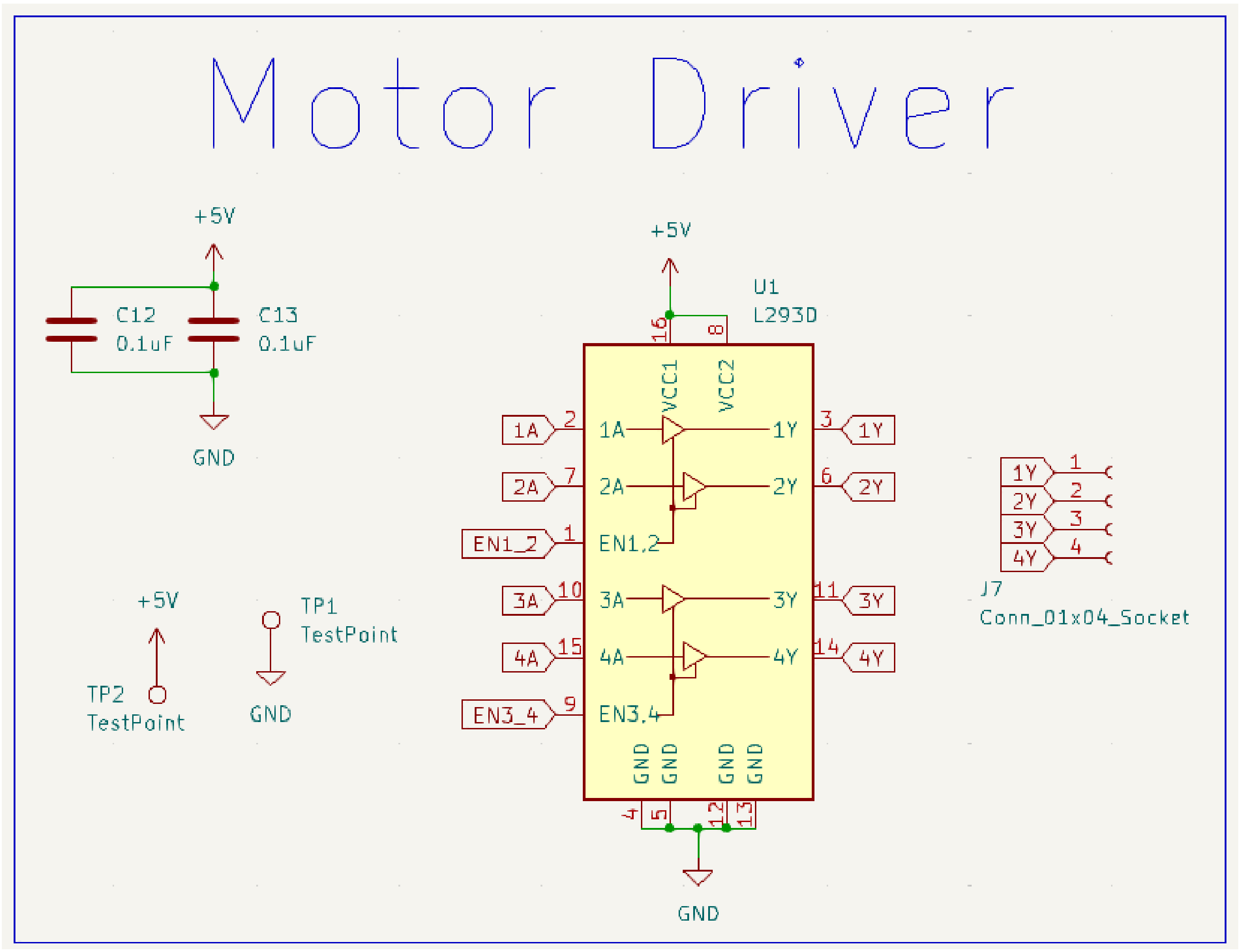
Make sure the values for passive components and labels reflect the below image.



Motor Driver Subsystem

- We will be using a [Quadruple Half-H Driver] (https://www.ti.com/lit/ds/symlink/l293d.pdf?ts=1749559495468&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FL293D) to create up to 2 H-bridges. One H bridge is capable of spinning a brushed DC motor or controlling a single phase of a multi-phase motor (stepper, brushless, etc.). For higher current requirements, you can solder on larger gauge wires to the test pads.
- Place the following:
- 5V and GND Power Symbols
- L293D IC: `L293D`
- 2x Generic Capacitors: `C`
- 2x Generic Test Points: `TestPoint`

Make sure the values for passive components and labels reflect the below image.

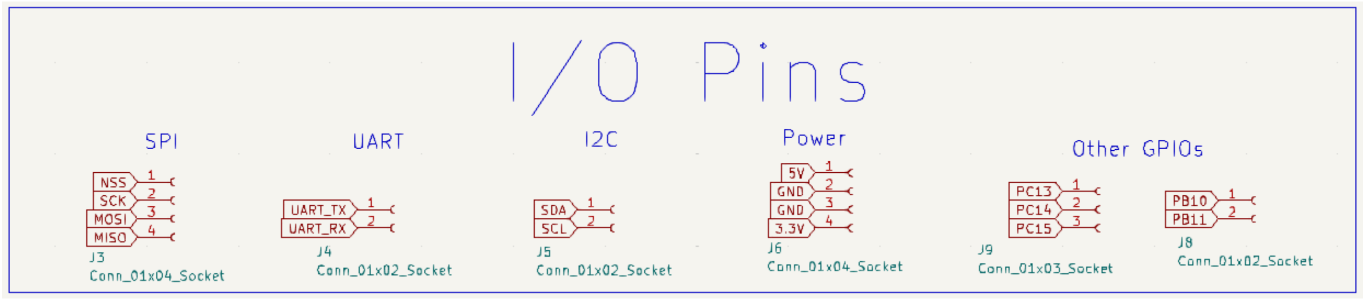


Input/Output Pins

We will be using socket connectors to allow communication, power, and general purpose connections to external components.

- Place the following:
- 3.3V, 5V, and GND Global Labels
- 3x Socket 1x2 Connectors: `Conn_01x02_Socket`
- 1x Socket 1x3 Connector: `Conn_01x03_Socket`
- 2x Socket 1x4 Connectors: `Conn_01x04_Socket`

Make sure the labels reflect the below image.



STM32 Microcontroller (MCU)

For this board, we will be using an STM32F103C8T6 MCU. This specific model and family of MCU is chosen for its industry level design tools and capabilities. As seen in the microcontroller setup in STM32CUBEIDE, there is a lot of configurability for this device.

To program the device, we will be using an STLink, a proprietary programmer which uses Serial Wire Debug to flash code onto the MCU.

We be placing the crystal oscillator our MCU configuration requires.

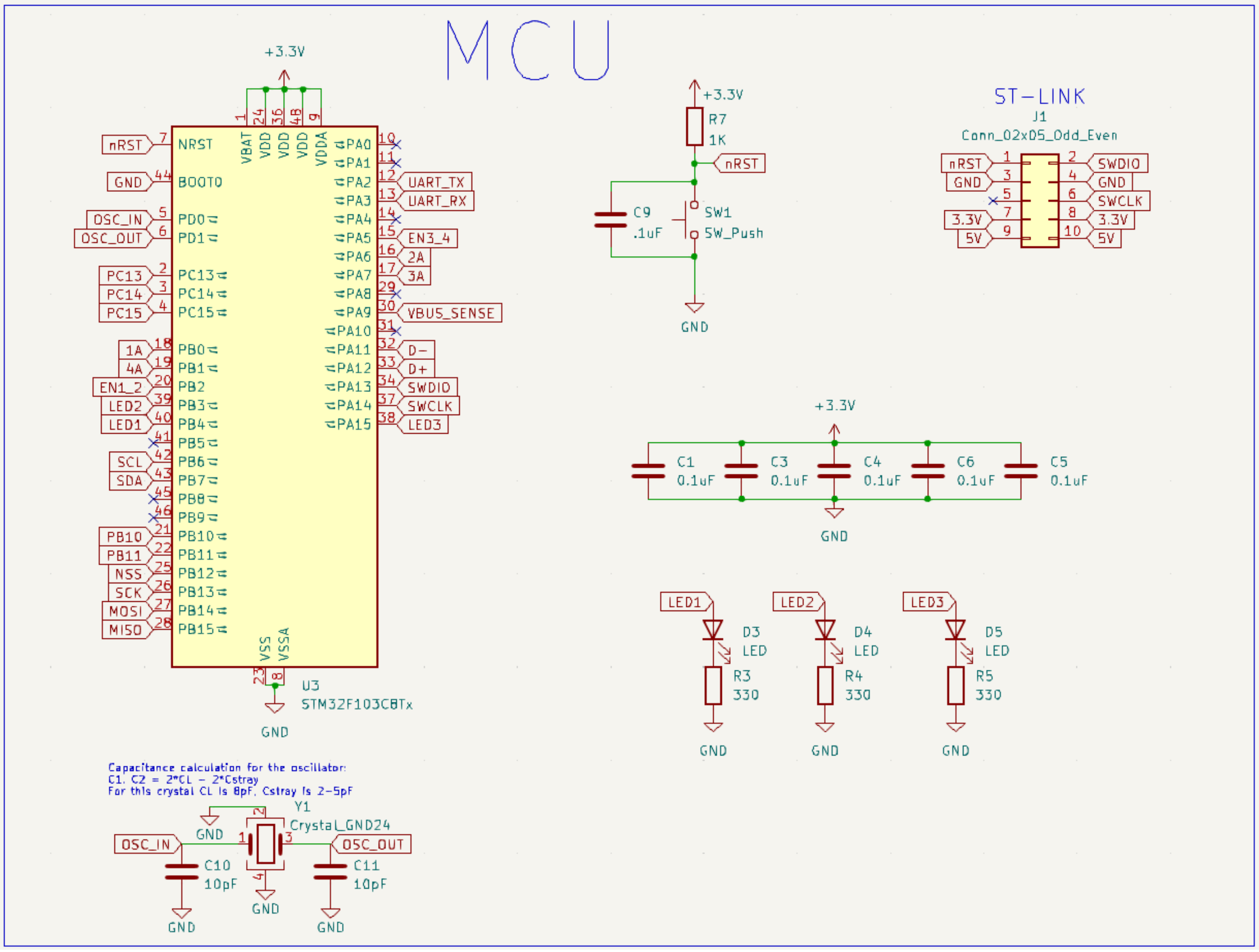
We require decoupling capacitors to filter out noise on the MCU's power lines.

We will place a physcial reset button to control the active low nRST pin.

We will finally place 3 LEDs for debugging or displaying current states of the MCU.

- Place the following:
- 3.3V, 5V, and GND Global Labels
- STM32F103C8T6 MCU: `STM32F103C8Tx`
- 8 MHz Crystal Oscillator: `Crystal_GND24`
- Socket 2x5 Connector: `Conn_02x05_Odd_Even`
- 8x Generic Capacitors: `C`
- 4x Generic Resistors: `R`
- 3x Generic LEDs: `LED`
- Generic Switch: `SW_Push`

Make sure the values for passive components and labels reflect the below image.



Mounting Holes

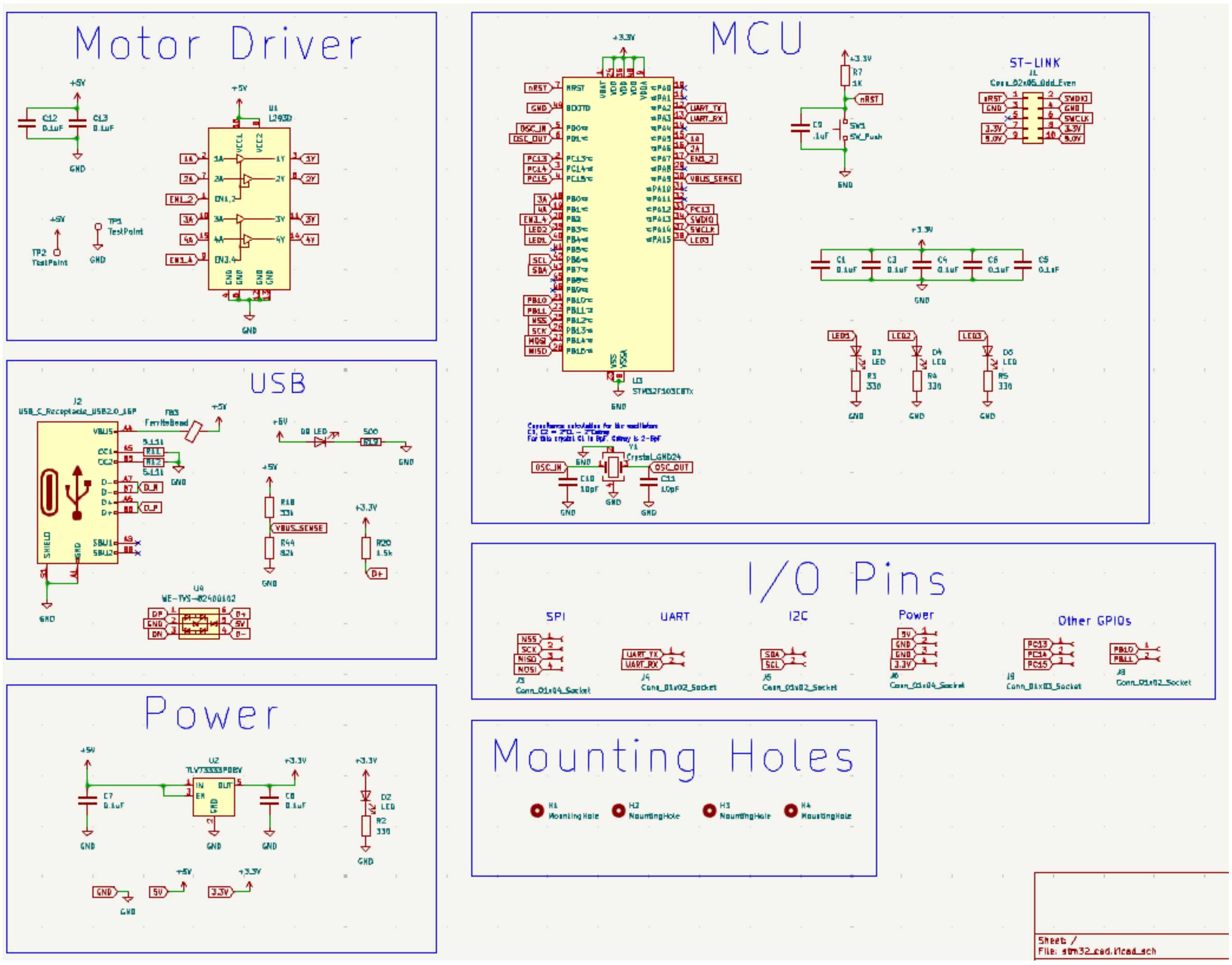
These will allow us to attach this board to an enclosure

Place:

- 4x Generic Mounting Holes: `MountingHole`

Complete Schematic

The schematic should now look like this:

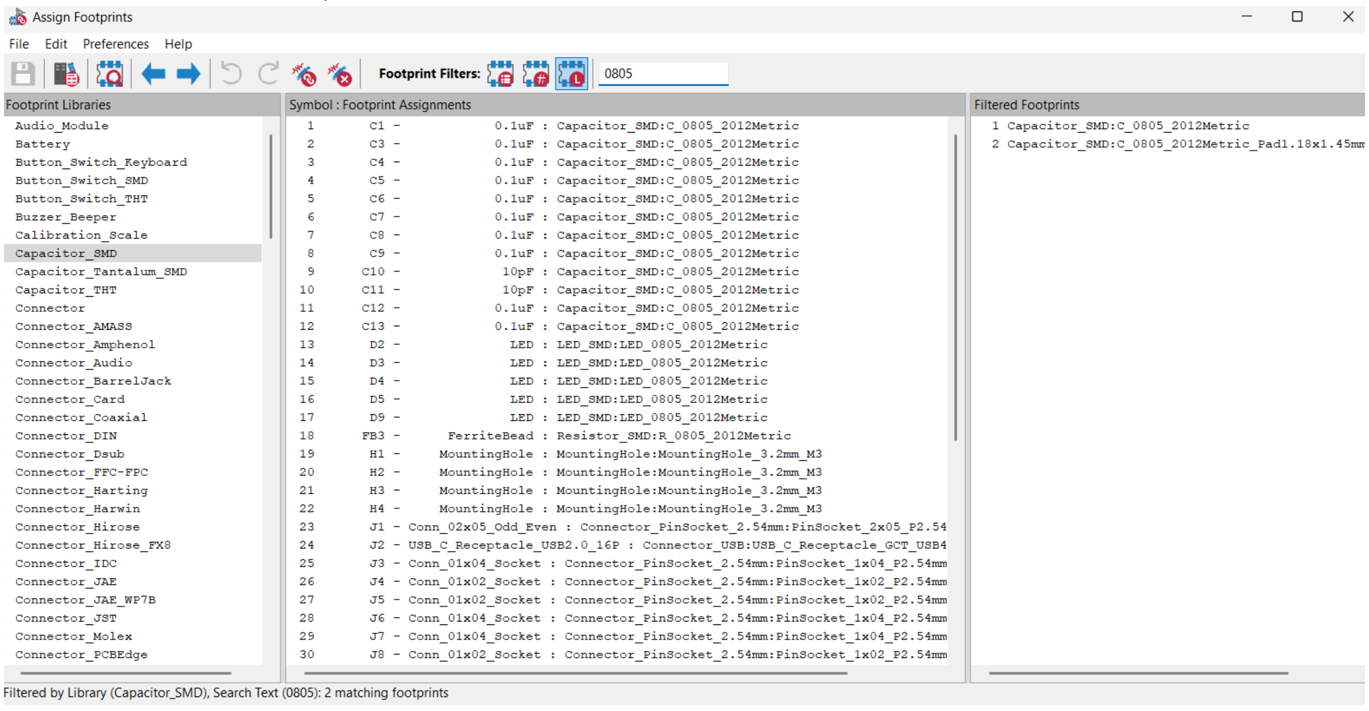


Assign Footprints

Now we have described our board electrically (nets). The software now knows abstractly what pins should be connected on which components. We now have to describe what each of the components will be physically, these descriptions are called footprints. Click on "Assign Footprints" on the top panel.

To assign your first footprint (capacitors for example), select the Capacitor_SMD library on the left pane. Next to "Footprint Filters", make sure the "L" icon is selected to search within the selected library and type 0805.

This is the size of the component. **0805 is 1.6 mm x 0.8 mm**



Footprint Libraries

```
Audio_Module
Battery
Button_Switch_Keyboard
Button_Switch_SMD
Button_Switch_THT
Buzzer_Beeper
Calibration_Scale
Capacitor_SMD
Capacitor_Tantalum_SMD
Capacitor_THT
Connector
Connector_AMASS
Connector_Amphenol
Connector_Audio
Connector_BarrelJack
Connector_Card
Connector_Coaxial
Connector_DIN
Connector_Dsub
Connector_FFC-FPC
Connector_Harting
Connector_Harwin
Connector_Hirose
Connector_Hirose_FX8
Connector_IDC
Connector_JAE
Connector_JAE_WP7B
Connector_JST
Connector_Molex
Connector_PCBEdge
```

Symbol : Footprint Assignments

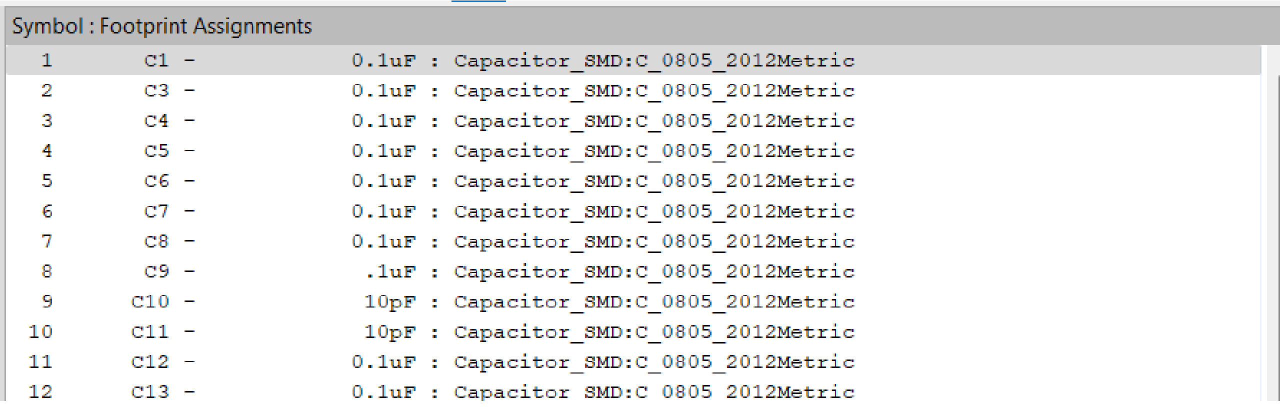
```
 1        C1 -             0.1uF : Capacitor_SMD:C_0805_2012Metric
 2        C3 -             0.1uF : Capacitor_SMD:C_0805_2012Metric
 3        C4 -             0.1uF : Capacitor_SMD:C_0805_2012Metric
 4        C5 -             0.1uF : Capacitor_SMD:C_0805_2012Metric
 5        C6 -             0.1uF : Capacitor_SMD:C_0805_2012Metric
 6        C7 -             0.1uF : Capacitor_SMD:C_0805_2012Metric
 7        C8 -             0.1uF : Capacitor_SMD:C_0805_2012Metric
 8        C9 -             0.1uF : Capacitor_SMD:C_0805_2012Metric
 9       C10 -              10pF : Capacitor_SMD:C_0805_2012Metric
10       C11 -              10pF : Capacitor_SMD:C_0805_2012Metric
11       C12 -             0.1uF : Capacitor_SMD:C_0805_2012Metric
12       C13 -             0.1uF : Capacitor_SMD:C_0805_2012Metric
13        D2 -               LED : LED_SMD:LED_0805_2012Metric
14        D3 -               LED : LED_SMD:LED_0805_2012Metric
15        D4 -               LED : LED_SMD:LED_0805_2012Metric
16        D5 -               LED : LED_SMD:LED_0805_2012Metric
17        D9 -               LED : LED_SMD:LED_0805_2012Metric
18       FB3 -       FerriteBead : Resistor_SMD:R_0805_2012Metric
19        H1 -      MountingHole : MountingHole:MountingHole_3.2mm_M3
20        H2 -      MountingHole : MountingHole:MountingHole_3.2mm_M3
21        H3 -      MountingHole : MountingHole:MountingHole_3.2mm_M3
22        H4 -      MountingHole : MountingHole:MountingHole_3.2mm_M3
23        J1 - Conn_02x05_Odd_Even : Connector_PinSocket_2.54mm:PinSocket_2x05_P2.54
24        J2 - USB_C_Receptacle_USB2.0_16P : Connector_USB:USB_C_Receptacle_GCT_USB4
25        J3 - Conn_01x04_Socket : Connector_PinSocket_2.54mm:PinSocket_1x04_P2.54mm
26        J4 - Conn_01x02_Socket : Connector_PinSocket_2.54mm:PinSocket_1x02_P2.54mm
27        J5 - Conn_01x02_Socket : Connector_PinSocket_2.54mm:PinSocket_1x02_P2.54mm
28        J6 - Conn_01x04_Socket : Connector_PinSocket_2.54mm:PinSocket_1x04_P2.54mm
29        J7 - Conn_01x04_Socket : Connector_PinSocket_2.54mm:PinSocket_1x04_P2.54mm
30        J8 - Conn_01x02_Socket : Connector_PinSocket_2.54mm:PinSocket_1x02_P2.54mm
```

Filtered Footprints

```
1 Capacitor_SMD:C_0805_2012Metric
2 Capacitor_SMD:C_0805_2012Metric_Pad1.18x1.45mm
```

Filtered by Library (Capacitor_SMD), Search Text (0805): 2 matching footprints

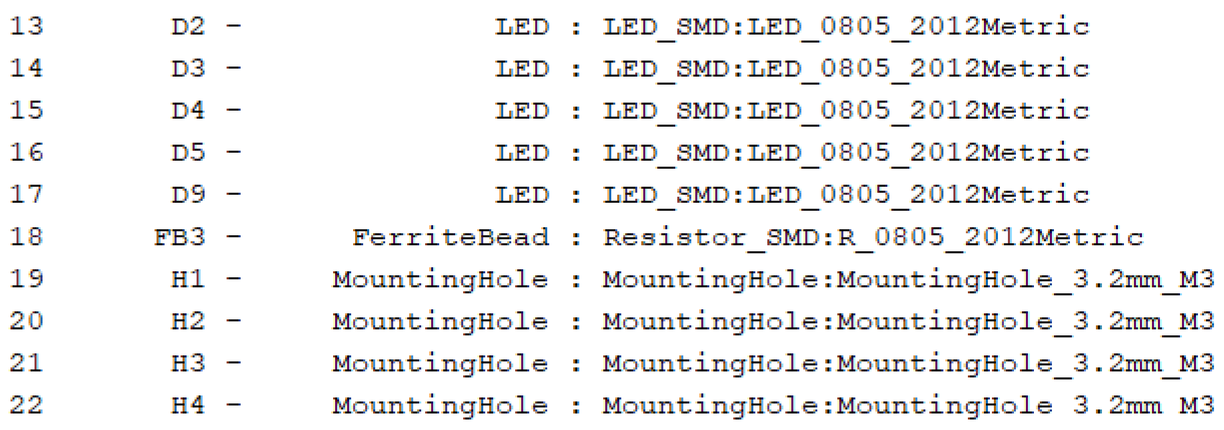
- All capacitors should be `Capacitor_SMD:C_0805_2012Metric`



Symbol : Footprint Assignments

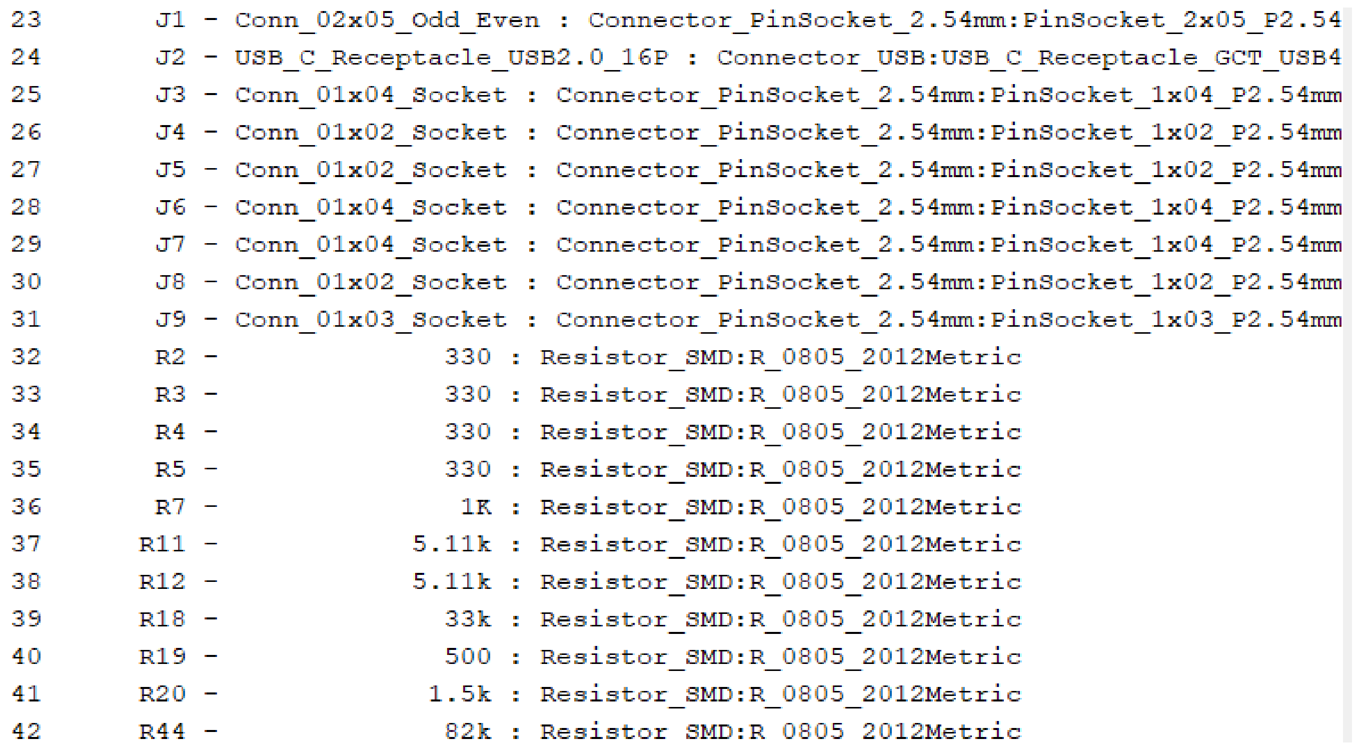
```
 1        C1 -             0.1uF : Capacitor_SMD:C_0805_2012Metric
 2        C3 -             0.1uF : Capacitor_SMD:C_0805_2012Metric
 3        C4 -             0.1uF : Capacitor_SMD:C_0805_2012Metric
 4        C5 -             0.1uF : Capacitor_SMD:C_0805_2012Metric
 5        C6 -             0.1uF : Capacitor_SMD:C_0805_2012Metric
 6        C7 -             0.1uF : Capacitor_SMD:C_0805_2012Metric
 7        C8 -             0.1uF : Capacitor_SMD:C_0805_2012Metric
 8        C9 -              .1uF : Capacitor_SMD:C_0805_2012Metric
 9       C10 -              10pF : Capacitor_SMD:C_0805_2012Metric
10       C11 -              10pF : Capacitor_SMD:C_0805_2012Metric
11       C12 -             0.1uF : Capacitor_SMD:C_0805_2012Metric
12       C13 -             0.1uF : Capacitor_SMD:C_0805_2012Metric
```

- All LEDs should be `LED_SMD:LED_0805_2012Metric`
- The ferrite bead should be `Resistor_SMD:R_0805_2012Metric` (This is physically accurate, but a ferrite bead not is not functionally the same as a resistor)
- The mounting holes should be `MountingHole:MountingHole_3.2mm_M3`



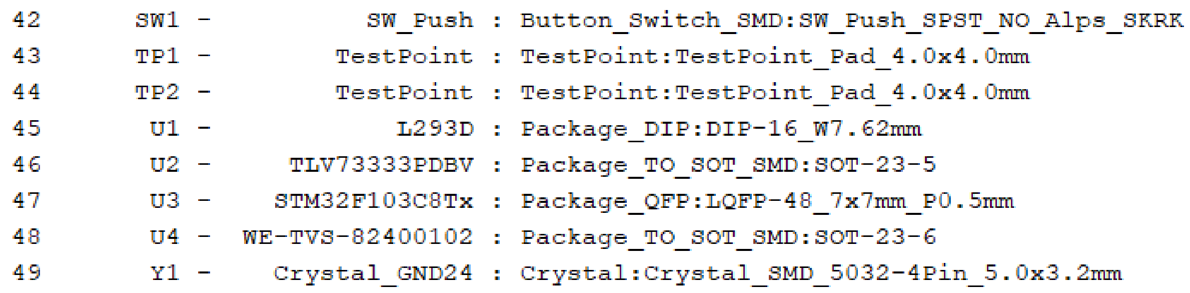
```
13        D2 -               LED : LED_SMD:LED_0805_2012Metric
14        D3 -               LED : LED_SMD:LED_0805_2012Metric
15        D4 -               LED : LED_SMD:LED_0805_2012Metric
16        D5 -               LED : LED_SMD:LED_0805_2012Metric
17        D9 -               LED : LED_SMD:LED_0805_2012Metric
18       FB3 -       FerriteBead : Resistor_SMD:R_0805_2012Metric
19        H1 -      MountingHole : MountingHole:MountingHole_3.2mm_M3
20        H2 -      MountingHole : MountingHole:MountingHole_3.2mm_M3
21        H3 -      MountingHole : MountingHole:MountingHole_3.2mm_M3
22        H4 -      MountingHole : MountingHole:MountingHole_3.2mm_M3
```

- The 02x05 connector should be `Connector_PinSocket_2.54mm:PinSocket_2x05_P2.54mm_Vertical`
- The USB connector should be `Connector_USB:USB_C_Receptacle_GCT_USB4085`
- The 01x04 connectors should be `Connector_PinSocket_2.54mm:PinSocket_1x04_P2.54mm_Vertical`
- The 01x03 connector should be `Connector_PinSocket_2.54mm:PinSocket_1x03_P2.54mm_Vertical`
- The 01x02 connectors should be `Connector_PinSocket_2.54mm:PinSocket_1x02_P2.54mm_Vertical`
- All resistors should be `Resistor_SMD:R_0805_2012Metric`



```
      J1 - Conn_02x05_Odd_Even : Connector_PinSocket_2.54mm:PinSocket_2x05_P2.54
      J2 - USB_C_Receptacle_USB2.0_16P : Connector_USB:USB_C_Receptacle_GCT_USB4
      J3 - Conn_01x04_Socket : Connector_PinSocket_2.54mm:PinSocket_1x04_P2.54mm
      J4 - Conn_01x02_Socket : Connector_PinSocket_2.54mm:PinSocket_1x02_P2.54mm
      J5 - Conn_01x02_Socket : Connector_PinSocket_2.54mm:PinSocket_1x02_P2.54mm
      J6 - Conn_01x04_Socket : Connector_PinSocket_2.54mm:PinSocket_1x04_P2.54mm
      J7 - Conn_01x04_Socket : Connector_PinSocket_2.54mm:PinSocket_1x04_P2.54mm
      J8 - Conn_01x02_Socket : Connector_PinSocket_2.54mm:PinSocket_1x02_P2.54mm
      J9 - Conn_01x03_Socket : Connector_PinSocket_2.54mm:PinSocket_1x03_P2.54mm
      R2 -             330 : Resistor_SMD:R_0805_2012Metric
      R3 -             330 : Resistor_SMD:R_0805_2012Metric
      R4 -             330 : Resistor_SMD:R_0805_2012Metric
      R5 -             330 : Resistor_SMD:R_0805_2012Metric
      R7 -              1K : Resistor_SMD:R_0805_2012Metric
     R11 -           5.11k : Resistor_SMD:R_0805_2012Metric
     R12 -           5.11k : Resistor_SMD:R_0805_2012Metric
     R18 -             33k : Resistor_SMD:R_0805_2012Metric
     R19 -             500 : Resistor_SMD:R_0805_2012Metric
     R20 -            1.5k : Resistor_SMD:R_0805_2012Metric
     R44 -             82k : Resistor_SMD:R_0805_2012Metric
```

- The button should be `Button_Switch_SMD:SW_Push_SPST_NO_Alps_SKRK`
- The test points should be `TestPoint:TestPoint_Pad_4.0x4.0mm`
- The crystal should be `Crystal:Crystal_SMD_5032-4Pin_5.0x3.2mm`
- The other footprints should auto populate, if not, use the footprints shown below.

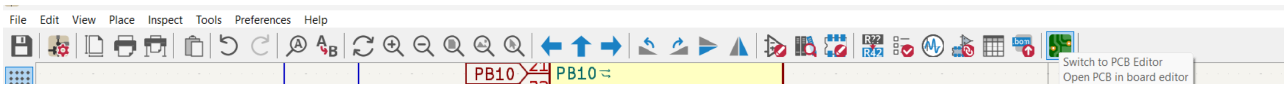


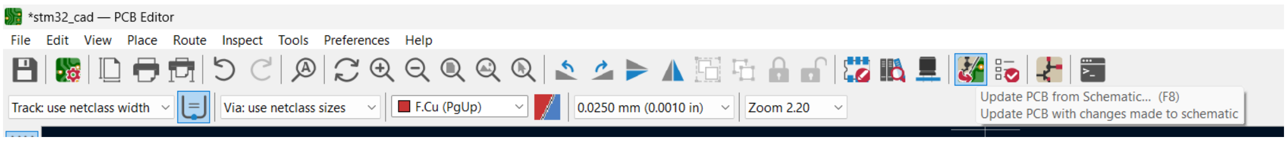
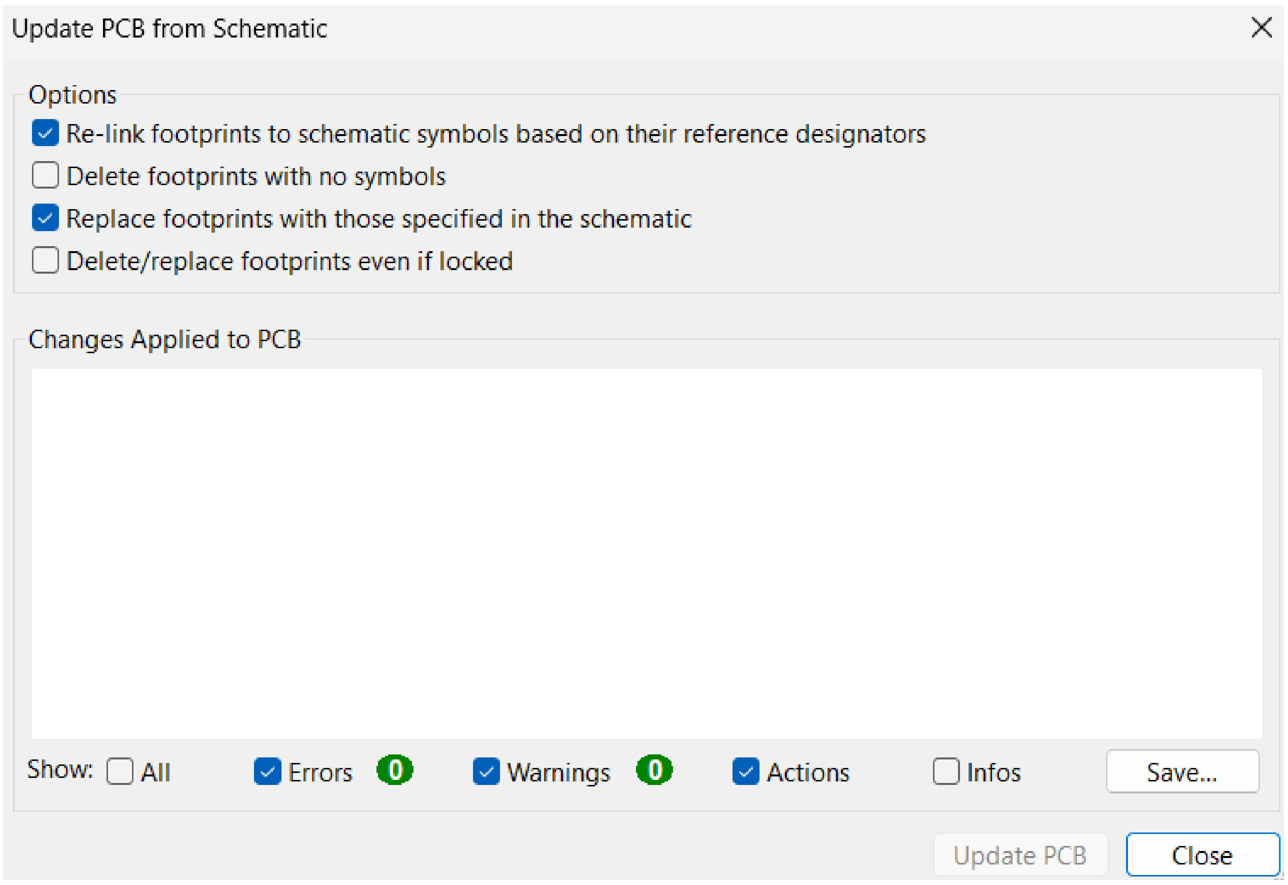
```
     SW1 -         SW_Push : Button_Switch_SMD:SW_Push_SPST_NO_Alps_SKRK
     TP1 -       TestPoint : TestPoint:TestPoint_Pad_4.0x4.0mm
     TP2 -       TestPoint : TestPoint:TestPoint_Pad_4.0x4.0mm
      U1 -          L293D : Package_DIP:DIP-16_W7.62mm
      U2 -    TLV73333PDBV : Package_TO_SOT_SMD:SOT-23-5
      U3 -   STM32F103C8Tx : Package_QFP:LQFP-48_7x7mm_P0.5mm
      U4 - WE-TVS-82400102 : Package_TO_SOT_SMD:SOT-23-6
      Y1 -    Crystal_GND24 : Crystal:Crystal_SMD_5032-4Pin_5.0x3.2mm
```

Click Apply, Save Schematic & Continue. Then click OK

KiCAD Layout

We are now on layout, which is where we decide how large out board will be and where to place our components (now footprints). We will also connect all of our nets using traces. Traces can vary in width and we will have one example of that. Traces also sometimes need to be impedance matched (at high frequencies) or routed as a differential pair, we will do the latter as an example, but it is not completely necessary for our applicaiton.

1. Click "Switch to PCB Editor" on the top panel


2. Click "Update PCB from Schematic"

*stm32_cad — PCB Editor
File Edit View Place Route Inspect Tools Preferences Help
Track: use netclass width
Via: use netclass sizes
F.Cu (PgUp)
0.0250 mm (0.0010 in)
Zoom 2.20
Update PCB from Schematic... (F8)
Update PCB with changes made to schematic
3. Click "Update PCB"


Update PCB from Schematic

Options

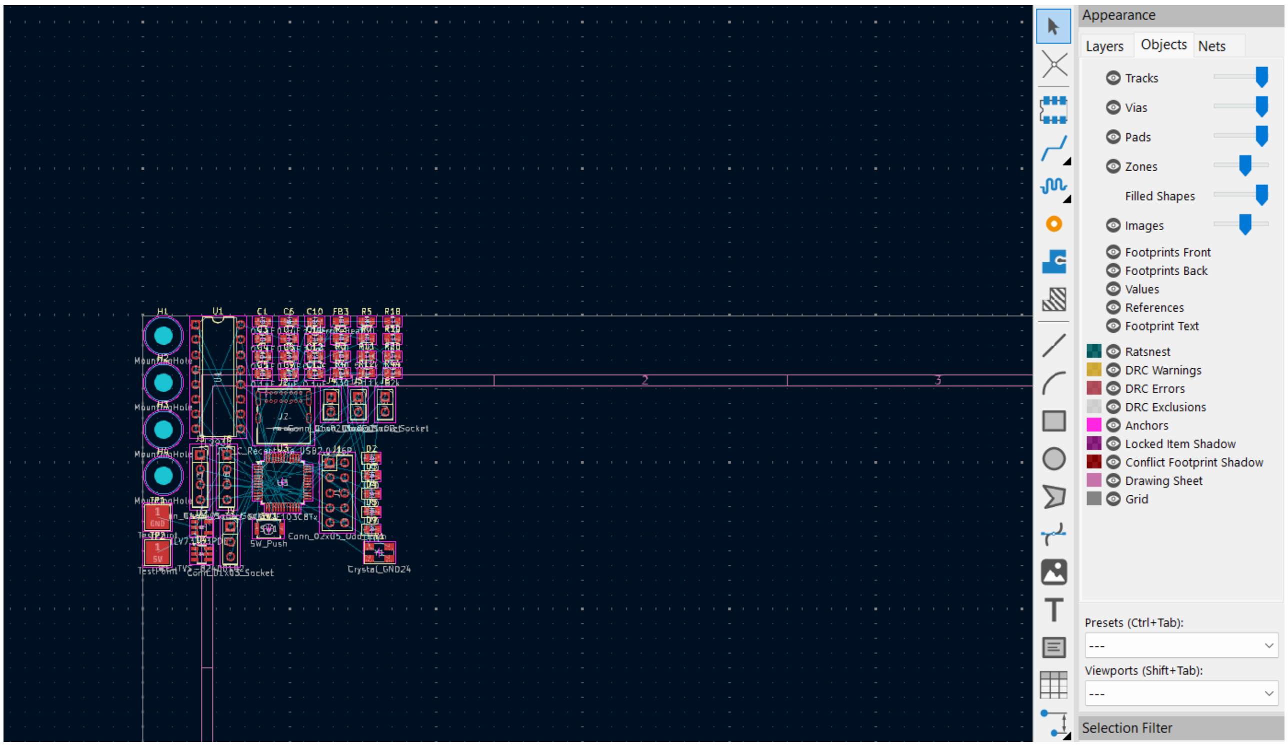
- [x] Re-link footprints to schematic symbols based on their reference designators
- [ ] Delete footprints with no symbols
- [x] Replace footprints with those specified in the schematic
- [ ] Delete/replace footprints even if locked

Changes Applied to PCB

Show: [ ] All [x] Errors 0 [x] Warnings 0 [x] Actions [ ] Infos Save...

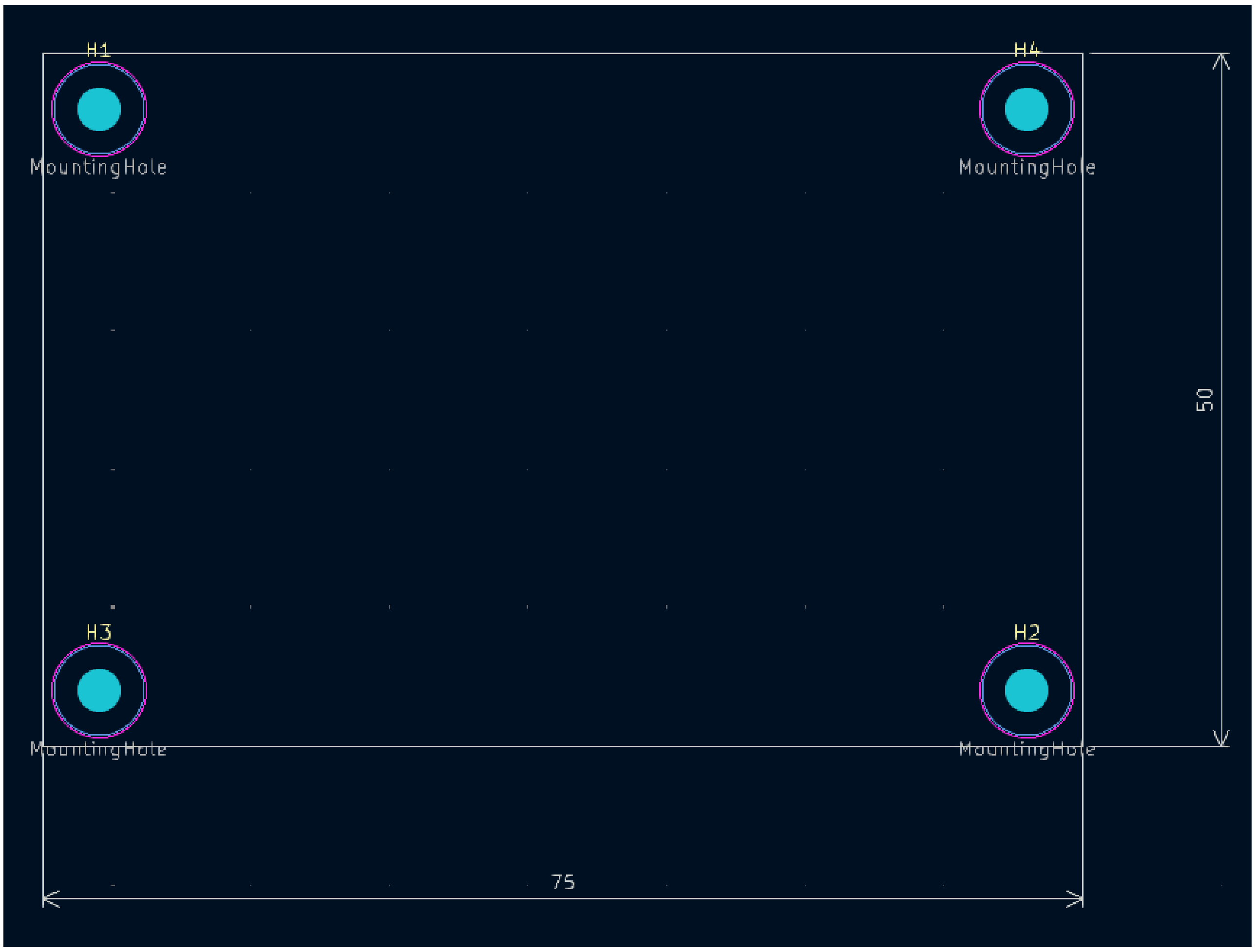
Update PCB Close

4. Click anywhere to drop the components. If you cannot see them, make sure your `Appearances` -> `Objects` settings look like the image below



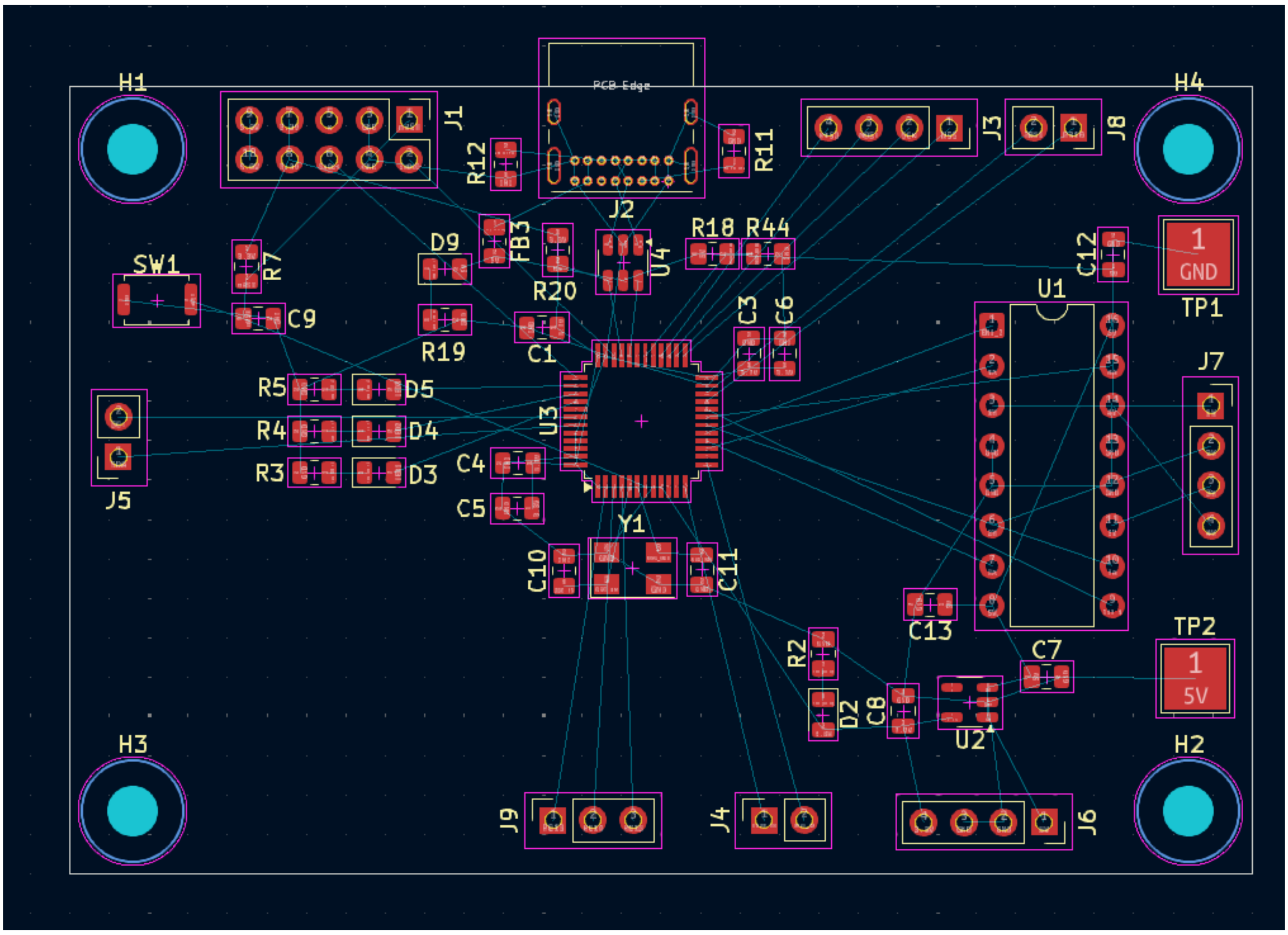
Defining the board size

1. `Appearance` -> `Layers` -> `Edge.Cuts`
2. Change the grid size at the top to 5.00mm
3. Select the rectangle tool and draw a 50mm x 75mm rectangle
4. Change the grid size to 1mm and place the mounting holes at the 4 corners of the board like in the image below.



Arranging components

Change the grid to whatever you want and arrange the rest of the components similar to shown below. You can turn off F.Fab and B.Fab to decrease cluttered text in the Appearance panel.



Keep these considerations in mind

1. Minimize signal (USB, UART, SPI, SWD, I2C) trace lengths
2. Minimize distance between crystal and MCU
3. Have at least 1 capacitor at each power source for MCU, LDO, and L293D
4. Keep MCU away from noisy power

Keep components grouped together similar to the schematic view can simplify your layout process

- *Hint: One way to do this is to click and drag to highlight a section in the schematic view, which will select it in the layout view also, then arrange groups of components and place them.*

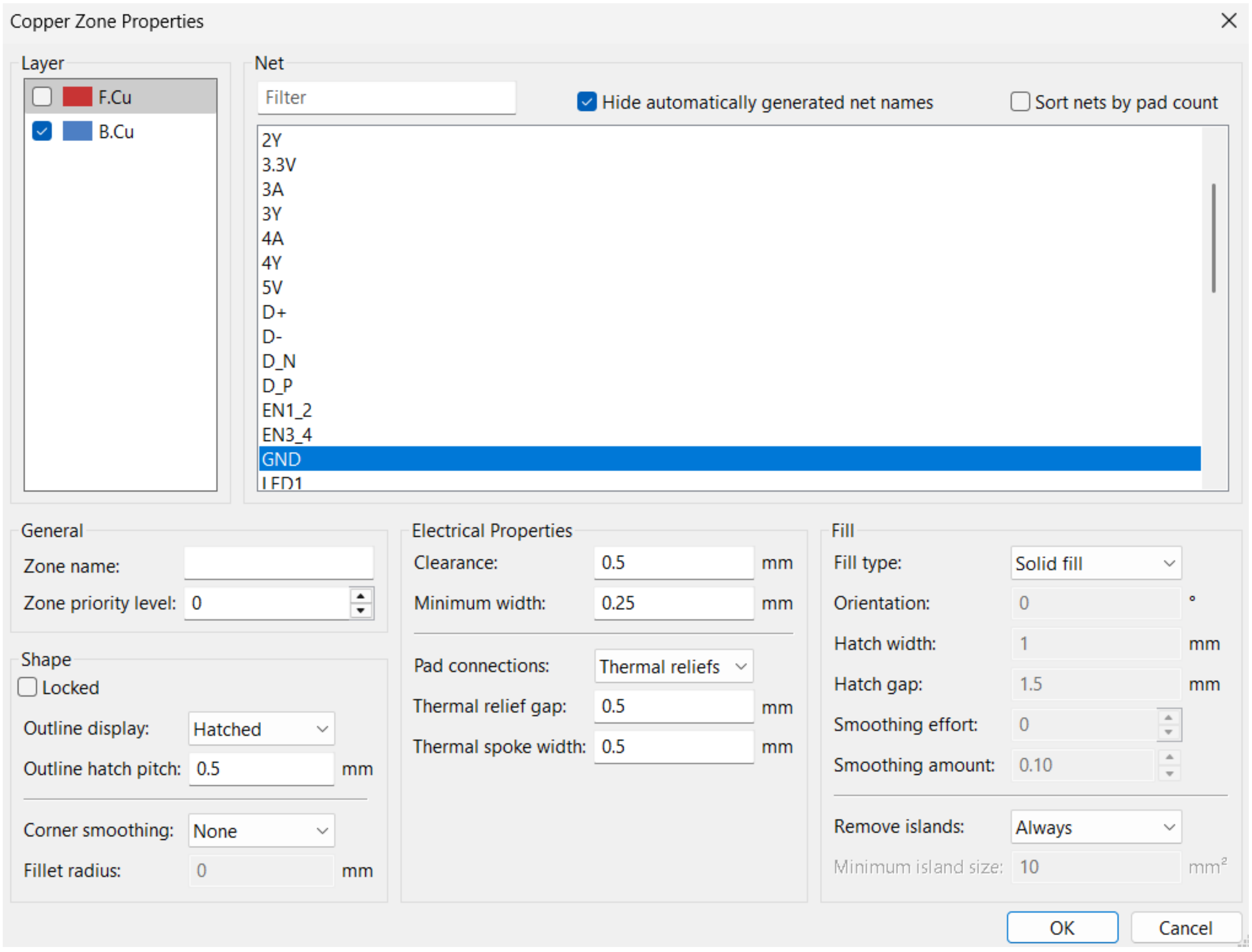
Drawing traces

All of the thin blue lines between device pins are called airwires. You will have to draw metal traces connecting the device pins until none remain.

There is one way around this which is using vias. Vias are holes drilled into the board which are then plated with copper or filled with conductive material. We will be placing a ground plane on the bottom of the board, meaning if you place a via on the top side, it will be effectively be connected to ground. Use this to your advantage.

Set the trace width to 0.25mm for all signal traces and 1mm for all power traces (3.3V, 5V) not going to the MCU. 0.5mm is fine for power traces going to the MCU.

Ground plane: `Ctrl + Shift + Z` or "Draw filled zone" on right panel. Configure it as shown below.



Layer: ☐ F.Cu, ☑ B.Cu

☑ Hide automatically generated net names ☐ Sort nets by pad count

Net list: 2Y, 3.3V, 3A, 3Y, 4A, 4Y, 5V, D+, D-, D_N, D_P, EN1_2, EN3_4, GND (selected), LED1

General

Zone name:	
Zone priority level:	0

Shape

☐ Locked

Outline display:	Hatched	
Outline hatch pitch:	0.5	mm
Corner smoothing:	None	
Fillet radius:	0	mm

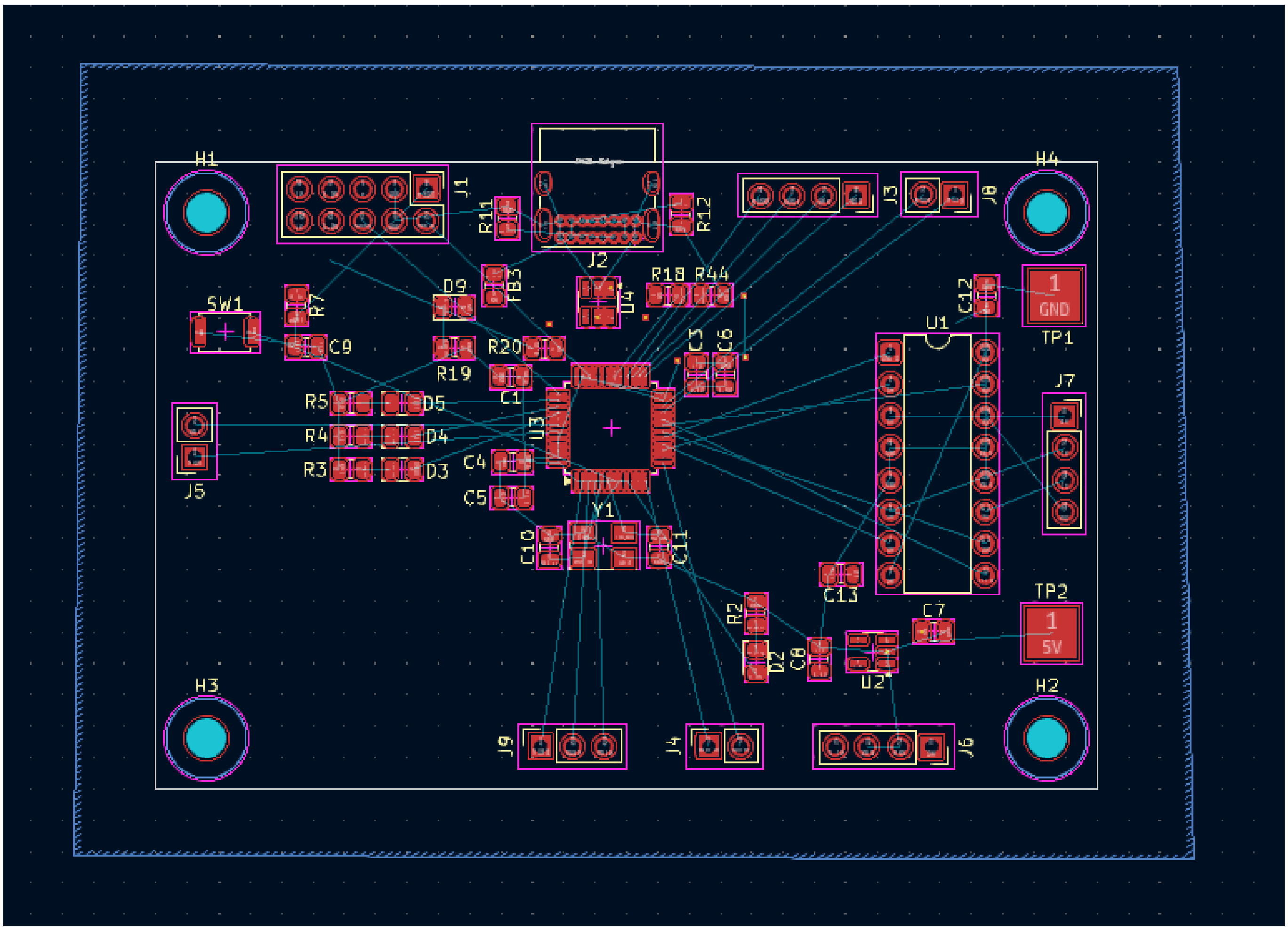
Electrical Properties

Clearance:	0.5	mm
Minimum width:	0.25	mm
Pad connections:	Thermal reliefs	
Thermal relief gap:	0.5	mm
Thermal spoke width:	0.5	mm

Fill

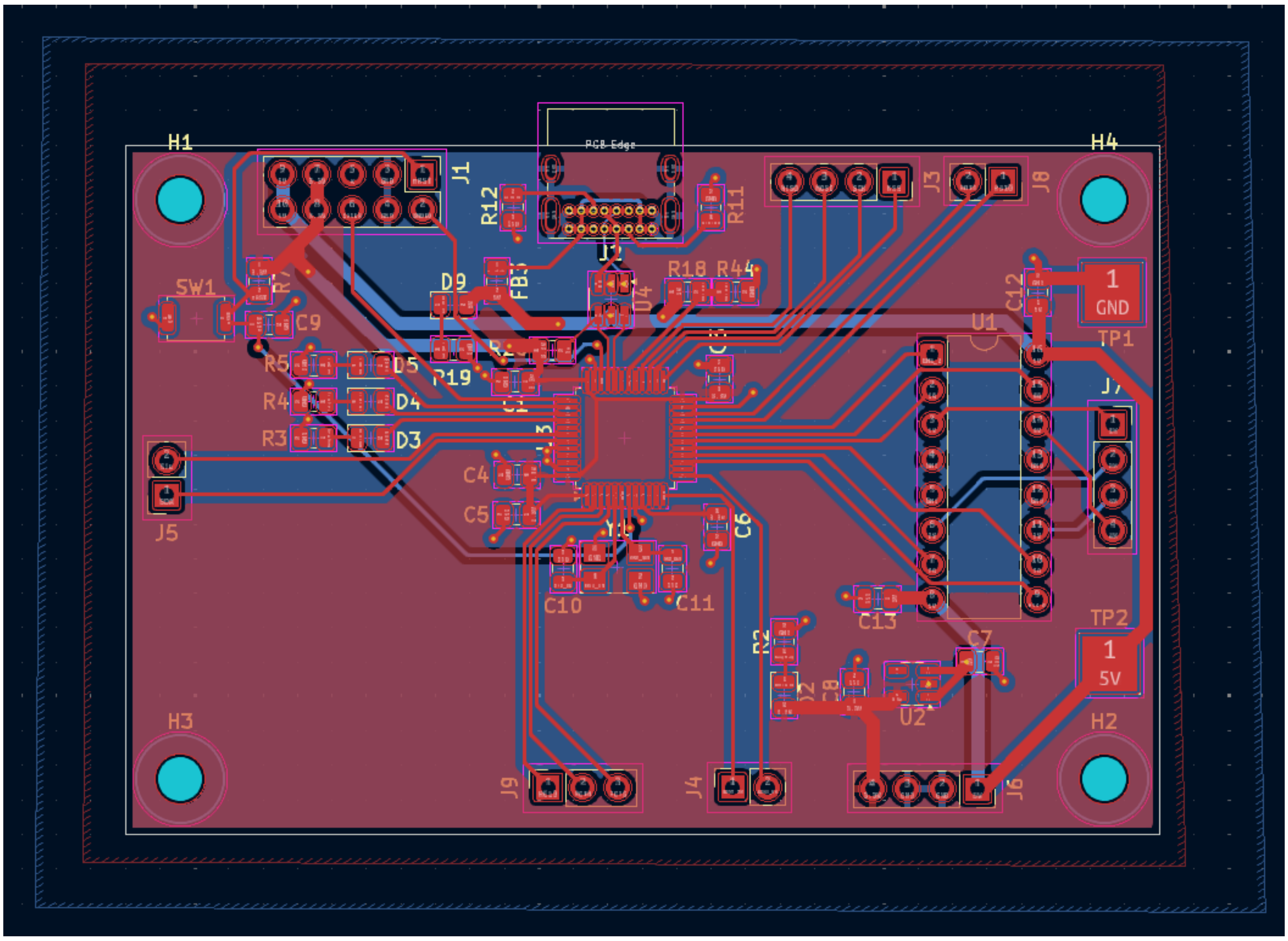
Fill type:	Solid fill	
Orientation:	0	°
Hatch width:	1	mm
Hatch gap:	1.5	mm
Smoothing effort:	0	
Smoothing amount:	0.10	
Remove islands:	Always	
Minimum island size:	10	mm²

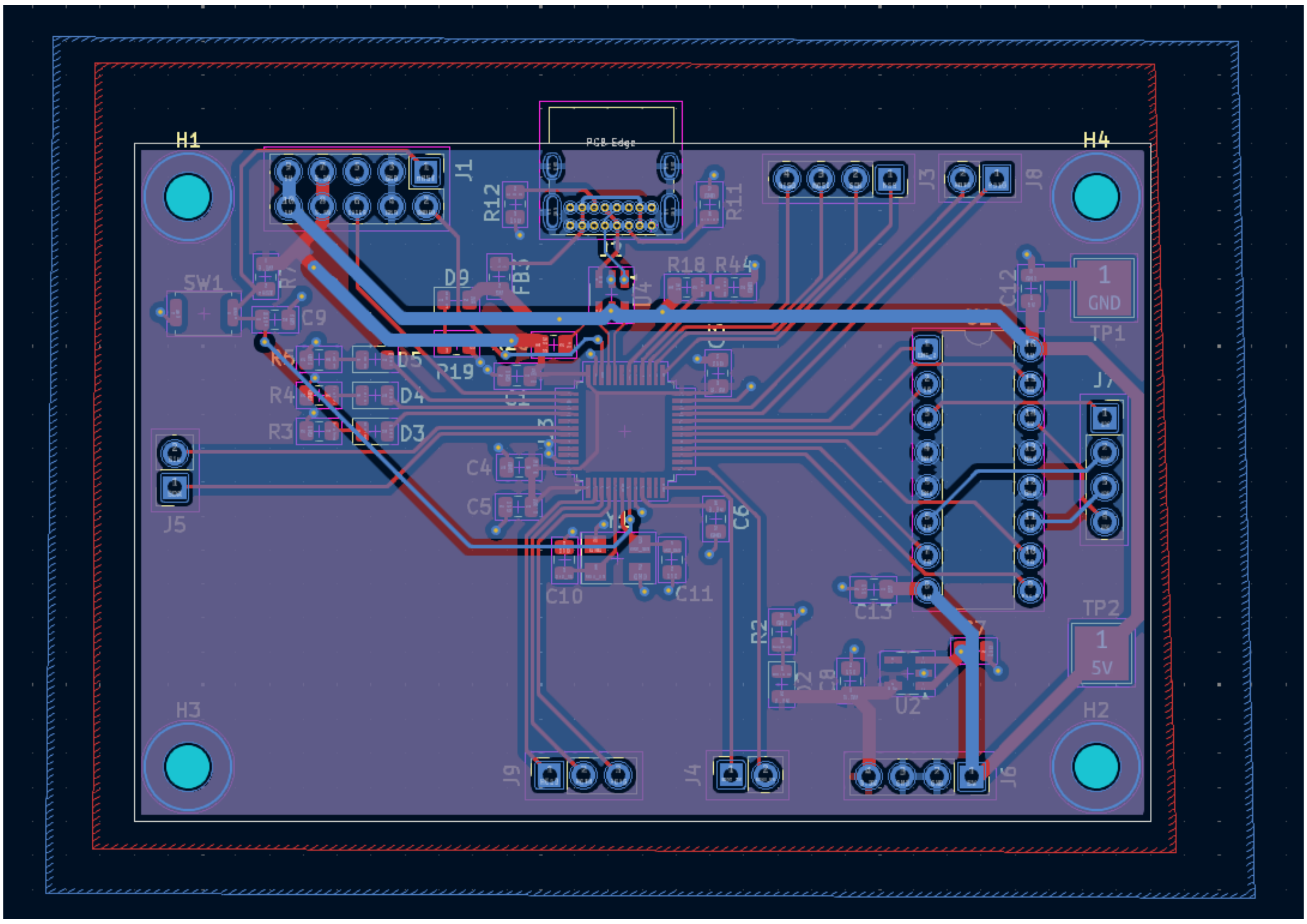
You can draw it with endpoints anywhere it just needs to encompass the entire board. Press B to fill



Add another filled zone on F.Cu for 3.3V.

Once all airwires are gone your board should look similar to this:

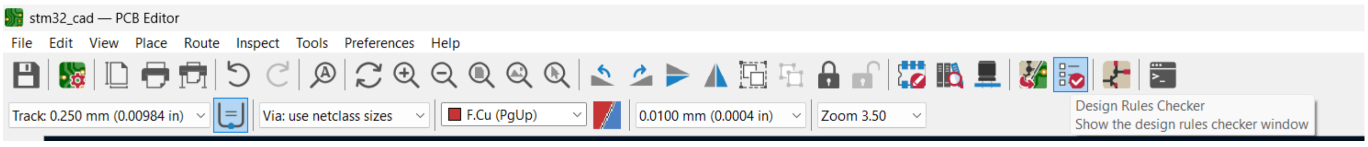




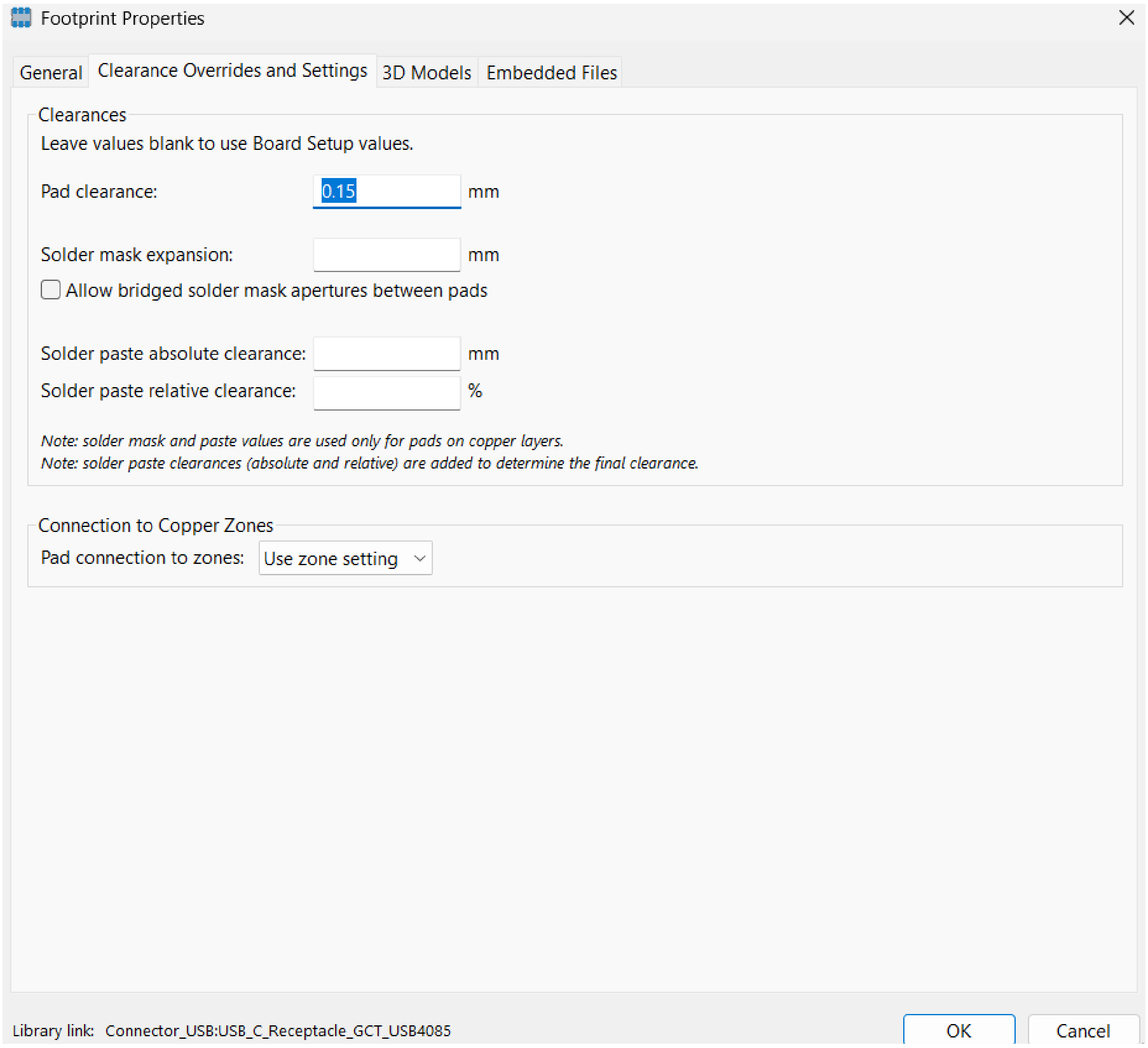
Design Rules Checker

Errors

To make sure that the board you have designed can actually be manufactured, we need to run Design Rules Checker (DRC). Note that KiCAD has default design rules (spacing of traces, sizes of vias, etc.), which differ from the manufacturer's design rules.


stm32_cad — PCB Editor
File Edit View Place Route Inspect Tools Preferences Help
Track: 0.250 mm (0.00984 in)
Via: use netclass sizes
F.Cu (PgUp)
0.0100 mm (0.0004 in)
Zoom 3.50
Design Rules Checker
Show the design rules checker window

You will likely get errors, probably >16. You must pay attention to errors. Warnings can be benign, but you should also check them. Some of the errors you get are "Clearance Violations" due to the spacing of the USB connector pins. To resolve this, double-click on the USB connector and go to `Clearance Overrides and Settings`->`Pad Clearance` and set this to 0.15 mm.


Footprint Properties

General | Clearance Overrides and Settings | 3D Models | Embedded Files

Clearances

Leave values blank to use Board Setup values.

Pad clearance: 0.15 mm

Solder mask expansion: mm

☐ Allow bridged solder mask apertures between pads

Solder paste absolute clearance: mm

Solder paste relative clearance: %

Note: solder mask and paste values are used only for pads on copper layers.
Note: solder paste clearances (absolute and relative) are added to determine the final clearance.

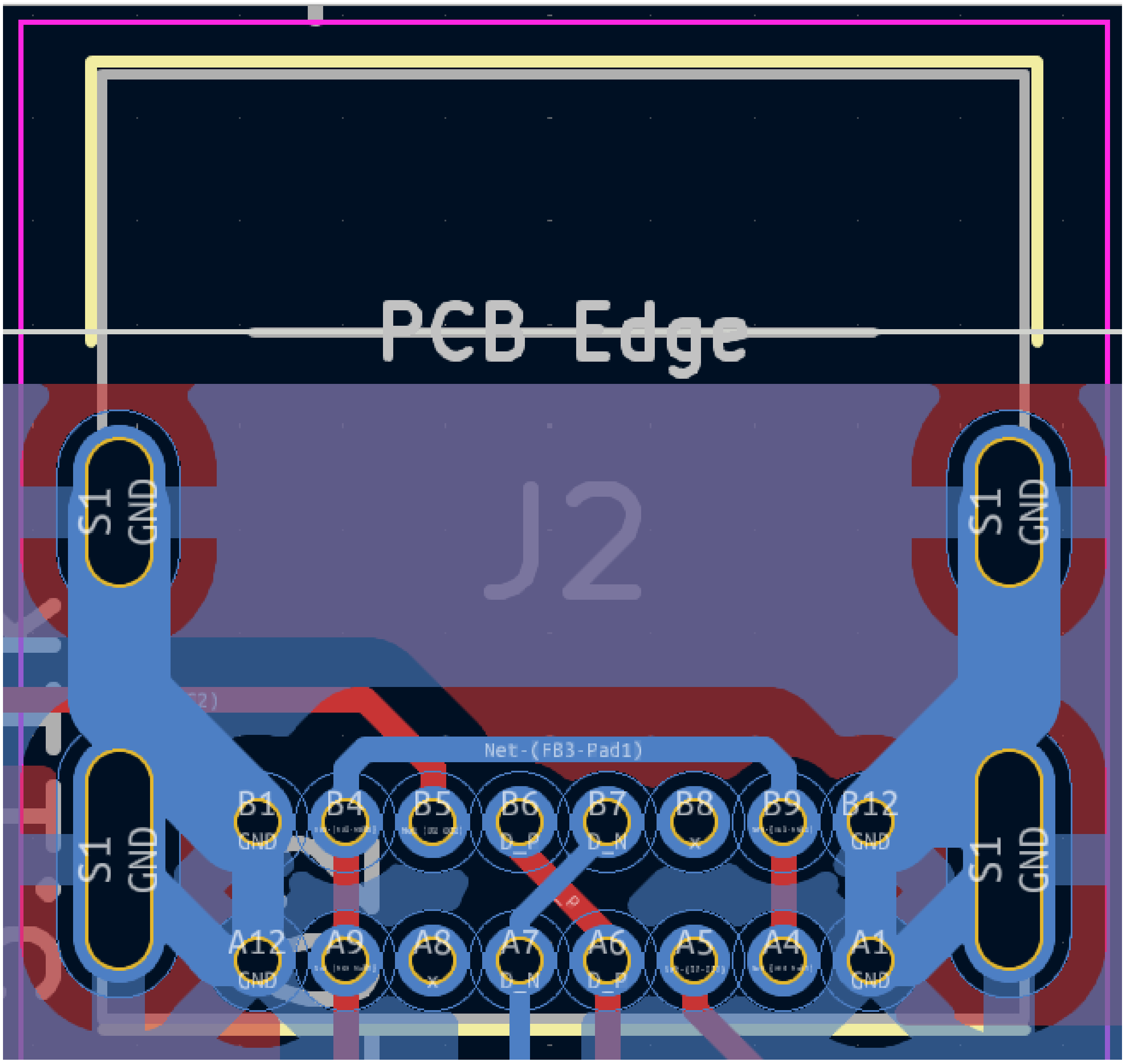
Connection to Copper Zones

Pad connection to zones: Use zone setting

Library link: Connector_USB:USB_C_Receptacle_GCT_USB4085

OK Cancel

This should greatly decrease your error count. You may also have errors related to thermal performance/relief. To resolve those, make sure power pins have multiple connections, like the USB connector shown below.



Warnings

Design rules violations that only have `Warning` severity will likely not impact the functionality or safety of your board. However, you must understand what your wanrings are and if they will matter during or after board fabrication.

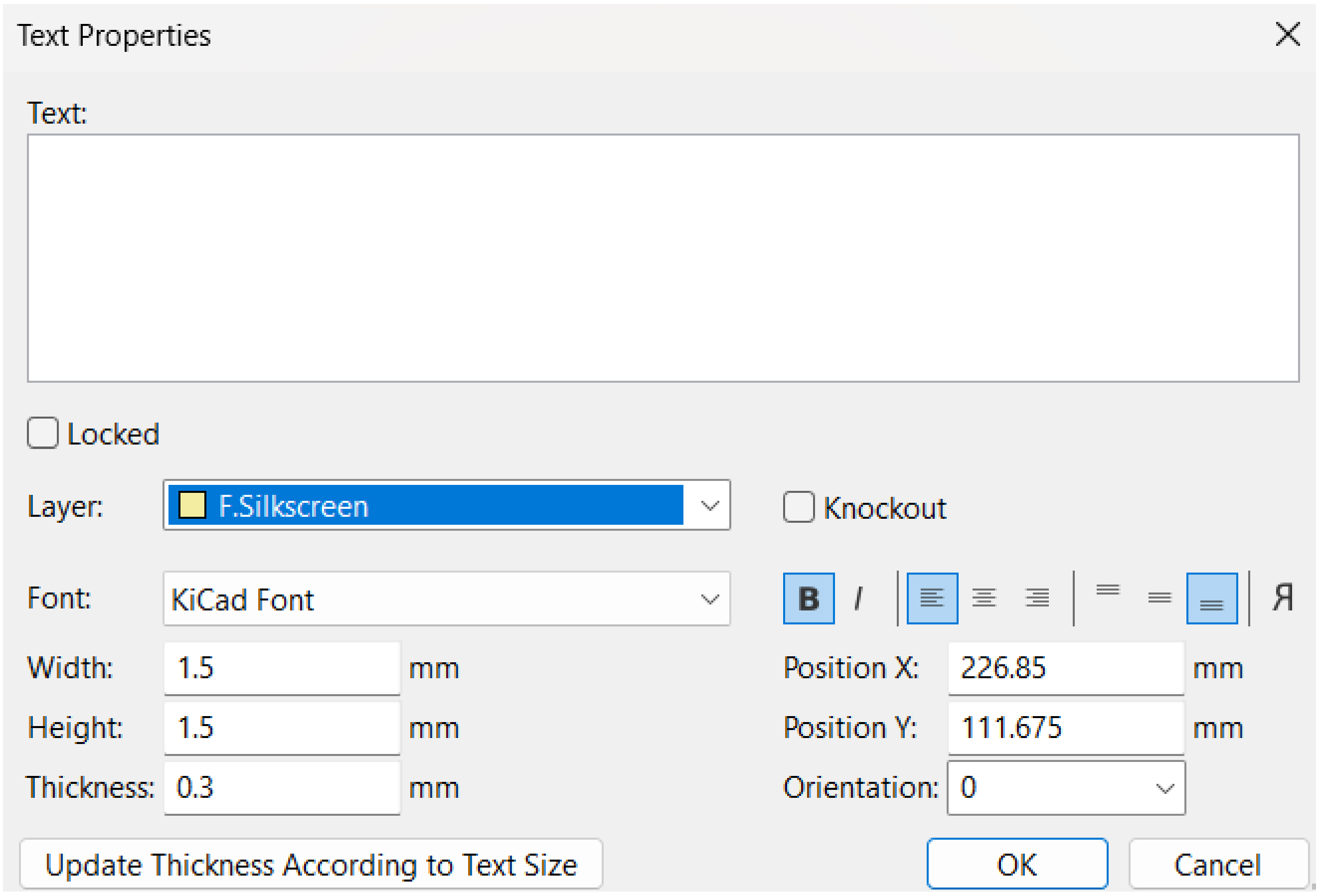
For example, there will be a warning: `Silkscreen clipped by board edge` because the USB connector will hang off the edge of the board. This warning can be ignored.

There may be other "Schematic Parity" warnings where an assigned footprint doesn't match symbol's footprint filters.

Adding text

We will now be adding some text to the board.

Click on the "T" icon on the right side and select F.Silkscreen



Text:

☐ Locked

Layer: F.Silkscreen

☐ Knockout

Font: KiCad Font

Width: 1.5 mm

Height: 1.5 mm

Thickness: 0.3 mm

Position X: 226.85 mm

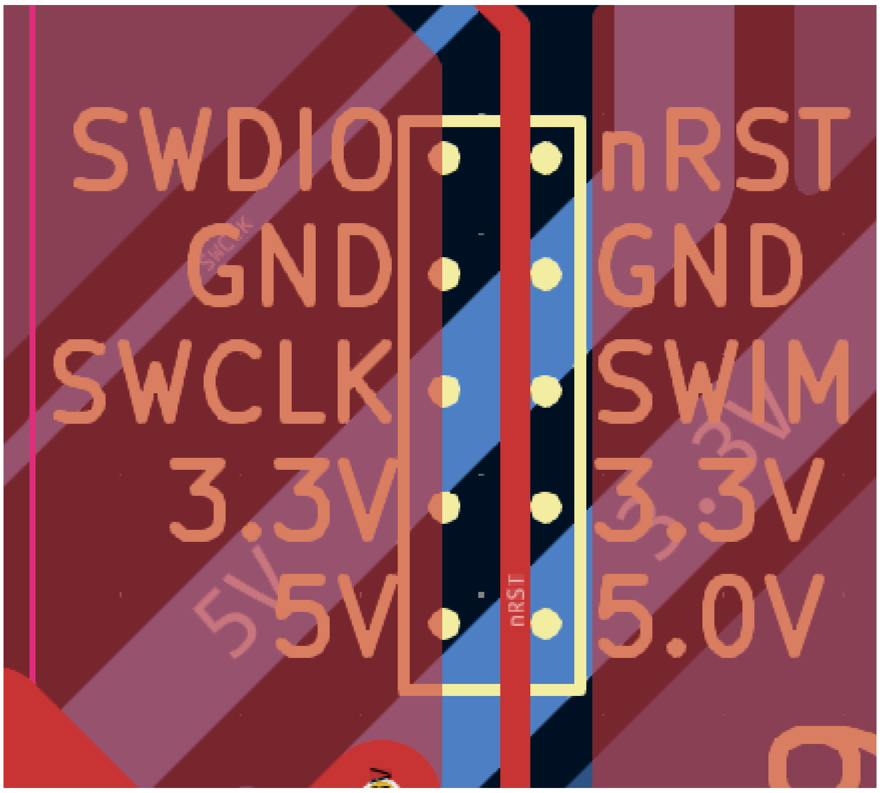
Position Y: 111.675 mm

Orientation: 0

You must have (but are not limited to) the following on your design:

- "CAD Assignment", Team number, "ECE 445"
- Labels for the socket outputs:`"UART", "SPI", "I2C", "5V", "3.3V", "GND"`
- Labels for motor driver outputs: `"1Y", "2Y", "3Y", "4Y"`
- Labels for the test points : `"5V", "GND"`
- Some sort of indication for ST LINK pins.

Recommended method: Small diagram using rectangles, dots, and text. Shown below



Programming Overview

Once you have assembled the board, you will test that there are no major issues. You should test, using a multimeter in Contuinity Mode (beeps if shorted), that none 5V, 3.3V, and GND are shorted. You should also check that the MCU 3.3V is connected to the LDO 3.3V, etc. Also test that the programming pins are not shorted.

Programming of the STM32 MCU is done using the ST Link, not over the USB port It is possible to program over USB, but this requires more firmware and thus is not possible for the first programming.

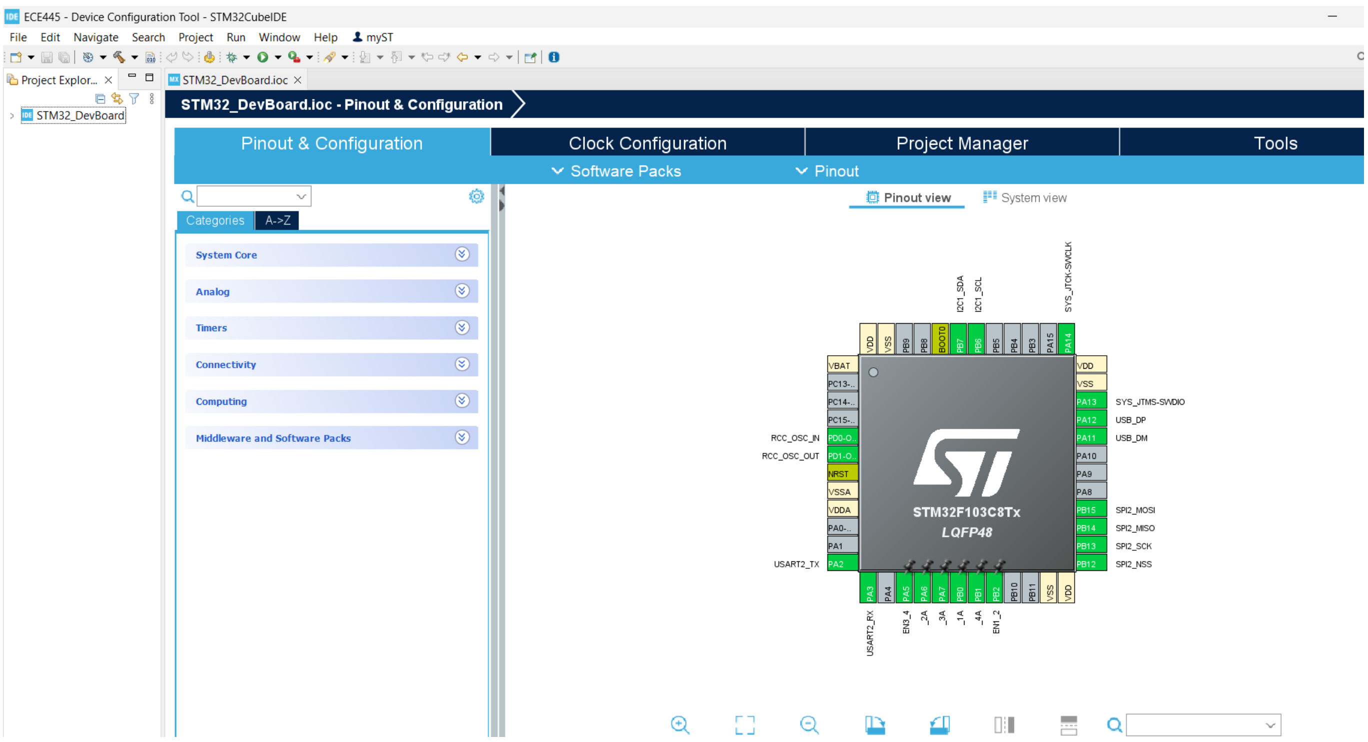
Plugging in the ST-Link

Depending on the ST Link version, the pins may be laid out such that you can plug it directly into your board (if the board has a female connector).

If you cannot directly plut it in, the necessary pins are GND, SWDIO, SWCLK, and 5V (and/or 3.3V). It is safe to plug both power pins in simultaneously, but not necessary.

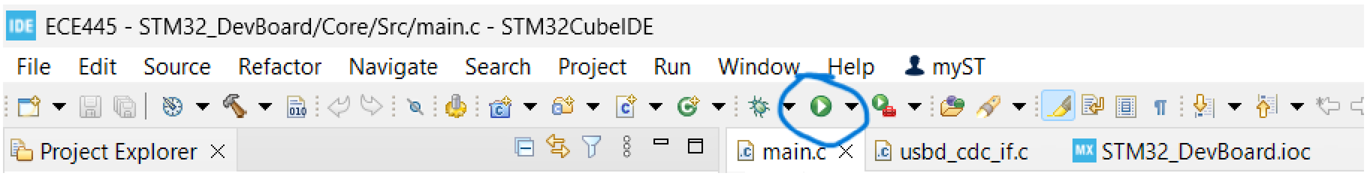
STM32Cube IDE

You have configured the .ioc file already, which helped us create the schematic and layout. Now, it is time to write the code that will run on the STM32F103C8 MCU. To generate the starter code from the .ioc file, click on the horizontal gear icon.



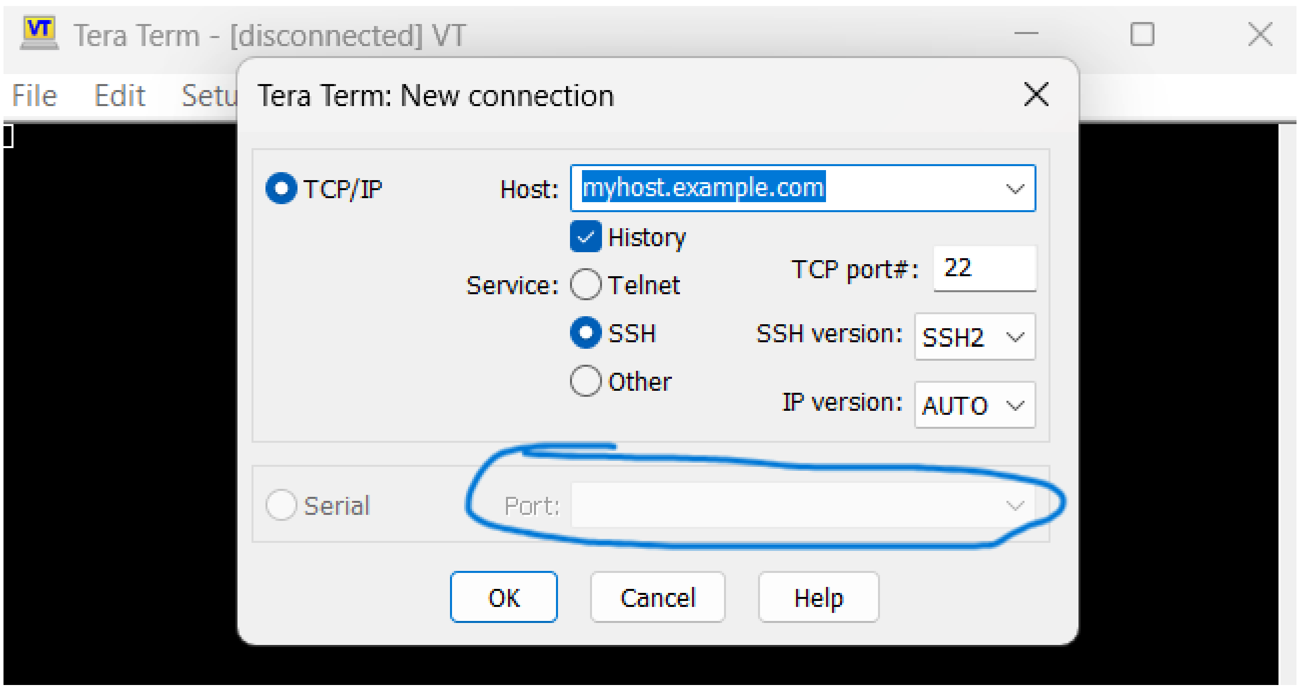
When asked to switch to C/C++ Perspective, click Yes.

To verify that you have soldered on the MCU correctly and attached the ST Link correctly, press the green "Run" button.


ECE445 - STM32_DevBoard/Core/Src/main.c - STM32CubeIDE
File Edit Source Refactor Navigate Search Project Run Window Help myST
Project Explorer
main.c
usbd_cdc_if.c
STM32_DevBoard.ioc

If you get a "Could not connect to target" (or similar) error, there is likely an issue with your soldering or connection since this error implies the issue is between the ST Link and the MCU as opposed to an issue with the ST Link itself.

Once the MCU has been programmed, you can connect the board to a computer via the USB C port. Download Tera Term or use Device Manager to view COM Ports. If the device has been programmed correctly and all the USB configuration in the .ioc was done correctly, you should see a COM port listed in Port. This verified that there are no issues with the physical connection between your computer and the MCU as well as the MCU was programmed with the right configuration.



TCP/IP Host: myhost.example.com
[✓] History
Service: ( ) Telnet (●) SSH ( ) Other
TCP port#: 22
SSH version: SSH2
IP version: AUTO
( ) Serial Port:

Programming Assignment

The next part of this tutorial is the final part of this assignment. You will have to create 2 different programs.

Python

You will write a Python script to send characters to your MCU over the COM port. You must assign one character to be "left" and one character to be "right".

Hints:

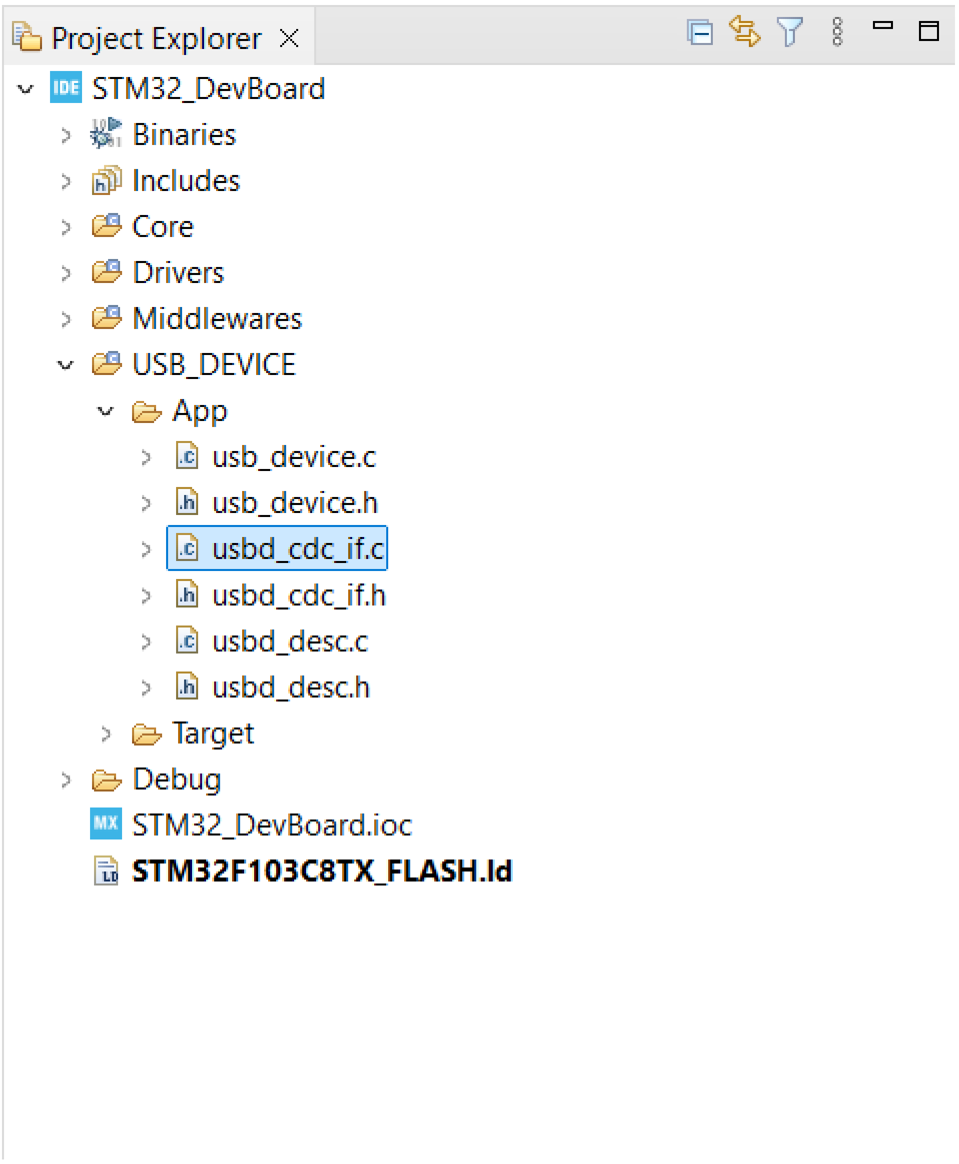
- Use the `serial` Python library with `import serial` for accessing the COM port
- Use the `keyboard` Python library with `import keyboard` to detect key presses.
- When sending data, you should typecast to byte string with `b'your_data'`

C/C++

You will write firmware on the MCU to read the characters sent from the USB COM port and turn a motor in the implied direction, using the GPIOs (1A, 1B, etc.).

USB CDC File

On the left hand side, in the Project Explorer, `USB_DEVICE` -> `App` -> `usbd_cdc_if.c`



You will need to implement the `CDC_Receive_FS` function so that it reads characters from the input buffer. The code snippet below shows (incompletely) a very direct method to control the motor within the USB CDC receive function.

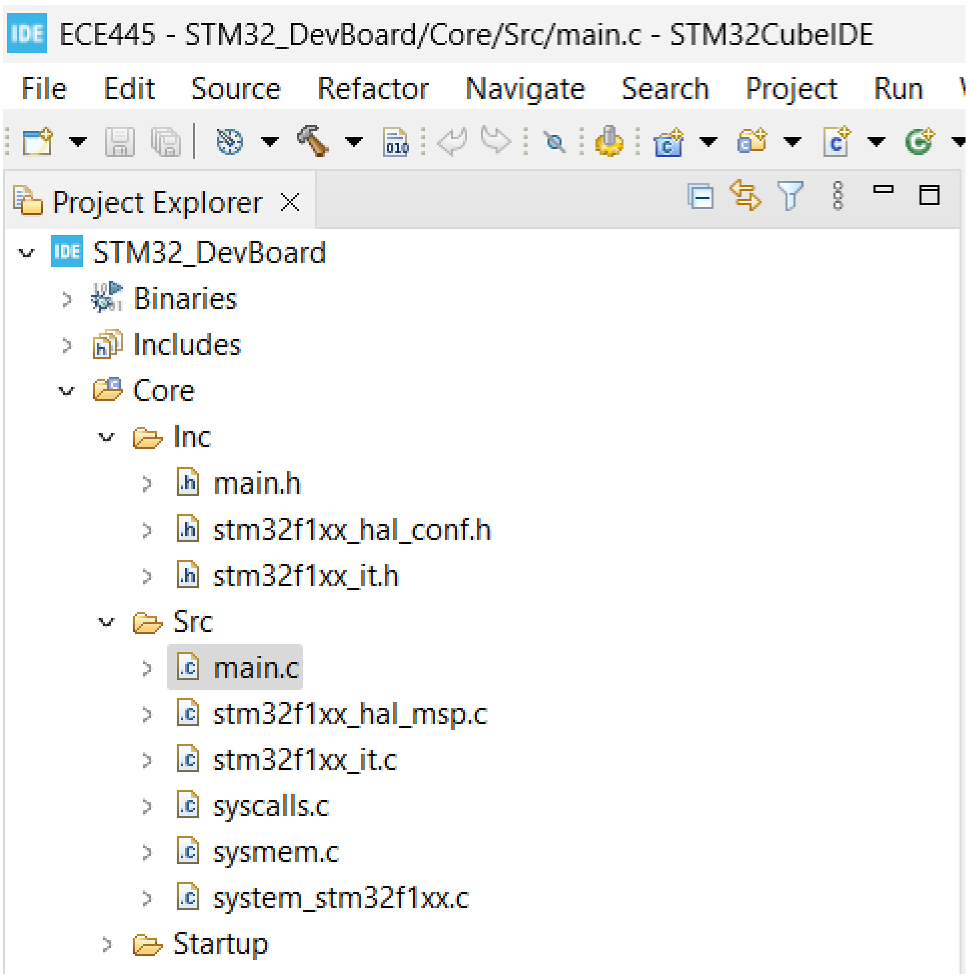
```
#include "main.h"

extern void stepMotor(uint8_t step);
volatile int8_t motor_step = 0;
volatile int8_t motor_step_command = 0;

static int8_t CDC_Receive_FS(uint8_t* Buf, uint32_t *Len)
{
/* USER CODE BEGIN 6 */
if (Buf[0] == 'TODO') {
// TODO: turn motor one direction
} else if (Buf[0] == 'TODO') {
// TODO: turn motor other direction
}
USBD_CDC_SetRxBuffer(&hUsbDeviceFS, &Buf[0]);
USBD_CDC_ReceivePacket(&hUsbDeviceFS);
return (USBD_OK);
/* USER CODE END 6 */
}
```

Main File

In project explorer, open `Core`->`Src`->`main.c`



You will need to read the L293D datasheet to figure out which pins should bd used to get the desired behavior: L293D Datasheet

The below code should be helpful, but will not work from just copying and pasting.

```
#include "usb_device.h"
#include "stm32f1xx_hal.h"

extern volatile int8_t motor_step;
extern volatile int8_t motor_step_command;

const uint8_t stepSequence[4][4] = {
{1, 1, 0, 0},
{0, 1, 1, 0},
{0, 0, 1, 1},
{1, 0, 0, 1},
};
```

```
void stepMotor(uint8_t step) {
HAL_GPIO_WritePin(GPIOB, TODO, stepSequence[step][0]);
HAL_GPIO_WritePin(GPIOC, TODO, stepSequence[step][1]);
HAL_GPIO_WritePin(GPIOC, TODO, stepSequence[step][2]);
HAL_GPIO_WritePin(GPIOC, TODO, stepSequence[step][3]);
}

while (1)
{
    if (motor_step_command != 0) {
for (int i = 0; i < 10; i++) { // Take 10 steps per command
if (motor_step_command == 1) {
motor_step = (motor_step + 1) % 4;
} else if (motor_step_command == -1) {
motor_step = (motor_step - 1 + 4) % 4;
}
stepMotor(motor_step);
HAL_Delay(100);
}
motor_step_command = 0;
}
/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */
}
```

Make sure all user code is structured like this. It may be deleted if not in a "user code" section.

```
/* USER CODE BEGIN */
YOUR CODE HERE
/* USER CODE END */
```

Demo

Once you have achieved the desired behavior, show a TA that you can control the motor direction from a computer.