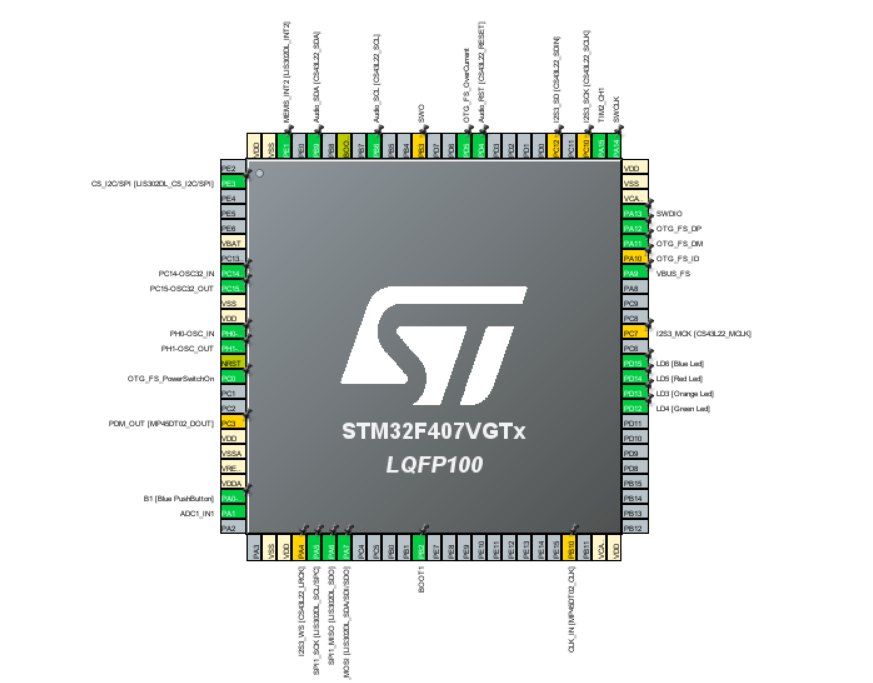
# **Sampling:**

The current sampling method involves using the 12-bit ADC on the STM32F4 Discovery Board. The data is sampled using the built-in ADC and transmitted to a PC over serial using USB Full-Speed. Below is a screenshot of the pin configuration of the Discovery board:



The sampling rate is configured using the built-in timer TIM2 which has a clock speed of 84 MHz. The counter period has thus been configured to 700 to create a sampling rate of 120 ksps. This has all been done in the STM32F407\_ADC file saved under the ADC folder. [Here’s the video that explains how to do this.](https://www.youtube.com/watch?v=r9UlXjrAOtc)

Note: The sampling can also be done without timers and instead setting the sampling rate of the ADC manually. Here’s a [link](https://community.st.com/t5/stm32-mcus-products/nucleo-stm32l432kc-adc-dma-circular-buffer-slower-than-expected/td-p/616791) explaining how to do this.

The sampled data is read using another Python script called SerialReader.py which reads the raw integers, formats them to be legible, and saves them to an output.txt file. This script is also saved under the ADC folder.

The connections between the Discovery board and the radar and amplifier work as follows:

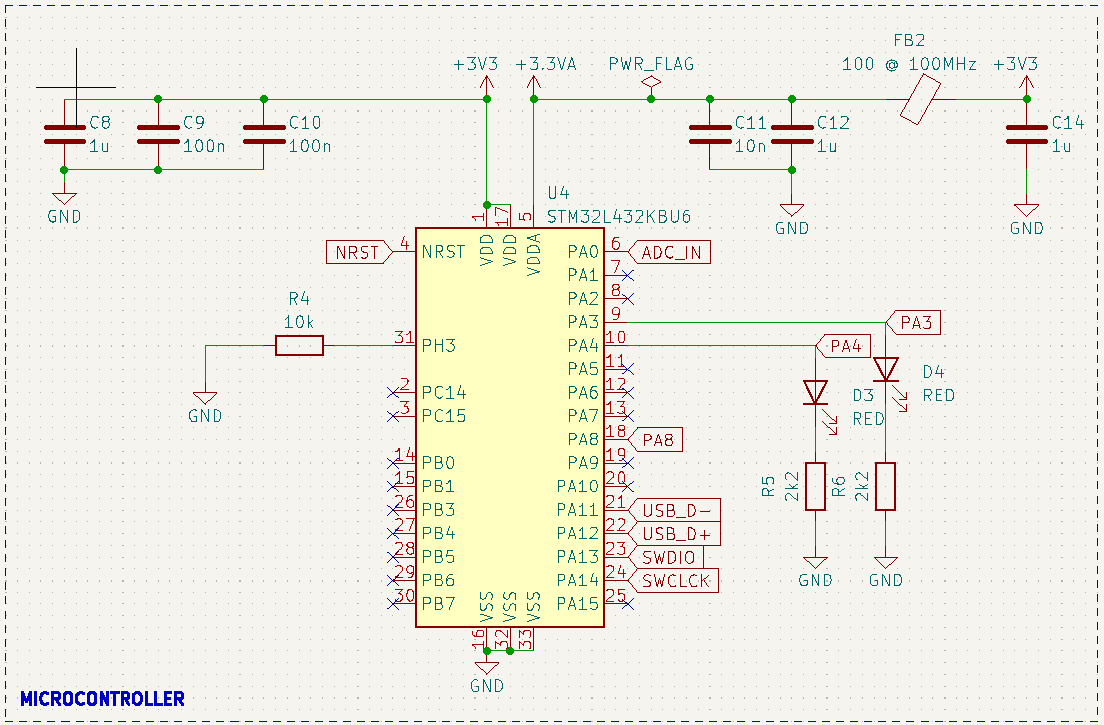
The radar is wired using a leftover Veroboard from last year which connects the CDM324 to the JYVA2 amplifier. The output of the amplifier is connected to pin PA1 on the Discovery board which has been configured to be the ADC input. The Veroboard is also connected to 5V and GND on the Discovery board.

The data stored in the output.txt file is then visualised using the spectrogram in the ReaderSpectrogram.ipynb script. This plots a normal spectrogram as well as a Gaussian Smoothed spectrogram for visualisation of the data.

# **PCB**

## **Version 1:**

The PCB was designed to include the amplification circuitry as well as the sampling circuitry. The sampling is done using an STM32L4 chip which has a 12-bit ADC. The circuitry for this PCB looks as follows:



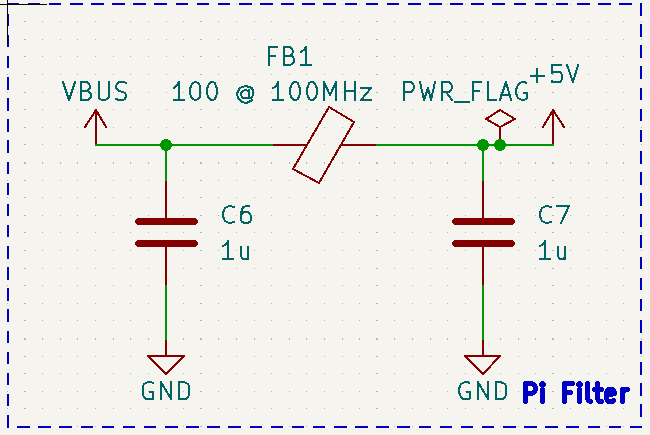
This was designed using the tutorials from Phil’s Lab as a reference. The two tutorials used are:

[His basic PCB design guide](https://www.youtube.com/watch?v=aVUqaB0IMh4)

[His RF PCB design guide](https://www.youtube.com/watch?v=14_jh3nLSsU&t=1499s)

The system is powered using USB which has ESD protection which was also designed using guidance from the RF PCB design guide. The 5V from the USB is dropped down to 3V3 using a voltage regulator (Phil’s Lab uses it in his basic PCB design guide) which is used to power the microcontroller.

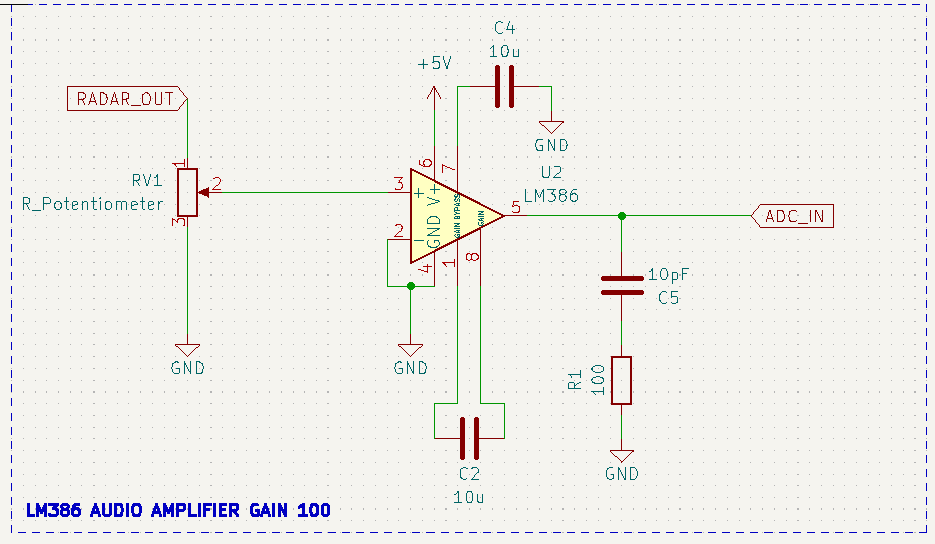
The 5V used to power the radar goes through a Pi Filter which is mentioned by Phil’s Lab in various videos.



This Pi filter is supposed to decrease the power supply noise from the USB.

The amplifier circuit was designed to replicate the LM386 amplifier module recommended by [Butterfield](https://blog.durablescope.com/post/BuildASpeedCameraAndTrafficLogger/).

[This is the amplifier module being replicated.](https://www.diyelectronics.co.za/store/audio/1943-lm386-audio-amplifier-module-200x-gain.html)



The PCB worked well on arrival. It is programmed using the STM32F0 Discovery board. How this is done is explained in the [datasheet](https://www.st.com/resource/en/user_manual/um1525-stm32f0discovery-discovery-kit-for-stm32-f0-microcontrollers-stmicroelectronics.pdf) on page 15. The code that was run can be found under scripts and is called **STM32F432KBU6\_ADC\_Config**. This is a replica of the **STM32F407\_ADC** script which runs on the STM32F4 discovery board. It must be noted that TIM1 and the USB clash meaning that TIM2 had to be used on the PCB. Furthermore, make double sure that the configuration of the timers matches up between the F4 ioc and the L4 ioc (I had to make a few extra changes to the L4 ioc to get it working). Also, this PCB then interfaces with the reader scripts the same as the [STM32F4 discovery board does](#_Sampling:). The L4 samples at 100 ksps in its current configuration and transmits packages of 1024 bytes at a time. The F4 samples at 120 ksps and transmits packages of 2048 bytes at a time.

# **Preliminary Testing Setup**

A preliminary testing setup using the Raspberry Pi 4, STM32F4 Discovery Board, JYVA2 and the CDM324 is being designed to be used while waiting for the PCB to arrive. This setup will be powered by a USB power which was selected using this [information](https://www.powerbankexpert.com/best-raspberry-pi-power-bank/). [The selected power bank.](https://www.takealot.com/romoss-30000mah-sense-8-portable-powerbank/PLID91262824?gad_source=1&gclid=Cj0KCQjwq86wBhDiARIsAJhuphlU8R9r_vR3qNQwqa36rNNXkwlNft8lyQeLt2XboAPI-klBbQuyImAaAgEkEALw_wcB&gclsrc=aw.ds.)

[Potential housing for this setup.](https://za.rs-online.com/web/p/general-purpose-enclosures/2006657)

[RTC for Raspberry Pi.](https://www.pishop.co.za/store/mini-rtc-module-real-time-clock)

I have used this [link](https://forums.raspberrypi.com/viewtopic.php?t=161133) to set up the RTC on the Pi. It should be setting the pi time to the rtc time when not connected to the internet.

I have used [this](https://randomnerdtutorials.com/raspberry-pi-remote-ssh-vs-code/) tutorial to SSH into the Raspberry Pi using VS code. If you switch the SD card on your pi you will need to delete the SSH history on your laptop before connecting using VS code. This is the command that opens the SSH history file: gamin\.ssh\known\_hosts

# **CDM324 and IPM-165 Raw Data Information:**

I was observing the IPM-165 raw output and found that the DC shift was changing a lot and scoped it properly in radar lab. It appears that the orientation of the radar affects the DC shift. When facing upwards where there was lots of clutter, the DC shift was around 200mV. However, pointing horizontally the DC shift was around 20mV. I expect that outside the DC shift will be even less due to the large open space and minimal nearby reflective surfaces. Here is some information obtained in the lab: