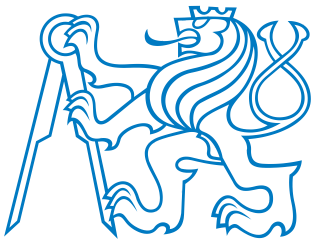
Czech Technical University in Prague

Department of Electromagnetic Field



BACHELOR THESIS

Surveillance FMCW Radar

Supervisor of the bachelor thesis: Ing. Viktor Adler, Ph.D

Study programme: Elektronika a komunikace

Prague TODO 2024

I hereby declare that I have independently written the submitted work and that I have cited all sources of information used in accordance with the Methodological Guidelines on Adherence to Ethical Principles in the Preparation of University Theses and the Framework Rules for the Use of Artificial Intelligence at CTU for Study and Educational Purposes in Bachelor's and Master's Studies.
In Prague, March 3, 2025

Title: Surveillance FMCW Radar
Author: Havránek Kryštof
Department: Department of Electromagnetic Field
Supervisor: Ing. Viktor Adler, Ph.D, Department of Electromagnetic Field

Abstract: TODO

Keywords: TODO

Název práce: Přehledový FMCW radar
Autor: Havránek Kryštof
Katedra: Katedra elektromagnetického pole
Vedoucí práce: Ing. Viktor Adler, Ph.D, Katedra elektromagnetického pole

Abstrakt: TODO

Klíčová slova: TODO

Contents

Abbreviations

Abbreviation	Meaning
AGC	Automatic Gain Control
CW	Continuous Wave
CFAR	Constant False Alarm Rate
DFT	Discrete Fourier Transform
FMCW	Frequency Modulated Continuous Wave
FFT	Fast Fourier Transform
devkit	Development Kit
SNR	Signal to Noise Ratio
SISO	Single Input Single Output
TSV	Tab Separated Values

Introduction

TODO

1. FMCW Radar Fundamentals

Unlike classical Continuous Wave (CW) radars, Frequency Modulated Continuous Wave (FMCW) radars do not broadcast a signal at a single frequency. Instead, they employ a linear frequency sweep across a defined range. This approach enables range estimation without requiring pulsed transmissions while still allowing speed measurements using the Doppler shift. However, velocity calculations in FMCW radars are more complex compared to single-frequency CW radars.

The "MW" suffix in FMCW radar denotes that the system operates in the microwave frequency range. These high frequencies allow for compact antenna arrays, even enabling on-chip integration. Additionally, the millimeter-wave (MMW) portion of the spectrum is typically license-free [**spektrumCTU**] and offers large bandwidths, reducing the risk of interference.

1.1 Comparison FMCW Radar to Pulse Radar

Distance measurement using radar predates FMCW technology by several decades. Early radar systems primarily relied on pulsed electromagnetic signals, measuring the time taken for the signal to reflect back. In such systems, speed can be determined using the Doppler effect as

$$v = \frac{f_{\text{dop}} c_o}{2 f_{\text{rad}}}, \tag{1.1}$$

where f_{dop} is doppler frequency, c_o is speed of light and f_{rad} is frequency of the radar signal. Distance is derived from the time of flight t of the signal as

$$d = \frac{c_o \cdot t}{2}. \tag{1.2}$$

While this approach is conceptually straightforward, it has several limitations, particularly in applications requiring high precision at close range. Achieving fine resolution in distance measurement necessitates very short pulses. However, to maintain sufficient signal-to-noise ratio (SNR), the transmitted pulse power must remain high, regardless of the number of pulses [**jankiraman2018**].

Maintaining high average transmission power poses legal and technical challenges. It increases the risk of interference with other devices and demands bulky, high-power circuitry – often requiring high voltages and even vacuum tubes. Consequently, pulsed radar is predominantly used in applications where fine range resolution is not essential, such as long-range target detection.

One key advantage of pulsed radar is its relatively simple data processing. In contrast, FMCW radar data processing is more complex due to the interdependence of distance and velocity measurements – both the frequency sweep and the Doppler shift contribute to frequency changes in the received signal.

1.2 Basic principles of ideal FMCW radar

Let us picture an ideal FMCW radar system sending a periodic chirp with frequency frequency sweep from f_{c} to $f_{\text{c}} + BW$, so called sawtooth waveform. Other

FMCW systems may use a different modulations such a linear triangular modulation or segmented linear frequency Modulation. These offer some advantages but the nature of the beat signal (which forms a sine wave with sawtooth modulation) is more complex. Especially in case of triangle when multiple targets are present [**jankiraman2018**]. SiRad Easy© kit technically uses a segmented linear frequency modulation due to its limited computational power [**sidarPRO**] but for the sake of simplicity we will stick to the ideal sawtooth waveform.

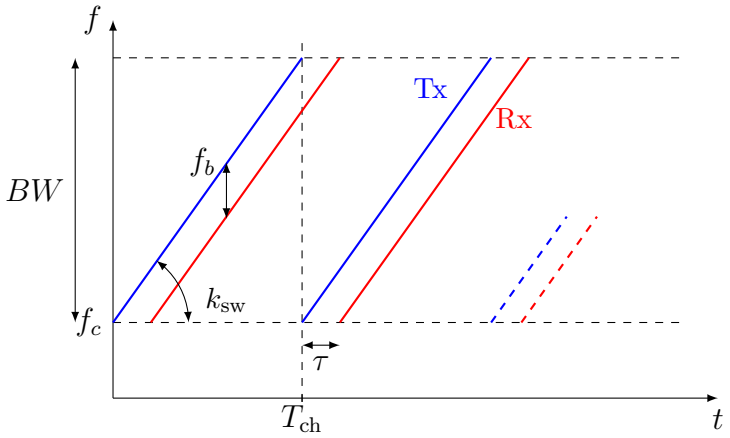


Figure 1.1: Ideal relation of frequency and time for received and sent signal

We can clearly see that in given time t the frequency spread from sent signal to received signal is proportional to the time delay τ. However doing some simple subtraction in spectrogram of the both signals isn't really feasible – the calculation need to take a smarter approach.

Let us define the chirp slope k_{sw} we can describe the change in frequency of the received signal as

$$\Delta f_{\mathrm{s}}(t) = k_{\mathrm{sw}} t = \frac{BW}{T_{\mathrm{ch}}} t, \tag{1.3}$$

where t is the time goes from 0 to chirp length T_{ch}. Standard equation of FM signal can be written as

$$s_{\mathrm{t}}(t) = A \cos\left(\omega_c t + 2\pi \int_0^t f_{\mathrm{s}}(t)\mathrm{d}t\right), \tag{1.4}$$

where A is amplitude of the signal, ω_c is carrier frequency and $f(s)$ is frequency of the signal. Substituting (1.3) into (1.4) we get the signal borrowed from the radar

$$s_{\mathrm{t}}(t) = A \cos(\omega_c t + \pi k_{\mathrm{sw}} t^2). \tag{1.5}$$

Signal bounced back from the target will have the same equation with the only difference being the time delay τ,

$$s_{\mathrm{r}}(t) = A \cos(\omega_c (t-\tau) + \pi k_{\mathrm{sw}} (t-\tau)^2). \tag{1.6}$$

Now we can calculate the product of the two signals, this can be done easily in the real world using a frequency mixer. The result of the multiplication is

$$\begin{aligned} s(t) = s_{\mathrm{r}}(t) \cdot s_{\mathrm{t}}(t) = \frac{A^2}{2} \cos\left(2(\omega_c - 2\pi k_{\mathrm{sw}} T_\tau)t + 2\pi k_{sw} t^2 + (\pi k_{\mathrm{sw}} \tau^2 - \omega_c \tau)\right) + \\ + \frac{A^2}{2} \cos\left(2\pi k_{\mathrm{sw}} \tau t + (\omega_c \tau - \pi k_{\mathrm{sw}} \tau^2)\right). \end{aligned} \tag{1.7}$$

First additive term will lead to a signal with very high frequency, well above $2\omega_c$, this term doesn't carry any useful information and is usually filtered out – either by low pass filter or the frequency mixer itself [**graham2005**]. Second term is so call beat signal whose frequency is directly proportional to the time delay τ. Applying a first time derivative to the cosine argument we get the frequency of the beat signal.

$$f_{\mathrm{b}} = \frac{1}{2\pi}\frac{\partial}{\partial t}\left(2\pi k_{\mathrm{sw}}\tau t + (\omega_c\tau - \pi k_{\mathrm{sw}}\tau^2)\right) = k_{\mathrm{sw}}\tau. \tag{1.8}$$

Calculating the distance to the target is now trivial, delay τ is equal to the time it takes for the signal to travel to the target and back

$$R = \frac{c_0\tau}{2}, \tag{1.9}$$

By substituting (1.8) into (1.9) we get the equation for distance

$$R = \frac{c_0 f_{\mathrm{b}}}{2k_{\mathrm{sw}}} = \frac{c_o f_{\mathrm{b}} T_{\mathrm{ch}}}{2BW}. \tag{1.10}$$

1.2.1 Limits of Range Measurement

Absolute limit for maximal distance is given by the time it takes for the signal to travel from the radar to the target and back. Would the distance be greater than a time of single chirp the signal would be interpreted as coming from a closer target. That gives us an maximal limit on beat frequency $f_{\mathrm{b}} = BW$.

However in most case the limit will imposed not by T_{ch} respectively BW but by sampling frequency f_{s}. In order to avoid aliasing the Nyquist-Shannon theorem must be satisfied thus limiting the maximal beat frequency to $f_{\mathrm{s}}/2$ and resulting in maximal distance of

$$R = \frac{c_o f_{\mathrm{s}}}{4k_{\mathrm{sw}}}. \tag{1.11}$$

While sampling with frequency f_{s} we get $N = \mathrm{f_s}T_{\mathrm{ch}}$ samples applying a DFT to the signal we get N samples in spectrum with frequency resolution of

$$\Delta f_{\mathrm{b}} = \frac{f_{\mathrm{s}}}{N} = \frac{1}{T_{ch}}. \tag{1.12}$$

We can see that the resolution of spectrum is only inversely proportional to the chirp length and doesn't have any relation to sampling frequency [**jankiraman2018**]. Now we can enter Δf_{b} into (1.10) to get the minimal distance that can be measured as

$$\Delta R = \frac{c_o}{2BW} \tag{1.13}$$

Thus in order to increase resolution in range a wider bandwidth is needed.

There are of course other effect impeding the resolution of the radar system – such a phase noise around targets or sweep nonlinearity. Sweep Linearity can be both in the ramp itself – leading to decreasing resolution with range (with both linear and quadratic errors present) [**graham2005**] (Appendix D and E). Or in sweep recovery (time to return to the start of the sweep) which leads to a fix decrease in resolution [**piper1995**]. Both are however largely compensated in modern radar systems by using a closed feedback loop [**graham2005**].

1.2.2 Speed Measurement

In order to demonstrate the effect of moving target on the beat frequency we can redefine the time delay τ as

$$\tau = \frac{2(R_0 + vt)}{c_o} \tag{1.14}$$

where R_0 is the initial distance to the target and v is the radial speed of the target. Within a single chirp there is no way to distinguish between the effects distance and speed of the target – thus multiple chirps are needed. Rewriting as

$$\tau = \frac{2(R_0 v(nT_{\text{ch}} + t_{\text{s}}))}{c_0} \tag{1.15}$$

where n is the number of chirps, T_{ch} is the chirp length and t_{s} denotes time withing a single chirp ($0 \leqslant t_{\text{s}} \leqslant T_{\text{ch}}$). Substituting (1.15) into low frequency part (1.7) leads to very complex equation, however according to [**suleymanov2016**] most of the terms can be neglected leading us to

$$s(t_{\text{s}}, n) = \frac{A^2}{2} \cos\left(\frac{4\pi k_{\text{sw}} R_0}{c_0} t_{\text{s}} + \frac{2\omega_c v n}{c_0} T_{\text{ch}} + \varphi_0\right), \tag{1.16}$$

where φ_0 is a phase shift given by the initial distance to the target. Its clear that first element describes predominantly the distance to the target and the second one the speed of the target. We can also see that speed will not affect a beat frequency in a single spectrum but will lead to a phase shift across multiple spectrums.

In order to calculate doppler shift frequency

$$f_{\text{d}} = \frac{2 f_{\text{c}} v}{c_0}, \tag{1.17}$$

we can use 2D Fourier transform – first in time within a single chirp and then in time across multiple chirps. This will lead to a so called range-Doppler map which on one axis contains information about speed and on second distance of the target [**suleymanov2016**].

Speed resolution is derived from a number of chirps N we are analyzing and their length T_{ch} as

$$\Delta v = \frac{c_0}{2 f_c} \frac{1}{N T_{\text{ch}}}. \tag{1.18}$$

On the other hand the increase in number of chirps will affect the distance measurement if the target movement speed is sufficiently high.

2. SiRad Easy©

Indie Semiconductor's SiRad Easy© is an FMCW radar system development kit designed primarily for automotive applications. Out of the box, it offers two headers a 24 GHz and a 122 GHz module both based on ICs from Indie Semiconductor (TRX-024-007 and TRA-120-001, respectively). Both are strictly SISO IC with two antennas – one for receiving another for transmitting so azimuth estimation is not possible by default.

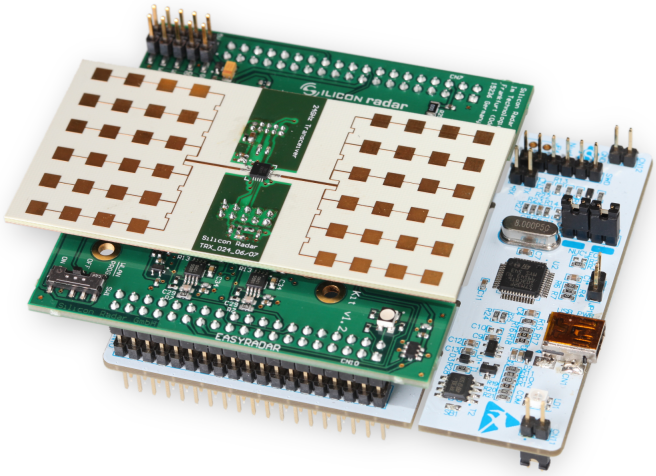


Figure 2.1: SiRad Easy© 24 GHz configuration

Direct communication with the radar board itself is not possible, or at least, the communication interface is undocumented. Instand there is always and intermediary in the form of STM32 Nucleo series microcontroller. To this microcontroller does the user connect either directly with UART over USB or with WiFi over on board ESP32. Both are relatively low bandwidth communication – the serial maxing at baudrate of 1 000 0000.

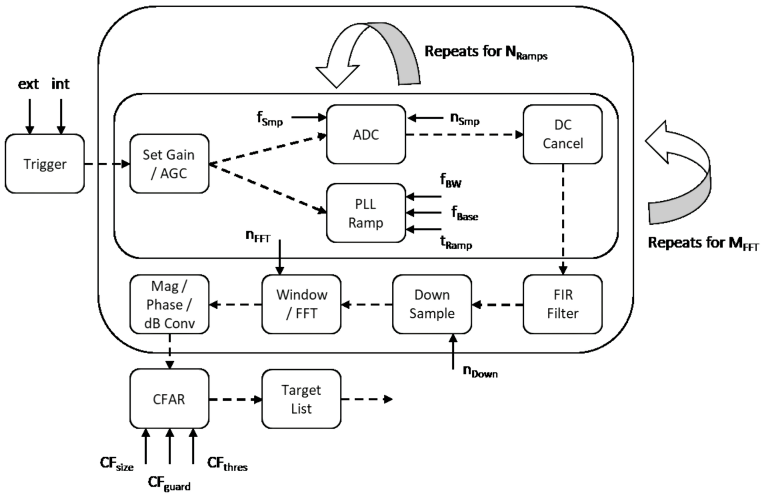


Figure 2.2: Flow of Radar Measurement on SiRad Easy©

The devkit is designed for easy integration into existing projects; however, this

unfortunately poses limitations for this thesis. In normal operation, the radar system functions as a black box (processing schema shown in 2.2), implementing its own filtering, FFT, and CFAR algorithms while only reporting a target list to the user.

This entire sequence can be triggered either internally (driven by a configurable oscillator) or externally (via a GPIO pin or UART input). Once trigger is received the device carries out a user configured number of chirps. The chirp length is determined by the number of sampling steps and the ADC sampling time as follows:

$$t_{\text{ramp}} = \frac{t_{\text{ADC}} \cdot (N_{\text{samples}} + 85)}{36 \text{ MHz}} \ \mu\text{s}, \tag{2.1}$$

with the manufacturer recommending an optimal time of 1 ms for good SNR [**sidarPRO**]. What happens to the sampled ramps remains unclear, as the manufacturer has not provided details. It is likely that the data are somehow averaged to improve the SNR of the reported signal. As only N_{samples} are reported by the radar but according to manufacturer provided equation (2.1) each ramp should be sampled with that number of samples. Additionally, due to the devkit's limited computational resources, the radar does not emit a continuous sawtooth waveform but rather a segmented one [**sidarPRO**].

2.1 Outline of Chosen Configuration

For the purpose of this project, the devkit was configured to output raw data from the 12-bit ADC in the form of in-phase and quadrature components of the signal. This allows for a more detailed analysis of the radar system and enables the implementation of custom signal processing algorithms. For these reasons, the default windowing and undocumented filtering was also disabled.

After configuration of the output regimes, it is necessary to find the correct balance between the number of ramps, samples, and sampling frequency. Since the radar will be in motion, frequent updates are crucial to ensure accurate reporting. Faster updates allow for higher platform speeds or make it possible to use neighboring samples to aid in evaluation of current one (As the angular distance between them remains relatively small.).

Reducing the number of ramps slightly improves the update rate, but the effect is minimal—decreasing the number of ramps from 16 to 1 results in only an 11 % increase in reporting speed. Similarly, changes in sampling frequency yield only minor improvements. The only parameter that significantly impacts the update rate is the number of samples—halving the number of samples leads to an approximately 50 % increase in speed. This is with the sampling frequency adjusted for optimal SNR (close to 1 ms). Even then, reports are received roughly every 60ms (for 256 samples), and the maximum detection range is more than halved.

Another parameter that requires consideration in rotary applications is AGC. Using it adds two additional ramps that are used solely to set gain value but it also can lead to inconsistent weights of values between two neighboring data sets. Tough former only introduces a small slowdown but the letter could be a problem. Especially since radar doesn't really report change in AGC value. Thus its safest to turn AGC off and set the gain manually depending on how the readings appear in post processing.

It is also important to note that the radar system is not well-suited for on-the-fly configuration changes. Applying a new configuration takes a considerable amount of time, and the radar does not provide any feedback to indicate when a new configuration can be safely applied. As a result, commonly used techniques such as alternating chirp slope or frequency modulation are not feasible.

Regarding the output format, the radar system supports two options: binary and TSV. Since the output speed does not differ significantly between the two, the primary deciding factor is the ease and speed of parsing. Even though MATLAB is not particularly optimized for parsing binary data, it still processes binary output approximately 40

2.2 24 GHz Header

Center of 24 GHz header is a SISO TRX-024-007 transceiver which integrates low noise amplifier, frequency mixer, filters and VCO into a single chip. It is primarily designed to operate in the ISM band (24.0–24.25GHz), with an additional ultra-wideband mode supporting 23–26 GHz [**sidarTRX24**]. On the SiRad Easy© there is no distinction made between those two modes and the user is free to set any bandwidth [**sidarPRO**] . The transmitter output power ranges from 2.5 dBm to 6 dBm, depending on the configuration [**sidarTRX24**]. A maximum range of 400 m is advertised [**sidarMANOld**], though this is likely under ideal conditions when observing a large target. Also as previously stated the radar will operate on much lower sampling frequency than needed to achieve such results.

As shown in Figure 2.1, the chip is connected to two microstrip patch antennas. The patches are arranged in a relatively standard configuration, forming a 6×4 array with spacing approximately equal to half the wavelength at 24 GHz.

Since the manufacturer did not provide any information about the radiation pattern of the array, a simulation was conducted using the CST simulation suite. Additionally, the manufacturer did not disclose the substrate parameters for the radar board specifically. Fortunately, the TRX-024-007 datasheet includes a board stack-up for the chip's evaluation board. It was assumed that the same stack-up would be used—18 μm copper for the traces and ground plane, with a 250 μm thick Rogers RO4350B substrate [**sidarTRX24**].

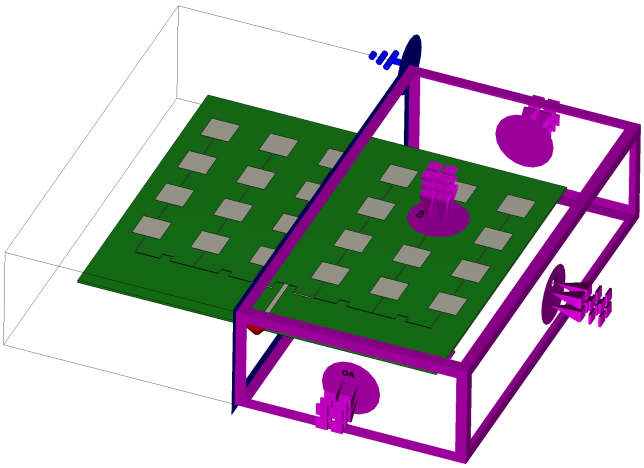


Figure 2.3: Simulated 24 GHz header with boundary conditions showed

After measuring the dimensions of the array using an optical microscope, the entire array was redrawn in CST Studio. Figure 2.3 shows the applied boundary conditions: the antenna was placed in open space, and an $H = 0$ condition was set along the symmetry plane to speed up the simulation.

After performing a standard time-domain simulation with an excitation signal ranging from 0 to 26GHz, the antenna array exhibited a minimum reflection coefficient $s_{11} \doteq -26.4$ dB at 23A.478 GHz with second minimum at 24.518 GHz (Figure 2.4). Lack of minimum at 24 GHz may be attributed to measurement difficulties (The etching quality of the copper traces was suboptimal.) and the neglecting variations of substrate parameter due to interactions of different layers.

At both frequencies with minimal reflection, a far-field radiation pattern was calculated. For clarity, only the 24.518 GHz pattern is shown in Figure 2.5. The main lobe width was measured at approximately 16 degrees (Figure 2.6) along the 180-degree norm (for orientation refer to the red cone on the PCB), with a peak gain of 18.6 dBi and side-lobe suppression of -13 dB. For the 90-degree norm, the main lobe width (Figure 2.7) was 30 degrees, with side-lobe suppression of -10 dB.

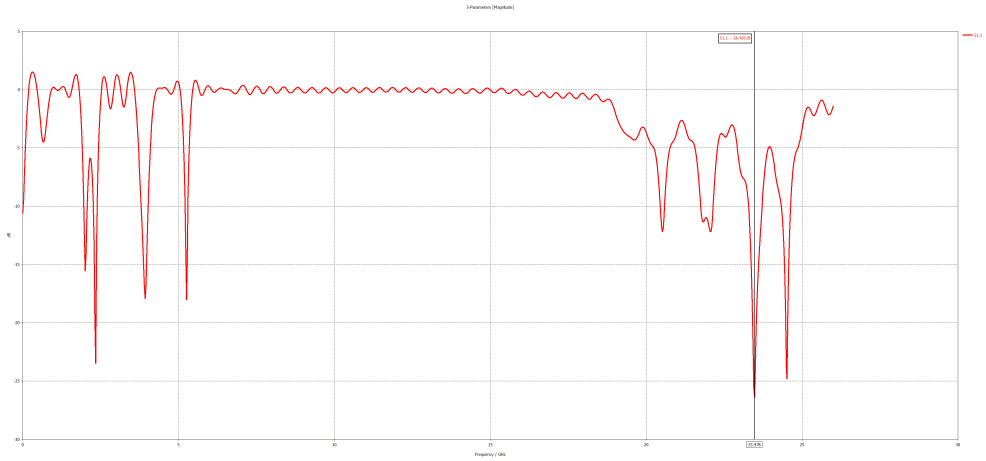


Figure 2.4: s_{11} parameter of the 24 GHz header

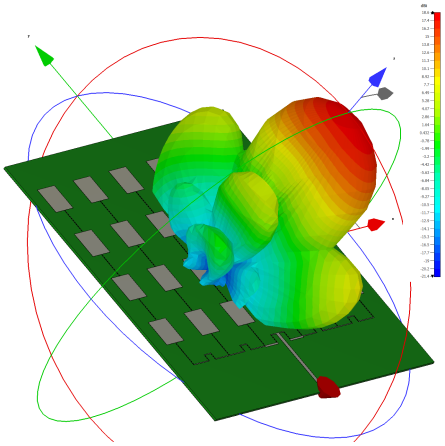


Figure 2.5: Radiation pattern of 24 GHz header – 3D view

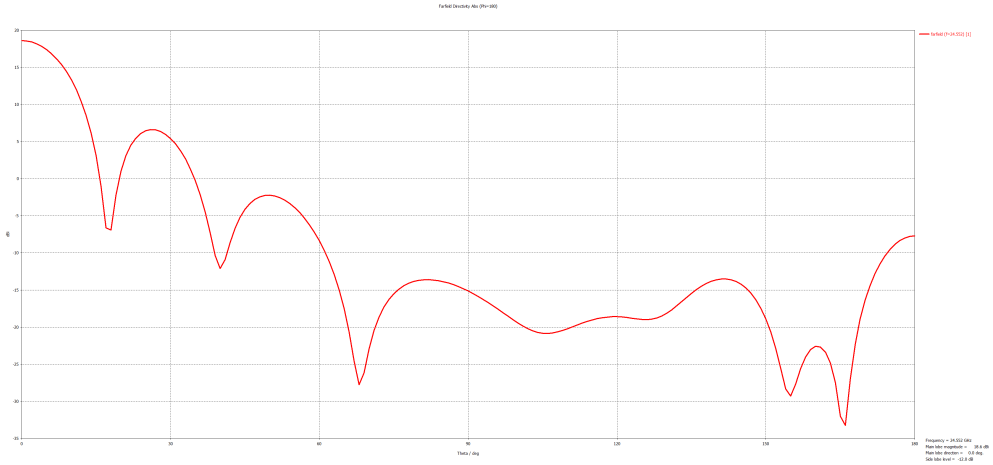


Figure 2.6: Radiation pattern of 24 GHz header – 180° norm

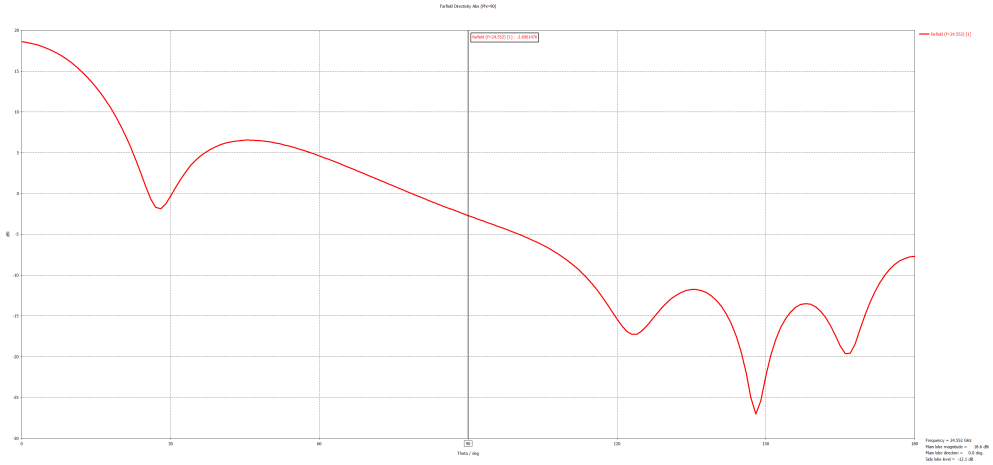


Figure 2.7: Radiation pattern of 24 GHz header – 90° norm

2.3 122 GHz Header

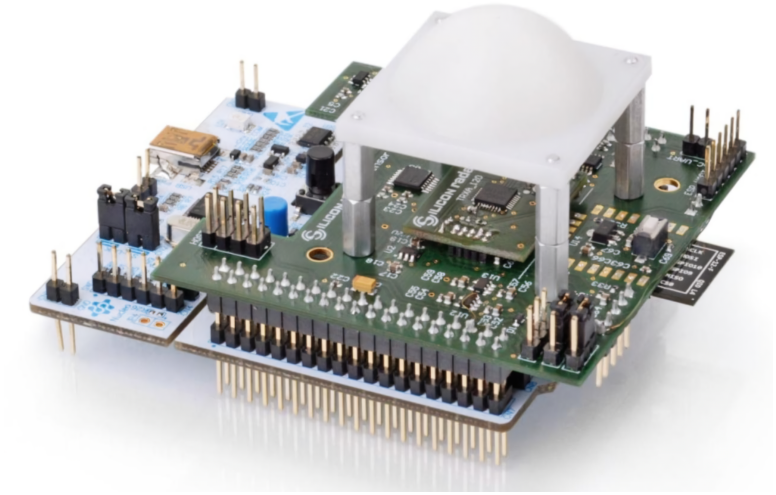


Figure 2.8: SiRad Easy© with 122 GHz header

The 122 GHz header is based on the TRX-120-001 transceiver, which, in addition to the essential components required for RF transmission and reception, also incorporates two on-chip antennas. It's designed to operate in the 122-123 GHz band, with output power ranging from -7 dBm to 1 dBm [**sidarTRX122**]. The chip is capable of detecting large targets at distances of up to 40 m [**sidarMANOld**].

Out of the box performance of the system is quite bad (Figure 2.9) with width of the main lobe being roughly ±40° in both E-plane and H-plane [**sidarTRX122**]. However, this can be significantly improved using the supplied collimator lens, reducing the main lobe width to ±4° [**sidarTRX122col**] (see Figure 2.9).

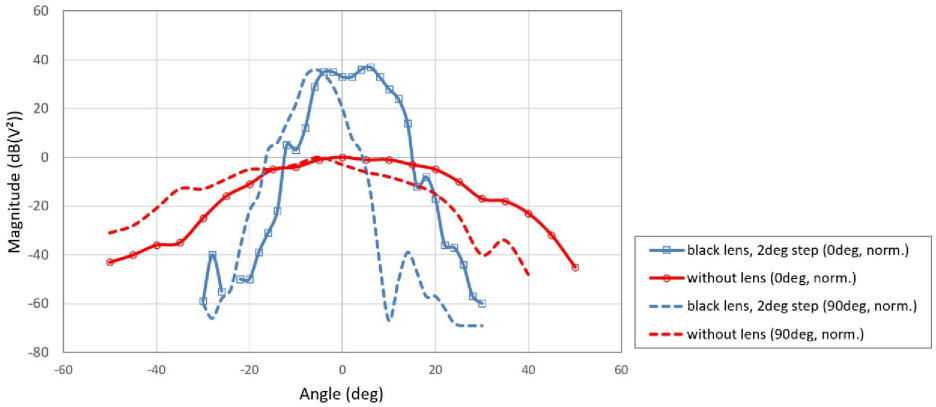


Figure 2.9: Radiation pattern of 122 GHz header comparison

3. Rotary Platform

Following chapter outlines design process and operation of a rotary platform specifically designed for SiRad Easy© radar system.

3.1 Platform Design Parameters

To begin, it is essential to outline the fundamental requirements for the platform. These stem from the physical capabilities of the SiRad Easy© and need to provide easy to use interface to control the platform.

3.1.1 Physical Capabilities

The primary constraints on the physical design arise from the radar's radiation pattern (considering both the 24 GHz and 122 GHz headers). These patterns determine the necessary clearance in front of the radar and, more importantly, the precision required for platform movement. As stated previously, the 24GHz radar has a main lobe width of approximately $\pm 7°$, while the 122 GHz radar has a main lobe width of $\pm 4°$ [**sidarTRX122**]. To minimize strong reflections from both the main lobe and side lobes, a conservative clearance of $\pm 45°$ in front of the radar was selected.

Due to the relatively low angular resolution of the radar, high platform precision is not required. A basic 200-step stepper motor with a step size of 1.8° is sufficient; however, smoother motion simplifies software compensation of the movement. Given the radar's low weight—measured at 120 g, including the mounting bracket, smoothness of movement can be achieved solely through motor microstepping.

High-speed movement is unnecessary for this application. The manufacturer specifies a maximum update frequency of 50 Hz, corresponding to a new measurement every 20ms [**sidarMAN**]. In this use case, the update frequency will be closer to 10-20 Hz. Using

$$t_{\text{angle}} = \frac{60}{360 \cdot N_{\text{RMP}}} \cdot \alpha, \tag{3.1}$$

where t_{angle} represents the time spent traveling an angle α in seconds, and N_{RPM} is the number of rotations per minute, we can calculate that even at a low rotational speed of 60 RPM, an 8-degree movement (matching the angular width of the main lobe for the 122 GHz radar) takes only 10 ms – too fast to properly interpret the data.

3.1.2 Software Requirements

Given its widespread adoption as an industry standard for controlling multi-axis machines, G-code over serial is a natural choice for the platform's communication format. Beyond the basic functionality typically offered by G-code interpreters, the platform must support additional features to reduce the user's manual control burden. These features include the ability to define movement limits and preprogram sequences of movements for autonomous execution by the platform.

For uplink communication, the platform must provide real-time information about its current position and speed. This data enables the user to make any mathematical corrections and properly interpret radar's gathered data.

3.2 Platform Construction

As the platform needs to transmit data from the rotating section to the stationary base, a slip ring is required. Due to the relatively low transmission speed of the radar and the absence of special requirements such as waterproofing, an affordable model, UH3899-01-0810 from Senring, was selected. This is a classical contact slip ring that features a dedicated USB 2.0 connection along with 8 additional signal wires, with an advertised insertion loss of less than 2 dB [**slipring**]. More problematic than loss is cross talk between signal wires when stepper motor responsible for tilting the radar is running. USB 2.0 connection doesn't seem that effected, however endstop used for homing requires adding a low pass filter to its output to help with noise.

Unfortunately, the manufacturer opted for a non-standard male-male USB 2.0 connection, requiring a female-female adapter to connect the radar to the slip ring. Additionally, when used with a poor-quality cable, the system exhibits signal integrity issues. These problems could likely be mitigated by integrating a signal conditioner into the transmission line.

The rest of the design is relatively simple. The fixed section mounts the slip ring with the stepper motor positioned underneath, directly driving a shaft connected to the rotating platform. The connection is secured using long M4 set screws that pass through the slip ring and hold the shaft in place. A 3D-printed housing serves only as a centering guide and is not load-bearing.

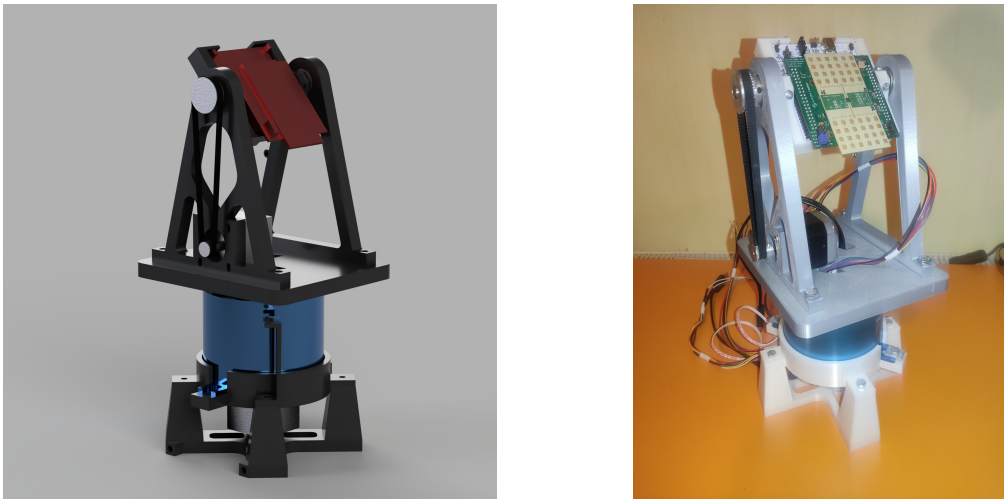
The rotating section features a simple A-frame design that elevates the radar, which is mounted on bearings, allowing it to tilt freely. To control the tilt, a second stepper motor is mounted on the rotating platform and linked to the radar via a 2:1 down-gearing ratio using a standard 8 mm belt. An optical endstop, used for homing of the platform, is mounted on the second support strut.

Since mechanical stresses are minimal, most parts can be 3D-printed using standard PLA filament. The only non-3D-printed components are the screws, bearings, and stepper motors. The final assembly (Figure 3.1) measures approximately 33 cm in height and has a footprint of 20 × 20 cm.

3.2.1 Platform Electronics

The electronic aspect of the platform is relatively simple, involving only two main tasks: driving the stepper motors and implementing a homing mechanism.

Given the low load on the stepper motors and the platform's inability to accumulate significant momentum, a basic stepper driver without feedback control is sufficient. For this purpose the A4988 stepper driver was chosen due to its low cost, microstepping capabilities and basic current control [**a4988**]. A minor drawback is the lack of feedback from the driver to the microcontroller, including the absence of stall detection. To simplify the design, A4988 development kits were used and soldered onto a prototyping board, eliminating the need for a custom PCB.



(a) 3D render (b) Photo

Figure 3.1: Form of the final assembly

For homing implementation, two potential solutions were considered: Hall effect sensors and optical gates. Hall effect sensors offer the advantage of angle sensing, allowing correction for positional drift during operation; however, they require precise alignment. If the orthogonal Hall effect sensor is not perfectly placed along the axis of rotation, calibration becomes necessary [**hall**]. While feedback would be beneficial, the microcontroller already tracks each step taken by the motor during normal operation, making it possible to determine the platform's position purely in software. Thus for simplicity and ease of integration, optical gates were selected.

The system is controlled by an ESP32 microcontroller. The ESP32-C6 version was chosen due to the author's extensive experience with this particular model. However, since the system does not require specialized peripherals or high processing power, any ESP32 variant would be sufficient.

3.3 Platform Software Realization

To maximize efficiency in processing commands and ensure accurate stepper motor control, the program workflow is divided into three distinct layers, as illustrated by figure 3.2.

The commonly used two-component architecture—where one component handles communication/command parsing and the other manages execution—was deemed unsuitable for this use case. Such an approach would complicate integration of programming interface and require just-in-time processing of commands, which could lead to performance issues.

In the chosen architecture, the degree of abstraction decreases with each successive layer, simplifying processing at each step. This design allows the final layer to operate with maximum efficiency, where transition from one command to the next is primarily limited by the inertia of stepper motors and not by the software.

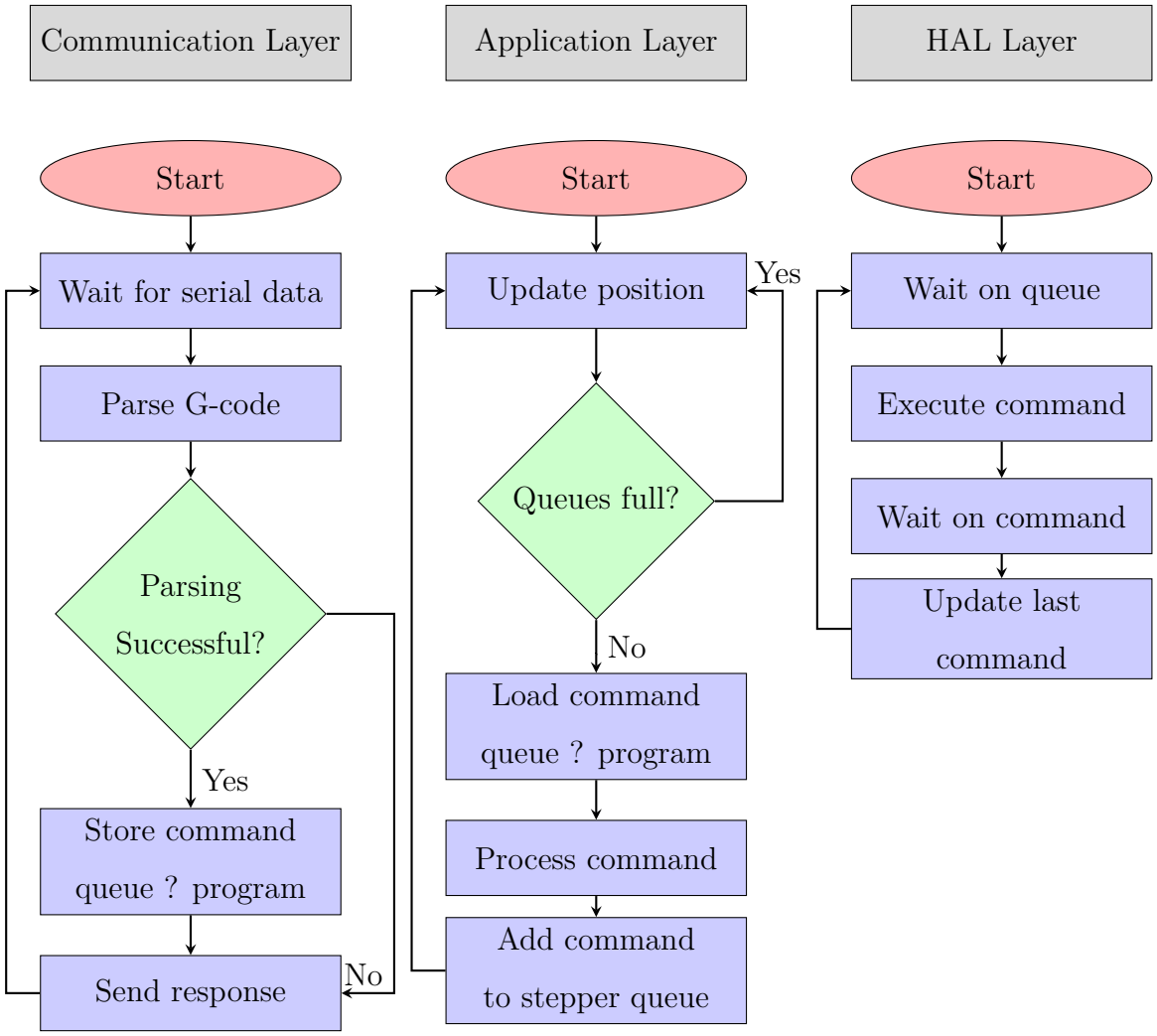


Figure 3.2: Program flow diagram

3.3.1 Communication layer

The communication layer manages incoming data over the serial line, with efficient handling facilitated with the aid of RTOS queues. Upon receiving data the text string is parsed and either pushed to a queue or added to programm declaration, in case we are currently declaring program.

Immediately after parsing, a response is send to the user confirming whether the command was parsed correctly or not. However, as the communication layer does not a can not check command within context of all previous commands, it is possible that command will be parsed correctly but its execution will fail in the application layer.

3.3.2 Application layer

The application layer performs two primary functions: tracking the current device position and scheduling commands to be sent to stepper motors. Aside from current position the program also keeps track of the end position of the last scheduled command. Thanks to this the application layer make necessary calculations to facilitate absolute positioning and enforce movement limits.

A key departure from standard G-code interpreters, like [**duet**], is how the

platform handles single-axis move commands. When a move command targets only one axis, the other axis remains free to read next command and begin its execution. If this behavior is undesirable, the user must issue commands for both axes. In relative positioning mode, a zero value results in no motion; in absolute positioning mode, the command must specify the current position to prevent movement.

This behavior is a necessary side effect of the spindle regime, which typically cannot be toggled on or off dynamically. Another consequence is the requirement for separate positioning modes for each axis. Continuous rotation prevents calculations of a move's end position, making it impossible to make calculation for absolute positioning commands – thus necessitating relative positioning. However it would be rather restrictive to force user to relative positioning on second axis, therefore the independent positioning settings.

In order to support or possible operating regimes a manual override mode was also implementing. This enables the user to manually push a move command directly to stepper queues totally skipping the application layer. Primary usecase of this mode is to allow tracking of targets or other application that require real time control of the platform. However in this regime no limits are enforced and the platform operates strictly in relative positioning mode.

3.3.3 HAL Layer

The final layer manages stepper motor control and provides the application layer with essential data for position calculations. In its loop, the program waits for the next command in the stepper queue. Upon receiving a command, it sets up execution, waits for one or both steppers to complete their movement, and then proceeds to the next command. Since limit and absolute positioning calculations are handled in the application layer whole routine remains highly efficient.

The main challenge lies in generating precise PWM signals (Used to control stepper motors drivers.) and stopping signal generation after a specific number of steps. Using the equation:

$$t_{\text{delay}}(s) = \frac{60}{2 \cdot N_{\text{steps}} \cdot s}, \tag{3.2}$$

where s is speed in RPM, N_{steps} is the number of steps (Anywhere from 200 to 1600 depending on microstepping.), and t_{delay} is the time between steps, we calculate that even at 30 RPM, the delay between output changes is 5 ms per step. With microstepping at a 2:1 ratio, this reduces to 2.5 ms – faster than lowest sleep interval on ESP32 and without sleeping the RTOS watchdog will trigger. Therefore, signal generation must leverage specialized microcontroller peripherals.

The ESP32 platform offers two options: Remote Controlled Transceiver (RMT) and Motor Control Pulse Width Modulation (MCPWM) combined with Pulse Counter (PCNT). While RMT allows smooth PWM frequency adjustments, it has several drawbacks. Such as the fact that generating a specific number of pulses is supported only on newer ESP32 models [**gitRMT**], synchronization is restricted to its proprietary API, and there is no straightforward way to track progress during a move [**espRMT**].

For these reasons, MCPWM and PCNT were chosen. MCPWM handles pulse generation, while PCNT counts steps, enabling easy synchronization, continuous rotation, and a robust API for step tracking [**espPCNT**]. The only limitation is the PCNT's 15-bit counter, which caps the maximum steps per move at 32.767.

Performance of the HAL Layer

Table 3.1 illustrates the stability of PWM generation by the MCPWM module at various speeds. Measurements were conducted using a Saleae Logic Pro 16 logic analyzer, with no microstepping enabled.

The results show that frequency deviation is minimal, though the generated speed is consistently marginally faster than the target, and the error increases slightly with higher speeds. Nevertheless, when measuring time of 24,000 steps at 120 RPM, the relative error in time duration (or speed) was only $\epsilon = -0.004\%$, demonstrating excellent accuracy.

Table 3.1: Stability of PWM generation

RPM	f_{desired} (Hz)	f_{low} (Hz)	f_{high} (Hz)	f_{avg} (Hz)
10	33.334	33.334	33.334	33.334
30	100	100	100.003	100.002
60	200	200	200.01	200.004
120	400	400	400.02	400.007

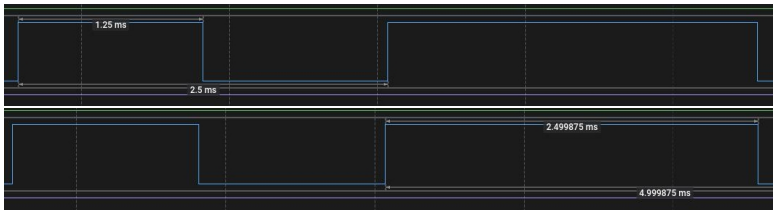


Figure 3.3: Moment of change between commands (120RPM ⇒ 60RPM)

An attempt was made to also measure the delay between switching commands, displayed in figure 3.3. The results indicate that the delay between commands is imperceptible. Similar outcomes were also observed for other command combinations.

This demonstrates the efficiency of the HAL layer in managing stepper motor control and transitioning seamlessly between commands. As long as stepper queues are supplied with commands in advance, the platform can operate without noticeable interruptions. Most importantly, the platform's timely and predictable behavior ensures that mathematical corrections to the radar data can be applied accurately.

3.4 MATLAB Control Interface

In its basic operating mode, platform control is entirely independent of the radar data processing. Or to be more precise, while radar data processing may apply corrections based on platform speed, acceleration, and position, it does not send

any commands to the platform itself. As a result, the control flow of the platform remains simple – users upload a program to the microcontroller and start its execution. To manage programs and provide a means of reading platform diagnostics, a simple MATLAB application was developed.

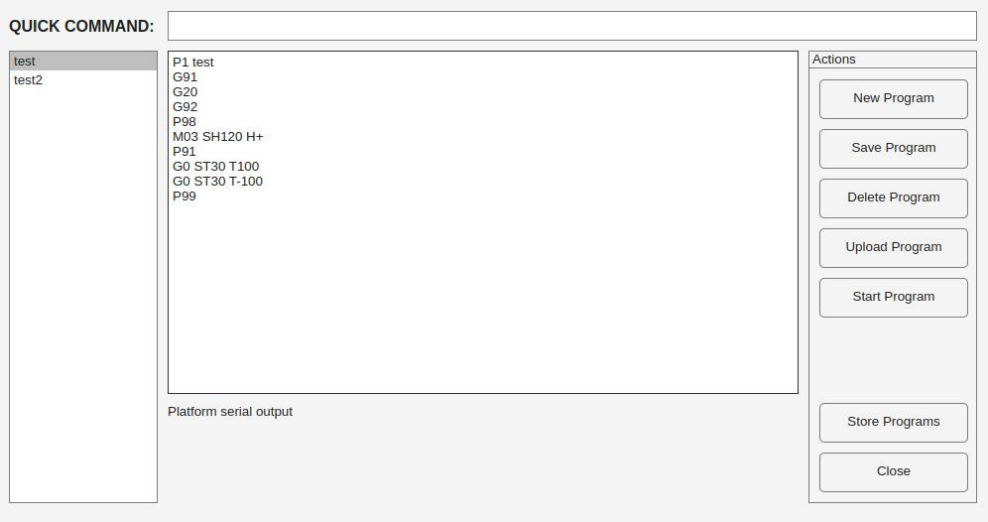


Figure 3.4: GUI of the platform control

After connecting to the platform (by selecting the correct port and baud rate in the preferences menu and clicking the "Connect" button in the main window), the user can navigate to the platform control window, as shown in Figure 3.4. On the left side, a sidebar displays a list of all stored programs. Clicking on any of these programs loads it into the editor window, where the user can edit it or send it to the platform using the buttons on the right.

Once the program is sent, execution can be initiated either manually by sending a start command from the `QUICK COMMAND` box—or by pressing the "Start" button. Using the "Start" button ensures that platform queues are empty and that the platform is homed before execution begins. To monitor feedback from the platform, users can refer to the text window at the bottom of the screen.

4. Radar Data Processing

TODO: target detection at least using CFAR

Conclusion

TODO

List of Figures

List of Tables