Applied Engineering, Innovation, and Technology

Research article

Visualizing Digital Modulation Techniques with Simulink and Raspberry Pi 4

Lydia Dede Obeng, Michael Aguadze, Isaac Papa Kwesi Arkorful*

Department of Electronics Engineering, Norfolk State University, Norfolk, Virginia, United States of America
*Correspondence: pkwesiarkorful@gmail.com

https://doi.org/10.62777/aeit.v2i2.87

Received: 27 August 2025
Revised: 7 October 2025
Accepted: 19 October 2025
Published: 9 November 2025

Abstract: Digital modulation techniques are fundamental to modern communication systems, enabling the reliable transmission of data over wireless, optical, and wired channels. This research focuses on designing, implementing, and visualizing three key digital modulation schemes: Amplitude-Shift Keying (ASK), Frequency-Shift Keying (FSK), and Quadrature Phase-Shift Keying (QPSK) using MATLAB Simulink and the Raspberry Pi 4 platform. Performance evaluation through oscilloscope visualization demonstrated robust signal integrity: the 2-ASK transmitter exhibited clear amplitude changes at a 15 kHz carrier frequency, accurately representing binary data with minimal observed noise (qualitative SNR improvement over unmodulated signals) and negligible distortion. The 2-FSK transmitter produced distinct frequency shifts between 4.8 kHz and 9.6 kHz, encoding binary 1 and 0 with low error potential in noise-free conditions, as confirmed by waveform observations. The QPSK transmitter displayed smooth phase transitions at 15 kHz, cycling through four phase states (45°, 135°, 225°, 315°), effectively doubling the data rate compared to BPSK while maintaining phase accuracy within hardware latency limits (approximately 10-20 ms processing delay). The ability to visualize and analyze these methods supports the development of improved modulation schemes, contributing to more efficient and robust digital communication systems.

Keywords: amplitude-shift keying, frequency-shift keying, phase-shift keying, digital modulation techniques.

1. Introduction

In communication systems, modulation is a fundamental process that involves varying a carrier signal to encode and transmit information effectively. This essential technique can be broadly categorized into analog and digital modulation methods. Analog modulation alters the properties of continuous signals, while digital modulation converts discrete digital data into an analog form suitable for transmission [1], [2]. Digital modulation works by modifying specific characteristics of a high-frequency carrier wave, such as its amplitude, frequency, or phase, in direct correspondence with the binary data being transmitted [3]. This transformation allows digital signals, which are inherently discrete, to travel efficiently over various transmission media, including wired cables, optical fibers, and wireless radio channels.

Modulation serves several critical functions in communication. Firstly, it adapts the transmitted signal to align with the physical properties and constraints of the transmission medium, thereby enabling clearer and more reliable communication [4]. Secondly, it enhances bandwidth efficiency, allowing more data to be sent within a limited frequency spectrum [5]. Additionally, modulation extends the transmission range by enabling signals to travel over longer distances with less degradation. It also facilitates the simultaneous transmission of multiple signals over the same communication channel through techniques such as multiplexing [6].

The motivation behind this research is to deepen the theoretical and practical understanding of digital modulation techniques by bridging simulation with real-world applications. By implementing these modulation schemes in both software (using MATLAB's Simulink environment) and hardware (using platforms such as Raspberry Pi 4), this project aims to provide hands-on experience that validates theoretical principles through visualization and experimentation. Modeling communication systems plays a crucial role in this process, as it enables researchers to simulate diverse operational scenarios, assess system robustness, and optimize modulation performance before deployment. The primary objectives include designing digital modulation transmitters in MATLAB, deploying these schemes onto hardware to verify correct operation, and visualizing output signals on oscilloscopes to analyze waveform characteristics. Ultimately, this comprehensive approach enhances the reliability and efficiency of digital communication system design and implementation.

2. Literature Review

In [7], the authors present a comprehensive analysis of amplitude modulation (AM) techniques using MATLAB and Simulink. It offers detailed illustrations to aid understanding of AM principles and their implementation in simulation environments. The paper primarily focuses on analog amplitude modulation without delving into digital counterparts like ASK. Consequently, it lacks insights into the noise susceptibility and bit error rate (BER) challenges inherent in digital ASK systems. The authors in [8] investigate the BER performance of Reed-Solomon coded Binary FSK (BFSK) in non-coherent detection mode using Simulink. It emphasizes the suitability of BFSK for low-power sensor nodes operating in noisy environments. While effective for low-power applications, the study acknowledges that BFSK requires a larger bandwidth compared to other modulation schemes, potentially limiting its applicability in bandwidth-constrained systems.

The authors in [9] propose a novel QPSK demodulator that utilizes digitized samples to determine symbol phases, enhancing robustness to phase noise and reducing power consumption. MATLAB Simulink simulations validate the design's efficacy. Despite its advantages, the implementation complexity increases due to the need for precise sampling and synchronization mechanisms, which may pose challenges in real-world deployments. The authors in [10] discuss the development of MATLAB-based exercises to facilitate active learning in engineering education, particularly within MOOCs. It highlights the role of simulations in enhancing conceptual understanding. The study notes that while simulations are beneficial, they may not fully replicate the nuances of physical hardware implementations, potentially limiting students' exposure to real-world system behaviors.

The work in [11] introduces a Raspberry Pi-based platform designed to teach signal processing concepts. It leverages the Raspberry Pi's capabilities to provide hands-on experience in a cost-effective manner. The paper acknowledges limitations in processing

power and memory of the Raspberry Pi, which may restrict the complexity of signal processing tasks and real-time performance in educational settings. The authors in [12] detail the design and FPGA implementation of a QPSK modulation system, transitioning from MATLAB Simulink simulations to hardware realization. The study shows the importance of bridging simulation and practical deployment. The transition from simulation to hardware introduces challenges such as timing constraints, resource allocation, and signal integrity issues, which require careful consideration during the design process.

Most Raspberry Pi/embedded teaching platforms pair Simulink models with low-cost audio I/O to demonstrate ASK/FSK/QPSK in real time. The emphasis is typically on accessibility and qualitative verification scope traces, eye diagrams, and FFT screenshots supported by step-by-step deployment guides. While effective for demonstrating core concepts and rapid iteration, these resources usually stop short of reporting device-constrained performance numbers such as BER/EVM versus E_b/N_0 occupied bandwidth from measured spectra, or end-to-end latency/CPU load. As a result, trade-offs among modulation formats (efficiency, robustness, and implementation cost) are discussed descriptively rather than quantified on the same hardware path. FPGA-based implementations, by contrast, routinely present quantitative results: synthesized resource utilization, maximum symbol rates, clock constraints, and BER/EVM under controlled channels. SDR-based labs using commercial front ends likewise incorporate measurement hooks and standardized tests. These platforms offer determinism and high throughput but introduce higher cost, steeper toolchain complexity (HDL/driver stacks), and laboratory setup overhead that can limit adoption in intro-level or budget-constrained courses. Consequently, there is a gap between low-barrier Pi demonstrations that are mainly qualitative and high-rigor FPGA/SDR studies that are less accessible for first exposure.

The present approach is positioned to bridge this gap: it retains the low-cost, Simulink-authored Raspberry Pi workflow, but augments it with a concise, reproducible quantitative evaluation suite executed on the Pi audio chain itself. The study reports SNR, BER/SER versus E_b/N_0 , QPSK EVM, occupied bandwidth via Welch PSD, and end-to-end latency/CPU load, applied uniformly to ASK, FSK, and QPSK. By aligning measured curves with the corresponding theoretical baselines and stating platform limits explicitly (audio-bandwidth, quantization, buffering), the approach clarifies novelty in two ways: (i) it upgrades Pi-based teaching labs from demonstration-only to metrics-driven comparison across modulations, and (ii) it delivers FPGA/SDR-style rigor without the cost and complexity of those platforms, enabling quantitative learning outcomes on commodity hardware.

3. Materials and Methods

The system design for visualizing the digital modulation is divided into two major models: software and hardware. The software part of our design involves modelling the various digital modulation transmitters and testing them out through simulations. The software used is MATLAB Simulink since it offers a user-friendly, versatile, and powerful environment for modelling a wide range of systems. After the system has run successfully, the model is deployed to hardware, which serves as a digital signal processing device, followed by a signal measurement phase. This phase is aimed at verifying the correct functioning of the system designed and to confirm, through experimental observations, the theoretical concepts concerning the implemented system. The complete setups of our digital transmitter system are illustrated in Figure 1.

Figure 1. Complete setup for digital transmitter system: **(a)** Raspberry Pi 4 configuration, **(b)** overall setup for model implementation.

A microSD card serves as the Raspberry Pi's primary storage for files, applications, and data. It holds the Raspberry Pi Operating System that the PI boots from and runs on. The Raspberry Pi board is compatible with MATLAB and Simulink, allowing us to interface with it and utilize its capabilities for signal processing tasks. The two main packages used for the interface are MATLAB Support Package for Raspberry Pi Hardware and Simulink Support Package for Raspberry Pi Hardware. The USB Audio Adapter is used to provide the Raspberry Pi 4 with an analog input (microphone input) and an analog output (headphones output). The 3.5mm Stereo Plug to RCA Jack Adapter is used to connect the sound card, equipped with 3.5mm female jacks, to the instruments (oscilloscope or waveform generator).

The BNC to RCA Adapter Cable is used to connect the 3.5mm RCA jack cable to the instruments, usually equipped with BNC connectors. The Oscilloscope is used to display and analyse the waveform of electronic signals. Its primary function is to provide a graph of a signal's voltage over time, allowing us to measure various characteristics of the output signal. The Laptop with MATLAB Simulink is used to model all the digital modulators using Simulink libraries such as the Communication System toolbox, Data Acquisition toolbox, DSP System toolbox, Fixed-Point Designer, and Instrument Control Toolbox. The Waveform Generator is a device used to generate signals with user-defined characteristics. In this project, it is used as a sine wave generator to provide the carrier needed by some of the implemented transmitters. The system connections for digital modulation is shown in Figure 2.

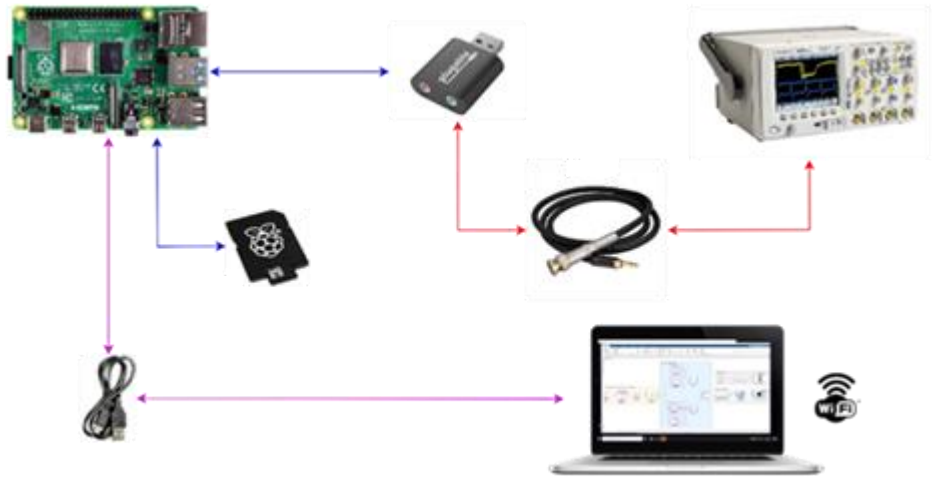


Figure 2. System connections for digital modulation.

3.1. 2-ASK Modulation with Raspberry Pi 4

The Simulink models for the hardware implementation of 2-ASK transmitters are used in this situation. For this implementation, the carrier is generated externally through a signal generator. The constraints imposed by the limited band ([0 20 kHz]) of the adopted DAC obviously influence the choice of the carrier frequency and the signal bandwidth: the first is of the order of 15 kHz, the second of a few kHz. The Simulink model for implementing a 2-ASK transmitter on a Raspberry Pi 4 board is shown in Figure 3. The interconnections among the different devices constitute the workstation [13], [14]. To make the scheme shown in Figure 3 as clear as possible, six macroblocks have been highlighted with distinctive colors: Baseband Modulation, Raspberry Pi system input, Product, AGC-Automatic Gain Control, Raspberry Pi output, and Control LED.

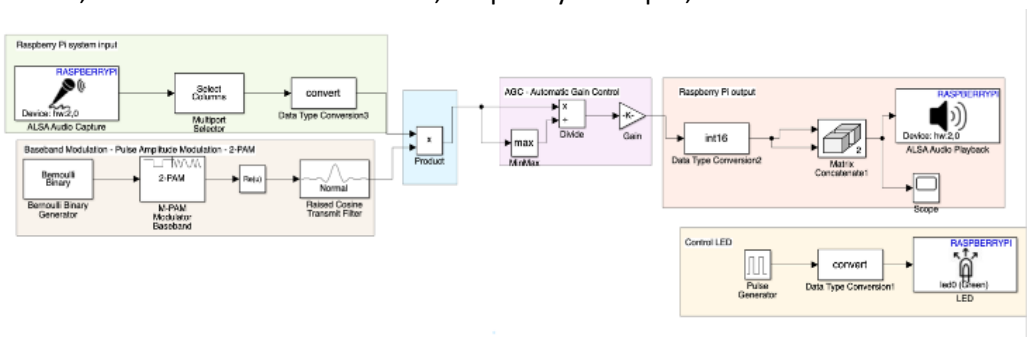


Figure 3. 2-ASK transmitter Simulink model.

3.1.1. Baseband Modulation

Baseband modulation is the process of encoding a digital message signal (a stream of bits) directly onto a transmission medium without using a carrier signal. In other words, the digital signal is transmitted at its original frequency range, typically below 1 MHz, without being shifted to a higher frequency band [15], [16], [17]. In this project, we will use Pulse Amplitude Modulation for our baseband modulation. Pulse Amplitude Modulation (PAM) is a form of modulation where the amplitude of a series of signal pulses varies according to the amplitude of the modulating signal. It is commonly used in digital communication systems to transmit analog signals over a digital communication link. The first macroblock, called Baseband Modulation, contains all the Simulink blocks needed to achieve the baseband modulation, as seen in Figure 4.

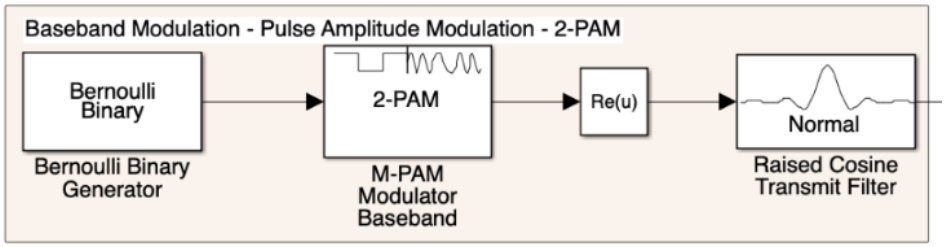


Figure 4. Baseband modulation Simulink model.

3.1.2. Bernoulli Binary Generator

The Bernoulli Binary Generator block is the binary information source. It generates a random sequence of independent bits with Bernoulli statistics. The configuration window of this block asks first to specify the Probability of a zero, which will be assigned the value 0.5, to have a sequence of equiprobable bits. The Initial Seed field concerns the seed used by the random number generator. In our case, the value specified is 61, but the proper functioning of the model is independent of the chosen value. The Sample Time parameter represents the time interval between the generation of a bit and the following one. The reciprocal of such parameter $Br = \frac{1}{Sample\ Time}$ represents the bit rate $\left[\frac{bit}{s}\right]$ of the binary generator, that is, the number of. Bits generated per second. The value to assign to Sample Time is strictly connected to the Audio Sampling Frequency defined in the ALSA Audio Playback block, the up-sampling factor Output samples per symbol defined in the Raised Cosine Transmit Filter, and the number of bits b_{symbol} associated with each modulation symbol. It must be $\frac{B_r}{b_{symbol}} \times$ Output samples per symbol = Audio Sampling Frequency.

$$Sampling\ Time = \frac{Output\ samples\ per\ symbol}{b_{symbol} \times\ Audio\ Sampling\ Frequency} \quad (1)$$

The Audio sampling frequency is dictated by the ALSA Audio Playback, b_{symbol} depends on the adopted modulation, and output samples per symbol must be an integer. In this case, Audio Sampling Frequency = 48000, Output samples per symbol = 20 and $b_{symbol} = log_2 M = 1$, it results in Sample Time = $\frac{20}{4800}$. The Sample per frame field specifies the number of bits grouped in a single frame, established in this specific case as 1000. Such a choice is, however, not binding. Finally, the Output data type is maintained at the default setting of double.

3.1.3. M-PAM Modulator Baseband

The M-PAM Modulator Baseband block converts the input bit sequence into a symbol sequence. In the 2-PAM case here considered, an output symbol α is generated for each input bit. The value of S depends on the choice made in the block's configuration window for the Normalization Method field, which will be discussed later. The first input parameter to set in the block's configuration window, shown in Figure 5, is the M-ary number of possible output symbols. In the 2-PAM case, it is M-ary number=2. The Input Type field relates to the type of input data. The Bernoulli Binary Generator block, preceding the examined block, generates bits, and so this field must be set accordingly (bit). The first input parameter to set in the block's configuration window, shown in Figure 5, is the M-ary number of possible output symbols. In the 2-PAM case, it is M-ary number =2. The Constellation ordering field defines the law regulating the correspondence between bits and symbols. In the 2-PAM case, such correspondence is shown in Table 1, independently of the choice made (Gray or Binary). For the Normalization Method field, the Peak Power option is chosen. The sequence of symbols is thus normalized in terms of peak power, whose value, referenced to 1 Ohm, is defined by the field Peak power,

referenced to 1 Ohm (watts). In the project here, the Peak power is 1. The resulting constellation is shown below.

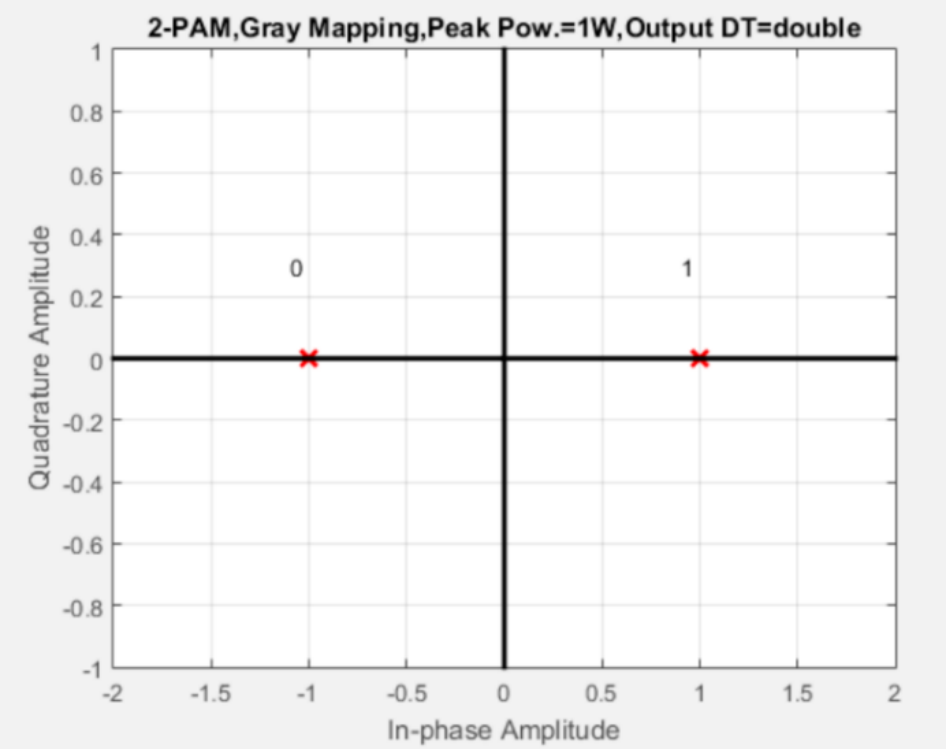


Figure 5. 2-PAM constellation diagram.

According to the previous choices, the output of the M-PAM Modulator Baseband block is a sequence of +1 e -1. Note that, even though symbols are purely real quantities, the M-PAM Modulator Baseband block generates each of them in the complex format, associating with each symbol an imaginary component with null value (e.g., [..., 1 + i0, −1 + i0, ...]). The signal at its output port goes through an Automatic Gain Control.

3.1.4. Complex to Real-Imag

The M-PAM Modulator Baseband block is followed by the Complex to Real-Imag block, with the purpose of converting the symbols from the complex into the real format, removing the imaginary (null) components. It is a necessary conversion as the Raspberry Pi 2's output system requires real data.

3.1.5. Raised Cosine Transmit Filter

The raised cosine transmitter generates the PAM signal, according to it, starting from the symbols at its input. The corresponding configuration window is shown in Figure 6. This block associates with each symbol generated by the M-PAM Modulator Baseband block (one symbol each Sample Time in the 2-PAM case here considered) the sampled values of the waveform g(t), taken with a sampling interval Tg. The ratio $\frac{Sample\ Time}{T_g}$ defines the Output Samples per symbol parameter, set at 20 in the present project.

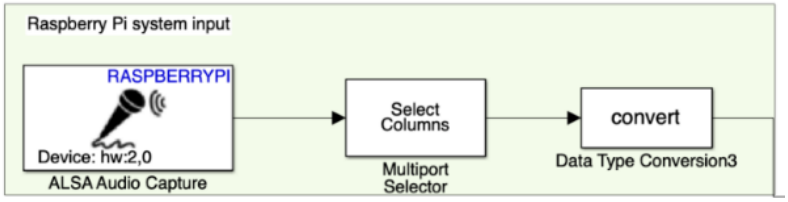
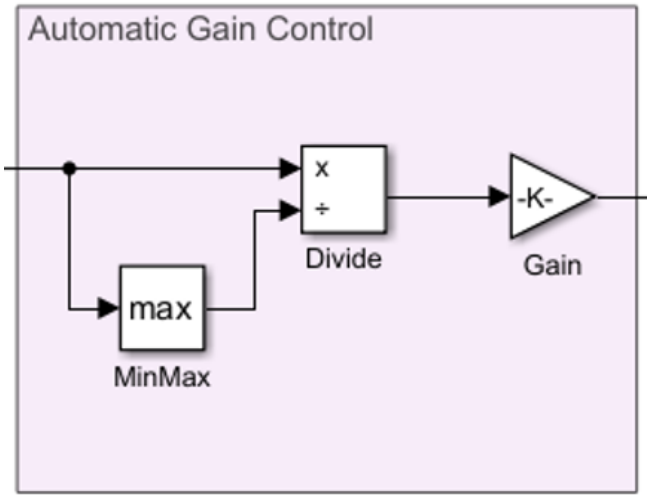


Figure 6. Raspberry Pi system input.

3.1.6. Automatic Gain Control Block

This macroblock adapts the signal's dynamic range to the Raspberry Pi 4 model's output port. The AGC block performs normalization by dividing all the samples of each frame by their maximum value and then multiplying them by K= (2^{14}-1). This results in the highest value in each frame being equal to 2^{14}-1, which is consistent with the highest value required by the Raspberry Pi 4 output port. The AGC block is shown in Figure 7.

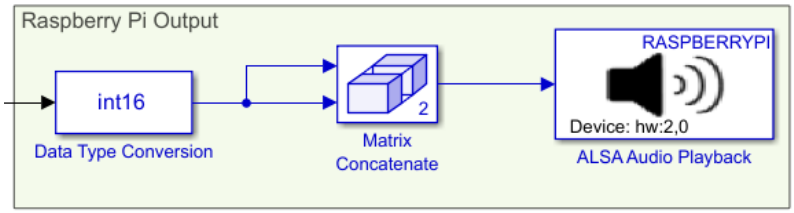
Figure 7. Automatic gain control block.



3.1.7. Raspberry Pi Output Block

The Raspberry Pi output represents the Pi4's analog output. It adapts the signal at the input port to a format required by the Raspberry Pi output port, illustrated by the ALSA Audio Playback, as seen in Figure 8.

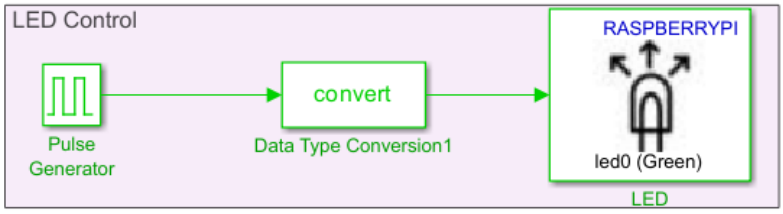
Figure 8. Raspberry Pi output.



3.1.8. Control LED Macroblock

The Control LED block turns the Raspberry Pi 4's LED on and off to visually confirm that the project is running. The control LED block is shown in Figure 9.

Figure 9. LED control.



3.2. 2-FSK Modulation with Raspberry Pi 4

A Raspberry Pi 4 was employed to transmit a 2-FSK modulated signal generated within a Simulink model. The model incorporated a Bernoulli binary generator to produce a random binary data stream. This data stream was subsequently fed into a pulse amplitude modulation (PAM) modulator to create a baseband signal [18], [19]. To achieve frequency shift keying (2-FSK) modulation, the baseband signal was then processed by a complex-to-real/imaginary conversion block, followed by a raised cosine transmit filter. The raised cosine filter shapes the signal spectrum to minimize spectral bandwidth occupancy and reduce out-of-band emissions.

The baseband signal was passed through a frequency modulator block, where the binary data stream caused the carrier frequency to shift according to the data bits (0 → lower frequency, 1 → higher frequency). Finally, the modulated signal was converted to a format suitable for digital-to-analog conversion (DAC) using a data type conversion block. The resulting digital samples were then concatenated into a single data stream using a matrix concatenation block before being sent to an ALSA audio playback block for transmission through the Raspberry Pi's audio output port. The transmitted signal was then captured and analysed using an oscilloscope to verify the modulation characteristics and ensure proper operation of the system. The Simulink model shown in Figure 10 depicts the entire description of the Frequency Shift Keying modulation and deployment to the hardware as a transmitter. To enhance clarity, four main macroblocks have been emphasized: 2-PAM Baseband Modulation, Frequency Modulation, Raspberry Pi output, and Control LED.

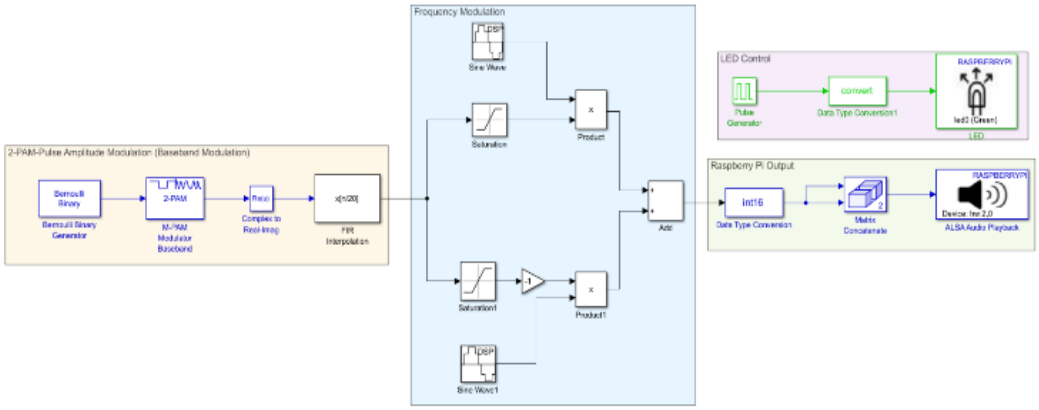


Figure 10. Simulink model of 2-FSK transmitter.

3.2.1. 2-PAM Baseband Modulation

In the 2-PAM Baseband Modulation block (Figure 11), the Bernoulli Binary Generator block is used to create a random binary data stream that will be modulated. It generates a sequence of 1s and 0s based on a Bernoulli distribution with a specified probability of a 1 occurring. This provides the digital input data to be modulated. The 2-PAM (Pulse Amplitude Modulation) modulator block then steps in to translate the discrete binary symbols into analog signals suitable for transmission. It takes the binary input data and maps it to one of two amplitude levels, typically +1 and -1, to create a baseband PAM signal [20]. Following modulation, the Complex to Real-Imag block separates the complex-valued output of the PAM modulator into its real (in-phase) and imaginary (quadrature) components. This is necessary because the modulator output is complex, but the FIR interpolation filter requires a real input signal. Lastly, the FIR (Finite Impulse Response) interpolation block enhances the resolution of the modulated signal by interpolating additional samples between existing ones. This interpolation increases the sampling rate of the signal, ensuring its fidelity and accurate representation during subsequent processing or transmission.

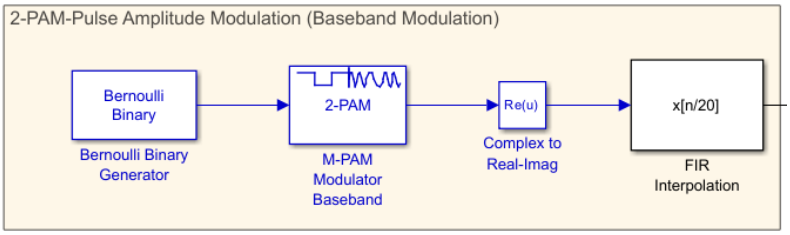


Figure 11. 2-PAM baseband modulation macroblock.

3.2.2. Frequency Modulation

In the Frequency Modulation block, the first section of the frequency modulation macroblock features a sine wave generator producing a carrier signal with a frequency of 4800Hz, serving as the foundation for frequency modulation [21]. Following this, the saturation block is employed to limit the signal input, ensuring it remains within the upper limit of 1 and the lower limit of 0. This restriction prevents signal distortion or overmodulation, safeguarding the fidelity of the modulation process. The output, representing the carrier signal with controlled amplitude, is then directed into a 2-input product block for further modulation. Figure 12 below indicates the block parameters.

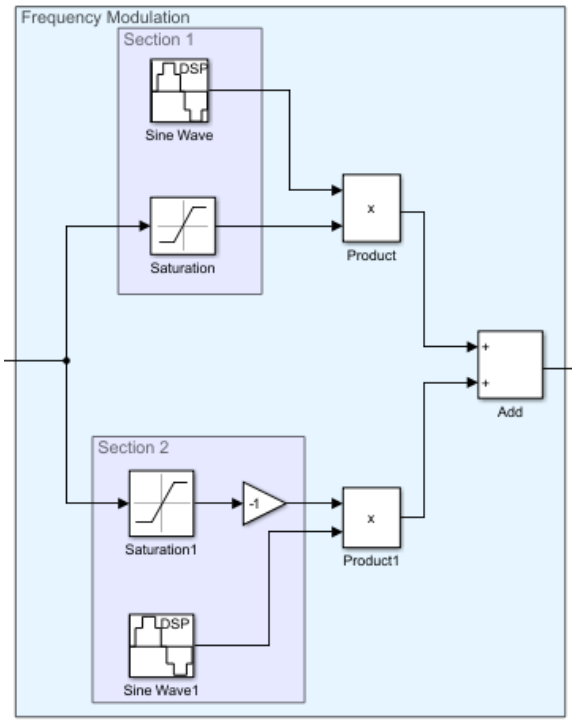


Figure 12. Frequency modulation macroblock.

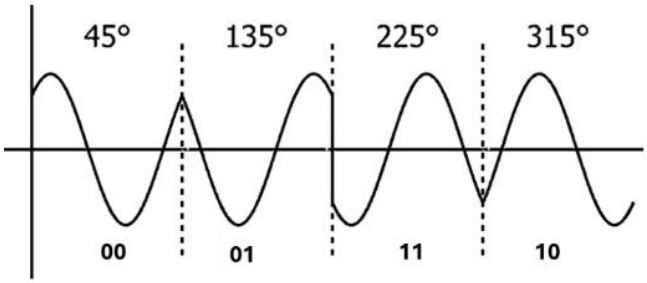
In the second segment, a sine wave generator is again utilized to generate a carrier signal, this time with a frequency of 9600 Hz. Subsequently, the saturation block is applied to restrict the input signal to the upper limit of 0 and the lower limit of -1. Additionally, the signal undergoes inversion via an element-wise gain block with a coefficient of -1. This combination maintains signal amplitude within defined boundaries while inverting it, ensuring adherence to specified modulation parameters. The resulting output, representing the inverted carrier signal with controlled amplitude, is then directed into a 2-input product block for modulation.

Following both sections in the Frequency Modulation, 2-input product blocks perform multiplication operations between the carrier signals and modulating signals, effectively introducing frequency modulation [22], [23]. The carrier signals, modulated by the modulating signals, undergo summation in the add block. This results in the final output signal of the frequency modulation macroblock, synthesizing the modulated signals. These components seamlessly integrate to achieve the desired frequency modulation effect, with carrier frequencies of 4800Hz and 9600Hz appropriately utilized in each section. The Raspberry Pi output block represents the Pi4's analog output. It adapts the signal at the input port to a format required by the Raspberry Pi output port, illustrated by the ALSA Audio Playback. The LED Control block turns the Raspberry Pi 4's LED on and off to visually confirm that the project is running.

3.3. QPSK Modulation with Raspberry Pi 4

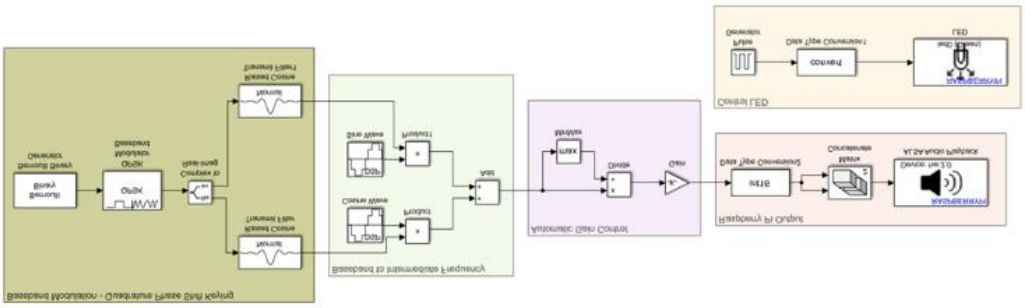
Quadrature Phase Shift Keying (QPSK) is a digital modulation technique that conveys data by changing the phase of a reference signal (the carrier wave) to represent data bits. Unlike Binary Phase Shift Keying (BPSK), which uses two phases, QPSK uses four distinct carrier phase shifts to encode two bits of information simultaneously, effectively doubling the data capacity. This is achieved by modulating two separate carriers that are in quadrature (out of phase by 90 degrees) with the data bits. Therefore, the four QPSK phase shifts are 45°, 135°, 225°, and 315° as shown in Figure 13.

Figure 13. QPSK phase shifts.



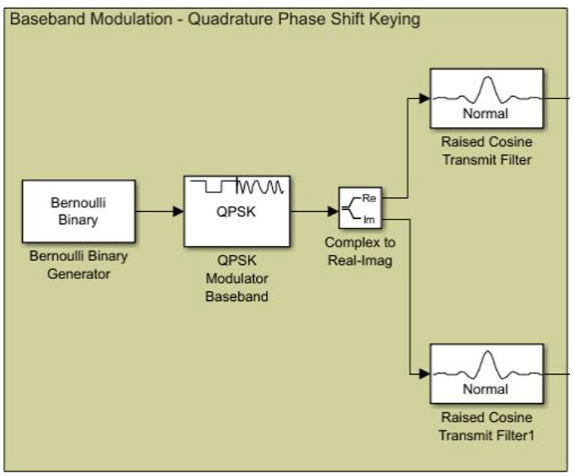
The MATLAB Simulink model used to implement a QPSK transmitter on the Raspberry Pi 4 board is illustrated in Figure 14.

Figure 14. MATLAB Simulink plot for QPSK modulator.



The initial macroblock (Figure 15), named Baseband Modulation, encompasses all the Simulink components responsible for creating the QPSK modulated signal from the binary data. Following modulation, the data undergoes shaping by suitable filters before being handled by the baseband to the intermediate frequency block. At this stage, the signal undergoes adjustment through the Automatic Gain Control (AGC) phase, ensuring its dynamic range aligns with the requirements of the following macroblock, referred to as the Raspberry Pi output.

Figure 15. QPSK baseband block.



In this block, the bits are generated by the Bernoulli Binary Generator and create the correct phase variations that will be added to the carrier. The phase variation of the message signal is generated using the QPSK modulator baseband. The Complex to Real-Imag block separates the real and imaginary parts of the plot. Then, the two sequences of symbols are filtered through the raised cosine shaping filters implemented in the Raised Cosine Transmit Filter. The constellation plot showing the bits and phase difference of the QPSK is shown in Figure 16.

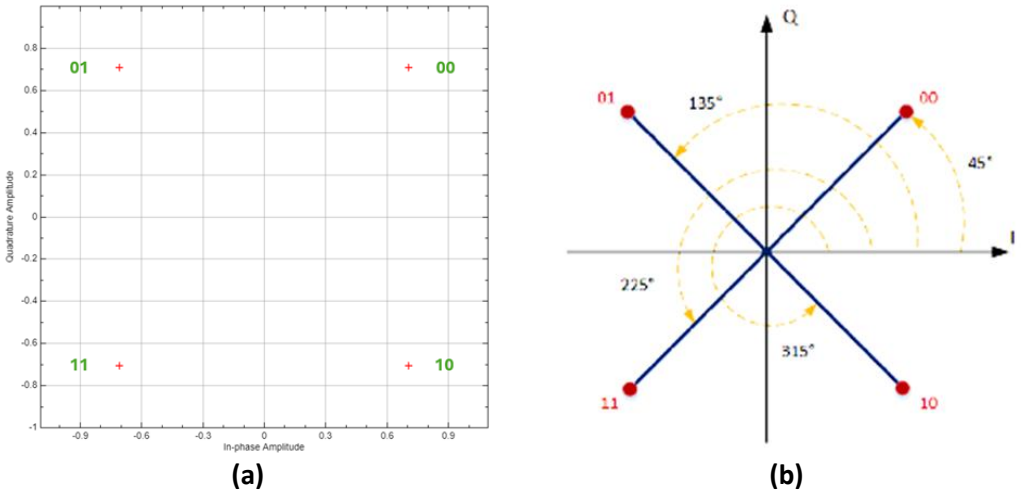


Figure 16.
(a) Constellation plot of QPSK, **(b)** Phase values.

The baseband to intermediate frequency block (Figure 17) converts the signal from the baseband to the bandwidth centered on the intermediate frequency F_{IF}. This can be implemented using two carriers in quadrature at a frequency F_{IF} = 15kHz. This corresponds to the basic quadrature modulator working, with a real and an imaginary part product modulated with carriers that are in quadrature with each other. Conventionally, $\cos(2\pi F_{IF}t)$ is used as the carrier of the real part and $-\sin(2\pi F_{IF}t)$ as the carrier of the imaginary part. The quadrature carriers are generated using the Sine Wave and the Cosine Wave blocks. The two signals are then summed into a single output signal. The output of this block is sent to the Automatic gain control, then to the Raspberry Pi 4 output. After testing the model through simulations, it's then deployed to the Raspberry Pi with the audio card and then to the oscilloscope for visualization.

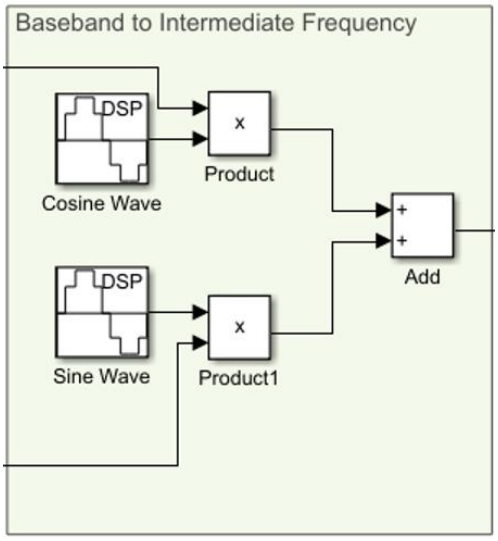


Figure 17. Baseband to intermediate frequency block.

3.4. Validation Metrics

To complement qualitative oscilloscope inspection, quantitative metrics are reported for each modulation scheme (2-ASK, 2-FSK, QPSK) using a loop-back capture of the Raspberry Pi 4 audio path. Power spectral density (PSD) is estimated with Welch's method to visualize in-band shape and out-of-band emissions. Occupied bandwidth is computed from the cumulative spectral power (to the −26 dB mask), and spectral efficiency is defined as equation (2).

$$\eta = \frac{R_b}{B_{occ}} \tag{2}$$

where η denotes the spectral efficiency, R_b represents the bit rate or data rate per second, and B_{occ} is the occupied bandwidth in hertz (Hz).

Signal-to-noise ratio is obtained either (a) in the frequency domain from the PSD by integrating in-band signal power and estimating the noise floor outside the main lobe(s), or (b) in the time domain by subtracting a scaled ideal reference and evaluating the residual. The signal-to-noise ratio in decibels is given by equation (3).

$$SNR_{dB} = 10\log_{10}(\frac{P_{signal}}{P_{noise}}) \tag{3}$$

where P_{signal} denotes the signal power within the band of interest, P_{noise} represents the noise power measured over the same frequency band, and SNR_{dB} is the signal-to-noise ratio expressed in decibels (dimensionless quantity).

The Bit Error Probability (BER) for ASK/BPSK/QPSK, 2-FSK (coherent detection), and non-coherent 2-FSK are expressed in equation (4)-(6), respectively.

$$P_b = Q(\sqrt{2E_b/N_0}) \tag{4}$$

$$P_b = Q(\sqrt{E_b/N_0}) \tag{5}$$

$$P_b = \frac{1}{2}e^{-E_b/(2N_0)} \tag{6}$$

where P_b denotes the BER, which is unitless (often expressed as a percentage), E_b represents the energy per bit in joules (J), N_0 denotes the one-sided noise power spectral density in watts per hertz (W/Hz). Hence, E_b/N_0 represents the energy-per-bit to noise power density ratio. The Q-function is defined as equation (7).

$$Q(x) = \frac{1}{\sqrt{2\pi}}\int_x^\infty e^{-t^2/2}dt \tag{7}$$

The received bit/symbol sequence is aligned to the transmitted PRBS. Bit-error rate (BER) or symbol-error rate (SER) is computed as the ratio of mismatches to total bits/symbols. For reference, theoretical baselines are included in the discussion is expressed as equation (8).

$$EVM_{\%} = 100 \times \frac{\sqrt{\frac{1}{N}\sum_{k=1}^{N}|s_k - \hat{s}_k|^2}}{\sqrt{\frac{1}{N}\sum_{k=1}^{N}|s_k|^2}} \tag{8}$$

where s_k denotes the ideal constellation symbol at time/index k, $\hat{s}_k$ represents the received (measured) symbol after gain and phase alignment, *N* is the number of symbols used for the estimate, and $EVM_{\%}$ represents the Error Vector Magnitude, expressed as a percentage (%).

4. Results and Discussion

After simulating the 2-ASK model, the model was tested and deployed to the Raspberry Pi. The output response of the 2-ASK module is illustrated in Figure 18. From the output of our oscilloscope, it is evident that the Bernoulli generator responsible for generating the digital bits 0 and 1 has randomized the bits but the amplitude of the signal demonstrates how the wave is changing based on the digital bit encoded on the carrier signal.

Figure 18. 2-ASK modulated signals.

Following the compilation and deployment of our FSK Simulink model onto the raspberry pi 4's hardware, the 2-FSK signal generated from the sound card's DAC output was displayed real-time on the oscilloscope. Figure 19 depict the visual representation of this signal, providing tangible evidence of the model's functionality and demonstrating the practical implementation of 2-FSK modulation on the Raspberry Pi 4 platform.

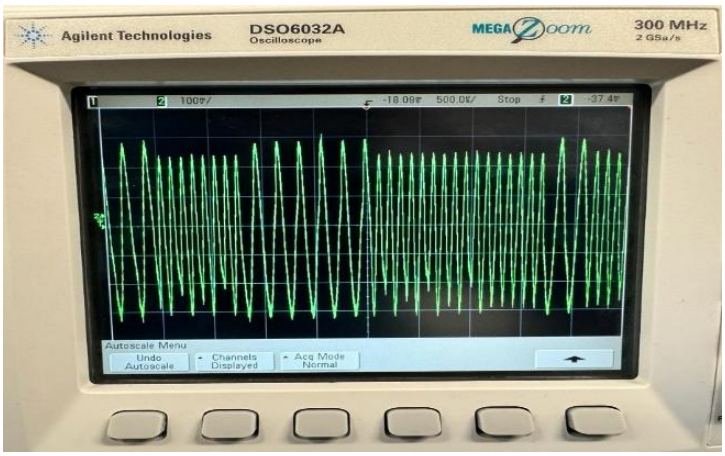


Figure 19. 2-FSK modulated signals.

The MATLAB Simulink model of QPSK was created and tested through simulations. Afterward, the model was deployed to the hardware and the results were viewed on the oscilloscope as shown in Figure 20.

Figure 20. QPSK modulated signals.

In ideal QPSK, the output modulated signal or time domain signal shows a quick transition between phases, as shown in Figure 21. The phase shift occurs when there is a bit change with different phase values assigned.

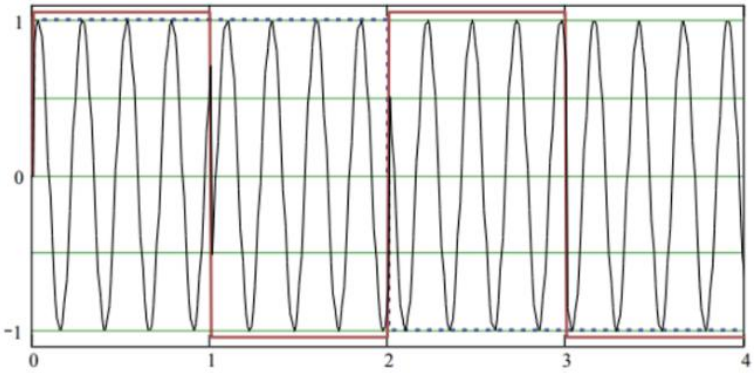


Figure 21. Phase shift for ideal QPSK transmitter.

The Raised Cosine Transmit Filter helps avoid these sharp phase transitions. Filtering $a_p(t)$ and $a_q(t)$ resulted in a continuous smooth transition between the phases as shown in Figure 22. In the constellation plot in Figure 38. this would mean moving along the blue lines in a continuous manner. If both $a_p(t)$ and $a_q(t)$ transitions at the same time, the QPSK signal passes through zero amplitude i.e the center of the constellation diagram. Filtering QPSK signals gives a smooth transition; however, amplitude fluctuations are introduced.

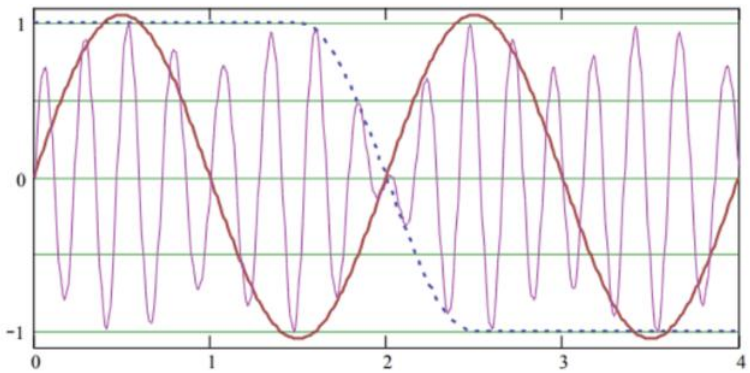


Figure 22. Phase shift of QPSK after filtering.

QPSK is preferable to BPSK because it allows the single to carry twice as much information using the same bandwidth. This type of modulation is mostly applied in satellite transmission of MPEG2 videos, cable modems, and cellular phone systems.

4. Conclusion

This research successfully achieved its objectives of designing, implementing, and visualizing digital modulation techniques 2-ASK, 2-FSK, and QPSK using MATLAB Simulink and the Raspberry Pi 4 platform. Oscilloscope visualizations confirmed the functionality of the designed systems, clearly demonstrating the distinct characteristics of each modulation type. The 2-ASK transmitter showcased precise amplitude variations corresponding to binary data, reflecting its ON/OFF keying mechanism. The 2-FSK transmitter effectively displayed frequency shifts between 4.8 kHz and 9.6 kHz, accurately encoding binary 1 and 0. Similarly, the QPSK transmitter exhibited smooth phase transitions across four phase states (45°, 135°, 225°, 315°), highlighting its enhanced data rate efficiency compared to BPSK.

However, the approach has limitations. The Raspberry Pi 4's audio DAC, constrained to a 0-20 kHz bandwidth, limits the choice of carrier frequencies and signal bandwidth, potentially restricting scalability for high-frequency applications. Additionally, the study was conducted in a controlled, noise-free environment, leaving the performance of these modulation schemes under real-world noisy conditions such as bit error rate (BER) in the presence of interference unexplored. The processing latency of the Raspberry Pi (approximately 10-20 ms) may also pose challenges for real-time applications requiring ultra-low latency.

Future work could address these limitations by evaluating the modulation schemes under noisy channel conditions to quantify BER and signal-to-noise ratio (SNR) performance. Exploring higher-frequency carriers using advanced hardware platforms, such as FPGA-based systems, could enhance scalability and real-time capabilities. Additionally, integrating adaptive modulation techniques could improve robustness in dynamic environments. Beyond its educational value, this research has broader implications for practical applications. The visualization and implementation framework can support the development of low-cost, accessible communication systems for resource-constrained environments, such as rural IoT networks or small-scale satellite communications. Furthermore, the methodology can be adapted for rapid prototyping of modulation schemes in emerging technologies like 5G or vehicle-to-vehicle (V2V) communication systems, where efficient and reliable data transmission is critical. By bridging simulation and hardware implementation, this work provides a foundation for both academic exploration and practical innovation in digital communication systems.

Funding: This research received no external funding.

Data Availability Statement: The data that support the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

[1] L. Giroto de Oliveira, B. Nuss, M. B. Alabd, A. Diewald, M. Pauli, and T. Zwick, 'Joint Radar-Communication Systems: Modulation Schemes and System Design', *IEEE Trans Microw Theory Tech*, vol. 70, no. 3, pp. 1521–1551, Mar. 2022, doi: 10.1109/TMTT.2021.3126887.

[2] D. Ma *et al.*, 'Spatial Modulation for Joint Radar-Communications Systems: Design, Analysis, and

Hardware Prototype', *IEEE Trans Veh Technol*, vol. 70, no. 3, pp. 2283–2298, Mar. 2021, doi: 10.1109/TVT.2021.3056408.

[3] C. Ofori, J. Cudjoe Attachie, and F. Obeng-Adjapong, 'A GSM-Based Fault Detection on Overhead Distribution Lines', *Jurnal Nasional Teknik Elektro*, vol. 12, no. 2, pp. 70–79, Jul. 2023, doi: 10.25077/jnte.v12n2.986.2023.

[4] S. Ansari, K. A. Alnajjar, M. Saad, S. Abdallah, and A. A. El-Moursy, 'Automatic Digital Modulation Recognition Based on Genetic-Algorithm-Optimized Machine Learning Models', *IEEE Access*, vol. 10, pp. 50265–50277, 2022, doi: 10.1109/ACCESS.2022.3171909.

[5] Y. Qin, 'Design of Digital Modulation for Long Distance Optical Communication', in *2024 4th International Conference on Electronic Information Engineering and Computer (EIECT)*, IEEE, Nov. 2024, pp. 692–695. doi: 10.1109/EIECT64462.2024.10866453.

[6] G. Pasolini, A. Bazzi, and F. Zabini, 'A Raspberry Pi-Based Platform for Signal Processing Education [SP Education]', *IEEE Signal Process Mag*, vol. 34, no. 4, pp. 151–158, Jul. 2017, doi: 10.1109/MSP.2017.2693500.

[7] Y. Xiang, 'Performance Analysis of an Ultra-Low Power MFSK System', UC San Diego, 2022. [Online]. Available: https://escholarship.org/uc/item/73k5d8ds

[8] D.-T. Nguyen, H.-N. Le, Q.-K. Trinh, T.-H.-T. Tran, and T.-A. Vu, 'A Novel Efficient Hardware Implementation of Symbol Timing and Carrier Phase Synchronizer for QPSK Receivers', in *2023 12th International Conference on Control, Automation and Information Sciences (ICCAIS)*, IEEE, Nov. 2023, pp. 39–44. doi: 10.1109/ICCAIS59597.2023.10382404.

[9] V. Avalos-Bravo, J. Toro-González, and E. Velazquez-Morales, 'Development of a MATLAB® MOOC to Enhance the Computational Skills of Students', in *Perspectives and Trends in Education and Technology (ICITED 2023)*, vol. 366, Singapore: Springer, 2023, pp. 131–139. doi: 10.1007/978-981-99-5414-8_14.

[10] K. Jathursajan and A. Wijethunge, 'Raspberry Pi-Based Bearing Fault Diagnosis by Bearing Audio and Vibration Signal Via Cost-Effective Accelerometer', in *Proceedings of the International Women in Engineering Symposium*, 2022, pp. 28–30.

[11] M. A. Al Zubaidy, S. L. Qaddoori, and N. T. Gadawe, 'Efficient design and implementation of the realtime multi types digital modulations system based FPGA', *Journal of Engineering Science and Technology (JESTEC)*, vol. 18, no. 2, pp. 974–989, Apr. 2023.

[12] M. Margarat, B. E. Caroline, and S. Soumiya, 'SOA-MZI-Based Nanoscale Optical Communication with various Modulation Formats', in *Nanoelectronic Devices for Hardware and Software Security*, Boca Raton: CRC Press, 2021, pp. 179–199. doi: 10.1201/9781003126645-9.

[13] J. Wang, J. Liu, S. Li, Y. Zhao, J. Du, and L. Zhu, 'Orbital angular momentum and beyond in free-space optical communications', *Nanophotonics*, vol. 11, no. 4, pp. 645–680, Feb. 2022, doi: 10.1515/nanoph-2021-0527.

[14] P. Velez *et al.*, 'Single-Frequency Amplitude-Modulation Sensor for Dielectric Characterization of Solids and Microfluidics', *IEEE Sens J*, vol. 21, no. 10, pp. 12189–12201, May 2021, doi: 10.1109/JSEN.2021.3062290.

[15] E. S. Lee, 'Frequency-Modulation-Based IPT With Magnetic Communication for EV Wireless Charging', *IEEE Transactions on Industrial Electronics*, vol. 70, no. 2, pp. 1398–1408, Feb. 2023, doi: 10.1109/TIE.2022.3158027.

[16] Z. Liu, Y. Liu, Z. Mai, Y. Yang, N. Zhou, and C. Yu, 'Enhancing weak-magnetic-field sensing of a cavity-magnon system with dual frequency modulation', *Phys Rev A (Coll Park)*, vol. 109, no. 2, p. 023709, Feb. 2024, doi: 10.1103/PhysRevA.109.023709.

[17] J. Li, H. Yang, Q. Yi, M. Lu, J. Shi, and T. Zeng, 'High-Frequency Modulated Transformer for Multi-Contrast MRI Super-Resolution', *IEEE Trans Med Imaging*, vol. 44, no. 7, pp. 3089–3099, Jul. 2025, doi: 10.1109/TMI.2025.3558164.

[18] T. A. Salih, 'Design and Implementation of a Low Power Consumption of ASK, FSK PSK, and QSK Modulators Based on FPAA Technology', *Int J Adv Sci Eng Inf Technol*, vol. 11, no. 4, pp. 1288–1294, Aug. 2021, doi: 10.18517/ijaseit.11.4.11299.

[19] K. Kaza, S. Pallapu, T. R. Chintala, and M. Samson, 'Reconfigurable FPGA Based ASK, FSK and PSK Modulator and Automatic Demodulator', in *2024 Asia Pacific Conference on Innovation in Technology (APCIT)*, IEEE, Jul. 2024, pp. 1–8. doi: 10.1109/APCIT62007.2024.10673600.

[20] X. H. Cao, X. J. Fan, M. Li, N. H. Zhu, and W. Li, 'Microwave photonic multi-frequency reconfigurable PSK/ASK/FSK formats modulation signals generation', *Opt Commun*, vol. 550, p. 129953, Jan. 2024, doi: 10.1016/j.optcom.2023.129953.

[21] W. Tang, 'Analysis of the principle and applications of state-of-art digital modulation techniques', *IET Conference Proceedings*, vol. 2024, no. 24, pp. 460–464, Jan. 2025, doi: 10.1049/icp.2024.4551.

[22] Y. Zhang, Z. Zhao, X. Feng, T. Zhao, and Q. Hu, 'Implementation of Underwater Electric Field Communication Based on Direct Sequence Spread Spectrum (DSSS) and Binary Phase Shift Keying (BPSK) Modulation', *Biomimetics*, vol. 9, no. 2, p. 103, Feb. 2024, doi: 10.3390/biomimetics9020103.

[23] G. N. Kareem, G. A. Gbotoso, and S. O. Omogoye, 'MATLAB analysis and simulink model for amplitude modulation technique', *World Journal of Advanced Engineering Technology and Sciences*, vol. 2, no. 2, pp. 021–028, May 2021, doi: 10.30574/wjaets.2021.2.2.0035.