# Report: PWM Drawer using ATmega32 and Graphical LCD

## 1. Introduction

The purpose of this project is to design and implement a PWM Drawer capable of displaying the frequency, duty cycle, and waveform of a PWM signal. The system uses an ATmega32 microcontroller and a graphical LCD to visualize and analyze PWM signals generated internally or provided externally.

## 2. Objectives

• Measure and display the frequency and duty cycle of a PWM signal.

• Visualize the waveform on a graphical LCD.

• Support signals from both internal and external sources.

## 3. System Design

The system consists of the following components:

• Microcontroller (ATmega32): Handles PWM generation, ADC sampling, and LCD control.

• Graphical LCD: Displays the waveform, frequency, and duty cycle.

• PWM Signal Source: Provides the PWM signal for analysis.

## 4. Methodology

### 4.1 Hardware Design

• PWM Generation: Timer1 on the ATmega32 is configured in PWM mode to generate a signal with adjustable frequency and duty cycle.

• Signal Input: The ADC is used to read the input PWM signal for waveform analysis.

• Graphical LCD Interface: The microcontroller communicates with the LCD to display information.

### 4.2 Software Implementation

The project software consists of several key functions:

• PWM Initialization: Configures Timer1 for PWM generation.

• ADC Initialization: Prepares the ADC for reading analog signals.

• Waveform Sampling: Samples the PWM signal and stores the data in a buffer.

• Data Processing: Calculates the frequency and duty cycle from the sampled data.

• LCD Display: Renders the waveform and displays calculated values.

## 5. Implementation Details

### 5.1 Key Functions

• init\_timer1\_pwm(): Configures Timer1 to generate an 18Hz PWM signal.

• init\_adc(): Initializes the ADC for analog-to-digital conversion.

• read\_adc(channel): Reads the analog signal from the specified ADC channel.

• display\_waveform(data, size): Draws the waveform on the LCD.

• measure\_pwm\_frequency(): Calculates the PWM frequency (placeholder implementation).

• measure\_duty\_cycle(): Calculates the duty cycle (placeholder implementation).

### 5.2 Workflow

1. Initialize the microcontroller peripherals.

2. Generate or read a PWM signal.

3. Sample the signal using the ADC.

4. Process the data to calculate the frequency and duty cycle.

5. Display the waveform and measurements on the graphical LCD.

## 6. Results

The system successfully measures and displays the frequency, duty cycle, and waveform of a PWM signal. Example values displayed:

• Frequency: 18 Hz

• Duty Cycle: 18%

## 7. Challenges and Solutions

• Signal Noise: Implemented filtering in software to handle ADC noise.

• LCD Update Speed: Optimized drawing routines to ensure smooth rendering.

## 8. Future Improvements

• Add real-time frequency and duty cycle calculations.

• Support higher-resolution ADC sampling for improved waveform accuracy.

• Expand the system to handle multiple signal channels simultaneously.

## 9. Conclusion

The PWM Drawer is a versatile and effective tool for analyzing PWM signals. By combining hardware and software functionalities, it provides an intuitive visualization of signal properties, making it valuable for educational and debugging purposes.

## 10. References

• ATmega32 Datasheet

• Graphical LCD Documentation

• AVR Programming Tutorials