# Project Specification

## Encoder GUI

## Endeavos Innovations

## Revision 1.0

7/9/2024

## Introduction

This project aims to capture, analyze, and display position and time data from an encoder via an intuitive GUI, featuring real-time charts and graphs

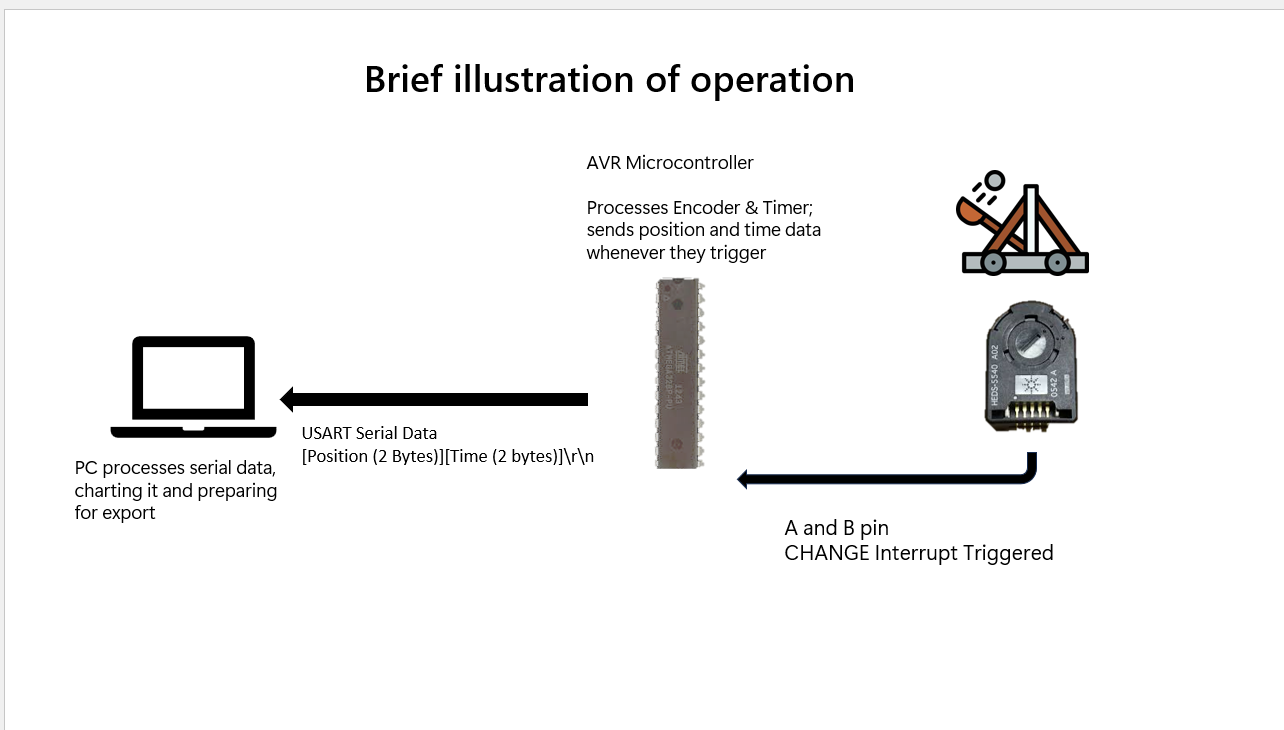
## Objective

The objective of the Encoder GUI project is to develop a comprehensive system for capturing, analyzing, and presenting position and time data from an encoder. This includes creating an intuitive graphical user interface (GUI) for user interaction, real-time visualization through charts and graphs, and data export capabilities. Initially implemented with LabVIEW for AVR microcontroller data visualization, the project aims to migrate to cross-platform frameworks like QT or wxWidgets to enhance versatility and usability in educational settings. The primary goal is to deliver a reliable educational tool that enriches learning experiences through clear and engaging data presentations. The project must be ready by August 1, 2024.

## System Overview

The overall system consists of a custom-built 3D printed encoder connected to an AVR microcontroller. The microcontroller transmits position and delta T data via USART, both on a regular timer basis and whenever an encoder interrupt is triggered. On the computer side, the data is received through USB and processed for further analysis and visualization.

## Brief Description of Operation



# Hardware

AVAGO Rotary Encoder

# Firmware (AVR Code)

## ReadEncoder Module

## [..\readEncoder.c](../readEncoder.c)

[..\readEncoder.h](../readEncoder.h)

## Overview

The `readEncoder` module is designed to interface with an encoder, read its data, and transmit the information via a serial port. This module ensures accurate timing and data transmission using interrupts and buffering techniques.

## Components

### 1. Global Variables

- volatile Encoder encoder: A structure to hold the encoder's state and position.

- volatile circular\_buffer\_t tx\_buffer: A circular buffer for storing data points to be transmitted.

### 2. Initialization Functions

- initTimer1(void):Configures Timer1 for timing and interrupt handling. Initializes to a prescaler of

-  initEncoder(): Sets up the Encoder with default state & position variables, and sets up the interrupts for it.

- initBuffer(&tx\_buffer); Initializes circular buffer with default values

### 3. Interrupt Service Routines (ISRs)

-ISR(INT0\_vect): Handles changes on the encoder's A pin.

- ISR(INT1\_vect): Handles changes on the encoder's B pin.

- ISR(TIMER1\_COMPA\_vect): Timer1 compare match interrupt, generates data points from encoder readings and stores them in the transmission buffer.

- ISR(USART\_UDRE\_vect): USART Data Register Empty interrupt, handles the transmission of data points from the buffer to the serial port.

### 4. Main Function

- Initializes the necessary peripherals (USART, Encoder, Interrupts, Timer1).

- Sets up Encoder to use built-in internal pull-up resistors.

- Enables global interrupts, configures clock speed to 16 Mhz.

- Runs an infinite loop.

### Key Macros and Constants

- F\_CPU: Defines the CPU clock speed as 16 MHz.

- PRESCALER\_VALUE: Timer prescaler value set to 64.

- SAMPLE\_RATE: Defines the sampling rate for interrupts (set to 1000ms).

- TIMER\_TICKS\_PER\_MILLISECOND: Calculates the number of timer ticks per millisecond.

- COMPARE\_VALUE: Timer compare value derived from the clock speed, prescaler, and sample rate.

## Functionality

The module reads encoder data and transmits it in a structured format via the serial port. It uses Timer1 for regular transmission with precise timing and manages data buffering with a circular buffer to ensure smooth and efficient data transmission with the USART module & its ISR. The INT0 and INT1 ISRs handle updating the encoder position when its signal changes according to the state table, and transmitting a DataPoint.

This setup ensures reliable encoder data acquisition and transmission, suitable for real-time applications requiring precise timing and data handling.

## Input/Output List

1. PD2 INT0(Pin 2 on Arduino): Encoder Pin A
2. PD3 INT1(Pin 3 On Arduino): Encoder Pin B

# Software

## Acquisition

### Hardware Interfaces

1. **Encoder**
   * **Model**: AVAGO Rotary Encoder
   * **Interface**: Digital Input
   * **Connections**:
     + **PD2 (INT0)**: Encoder Pin A
     + **PD3 (INT1)**: Encoder Pin B
2. **Microcontroller**
   * **Model**: AVR Microcontroller
   * **Clock Speed**: 16 MHz
   * **Interfaces**:
     + **Timer1**: Used for timing and generating interrupts at specified intervals.
     + **USART**: Used for serial communication to transmit encoder data.
3. **Signal Conditioning**
   * **Pull-up Resistors**: Enabled for Encoder pins to ensure clean digital signals.
4. **Digitizers and Instruments**
   * **Timer1 Configuration**: Configured with a prescaler of 64, compare value based on desired sampling rate (1000 ms).
   * **USART Configuration**: Configured for data transmission when the buffer is not empty.
5. **Control Devices**
   * **Timer Interrupts**: Utilized for generating periodic data points.
   * **USART Interrupts**: Utilized for managing data transmission.

## Analysis

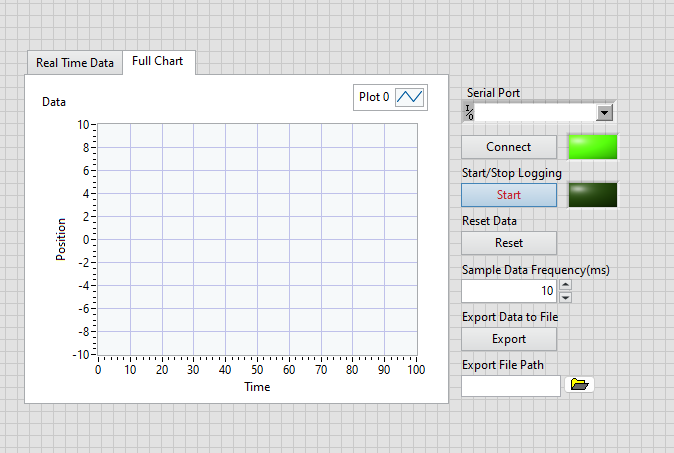
Once the position & time data arrives from the Microcontroller through the serial port, basic processing will be required to convert it into the desired units and present it. For example, say we get this line from the serial:

**10101001 10010101 11001001 10010101\r\n**

## Presentation

### User Interface

Below is a prototype of the UI. As shown, there is functionality for connection, toggling logging, resetting data, adjusting the frequency of data collection, and exporting on the right. On the left we have a pane to toggle between the real time chart, and the full chart data.



### Data Files

## Connectivity

## Priority Matrix Test Methodology Appendix A: Glossary Appendix B: Input/Output Channel List Appendix C: Sample Report Appendix D: Product Specifications