ECE 263: Embedded System Design

LAB 6: Digital Tape Measure

We certify that this work is original and not a product of anyone's work but our own.

<Bishoy Mikhail>:_____

<Leandro Neves>:_____

Submitted: March 27th, 2023

Due by: March 27th, 2023

Graded by: _____ Date: _____

Contents

1 Abstract **2**

2 Introduction **2**

3 Methods **2**

3.1 Hardware Components and Setup 2

3.2 Initialization and Interrupt Setup 3

3.3 Ultrasonic Distance Measurement 3

4 Results and Discussion **4**

5 Conclusion **5**

6 Recommendations **6**

7 Reflection **6**

8 Appendix **7**

List of Figures

1 LAB 6 Circuitry 3

2 LAB 6 Circuit Functionality Testing 5

List of Tables

List of Files

1 C Code for LAB 6 7

1 Abstract

This lab report details the design and implementation of a digital tape measure system using an ATmega328PB microcontroller and an HC-SR04 ultrasonic module for distance measurement. The system displays measurement information on a 16x2 LCD screen. The report covers the software and hardware design, testing, and evaluation results. The implementation successfully achieved accurate distance measurements and demonstrates the use of valuable programming techniques.

2 Introduction

In this report, the results of Lab 6 will be presented and analyzed. The objective of the lab was to design and implement a digital tape measure system using the ATmega328PB microcontroller and an HC-SR04 ultrasonic module. The system was intended to accurately measure distances and display the results on a 16x2 LCD screen. The design and implementation of the hardware and software were the key objectives of this lab, and it required a thorough understanding of microcontroller C programming and hardware interfacing. This report will thoroughly discuss the methodology used to achieve the objective and the results obtained.

3 Methods

3.1 Hardware Components and Setup

The hardware components used in this project include:

- ATmega328P microcontroller
- HC-SR04 ultrasonic sensor
- 16x2 LCD display

The HC-SR04 sensor is connected to the microcontroller as follows:

- Trigger pin connected to PC3
- Echo pin connected to PB0

The LCD display is connected to the microcontroller as follows:

- RS (Register Select) pin connected to PC4
- E (Enable) pin connected to PC5

- Data pins D0-D7 connected to PORTD

The program was developed using the AVR-GCC compiler and uploaded to the microcontroller via a using a debug wire.

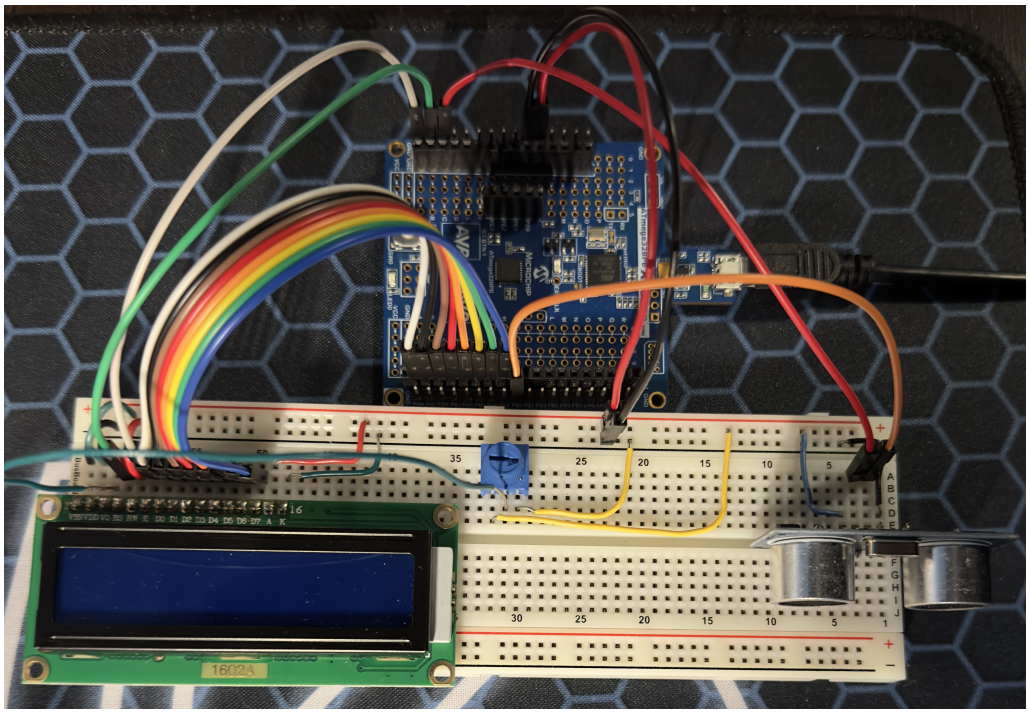


Figure 1: LAB 6 Circuitry

3.2 Initialization and Interrupt Setup

In the main() function, the InitLCD() function is called to initialize the LCD display. The DDRB register is set to 0b10 to configure PB1 as an output for the trigger signal. Timer/Counter1 is configured in input capture mode with noise canceler and edge select (rising edge) enabled, and pre-scaler set to 64. The input capture interrupt is enabled using the TIMSK1 register, and interrupts are globally enabled using the sei() function.

3.3 Ultrasonic Distance Measurement

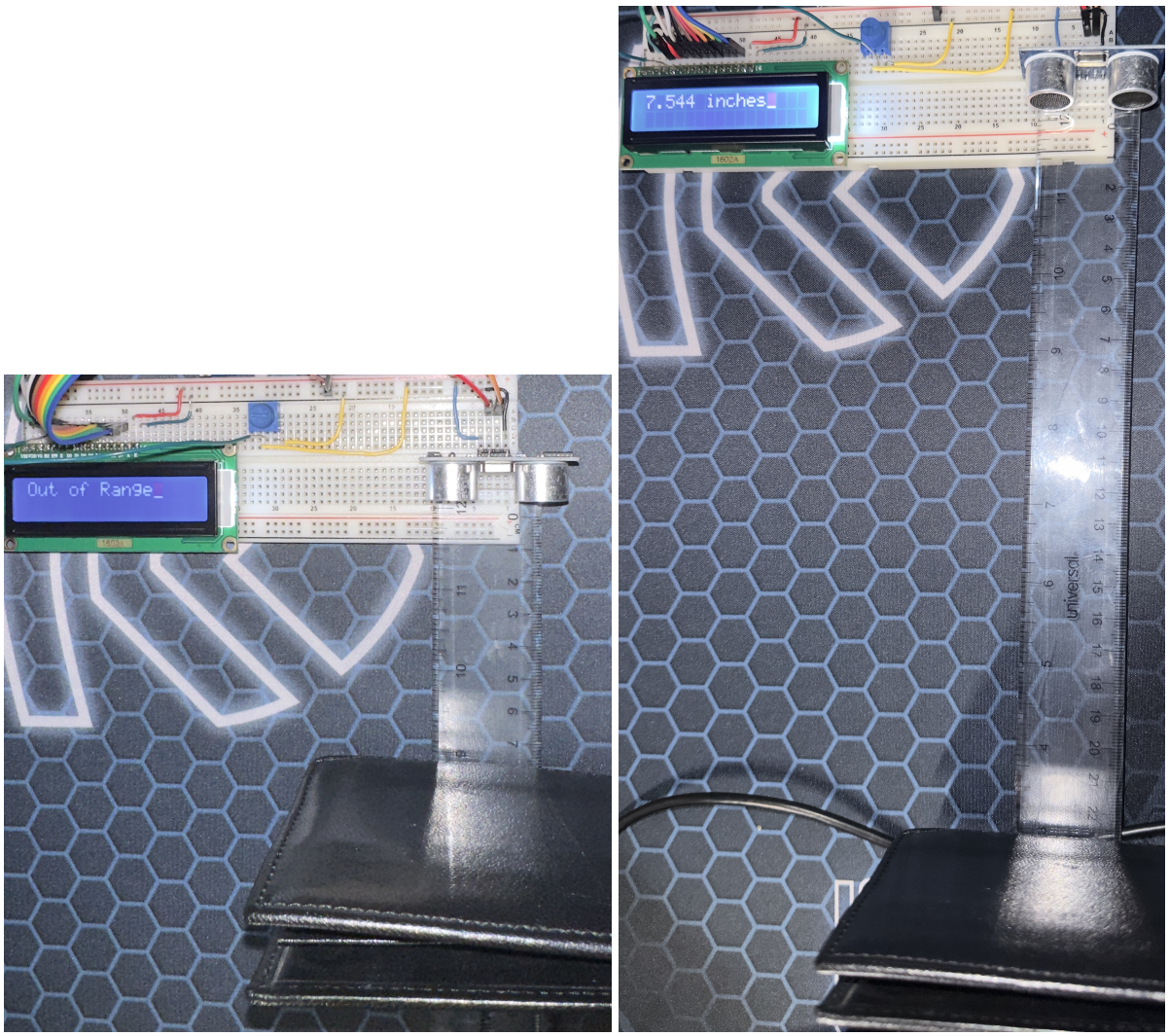
The 'ISR(TIMER1_CAPT_vect)' interrupt service routine is triggered on each rising edge of the echo signal from the HC-SR04 sensor. The 'previous_value' variable is updated with the previous value of 'new_value', and 'new_value' is updated with the current value of the input capture register (ICR1).

In the main loop, the trigger signal is sent by setting PC3 high for 10 microseconds, then low for 38 milliseconds to allow the ultrasonic waves to propagate and bounce back from an object. The time difference between the previous and current echo signals is calculated using 'uint16_t diff = new_value - previous_value', and the distance to the object in inches is calculated using 'distance = (3.5f * diff) / 148.0f'.

If the distance is within the range of 6 to 84 inches, the distance is displayed on the LCD display using 'sprintf()' to format the string and 'PrintString()' to send the string to the display. If the distance is outside this range, the message "Out of Range" is displayed instead. The timer is reset using 'TCNT1 = 0' to prevent overflow and the loop waits for 1 second using '_delay_ms(1000)' before repeating.

4 Results and Discussion

After writing the full program and building the circuitry for the device, the code was successfully compiled and run. To test the functionality of the constructed device, objects were placed at different distances from the ultrasonic sensor and the results were observed on the LCD screen. For example, Figure 2a shows an object placed approximately 3 inches away from the sensor. As stated previously, the sensor's minimum reading range is 6 inches, so theoretically, the LCD screen should display "Out Of Range". The LCD in Figure 2a agrees with the theoretical conclusion, showing the same message on the LCD. Figure 2b shows an object placed 9 inches away from the sensor which is identified as 7.544 inches on the LCD screen. There is clearly some error present in the device, in this example the percent error is 16.178%. This error could stem from the code, the hardware quality, or even the measuring equipment. Nevertheless, the device is responsive and the distance properly changes in response to the object moving farther away from the sensor. The next logical step with this project is to work on optimizing it and increasing its accuracy.



(a) 3-Inches From Sensor (b) 9-Inches From Sensor

Figure 2: LAB 6 Circuit Functionality Testing

5 Conclusion

Based on the results presented in this report, the objective of designing and implementing a digital tape measure system using the ATmega328PB microcontroller and an HC-SR04 ultrasonic module has been successfully achieved. Through careful programming and hardware interfacing, accurate distance measurements were obtained and displayed on a 16x2 LCD screen. The methodology used in this project demonstrates valuable programming techniques and provides insight into the practical applications of microcontroller systems. Overall, this project serves as a valuable exercise in microcontroller programming and provides a solid foundation for future projects in this area.

6 Recommendations

I do not have many recommendations for someone doing Lab 6 for the first time. What I would say is to get comfortable with the two major components in this lab, one being the LCD display and the other being the HC-SR04 ultrasonic module. Once you understand these components, you can more easily understand and conceptualize how to construct the circuit (hardware), which will lead to an easier time writing the code. Furthermore, the amount of code and its complexity can be overwhelming, so I recommend cutting everything up into palatable sections.

7 Reflection

Reflecting on this lab, it was very useful and a good learning experience. A lot of skills were learned and developed in this lab, which is great. I have never done hardware interfacing to this degree so it really put into perspective what it takes for seemingly simple designs and devices.

8 Appendix

The following is the program for LAB 6 written in C which uses an ultrasonic sensor to measure distance within a specific range and display it on an LCD screen.

File 1: C Code for LAB 6

```
/*
 ?   LAB_6: Digital Tape Measure

     Author:  Leandro DaSilva Neves
 ????Partner: Bishoy Mikhail

ATmega------LCD
PD7---------D7
PD6---------D6
PD5---------D5
PD4---------D4
PD3---------D3
PD2---------D2
PD1---------D1
PD0---------D0
PC5---------E
PC4---------RS
GND---------R/W (we are only writing)
GND---------Vss
Vcc---------Vdd

+5--\/\/\/\/--GND
????   V0

Optional:
Vcc---------A
GND---------K

ATmega---------Sensor
PB0(ICP1)------Echo
PC3------------Trig
*/
```

```
/*****************************************************/
#include <avr/io.h>
#include <stdio.h>
#include <avr/interrupt.h>
#define F_CPU 16000000
#include <util/delay.h>
/*****************************************************/
void InitLCD();
void WriteCommand(uint8_t d);
void WriteData(uint8_t d);
void PrintString(char s[]);
void GoToXY(uint8_t x, uint8_t y);
/*****************************************************/
volatile uint16_t new_value = 0;
volatile uint16_t previous_value = 0;
/*****************************************************/
#define EBIT 5   //Set E to PC5
#define RSBIT 4  //Set RS to PC4
#define TBIT 3   //Set trigger to PC3
#define EcoPIN 0 //Set Echo to PB0
/*****************************************************/
inline void E_HIGH() {PORTC = PORTC |= 1<<EBIT;}     //Setting E to
   ↪ high/low
inline void E_LOW() {PORTC = PORTC &= ~(1<<EBIT);}
inline void RS_HIGH() {PORTC = PORTC |= 1<<RSBIT;}   //Setting RS to
   ↪ high/low
inline void RS_LOW() {PORTC = PORTC &= ~(1<<RSBIT);}
inline void T_HIGH() {PORTC = PORTC |= 1<<TBIT;}     //Setting Trigger
   ↪  to high/low
inline void T_LOW() {PORTC = PORTC &= ~(1<<TBIT);}
/*****************************************************/
inline void delay250ns()
{
 asm volatile(    //Defining LCD Function and Setup
 "swap␣r16\t\n"
 "swap␣r16\t\n"
 "swap␣r16\t\n"   //Use swap R16 to have compiler not optimize
    ↪ delay250
```

```
 "swap␣r16\t\n");
}

int main(void)
{
 InitLCD();

 DDRB = 0b10;
 TCCR1A = 0b00<<COM1A0 | 0b00<<COM1B0 | 0b00<<WGM10;
 TCCR1B = 0<<ICNC1 | 1<<ICES1 | 0b011<<CS10 | 0b00<<WGM12; //
    ↪ Initializing and setting up Interrupt Registers and ports for
    ↪ input and outputting
 TIMSK1 = 1<<ICIE1;
 SEI(); //Also enabling interrupts

 while (1)
 {
   float distance = 0.0; //initialize meas so it clears each time and
      ↪ doesn't get FUNKY
   T_HIGH();
   _delay_us(10); //Send the 8 pulses
   T_LOW();
   _delay_ms(38); //Wait for pulses to return
    GoToXY(0,0);
   PrintString("␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣");
   GoToXY(0,1); //Clearing the display
   PrintString("␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣");
   CLI(); //Disable interrupts so you can do the MATHS
   uint16_t diff = new_value - previous_value;
   distance = (3.5f * diff) / 148.0f;
   SEI(); //Re-enable interrupts for wonderful interrupt-ness
   char S[20];
   if((distance >= 6)&(distance <= 84))
   {
     sprintf(S, "%2.3f␣inches", distance);
     GoToXY(0, 0);
     PrintString(S);
   }
```

```
    else
    {
      GoToXY(0,0); PrintString("Out of Range");
    }
    TCNT1 = 0;  //reset timer so it don't blow up
    _delay_ms(1000);
  }
}

ISR (TIMER1_CAPT_vect)
{
 previous_value = new_value;
 new_value = ICR1;
 TCCR1B ^= 1 << ICES1;
}

void InitLCD()
{
 DDRC |= 0xFF; //D is output
 _delay_ms(100);
 DDRD |= 0xFF;
 WriteCommand(0x30);
 _delay_ms(5);
 WriteCommand(0x38);
 _delay_us(100);
 WriteCommand(0x08);
 WriteCommand(0x01);
 _delay_ms(2);
 WriteCommand(0x06);
 WriteCommand(0x0F);
}

void WriteCommand(uint8_t d)
{
 E_LOW();
 RS_LOW();
 E_HIGH();
 PORTD = (d & 0xFF);
```

```
 delay250ns();
 E_LOW();
 delay250ns();
 _delay_us(39);
}

void WriteData(uint8_t d)
{
 E_LOW();
 RS_HIGH();
 E_HIGH();
 PORTD = (d & 0xFF);
 delay250ns();
 E_LOW();
 delay250ns();
 _delay_us(39);
}

void PrintString(char s[])
{
 uint8_t i=0;
 while (s[i])
 WriteData(s[i++]);
}

void GoToXY(uint8_t x, uint8_t y)
{
 WriteCommand(0x80 | (y*0x40 + x));
}
```