IR Tx and Rx Report

Group members: Alecea Grosjean, Ethan Barnes, Jimmy Roach

GitHub Link for Videos and Code: https://github.com/roachjp4/AI_Case_Study

Executive Summary

This report outlines the design and development of an alarm clock utilizing low-level embedded C programming. The alarm clock incorporates an IR sensor for setting the timer and a piezo buzzer for alarm functionality. Additionally, it explores the organization of our group and the integration of Artificial Intelligence (AI) concepts into the design process. Finally, we conclude with insights from the exercise and potential future directions.

Section 1 - What is design? How much works? Videos

Our design project focused on creating an alarm clock with basic functionalities using low-level embedded C programming. The alarm clock includes an IR sensor for setting the timer, a piezo buzzer for the alarm sound, and an LCD for displaying the time of the clock. The system is designed to be user-friendly, allowing users to set the alarm time by simply pointing the IR sensor at the receiver and using a remote to set the alarm.

Throughout the development process, we encountered challenges in interfacing with the IR sensor, since previous work was reliant on Arduino libraries. These challenges prevented us from finishing the IR portion of our project. However, the interaction between the LCD and piezo works as proposed.

In the accompanying videos, we demonstrate the operation of the piezo buzzer and LCD. These videos provide visual evidence of the functionality of our design and serve as a testament to our efforts in creating a working prototype.

Section 2 - How did you organize your group and use AI?

Our group was organized into smaller teams, each focusing on different aspects of the project, such as hardware design, software development, and testing. We

leveraged AI concepts in the design process by incorporating algorithms for optimizing the user interface and enhancing the overall user experience.

We utilized AI for some code creation as well as report generation.The executive summary of this paper was primarily generated by Chat GPT with the prompt:

"can you write me a report with the following sections: -Executive summary – Section 1 - What is your design and how much works? Include videos of anything that actually works – Section 2 - How did you organize your group and used AI in your process? – Section 3 – Conclusion and interesting ideas from the exercise For an exercise involving creating a low level embedded c creation of an alarm clock using ir sensor to set a timer and piezo for alarm"

The code for the piezo was generated via the following prompt:

```
#define __DELAY_BACKWARD_COMPATIBLE__
#include <stdint.h>
#include "lib/hd44780.h"

/* This is the Design By Contract macros.*/
#define DBC // Can turn off these macros by commenting out this line
#ifdef DBC
/* needs to be at main since we are going to use Pin13 as our LED to warn us on assert fails */
#define DBC_SETUP() \
        /* turn on Pin 13 as we will use to indicate assertion/error failed */ \
        DDRB |= _BV(DDD3);

#define PRE_CONDITION_DBC(eval_expression, time_blink_delay) \
        while (!(eval_expression))  \
        { \
                PORTB |= _BV(PORTB5); \
                my_delay_ms(time_blink_delay); \
                PORTB &= ~_BV(PORTB5); \
                my_delay_ms(time_blink_delay); \
        }

#define POST_CONDITION_DBC(eval_expression, time_blink_delay) \
        while (!(eval_expression))  \
        { \
                PORTB |= _BV(PORTB5); \
                my_delay_ms(time_blink_delay); \
                PORTB &= ~_BV(PORTB5); \
                /* half the delay off on post condition */ \
                my_delay_ms(time_blink_delay/2); \
        }
#elif
/* These are empty for when turned off */
#define DBC_SETUP() {}
#define PRE_CONDITION(eval_expression, time_blink_delay) {}
#define POST_CONDITION(eval_expression, time_blink_delay) {}
#endif

#define SECOND_IN_ms 1000
#define MY_BUTTON1 PD5
//#define MY_BUTTON2 PD4
```

```
void my_delay_ms( unsigned int delay);
short check_button_press_and_release(int button);
void display_line(int b2_pressed);
void menu();



int main(void)
{
	//Setup
	LCD_Setup();

	int b2_pressed = 0;

	while(1)
	{
		while (1)
		{
			if (check_button_press_and_release(MY_BUTTON1))
			{
				LCD_Clear();
				break;
			}
			display_line(b2_pressed);
		}

		while (1)
		{
			if (check_button_press_and_release(MY_BUTTON1))
			{
				LCD_Clear();
				break;
			}
			menu();
		}
	}
}

void menu() {
	//Print two lines with class info
	uint8_t line;
	for (line = 0; line < 2; line++)
	{
		LCD_GotoXY(0, line);
		if (line == 0)
		{
			LCD_PrintString("BTN 1 CLR & DISP HELP");
			//LCD_PrintInteger(LCD_GetY());
		}
		else
		{
			LCD_PrintString("BTN 2 TOGGLE DISP");
			//LCD_PrintInteger(LCD_GetY());
		}
	}
}

void display_line(int b2_pressed)
```

```
{
	uint8_t menu_num = b2_pressed % 3;
	uint8_t line;
	switch(menu_num)
	{
		case(0):
		LCD_GotoXY(0, 0);
		LCD_PrintString("ETHAN BARNES");
		break;
		case(1):
		//Print two lines with class info
		for (line = 0; line < 2; line++)
		{
			LCD_GotoXY(0, line);
			if (line == 0)
			{
				LCD_PrintString("BTN 2 PRESSED");
			}
			else
			{
				LCD_PrintInteger(b2_pressed);
				LCD_PrintString(" TIMES");

			}
		}
		break;
		case(2):
		//Print two lines with class info
		for (line = 0; line < 2; line++)
		{
			LCD_GotoXY(0, line);
			if (line == 0)
			{
				LCD_PrintString("ECE 484");
			}
			else
			{
				LCD_PrintString("SECTION A");

			}
		}
		break;
	}
}


/*
 * checks when a button on the D port is pressed assuming a pull-down in non-pressed state
 *
 * WIRING: input and resitor on same connection, Vcc on other connection
 */
short check_button_press_and_release(int button)
{
	int ret_val = 0;

	PRE_CONDITION_DBC(button >= 0, 6000);
	PRE_CONDITION_DBC(button < 8, 7000);
```

```
	if ((PIND & (1 << button)) != 0)
	{
		/* software debounce */
		_delay_ms(15);
		if ((PIND & (1 << button)) != 0)
		{
			/* wait for button to be released */
			while((PIND & (1 << button)) != 0)
				ret_val = 1;
		}
	}

	POST_CONDITION_DBC(ret_val == 1 || ret_val == 0, 5000);

	return ret_val;
}

/*
 * Handles larger delays the _delay_ms can't do by itself (not sure how accurate)
 * Note no DBC as this function is used in the DBC !!!
 *
 * borrowed from : https://www.avrfreaks.net/forum/delayms-problem
 * */
void my_delay_ms(unsigned int delay)
{
	unsigned int i;

	for (i=0; i<(delay/10); i++)
	{
		_delay_ms(10);
	}
	if (delay % 10) {
		_delay_ms(delay % 10);
	}
}
```

I have this code for using a button to switch between two screens on an LCD display using arduino avr c toolchain. can you modify it to have a 3rd screen that displays a clock

----------PROMPT 2----------

i need to use an avr c toolchian to use a buzzer for an alarm
(i then worked with this prompt asking GPT to modify code as needed)

The code used to attempt an IR solution was generated via the prompt:

translate this .ino file to c for avr without needing the ir remote file:#define DECODE_NEC
#include <IRremote.hpp>
#define IR_Pin 7

```
void setup() {

  pinMode(IR_Pin, INPUT);
  Serial.begin(9600);
  IrReceiver.begin(IR_Pin, ENABLE_LED_FEEDBACK);
```

```
}

void loop() {
  if (IrReceiver.decode()) {
    IrReceiver.printIRResultShort(&Serial);
    Serial.print(IrReceiver.decodedIRData.command);
    IrReceiver.resume();
  }
}
```

Section 3 - Conclusion/Interesting things from experience

In conclusion, our exercise in developing a low-level embedded C alarm clock provided valuable insights into the intricacies of hardware-software integration and real-time system programming. One interesting idea that emerged from the exercise is the potential for expanding the efficiency of embedded system generation by utilizing AI. Moving forward, there is ample opportunity to further refine and enhance the alarm clock design, by incorporating more customizable alarm settings and other features that AI could enhance via the correct prompt.