Programmable Bidirectional IR Remote-Control Interface
CSE 3442-001

Sean-Michael Woerner
1001229459
December 4, 2020

Table of Contents

Introduction

Overview

The goal of this project is to build a device that serves as a bidirectional IR interface, capable of learning and playing NEC format commands. This device can be used to communicate with other devices through sending and receiving IR commands and has the potential to be used as an IoT communications device with further development.

Theory of Operation

Hardware

The project will consist of a single board containing sockets for the M4F board, 38 kHz IR decoder, IR LED, and associated circuitry.

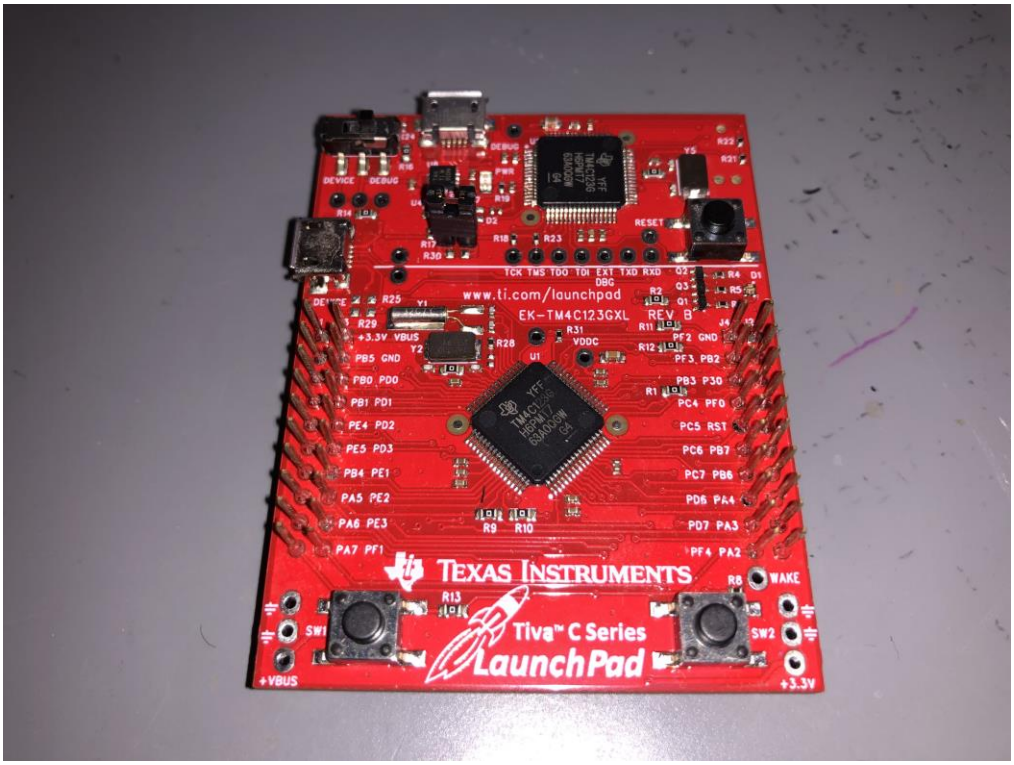

Figure 1a – TM4C123GH6PM Board

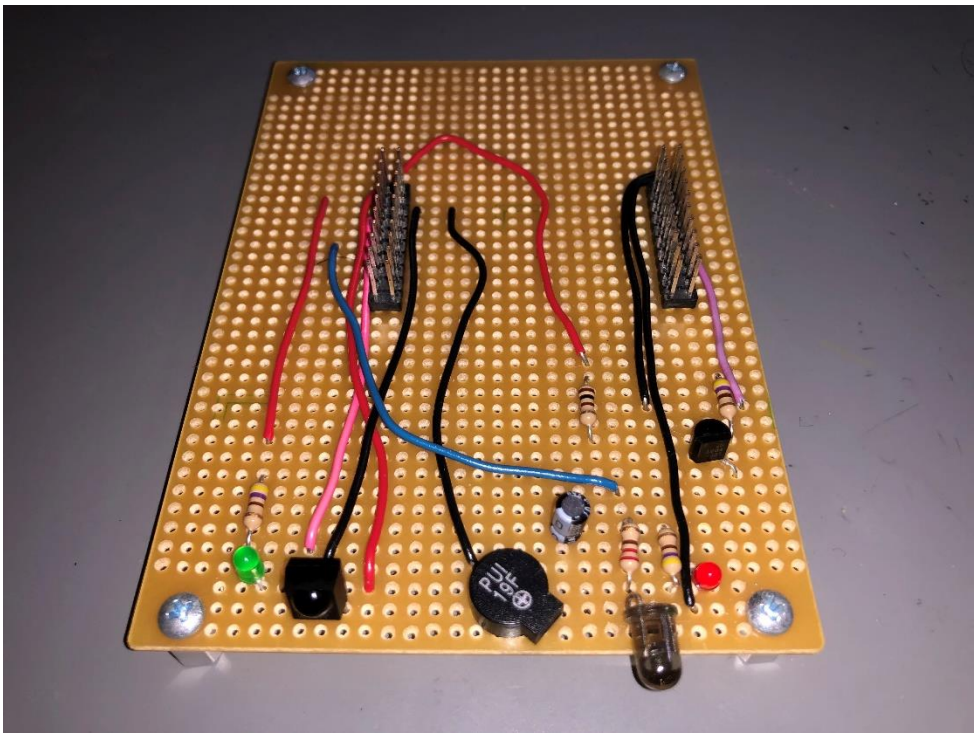

Figure 1b – PC Board with circuitry

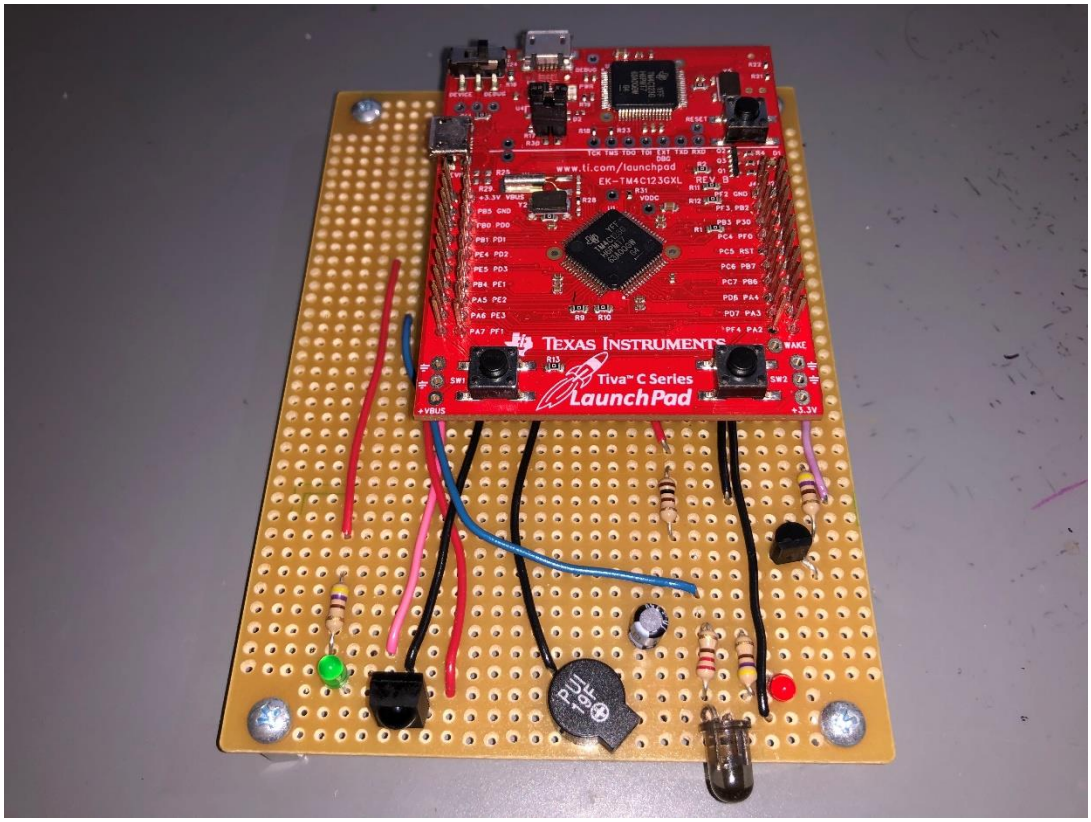

Figure 1c – IR Bidirectional Communications Device

Schematics

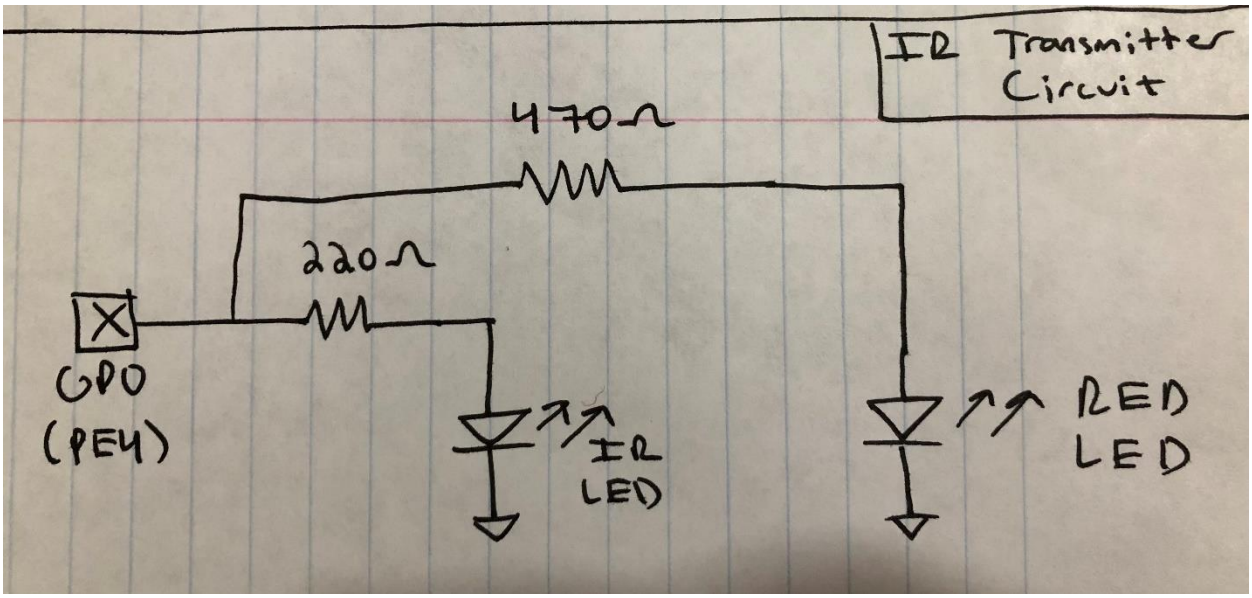

Figure 2a – IR Transmitter Circuit

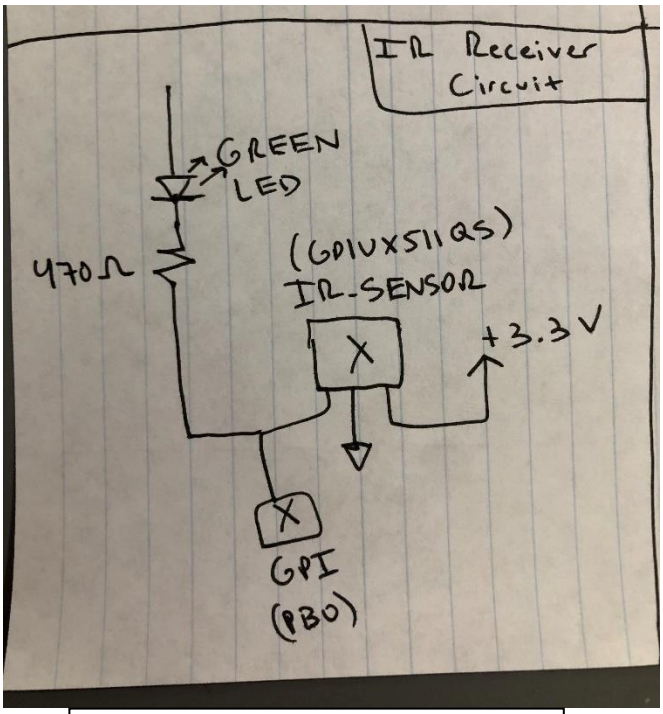

Figure 2b – IR Receiver Circuit

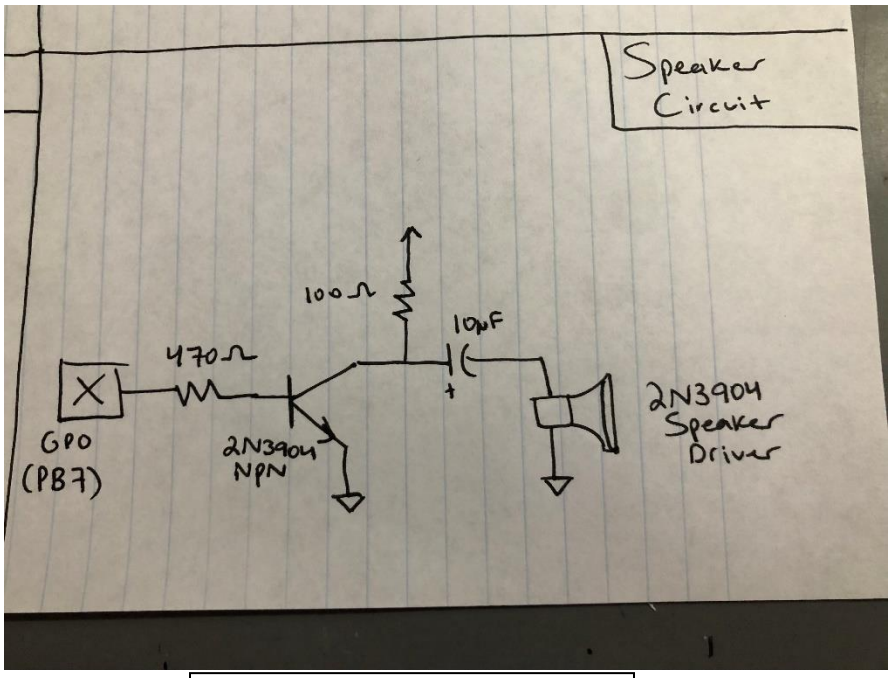

Figure 2c – Speaker Circuit

Components

Part	Quantity
EK-TM4C123GXL controller board	1
GP1UX511QS 38kHz IR detector/demodulator	1
IR LED	1
Green LED (receive LED)	1
Red LED (transmit LED)	1
470 ohm (red/green LED series limiting resistor)	2
220 ohm (IR LED series limiting resistor)	1
AT-1127-ST-2-R transducer	1
100ohm resistor (speaker current limit)	1
470ohm resistor (speaker driver collector resistor)	1
10uF capacitor (speaker AC coupler)	1
2N3904 (speaker driver)	1
10x2 100mil pitch unshrouded header	2
Wire (22-24 AWG solid wire, 3+ colors) red for 3.3V only, black for ground only	1
PC Board	1

Figure 3 – Components

Software

The software will support the following functionality:

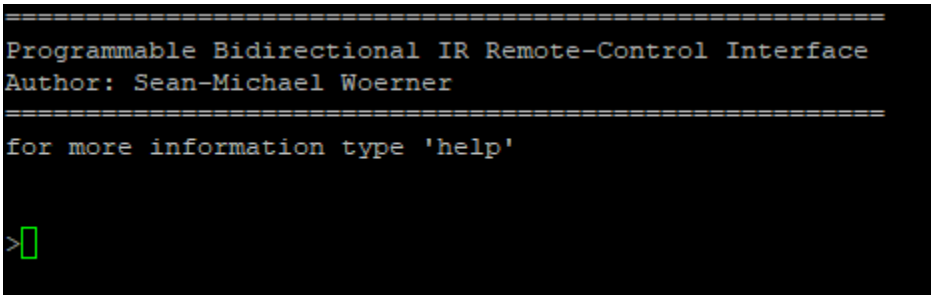

Figure 4a – Boot-up Screen

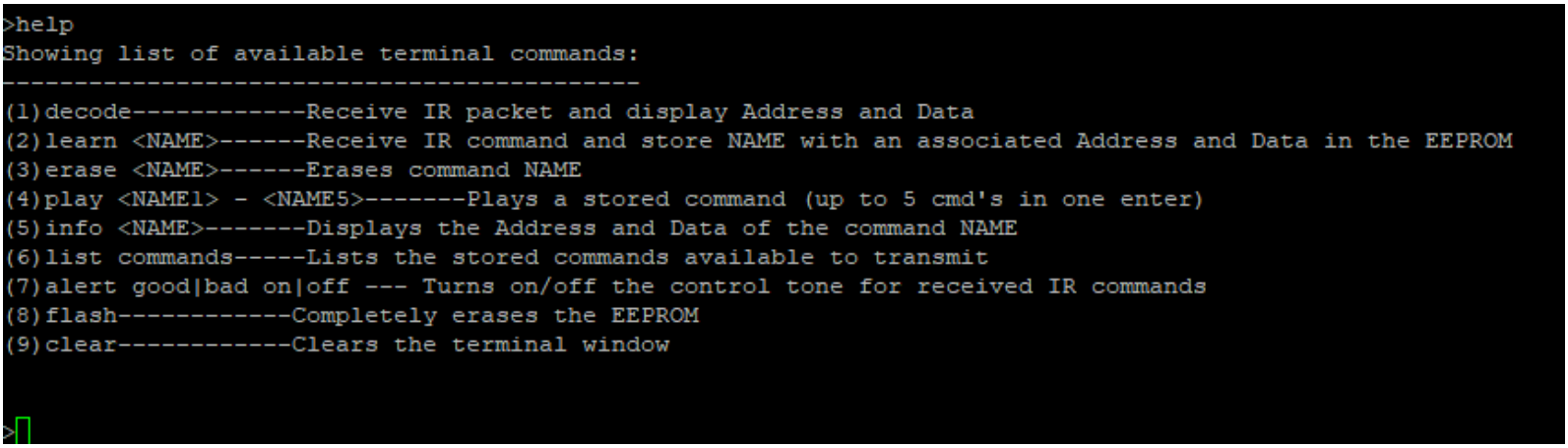

Figure 4b – Terminal Commands

UART Commands

The virtual COM port connected to UART0 will deliver these instructions:

	IR Remote Functions	
	decode <1-21>	Waits to receive an NEC IR packet and then displays the address and data
	learn NAME	Receieves an NEC IR command and stores as NAME with the assosciated address and data in EEPROM
	erase NAME	Erases a command NAME
EXTRA CREDIT	play NAME1, NAME2, …, NAME5	Plays up to 5 stored commands
	info name	Displays the address and data of the command NAME
	list commands	Lists the stored commands
	Control and Status	
	alert good|bad on|off	Turns on/off the good or bad alert tones for receieved IR commands
	flash	Completley erases the EEPROM
	clear	Clears the terminal window
	help	Displays a list of availiable terminal commands

Figure 5 – UART Commands

Available IR Codes for Reception and Transmission

#	Name	ADD	~ADD	DATA	~DATA
1	CH-	00000000	00000000	10100010	01011101
2	CH-	00000000	00000000	01100010	10011101
3	CH+	00000000	00000000	11100010	00011101
4	PREV	00000000	00000000	00100010	11011101
5	NEXT	00000000	00000000	00000010	11111101
6	PLAY/PAUSE	00000000	00000000	11000010	00111101
7	VOL-	00000000	00000000	11100000	00011111
8	VOL+	00000000	00000000	10101000	01010111
9	EQ	00000000	00000000	10010000	01101111
10	0	00000000	00000000	01101000	10010111
11	100+	00000000	00000000	10011000	01100111
12	200+	00000000	00000000	10110000	01001111
13	1	00000000	00000000	00110000	11001111
14	2	00000000	00000000	00011000	11100111
15	3	00000000	00000000	01111010	10000101
16	4	00000000	00000000	00010000	11101111
17	5	00000000	00000000	00111000	11000111
18	6	00000000	00000000	01011010	10100101
19	7	00000000	00000000	01000010	10111101
20	8	00000000	00000000	01001010	10110101
21	9	00000000	00000000	01010010	10101101

Figure 6 – IR Codes

General Operation

The IR code reception and transmission will be implemented within interrupts so that the UART interface can receive and transmit characters without interruption or loss of data.

When an IR code using NEC format is received with a valid address and data, and the good alerts are enabled, a higher-pitched tone will commence indicating success. If an IR code using NEC format is received containing errors and the bad alerts are enabled, a lower-pitched tone will commence indicating an error.

When a terminal command is received, a search will begin looking through the list of available commands and then will perform the desired action if available. For example, if a user enters the command:

"learn VOLUP 0 168", the software will learn the command VOLUP and if the user then enters "play VOLUP VOLUP VOLUP" the software will play the IR command "VOLUP" three times while printing the given address and data. The receiving device, if programmed properly, should turn up the volume three times.

With all the listed parts working together, the device can be programmed to carry out desired functionality, all programmed by the user.

Code

Below is a list of the included files that are contained within the project's folder:

1. CommonTerminalInterface.c
    - Receives an entered terminal line from the user and parses the different fields to determine the command entered along with attached arguments.

2. eeprom.c
    - Saves a user defined list of commands and rules in the EEPROM_EERDWR_R register to be accessed when called upon. Also contains functionality to get information of or erase a selected, saved command or rule.
3. eeprom.h
    - Header file for eeprom.c. Contains inclusions, definitions, variables, and functions that may be used in eeprom.c.
4. IR_Receiver.c
    - Uses timers and interrupts to sample a received IR command. IR reception is caught on pin PB0 which contains the input line from the GP1UX511QS IR detector. Once a signal is detected, a falling edge interrupt commences that turns on Timer1 ISR, running for 103 iterations, turning off interrupts afterward. At each iteration, an appropriate time value is given to the timer to correctly sample the IR signal. Each iteration toggles pin PB1 which will see if the IR signal is high or low, saving the address and data results to corresponding arrays. The resultant arrays are converted from binary to decimal and sent to getButton() which determines the given IR command based on the received address and data.
5. IR_Transmitter.c

- Initializes hardware for the speaker (pin PB7) and IR LED (pin PE4). The speaker and IR LED both use pulse width modulation in order to emit a specified frequency wave, rather than a digital high or low. The IR_LED will be configured to emit a 38.2 kHz wave with 50% duty cycle to follow the NEC format. The speaker will be configured on an as-needed basis, changing the frequency value for desired pitch. The function playComment() takes in an address and data value which is then converted to binary, formatted to mimic a signal that a NEC format remote control would send, and saved to an output array. Once the output array is filled, the function IR_Interrupt() follows a similar process as IR_Receiver.c. Timer0 is used with appropriate time values that toggles pin PE4 to emit a 38.2 kHz signal based on the received address and data values, outputting the given command.

6. main,c
    - Initializes all hardware and begins the project. On start-up, the boot-up sound commences and the user is brought to the main menu in the terminal. From here, a loop begins waiting for user input and checks for which command or rule the user has given. Execution of the command or rule begins once found to be valid.
7. project.h
    - Header file for main.c. Contains inclusions, definitions, variables, and functions that may be used in main.c.
8. serial_s.s
    - Contains blocking functions that writes serial data when called from UART0
9. tm4c123gh6pm_startup_ccs.c

- Replaces the default interrupt handler for the following:
    a) GPIO PORT B – onFallingEdge
    b) Timer 0 subtimer A – IR_Interrupt
    c) Timer 1 subtimer A - timerISR

10. tm4c123gh6pm.cmd
    - Default link command file for the TM4C123GH6PM board

11. tm4c123gh6pm.h
    - Contains all addresses and values of pins on the TM4C123HG6PM board

12. uart0.h
    - Header file for uart0.c. Contains inclusions, definitions, variables, and functions that may be used in uart0.c.

13. uart0.c
    - Initializes hardware that configures UART0 for 115200 baurdrate with 8N1 format. Contains the blocking functions that are also used in serial_s.s

14. wait.c
    - Includes a functions that waits for a given amount of microseconds.

15. wait.h
    - Header file for wait.c. Contains inclusions, definitions, variables, and functions that may be used in wait.c.

Conclusion

After building the hardware and compiling the included program files, a programmable bidirectional IR remote-control interface was successfully assembled. Now, a user can program this device to communicate with other NEC format IR receiving devices or even use this device as a foundation for an IoT system with additional development.