Lab 3 Report

Nathan Grant & Jiss Xavier

I. INTRODUCTION

Lab 3 consisted of using two TV remotes, two IR receiver modules, and two OLED displays. The big picture idea of this lab is as follows. To start, use the Saleae Logic Analyzer to decode the remote's different IR signals for its different buttons. Once this is done, the next step is to write software that can use these readings from the IR receiver module to detect which button on the remote is being pressed. From there, by identifying the button pressed on the remote, you use the number of times a button is pressed to type different characters on the OLED display using SPI, and then eventually have them sent to the other OLED display through UART communication.

II. BACKGROUND

While a lot of the software and hardware used in this lab are similar and carried over from the first lab, there are a few new concepts and tools used in this lab that are new.

This lab revolves around IR signals that are produced by microcontrollers. In our case, we used the AT&T S10-S2 Remote Control to generate our IR signals. Additionally, we used the Vishay TSOP31336 or 1236 or 31236 IR receiver to record the IR signals generated by our remote. We had to build a simple circuit in which the Vishay IR receiver was connected to ground and power, with its output pin interfaced to our LaunchPad to observe the IR signals recorded. We incorporated a 100-ohm resistor and 100 micro Farad capacitor into this circuit to help reduce the noise that might occur on the power source. We also configured the AT&T remote control by following simple instructions that were provided to us through the lab manual.

In essence, we had our AT&T remote control that produced IR signals that were recorded by the Vishay IR receiver module and then passed onto our LaunchPad using the GPIO Input Pin to be decoded and used. The IR signal produced by the remote control could take on many different transmission protocols, and in our case, we observed that the IR signal followed the NEC IR transmission protocol. In this particular transmission protocol, the transmission begins with a burst with the length of 9 ms followed by a pause of 4.5 ms then the data. '0's and '1's are characterized by different pulse widths where a 1 ms burst corresponds to a '0' and a 2 ms burst corresponds to a '1'. Lastly, a recurring pulse of a fixed with different from all others is used as repeat bits to indicate the button is being held down.

The IR interface and remote are then combined with our ADAFruit OLED which uses SPI to display the messages generated on the LaunchPad from the IR receiver. UART, Asynchronous Serial Communication, is used to transmit data between two boards which is something we have seen before in Lab 1.

III. GOALS

Part 1: Capturing and Characterizing IR Transmission using Saleae logic

The objective of the first portion of the lab is to characterize the infrared transmissions generated by the remote control when different keys are pressed. Every button on the remote is encoded with a different pattern of varying length pulses, and we are tasked with differentiating between the different button presses. We begin by configuring our remote and setting up the necessary circuit needed for the IR receiver. In the first part, we use the Saleae USB logic analyzer to capture the readings when buttons 1 through 9, along with the 'Delete' and 'Enter' buttons on the remote are pressed. Using the readings captured when the different buttons are pressed, we are to characterize the IR transmission format along with the different timings, pulse lengths, and patterns observed. For each button, a unique binary sequence is produced that will be used during the next portion of the lab to decode the button press.

Part 2: Decoding IR Transmissions / Application Program

In this part, we first interface the microcontroller to the IR receiver module. We utilize a GPIO input pin to read the data recorded by the IR receiver module. We are to build software that decodes different button presses by utilizing interrupts. We are to use the data recorded in Part 1 of the lab to build this software. More specifically, we utilize GPIO interrupts that are triggered by falling edges along with Timer interrupts that are used to measure the pulse widths so that the values can be used for decoding. By observing the different pulse widths, we are to build the unique binary sequence produced by each key ignoring the repeat bits. Our software is to distinguish between different buttons on the remote and indicates which key is pressed by decoding the IR signal and binary sequence that is observed when a specific button is pressed.

Part 3: Board to Board Texting Using Asynchronous Serial Communication (UART)

Having completed Parts 1 and 2 of the lab, we are to expand upon our application from Part 2 to allow for texting between LaunchPads. In this part, we begin by implementing software that allows for multi-tap texting, in which keys are pressed multiple times to run through the different letters, spaces and backspaces are also to be implemented. Next, we are to display the messages we create using the remote on an OLED display in which the color of the text can be altered by pressing '1'. Lastly, we are to send the messages created on each board back and forth using UART. The OLED would display the message to be sent on the top half of the display and the messages received on the bottom half of the display.

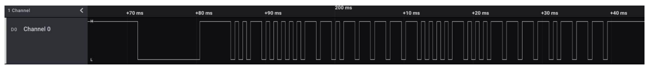
What did we learn?

In this lab, we learned how to decode IR signals generated by the microcontroller by analyzing the pulse widths of the signal produced when a button is pressed. Furthermore, we learned how to interface an IR receiver to our LaunchPad using the GPIO input pin and read in the data recorded by the IR receiver. Additionally, from a software implementation standpoint, we learned how to implement interrupts whether they be GPIO interrupts or timer interrupts. We learned how to interrupt on falling edges using GPIO interrupts and then measure the pulse width using the timer interrupts found in timer_if. This allowed us to decode the different keys of the remote allowing us to create a multi-tap texting application that could be displayed on an OLED. Although we were unable to get messaging between the boards to work, we learned to use UART to interface our boards.

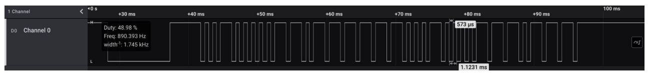
IV. METHODS

The first step in this lab was learning the values outputted by the IR Receiver module from each related button on our TV remote. In order to do this, we first had to build the IR receiver circuit and configure the remote, by following the instructions in the lab document. We used the Saleae Logic Analyzer, combined with our knowledge from the internet of NEC decoding, to get a value of 1's and 0's outputted by the receiver module for all of the relevant buttons on the remote. Once we had what each button's output was, we knew what value was being sent to the microcontroller for every button press.

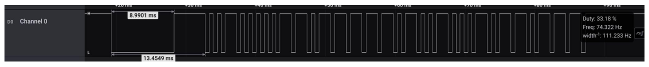
Button for '0':



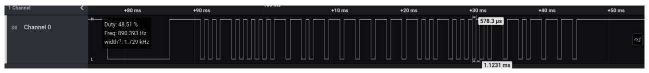
Button for '1':



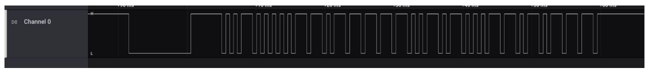
Button for '2':



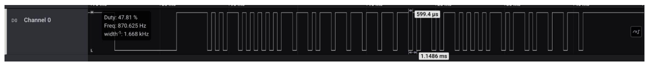
Button for '3':



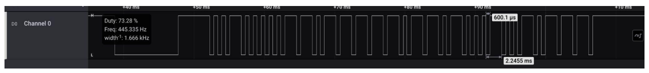
Button for '4':



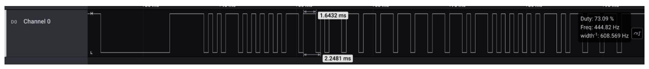
Button for '5':



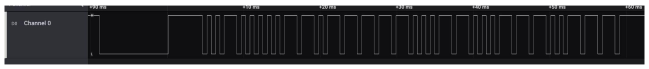
Button for '6':



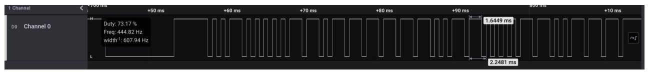
Button for '7':



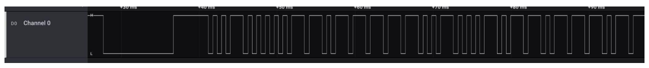
Button for ‘8’:



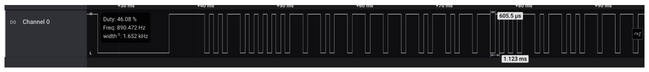
Button for ‘9’:



Button for ‘Enter’:

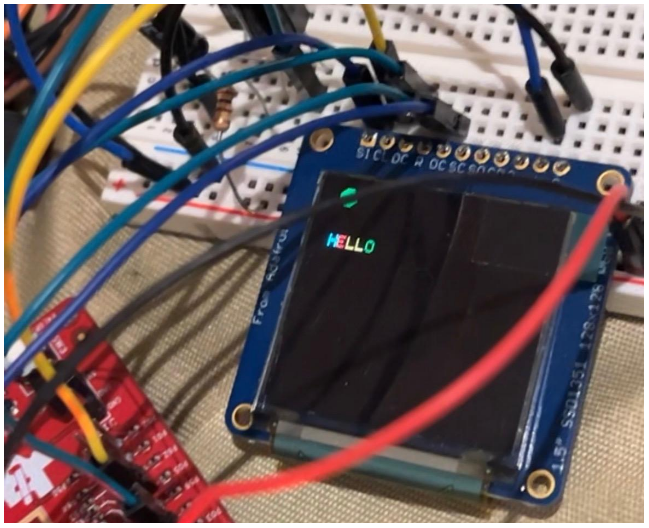


Button for ‘Delete’:



The next step was for us to write code in order to decode these signals in software, since the receiver module outputted high signals with different pulse widths, indicating whether the value was a 1 or a 0. Since we already decoded these signals by hand, we knew what they should translate to, but we had to make sure our software was decoding the same consistent values that we were expecting. In order to do this, we used falling edge interrupts as well as timer interrupts. First, we configured a falling edge gpio interrupt on the pin that was receiving the IR output. This would then trigger an interrupt on every falling edge of the output signal. Since we knew the time between falling edges was different for a 1 and a 0, we could use the time between these falling edges to decode the signal. We did this by also using a timer interrupt. We first tried using a timer interrupt that caused an interrupt every millisecond, but found that this was not precise enough to accurately decode the difference between a 1 or 0 value. So, we altered our timer interrupt to throw an interrupt every 100 microseconds, which we found provided enough precision. Essentially, we threw an interrupt every 100 microseconds, and then increment a variable in the interrupt handler, to keep track of how many interrupts had been thrown. In our falling edge gpio handler, we would check this timer value on every falling edge, and by comparing the difference between the timer variable on each successive falling edge, we would know whether the value provided was a 0 or 1. We combined all of these 1 or 0 values in a string variable called “sequence”, and then compared them to what we knew each button press should decode to. Once we verified that the software decoding matched all of our “by-hand” decodings from the logic analyzer, we knew we were accurately decoding each button.

Once we had the button decoding down, the next challenge was to check for successive button presses of the same button, in order to change the letter. We could still keep track of time from our running timer interrupt, and so by a little trial and error (seeing how much time felt natural to change the letter vs. typing the same letter twice), combined with OLED functions from the previous lab, we were able to output whatever characters we wanted to the OLED screen. We also implemented the font color by having a little colored ball in the top left of our screen, depicting what color the current text would be output in. We stored 5 colors in an array, and would simply change both the font color and the ball color every time the button 1 was pressed on the remote, so the user knew which color the successive letters would be typed in.



The last functionality of this lab was to do all of this logic on both boards and be able to send the typed messages back and forth. Unfortunately, although we got all of the aforementioned functionality working on both labs, we were unable to successfully configure the UART interrupts in order to send the messages back and forth from one board to the other.

V. CONCLUSION

This lab turned out to be an incredibly challenging and time-consuming lab as many students including ourselves faced several technical challenges and found the lab overall difficult to complete. We were able to quickly set up the circuit needed for the IR receiver and obtain the data produced when different keys are pressed using the Saleae logic analyzer. However, when it came to building the software that would decode the signal produced when different keys on the remote are pressed, we faced several obstacles as we had to deal with new concepts and technologies such as timers and interrupts. Despite this, we were able to get help and guidance from our TA, Asmita, which allowed us to get the software ready and functioning. Once, we had the code for decoding the signals working, we were able to quickly implement the multi-tap texting application and interface the application with our OLED display. We were able to get an intuitive and pleasant multitap messaging experience implemented using our LaunchPad, OLED, and IR receiver. We then began implementing the messaging over UART and faced several issues and bugs when trying to interface the two LaunchPads. Unfortunately, we ran out of time in the end and weren't quite able to complete the UART messaging functionality. However, we believe that with some more time and guidance from our TA, we would have been able to get this done. Overall, this lab was incredibly time-consuming and challenging. Much like other students in the class, we were plagued with several different hardware and software difficulties that made completing this lab difficult. Now having completed it, we can confidently say that we have learned a lot from this experience that we will carry forward into future labs.