**EEC172 Lab 2 Report**

Name: Chen Chen SID: 998820114

Name: Weidong Wu SID: 912341051

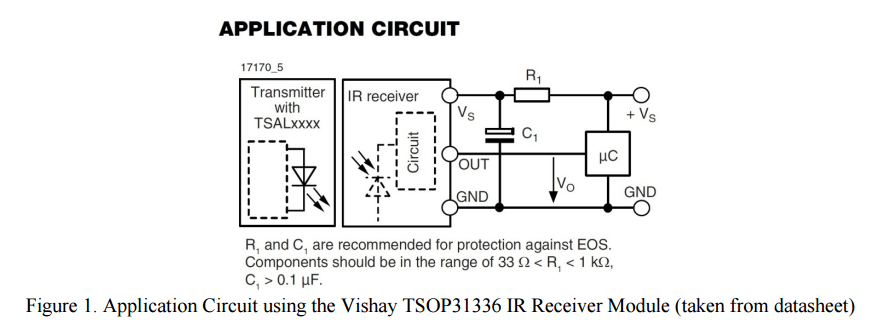
**Introduction**

In this lab, we used our Saleae logic analyzer and an IR receiver module to characterize the transmissions for buttons from an AT&T IR remote control for a specific TV. We decoded different waveforms coming from the remote control with an IR receiver and CC3200 Timer Interrupts. We interfaced the CC3200 Launchpad to the IR receiver module and write a program that uses interrupts to monitor the signal from the IR receiver module. By analyzing the series of pulses, our program is able to determine which button was pressed on the IR remote control programmed for our particular TV code. We then used our IR remote control to compose text messages using the multi-tap text entry system and send text messages back and forth between two CC3200 LaunchPad boards over an asynchronous serial (UART) communication channel.

**Methods and Procedures**

**Part I. Interfacing to the OLED using the Serial Peripheral Interface (SPI)**

After we received new equipments, we wired up IR receiver to our processor and connected the OUT signal to our Saleae logic probe according to the figure below. The TSOP31336 has three pins which we need to interface with: power (Vs), ground (GND) and output signal (OUT). R1 and C1in the Figure below are recommended to help filter noise that might occur on the power source and to protect against electrical over-stress (EOS) from power supply variations. Vs was set to +3.3V and powered directly from CC3200. We used R1 = 100Ω and C1 = 100µF.

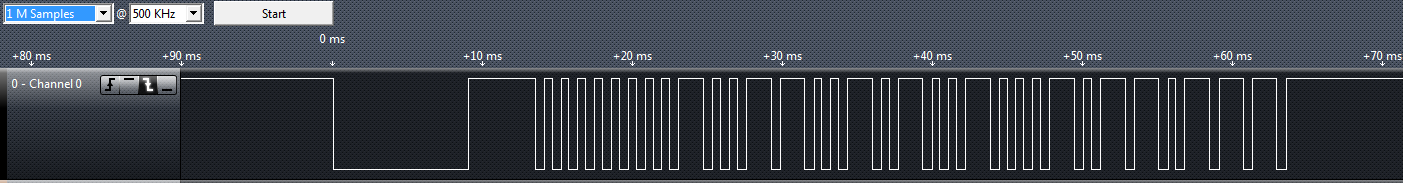


In order to switch AT&T S10-S3 Remote Control to programing mode, we followed the “Remote Control Configuration” taken directly from the Lab Manual. The exact steps can be found in Appendix II.

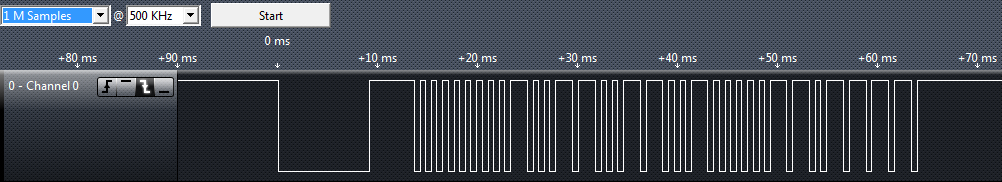
Next, we read through the TSOP31336 IR Receiver Module Data Sheet and found out how the module works. We also used Saleae logic probe to capture couple waveforms to confirm our understanding of the IR communication protocol. Our 4-digit Remote code is 1020. By capturing and reproducing two wave forms for each button, we are managed to record and see the difference between each signal. Our remote control seems to use The NEC Code protocol. Pressing each button starts the transmission with a leader code and a pause, and 1s and 0s are represented by having different pulse distances. An interesting observation is that although the lights in lab typically do not generate noise on our devices, the signal from other groups in the same lab might somehow interfere with our experiment.

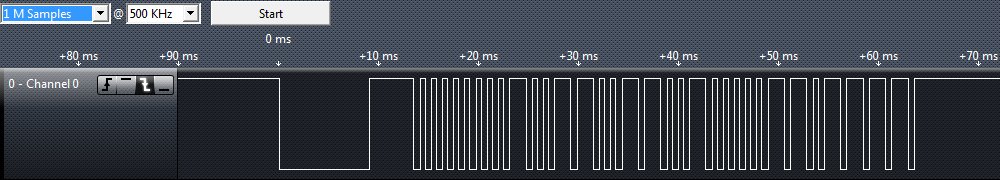
**Captured Waveforms:**

0: Address bits: 0000 0000 1001 1001; Command Bits: 0100 1000 1011 0111

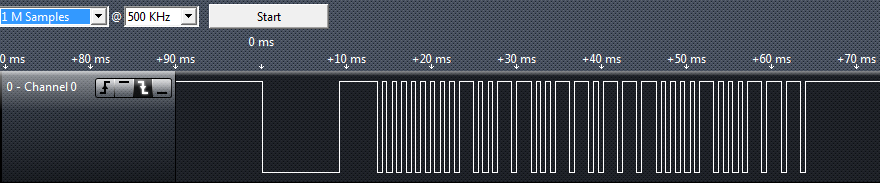


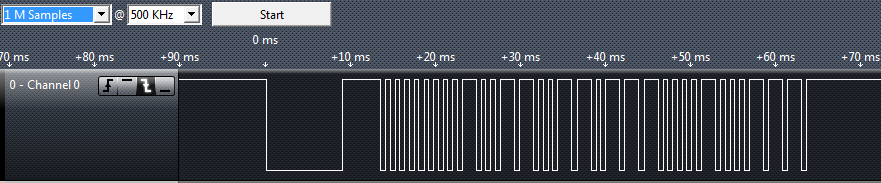


1: Address bits: 0000 0000 1001 1001; Command Bits: 1001 0000 0110 1111  


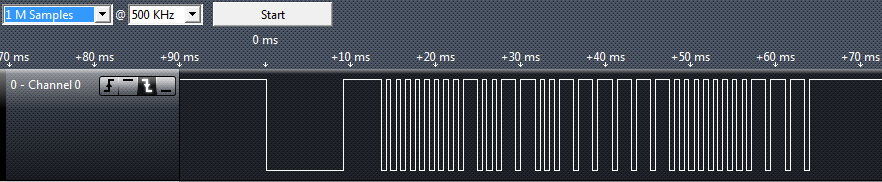


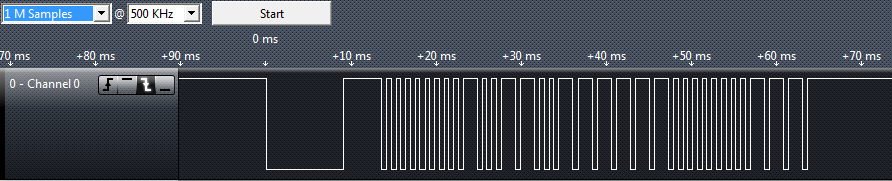
2: Address bits: 0000 0000 1001 1001; Command Bits: 1011 1000 0100 0111



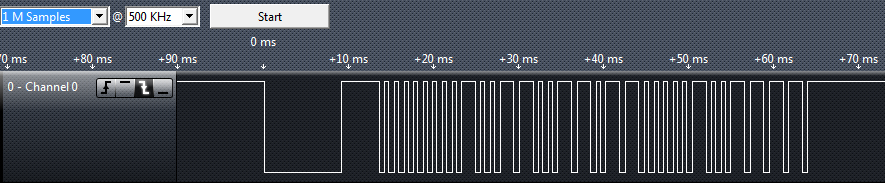


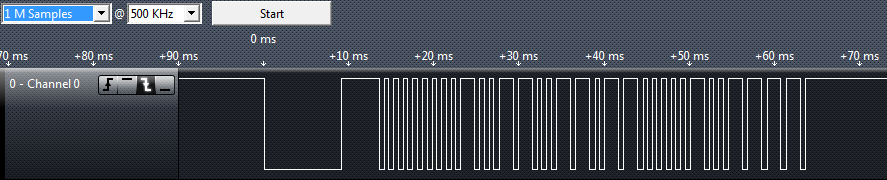
3: Address bits: 0000 0000 1001 1001; Command Bits: 1111 1000 0000 0111



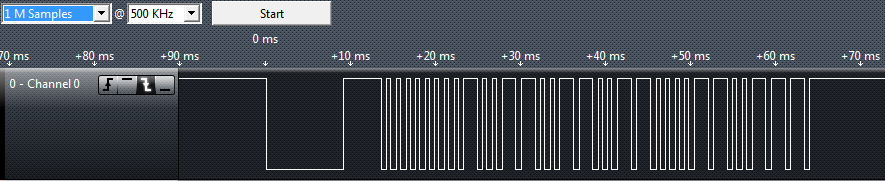


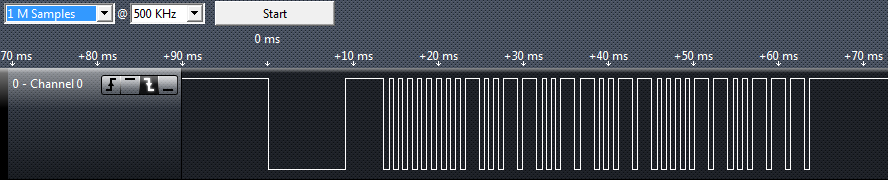
4: Address bits: 0000 0000 1001 1001; Command Bits: 1011 0000 0100 1111



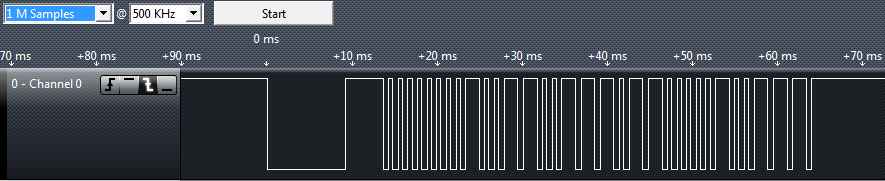


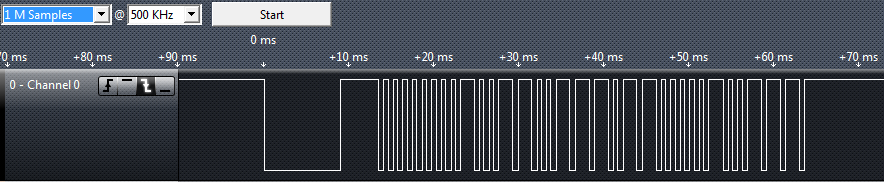
5: Address bits: 0000 0000 1001 1001; Command Bits: 1001 1000 0110 0111



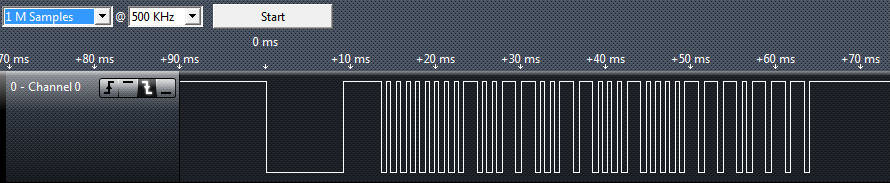


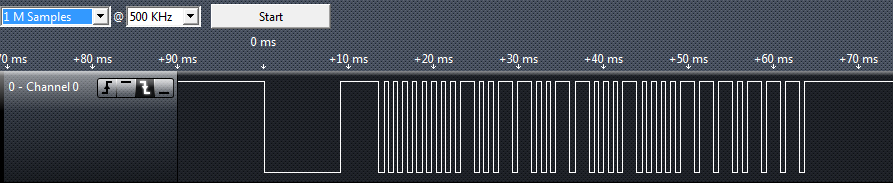
6: Address bits: 0000 0000 1001 1001; Command Bits: 1101 1000 0010 0111



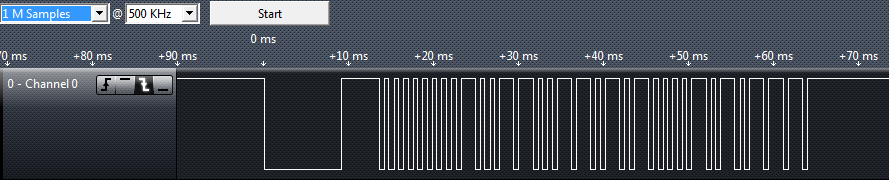


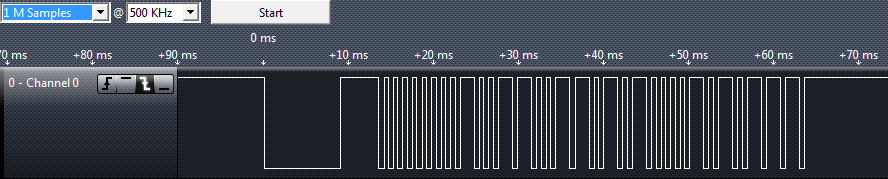
7: Address bits: 0000 0000 1001 1001; Command Bits: 1000 1000 0111 0111



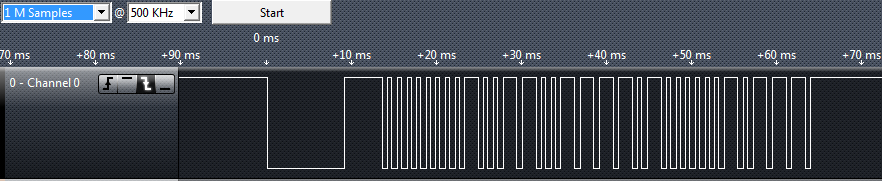


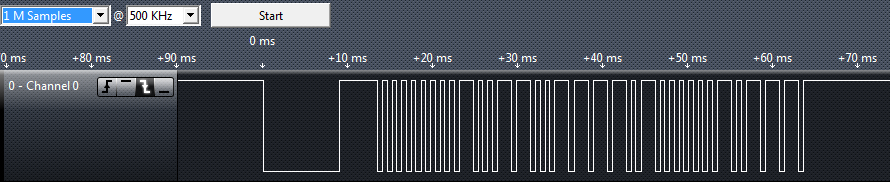
8: Address bits: 0000 0000 1001 1001; Command Bits: 1010 1000 0101 0111



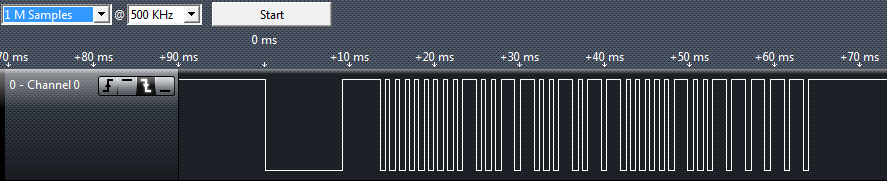


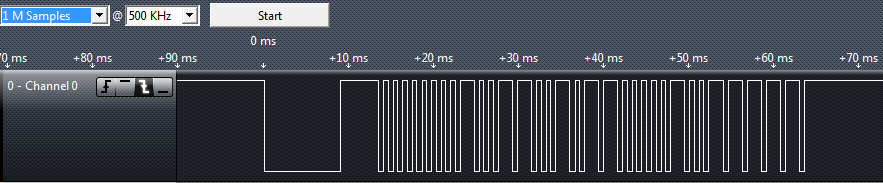
9: Address bits: 0000 0000 1001 1001; Command Bits: 1110 1000 0001 0111



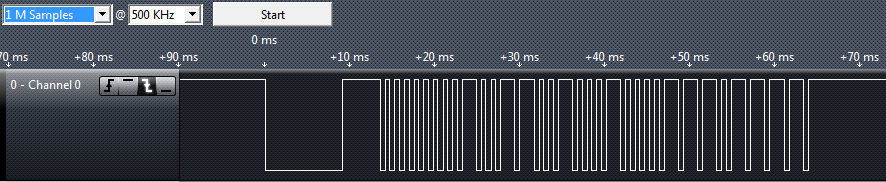


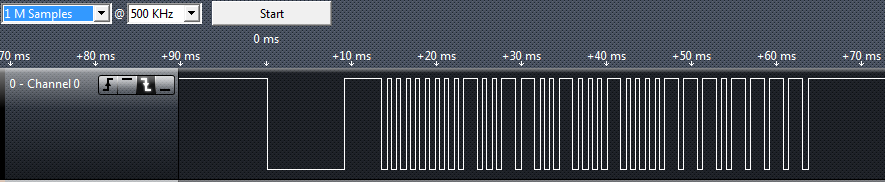
MUTE: Address bits: 0000 0000 1001 1001; Command Bits: 0110 0000 1001 1111





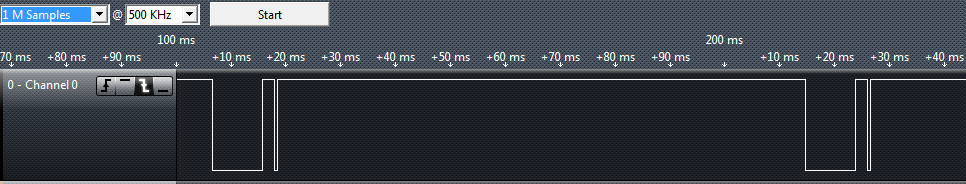
ENTER: Address bits: 0000 0000 1001 1001; Command Bits: 0001 0000 1110 1111





Repeat signal



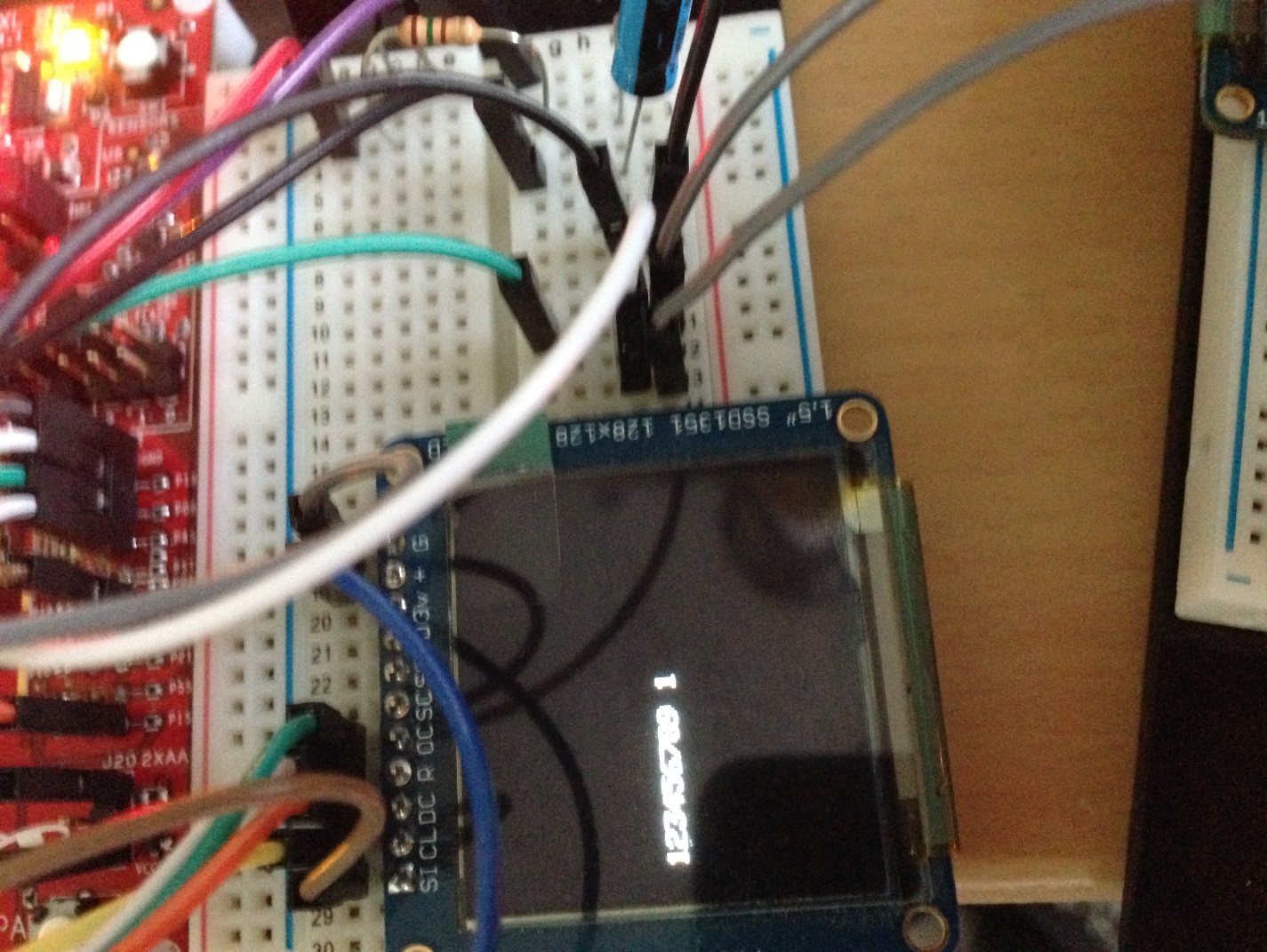


Our IR remote has a modulation carrier frequency of 36 kHz so a logic analyzer sampling frequency that is significantly above that will yield accurate timing measurements. We set the sampling options to: 1 M Samples @ 500 KHz to capture 2 seconds of data.

**Part II: Decoding IR Transmissions / Application Program**

In this part, we tried to interface the microcontroller to the Vishay TSOP31336 IR receiver module. Firstly, we connect the OUT signal of the TSOP31336 to an unused GPIO pin. Similar to previous labs, we use the PIN MUX utility to configure the pin we choose for GPIO input, SPI and UART transmission. We then used interrupts to detect the rising edges of the IR remote input signal, and use 2 of the microcontroller's timers to determine the pulse widths.

**Button Recognition shown on OLED and Terminal**



Button Recognition is achieved by encoding each button to a unique MACRO value that is defined at the start of our program. Each button has a unique code generated by simply treating the signal bits as a binary number. For example, the binary code for button “6” is 0000 0000 1001 1001 1101 1000 0010 0111, which can be converted into decimal number 10082343. This is defined as a MACRO for button “6”. Each time when the timer interrupt is triggered for the GPIO pin. The timer will start counting the pulse widths and determine signal value. Different range of pulse widths are interpreted as different signals. For example, if the Interval is from 80,000 to 100,000 cycles, the signal is interpreted as a '0'. The bits are instantaneously added to a number which will be compared to our MACRO after each successful signal transmission in order to determined which button is pressed. We also wrote codes to ignore the garbage bits which are not significant in determine the button identity.

It is worth mentioning that in order to achieve the above functionality, interrupt routines need to be properly implemented. In our code, TimerA3\_A was configured to be an edge capturing timer which is dedicated to detect a positive edge and turn subsequent signals into actual bits. TimerA3\_B was configured as a periodic timer whose only job is to assist Timer A and count the missed timeout cycle in timer A. All the configuration, initialization and interrupt vector registrations can be found in our code in Appendix I.

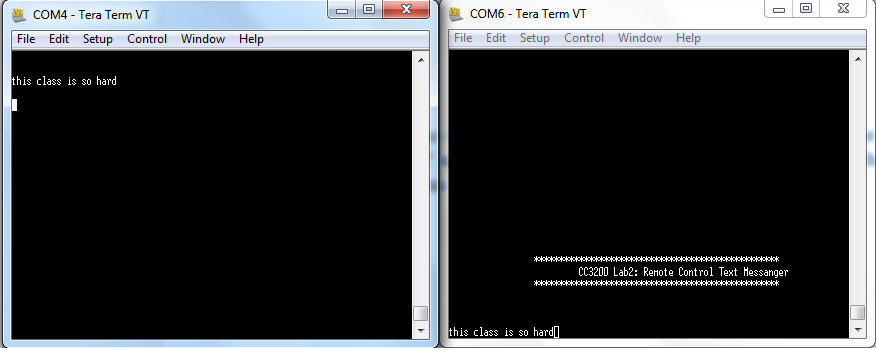
After doing this part, we were able to recognize each button pretty well. This will be crucial in the next part where we specify the behaviors and functions of each button.

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

In this part, we connect our two CC3200 LaunchPads using asynchronous serial lines TX and RX of UART1. UART0 on each of the board is occupied by Terminal/Console Window for debugging purposes. Each of our two CC3200 LaunchPads is connected to an Adafruit color OLED display through SPI to display text. We eventually developed an application that uses our IR remote control to compose text messages using the multi-tap text entry system, and sends text messages back and forth between our two CC3200 LaunchPad boards over UART1.

According to the Lab Manual, our application must allow the user to input a text string via the IR and display the message on the local OLED. After the ENTER key is pressed the message must be transmitted to the remote board and displayed on the remote OLED display. In multi-tap texting, the intermediate characters should be displayed. For example, when sending the character 'z' the '9' button will be pressed 4 times, so the intermediate characters 'w', 'x', 'y' will appear at the current character position before 'z' appears. Pressing the 'DELETE' key should delete the last character, to allow for correction of an errant character before a message is sent.

**Text Messaging Shown on OLED and Terminal with Sender at the Left and Receiver at the Right**





This Part of the Lab is more complicated. It is done in three main functional components: Remote Control Input (including multi-tap texting), OLED display and UART communication. All detailed implementation can be found in Appendix I.

**Remote Control Input**

We implement this part of the code using a globally declared two-dimensional matrix to keep track of what the current editing character is. A row index and a column index are also declared globally to do the book keeping. They are both initialized to be -1. Pressing a button basically changes the row index and column index that is pointing to a certain element in the matrix. The array is Initialized in the function InitCharArray(). Basically, the row index is the same as the number that is pressed and the column index will rotate when the same button is pressed for multiple times. The current editing character will be push to a global buffer either when a different button is pressed.

Pressing the MUTE button on our control would simply pop the latest character in the buffer and reset the button indices to -1.

Pressing the ENTER button will trigger a UART communication session with the other board which will be discussed further.

**OLED display**

This part of the code are designed to accurately reflect the state of the system. We have functions to delete and display a single character and to move the global cursor. All the functions are internally trimmed to not go out of the screen. The purpose of this part of the program is to accurately reflect what is in the transmit buffer and what is the character that is being current edited.

**UART Communication**

This UART routine will be triggered after the ENTER button is pressed. This is achieved by using both UART receive timeout and receive register interrupts. We played around the trigger level and eventually settled at 4\_8 which is ½ of the 16 byte buffer. When the interrupt routined is served, the receiving board will read all the available characters in its receive register. The following images are waveforms captured during the UART session. It is worth noting that we delay the transmission of each character, which gives rise to the long delay between each character in the second figure.

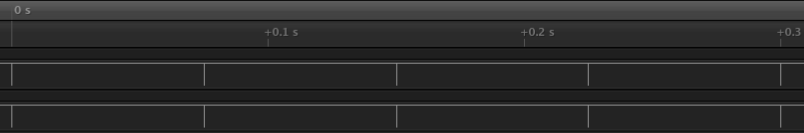
**UART Waveform for Character 'c'**

Bit Pattern (From LSB to MSB): (Start Bit: 0) 1100 0110 (End Bit: 1)

ASCII Code for 'c' (From MSB to LSB): 0110 0011

They match each other.

**UART Waveform for Multiple Characters**

****

**Challenges and Potential Improvements**

1. There are apparently two different types of IR receiver in the lab and the one we got was not the one in the technical document. This caused a little bit of confusion which was later solved by changing to the right one.

2. When we started, we were very confused about how to print each character on the screen. Also we were not sure how does the board configuration work. For the last part, we kept getting garbage characters at the end of strings on the screen. Finally we realize that it was because IR interference. More importantly, we realized the disastrous impacts of the bad programming practices in our program. We implement everything in the interrupt routine which is not encouraged. Although we eventually tweaked the system and had it working. We are definitely not going to do it again in subsequent labs.

**Conclusion**

After finishing this lab, we have learnt how to use your Saleae logic analyzer and an IR receiver module to characterize the transmissions for buttons from an AT&T IR remote control for a specific TV. We also learnt the importance of good programming practices especially in systems involving data transmission and timer interrupts.

**Reference**

AT&T U-verse TV Standard Remote Control User Guide

CC3200 Peripheral Driver Library User's Guide

CC3200 SimpleLink™ Wi-Fi® and Internet-of-Things Solution, a Single-Chip Wireless MCU Data Sheet

**Appendix I: Code for main.c and pin\_mux\_config.c**

**Appendix II: Remote Control Configuration**

Your remote may have been previously configured so you should first reset the remote to factory defaults as follows:

1. Press and hold the AT&T key along with the OK key. Hold both keys for one second, then release. All the mode keys should flash twice.

2. Enter 900 using the numeric keypad. The AT&T key will give a long flash to confirm success.

3. If the remote control times-out before you complete the procedure, you will need to start over.

By default, the backlight is set to ON. You should disable backlighting to extend battery life. To toggle the backlight setting from ON to OFF (or vice versa), use the following steps.

1. Press and hold the AT&T key along with the OK key. Hold both keys for one second, then release. All the mode keys should flash twice.

2. Enter 991 using the numeric keypad. The AT&T key will give a long flash to confirm success.

You will program a Device code for a specific TV, as assigned by your TA. A TV Device code is a 4-digit number that must start with “1”. The IR transmission encoding for a specific TV varies by manufacturer and TV model so there are different Device codes for different TVs.

1. Press and hold the TV key along with the OK key. Hold both keys for one second, then release. All the mode keys should flash twice to indicate that you are in the programming mode.

2. Enter the 4-digit Device Code using the numeric keypad. 3. The TV key will give a long flash to confirm success.

Next you will configure the remote so that Channel and Volume commands always control one device, namely, the TV that you just specified. For channel control using the numeric keypad to be sent to your TV , perform the following steps:

1. Press and hold the AT&T key along with the OK key. Hold both keys for one second, then release. All the mode keys should flash twice.

2. Enter 966 using the numeric keypad. The AT&T key will flash twice to confirm success.

3. Press the TV key. The TV key will give a long flash to confirm success.

For volume control commands to be sent to your TV, perform the following steps:

4. Press and hold the AT&T key along with the OK key. Hold both keys for one second, then release. All the mode keys should flash twice.

5. Enter 955 using the numeric keypad. The AT&T key will flash twice to confirm success.

6. Press the TV key. The TV key will give a long flash to confirm success.