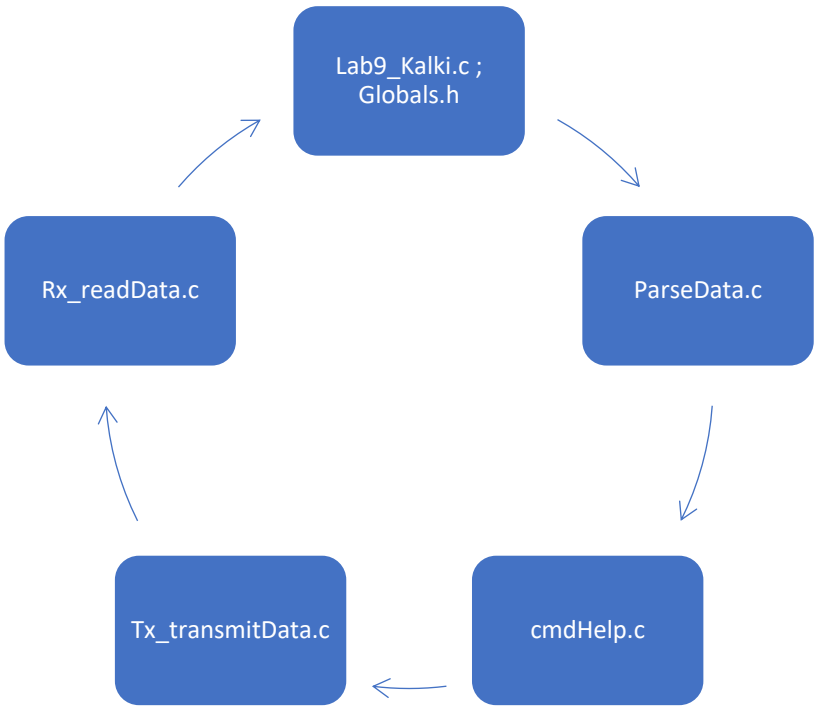
CSE 5400 Project Report

Name: Sai Kalki Abhinav, Gannavarapu
ID : 1002060436

Overview:

We are creating data packets that follow IR LED data format (config pulse and stop bit) and receiving that using a receiver connected to TIVA.
We have also encoded intelligence into this using putty so that the input commands are translated into bits that need to be transmitted / received
Based on this we have formed a shell interface that can be used to initiate transmission of different bytes with different functionalities implemented for them. If we are on the receiving side, then we handle the commands differently for different packets received. For example, we could send out a poll command and receive an acknowledgement from another device as a response OR we could directly set the LEDs on another device ON /OFF based on what their node addresses are. The underlying functionality of this IR transceiver is described below.

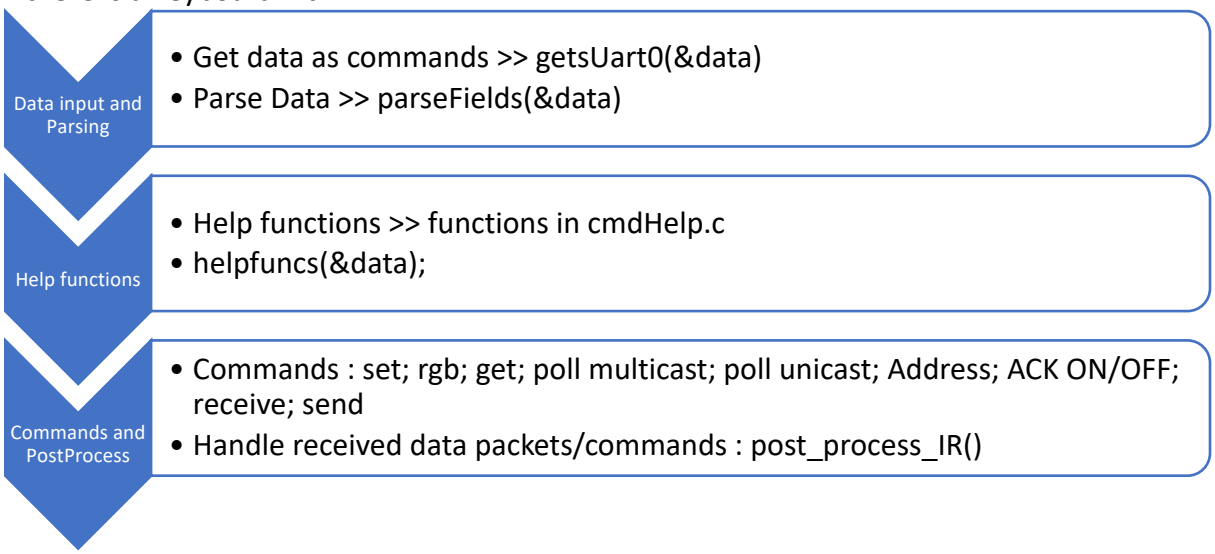
Project Flow:



Lab9_Kalki.c

- This contains the initialization of the following hardware:
    - UART for taking input from putty and displaying data on putty
    - Baud rate of 115200
    - Receiver Hardware: Wide Timer 1, PC6 with GPIO interrupts ON
    - Transmitter Hardware: PWM module 0, gen 0 on PB6; Timer 1 in One shot count down mode
    - EEPROM and RGB control with PWM is initialized along with receiver hardware
    - #include “Globals.h” >> Contains the #defines of all the project

- If there is a keyboard hit:



Tx_transmitData.c

- Initialized a 2D FIFO, Tx_FIFO assuming each letter inputted is stored as 8 bits of data
- FIFO_wrIndex , FIFO_rdIndex and Tx_temprd are FIFO read and write indexes
- Transmit_packet is the global packet of data being sent out (used for readability)
- isMSB =1 for MSB processing and 0 for LSB processing

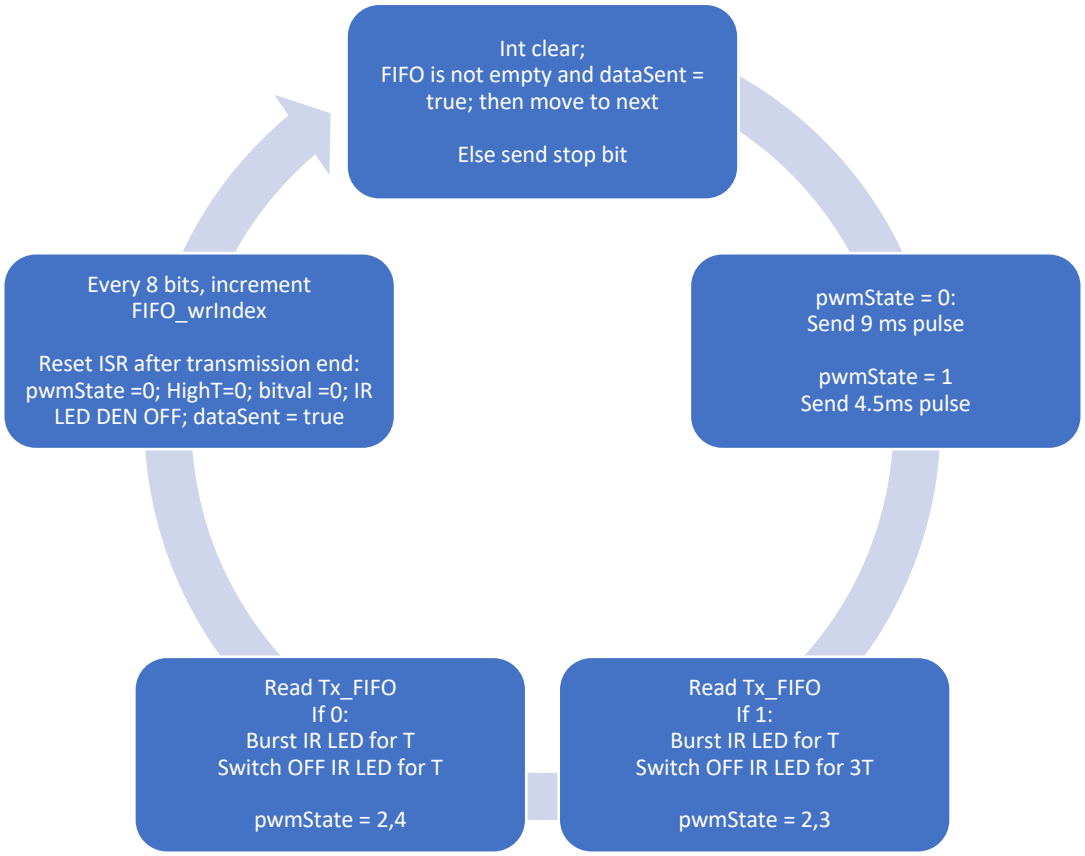
void init_Txpwm()

- PWM module 0 Gen 0 is enable (PB6) and a one shot count down timer 1 A is configured
- IR LED is connected to PB6. DEN bit is used to control the IR LED burst and Timer is used to control how long the burst will be ON /OFF for

void Tx_sendByte(uint8_t byte)

- Loads data into the Tx_FIFO one byte at a time
- If the ISR has not already been triggered (based on the ISR state machine conditions) then after the first byte is loaded **transmitIsr()** is triggered to start transmission (prime the pump)

void transmitIsr(): State machine



void setIR(USER_DATA* data)

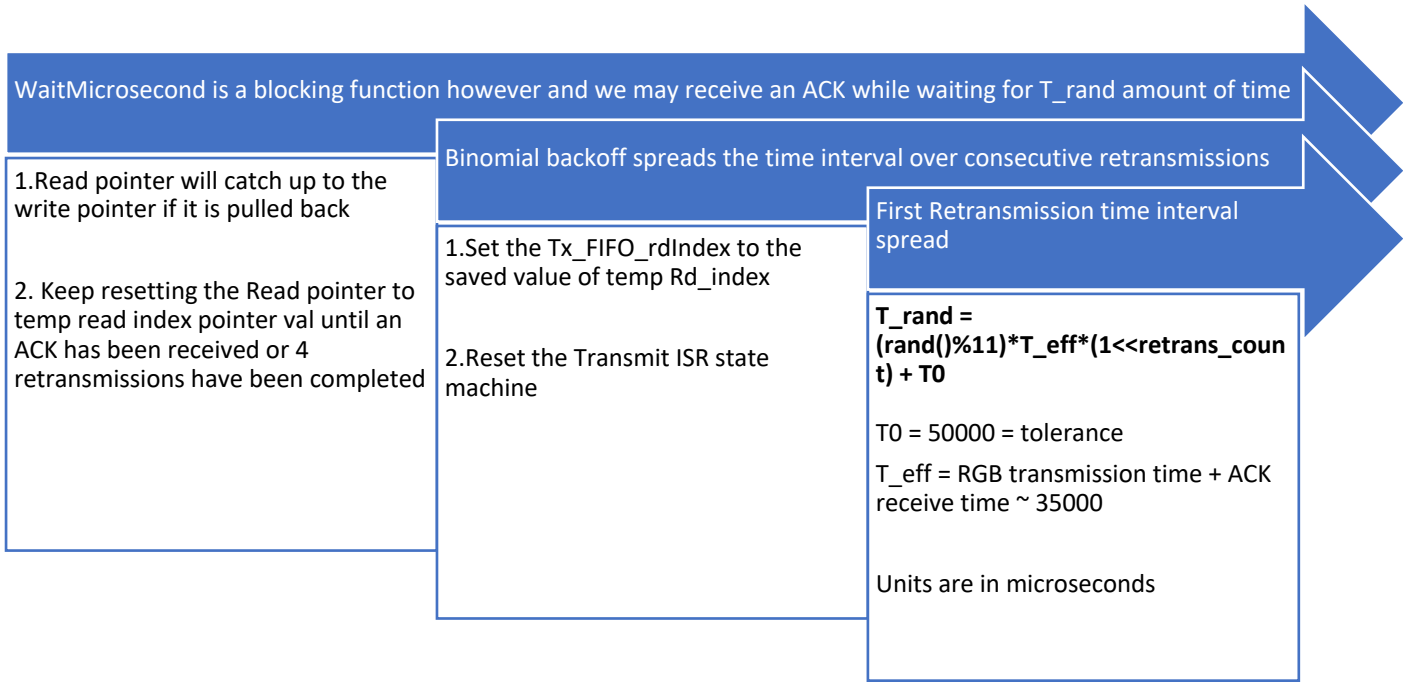
- Get the DST_ADD and DATA from putty. Reset the transmit ISR state machine (dataSent = false)
- Load the transmit_packet with DST_ADD, SRC_ADD(stored in EEprom), ARG_LIST, CHECKSUM
- Use **Tx_sendByte(int byte)** to send each of the above bytes
- Store the FIFO_rdIndex into temp_rdindex to be used for retransmission by retransmit()

Supported Commands:

- The transmission format is the same for all Commands and can be inferred from above. Below we can see what we are transmitting:
  - **void setIR**(USER_DATA* data) : DST_ADD, SRC_ADD(stored in EEprom), ARG_LIST[0], CHECKSUM
  - **void setRGB(USER_DATA* data)**) : DST_ADD, SRC_ADD(stored in EEprom), ARG_LIST[0], ARG_LIST[1], ARG_LIST[2], CHECKSUM
  - **void getIR(USER_DATA* data)**: DST_ADD, SRC_ADD(stored in EEprom), CHECKSUM
  - **void getpoll(USER_DATA* data)**: DST_ADD, SRC_ADD(stored in EEprom), CHECKSUM
    - DST_ADD is 0XFF for multicast poll and specified address for unicast poll
- **void sendACK**() :
  - Acknowledgement packet: DST_ADD,SRC_ADD, CMD_ACK,CHECKSUM
- **void get_response**()
  - Get response packet: DST_ADD, SRC_ADD, CMD_GETRESPONSE, ARGLIST[0],CHECKSUM
- **void freeform_Send(USER_DATA* data)**:
  - This is used as a debug functionality to send specific arguments. In putty used the following input
  - send>>cmd>>dst_add>>src_add>>arg0>>arg1>arg2>>checksum

- **void retransmit**(): Retransmit the commands if you are expecting an ACK but did not receive one

Retransmission strategy:



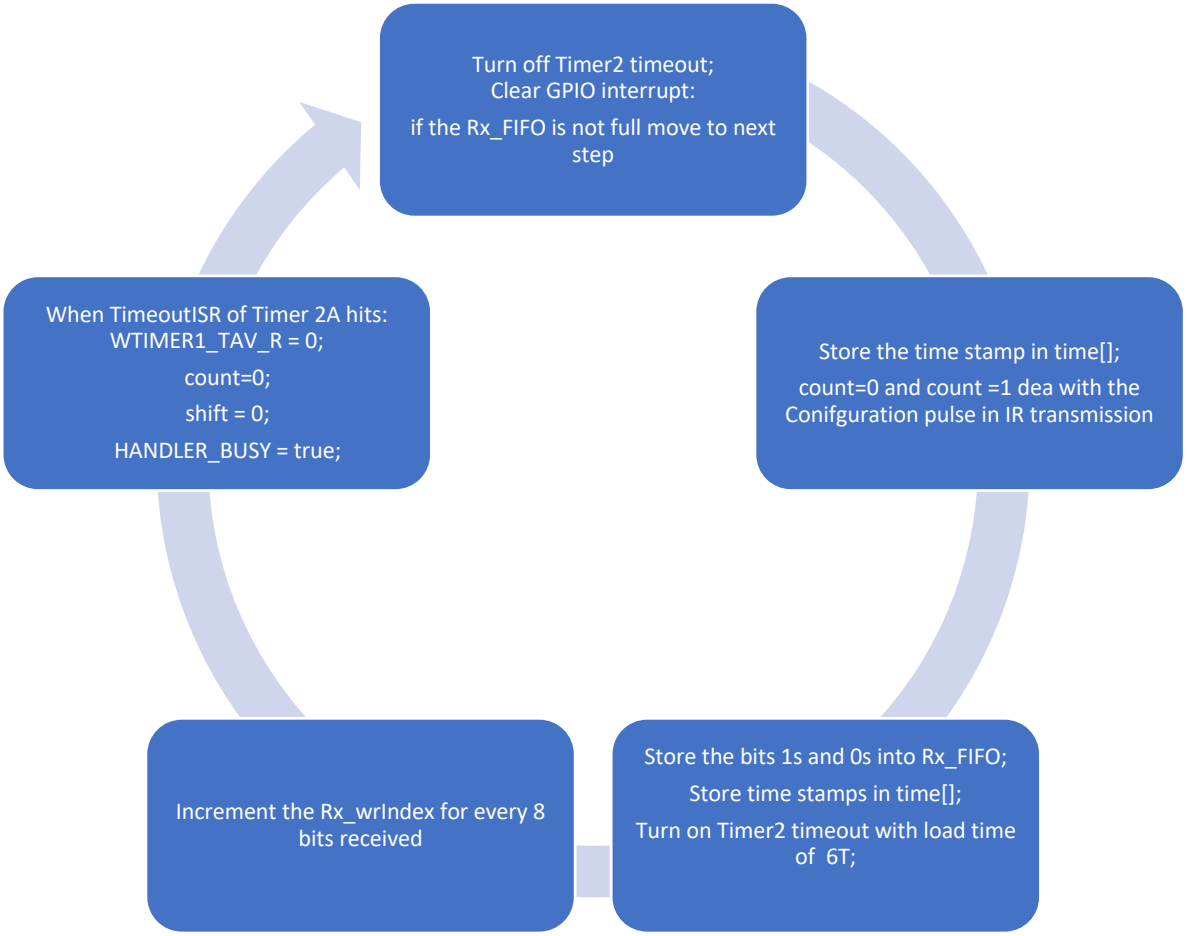
- For POLL and sending an ACK we are also waiting a random amount of time calculated by `T = (rand()%11)*(200000);`

Rx_readData.c

- Initialized a 2D FIFO Rx_FIFO[][] to store bits sent
- The time[] array stores the timestamps based on the falling edges of the receiver output
- SIGNAL_IN on PC6 (WT1CCP0)
- Wide Timer 1 in count up mode configures the timer in counter mode and we use this to store the time stamps into the time[] array
- Timer 2A is configured in 1-shot count down mode to be used for detecting the stop bit / end of transmission
- Port F R, G, B LEDs are configured, and they are configured to run in the alternate PWM mode on PF1, PF2, PF3. We load in the comparator values to control the respective LEDs brightness level

- **void receiveIsr**(): Triggered by GPIO interrupts from SIGNAL IN on PC6

Receiver State Machine:



void post_process_IR()

- Go through the Rx_FIFO until Rx_rdIndex != Rx_wrIndex
- Store the decimal values of the bits received in Rx_packet_handle[] which is a 1D array and easy to read data out of for further processing
- Store data into the receive_packet (similar to transmit_packet)
- If the DST_ADD of the received data is not one of our current addresses or 0XFF then print an error messaage and flush the FIFO
- Calculate the checksum and compare it against the received checksum. If both are not equal then print an error message and flush the FIFO
- If no errors then **Rx_cmd_handler**() is called

Rx_cmd_handler()

- Set the RGB LEDs based on the command received.
- For set command, set the respective LED ON with corresponding brightness levels. Input is in the acceptable range from 0 -255

- For rgb , set the 3 comparator values from the 3 arguments received
- If get is the command received, then build a GET_RESPONSE packet with the value of the comparator at the requested address loaded in the packet and transmit the built packet
  - Get response packet: DST_ADD, SRC_ADD, CMD_GETRESPONSE, ARGLIST[0],CHECKSUM
- If poll is the received command, then build an acknowledgement packet and send it out
  - Acknowledgement packet: DST_ADD, SRC_ADD, CMD_ACK,CHECKSUM
- Before exiting check if the SET/RGB commands requested an ACK and if they did build an ack packet and send it out as mentioned above

void sendACK()

- Acknowledgement packet: DST_ADD, SRC_ADD, CMD_ACK,CHECKSUM
- This is transmitted using Tx_sendByte() function mentioned above

void get_response()

- Get response packet: DST_ADD, SRC_ADD, CMD_GETRESPONSE, ARGLIST[0],CHECKSUM
- This is transmitted using Tx_sendByte() function mentioned above

void printPacket();

- If **"receive"** is typed in putty then the last received packet is printed out

Putty supported Commands

The following are the putty supported commands

```
putsUart0("clear >> clears putty \n");
putsUart0("reboot >> reboots the texas board \n");
putsUart0("BlueON, BlueOFF \n");
putsUart0("RedON, RedOFF \n");
putsUart0("GreenON, GreenOFF \n");
putcUart0('\n');
putsUart0("COMMANDS.... \n");
putsUart0("ACK ON | ACK OFF \n");
putsUart0("get DST_ADD \n");
putsUart0("poll | poll DST_ADD .. ACK should be OFF \n");
putsUart0("set DST_ADD DATA \n");
putsUart0("rgb DST_ADD DATA1 DATA2 DATA3 \n");
putsUart0("send DATA1 DATA2.. send free form data \n");
putsUart0("receive...print the last received packet \n");
putsUart0("clr .... reset state for receiveISR \n");
putsUart0("MSB .... process with MSB first \n");
putsUart0("LSB .... process with LSB first \n");
putsUart0("trans_off...Turn off retransmission \n");
```