**Pseudocode for TransmitService.c**

This service is responsible for framing, sending data packets and executing Goggle.

**Data private to the module:**

```
static uint8_t data_array[5];
static uint8_t array_size;
static uint8_t frame_id;
static uint8_t destination[2];
static uint8_t Transmit_IF = 0; //flag to post packet sent event
static uint8_t Broadcast_Flag = 0;// goes high on receiving ES_Sing
static uint8_t bytes_remaining;
static uint8_t index;
static uint8_t MyPriority;
static GoggleState_t CurrentState;
static uint8_t FrameLengthMSB;
static uint8_t FrameLengthLSB;
static uint8_t API_ID;
static uint8_t frame_id;
#define DestAddressMSB 0x21//changed for checking
static uint8_t DestAddressLSB;
static uint8_t Options;
static uint8_t XBEE_Packet_Size;
static uint8_t XBeePacket[Max_Size];
```

**Constants:**

```
#define Control_Size 4 //also sing Ed is in control byte
#define Pairing_Size 5
#define Unpairing_Size 5
#define Broadcast_Size 1
#define One_Sec 100
#define Mil_Sec 20// 200ms timer
#define Transmit_Time 20//packet to be sent after 200ms
#define Pair_Timer 0//to send control byte in 1sec
#define Ack_Timer 1//to send pairing request again every 200ms
#define Transmit_Timer 2// to scan and transmit control packet every 200ms
#define Max_Size 14//maximum size of packet
```

**InitTransmitService**

Initialize CurrentState to Waiting_to_Pair
Configure the registers for UART by setting the baud rate, choosing the 8 bit counter and setting the PPS register to RB6. Configure RB7 as receive
Configure output and input pins for:

- Sing Ed button,
- IT status LED,
- Pairing status LED,

- Foot Pedal,
- Vibe motors

Post ES_init to the TransmitService

**PostTransmitService**
Return PostToService

**RunTransmitService**
The EventType field of ThisEvent will be one of: ES_TIMEOUT, ES_Pair, ES_AckReceived, ES_Unpair, ES_Broadcast

Returns ES_NO_EVENT if No Error detected, ES_ERROR otherwise
local var ReturnValue initialized to ES_NO_EVENT
Based on the state of the CurrentState variable choose one of the following blocks of code:

**If CurrentState is : Waiting_to_Pair**

Initialize the Pair LED to off
If the Event is ES_Init
move to Waiting_to_Pair state

Endif

If the Event is ES_Pair
    Call Pair_Packet( ) to make pair data
    Call Construct_XBEE_Packet() to make XBEE packet
    Call Start_Transmit() to start transmit
    Initiaize the Pair timer and Ack timer
    Set the CurrentState to Trying_to_Pair
Endif

Break

**If CurrentState is : Trying_to_Pair**

If the Event is ES_TIMEOUT with param as Ack timer
    Send pair packet again
    Initialize the ack timer
Endif

If the Event is ES_TIMEOUT with param as Pair timer
    Move back to Waiting_to_Pair state
Endif

If the Event is ES_AckReceived
    If the frame header is 0xff and it is addressed to you

Call Control_Packet()
Then call Construct_XBEE_Packet and Start Transmit functions
Move to Paired state
Endif
Endif

Break

**If CurrentState is Paired:**

If Event is TIMEOUT and param is Transmit Timer
Construct and send a control packet
Initialize Transmit Timer
Endif

If Event is TIMEOUT and param is Pair_Timer
Call Unpair_packet() to send an unpair packet to WHA:LE
Call Construct_Packet and Start Transmit Functions
Move to Waiting_to_Pair
Endif

If Event is ES_Unpair Call Unpair_packet() to send an unpair packet to WHA:LE
Call Construct_Packet and Start Transmit Functions
Move to Waiting_to_Pair
Endif

If the Event is ES_Broadcast
Raise the Broadcast_Flag
Call control packet()
Construct packet and Start transmit
Initialite Transmit_Timer
Endif

Break

If case if default
Break

return ReturnEvent

**Function: static void Construct_XBEE_Packet()**
Initialize checksum to 0
Start filling array XBEEPacket in the following manner starting from index 0:

- Write delimiter
- Increment checksum
- Length of packet
- Increment checksum

- APID
- Increment checksum
- Frame_id
- Increment checksum
- Destination MSB
- Increment checksum
- Destination LSB
- Increment checksum
- Then add data packet byte and increment checksum at after every step
- Set the last byte of XBEEPacket = 0xff-checksum
- Define XBEE_Packet_Size

**Function : static bool Start_Transmit():**
Initialize bytes-remaining to XBEE_Packet_Size
And index to 0
If the transmit flag is set
    Load first byte of data packet
    Dec bytes_remaining
    Inc index
    If transmit flag is set
        Load 2nd byte
        Dec bytes_remaining
        Inc index
        Enable transmit interrupt
        Enable peripheral interrupt
        Enable global interrupt
        Endif
    Return true
    Endif
    Else
    Return false

**Function : static void Pair_Packet()**
Define data_array and fill up its elements as per communication protocol as follows:

- At 0: pair_header
- At 1: Goggle_MSB
- At 2: Goggle_LSB
- At 3: DestinationMSB
- At 4: Query the Destination MSB as it will change from WHA:LE to WHA:LE

Define array size as 5
Frame_id as Pair_header
Destination[0] as  DestinationMSB
Destination[1] as QueryDestination()

**Function : static void Control_Packet()**

Initaite Update_Flag = 0// the contents of packet will update only when this flag is 0
i.e after every 3 seconds, otherwise send previous control packet
Intitiate Old_Driving  and Old_Direction
Data_array[0]= control_header
If update_flag is 0
    Data_array[1]= QueryDriving()
    Save the value of Data_array[1] in Old_Driving
    Data_array[2]= QueryDirection()
    Save the value of Data_array[2] in Old_Direction
Endif

Else set data array [1] and [2] to Old_Driving and Old_Direction respectively
Data_array[3] = QuerySpecial() to set special commands
Define array size as 4
Frame_id as Control_Header
destination[0] = DestAddressMSB
destination[1] = QueryDestination()
Inc Update_Flag by 200

**Function : static void Unpair_Packet()**
Initialize elements of data_array as follows:

- At 0: Unpair_header
- At 1: 0xDE
- At 2: 0xAD
- At 3: 0xBE
- At 4: Q0xEF

Define array_size as 5
Frame_id as Unpair_Header
destination[0] = DestAddressMSB
destination[1] = QueryDestination()

**Function void Transmit_Next()//** this function will be called from ISR as the variables that it uses are in TransmitService, so function is defined here
Load next byte into transmit register at interrupt
Dec bytes_remaining
Inc index
If Bytes_remaining  == 0
    Disable transmit interrupt
Endif

**Function : static uint8_t QueryDriving(void)**
If left-hand motion is deducted and foot pedal is not pressed or right-hand motion and foot pedal is not pressed
    Return move forward and specify speed
Endif

Else If foot pedal is deducted

Return move backwards
Endif

Else return stop

**Function : static uint8_t QueryDirection(void)**
If left-hand motion is deducted and right-hand motion
Return move straight
Endif

Else If only left-hand motion is detected
Return move left
Endif

Else If right-hand motion is detected
Return move right
Endif

Else return stop

Return stop

**Function : static uint8_t QuerySpecial(void)**
If Broadcast_Flag is clear
Return don't sing Ed
Endif
Else return sing Ed

**Function : static uint8_t QueryDestination(void)**
Divide the 0-1024 values into 8 parts to provide pairing slot for each of the 8 teams. The values will come from ADC values from potentiometer for Team select
Return value is based upon the team selected chosen from Communication protocol