Logitech Z-5300 ... Hacked and Wireless !

I know this post is getting old and probably the original authors moved on but some people might still be interested in the alternative for the remote pod of the Z-5300 I propose!

After few hours of investigation (days in fact) I managed to get the system working without the pod but simply with an Arduino UNO and few wiring. This approach doesn't require any soldering or even opening of the Sub unit (in fact I never opened the sub to do this hack).

From an ethical standpoint, I would not have disclosed the following information as it is Intellectual property of Logitech but considering that the product is obsolete and the spare parts are not available anymore (beside ebay market) I consider that it is people's right to build up substitute to enjoy their material longer or differently.
Furthermore, building a hacked remote control when one lost or broke it allows you to avoid a new purchase and disposal of perfectly working equipment. Thus extending the equipment lifetime is also a way to minimize out impact on environment!

I. Why hurting myself with this project

I use my Z-5300 to watch movies in 5.1 and it is also connected to an Raspberry Pi with HifiBerry and AirPlay receiver. I also still have a perfectly working remote pod so why making this project? The answer is that since I introduced a TOSLINK to RCA 5.1channel coverter, it became impossible to use the TV remote to adjust the sound volume. Thus I had to get and reach the pod to change the volume or adjust the Sub, Fad and Center levels.

This post is not covering the remote control via infrared part as there are sufficient amount of tutorials already detailing the process. Also the code is shoud be different with each remote control you may intend to use. You can still modify the code I am proposing here and make your remote controlled system for the pod substitute.

What follows is to be used at your own risk and under your own responsibility. I cannot be held responsible for any damage you may have on any equipme while trying to reproduce this system.

My electronic knowledge comes from my old student times which means around 10 years and no real practice in between so purist may find what follows a bit harsh but it worked fine for me.

I also tried to make this tutorial the simplest possible (sometime "for dummies" style) so that complete novice in electronic and Arduino can step by step reproduce the project while understanding what they are doing.

II. The remote pod: how it works

This is a pretty simply designed board composed of a selector and 3 push buttons:
Power
Mode select
Matrix

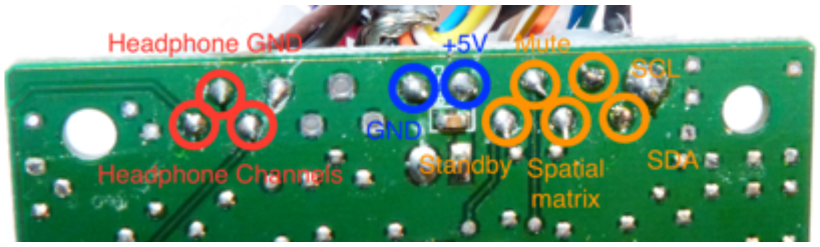
The pod also has an audio jack connector which I have never used.

Finally it is connected to the sub unit via a VGA connector similar to the one used for display.

The pod contains a microcontroller in charge of collecting the information via the buttons and send the commands accordingly to the sub internal microcontroller

III. The pod pin allocation

Based on a previous post that identified the pin (and I thank the author for that precious kick off help) I identified 3 groups for the pins:
Group Jack: this group is connected independently to the jack on the pod board. I will ignore this as it has no impact on the control electronic and is not connected to the pod micro controller.



Group power

- Pin1: GND
- Pin6: +5V

This just powers the pod electronic board of the pod. It is constantly on when you turn on the sub via the main switch in the back of the sub unit.

Group control

- Pin11: Standby
- Pin12: Mute
- Pin13: matrix
- Pin14: SCL
- Pin15: SDA

The pins 14 and 15 hold the key for the control of the unit and will have all our attention. However we will not completely ignore the pin11 and 12 as they help us turning the system out of standby mode....

IV. The control principles

a. Powering up:
The pod is always powered up by the sub when the switch is on.

b. Getting out of standby:
When you press the on/off button here is what happen in a sequence:
- the standby pin 11 is set to the ground which activate the sub
- The mute pin 12 is also set to the ground
- The pod sends a series of bytes to the sub via the pins 14 and 15 working on i2c principles (I'll come back more in details on that later)

The series of command sent by the pod at startup has one single objective. The command series aims to gradually ramp up the volume in order to reach the last level set by the user when the latter turned the pod off. This user setting is stored in the pod and not in sub. For proof if you set the volume to a certain level, turn off the pod, disconnect the pod while maintaining the main general switch on, the unit will get in standby mode. Reconnecting the pod and turning it back on will not set the volume to last user setting but to minimum. Note that I say “minimum” and not zero. The zero for the pod means “volume to the maximum”. I'll come back on that.

c. Matrix button
I never liked the effect of this button but anyway. Pressing the button will set pin 13 to ground activating the effect. The pod also sends a tiny volume reduction command in parallel more for user comfort than for actual usefulness.

d.mode selector
The mode selector has no direct impact on the sub. It simply indicate to the pod what type of command to send to the sub when turning the selector. It also starts a timer to get back to volume mode if the user does not change the setting of the selected parameter after a certain time.

d.the selector
The selector is what triggers the command. It sends 2 types of indications to the pod's Micro controller: up or down
Every time you turn it and increase or decrease by one indentation the micro controller sends to the sub one command and one only!

V. The commands

The pod communicates with the sub using the i2c/twi standard on pin 14 & 15. Please go on wiki if you don't see what I am talking about.
Basically each command is composed of 3 instructions:

- the address: the pod is the master and is sending the address 68 which is the sub microcontroller address. This is also followed by the write instruction to indicate to the sub that more will follow
- The speaker code give an indication as of which speaker the command will apply. If for example this code is 0 that means general volume. I'll detail the different code I identified later
- The level to be set to.

As an example if I want to set the volume to the middle position, here is what happens in a sequence

1. Start
2. Address 68 + 0 write
3. Nature 0
4. Level 30
5. Stop

In human world that would mean:

1. "Hello everyone I am on the line as the master in the pod and I am opening the channel!"
2. "Hey sub! Yes, you with the number 68. I am about to give you instructions"
3. "So you will now change the volume"
4. "You will set it to 30 on a scale from 0 to 61"
5. "That's it for now I hang up!"

In the i2c standard, with exception to the start and stop signals each step corresponds to one byte (8bits) separated by an acknowledgment 0 bit emitted by the sub. That's theory because as you will see reality is a bit more complex.

Indeed our beloved sub is acknowledging by sending a 1 bit in spite of the standard definition.

Having said that here are the different commands I could identify:
Address: 68 (7bit) + 0 write bit
Which in bit world correspond to 10001000

Speaker codes:

- 0 (00000000 in bit world) applies to all speaker simultaneously and thus aims to set the volume
- 1: Front left speaker. The standard remote pod does not use it. If you want to finer tune the setup with the remote you may be interested.
- 2: Front right speaker. Idem as FL.
- 3: Center speaker
- 4: Rear Left speaker (RL)
- 5: Rear Right speaker (RR)
- 7: Subwoofer (SW)
- 8: this code is not designing any speaker (obviously) but is used to mute or unmute one or several specific speakers in the system

Level:
For volume:

- 80: equivalent to mute
- 61: level minimum
- 0: level maximum
- All values between 61 and 0 indicates a level

For Subwoofer

- 85: full left
- 80 almost full right
- 89 full right
- All value between 85 and 80 indicates the centering position

For center

- 12: sub to minimum
- 0. Sub to maximum
- All values between 12 and 0 indicate the sub level to adopt

For fader: (applies to RR &RL)

- 12: full back
- 0: full front
- All values in between

The same code range applies also for the front speakers but the pod is not equipped to send any signal applicable to front right and left speakers. The system is set at maximum 0 as initial setup.

The speaker code 8 works differently compared to the previous ones.

As explained the code 8 aims to Mute or Unmute the volume of one or several speaker depending on the values. With the pod the commands using the speaker code 8 is sent when the Center or Fader volumes are turned to minimum. To understand how the command works requires decomposing the Byte that is sent as associated instruction for speaker code 8.
When the Fader is set to minimum, the pod sends the instruction [8 ; 6] and thus turns both Rear speakers back on.
In binary 6 is equivalent to:
[0 I 0 I 0 I 0I 0 I 1 I 1 I 0]
When turning the fader back up, the first command will be [8;0] to turn the Rear speakers back on.

If you measure the same thing on the Center functions, the pod will switch the center speaker on mute by sending the instruction [8 ; 8].
[0 I 0 I 0 I 0I 1 I 0 I 0 I 0]

In fact exact bit set to 1 in the instruction byte will set the corresponding speaker on mute.

We can match each speaker with the following mapping:
[0 I 0 I FL I FRI C I RL I RR I 0]

As an example, If I want to mute all the speaker on the rights and the center, I should submit the command: [8 , 26]
Which in binary mapping would give : [0 I 0 I 0 I 1I 1 I 0 I 1 I 0]

Before setting a volume value for a specific speaker, don't forget to set it back on unmute by changing the bit associated down to 0 in the command.

Now that we understand the working principles better, let's start creating our own pod

VI. The electronic setup

What follows is pure prototyping. It works (for me at least) but is far from representing a finalized solution. I hope it will get you through the difficulties I encountered to be able to control the sub with an alternative hardware to the supplied pod (especially if you lost it).

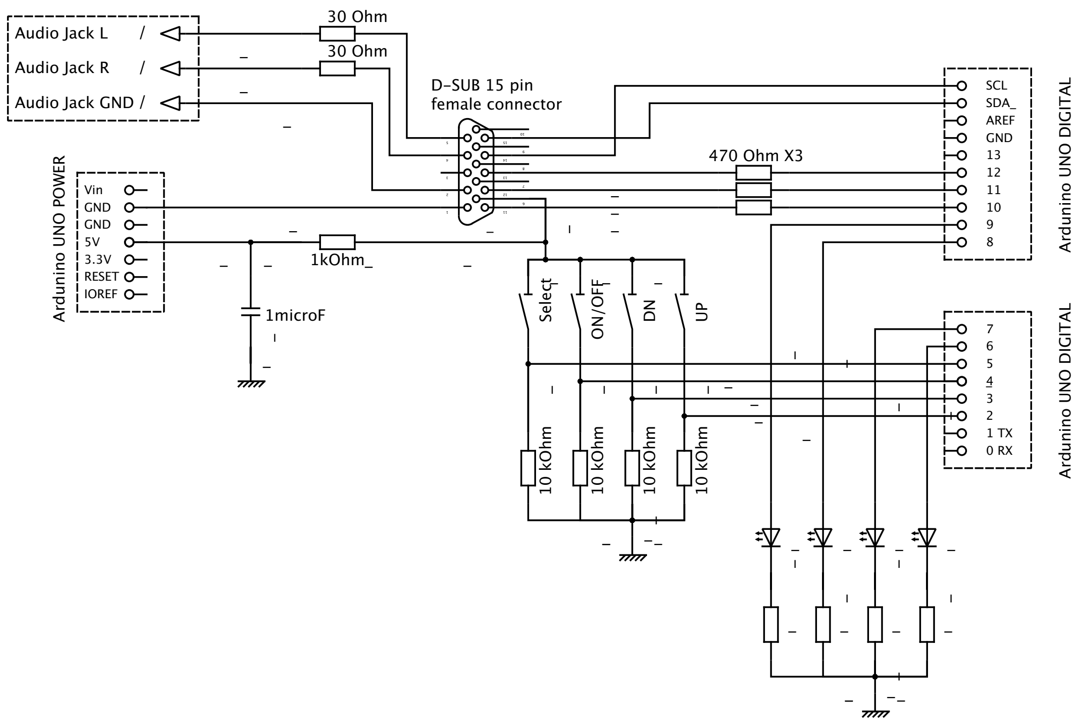
The design I propose here doesn't include either a remote control capability. If you wish to get more details on how to set a IR and basics for remote control please visit the following website: http://www.instructables.com/id/Arduino-

Infrared-Remote-tutorial/?ALLSTEPS

a. Material

- An Arduino UNO
- A breadboard
- A VGA separator (I personal used a VGA extender cable I sacrificed to do the separation of the pin cables)
- A VGA cable with a female connector (usually in extenders)
- Some male female inverters (depends on the cable and separator configuration you have)
- 4 push buttons
- 3 resistors 470 Ohms
- 4 resistors 220 Ohms
- 1 resistors 1 kOhms
- 4 resistors 10 kOhms
- 2 resistors 30 Ohms
- 1 capacitor 1microF
- 4 LED (same or different colors as your preference)
- Jumper cables

b. Build up the electronic circuit



We are going to make a simple 4 buttons pod:

- On/Off
- Selector
- Volume UP
- Volume DOWN

Set up the 3 buttons the same way with one resistance to the ground and the 5v connected to the push button
The line between the resistance and the button shall be connected to one digital output from the Arduino

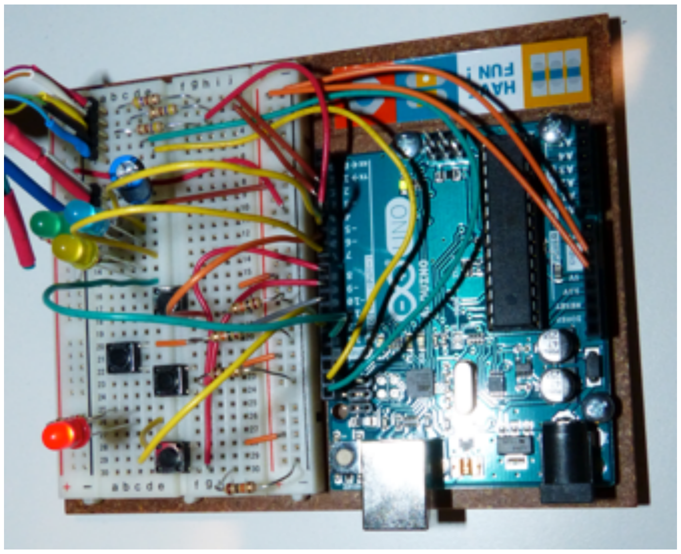
Using the VGA separator, connect the following pins to the Arduino:

Pin 1 ground to Arduino ground
Pin 6 5v to Arduino 5v (optional if Arduino is powered via USB)
Pin14 to SCL of the Arduino
Pin15 to SDA of the Arduino

If you want to focus on the volume control first and see the on/off function for another time then directly connect the pins 11 and 12 to the ground. Otherwise connect them to some Arduino digital output with a 470 Ohm resistance to protect.

That's it you have created the remote! At least from a hardware perspective.

It should look like something like this: (pin connection on the picture may differ from the picture)



VII. Programing the Arduino

Some of you may tell me that they tried this but never get the Arduino or any i2c master device successfully sending commands to the sub.
Some people have successfully managed to send i2c commands by directly soldering the SCL and SDA from the Arduino to the sub internal microcontroller because the i2c commands were not accepted via the VGA. I have managed to have it working without that method because I found out that the system is not complying with all specification of the i2c standard!

When you look at the i2c specs from NXP and the way the Arduino Wire library works, after each byte transmitted the slave has to Acknowledge reception via a 0bit!
That's not the case with the sub microcontroller in the Z5300. It acknowledges by sending a 1bit which leads the Arduino to think that the sub did not acknowledge reception.
As a consequence the Arduino immediately stops sending information after the address byte.
The sub thinks then that the pod only sends a series of zero which means "volume to max" and then set all the speakers to maximum.

This is the real challenge I had to face here!

As the only thing you have control on is the Arduino software, my idea was to adapt the Wire library so that the Acknowledgement becomes a 1bit instead of a 0 to adapt the the protocol to the one expected by the Subwoofer.
I went and reviewed the Wire.h file and found out that the real file to modify should be one of the files in the library folders called twi.h.
You will find the right file at the following folder (I'm using a Mac, I have no idea where the file might be on windows… sorry) in the Arduino application packages:

Contents/Java/hardware/tools/avr/avr/include/util/twi.h

The simple thing to do (after saving a copy of the original twi.h file just in case) is to invert the NACK and ACK value definition for the master transmitters MT.

In the twi.h file, modify the following values to reverse the Acknowledgment protocol:

```
/* Master Transmitter */
/** \ingroup util_twi
    \def TW_MT_SLA_ACK
    SLA+W transmitted, ACK received */
#define TW_MT_SLA_ACK            0x20   // Original value 0x18

/** \ingroup util_twi
    \def TW_MT_SLA_NACK
    SLA+W transmitted, NACK received */
#define TW_MT_SLA_NACK           0x18  // Original value 0x20
```

```
/** \ingroup util_twi
    \def TW_MT_DATA_ACK
    data transmitted, ACK received */
#define TW_MT_DATA_ACK           0x30    //Original value 0x28

/** \ingroup util_twi
    \def TW_MT_DATA_NACK
    data transmitted, NACK received */
#define TW_MT_DATA_NACK          0x28         // Original value 0x30
```

Once done, you just have to write the following code lines and load the program in the Arduino. If you are not satisfied with the pin allocation for the button and the LED, you just have to change the values in the allocation code line in the beginning of the following program in the Arduino programmer software:

```
#include <Wire.h> //Library Wire with modified Acknowledged function in I2C
protocol
#include <IRremote.h> // Optional Remote control library (for those who want to
control the Z5300 with a remote control

//Bouton
const char up_pin = 2;// Up
const char dow_pin = 3;// Down
const int onoff_pin = 4; //on/off
const char sel_pin = 5; // Option Selector
//OUtput fonctions to SUB
const char stdy_pin = 10; //standby
const char mute_pin = 11; // Mute
const char mat_pin = 12;// Matrix
//LED status indicator
const char red_pin = 6; //Red LED for Volume
const char gre_pin = 7; // Green LED for Sub
const char yel_pin = 8; //Yellow PIN for Fader
const char blu_pin = 9; // Blue LED for Center

//Pin for Remote Control Input
const char irr_pin = 13; //IRRemote input (if any IR detector) on pin 13

int select = 0;// 0 volume, 4
int vol = 80;//original 80 for test 30
int sub = 82;// Original volume for the sub set to 82 initial setup when sub
turned on
int fad = 6; / Original volume for the fader set to 6 initial setup when sub
turned on
int cent = 6;/ Original volume for the Center set to 6 initial setup when sub
turned on
int lr = 0;// Original volume for the sub set to 82 initial setup when sub
turned on
int timer_select;

bool onoff= 0;//On off indicator boolean
bool mute = 0;//Mute indicator boolean
bool matrix = 0;// Matrix mode indicator boolean (not used here. Just add a
button on a pin and an associated function to toogle mat_pin with digitalWrite
function)



    IRrecv irrecv(irr_pin);//Start IR Remote receiver
    decode_results results;// store the result of the last IR command received

void setup() {
    //Setup of Serial IN/OUPUTS
    Serial.begin(9600);
    Wire.begin(); // join i2c bus (address optional for master)
    irrecv.enableIRIn(); // Start the IR Remote receiver

    // Buttons OUTPUT
    pinMode(up_pin, INPUT);//up Button
    pinMode(dow_pin, INPUT);//down Button
    pinMode(onoff_pin, INPUT);//on/off button
    pinMode(sel_pin, INPUT);//selector pin

    //output onoff mute and matrix for the SUB
    pinMode(stdy_pin,OUTPUT);
    pinMode(mute_pin,OUTPUT);
    pinMode(mat_pin,OUTPUT);

    //Setup off mode and matrix OFF
    digitalWrite(stdy_pin,HIGH);
    digitalWrite(mute_pin,HIGH);
    digitalWrite(mat_pin,LOW);

    // LED setting
    pinMode(red_pin,OUTPUT);//set red LED for volume
    pinMode(gre_pin,OUTPUT);//set green LED for sub
    pinMode(yel_pin,OUTPUT);//set yellow LED for Fader
    pinMode(blu_pin,OUTPUT);//set blue LED for Center

    // Set the off status as startup status
    digitalWrite(stdy_pin,HIGH);
    digitalWrite(mute_pin,HIGH);
    digitalWrite(mat_pin,HIGH);
    Subset(0,80);//Reinitialisation of sound to lowest

    Serial.println("Good to control!");
}


void loop() {

 int temp_timer = millis();//record time for the loop cycle

// ON/OFF Toogle Button
  if(digitalRead(onoff_pin)==HIGH){
    Toogleonoff();
    delay(100);
  }


if(onoff==1){// give access to functions volume capable of adjusting the vol
varialble only if the system is on

 // Mode Selection Button
  if(digitalRead(sel_pin)==HIGH){
  select=modeselect(select);
  timer_select=millis();
   // matrix=abs(matrix-1);
  //digitalWrite(mat_pin,matrix);
  //delay(100);
  while(digitalRead(sel_pin)==HIGH){

  }
  }
// BUTTON VOLUME UP
  if(digitalRead(up_pin)==HIGH){
      digitalWrite(red_pin,LOW);// turnoff Red LED to show command reception
     timer_select=millis();// reset the automatic return to volume mode to new
value due to press button action
     switch(select){//select the mode and command to send accordingly
      case 0:
        vol=volumecal(vol,1);
        Subset(select,vol);//decrementation the volume level
      break;
      case 7:
        sub=Calc_Sub(sub,1);    // decementation of Sub level
        Subset(select,sub);     //Set level to Subwoofer
      break;
      case 4:
        fad=Calc_FadCent(fad,1);// decementation of Fader level
        Subset(select,fad);     //Set level to Rear Left Speaker
        Subset(select+1,fad);   //Set level to Rear Rigth Speaker
      break;
      case 3:
        cent=Calc_FadCent(cent,1);// decementation of SubCenter level
        Subset(select,cent);      //Set level to Center Speaker
      break;
      case 1:
        lr=Calc_FadCent(lr,1); // decementation of Front level
        Subset(1,lr);     //Set level to Front Left Speaker
        Subset(2,lr);     //Set level to Front Right Speaker
      break;
    }
    delay(100);
    digitalWrite(red_pin,HIGH); // turn LED Red back ON
   }


// BUTTON VOLUME DOWN Activate
  if(digitalRead(dow_pin)==HIGH){
    timer_select=millis();
    switch(select){
      case 0:
        vol=volumecal(vol,0);
        Subset(select,vol);//decrementation the volume level
      break;
      case 7:
        sub=Calc_Sub(sub,0);
        Subset(select,sub);//Volume Subwoofer
      break;
      case 4:

        fad=Calc_FadCent(fad,0);
        Subset(select,fad);//Volume Rear Left
        Subset(select+1,fad);//Volume Rear Right
      break;
      case 3:
        cent=Calc_FadCent(cent,0);
        Subset(select,cent);
      break;
      case 1:
        lr=Calc_FadCent(lr,0);
        Subset(1,lr);//Volume Front Left
        Subset(2,lr);//Volume Front Right
      break;
    }

    delay(100);
  }
}

 if(temp_timer-timer_select>5000 & select!=0){ // return to volume mode if no
button is pressed after 5 seconds
 select=modeselect(2);// set on general volume mode
 }
/*Remote control Parts
 * This section until the end of the loop function enables to control the Mute
and the general volume via the remote control
 * This is using an infrared detector suitable with the remote control you wish
to use
 * This code is adapted to my LG TV remote control and will not work if you
have a different brand or model
 * If you wish to use the following you will have to adapt the code to the
remote you wish to use therefore you should identify the way
 * you can control the remote you wish to use with the Arduhno and understand
its protocol (see IRRemote tutorial to proceed)
 */
 if (irrecv.decode(&results)) {
    Serial.println(results.value, HEX);

     if(results.value==0x20DF4EB1){ // code received for ON/OFF of the Unit
     Toogleonoff();
    delay(100);
     }


if(onoff==1){
    if( results.value==0x20DF906F){
      mute=abs(mute-1);
      //digitalWrite(mute_pin,mute);
      digitalWrite(red_pin,LOW);
      if(mute==1){
           digitalWrite(mute_pin,HIGH);
        //Subset(8,32);
      }
      else{
           digitalWrite(mute_pin,LOW);
       // Subset(8,31);

      }
      delay(100);
    digitalWrite(red_pin,HIGH);
    }


    if(results.value==0x20DF40BF){
      bailout:
      digitalWrite(red_pin,LOW);// create a blink LED to acknowledge remote
data reception
      vol=volumecal(vol,1); //decrementation of vol value to increase the
volume level
      Subset(0,vol);  // apply the volume instruction ot the SUB
      irrecv.resume();
      int tim1=millis();
      int tim2;
      while(irrecv.decode(&results)==0  && (tim2-tim1)<200 ){
        tim2=millis();
        delay(50);
        digitalWrite(red_pin,HIGH);
      }
      if (irrecv.decode(&results)){
        goto bailout;

        }

      //irrecv.resume();
        delay(100);
        digitalWrite(red_pin,HIGH);

        }
        else if(results.value==0x20DFC03F){
     keepdown:
    digitalWrite(red_pin,LOW);// create a blink LED to acknowledge remote data
reception
      vol=volumecal(vol,0); //decrementation the volume level
      Subset(0,vol);  // apply the volume instruction ot the SUB
      irrecv.resume();
      int tim1=millis();
      int tim2;
      while(irrecv.decode(&results)==0  && (tim2-tim1)<200 ){
        tim2=millis();
        delay(50);
        digitalWrite(red_pin,HIGH);

      }
      if (irrecv.decode(&results)){
        goto keepdown;

        }
        delay(100);
        digitalWrite(red_pin,HIGH);
        }
      }
              irrecv.resume();

/*************************************************/

  }
}


/* Function volumecal
 *  Desc    Increase or decrease the input value to be used for the volume
incrementation applied.
 *          Function considers the sides effects and
 *  Input   x:    integer current volume level value
 *          sens: boolean intdicating the increase (1) or the decrease (0) of
volume level value byt one unit
 *  Output  Return a value for the
 */
int volumecal(int x, bool sens){
  if (x==80){
    return 80-19*sens;
  }
    else if(x==61){
      return 60+abs(sens-1)*20;
    }
    else{
      return max(x-1+abs(sens-1)*2,0);
    }
}

/* Function volumeset(int )
```

```
 *  Desc    send the command to the sub
 *
 *  Input   nat: nature or parameter to be sent
 *          x: level of information to be sent
 */
 void Subset(int nat,int x){
  Wire.beginTransmission(68); // transmit to device #68
  Wire.write(nat);
  Wire.write(x);
  Wire.endTransmission();
  Serial.print(nat);
  Serial.print(", ");
  Serial.println(x);

}

/* Function Toogleonoff()
 *  Desc    Toogle On/Off of the sub with a smooth ramp up or down effect on
the volume
 *
 *  Input
 */
void Toogleonoff(){
  select = modeselect(2);
  onoff=abs(onoff-1);//Toogle the value onoff as boolean
    //Serial.println("ON/OFF");
    int x=(vol)*onoff+80*abs(onoff-1);// target Volume to reach in the process
    int temp=80*onoff+(vol)*abs(onoff-1); //original standpoint from where to
start in the process

    //turn the SUB ON in case the sub is currently off
    if(onoff==1){
      digitalWrite(stdy_pin,LOW);
      delay(50);
      digitalWrite(mute_pin,LOW);
      delay(50);
    }
    //Ramp-up or down the sound depending the toogle to get from or to the set
volume level
    while(x!=temp){
      Subset(0,temp);
      delay(50);
      temp=volumecal(temp,onoff);
    }
    //Turn the SUB OFF in case the sub is currently on
    if(onoff==0){
      Subset(0,80);
      digitalWrite(mute_pin,HIGH);
      delay(50);
      digitalWrite(stdy_pin,HIGH);
      delay(50);
    }
    delay(10);
    digitalWrite(red_pin,onoff);
}

/* Function Calc_FadCent
 *  Desc    Toogle On/Off of the sub with a smooth ramp up or down effect on
the volume
 *
 *  Input
 */

 int Calc_FadCent(int x, bool sens){
  return min(12,max(x-1+abs(sens-1)*2,0));
 }


 int Calc_Sub(int x, bool sens){
  if(x==89){
    return max(89*sens,80+sens*9);
  }
  if(x==80){
    return 81+sens*8;
  }
  else{
    return min(85,max(x-1+abs(sens-1)*2,80));
  }
 }



int modeselect(int x){
  //Reset all LED to off mode
  digitalWrite(red_pin,LOW);// volume
  digitalWrite(gre_pin,LOW);// Subwoofer
  digitalWrite(yel_pin,LOW);//Fader
  digitalWrite(blu_pin,LOW);//Center
  switch(x){
    case 0:
      digitalWrite(gre_pin,HIGH);
      return 7;
    break;
    case 7:
      digitalWrite(yel_pin,HIGH);
      return 4;
    break;
    case 4:
      digitalWrite(blu_pin,HIGH);
      return 3;
    break;
    case 3:
     // Optional Front Left,
      return 1;
    break;
    default:
      digitalWrite(red_pin,HIGH);
      return 0;
    break;
  }


}
```

Congratulation, you should now have a running program.

You can now adapt the remote to your needs with more buttons or like me add an infrared receptor to have it remote controlled (which that code already enables.

The previous code is not able to turn individually the speaker on mute. I haven't done the program with code 8 functions yet (mentioned earlier).

In the meantime this solution is a pretty robust approach to make a substitute to the original control pod in case you lost or broke it (and don’t want to pay $60, or € on a used spare)

Enjoy!!!