**AUX voor Volvo HU-XXX**

http://www.stichtingris.com/aux-voor-volvo-hu-xxx

We kennen het allemaal, je auto zo origineel mogelijk houden. Dit betekend ook de radio. Vooral bij de Volvo radio's geld dit want haal je ze eruit moet je weer een verloopje, Passend DIN-frame, andere radio enz enz. Natuurlijk is er ook de mogelijkheid om er een Grom of iMiV kastje tussen te hangen. Geweldige apparaten met erg veel mogelijkheden.

Maar ja, die prijs. Onder de 100 euro word al lastig en het enige dat ik wil is mijn iPhone via de radio liedjes laten spelen.

Na wat zoekwerk kwam ik uit bij een Amerikaanse site waar een aantal heren al erg druk waren geweest, hierdoor geïnspireerd ben ik zelf gaan zoeken en proberen. Waarom vraag je je af? Omdat ik het leuk vind en omdat je volgens deze heren onder de 10 euro klaar bent. Laat duidelijk zijn dat ik niemand voor het hoofd wil stoten of een boterham uit zijn mond wil trekken, ik heb het grootste respect voor de bedrijven die zaken als Grom en iMiV ontwikkelen, verkopen en inbouwen maar ja zelf een beetje klungelen vind ik ook heerlijk en als ik dan een paar euro kan besparen ( ben tenslotte nog een Nederlander) is dat helemaal geweldig.

Goed wat hebben we nodig:

Een 8-pin DIN stekker ------------> [LINK](https://www.allekabels.nl/din-kabel/130/1018991/din-stekker-8-polig.html)

Arduino Nano ( clone ) met USB-kabel -------------> [LINK](https://www.martoparts.nl/ATmega328P-Arduino-Nano-V3)

Metertje netwerk/ethernet kabel

Jack-plug -------------> [LINK](https://www.martoparts.nl/ATmega328P-Arduino-Nano-V3)

Wat dunne audiokabel

Natuurlijk moet je een beetje kunnen solderen, maar dat is ook het meest technische dat we nodig zullen hebben. Het printplaatje zal geprogrammeerd moeten worden maar aangezien er geweldige mensen zijn die dit heel goed kunnen en dit met de wereld willen delen is dit niets meer dan knip en plak werk.

Om het printplaatje te programmeren kun je op de site van Arduino het IDE programma downloaden om de code op het printplaatje te krijgen, deze vind je [HIER](https://www.arduino.cc/en/Main/Software) . Even je eigen besturingssysteem aanklikken en voila. Ik kwam er later achter dat je echt even op zoek moet gaan naar de juiste drivers om het bordje werkend te krijgen met het programma van Arduino, drivers verschillen echt per PC dus je zult er een aantal moeten proberen. Ik ben gewoon gaan snuffelen in de forums van de Arduino website, helaas ben ik wel de link kwijt voor de driver die voor mij werkte dus die kan ik helaas niet geven.

Goed we gaan beginnen!

Als eerste de layout van een JackPlug, heel simpel ROOD- Rechts, WIT- Links en ZWART- Massa.

Dan pakken we de EthernetKabel, hiervan gebruiken we 5 draden, Oranje, Wit, Blauw, Groen en Rood

Vervolgens gaan we deze aan de 8-Pin stekker solderen, beetje klein en friemelen maar komt allemaal goed.

Soldeer de kabels volgens onderstaand schema aan de stekker.

Vervolgens gaan we de andere kant van de kabel aan het bordje solderen, het schema van net geeft ook precies aan op welke plek welk draadje komt. Dus kun je lezen, kun je dit ook vast prima doen. De locaties staan namelijk ook gemarkeerd op het Bordje.

Hier zie je dan het eindresultaat van je knutselproject. Misschien niet oogstrelend mooi maar het werkt en niemand die het hierna nog zal zien. ;-)

Nu gaan we het bordje programmeren, vraag me niet wat wat doet trouwens. Ik heb de code op internet gevonden en deze werkt voor mij. Sluit het bordje aan op je computer en open het Arduino programma.

Kopieer en plak de code ( deze vind je hier onderaan de pagina ) en upload deze naar het bordje. Je krijgt vanzelf een melding als dit is gebeurt en geslaagd is. Haal nu het bordje weer los van je computer want het programmeren zit er alweer op.

Ik heb zelf de kabel nog wat verstevigd met tape. Dan is het nu tijd om naar je auto te gaan.

Ikzelf heb de onderstaande Radio, wat ik begrepen heb is deze tutorial toepasbaar op alle HU-XXX modellen.

Hieronder zie je de rood omcirkelde aansluiting voor de wisselaar die je nodig hebt. De aansluiting onder de gele pijl is geloof ik voor een versterker.

Sluit hier je geweldig gefabriceerde stekker op aan.

Zoals je ziet geeft het bordje aan dat hij "power" heeft en functioneert.

Zet je radio aan en ga door middel van aan de SOURCE knop te draaien naar de CD-CHANGER optie. Voorheen was deze optie niet zichtbaar maar zoals je ziet "Fopt" het printplaatje je radio nu door te zeggen "Hallo, ik ben een wisselaar, echt waar."

Dan is het alleen nog een kwestie van jackplug in je telefoon of mp3 speler en genieten van honderden mb's aan muziek.

Mocht je geïnspireerd zijn geraakt door deze How-To overweeg dan eens een donatie aan onze stichting.

De Code.

Let op alles in het geel is de code die gekopieerd en geplakt dient te worden. Have Fun!

De code is ook te vinden onder [DEZE LINK!](https://gist.githubusercontent.com/klalle/1ae1bfec5e2506918a3f89492180565e/raw/c28def1d0ea2b01066ab0fd0a40786b3dfb1bc34/Melbus_Mitsubishi%2520HU.ino)

#define SERDBG 1   
const uint8\_t MELBUS\_CLOCKBIT\_INT = 1; //interrupt numer (INT1) on DDR3  
const uint8\_t MELBUS\_CLOCKBIT = 3; //Pin D3 - CLK  
const uint8\_t MELBUS\_DATA = 4; //Pin D4 - Data  
const uint8\_t MELBUS\_BUSY = 5; //Pin D5 - Busy

volatile uint8\_t melbus\_ReceivedByte = 0;  
volatile uint8\_t melbus\_CharBytes = 0;  
volatile uint8\_t melbus\_OutByte = 0xFF;  
volatile uint8\_t melbus\_LastReadByte[12] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};  
volatile uint8\_t melbus\_SendBuffer[9] = {0x00,0x02,0x00,0x01,0x80,0x01,0xC7,0x0A,0x02};  
volatile uint8\_t melbus\_SendCnt=0;  
volatile uint8\_t melbus\_DiscBuffer[6] = {0x00,0xFC,0xFF,0x4A,0xFC,0xFF};  
volatile uint8\_t melbus\_DiscCnt=0;  
volatile uint8\_t melbus\_Bitposition = 0x80;

volatile bool InitialSequence\_ext = false;  
volatile bool ByteIsRead = false;  
volatile bool sending\_byte = false;  
volatile bool melbus\_MasterRequested = false;  
volatile bool melbus\_MasterRequestAccepted = false;

volatile bool testbool = false;  
volatile bool AllowInterruptRead = true;  
volatile int incomingByte = 0; // for incoming serial data

//Startup seequence  
void setup() {  
//Data is deafult input high  
pinMode(MELBUS\_DATA, INPUT\_PULLUP);  
  
//Activate interrupt on clock pin (INT1, D3)  
attachInterrupt(MELBUS\_CLOCKBIT\_INT, MELBUS\_CLOCK\_INTERRUPT, FALLING);   
//Set Clockpin-interrupt to input  
pinMode(MELBUS\_CLOCKBIT, INPUT\_PULLUP);  
#ifdef SERDBG  
//Initiate serial communication to debug via serial-usb (arduino)  
Serial.begin(230400);  
Serial.println("Initiating contact with Melbus:");  
#endif  
//Call function that tells HU that we want to register a new device  
melbus\_Init\_CDCHRG();  
}

//Main loop  
void loop() {  
//Waiting for the clock interrupt to trigger 8 times to read one byte before evaluating the data  
#ifdef SERDBG  
if (ByteIsRead) {  
//Reset bool to enable reading of next byte  
ByteIsRead=false;

if (incomingByte == ' ') {  
if(melbus\_LastReadByte[11] == 0x0 && (melbus\_LastReadByte[10] == 0x4A || melbus\_LastReadByte[10] == 0x4C || melbus\_LastReadByte[10] == 0x4E) && melbus\_LastReadByte[9] == 0xEC && melbus\_LastReadByte[8] == 0x57 && melbus\_LastReadByte[7] == 0x57 && melbus\_LastReadByte[6] == 0x49 && melbus\_LastReadByte[5] == 0x52 && melbus\_LastReadByte[4] == 0xAF && melbus\_LastReadByte[3] == 0xE0 && melbus\_LastReadByte[2] == 0x0)  
{  
melbus\_CharBytes=8; //print RDS station name  
}

if(melbus\_LastReadByte[1] == 0x0 && melbus\_LastReadByte[0] == 0x4A)  
{  
Serial.println("\n LCD is master: (no CD init)");  
}  
else if(melbus\_LastReadByte[1] == 0x0 && melbus\_LastReadByte[0] == 0x4C)  
{  
Serial.println("\n LCD is master: (???)");  
}  
else if(melbus\_LastReadByte[1] == 0x0 && melbus\_LastReadByte[0] == 0x4E)  
{  
Serial.println("\n LCD is master: (with CD init)");  
}  
else if(melbus\_LastReadByte[1] == 0x80 && melbus\_LastReadByte[0] == 0x4E)  
{  
Serial.println("\n ???");  
}  
else if(melbus\_LastReadByte[1] == 0xE8 && melbus\_LastReadByte[0] == 0x4E)  
{  
Serial.println("\n ???");  
}  
else if(melbus\_LastReadByte[1] == 0xF9 && melbus\_LastReadByte[0] == 0x49)  
{  
Serial.println("\n HU is master: ");  
}  
else if(melbus\_LastReadByte[1] == 0x80 && melbus\_LastReadByte[0] == 0x49)  
{  
Serial.println("\n HU is master: ");  
}  
else if(melbus\_LastReadByte[1] == 0xE8 && melbus\_LastReadByte[0] == 0x49)  
{  
Serial.println("\n HU is master: ");  
}  
else if(melbus\_LastReadByte[1] == 0xE9 && melbus\_LastReadByte[0] == 0x4B)  
{  
Serial.println("\n HU is master: -> CDC");  
}  
else if(melbus\_LastReadByte[1] == 0x81 && melbus\_LastReadByte[0] == 0x4B)  
{  
Serial.println("\n HU is master: -> CDP");  
}  
else if(melbus\_LastReadByte[1] == 0xF9 && melbus\_LastReadByte[0] == 0x4E)  
{  
Serial.println("\n HU is master: ");  
}  
else if(melbus\_LastReadByte[1] == 0x50 && melbus\_LastReadByte[0] == 0x4E)  
{  
Serial.println("\n HU is master: ");  
}  
else if(melbus\_LastReadByte[1] == 0x50 && melbus\_LastReadByte[0] == 0x4C)  
{  
Serial.println("\n HU is master: ");  
}  
else if(melbus\_LastReadByte[1] == 0x50 && melbus\_LastReadByte[0] == 0x4A)  
{  
Serial.println("\n HU is master: ");  
}  
else if(melbus\_LastReadByte[1] == 0xF8 && melbus\_LastReadByte[0] == 0x4C)  
{  
Serial.println("\n HU is master: ");  
}

if (melbus\_CharBytes) {  
Serial.write(melbus\_LastReadByte[1]);  
melbus\_CharBytes--;  
}else   
{  
Serial.print(melbus\_LastReadByte[1],HEX);  
Serial.write(' ');  
}  
}

}  
#endif

//If BUSYPIN is HIGH => HU is in between transmissions  
if (digitalRead(MELBUS\_BUSY)==HIGH)   
{  
//Make sure we are in sync when reading the bits by resetting the clock reader

#ifdef SERDBG  
if (melbus\_Bitposition != 0x80) {  
Serial.println(melbus\_Bitposition,HEX);  
Serial.println("\n not in sync! ");  
}  
#endif  
if (incomingByte != 'k') {  
melbus\_Bitposition = 0x80;  
melbus\_OutByte = 0xFF;  
melbus\_SendCnt=0;  
melbus\_DiscCnt=0;  
DDRD &= ~(1<<MELBUS\_DATA);  
PORTD |= (1<<MELBUS\_DATA);  
}

}   
#ifdef SERDBG  
if (Serial.available() > 0) {  
// read the incoming byte:  
incomingByte = Serial.read();

}  
if (incomingByte == 'i') {  
melbus\_Init\_CDCHRG();  
Serial.println("\n forced init: ");  
incomingByte=0;

}  
#endif  
if ((melbus\_Bitposition == 0x80) && (PIND & (1<<MELBUS\_CLOCKBIT)))  
{  
delayMicroseconds(7);  
DDRD &= ~(1<<MELBUS\_DATA);  
PORTD |= (1<<MELBUS\_DATA);  
}

}

//Notify HU that we want to trigger the first initiate procedure to add a new device (CD-CHGR) by pulling BUSY line low for 1s  
void melbus\_Init\_CDCHRG() {  
//Disabel interrupt on INT1 quicker then: detachInterrupt(MELBUS\_CLOCKBIT\_INT);  
EIMSK &= ~(1<<INT1);   
  
// Wait untill Busy-line goes high (not busy) before we pull BUSY low to request init  
while(digitalRead(MELBUS\_BUSY)==LOW){}   
delayMicroseconds(10);  
  
pinMode(MELBUS\_BUSY, OUTPUT);  
digitalWrite(MELBUS\_BUSY, LOW);  
delay(1200);  
digitalWrite(MELBUS\_BUSY, HIGH);  
pinMode(MELBUS\_BUSY, INPUT\_PULLUP);

//Enable interrupt on INT1, quicker then: attachInterrupt(MELBUS\_CLOCKBIT\_INT, MELBUS\_CLOCK\_INTERRUPT, RISING);  
EIMSK |= (1<<INT1);   
}

//Global external interrupt that triggers when clock pin goes high after it has been low for a short time => time to read datapin  
void MELBUS\_CLOCK\_INTERRUPT() {

//Read status of Datapin and set status of current bit in recv\_byte  
  
if(melbus\_OutByte & melbus\_Bitposition){  
DDRD &= (~(1<<MELBUS\_DATA));  
PORTD |= (1<<MELBUS\_DATA);  
}  
//if bit [i] is "0" - make databpin low  
else{  
PORTD &= (~(1<<MELBUS\_DATA));  
DDRD |= (1<<MELBUS\_DATA);  
}  
  
  
if (PIND & (1<<MELBUS\_DATA)){  
melbus\_ReceivedByte |= melbus\_Bitposition; //set bit nr [melbus\_Bitposition] to "1"  
}  
else {  
melbus\_ReceivedByte &=~melbus\_Bitposition; //set bit nr [melbus\_Bitposition] to "0"  
}  
  
  
//if all the bits in the byte are read:   
if (melbus\_Bitposition==0x01) {  
  
//Move every lastreadbyte one step down the array to keep track of former bytes   
for(int i=11;i>0;i--){  
melbus\_LastReadByte[i] = melbus\_LastReadByte[i-1];  
}

if (melbus\_OutByte != 0xFF) {  
melbus\_LastReadByte[0] = melbus\_OutByte;  
melbus\_OutByte = 0xFF;  
} else {  
  
//Insert the newly read byte into first position of array  
melbus\_LastReadByte[0] = melbus\_ReceivedByte;  
}  
//set bool to true to evaluate the bytes in main loop  
ByteIsRead = true;

//Reset bitcount to first bit in byte  
melbus\_Bitposition=0x80;  
if(melbus\_LastReadByte[2] == 0x07 && (melbus\_LastReadByte[1] == 0x1A || melbus\_LastReadByte[1] == 0x4A) && melbus\_LastReadByte[0] == 0xEE)  
{  
InitialSequence\_ext = true;  
}  
else if(melbus\_LastReadByte[2] == 0x0 && (melbus\_LastReadByte[1] == 0x1C || melbus\_LastReadByte[1] == 0x4C) && melbus\_LastReadByte[0] == 0xED)  
{  
InitialSequence\_ext = true;  
}  
else if((melbus\_LastReadByte[0] == 0xE8 || melbus\_LastReadByte[0] == 0xE9) && InitialSequence\_ext == true){  
InitialSequence\_ext = false;

//Returning the expected byte to the HU, to confirm that the CD-CHGR is present (0xEE)! see "ID Response"-table here http://volvo.wot.lv/wiki/doku.php?id=melbus  
melbus\_OutByte = 0xEE;  
}  
else if((melbus\_LastReadByte[2] == 0xE8 || melbus\_LastReadByte[2] == 0xE9) && (melbus\_LastReadByte[1] == 0x1E || melbus\_LastReadByte[1] == 0x4E) && melbus\_LastReadByte[0] == 0xEF)  
{  
// CartInfo  
melbus\_DiscCnt=6;  
}  
else if((melbus\_LastReadByte[2] == 0xE8 || melbus\_LastReadByte[2] == 0xE9) && (melbus\_LastReadByte[1] == 0x19 || melbus\_LastReadByte[1] == 0x49) && melbus\_LastReadByte[0] == 0x22)  
{  
// Powerdown  
melbus\_OutByte = 0x00; // respond to powerdown;  
melbus\_SendBuffer[1]=0x02; // STOP  
melbus\_SendBuffer[8]=0x02; // STOP  
}  
else if((melbus\_LastReadByte[2] == 0xE8 || melbus\_LastReadByte[2] == 0xE9) && (melbus\_LastReadByte[1] == 0x19 || melbus\_LastReadByte[1] == 0x49) && melbus\_LastReadByte[0] == 0x52)  
{  
// RND  
}  
else if((melbus\_LastReadByte[2] == 0xE8 || melbus\_LastReadByte[2] == 0xE9) && (melbus\_LastReadByte[1] == 0x19 || melbus\_LastReadByte[1] == 0x49) && melbus\_LastReadByte[0] == 0x29)  
{  
// FF  
}  
else if((melbus\_LastReadByte[2] == 0xE8 || melbus\_LastReadByte[2] == 0xE9) && (melbus\_LastReadByte[1] == 0x19 || melbus\_LastReadByte[1] == 0x49) && melbus\_LastReadByte[0] == 0x2F)  
{  
// FR  
melbus\_OutByte = 0x00; // respond to start;  
melbus\_SendBuffer[1]=0x08; // START  
melbus\_SendBuffer[8]=0x08; // START  
}  
else if((melbus\_LastReadByte[3] == 0xE8 || melbus\_LastReadByte[3] == 0xE9) && (melbus\_LastReadByte[2] == 0x1A || melbus\_LastReadByte[2] == 0x4A) && melbus\_LastReadByte[1] == 0x50 && melbus\_LastReadByte[0] == 0x01)  
{  
// D-  
melbus\_SendBuffer[3]--;  
melbus\_SendBuffer[5]=0x01;  
}  
else if((melbus\_LastReadByte[3] == 0xE8 || melbus\_LastReadByte[3] == 0xE9) && (melbus\_LastReadByte[2] == 0x1A || melbus\_LastReadByte[2] == 0x4A) && melbus\_LastReadByte[1] == 0x50 && melbus\_LastReadByte[0] == 0x41)  
{  
// D+  
melbus\_SendBuffer[3]++;  
melbus\_SendBuffer[5]=0x01;  
}  
else if((melbus\_LastReadByte[4] == 0xE8 || melbus\_LastReadByte[4] == 0xE9) && (melbus\_LastReadByte[3] == 0x1B || melbus\_LastReadByte[3] == 0x4B) && melbus\_LastReadByte[2] == 0x2D && melbus\_LastReadByte[1] == 0x00 && melbus\_LastReadByte[0] == 0x01)  
{  
// T-  
melbus\_SendBuffer[5]--;  
}  
else if((melbus\_LastReadByte[4] == 0xE8 || melbus\_LastReadByte[4] == 0xE9) && (melbus\_LastReadByte[3] == 0x1B || melbus\_LastReadByte[3] == 0x4B) && melbus\_LastReadByte[2] == 0x2D && melbus\_LastReadByte[1] == 0x40 && melbus\_LastReadByte[0] == 0x01)  
{  
// T+  
melbus\_SendBuffer[5]++;  
}  
else if((melbus\_LastReadByte[4] == 0xE8 || melbus\_LastReadByte[4] == 0xE9) && (melbus\_LastReadByte[3] == 0x1B || melbus\_LastReadByte[3] == 0x4B) && melbus\_LastReadByte[2] == 0xE0 && melbus\_LastReadByte[1] == 0x01 && melbus\_LastReadByte[0] == 0x08 ){  
// Playinfo  
melbus\_SendCnt=9;  
}   
if (melbus\_SendCnt) {  
melbus\_OutByte=melbus\_SendBuffer[9-melbus\_SendCnt];  
melbus\_SendCnt--;  
} else if (melbus\_DiscCnt) {  
melbus\_OutByte=melbus\_DiscBuffer[6-melbus\_DiscCnt];  
melbus\_DiscCnt--;  
}  
} else {  
//set bitnumber to address of next bit in byte  
melbus\_Bitposition>>=1;  
}  
EIFR |= (1 << INTF1);  
}

Top of Form

https://www.paypalobjects.com/nl_NL/i/scr/pixel.gif

Bottom of Form

[Gebruiksvoorwaarden](http://stichtingris.jouwweb.nl/gebruikersvoorwaarden)

[Privacy](http://stichtingris.jouwweb.nl/privacy-document)

pagina-content

© 2016 - 2017 Stichtingris.com