4 Broncode

4.1 Esp8266 (init.lua)

4.1.1 Titels opslaan

```
-- Deze functie zet de juiste liedjes in de variabelen
function setSong(nr, titel)
        if(titel ~= nil) then
                if(nr == 1) then
                        songTitle1 = titel
                elseif(nr == 2) then
                        songTitle2 = titel
                elseif(nr == 3) then
                        songTitle3 = titel
                elseif(nr == 4) then
                        songTitle4 = titel
                else
                        uart.write(0,"ERROR")
                end
        end
end
```

Functie1: Liedjes in juiste var zetten.

De eerste functie is gemaakt om via uart de titels van de liedjes door te sturen. De microcontroller kan nu via uart bv. setSong(1,”Missing - edx”) sturen. Lua zal dit dan begrijpen en automatisch deze functie starten. Deze functie zal dan het juiste liedje in de juiste variabele zetten, of een error terugsturen als de index van het liedje niet bestaat.

4.1.2 Placeholders vervangen

```
-- Een functie die in het html bestand zoekt naar de plaats om de titels in te vullen
function checkIfSongTitle( line )
    if (line:find("##SONG1##") ~= nil) then
        line = songTitle1 .. " (" .. song1 .. ")"
    elseif (line:find("##SONG2##") ~= nil) then
        line = songTitle2 .. " (" .. song2 .. ")"
    elseif (line:find("##SONG3##") ~= nil) then
        line = songTitle3 .. " (" .. song3 .. ")"
    elseif (line:find("##SONG4##") ~= nil) then
        line = songTitle4 .. " (" .. song4 .. ")"
    end
    return line;
```

Functie2: Placeholder vervangen.

Deze functie vervangt bij het voorkomen van een placeholder, de lijn door de titel van een liedje.

4.1.3 Opzetten en verwerken http webserver

```
-- We starten de server op op poort 80
srv=net.createServer(net.TCP)
srv:listen(80,function(conn)
    conn:on("receive", function(client,request)
        local _, _, method, path, vars = string.find(request, "([A-Z]+) (.+)?(.+) HTTP");
        if(method == nil)then
            _, _, method, path = string.find(request, "([A-Z]+) (.+) HTTP");
        end
        local _GET = {}
        if (vars ~= nil)then
            for k, v in string.gmatch(vars, "(%w+)=(%w+)&*") do
                _GET[k] = v
            end
        end

        count(_GET.next)        -- We tellen het aantal stemmen

        file.open('webpaginaOnline.html','r')
        local line = file.readline()
        while (line) do
            line = checkIfSongTitle(line)      -- We kijken of er een titel moet komen
            client:send(line)                  --  en vervanen als het nodig is
            line = file.readline()
        end
        file.close()
        client:close();

        collectgarbage();
    end)
end)
```

Functie3: Http webserver.

De webserver wordt gestart op poort 80, en er wordt daarna gewacht op een http get request.
Als er een get request toekomt, wordt dit gestoken in de variabele GET. Als er op 1 van de knoppen op de website gedrukt wordt, wordt er een get request gestuurd met een waarde in “next”. Deze waarde wordt meegegeven als parameter aan de functie count. Deze functie houd een telling bij van de liedjes waarop gestemd is.
Als er geen request met next is, wordt gewoon de pagina geladen. Dit laden gebeurd lijn per lijn uit een html file die opgeslagen is op de flash van de esp module.
Bij het lijn per lijn lezen van de file, wordt elke lijn door de functie checkIfSongTitle gehaald, deze functie kijkt of er in de html file een placeholder staat waar de songtitels moeten komen.

4.1.4 Opvragen meest gestemde lied

```
-- Returnt het meest gestemde liedjesId
function getBest()
	local lijst = {}
	lijst[1] = song1
	lijst[2] = song2
	lijst[3] = song3
	lijst[4] = song4
	table.sort(lijst)

	if(lijst[4] == song1) then
		uart.write(0,"1\n")
	elseif(lijst[4] == song2) then
		uart.write(0,"2\n")
	elseif(lijst[4] == song3) then
		uart.write(0,"3\n")
	elseif(lijst[4] == song4) then
		uart.write(0,"4\n")
	end

	resetSongs()
end
```

Functie4: Beste liedje opvragen.

Deze functie is gemaakt om via uart het meest gestemde liedje op te vragen. Als de microcontroller getBest() stuurt via uart, dan zal de esp het meest gestemde liedje terugsturen. Dit wordt gedaan door het sorteren van een lijst, en het laatste item in de lijst is dan het meest gestemde liedje. De functie resetSongs zet de tellers van de liedjes op 0.

4.2 Stm32
4.2.1 Sample timer interrupt

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
  static uint16_t teller = 0;

  // We willen enkel de compare van timer3
  if(htim->Instance ==  TIM3)
  {
        if(wavBufferSelect == 0)  // Kijken uit welke buffer we moeten lezen
        {
              __HAL_TIM_SET_COMPARE(&htim4,TIM_CHANNEL_3,
                    (wavBuffer0[teller] + 100));
        }
        else
        {
              __HAL_TIM_SET_COMPARE(&htim4,TIM_CHANNEL_3,
                    (wavBuffer1[teller] + 100));
        }
        teller++;

        if(teller >= 512)
        {
              bufferEnd = 1;
              wavBufferSelect = ! wavBufferSelect;
              teller = 0;
        }
  }
}
```

Functie1: Muziek op pwm zetten

Dit is de functie die opgeroepen wordt als timer3 een compare match heeft. Deze functie wordt dus aan 44100Hz opgeroepen en past aan deze frequentie de pwm output aan. De waarde die de pwm zal outputen wordt gehaald uit de 2 bufferArrays die afgewisseld worden als ze helemaal uitgelezen zijn.

4.2.2 Begin programma

```
uint8_t aantalNummers = 0;

// De variabele waar de file inzit
FIL MyFile;

HAL_UART_Transmit(&huart2,"Wireless jukebox V1.0",21,10);

// We halen alle titels op vanaf de sd-kaart en slagen deze op in een array
char * titels[40];
aantalNummers = getTitels(&MyFile,titels,20);

// We sturen de eerste liedjes naar de esp
uint8_t selectedSongs[4] = {0};
randomNummers(selectedSongs,aantalNummers);

// We gaan nu de eerste wav buffer vullen, hiervoor pakken we steeds nummer 0
probeer(f_open(&MyFile,"0.wav",FA_READ),"openen file 0.wav");
getData(&MyFile,wavBuffer0,512);
getData(&MyFile,wavBuffer1,512);

// Timer3 en pwm output starten
startPwm();

// We zetten de eerste titels op de esp
HAL_Delay(5000);          // We moeten wachten zodat de esp zeker opgestart is
sendSongsToEsp(titels,selectedSongs);
```

Functie2: Alles initialiseren

Na het sturen van een welkom message over de uart gaan we op de sd-kaart zoeken naar alle titels van de liedjes die erop staan. Deze worden opgeslagen in de array titels. Hieruit krijgen we ook het aantal liedjes dat op de sd-kaart staat.
Vervolgens gaan we een array van 4 elementen vullen met 4 random id's van liedjes. Als eerste liedje openen we altijd lied0 op de kaart en we starten met het vullen van de 2 buffers. Eens de 2 buffers beide opgevuld zijn kunnen we de pwm starten en kan de muziek beginnen spelen.
Hierna wordten de titels van de huidige random nummers doorgestuurd naar de esp zodat er op gestamd kan worden.

4.3.3 getTitels (wav.c)

```
uint16_t getTitels(FIL * file, char ** titels, uint16_t maxTitels)
{

	uint16_t teller = 0;
	char buffer[100];

	// We proberen de file met de titels te openen op de sd-kaart
	probeer(f_open(file, "titels.txt",FA_READ),"open titels.txt");

	// Nu lezen we alle lijnen met titles in de file
	while(f_eof(file) == 0)
	{
		f_gets(buffer, sizeof(buffer),file);	// We lezen 1 lijn
		// We zoeken wat plaats voor de nieuwe titel
		titels[teller] = (char *) malloc(strlen(buffer));
		// We zetten de titel mee in de array
		strcpy(titels[teller],buffer);

		// Als we het max aantal titels gelezen hebben, moeten we zeker
			stoppen
		if(teller > maxTitels) break;
		teller ++;
	}
	f_close(file);
	return teller;
```

Functie3: Alle titels van de sd-kaart halen

In deze functie gaan we alle titels vanaf de sd-kaart ophalen uit het bestand met de titels en slaagt deze op in de titels array via een malloc. Eerst wordt de data ingelezen in een buffer en vervolgens wordt er genoeg plaats in het geheugen gezocht om deze titel op te slaan. Als alle titels opgeslagen zijn wordt er nog een aantal titels teruggestuurd.

4.3.4 startPwm (wav.c)

```
void startPwm(void)
{
   HAL_TIM_Base_Start_IT(&htim3);
   HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_3);
}
```

Functie4: Pwm starten

Deze functie start de pwm sample timer en activeert de pwm output.

4.3.5 randomNummers (esp.c)

```
void randomNummers(uint8_t * selectedSongs, uint8_t aantalSongs)
{
	uint8_t temp = 0;
	uint8_t teller = 0;

	for (teller = 0; teller < 4; teller++)
	{
		selectedSongs[teller] = 5;
	}

	teller = 0;

	// we nemen 4 random nummers en slagen deze op, ze mogen niet dezelfde
		zijn
	while(teller <= 4)
	{
		temp = rand() % aantalSongs;
		if(temp != selectedSongs[0] && temp != selectedSongs[1] &&
			temp != selectedSongs[2] && temp != selectedSongs[3] &&
			temp <= aantalSongs-1)
		{
			selectedSongs[teller] = temp;
			teller++;
		}

	}
```

Functie5: random nummers genereren.

In deze functie gaan we 4 verschillende random nummers genereren hiervoor genereren we elke keer een random nummer en kijken we of dit nog niet gegenereerd was deze keer.

4.3.6 sendSongsToEsp (esp.c)

```
void sendSongsToEsp(char * titels[], uint8_t selectedSongs[])
{
	uint8_t teller = 0;
	// We sturen 4 liedjes naar de esp
	for(teller = 0; teller<4; teller++)
	{
		setSong(1,titels[selectedSongs[0]]);
		HAL_Delay(100);
		setSong(2,titels[selectedSongs[1]]);
		HAL_Delay(100);
		setSong(3,titels[selectedSongs[2]]);
		HAL_Delay(100);
		setSong(4,titels[selectedSongs[3]]);
		HAL_Delay(100);
	}
```

Functie6: titels naar esp sturen

Deze functie stuurt de 4 titels van de random gegenereerde liedjes door naar de esp zodat erop gestemd kan worden. Er zit een for loop in om dit een paar keer te doen om er zeker van de zijn dat de liedjes er goed opkomen.

```
void setSong(uint8_t nummer, char titel[])
{
	char buffer[100];
	sprintf(buffer,"setSong(%d,'%s')\n",nummer,titel);
	HAL_UART_Transmit(&huart1,(uint8_t *)buffer,strlen(buffer),10);
```

Functie7: titel doorsturen via uart.

Deze functie stuurt 1 titel door via uart in het formaat dat de esp verwacht.

4.3.7 Main while

```
if((songEnd == 1) && (bufferEnd == 1))  // als liedje gedaan is
{
        // We stoppen de pwm
        stopPwm();

        // We sluiten de file
        f_close(&MyFile);

        volgendLiedId = getBest() - 1;

        // Id van het volgend liedje ophalen
        volgendLiedId = selectedSongs[volgendLiedId];

        // Zo staat het liedje op de schijf
        sprintf(openSongString,"%d.wav",volgendLiedId);

        // We openen de nieuwe file
        probeer(f_open(&MyFile,openSongString,FA_READ),"open nieuw liedje");

        // Liedjes voor de volgende stemming kiezen
        randomNummers(selectedSongs,aantalNummers);
        sendSongsToEsp(titels,selectedSongs);

        // Alles terug in de beginpositie
        getData(&MyFile,wavBuffer0,512);
        wavBufferSelect = 0;
        bufferEnd = 0;
        songEnd = 0;
        startPwm();
```

Functie8: als liedje gedaan is.

Als het liedje gedaan is dan stoppen we de pwm, en vragen aan de esp wat het volgende liedje is. Het volgend liedjes id wordt dan omgezet in de titel van het liedje op de sd-kaart en deze wordt dan geladen. Dit liedje wordt dan open gedaan en alles wordt herhaald, we genereren nieuwe random nummers en sturen ze terug naar de esp. Hierna worden alle variabelen en buffers terug klaargezet om opnieuw muziek af te spelen.

```
if(bufferEnd == 1)
{
        bufferEnd = 0;
        // Kijken in welke buffer de interupt aan het lezen is
        if(wavBufferSelect == 0)
        {
                // Interupt leest in 0, wij schrijven in 1
                if(getData(&MyFile,wavBuffer1,512) == 1 ) songEnd = 1;
        }
        else
        {
                if(getData(&MyFile,wavBuffer0,512) == 1 ) songEnd = 1;
        }

   }
```

functie9: wavBuffers vullen.

Dit stuk code bevind zich in de main en zal true zijn als de interrupt 512 bytes naar de pwm gestuurd heeft en dus aan het einde van zijn buffer is.
Als het einde bereikt is schakelt de interrupt automatisch over naar de 2de buffer, en kunnen wij dus de lege buffer terug vullen met data van de sd-kaart.

4.3.8 getData (wav.c)

```
uint8_t getData(FIL * fp, char data[],int aantal)
{
    int teller = 0;
    static uint8_t dataGevonden = 0;

    char byte;
    uint8_t bytesRead;
    uint16_t i;

    // eerst moeten we de header over springen tot aan de data chunck
    while(f_eof(fp) == 0)          // we lezen tot aan het einde van de file
    {
        f_read(fp,&byte,1 * sizeof(char),(UINT*)&bytesRead);

        // Kijken of we de data header al gevonden hadden
        if(dataGevonden == 0)
        {
            // we hebben een d gevonden, nu zoeken we naar "ata" van data
            if(GET3CHARS = "ata")
            {
                dataGevonden = 1;
            }
        }
        else        // de data header is gevonden, vanaf nu komt de data
        {
            // We slagen het aantal gevraagde databytes op in de array
            data[teller] = byte;
            teller ++;

            if(teller == aantal)    //Als we genoeg data gelezen hebben stoppen we
            {
                return 0;
            }
        }
    }
    return 1;
```

Functie10: Data uit wav file halen.

Deze functie haalt een opgegeven aantal bytes uit een opgegeven wav file. Eerst wordt er gekeken of we al voorbij de wav-header zijn, als dit het geval is dan kunnen we data uitlezen en opslaan in de meegegeven buffer. Als dit niet het geval is dan moeten we eerst zoeken naar de "DATA" chuck, en dan kunnen we pas de data uitlezen.