Aan het begin van dit jaar (2023) werd mij gevraagd om een geluidje te maken bij een station wanneer een trein eraan komt. Dit kan uitgebreid worden met knipperlichten, teksten, bewegingen en andere zaken die, schijnbaar, tegelijkertijd gaan. Ik ben begonnen met een demo boordje waarop verschillende zaken met een druk op de knop het openen en sluiten van twee dubbel treinloods deuren simuleerde. Het volgende scenario werd met een druk op de knop uitgevoerd:

1. De deur die open/gesloten werd met een (Duitse) stem wat er met de rechter deur gebeurd.
2. Een knipperlicht gaat knipperen
3. De rechter deur gaat piepend open/sluiten (servo gaat bewegen)
4. Een stem klinkt dat de linker deur open/gesloten wordt
5. De tweede deur gaat piepend bewegen
6. Wanneer de eerste deur gesloten/geopend is dan zal een stem dat vertellen
7. Wanneer de tweede deur gesloten/geopend is dan zal een stem dat vertellen
8. Het knipperlicht gaat uit.

Parallel hieraan worden verschillende teksten in een LCD display getoond.

Verschillende mensen hebben mij gevraagd hoe ik dit voor elkaar kreeg en sommigen vroegen daarbij of ik het hen wilde uitleggen. Dat wil ik best.

Dit stukje is dan wel erg technisch en ik begrijp dat verschillende mensen dit niet zullen begrijpen. Begrippen zoals Arduino, loop(), setup() maar ook preemptive statemachine zullen voor velen onduidelijk zijn en blijven. Toch denk ik dat dit artikeltje in dit kwartaalblad geplaatst kan worden; niet iedereen zal hier wat aan hebben maar sommigen:

* Zullen aan de hand van dit artikel hun Arduino project kunnen verbeteren
* Anderen kunnen met mijn hulp betere sketches in de Arduino kunnen schrijven
* Voor weer anderen zal dit een opstapje zijn om de (geweldige wereld van de) Arduino te leren.

Ook wanneer men geen kennis heeft van de Arduino, kan het handig zijn om ideeën te opperen, zowel voor de vereniging als ook voor je eigen privé baan en mensen te vragen om het programmeren te doen.

Ik zal ook delen van bestaande code bespreken. Deze code kan men van het internet afhalen. Dit staat namelijk in mijn persoonlijke GitHub repository. De code kan dus gedownload worden via de link:

<https://github.com/knijff1961/MVA>

Hier is een knopje genaamd Code waar een zip file gedownload kan worden. Ik zal proberen om deze repository (of bibliotheek) bij tee houden.

**De peremptive statemachine**

Mensen die met de Arduino vertrouwd zijn weten dat er twee routines in de sketch voorkomen:

1. De setup() die de Arduino verteld wat er allemaal aan de Arduino “hangt” in hoe al deze hardware geïnitialiseerd dient te worden.
2. De loop() welke continue door de Arduino wordt aangeroepen. Deze loop blijft aangeroepen worden tot de stroom van de Arduino afgaat.

Laten we een eenvoudig programmaatje nemen: een knipperlicht; elke seconde aan-uit-aan-uit- etc.

Stel een ledje is verbonden met pin 9 van de arduino. Dit is een output dus in de setup zal de code moeten staan zoals:

void setup() {

pinMode(9, OUTPUT);

}

En de loop kan dan zijn:

void loop() {

digitalWrite(9, LOW);

delay(1000);

digitalWrite(9, LOW);

delay(1000);

}

Eerst wordt de led aangezet, dan wordt er een seconde (1000 ms) gewacht. Hierna zal de led uitgezet worden en wordt er weer een seconde gewacht. En dit blijft zich herhalen.

Heel eenvoudig dus en inderdaad de led zal om de seconde aan of uitgaan.

Stel nu, we hebben 2 knipperlichten (Led pin 9 & led pin 10). De ene moet om de seconde knipperen, de ander om de 700 ms. Nu wordt het wat lastiger; we kunnen niet zomaar vertragingen erin plakken want hoelang moeten deze delays zijn. Toch is er een oplossing: preemptive state machines.

Om het een en ander te verduidelijken hebben gaan we een café bekijken. In de keuken hangt een klok en in het cafe zijn twee tafels met mensen. De ene tafel wilt om de 10 minuten koffie en tien minuten later thee. Daarna, na tien minuten koffie en na 10 minuten weer thee etc.

Een mogelijke pseudo code is:

* Breng koffie
* Wacht 10 minuten
* Breng thee
* Wacht 10 minuten

Maar je kunt ook zeggen:

* Breng koffie en bepaal de tijd wanneer er thee gebracht moet worden. (10 minuten later dus)
* Doe allerlei andere dingen maar controleer wel regelmatig de tijd.
* Wanneer de tijd om is breng thee en bepaal opnieuw de tijd dat er koffie gehaald dient te worden.
* Doe allerlei andere dingen maar controleer wel regelmatig de tijd.

En doe dit voor een onbepaald tijd. Twee dingen moet e dus bijhouden: De tijd dat je iets moet doen (dit noemen we de timeout()) en wat je zal moeten doen (De state). Dit laatste wordt bepaald wat je gedaan hebt. Dus zolang er geen timeout is: ga direct de keuken in, bepaal de huidige tijd en ga weer het café (loop()) in.

Bovenstaande code kan dus herschreven worden door middel van:

Is er op deze tafel een timeout() {

Bepaal destatus:

Is de status “koffie gehad” {

Geef thee

Zet de timeout 10 minuten later

} anders is de status “thee gehad”

Geef koffie

Zet de timeout 10 minuten later

}

}

**NOOT:** Dit is dus een statemachine!

2 belangrijke zaken zijn nu van belang:

Deze code moet in de loop() geplaatst worden (net zoals de tafel in de café staat)

Zolang er niets gedaan hoeft te worden (er is dus geen timeout()) zal de loop() verlaten moeten worden om de nieuwe tijd te bepalen. Deze tijd kan gebruikt worden om een tweede statemachine te maken en deze OOK uitvoeren. Wanneer een tweede tafel (statemachine) bezet is en deze persoon ook om en om koffie wilt hebben maar dan om de 7 minuten, dan kan de code hiervoor gewoon in de loop gezet worden. Mocht de (zeldzame) situatie zich voordoen dat beide tijden precies op hetzelfde moment afloopt, dan zal het toch lijken alsof de twee tafels binnen de 10, resp. 7 minuten afgehandeld wordt.

Het leuke is dat er nog een derde, vierde etc. tafel erbij kunnen komen. De tafels kunnen gewoon allemaal afgehandeld worden zolang het café open is.

Ik heb een class gemaakt die de tijd, state en timeout kan bijhouden. De class naam is:   
 CPreEmptiveTimer.

De class beschrijft 2 functies (genaamd methods) en één status waarde (property).

|  |  |  |
| --- | --- | --- |
| Naam | M/P | Beschrijving |
| void setWaitConst(unsigned long lMs) | Method | Hiermee wordt de wachttijd gezet |
| bool timeout() | Method | Geeft aan wanneer een timeout is opgetreden. |
| int iState | Property | De status van de state machine |

Wanneer een knipperlicht gemaakt wordt met dit object dan kan de volgende code gebruikt wortden:

De code met 2 knipperlichten waarvan de eerste om de 1000 ms knippert en de tweede hetzelfde doet om de 700 ms ziet er dan als volgt uit.

#include <CPreEmptiveTimer.h>

CPreEmptiveTimer cPreEmtiveTimer1;

CPreEmptiveTimer cPreEmtiveTimer2;

void setup() {

pinMode(9, OUTPUT);

pinMode(10, OUTPUT);

cPreEmtiveTimer1.iState = 0;

cPreEmtiveTimer1.setWaitConst(1000); // 1 sec

cPreEmtiveTimer2.iState = 0;

cPreEmtiveTimer2.setWaitConst(700); // 700 ms)

}

void loop() {

unsigned long currentMillis = millis();

if(cPreEmtiveTimer1.timeout(currentMillis)) {

switch(cPreEmtiveTimer1.iState) {

case 0:

digitalWrite(9, HIGH); // LED aan

cPreEmtiveTimer1.iState = 1; // nieuwe status

cPreEmtiveTimer1.setWaitConst(1000); // wacht 1 sec

break;

case 1:

digitalWrite(9, LOW); // LED uit

cPreEmtiveTimer1.iState = 0; // nieuwe status

cPreEmtiveTimer1.setWaitConst(1000); // wacht 1 sec

break;

}

}

if(cPreEmtiveTimer2.timeout(currentMillis)) {

switch(cPreEmtiveTimer2.iState) {

case 0:

digitalWrite(10, HIGH); // LED aan

cPreEmtiveTimer2.iState = 1; // nieuwe status

cPreEmtiveTimer2.setWaitConst(700); // wacht 1 sec

break;

case 1:

digitalWrite(10, LOW); // LED uit

cPreEmtiveTimer2.iState = 0; // nieuwe status

cPreEmtiveTimer2.setWaitConst(700); // wacht 1 sec

break;

}

}

}

Deze code staat in het voorbeeld programma   
 Libraries/CPreEmptiveTimer/examples/CPreEmptiveTimerSimple/CPreEmptiveTimerSimple.ino