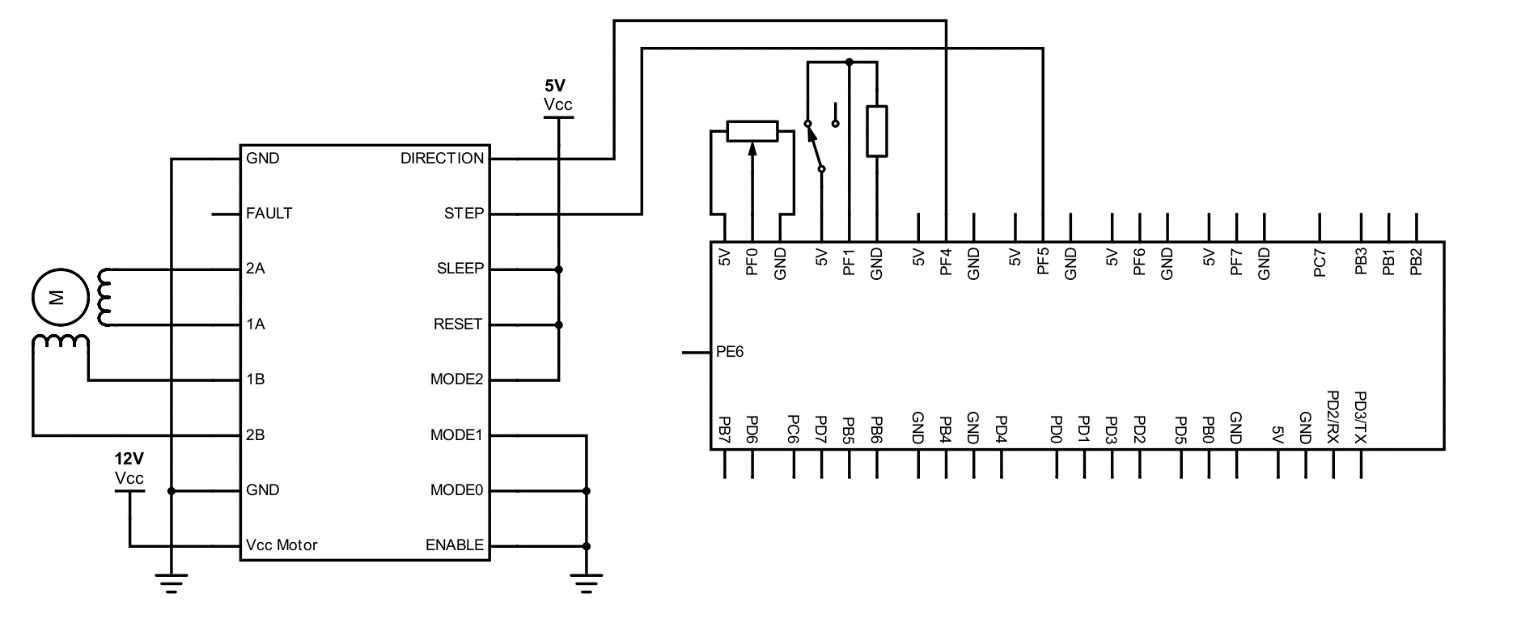
**Labo Pololu DRV8825 driver**

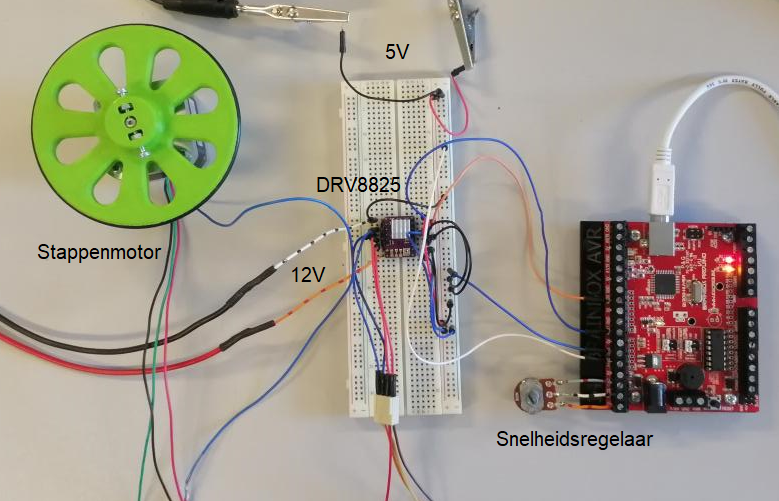
**Doelstelling**

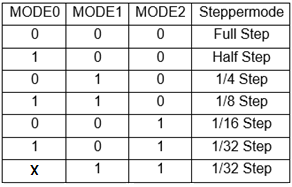
Het doel van dit labo is om te begrijpen hoe ik een DRV8825 microstep driver moet aansturen via een ATMEGA32U4 en daarmee een stappenmotor laten draaien op verschillende modes. Want ik moet er voor de robot 2 tegelijkertijd kunnen aansturen en snel van snelheid en draairichting laten veranderen.

**CH2**

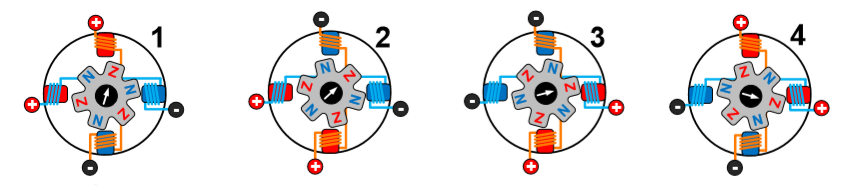
**CH1**

**CH1**

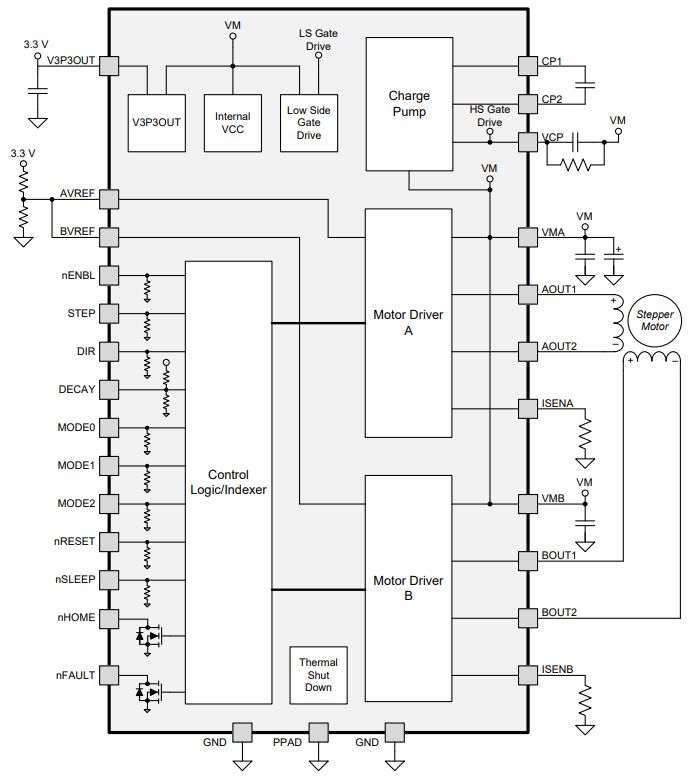
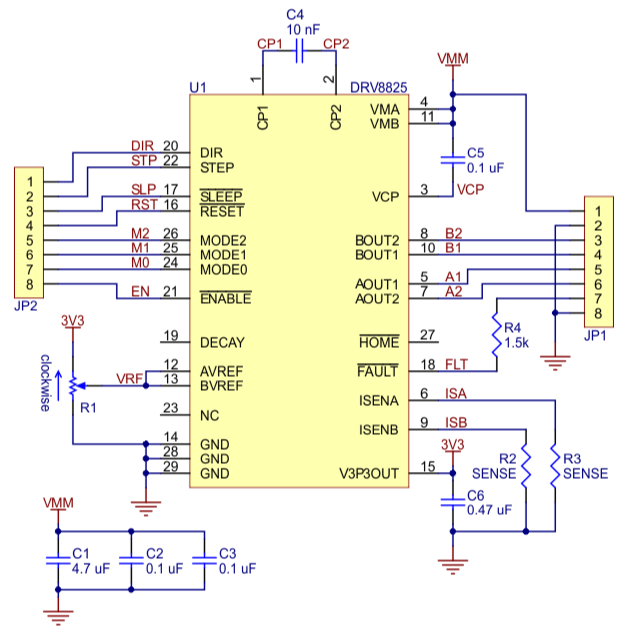
****

Het is een vrij simpel schema. Ik had mezelf de opdracht gegeven om een project te maken waarin ik een stappenmotor kon besturen. Ik zou de snelheid moeten kunnen regelen met een potentiometer en de draairichting kunnen veranderen met een soort schakelaar (mijn verwachtingen). Ik kan ook kiezen tussen Full step, Half step, 1/4 step, 1/8 step, 1/16 step en 1/32 step met de MODE pinnen (staat in datasheet) maar dat kon ik gewoon aanpassen door de draden op de breadboard te veranderen van 5V naar GND.

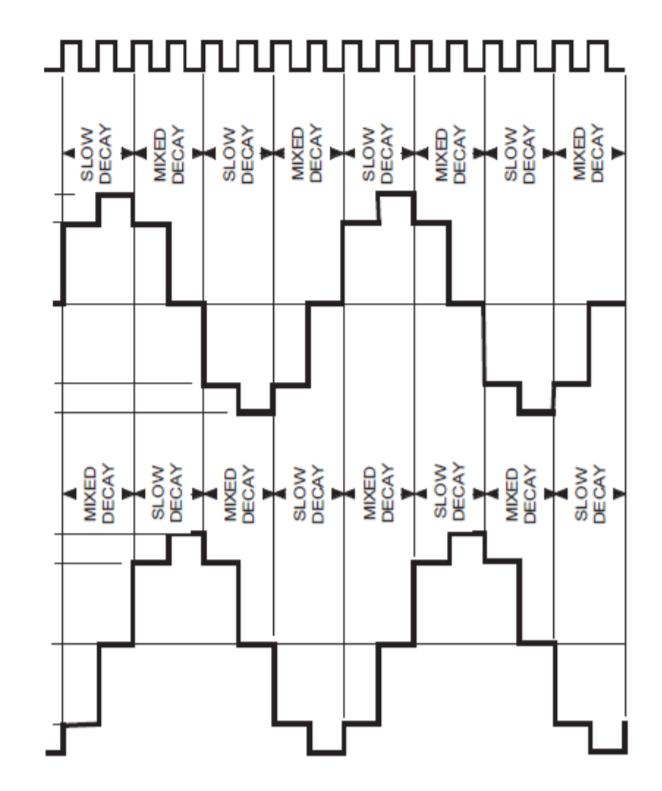
**Stappenmotor**

****Stappenmotoren zijn een soort DC motoren die in stappen draaien i.p.v. in een constante cirkel. Er zijn 2 soorten stappenmotoren : een bipolaire en een unipolaire. Ik gebruik een bipolaire. In de foto hieronder zie je een voorbeeld van “fullstep” sturing. We zien 4 spoelen (stator) en een magneet met alternerende polen (rotor), de “+” betekent dat daar de stroom binnenkomt en de “-” betekent dat daar de stroom dan buitenkomt. De spoelen waar een “+” boven staat trekken dan de noordpool aan en de “-“ trekken de zuidpool aan. Door de 4 spoelen van de motor juist aan te sturen kunnen we de rotor in stappen laten draaien (zie foto hieronder). Natuurlijk is deze foto een simpele versie van dit, een echte stappenmotor heeft veel meer alternerende polen aan de rotor en veel meer spoelen aan de stator.

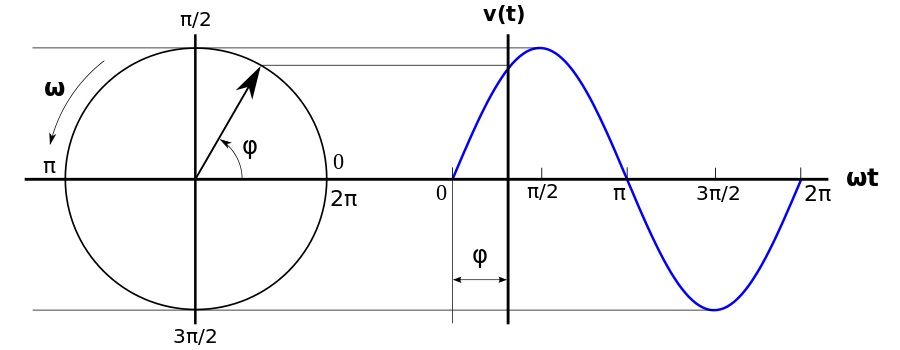
**DRV8825**

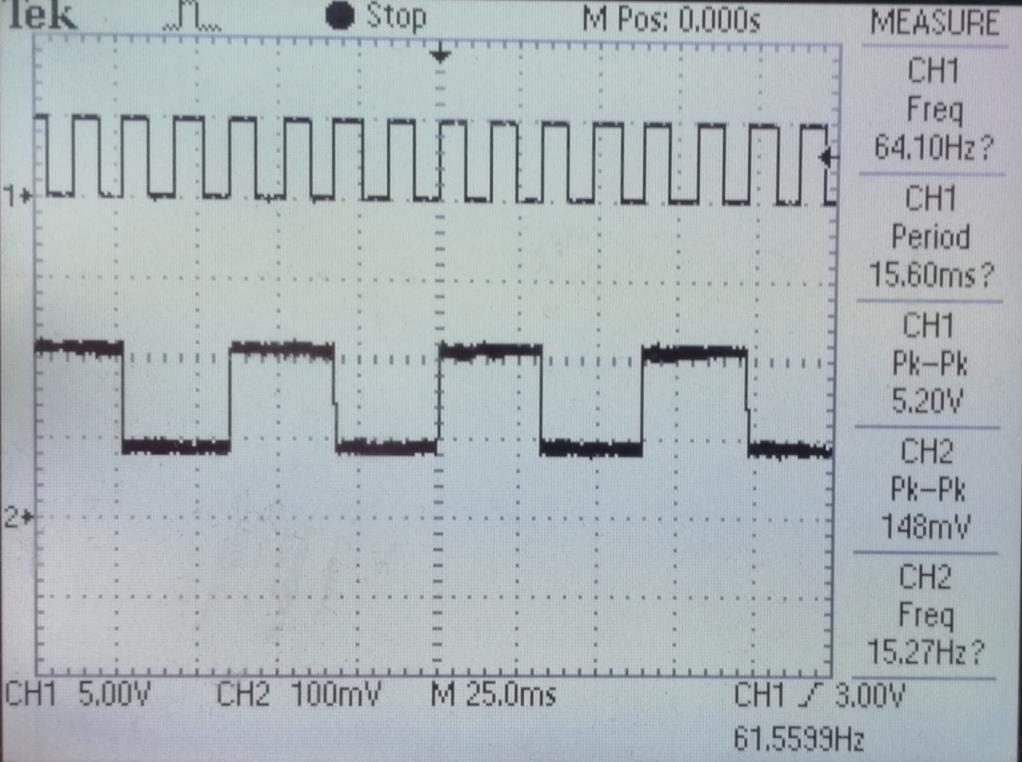
Hier zie je 2 duidelijke schema’s die het blokschema en de externe componenten op de driver laten zien. Als we naar het blokschema kijken zien we ingangen die we zelf moeten aansluiten (zoals de sleep, modes en step pin) en welke pull up en pull down weerstanden hebben. We zien dat de data gaat naar een controller die dan bepaald wanneer en welke soort signalen moeten gestuurd worden naar de motor drivers. De soort signalen kunnen we zien verder in het verslag bij metingen. Deze driver wordt aangestuurd door pulsen te sturen naar de step pin. Met de driver kunnen we de motor dus heel simpel en op veel verschillende manieren aansturen, zoals microstep. Fullstep is simpel daar worden gewoon de pinnen verschillend hoog en laag gemaakt. Maar bij microstep word bij elke puls een volgende stap gezet in een blokvormig AC sinus signaal (duidelijker te zien bij de metingen). Het nut van microstepping is om de stappenmotor zo nauwkeurig mogelijk te laten draaien. Dit is nodig in bijvoorbeeld een 3D printer, een PCB freesmachine of een balancerende robot.

**Metingen**

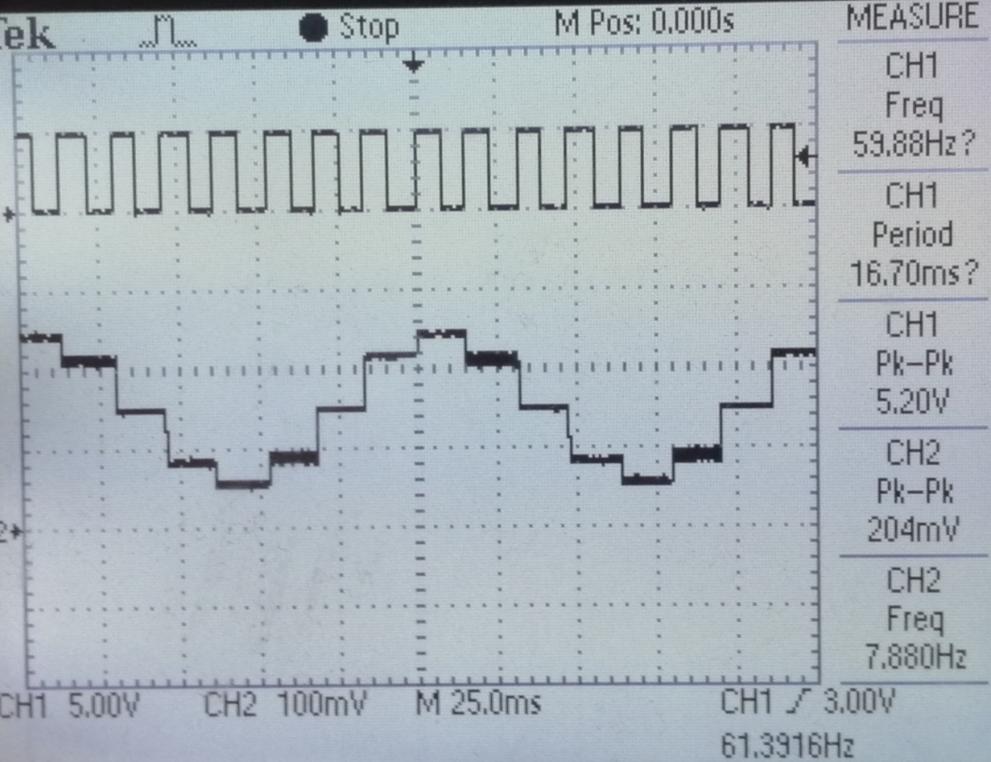
Voor alle metingen op CH2 (de faseverschuivingen en stappen) moest ik een stroomprobe gebruiken. De reden hiervoor is omdat de DRV8825 de stroom moduleert voor microstepping. De probe die ik gebruik (zie foto, Pico TA018) heeft 2 modes: 1mV/10mA met max 20A en 1mV/100mA met max 60A. Dus ik gebruik de eerste mode en dat betekent dat de spanning van CH2 de stroom voorstelt (bvb 150mV = 1,5A). Maar je moet wel de gemeten spanning door 2 delen (dus 150mV = 0,75A) want net zoals bij een AC signaal (zoals op de onderste foto) is het nulpunt in de helft en omdat we Pk-Pk/Peak to Peak meten, meten we de positieve waarden en de negatieve waarden en tellen de absolute waarde op. Daarom moeten we het door 2 delen.

Op de scoopbeelden zal het ook lijken dat we maar 1 signaal naar de motor sturen. Maar eigenlijk stuur ik 2 signalen met een faseverschuiving van 90° , zoals we zien op de foto.

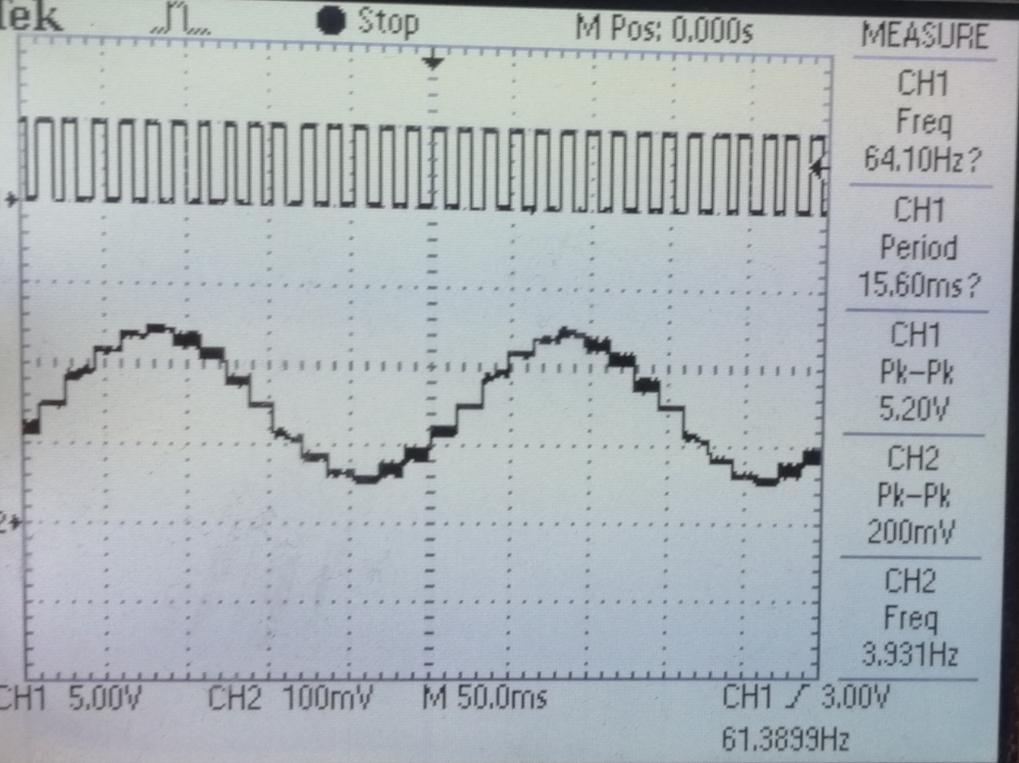
****

****

Dit is Full Step dus alle MODE pinnen zijn aan de GND verbonden (zie tabel bij schema). Door de pulsen zichtbaar op CH1 naar de driver te sturen, krijgen we de stappen zichtbaar op CH2. We kunnen ook zien dat de frequentie van CH1 gelijk is aan de frequentie van CH2 gedeeld door 4 (64Hz / 4 = 16Hz). Normaal wordt dit signaal 2 keer naar de motor gestuurd, maar ik kan er maar 1 tegelijkertijd meten. Het is ook duidelijk hoe elke stap start bij een rising edge van CH1. We zien ook de stroom door de motor weer met CH2 Pk-Pk.

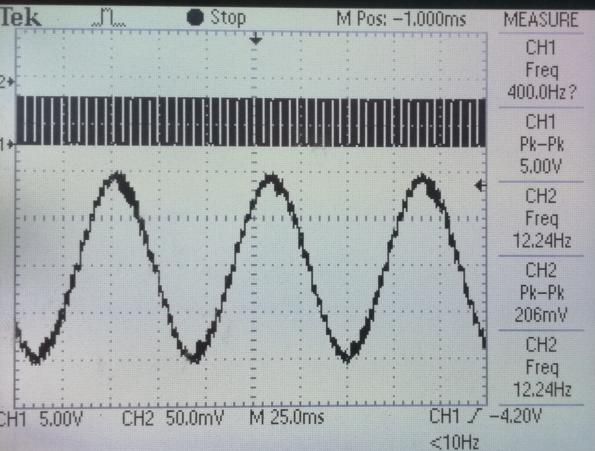


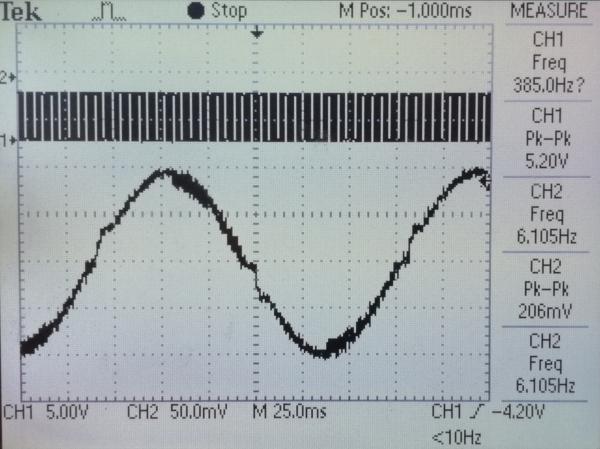
Dit is Half Step dus alle MODE pinnen zijn hetzelfde behalve MODE0, die is hoog (zie tabel bij schema). Hier zien we duidelijk dat het signaal een soort blokvormige sinusgolf wordt. Het is ook duidelijk hoe bij elke rising edge van CH1 een verandering in CH2 komt. Het is ook zo dat de frequentie van CH1 gelijk is aan de frequentie van CH2 gedeeld door 8 (60Hz / 8 = 7.5Hz). We zien ook de stroom door de motor weer met CH2 Pk-Pk.



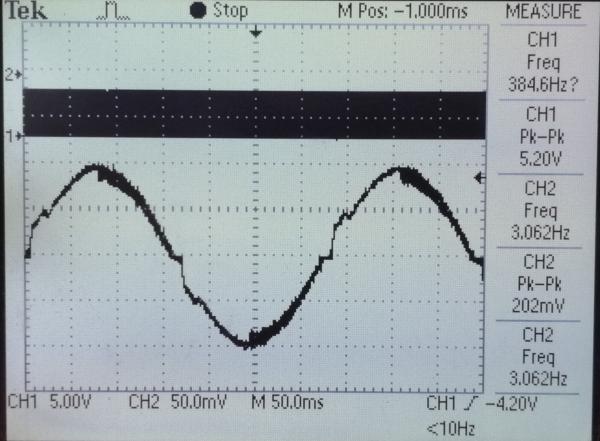
Dit is 1/4 Step dus de MODE pinnen zijn aangesloten in een 010 configuratie (zie tabel bij schema). We zien duidelijk dat het nog steeds een blokvormig sinus signaal is maar deze keer met kleinere blokken. Het is ook zo dat de frequentie van CH1 gelijk is aan de frequentie van CH2 gedeeld door 16 (64Hz / 16 = 4Hz). We zien ook de stroom door de motor weer met CH2 Pk-Pk.

Dit is 1/8 Step dus de MODE pinnen zijn aangesloten in een 110 configuratie (zie tabel bij schema). Hier is het duidelijk dat er nog steeds een blokvormig sinus signaal is, maar deze keer met nog zeer kleine blokken. We zien ook de stroom door de motor met de CH2 Pk-Pk meting. Het is ook zo dat de frequentie van CH1 gelijk is aan de frequentie van CH2 gedeeld door 32 (400Hz / 32 = 12.5Hz).





Dit is 1/16 Step dus de MODE pinnen zijn aangesloten in een 001 configuratie (zie tabel bij schema). We zien deze keer gewoon niet dat het nog blokvormig is want de blokvormige delen zijn gewoon te klein. We zien ook de stroom door de motor weer met de CH2 Pk-Pk meting. Het is ook zo dat de frequentie van CH1 gelijk is aan de frequentie van CH2 gedeeld door 64 (385Hz / 64 = 6.015Hz).



Dit is 1/32 Step dus de MODE pinnen zijn aangesloten in een 101, 011 of 111 configuratie (zie tabel bij schema). We zien ook hier niet dat het nog blokvormig is omdat de blokvormige delen gewoon te klein zijn. We zien ook de stroom door de motor weer met de CH2 Pk-Pk meting. De frequentie van CH1 is gelijk aan de frequentie van CH2 gedeeld door 128 (385Hz / 128 = 3.007Hz).

**Software**

unsigned int ADC0**;** // Variable waar de ADC waarde in wordt opgeslagen

void setup**()**

**{**

DDRF **|=** 0b00110000**;** // Maak PF5 en PF4 outputs en verander de andere pinnen niet (door het masker)

Serial**.**begin**(**9600**);** // Bautrate van de seriële monitor zetten op 9600

unsigned int READ\_ADC\_INT\_CHANNEL**(** unsigned char channel **);** // Prototype functie voor ADC naar een INT

**}**

void loop**()**

**{**

ADC0 **=** READ\_ADC\_INT\_CHANNEL**(**0**);** // Lees de potmeter waarde op PF0 in en slaag het op in de variable ADC0

ADC0 **=** map**(**ADC0**,** 0**,** 1023**,** 1**,** 1005**);** // De minimum waarde van de ADC meting is i.p.v. 0 nu 1 en de maximum waarde is 105 i.p.v. 1023

Serial**.**println**(**ADC0**);** // De potmeter waarde en de stand van de schakelaar op de seriële monitor weergeven

Serial**.**print**(**"\t"**);**

Serial**.**println**(**digitalRead**(**22**));**

**if(**ADC0 **<=** 1000**)** // Laat de motor alleen draaien als er al een klein beetje aan de potmeter is gedraaid

**{**

PORTF **^=** 0b00100000**;** // Maak pulsen waarvan je de frequentie van kan aanpassen met de potmeter

delay**(**ADC0**);** // Er kan ook een for loop gebruikt worden om te delay te regelen maar dit is simpeler en korter

**}**

**if(**PINF **&** 0b00000010**)** // Als PF1 hoog is/als de schakelaar is overgehaald

**{**

PORTF **|=** 0b00010000**;** // Maak alleen PF4 hoog

**}**

**else**

**{**

PORTF **&=** 0b11101111**;** // Maak alleen PF4 laag

**}**

**}**

unsigned int READ\_ADC\_INT\_CHANNEL**(** unsigned char channel **)**

**{**

DDRF **&=** **~(**1**<<**channel**);** // config selected channel as input

// Right adjust + choice of channel

ADMUX **=** 0b01000000 **+** channel**;**

// Activate ADC - Stop conversion - prescaler 128 (for ADC 50K<..<200k) (16Mhz / 128 = 125000)

ADCSRA **=** 0b10000111**;**

// start conversion and wait for completion

ADCSRA **|=** **(**1**<<**ADSC**);**

**while** **(**ADCSRA **&** **(**1**<<**ADSC**));**

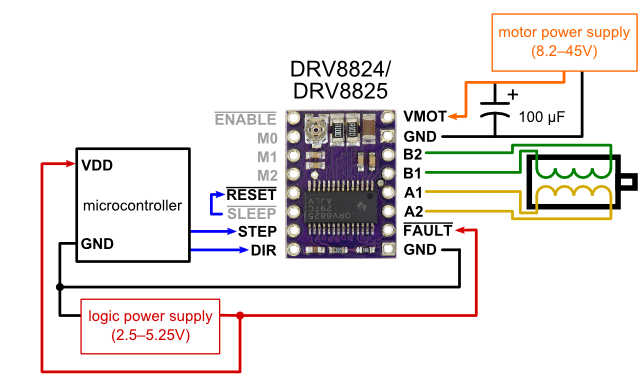
// return A/D conversion result

unsigned char ADCLBuff **=** ADCL**;** // first read ADCL!!

**return** **(** **((**unsigned int**)** ADCH **)** **<<** 8 **)** **+** ADCLBuff**;**

**}**

**Besluit**

Door dit labo heb ik uit eigen ervaring en testen geleerd hoe de DRV8825 driver werkt. Voorbeelden van dingen die ik geleerd heb zijn : de regelbare weerstand op de driver zorgt voor een stroomlimiet door de driver. In mijn schema heb ik niets gedaan met de FAULT pin, maar voor beveiliging ga ik die met een weerstand verbinden aan de SLEEP pin. Zodat als er een fout gebeurt, de driver in sleep modus gaat. Ten slotte zou ik ook nog condensatoren hebben moeten gebruiken aan de voeding voor de mot or om de spanning en stroompieken te filteren. Zoals op de foto hieronder.