Verslag Fablab Smart Objects

Dylan Duunk

June 19, 2018

Contents

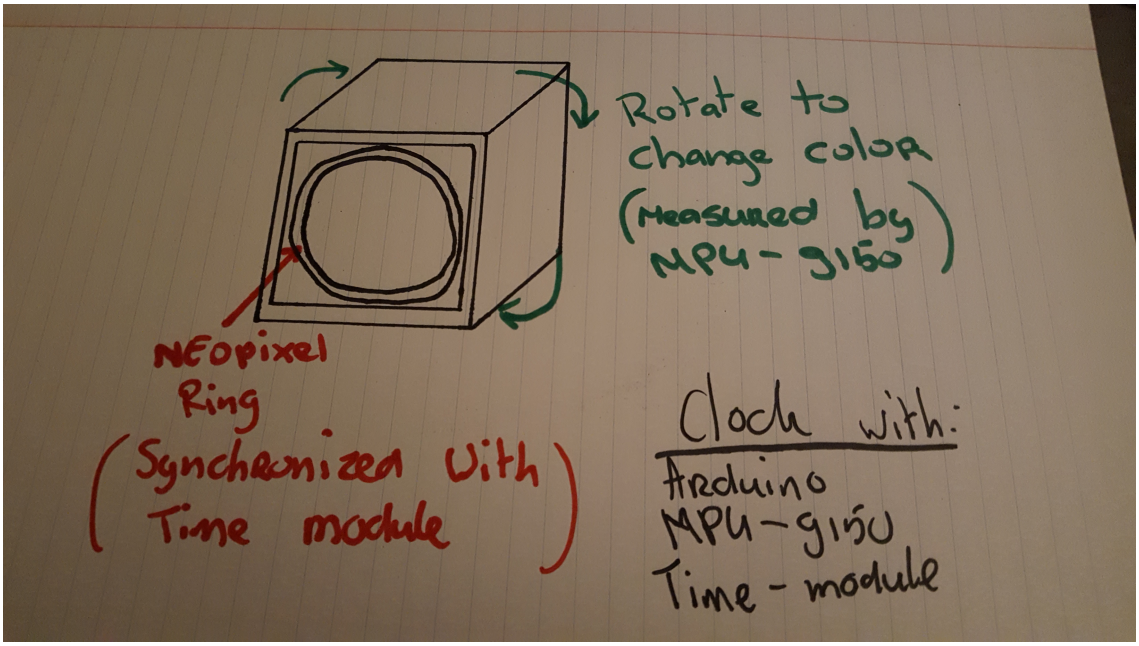
1 Inleiding

Voor het keuzevak Fablab Smart Objects is ons de opdracht geven om een "Smart Object" te maken. Dit wordt gedaan met de samenwerking van een Arduino (of andere microcontroller) en verschillende sensoren. In dit document beschrijf ik mijn "Smart Object".

2 Concept

Ik heb ervoor gekozen om een klok te maken en daar mijn eigen draai aan te geven. Wat mijn klok "smart" maakt is een sensor die graden/rotatie meet. Hiermee kan ik een klok maken die omgedraait kan worden en op deze manier van kleur veranderd. Ik maak deze klok met behulp van een NeoPixel Ring, een RTC-Module en een MPU-9150.

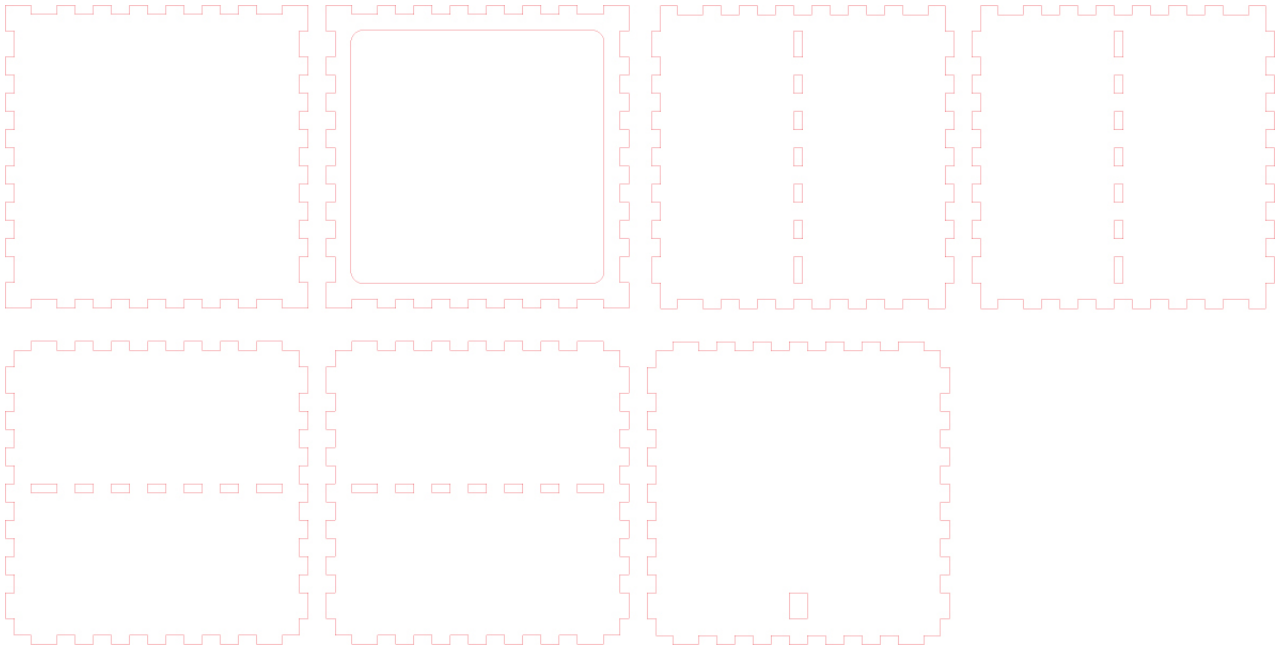
Dit is mijn concept ontwerp:



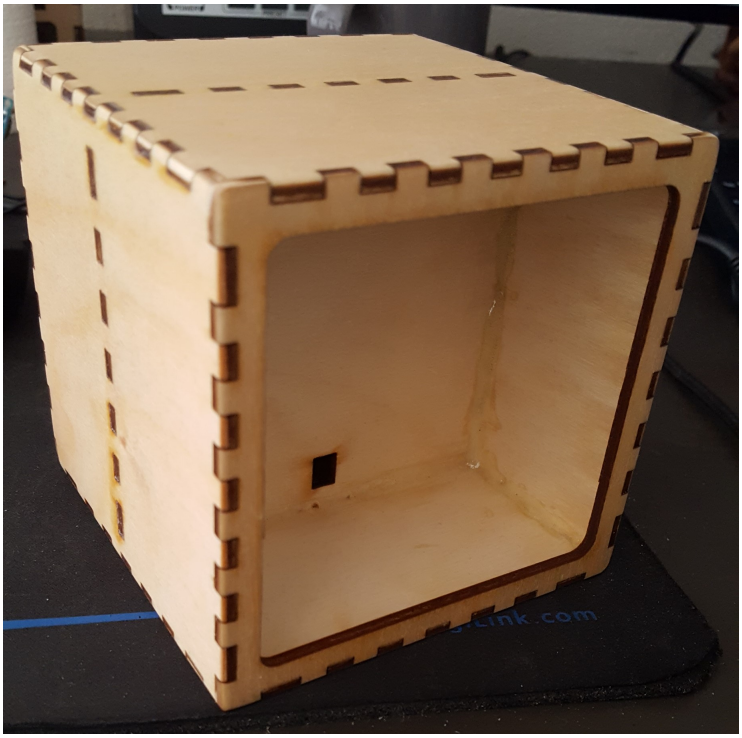
3 Product

3.1 Ontwerp

Dit is het ontwerp van de behuizing voor de lazersnijder.



Dit ontwerp heb ik uitgesneden op de lazersnijder en in elkaar gezet. Vervolgens heb ik het goed geschuurd en ingesmeerd met meubelolie voor een betere afwerking. Dit is het resultaat daarvan:

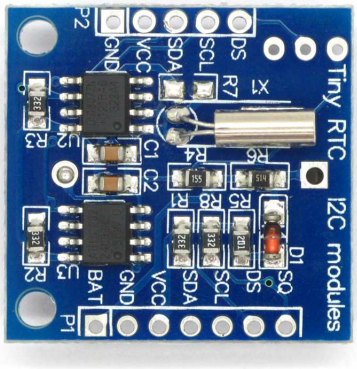


3.2 Hardware

Hier beschrijf ik de gebruikte sensoren:

3.2.1 RTC-Module

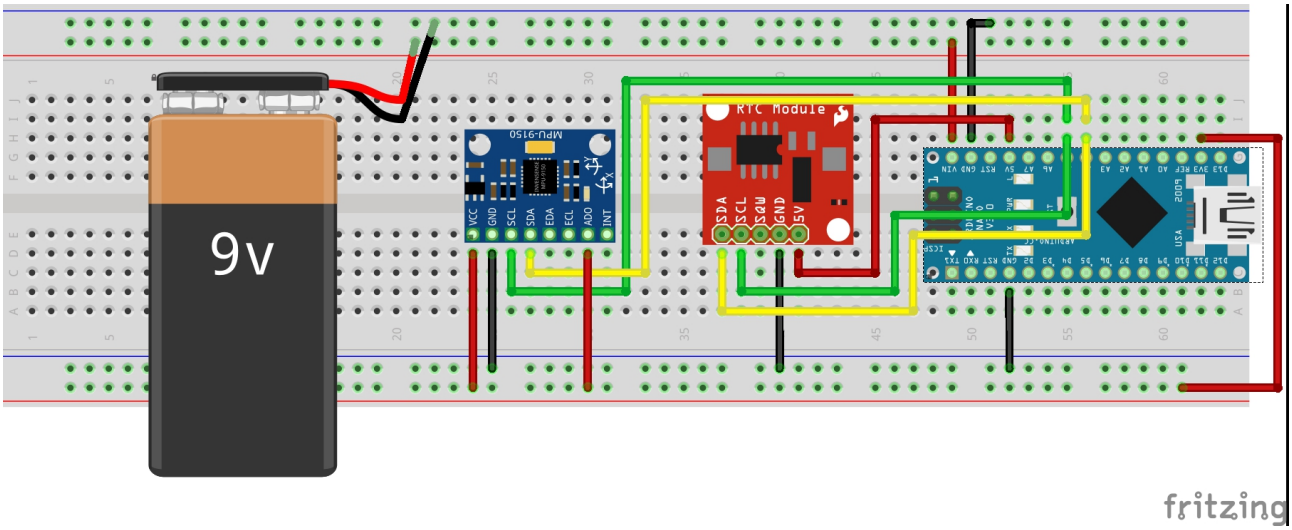
De RTC-Module is een tijdsensor. Deze sensor werkt ook met I2c met de Arduino. Het I2c adres van deze sensor is 0x68. Dit adres is van belang als er meerdere I2c apparaten worden aangesloten op de Arduino.



3.2.2 MPU-9150

De MPU-9150 is een 9-axis bewegingsregistrator sensor. Deze sensor werkt via I2c met de Arduino. Zoals al eerder is verteld is het I2c adres van belang bij het gebruik van meerdere I2c apparaten. Dit is waarom: De MPU-9150 gebruikt als standaard I2c adres ook 0x68 (hetzelfde adres als de RTC-Module) hierdoor onstaan problemen met sensoren uitlezen. Gelukkig is het mogelijk om het I2c adres van de MPU-9150 te veranderen naar 0x69, dit is mogelijk door AD0 aan te sluiten op 3.3VCC.

3.2.3 Elektrisch Schema



3.3 Code

Coderen speelt een grote rol bij het ontwikkelen van een "Smart Object". Met coderen bepaal je hoe het object werkt en hoe je het wilt gebruiken. Hier laat ik verschillende stukken code en libraries zien.

3.3.1 Libraries

Voor de code heb ik gebruik gemaakt van verschillende libraries, hier zijn de belangrijkste:

Library voor de NeoPixel Ring:
`https://github.com/adafruit/Adafruit_NeoPixel`

Libraries voor de RTC-Module:
`https://github.com/adafruit/RTClib`
`https://www.arduino.cc/en/reference/wire`

Libraries voor de MPU-9150:
`https://github.com/jrowberg/i2cdevlib/blob/master/Arduino/I2Cdev/I2Cdev.h`
`https://github.com/alexvonduar/mpu9150-arduino-lib/blob/master/libraries/MPU9150Lib/MPU9150Lib.h`

3.3.2 Code fragmenten

Dit is de code voor de afhandeling van de rotatie:

Listing 1: NEOPIXEL.cpp

```
void NEOPIXEL_RUN() {
    if (MPU.xValue > 0 && MPU.xValue < 90) {
        mode = 1;
    } else if (MPU.xValue > 90 && MPU.xValue < 180) {
        mode = 2;
    } else if (MPU.xValue < 0 && MPU.xValue > -90) {
        mode = 3;
    } else if (MPU.xValue < -90 && MPU.xValue > -180) {
        mode = 4;
    }
    timeCycle(second, minute, hour, mode);
}
```

Hier verander ik de kleur t.o.v. het variable "mode" die van het vorige stukje code afkomt. Ook zorg ik dat de tijd over de 24 led's van de NeoPixel Ring wordt verspreidt.

Listing 2: NEOPIXEL.cpp

```
void timeCycle(int sec, int min, int h, int mode) {
    switch(mode) {
        case 1: cSec = strip.Color(255, 0, 0); //red
                cMin = strip.Color(0, 255, 0); //green
                cHour = strip.Color(0, 0, 255); //blue
                break;
        case 2: cSec = strip.Color(0, 255, 0); //green
                cMin = strip.Color(0, 0, 255); //blue
                cHour = strip.Color(255, 0, 0); //red
                break;
        case 3: cSec = strip.Color(0, 0, 255); //blue
                cMin = strip.Color(255, 0, 0); //red
                cHour = strip.Color(0, 255, 0); //green
                break;
        case 4: cSec = strip.Color(255, 0, 0);
                cMin = strip.Color(255, 255, 0);
                cHour = strip.Color(255, 0, 255);
                break;
    }
```

```
    if(((sec - 1) / 2.5) < 0 || ((min - 1) / 2.5) < 0 ||
       ((h - 1) * 2) < 0) {
        strip.setPixelColor(23, 0, 0, 0);
        strip.show();
    } else if (((sec - 1) / 2.5) < 1 || ((min - 1) / 2.5) < 1 ||
       ((h - 1) * 2) < 1) {
        strip.setPixelColor(0, 0, 0, 0);
        strip.show();
    } else {
        strip.setPixelColor(((sec - 1) / 2.5), 0, 0, 0);
        strip.setPixelColor(((min -1) / 2.5), 0, 0, 0);
        strip.setPixelColor(((h - 1) * 2), 0, 0, 0);
        strip.show();
    }

    strip.setPixelColor(sec / 2.5, cSec);
    strip.setPixelColor(min / 2.5, cMin);
    strip.setPixelColor(h * 2, cHour);
    strip.show();
}
```

In dit stukje code wordt de RTC-Module gesynchronizeerd met de tijd van je computer. Dit gebeurt tijdens het compileren van de code.

Listing 3: RTC.cpp

```
void RTC_INIT() {
    while (!Serial);
    if (!rtc.begin()) {
        Serial.println("Couldn't find RTC");
        while (1);
    }
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
}
```

Dit is de code om de MPU-9150 te initializeren:

Listing 4: MPU.cpp

```
void MPU_INIT() {
    Serial.print("Arduino9150 starting using device ");
    Serial.println(DEVICE_TO_USE);
    Wire.begin();
```

```
    MPU.selectDevice(DEVICE_TO_USE);
    MPU.init(MPU_UPDATE_RATE, MPU_MAG_MIX_GYRO_AND_MAG, MAG_UPDATE_RAT
}
```

Voor de volledige code zie: `https://github.com/DDuunk/Clock`