Heilmann G4 Hugo

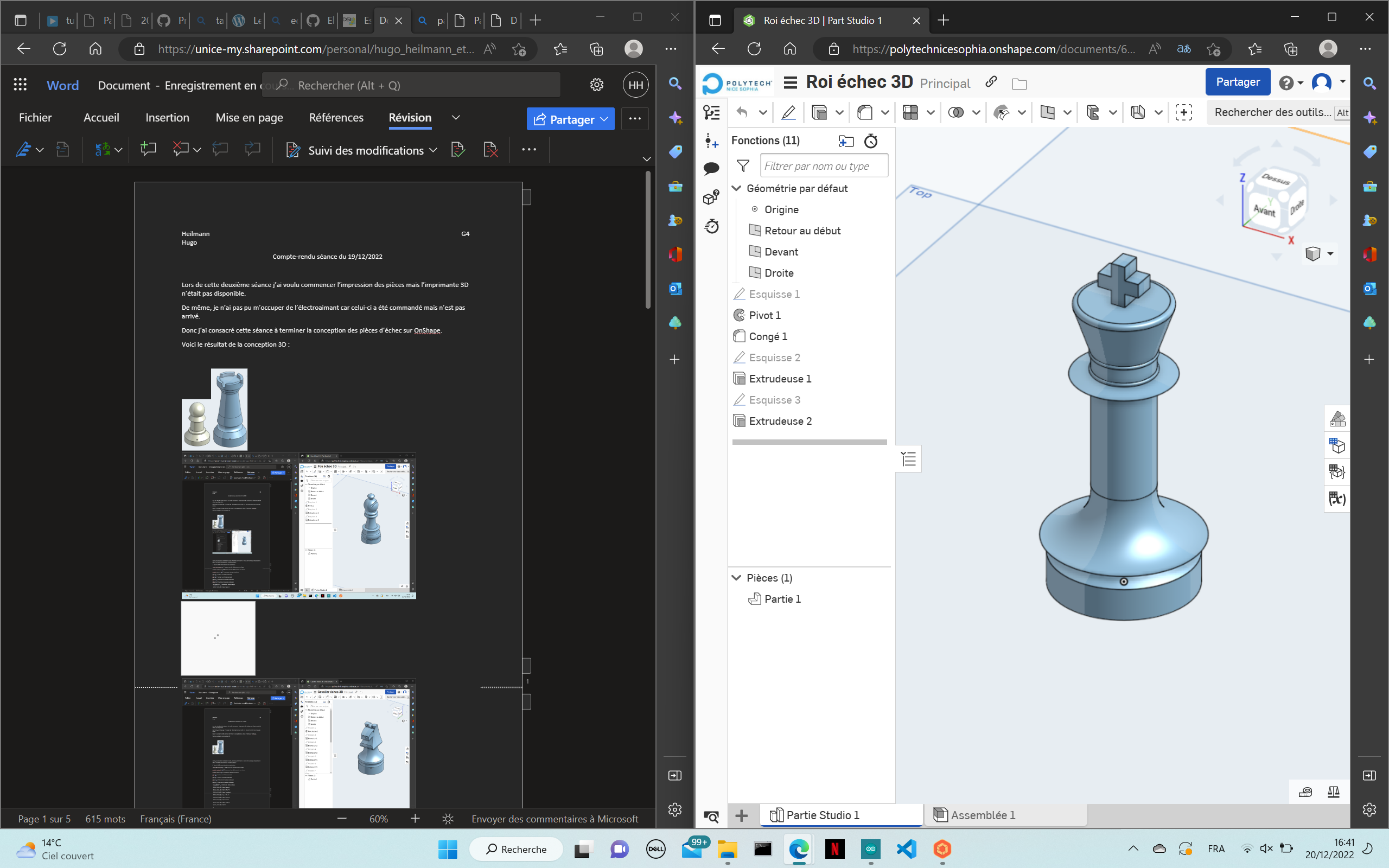
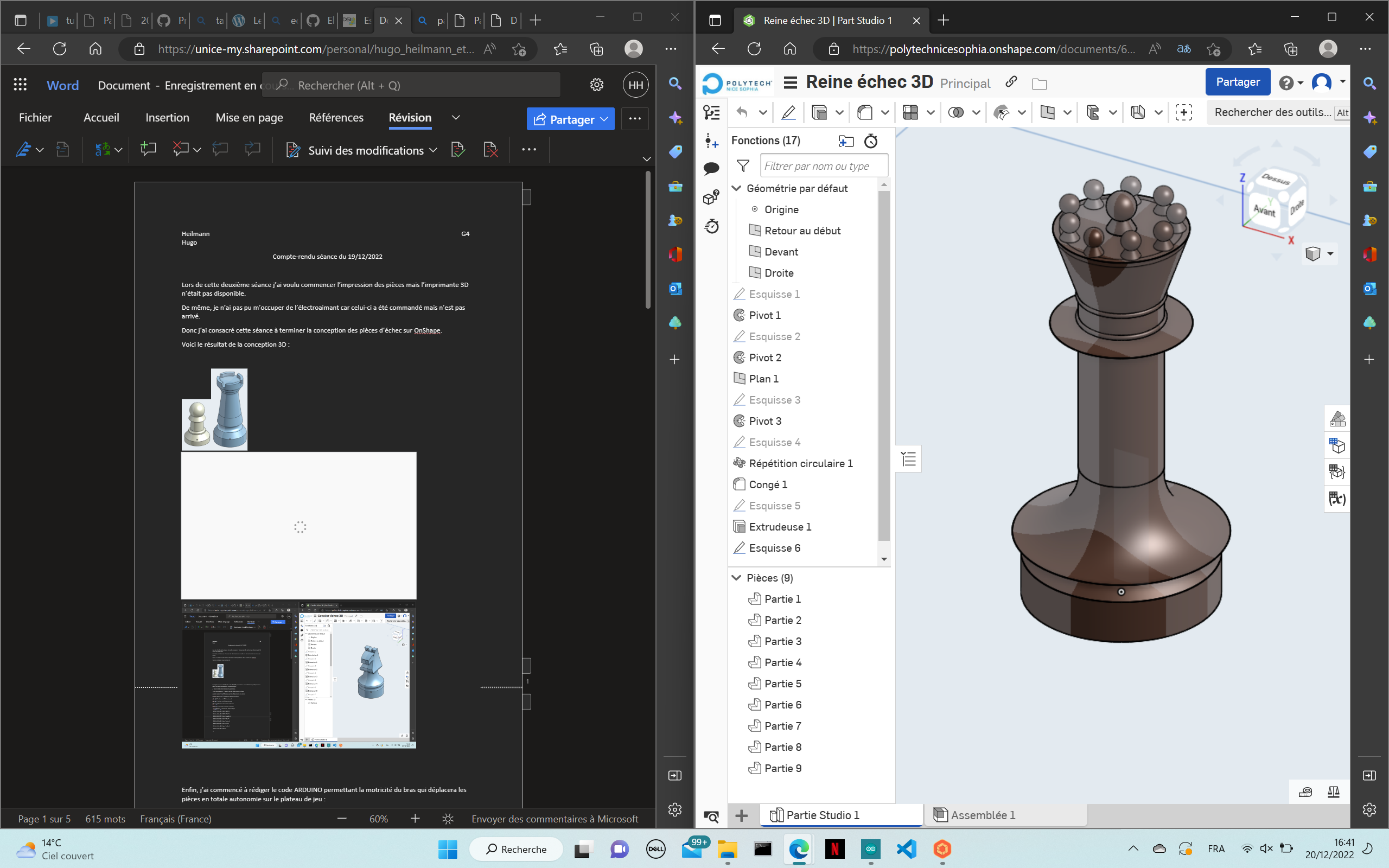
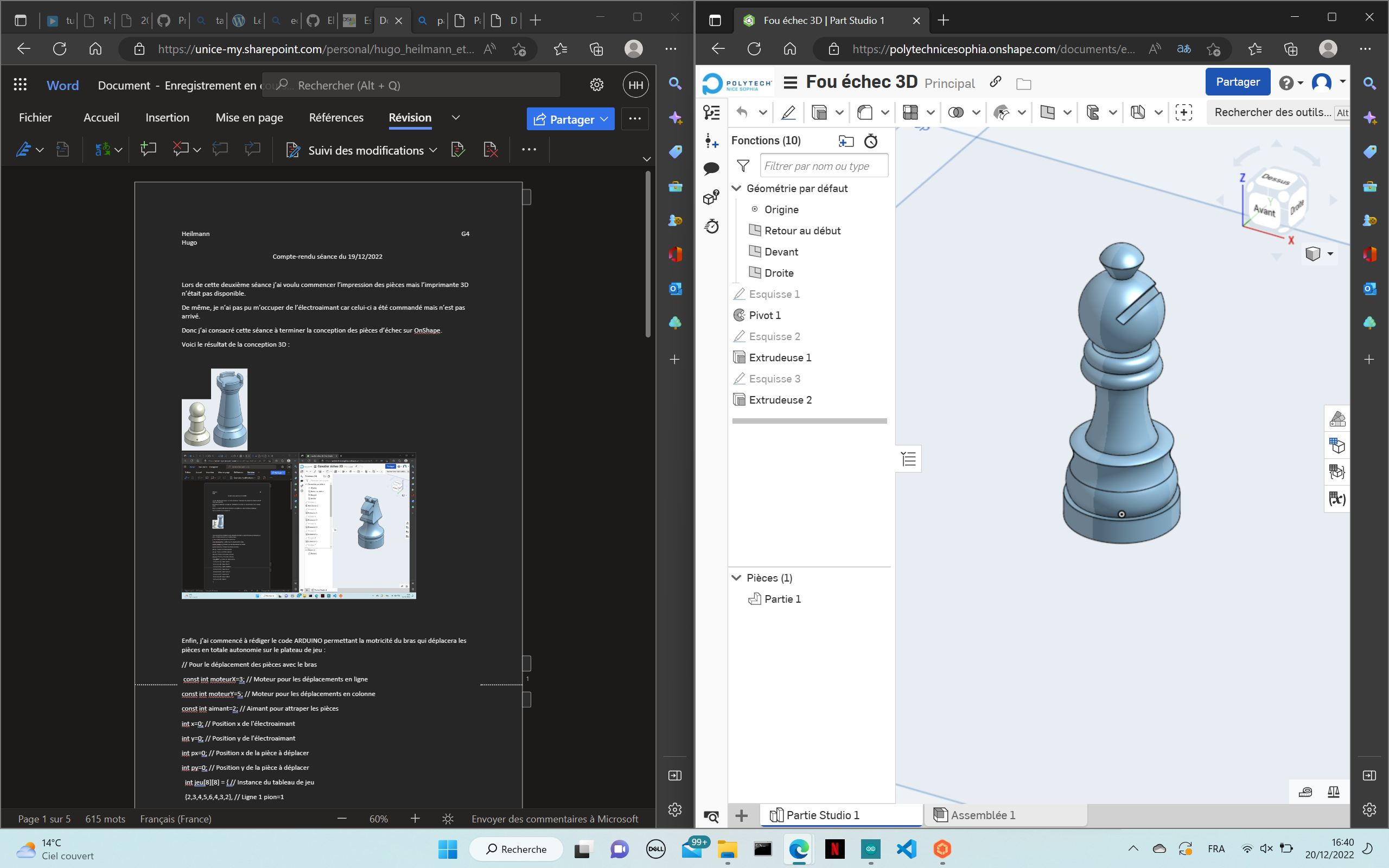
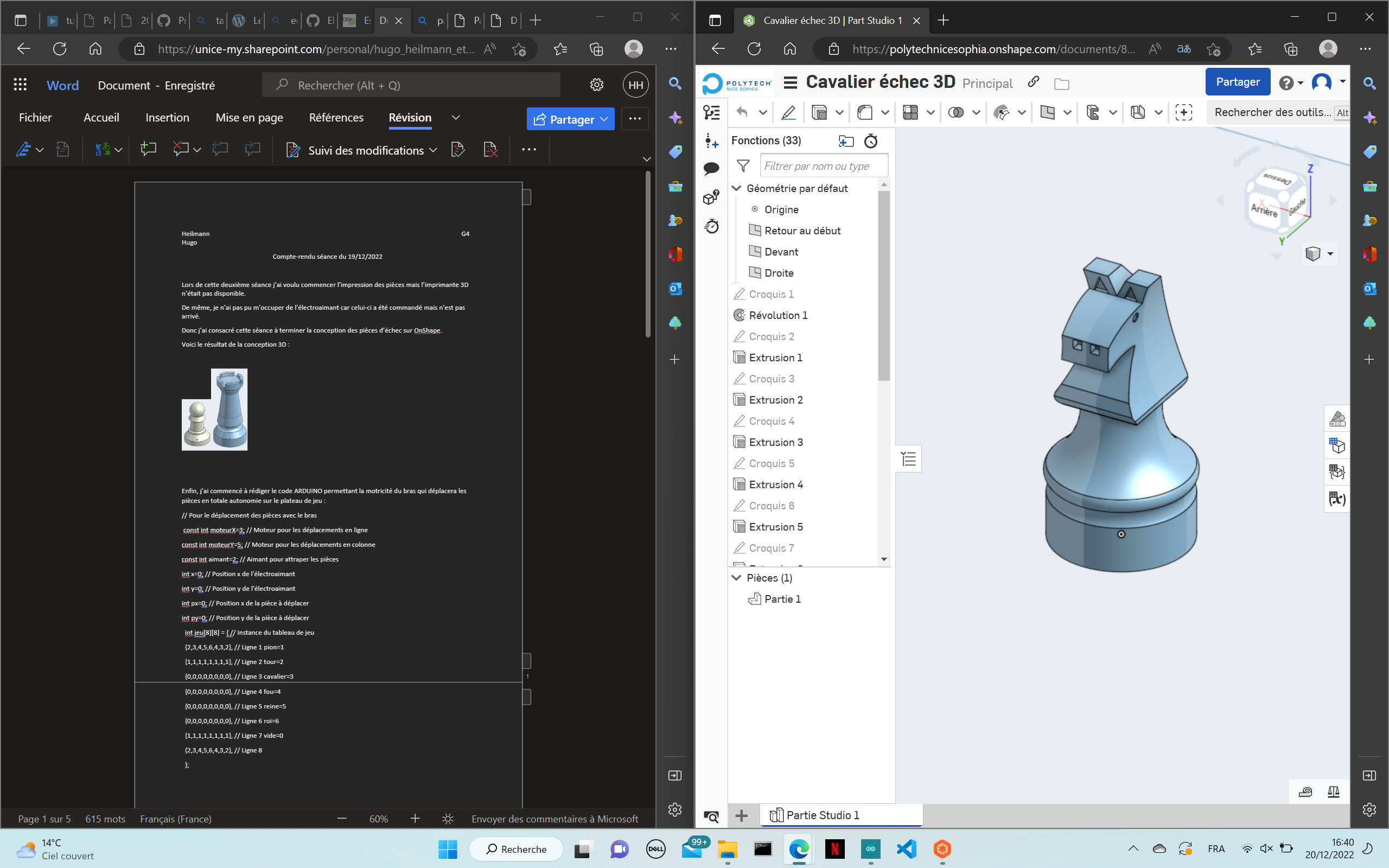
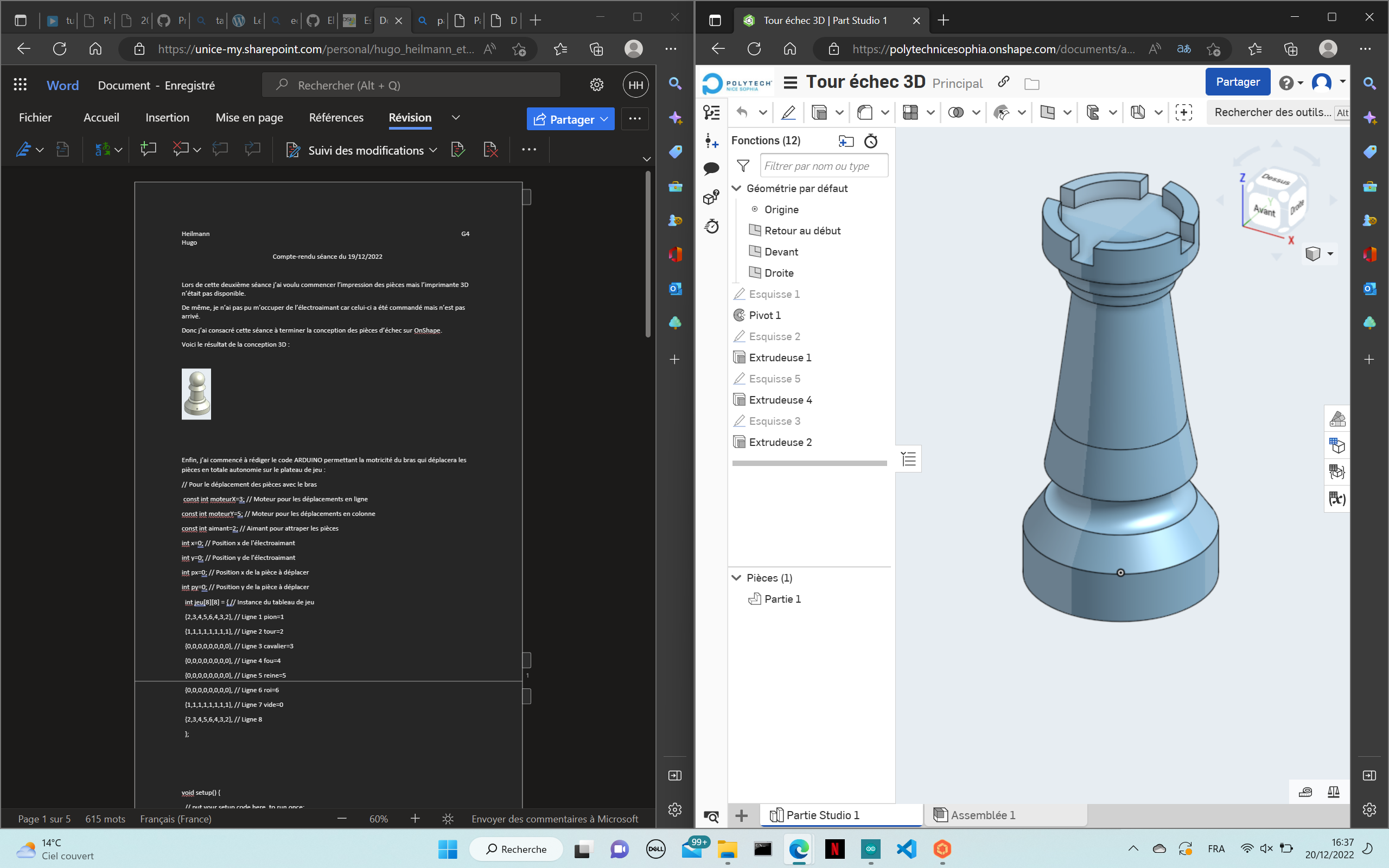
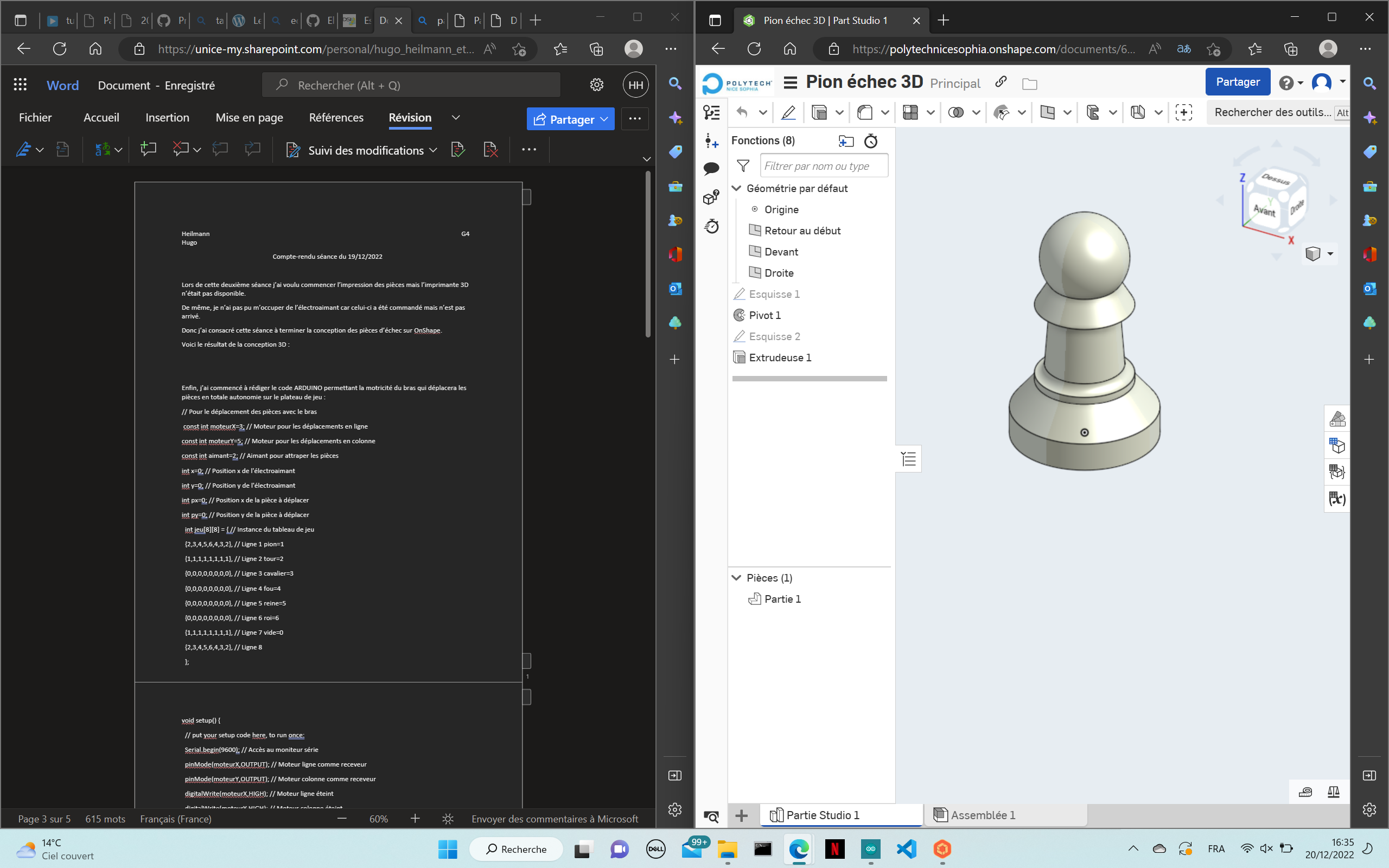
Compte-rendu séance du 19/12/2022

Lors de cette deuxième séance j’ai voulu commencer l’impression des pièces mais l’imprimante 3D n’était pas disponible.

De même, je n’ai pas pu m’occuper de l’électroaimant car celui-ci a été commandé mais n’est pas arrivé.

Donc j’ai consacré cette séance à terminer la conception des pièces d’échec sur OnShape.

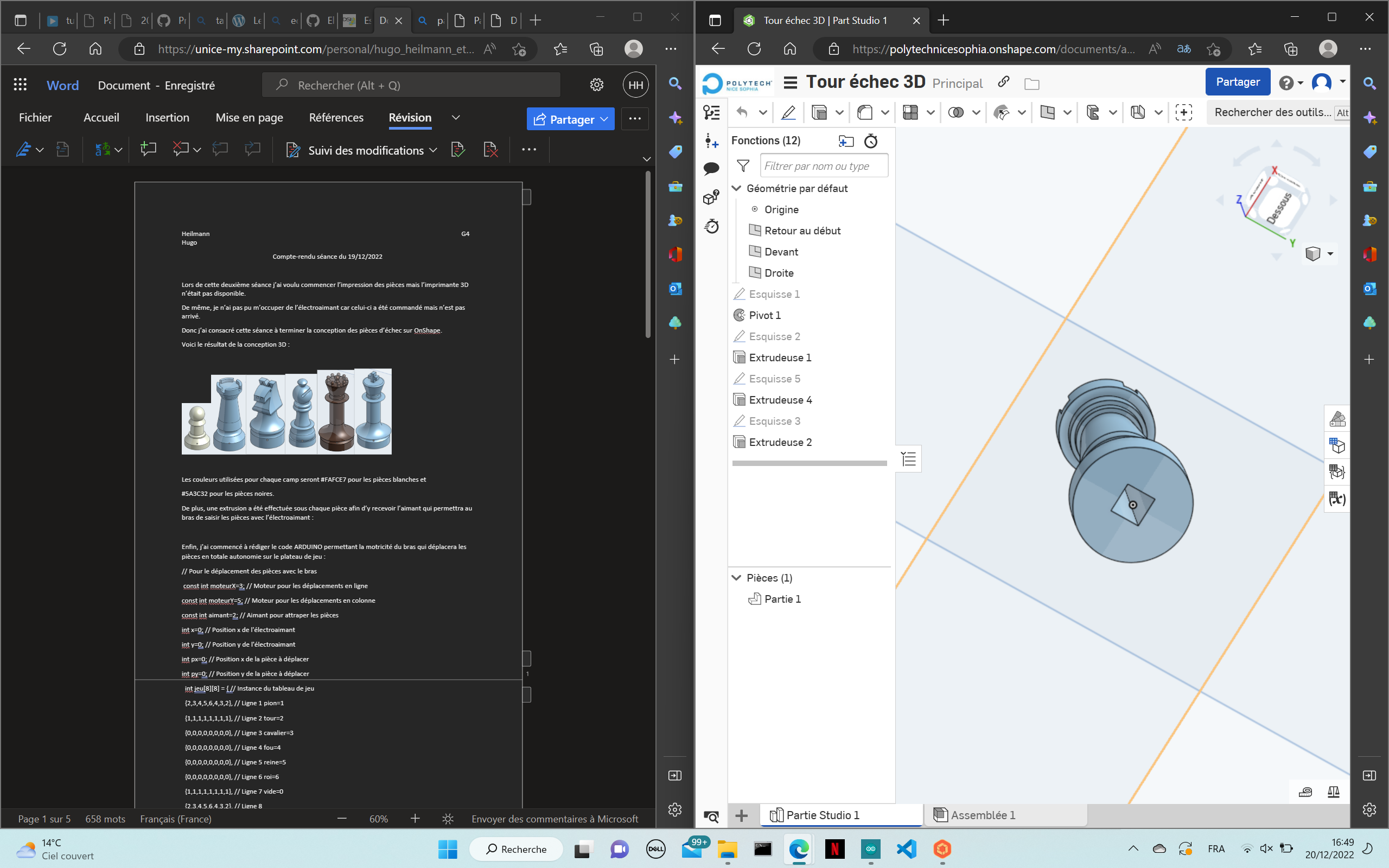
Voici le résultat de la conception 3D :



Les couleurs utilisées pour chaque camp seront #FAFCE7 pour les pièces blanches et

#5A3C32 pour les pièces noires.

De plus, une extrusion a été effectuée sous chaque pièce afin d’y recevoir l’aimant qui permettra au bras de saisir les pièces avec l’électroaimant :



Enfin, j’ai commencé à rédiger le code ARDUINO permettant la motricité du bras qui déplacera les pièces en totale autonomie sur le plateau de jeu :

// Pour le déplacement des pièces avec le bras

const int moteurX=3; // Moteur pour les déplacements en ligne

const int moteurY=5; // Moteur pour les déplacements en colonne

const int aimant=2; // Aimant pour attraper les pièces

int x=0; // Position x de l'électroaimant

int y=0; // Position y de l'électroaimant

int px=0; // Position x de la pièce à déplacer

int py=0; // Position y de la pièce à déplacer

int jeu[8][8] = { // Instance du tableau de jeu

{2,3,4,5,6,4,3,2}, // Ligne 1 pion=1

{1,1,1,1,1,1,1,1}, // Ligne 2 tour=2

{0,0,0,0,0,0,0,0}, // Ligne 3 cavalier=3

{0,0,0,0,0,0,0,0}, // Ligne 4 fou=4

{0,0,0,0,0,0,0,0}, // Ligne 5 reine=5

{0,0,0,0,0,0,0,0}, // Ligne 6 roi=6

{1,1,1,1,1,1,1,1}, // Ligne 7 vide=0

{2,3,4,5,6,4,3,2}, // Ligne 8

};

void setup() {

// put your setup code here, to run once:

Serial.begin(9600); // Accès au moniteur série

pinMode(moteurX,OUTPUT); // Moteur ligne comme receveur

pinMode(moteurY,OUTPUT); // Moteur colonne comme receveur

digitalWrite(moteurX,HIGH); // Moteur ligne éteint

digitalWrite(moteurY,HIGH); // Moteur colonne éteint

digitalWrite(aimant,HIGH); // Aimant desactivé

}

void loop() {

// put your main code here, to run repeatedly:

// px=digitalRead(...); valeur x de l'emplacement de la pièce à déplacer(valeur de 0 à 7 inclus)

// py=digitalRead(...); valeur y de l'emplacement de la pièce à déplacer(valeur de 0 à 7 inclus)

}

void aller\_case(int goX, int goY) {

int diffX=goX-x;

int diffY=goY-y;

avancer\_caseX(diffX);

avancer\_caseY(diffY);

x+=diffX;

y+=diffY;

px=x;

py=y;

}

void avancer\_caseX(int ax) { // avancer/reculer jusqu'à la colonne ax

int nbPas=ax\*50; // Demi-case environ 50px ?

for (int i=0;i<nbPas;i++) {

digitalWrite(moteurX,LOW); // on allume le moteur ligne pendant nbPas

}

digitalWrite(moteurX,HIGH); // on éteint le moteur ligne

x=ax;

}

void avancer\_caseY(int ay) { // avancer/reculer jusqu'à la ligne ay

int nbPas=ay\*50; // Demi-case environ 50px ?

for (int i=0;i<nbPas;i++) {

digitalWrite(moteurY,LOW); // on allume le moteur colonne pendant nbPas

}

digitalWrite(moteurY,HIGH); // on éteint le moteur colonne

y=ay;

}

void revient\_origine() { // Faire revenir le bras à l'origine

avancer\_caseX(-x);

avancer\_caseY(-y);

x=0;

y=0;

}

void dead() {

// gérer ici l'endroit où sont envoyés les pièces mortes

jeu[px][py]=0; // rend la case de la pièce morte vide

// Déplace la pièce dans la zone morte

digitalWrite(aimant,LOW); // active l'aimant

aller\_case(-20,-20); // les nombres négatifs représentent la zone morte

// il faut modifier cette partie pour que les pièces mortes soient côte à côte

digitalWrite(aimant,HIGH); // désactive l'aimant

}

void deplacer\_piece(int i, int j) {

if (already\_taken==true) {

aller\_case(i,j);

jeu[i][j].dead();

}

digitalWrite(aimant,LOW); // active l'aimant pour attraper la pièce

avancer\_caseX(i); // se déplace sur la ligne voulue (avec la pièce)

avancer\_caseY(j); // se déplace sur la colonne voulue (avec la pièce)

digitalWrite(aimant,HIGH); // désactive l'aimant pour lacher la pièce

int piece=jeu[px][py]; // Conserve la valeur de la pièce à déplacer

jeu[px][py]=0; // Rend la case de la pièce que l'on déplace vide

jeu[i][j]=piece; // Donne la valeur de la pièce déplacée à la case d'arrivée

}

/\*

boolean already\_taken(int i, int j) {

if (jeu[i][j]==0){

return false;

}

return true;

}

\*/

Une partie du programme sera à modifier plus tard lorsque les moteurs seront définitivement codés afin de savoir combien de pas il faut effectuer avec le moteur afin de se déplacer d’une seule case.