Wouter Vandorpe
Weststraat 138 A 203
9940 Sleidinge
Belgium
Tel: +32478207902
Tel: +32492431139
Email: zolly.fit@hotmail.com
Email: zolly.wvd@gmail.com

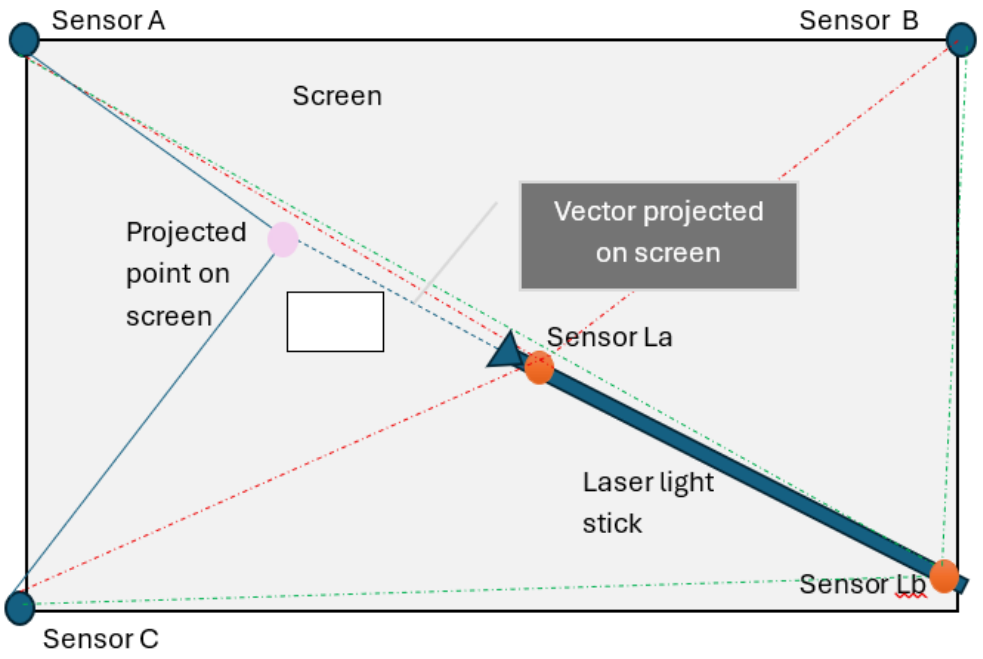
# Virtual Air Plot

## 1. Introduction

The possibility to write on a screen with a moving object in mid air.
We will use a laser to do so with some sensors attached to it (front and end).
The laser only is for visual to see where we write. With software attached on the sensors it is possible to plot the movement on the screen.

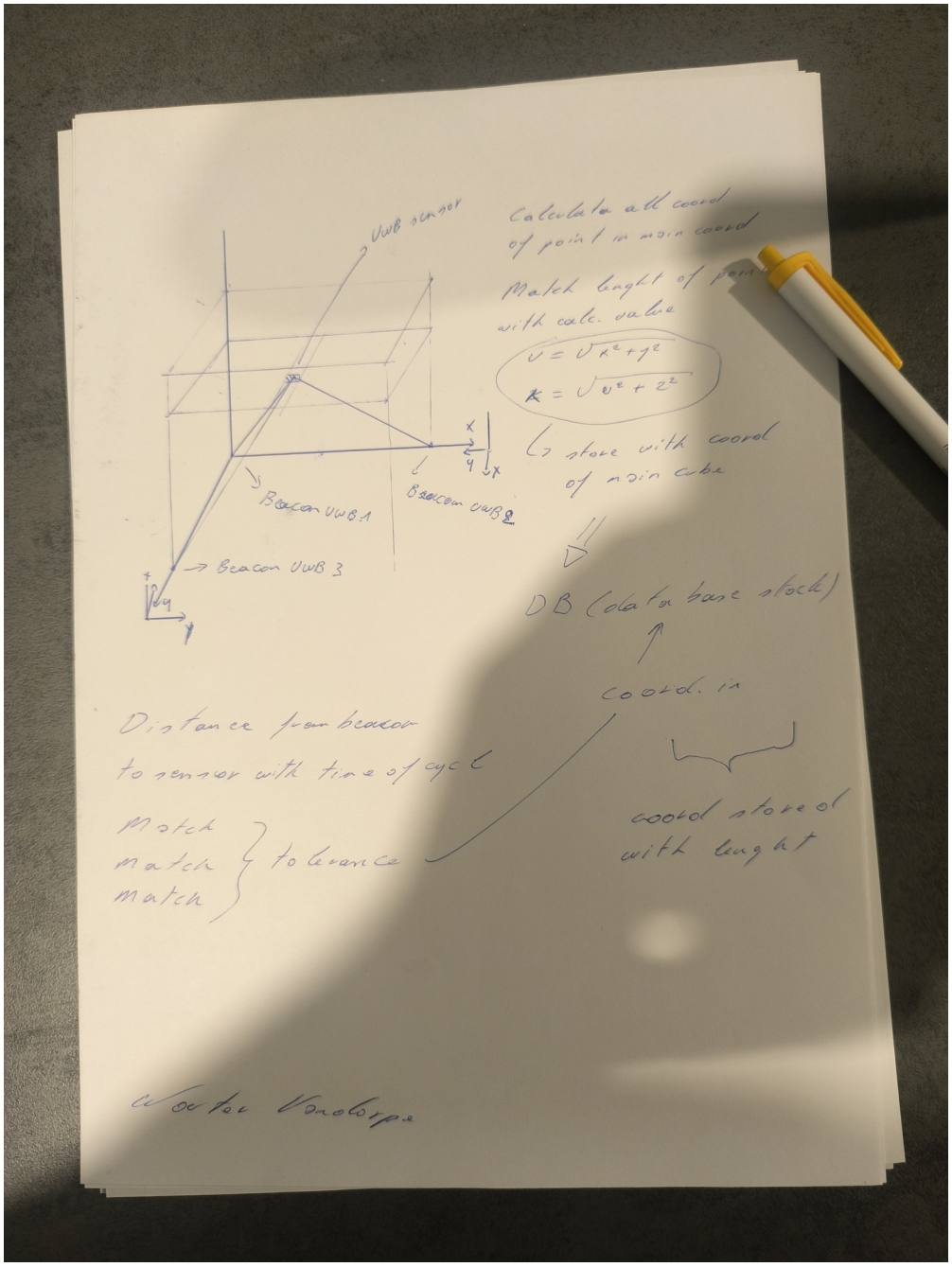
In our case we use 3 beacons to define the surface we can write. They are the corners of the surface with known coordinates.

## 2. Schematics




We got the projected point in pink on the screen. This comes form the light of the laser with the vector on the surface. The Laser light stick (LLS) has 2 UWB sensors inside. The laser projects the laser point on the screen. The coordinates of the laser point can be calculated with the sensors. Distance from sensor La to Sensor A,B,C and distance from sensor Lb to sensor A,B,C. Here we can get the coordinates of this points which makes it possible to calculate the vector point on the screen.

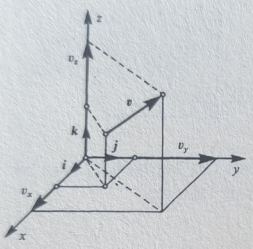
## 3. Calculation



Het coördinatenstelsel moet verder een rechts stelsel zijn, d.w.z. wordt de positieve $x$-as in het $x,y$-vlak in tegenwijzerrichting naar de positieve $y$-as gedraaid en tegelijkertijd langs de positieve $z$-as verplaatst, dan moet een rechtse schroef ontstaan.
De $x,y$-, $y,z$- en $z,x$-vlakken heten de grondvlakken van het coördinatenstelsel.

### b. VOORSTELLING VAN EEN VECTOR IN EEN COÖRDINATENSTELSEL

Wordt een vector $\boldsymbol{v}$ op de assen geprojecteerd, dan ontstaan drie vectoren $\boldsymbol{V}_x$, $\boldsymbol{V}_y$ en $\boldsymbol{V}_z$, de componenten van $\boldsymbol{V}$.




$\boldsymbol{V} = \boldsymbol{V}_x + \boldsymbol{V}_y + \boldsymbol{V}_z$ (componentenvoorstelling)

Als de lengte van de componenten wordt aangegeven met $v_x$, $v_y$ en $v_z$ dan is $\boldsymbol{v}_x = v_x\boldsymbol{i}$, $\boldsymbol{v}_y = v_y\boldsymbol{j}$ en $\boldsymbol{v}_z = v_z\boldsymbol{k}$ en geldt:

$\boldsymbol{v} = v_x\boldsymbol{i} + v_y\boldsymbol{j} \pm v_z\boldsymbol{k}$ (basisvoorstelling)

De getallen $v_x$, $v_y$ en $v_z$ worden de vectorcoördinaten van $\boldsymbol{v}$ genoemd.

Bij elke vector $\boldsymbol{v}$ hoort een geordend drietal $v_x$, $v_y$, $v_z$, dat anderzijds in bepaalde volgorde een vector eenduidig bepaald.
Ter verkorting van de schrijfwijze wordt veelal van een geordend drietal gebruik gemaakt:

$$\boldsymbol{v} = (v_x; v_y; v_z)$$

of:

$$\boldsymbol{v} = \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} \text{ (blokvoorstelling)}$$

Bijzondere gevallen:

1. De basisvectoren:

$$\boldsymbol{i} = (1;0;0) = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}; \boldsymbol{j} = (0;0;1) = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \text{ en } \boldsymbol{k} = (0;0;1) = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$



Example of the calculation where we get the matching coordinates of the measured values.

```
import math
#####################################
#Fill in the classname of the object#
#####################################
class clInterfaceScript:
    #####################################
    #Constructor class (stay's this way)#
    #####################################
    def __init__(self):
        print ('clInterfaceScript::__init__->start')
```

```
    self.Tracker_CoordX=[]
    self.Tracker_Length_X=[]
    self.Tracker_CoordY=[]
    self.Tracker_Length_Y=[]
    self.Tracker_CoordZ=[]
    self.Tracker_Length_Z=[]

    try:
        print ('clInterfaceScript::__init__->start')
    except:
        traceback.print_exc()
####################################
#Destructor class (stay's this way)#
####################################
def __del__(self):
    print ('clInterfaceScript::__del__->start')

###########
#Functions#
###########
def calculate_all(self,x_dimension,y_dimension,z_dimension):
    print('calculate_all')
    i = 0
    j = 0
    z = 0

    while i < x_dimension:
        Tracker_Length_2 = []
        while j < y_dimension:
            Tracker_Length_3 = []
            while z < z_dimension:
                self.Tracker_CoordX.append([i,j,z])
                v = math.sqrt((i*i)+(j*j))
                p = math.sqrt((v*v)+(z*z))
                Tracker_Length_3.append(p)
                print (p)
                z = z + 1
            z = 0
            j = j + 1
            Tracker_Length_2.append(Tracker_Length_3)
        j = 0
```

```
  i = i + 1
  self.Tracker_Length_X.append(Tracker_Length_2)


          #Calculate the second trio
i=0
j=0
z=0

while i < x_dimension:
  Tracker_Length_2 = []
  while j < y_dimension:
     Tracker_Length_3 = []
     while z < z_dimension:
        self.Tracker_CoordY.append([i,j,z])
        v = math.sqrt((i*i)+(j*j))
        p = math.sqrt((v*v)+(z*z))
        Tracker_Length_3.append(p)
        print (p)
        z = z + 1
     z = 0
     j = j + 1
     Tracker_Length_2.append(Tracker_Length_3)
  j = 0
  i = i + 1
  self.Tracker_Length_Y.append(Tracker_Length_2)

     #Calculate the third trio
i=0
j=0
z=0


while i < x_dimension:
  Tracker_Length_2 = []
  while j < y_dimension:
     Tracker_Length_3 = []
     while z < z_dimension:
        self.Tracker_CoordZ.append([i,j,z])
        v = math.sqrt((i*i)+(j*j))
        p = math.sqrt((v*v)+(z*z))
```

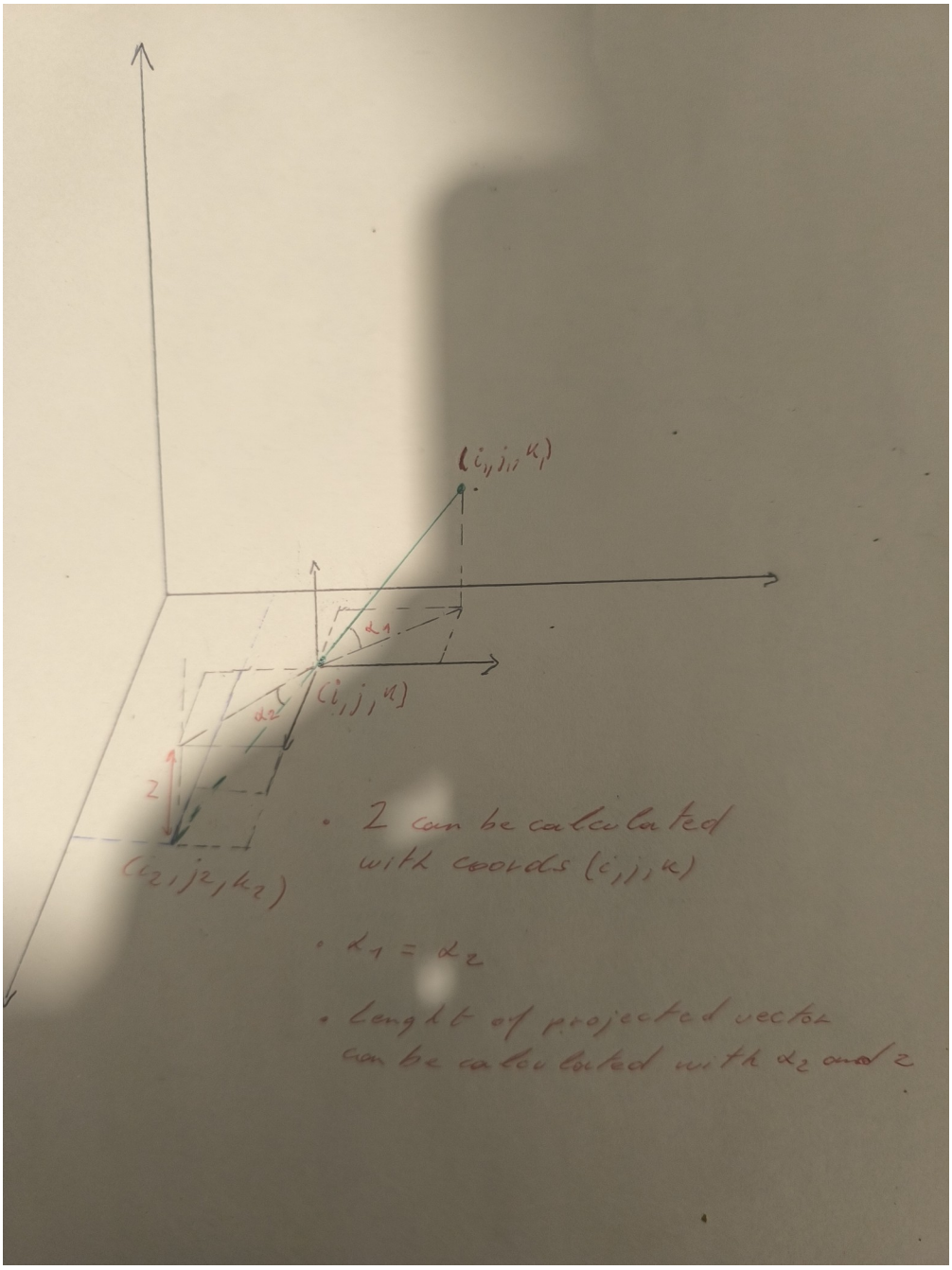
```
                Tracker_Length_3.append(p)
                print (p)
                z = z + 1
            z = 0
            j = j + 1
            Tracker_Length_2.append(Tracker_Length_3)
        j = 0
        i = i + 1
        self.Tracker_Length_Z.append(Tracker_Length_2)

    def bestfit(self,trio_1,trio_2,trio_3,dim1,dim2,dim3):
        print('START bestfit')
        tol = 1

        element_1=[]
        element_1_Y=[]
        BlockListNumberX=[]
        counter_X_L1 = 0
        print(len(self.Tracker_Length_X))
        for element_1 in self.Tracker_Length_X:
            counter_X_L2 = 0
            for element_1_Y in element_1:
                counter_X_L3 = 0
                for element_1_Z in element_1_Y:
                    if trio_1 < (element_1_Z + tol) and trio_1 > (element_1_Z - tol):
                        for elements_coord in self.Tracker_CoordX:
                            if (elements_coord[0] == counter_X_L1) and (elements_coord[1] ==
counter_X_L2) and (elements_coord[2] == counter_X_L3):
                                BlockListNumberX.append(elements_coord)
                                print ('--------------------')
                                print (elements_coord)
                                print (element_1_Z)
                                print (len(BlockListNumberX))
                                print ('--------------------')
                    counter_X_L3 = counter_X_L3 + 1
                counter_X_L2 = counter_X_L2 + 1
            counter_X_L1 = counter_X_L1 + 1

        BlockListNumberY=[]
        element_1=[]
        element_1_Y=[]
```

```
        counter_Y_L1 = 0
        print(len(self.Tracker_Length_Y))
        for element_1 in self.Tracker_Length_Y:
            counter_Y_L2 = 0
            for element_1_Y in element_1:
                counter_Y_L3 = 0
                for element_1_Z in element_1_Y:
                    if trio_2 < (element_1_Z + tol) and trio_2 > (element_1_Z - tol):
                        for elements_coord in self.Tracker_CoordY:
                            if elements_coord[0] == counter_Y_L1 and elements_coord[1] ==
counter_Y_L2 and elements_coord[2] == counter_Y_L3:
                                BlockListNumberY.append(elements_coord)
                                print ('--------------------')
                                print (elements_coord)
                                print (element_1_Z)
                                print (len(BlockListNumberY))
                                print ('--------------------')
                    counter_Y_L3 = counter_Y_L3 + 1
                counter_Y_L2 = counter_Y_L2 + 1
            counter_Y_L1 = counter_Y_L1 + 1

        BlockListNumberZ=[]
        element_1_Y=[]
        counter_Z_L1 = 0
        print(len(self.Tracker_Length_Z))
        for element_1 in self.Tracker_Length_Z:
            counter_Z_L2 = 0
            for element_1_Y in element_1:
                counter_Z_L3 = 0
                for element_1_Z in element_1_Y:
                    if trio_3 < (element_1_Z + tol) and trio_3 > (element_1_Z - tol):
                        for elements_coord in self.Tracker_CoordZ:
                            if elements_coord[0] == counter_Z_L1 and elements_coord[1] ==
counter_Z_L2 and elements_coord[2] == counter_Z_L3:
                                BlockListNumberZ.append(elements_coord)
                                print ('--------------------')
                                print (elements_coord)
                                print (element_1_Z)
                                print (len(BlockListNumberZ))
                                print ('--------------------')
                    counter_Z_L3 = counter_Z_L3 + 1
```

```
            counter_Z_L2 = counter_Z_L2 + 1
        counter_Z_L1 = counter_Z_L1 + 1

    blockTolerance=2
    print ('Continue')
    print ("Len of BlockListNumberX %s",(len(BlockListNumberX)))
    print ("Len of BlockListNumberY %s",(len(BlockListNumberY)))
    print ("Len of BlockListNumberZ %s",(len(BlockListNumberZ)))
    i = 0
    j = 0
    k = 0
    breakFunct = 0
    while i < len(BlockListNumberX):
        j = 0
        while j < len(BlockListNumberY):
            k = 0
            while k < len(BlockListNumberZ):
                elements_coord_1 = BlockListNumberX[i]
                elements_coord_2 = BlockListNumberY[j]
                elements_coord_3 = BlockListNumberZ[k]
                print ("Element: %s %s %s",
(elements_coord_1,elements_coord_2,elements_coord_3))

                if (elements_coord_1[0] == ((dim2 - elements_coord_2[1])) and
                    elements_coord_1[0] == ((elements_coord_3[1])) and
                    elements_coord_1[1] == (elements_coord_2[0]) and
                    elements_coord_1[1] == ((dim3 - elements_coord_3[0])) and
                    elements_coord_1[2] < (elements_coord_2[2] + blockTolerance) and
                    elements_coord_1[2] > (elements_coord_2[2] - blockTolerance) and
                    elements_coord_1[2] < (elements_coord_3[2] + blockTolerance) and
                    elements_coord_1[2] > (elements_coord_3[2] - blockTolerance)):
                        print ('---------------')
                        print (elements_coord_1)
                        print (elements_coord_2)
                        print (elements_coord_3)
                        return [elements_coord_1]

                k = k + 1
            j = j + 1
        i = i + 1
    print('STOP bestfit')
```

```
########################################
#Created for manual testing of the methods on the class  #
########################################################
def main():
    meInterfaceScript = clInterfaceScript()
    #Test the function manually for a surface of 10 hight and with and length
    meInterfaceScript.calculate_all(10,10,10)
    #Measured values are 7 and get the best fit coordinates
    bestFitValue = meInterfaceScript.bestfit(7,7,7,10,10,10)

if __name__ == "__main__":
    main()
```

Once the coordinates are located into the cube we can calculate the vector to the surface.

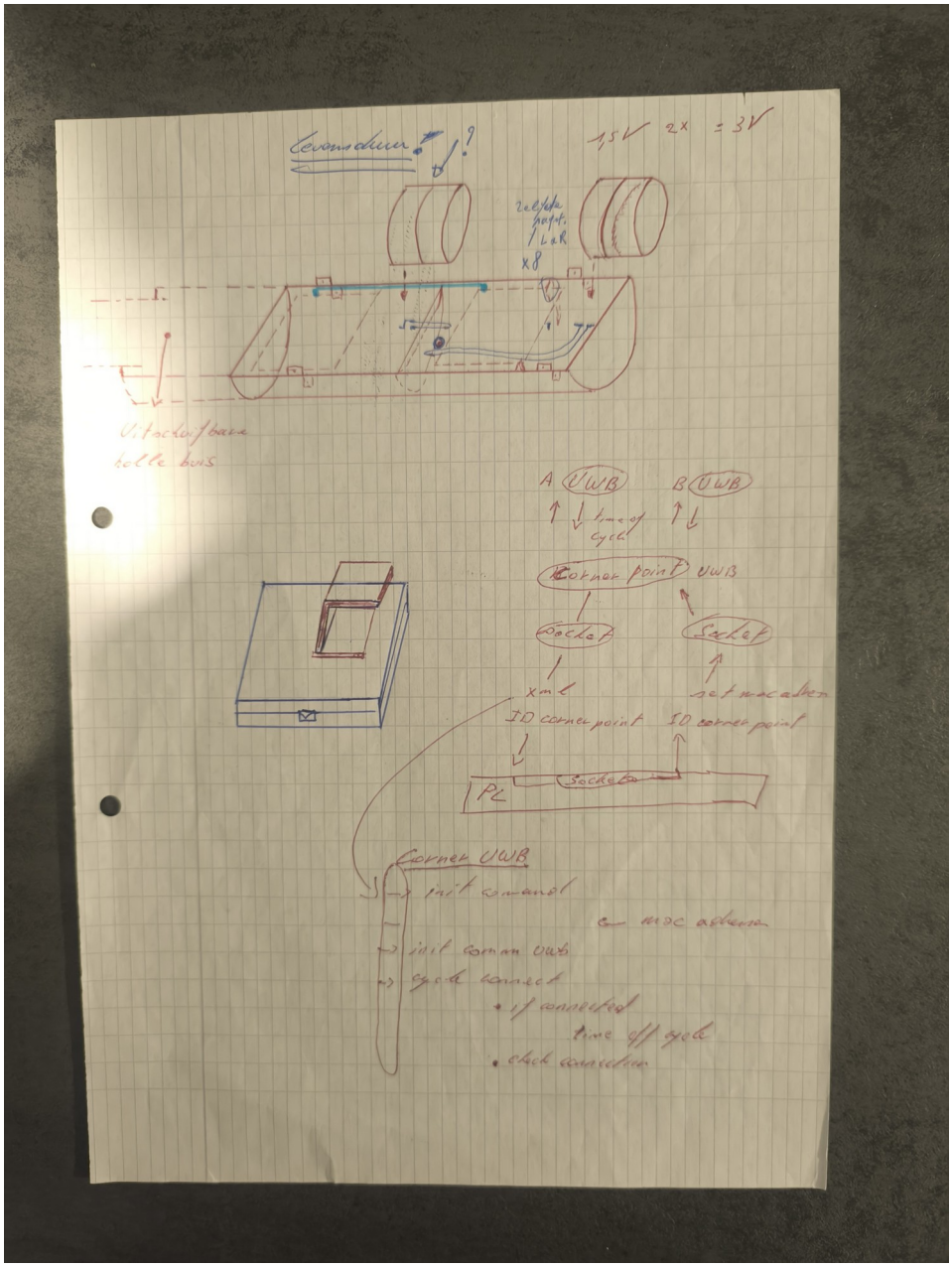



**4. Technical**

To run the script and define the matching point it takes to much time to do this en every measuring cycle. Therefore at calibration time we can fill the database with the probable distances for all points according mm, cm, or smaller. Once the measure values come in we can query the database with a cycle of approx 0.2 msec and get the coordinates to calculate where point is located on the surface.

**5. The lasermold**

I have been drawing a small model for the mold but this is just a pointer.
Original paper is this one.



**6. Material**

For building the prototype you can use the following material.

**2 sensors:**

**UWB Indoor Positioning Module Tag + Base Station Ultra-wideband Short-range High-precision Ranging Module BU01**

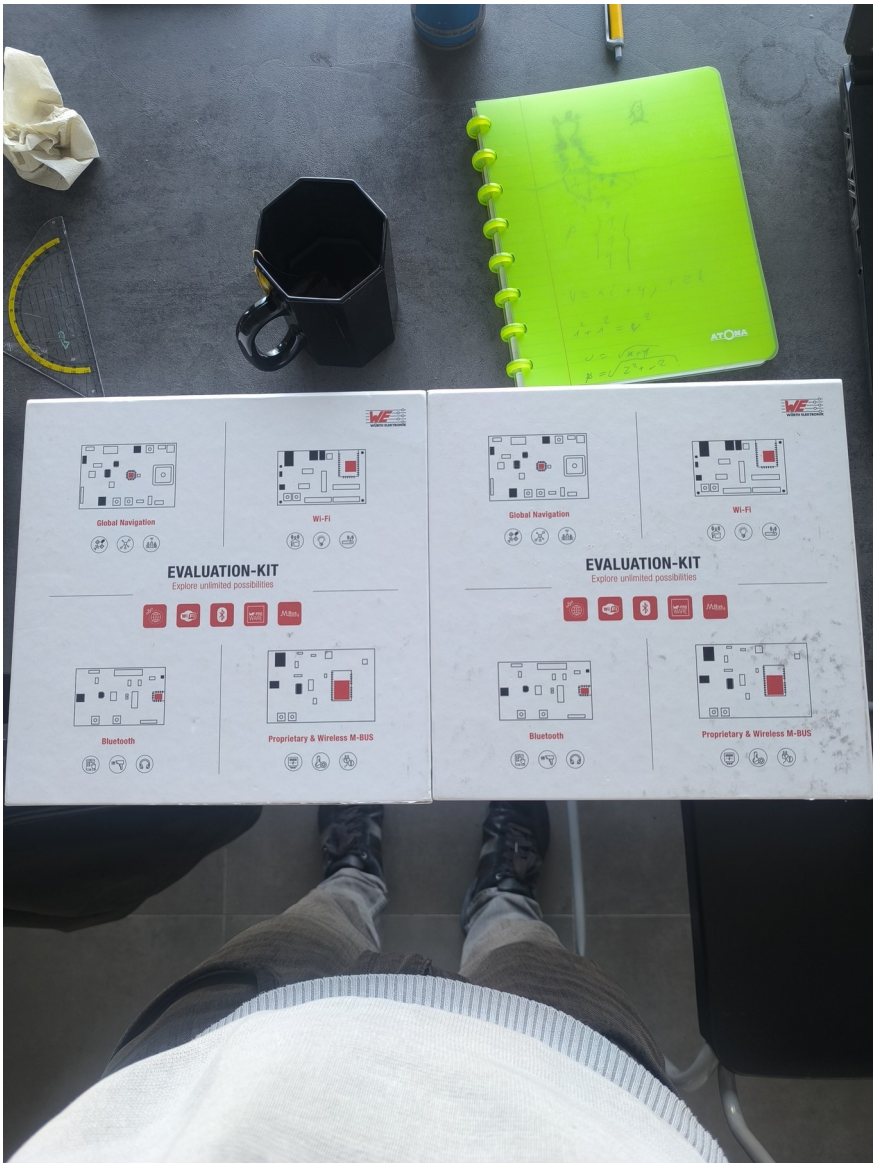
**https://nl.aliexpress.com/item/1005001605414048.html?spm=a2g0o.productlist.main.13.35f24ae5KnoPAD&algo_pvid=806e09fd-16ce-4921-afdf-5e9dfdce3d4c&algo_exp_id=806e09fd-16ce-4921-afdf-5e9dfdce3d4c-10&pdp_ext_f=%7B%22order%22%3A%2235%22%2C%22eval%22%3A%221%22%7D&pdp_npi=6%40dis%21EUR%211.95%211.46%21%21%212.23%211.67%21%402103864c17561416654132605efa01%21112000046030523389%21sea%21BE%212643826211%21X%211%210%21n_tag%3A-29919%3Bd%3Ae4f0e6f9%3Bm03_new_user%3A-29895&curPageLogUid=3OBr300XeGr1&utparam-url=scene%3Asearch%7Cquery_from%3A%7Cx_object_id%3A1005001605414048%7C_p_origin_prod%3A**

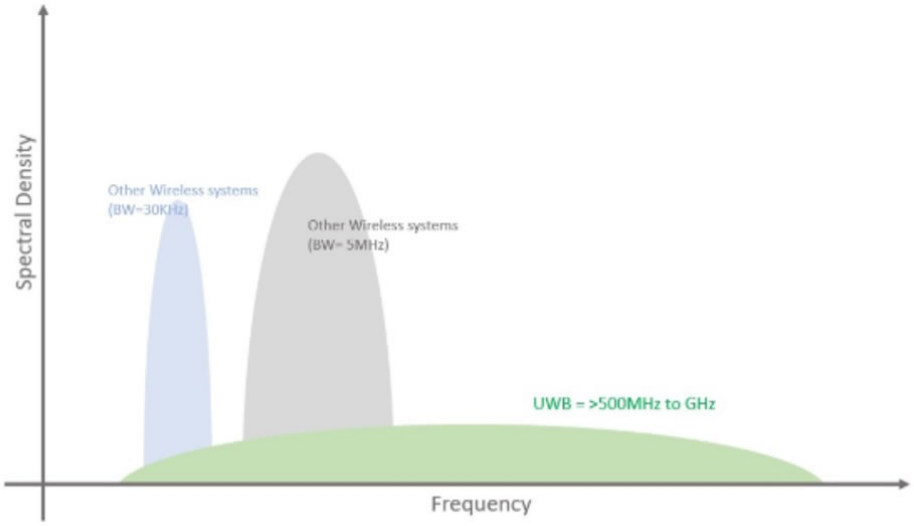
**3 Arduino's**
**ESP 32 UWB Pro Development board Wireless module**
**https://www.alibaba.com/product-detail/ESP-32-UWB-Pro-Development-board_1601201768432.html**

**DW1000 and ESP32**
**https://forum.qorvo.com/t/esp32-dw3000-ds-twr-sample-code/14367/9**

**7. Accuracy**

The accuracy of UWB is not that good for the moment, therefore I suggest to modify the development boards of Würth with bluetooth (mm) with UWB chips. This could make the accuracy a lot closer to the microns.


Global Navigation
Wi-Fi
EVALUATION-KIT
Explore unlimited possibilities
Bluetooth
Proprietary & Wireless M-BUS
Global Navigation
Wi-Fi
EVALUATION-KIT
Explore unlimited possibilities
Bluetooth
Proprietary & Wireless M-BUS




**8. Other applications**

- CNC machine positioning (when uwb on development boards are accurate)
- Location in holograms: opening folders and controlling hollogram computers
- Rotating at snap point objects in holograms
- Robot positioning without step motor
- ...