2UV project

Wouter Vandorpe

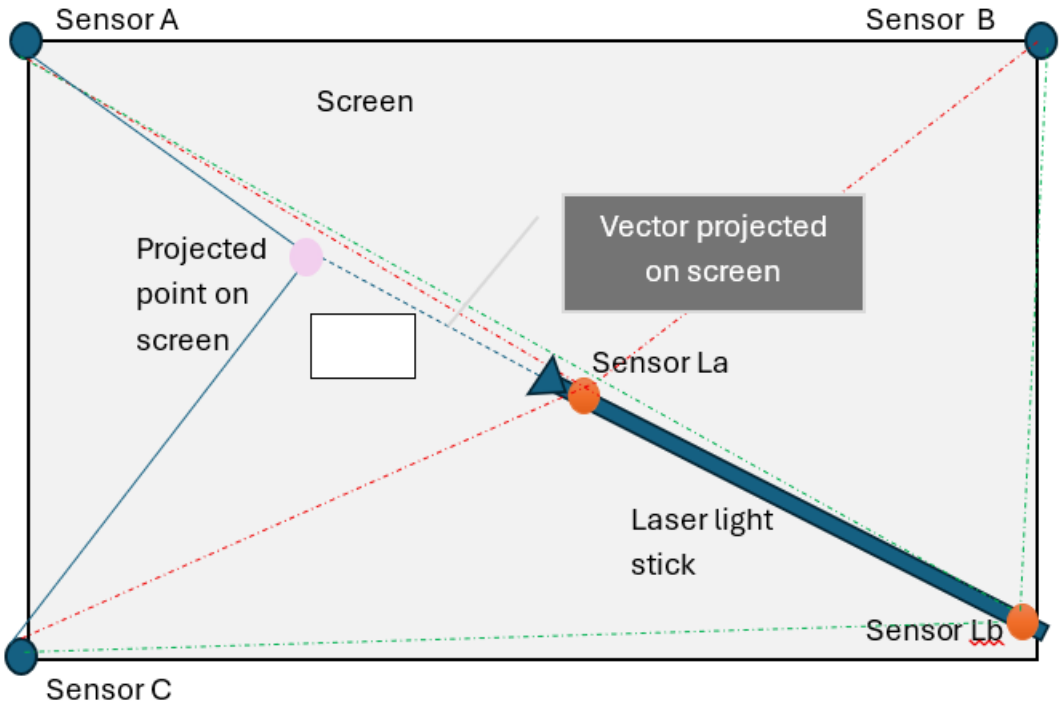
Weststraat 138 A203
Sleidinge 9940

Virtualwrite

1) Introduction:

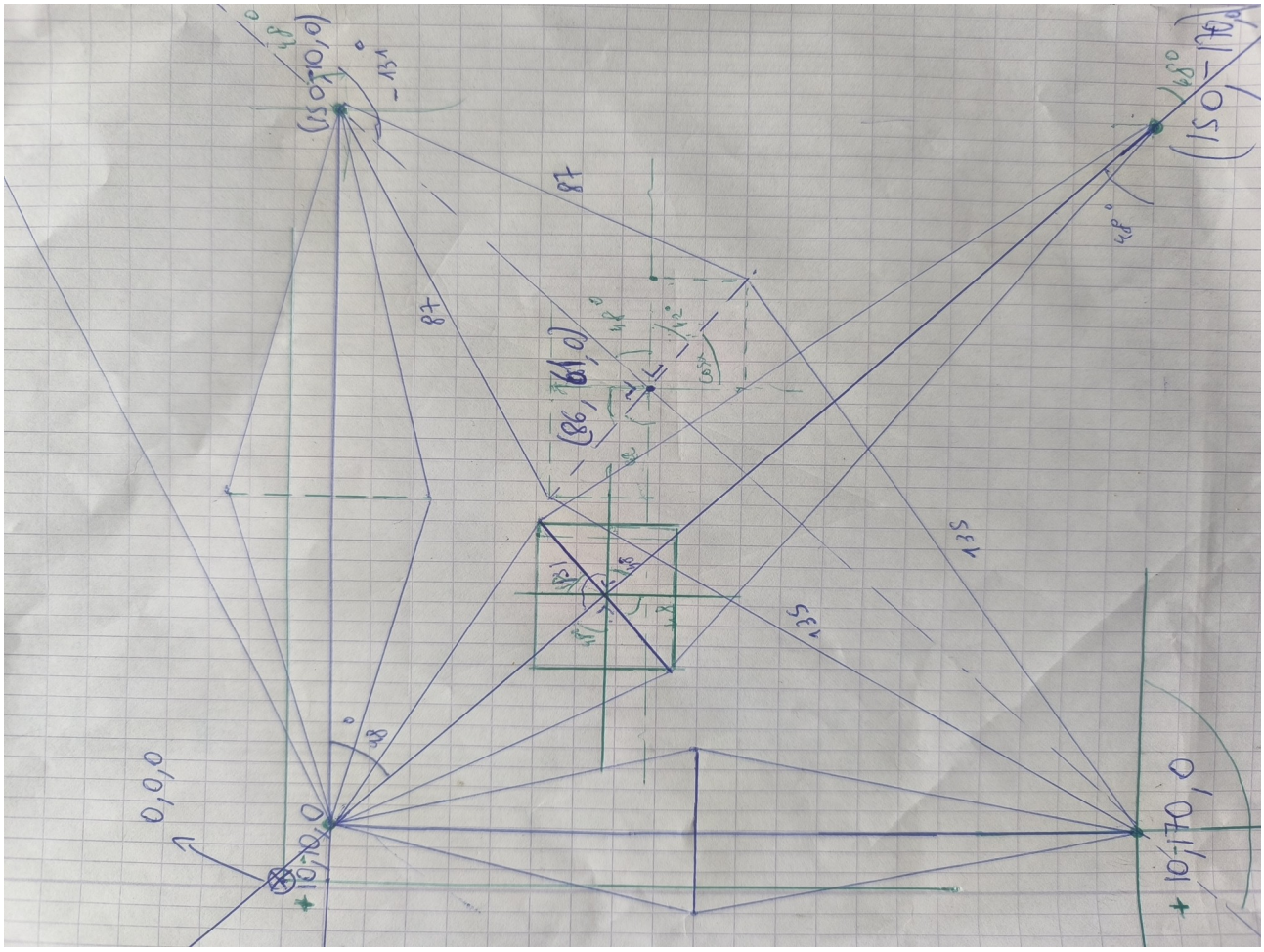
The possibility to write on a screen with a moving object in the air. We will use a laser to do this. In this product we will use the technology of UWB for location and calculation of the variables who are displayed. In this document you will find the maths used and the equipment. Following with the schematics who represent the formation.

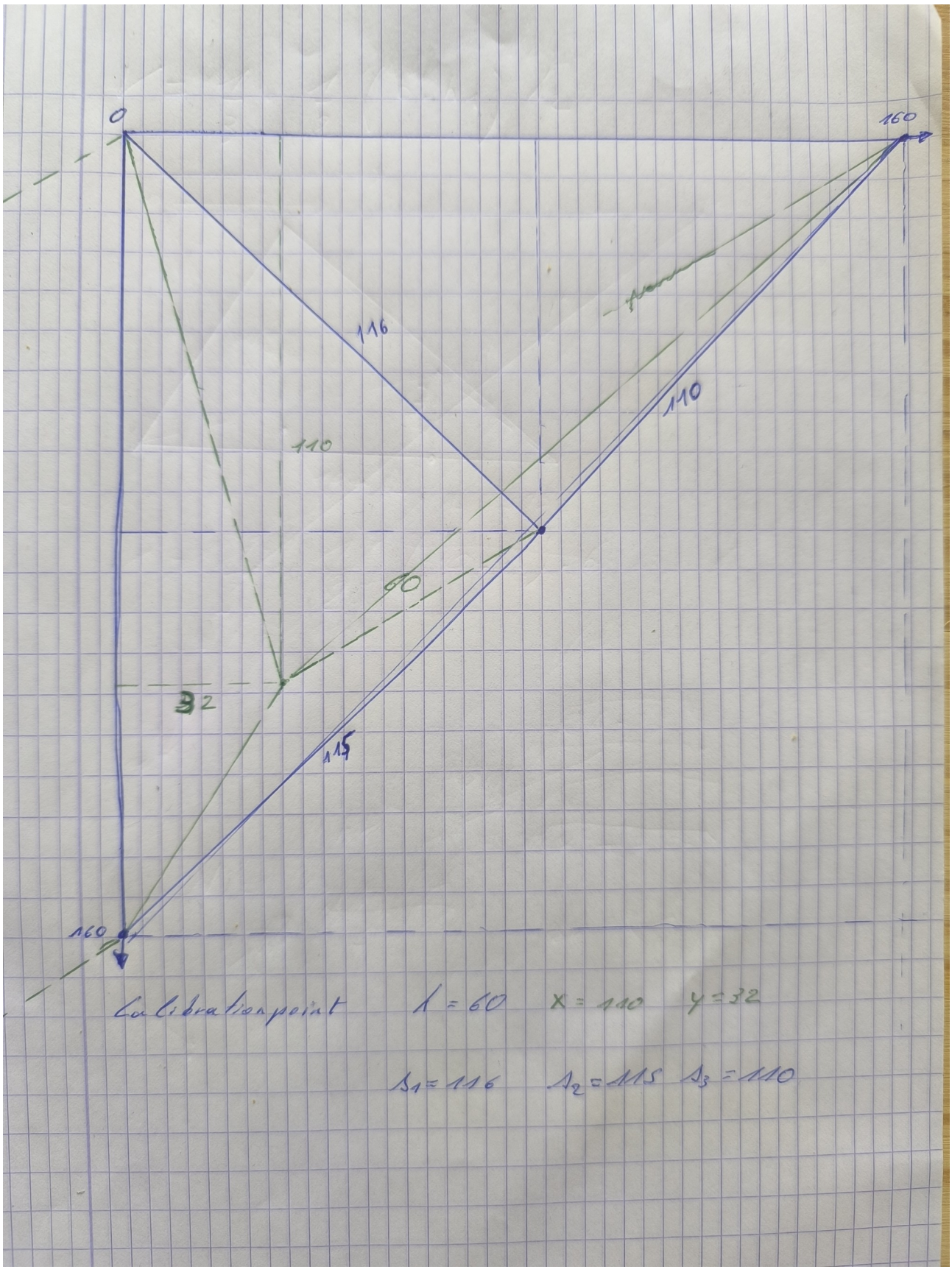
2) Schematics:



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. Which can calculate the vector on the screen.

3) Maths





Calibration point $\lambda = 60$ $X = 110$ $Y = 32$

$\lambda_1 = 116$ $\lambda_2 = 115$ $\lambda_3 = 110$

4) Material

- 5 UWB sensors, 3 sensors for the moment Arduino UWB with Wifi, over this Wifi we can implement software with sockets to transfer the data to the server.
- Calibration rule
- Laser pointer with a long stick
- Software which captures the UWB points.

5) Workflow

First we do the calibration of the 2 points. This means we set the height at a certain position with a rule. Making sure the laser is straight.
In an angle of 90 degrees on the surface.
Once the calibration is made the stick can be used to write on the screen through receiving the coordinates.
Coordinates are captured and displayed on the screen. This with only the projected vector point.

6) UWB communication with workstation

The communication here is based on a rssi signal from wifi so same must be implemented for UWB. Here we choose the time of cycle of a package to calculate the distance of the sensor point.

Example: Tracker device with rssi signal

```
#include <Arduino.h>
#include <WiFi.h>

#include <WiFiClient.h>
#include <WiFiServer.h>

const char* paSsid = "2UVTracker_0002";
const char* paPassword = "password213";


IPAddress local_IP(192,168,4,10);
IPAddress gateway(192,168,4,1);
IPAddress subnet(255,255,255,0);

void setup() {
  Serial.begin(115200);

  WiFi.mode(WIFI_AP); //Set mode of the Wifi
  delay(250);
  Serial.println(WiFi.softAPConfig(local_IP, gateway, subnet) ? "Ready" : "Failed!");
  delay(250);
  Serial.println(WiFi.softAP(paSsid, paPassword) ? "Ready" : "Failed!");
  delay(1000);
  // Use your network credentials as SSID and password
  IPAddress IP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(IP);
```

```
  //server.begin();
}
void loop() {
  delay(1000);
  Serial.println("Server running");
  Serial.print("NetworkID:");
  String current_ssid = WiFi.SSID();
  Serial.println(current_ssid);
  /*
  WiFiClient client = server.available();
  if (client) {
    Serial.println("New client");
    String currentLine = "";
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
        if (c == '\n') {
          if (currentLine.length() == 0) {
            // Send the HTTP response
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println();
            client.print("<html><body><h1>Hello from ESP32!</h1></body></html>");
            client.println();
            break; // Break out of the loop when the response is sent
          } else {
            currentLine = "";
          }
        } else if (c != '\r') {
          currentLine += c;
        }
      }
    }
    client.stop();
    Serial.println("Client disconnected");
  }
  */
}
```

Beacon with rssi signal

```
#include <WiFi.h>

const char *ssid = "_LinksysSetupCBC";
const char *password = "password";

String glObjectsName[50];
int glObjectsRSSI[50];
int glObjectsInField[50];
```

```
int glRunCount = 0;
String glDeviceName = "2UVBEACON_001_Z";


void scan_Networks()
{
    int numberOfNetworks = WiFi.scanNetworks();
    for (int i = 0; i < numberOfNetworks; i++)
    {

      Serial.print(WiFi.SSID(i));
      Serial.print(" (");
      Serial.print(WiFi.RSSI(i));
      Serial.print(" dBm)");
      delay(10);

      bool addToList = true;
      for (int j = 0; j < 50; j++)
      {
        if (glObjectsName[j] == String(WiFi.SSID(i)))
        {
          glObjectsRSSI[j] = WiFi.RSSI(i);
          glObjectsInField[j] = glRunCount;
          addToList = false;
        }
      }
      if (addToList)
      {
        for (int j = 0; j < 50; j++){
          if (glObjectsName[j] == String(""))
          {
            glObjectsRSSI[j] = WiFi.RSSI(i);
            glObjectsName[j] = String(WiFi.SSID(i));
            glObjectsInField[j] = glRunCount;
            break;
          }
        }
      }

    }
}

void setup() {
  Serial.begin(115200);
  delay(10);

  Serial.println();
  Serial.print("[WiFi] Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);
  // Auto reconnect is set true as default
```

```
  // To set auto connect off, use the following function
  WiFi.setAutoReconnect(true);
}

void loop() {

  // put your main code here, to run repeatedly:
  Serial.println(F("Scanning will start"));
  scan_Networks();


  // Reconnect if disconnected the WIFI
  if (WiFi.status() == WL_DISCONNECTED)
  {
    WiFi.begin(ssid, password);
    if (WiFi.status() == WL_CONNECTED)
    {
      Serial.println("Local IP:" + WiFi.localIP());
    }
  }

  //Construct the string
  glRunCount = glRunCount + 1;
  if (glRunCount == 50000)
  {
    glRunCount = 1;
  }
  for (int i = 0; i < 50; i++)
  {
    if (glObjectsInField[i] < (glRunCount - 20))
    {
        glObjectsRSSI[i] = 0;
        glObjectsName[i] = String("");
        glObjectsInField[i] = 0;
    }
  }
  String loMessage = "<root><master>" + glDeviceName + "</master><devices>";
  for (int i = 0; i < 50; i++)
  {
    if (glObjectsName[i] != String(""))
    {
        if (  String(glObjectsName[i][0]) == "2" &&
              String(glObjectsName[i][1]) == "U" &&
              String(glObjectsName[i][2]) == "V" &&
              String(glObjectsName[i][3]) == "T")
                loMessage = loMessage + "<device><name>" + glObjectsName[i] + "</name><rssi>" +
String(glObjectsRSSI[i]) + "</rssi><time>" + String(glObjectsInField[i]) + "</time></device>";
    }
  }
  loMessage = loMessage + "</devices></root>";

  //Send the string
```

```
  // Use NetworkClient class to create TCP connections
  NetworkClient client;
  const uint16_t port = 4000;
  const char *host = "192.168.1.231";

  if (!client.connect(host, port)) {
    Serial.println("Connection failed.");
    Serial.println("Waiting 5 seconds before retrying...");
    delay(1000);
    return;
  }
  client.print(loMessage);

  //delay(1000);

}
```

Receiver script in python

```
from bluepy.btle import Scanner, DefaultDelegate, BTLEException, Peripheral, UUID
import socket
import xml.etree.ElementTree as ET
import datetime
from threading import Thread
import threading
import math




class clTracker:
    def __init__(self, paName):
        self.meName = paName
        self.meRssiToBeacon = []

class clPoint:
    def __init__(self, x: float, y: float, z: float = 0.0):
        self.x = x
        self.y = y
        self.z = z

    def distance_to(self, other: 'Point') -> float:
        return math.sqrt((self.x - other.x)**2 + (self.y - other.y)**2 + (self.z - other.z)**2)

class clBeacon:
    def __init__(self, paName, paPoint: clPoint):
```

```
        self.meName = paName
        self.mePoint = paPoint




####################################
#Fill in the classname of the object#
####################################
class clSocketBoxSimulator:
    ####################################
    #Constructor class (stay's this way)#
    ####################################
    def __init__(self):
            print ('clSocketBoxSimulator::__init__->start')
            self.commandID = ""
            self.timerSendFile = 0
            self.timerRecieveFile = 0
            self.timerRunFile = 0
            self.paused = 0
            self.error = ''
            self.TrackerIDs = []
            self.counter = 50

            #Set up the interface communication
            try:
                self.commandID = ""
                self.timerSendFile = 0
                self.timerRecieveFile = 0
                self.timerRunFile = 0
                self.error = ''
            except:
                traceback.print_exc()
    ####################################
    #Destructor class (stay's this way)#
    ####################################
    def __del__(self):
            print ('clSocketBoxSimulator::__del__->start')

    ############
    #Functions#
    ############
    def run_server(self):
        # create a socket object
        server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

        server_ip = "192.168.1.231"
        port = 4001

        # bind the socket to a specific address and port
        server.bind((server_ip, port))
        #server.close()
```

```
        # listen for incoming connections
        server.listen(10)
        print('Listening on ' + server_ip + ':' + str(port))

        try:
            while True:
                client_socket, addr = server.accept()
                print('Accepted connection from ' + str(addr))
                client_handler = Thread(target=self.handle_client, args=(client_socket,))
                client_handler.start()

                #client_scanner = Thread(target=self.handle_scan_for_ble,args=())
                #client_scanner.start()
                # close server socket
        except:
            server.close()

    def handle_client(self,client_socket):
        while True:
            try:
                message = client_socket.recv(1024).decode('utf-8')
                if not message:
                    print('Nothing received')
                else:
                    #print('Received: ' + message)
                    ########################## parse the input
#######################################
                    self.parseXML(message)
                    ########################## send the response
#######################################
                    #W0R0D0P0C0T5648779879
                    #Waiting
                    #Running
                    #DataTransfer
                    #Pauzed
                    #Connected
                    #Timestamp
                    returnState = ""

                    if self.timerRunFile > 0:
                        returnState = returnState + 'W0R1'
                    else:
                        returnState = returnState + 'W1R0'
                    if self.paused == 0:
                        returnState = returnState + 'P0'
                    else:
                        returnState = returnState + 'P1'
                    if self.timerSendFile == 0 and self.timerRecieveFile == 0:
                        returnState = returnState + 'D0'
                    else:
                        returnState = returnState + 'D1'
                    #Connected
```

```
                returnState = returnState + 'C1'
                loTimeStamp = datetime.datetime.now()
                returnState = returnState + 'T' + loTimeStamp.strftime("%f")

                loObjects = ''
                for i, loTracker in enumerate(self.TrackerIDs):
                    loObjects = loObjects + '<object name=\"' + str(loTracker[0]) + '\"><x value=\"' +
str(loTracker[1]) + '\"/><y value=\"' + str(loTracker[2]) + '\"/><z value=\"' + str(loTracker[3]) +
'\"/></object>'
                #<object name=\"Magazijn_1001\"><x value=\"30.5\"/><y value=\"30.5\"/><z
value=\"1.0\"/></object><object name=\"Employee_200\"><x value=\"60.5\"/><y
value=\"50.5\"/><z value=\"0.5\"/></object>
                returnString = ('<returnHardwareDevice><id>' + self.commandID + '</id><state>' +
returnState + '</state><datas>' + loObjects + '</datas><error>' + self.error +
'</error></returnHardwareDevice>')

                #print('Response: ' + returnString)
                #print('sendall start')
                client_socket.sendall(returnString.encode('utf-8'))
                #print('sendall not handeled')
            except:
                print('Exception')
                break
        client_socket.close()

    def parseXML(self,request):
        #print('START parseXML(requestXML)')
        loRoot = ET.fromstring(request)
        loSelectedChild = None
        loSelectedChildren = []
        for loChild in loRoot:
            #print(loChild.tag, loChild.attrib)
            if loChild.tag == 'id':
                self.commandID = loChild.text

            loPerform = loChild.get('do')
            if (loPerform == 'true'):
                loSelectedChild = loChild
                for loChildParams in loSelectedChild:
                    loSelectedChildren.append(loChildParams.text)
                break

        if loSelectedChild.tag == "connect":
            self.commandConnect(loSelectedChildren)
        elif loSelectedChild.tag == "disconnect":
            self.commandDisconnect(loSelectedChildren)
        elif loSelectedChild.tag == "state":
            self.commandState(loSelectedChildren)
        elif loSelectedChild.tag == "run":
            self.commandRun(loSelectedChildren)
        elif loSelectedChild.tag == "abort":
            self.commandAbort(loSelectedChildren)
```

```
      elif loSelectedChild.tag == "hold":
         self.commandHold(loSelectedChildren)
      elif loSelectedChild.tag == "continue":
         self.commandContinue(loSelectedChildren)
      elif loSelectedChild.tag == "sendFile":
         self.commandSendFile(loSelectedChildren)
      elif loSelectedChild.tag == "recieveFile":
         self.commandRecieveFile(loSelectedChildren)
      elif loSelectedChild.tag == "scriptCommand":
         self.commandScriptCommand(loSelectedChildren)
      elif loSelectedChild.tag == "optionalCommand":
         self.commandOptionalCommand(loSelectedChildren)
      #print('STOP parseXML(requestXML)')

   def commandConnect(self,paParams):
      print('START commandConnect(paParams')
      print('STOP commandConnect(paParams')

   def commandDisconnect(self,paParams):
      print('START commandDisconnect(paParams)')
      print('STOP commandDisconnect(paParams)')

   def commandState(self,paParams):
      #print('START commandState(paParams)')

      ##################################################
      #Moving of 2 objects into the field
      ##################################################
      self.counter = self.counter - 5
      if self.counter < 1:
         self.counter = 50

      #if self.counter >= 40:
      #    self.objectID = '<object name=\"Magazijn_1001\"><x value=\"70.5\"/><y
value=\"70.5\"/><z value=\"1.0\"/></object><object name=\"Employee_200\"><x
value=\"80.5\"/><y value=\"50.5\"/><z value=\"0.5\"/></object>'
      #elif self.counter >= 30:
      #    self.objectID = '<object name=\"Magazijn_1001\"><x value=\"60.5\"/><y
value=\"60.5\"/><z value=\"1.0\"/></object><object name=\"Employee_200\"><x
value=\"65.5\"/><y value=\"40.5\"/><z value=\"0.5\"/></object>'
      #elif self.counter >= 20:
      #    self.objectID = '<object name=\"Magazijn_1001\"><x value=\"40.5\"/><y
value=\"40.5\"/><z value=\"1.0\"/></object><object name=\"Employee_200\"><x
value=\"50.5\"/><y value=\"30.5\"/><z value=\"0.5\"/></object>'
      #elif self.counter >= 10:
      #    self.objectID = '<object name=\"Magazijn_1001\"><x value=\"30.5\"/><y
value=\"30.5\"/><z value=\"1.0\"/></object><object name=\"Employee_200\"><x
value=\"35.5\"/><y value=\"20.5\"/><z value=\"0.5\"/></object>'
      #elif self.counter >= 0:
      #    self.objectID = '<object name=\"Magazijn_1001\"><x value=\"15.5\"/><y
value=\"15.5\"/><z value=\"1.0\"/></object><object name=\"Employee_200\"><x
value=\"10.5\"/><y value=\"10.5\"/><z value=\"0.5\"/></object>'
```

```
    ##################################################

    if self.timerRecieveFile > 0:
        self.timerRecieveFile = self.timerRecieveFile - 1

    if self.timerRunFile > 0:
        self.timerRunFile = self.timerRunFile - 1

    if self.timerRunFile < 5:
        self.error = ''

    if self.timerSendFile > 0:
        self.timerSendFile = self.timerSendFile -1

    #print('STOP commandState(paParams)')

def commandRun(self,paParams):
    print('START commandRun(paParams)')
    self.timerRunFile = 10
    self.error = 'approx 10 seconds running'
    print('STOP commandRun(paParams)')

def commandAbort(self,paParams):
    print('START commandAbort(paParams)')
    print('STOP commandAbort(paParams)')

def commandHold(self,paParams):
    print('START commandHold(paParams)')
    self.paused = 1
    print('STOP commandHold(paParams)')

def commandContinue(self,paParams):
    print('START commandContinue(paParams)')
    self.paused = 0
    print('STOP commandContinue(paParams)')

def commandSendFile(self,paParams):
    print('START commandSendFile(paParams)')
    self.timerSendFile = 10
    print('STOP commandSendFile(paParams)')

def commandRecieveFile(self,paParams):
    print('START commandRecieveFile(paParams)')
    self.timerReceiveFile = 10
    print('STOP commandRecieveFile(paParams)')

def commandScriptCommand(self,paParams):
    print('START commandScriptCommand(paParams)')
    print('STOP commandScriptCommand(paParams)')

def commandOptionalCommand(self,paParams):
    print('START optionalCommand(paParams)')
```

```
        print('STOP optionalCommand(paParams)')

####################################
#Fill in the classname of the object#
####################################
class clSocketBoxBeaconServer:
    ####################################
    #Constructor class (stay's this way)#
    ####################################
    def __init__(self, paSocketBoxSimulator: clSocketBoxSimulator, paBeacons):
            print ('clSocketBoxBeaconServer::__init__->start')
            self.myMac = ""
            self.trackerX = []
            self.trackerY = []
            self.trackerZ = []
            self.meSocketBoxSimulator = paSocketBoxSimulator
            self.meBeacons = paBeacons
            self.meTrackers = []

    ####################################
    #Destructor class (stay's this way)#
    ####################################
    def __del__(self):
            print ('clSocketBoxBeaconServer::__del__->start')

    ############
    #Functions#
    ############
    def run_server(self):
        # create a socket object
        server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

        server_ip = "192.168.1.231"
        port = 4000

        # bind the socket to a specific address and port
        server.bind((server_ip, port))
        #server.close()
        # listen for incoming connections
        server.listen(10)
        print('Listening on ' + server_ip + ':' + str(port))

        try:
            while True:
                client_socket, addr = server.accept()
                print('Accepted connection from ' + str(addr))
                client_handler = Thread(target=self.handle_client, args=(client_socket,))
                client_handler.start()

                #client_scanner = Thread(target=self.handle_scan_for_ble,args=())
                #client_scanner.start()
                # close server socket
```

```
        except:
            server.close()

    def handle_client(self,client_socket):
        while True:
            try:
                message = client_socket.recv(1024).decode('utf-8')
                if not message:
                    print('Nothing received')
                else:
                    #print('Received: ' + message)
                    ########################## parse the input
########################################
                    self.parseXML(message)
                    self.calculatePoints()
                    print ('Received from Beacon:' + message)
                    client_socket.close()
                    break
            except:
                print('Exception')
                break
        client_socket.close()


    def parseXML(self,xmlStringForObject):
        try:
            loName = ""
            loDistanceToObject = ""

            #Parse xml from the beacon
            xml_string = xmlStringForObject
            print(xml_string)
            root = ET.fromstring(xml_string)
            #<root><master>" + glDeviceName + "</master><devices>somedevice</devices></root>

            #Find beacon name
            for master in root.iter(tag='master'):


                loBeaconFound = False
                for n, loBeacon in enumerate(self.meBeacons):

                    if loBeacon.meName == master.text:
                        print ('Beacon found: ' +  master.text)
                        loBeaconFound = True

                        #Find tracker name
                        for device in root.iter(tag = 'device'):
                            for loTrackerToFind in device.iter(tag = 'name'):

                                #Check Tracker already in list
                                loTrackerFound = False
```

```
                        for l, loTracker in enumerate(self.meTrackers):
                            if loTracker.meName == loTrackerToFind.text:
                                loTrackerFound = True

                                #Update rssi value too
                                for devRSSI in device.iter(tag = 'rssi'):
                                    loDistanceToObject = devRSSI.text

                                #Check if beacon alredy in the tracker
                                loBeaconInTrackerFound = False
                                for m, loObjectBeaconInTrackerAndRssi in
enumerate(loTracker.meRssiToBeacon):
                                    loObjectBeaconInTracker = loObjectBeaconInTrackerAndRssi[0]
                                    loObjectRssiInTracker = loObjectBeaconInTrackerAndRssi[1]
                                    if loObjectBeaconInTracker.meName == loBeacon.meName:
                                        loBeaconInTrackerFound = True
                                        loObjectBeaconInTrackerAndRssi[1] = loDistanceToObject

                                #Beacon not in tracker list add
                                if not loBeaconInTrackerFound:
                                    loTracker.meRssiToBeacon.append([loBeacon,loDistanceToObject])


                        #Tracker not in list so add to list
                        if not loTrackerFound:
                            #Add rssi value too
                            for devRSSI in device.iter(tag = 'rssi'):
                                loDistanceToObject = devRSSI.text
                            loTrackerNew = clTracker(loTrackerToFind.text)
                            loTrackerNew.meRssiToBeacon.append([loBeacon,loDistanceToObject])
                            self.meTrackers.append(loTrackerNew)
                #For a specific master
                #Loop all tarcker devices and check if this master (beacon) has this tracker device
                for p, loTracker in enumerate(self.meTrackers):
                    loTrackerFoundToResetBeaconValue = True
                    for r, loBeaconToCheck in enumerate(loTracker.meRssiToBeacon):
                        #Beacon has the same name as in the list
                        if(loBeaconToCheck[0].meName == master):
                            for device in root.iter(tag = 'device'):
                                for loTrackerToFind in device.iter(tag = 'name'):
                                    if (loTracker.meName == loTrackerToFind.text):
                                        loTrackerFoundToResetBeaconValue = False
                    if loTrackerFoundToResetBeaconValue:
                        for r, loBeaconToCheck in enumerate(loTracker.meRssiToBeacon):
                            #Beacon has the same name as in the list
                            if(loBeaconToCheck[0].meName == master):
                                loBeaconToCheck[1] = str(-1000)
                if not loBeaconFound:
                    print('Beacon not found in list error ....')

            except Exception as e:
                print("Error parseXML:", e)
```

```
    ############################################################
    #Created for manual testing of the methods on the class  #
    ############################################################
    def sortDesc(self, paTracker: clTracker):
        try:
            loTracker = paTracker
            for i, loBeaconAndRssi in enumerate(loTracker.meRssiToBeacon):
                j = i + 1
                while j < len(loTracker.meRssiToBeacon):
                    if (int(loTracker.meRssiToBeacon[i][1]) < int(loTracker.meRssiToBeacon[j][1])):
                        temp = loTracker.meRssiToBeacon[i]
                        loTracker.meRssiToBeacon[i] = loTracker.meRssiToBeacon[j]
                        loTracker.meRssiToBeacon[j] = temp
                    j = j + 1

            return loTracker
        except Exception as e:
            print("clSocketBoxBeaconServer::sortDesc -> Error:", e)
            return None

    def calculatePoints(self):
        try:
            #Set the trackerIDs to zero
            self.meSocketBoxSimulator.TrackerIDs = []

            for p, listTracker in enumerate(self.meTrackers):
                loTracker = self.sortDesc(listTracker)
                #If there are more than one beacons to detect the tracker
                if len(loTracker.meRssiToBeacon) == 2:
                    if (int(loTracker.meRssiToBeacon[0][1]) != -1000) and
(int(loTracker.meRssiToBeacon[1][1]) != -1000):
                        #Distance calculation
                        loFirstBeacon = loTracker.meRssiToBeacon[0][0]
                        loSecondBeacon = loTracker.meRssiToBeacon[1][0]
                        loFirstBeaconDistance = self.distanceCalculation(int(loTracker.meRssiToBeacon[0]
[1]))
                        loSecondBeaconDistance =
self.distanceCalculation(int(loTracker.meRssiToBeacon[1][1]))
                        print ("The %s has distance %.2f" %
(loFirstBeacon.meName,loFirstBeaconDistance))
                        print ("The %s has distance %.2f" %
(loSecondBeacon.meName,loSecondBeaconDistance))

                        #Base calculation
                        loBaseDistance = loFirstBeacon.mePoint.distance_to(loSecondBeacon.mePoint)
                        print("The base distance %.2f" % (loBaseDistance))

                        #Level Distance 1 and 2
                        if (loFirstBeaconDistance + loSecondBeaconDistance) < loBaseDistance:
                            loTemp = loBaseDistance - (loFirstBeaconDistance + loSecondBeaconDistance)
                            #+1 to create a triangle
```

```
                    loFirstBeaconDistance = loFirstBeaconDistance + (loTemp/2) + 1
                    loSecondBeaconDistance = loSecondBeaconDistance + (loTemp/2) + 1
                    print("Leveled distance %s: %.2f" %
(loFirstBeacon.meName,loFirstBeaconDistance))
                    print("Leveled distance %s: %.2f" %
(loSecondBeacon.meName,loSecondBeaconDistance))


                #Calculate coord
                result =
self.coordCalc(abs(loBaseDistance),loFirstBeaconDistance,loSecondBeaconDistance,loFirstBeaco
n.mePoint,loSecondBeacon.mePoint,clPoint(170.0,-965.0,0.0))
                if result:
                    print(f"Calculated location: {result}")

self.meSocketBoxSimulator.TrackerIDs.append([listTracker.meName,result[0],result[1],0.05])
                else:
                    print(f"Result could not be caluclated ... ")

            else:
                print ("The Tracker has id: %s with beaconID_1 %s values Rssi: %.2f beaconID_2
%s Rssi: %.2f" % (loTracker.meName,loTracker.meRssiToBeacon[0][0].meName,
float(loTracker.meRssiToBeacon[0][1]), loTracker.meRssiToBeacon[1][0].meName,
float(loTracker.meRssiToBeacon[1][1])))
        elif len(loTracker.meRssiToBeacon) > 2:
            if (int(loTracker.meRssiToBeacon[0][1]) != -1000) and
(int(loTracker.meRssiToBeacon[1][1]) != -1000):
                #Distance calculation
                loFirstBeacon = loTracker.meRssiToBeacon[0][0]
                loSecondBeacon = loTracker.meRssiToBeacon[1][0]
                loThirdBeacon = loTracker.meRssiToBeacon[2][0]
                loFirstBeaconDistance = self.distanceCalculation(int(loTracker.meRssiToBeacon[0]
[1]))
                loSecondBeaconDistance =
self.distanceCalculation(int(loTracker.meRssiToBeacon[1][1]))
                loThirdBeaconDistance =
self.distanceCalculation(int(loTracker.meRssiToBeacon[2][1]))
                print ("The %s has distance %.2f" %
(loFirstBeacon.meName,loFirstBeaconDistance))
                print ("The %s has distance %.2f" %
(loSecondBeacon.meName,loSecondBeaconDistance))
                print ("The %s has distance %.2f" %
(loThirdBeacon.meName,loThirdBeaconDistance))

                #Base calculation
                loBaseDistance = loFirstBeacon.mePoint.distance_to(loSecondBeacon.mePoint)
                print("The base distance %.2f" % (loBaseDistance))

                #Level Distance 1 and 2
                if (loFirstBeaconDistance + loSecondBeaconDistance) < loBaseDistance:
                    loTemp = loBaseDistance - (loFirstBeaconDistance + loSecondBeaconDistance)
                    #+1 to create a triangle
```

```
                    loFirstBeaconDistance = loFirstBeaconDistance + (loTemp/2) + 1
                    loSecondBeaconDistance = loSecondBeaconDistance + (loTemp/2) + 1
                    print("Leveled distance %s: %.2f" %
(loFirstBeacon.meName,loFirstBeaconDistance))
                    print("Leveled distance %s: %.2f" %
(loSecondBeacon.meName,loSecondBeaconDistance))

                #Calculate coord
                result =
self.coordCalc(loBaseDistance,loFirstBeaconDistance,loSecondBeaconDistance,loFirstBeacon.me
Point,loSecondBeacon.mePoint,loThirdBeacon.mePoint)
                if result:
                    print(f"Calculated location: {result}")

self.meSocketBoxSimulator.TrackerIDs.append([listTracker.meName,result[0],result[1],0.05])
                else:
                    print(f"Result could not be caluclated ... ")




            else:
                print ("The Tracker has id: %s with beaconID_1 %s values Rssi: %.2f beaconID_2
%s Rssi: %.2f beaconID_3 %s values Rssi: %.2f" %
(loTracker.meName,loTracker.meRssiToBeacon[0][0].meName,
float(loTracker.meRssiToBeacon[0][1]), loTracker.meRssiToBeacon[1][0].meName,
float(loTracker.meRssiToBeacon[1][1]), loTracker.meRssiToBeacon[2][0].meName,
float(loTracker.meRssiToBeacon[2][1])))
            else:
                print ("The Tracker has id: %s has not more than 1 detection Beacon" %
(loTracker.meName))



    except Exception as e:
        print("Error calculatePoints:", e)

  def distanceCalculation(self,rssiValue) -> float:
    powerAtOneMeter = -49
     #3.2 to short
     #2.6 good



     #WROOM -49 and 2.3
     #Mini Esp32 -57 and 2
    scaleFactor = 2.5
    return (pow(10,((powerAtOneMeter-(rssiValue))/(10*scaleFactor))) * 100)

  def coordCalc (self,paDistanceBase,paDistance1,paDistance2, paPoint1: clPoint, paPoint2:
clPoint, paPoint3: clPoint) -> list[float, float]:
    try:
        #Calculate the area
        perimeter = paDistanceBase + paDistance1 + paDistance2
```

```
        s = perimeter/2
        area = math.sqrt(s*(s-paDistanceBase)*(s-paDistance1)*(s-paDistance2))
        #area = base * height/2
        height = (2*area)/paDistanceBase


        #Get point length on the Base
        #c²=x²+y² => c = sqrt(x²+y²)
        pointLenghtOnBase = 0
        if height <= paDistance1:
           pointLenghtOnBase = math.sqrt(pow(paDistance1,2) - pow(height,2))
        else:
           pointLenghtOnBase = math.sqrt(pow(height,2) - pow(paDistance1,2))
        print("The distance to the point is: %.2f and height is: %.2f" % (pointLenghtOnBase,
height))


        cosAlfa = (paPoint1.x - paPoint2.x)/paDistanceBase
        radians = math.acos(cosAlfa)
        sinAlfa = math.sin(radians)
        degrees = radians * (180.0 / math.pi)
        print("The Angle of the base is: %.2f in radians is: %.2f" % (degrees, radians))
        pointLenghtOnBase_X = cosAlfa * pointLenghtOnBase
        pointLenghtOnBase_Y = sinAlfa * pointLenghtOnBase
        print("The pointLenghtOnBase_X: %.2f" % (pointLenghtOnBase_X))
        print("The pointLenghtOnBase_Y: %.2f" % (pointLenghtOnBase_Y))

        pointlenghtOnBase_OrigCoord_X = 0
        pointlenghtOnBase_OrigCoord_Y = 0
        #Quadtrant 3
        if (paPoint1.x > paPoint2.x) and (paPoint1.y > paPoint2.y) and (paPoint1.y < paPoint3.y):
           print ("Quadtrant 3 A")
           pointlenghtOnBase_OrigCoord_X = paPoint1.x - abs(pointLenghtOnBase_X)
           pointlenghtOnBase_OrigCoord_Y = paPoint1.y + abs(pointLenghtOnBase_Y)
        elif (paPoint1.x > paPoint2.x) and (paPoint1.y >= paPoint2.y) and (paPoint1.y >
paPoint3.y):
           print ("Quadtrant 3 B")
           pointlenghtOnBase_OrigCoord_X = paPoint1.x - abs(pointLenghtOnBase_X)
           pointlenghtOnBase_OrigCoord_Y = paPoint1.y - abs(pointLenghtOnBase_Y)
        #Quadtrant 1
        elif (paPoint1.x < paPoint2.x) and (paPoint1.y <= paPoint2.y) and (paPoint1.y <
paPoint3.y):
           print ("Quadtrant 1 A")
           pointlenghtOnBase_OrigCoord_X = paPoint1.x + abs(pointLenghtOnBase_X)
           pointlenghtOnBase_OrigCoord_Y = paPoint1.y + abs(pointLenghtOnBase_Y)
        elif (paPoint1.x < paPoint2.x) and (paPoint1.y <= paPoint2.y) and (paPoint1.y >
paPoint3.y):
           print ("Quadtrant 1 B")
           pointlenghtOnBase_OrigCoord_X = paPoint1.x + abs(pointLenghtOnBase_X)
           pointlenghtOnBase_OrigCoord_Y = paPoint1.y - abs(pointLenghtOnBase_Y)
        #Quadtrant 2
        elif (paPoint1.x >= paPoint2.x) and (paPoint1.y < paPoint2.y) and (paPoint1.x <
```

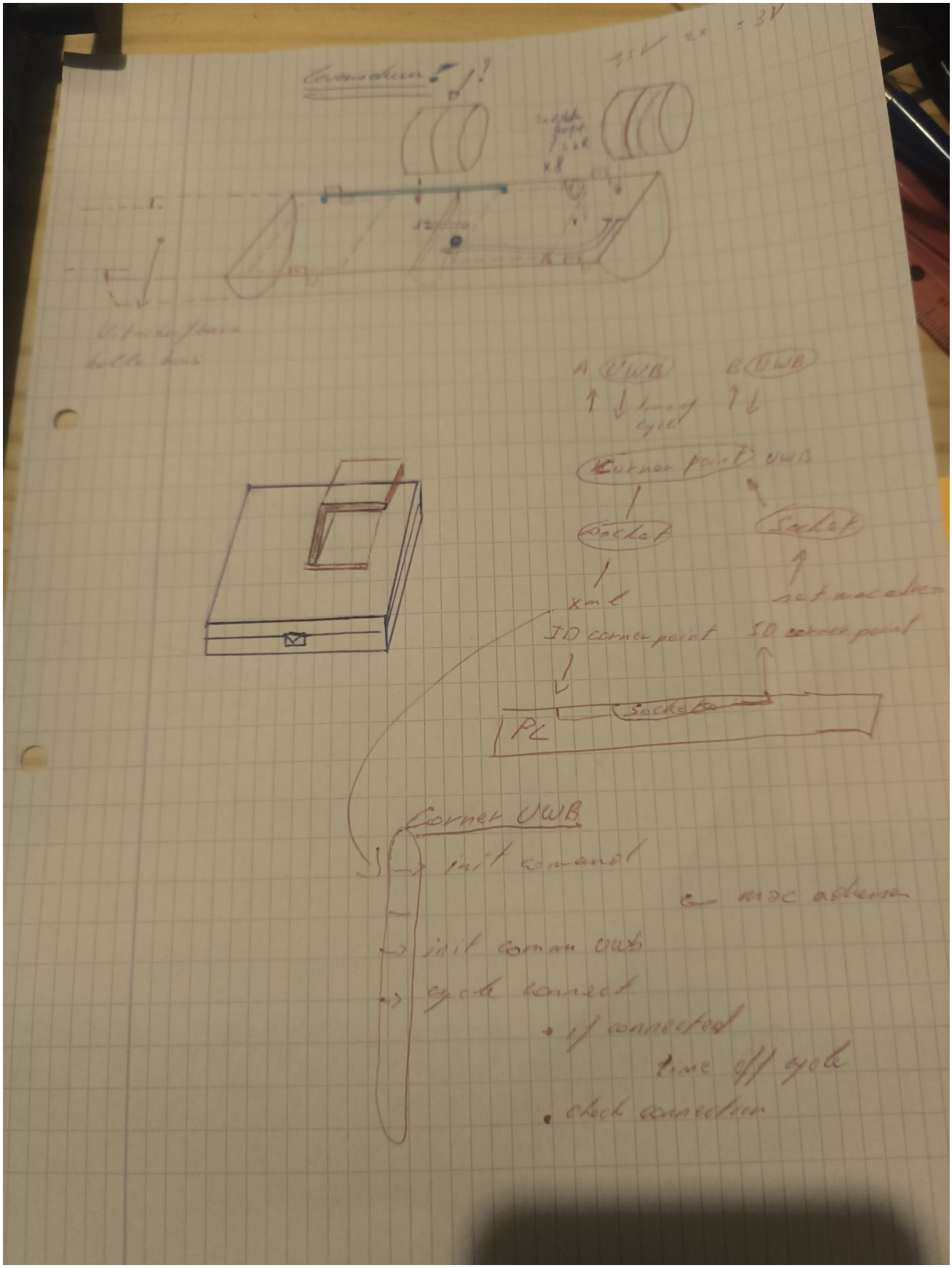
```
paPoint3.x):
            print ("Quadtrant 2 A")
            pointlenghtOnBase_OrigCoord_X = paPoint1.x + abs(pointLenghtOnBase_X)
            pointlenghtOnBase_OrigCoord_Y = paPoint1.y + abs(pointLenghtOnBase_Y)
        elif (paPoint1.x >= paPoint2.x) and (paPoint1.y < paPoint2.y) and (paPoint1.x >
paPoint3.x):
            print ("Quadtrant 2 B")
            pointlenghtOnBase_OrigCoord_X = paPoint1.x - abs(pointLenghtOnBase_X)
            pointlenghtOnBase_OrigCoord_Y = paPoint1.y + abs(pointLenghtOnBase_Y)
        #Quadtrant 4
        elif (paPoint1.x <= paPoint2.x) and (paPoint1.y > paPoint2.y) and (paPoint1.x <
paPoint3.x):
            print ("Quadtrant 4 A")
            pointlenghtOnBase_OrigCoord_X = paPoint1.x + abs(pointLenghtOnBase_X)
            pointlenghtOnBase_OrigCoord_Y = paPoint1.y - abs(pointLenghtOnBase_Y)
        elif (paPoint1.x <= paPoint2.x) and (paPoint1.y > paPoint2.y) and (paPoint1.x >
paPoint3.x):
            print ("Quadtrant 4 B")
            pointlenghtOnBase_OrigCoord_X = paPoint1.x - abs(pointLenghtOnBase_X)
            pointlenghtOnBase_OrigCoord_Y = paPoint1.y - abs(pointLenghtOnBase_Y)


        print("pointlenghtOnBase_OrigCoord_X = %.2f" %(pointlenghtOnBase_OrigCoord_X))
        print("pointlenghtOnBase_OrigCoord_Y = %.2f" %(pointlenghtOnBase_OrigCoord_Y))

        #Check the direction of the hight
        loLeftOrRight = ""
        loUpperOrLower = ""

        if (degrees == 0.0 or degrees == 360.0):
            if (pointlenghtOnBase_OrigCoord_Y <= paPoint3.y):
                print ("Degrees 0 = LE")
                loLeftOrRight = "LE"
            elif (pointlenghtOnBase_OrigCoord_Y >= paPoint3.y):
                print ("Degrees 0 = RI")
                loLeftOrRight = "RI"
        elif (degrees == 90.0):
            if (pointlenghtOnBase_OrigCoord_X <= paPoint3.x):
                print ("Degrees 90 = RI")
                loLeftOrRight = "RI"
            elif (pointlenghtOnBase_OrigCoord_X >= paPoint3.x):
                print ("Degrees 90 = LE")
                loLeftOrRight = "LE"
        elif (degrees == 180.0):
            if (pointlenghtOnBase_OrigCoord_Y <= paPoint3.y):
                print ("Degrees 180 = LE")
                loLeftOrRight = "LE"
            elif (pointlenghtOnBase_OrigCoord_Y >= paPoint3.y):
                print ("Degrees 180 = RI")
                loLeftOrRight = "RI"
        elif (degrees == 270.0):
            if (pointlenghtOnBase_OrigCoord_X <= paPoint3.x):
```

```
                print ("Degrees 270 = LE")
                loLeftOrRight = "LE"
            elif (pointlenghtOnBase_OrigCoord_X >= paPoint3.x):
                print ("Degrees 270 = RI")
                loLeftOrRight = "RI"
        elif (pointlenghtOnBase_OrigCoord_X <= paPoint3.x):
            print ("corner RI")
            loLeftOrRight = "RI"
        elif (pointlenghtOnBase_OrigCoord_X >= paPoint3.x):
            print ("corner LE")
            loLeftOrRight = "LE"

        pointOfLocation_X = 0.0
        pointOfLocation_Y = 0.0
        pointOfLocation_OrigCoord_X = 0.0
        pointOfLocation_OrigCoord_Y = 0.0

        #coord of the point pointLenghtOnBase
        #Select quadrant 3
        if (paPoint1.x > paPoint2.x) and (paPoint1.y >= paPoint2.y):
            print("Entering quadrant 3")
            #Triangle goes right
            if loLeftOrRight == "RI":
                pointOfLocation_Y = cosAlfa * height
                pointOfLocation_X = sinAlfa * height
                pointOfLocation_OrigCoord_X = pointlenghtOnBase_OrigCoord_X -
abs(pointOfLocation_X)
                pointOfLocation_OrigCoord_Y = pointlenghtOnBase_OrigCoord_Y -
abs(pointOfLocation_Y)
            #Triangle goes left
            elif loLeftOrRight == "LE":
                pointOfLocation_Y = cosAlfa * height
                pointOfLocation_X = sinAlfa * height
                pointOfLocation_OrigCoord_X = pointlenghtOnBase_OrigCoord_X -
abs(pointOfLocation_X)
                pointOfLocation_OrigCoord_Y = pointlenghtOnBase_OrigCoord_Y +
abs(pointOfLocation_Y)
        #Select quadrant 1
        elif (paPoint1.x < paPoint2.x) and (paPoint1.y <= paPoint2.y):
            #degreesHelp = 180 - degrees
            #radiansHelp = degreesHelp * (math.pi / 180)
            #if ((degreesHelp >= 0) and (degreesHelp <= 90)):
            print("Entering quadrant 1")
            #Triangle goes right
            if loLeftOrRight == "RI":
                pointOfLocation_Y = cosAlfa * height
                pointOfLocation_X = sinAlfa * height
                pointOfLocation_OrigCoord_X = pointlenghtOnBase_OrigCoord_X +
abs(pointOfLocation_X)
                pointOfLocation_OrigCoord_Y = pointlenghtOnBase_OrigCoord_Y -
abs(pointOfLocation_Y)
            #Triangle goes left
```

```
            elif loLeftOrRight == "LE":
                pointOfLocation_Y = cosAlfa * height
                pointOfLocation_X = sinAlfa * height
                pointOfLocation_OrigCoord_X = pointlenghtOnBase_OrigCoord_X +
abs(pointOfLocation_X)
                pointOfLocation_OrigCoord_Y = pointlenghtOnBase_OrigCoord_Y +
abs(pointOfLocation_Y)
        #Select quadrant 2
        elif (paPoint1.x >= paPoint2.x) and (paPoint1.y < paPoint2.y):
            print("Entering quadrant 2")
            #Triangle goes right
            if loLeftOrRight == "RI":
                pointOfLocation_Y = cosAlfa * height
                pointOfLocation_X = sinAlfa * height
                pointOfLocation_OrigCoord_X = pointlenghtOnBase_OrigCoord_X +
abs(pointOfLocation_X)
                pointOfLocation_OrigCoord_Y = pointlenghtOnBase_OrigCoord_Y +
abs(pointOfLocation_Y)
            #Triangle goes left
            elif loLeftOrRight == "LE":
                pointOfLocation_Y = cosAlfa * height
                pointOfLocation_X = sinAlfa * height
                pointOfLocation_OrigCoord_X = pointlenghtOnBase_OrigCoord_X -
abs(pointOfLocation_X)
                pointOfLocation_OrigCoord_Y = pointlenghtOnBase_OrigCoord_Y +
abs(pointOfLocation_Y)

        #Select quadrant 4
        elif (paPoint1.x <= paPoint2.x) and (paPoint1.y > paPoint2.y):
            print("Entering quadrant 4")
            #Triangle goes right
            if loLeftOrRight == "RI":
                pointOfLocation_Y = cosAlfa * height
                pointOfLocation_X = sinAlfa * height
                pointOfLocation_OrigCoord_X = pointlenghtOnBase_OrigCoord_X +
abs(pointOfLocation_X)
                pointOfLocation_OrigCoord_Y = pointlenghtOnBase_OrigCoord_Y -
abs(pointOfLocation_Y)
            #Triangle goes left
            elif loLeftOrRight == "LE":
                pointOfLocation_Y = cosAlfa * height
                pointOfLocation_X = sinAlfa * height
                pointOfLocation_OrigCoord_X = pointlenghtOnBase_OrigCoord_X -
abs(pointOfLocation_X)
                pointOfLocation_OrigCoord_Y = pointlenghtOnBase_OrigCoord_Y -
abs(pointOfLocation_Y)

        return (pointOfLocation_OrigCoord_X,pointOfLocation_OrigCoord_Y)
    except Exception as e:
        print("Error coordCalc:", e)
        return None
```

```
def main():
    print('Start SocketBoxLocationBeacon')

    #Define the beacons with name and location
    loBeacons = []
    loBeacons.append(clBeacon("2UVBEACON_001_X",clPoint(170.0,-310.0,0.0)))
    loBeacons.append(clBeacon("2UVBEACON_001_Y",clPoint(895.0,-310.0,0.0)))
    loBeacons.append(clBeacon("2UVBEACON_001_Z",clPoint(170.0,-965.0,0.0)))


    #Create the socket communicator
    meSocketBoxSimulator = clSocketBoxSimulator()
    meSocketBoxBeaconServer = clSocketBoxBeaconServer(meSocketBoxSimulator, loBeacons)
    #Start the BLE threading
    x = threading.Thread(target=meSocketBoxBeaconServer.run_server, args=())
    x.start()

    #Test the function manually
    #while True:
    #    try:
    meSocketBoxSimulator.run_server()
    #    except:
    #        print('Exception')


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

Model:


UWB
UWB
Corner point UWB
Socket
Socket
IO corner point
IO corner point
Socket
PC
Corner UWB
init command
init comm UWB
check connect
if connected
check connection

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

*Distance = ToF * Speed of light*

ToF = [(TRF - TSP) - (TSR - TRP)] / 4

The Two-Way Ranging (TWR) Process

1. 1. **Poll Message**:
   The tag (mobile device) sends a "Poll" message to an anchor (fixed device), recording the time of transmission (TSP).
2. 2. **Response Message**:
   The anchor receives the Poll message, notes the reception time (TRP), and sends a "Response" message back to the tag at a later time (TSR).
3. 3. **Final Message**:
   The tag receives the Response message, records the reception time (TRR), and sends a "Final" message back to the anchor, including its ID, TSP, TRR, and the time of the Final message transmission (TSF).
4. 4. **ToF Calculation by Anchor**:
   The anchor receives the Final message at time (TRF). It then uses all the timestamps to calculate the overall Time of Flight (ToF).

ToF Calculation Formula
The specific calculation for ToF from the timestamps is: ToF = [(TRF - TSP) - (TSR - TRP)] / 4.
Distance Calculation
Once the ToF is known, the distance is calculated using the fundamental formula: Distance = ToF x Speed of Light.
Why UWB is Effective

Übersicht mit KI

TWR (Two-Way Ranging) is . Arduino users can implement TWR for real-time location systems (RTLS) or access control using hardware like the Qorvo DWM3000EVB or the Arduino Portenta UWB Shield, often with libraries available on platforms like Arduino Docs and GitHub.
How Two-Way Ranging (TWR) Works

1. **Initiator (Device A)**: sends a message to the Responder (Device B) and records the time it was sent (T1).

- **Device B**: receives the message, processes it, and sends a response back to Device A after a known delay.
- **Device A**: receives the response, records the time (T2), and then calculates the distance using the round-trip time (T2 - T1) and known processing delays.
- By sending multiple messages, TWR can compensate for clock synchronization issues between devices, improving accuracy.

4.

Implementing TWR with Arduino

- **Hardware**:

You will need an Arduino-compatible board and UWB modules, such as the Qorvo DWM3000 modules, often available on development shields like the DWM3000EVB, or integrated into boards like the Arduino Stella or the Portenta UWB Shield.

- **Libraries**:

Libraries are available to simplify TWR implementation:

- **Arduino Docs**: Provide libraries and user guides for specific hardware like the Arduino Stella and Portenta UWB Shield.
- **GitHub**: Offers example code and libraries, such as the RAK13801_UWB library for TWR, and community-driven projects like DecaDuino.
-
- **Use Cases**:
- **Real-Time Location Systems (RTLS)**: For precise positioning of objects or people in industrial environments, enabling accident prevention and congestion reduction.
- **Access Control**: To create secure, contactless entry systems for workspaces or vehicles.
- **Proximity Detection**: To enable automatic equipment operation when an authorized user is within a specific distance.

Advanced TWR Techniques

- **Single-Sided Two-Way Ranging (SS-TWR)**: A basic implementation involving a single exchange of messages.
- **Double-Sided Two-Way Ranging (DS-TWR)**: More complex, involving multiple messages in both directions to further improve accuracy, for example, Skew-Aware Single-Sided Two-Way Ranging (SASS-TWR) or Asymmetric Double-Sided Two-Way Ranging (ADS-TWR), which compensate for clock skew.

Dw3000ds

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