

Róbot armur stýrður með handaskynjun

Efnisyfirlit

<i>Inngangur.....</i>	4
<i>Meginmál.....</i>	4
- <i>Framkvæmd.....</i>	4
- <i>Virknilýsing</i>	5
- <i>Formúlur og útkomur.....</i>	6
- <i>Efniskostnaður</i>	7
<i>Lokaorð</i>	8
<i>Myndaskrá</i>	9
<i>Viðauki 1: Kóði</i>	10

Inngangur

Ástæðan fyrir því að ég tók þetta verkefni að mér er því mig langaði að gera eitthvað sem nær til allra og allir geta sýnt áhuga á. Ég fékk hugmyndina frá myndabandi af Jeff Bezos vera að stjórna róbot örmum og hlæja af gleði. Ef ríkasti maður heimsins hefur gaman að því að leika sér með róbot arma er hægt að treysta því að flestir geta haft gaman að því. Þetta verkefni nýtir einnig kunnáttu mína í forritun, skilning á tölvum og smáspennu.



Mynd 1: Jeff Bezos að prúfa róbot arma stjórnað með handahreyfingum

Meginmál

- Framkvæmd

Margir möguleikar eru á framkvæmd við slíkt verkefni. Hvern skonar róbot arm á að nota? Hvernig skynja ég handahreyfingar? Hvernig stjórna ég hreyfingu mótoranna? Fyrst langaði mig að búa til stóran róbot arm eins og þennan til hægri. En tveir þættir koma í veg fyrir það. Kostnaður svona stepper mótorra í hann er mikill og það eru margar flóknar hreyfingar og snúningar sem hann getur gert. Mig langar sjálfur að forrita script sem finnur hvernig armurinn á að hreyfa sig miðað við staðsetningu handar, en það gekk ekki upp og ég þurfti að sætta mig við að nota svipaðan kóða frá Github sem dugar fyrir sýningu á hreyfingum.



Mynd 2: BCN3D Róbot armar

Skynjun á handahreyfingum er hægt að framkvæma á marga vegu. Hægt er að vera með hanska með hreyfiskynjara eða með skynjurum fyrir puttahreyfingar. Mig langar að breyta alvöru staðsetningu handar í staðsetningu róbot armsins og skynjun með myndavél er eina raunhæfa aðferðin að því. Það að nota myndavél er með svalari lausnunum að mínu mati því það þarf að nota vitvél. Með myndavél og nýjum aðgengilegum skjölum að vitvélum get ég skynjað staðsetningu handar, og puttahreyfingar með ódýrri myndavél. Eini gallinn er hvernig fær maður tölvu til að nýta sér allar þessar upplýsingar eins og manneskja?

Fyrir handaskynjun nota ég Mediapipe eftir Google. Mediapipe er verkefni til að gera skynjun líkama, andlita og handa með vitvél aðgengilegt öllum. Mediapipe er open source kóði sem hægt er að breyta, bæta og nýta sér í persónuleg verkefni.

- Virknilysing

Mediapipe er verkefni þar sem Google dældi helling af myndum í forrit, meðal annars myndum af höndum, og fékk fólk til að merkja hvar hendi er staðsett. Svo er fleiri myndum dælt í forrit en núna giskar talvan sjálf hvar hendi er staðsett og manneskja segir hvort það er rétt eða rangt. Þegar þetta ferli hefur verið endurtekið margoft getur talvan giskað nokkuð vel hvar hendi sé á mynd. Google hefur gert þetta aðgengilegt öllum og er ég að nýta mér það til að skynja staðsetningu handa á myndaramma.

Hnitin eru fundin með að þekkja stærð myndarinnar sem vitvélin les. Mediapipe er með innbyggt fall sem finnur staðsetningu handar og skilar út X og Y hnitum í pixlum. Þar sem stærð myndarinnar er

600x400 pixlar, myndi ég fá staðsetningargildi þar sem X getur verið 0-600 og Y 0-400. Hnitin eru í millimetrum fyrir arminn.

Z hnitin eru fundin með að finna lengd á milli tveggja punkta á hendi notenda.

Eini gallinn við að reikna Z hnit með lengd á milli tveggja punkta er að myndavélar sjá ekki fjarlægð línulega. Þannig að hlutfall á fjarlægð og lengd handar er ekki 1:1 hlutfall. Þannig því lengra sem hendi er frá myndavélinni, því minni munur er á Z hreyfingu.

Hægt er að leysa þetta með að setja formúlu í forritið sem reiknar fjarlægð með þekkri brennivídd myndavélarinnar. Brennivídd vefmyndavélarinnar á fartölvunni minni er 50mm. Þessi lausn er ekki enn komin í forritið en tel ég þetta vera tiltulega auðvelda lausn sem er hægt að bæta við forritið.

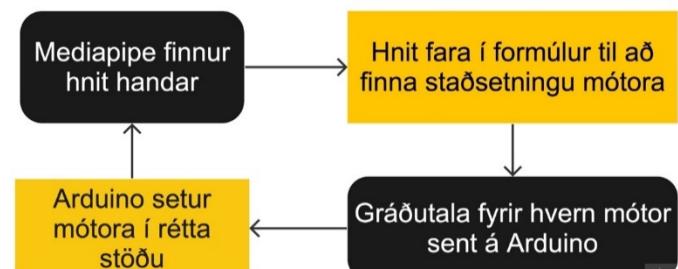
Hendin framan á róbot arminum getur opnað og lokað til að halda utan um hluti og er séð hversu opin eða lokað hún á að vera með lengd á milli þumalputtar og vísifingurs.

Lengd á milli hluta er fundin með Pýbagórasarreglunni.

Sami galli gildir fyrir lengd á milli þumals og vísifingur úr fjarlægð. Lausnin við þessi vandamáli er að finna hlutfall á milli lengd tveggja punkta á hendi og fjarlægð frá myndavél. Lengd á milli puttanna



Mynd 3: Prufun á handaskynjun



Mynd 4: Vinnuferill kóðans



Mynd 5: Róbot armurinn

væri svo margfaldað með þessu hlutfalli. Mótorarnir eru keyrðir með Arduino tölvu í gegnum PWM pinna. 5 volta spennubreytir veitir öllum mótorum spennu til að hreyfa sig en Arduino talvan er einungis til að segja þeim hvernig þeir skulu snúa sér. Í Arduino tölvunni er búið að setja upp hugbúnaðinn Firmata sem gerir henni kleift að tala auðveldlega við python kóða.

```
board = Arduino("/dev/cu.usbmodem1101")
board.digital[3].mode = SERVO
board.digital[5].mode = SERVO
board.digital[6].mode = SERVO
board.digital[9].mode = SERVO
```

Mynd 6: Bútur úr kóðanum sem sýnir skilgreiningu Arduino og pinna

```
board.digital[5].write(180 - baseAngle)
board.digital[6].write(armAngle)
board.digital[9].write(vArmAngle)
```

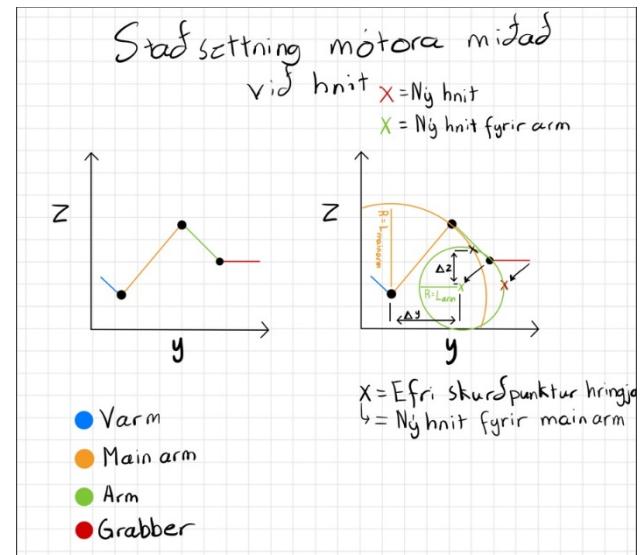
Mynd 7: Bútur úr kóðanum sem sýnir hvernig

- Formúlur og útkomur

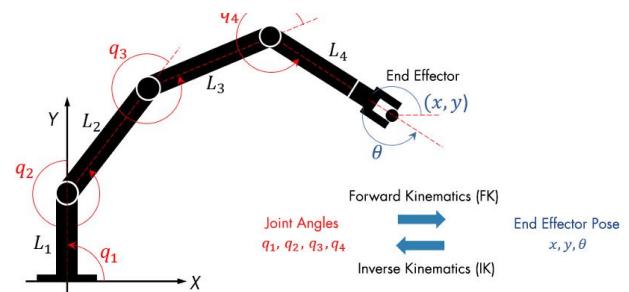
Reikningur að formúlum hefur reynst vera erfitt ferli því alltaf er eitthvað sem virkaði ekki og ég skildi ekki hvað var að. Upprunalega hugmyndin var að finna skurðpunkta hringja eins og sést á mynd nr 8. Þessi lausn virkar en reyndist flóknari þegar snúningi base mótorsons var bætt við. Enn veit ég ekki afhverju þetta virkar ekki, en ég er sannfærður um að eftir smá andlega þásu frá verkefninu get ég fengið mína lausn til að virka. Aðal formúlur notaðar eru hornaföll og pýbagórasareglan. Ég forritaði meirihluta kóðans áður en ég setti arminn saman og voru þá margir hlutir sem ég hafði ekki hugsað fyrir. Tvö dæmi eru að þar sem main arm, og v arm mótorarnir eru öfugu megin við hvorn annan þarf snúningur annars mótorsins að vera speglaður. Annað dæmi er að tannhjólin base mótorinn snýr eru ekki með 1:1 hlutfall. Þannig að þó base mótorinn snýst um 180° snýst armurinn aðeins um u.b.b 110° .

Þegar fór að líða á önnina var ég farinn að átta mig á að ég væri ekki að fara að finna út úr formúlunum

mínunum fyrir útskrift og vildi ég ekki vera með óhreyfanlegan arm til sýnis í kynningunni. Ég gafst upp á þrjóskunni og ákvað að leita á netinu að upplýsingum. Það sem ég fann er að það sem ég hafði verið að búa til er forrit fyrir það sem er kallað á ensku „Inverse kinematics“, sem er að finna staðsetningu mótora eftir óska hnitudum og lengdum arms. Formúlurnar fyrir þessa reikninga eru töluvert flóknara en það sem ég hafði verið að gera, en ég hafi samt verið með rétta grunnin þar sem hornaföll eru aðal reikningsaðferðirnar. Ég leitaði svo að þessum formúlum fyrir



Mynd 8: Upprunaleg hugmynd á inverse kinematics fyrir arminn



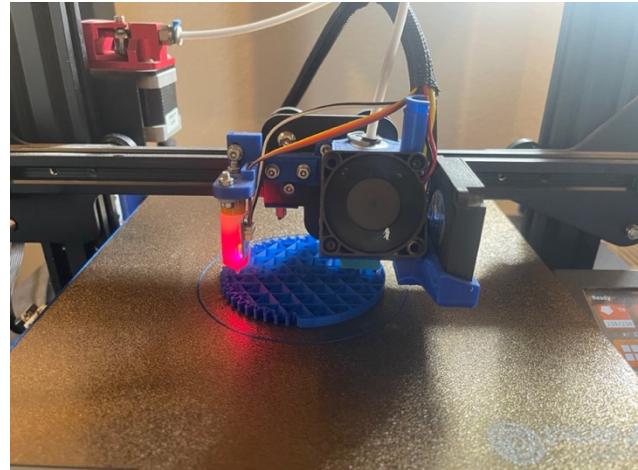
Mynd 9: Einföld lýsing á hugmynd inverse kinematics aðferðinnar

arminn sem ég hafði prentað og fann ég óþarfa langan kóða fyrir inverse kinematics fyrir þennan róbot arm á Github. Sá kóði var líka með föllum til að sýna staðsetningu armsins í þrívíddar grafi, og klasa fyrir bæði gerð eitt og tvö af arminum, þar sem tvær gerðir hafa verið gerðar. Þannig ég þurfti aðeins að nýta mér lítinn hluta af kóðanum. Allur kóðinn er um 1.300 línur. Eftir að hafa bætt þessum kóða við minn kóða virkaði það samt ekki. Tveir hlutir voru að fara úrskeiðis, X og Y hnitin voru nýtt öfugt í þessum nýja kóða, þannig Y hnitin voru hreyfingar frá vinstri til hægri og fram og aftur hreyfingar voru X hnitin. Ekki var neitt skrifað um þetta í kóðanum og fann ég út úr því frá myndum af þrívíddar gröfum á Github síðunni. Auðvelt var að laga það en annar hluturinn sem virkaði ekki var fall sem skoðaði hvort armurinn gæti komist að hnitudum.

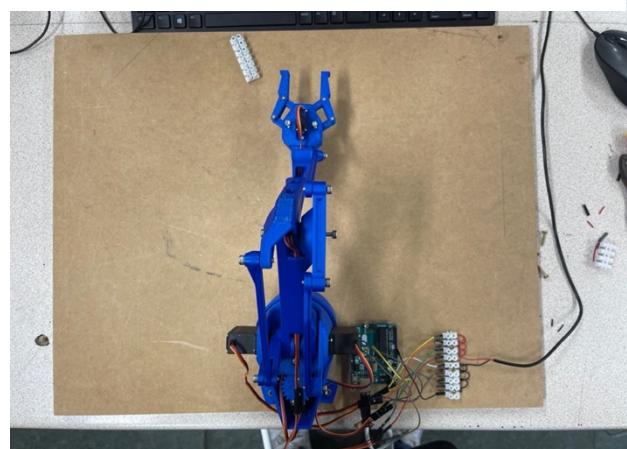
Svo að mótorarnir séu ekki að vinna á móti hvorum öðrum eða fara útfyrir vinnusvæði armsins var sett fall í kóðann sem skoðar einmitt það. Ef fallið finnur að það sé tilfellið, leyfir hann ekki hreyfinuna. Þetta fall virkaði ekki og leyfði aðeins hreyfingu á örlítili línu frá vinstri til hægri. Þar sem þessi kóði og formúlurnar eru fyrir utan mína kunnáttu í stærðfræði ákvað ég að taka fallið úr kóðanum. Það leysti vandamálið og armurinn gat núna hreyft sig um allt vinnusvæðið. Til að koma í veg fyrir að mótorarnir vinna á móti hvor öðrum setti ég limit á hversu lág hnít skynjaðar handar notandans. Þannig ef hending fer niður fyrir Y = 115 helst Y sem 115.

- Efniskostnaður

Armurinn er þrívíddar prentaður úr PLA plasti. PLA plast er ódýrt og auðvelt að prenta en það er ekki eins sterkt né endist jafn lengi og annað sem hægt er að fá. En ég taldi það vera fínt í þetta verkefni. Þar sem plastið er ódýrt kostaði það undir 1.000kr að prenta arminn, en skrúfurnar til að fest hann saman bætir eitthvað aðeins í það. Kostnaður á öllum mótorum er ekki meiri en 4.000kr án sendingarkostnaðar. Aðrir aukahlutir kostuðu meira þar sem þeir voru pantaðir sér frá Amazon á 10.000kr. Arduino talva var núþegar til í skólanum meðal mótotengja, dupont snúrra og viðarplötu. Erfitt er að reikna nákvæman kostnað þar sem margir hlutir voru til staðar í skólanum en tel ég verkefnið hafa kostað u.p.b 25.000kr fyrir utan fartölvu og vefmyndavél.



Mynd 10: Hluti armsins í prentun



Mynd 11: Loka útlit verkefnisins

Lokaorð

Allt í allt tel ég mig hafa uppfyllt grunnhugmynd verkefnisins. Það eru nokkrir hlutir sem ég hefði viljað gera auðruvísi núna. T.d að hafa leitað upplýsinga á netinu fyrr í staðin fyrir að reyna að finna formúlurnar sjálfur. Hefði ég gert það tel ég mig hafa haft meiri tíma í að finnpússa hreyfingar og gert kóðan auðveldari í uppsetningu. Það eru nokkrir hlutir sem mig langar að bæta við verkefnið þótt ég geri það ekki fyrr en eftir útskrift. Eins og að kynna mér inverse kinematics betur til að gera minn eigin kóða til að hreyfa hendina, mögulega þá líka búa til Python eða C++ pakka sem væri hægt að nota fyrir kennslu til að geta gert hreyfingar armsins auðveldari til að forrita. Þar sem ein hugmynd fyrir notkun armsins í framtíðinni er við kennslu.

Myndaskrá

Mynd 1: Jeff Bezos að prufa róbot arma stjórnað með handahreyfingum	4
Mynd 2: BCN3D Róbot armar	4
Mynd 3: Prufun á handaskynjun.....	5
Mynd 4: Vinnuferill kóðans	5
Mynd 5: Róbot armurinn	5
Mynd 6: Bútur úr kóðanum sem sýnir skilgreingu Arduino og pinna	6
Mynd 7: Bútur úr kóðanum sem sýnir hvernig breytt er um staðsetningu mótoranna	6
Mynd 8: Upprunaleg hugmynd á inverse kinematics fyrir arminn	6
Mynd 9: Einföld lýsing á hugmynd inverse kinematics aðferðinnar	6
Mynd 10: Hluti armsins í prentun.....	7
Mynd 11: Loka útlit verkefnisins	7

Viðauki 1: Kóði

```
import math
import cv2
import mediapipe as mp
import time
from pyfirmata import Arduino, SERVO
from CoordinatesV5 import *

board = Arduino("/dev/cu.usbmodem1101")
board.digital[3].mode = SERVO
board.digital[5].mode = SERVO
board.digital[6].mode = SERVO
board.digital[9].mode = SERVO

mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles
mp_hands = mp.solutions.hands

cap = cv2.VideoCapture(0)

cap.set(3, 300)
cap.set(4, 300)

robotArm = EEZYbotARM_Mk2(0, 0, 0)

with mp_hands.Hands(
    model_complexity=0,
    min_detection_confidence=0.8,
    min_tracking_confidence=0.5) as hands:
    while cap.isOpened():
        success, image = cap.read()
        if not success:
            print("Ignoring empty camera frame.")
            print("Ignoring empty camera frame.")
            # If loading a video, use 'break' instead of 'continue'.
            continue

        results = hands.process(image)

        # Draw the hand annotations on the image.
        image.flags.writeable = True
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

        timeNow = time.time()

        if results.multi_hand_landmarks:
            time.sleep(0.016)
            for hand_landmarks in results.multi_hand_landmarks:

                handCoordinatesX = round(450 - hand_landmarks.landmark[mp_hands.HandLandmark.MIDDLE_FINGER_MCP].y * 500
                                         , 2)
                if handCoordinatesX < 115:
                    handCoordinatesX = 115
                handCoordinatesY = round(300 - hand_landmarks.landmark[mp_hands.HandLandmark.MIDDLE_FINGER_MCP].x * 600
                                         , 2)
                handCoordinatesZ = round(150 -
                                         math.sqrt(
                                             (hand_landmarks.landmark[mp_hands.HandLandmark.INDEX_FINGER_MCP].x -
                                              hand_landmarks.landmark[mp_hands.HandLandmark.PINKY_MCP].x)**2 +
                                             (hand_landmarks.landmark[mp_hands.HandLandmark.INDEX_FINGER_MCP].y -
                                              hand_landmarks.landmark[mp_hands.HandLandmark.PINKY_MCP].y)**2
                                         ) * 400)
                zRatio = 1 + (handCoordinatesZ / 30) #Hugmynd, gera 1 + (handcoordinates / 30)
                print(zRatio)
```

```

        print(zRatio)
        grabber = (65 - (round(math.sqrt(
            (hand_landmarks.landmark[mp_hands.HandLandmark.INDEX_FINGER_TIP].x -
            hand_landmarks.landmark[mp_hands.HandLandmark.THUMB_TIP].x)**2 +
            (hand_landmarks.landmark[mp_hands.HandLandmark.INDEX_FINGER_TIP].y -
            hand_landmarks.landmark[mp_hands.HandLandmark.THUMB_TIP].y)**2
        ) * 200))) * zRatio
    if(grabber < 0):
        grabber = 0
    mp_drawing.draw_landmarks(
        image,
        hand_landmarks,
        mp_hands.HAND_CONNECTIONS,
        mp_drawing_styles.get_default_hand_landmarks_style(),
        mp_drawing_styles.get_default_hand_connections_style())
    print(f"Hand coordinates X Y Z: {handCoordinatesX, handCoordinatesY, handCoordinatesZ}")
    print(f"Grabber motor pos: {grabber}")
    board.digital[3].write(grabber)
    try:
        x, y ,z = handCoordinatesX, handCoordinatesY, handCoordinatesZ
        q1, q2, q3 = EEZYbotARM_Mk2.inverseKinematics(robotArm, x, y, z)
        baseAngle, armAngle, vArmAngle, = EEZYbotARM_Mk2.map_kinematicsToServoAngles(robotArm
        , q1 = q1, q2 = q2, q3 = q3)
        print(baseAngle, armAngle, vArmAngle)
        board.digital[5].write(180 - baseAngle)
        board.digital[6].write(armAngle)
        board.digital[9].write(vArmAngle)
    except:
        print("Robot arm cannot reach coordinates")
        pass
    print("Robot arm cannot reach coordinates")
    pass

# Flip the image horizontally for a selfie-view display.
cv2.imshow('MediaPipe Hands', cv2.flip(image, 1))
if cv2.waitKey(1) & 0xFF == ord("q"):
    break
cap.release()

class EEZYbotARM:
    """
    --Description--
    This is a parent class for the EExybot Robotic arm

    **Please note that you must always instantiate on a child class (either EEZYbotARMMk2() or EEZYbotARMMk1())
    rather than instansiating the parent 'EEZYbotARM' class**

    --Methods--
    Description of available methods in this class:
    - __init__ --> Initialise a robotic arm with current joint position
    - updateJointAngles --> Update the stored joint angles for the robot arm
    - checkErrorJointLimits --> Check if any of the supplied joint angles are outside of physical limits
    - FK_EEZybotARM --> Perform forward kinematics to find the end effector position, given joint angles
    - IK_EEZybotARM --> Perform inverse kinematics to find the joint angles, given end effector position
    - plot_EEZybotARM --> Plot the robotic arm in 3D using matplotlib Axes 3D
    """

    # Class attribute
    # e.g. species = 'mammal'

    # Initializer / Instance attributes
    def __init__(self, initial_q1, initial_q2, initial_q3):
        self.q1 = initial_q1
        self.q2 = initial_q2
        self.q3 = initial_q3

```

```

def inverseKinematics(self, x_EE, y_EE, z_EE):
    """
    --Description--
    Function: to find the Inverse Kinematics for the EEZYbotARM Mk2

    Description: given x y z positions of the desired EE position inside the
    manipulator workspace, find the corresponding joint angles

    --Parameters--
    x_EE, y_EE, z_EE -> the cartesian position of the end effector in the world frame

    --Returns--
    q1, q2, q3 -> Corresponding joint angles in degrees

    """
    # DH parameters (Proximal Convention)
    L1 = self.L1
    L2 = self.L2
    L3 = self.L3
    L4 = self.L4

    # Find the value for the fist angle
    q1 = atan2(y_EE, x_EE)

    # Find the values for the position of joint #4 for x, y, z
    x_4 = x_EE - (L4 * cos(q1))
    y_4 = y_EE - (L4 * sin(q1))
    z_4 = z_EE

    # Find the length of the third side of the (virtual) triangle made by L2,
    # Find the length of the third side of the (virtual) triangle made by L2,
    # L3 in the vertical plane of the robot arm

    # --- Specify the z position of joint #1
    z_1 = L1

    # --- First we find the z distance between joint #1 and joint #4 (in the world frame)
    z_1_4 = z_4 - z_1

    # --- Find the true distance (in x, y plane) to joint #4
    xy_4 = sqrt((x_4**2)+(y_4**2))

    # --- Then we find the length of the virtual side made by the right angle triangle
    v_side = sqrt((z_1_4**2) + (xy_4**2))

    # Find the value for the angle at joint #3
    q3 = - (pi - acos((L2**2 + L3**2 - v_side**2)/(2 * L2 * L3)))

    # Find the value for the angle at joint #2 %DEBUG HERE
    # --- Find the value of the angle from the x-y plane to the virtual side
    q2_a = atan2(z_1_4, xy_4)

    q2_b = acos((v_side**2 + L2**2 - L3**2)/(2 * v_side * L2))

    q2 = q2_a + q2_b # NOTE there's some more work to do here to make this correctly summation or subtraction

    # Print the input world frame position of the end effector
    #print('Input Position of End Effector: \n')
    #print('x_EE: {}'.format(x_EE))
    #print('y_EE: {}'.format(y_EE))
    #print('z_EE: {} \n'.format(z_EE))

```

```

#print('z_EE: {} \n'.format(z_EE))

# Print the output joint angles
#print('Output joint angles: \n')
#print('q1: {:.2f}'.format(q1 * 180/pi))
#print('q2: {:.2f}'.format(q2 * 180/pi))
#print('q3: {:.2f} \n'.format(q3 * 180/pi))

# round values
q1 = round(q1 * 180/pi, 2)
q2 = round(q2 * 180/pi, 2)
q3 = round(q3 * 180/pi, 2)

return q1, q2, q3

```

class EEZYbotARM_Mk2(EEZYbotARM):

```

"""
--Description--
This is a child class for the EEzybot Robotic arm MK2 (inherits from EEZYbotARM() class)

**Please note that you must always instantiate on a child class (either EEZYbotARMMk2() or EEZYbotARMMk1())
rather than instansiating the parent 'EEZYbotARM' class**

--Methods--
Description of available methods in this class:
- q3CalcLimits --> Calculate the physical limits for joint 3 (because these depend on the angle of joint 2)
- map_kinematicsToServoAngles --> Map angles defined in kinematics to physical servo angles on the robot
"""

```

```

# Class attribute
# DH parameters (Proximal Convention)
L1 = 92 # mm
L2 = 135
L3 = 147
L4 = 87

# --- Lengths of links which attach to the hoarm
L2A = 60
LAB = 140
LB3 = 60

# Joint limits
q1_min = -30 # degrees
q1_max = 30
q2_min = 39
q2_max = 120
q2_max = 120
# q3_min, q3_max are given by q3CalcLimits() function

```

```

def q3CalcLimits(self, **kwargs):
"""
Calculate q3 angle limits for the EzzyBot Arm given a value for the angle q2 in degrees
These limits have been determined experimentally for the EEzybotMK2

If no q2 value is given then the current value of q2 is used

--Optional kwargs Parameters--
@q2 -> the value of the angle q2 (in degrees)

--Returns--
q3_min, q3_max -> the min and max limits for the angle q3 (in degrees)
"""

# Use **kwarg if provided, otherwise use current q2 value
q2 = kwargs.get('q2', self.q2)

# calculate q3 min limits in degrees
q3_min = (-0.6755 * q2) - 70.768
q3_max = (-0.7165 * q2) - 13.144

return q3_min, q3_max

```

```

    return q3_min, q3_max

def map_kinematicsToServoAngles(self, **kwargs):
    """
    --Description--
    Take the three angles for the EzzyBot Arm defined in kinematics.
    Use these to calculate the required physical servo position with respect to the reference position.
    If three angles are not provided as **kwargs then the current values for the arm are used

    The reference positions for the three servos are as follows:

    EzzyBot base (q1) : 90 degree servo position is facing directly forwards
    Main arm (q2): 90 degree servo position is with main arm perpendicular (at 90 degrees to) base
    Horarm (q3): 90 degree servo position is with horarm servo link at 45 degrees to base

    The function will be updated to raise an error message when any of the returned angles are outside of the

    --Optional **kwargs Parameters--
    @q1 -> the value of the angle q1 (in degrees) as used in the kinematics model
    @q2 -> the value of the angle q2 (in degrees) as used in the kinematics model
    @q3 -> the value of the angle q3 (in degrees) as used in the kinematics model

    --Returns--
    servoAngle_q1, servoAngle_q2, servoAngle_q3 -> values in degrees for output to the physical servos

    """

    # Use **kwargs if provided, otherwise use current values
    q1 = kwargs.get('q1', self.q1)
    q2 = kwargs.get('q2', self.q2)
    q3 = kwargs.get('q3', self.q3)
    q3 = kwargs.get('q3', self.q3)

    # Check none of the angles are outside of joint limits! So that servos cannot get damaged
    #self.checkErrorJointLimits(q1=q1, q2=q2, q3=q3)

    # Calculate for q1
    servoAngle_q1 = ((-2.0497)*q1) + 91.726 # from experimentation !
    servoAngle_q1 = round(servoAngle_q1, 2)

    # Calculate for q2
    servoAngle_q2 = 180 - q2 # approximate adjusted q2 value
    servoAngle_q2 = round(servoAngle_q2, 2)

    # Calculate for q3
    q3_a = 180 - (- q3) # approximate adjusted q3 value
    servoAngle_q3 = q2 - 45 + q3_a
    servoAngle_q3 = round(servoAngle_q3, 2)

    return servoAngle_q1, servoAngle_q2, servoAngle_q3

```