18 ΝΟΕΜΒΡΙΟΥ 2022

ΚΑΔΡΕΒΗ ΜΑΡΙΑ - ΕΛΕΝΗ

ΑΝΑΠΤΥΞΗ ΠΕΡΙΦΕΡΕΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ ΣΕ ΠΕΡΙΒΑΛΛΟΝ EMU8086

ΥΠΕΥΘΥΝΟΣ ΚΑΘΗΓΗΤΗΣ: Μ.ΔΑΣΥΓΕΝΗΣ

ΕΝΟΤΗΤΕΣ

ΕΙΣΑΓΩΓΗ

Στόχος της εργασίας αυτής είναι η αναλυτική περιγραφή των βημάτων που απαιτούνται για την δημιουργία μιας εικονικής συσκευής σε παραθυρικό περιβάλλον με χρήση γλώσσας python, η οποία θα μπορεί να επικοινωνεί με τον προσομοιωτή EMU8086.

Ως παράδειγμα θα σχεδιαστεί ένας ρομποτικός βραχίονας, τεσσάρων βαθμών ελευθερίας ο οποίος θα κινείται μέσω εντολών από ένα πρόγραμμα το οποίο θα τρέχει στον προσομοιωτή EMU8086.

ΕΠΙΚΟΙΝΩΝΙΑ ΜΕΤΑΞΥ EMU8086 ΚΑΙ ΣΥΣΚΕΥΗΣ

Ο EMU8086 επικοινωνεί με τις εικονικές συσκευές μέσω των θυρών 0-65535 με τις εντολές assembly IN και OUT.Για την προσομοίωση της λειτουργίας των θυρών με τον προσομοιωτή EMU8086 χρησιμοποιείται ένα αρχείο **C:\emu8086\emu8086.io** με 65536 byte όπου κάθε byte αντιστοιχεί και στην τιμή της αντίστοιχης θύρας. Το όνομα και η θέση του αρχείου προσδιορίζονται στο αρχείο EMU8086.ini.

EMUPORT=c:\emu8086\emu8086.io

Για παράδειγμα αν ο χρήστης επιθυμεί να διαβάσει την τιμή της θύρας 12 θα δώσει την εντολή

```
IN AL, 12      ; Διαβάζει την τιμή της θύρας 12 στον καταχωρητή AL
```

ενώ αν θέλει να γραψει στην τιμή 35 στην θύρα 12 θα δώσει τις εντολές

```
MOV AL, 35     ; τοποθετεί την τιμή 35 στον καταχωρητή AL
OUT 12, AL     ; γράφει την τιμή του AL στην θύρα 12
```

Αντίστοιχα η εικονική συσκευή θα πρέπει να μπορεί να διαβάσει ή να γράψει σε ένα από τα 65536 byte στο ίδιο αρχείο για να «ενημερώσει» την αντίστοιχη θύρα. Για τον σκοπό αυτό θα πρέπει να γραφτούν δύο βασικές ρουτίνες οι οποίες θα διαβάζουν και θα γράφουν ένα byte στην επιθυμητή θέση (θύρα) στο αρχείο **c:\emu8086\emu8086.io**

```
sIO_FILE = "C:\emu8086\emu8086.io"

def WRITE_IO_BYTE(lPORT_NUM, mybyte):
    f = open(sIO_FILE, "r+b")
    f.seek(lPORT_NUM)
    f.write(mybyte)
    f.close()

def READ_IO_BYTE(lPORT_NUM):
    mybyte = chr(0)
    f = open(sIO_FILE, "rb")
    f.seek(lPORT_NUM)
    mybyte = f.read(1)
    f.close()
    return mybyte
```

ΔΗΜΙΟΥΡΓΙΑ ΑΠΛΗΣ ΠΑΡΑΘΥΡΙΚΗΣ ΕΦΑΡΜΟΓΗΣ

Για την δημιουργία μιας παραθυρικής εφαρμογής με απλά γραφικά η python παρέχει μια πολύ εύχρηστη βιβλιοθήκη την **turtle** η οποία και θα χρησιμοποιηθεί για την σχεδίαση του ρομποτικού βραχίονα**.** Για να έχουμε πρόσβαση στην βιβλιοθήκη πρέπει να την εισάγουμε στον πρόγραμμά μας με την εντολή:

```
Import turtle
```

Για την δημιουργία ενός παράθυρου με συγκεκριμένες διαστάσεις και χρώμα χρησιμοποιώ τον παρακάτω κώδικα:

```
WIDTH = 600
HEIGHT = 400
wndw = turtle.Screen()                    #Δημιουργώ ένα παράθυρο wndw στην οθόνη
wndw.bgcolor("light green")               #ορίζω το χρώμα του υπόβαθρου
wndw.setup(width=WIDTH, height=HEIGHT)    #ορίζω το μεγεθος του παράθυρου
wndw.title("Robotic Arm")                 #ορίζω το ονομα του παράθυρου
wndw.tracer(0)                            #ορίζω τον ρυθμό ανανέωσης του παραθυρου
```

ΣΧΕΔΙΑΣΗ ΕΝΟΣ ΟΡΘΟΓΩΝΙΟΥ

Η βιβλιοθήκη turtle μας επιτρέπει την εύκολη σχεδίαση αντικειμένων με την λογική ενός μολυβιού το οποίο κινούμε διαδοχικά στις επιθυμητές διευθύνσεις και αποστάσεις είτε ακουμπώντας το στην επιφάνεια (γράφει) είτε όχι (μετακινείται χωρίς να γράφει). Ένα ορθογώνιο παραλληλόγραμμο που ξεκινά στην θέση x0, y0 έχει χρώμα **color,** πλάτος **width,** ύψος **height**, κα είναι γεμάτο **fill=1** ή άδειο **fill = 0**, μπορεί να σχεδιαστεί με τον παρακάτω κώδικα:

```
#create  an orthogonal staring at x0, y0
def make_box(color, x0, y0 ,width, height, fill) :

    t = turtle.Turtle()
    t.hideturtle()
    t.pensize(5)
    t.penup()
    t.goto(x0, y0)
    t.color(color)
    t.pendown()
    t.fillcolor(color)
    if (fill == 1): t.begin_fill()

    # drawing first side
    t.forward(width) # Forward turtle by width units
    t.right(90)      # Turn turtle by 90 degree

    # drawing second side
    t.forward(height) # Forward turtle by height units
    t.right(90)       # Turn turtle by 90 degree

    # drawing third side
    t.forward(width) # Forward turtle by width units
    t.right(90)      # Turn turtle by 90 degree

    # drawing fourth side
    t.forward(height) # Forward turtle by height units
    t.right(90)       # Turn turtle by 90 degree
```

```
    if (fill==1): t.end_fill()
```

ΚΑΘΟΡΙΣΜΟΣ ΘΕΣΗΣ ΤΟΥ ΟΡΘΟΓΩΝΙΟΥ

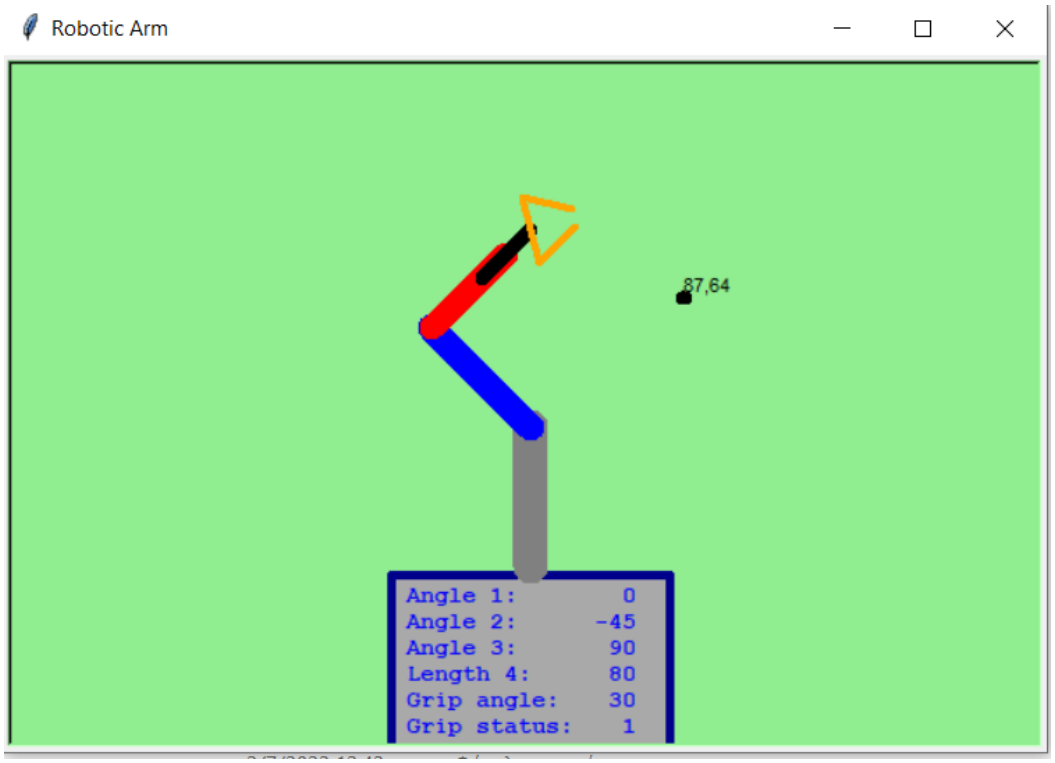
Αν θέλουμε να διαβάσουμε την αρχική θέση του ορθογώνιου μέσω των θυρών 11 και 12 (οι τιμές θα δίνονται από κώδικα που τρέχει στο EMU8086, θα πρέπει να χρησιμοποιήσουμε την ρουτίνα READ_IO_BYTE για να διαβάσουμε την τιμή από την αντίστοιχη θέση στο αρχείο EMU8086.io

```
Import turtle

WIDTH = 600
HEIGHT = 400
wndw = turtle.Screen()                  #Δημιουργώ ένα παράθυρο wndw στην οθόνη
wndw.bgcolor("light green")             #ορίζω το χρώμα του υπόβαθρου
wndw.setup(width=WIDTH, height=HEIGHT) #ορίζω το μεγεθος του παράθυρου
wndw.title("Robotic Arm")               #ορίζω το ονομα του παράθυρου
wndw.tracer(0)                          #ορίζω τον ρυθμό ανανέωσης του παραθυρου


x0 = READ_IO_BYTE(11)
y0 = READ_IO_BYTE(12)
color = "white"
width = 50
height = 50
fill = 1
make_box(color, x0, y0 ,width, height, fill)
```

ΔΟΜΗ ΤΟΥ ΡΟΜΠΟΤΙΚΟΥ ΒΡΑΧΙΟΝΑ

Ο ρομποτικός βραχίονας αποτελείται από:

- Τρία τμήματα APM1 (γκρίζο), APM2(μπλε), APM3(κόκκινο) στα οποία μπορούν να καθοριστούν το μήκος και τα όρια της γωνίας περιστροφής του καθενός
- Ένα τμήμα APM4 (μαύρο) στο οποίο μπορεί να καθοριστεί το μήκος και η μετατόπιση σχετικά με το τμήμα APM3 (εμβολοειδής κίνηση)
- Μια αρπάγη (GRIP – πορτοκαλί) στην οποία μπορούν να καθοριστούν το μήκος, η γωνία ανοικτή-κλειστή και η γωνία περιστροφής σχετικά με τον τρίτο άξονα.



Τα μήκη των τμημάτων APM1 και APM2 APM3 και APM4 μπορούν να αλλάξουν μόνο με αλλαγή του κώδικα και δημιουργία νέου εκτελέσιμου. Σε μελλοντική έκδοση θα μπορούσαν είτε να διαβάζονται ως παράμετροι από ένα αρχείο είτε να καθορίζονται προγραμματιστικά από τον χρήστη.

Ομοίως και τα όρια περιστροφής του κάθε τμήματος και τα χαρακτηριστικά της αρπάγης μπορούν να αλλάξουν μόνο με αλλαγή του κώδικα.

Οι γωνίες περιστροφής και η μετακίνηση του APM4 είναι παράμετροι που καθορίζονται από τον χρήστη και την επακόλουθη μετακίνηση του βραχίονα.

ΠΡΟΓΡΑΜΜΑΤΙΖΟΜΕΝΕΣ ΠΑΡΑΜΕΤΡΟΙ

Ο ρομποτικός βραχίονας διαβάζει τις προγραμματιζόμενες παραμέτρους από συγκεκριμένες θύρες όπως καταγράφεται παρακάτω:

Port 11	γωνία του APM1 σε μοίρες με εύρος τιμών (-60,60) – *Ο χρήστης μπορεί να εγγράψει μια τιμή από -128 ως 127 και μετά την ολοκλήρωση της κίνησης ο βραχίονας εγγράφει στην θύρα την τελική γωνία περιστροφής του APM1. Αν η τιμή που δίνει ο χρήστης είναι εκτός των ορίων που έχουν προσδιοριστεί στον κώδικα η τελική θέση θα είναι το όριο της συγκεκριμένης παραμέτρου.*	1 byte

Port 12	γωνία του ΑΡΜ2 σε μοίρες με εύρος τιμών (-45,45) *Ισχύει ότι και στην Θύρα 11*	1 byte
Port 13	γωνία του ΑΡΜ3 σε μοίρες με εύρος τιμών (-125,125) *Ισχύει ότι και στην Θύρα 11*	1 byte
Port 14	μήκος του ΑΡΜ3 με εύρος τιμών (60-100) *Ισχύει ότι και στην Θύρα 11*	1 byte
Port 15	γωνία περιστροφής αρπάγης σε μοίρες με εύρος τιμών (-60,60) *Ισχύει ότι και στην Θύρα 11*	1 byte
Port 16	κατάσταση αρπάγης 1 ανοιχτή και 0 κλειστή	1 byte

Εκτός από τις παραμέτρους που καθορίζουν την θέση του βραχίονα, υπάρχουν και άλλες παράμετροι που βοηθούν στην επικοινωνία του χρήστη με τον βραχίονα οι οποίες είναι:

Port 20	Κατάσταση ρομποτικού βραχίονα Ο έτοιμος, 1 απασχολημένος (μόνο ανάγνωση από τον χρήστη)	1 byte
Port 21	Τερματισμός λειτουργίας ρομποτικού 0 ενεργό 1 τερματισμός – εγγραφή από χρήστη.	1 byte
Port 30 + 31	Συντεταγμένη x του αντικειμένου – μόνο για ανάγνωση από χρήστη	2 byte
Port 32 + 33	Συντεταγμένη y του αντικειμένου – μόνο για ανάγνωση από χρήστη.	2 byte

Για την προσομοίωση της λειτουργίας των θυρών με τον προσομοιωτή EMU8086 χρησιμοποιείται ένα αρχείο **C:\emu8086\emu8086.io** με 65536 byte όπου κάθε byte αντιστοιχεί και στην τιμή της αντίστοιχης θύρας. Για παράδειγμα αν ο χρήστης γράψει την τιμή 25 στο byte 13 του αρχείου σημαίνει ότι ο ΑΡΜ3 θα περιστραφεί σε γωνία 25 μοιρών δεξιόστροφα σχετικά με τον ΑΡΜ2. Αν ο χρήστης επιθυμεί να διαβάσει την τρέχουσα γωνία του ΑΡΜ1 θα διαβάσει το byte 11 από το ίδιο αρχείο.

ΣΧΕΔΙΑΣΗ ΡΟΜΠΟΤΙΚΟΥ ΒΡΑΧΙΟΝΑ

Όπως και στο παράδειγμα του ορθογώνιου, ο βραχίονας μπορεί να σχεδιαστεί με εντολές της βιβλιοθήκης turtle. Οι τιμές για την θέση κάθε τμήματος του βραχίονα μπορούν να διαβάζονται από τις θύρες που αναγράφονται στην προηγούμενη ενότητα χρησιμοποιώντας την ρουτίνα **READ_IO_BYTE()**

```
#length of robotic arms
arm1_l = 80
arm2_l = 80
arm3_l = 60
arm4_l = 40
```

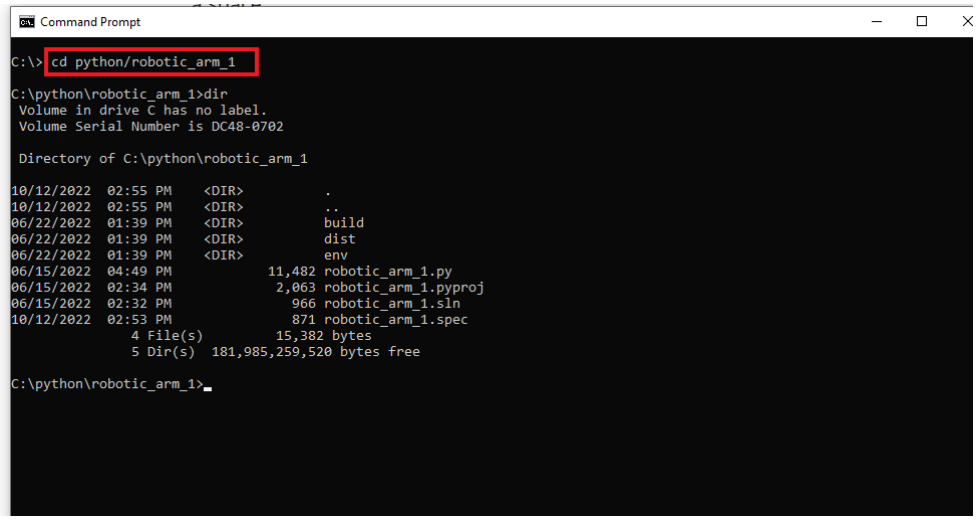
```
# Each tuple in list arms describes the color, the length, and the angle range of
the arm
arm1 = ["grey", arm1_l, -60,60]
arm2 = ["blue", arm2_l, -45, 45]
arm3 = ["red", arm3_l,  -125, 125, arm4_l]

#Read robotic arm parameters
……

#Draw robotic arm

# Create the drawing pen for the robotic arms
pen = turtle.Turtle()
pen.hideturtle()
pen.speed(0)

def draw_robotic_arm(ang1, ang2, ang3, a4_l, pen):

    #draw first arm

    pen.pensize(20)
    pen.penup()
    pen.goto(0, -90)
    pen.color(arm1[0])
    pen.setheading(90)
    pen.rt(ang1)
    pen.pendown()
    pen.fd(arm1[1])

    #draw second arm
    pen.pensize(16)
    pen.penup()
    pen.color(arm2[0])
    pen.rt(ang2)
    pen.pendown()
    pen.fd(arm2[1])


    #draw third arm

    pen.pensize(14)
    pen.penup()
    pen.color(arm3[0])
    pen.rt(ang3)
    pen.pendown()
    pen.fd(arm3[1])

    pen.pensize(8)
    pen.penup()
    pen.color("black")
    pen.backward((arm3_l+arm4_l)-a4_l)

    pen.pendown()

    pen.fd(arm4_l)
```

ΜΕΤΑΤΡΟΠΗ ΑΡΧΕΙΟΥ .py ΣΕ ΕΚΤΕΛΕΣΙΜΟ

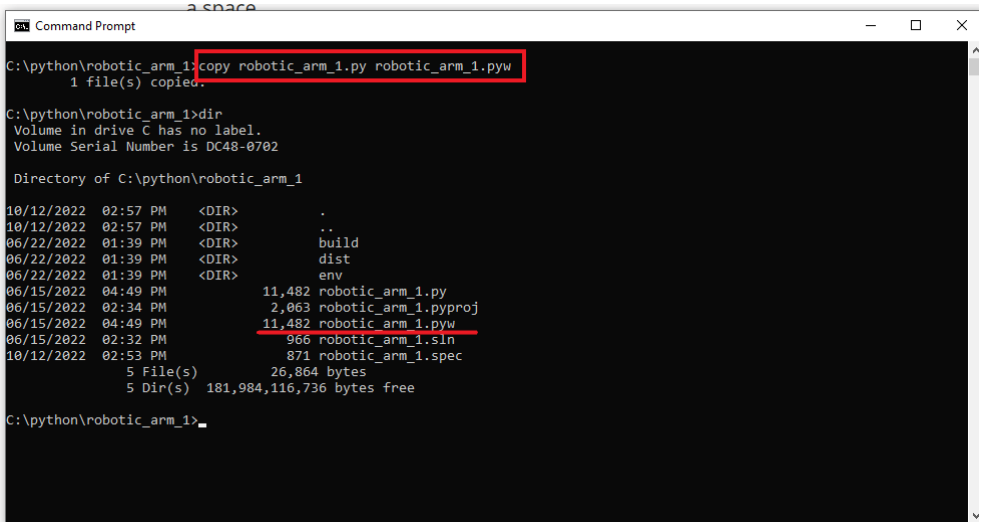
Αναλυτική επεξήγηση μετατροπής ενός αρχείου python σε εκτελέσιμο με την χρήση παραδείγματος.

Για την δημιουργία εκτελέσιμου:

1. Ανοίγουμε ένα command window.
2. Πηγαίνουμε στο φάκελο όπου υπάρχει το αρχείο robotic_arm_1.py

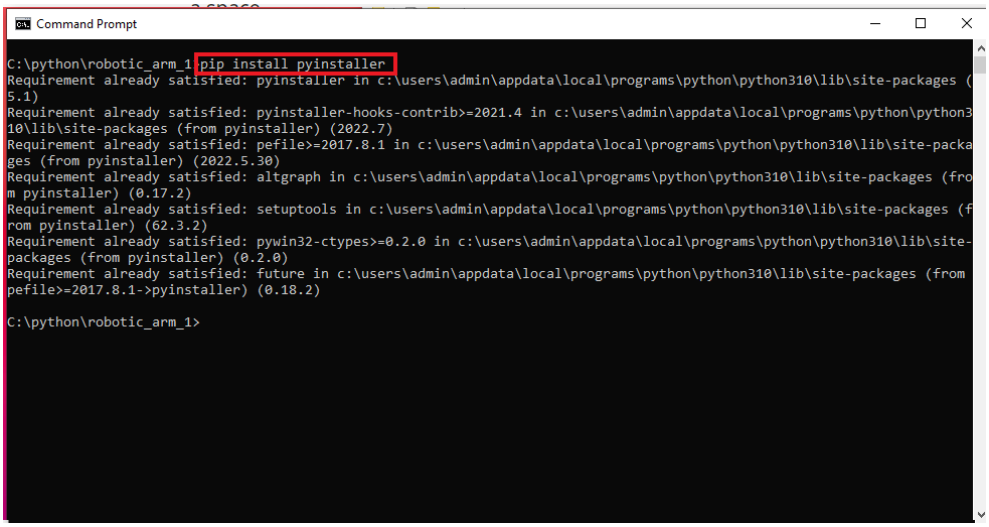


3. Το αντιγράφουμε σε ένα αρχείο με επέκταση *.pyw για να δημιουργηθεί εκτελέσιμο χωρίς να ανοίγει και το command window

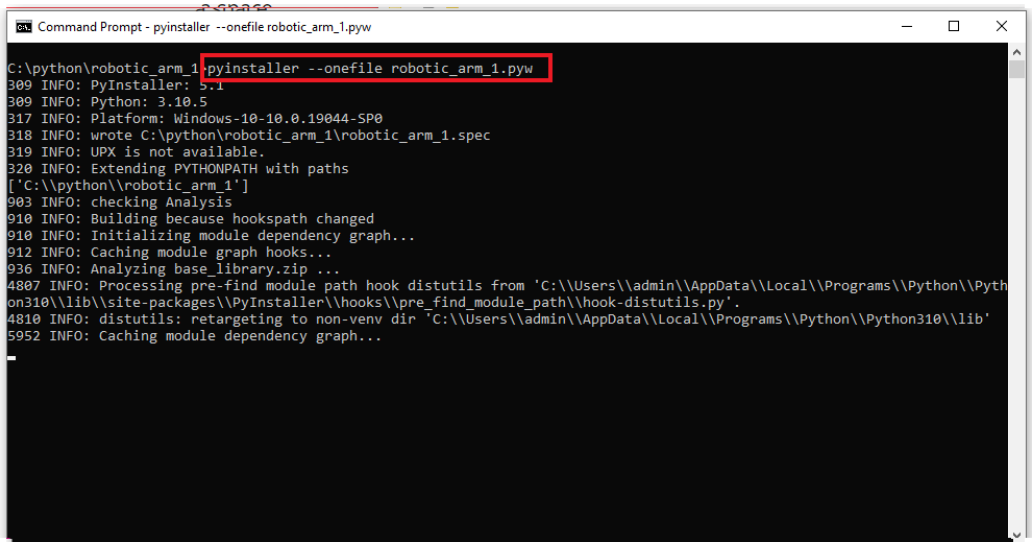


4. Πηγαίνουμε στο φάκελο με το *.pyw αρχείο

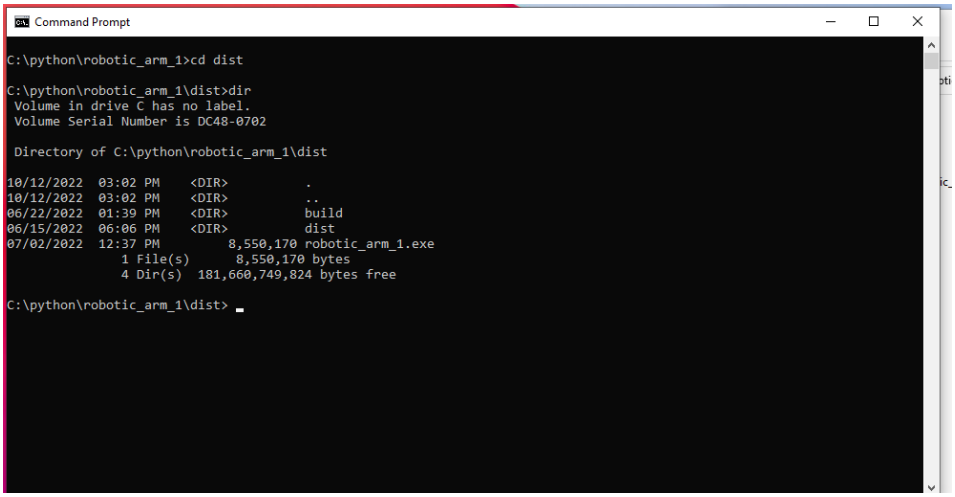
5. Εκτελούμε την εντολή
   - pip installer pyinstaller – Εάν δεν έχουν το pyinstaller εγκατεστημένο.



Εάν το έχουμε εκτελούμε την παρακάτω εντολή
   - Pyinstaller –-onefile robotic_arm_1.pyw

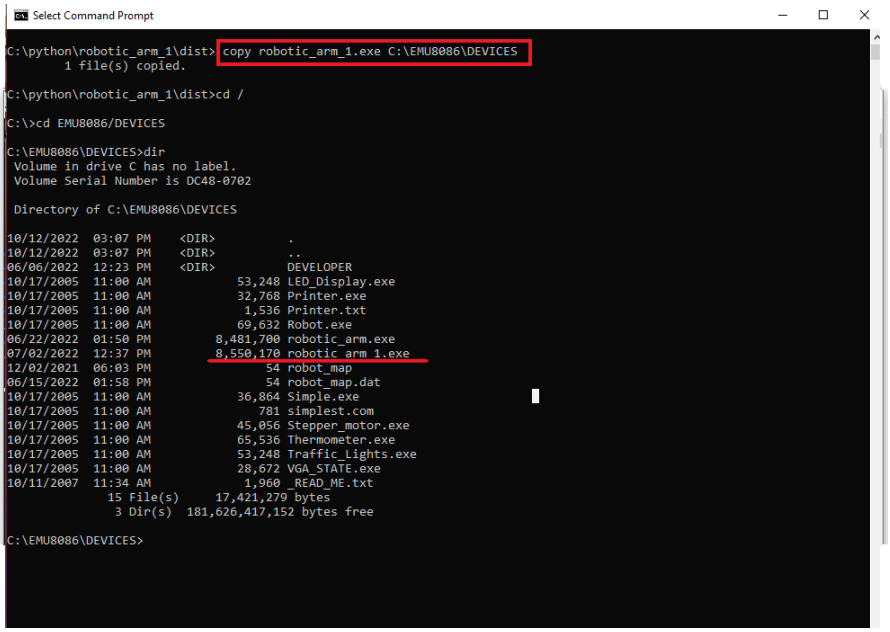


6. Το εκτελέσιμο θα δημιουργηθεί σε ένα υποφάκελο dist μέσα στο φάκελο του project



Εάν θέλουμε να δημιουργήσουμε 32bit exe θα πρέπει να έχουμε κατεβασμένη την έκδοση 3.7 32bit και το PATH για το pyinstaller να είναι στο περιβάλλον της 3.7 python.

7. Αντιγραφή του εκτελέσιμου στο φάκελο devices του EMU8086



ΑΝΑΛΥΤΙΚΟ ΠΑΡΑΔΕΙΓΜΑ

Παράδειγμα δημιουργίας και προγραμματισμού ενός περιφερειακού για το emu8086

ΖΗΤΟΥΜΕΝΟ:

Να δημιουργηθεί emu8086 peripheral (visual basic) που θα εμφανίζεται ένας ρομποτικός βραχίονας 4 DOF και θα μετακινείται ανάλογα με τις εντολές IN/OUT σε συγκεκριμένες θύρες.

ΑΝΑΛΥΤΙΚΗ ΕΠΕΞΗΓΗΣΗ ΤΟΥ ΚΩΔΙΚΑ:

Γίνεται εισαγωγή της βιβλιοθήκης **turtle** η οποία υπάρχει μέσα την python και αποτελεί τον πυρήνα των γραφικών του προγράμματος.

sIO_FILE = "C:\emu8086\emu8086.io" : εξωτερικό αρχείο με το οποίο γίνεται η σύνδεση το προγράμματος της python με το emu8086

```
import turtle
import time                                    #library for time

sIO_FILE = "C:\emu8086\emu8086.io"
```

1. **ΡΟΜΠΟΤΙΚΟΣ ΒΡΑΧΙΟΝΑΣ**

ΟΡΙΣΜΟΣ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ ΡΟΜΠΟΤΙΚΟΥ ΒΡΑΧΙΟΝΑ

18-21: Αρχικοποίηση κάθε τμήματος του ρομποτικού βραχίονα. Ορίζονται το χρώμα και το μήκος του τμήματος καθώς και τα όρια της γωνίας περιστροφής του κάθε τμήματος του ρομποτικού βραχίονα και αυτό διότι θέλαμε να κάνουμε πιο ρεαλιστικό το πρόγραμμα (ένας βραχίονας δεν μπορεί να κάνει οποιαδήποτε κίνηση)

24-30: Ορισμός Αρχικής θέσης κάθε τμήματος του ρομποτικού βραχίονα. Με **new_** ορίζονται οι παράμετροι που εισάγει ο χρήστης και **cur_** είναι οι τρέχουσα τιμή της κάθε παραμέτρου.

```
arm1 = ["grey", arm1_l, -60,60]
arm2 = ["blue", arm2_l, -45, 45]
arm3 = ["red", arm3_l,  -125, 125, arm4_l]
grip = ["blue", 30, -60, 60]

#starting robotic arm position
new_angle1 = cur_angle1 = 0
new_angle2 = cur_angle2 = -45
new_angle3 = cur_angle3 = 90
new_length4 = cur_length4 = int(arm3_l + (arm4_l/2))
new_grip_angle = cur_grip_angle = 30
new_grip_status = cur_grip_status = 1
terminate = False
```

ΡΟΥΤΙΝΑ ΓΙΑ ΣΧΕΔΙΑΣΗ ΑΡΠΑΓΗΣ

239-268: Κώδικας για τον σχεδιασμό της αρπάγης. Αφού ελεγχθεί η επιθυμητή γωνία ως προς τα όρια κίνησης της αρπάγης με την βοήθεια των εντολών που υπάρχουν στην βιβλιοθήκη **turtle** σχεδιάζεται η αρπάγη ανοιχτή ή κλειστή σύμφωνα με την μεταβλητή status

```
    #truncate grip_angle
    if grip_angle < grip[2] : grip_angle = grip[2]
    elif grip_angle > grip[3] : grip_angle = grip[3]

    pen.pensize(4)
    if status==0:  # grip open
        pen.penup()
        pen.color(grip[0])
        pen.rt(grip_angle+90)
        pen.pendown()
        pen.fd(20)
        pen.left(90)
        pen.fd(grip[1])
        pen.back(grip[1])
        pen.lt(90)
        pen.fd(40)
        pen.rt(90)
        pen.fd(grip[1])
    else:        #grip closed
        pen.penup()
        pen.color(grip[0])
        pen.rt(grip_angle+90)
        pen.pendown()
        pen.fd(20)
        pen.left(120)
        pen.fd(grip[1])
        pen.back(grip[1])
        pen.lt(60)
        pen.fd(40)
        pen.rt(120)
        pen. fd(grip[1])
```

ΡΟΥΤΙΝΑ ΓΙΑ ΣΧΕΔΙΑΣΜΟ ΒΡΑΧΙΟΝΑ

271-317: Ρουτίνα για τον Σχεδιασμό του ρομποτικού βραχίονα σύμφωνα με τις επιθυμητές παράμερους. Μετά το πέρας του σχεδιασμού ενημερώνονται οι παράμετροι **cur_** στις οποίες είναι αποθηκευμένη η τρέχουσα κατάσταση του βραχίονα.

```
def draw_robotic_arm (ang1, ang2, ang3, a4_l, gr, grip_angle, pen):

    global cur_angle1, cur_angle2, cur_angle3, cur_length4,
cur_grip_angle, cur_grip_status
    #Draw first arm

    pen.pensize(20)
    pen.penup()
    pen.goto(0, -90)
    pen.color(arm1[0])
    pen.setheading(90)
    pen.rt(ang1)
    pen.pendown()
    pen.fd(arm1[1])

    #draw second arm
    pen.pensize(16)
    pen.penup()
    pen.color(arm2[0])
    pen.rt(ang2)
    pen.pendown()
    pen.fd(arm2[1])


    #draw third arm

    pen.pensize(14)
    pen.penup()
    pen.color(arm3[0])
    pen.rt(ang3)
    pen.pendown()
    pen.fd(arm3[1])
```

```
    pen.pensize(8)
    pen.penup()
    pen.color("black")
    pen.backward((arm3_l+arm4_l)-a4_l)
    pen.pendown()
```

```
    pen.fd(arm4_l)

    draw_grip(gr, grip_angle, pen)

    cur_angle1 = ang1
    cur_angle2 = ang2
    cur_angle3 = ang3
    cur_length4 = a4_l
    cur_grip_angle = grip_angle
    cur_grip_status = gr
```

ΑΝΑΛΥΣΗ ΤΩΝ ΜΕΘΟΔΩΝ pen.

Για τον σχεδιασμό των γραφικών του βραχίονα χρησιμοποιείται όπως είπαμε και παραπάνω η βιβλιοθήκη turtle όπου μέσω αυτής χρησιμοποιείται το pen (το στυλό) όπου με αυτό καλούνται διάφοροι χρήσιμοι μέθοδοι για τον σχεδιασμό.

Χρήσιμες μέθοδοι της βιβλιοθήκης turtle με την χρήση pen. (οι οποίοι χρησιμοποιούνται και στο παράδειγμά μας παραπάνω) :

1. Pen.penup() / pen.pendowm() : Σηκώνει το στυλό ώστε η χελώνα να μην τραβήξει γραμμή καθώς κινείται (Pen.penup()) και αντίστοιχα κατεβάζει το στυλό ώστε η χελώνα να τραβήξει γραμμή καθώς κινείται.
2. Pen.pensize() : Αυτή η μέθοδος χρησιμοποιείται για τον καθορισμό ή την επιστροφή του πάχους γραμμής.
3. Pen.color() : Αυτή η μέθοδος χρησιμοποιείται για την αλλαγή του χρώματος του μελανιού του σχεδίου του στυλό. Το προεπιλεγμένο χρώμα είναι το μαύρο.
4. Pen.goto() : Αυτή η μέθοδος χρησιμοποιείται για να μετακινήσει ο στυλό σε απόλυτη θέση που εμείς ορίζουμε μέσα στην παρένθεση.
5. Pen.backward() : Αυτή η μέθοδος χρησιμοποιείται για να μετακινήσει το στυλό προς τα πίσω με βάση τη τιμή του ορίσματος που παίρνει.
6. Pen.fd() : Αυτή η μέθοδος μετακινεί τη χελώνα προς τα εμπρός κατά μία ορισμένη απόσταση.
7. Pen.rt() : Η μέθοδος αυτή κινεί το στυλό δεξιά κατά μονάδες γωνίας που ορίζονται ως παράμετροι.
8. Pen.setheading() : Η μέθοδος αυτή ορίζει τον προσανατολισμό της χελώνας σε γωνία.

2. **ΕΜΦΑΝΙΣΗ ΠΑΡΑΜΕΤΡΩΝ ΣΤΗΝ ΟΘΟΝΗ**

ΡΟΥΤΙΝΕΣ ΓΙΑ ΤΗΝ ΕΜΦΑΝΙΣΗ ΤΡΕΧΟΥΣΑΣ ΚΑΤΑΣΤΑΣΗΣ ΤΟΥ ΒΡΑΧΙΟΝΑ ΣΤΗΝ ΟΘΟΝΗ

Χρησιμοποιούνται δύο ρουτίνες για την εμφάνιση των παραμέτρων στην βάση του ρομποτικού βραχίονα. Μια για την εμφάνιση των ονομάτων και μια για την εμφάνιση των τιμών. Μόνο οι τιμές αλλάζουν συνεχώς ενώ τα ονόματα μόνο όταν γίνεται ανανέωση όλου του παραθύρου. Έτσι επιταχύνεται η λειτουργία του βραχίονα και βελτίωση του οπτικού αποτελέσματος.

```
def show_parameter_names():
    x0=-70
    y0=-115
    step = 15
    t1.clear()
    t1.ht()
    t1.color('blue')

    t1.penup()
    t1.goto(x0, y0)
    t1.pendown()
    t1.write("Angle 1:    ", font=("Courier", 10, "bold") )

    t1.penup()
    t1.goto(x0, y0-step)
    t1.pendown()
    t1.write("Angle 2:    " , font=("Courier", 10, "bold"))

    t1.penup()
    t1.goto(x0, y0-2*step)
    t1.pendown()
    t1.write("Angle 3:    " , font=("Courier", 10, "bold"))

    t1.penup()
    t1.goto(x0, y0-3*step)
    t1.pendown()
    t1.write("Length 4:   " , font=("Courier", 10, "bold"))

    t1.penup()
    t1.goto(x0, y0-4*step)
    t1.pendown()
    t1.write("Grip angle: " , font=("Courier", 10, "bold"))

    t1.penup()
    t1.goto(x0, y0-5*step)
    t1.pendown()
    t1.write("Grip status:" , font=("Courier", 10, "bold"))
```

```
 t1.penup()
    t1.goto(x0, y0-5*step)
    t1.pendown()
    t1.write("Grip status:" , font=("Courier", 10, "bold"))

def show_parameter_values():

    x0=30
    y0=-115
    step = 15

    t2.clear()
    t2.ht()
    t2.color('blue')
    t2.penup()
    t2.goto(x0, y0)
    t2.pendown()
    t2.write( '%4s' %(cur_angle1), font=("Courier", 10, "bold"))

    t2.penup()
    t2.goto(x0, y0-step)
    t2.pendown()
    t2.write( '%4s' %(cur_angle2), font=("Courier", 10, "bold"))

    t2.penup()
    t2.goto(x0, y0-2*step)
    t2.pendown()
    t2.write('%4s' %(cur_angle3), font=("Courier", 10, "bold"))

    t2.penup()
    t2.goto(x0, y0-3*step)
    t2.pendown()
    t2.write('%4s' %(cur_length4), font=("Courier", 10, "bold"))

    t2.penup()
    t2.goto(x0, y0-4*step)
    t2.pendown()
    t2.write('%4s' %(cur_grip_angle), font=("Courier", 10, "bold"))

    t2.penup()
    t2.goto(x0, y0-5*step)
    t2.pendown()
    t2.write('%4s' %(cur_grip_status), font=("Courier", 10, "bold"))
```

Ρουτίνα για τον σχεδιασμό παραλληλόγραμμου (με γέμιση ή όχι)

```
def make_box(color, x0, y0 ,width, height, fill) :
    #Draw the base of the robotic arm (grey orthogonal(100X200))
    t = turtle.Turtle()
    t.hideturtle()
    t.pensize(5)
    t.penup()
    t.goto(x0, y0)
    t.color(color)
    t.pendown()
    t.fillcolor(color)
    if (fill == 1): t.begin_fill()

    # drawing first side
    t.forward(width) # Forward turtle by width units
    t.right(90) # Turn turtle by 90 degree

    # drawing second side
    t.forward(height) # Forward turtle by height units
    t.right(90) # Turn turtle by 90 degree

    # drawing third side
    t.forward(width) # Forward turtle by width units
    t.right(90) # Turn turtle by 90 degree

    # drawing fourth side
    t.forward(height) # Forward turtle by height units
    t.right(90) # Turn turtle by 90 degree

    if (fill==1): t.end_fill()
```

3. ΑΝΤΙΚΕΙΜΕΝΟ

ΚΑΤΑΓΡΑΦΗ ΑΝΤΙΚΕΙΜΕΝΟΥ ΣΤΙΣ ΘΥΡΕΣ (30,31,32,33)

319-325: Ρουτίνα για την καταχώρηση συντεταγμένων του αντικειμένου που ορίζεται με το ποντίκι στον χωρο του βραχίονα. Χρησιμοποιουνται δύο προσημασμένοι 16μπιτοι αριθμοί για τις συντεταγμένες x και y.

```
def write_item_to_port(x,y):
    x1 = (x).to_bytes(2,'little',signed = True)
    y1 = (y).to_bytes(2,'little',signed = True)
    WRITE_IO_BYTE(30,(x1[0]).to_bytes(1,'little'))
    WRITE_IO_BYTE(31,(x1[1]).to_bytes(1,'little'))
    WRITE_IO_BYTE(32,(y1[0]).to_bytes(1,'little'))
    WRITE_IO_BYTE(33,(y1[1]).to_bytes(1,'little'))
```

ΣΧΕΔΙΑΣΗ ΚΑΙ ΚΑΘΑΡΙΣΜΟΣ ΑΝΤΙΚΕΙΜΕΝΟΥ

328-355: Σχεδιασμός αντικειμένου place_object(x,y) και διαγραφή αντικειμένου clear_objects(x,y).

```
t3 = turtle.Turtle()
t3.hideturtle()

def place_object(x,y):
    global t3,item
    if (item[0] == True):
        t3.clear()

    item[0] = True
    item[1] = int(x)
    item[2] = int(y)
    t3.hideturtle()
    t3.penup()
    t3.goto(x,y)
    t3.rt(180)
    t3.begin_fill()
    t3.circle(4)
    t3.end_fill()
    t3.write(str(int(x)) + ',' + str(int(y)))
    write_item_to_port(item[1], item[2])


def clear_objects(x,y):
    global t3,item
    t3.clear()
    item[0] = False
```

```
    item[1] = 0
    item[2] = 0
```

4. **ΠΑΡΑΜΕΤΡΟΙ**

114-118: Εγγραφή ενός Byte σε εξωτερικό αρχείο

120-126: Ανάγνωση ενός byte από εξωτερικό αρχείο

120-134: Ανάγνωση όλων των παραμέτρων από το εξωτερικό αρχείο

```
def READ_IO_BYTE( lPORT_NUM):
    mybyte = chr(0)
    f = open(sIO_FILE, "rb")
    f.seek(lPORT_NUM)               #goes to the position of where we want
to read
    mybyte = f.read(1)
    f.close()
    return mybyte

def READ_IO_PARAMETERS( lPORT_NUM):
    f = open(sIO_FILE, "rb")
    f.seek(lPORT_NUM)
    myparams = f.read(7)
    f.close()
    #print(myparams)
    return myparams
```

138-157: Μετατροπή το παραμέτρων σε μη προσημασμένα byte (0-255) και αποθήκευση στο εξωτερικό αρχείο για να είναι προσβάσιμες από τον κώδικα που τρέχει στον προσομοιωτή EMU8086.

```
def update_position():
    if cur_angle1 < 0: temp = cur_angle1 + 256
    else: temp = cur_angle1
    WRITE_IO_BYTE(11,(temp).to_bytes(1, "little"))

    if cur_angle2 < 0: temp = cur_angle2 + 256
    else: temp = cur_angle2
    WRITE_IO_BYTE(12,(temp).to_bytes(1, "little"))

    if cur_angle3 < 0: temp = cur_angle3 + 256
    else: temp = cur_angle3
    WRITE_IO_BYTE(13,(temp).to_bytes(1, "little"))
```

```
    WRITE_IO_BYTE(14,(cur_length4).to_bytes(1, "little"))

    if cur_grip_angle < 0: temp = cur_grip_angle + 256
    else: temp = cur_grip_angle
    WRITE_IO_BYTE(15,(temp).to_bytes(1, "little"))

    WRITE_IO_BYTE(16,(cur_grip_status).to_bytes(1, "little"))
```

161-198 : Ανάγνωση παραμέτρων από εξωτερικό αρχείο. Σύγκριση και έλεγχος τιμών και περικοπή εντός των επιτρεπτών ορίων.

```
def read_new_position():
    global new_angle1, new_angle2, new_angle3, new_length4,
new_grip_angle, new_grip_status

    temp = READ_IO_PARAMETERS(11)
    new_angle1 = (temp[0])
    new_angle2 = (temp[1])
    new_angle3 = (temp[2])
    new_length4 = (temp[3])
    new_grip_angle = (temp[4])
    new_grip_status = (temp[5])

    #print(temp[0], temp[1],temp[2],temp[3],temp[4],temp[5])

    #print(new_angle1, new_angle2, new_angle3, new_length4,
new_grip_angle, new_grip_status)

    # convert parameters to signed intergers
    if new_angle1 >128: new_angle1 = new_angle1-256
    if new_angle2 >128: new_angle2 = new_angle2-256
    if new_angle3 >128: new_angle3 = new_angle3-256
    if new_grip_angle >128: new_grip_angle = new_grip_angle-256

    #truncate parameters to arm bounds
    if (new_angle1 < arm1[2]):  new_angle1 = arm1[2]
    elif(new_angle1 > arm1[3]): new_angle1 = arm1[3]

    if (new_angle2 < arm2[2]):   new_angle2 = arm2[2]
    elif(new_angle2 > arm2[3]):  new_angle2 = arm2[3]

    if (new_angle3 < arm3[2]):   new_angle3 = arm3[2]
    elif(new_angle3 > arm3[3]):  new_angle3 = arm3[3]
```

```
    if (new_length4 < arm3[1]):   new_length4 = arm3[1]
    elif(new_length4 > (arm3[1]+arm3[4])):  new_length4 =
(arm3[1]+arm3[4])

    if (new_grip_angle < grip[2]):    new_grip_angle = grip[2]
    elif(new_grip_angle > grip[3]):   new_grip_angle = grip[3]

    if (new_grip_status != 0): new_grip_status = 1
```

5.ΛΕΙΤΟΥΡΓΙΑ ΒΡΑΧΙΟΝΑ

359-363: Σχεδιασμός οθόνης (background) προγράμματος

```
#setup  window for robotic arm
wndw = turtle.Screen()                                  #makes a screen from
library turtle with name wndw
wndw.bgcolor("light green")                                   #background
color
wndw.setup(width=600, height=400)                       #size of window
wndw.title("Robotic Arm")                               #window name
wndw.tracer(0)                                          #set the refresh time
```

367-370: Δημιουργία βάσης του Ρομποτικού Βραχίονα

373-375: Δημιουργία pen για την σχεδίαση των γραφικών

```
# draw the base of the robotic arm
make_box("dark gray", -80, -95, 160, 100, 1)


make_box("dark blue", -80, -95, 160, 100, 0)

# Create the drawing pen for the robotic arms
pen = turtle.Turtle()
pen.hideturtle()
pen.speed(0)
```

381-386: Εμφάνιση αρχικών παραμέτρων και αρχικοποίηση του ρομποτικού βραχίονα με τις αρχικές τιμές.

```
write_item_to_port(0,0)
user_exit = 0
WRITE_IO_BYTE(21,(0).to_bytes(1,"little"))

show_parameter_names()
```

```
show_parameter_values()
```

388-394: Σχεδίαση του ρομποτικού βραχίονα με τις τρέχουσες τιμές που δίνει ο χρήστης

```
draw_robotic_arm(cur_angle1,cur_angle2,cur_angle3, cur_length4,
cur_grip_status, cur_grip_angle, pen)
update_position()
wndw.update()
time.sleep(3)

show_parameter_names()
show_parameter_values()
```

395-τελος: Ενημέρωση της θέσης του ρομποτικού βραχίονα σύμφωνα με τις νέες παραμέτρους και εμφάνιση αντικειμένου αν υπάρχει. Η ενημέρωση γίνεται κάθε 100 ms

```
try:
    while (user_exit == 0):
        read_new_position()

    if (cur_angle1 != new_angle1) or (cur_angle2 != new_angle2)
or  (cur_angle3 != new_angle3) or (cur_length4 != new_length4) or
(cur_grip_status != new_grip_status) or (cur_grip_angle !=
new_grip_angle):
        pen.clear()
        WRITE_IO_BYTE(20,(1).to_bytes(1, "little"))
        draw_robotic_arm(new_angle1,new_angle2, new_angle3,
new_length4,new_grip_status, new_grip_angle, pen)
        update_position()


    show_parameter_values()
    wndw.update()
    WRITE_IO_BYTE(20,(0).to_bytes(1,"little"))
    user_exit = int.from_bytes(READ_IO_BYTE(21), "little")

    wndw.onclick(place_object,1)
    wndw.onclick(clear_objects,3)

    time.sleep(0.1)
    t2.clear()
except:
    pass
```

ΠΑΡΑΔΕΙΓΜΑ ΚΩΔΙΚΑ ΣΕ ΓΛΩΣΣΑ ASSEMBLY

ΓΙΑ ΤΟΝ ΕΛΕΓΧΟ ΤΟΥ ΡΟΜΠΟΤΙΚΟΥ ΒΡΑΧΙΟΝΑ ΜΕΣΩ ΤΟΥ EMU8086

```
#start=Robotic_arm.exe#
Title XXXXXXXXXX


   TITLE HELLOWORLD
   DATA SEGMENT
        x1 db 0
        MSG DB 10,13,"$"
   DATA ENDS


   CODE SEGMENT
        START: MOV AX, DATA
               MOV DS, AX


            ; DEMO CODE FOR ROBOTIC ARM


           MOVE_ARM1:
              IN AL,20    ; Έλεγχος αν ο βραχίονας είναι έτοιμος
              CMP AL,0
              JNE MOVE_ARM1


              in al, 30    ; Ανάγνωση LSB x συντεταγμένης αντικειμένου

              in al, 31    ; Ανάγνωση MSB x συντεταγμένης αντικειμένου

              in al, 32     ; Ανάγνωση LSB y συντεταγμένης αντικειμένου

              in al, 33     ; Ανάγνωση MSB y συντεταγμένης αντικειμένου


              IN AL, 11   ; Read angle of arm 1
              ADD Al, 10  ; turn  10 degrees

              OUT 11, AL   ; update angle of arm1

           MOVE_ARM2:
              IN AL,20    ; Έλεγχος αν ο βραχίονας είναι έτοιμος
              CMP AL,0
              JNE MOVE_ARM2


              IN AL, 12   ; Read angle of arm 2
              ADD Al, 15  ; increment angle by 15 degrees
              OUT 12, AL   ; update parameter of ARM2 angle.


            MOVE_ARM3:      ; Έλεγχος αν ο βραχίονας είναι έτοιμος
              IN AL,20
              CMP AL,0
              JNE MOVE_ARM3
```

```
    IN AL, 13     ; Read angle of arm 3
    MOV Al,180    ; Change angle to 180 degrees.
    OUT 13, AL    ; Update  parameter for ARM 3

  MOVE_ARM4:      ; Έλεγχος αν ο βραχίονας είναι έτοιμος
    IN AL,20
    CMP AL,0
    JNE MOVE_ARM4


    IN AL, 14     ; Read length of arm 3
    ADD Al, 15    ; stretch arm my 15 units
    OUT 14, AL    ; Update parameter for Arm4 length


  MOVE_GRIP:      ; Έλεγχος αν ο βραχίονας είναι έτοιμος
    IN AL,20
    CMP AL,0
    JNE MOVE_GRIP


    IN AL, 15     ; Read grip angle
    SUB Al,30     ; turn grip -30 degrees
    OUT 15, AL    ; Update parameter for grip angle


  OPEN_GRIP:      ; Έλεγχος αν ο βραχίονας είναι έτοιμος
    IN AL,20
    CMP AL,0
    JNE OPEN_GRIP


    IN  AL, 16    ; Read grip status
    MOV AL, 0     ; open Grip
    OUT 16, AL    ; Update parameter for grip status



    MOV AL, 1     ; terminate robotic arm
    OUT 17, AL    ;



    MOV AH, 4CH
    INT 21H
CODE ENDS
END START
```