Robotic arm using 8086 trainer MTS-86C

1- حسين حمزة
2- مرتضى حكمان
3- احمد حازم
4- محمود عباس
5 – ياسر صلاح

شعبة A

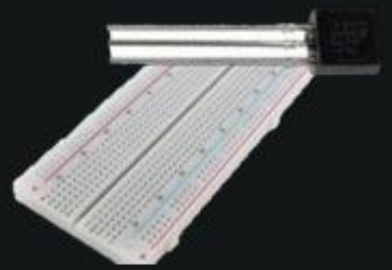
Components of the project

01 8086 trainer MTS-86C

02 Conveyor belt

03 3 stepper motors

04 Transistors and wires , breadboard




MTS-86C

WHAT IS THE MTS-86C TRAINER

The MTS-86C trainer is likely a piece of hardware used for educational purposes or for training individuals in the programming and operation of the Intel 8086 microprocessor. The Intel 8086 is a 16-bit microprocessor introduced in 1978, which was widely used in early personal computers and embedded systems.

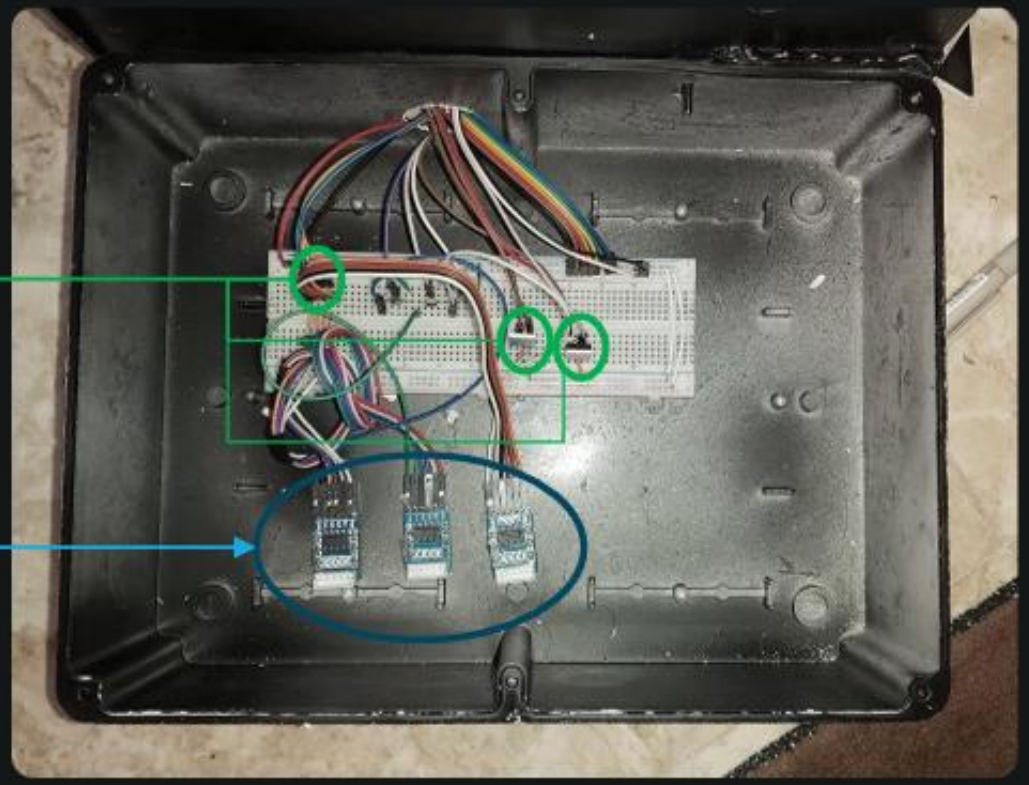
How does the arm work?

The robotic arm is moved by a stepper motors. There are 3 steppers. The first is responsible for moving the arm right or left. The second stepper is responsible for moving the arm up or down. The last stepper is responsible for picking things up from the conveyor belt.

How does the arm work?

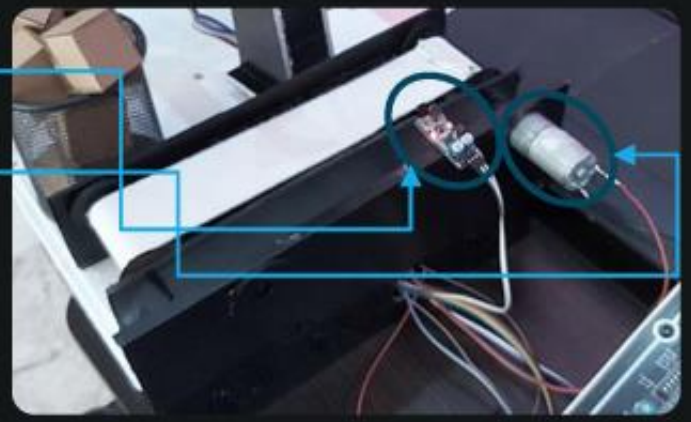
STEPPER
MOTORS DRIVERS

P-MOSFET
TRANSISTORS
(for enabling at 0)



The Conveyor Belt?

The conveyor belt is basically controller with one dc motor with 80rpm power, and it’s automatically switched on or off with the IR sensor.
{NO OBJECT >> DC MOTTOR = 0)
{OBJECT DETECTED >> DC MOTOR = 1)



CODE SNIPPET

```
;Automatic Arm Porject using MIS-86C

;Members' names:-

;Ahmed Hazem
;Hussien Hamza
;Yasser Salah
;Mahmoud Abbas
;Morteda Hakaan


;CODE:-
COMMAND1 EQU 0FFFFH
PORTA1 EQU 0FFF8H
PORTB1 EQU 0FFF9H       ;Addresses Define PPI-1
PORTC1 EQU 0FFFAH

COMMAND2 EQU 0FFFEH
PORTA2 EQU 0FFF4H
PORTB2 EQU 0FFF5H       ;Addresses Define PPI-2
PORTC2 EQU 0FFF6H

;-------------------------------------------------MAIN PORGRAM-------------------------------------------------------------------
org 100h

CODE    SEGMENT
    ASSUME CS:CODE,DS:CODE
START:  MOV SP,2000H            ;this code initializes the stack pointer to address 2000H
        MOV AX,CS               ;and then sets the data segment register to the same value as the code segment
        MOV DS,AX               ;This is a typical setup for x86 assembly programs.

MOV DX, COMMAND1
MOV AL , 82H                    ;Initializes PPI-1
OUT DX, AL

MOV DX, COMMAND2
MOV AL, 00H                     ;Initializes PPI-2
OUT DX , AL

L1:

MOV DX,PORTA1
MOV AL,00H                      ;Turning ON the DC motor
OUT DX,AL

MOV DX , PORTB1
IN_LOOP:
IN  AL , DX                     ;Waiting to detect object
CMP AL , 0FEH
JNZ IN_LOOP

CALL DELAY_DC

MOV DX,PORTA1
MOV AL,0FH                      ;Turning OFF the DC motor
OUT DX,AL

;-------------------------------------------------RUN THE ARM-------------------------------------------------------------------

MOV DX,PORTA2
MOV CX, 12BH
MOV BL , 28H
CALL TRANS                      ; ROTATE CW - STEPPER 2 DOWN
CALL RCW
CALL DELAY2

MOV CX, 15BH
MOV BL , 48H
CALL TRANS                      ; ROTATE CW - STEPPER 3 PICK
CALL RCW
CALL DELAY2

MOV CX, 12BH
MOV BL , 28H
CALL TRANS                      ; ROTATE CCW - STEPPER 2 UP
CALL RCCW
CALL DELAY2

MOV CX, 25BH
MOV BL , 18H
CALL TRANS                      ; ROTATE CW - STEPPER 1 RIGHT
CALL RCW
CALL DELAY2


MOV CX, 88H
MOV BL , 28H                   ; ROTATE CW - STEPPER 2 DOWN
CALL TRANS
CALL RCW
CALL DELAY2

MOV CX, 15BH
MOV BL , 48H
CALL TRANS                      ; ROTATE CW - STEPPER 3 RELEASE
CALL RCCW
CALL DELAY2

MOV CX, 88H
MOV BL , 28H
CALL TRANS                      ; ROTATE CW - STEPPER 2 UP
CALL DELAY2

JMP L
;-------------------------------------------------END MAIN PORGRAM---------------------------------------------------------------

;-------------------------------------------------PROCEDURES---------------------------------------------------------------

TRANS PROC
    PUSH DX

    MOV DX,PORTB2
    MOV AL,BL
    OUT DX,AL                   ; Select the transistor

    POP DX
RET
TRANS ENDP


RCW PROC
MOV AL,11H
    L1:

    PUSH CX
    CALL CW                     ; Clock wise - multi steps
    POP CX

    LOOP L1
    RET
RCW ENDP

CW PROC
    OUT DX , AL
    CALL DELAY                  ; Clock wise - one step
    ROR AL,1H
    RET
CW ENDP


PROC RCCW
MOV AL,88H
    L3:

    PUSH CX
    CALL CCW                    ; Counter clock wise - multi steps
    POP CX

    LOOP L3
    RET
RCCW ENDP

CCW PROC
    OUT DX , AL
    CALL DELAY                  ; Counter clock wise - one step
    ROL AL,1H
    RET
CCW ENDP

;--------------Delay procedures-------------------

DELAY PROC NEAR USES CX
MOV CX , 300
D1:
LOOP D1
RET
DELAY ENDP

DELAY2 PROC NEAR USES CX
MOV CX , 5H
D:
PUSH CX
MOV CX,2000
    D2:
    LOOP D2
POP CX
LOOP D
RET
DELAY2 ENDP

DELAY_DC PROC NEAR USES CX
MOV CX , 0FFFFH
D_DC:
LOOP D_DC
RET
DELAY_DC ENDP

;End code
```

```
CALL DELAY2

JMP L
;-------------------------------------------------END MAIN PORGRAM---------------------------------------------------------------

;-------------------------------------------------PROCEDURES---------------------------------------------------------------

TRANS PROC
    PUSH DX

    MOV DX,PORTB2
    MOV AL,BL
    OUT DX,AL                   ; Select the transistor

    POP DX
RET
TRANS ENDP


RCW PROC
MOV AL,11H
    L1:

    PUSH CX
    CALL CW                     ; Clock wise - multi steps
    POP CX

    LOOP L1
    RET
RCW ENDP

CW PROC
    OUT DX , AL
    CALL DELAY                  ; Clock wise - one step
    ROR AL,1H
    RET
CW ENDP


PROC RCCW
MOV AL,88H
    L3:

    PUSH CX
    CALL CCW                    ; Counter clock wise - multi steps
    POP CX

    LOOP L3
    RET
RCCW ENDP

CCW PROC
    OUT DX , AL
    CALL DELAY                  ; Counter clock wise - one step
    ROL AL,1H
    RET
CCW ENDP

;--------------Delay procedures-------------------

DELAY PROC NEAR USES CX
MOV CX , 300
D1:
LOOP D1
RET
DELAY ENDP

DELAY2 PROC NEAR USES CX
MOV CX , 5H
D:
PUSH CX
MOV CX,2000
    D2:
    LOOP D2
POP CX
LOOP D
RET
DELAY2 ENDP

DELAY_DC PROC NEAR USES CX
MOV CX , 0FFFFH
D_DC:
LOOP D_DC
RET
DELAY_DC ENDP

;End code
```

THE CHALLENGING STRUGGLES

As simple as it looks like, although it wasn't really that easy.
our first challenge was working with the trainer manually, because of the malfunction of the serial port.
Second issue was with the 8255 reading an input by itself (which was solved by giving all pin 1, and reading for 0).
Third issue was the trainer is missing ROM chip, so we had to enter the code manually each time we had a power outage !! , and we're talking about entering about 300 bytes ! Manually!!.

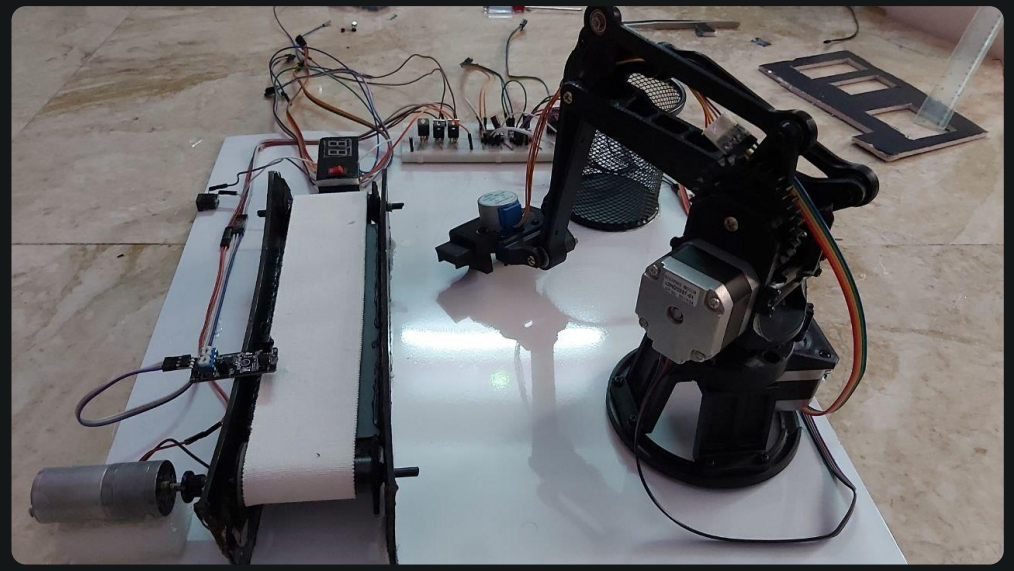
But yes, we were able to overcome all these issues and come to this final project with the 8086 trainer.

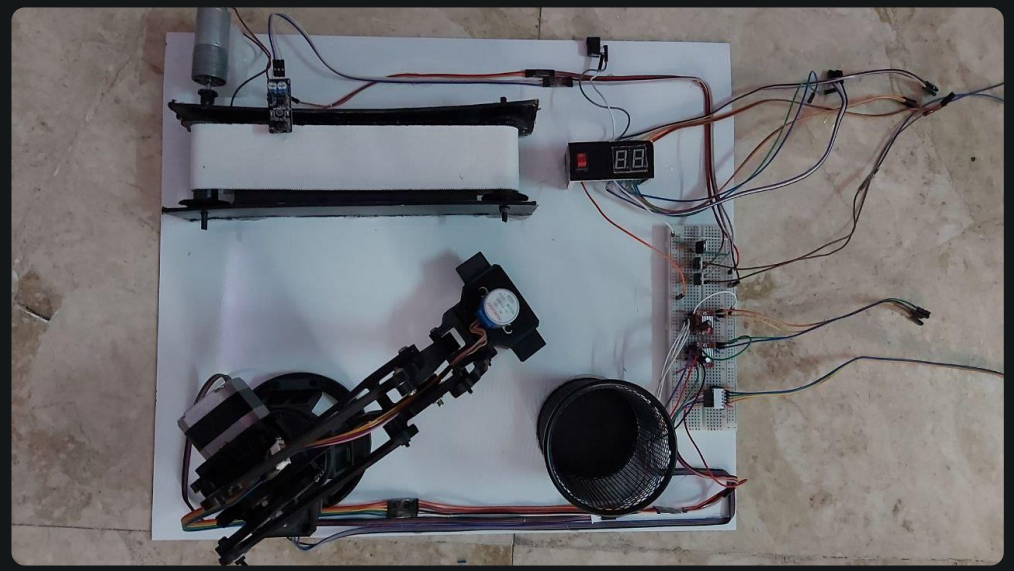
UPDATE

THE NEXT SLIDE WILL SHOW THE NEWEST UPDATE FOR OUR ARM PROJECT AFTER WE 3D PRINTED THE ARM AND CHANGED THE SURFACE AND THE APPEARNECE OF THE WHOLE PROJECT.

WHAT HAVE WE ADDED?:

- 2 SEVEN SEGMENTS
- 2 NEMA17 STEPPER MOTORS
- 3D PRINTED ARM
- THE FINAL CODE
- SURFACE AND BOX
- SHUTDOWN SWITCH





THANK YOU

*DEMO PROVIDED WITHIN THE SAME FOLDER