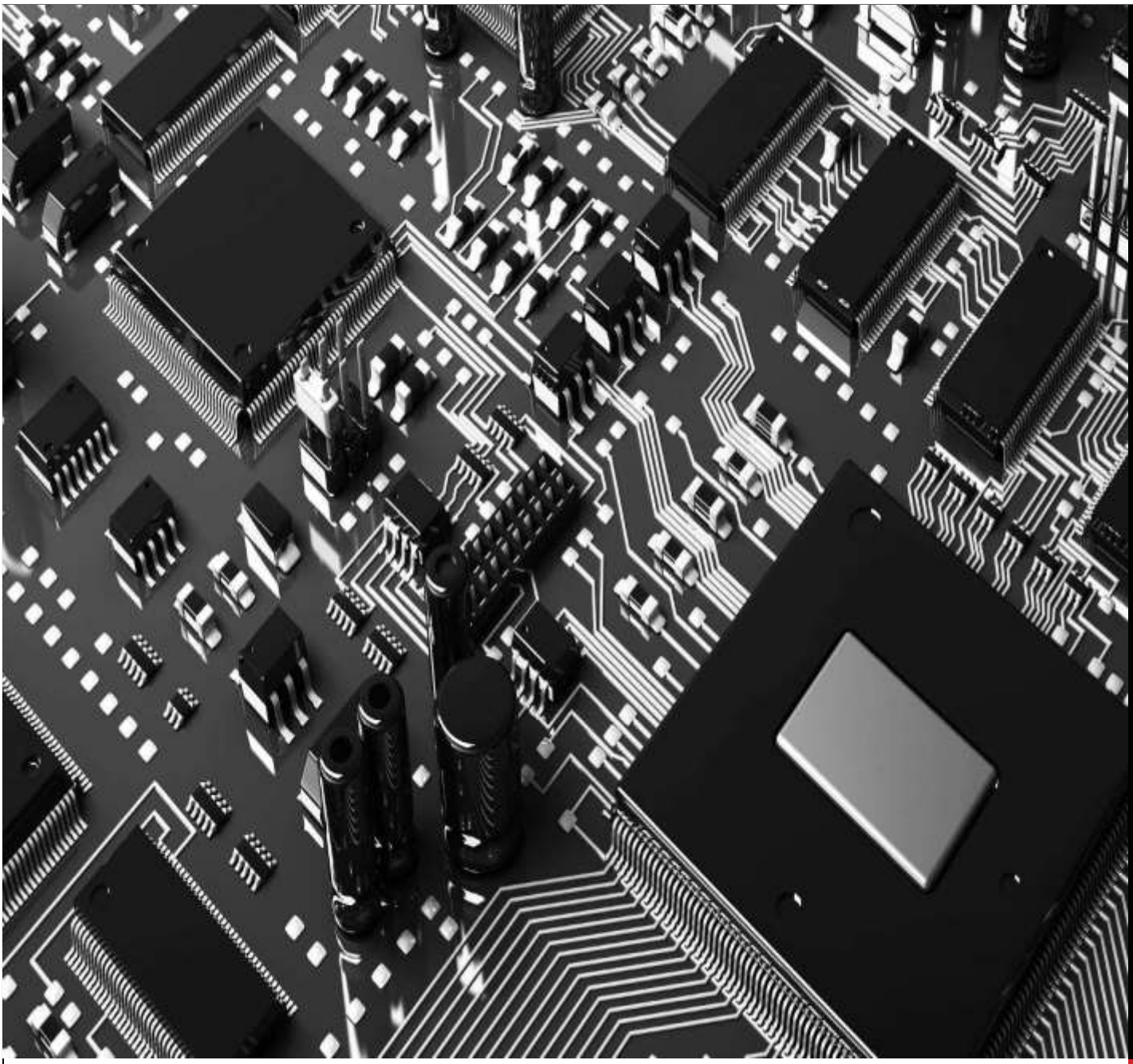


SORTING MACHINE

INTEL 8085 PROJECT

EC 316 INTEL 8085 MICROPROCESSOR LAB REPORT

LALIT ARORA 079EC14

NITIN SHRIVASTAVA 098EC14

ELECTRONICS AND COMMUNICATION DEPARTMENT

NETAJI SUBHAS INSTITUTE OF TECHNOLOGY

SYNOPSIS

This project uses the 8085 microprocessor to implement a **color recognition based sorting machine**. We are using **Intel Microprocessor 8085** (introduced in 1976) as the brain of our sorting machine. The output of manufacturing unit of the industry is transferred towards the sorting unit. The items are sensed by the sensor and passed to the workspace. Workspace will be controlled by two 12-volt bidirectional stepper motors and a color sensor TCS320 and an infrared sensor. Infrared Sensor will detect the object presence on the workspace and then color sensor will sense the color and accordingly stepper motors will rotate at particular angle/s and direct the object in that direction. This process will go on until the whole lot of manufacturing unit output is sorted and sent to desired destinations for packaging or dispatch.

JUSTIFICATION

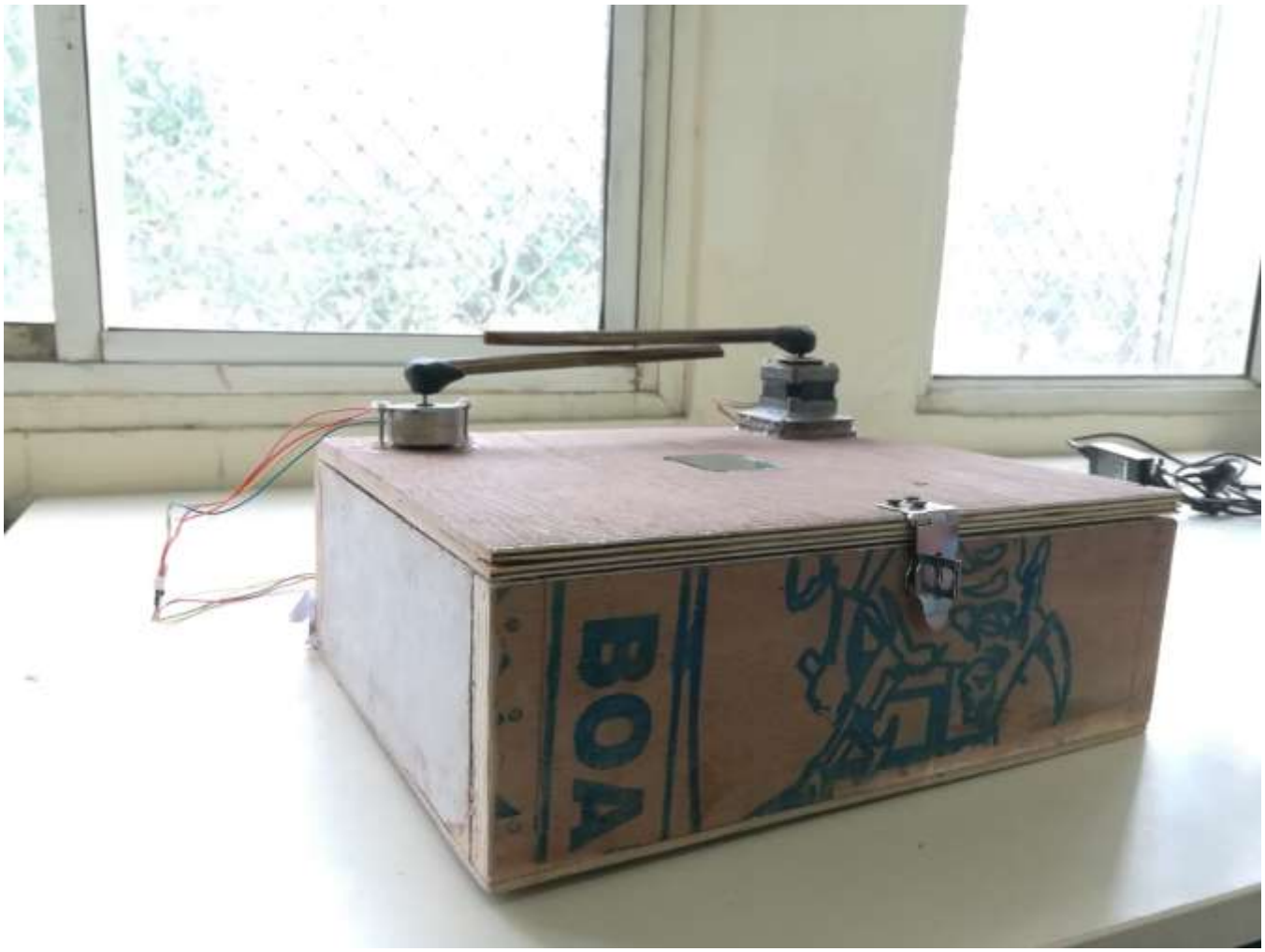
We had planned a general project to prototype with 8085 Microprocessor but **Prof. D.V. Gadre** told us that he is expecting some gadget or an electronic measuring device from us. This helped us to think out of the box and we came up with this innovative idea of making a Sorting Machine which is frequently used in many small scale and large scale industries using PLCs.

This project has taught us the value of electronics and how small independent parts can be brought together to create something special.

This experience has been wonderful and we have learnt how to be patient and how to work together without getting on each other’s nerves. We have also learnt the importance of time management and self-belief.

ACKNOWLEDGEMENT

We are extremely grateful to **Prof. D.V. Gadre** for encouraging us throughout the entire process of making this project. Without his guidance we wouldn't have been able to accomplish this project. A very special thanks to our parents and our friends and seniors for helping us throughout.



OBJECTIVE

Before diving into the project hardware and software specifications, we should first explain the objective of making this project and the reason which fascinated us for working on this project for last 2 to 3 months.

Programmable Logic Controller abbreviated as PLC is a digital computer used for the automation of various electro-mechanical processes in industries. These controllers are specially designed to survive in harsh situations and shielded from heat, cold, dust, and moisture etc. **PLC** consists of a microprocessor which is programmed using the computer language. The program is written on a computer and is downloaded to the PLC via cable. These loaded programs are stored in non – volatile memory of the PLC. During the transition of relay control panels to PLC, the hard wired relay logic was exchanged for the program fed by the user. A visual programming language known as the Ladder Logic was created to program the PLC.

LabVIEW is a graphical development environment that complements IEC 61131-3 based PLCs by incorporating PC and embedded technologies for real-time analysis, monitoring, advanced control, and predictive maintenance. You can significantly achieve better throughput, yield and uptime by adding LabVIEW to your existing PLC-based systems. LabVIEW allow engineers to use streamlined FPGA targeting tools to implement high performance, hardware-based machine monitoring and protection systems and the new LabVIEW Touch Panel Module, which helps them to use the same software to create Windows CE-based HMIs.

We were familiar with this concept and we thinking to implement this on a prototype level. Then only we thought to make a Sorting Machine with 8085 Intel Microprocessor. Various method of sorting includes color/optical recognition, laser sorting and sorting based on images processed by some third party computer software like MATLAB. In our project, sorting is implemented by color recognition using sensor TCS3200, Object Detection using Infrared sensor and 12V Stepper Motors to sort the objects in different directions.

TABLE OF CONTENTS

1.HARDWARE

1.1 BILL OF MATERIALS (BOM)

Part	Value	Device	Package	Description
8085-MPU		8085	DIL40	MICROCOMPUTER/PERIPHERAL DEVICE
8254-COUNTER		8253	DIL24-6	MICROCOMPUTER/PERIPHERAL DEVICE
8255-PPE		8255A	DIL40	MICROCOMPUTER/PERIPHERAL DEVICE
ADDLATCH	74HCT573N	74HCT573N	DIL20	8-bit D latch BUS DRIVER
C7		C-EU025-025X050	C025-025X050	CAPACITOR, European symbol
C8		CPOL-EUE2.5-6	E2,5-6	POLARIZED CAPACITOR, European symbol
C9		CPOL-EUE2.5-6	E2,5-6	POLARIZED CAPACITOR, European symbol
C16		C-EU025-025X050	C025-025X050	CAPACITOR, European symbol
C17		CPOL-EUE2.5-6	E2,5-6	POLARIZED CAPACITOR, European symbol
C18		C-EU025-025X050	C025-025X050	CAPACITOR, European symbol
C19		C-EU025-025X050	C025-025X050	CAPACITOR, European symbol
C20		C-EU025-025X050	C025-025X050	CAPACITOR, European symbol
C24		C-EU025-025X050	C025-025X050	CAPACITOR, European symbol
C25		C-EU025-025X050	C025-025X050	CAPACITOR, European symbol
C26		C-EU025-025X050	C025-025X050	CAPACITOR, European symbol
C27		C-EU025-025X050	C025-025X050	CAPACITOR, European symbol
C29		C-EU025-025X050	C025-025X050	CAPACITOR, European symbol
C31		C-EU025-025X050	C025-025X050	CAPACITOR, European symbol
C33		C-EU025-025X050	C025-025X050	CAPACITOR, European symbol
COUNTER0		M03	03P	AMP QUICK CONNECTOR
COUNTER1		M03	03P	AMP QUICK CONNECTOR
COUNTER2		M03	03P	AMP QUICK CONNECTOR
CRYSTAL		CRYSTALHC49US	HC49US	Crystals
IC1	74HCT04N	74HCT04N	DIL14	Hex INVERTER
MEM-DEC	74HCT138N	74HCT138N	DIL16	3-line to 8-line DECODER/DEMULTIPLEXER
P1-10		M10	10P	AMP QUICK CONNECTOR
P11-20		M10	10P	AMP QUICK CONNECTOR
P21-30		M10	10P	AMP QUICK CONNECTOR
P31-40		M10	10P	AMP QUICK CONNECTOR
PORTB		M08	08P	AMP QUICK CONNECTOR
PORT_A_MOTORS		M08	08P	AMP QUICK CONNECTOR
PORT_C_H_IR		M04	04P	AMP QUICK CONNECTOR
PORT_C_L_TCS1		M04	04P	AMP QUICK CONNECTOR

PWR	LED5MM	LED5MM	LED5MM	
PWR5		M02	02P	AMP QUICK CONNECTOR
R2		R-EU_0204/7	0204/7	RESISTOR, European symbol
R3		R-EU_0204/7	0204/7	RESISTOR, European symbol
R4		R-EU_0204/7	0204/7	RESISTOR, European symbol
R5		R-EU_0204/7	0204/7	RESISTOR, European symbol
R6		R-EU_0204/7	0204/7	RESISTOR, European symbol
R7		R-EU_0204/7	0204/7	RESISTOR, European symbol
R8		R-EU_0204/7	0204/7	RESISTOR, European symbol
R9		R-EU_0204/7	0204/7	RESISTOR, European symbol
R10		R-EU_0204/7	0204/7	RESISTOR, European symbol
R12		R-EU_0204/7	0204/7	RESISTOR, European symbol
R13		R-EU_0204/7	0204/7	RESISTOR, European symbol
R14		R-EU_0204/7	0204/7	RESISTOR, European symbol
R15		R-EU_0204/7	0204/7	RESISTOR, European symbol
RAM-62256P		62256P	DIL28-6	MEMORY
RESET		10-XX	B3F-10XX	OMRON SWITCH
ROM-58C256		58C256P	DIL28-6	MEMORY
SID		10-XX	B3F-10XX	OMRON SWITCH
SOD	LED5MM	LED5MM	LED5MM	
TCS0		M04	04P	AMP QUICK CONNECTOR
USB	USB(POWER)	USB(POWER)	USB(POWER)	

1.2 COMPONENTS AND THEIR INTERFACING

1.2.1 Motherboard

We started off with an idea of having all the I/O and memory devices on the motherboard but prototyping an automation project on a single board will mess all the things. Our Motherboard contains:

- The Clock
  The maximum allowed frequency for operation in 8085 is 5 MHz A source crystal of 4Mhz is used therefore our system will work at 2 MHz frequency. A 22pf capacitor is used with crystal to provide enough loading to crystal to generate oscillations.

- Reset Switch
  This switch will take back the whole system to its initial coordinates whenever system got struck in some undefined location or an infinite loop (Boot loop). This switch has a very important role which will only be the support when system had lost somewhere.

- SID SOD

  These special pins provide a 1-bit serial communication from one pin to other using SIM and RIM commands of 8085 Microprocessor Instruction Set. In our project these pins are used to show the communication between a push button and a 5mm red led.

- MEMORIES: RAM 62256P & ROM 58C256

  RAM is a volatile memory which loses its data at power off. So we want that our code will stay saved hence we need a non-volatile memory ROM. After analyzing our project requirements, we come up with a decision of interfacing a 32K RAM 62256P and 32K ROM 58C256. After every power off or reset, program counter directs to location "0000H" hence we have to interface in such a way that whenever my system gets reset or switched on, program counter will point to the first location of my ROM or first location of my program such that it will not cause any intervention to my system.

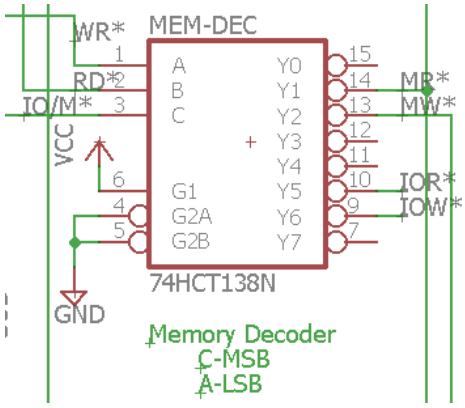
  Hence to achieve our decision we have to interface such that my ROM starting address should be "0000H".
  Address lines provided by 64K 8085 = 16 lines
  Address lines required by 32K Memory = 15 lines
  Hence we can use A15 line to distinguish between RAM and ROM. We used a NOT Gate 74HCT04N to get chip select signals for our ROM and RAM as shown in Schematic.

  How to Demultiplex the address signals AD0 to AD7?

  According to 8085 Datasheet, we can see that the lines AD0 to AD7 are multiplexed i.e. we can use them as address lines and data lines both. To distinguish or demultiplex these lines we need an Address Latch 74HCT573N which will be enabled by a signal named Address Latch Enable abbreviated as ALE. 8085 has a pin named ALE which goes HIGH on start of every Machine cycle for one T state. As ALE goes high, lines AD0 to AD7 contains lower byte address which is latched to output of Address Latch (A0 to A7) and after first T state we can use these lines as data lines (D0 to D7).

- Control Signals

  We need some Control Signals which will be generated when input and output commands are called from program. These signals help us to generate Read and Write signals for Memory and I/O devices. These signals are generated using a 3x8 Decoder 74HCT138N. Connections are shown in given figure:



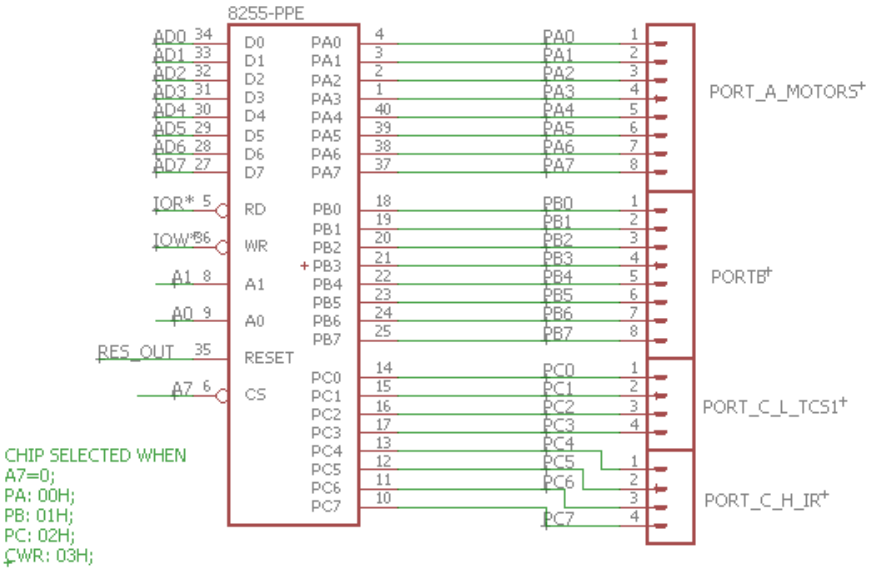
Four useful signals decoded from this decoder are **MR*** (Memory Read), **MW*** (Memory Write), **IOR*** (I/O Read) and **IOW*** (I/O Write).

1.2.2 Programmable Peripheral Interface (PPI)

After working over the interfacing of memories, it's time to connect peripherals which will communicate with 8085 during the sorting process. The requirements are brought down on paper and we came to know that we need

- ✓ Output Port for our Color Sensor setting pins
- ✓ Input Pin for Infrared Led input (IR)
- ✓ Output Port for 2 Bidirectional Stepper Motors.

Hence in all, 8255A PPI is the most appropriate contender of all other methods of interfacing includes using latches and buffers separately for each output and input device respectively.



8255A is a 40 pin DIP Programmable peripheral interface IC specially designed to communicate with INTEL Microprocessors. It has 3 I/O ports named PORTA, PORTB and PORTC. We used these ports as follows:

- PORT A – 2 Bidirectional Stepper Motors (12V) Output
- PORT B – Not Connected
- PORT C (upper) – Infrared Sensor Input
- PORT C (lower) – Color Sensor TCS3200 settings Output

8255A can work in 3 modes. First mode is MODE 0 which is simple I/O interface. Second mode is MODE 1 in which PORT A and PORT B can communicate with peripherals using Handshake signals using PORT C. Third and last mode is MODE 2 in which PORT A works in Bidirectional Mode using Handshake signals from PORT C and PORT B can be used as simple I/O.

We are working in MODE 0 which is Simple I/O because we do not need any special Handshake signals.

Enabling of 8255A is done through pin A7. 8255A has active low chip select hence when A7=0 the 8255A will be enabled.

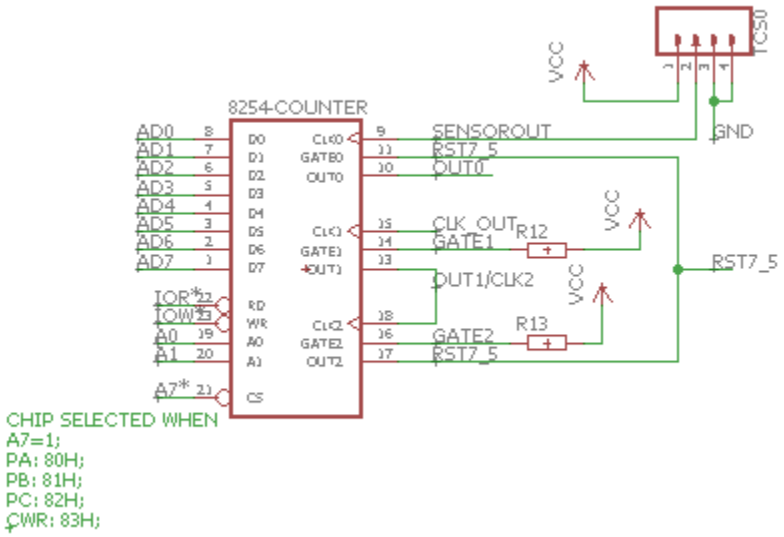
Read and Write control signals are provided by the Decoded control signals explained in last section i.e. IOR* and IOW*.

Reset pin of 8255A is connected to Reset Out Pin of 8085 so that whenever 8085 get reset then all the other peripherals will also get reset.

1.2.3 Programmable Interval Timer/Counter

Interfacing of peripherals is done but main requirement was to measure the frequency of a square wave send as output signal of color sensor TCS3200. Working of sensor will be explained later in this report. So, we deduced a method to measure it using versatile 8254 24 pin DIP Programmable Interval Timer IC which is compatible with 8085. Procedure to measure frequency will be to load the counter of 8254 with a particular hex equivalent of a decimal number and decrement it for 1sec. As 8254 will decrement only when its Clock transit from LOW to HIGH. Therefore, Clock terminal of 8254 Counter 0 is connected to Sensor Output. Hence 8254 will be able to calculate the number of HIGH-LOW transitions or oscillations in 1 sec of Output wave. This value can be calculated by subtracting the remaining count from the initial loaded count. This subtracted result will tell you the frequency which is used by 8085 to decide and control Bidirectional Stepper motors.

8254 IC has 3 16-bit counters which can work independently in 6 modes from MODE 0 to MODE 5. 8254 decrements the loaded count and continues as per the mode in which it is working. Explaining all the modes of 8254 is not right here so we are explaining only those Modes which were used in this project. For rest reader can consult Datasheet of 8254.



Counter 0 is working in MODE 0: Interrupt on Terminal Count. In this mode, OUT is initially low, and will remain low until the Counter reaches zero. OUT then goes high and remains high until a new count or a new Mode 0 Control Word is written into the Counter. We use this mode to count the number of HIGH LOW transitions in CLK or Sensor Output Wave and hence we were able to estimate the frequency of the signal. But the major problem was that we have to trigger the gate of Counter 0 after 1 sec. Hence, how to generate an interrupt after 1 sec. For this we used Counter 1 and Counter 2 in cascade to generate 1 sec interrupt.
First Counter 1 was loaded with hex equivalent of 50000 and initiated in MODE 2. Mode 2 is Rate Generator. This Mode functions like a divide-by-N counter. It is typically used to generate a Real Time Clock interrupt. OUT will initially be high. When the initial count has decremented to 1, OUT goes low for one CLK pulse. OUT then goes high again, the Counter reloads the initial count and the process is repeated. Mode 2 is periodic; the same sequence is repeated indefinitely. For an initial count of N, the sequence repeats every N CLK cycles.

Secondly Counter 2 is loaded with hex equivalent of 40 and initiated in Mode 4. Mode 4 is a Software Triggered Strobe. OUT will be initially high. When the initial count expires, OUT will go low for one CLK pulse and then go high again. The counting sequence is ``triggered'' by writing the initial count. Out of the counter 2 is connected to RST 7.5 Interrupt pin and Gate terminal of Counter 0.

Counter 1 loaded with: 50000

Counter 2 loaded with: 40

Out of counter 1 is connected to clock of counter 2. (cascaded).

Hence net loaded value: 50000x40=2000000 = 2Mhz which will provide interrupt after every 1 sec because Clock Out of 8085 is also 2Mhz.

In all, Counter 1 and Counter 2 in cascade providing interrupt after every 1 sec and Counter 0 is triggered with this interrupt and tells the remaining count according to its clock which is sensor output.

Enabling of 8254 is done with A7*. 8254 has active low enable pin hence when A7 =1 or A7* = 0, then 8254 will be enabled.

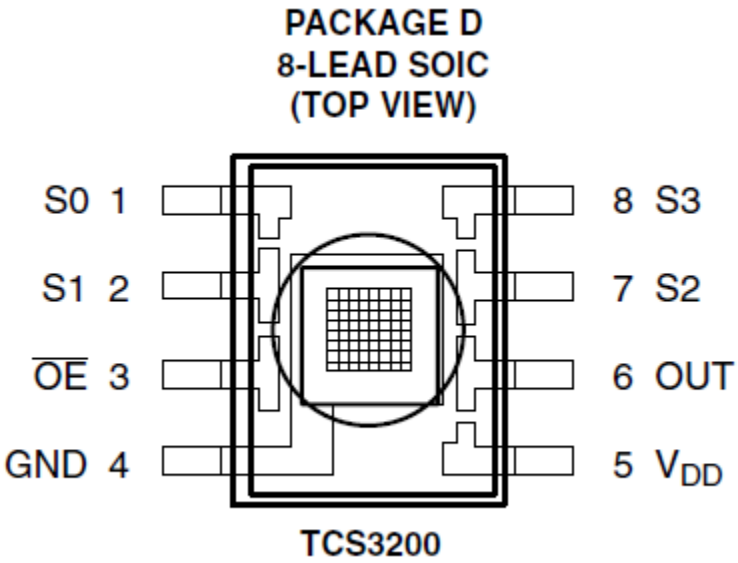
Read and Write control signals are provided by the Decoded control signals explained in last section i.e. IOR* and IOW*.

1.2.4 Color Sensor TCS3200

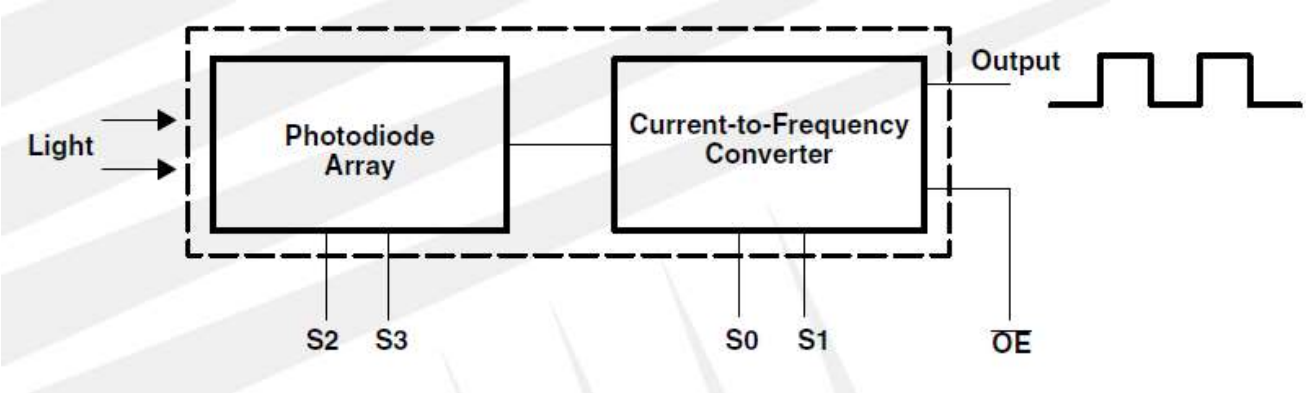
The TCS3200 and TCS3210 programmable color light-to-frequency converters that combine configurable silicon photodiodes and a current-to-frequency converter on a single monolithic CMOS integrated circuit. The output is a square wave (50% duty cycle) with frequency directly proportional to light intensity (irradiance). The full-scale output frequency can be scaled by one of three preset values via two control input pins. Digital inputs and digital output allow direct interface to a microcontroller or other logic circuitry. Output enable (OE) places the output in the high-impedance state for multiple-unit sharing of a microcontroller input line.

In the TCS3200, the light-to-frequency converter reads an 8 x 8 array of photodiodes. Sixteen photodiodes have blue filters, 16 photodiodes have green filters, 16 photodiodes have red filters, and 16 photodiodes are clear with no filters.

Pinout of TCS3200 is shown in figure



Working of TCS3200 is shown in figure



In our project when object is detected on Workspace then TCS3200 send light signals on the object and gives the output as a 50% duty cycle square wave as shown in the above figure. The frequency is proportional to RGB content in the object. Hence we used 8254 to calculate the frequency of the square wave coming out as output signal from TCS3200.

S0 and S1 are used to set the scaling factor of frequency.

S0	S1	OUTPUT FREQUENCY SCALING (f_o)
L	L	Power down
L	H	2%
H	L	20%
H	H	100%

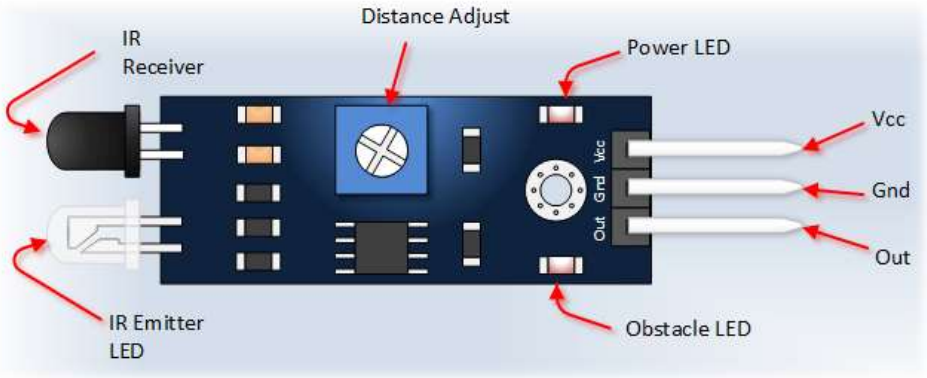
S2 and S3 are set to get the output square wave for different primary colors as follows:

S2	S3	PHOTODIODE TYPE
L	L	Red
L	H	Blue
H	L	Clear (no filter)
H	H	Green

Hence 8255A PPI PORT C (lower) is used to send the signals for S0, S1, S2, S3.

1.2.5 Infrared Sensor

Infrared sensor is used as proximity sensor which will continuously transmit and receive infrared radiations from a IR Led and IR photodiode. The received signal output voltage is compared with the reference voltage to detect whether the object is placed on workspace or not. If value is above the reference voltage, then it sends logic HIGH to 8085 else send logic LOW. Comparison is done using dual op amp IC LM358. It is responsible to enable the Color Sensor TCS3200.



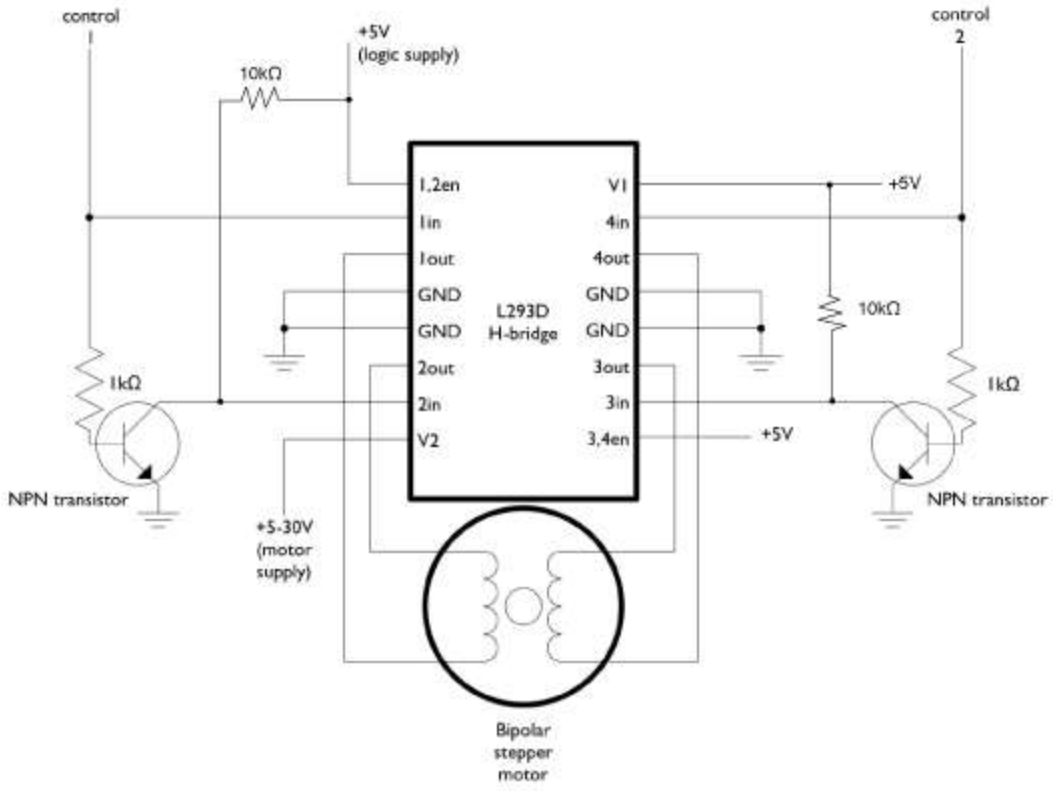
1.2.6 MOTOR BOARD

Apart from Motherboard and peripherals, we had also designed a Motor Controller Board. This board contains:

I. 2 Motor Driver Modules L293D
II. Voltage Regulator 7805
III. Connectors
IV. Power Supply Ports (12V)

The reason of making a separate board for motor control was to provide isolation to Motherboard from 12V power supply. The motor control board is powered with 12V DC supply using a 12V 2A adaptor and then a 5V supply is generated using 7805 voltage regulator.

This 5V supply is used to power L293D motor driver IC and 12V supply is used in parallel to power stepper motors. L293D is a motor driver IC containing two H Bridges to run two coils at a time. Stepper Motors have 4 coils which when energized in a particular manner to move the rotor of the motor by a particular step angle.

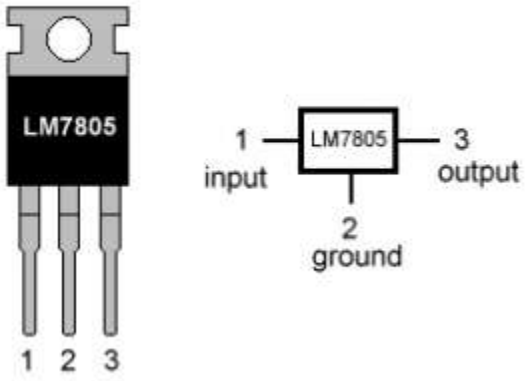


Interfacing of Bidirectional Stepper Motor is shown in above figure. To energize a coil of stepper motor, we need to send a particular 4-bit code to 8085 which will send that code to output ports of 8255A with which L293D is connected and motors will respond to that. Coding part of Motors will be explained in Software Part of this Report.

Procedure employed in making the board

1. We started with a Zero Board to make Motor Control Board.
2. First we designed the schematic on a paper and marked the places on the Zero board before soldering the components directly.
3. First power supply ports are soldered with 7805 Voltage Regulator connected with main supply to generate to parallel lines of 5V and 12V.

LM7805 PINOUT DIAGRAM



4. After completing the supply Connections, Motor Driver modules are soldered using Female Headers.
5. Male Connectors (M0s) are soldered for connection of motors with the Motherboard.
6. At last two 12V Bidirectional Stepper Motors (with wooden propellers) are connected to the connectors on Motor Control Board.

1.3 SCHEMATIC

After preparing the whole schematic on the paper and Motor Control Board on Zero Board, it's time to make the schematic on Computer Software so that we can design the PCB out of that.

We choose Autodesk Eagle 8.2.0 to design our schematic. This is the latest version of Eagle Software in which Auto Router algorithm is highly optimized. We added the required libraries from Adafruit and CeDT. Schematic is shown on the next page.

1.4 BOARD LAYOUT

Board Layout is generated from Schematic by placing the components on the board and Auto Routing the board with given specifications:

Net Class: Power Net Width (Vcc and GND): 14 mil

Net Width: 24mil

Clearance :12mil

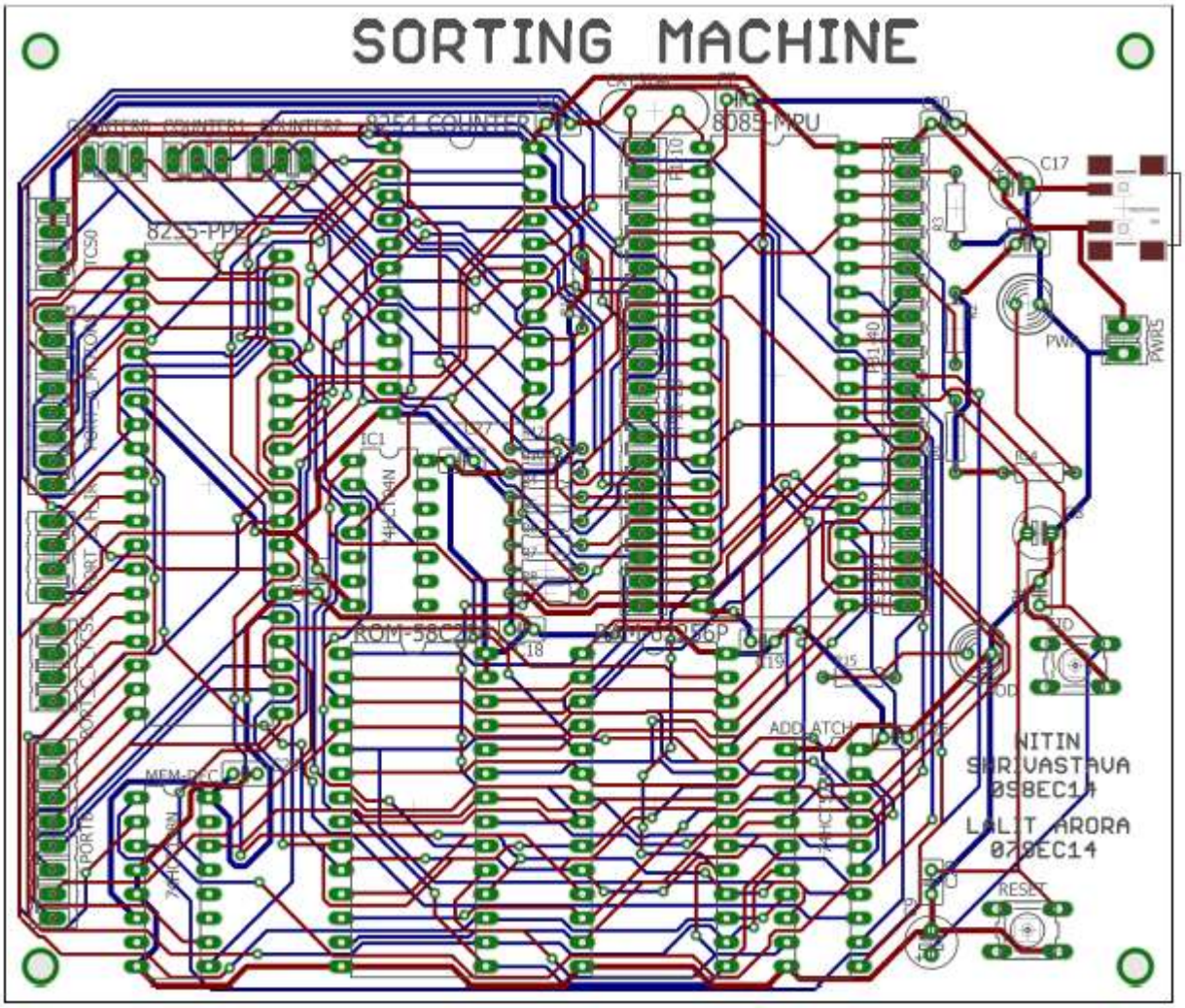
1.5 PCB FABRICATION

For Fabrication of PCB, we generated Gerber Files of our Board using Cam Processor Jobs gerb274x.cam and excellon.cam. After generating Gerber Files, we used Free DFM Checker of PCBPower and 4pcb to compile our board for fabrication errors. At last, we ordered the PCB for fabrication at PCBPower by logging to their website: https://www.pcbpower.com/

Fabrication part of Motor Control Board is explained in previous section.


SORTING MACHINE
ETOK
NITIN
SHRIVASTAVA
098EC14
LALIT ARORA
079EC14
RESET
AT28C256
20PI
0317

SORTING MACHINE


SORTING MACHINE
8085-MPU
8255-PPI
8254 COUNTER
C17
PWR
PWR5
ROM-58C256
ADD LATCH
NITIN
SRIVASTAVA
098EC14
LALIT ARORA
078EC14
RESET

2. SOFTWARE

MEMORY LOCATIONS

Interfacing of Memory Devices RAM and ROM was described above and concept of reset of 8085 was also explained in Hardware section of our report. After every reset of power on, program counter points to address location “0000H” Hence we interfaced ROM and RAM with an investor using A15 Pin. A15 when LOW, will enable ROM and if A15 is HIGH, will enable RAM. So address locations will be as shown below:

Starting address of ROM must be: 0000H

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Size of ROM : 32K i.e. $\log_2(32 * 1024)$ = 15.

So if all the 15 address lines goes HIGH then the final address for ROM will be: 7FFFH

0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Starting address of RAM will be: 8000H

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Size of RAM : 32K i.e. $\log_2(32 * 1024)$ = 15.

So if all the 15 address lines goes HIGH then the final address for ROM will be: FFFFH

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

PROGRAMMABLE PERIPHERAL INTERFACE 8255 ADDRESS

PORT A

0	0	0	0	0	0	0	0

PORT B

0	0	0	0	0	0	0	1

PORT C

0	0	0	0	0	0	1	0

CONTROL WORD REGISTER

0	0	0	0	0	0	1	1

PROGRAMMABLE INTERVAL TIMER/COUNTER 8254 ADDRESS

COUNTER 0

1	0	0	0	0	0	0	0

COUNTER 1

1	0	0	0	0	0	0	1

COUNTER 2

1	0	0	0	0	0	1	0

CONTROL WORD REGISTER

1	0	0	0	0	0	1	1

MOTOR CONTROL DATA LINES CODES

Stepper Motor has 2 coils which are controlled by 4 data lines of 8085. In our case we are using PORT A of 8255 to send data to L293D module which is driving the motors in 8 step sequence method. In this method 8 pulses are send to make a step angle rotation and you can

make a loop to rotate any multiple of that angle. Bidirectional Stepper motors are used hence they can rotate the propeller in any of direction.

8 pulses code in sequence is shown below:

Anti-Clockwise: 04H,06H,02H,0AH,08H,09H,01H,05H

Clockwise 05H,01H,09H,08H,0AH,02H,06H,04H

Testing code is shown under the testing section of this report. We first interfaced motors with Arduino Nano using PORT MANIPULATION. In this we have to send the data to Arduino in form of hex or binary pulses using Embedded C Programming. DDR register is used to assign a certain port as Input or Output. PORT register is used to send data to port in form of binary or hexadecimal.

You can have a look to the PORT MANIPULATION code from my GitHub Repository: https://github.com/MCodez/Embedded-Systems

3. TESTING

Testing is an important phase while making a project. We decided to test each and every IC on our Motherboard first and each peripheral by writing a code. Testing can also be classified in two categories: Hardware Testing and Software Testing. First we will brief on Hardware Testing and then dive in Software Testing Part.

HARDWARE TESTING

PCBPower has provided us the testing result with the invoice of our Board Fabrication. Electrical Testing of nets and connections is done by PcbPower before dispatch but for calming us, we testing nets using Multimeter Continuity Check.

Power check: To check any power issues, we powered our board with Micro USB from a Power Bank and Power Led lights up. We switched Multimeter on DC Voltmeter mode for range of 20V and checked the voltages between each IC's Vcc and Gnd pins.

Clock Check: To check the clock, we used Oscilloscope at our CEPD Lab Room No 221 ECE Department. We provided the power to our board and put the oscilloscope probe at Clock Out pin of 8085 (Pin 37) and other probe to Ground. We had installed 4Mhz crystal with 22pf capacitor as a clock to our microprocessor 8085. So at Clock Out 8085 sends a divide by 2 clock signal. So we should read 2Mhz square wave on oscilloscope. Result was satisfactory as we were getting a square wave of 1.9Mhz with much distortions.

SOFTWARE TESTING

Peripherals like 8255 and 8254 cannot be checked with Multimeter or Oscilloscope satisfactorily. We need to write a code to confirm their working. Memory Testing or RAM Testing is done by writing an assembly code which is explained below. We did also write codes for SID and SOD serial transmission which checks for 8085 working. Peripheral working will test Decoders and Address Latch working automatically. Sensor Testing and Motor Testing is done using Arduino Uno and Oscilloscope before interfacing with 8085 Microprocessor.

Jubin Simulator is used with Hex2Bin exe to convert hex file to binary which is directly uploaded to ROM using EEPROM Programmer GUI.

SID SOD Serial Transmission or 8085 Check:

To check Sid and Sod lines, we send data to Sod through Sid pin. A switch is connected to SID pin as shown in schematic and SOD pin is connected with a Red 5mm Led. An assembly code is written to light on the led as switch is pressed and off as switched is not pressed. Code is a follows:

```
1   .org 0000h
2   JMP sid
3   .org 003CH
4   JMP isr
5   sid: RIM         ; READ SID LINE
6       ANI 80H     ;D7 BIT ISISOLATED
7       ORI 40H     ;VALUE IS TOGGLED
8       SIM
9       JMP SID
10   isr:  RET
```

Result: SID SOD and 8085 are working fine.

RAM & PROGRAMMABLE PERIPHERAL INTERFACE 8255A TEST

To test RAM, we wrote a particular byte at all locations of RAM. After writing we read all the locations and compare the data with the fixed byte which was first entered in locations. If there will be any disperency then it will make upper nibble of PORT A of 8255 HIGH else after all comparisons it will make the whole PORT A HIGH. PORT A of 8255 is connected with an 8 leds of led panel containing 10 leds. Code is as follows:

```
    LXI H,8000
LABEL1:    MVI A,50
    MOV M,A
    MOV B,H
    MOV A,L
    ANA B
    XRI FF
    JZ LABEL0
    INX H
    JMP LABEL1
LABEL0:    LXI H,0800
    MVI A,80
    OUT 03
LABEL3:    MOV A,M
    XRI 50
    JNZ LABEL2
    MOV B,H
    MOV A,L
    ANA B
    XRI FF
    JZ LABEL2
    INX H
    JMP LABEL3
    MVI A,F0
    OUT 00
LABEL2:    MVI A,FF
    OUT 00
    RST 1
```

Result: PORT A showed FFH hence RAM as well as 8255 PORT both are fine.

PROGRAMMABLE INTERVAL TIMER/COUNTER 8254 & INTERRUPT RST 7.5 TEST

To test Programmable Timer IC 8254 and Interrupt RST 7.5, a joint code is written which works as follows. Counter 1 and Counter 2 are used to generate a 1 sec delay i.e. after every 1 sec 8254 will generate an interrupt RST 7.5 and program counter will point to ISR location. In ISR, code to increment B register is written which is then displayed at PORT B of 8255A. Hence in all 8254,8255 and Interrupt 7.5 all are used in this code. Code is as follows:

```
.ORG 0000H
JMP 0040H
.ORG 003CH
JMP 0500H
.ORG 0040H
LXI SP,0FFFFH
MVI A,00BH
SIM
EI
MVI A,80H ; all output on 8255
OUT 03H
MVI B,00H
MVI A,074H ; counter 1 mode 2
OUT 83H
```

```
MVI A,050H ; counter 1 least significant bit
OUT 81H
MVI A,0C3H ; counter 1 most significant bit
OUT 81H
MVI A,094H ; counter 2 mode 2
OUT 83H
MVI A,050H ; counter 2 least significant bit
OUT 82H
MVI A,00H ; port b always 0
OUT 01H
LOOP:    MOV A,B
    OUT 00H
    JMP LOOP
.ORG 0500H
INR B
EI
RET
```

COLOR SENSOR TCS3200 TEST

To test Color Sensor in preliminary stages, we wrote a code to send signals to PORT C (upper) for setting color sensor scaling and selecting photo diode color.S0 ,S1,S2 and S3 are connected with upper nibble of PORT C. And output of Color Sensor is send to Oscilloscope using its probe. Other probe is connected with ground.

Result: Output waveform on Oscilloscope was a perfect square wave with frequency proportional to the color content in the color of object.

MOTOR BOARD TEST

Motor Board contains Motor Driver Modules with 12V Bidirectional Stepper Motors. To test the modules, we wrote subroutines for sending 8 step sequence data to motor driver using PORT A of 8255A. Pulses codes are explained in previous section of this report. Here is included a code snippet for Anti clockwise and Clockwise rotation of Stepper Motors.

```
C:
MVI D,03H
DRIVE_C:MVI A , 06H
OUT 00H
CALL DELAY
MVI A,05H
OUT 00H
CALL DELAY
MVI A,09H
OUT 00H
CALL DELAY
MVI A,0AH
```

```
OUT 00H
CALL DELAY
DCR D
JNZ DRIVE_C
JMP START

AC1:
MVI D,03H
DRIVE_AC1:MVI A , 0AH
OUT 00H
CALL DELAY
MVI A,09H
OUT 00H
CALL DELAY
MVI A,05H
OUT 00H
CALL DELAY
MVI A,06H
OUT 00H
CALL DELAY
DCR D
JNZ DRIVE_AC1
```

4.CONCLUSION

We succeeded in finishing our project though with some delay but with amazing results. Color Recognition based Sorting Machine prototyped using Intel 8085 Microprocessor is efficiently sorting three primary colors Red, Green and Blue as proposed before the project commencement. A wooden box of dimensions 13 x10 inch^2 is used to enclose the whole circuitry including motherboard, motor control board and Sensors. Workspace is made with a glass plate of dimensions 1.5 x 1.5 inch^2. A thermocol cube coated with primary colors is used as an object, which is placed on glass workspace. Infrared detects it and Color Sensor senses its color and accordingly signals are send to motors to rotate at particular angle to direct the object an again come back to original calibrated position. Supply needed to run our Sorted Machine is 12V DC which is given using an adaptor (220V AC to 12V DC) and a separate 5V DC power supply for Motherboard.

5. DOCUMENTATION

The whole documentation of project is done and this Report is prepared. All the necessary files including Schematic, Board layouts, datasheets needed to complete this project are included in GitHub Repository: https://github.com/MCodez/Sorting-Machine-Intel-8085-Microprocessor-Project

6. BIBILIOGRAPHY

Gaonkar, Ramesh S. *Microprocessor Architecture, Programming, and Applications with the 8085*. Upper Saddle River, NJ: Prentice Hall, 2002. Print.

http://www.alldatasheet.com

https://github.com/MCodez/Embedded-Systems