BALL SORTING MACHINE: Chaos.Bot 2004

AER201Y:Engineering Design

Instructor: **Dr. M. Reza Emami**

TA: **Ryan Martens**

Team: #60

Members: Jan-Hung Chen
Stephanie Elaine Gar-Wai Chiu
Aditya Ganti Mahapatruni

Engineering Science 0T6

Date: Tuesday, March 23rd 2004

ACKNOWLEDGMENTS

Team 60 would like to thank the following people/groups/inanimate objects (in no particular order):

Professor Emami for being helpful and attentive, Ryan Martens for sympathizing, Stephanie's family for listening to her rants, food, Aditya's uncle for providing transportation, fellow design teams for help and support, Microchip© for free PIC chips, food, Blue Tack© and Scotch tape for holding everything together, purveyors of cheap food, the patient staff at Supremetronic Inc., the Eng Sci computer lab regulars, food, the cold Canadian winter for keeping us awake, and food (did we mention that already?).

ABSTRACT

This report documents the technical aspects of the design process of a ball-sorting machine. The project was initiated by a request for proposal put forth by a sports utility retailer. The main functional requirements of the machine were that it had to sort up to twenty balls at a time into their different types: tennis, golf, squash, white ping-pong, and orange ping-pong. The machine was to be fully automated, with the balls being sorted in less than two minutes with the press of a start button. The solution involved coordination between three team members, each working on his or her own subsystem and eventually integrating the components together. These subsystems were the electromechanical, microcontroller, and circuits systems. The general concept for the design was a series of sorting ramps. The balls moved down the machine, from ramp to ramp, with one type being sorted out at each ramp. The final result for the most part met the goals set out at the beginning. The machine was simple and fast, utilizing gravity and sorting twenty assorted balls in less than thirty seconds. However, this resulted in less control over the flow of the balls and therefore some inconsistency in sorting and counting.

TABLE OF CONTENTS

Acknowledgements……………………………………………………………………….II
Abstract………………………………………………………………………………… III
Table of Contents……………………………………………………………………...IV
Abbreviations…………………………………………………………………………..VI
Notation………………………………………………………………………………..VI

CHAPTER 1…………………………………………………………………………...1
Introduction, Project Concept, Design Parameters, History, Limitations, Budget, Problem Division
1.1 Introduction……………………………………………………………………..2
1.2 Project Concept.………………………………………………………………...2
1.2.1 Acceptance Criteria and Design Parameters……………………………....2
1.3 Perspective…………………………………………...…………………….4
History and Background
1.4 Design Limitations………………………………………………………………5
1.5 Budget ……………………………………………………………………..8
1.5.1 Electromechanical expenditure…………………………………………….8
1.5.2 PIC Microcontroller expenditure…………………………………………..8
1.5.3 Circuits expenditure………………………………………………………..9
1.6 Division of the Problem………………………………………………………..10
1.6.1 Microcontroller……………………………………………………………10
1.6.2 Electromechanical…………………………………………………………10
1.6.3 Circuits…………………………………………………………………….10

CHAPTER 2: Electromechanical……………………………….….…………….11
2.1 Assessment of the problem……………………………………………......12
2.2 Solution……………………………………………………………………12
2.2.1 Loading bin mechanism……………………………………………………12
2.2.2 Sorting sequence………………………………………………………...13
2.2.3 Parallel sorting……………………………………………………………..14
2.2.4 Distinguishing white from orange ping-pong……………………………...14
2.2.5 Counting……………………………………………………………………15
2.2.6 Materials selection …………………………………………………………15
2.3 Suggestions for improvement………………………………………………15

CHAPTER 3: Circuits
3.1 Assessment of the problem……………………………………………………...19
3.2 Solution…………………………………………………………………………20
3.2.1 Counting……………………………………………………………………20
3.2.2 Actuators……………………………………………………………………22
3.2.3 PIC interface………………………………………………………………..22
3.2.4 Power distribution…………………………………………………………..23
3.3 Suggestions for improvement…………………………………………………23

CHAPTER 4: PIC Microcontroller
4.1 Assessment of the problem………………………………………..……………25

4.2 Solution……………………………………………………………………….25
4.2.1 Timer……………………………………………………………………….25
4.2.2 LCD Connection………………………………………………………...25
4.2.3 Keypad connection………………………………………………………26
4.2.4 PIC………………………………………………………………………….27
4.3 Suggestions for improvement…………………………………………...………….27

CHAPTER 5
Integration, Improvement suggestions, limitations, conclusions, Results
5.1 Integration………………………………………………………………………….29
5.1.1 Circuit mounting on machine frame………………………………………..29
5.1.2 Loading bin………………………………………………………………….29
5.1.3 Contact switch placement…………………………………………………..29
5.1.4 Integration of colour sensor box.……………………………………………30
5.1.5 PIC integration………………………………………………………………30
5.1.6 Ping-pong rate control………………………………………………………31
5.1.7 Debugging…………………………………………………………………..31
5.1.8 Testing……………………………………………………………………….31
5.2 System improvement suggestions…………………………………………………..32
5.3 Limitations and restrictions imposed by current methods………………………….33
5.4 Conclusions…………………………………………………………………………34
5.5 Results……………………………………………………………………………….34

CHAPTER 6:
6.1 References………………………………………………………………………….36
6.2 Bibliography………………………………………………………………………..36
6.3 Appendix A—Pin Descriptions…………………………………………………….A-1
6.4 Appendix B—Sample Calculations………………………………………………..B-1
6.5 Appendix C—Circuit Schematics…………………………………………………..C-1
6.6 Appendix D—Machine Schematics………………………………………………..D-1
6.7 Appendix E—Photos……………………………………………………………… …E-1
6.8 Appendix F—Standard Operating Procedure……………………………………….F-1
6.9 Appendix G--Microcontroller Algorithm and Code……………………………….G-1
6.10 Appendix H—Datasheets…………………………………………………………..H-1

ABBREVIATIONS

AC	Alternating Current
CMOS	Complementary Metal Oxide Semiconductor
DC	Direct Current
EPROM	Erasable Programmable Read Only Memory
IC	Integrated Chip
I/O	Input/Output
IR	Infrared
LCD	Liquid Crystal Display
LED	Light Emitting Diode
LS	Low power Schottky
PIC	Peripheral Interface Controller
RA0, RC2, etc.	Pins on input/output ports A, C, etc.
RAM	Random Access Memory
SPDT	Single-pole double-throw (switch)
TTL	Transistor-Transistor Logic

NOTATION

C	Capacitance, units are Farads [F]
R	Resistance, units are Ohms [Ω]
t_{high}	Time interval spent on high logic level, units are Seconds [s]
t_{low}	Time interval spent on low logic level, units are Second [s]

CHAPTER 1:

Introduction, Project Concept, Design Parameters, History, Limitations, Budget, Problem Division

1.1 INTRODUCTION

The task at hand, as requested by a sports utility retailer, was to build an automated machine that takes in an assortment of up to twenty balls made up of tennis, squash, golf, white ping-pong and orange ping-pong balls and sorts them according to type and colour. The entire operation was to be performed in under two minutes and run with only the push of a start button.

While it is possible to sort tennis, golf, squash and ping-pong balls by hand, a business would greatly benefit from an automated machine that performs this task accurately, such as the final result of this project. Obviously, customers expect inventory to be properly sorted and conveniently located at all times. Therefore, it is necessary to constantly ensure that the balls are sorted. Delegating this responsibility to employees means that they have less time to perform more important and less tedious tasks. An automated ball-sorting machine will free up time and get the task done quickly and accurately without supervision.

1.2 PROJECT CONCEPT

The final design of the machine sorts out each type of ball in different stages. Each stage of the sorting machine utilizes one characteristic to isolate one type of ball and some sort of counting mechanism to record the number of balls of that type that pass by. Each type of ball is collected in its own bin as the balls are sorted. The sorting operation begins with the press of a button and ends automatically when all the balls are sorted and counted. At the end of the operation, sorting statistics on the number of balls in each category, the total number of balls, and the total operation time are displayed depending on which buttons the user presses. All that is required to power the machine is to plug it into an AC wall socket.

1.2.1 Acceptance Criteria and Design Parameters

To reach these goals, specific objectives were needed to strive for the best design possible:

- Easy to use: The balls should be easy to load, the display should provide clear and simple communication with the user, and sorted balls should be easily retrievable
- Easy to produce: The design must be manufactured given a limited amount of time and resources
- Time efficient: The design should sort the balls as quickly as possible, since twenty balls must be sorted in two minutes

- Weight and size efficient: The machine can weigh no more than ten kilograms and must fit in a 1.5 x 1.5 x 1.5 m^3 envelope
- Energy efficient: Power consumption of the machine should be kept to a minimum
- Cost efficient: Given the budget constraints, design should consist of affordable components
- Safe: The machine should run smoothly and pose no hazard to the user
- Reliable: The machine must always sort the balls and give information accurately. The machine should never freeze or jam during operation. Also, the machine should be able to deliver repeatedly the same quality of results.
- Be equipped to handle all cases: The design must accommodate any combination of balls, including, for example, twenty tennis balls
- Simple and elegant: The design should be as simple as the goal allows, making it easier to fabricate at a high quality and easier to adjust or fix
- Fully automated: All that the user should have to do to operate the machine is load the balls into a bin and press a button on the keypad

1.3 PERSPECTIVE

History and Background

The availability of ball sorting machines presently is very limited. In fact, machines that sort tennis, golf, squash and ping-pong balls simply cannot be found. There exist devices that sort other types of balls and on different scales, but even these are few and far between. Examples include a machine that collects, washes, and sorts hollow plastic balls found in recreational ball pits and a simpler ball-sorting device consisting of a plastic container with a few sieve-like boundaries that sort out small objects into different levels. The most applicable designs prior to this project were built by students at the University of Melbourne. Their task was to sort tennis and ping-pong balls into groups of three, each with one tennis ball and two ping-pong balls. The most common construction materials used were cardboard and duct tape and the designs required no power. While this began to approach the level of complexity needed for the problem presented, our design encompasses more sorting mechanisms and calls for a more lasting prototype. It also includes means of communicating useful statistics from the sorting operation to the user.

1.4 DESIGN LIMITATIONS

A device was needed for the project that would make the machine intelligent so the user could communicate with it to get the sorting statistics and have better control of the machine. Presently, two popular solutions to this problem are the microprocessor and microcontroller.

Microprocessors and microcontrollers are, in one sense, the same thing. They all have an ASIC (Application Specific Integrated Circuit) that fetches and executes instructions based on the programs stored in them. These devices are controlled by software and have great flexibility in terms of their functions.

The first "microprocessor" was created by Intel in 1971. In the three decades since the invention of the first microprocessor, there has been tremendous development and innovation in this field of engineering. All kinds of microprocessors and microcontrollers have been invented and they all have different application spaces and features. A typical microprocessor or microcontroller includes the following: a CPU (central processing unit), RAM (Random Access Memory), EPROM/PROM/ROM (Erasable Programmable Read Only Memory), and I/O (Input/Output) ports with interrupt service.

Nowadays, microprocessors are often used for more advanced applications because of its high processing speed and ability of handle complex system. However, they also require additional system such as external RAM, ROM and I/O conversion.

Microcontrollers are typically used at what is called the "low-end" of computing since they are a lot slower compared to microprocessors. However, this does not mean that microcontrollers are less useful. They are designed to target specific applications which are self-contained and involve limited input and output. For the proposed design, which has these characteristics, microcontrollers are an ideal choice because of their low cost, low power consumption and simplicity.

Simpler integrated chips (IC's) are available as common circuit components as they are abundantly available and extremely affordable. There are two main families of chips in the market: transistor-transistor logic (TTL) and complementary metal oxide semiconductor (CMOS). They differ in the type of transistor used to drive the circuits (Course notes, 5-42), which has a large impact on some of their important characteristics. In general, CMOS chips can operate on a wider range of voltages and require less current and therefore less power to drive their circuitry. However, they are more sensitive to static electricity and can be damaged more easily. Also, they are not suited to the higher frequencies needed by the microcontroller. In this project, 7400 series TTL chips were used, and in particular, the low power Schottky (LS) variety. The only exception is the 4050 buffer. Despite the fact that this chip is a CMOS chip, it is still capable of driving TTL logic chips.

One chip central to this project is the NE555 timer. It is capable of producing a square wave signal of very precise pulse width based on an external resistor-capacitor (RC) circuit. As the RC circuit charges and discharges according to the time constant dependent on the resistor and capacitor used, the signal oscillates between the two logic levels (Course notes, 5-50). When the timer is used in astable mode, the time periods spent on each logic level are given below:

$$t_{high} = 0.693CR_1 \qquad \textbf{(1)}$$
$$t_{low} = 0.693CR_2 \qquad \textbf{(2)}$$

where R represent the resistor value and C represents the capacitor value. Circuits using this principle are described in the circuits subsystem, along with sample calculations.

Mechanical components such as D.C motors, servomotors and solenoids have been in use since the late 19th century. Miniature and specific mechanical components have to be produced and are used extensively in various industries. Modern applications include cars, airplanes and robots.

A Servo is a small device that incorporates a three wire DC motor, a gear train, a potentiometer, an integrated circuit, and an output shaft bearing. Of the three wires that stick out from the motor casing, one is for power, one is for ground, and one is a control input line. The shaft of the servo can be positioned to specific angular positions by sending a coded signal. As long as the coded signal exists on the input line, the servo will maintain the angular position of the shaft. If the coded signal changes, then the angular position of the shaft changes.

A solenoid is an electro-mechanical component that converts electrical energy into mechanical power. Electrical current is supplied to a tight coil and the resulting magnetic field is increased by surrounding the coil with a highly permeable iron frame. The magnetic field then acts upon a plunger, drawing it from its unpowered, extended position to a seated position against a backstop or pole piece. The linear force on the plunger from the magnetic field is extremely nonlinear with position, i.e. the force is relatively high immediately adjacent to the seated position and falls off rapidly with increased distance from the seated position.

Many rotary solenoids have the same fundamental design as a linear push-pull solenoid. Linear motion is translated into rotary motion via three small bearing balls that ride on an inclined plane as the plunger closes, converting a relatively small axial motion into a rotary stroke. True rotary solenoids operate on similar principles, but the magnetic arrangement allows for direct rotational motion with no attendant axial stroke. As with linear solenoids, the torque from the plunger is greatest near the energized, seated position. Short-stroke solenoids have, in general, greater starting torque than longer stroke solenoids of the same construction.

Materials selection and fabrication is something that can be learnt only through trial and error and experience. In this project, a first machine was built whose frame was made out of wood. Upon realization that it was too heavy, the design was changed to a metal frame because it was more elegant and lighter. The final prototype of the machine contains one

HS311 standard servo powered at 5V, one rotary solenoid powered at 12V and a DC motor powered at 12V, as well as metal for the frame, wood dowels, foamboard and coroplast ® corrugated plastic sheet. These components and materials are abundantly available and are easy to work with.

1.5 BUDGET

1.5.1 Electromechanical					
#	**Item**	**Quantity Used**	**Unit Cost(C$)**	**Total Cost (C$) (inc. Tax)**	**Supplier**
1	2x1/2x1/16" metal frame	3	6.90	23.81	Rona Home and Garden
2	3/8 x 36" wood square dowel	15	0.99	17.08	Canadian Tire
3	4-40 x ½"	2	1.29	2.96	
4	4-40 x ¾"	2	1.29	2.96	
5	Chain for loading bin	1	0.44	0.51	
6	Braces and brackets	12	0.29	4.00	
7	Loading Bin container	1	3.49	4.01	
8	Metal rod (to support loading bin)	1	0.75	0.86	
9	Foam board std. size	1	5.49	6.31	Grand and Toy
10	Collecting bins (varying sizes)	5	1.00	5.75	Dollarama
11	HITEC HS311 Standard Servo motor	1	14.00	14.00	Online www.hitecrcd.com
12	DC motor with gearbox	1	4.95	4.95	Active Surplus
13	30^o Rotary solenoid	1	2.95	2.95	
14	Coroplast plastic	1	0.75	0.75	
15	Contact switches with roller tips	3	1.00	3.45	Supremetronic Inc.
16	Miscellaneous copper and aluminium sheets		0.30	0.30	Machine shop
Electromechanical Subsystem Total				$95.35	
1.5.2 PIC Microcontroller					
17	PIC 16F877A with PIC Development board	1	30.00	30.00	Design Store, Microchip www.microchip.com
18	MM74C922N keypad encoder	1	6.00	6.00	Design Store
19	16x1 LCD display	1	6.00	6.00	
20	4x4 Keypad	1	4.00	4.00	
PIC Subsystem Total				$46.00	

1.5.3 Circuits					
#	**Item**	**Quantity Used**	**Unit Cost(C$)**	**Total Cost (C$) (inc. Tax)**	Supplier
21	74LS279 RS Latch	1	0.70	0.70	Suprementronic Inc, Active Surplus Electronics and Above all Electronics Surplus Ltd.
22	Superbright white LED	2	1.75	3.50	
23	CL138 Phototransistor	2	0.75	1.50	
24	100K trimmer potentiometer	2	1.50	3.00	
25	74LS14 Schmitt trigger inverter	2	0.70	1.40	
26	4050 Buffer	5	0.70	3.50	
27	LM358 OP AMP	1	0.70	0.70	
28	NE555 timer	4	0.85	3.40	
29	74LS157 quad 2 to 1 multiplier	1	0.70	0.71	
30	1N4148 diode	4	0.10	0.40	
31	1N4004 diode	1	0.15	0.15	
32	TIP 122 transistor	2	0.85	1.70	
33	TIP 30 transistor	1	0.85	0.85	
34	7404 NOT gate	2	0.70	1.40	
35	TIP 112 transistor	1	0.85	0.85	
36	Double header male strip	1	0.35	0.35	
37	8-pin IC socket	5	0.05	0.25	
38	14-pin IC socket	3	0.08	0.24	
39	16-pin IC socket	7	0.09	0.63	
40	18-pin IC socket	1	0.10	0.10	
41	Capacitor	10	0.10	1.00	
42	Resistor	30	0.01	0.30	
43	Power supply	1	10.00	10.00	
44	Circuit solder boards (IC type)	2	3.15	6.30	
45	Circuit solder boards (8x4 block type)	2	1.65	3.30	
46	Ribbon cable	4ft	0.35/ft	1.40	
47	Small mounting screws	30	0.02	0.60	
48	Solid wire	15ft	0.05/ft	0.75	
49	Stranded wire	75ft	0.05/ft	3.75	
50	3-pin molex connectors	4	0.20	0.80	
51	2-pin molex connectors	35	0.15	5.25	
	Circuits Subsystem total			**$58.78**	

GRAND TOTAL	**C$199.93**

The total amount spent on constructing the machine was C$199.93. This includes the amounts for each subsystem, and is within the proposed amount of $200.

1.6 DIVISION OF THE PROBLEM

On this project, the work was split up into three subsystems, with each team member assigned to one of the specialties. These subsystems were as follows:

> ***1.6.1*** *Microcontroller:* Jan, the microcontroller team member was responsible for writing the software for the microcontroller, a device which can store and execute a set of instructions. This included interfacing the keypad and LCD display, sending out control signals to appropriate mechanisms, taking in signals from counting circuits, count the balls based on these incoming signals, and count the amount of time elapsed during each sorting operation. After the program code was developed, Jan also aided the electromechanical member in constructing the physical aspects of the machine.
>
> ***1.6.2*** *Electromechanical:* The electromechanical member, Aditya, was in charge of building the machine's physical structure, incorporating actuating mechanisms into the operation of the machine, and determining positioning of components. Actuating mechanisms include opening the loading bin into which the unsorted balls are poured and any other moving parts that control the flow of the balls.
>
> ***1.6.3*** *Circuits:* This subsystem was the responsibility of Stephanie, who was to supply power to the machine and design and construct the circuits to interface with the microcontroller and drive the mechanisms. This included control of actuators and directing input from ball sensors to the microcontroller.

In the integration stage that followed the assembly of each subsystem, all team members worked to incorporate all the elements together and the distinctions became less important.

CHAPTER 2: Electromechanical Subsystem

2.1 ASSESSMENT OF THE PROBLEM

The ball sorting machine is expected to sort into separate containers each of tennis, golf, squash, white and orange ping-pong balls. Moreover, it must do so within 2 minutes, without generating loud sound. The loading bin mechanism must be a stocking mechanism as opposed to a sort-as-you-load mechanism. This means that balls are stored in a reserve until a button is pressed which actuates the sorting sequence(s). Sorting is to be done by size, weight and colour as applicable. A mechanism had to be implemented to put a suitable separation between ping-pong balls entering the colour sensor. Within the colour sensor itself, a mechanism to separate white ping-pong from orange ping-pong had to be implemented. A method had to be found to speed up the running time. Finally, the materials to be used in machine fabrication had to be lightweight and easily workable.

2.2 SOLUTION

1. 2.2.1 Loading bin mechanism: The mechanism was implemented using a store-bought loading bin as shown on the left. The bottom part of the loading bin was cut away, and a foamboard flap was attached in the opening using chains. This allowed the flap to be free hanging such that even a single ping-pong exerted enough force on it to move past it. The stocking mechanism was implemented using a HiTec HS-311 standard servomotor, powered at 5V. The servo is connected to an arm which sticks up and covers the middle of the flap.

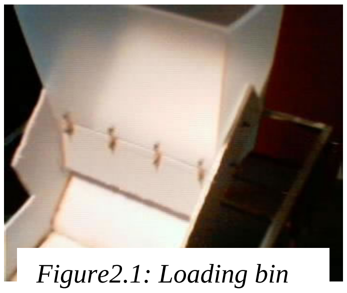


Figure2.1: Loading bin

When 20 assorted balls are loaded into the bin, taking configuration to be approximately 5 tennis (60g each), 4 golf (46g each), 3 squash (24g each) and 8 ping-pong balls (3g each), the total weight adds up to 580grams resting against the arm of the servo. Taking a factor of safety of 1.3 for a 3.8cm arm attached to the servo, a torque of 2.87kg.cm[1] is placed on the servo.

[1] Refer to appendix B for calculations

The torque for this model is rated at slightly higher than 3.0kg.cm[2] when powered at 5.0V. This torque is more than sufficient when an assorted combination of balls weighing 750g needs to be sorted.

2. <u>2.2.2 Sorting sequence:</u> When the servo opens, it turns 180^{o} and the stored balls flow out rapidly.
   The machine implements a 4-stage sorting sequence:

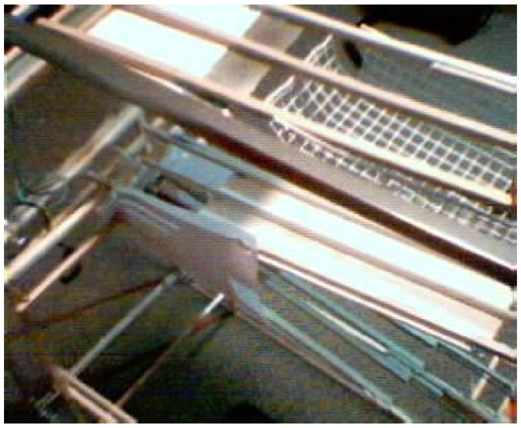


Figure 2.2: *3 stages of the sorting ramps*

   - **Stage 1**: Tennis ball is sorted from all other smaller balls. The picture to the right shows 3 of the 4 stages. The topmost ramp has rails that are wide enough for smaller balls, but allow tennis balls to continue on to a collec into the net and move onto stage 2.

   - **Stage 2**: On this ramp, a large piece of foam board is attached to the underside of the ramp to accelerate the balls. Golf is separated in a similar fashion to the tennis balls: the rails are big enough to allow squash and ping-pong to drop through, but not wide enough for the golf balls to drop through. Golf balls move onto a collector ramp.

   - **Stage 3**: Only squash and golf pass through from the previous two stages onto this ramp. Here, one pair of rubber bands per ramp is used as a weight sensor. They are stretched over long screws attached to the side of the ramp. When a squash ball (24grams) passes over this arrangement, its weight causes it to drop through the rubber bands and onto the collecting ramps below, while the ping-pong ball (3g) passes over the stretched rubber bands.

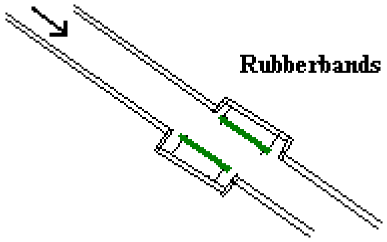


Fig 2.3: Squash ball sorting arrangement

[2] Refer to appendix for HS311 servo datasheet

Stage 4: Stage 4 is the colour sensor box. However, before the ping-pong balls move into the box, they are stored on a smaller ramp. A 12VD.C motor with gear combination is used to feed the balls one-by-one into the colour sensor box. The motor is attached to a Coroplast ® corrugated plastic cut into a H-shape as shown. Every time the hole in the H-shape aligns with the hole on the ping-pong storage ramp, one ping-pong ball drops through and goes into the colour sensor ramp. The motor rotates roughly at a speed allowing 2 ping-pong balls per second into the colour sensor box.

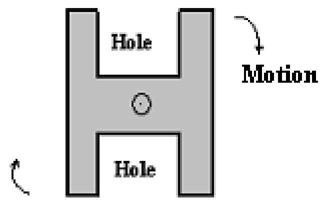


Fig. 2.4: Corrugated plastic of this shape is attached to the D.C motor

3. <u>2.2.3 Parallel sorting:</u> As identified under the problem assessment, time was an important criterion. When it was realized from our unsuccessful first design that sorting balls one-by-one was slow and caused jamming for later balls if the balls in front got stuck, a better parallel sorting idea was implemented. So when the loading bin servo is activated, balls pour onto two sets of ramps arranged side by side. This parallel sorting continues up to stage 3.

4. <u>2.2.4 Distinguishing white from Orange ping-pong:</u> Colour theory was used to implement sorting between white and orange ping-pong balls. A super bright white LED shines on the ball; reflected light from the ball shines on a phototransistor covered with blue filter paper. Why blue? Because blue is complimentary to orange. (e.g. white = Red + Green + Blue; Orange = Red+Green). A white ball would be registered by low resistance of the phototransistor while an orange ball would be characterized by much higher resistance.

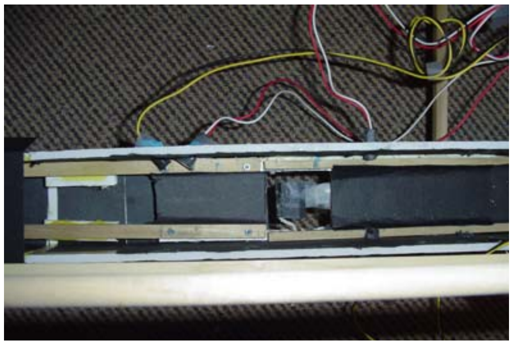


Fig. 2.5: *Colour sensor interior, showing positions of LED and phototransistor*

Depending on the resistance registered by the sensor, a rotary solenoid is activated (or not). When activated, the solenoid rotates 30 degrees and moves a paper flap, which allows the white ball to drop through but the orange ball continues uninterrupted.

5. 2.2.5 Counting: To count tennis, golf and squash balls, switches with a roller at the end of their tips were used on collecting ramps. White ping-pong is counted every time the solenoid activates. Orange ping-pong is counted via a break-beam sensor setup, where a continuously shining beam of light falling on a phototransistor is broken by the passing of an orange ball. When this happens, the count is incremented by 1.

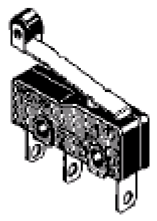


Fig. 2.6: Switch used for tennis, golf and squash

6. 2.2.6 Materials selection: Metal (Aluminium composite) was chosen because it was light, strong and relatively inexpensive compared to wood. Square wooden rods were used for ramps. Collecting bins and loading bin were store-bought, as were some of the intermediary ramps.

2.3 SUGGESTIONS FOR IMPROVEMENT

The servomotor used cannot support the weight of more than 12 tennis balls. Its current torque specifications are 3.0kg.cm for 5V. To operate 20 tennis balls, its torque should be 4.6kg.cm minimum.[3] This can be achieved using a more powerful servo or by using two of the same HS311 servos to open the loading bin so that each servo carries an effective weight of 10 tennis balls.

Design was changed and modified as seen fit. This lead to the use of different kinds of materials (e.g. foam board, coroplast ® plastic) for the same purpose in different parts of the machine. An improvement would be to make one material the standard for each job type so that the machine looks elegant and clean.

Having a one-by-one feeding mechanism right at the loading bin improves accuracy of sorting and completely eliminates jamming problems. If balls are released one-by-one to be sorted, the running time would be slower, but sorting accuracy would be improved close to 100%.

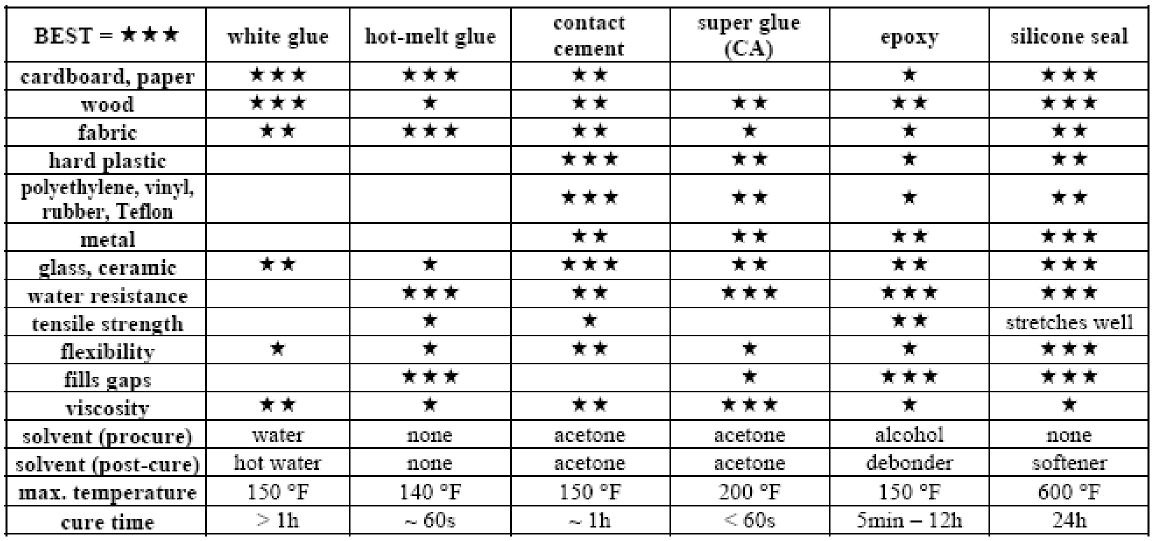
[3] See Appendix B for calculations

The problems presented above are summarized in the table below, showing salient features of the electromechanical subsystem.

Problem	Solutions considered	Solution chosen	Reason
Loading bin must be a stocking mechanism	• 12V DC motor-controlled gate. • Two solenoids to block either side of gate • Servomotor to block gate	Servomotor, powered at 5V	The servo was the only component that had the required torque to hold out against a loaded bin weighing 0.6kg
Increase separation between ping-pong balls going into the colour sensor	• Linear solenoid that activated once every second to let ball pass through • DC motor rotating at a certain frequency • Increasing the number of ramps to increase separation	DC motor	Implementation was easiest, and by means of a simple piece of corrugated plastic, the desired result was achieved
Distinguish white from orange ping-pong	• Linear solenoid • Rotary solenoid	Rotary solenoid	Rotary solenoid kept the design simple, and eliminated the need to stop the ball to detect colour. So colour could be detected even when the balls were in motion.

Table 2.1*: Comparison of major problems and solutions*

Contact cement was used to join foamboard to wood. Wood glue was used to attach wood dowels together.



BEST = ★★★	white glue	hot-melt glue	contact cement	super glue (CA)	epoxy	silicone seal
cardboard, paper	★★★	★★★	★★		★	★★★
wood	★★★	★	★★	★★	★★	★★★
fabric	★★	★★★	★★	★	★	★★
hard plastic			★★★	★★	★	★★
polyethylene, vinyl, rubber, Teflon			★★★	★★	★	★★
metal			★★	★★	★★	★★★
glass, ceramic	★★	★	★★★	★★	★★	★★★
water resistance		★★★	★★	★★★	★★★	★★★
tensile strength		★	★		★★	stretches well
flexibility	★	★	★★	★	★	★★★
fills gaps		★★★		★	★★★	★★★
viscosity	★★	★	★★	★★★	★	★
solvent (procure)	water	none	acetone	acetone	alcohol	none
solvent (post-cure)	hot water	none	acetone	acetone	debonder	softener
max. temperature	150 °F	140 °F	150 °F	200 °F	150 °F	600 °F
cure time	> 1h	~ 60s	~ 1h	< 60s	5min – 12h	24h

Table 2.2: *Comparison of adhesives. Contact cement and white wood glue were extensively used in machine fabrication*

CHAPTER 3: Circuits Subsystem

3.1 ASSESSMENT OF THE PROBLEM

The major tasks to be accomplished by the circuits subsystem were to interface with the PIC, drive the actuators, provide power, and assemble ball sensing mechanisms. The approach taken called for minimal input and output to and from the PIC, with as many of the circuit functions being independent as possible.

Counting

Since one of the objectives of the design was to count the number of each kind of ball that was sorted, some form of sensor and corresponding circuit was required for each type of ball. For the tennis, golf, and squash balls, one contact switch was employed for each. These consisted of a flat arm that closed the switch every time a ball rolled over it. A circuit was needed for each to transmit the signal to the PIC and to smooth any irregularities in the signal resulting from the contacts bouncing when brought together.

The only balls that had to be sorted using a method non-mechanical in nature were the white and orange ping-pong balls. This task fell to the circuits subsystem, as the goal here was to sort them without having to use the PIC to distinguish between the different coloured balls. The most apparent approach to this problem was to use a blue filter paper to filter out most of the light reflecting off of the orange balls, since blue and orange are complementary colours. Light from a superbright white LED reflecting off of a white ping-pong ball would still make it through the filter, since blue is a component of white light. Then, by using some sort of light detector, a difference in voltage signals set off by the two ball colours could be exploited. This had to be translated into a positive pulse signal to the PIC and the momentary opening of the rotary solenoid underneath the trapdoor leading to the white ping-pong ball bin. A similar but less complicated mechanism was needed to detect the passage of the sorted orange balls in order to count them. In this case, only the signal the PIC was needed. Simple contact switches could not be used due to the tiny weight of the ping-pong balls.

Actuators

The actuators that needed circuit control were the servo motor that acted as a gate for the loading bin, the DC gearhead motor that provided spacing for the ping-pong balls, and the rotary solenoid that allowed the white ping-pong balls to drop into their bin. The only input to the servo and DC motor circuits was a signal from the PIC that was high during the entire sorting operation and low at all other times. The servo circuit had to hold the arm stationary and blocking the loading bin door while the signal was low and then swing open by at least ninety degrees and hold while the signal was high. The operation of the DC motor was much simpler, with the motor running while the signal was high and stationary while the signal was low. Speed was not a concern because of the gears on the motor, so speed control was not necessary. As opposed to the motors, the control of the rotary solenoid was independent of the PIC. The movement of the solenoid was triggered by a light sensor that gave a positive pulse whenever a white ping-pong ball rolled by it. The rotary solenoid was to open for a set time and then close on its own.

PIC Interface

Interfacing with the PIC consisted of a few tasks. First of all, the PIC had to be connected to the keypad using a keypad encoder chip. The most essential task in terms of fulfilling the main design objectives was connecting all the inputs and outputs. Besides the keypad and LCD display, the only output from the PIC was the signal to the servo and DC motors. The PIC also had to be hooked up to five signals coming from each ball detector circuit. Finally, a means of counting the seconds of operation was needed, in accordance with the design guidelines.

Power Distribution

In order for the entire system to run, a means of supplying power to all the circuits, the PIC development board, and the actuators was needed. Also important for signal transmission was the need for a common ground among all the circuits.

3.2 SOLUTION

Components of the machine design constantly evolved over the course of the project life. An attempt is made here to summarize briefly the impact this had on the circuits subsystem, along with the more detailed explanations of the final circuit designs. All circuit schematic figures referred to in this section can be found in Appendix C, unless specified otherwise.

3.2.1 Counting

The contact switches used are simple single pole-double throw (SPDT) switches which connect the common (C) terminal with the normally open (NO) terminal when the switch is open and connect it with the normally closed (NC) terminal when the switch is pressed. This setup made the RS latch debouncing circuit easily adaptable to this situation. This circuit corresponds to Fig. 1. As the switches bounces towards and away from one of the poles, the initial signal from the switch being triggered is held. A single

RS latch 74LS279 chip provides four latches and therefore could be used for all three of the switch debouncing circuits. Low Power Schottky logic chips are used exclusively (aside from the 4050 buffers) in all the circuits for consistency and stability. Each debouncing circuit is connected to a switch and to the PIC using detachable molex connectors.

The number of viable options for the light sensor required as outlined in the previous section is limited. The three main affordable options were using a photoresistor, a phototransistor, or a photodiode. Of these, the phototransistor was chosen because of its affordability and ability to provide a sharp change in voltage with small changes in lighting through minimal adjustments. To detect the white balls, a phototransistor and a superbright white LED are set up in a reflector configuration (see Fig. 5 in Appendix E). Both the phototransistor and LED are covered on the sides to prevent unwanted effects from stray lighting and the whole arrangement is encased in a long covered tunnel to provide reliably dark conditions. The phototransistor is also covered with blue filter paper for reasons mentioned previously. The two items are positioned on the side of the tunnel such that when a ball crosses in front of them, the light from the LED reflects off of the ball and onto the phototransistor. At all other times, the phototransistor receives no light. The voltage drop across the transistor is greater when it receives no light and smaller when there is light incident on it. The circuit, shown in Fig. 2, therefore has the phototransistor in series with a resistor with the useful signal being the voltage drop across the resistor. A 100kΩ trimmer potentiometer is used as the resistor so as to make the signal adjustable as required for different lighting conditions. Increasing the resistance increase the voltage of the signal. The signal is then fed into an op amp in non-inverting amplifier configuration to make sure that the signal is at a suitable level to trigger a high logic signal. Finally, the analogue signal is shaped into a digital signal by the Schmitt trigger inverters. A pull-down resistor was necessary at the first inverter to trigger the right logic levels. Using two inverters produces a positive square pulse when white ping-pong balls rolls by the light sensor. This pulse is sent to the solenoid timing circuit and also through a buffer to the PIC to be counted.

The last type of ball to be counted is the orange ping-pong ball. A few options were also tested for this task. Among these were a discrete IR emitter-detector pair and a reflective optosensor. It was thought that using IR sensors would eliminate the need to shield the sensor from light. However, it was found that ambient visible light affected them too much. Since it was apparent that this sensor would also have to be encased, the safe option of using the same phototransistor and LED as for the white ball was employed. Instead of detecting the reflection off of the balls, the two are positioned across from each other so that when a ball rolls between them, the beam of light is broken and no light falls on the phototransistor. The circuit (Fig. 2) is essentially the same, except that the amplifier is not needed due to the large difference in voltage between the two signals. The final signal needs only to be sent to the PIC.

3.2.2 Actuators

The loading bin mechanism was changed twice during the course of the project. Originally, there was a rotating paddle powered by a DC motor. Later, this was scrapped in favour of a gate that was allowed to open when two linear solenoids retracted. However, the solenoid did not have a long enough stroke. Finally, a servo motor replaced this setup. The servo motor uses pulse width modulation to determine the arm position. A different square pulse width is needed for each of the two arm positions. Therefore, two NE555 timer chips are fed into a multiplexer whose select pin is controlled by the PIC output signal (see Fig. 3). Depending on whether the signal is high or low, the multiplexer lets through the input from one timer or the other. The resistor connected to +5V and the capacitor connected to pin 6 are the parameters that affect the pulse with. The calculations to determine the approximate value needed are shown in Appendix B. The precise values were found by testing, with the resistances being varied. Although the motor was tested using a 90° rotation, the final circuit produced a 180° rotation, with the resistances shaping the pulse widths so that the motor reached the end of its turning range at each position.

Also controlled by the PIC output signal is the DC motor used for spacing the ping-pong balls. The circuit (Fig. 4) is simple and merely allows current to flow through the motor at 5 volts when the signal is high. This is accomplished using a TIP122 transistor chosen for its high power tolerance.

The rotary solenoid presented a more substantial challenge in terms of supplying power and control. The solenoid would not work on any less than 24 volts, so both the +12V/-12V lines were needed. A combination of an NPN and a PNP transistor is needed since each transistor alone cannot handle a negative voltage (Fig. 5). A clamping diode is needed to prevent voltage spikes. Another challenge was in causing the solenoid to open for a set amount of time (about one second) based on a short pulse from the white ball light sensor circuit. The solution was to use a oneshot NE555 timer circuit (Fig. 6). The pulse trigger the timer via a TIP112 transistor and the timer subsequently sends out a pulse of length determined by the timing resistor and capacitor to the solenoid.

3.2.3 PIC Interface

Keypad input is connected to the PIC through a circuit for the MM74C922N keypad encoder chip. The circuit is shown in Fig.7. The keypad is connected by a ribbon cable and the inputs A to D are connected to the appropriate pins on the PIC, also by means of a ribbon cable. The pin assignments for the ribbon are given in Table 2 in Appendix A. The output and inputs are connected to the PIC in the same way. Each input and output has a 1kΩ resistor in series to limit the current sunk and sourced by the PIC port. An NE555 timing circuit in astable mode (Fig. 8) provides a short pulse to one of the PIC pins every second. The calculations for the pulse width and frequency are similar to those for the servo circuit.

3.2.4 Power Distribution

A surplus AT computer power supply provides all the circuits and actuators with power. Two of the +5V/GND lines are connected to two sets of circuits to power the logic components and the motors. These two sets of circuits are also connected to each other to ensure a common ground is established among all the circuits. The PIC development board requires a separate +12V/GND line and the rotary solenoid uses a +12V/-12V line.

3.3 SUGGESTIONS FOR IMPROVEMENT

The actuators are the components that could use the most improvement. The servo motor circuit did not work as well as it did on its breadboard prototype, but there was insufficient time to remedy the situation. As the result, the motor control was adequate, but the angle control could have been better. The DC gearhead motor circuit also suffered from time restrictions. The circuit used is sufficient, especially considering that the motor almost never runs for more than 30 seconds straight. However, introducing pulse width modulation through a 555 timer would have allowed for speed control.

There is also a minor problem with interference on the PIC. On occasion, the display flickers after the operation is finished to show a different ball result even when the keypad remains untouched. The most effective ways of minimizing this were the addition of buffers on certain input lines and grounding unused pins in ports A and C. However, this did not completely eliminate the problem. An improvement for the future would be to find a way to fix the problem so that the display becomes stable.

CHAPTER 4: PIC Microcontroller Subsystem

4.1 ASSESSMENT OF THE PROBLEM

The ball sorter machine is expected to inform the user of the sorting statistics (the total number of balls, number in each category and the overall operation time). The user has to be able to communicate with the machine through a keypad to start the machine and to review the sorting statistics. Also, the machine has to stop after all the balls are sorted. Therefore the problems are: how to implement the timer that can display the operation time, how to count number of balls, how to store the information so the user can review the statistics after the balls are sorted and when and how to stop the operation.

4.2 SOLUTION

A device is needed that can be programmed to control the machine intelligently and store the sorting statistics. For the complexity of the problem, a microcontroller would be an ideal choice because we are targeting for a particular application and it is cheap and consumes little power.

A PIC16F877A was used as the microcontroller with a 16×1 LCD display and a 4×4 keypad as the interface between the machine and the user. The keypad is connected to the PIC through the MM74C922N keypad encoder. When the user presses the “start” button on the keypad, the PIC will send a signal to the circuit to start the machine. Once the “start” button is pressed, the timer begins. We build a timer circuit with a 555 timer that sends signals to the PIC every second. The PIC then displays the number of signals received on the LCD as the operation time. There are also counters such as switches that send signals to the PIC when the balls trigger them. The PIC then stores the number of signals received from each counter. Once the machine is stared, the PIC will wait for signals from the counters for 15 seconds. If no signal is sent in 15 seconds, the PIC sends a signal to the circuit to stop the operation and displays the “finish” message on the LCD. On the other hand, every time the PIC receives a signal, it will reset the waiting time to 5 seconds. After the waiting time, PIC sends a signal to stop the operation and display “finish” message on LCD. At this stage, the user has access to information (total number of balls, number in each category and the overall operation time) stored in the PIC through the keypad and the LCD.

1. **4.2.1** Timer:
   The timer circuit is covered in detail in the circuit subsystem section.

2. **4.2.2** LCD connection:
   The standard 16×1 LCD has a 14-pin interface: 8 data lines (D_0 to D_7), 3 control lines (RS, W/R, E), and 3 power lines (V_{DD}, $V_{SS,}$ V_{EE}). V_{DD} (pin 2) and V_{SS} (pin 1) are the module’s positive and negative power supply leads. V_{EE} (pin 3) is the display contrast control. A potentiometer placed between supply voltages, with its wiper connected to V_{EE}, allows for manual adjustment of the contrast of the display. D_0 to D_7 (pin 7 to 14) are the data bus lines. The LCD is connected to the PIC through 4-bit

data transfer mode. Therefore, only four data lines (D_4 to D_7) are used. The data is sent to the PIC as two 4-bit numbers.
When RS (pin 4) is low, data types transferred to the display are interpreted as commands such as clear display. When RS is set high, character data can be transferred to the display. For this project, we only need to write to the display, so the R/W is set low. In order to send data to the display successfully, the E (enable control input) line is also needed. When writing to the display, data on the D_4 to D_7 lines is transferred to the display when the enable input receives a high-to-low transition.
Therefore, one subroutine for writing commands and one subroutine for writing data to the display are needed. For both subroutines, a short high-to-low transition for the E line is given.

3. **4.2.3** Keypad connection:

The 4×4 keypad is connected to the PIC through an encoder. The MM74C922N translates the pressed key on the keypad into a 4-bit binary number, which is then sent to PORTA (RA0 to RA3) of the PIC. Based on the binary number received, the PIC responds accordingly. The encoder assumes that the keys of the keypad are labeled from 0 to 15 starting with 0 at the 1st row and 1st column and ending with 15 at the 4th row and 4th column.

4. **4.2.4** PIC:

The program contains three main sections. The first part is to wait for the user to press the start button to start the machine. After the machine starts, the program moves on to the next section. In the second part, the PIC polls the timer circuit and all the counters and stores the numbers of signals received from each counter and the timer. If the PIC receives no signal from the counter for a period of time, the program will exit this part, display "finish" on the LCD and moves to the last section. The last section of the program is just a loop that waits for the 4-bit binary numbers from the keypad encoder. According to the request from the user, the PIC sends the data to the LCD to display the sorting statistics.

Pins	Function	Pins	Function
RA0	Takes input from the keypad through the encoder.	RC4	White ping-pong ball counter
RA1		RC5	Orange ping-pong ball counter
RA2		RD2	RS line for the LCD
RA3		RD3	E line for the LCD
RC0	Receives signal from the timer circuit.	RD4	Data lines for the LCD
RC1	Tennis ball counter	RD5	
RC2	Golf ball counter	RD6	
RC3	Squash ball counter	RD7	

4.3 SUGGESTIONS FOR IMPROVEMENT

The program we have now can only hold information for one run. The machine has to be reset for the next run and the information would be lost. One improvement that can be made is to make the display hold a longer log (of the previous operations) with more information. For example, after 3 runs, the user has access to the sorting statistics for the 3 individual runs.

Also, once the emergency button is pressed, the machine stops, but all the information is lost too. It could be re-designed so that even when the operation is interrupted, the PIC will retain the information. This way, enables us to know how many balls have been sorted and how long the machine had operated before it was stopped.

CHAPTER 5

Integration, Improvement suggestions, Limitations, Conclusions, Results

5.1 INTEGRATION

Integration proved to be a bigger challenge than initially thought. Throughout the stages where all three subsystems were coming together, it was found that several major and numerous minor changes had to be made to keep the machine accurate to design specifications. These stages of integration, along with problems and solutions are presented here.

1. **5.1.1** Circuit mounting on machine frame: Individual circuit boards were screwed onto larger coroplast® corrugated plastic sheets, which were then fixed to the metal frame of the machine. The plastic sheets were attached only to one side of the machine to give it an organized look.

2. **5.1.2** Loading bin: Although it was initially proposed that a 6V D.C motor would be used to operate the door of the loading bin, it was later found to be ineffective because of frequent jamming of balls. This D.C motor was then replaced by two linear 12V solenoids, each one placed on either side of a re-designed loading bin. When powered, the solenoids would retract and let a flap be freely opened by the balls. However, this idea, although better than the previous, could not give 100% efficiency because the combination of solenoids did not have enough force to retract against the weight of 20 assorted balls (estimated at 0.6kgs). Applying oil could not solve the problem because of point source contact. A third solution, which worked beautifully, was to use a servomotor (HS311 standard) to block the gate. When powered at 5.0 V, it had sufficient torque (rated at 3.0 kg.cm) to move against the weight of the assorted balls.

3. **5.1.3** Contact switch placement: Contact switches had to be positioned with great care because they provided the basis of counting when a ball rolled over them. For tennis, golf and squash, the proposed idea was to use switches with a lever (longer lever for tennis, shorter for golf and squash), which when pressed, send a signal to the PIC to register a count. There was the obvious problem of balls not clicking the lever. The solution implemented was to use the same type of switches, but with little rollers at the tip of the levers to ensure better clicking contact.

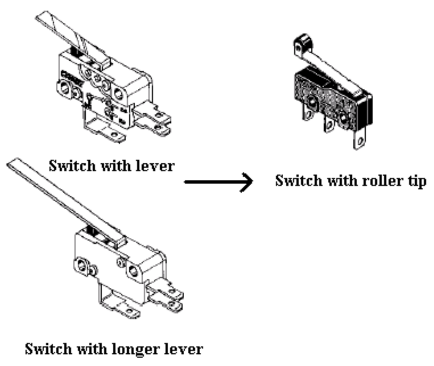


Figure showing the old switches on the left and the new switch on the right of the arrow

4. **5.1.4** Integration of colour sensor box: Two pairs of white LED and phototransistor had to be placed inside the box. One pair was be used to detect colour and activate the rotary solenoid, while the other pair was used to count orange ping-pong balls. The whole box itself had to be positioned at an incline so that the ping-pong balls going inside had the correct speed to be detected by the sensors. A hole was cut on the bottom of the box to place the arm of the rotary solenoid. A paper flap was attached to this arm. Different kinds of paper were used to make the flap which would open for white ping-pong and remain closed when an orange ping-pong was detected. Black construction paper was finally agreed upon, with its edges strengthened with tape.

5. **5.1.5** PIC integration: The PIC development board, LCD display, and keypad were mounted easily on a piece of corrugated plastic. Integration with the circuits and the microcontroller was straightforward. The main tasks were to power the PIC development board, connect the inputs and outputs to the proper pins, and connect the keypad and LCD display to the PIC development board. A simple DC plug was used to connect the development board to a +12V/GND line. A double row header strip with a total of 34 pins that the main data bus can connect to is soldered onto a circuit board. That way, the inputs and outputs can be connected to that circuit using detachable molex connectors. Small series resistors are located on each input and output line to limit the current reaching the PIC, since there is a limit on the amount of current each port can sink or source. Also, each input or output line leads to either a 4050 buffer or logic chip, adding another line of protection for the PIC in case of a current surge. On the same circuit board is another double row of pins for the keypad to hook up to using a ribbon cable. The inputs from this are used in the keypad encoder circuit. The LCD display is also connected to the development board using a ribbon cable. The single row of pins on the LCD display had to be connected to a double row of pins of the development board.

   The main problems in integration of the PIC and circuits involved the input pins. It was found that the counts for the balls was not always accurate, and that sometimes one input signal would remain high for the entire time. This was remedied by placing buffers on the white and orange ball signal lines to keep the signals as close to ground voltage as possible when the signal was low. Another major problem was erratic behaviour of the program, as identified by unpredictable microcontroller program erasure and flickering of the display. It was deduced that the LCD display connection was fine and that the problem was with the PIC itself. The problem was also reduced by use of the buffers. However, the flickering problem was still an issue. Power lines nearby were twisted and placed away from the PIC circuit. Also, the unused pins in ports A and C were grounded. This went a long way in reducing the flickering problem, although it stills showed up every once in a while. This problem prevented the use of the reset option originally planned for the microcontroller program since unpredictable flickering sometimes resulted in premature reset of the program.

Not only would eliminating the problem improve the post-operation display, it would allow the reintroduction of the reset function.

6. **5.1.6** Ping-pong rate control: Numerous non-electrical ways were tried to increase the separation of the ping-pong balls going into the colour sensor. These ranged from putting more ramps before the colour sensors to making minute adjustments to try and make ping-pong balls spin down from the ramps rather than drop down. However, these methods only produced moderate success. A solenoid was then decided upon. Controlling the solenoid activation at a specific rate would allow one ball through at the specified rate. However, solenoids suited for this job were sold out at Active Surplus. A final decision was taken: use a D.C motor with gears to achieve the required speed. This idea was vaguely similar to the feeding mechanism of the coin sorter from Assignment 1: Reverse engineering.
   Refer to appendix D, fig.4

7. **5.1.7** Debugging: Program code was modified slightly when the PIC was found to stop too early, mid-way through sorting. The delay between last ball count and program termination was increased from 4sec to 5sec. Moreover, if no balls were put in and the program was started, the PIC would terminate after 15seconds. This allowed termination in the unlikely event of mass jamming early on.

8. **5.1.8** Testing: When the machine was tested with 20 assorted balls, there were numerous problems relating to jamming, flying balls and mis-sorting.
   - To correct this, the first ramp was made slightly steeper and wider so that two tennis balls rolling side by side did not jam. The space where all the smaller (golf, squash, ping-pong) balls were collected after tennis had been sorted was made larger. This allowed any fast-moving ping-pong ball to be collected even when a tennis ball was pushing it.
   - The ramp configuration leading to the collection of golf balls had to be modified. A file was used to trim down the dowel height.
   - Higher walls had to be put in place along the ramps to prevent balls flying.
   - The colour sensor had to be calibrated several times to accommodate the changes in the other parts of the machine.
   - Various other trimming, filing and modifications were made to improve accuracy of counting and sorting.

5.2 SYSTEM IMPROVEMENT SUGGESTIONS

- Instead of the collecting ramp for squash, a slide would be better because squash balls are sticky. On the first run during the competition, two squash balls, already sorted, just had to drop into their collecting bins, but got stuck to each other on the ramp.

- Incorporating a reset function on the keypad rather than using the reset button on the PIC development board would make it more user-friendly.

- On an aesthetic note, different, eclectic materials were used for the same reason, e.g., to make boundary walls to prevent balls from flying out. These walls could be made of the same material to give a clean, uniform look to the machine.

- More than 20 balls could easily sorted with 1 servomotor blocking the gate. However, this can be extended to two servomotors, which would place less stress on each individual servo, thereby prolonging the servo’s life. Moreover, the existing design can be extended so that the loading bin is a two-stage mechanism, whereby half the balls in the loading bin get sorted when the first servo opens, and a few seconds later, when the second servo opens the rest of the balls are released.

- Enclosing circuits so that they are not exposed.

- Using thicker construction paper for the colour sensors. On the second run during the competition, the sorting was almost perfect, except that an orange ping-pong got sorted into the white collecting bin. This was because of direct sunlight shining on the colour sensor box, possibly messing up the calibration.

- Incorporating memory into the PIC so that information from one run can be stored and referred to a short while later. I.e., a second run would not erase the previous data.

5.3 LIMITATIONS AND RESTRICTIONS IMPOSED BY CURRENT METHODS

- While all other sorting stages of the machine are done in parallel, there is only one colour sensor box. This causes ping-pong balls to be delayed as they wait to be sorted one-by-one. Having another set of colour sensors could improve this delay, but this makes the machine that much more expensive.

- The current method for counting tennis, golf and squash balls is by using switches with a roller attached at the tip (see integration section). This requires the ball to roll such that the switch is pressed as the ball goes over it.

- Solenoids drain too much power and could only be used sparingly on the machine.

- The PIC can now only hold data from one run which is lost in case of power failure. Moreover, when the emergency stop is pressed, all information is lost.

5.4 CONCLUSIONS

The goal of this project was to build a working prototype of a machine that could sort balls of different sizes and colours (tennis, golf, squash, orange and white ping-pong). Within the proposed constraints of cost, power and weight, a first machine was designed and partly constructed but it was found to be over the 10kg limit. Following this, a new design that retained parts of the old design and incorporated newfound knowledge was created. This new machine, at 7.3 kg, sorted two balls in parallel, compared with the one-by-one sorting of the first machine. This cut down running time in half.

Utilizing the force of gravity to drive the balls minimized the number of moving parts. This increased simplicity, reliability, ease of construction, and cost effectiveness. Sorting the balls by size is the fastest and simplest method. However, since the ping-pong and squash balls are the same size, this method was applied to sort out the tennis and golf balls only. Because a squash ball is many times heavier than a ping-pong ball, they were easily separated by weight using rubber bands. Finally, using a colour sensor separated the white and orange ping-pong balls.

However, using gravity to drive the balls meant the user had no control over them. This somewhat limited the sorting and caused occasional mis-sorting.

5.5 RESULTS

The machine sorts balls fast and is lightweight at 7.3 kg. Because of the parallel sorting mechanism, the machine is able to sort 20 balls in well under 30 seconds. During the competition, the first run was clocked at 20 seconds and the second run was 19 seconds, which were relatively faster than other machines present there. The large capacity of the loading bin allows more than 20 balls to be sorted, with that number higher if using smaller balls only. The contact switches work perfectly with the circuits but a count may be missed if the switch is pressed too quickly in succession. Even with a debouncer circuit, this problem could not be solved. However, in typical runs, the distance between balls is sufficiently great that they are properly registered.

CHAPTER 6:

References, Bibliography. Tables and Appendices.

6.1 REFERENCES

[1] J. Richard Hollrock, "Apparatus for washing and sorting plastic balls," United States Patent and Trademark Office. http://www.uspto.gov

[2] "Giant Sieve Sorter," Exploratorium. http://www.exploratorium.edu/snacks/giant_sieve_sorter/

[3] John Weir, "Level 300 Design," Engineering Design Melbourne. http://www.mame.mu.oz.au/eng_design/learning/level300.html

[4] M. Reza Emami, *AER201Y Engineering Design Course Manual*, 4th ed., Sep. 2003, p.5-49.

[5] M. Reza Emami, *AER201Y Engineering Design Course Manual*, 4th ed., Sep. 2003, p.5-60.

[6] M. Reza Emami, *AER201Y Engineering Design Course Manual*, 4th ed., Sep. 2003, p.5-51.

[7] R. Carter, "Re: Need 5V Driver Circuit for +12/-12V Solenoid." Online Posting, The Seattle Robotics Society. 1 July 2002 http://groups.yahoo.com/group/SeattleRobotics/message/11271

[8] R. Paisley, "LM555 and LM556 Timer Circuits." http://home.cogeco.ca/~rpaisley4/LM555.html

6.2 BIBLIOGRAPHY

Bill Davies, *Practical Robotics*, Richmond Hill: WERD Technology, 1997.

Listing of 7400 series logic chip datasheets. http://www.fe.up.pt/~victorm/TTL.htm

M. Reza Emami, *AER201Y Engineering Design Course Manual*, 4th ed., Sep. 2003.

Phillips Semiconductors product datasheets. http://www.philipslogic.com/products/

6.3 Appendix A – Pin Descriptions

Circuit Connector Descriptions

Connections are split up into the actual physical circuit boards on which each connection was located. All board-to-board connections were made using 2 or 3 pin small molex connectors which were detachable. Pictures of circuit boards can be found in Appendix E.

PIC Circuit Board (see Fig. 6)

1) +5V/GND in from power supply
2) +5V/GND out to DC motor circuit
3) +5V/GND out to switch debouncer circuit
4) signal in from tennis ball debouncer (RC1)
5) signal in from golf ball debouncer (RC2)
6) signal in from squash ball debouncer (RC3)
7) signal in from white ping-pong ball counter(RC4)
8) signal in from orange ping-pong ball counter(RC5)
9) signal out to DC motor circuit (RC7)
10) ribbon cable connected to keypad
11) ribbon cable (data bus) connected to PIC development board

DC Motor Circuit Board (see Fig. 6)

1) +5V/GND in from PIC circuit
2) +5V/GND out to servo motor circuit
3) signal in from PIC circuit (RC7)
4) signal out to servo motor circuit
5) DC motor leads

Servo Motor Circuit Board (see Fig. 6)

1) +5V/GND in from DC motor circuit
2) signal in from DC motor circuit
3) +5V/GND and signal out to servo motor

Switch Debouncer Circuit Board (for tennis, golf, squash) (see Fig. 9)

1) +5V/GND in from white ping-pong ball signal circuit
2) +5V/GND out to PIC circuit

3) C/NO/NC in from tennis ball switch
4) signal out to PIC circuit (RC1) for tennis ball counter
5) C/NO/NC in from golf ball switch
6) signal out to PIC circuit (RC2) for golf ball counter
7) C/NO/NC in from squash ball switch
8) signal out to PIC circuit (RC3) for squash ball counter

White Ping-pong ball Signal Circuit Board (includes rotary solenoid timing circuit, see Fig. 7)

1) +5V/GND in from power supply
2) +5V/GND out to phototransistor circuit
3) +5V/GND out to switch debouncer circuit
4) +5V/GND out to rotary solenoid circuit
5) signal in from phototransistor circuit
6) signal out to PIC (RC4) for white ball counter
7) signal out to rotary solenoid circuit

Phototransistor Circuit Board (include phototransistors and LEDs, see Fig. 7)

1) +5V/GND in from white ping-pong ball signal circuit
2) signal out to white ping-pong ball signal circuit
3) signal out to PIC (RC5) for orange ball counter
4) white ping-pong ball phototransistor leads
5) white ping-pong ball LED leads
6) orange ping-pong ball phototransistor leads
7) orange ping-pong ball LED leads

Rotary Solenoid Circuit Board (see Fig. 8)

1) +5V/GND in from white ping-pong ball signal circuit
2) +12V/-12V in from power supply
3) signal in from white ping-pong ball circuit
4) rotary solenoid leads

PIC Connections

The PIC inputs and outputs were connected through a 34-pin data bus, with the locations of each pin on the PIC circuit board shown below. Descriptions of each pin are also shown below

					RC6	RC4	RC2	RC0						RA4	RA2	RA0
					RC7	RC5	RC3	RC1						RA5	RA3	RA1

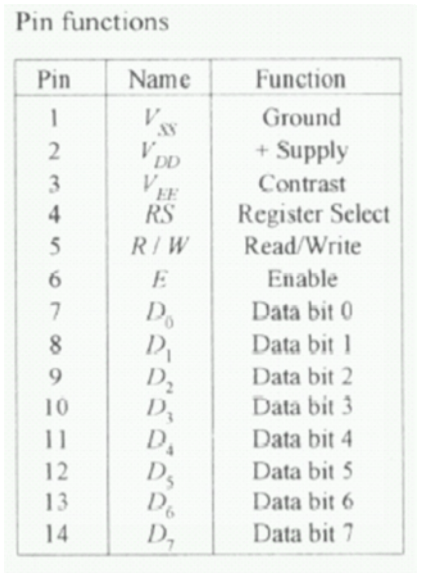
Microcontroller Data Bus Pin Descriptions

Table 1

Pin	**Connection**
RC0	PIC second timer
RC1	Tennis ball counter
RC2	Golf ball counter
RC3	Squash ball counter
RC4	White ping-pong ball counter
RC5	Orange ping-pong ball counter
RC6	Grounded
RC7	Signal out to DC and servo motor circuits
RA0	Data A on keypad encoder
RA1	Data B on keypad encoder
RA2	Data C on keypad encoder
RA3	Data D on keypad encoder
RA4	Grounded
RA5	Grounded
RD2	RS line for the LCD
RD3	E line for the LCD
RD4-RD7	Data lines for the LCD

LCD Display Pin Descriptions

Table 2

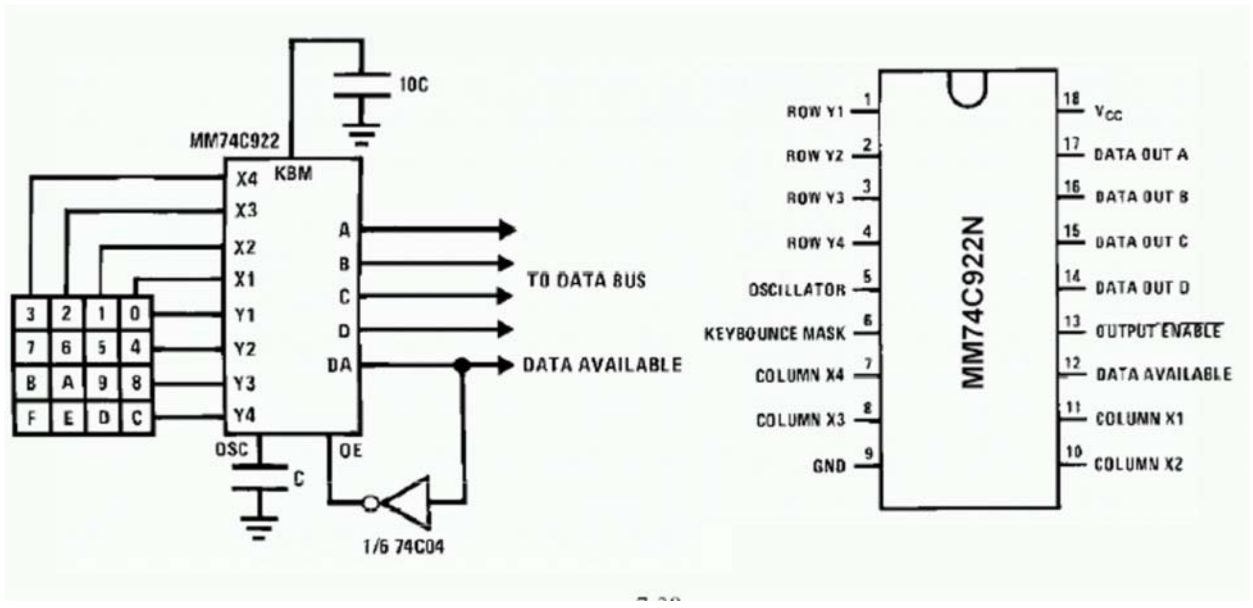


Pin functions

Pin	Name	Function
1	V_{SS}	Ground
2	V_{DD}	+ Supply
3	V_{EE}	Contrast
4	RS	Register Select
5	R / W	Read/Write
6	E	Enable
7	D_0	Data bit 0
8	D_1	Data bit 1
9	D_2	Data bit 2
10	D_3	Data bit 3
11	D_4	Data bit 4
12	D_5	Data bit 5
13	D_6	Data bit 6
14	D_7	Data bit 7

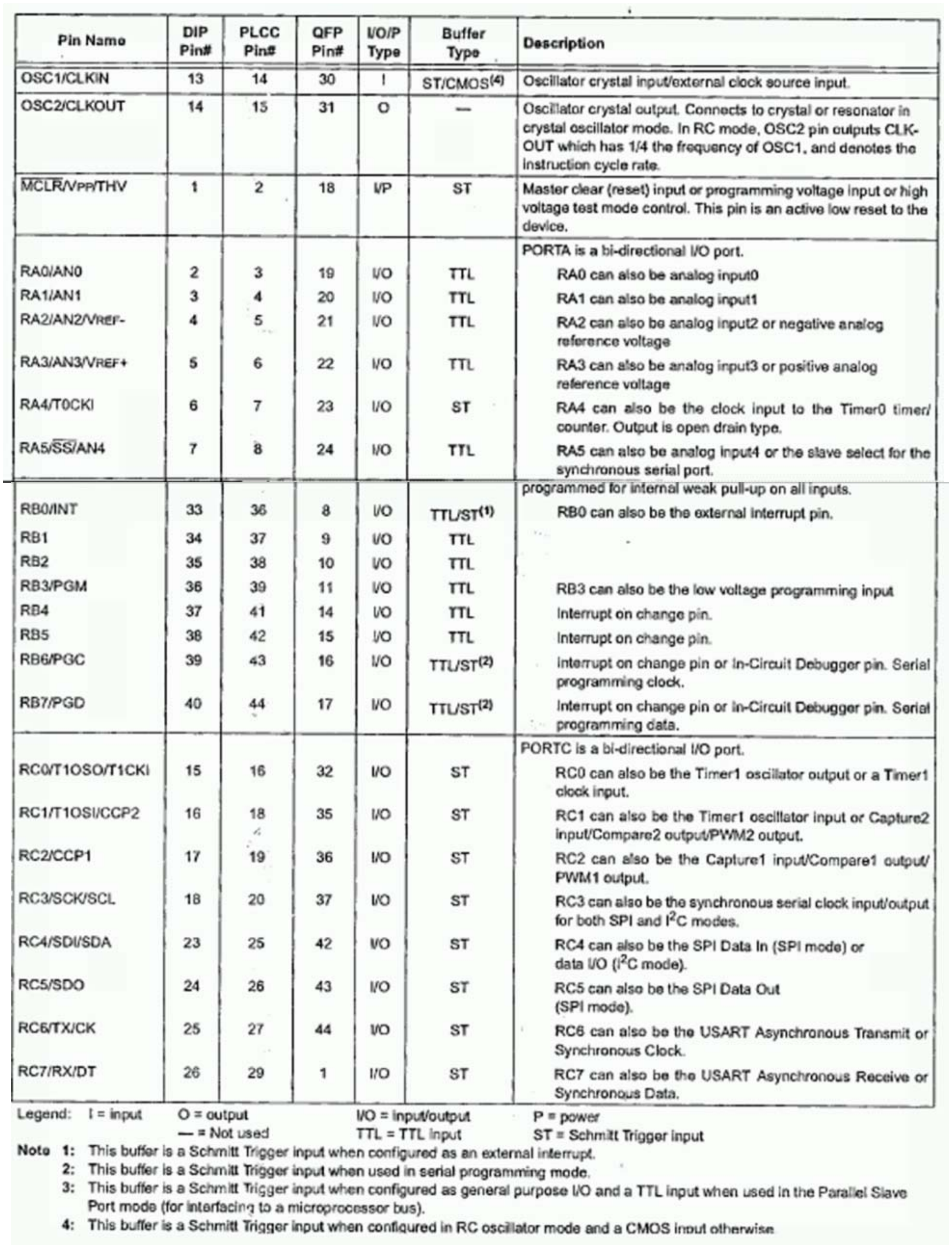
Keypad Encoder Circuit and Pin Descriptions

Fig. 1



PIC16F877 Tables

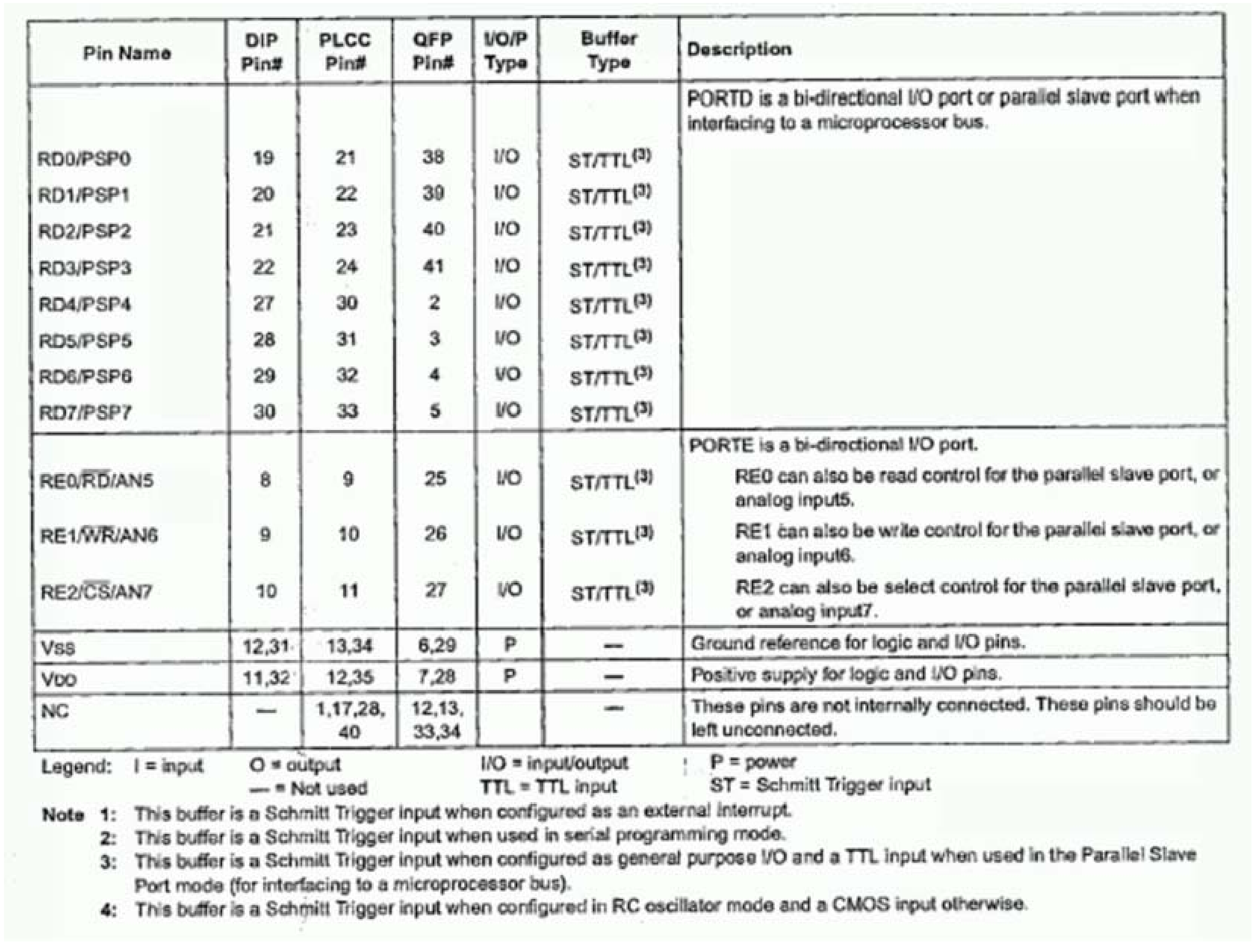
Table 3. Complete pin-out description. (cont'd on next page)



Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	13	14	30	I	ST/CMOS[(4)]	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	14	15	31	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLK-OUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
$\overline{\text{MCLR}}$/VPP/THV	1	2	18	I/P	ST	Master clear (reset) input or programming voltage input or high voltage test mode control. This pin is an active low reset to the device.
						PORTA is a bi-directional I/O port.
RA0/AN0	2	3	19	I/O	TTL	RA0 can also be analog input0
RA1/AN1	3	4	20	I/O	TTL	RA1 can also be analog input1
RA2/AN2/VREF-	4	5	21	I/O	TTL	RA2 can also be analog input2 or negative analog reference voltage
RA3/AN3/VREF+	5	6	22	I/O	TTL	RA3 can also be analog input3 or positive analog reference voltage
RA4/T0CKI	6	7	23	I/O	ST	RA4 can also be the clock input to the Timer0 timer/counter. Output is open drain type.
RA5/$\overline{\text{SS}}$/AN4	7	8	24	I/O	TTL	RA5 can also be analog input4 or the slave select for the synchronous serial port.
						programmed for internal weak pull-up on all inputs.
RB0/INT	33	36	8	I/O	TTL/ST[(1)]	RB0 can also be the external interrupt pin.
RB1	34	37	9	I/O	TTL	
RB2	35	38	10	I/O	TTL	
RB3/PGM	36	39	11	I/O	TTL	RB3 can also be the low voltage programming input
RB4	37	41	14	I/O	TTL	Interrupt on change pin.
RB5	38	42	15	I/O	TTL	Interrupt on change pin.
RB6/PGC	39	43	16	I/O	TTL/ST[(2)]	Interrupt on change pin or In-Circuit Debugger pin. Serial programming clock.
RB7/PGD	40	44	17	I/O	TTL/ST[(2)]	Interrupt on change pin or In-Circuit Debugger pin. Serial programming data.
						PORTC is a bi-directional I/O port.
RC0/T1OSO/T1CKI	15	16	32	I/O	ST	RC0 can also be the Timer1 oscillator output or a Timer1 clock input.
RC1/T1OSI/CCP2	16	18	35	I/O	ST	RC1 can also be the Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output.
RC2/CCP1	17	19	36	I/O	ST	RC2 can also be the Capture1 input/Compare1 output/PWM1 output.
RC3/SCK/SCL	18	20	37	I/O	ST	RC3 can also be the synchronous serial clock input/output for both SPI and I^2C modes.
RC4/SDI/SDA	23	25	42	I/O	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^2C mode).
RC5/SDO	24	26	43	I/O	ST	RC5 can also be the SPI Data Out (SPI mode).
RC6/TX/CK	25	27	44	I/O	ST	RC6 can also be the USART Asynchronous Transmit or Synchronous Clock.
RC7/RX/DT	26	29	1	I/O	ST	RC7 can also be the USART Asynchronous Receive or Synchronous Data.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as an external interrupt.
2: This buffer is a Schmitt Trigger input when used in serial programming mode.
3: This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise



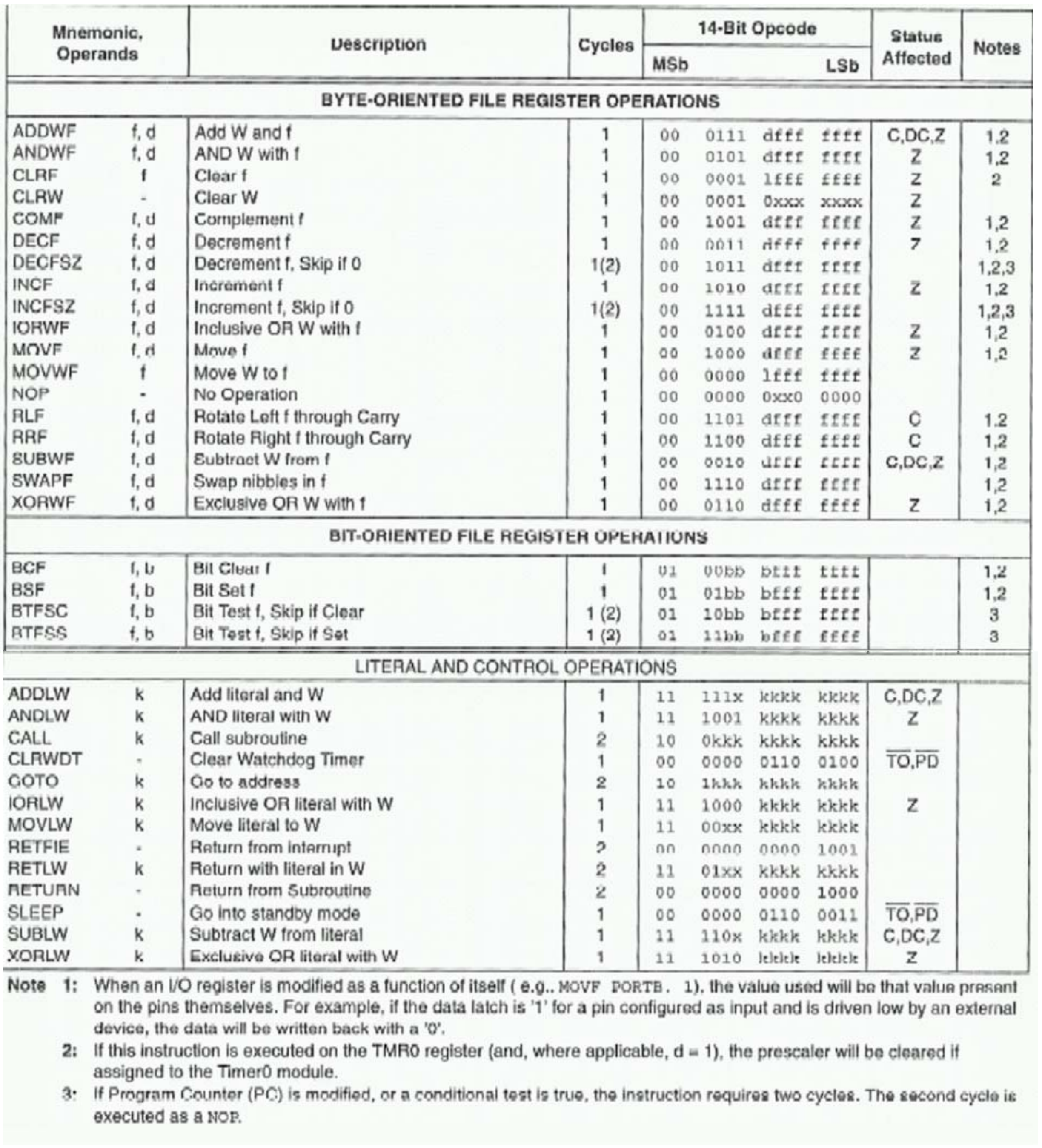
Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
						PORTD is a bi-directional I/O port or parallel slave port when interfacing to a microprocessor bus.
RD0/PSP0	19	21	38	I/O	ST/TTL(3)	
RD1/PSP1	20	22	39	I/O	ST/TTL(3)	
RD2/PSP2	21	23	40	I/O	ST/TTL(3)	
RD3/PSP3	22	24	41	I/O	ST/TTL(3)	
RD4/PSP4	27	30	2	I/O	ST/TTL(3)	
RD5/PSP5	28	31	3	I/O	ST/TTL(3)	
RD6/PSP6	29	32	4	I/O	ST/TTL(3)	
RD7/PSP7	30	33	5	I/O	ST/TTL(3)	
						PORTE is a bi-directional I/O port.
RE0/$\overline{\text{RD}}$/AN5	8	9	25	I/O	ST/TTL(3)	RE0 can also be read control for the parallel slave port, or analog input5.
RE1/$\overline{\text{WR}}$/AN6	9	10	26	I/O	ST/TTL(3)	RE1 can also be write control for the parallel slave port, or analog input6.
RE2/$\overline{\text{CS}}$/AN7	10	11	27	I/O	ST/TTL(3)	RE2 can also be select control for the parallel slave port, or analog input7.
Vss	12,31	13,34	6,29	P	—	Ground reference for logic and I/O pins.
VDD	11,32	12,35	7,28	P	—	Positive supply for logic and I/O pins.
NC	—	1,17,28, 40	12,13, 33,34		—	These pins are not internally connected. These pins should be left unconnected.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as an external interrupt.
2: This buffer is a Schmitt Trigger input when used in serial programming mode.
3: This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

Table 4. Mnemonics for programming the PIC16F877

.



Mnemonic, Operands		Description	Cycles	14-Bit Opcode				Status Affected	Notes
				MSb			LSb		
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z	1,2
CLRF	f	Clear f	1	00	0001	lfff	ffff	Z	2
CLRW	-	Clear W	1	00	0001	0xxx	xxxx	Z	
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z	1,2
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff		1,2,3
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00	1111	dfff	ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	1,2
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z	1,2
MOVWF	f	Move W to f	1	00	0000	lfff	ffff		
NOP	-	No Operation	1	00	0000	0xx0	0000		
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff		1,2
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		3
LITERAL AND CONTROL OPERATIONS									
ADDLW	k	Add literal and W	1	11	111x	kkkk	kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11	1001	kkkk	kkkk	Z	
CALL	k	Call subroutine	2	10	0kkk	kkkk	kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00	0000	0110	0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10	1kkk	kkkk	kkkk		
IORLW	k	Inclusive OR literal with W	1	11	1000	kkkk	kkkk	Z	
MOVLW	k	Move literal to W	1	11	00xx	kkkk	kkkk		
RETFIE	-	Return from interrupt	2	00	0000	0000	1001		
RETLW	k	Return with literal in W	2	11	01xx	kkkk	kkkk		
RETURN	-	Return from Subroutine	2	00	0000	0000	1000		
SLEEP	-	Go into standby mode	1	00	0000	0110	0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11	110x	kkkk	kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11	1010	kkkk	kkkk	Z	

Note 1: When an I/O register is modified as a function of itself (e.g., `MOVF PORTB, 1`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned to the Timer0 module.

3: If Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

6.4 Appendix B – Sample Calculations

Calculation for servo motor torque required

$$Total\ mass = 580g = 0.58kg$$
$$Safetyfactor = 1.3$$
$$Mass, M = 0.58x1.3 = 0.754kg$$
$$Length\ of\ servo\ arm, L = 3.8cm$$
$$Torque\ acting\ on\ servo = ML = 2.87kg.cm$$

To operate 20 Tennis balls, $\tau_{reqd} = 20\,x\,0.06kg\,x\,3.8cm = 4.6kg.cm$

Calculation for servo motor pulse width

Centre position requires a pulse width of 1.5ms (t_{high}) and 50Hz refresh rate. This works out to a period of 20ms or less. Then, t_{low}<18.5ms.

According to formulas 1 and 2 given in the introduction:

$t_{high} = 0.693CR_1 = 1.5 \times 10^{-3}$s ➔ $R_1 = 2.2k\Omega$
$t_{low} = 0.693CR_2 < 18.5 \times 10^{-3}$s ➔ $R_2 < 26.7k\Omega$

R_1 is connected to +5V and R_2 is in parallel with the diode.

6.5 Appendix C – Circuit Schematics

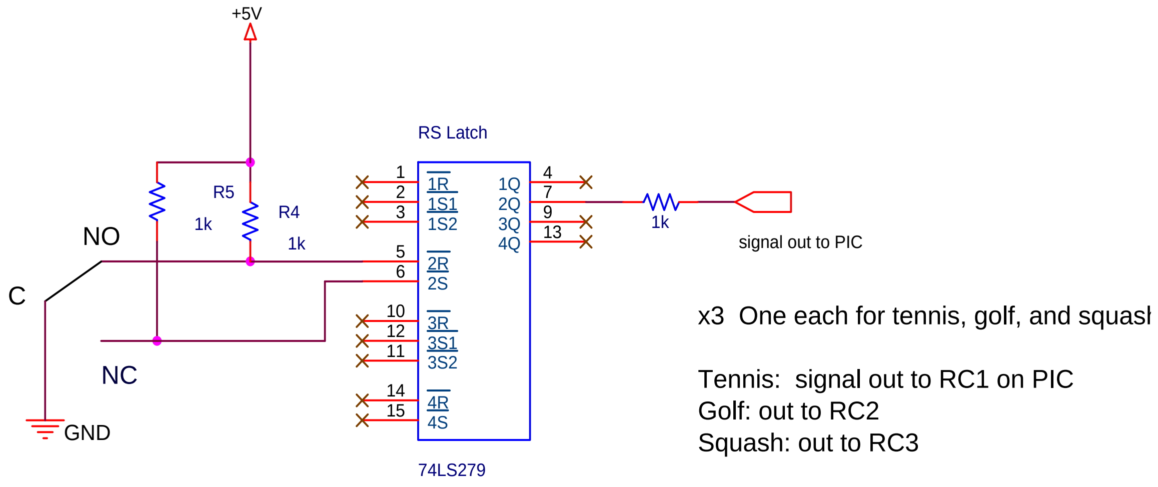
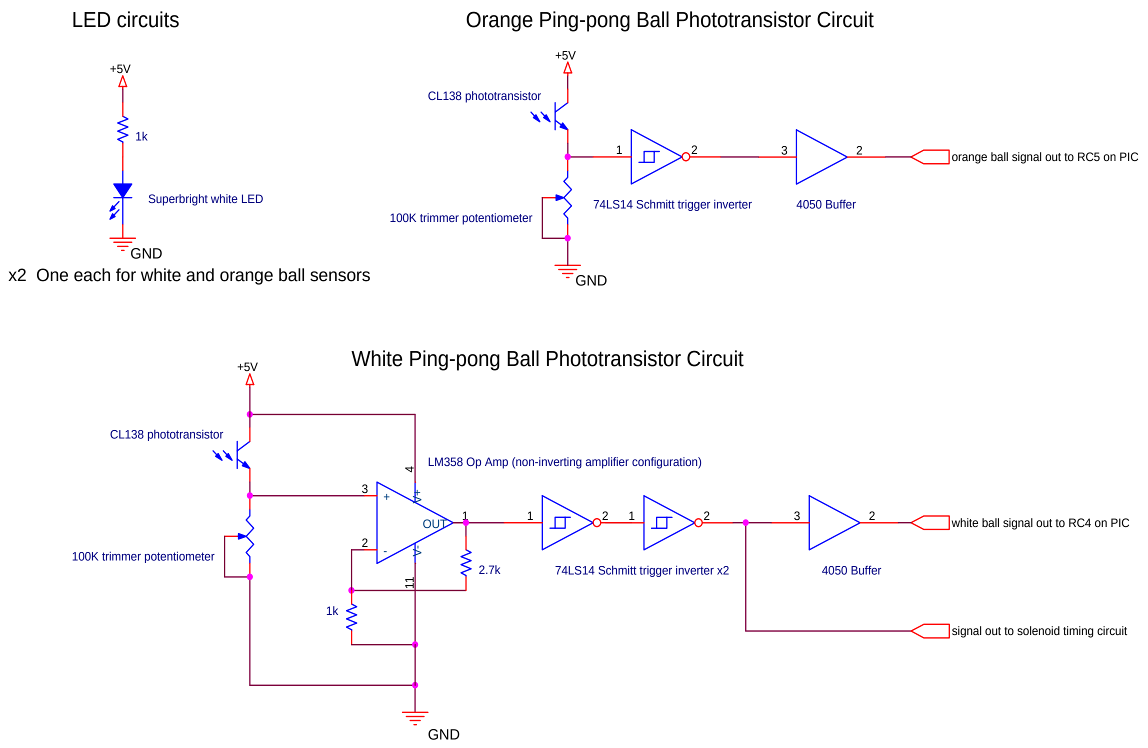


Fig. 1. Switch debouncer circuit, adapted from page 5-49 in the course notes[4].



LED circuits
+5V
1k
Superbright white LED
GND
x2 One each for white and orange ball sensors
Orange Ping-pong Ball Phototransistor Circuit
+5V
CL138 phototransistor
100K trimmer potentiometer
74LS14 Schmitt trigger inverter
4050 Buffer
orange ball signal out to RC5 on PIC
GND
White Ping-pong Ball Phototransistor Circuit
+5V
CL138 phototransistor
100K trimmer potentiometer
LM358 Op Amp (non-inverting amplifier configuration)
OUT
2.7k
1k
74LS14 Schmitt trigger inverter x2
4050 Buffer
white ball signal out to RC4 on PIC
signal out to solenoid timing circuit
GND

Fig. 2. Colour sensing circuit. Op amp portion taken from page 5-60 in the course notes[5].

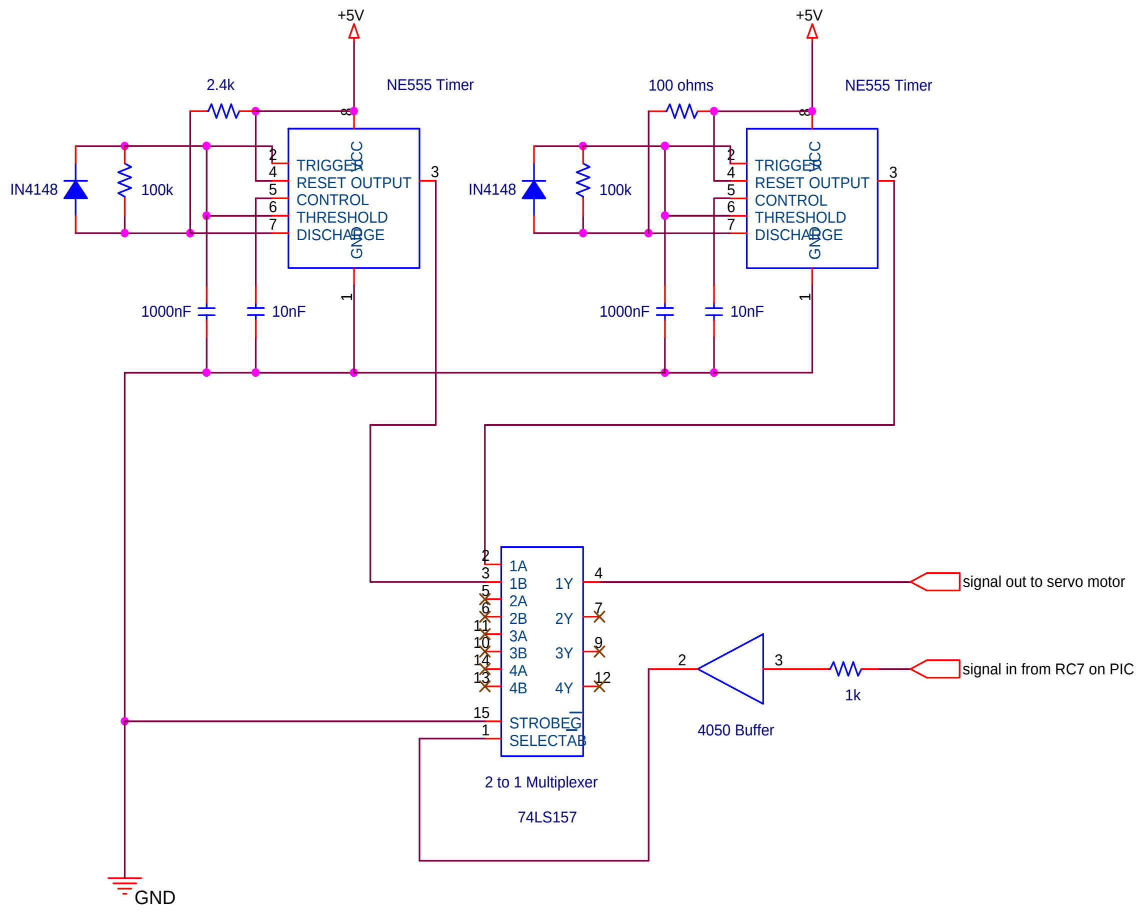


Fig. 3. Servo motor driver circuit. NE555 timer circuit taken from page 5-51 in the course notes[6].

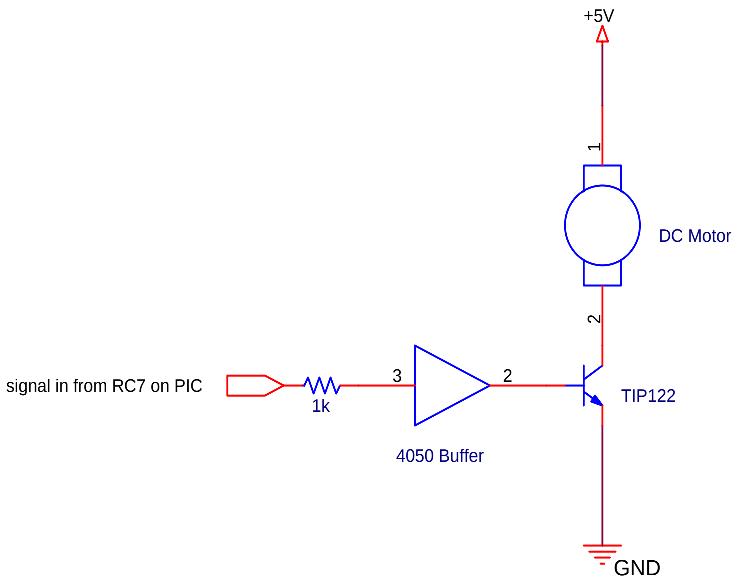


Fig. 4. DC motor driver circuit

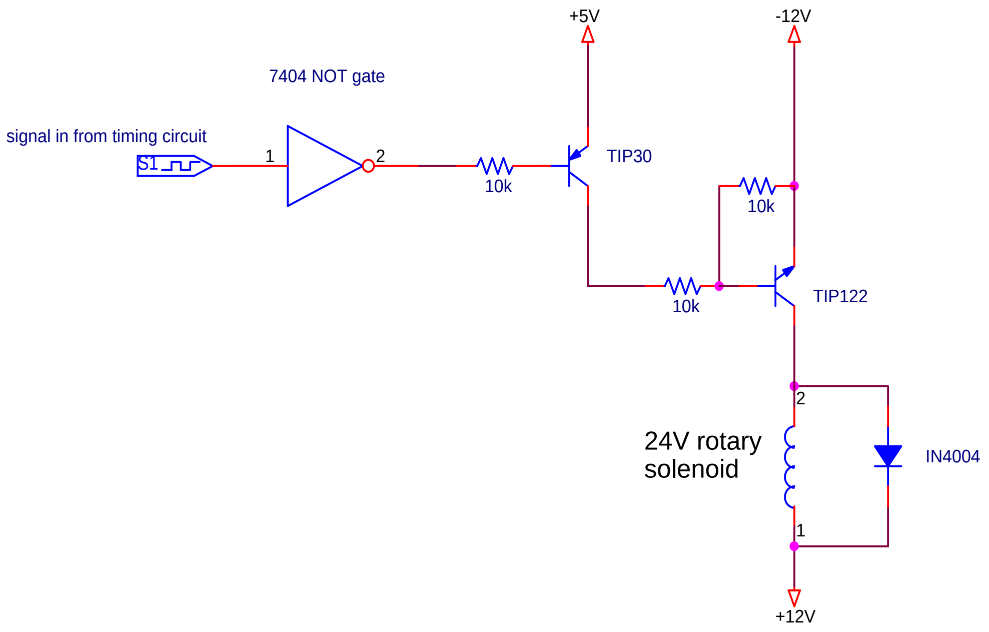


Fig. 5. Rotary solenoid driver circuit. Adapted from Seattle Robotics message post[7].

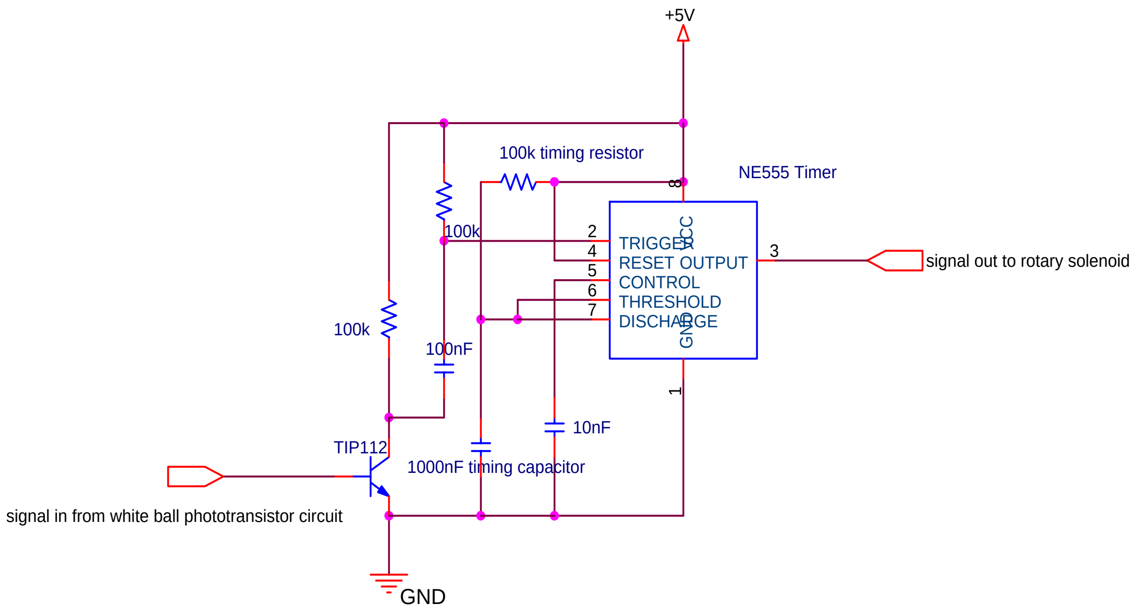


Fig. 6. Rotary solenoid timing circuit. Adapted from "1Second Oneshot Monostable Oscillator"[8].

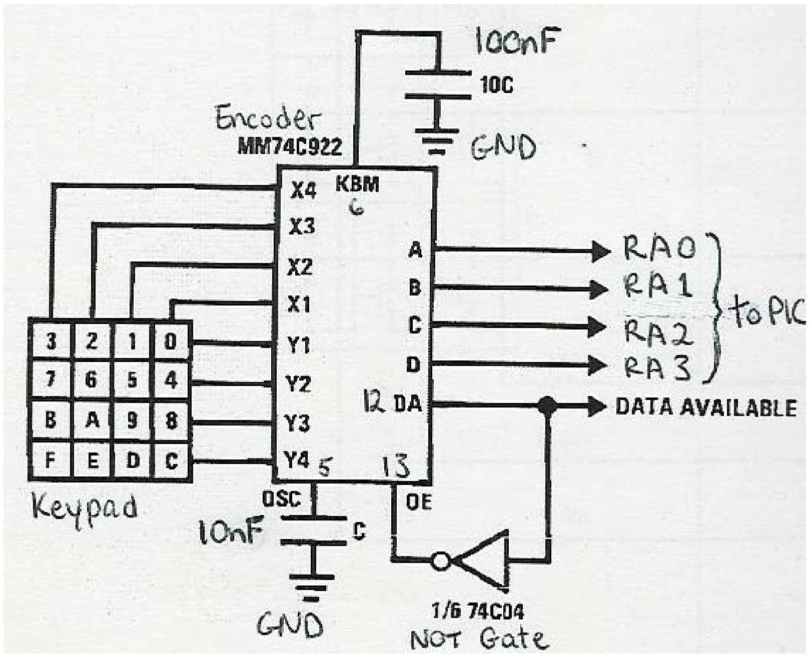


Fig. 7. Keypad encoder circuit. Taken from page 7-38 in the course notes[9].

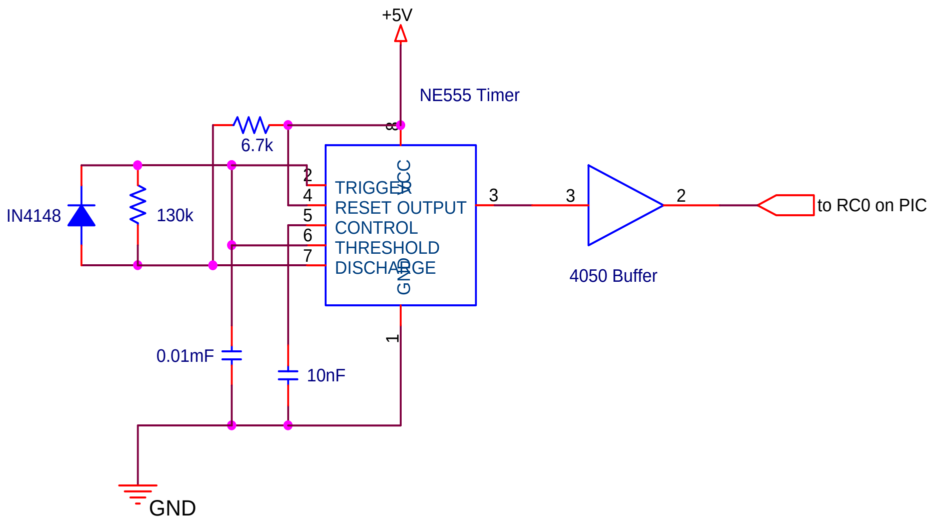
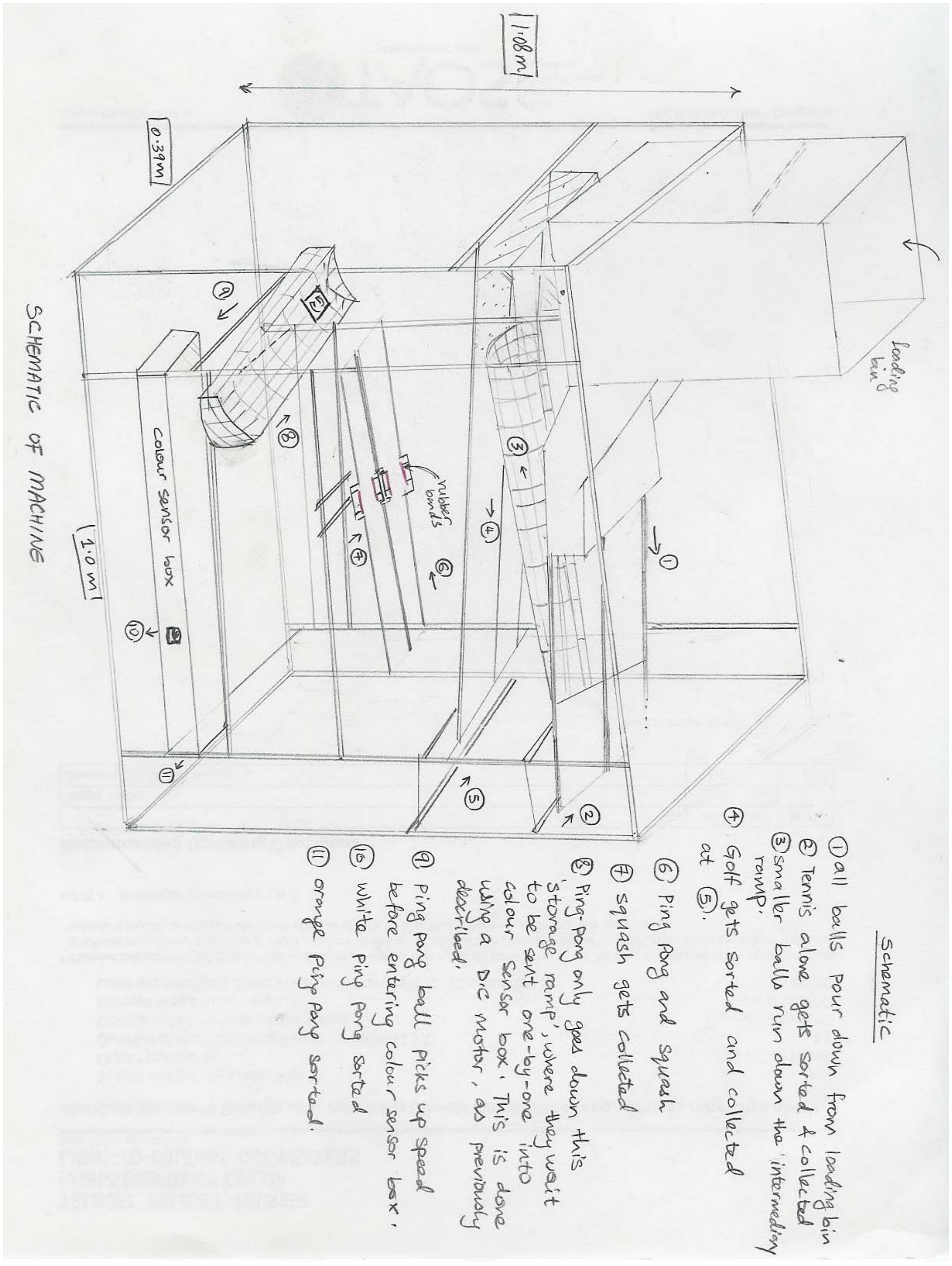


Fig. 8. PIC timing circuit. Similar to servo motor circuit.

6.6 Appendix D – Machine Schematics



Schematic

① all balls pour down from loading bin
② Tennis alone gets sorted & collected
③ smaller balls run down the 'intermediary ramp'.
④ Golf gets sorted and collected at ⑤.
⑥ Ping Pong and squash
⑦ squash gets collected
⑧ Ping-Pong only goes down this 'storage ramp', where they wait to be sent one-by-one into colour sensor box. This is done using a DC motor, as previously described.
⑨ Ping Pong ball picks up speed before entering colour sensor box.
⑩ white Ping Pong sorted
⑪ orange Ping Pong sorted.

Fig. 1. Sketch of the entire machine.

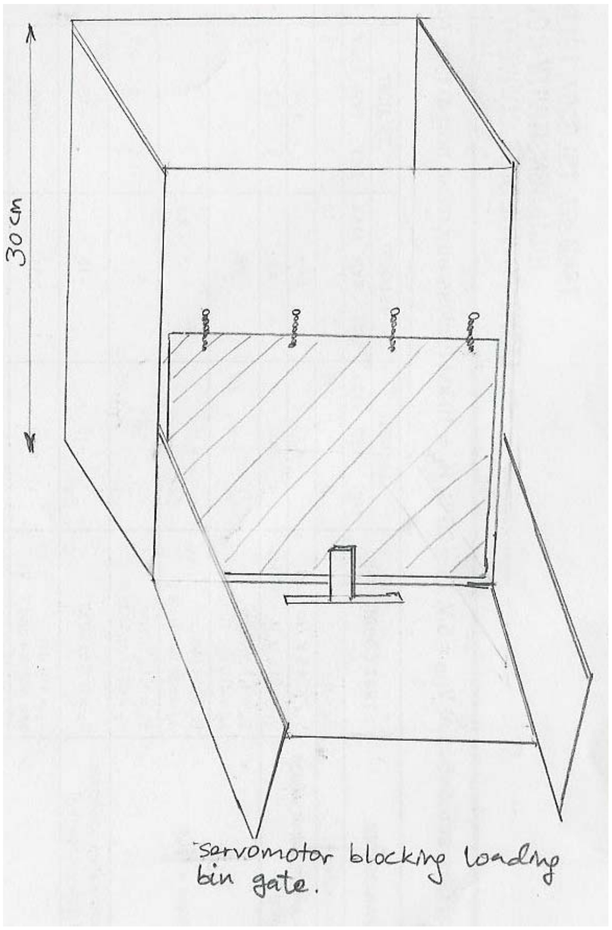


Fig. 2. The loading bin.

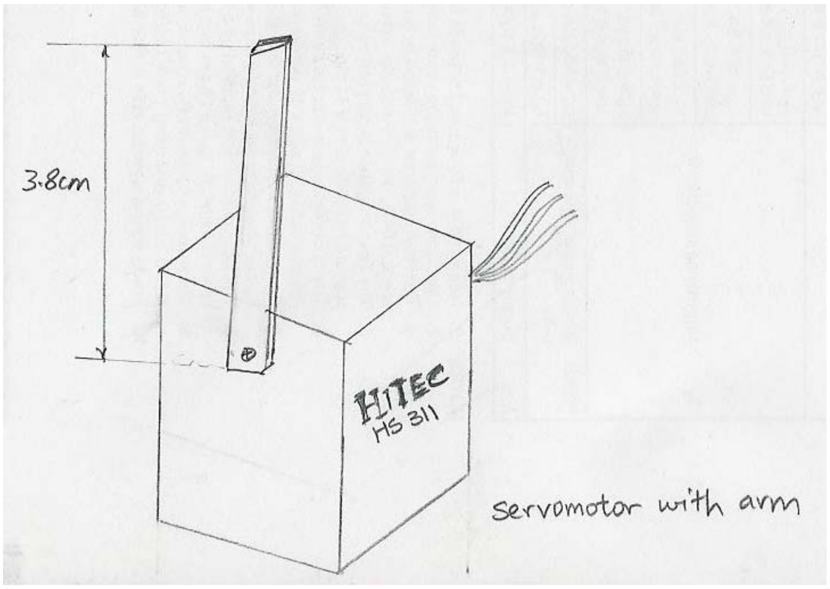


Fig. 3. The servo motor

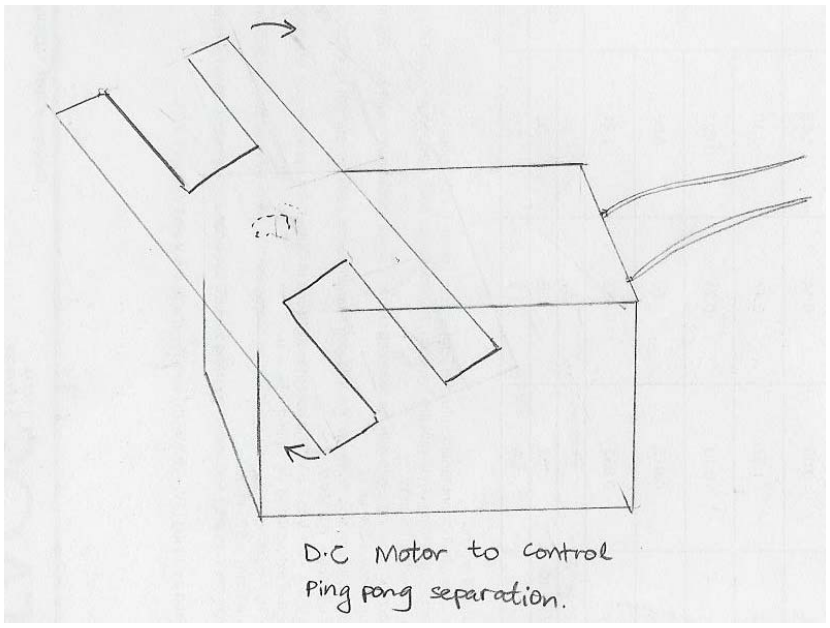
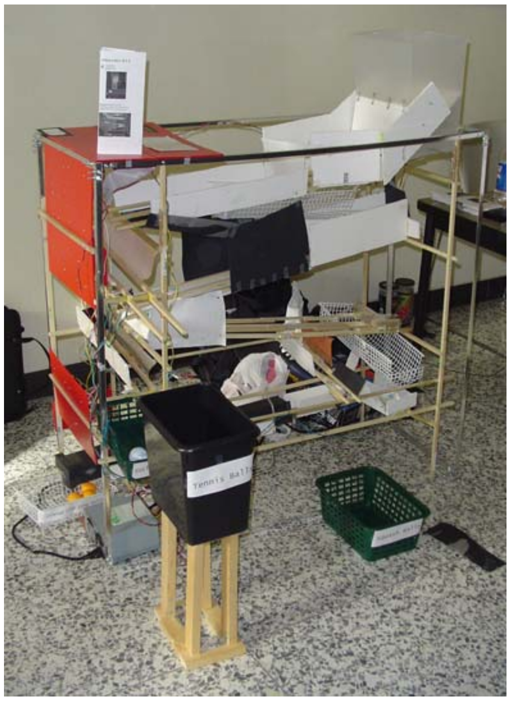


Fig. 4. The DC gearhead motor.

6.7 Appendix E -- Photographs



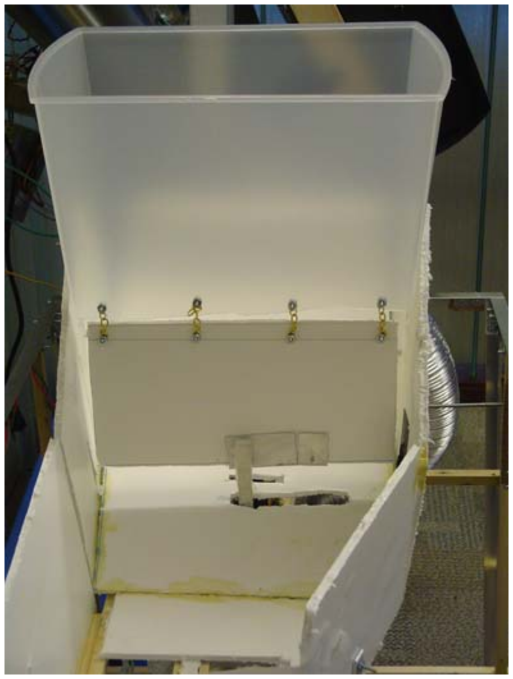


Fig. 1. (left) The machine in its entirety.

Fig. 2. (right) The loading bin with servo motor-controlled gate.

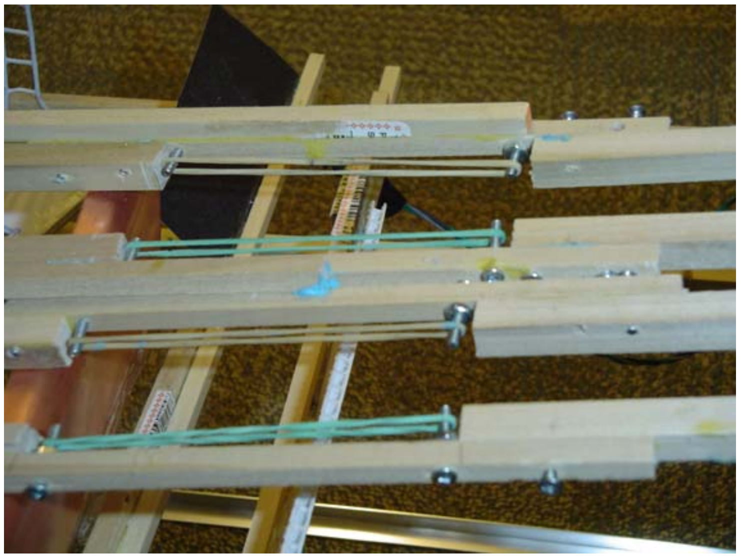


Fig. 3. Elastic bands allow squash balls to drop while ping-pong balls pass over.

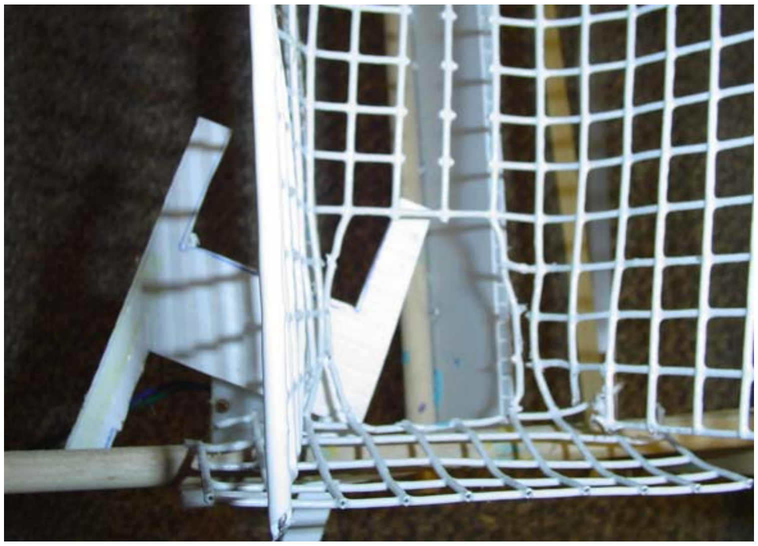


Fig. 4. DC motor controls spacing between ping-pong balls.

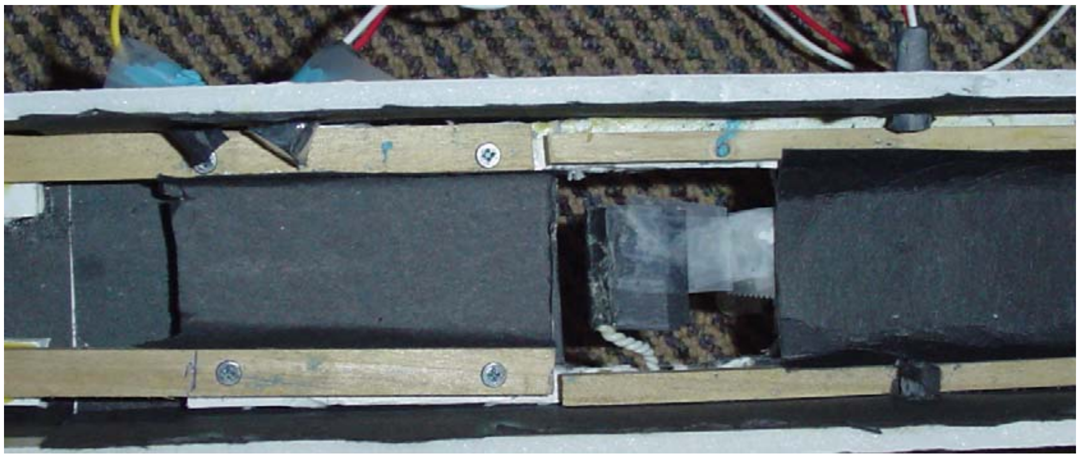


Fig. 5. Colour sensing tunnel. White ball LED and phototransistor on the left, orange ball sensor on the right, rotary solenoid flap in the middlle.

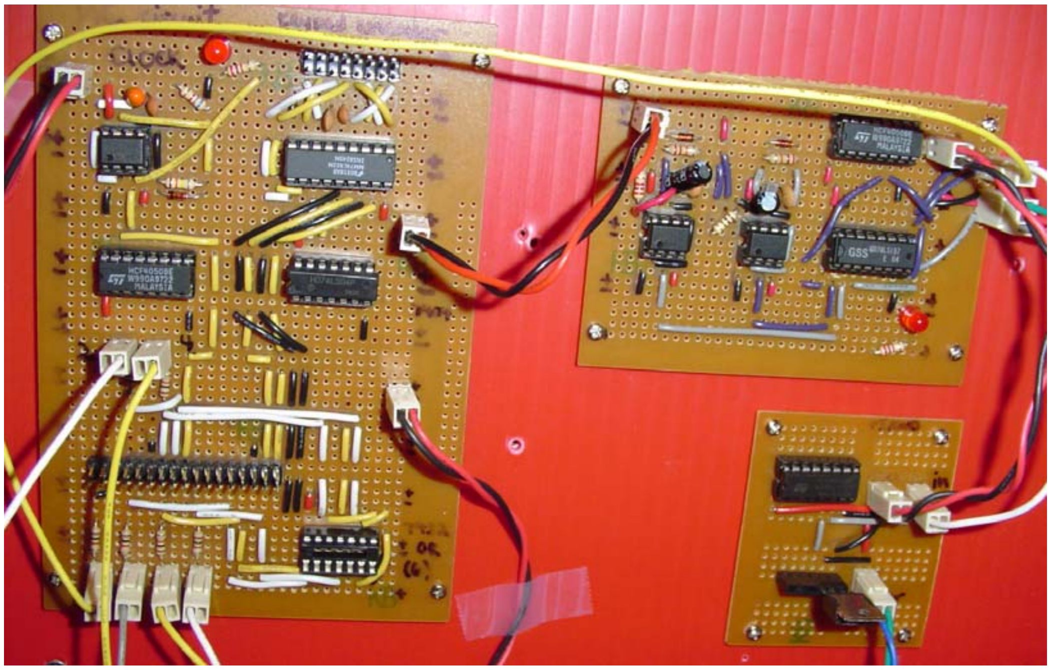


Fig. 6. Upper cluster of circuits. Clockwise from left: PIC circuit, servo motor circuit, DC motor circuit.

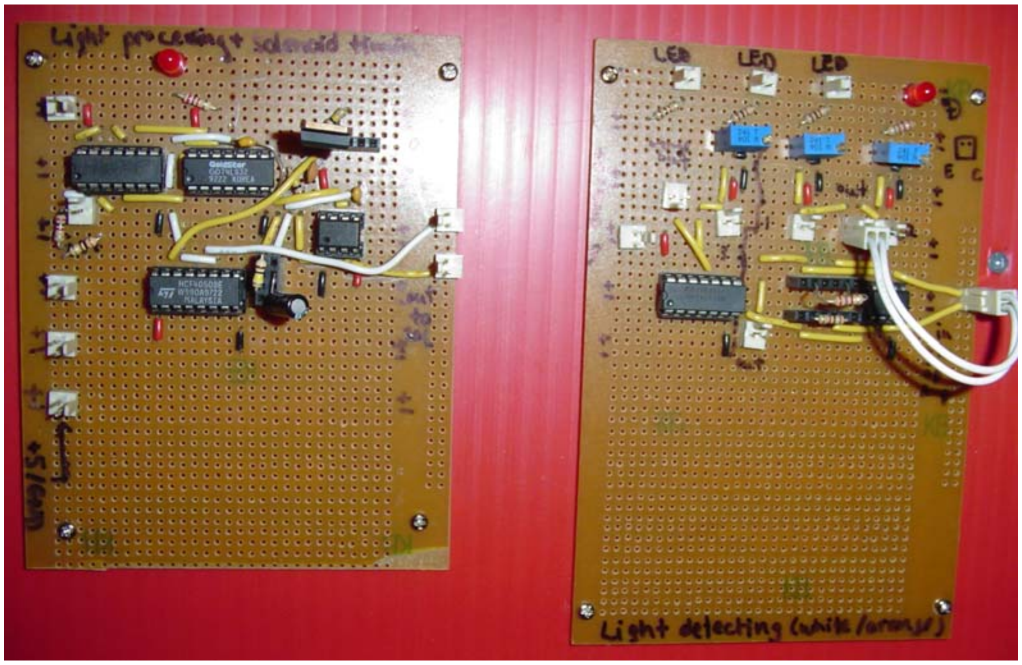

Light processing + solenoid timer
LED
LED
LED
+5/Gnd
Light detecting (white/orange)

Fig. 7. Lower cluster of circuits. Left: white ping-pong ball processing circuit. Right: Light detecting circuit.

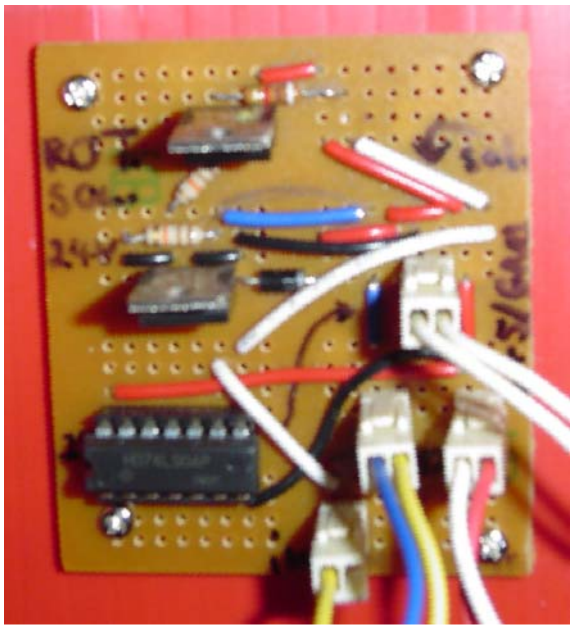


Fig. 8. Rotary solenoid circuit.

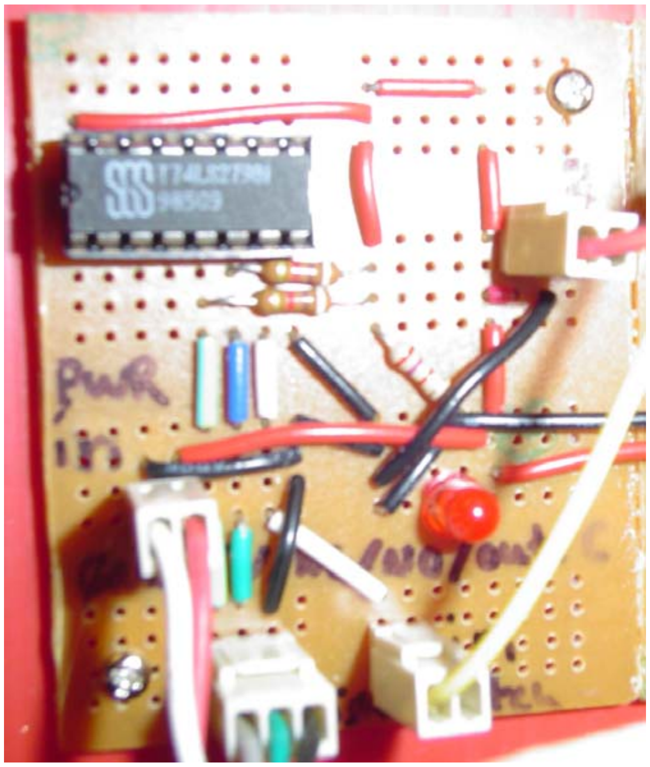


Fig. 9. One third of the switch debouncer circuit.

6.8 Appendix F – Standard Operating Procedures

Tools and supplementary materials required: chip puller and replacement PIC16F877A

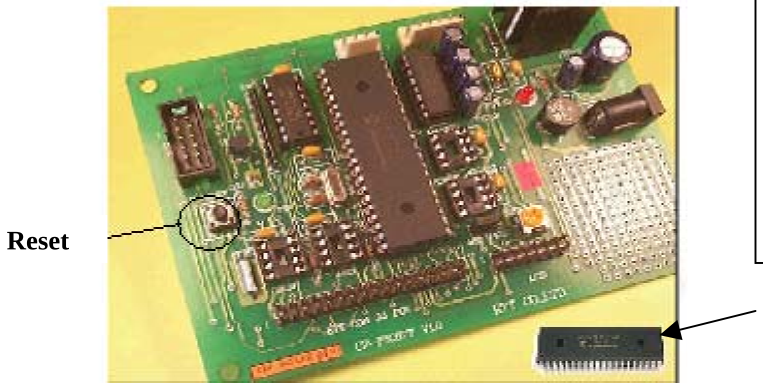
Warning: Please follow the exact operation procedure to prevent damage to machine

1. Plug in the power supply cord into a regular 110VAC 60Hz wall socket. **Make sure your hand is dry to prevent injury from electrical shock!**
2. Turn on the master power switch connected to the power supply. At this point, the servomotor, rotary solenoid and DC motor may move momentarily.
3. Before loading any balls:
    - Ensure that the servomotor is blocking the gate with the shaft attached to it in upright position.
    - Ensure that the display on LCD reads: "Press D". If instead, there appear to be square blocks on the screen, proceed to step **A** below.
    - Ensure that the collecting bins are placed in their correct positions, as marked on the machine.
4. Put up to a maximum of 20 assorted balls of the following types: Tennis, Golf, squash, white and orange ping-pong balls. Recommended squash ball type is the blue-dot model.
5. Press 'D' only when steps 1-4 have been strictly observed. Pressing D starts sorting.
6. Wait for the display to read 'Finished'. Typical operation takes less than 30 seconds.
7. To get the count of each ball type, follow the legend provided below.
8. To run a second time, toggle either the Reset button shown on the diagram below or the main power switch.
9. When not in use, store away from moisture.
10. If machine encounters other problems, call the toll-free number provided for phone support.

Troubleshooting Instructions

A. If LCD screen displays blocks or nothing, turn off the power switch and unplug it from the wall socket. Locate the PIC development board on the underside of the keypad console. The diagram on the right shows what a PIC development board looks like. Using the chip puller provided, gently pull the PIC free of its board. Insert a new PIC that shipped with the product. Ensure correct position of PIC. Pin #1 is marked.

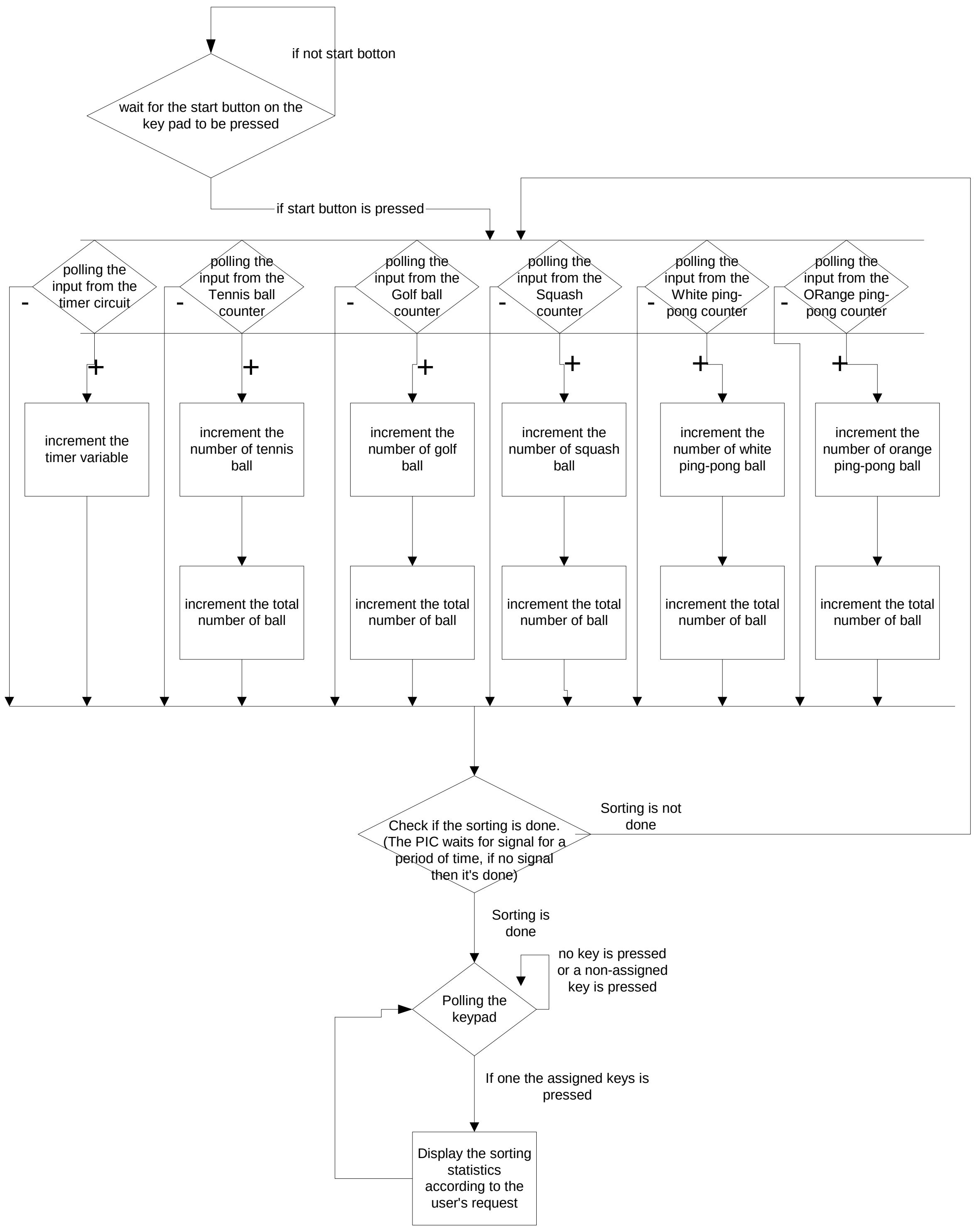
Legend:
D: Start Operation
1: Tennis count
2: Golf
3: Squash
4: White Ping-pong
5: Orange Ping-pong
6: Total ball count
8: Total running time



PIC development board, with PIC on bottom right

6.9 Appendix G – Microcontroller Algorithm and Code

Fig. 1. Flowchart illustrating the microcontroller algorithm.



if not start botton
wait for the start button on the key pad to be pressed
if start button is pressed
polling the input from the timer circuit
polling the input from the Tennis ball counter
polling the input from the Golf ball counter
polling the input from the Squash counter
polling the input from the White ping-pong counter
polling the input from the ORange ping-pong counter
-
+
increment the timer variable
increment the number of tennis ball
increment the number of golf ball
increment the number of squash ball
increment the number of white ping-pong ball
increment the number of orange ping-pong ball
increment the total number of ball
Check if the sorting is done. (The PIC waits for signal for a period of time, if no signal then it's done)
Sorting is not done
Sorting is done
Polling the keypad
no key is pressed or a non-assigned key is pressed
If one the assigned keys is pressed
Display the sorting statistics according to the user's request

Fig. 2. The microcontroller assembly language code. Text in italics indicated programmer comments.

```
list P=PIC16F877, F=INHX8M, C=160, N=80, ST=OFF, MM=OFF, R=DEC
include "P16F877.INC"
__config (_CP_OFF & _PWRTE_ON & _XT_OSC & _WDT_OFF & _BODEN_OFF)
errorlevel -302


;;;;;;;;;;;;;;;;;;;;;;;;;;;equates;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
#define     timer  PORTC,0
#define     SW1    PORTC,1
#define     SW2    PORTC,2
#define     SW3    PORTC,3
#define     SW4    PORTC,4
#define     SW5    PORTC,5
#define     SW6    PORTC,6

Bank0RAM          equ H'20'
MaxCount          equ 100
TenMsH            equ 13
TenMsL            equ 250

#define   RS        PORTD,2
#define   E         PORTD,3

com       EQU       0x20           ; buffer for Instruction
dat         EQU       0x21           ; buffer for data

;;;;;;;;;;;;;;;;;;;;;;;;;Variables;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


    cblock                Bank0RAM
    check_timer
    check_check_timer
    check1
    check_check1
    check2
    check_check2
    check3
    check_check3
    check4
    check_check4
    check5
    check_check5
    check6
    check_check6
    check_key1
    check_key2
    check_key3
    check_key4
    check_key5
    check_key6
    check_key7
    check_key8
    temp
    temp1
    tempkey
    tempkey1
    tempSW


   counter_timer
   counter_timer2
   counter_timer3
   counter_timer4
   counter_timer5
   counter1
   counter12
   counter13
   counter2
   counter22
   counter23
   counter3
   counter32
   counter33
   counter4
   counter42
   counter43
   counter5
   counter52
   counter53
   counter6
   counter62
   counter63
   COUNTH
   COUNTL
   finish?
        endc

;;;;;;;;;;;;;;;;;;Vectors;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        org H'000'
        goto Mainline
;        org H'004'
;        goto Stop

;;;;;;;;;;;;;;Mainline program;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;**********************************************************************

;;;;;;;;;;;;;;First Section;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

;This is the first loop that waits for the user to start the machine. ;Once the user presses "D" on the keypad, the program will go to the ;next loop to start the counting and timing.

```
Mainline
    call Initial
     movlw     "P"
     call      WR_DATA
     movlw     "r"
     call      WR_DATA
     movlw     "e"
     call      WR_DATA
     movlw     "s"
     call      WR_DATA
     movlw     "s"
     call      WR_DATA
     movlw     " "
     call      WR_DATA
     movlw     "D"
     call      WR_DATA

wait

;00001100 is the binary number the keypad send to the PIC through the
;encoder


     movf    PORTA, W
     movwf   tempkey
     andlw   B'00001111'
     sublw   B'00001100'
     btfsc   STATUS, Z
     goto    Start
     goto    wait


;*******************************************************************

;;;;;;;;;;;;;;;Second Section;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;Display the initial setting of the timer Start
     call      Initial
     bsf       PORTC, 7
     movlw     B'10000010'
     call     WR_INS
     movlw     "T"
     call      WR_DATA
     movlw     "i"
     call      WR_DATA
     movlw     "m"
     call      WR_DATA
     movlw     "e"
     call      WR_DATA
     movlw     ":"
     call      WR_DATA

     movlw     B'11000000'
     call     WR_INS
     movlw     "0"
     call      WR_DATA
     movlw     "0"
     call      WR_DATA
     movlw     "0"
     call      WR_DATA

     movlw     B'11000100'
     call      WR_INS
     movlw     "s"
     call      WR_DATA
```

; This is the main function of the program. It times the operation time ;and counts the number of the balls in each category by counting the ;pins.

```
MainLoop
```

; We did the timer externally. We build a timer circuit with a 555 ;timer circuit and it sends a signal to the pin every second.

```
    btfsc  timer
    goto   timer_on
```

; Since the signal from the pin stays high for many cycles, we need the ;“check” variable to avoid the program gets into the loop more than ;once and thus counts more than once ;for one signal.

```
    bcf       check_timer,0
```

; SW’s are the switches on the machine that send signal to the PIC ;whenever they are ;pressed. The following SWon subroutines will then ;count the number of signals sent. They also have the “check” variables ;to avoid counting more than once for a single signal.

;All the switches and the timer circuit are connected to the pins in ;the PORT C.

```
    clrw
    bcf     STATUS, Z
    clrf    tempSW
    movf    PORTC, W
    movwf   tempSW
    andlw   B'00111110'
    sublw   B'00000010'
     btfsc   STATUS, Z
     goto    SW1on
     bcf     check1,0

     clrw
     bcf     STATUS, Z
     movf    tempSW, W
     andlw   B'00111110'
     sublw   B'00000100'
     btfsc   STATUS, Z
     goto    SW2on
     bcf     check2,0

    clrw
     bcf     STATUS, Z
     movf    tempSW, W
     andlw   B'00111110'
     sublw   B'00001000'
     btfsc   STATUS, Z
     goto    SW3on
     bcf     check3,0

     clrw
     bcf     STATUS, Z
     movf    tempSW, W
     andlw   B'00111110'
     sublw   B'00010000'
     btfsc   STATUS, Z
     goto    SW4on
     bcf     check4,0

     clrw
     bcf     STATUS, Z
     movf    tempSW, W
     andlw   B'00111110'
     sublw   B'00100000'
     btfsc   STATUS, Z
     goto    SW5on
     bcf     check5,0



      goto    MainLoop
```

;The following “on” subroutines are the actual functions that increment ;the counter ;variables.

```
timer_on

         movf      check_timer, W
         movwf     temp1
         movlw     B'00000010'    ; Clear ram
         call      WR_INS
         clrw
         movf      temp1, W
         movwf     check_timer
         btfsc     check_timer,0
         goto      returntimer
         bsf       check_timer,0
         bsf       check_check_timer,0
```
;Increment the timer variable. Also decrement of “finish?” variable. ;The machine will operate for at least 15 seconds to wait for any ;switch to be pressed. If nothing happens for 15 seconds, the program ;will finish counting the timing and move to the next section.

```
         movf      check_timer, W
         movwf     temp
         decfsz    finish?, F
         goto      move_on
         goto      finish_sorting
move_on
         incfsz    counter_timer, F
         decfsz    counter_timer2, F    ; check if move to the second digit
         goto      one_digit_timer
         goto      two_digit_timer

two_digit_timer
         movlw     B'11000001'
         call      WR_INS

         movlw     0
         movwf     counter_timer
         movlw     10
         movwf     counter_timer2
         incfsz    counter_timer3

         decfsz    counter_timer4, F
         goto      return_from_3digit
         goto      three_digit_timer

return_from_3digit


         movf      counter_timer3, W
         addlw     B'00110000'
         call      WR_DATA

         goto      one_digit_timer

three_digit_timer
         movlw     B'11000000'
         call      WR_INS

         movlw     0
         movwf     counter_timer3
         movlw     10
         movwf     counter_timer4

         incfsz    counter_timer5
         movf      counter_timer5, W
         addlw     B'00110000'
         call      WR_DATA

         goto      return_from_3digit

one_digit_timer

         movlw     B'11000010'
         call      WR_INS
         movf      counter_timer, W
         movwf     temp1
         movf      temp, W
         movwf     check_timer
         clrw
         movf      counter_timer, W
         addlw     B'00110000'
         call      WR_DATA
         clrw
         movf      temp1, W
         movwf     counter_timer
         goto      returntimer
SW1on


         movf      check1, W
         movwf     temp1
         clrw
         movf      temp1, W
         movwf     check1
         btfsc     check1,0
         goto      MainLoop

         bsf       check1,0
         bsf       check_check1, 0
         movf      check1, W
         movwf     temp



;As mentioned before, the program will wait for 15 seconds and then ;finish
the counting ;and timing. However, every time a switch is ;pressed, it will
reset the “finish?” variable and the machine will ;wait for 5 seconds from
the point the switch is pressed.
          movlw    5
          movwf    finish?

         incfsz    counter1, F
         decfsz    counter12, F
         goto      one_digit_1
         goto      two_digit_1

two_digit_1

         movlw     0
         movwf     counter1
         movlw     10
         movwf     counter12
         incfsz    counter13

one_digit_1
         movf      counter1, W
         movwf     temp1
         movf      temp, W
         movwf     check1
         clrw

         clrw
         movf      temp1, W
         movwf     counter1
; We not only increment the counter for this switch, but also increment ;the
counter for the ;total number of balls.
         goto      TotalBallCount


SW2on

         movf      check2, W
         movwf     temp1
         clrw
         movf      temp1, W
         movwf     check2
         btfsc     check2,0
         goto      MainLoop

         bsf       check2,0
         bsf       check_check2, 0
         movf      check2, W
         movwf     temp

          movlw    5
          movwf    finish?

         incfsz    counter2, F
         decfsz    counter22, F
         goto      one_digit_2
         goto      two_digit_2

two_digit_2

         movlw     0
         movwf     counter2
         movlw     10
         movwf     counter22
         incfsz    counter23

one_digit_2
         movf      counter2, W
         movwf     temp1
         movf      temp, W
         movwf     check2
         clrw
         clrw
         movf      temp1, W
         movwf     counter2

         goto      TotalBallCount

SW3on

         movf      check3, W
         movwf     temp1
         clrw
         movf      temp1, W
         movwf     check3
         btfsc     check3,0
         goto      MainLoop

         bsf       check3,0
         bsf       check_check3, 0
         movf      check3, W
         movwf     temp

          movlw    5
          movwf    finish?

         incfsz    counter3, F
         decfsz    counter32, F
         goto      one_digit_3
         goto      two_digit_3

two_digit_3

         movlw     0
         movwf     counter3
         movlw     10
         movwf     counter32
         incfsz    counter33
         bsf       index3,0

one_digit_3
         movf      index3, W
         addlw     B'10000000
         call      WR_INS
         movf      counter3, W
         movwf     temp1
         movf      temp, W
         movwf     check3
         clrw
         clrw
         movf      temp1, W
         movwf     counter3

          goto      TotalBallCount

SW4on


         movf      check4, W
         movwf     temp1
         clrw
         movf      temp1, W
         movwf     check4
         btfsc     check4,0
         goto      MainLoop

         bsf       check4,0
         bsf       check_check4, 0
         movf      check4, W
         movwf     temp

          movlw    5
          movwf    finish?
         incfsz    counter4, F
         decfsz    counter42, F
         goto      one_digit_4
         goto      two_digit_4

two_digit_4

         movlw     0
         movwf     counter4
         movlw     10
         movwf     counter42
         incfsz    counter43
         bsf       index4,0

one_digit_4
         movf      counter4, W
         movwf     temp1
         movf      temp, W
         movwf     check4
         clrw
         clrw
         movf      temp1, W
         movwf     counter4

         goto      TotalBallCount

SW5on

         movf      check5, W
         movwf     temp1
         clrw
         movf      temp1, W
         movwf     check5
         btfsc     check5,0
         goto      MainLoop

         bsf       check5,0
         bsf       check_check5, 0
         movf      check5, W
         movwf     temp

          movlw    5
          movwf    finish?

         incfsz    counter5, F
         decfsz    counter52, F
         goto      one_digit_5
         goto      two_digit_5

two_digit_5

         movlw     0
         movwf     counter5
         movlw     10
         movwf     counter52
         incfsz    counter53
         bsf       index5,0

one_digit_5
         movf      counter5, W
         movwf     temp1
         movf      temp, W
         movwf     check5
         clrw
         clrw
         movf      temp1, W
         movwf     counter5

         goto      TotalBallCount

TotalBallCount

         incfsz    counter6, F
         decfsz    counter62, F
         goto      one_digit_6
         goto      two_digit_6

two_digit_6

         movlw     0
         movwf     counter6
         movlw     10
         movwf     counter62
         incfsz    counter63

one_digit_6
         movf      counter6, W
         movwf     temp1
         movf      temp, W
         movwf     check6
         clrw
         movf      temp1, W
         movwf     counter6
         goto      MainLoop

finish_sorting
     bcf       PORTC, 7
     movlw     "F"
     call      WR_DATA
     movlw     "i"
     call      WR_DATA
     movlw     "n"
     call      WR_DATA
     movlw     "i"
     call      WR_DATA
     movlw     "s"
     call      WR_DATA
     movlw     "h"
     call      WR_DATA
     movlw     ":"
     call      WR_DATA

     goto     display
```

```
;;*******************************************************************

;;*******************************************************************

;;;;;;;;;;;;;;;;;;;;;Third Section;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

;This section is just a loop that waits for the 4-bit binary numbers ;from the keypad ;encoder. According the request from the user, the PIC ;sends the data to the LCD to display to sorting statistics.

```
display
     clrw
     bcf     STATUS, Z
     clrf    tempkey
     movf    PORTA, W

     movwf   tempkey
     andlw   B'00001111'
     sublw   B'00000011'
     btfsc   STATUS, Z
     goto    ONE

         clrw
     bcf     STATUS, Z
     movf    tempkey, W
     andlw   B'00001111'
     sublw   B'00000010'
     btfsc   STATUS, Z
     goto    TWO

         clrw
     bcf     STATUS, Z
     movf    tempkey, W
     andlw   B'00001111'
     sublw   B'00000001'
     btfsc   STATUS, Z
     goto    THREE

         clrw
     bcf     STATUS, Z
     movf    tempkey, W
     andlw   B'00001111'
     sublw   B'00000111'
     btfsc   STATUS, Z
     goto    FOUR

         clrw
     bcf     STATUS, Z
     movf    tempkey, W
     andlw   B'00001111'
     sublw   B'00000110'
     btfsc   STATUS, Z
     goto    FIVE
```

```
        clrw
    bcf     STATUS, Z
    movf    tempkey, W
    andlw   B'00001111'
    sublw   B'00001011'
    btfsc   STATUS, Z
    goto    SEVEN

        clrw
    bcf     STATUS, Z
    movf    tempkey, W
    andlw   B'00001111'
    sublw   B'00001010'
    btfsc   STATUS, Z
    goto    EIGHT

    goto    display

ONE
          btfss     check_key1, 0
          goto      display

          bcf       check_key1,0
          bsf       check_key2,0
          bsf       check_key3,0
          bsf       check_key4,0
          bsf       check_key5,0
          bsf       check_key6,0
          bsf       check_key7,0
          bsf       check_key8,0

          movlw     B'00000001'    ; Clear ram
          call      WR_INS

         movf      counter13, W
         addlw     B'00110000'
         call      WR_DATA
         movf      counter1, W
         addlw     B'00110000'
         call      WR_DATA

         movlw     B'11000000'
         call      WR_INS

         movlw     "T"
         call      WR_DATA
         movlw     "e"
         call      WR_DATA
         movlw     "n"
         call      WR_DATA
         movlw     "n"
         call      WR_DATA
         movlw     "i"
         call      WR_DATA
         movlw     "s"
         call      WR_DATA
         goto      display
TWO
          btfss     check_key2, 0
          goto      display

          bsf       check_key1,0
          bcf       check_key2,0
          bsf       check_key3,0
          bsf       check_key4,0
          bsf       check_key5,0
          bsf       check_key6,0
          bsf       check_key7,0
          bsf       check_key8,0

          movlw     B'00000001'    ; Clear ram
          call      WR_INS

         movf      counter23, W
         addlw     B'00110000'
         call      WR_DATA
         movf      counter2, W
         addlw     B'00110000'
         call      WR_DATA

         movlw     B'11000000'
         call      WR_INS

         movlw     "G"
         call      WR_DATA
         movlw     "o"
         call      WR_DATA
         movlw     "l"
         call      WR_DATA
         movlw     "f"
         call      WR_DATA

         goto      display
THREE
          btfss     check_key3, 0
          goto      display

          bsf       check_key1,0
          bsf       check_key2,0
          bcf       check_key3,0
          bsf       check_key4,0
          bsf       check_key5,0
          bsf       check_key6,0
          bsf       check_key7,0
          bsf       check_key8,0

          movlw     B'00000001'    ; Clear ram
          call      WR_INS

         movf      counter33, W
         addlw     B'00110000'
         call      WR_DATA
         movf      counter3, W
```

```
          addlw     B'00110000'
          call      WR_DATA

          movlw     B'11000000'
          call      WR_INS

          movlw     "S"
          call      WR_DATA
          movlw     "q"
          call      WR_DATA
          movlw     "u"
          call      WR_DATA
          movlw     "a"
          call      WR_DATA
          movlw     "s"
          call      WR_DATA
          movlw     "h"
          call      WR_DATA

          goto      display

FOUR
           btfss     check_key4, 0
           goto      display

           bsf       check_key1,0
           bsf       check_key2,0
           bsf       check_key3,0
           bcf       check_key4,0
           bsf       check_key5,0
           bsf       check_key6,0
           bsf       check_key7,0
           bsf       check_key8,0

           movlw     B'00000001'    ; Clear ram
           call      WR_INS

          movf      counter43, W
          addlw     B'00110000'
          call      WR_DATA
          movf      counter4, W
          addlw     B'00110000'
          call      WR_DATA

          movlw     B'11000000'
          call      WR_INS

          movlw     "W"
          call      WR_DATA
          movlw     "h"
          call      WR_DATA
          movlw     "i"
          call      WR_DATA
          movlw     "t"
          call      WR_DATA
          movlw     "e"
          call      WR_DATA
```

```
         movlw     "P"
         call      WR_DATA
         movlw     "-"
         call      WR_DATA
         movlw     "P"
         call      WR_DATA

          goto     display

FIVE
          btfss     check_key5, 0
          goto      display

          bsf       check_key1,0
          bsf       check_key2,0
          bsf       check_key3,0
          bsf       check_key4,0
          bcf       check_key5,0
          bsf       check_key6,0
          bsf       check_key7,0
          bsf       check_key8,0

          movlw     B'00000001'    ; Clear ram
          call      WR_INS

         movf      counter53, W
         addlw     B'00110000'
         call      WR_DATA
         movf      counter5, W
         addlw     B'00110000'
         call      WR_DATA

         movlw     B'11000000'
         call      WR_INS

         movlw     "O"
         call      WR_DATA
         movlw     "r"
         call      WR_DATA
         movlw     "a"
         call      WR_DATA
         movlw     "n"
         call      WR_DATA
         movlw     "g"
         call      WR_DATA
         movlw     "e"
         call      WR_DATA
         movlw     "P"
         call      WR_DATA


         goto      display

SEVEN
          btfss     check_key6, 0
          goto      display
```

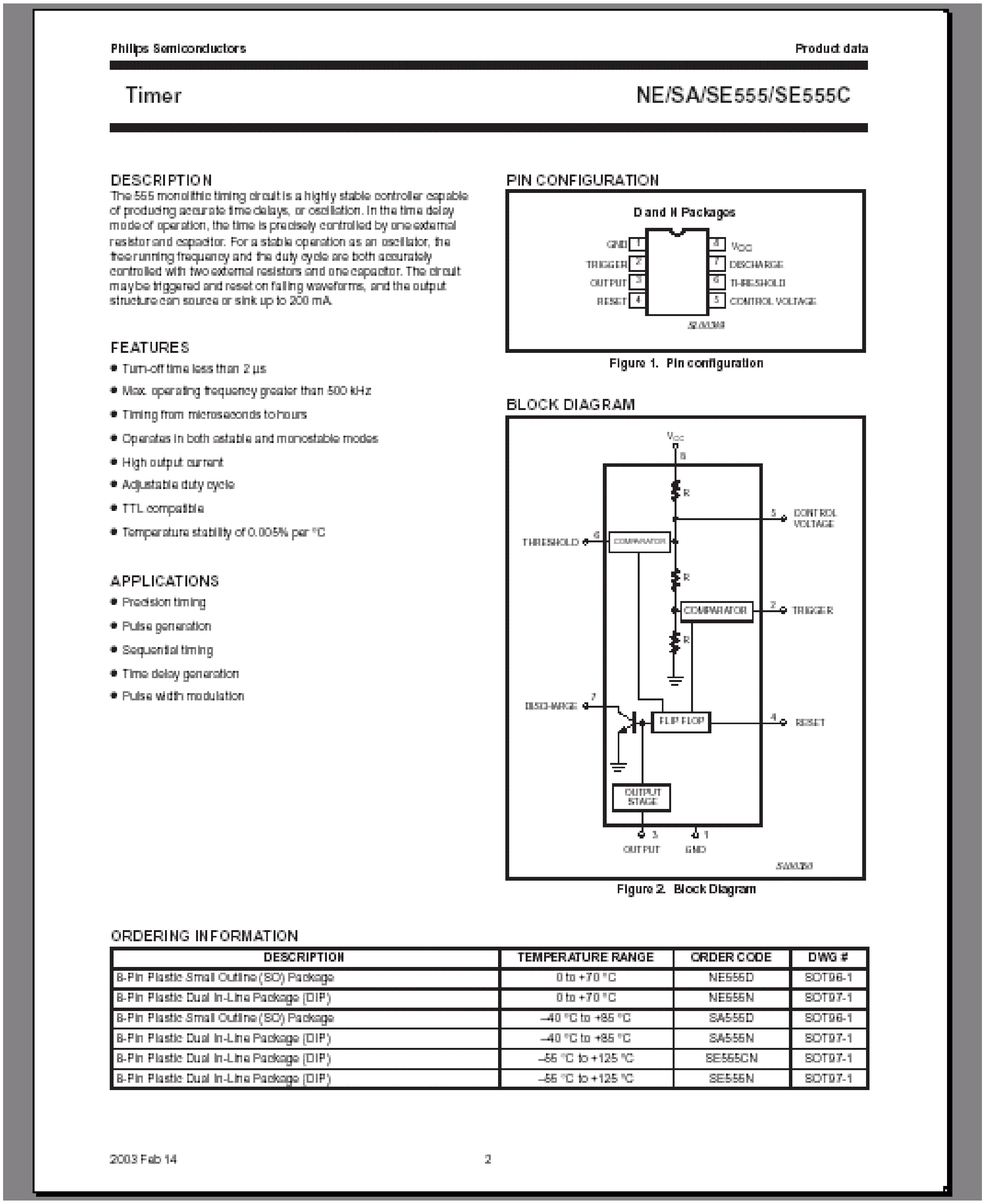
```
           bsf       check_key1,0
           bsf       check_key2,0
           bsf       check_key3,0
           bsf       check_key4,0
           bsf       check_key5,0
           bcf       check_key6,0
           bsf       check_key7,0
           bsf       check_key8,0

           movlw     B'00000001'    ; Clear ram
           call      WR_INS

          movf      counter63, W
          addlw     B'00110000'
          call      WR_DATA
          movf      counter6, W
          addlw     B'00110000'
          call      WR_DATA

          movlw     B'11000000'
          call      WR_INS

          movlw     "T"
          call      WR_DATA
          movlw     "o"
          call      WR_DATA
          movlw     "t"
          call      WR_DATA
          movlw     "a"
          call      WR_DATA
          movlw     "l"
          call      WR_DATA
          goto      display

EIGHT
           btfss     check_key7, 0
           goto      display

           bsf       check_key1,0
           bsf       check_key2,0
           bsf       check_key3,0
           bsf       check_key4,0
           bsf       check_key5,0
           bsf       check_key6,0
           bcf       check_key7,0
           bsf       check_key8,0

           movlw     B'00000001'    ; Clear ram
           call      WR_INS

          movf      counter_timer5, W
          addlw     B'00110000'
          call      WR_DATA
          movf      counter_timer3, W
          addlw     B'00110000'
          call      WR_DATA
          movf      counter_timer, W
```

```
         addlw     B'00110000'
         call      WR_DATA
         movlw     "s"
         call      WR_DATA

         movlw     B'11000000'
         call      WR_INS

         movlw     "T"
         call      WR_DATA
         movlw     "i"
         call      WR_DATA
         movlw     "m"
         call      WR_DATA
         movlw     "e"
         call      WR_DATA


         goto      display

;;*********************************************************************


;*********************************************************************
;;;;;;;;;;Initial subroutine;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;Initialize all the PORT settings and the variables

Initial
          call      delay
          call      delay
          bsf       STATUS,RP0     ; select bank 1
          clrf      TRISD          ; All port D is output
          bcf       STATUS,RP0     ; select bank 0

          movlw     B'00110011'    ;
          call      WR_INS
          movlw     B'00110010'
          call      WR_INS
          movlw     B'00101000'    ; 4 bits, 2 lines,5X7 dot
          call      WR_INS
          movlw     B'00001100'    ; display on/off, blinking cursor
          call      WR_INS
          movlw     B'00000110'    ; Entry mode
          call      WR_INS
          movlw     B'00000001'    ; Clear ram
          call      WR_INS



        bsf       STATUS,RP0
        movlw     B'01111111'
        movwf     TRISC
        bcf       STATUS,RP0
        bcf      STATUS, RP0
        bcf      STATUS, RP1
        clrf     PORTA

        bsf      STATUS, RP0
        movlw    0x06
        movwf    ADCON1
          movlw    0xCF

          movwf    TRISA
          bcf      STATUS,RP0

          bcf      PORTC, 7
          movlw    B'00000000'
          movwf    counter_timer
          movlw    0
          movwf    check_timer
          movlw    B'00000000'
          movwf    index1
          movlw    10
          movwf    counter_timer2
          movlw    0
          movwf    counter_timer3
          movlw    10
          movwf    counter_timer4
          movlw    0
          movwf    counter_timer5

          movlw    B'00000000'
          movwf    counter1
          movlw    B'00000000'
          movwf    check1
          movlw    B'00000000'
          movwf    index1
          movlw    10
          movwf    counter12
          movlw    0
          movwf    counter13

          movlw    B'00000000'
          movwf    counter2
          movlw    B'00000000'
          movwf    check2
          movlw    B'00000000'
          movwf    index2
          movlw    10
          movwf    counter22
          movlw    0
          movwf    counter23

          movlw    B'00000000'
          movwf    counter3
          movlw    B'00000000'
          movwf    check3
          movlw    B'00000000'
          movwf    index3
          movlw    10
```

```
movwf    counter32
movlw    0
movwf    counter33

movlw    B'00000000'
movwf    counter4
movlw    B'00000000'
movwf    check4
movlw    B'00000000'
movwf    index3
movlw    10
movwf    counter42
movlw    0
movwf    counter43

movlw    B'00000000'
movwf    counter5
movlw    B'00000000'
movwf    check5
movlw    B'00000000'
movwf    index5
movlw    10
movwf    counter52
movlw    0
movwf    counter53

movlw    B'00000000'
movwf    counter6
movlw    B'00000000'
movwf    check6
movlw    B'00000000'
movwf    index3
movlw    10
movwf    counter62
movlw    0
movwf    counter63

movlw    15
movwf    finish?

movlw    1
movwf    check_check1
movlw    1
movwf    check_check2
movlw    1
movwf    check_check3
movlw    1
movwf    check_check4
movlw    1
movwf    check_check5
movlw    1
movwf    check_check6
movlw    1
movwf    check_check_timer


bsf       check_key1,0
```

```
          bsf       check_key2,0
          bsf       check_key3,0
          bsf       check_key4,0
          bsf       check_key5,0
          bsf       check_key6,0
          bsf       check_key7,0
          bsf       check_key8,0



        return

;**********************************************************************

;****************************************
; Write command to LCD
; Input  : W
; output : -
;****************************************


WR_INS    bcf       RS        ; clear RS
          movwf     com      ; W --> com
          andlw     0xF0      ; mask 4 bits MSB  W = X0
          addlw     8
          movwf     PORTD     ; Send 4 bits MSB
          call      delay
          bcf       E         ;
          call      delay     ; __    __
          bsf       E         ;      |__|
          swapf     com,w
          andlw     0xF0      ; 1111 0010
          addlw     8
          movwf     PORTD     ; send 4 bits LSB
          call      delay
          bcf       E         ;
          call      delay     ; __    __
          bsf       E         ;   |__|
          call      delay
          clrw
          return

;***************************************
; Write data to LCD
; Input  : W
; Output : -
;***************************************
WR_DATA   bsf       RS
          movwf     dat
          movf      dat,w
          andlw     0xF0
          addlw     0xC
          movwf     PORTD
          call      delay
          bcf       E
          call      delay     ; __    __
          bsf       E         ;   |__|
          swapf     dat,w
          andlw     0xF0
          addlw     0xC
          movwf     PORTD
          call      delay
          bcf       E         ;
          call      delay     ; __    __
          bsf       E         ;   |__|
          clrw
          return

;****************************************
; Delay
;****************************************
delay
TenMs
        movlw TenMsH
        movwf COUNTH
        movlw TenMsL
        movwf COUNTL
Ten_1
        decfsz COUNTL,F
        goto Ten_1
        decfsz COUNTH,F
        goto Ten_1

    return


end
```

6.10 Appendix H – Selected Datasheets



Timer NE/SA/SE555/SE555C

DESCRIPTION

The 555 monolithic timing circuit is a highly stable controller capable of producing accurate time delays, or oscillation. In the time delay mode of operation, the time is precisely controlled by one external resistor and capacitor. For a stable operation as an oscillator, the free running frequency and the duty cycle are both accurately controlled with two external resistors and one capacitor. The circuit may be triggered and reset on falling waveforms, and the output structure can source or sink up to 200 mA.

FEATURES

- Turn-off time less than 2 µs
- Max. operating frequency greater than 500 kHz
- Timing from microseconds to hours
- Operates in both astable and monostable modes
- High output current
- Adjustable duty cycle
- TTL compatible
- Temperature stability of 0.005% per °C

APPLICATIONS

- Precision timing
- Pulse generation
- Sequential timing
- Time delay generation
- Pulse width modulation

PIN CONFIGURATION

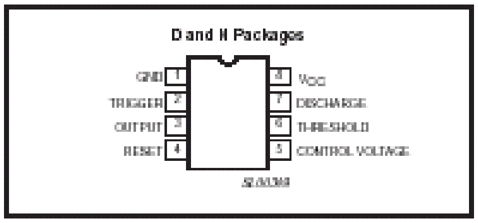


Figure 1. Pin configuration

BLOCK DIAGRAM

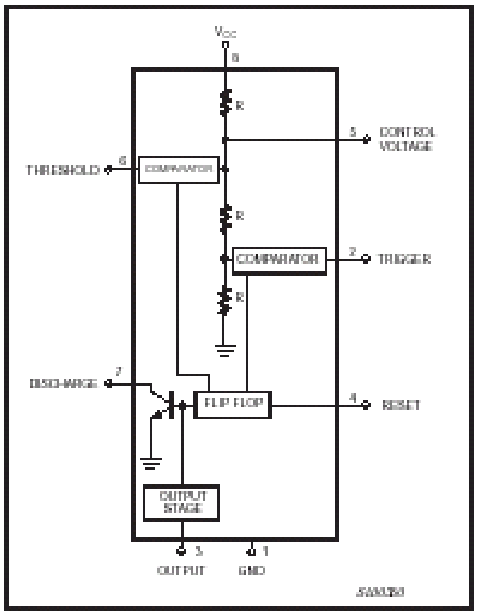


Figure 2. Block Diagram

ORDERING INFORMATION

DESCRIPTION	TEMPERATURE RANGE	ORDER CODE	DWG #
8-Pin Plastic Small Outline (SO) Package	0 to +70 °C	NE555D	SOT96-1
8-Pin Plastic Dual In-Line Package (DIP)	0 to +70 °C	NE555N	SOT97-1
8-Pin Plastic Small Outline (SO) Package	–40 °C to +85 °C	SA555D	SOT96-1
8-Pin Plastic Dual In-Line Package (DIP)	–40 °C to +85 °C	SA555N	SOT97-1
8-Pin Plastic Dual In-Line Package (DIP)	–55 °C to +125 °C	SE555CN	SOT97-1
8-Pin Plastic Dual In-Line Package (DIP)	–55 °C to +125 °C	SE555N	SOT97-1