LED SWIM PACER

By

Jonathan Lee

Yi-Liang Chen

Final Report for ECE 445, Senior Design, Spring 2012

TA: Ryan May

2 /May 2012

Project No. 23

Abstract

The LED Swim Pacer allows a swimmer to pace and improve himself. This is a sequential LED strip to be placed underwater and allows the swimmer to see and follow the LED lights. The swimmer can set his pace from the waterproof control box on the side of the pool. Our swim pacer consists of different modes for the user to choose from which allows the swim coach to interact with swimmers by controlling the speed of the pacer.

The user interface from the controller box allows easy control over the pacer and lets the swim coach monitor the performance of the swimmer with respect to the pacer speed.

Our project design was very successful and performed as intended. There are still improvements to be made for the power source, brightness of the LED and the programming. The process of designing the components of this project and the in-depth tests are extensively described in this document.

Contents

1. Introduction

This project is brought forth by Coach Howard as he wanted a training tool to help him improve the performance of his swimmers. There are several running pacers on the market which trains the runners to run faster as the pace increase; however there are no swimming pacers. Coach Howard wanted a swim pacer that allows him to interact with his swimmers by setting paces for the swimmers to follow. This is a more efficient method than just verbally telling swimmers to swim faster or telling them how fast they swam for the previous laps.

1.1 Purpose

We designed and built a variable speed sequential strip of LED lights which can be controlled from a control box. This allows the user, either the coach or the swimmer, to easily alter the speed of the LED lights. The LED strip gives a visual representation of the pace which lets the swimmers be aware of their speed and therefore improving their pace and consistency.

1.2 Function

This LED swim pacer uses sequential LED lights to pace the swimmers. Placing it directly under the swimming pool, it will allow the swimmer to easily spot the LED as it turns on. The swim pacer strip is in one inch clear tubing which is simple to deploy. The waterproof control box has a display and 4 push buttons making it easy to operate. The LED strip consists of green and red LEDs with the red ones at each end of the strip to warn swimmers that the end of the pool is close. The red LEDs are programmed to be faster to account for the push off from the pool wall.

1.3 Block Diagram

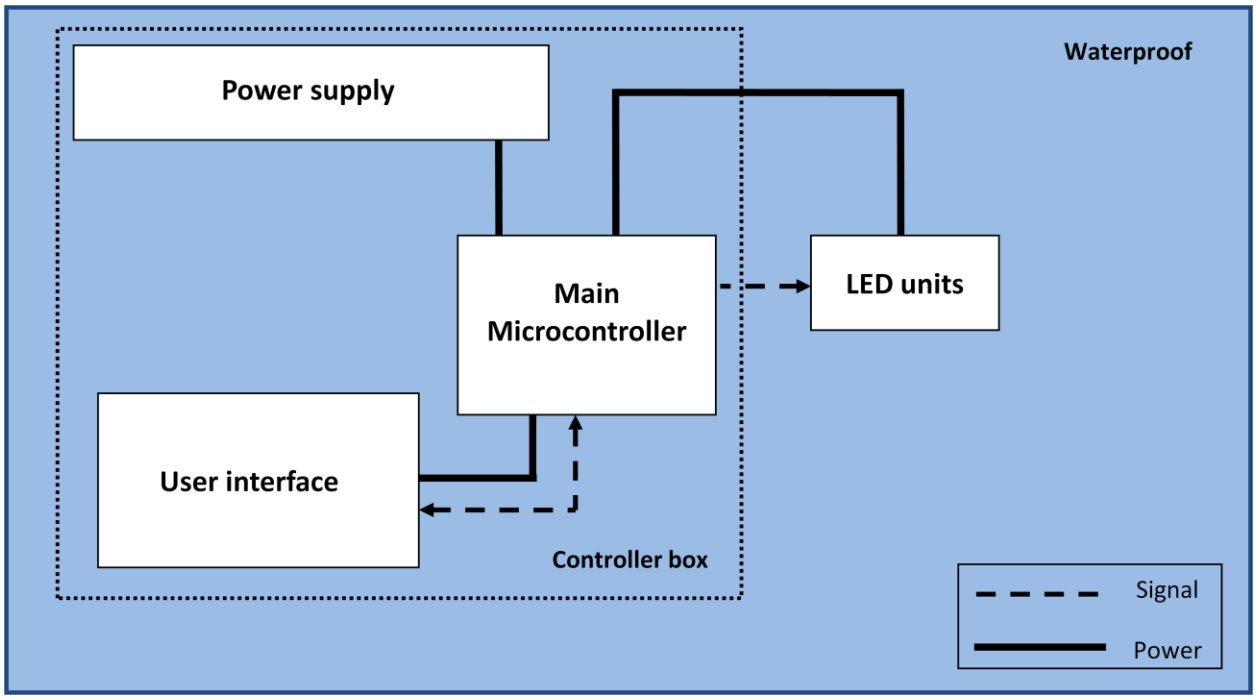

Figure 1 Final Block Diagram

Our design can be split up into two different sections, one is the user interface and the other is the LED units. Both are controlled by the main microcontroller.

1.3.1 Main Microcontroller

The main microcontroller controls the LCD display, user button inputs and computes that data. Using the data set by the users, it calculates the correct clock cycle and sends it out to the LED units.

1.3.2 User Interface

The user interface includes a 16 character x 2 line LCD and also four push buttons. The LCD displays the pacer menu and prompts the users for input. The four push buttons allows the user to select the modes and the timing he desires.

1.3.3 LED units

The LED units are a strip with a total length of 25 meters. It consists of 6 bidirectional shift registers and each of the shift register controls eight LEDs. Each of the LEDs unit consists of two 100 ohm resistor and two LEDs.

1.3.4 Power supply

The power supply of this pacer is six AA alkaline batteries. They will provide a total of 9 volts to the main microcontroller. From the main microcontroller, the on board voltage regulator will provide 5 volts to the LCD display and LED units.

1.4 Performance Requirements

One of the main requirements for the swim pacer is that it needs to be accurate to the 100ms. Several tests for different timings were made and a stopwatch was used to measure the time. The Swim Pacer is verified to be accurate to the 100ms from the data in Appendix C, Tables 5 to 7. The LCD also correctly displays the intended messages and changes the messages accordingly from the user inputs. The whole pacer is designed to be waterproof and the control box is sealed tight by using silicone sealant. The first 4 meters of LEDs are programmed to be faster and they are also tested to ensure that the pacer is still within the 100ms accuracy. More detailed process of the verification and design are further explained below.

1.5 Changes

One of the major changes that we made was that in the original design, we planned on using two microcontrollers: one for controlling the user interface and the other for controlling the LED units. After analysis and testing, we decided to use one microcontroller to control the whole pacer. This decreases the space in the control box and makes the pacer more efficient.

2 Design

2.1 Design Procedure

2.1.1 Main Microcontroller

In our initial design, we plan on using two microcontrollers, one for user interface and the other for LED units. We had programmed most of the code for the two microcontrollers; however, after analysis and testing, we decided that one microcontroller is sufficient. Using just one microcontroller makes the project more efficient and also save space in the control box. We also had plans to design several microcontrollers inside the tubing but it's more efficient to just use one microcontroller to control all the LEDs. The whole design schematics can be found in Appendix B, Figure 20.

2.1.2 User Interface

There are several options for the user interface. We can get different LCD and buttons however, since the ECE parts shop had a 16x2 LCD with 4 push buttons embedded, we decided to implement our user interface design over it. We tried to reduce the buttons required for this swim pacer because we don't want the user interface to be too complicated for the users.

2.1.3 LED Units

We planned on making this swim pacer a sequential LED strip because it's easy to deploy and simple for the swimmers to utilize. Placing it under the pool also allows the swimmers to spot the lights easily because swimmers are always looking down at a line which guides them to swim straight. We could have reduced the LEDs to just 1 meter separation and easily decrease the amount of wiring and design needed. However we found that it's more suitable to make the LED in an increment distance of 50cm so that the gap is not too big for slow swimmers. One other design is to wire all the LEDs to the microcontroller without using any shift registers. It will end up with about 50 wires going through the tube which is very inefficient. So for our design, only 13 wires are connected to the microcontroller and the rest of the 50 LEDs are controlled by bidirectional shift registers built into the LED strip. This significantly reduces the wiring needed.

2.1.4 Power Supply

In our original design, we wanted to connect four AA alkaline batteries in series and add another four AA alkaline batteries in parallel. This will supply the microcontrollers with 6 volts. After some testing, we decided to make the batteries by connecting 6 of them in series. This will provide 9 volts to the microcontroller. Another design is to use an AC outlet for power, however this pacer will be placed under the pool and we can't afford to create electrical hazards for the swimmers. Although this will improve the usage time of the pacer, we found it to be safer to stick with alkaline batteries.

2.2 Design details

2.2.1 Main Microcontroller

The main microcontroller controls both the user interface and the LED unit.

For the user interface, the microcontroller displays the menu system and also prompts the user for inputs. Looking at Appendix A, Figure 15, it shows the flowchart of how the LCD message changes as it responds to the user inputs. Figure 16, shows the flowchart of how the microcontroller prompt the user for input and stores that data, waiting for the user to begin the pacer.

The swim pacer consists of 3 different modes for the users to choose from. The mode overview is in Appendix A, Figure 13. Mode 1 allows user to select one timing for the lap, mode 2 allows users to choose from two different timings. In mode 2, users can choose to swim at a faster pace and then a slow pace or vice versa. In mode 3, there will be 3 different timings for the users to select from. The pacer in mode 3 can vary from fast, slow, fast to slow, slow fast. It all depends on how the user wants to train himself. Figure 2 below gives a general idea about the different modes

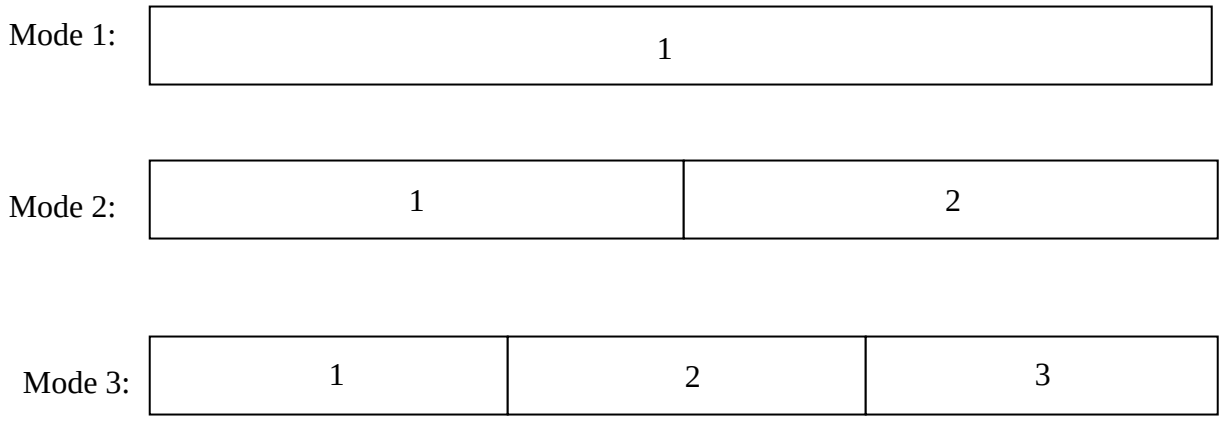

Figure 2 Different timings to choose for in different modes

In appendix A, figure 17, it shows the microcontroller flowchart for how the timing selected by the user get computed and sent out to the shift registers. If the speed updates or the reset button is push, the microcontroller will act accordingly.

The calculation for the clock cycle time is very important because the pacer needs to be accurate to the 100ms. One of the functions of the swim pacer is that in the first 4 meters, the LEDs will move at a faster pace to account for the push off from the pool wall. In the first calculation below, it will ignore this function.

Not accounting for increased pace for first 4 meters

$$ClockCycleTime = SEC * 1000 / 50 \quad (2.1)$$

Looking at equation 2.1, it shows how to get the clock cycle time. SEC is the timing that the user selected and it's converted to milliseconds by multiplying it by 1000. It is then divided by 50 LEDs. Note that there are 51 LEDs but the one at the ends doesn't count in the number of clock cycles. Using an example of 25 seconds, we get a clock cycle time of 500ms. This means that the first LED delays for 500ms and turns off and the next LED will turn on and so on.

Accounting for increased pace for first 4 meters

$$ClockCycleTime = \{[4 / (25 / SEC)] / 2\} * 1000 / 8 \quad (2.2)$$

$$ClockCycleTime = \{SEC - \{[4 / (25 / SEC)] / 2\}\} * 1000 / 42 \quad (2.3)$$

For the first 4 meters, it is now programmed to run faster. Looking at equation 2.2, 4 meters is divided by the speed that the user selected. This will give the total time to travel the first 4 meters. This time is then divided by 2 to increase the speed and then it's divided by 8 LEDs. 8 is the number of LEDs in the first 4 meters.

For the rest of the swim pacer, looking at equation 2.3; the time required to swim the first 4 meters is subtracted from the total time that the user selected. With the remaining time, it's divided by the rest of the LEDs. These two equations allows the pacer to be fast at the beginning and the total time for the lap will still be the same as what the user selected.

With the rest of the modes, using the same equations just different number of LEDs, it allows the LED strip to move at different speeds and the accuracy of each lap in mode 2 and 3 are also accurate to the 100ms.

After the main microcontroller received the data from user input, and it processes the data and calculates the clock cycle. Then it sends out individual data input, universal clock and left/right shift select to the 6 LED units. At the same time the microcontroller will update the LCD monitor as well. Any adjustment from the user will be updated to the main microcontroller which will alter the speed accordingly.

The swim pacer code is located in Appendix F.

2.2.2 User Interface

The user interface consist of a 16x2 LCD and 4 push buttons. The LCD will display the menu and prompt user for input. The push buttons allows the user to input their desired mode and timings. The detail LCD logic diagram is in Appendix B, figure 18 and the LCD pin I/O is in Appendix B, Table 4. Figure 3 below shows how the LCD is connected to the main microcontroller. The LCD connected by 4-bit mode which only requires DB4 to DB7. It requires fewer wires to be connected to the microcontroller. On the bottom of the Figure 3 circuit, are the four push buttons. Each of the buttons has a maximum rating of 24Vdc and 50mA. Since the max current is only 50mA, the current from the power supply needs to be reduced, that's why there's a 100 ohm resistor between the 5v source and switch. The power from the microcontroller has a power of 5volts and 0.33A.

$$V = IR \tag{2.4}$$

$$R = V / I$$

$$R = 5V / 0.33A = 15.1515\Omega \tag{2.5}$$

Add $100\,\Omega$ resistor

$$I = V / R = 5V / 115.1515\Omega = 0.0434A \tag{2.6}$$

The current is reduced to 0.43A which is less than it's max current of 0.05A. This allows the buttons to work properly.

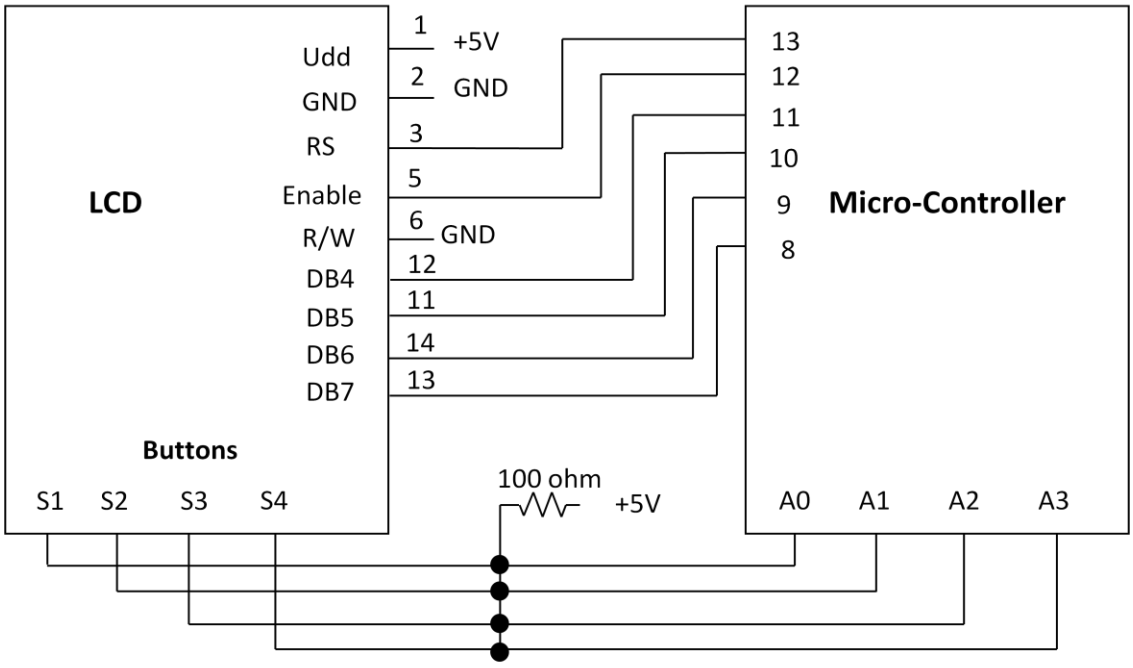

Figure 3 LCD circuit

The instructions for the buttons can be found in Appendix A, Figure 14. The user interface isn't that complex and users can easily understand and memorize how to operate the swim pacer within minutes of practice.

2.2.3 LED Units

The LED unit contains one 8-bit bidirectional shift register (MC74HC299N) and 8 sets of parallel LED bulb, and each LED is connected with a 100Ω resister. The shift register gets input data, shift left/right select, and clock signals from main microcontroller. The shift register will shift the input data to left or right according to the select signal input, and the shift speed is determine by the clock we created in microcontroller. The schematics can be found in Appendix B, Figure 19.

Since the LED strip is 25meters long, we calculated the wire resistance to make sure that it won't cause any significant resistance that will result in the design failing to work. 22 guage wires were used in the swim pacer and we calculated the resistance for 24meters.

$$R = \rho(l / A) \tag{2.7}$$

$$\text{Resistivity}: 0.01614\Omega / \ ft \tag{2.8}$$

$$R = 0.01614 * 78.74\, ft = 1.271\Omega \tag{2.9}$$

We get a resistivity of roughly $1.271\,\Omega$ which is not significant enough to affect the bidirectional shift register or the LEDs.

The following Figure 4 is the shift register simulation. Once the pacer begins, a high signal is sent from the microcontroller to register 1 and the signal is shifted through all 8 LEDs and then the next register will get a high data signal from the microcontroller and so on.

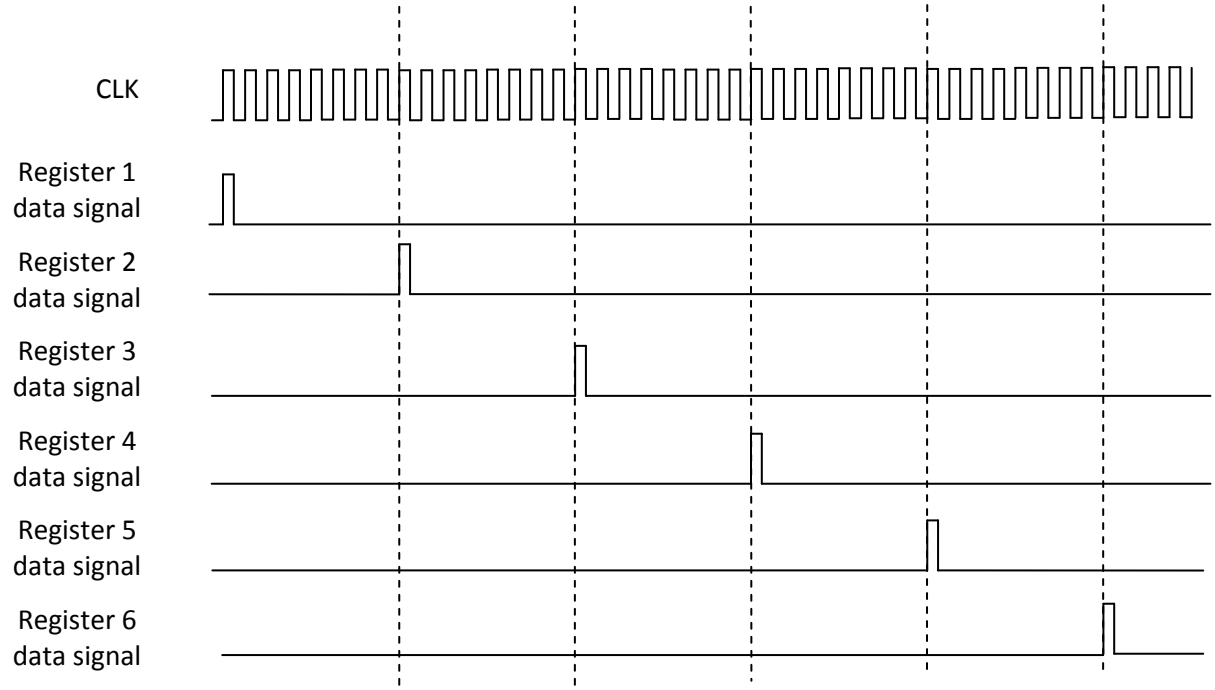

Figure 4 Shift Register simulation

In Appendix C, Figure 21, it shows the simulation for 48 LEDs. As you can see, at each clock cycle only one LED is high. The LEDs turns on one at a time without overlapping. This is tested in our verification in part 3.

2.2.4 Power Supply

We are using 6 AA alkaline batteries with a total of 18watts-hour. This will be used to power the main microcontroller. The LCD and LED units will get their power from the main microcontroller.

Looking at Table 1, you can see the modules and their power usage.

Table 1 Power Supply Overview

Modules	**Power Usage(watts)**
LED	0.1362 watts(for a pair)
Microcontrollers	0.21 watts
Shift registers	1.50 watts
LCD	1.5 watts
Total	**3.3462 watts**

$$18watt-hour / 3.3462watts = 5.38hours \quad (2.10)$$

From the equation 2.10, the 6 AA alkaline batteries can provide more than 5 hours of continuous swim pacer usage. This is a good amount of time for a day's swim session.

3. Design Verification

The testing began with verifying each of the modules individually. Once each modular section was verified and confirmed to have the correct operation, in theory, the whole design will work correctly. The whole system was tested together to check if they are indeed working correctly.

Please refer to Appendix D for the Requirement and Verification Table. All the instruments required to test our project were provided by the ECE445 laboratory. The sections below describe our successful verifications of each module.

3.1 Testing

3.1.1 LCD

The LCD needs to display the main menu when it's turned on. Powered by the 5V from the microcontroller, the LCD is programmed to display "Hello World". Once that test is verified, we can confirm that the wires and LCD are working properly. After that, the main menu is programmed in the microcontroller and the LCD can display it correctly (Figure 5). We then tested each of the buttons to make sure that the microcontroller is receiving the signals from them. Looking at Figure 6, it shows that the microcontroller reads the inputs from the user and responds to the button that is pushed. When the LCD prompted the user to select the desired timing, the microcontroller correctly read the data and reconfirmed the timing that he selected (Figure 7). The LCD is tested and verified to be working correctly.

Figure 5 LCD displaying the main menu

Figure 6 Reading the user inputs

Figure 7 Received correct timing from user input

3.1.2 Main Microcontroller

The main microcontroller doesn't only need to send signals to the LCD and receive data from the user inputs; it has to calculate the correct clock cycle time according to the user input and send it to the LED units. Referring back to part 2, design, it shows the clock cycle time equation, Equation (2.1). To test if the microcontroller is processing the data correctly, a test speed of 25seconds was sent to the microcontroller and after calculation, the clock cycle time is displayed on the LCD, Figure 8. This shows that the clock cycle time is 500ms which is correct.

$$Time = ClockCycleTime * 50LEDs / 1000 \qquad (3.1)$$
$$500ms * 50LEDs / 1000 = 25\sec$$

From equation 3.1, it shows that the sum of each of the clock cycle time adds up to match the timing that the user selected.

Figure 8 Clock cycle time (ms) from the test data

3.1.3 LED units

The LED units should light up when it receives a high signal from the shift registers and turn off accordingly. The LED and shift registers are tested on a breadboard to confirm that they turn on and off correctly. Looking at Figure 9, at any moment, there is only one LED lit up. Several tests on the breadboard verify that the LED units are working as intended and passed the test. This matches with our simulation located in Appendix C, Figure 21.

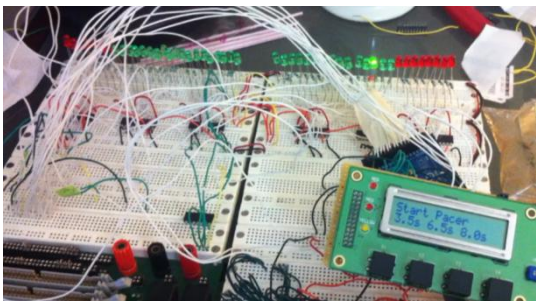

Figure 9 LED strip prototype on breadboard

3.1.4 LED strip

The LED strip has a minimum of 10 seconds (Figure 10) and maximum time of 40 seconds (Figure 11). The program prevents the user to select times out of that range. To test if the LED strip correctly shows a pace of 10 seconds and 40 seconds, a stop watch is used to verify that the pace is accurate to the 100ms. Refer to Appendix C, Tables 5 to 7 for the test results. Some of the times were off by more than 100ms however, this is caused by the reaction time that the stopwatch user has. Overall, the pacer is accurate to the 100ms and passes the verification.

Figure 11 Minimum of 10 seconds for the pacer

Figure 10 Maximum of 40 seconds for the pacer

3.1.5 Power Supply

6 Energizer AA alkaline batteries are connected in series to provide power to the main microcontroller. Looking at Figure 12, the input voltage from the batteries has a voltage of 8.56V. The microcontroller's on board voltage regulator provides 4.98V to the LCD and the LED units. This passes the verification which required the microcontroller to provide a voltage not less than 2 volts or more than 7 volts.

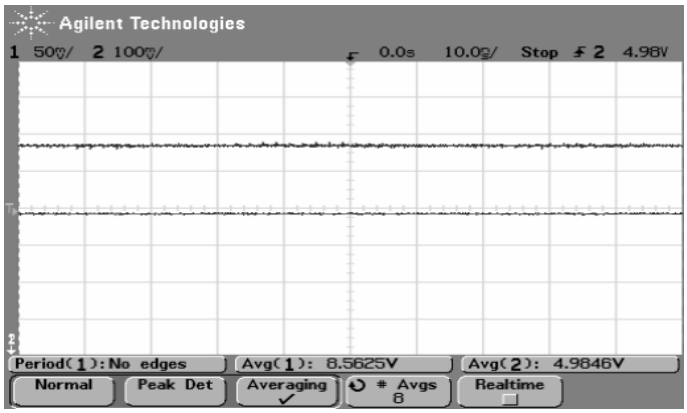

Figure 12 Input voltage from batteries: 8.56V
Output voltage from regulator 4.98V

3.2 Discussion of results and failed verifications

All of the verifications have passed. Each of the modules was verified individually and all of them met the functional performance requirements. Since each of the modules worked as designed, this indicated that our whole design should work. Combining them together, the whole project was once again verified. We are pleased with the performance of our project as it can display the main menu once it powers on and is able to read the timing that the user selected. Using that data, the microcontroller can then correctly compute the right clock cycle time and send them to the bidirectional shift registers.

4. Costs

4.1 Parts

Table 2 Parts Costs

Part Name	Part Number	Manufacturer	Quantity	Retail Cost ($)	Bulk Purchase Cost ($)	Actual Cost ($)
16x2 LCD display with 4 push buttons	HD44780	Hitachi	1	$13.95	$8.95	Free
Arduino Mega 2560	ATmega2560	Arduino	1	$69.99	$53.95	$69.99
LED (Green)	LTL-4238	Lite-On Inc.	60	$16.80	$6.00	Free
LED (Red)	LTL-307EE	Lite-On Inc.	40	$11.20	$4.00	Free
Wire(100ft) 22 Gauge	-	Alpha	7	$34.93	$28.00	Free
Energizer alkaline Battery AA	-	Energizer	6	$9.42	$9.00	$9.42
Resister ¼ watt 100 ohm	-	Ohmite	100	$15.00	$12.00	Free
Clear PVC Tubing (1 inch) 100 ft	-	Anderson Barrows Metals	1	$82.92	$69.69	$82.92
Clear Silicone Sealant	00684	Dap Corning	1	$6.95	$6.90	$6.95
6.7x4.7x2 Nema Plastic Box	PN-1324	BUD	1	$13.47	$13.00	Free
Bidirectional shift register	MC74HC299N	Motorola	6	$9.00	$6.00	Free
Dual AA battery holder	12BH322B-GR	Mouser	3	$2.61	$2.40	Free
4.5x6.5 inch Solderable Perf Boards	276-147	Radio Shack	6	$22.56	$19.50	Free
9Volt battery clip	Keystone233	Keystone	3	$0.63	$0.30	Free
Total				**$309.43**	**$239.69**	**$169.28**

The majority of the parts are reimbursed by the ECE department and obtained from the ECE parts shop. We believe that we can replace the AA batteries with other types of batteries and reduce the size of the LED module. This will decrease the size of our clear PVC tubing from 1 inch in diameter to something thinner. It will also lower the total budget. The wires used in the LED strip could also be reduced to save some cost. Looking at Table 4, it shows that the retail cost of one unit is $309.43. The cost of mass

production using the bulk purchase is roughly $239.69. The amount that our team and the department paid for is $169.28 for this swim pacer unit.

4.2 Labor

Table 3 Labor Cost

Name	Hourly Rate($/hr)	Total hours to complete (hrs)	Total ($)	Total x 2.5 ($)
Jonathan Lee	$32/hr	150hrs	$4800	$12,000
Yi-Liang Chen	$32/hr	150hrs	$4800	$12,000
			Grand Total	$24,000

Using the equation:

$$\textit{Total Labor Cost(\$) = \# of group members*ideal salary(hourly rate)*actual hours spent*2.5(multiplier)} \quad (4.1)$$

In equation 4.1, the number of group members is two and factoring in a multiplier of 2.5 for electronics and machine shop hours, we get the total labor cost of $24,000.

$$\textit{Total Cost(\$) = Total Labor Cost(\$) + Total Parts Actual Costs(\$)} \quad (4.2)$$

The total cost of the entire project using equation 4.2, we get a final amount of $24,169.28.

5. Conclusion

5.1 Accomplishments

We are very pleased with the result of what we were able to accomplish. All the goals where met and we were able to get the swim pacer to work during the demonstration. We managed to make our whole product waterproof and ready to be placed under the swimming pool for testing whereas other swim pacer groups wasn't able to do so during the demonstration. During the testing stage, when both of our microcontrollers failed to work, we redesigned our project immediately, changing two microcontrollers into one. There wasn't might needed changes in our programming and we managed to save space inside the controller box. Although we had to make changes to the microcontroller and batteries in our design, it didn't hinder our project and we still managed to get the swim pacer to work accordingly.

5.2 Uncertainties

Although the swim pacer is built to be waterproof, it hasn't yet been tested under the pool for an extended period of time. The 100 feet tube needs to be underwater and the amount of weights needed to keep it submerged and stable is also unconfirmed.

5.3 Ethical considerations

The safety of the users is very important and we don't believe the project have any ethical issues conflicting with the IEEE ethic code. IEEE Code of Ethics issue #9 states, "to avoid injuring others, their property, reputation, or employment by false or malicious action". Since the product will be submerged underwater, there might be a chance of causing electric shock to users. Therefore the product will be sealed tight and waterproofed. The main controller will also be waterproofed even though it will be on the ground most of the time. The power is supplied by batteries, not AC line input therefore it will reduce the potential danger of electric shock.

5.4 Future work

Due to the limited amount of time to design, build and test this project, we could not get everything the way we wanted to. There had to be changes in our design during the built but it turns out to be better than what we initially designed. For future work, we plan to use more powerful LED lights to allow the swimmers to see better in outdoor pools. We also plan to make allow the controller box to be powered by an AC outlet. This can save the trouble of changing batteries and bringing extra batteries during a swim session. With our swim pacer built, we realized that we can cut down on the size of the tubing and reduce the weight of the product. We also plan to make the control box wireless so that it doesn't have to be connected to the LED strip.

References

[1] "Arduino Examples", 2012. Available at: http://arduino.cc/en/Guide/Windows

[2] "Arduino Forum", 2012. Available at: http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1238350686

[3] "LCD interfacing with Microcontrollers tutorial", 2012. Available at: http://www.8051projects.net/lcd-interfacing/lcd-4-bit.php

[4] IEEE code of ethics, 2012. Available at: http://www.ieee.org/portal/pages/iportals/aboutus/ethics/code.html

[5] LCD Front Panel Set with HD44780 controller, 4 push buttons data sheet. Available at: http://www.piclist.com/techref/io/lcd/panel1.htm

[6] 8-bit bidirectional shift register, MC74HC299 Data Sheet. Available at: http://www.alldatasheet.com/datasheet-pdf/pdf/167380/MOTOROLA/MC74HC299.html

Appendix A Flow Charts, Diagrams

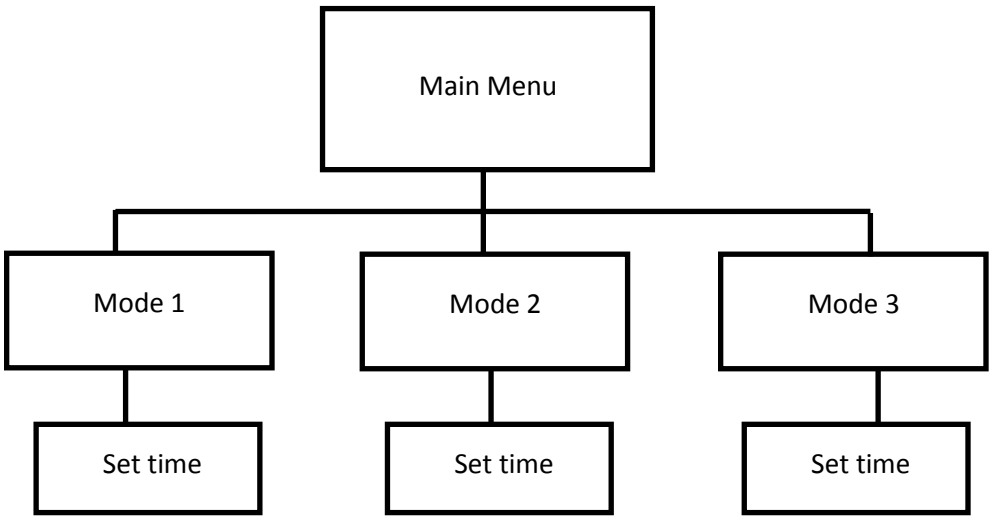

Figure 13 Program Overview

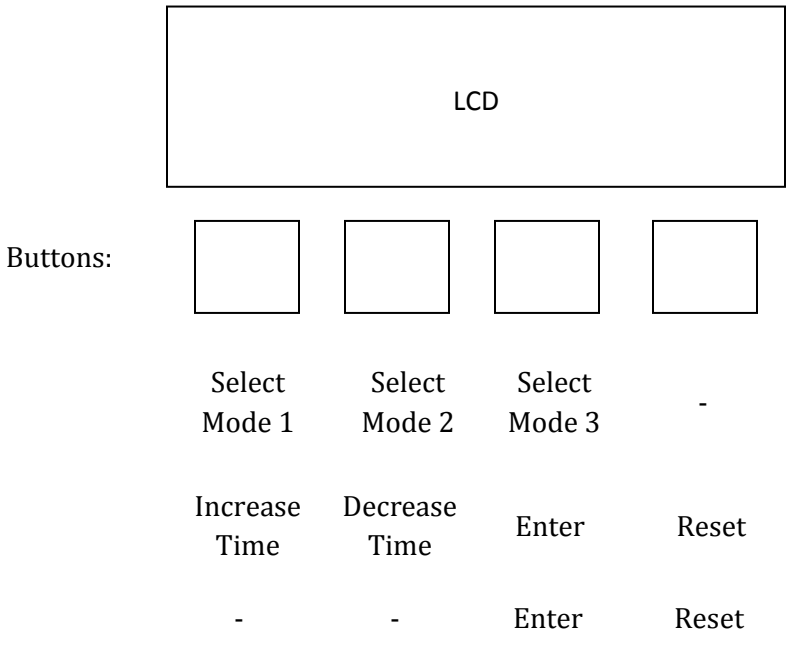

Figure 14 Swim pacer instructions

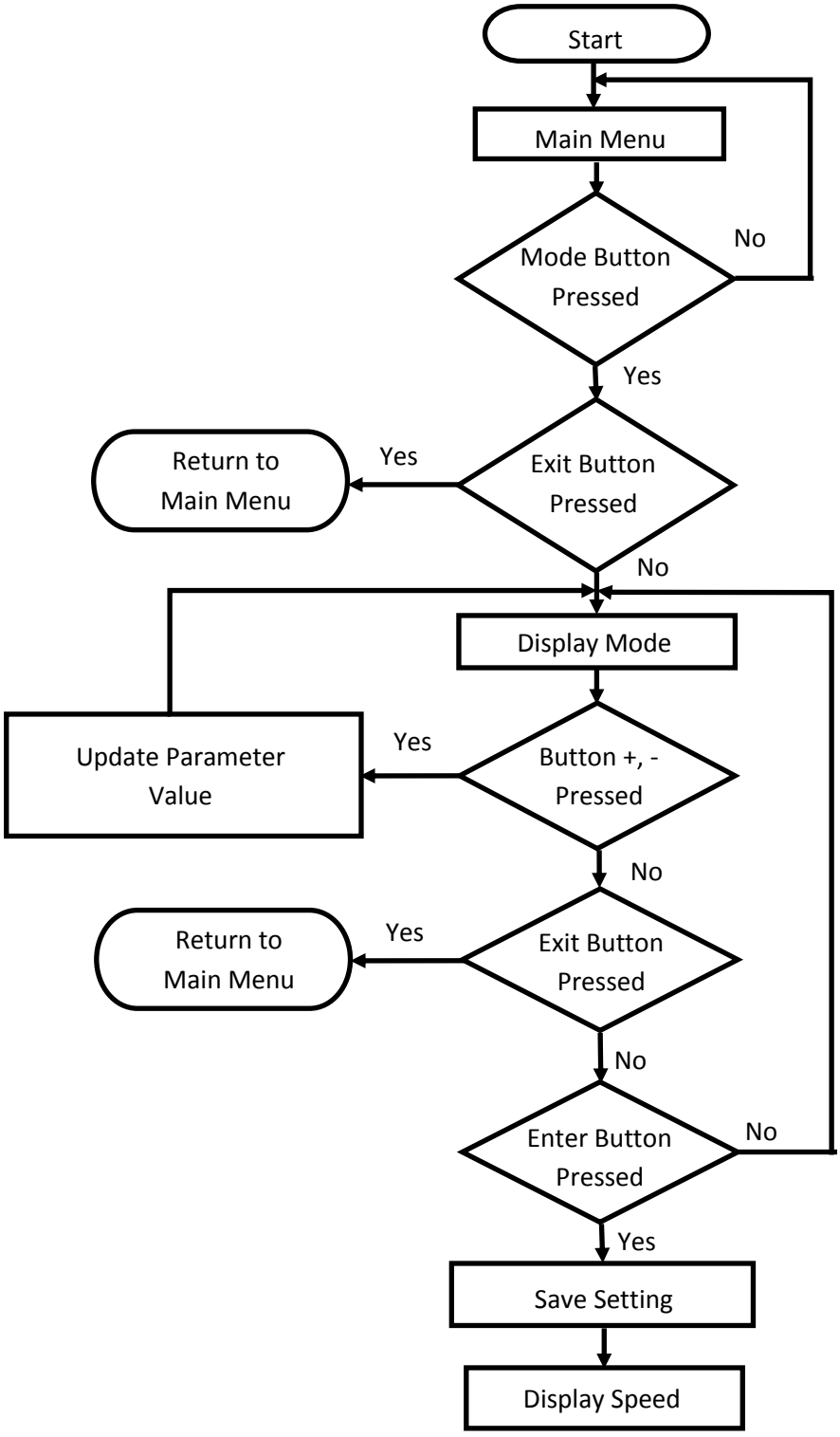

Figure 15 LCD flowchart

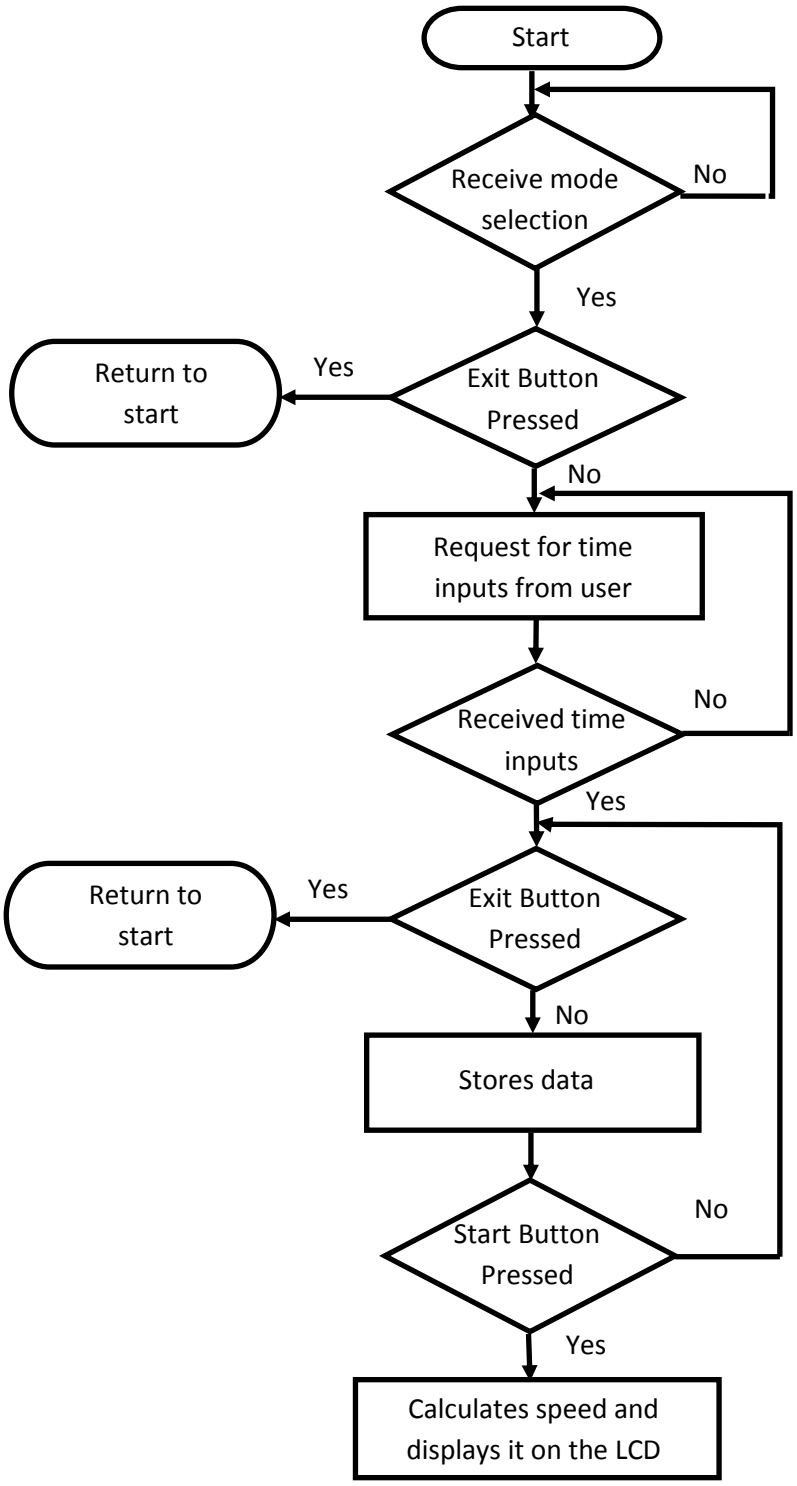

Figure 16 Microcontroller flowchart (LCD)

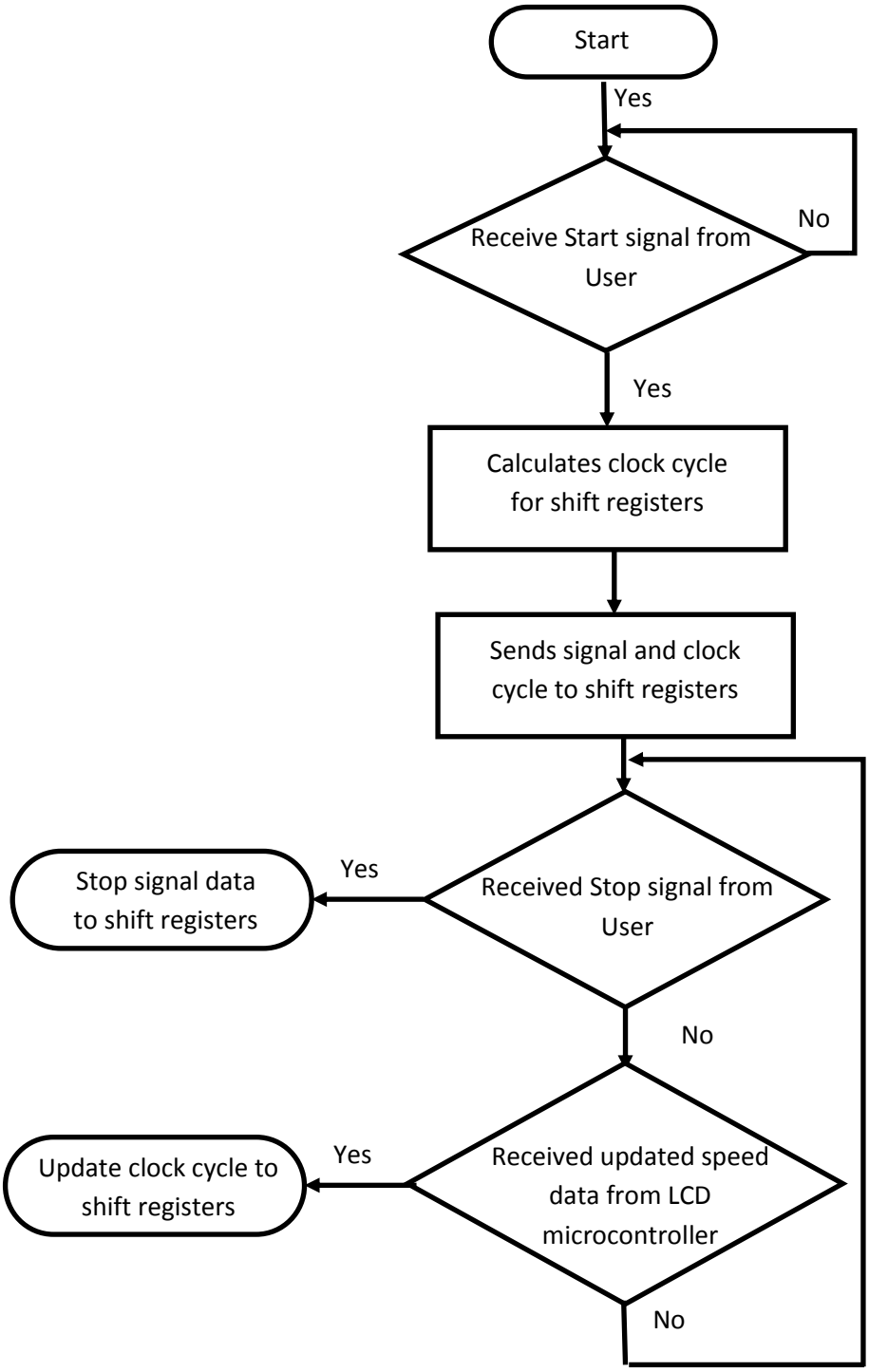

Figure 17 Microcontroller flowchart (LED units)

Appendix B Schematics

Table 4 LCD pin I/O

Pin NO	Symbol	Function
1	V_{DD}	+5V
2	V_{SS}	Ground
3	RS	Register select
4	NC	No connection
5	E	Read/write enable
6	R/W	Read/write
7	DB1	Data bit
8	DB0	Data bit
9	DB3	Data bit
10	DB2	Data bit
11	DB5	Data bit
12	DB4	Data bit
13	DB7	Data bit
14	DB6	Data bit
15	CLED	LED C(+)
16	ALED	LED A(+)
17	S1	Switch 1
18	BLED	LED B(+)
19	S3	Switch 3
20	S2	Switch 2
21	S4	Switch 4
22	NC	No connection
23	V_{SS}	Ground
24	V_{DD}	+5V

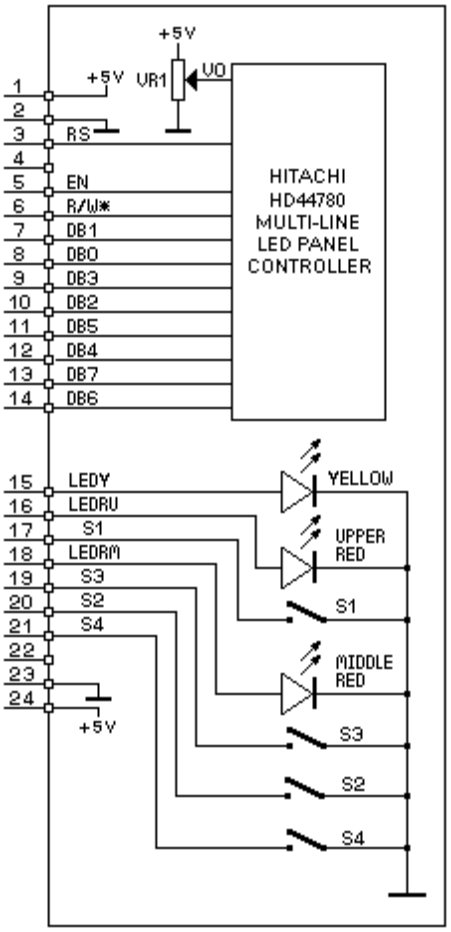

Figure 18 LCD Logic Diagram

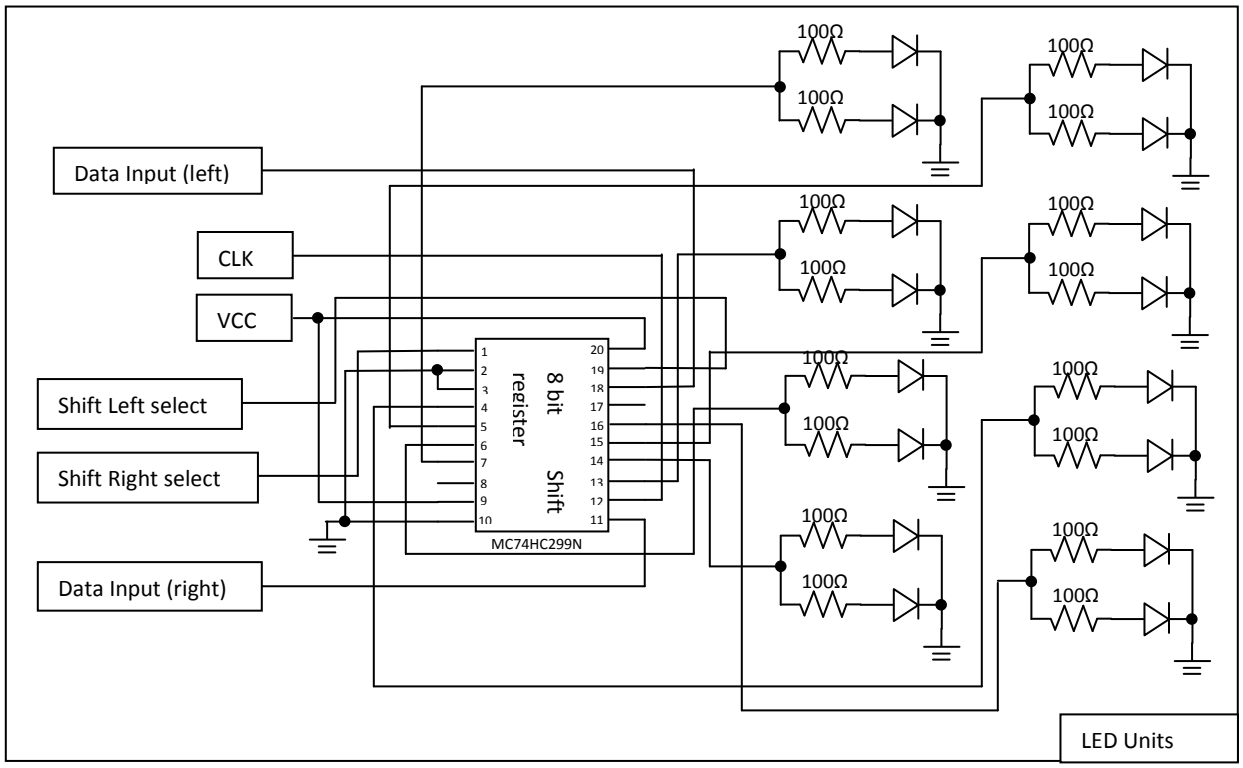

Figure 19 Shift Register Schematics

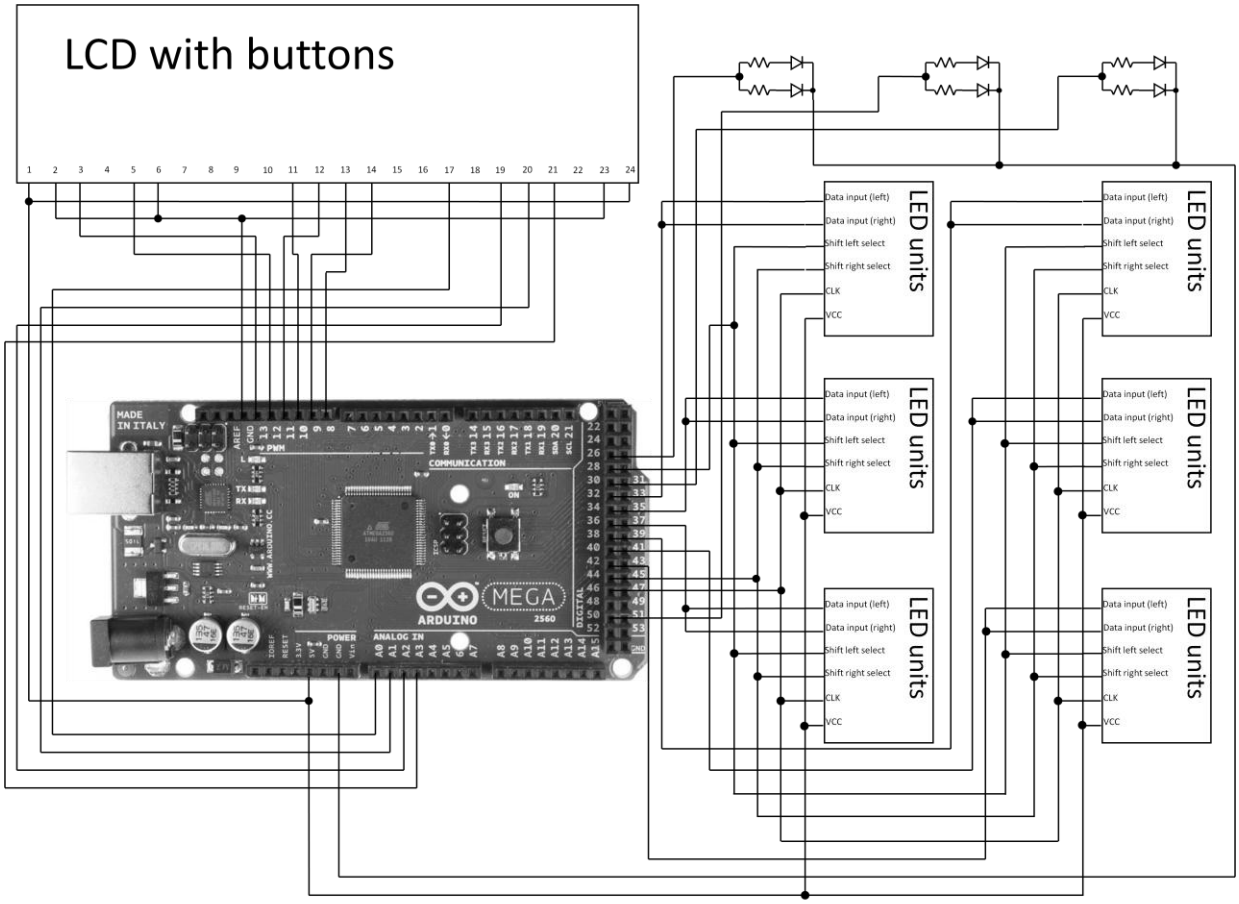

Figure 20 Design Schematics

Appendix C Test Data

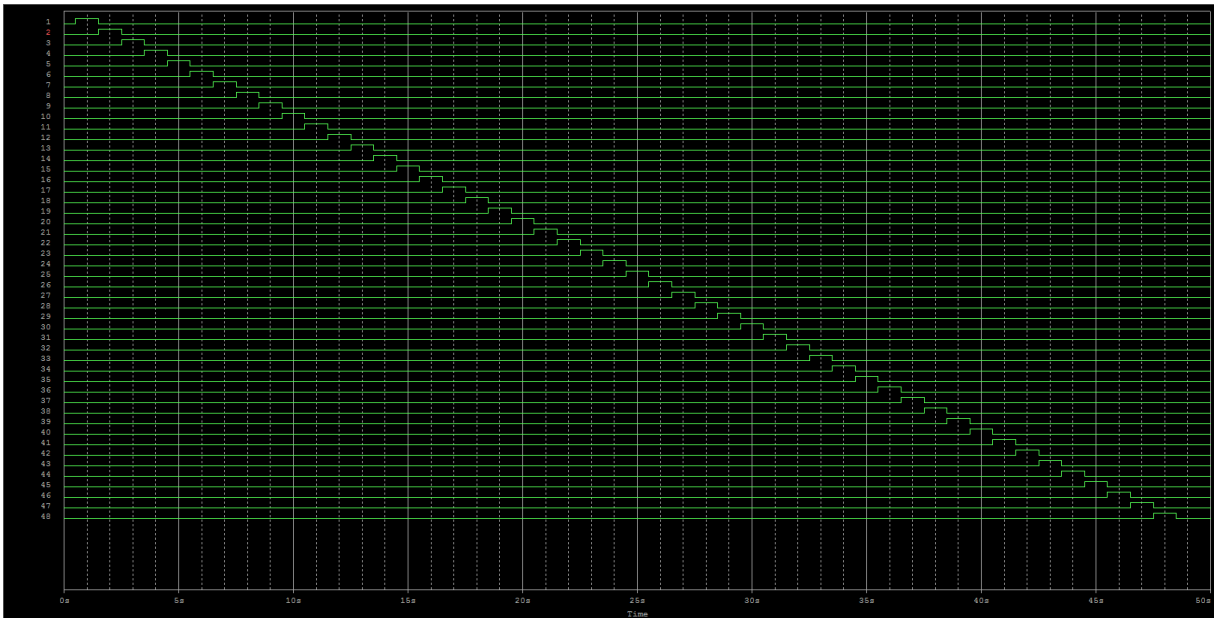

Figure 21 Simulation for 48 LED lights. This shows that each LED turns on and off without repeating for a distance of 25meters

The following tables are testing timings for each for the 3 modes. Different times are set for the pacer and a stop watch is used to determine the actual time for the swim pacer to run a lap. Several trials are made and the average is at the bottom.

Table 5 Mode 1: Timing

Set time: 10 sec	Set time: 25 sec	Set time: 40 sec
Actual time(sec)	**Actual time(sec)**	**Actual time(sec)**
10.0	25.1	40.0
10.1	25.1	40.1
9.9	25.0	40.0
10.0	25.1	40.1
10.2	24.9	40.0
10.0	25.0	40.0
Average: 10.03	Average: 25.03	Average: 40.03

Table 6 Mode 2: Timing

Set times: 7 sec, 13 sec	**Set times: 5 sec, 7 sec**	**Set times: 10 sec, 20 sec**
Actual total time(sec)	**Actual total time(sec)**	**Actual total time(sec)**
20.0	12.2	30.0
20.1	12.1	30.0
20.1	12.1	30.1
20.0	12.0	30.0
20.1	12.0	30.1
20.2	12.1	30.1
Average: 20.08	Average: 12.08	Average: 30.05

Table 7 Mode 3: Timing

Set times: 7 sec, 3 sec, 4 sec	**Set times: 7 sec, 7 sec, 12 sec**	**Set times: 14 sec, 14 sec,10 sec**
Actual total time(sec)	**Actual total time(sec)**	**Actual total time(sec)**
14.3	26.2	38.0
14.2	26.0	38.1
14.0	26.0	38.1
13.9	26.1	38.0
14.0	26.0	38.0
14.1	26.1	38.0
Average: 14.08	Average: 26.07	Average: 38.03

Appendix D Requirement and Verification Table

Table 8 System Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
1. LCD display information and interact with user inputs correctly a. Display main menu when the power is turned on and the mode it's in b. Print and retrieve data c. Transmit data to main microcontroller	1. LCD display information and interact with user inputs correctly a. Powered by the microcontroller, LCD will be tested by receiving a test data from the LCD microcontroller to confirm that it's displaying the information correctly b. A test data will be sent from the user input which can be displayed on the Arduino serial monitor to confirm accurate information is received c. A test data from the user input will be transmitted to the main microcontroller and these data will be displayed on the Arduino serial monitor to inspect the correctness of the data	Y
2. Main microcontroller computes data correctly a. Display the computed clock cycle time on the LCD b. Sum of each of the clock cycle time has to be added up to match the timing that the user selected	2. Main microcontroller computes data correctly a. A test timing is use in the clock calculation equation and displayed on the LCD to confirm that it's correct b. Since the pacer has different clock cycle times for each sections, the LCD will display the total clock cycle time for	Y

	each section and added up to verify that the total clock cycle time is equal to the timing that the user selected	
3. LED unit should accurately light up when it receives a high signal from the shift registers a. Turn on and off accordingly	3. LED unit should accurately light up when it receives a high signal from the shift registers a. The LED and the shift register will be tested on a breadboard to confirm that it can be turned on and off correctly from the signal of the shift registers	Y
4. The LED strip will have a minimum time of 10 seconds and maximum time of 40 seconds a. Microcontroller only allows users to select a minimum of 10 seconds and the LEDs will travel 25meters at the minimum time of 10 seconds b. Microcontroller only allows users to select a maximum speed of 40 seconds and the LEDs will travel 25 meters at the maximum of 40 seconds	4. The LED strip will have a minimum time of 10 seconds and maximum time of 40 seconds a. The value of the time will be displayed on the LCD, if it is less than 10 seconds, then the test fails. If it passes the test, the LED strip will be tested at a time of 10 seconds and the LEDs needs to run a lap in 10 seconds, accurate to the 100ms. If difference is more than 100ms, test will fail b. The value of the time will be displayed on the LCD, if it is more than 40 seconds, then the test fails. If it passes the test, the LED strip will be tested at a time of 40 seconds and the LEDs needs to run a lap in 40 seconds, accurate to the 100ms. If difference is more than 100ms, test will fail	Y
5. Power Supply a. Supply +5V to the microcontrollers	5. Power Supply a. The power supply will be tested with a	Y

		multimeter to ensure that it can power the microcontrollers and all six of the shift registers. It should measure 5v for a pass and if it's less than 2v or more than 7v, then the test will fail	

Appendix E Pictures

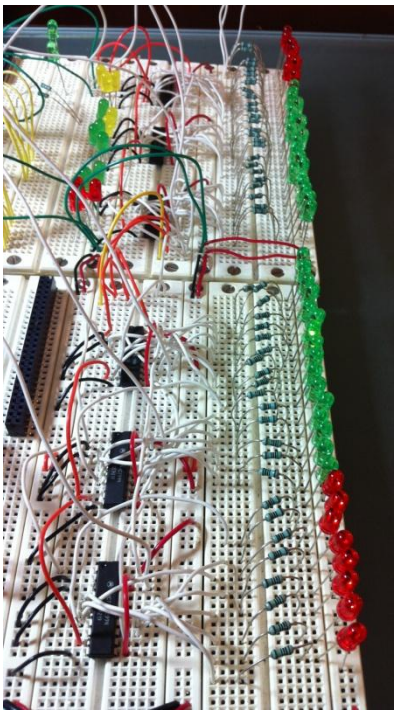

Figure 22 LED prototype

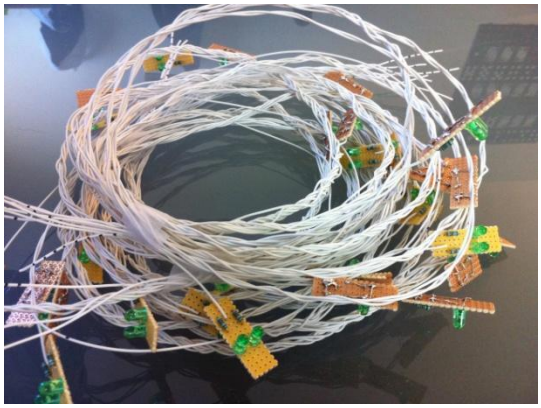

Figure 23 LED Strip

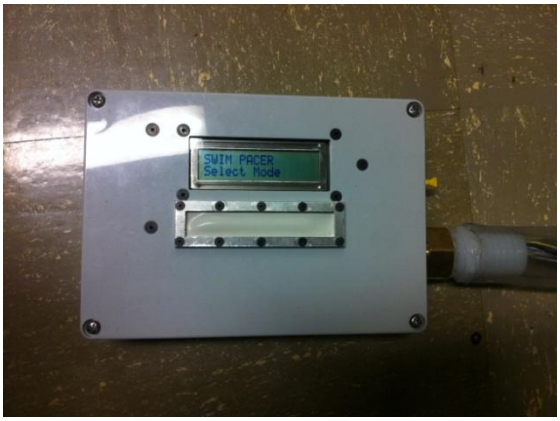

Figure 24 Control Box containing LCD, microcontroller and batteries

Figure 25 Final LED Swim Pacer

Appendix F Swim Pacer Code

```
// set pin numbers:
/*
Swim Pacer
The circuit:
* LCD RS pin to digital pin 13
* LCD Enable pin to digital pin 12
* LCD D4 pin to digital pin 11
* LCD D5 pin to digital pin 10
* LCD D6 pin to digital pin 9
* LCD D7 pin to digital pin 8
* LCD R/W pin to ground
* switch 1 to Analog 0
* switch 2 to Analog 1
* switch 3 to Analog 2
* switch 4 to Analog 3

 */

// include the library code:
#include <LiquidCrystal.h>
// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(13, 12, 11, 10, 9, 8);
const int analogPin0 = A0;
const int analogPin1 = A1;
const int analogPin2 = A2;
const int analogPin3 = A3;
const int threshold = 100;
int analog0 = 0;
int analog1 = 0;
int analog2 = 0;
int analog3 = 0;
int entered = false;
int entersec=false;
int entersec2a=false;
int entersec2b=false;
int entersec3a=false;
int entersec3b=false;
int entersec3c=false;
int enterstart= false;
int enterstop = false;
int check = 0; // check if the timing has a 0.5 in it
int check2 =0;
int mode = 0;
float sec = 25; // time
float sec2a = 12; // time
float sec2b = 12; // time
float sec2t = 0; // time total for mode 2
float sec3a = 8; // time
float sec3b = 8; // time
float sec3c = 8; // time
float sec3t = 0; // time total for mode 3
float secinc = 0.5; // time increment
float spd=0; // speed
int sectrans = 0; // speed that is transferred to main controller
const int led1 =  26;        // the number of first LED pin
const int select2 =  28;
const int led3 =  30;        // the number of third LED pin
const int r1 =  32;          // the number of first shift register pin
const int r2 =  34;          // the number of second shift register pin
const int r3 =  36;          // the number of third shift register pin
const int r4 =  38;          // the number of fourth shift register pin
```

```
const int r5 =  40;          // the number of fifth shift register pin
const int r6 =  42;          // the number of sixth shift register pin
const int select =  44;      // the number of the slect signal pin
const int slclk =  46;     // the number of the slow clock pin
const int startbit = 48;
const int led2 =  50;        // the number of second LED pin


int num=0;
int x=0;       //clock number
int time=250;
int time2=250;
int time2a;
int time2b;
int time3a;
int time3b;
int time3c;

void setup()
 {
 // set the digital pin as output:
 pinMode(slclk, OUTPUT);
 pinMode(select, OUTPUT);
 pinMode(select2, OUTPUT);
 pinMode(led1, OUTPUT);
 pinMode(led2, OUTPUT);
 pinMode(led3, OUTPUT);
 pinMode(r1, OUTPUT);
 pinMode(r2, OUTPUT);
 pinMode(r3, OUTPUT);
 pinMode(r4, OUTPUT);
 pinMode(r5, OUTPUT);
 pinMode(r6, OUTPUT);
 lcd.begin(16, 2);
 // Print a message to the LCD.
 lcd.print("SWIM PACER");
 pinMode(22, OUTPUT); // reset
 }



void loop()
{
 digitalWrite(22,HIGH );
 while (entered ==false) // when a mode is selected
 {
   entersec = false;
   entersec2a =false;
   entersec2b =false;
   entersec3a =false;
   entersec3b =false;
   entersec3c =false;
   enterstart = false;
   enterstop = false;
   lcd.setCursor(0, 1);
   lcd.print("Select Mode");
   // read the state of the pushbutton value:
   int analog0 = analogRead(analogPin0);
   int analog1 = analogRead(analogPin1);
   int analog2 = analogRead(analogPin2);
   int analog3 = analogRead(analogPin3);
   // check if the pushbutton is pressed.
   // if it is, the buttonState is HIGH:
   if (analog0 <threshold)
   {
     lcd.setCursor(0, 1);
```

```
    lcd.print("Mode 1                "); // display mode 1
    mode=1;
    entered=true;
   }
   if (analog1 <threshold)
   {
    lcd.setCursor(0, 1);
    lcd.print("Mode 2                "); // display mode 2
    mode=2;
    entered=true;
   }
   if (analog2 <threshold)
   {
    lcd.setCursor(0, 1);
    lcd.print("Mode 3                "); // display mode 3
    mode=3;
    entered=true;
   }
  }
// next level after mode is selected ***********************************************************************
  switch (mode)
  {
   case 1:
   delay(2000);
   lcd.clear();
   lcd.begin(16, 2);
   lcd.print("Mode 1                    ");
   lcd.setCursor(0, 1);
   lcd.print("Select time(sec)            ");
   delay(1000);
   while (entersec==false) //press 1 or 2 to increase or decrease speed
   {
   analog0 = analogRead(analogPin0);
   analog1 = analogRead(analogPin1);
   analog2 = analogRead(analogPin2);
   analog3 = analogRead(analogPin3);
   if (analog0 <threshold) //increase speed when button 1 is pushed
   {
    if (sec <= 39.5)
    {
     sec = sec + secinc;
    }
    else
    {
     sec=40;
    }
   delay(300);
   }
   if (analog1 <threshold) //decrease speed when button 1 is pushed
   {
    if (sec >= 10.5)
    {
     sec = sec - secinc;
    }
    else
    {
    sec=10;
    }
    delay(300);
   }
   if (analog2 <threshold) // enter is pressed
   {
    entersec= true;
   }
   if (analog3 <threshold) // exit is pressed
   {
```

```
    reset();
   }
   if (sec == 10) // set the minimum timing
   {
    lcd.setCursor(0, 1);
    lcd.print("Min is 10 sec                ");
   }
   else if (sec == 40) // set the maximum timing
   {
    lcd.setCursor(0, 1);
    lcd.print("Max is 40 sec                ");
    }
   else // display current timing
   {
    lcd.setCursor(0, 1);
    lcd.print(sec,1);
    lcd.print(" sec             ");
   }
  }
  break;
// next level after mode2 is selected ***********************************************************************
  case 2:
  delay(2000);
  lcd.clear();
  lcd.begin(16, 2);
  lcd.print("Mode 2                ");
  lcd.setCursor(0, 1);
  lcd.print("Select 1st time             ");
  delay(1000);
  while (entersec2a==false) //press 1 or 2 to increase or decrease speed
  {
   analog0 = analogRead(analogPin0);
   analog1 = analogRead(analogPin1);
   analog2 = analogRead(analogPin2);
   analog3 = analogRead(analogPin3);
   if (analog0 <threshold) //increase speed when button 1 is pushed
   {
    if (sec2a <= 19.5)
    {
     sec2a = sec2a + secinc;
    }
    else
    {
     sec2a=20;
    }
    delay(300);
   }
   if (analog1 <threshold) //decrease speed when button 1 is pushed
   {
    if (sec2a >= 5.5)
    {
     sec2a = sec2a - secinc;
    }
    else
    {
     sec2a=5;
    }
    delay(300);
   }
   if (analog2 <threshold) // enter is pressed
   {
    entersec2a= true;
   }
   if (analog3 <threshold)  // exit is pressed
   {
    reset();
```

```
 }
 if (sec2a == 5) // set the minimum timing
 {
  lcd.setCursor(0, 1);
  lcd.print("Min is 5 sec                ");
 }
 else if (sec2a == 20) // set the maximum timing
 {
  lcd.setCursor(0, 1);
  lcd.print("Max is 20 sec                ");
 }
 else // display current timing
 {
  lcd.setCursor(0, 1);
  lcd.print(sec2a,1);
  lcd.print(" sec              ");
 }
}
lcd.clear();
lcd.begin(16, 2);
lcd.print("Mode 2             ");
lcd.setCursor(0, 1);
lcd.print("Select 2nd time          ");
delay(1000);
while (entersec2b==false) //press 1 or 2 to increase or decrease speed
{
 analog0 = analogRead(analogPin0);
 analog1 = analogRead(analogPin1);
 analog2 = analogRead(analogPin2);
 analog3 = analogRead(analogPin3);
 if (analog0 <threshold) //increase speed when button 1 is pushed
 {
  if (sec2b <= 19.5)
  {
   sec2b = sec2b + secinc;
  }
  else
  {
   sec2b=20;
  }
 delay(300);
 }
 if (analog1 <threshold) //decrease speed when button 1 is pushed
 {
  if (sec2b >= 5.5)
  {
   sec2b = sec2b - secinc;
  }
  else
  {
   sec2b=5;
  }
  delay(300);
 }
 if (analog2 <threshold)
 { // enter is pressed
  entersec2b= true;
 }
 if (analog3 <threshold)
 { // exit is pressed
  reset();
 }
 if (sec2b == 5) // set the minimum timing
 {
  lcd.setCursor(0, 1);
  lcd.print("Min is 5 sec           ");
```

```
  }
  else if (sec2b == 20) // set the maximum timing
  {
   lcd.setCursor(0, 1);
   lcd.print("Max is 20 sec              ");
  }
  else // display current timing
  {
   lcd.setCursor(0, 1);
   lcd.print(sec2b,1);
   lcd.print(" sec             ");
  }
 }
 break;
// next level after mode3 is selected ********************************************************************
 case 3:
 delay(2000);
 lcd.clear();
 lcd.begin(16, 2);
 lcd.print("Mode 3            ");
 lcd.setCursor(0, 1);
 lcd.print("Select 1st time           ");
 delay(1000);
 while (entersec3a==false) //press 1 or 2 to increase or decrease speed
 {
  analog0 = analogRead(analogPin0);
  analog1 = analogRead(analogPin1);
  analog2 = analogRead(analogPin2);
  analog3 = analogRead(analogPin3);
  if (analog0 <threshold) //increase speed when button 1 is pushed
  {
   if (sec3a <= 13.5)
   {
    sec3a = sec3a + secinc;
   }
   else
   {
    sec3a=14;
   }
   delay(300);
  }
  if (analog1 <threshold) //decrease speed when button 1 is pushed
  {
   if (sec3a >= 3.5)
   {
    sec3a = sec3a - secinc;
   }
   else
   {
    sec2b=3;
   }
   delay(300);
  }
  if (analog2 <threshold) // enter is pressed
  {
   entersec3a= true;
  }
  if (analog3 <threshold) // exit is pressed
  {
   reset();
  }
  if (sec3a == 3) // set the minimum timing
  {
   lcd.setCursor(0, 1);
   lcd.print("Min is 3 sec             ");
  }
```

```
 else if (sec3a == 14) // set the maximum timing
 {
  lcd.setCursor(0, 1);
  lcd.print("Max is 14 sec             ");
 }
 else // display current timing
 {
  lcd.setCursor(0, 1);
  lcd.print(sec3a,1);
  lcd.print(" sec             ");
 }
}
lcd.clear();
lcd.begin(16, 2);
lcd.print("Mode 3            ");
lcd.setCursor(0, 1);
lcd.print("Select 2nd time           ");
delay(1000);
while (entersec3b==false) //press 1 or 2 to increase or decrease speed
 {
  analog0 = analogRead(analogPin0);
  analog1 = analogRead(analogPin1);
  analog2 = analogRead(analogPin2);
  analog3 = analogRead(analogPin3);
  if (analog0 <threshold) //increase speed when button 1 is pushed
  {
   if (sec3b <= 13.5)
   {
    sec3b = sec3b + secinc;
   }
   else
   {
    sec3b=14;
   }
   delay(300);
  }
  if (analog1 <threshold) //decrease speed when button 1 is pushed
  {
   if (sec3b >= 3.5)
   {
    sec3b = sec3b - secinc;
   }
   else
   {
    sec3b=3;
   }
   delay(300);
  }
  if (analog2 <threshold) // enter is pressed
  {
   entersec3b= true;
  }
  if (analog3 <threshold)  // exit is pressed
  {
   reset();
  }
  if (sec3b == 3) // set the minimum timing
  {
   lcd.setCursor(0, 1);
   lcd.print("Min is 3 sec             ");
  }
  else if (sec3b == 14) // set the maximum timing
  {
   lcd.setCursor(0, 1);
   lcd.print("Max is 14 sec             ");
  }
```

```
   else // display current timing
   {
    lcd.setCursor(0, 1);
    lcd.print(sec3b,1);
    lcd.print(" sec             ");
   }
}
lcd.clear();
lcd.begin(16, 2);
lcd.print("Mode 3            ");
lcd.setCursor(0, 1);
lcd.print("Select 3rd time            ");
delay(1000);
while (entersec3c==false) //press 1 or 2 to increase or decrease speed
{
 analog0 = analogRead(analogPin0);
 analog1 = analogRead(analogPin1);
 analog2 = analogRead(analogPin2);
 analog3 = analogRead(analogPin3);
 if (analog0 <threshold) //increase speed when button 1 is pushed
 {
  if (sec3c <= 13.5)
  {
   sec3c = sec3c + secinc;
  }
  else
  {
   sec3c=14;
  }
  delay(300);
 }
 if (analog1 <threshold) //decrease speed when button 1 is pushed
 {
  if (sec3c >= 3.5)
  {
   sec3c = sec3c - secinc;
  }
  else
  {
   sec3c=3;
  }
  delay(300);
 }
 if (analog2 <threshold) // enter is pressed
 {
  entersec3c= true;
 }
 if (analog3 <threshold) // exit is pressed
 {
  reset();
 }
 if (sec3c == 3) // set the minimum timing
 {
  lcd.setCursor(0, 1);
  lcd.print("Min is 3 sec             ");
 }
 else if (sec3c == 14) // set the maximum timing
 {
  lcd.setCursor(0, 1);
  lcd.print("Max is 14 sec             ");
 }
 else // display current timing
 {
  lcd.setCursor(0, 1);
  lcd.print(sec3c,1);
  lcd.print(" sec             ");
```

```
   }
  }
  break;
  }
// next level after timing is selected - show time**********************************************************************
  switch (mode)
  {
   case 1:
   lcd.setCursor(0, 1); //confirm the timing that user entered
   lcd.print("Entered ");
   lcd.print(sec,1);
   lcd.print("sec");
   delay(2000);
   break;
   case 2:
   lcd.setCursor(0, 0);
   lcd.print("Mode 2");
   lcd.setCursor(0, 1); //confirm the timing that user entered
   lcd.print("Entered ");
   lcd.print(sec2a,1);
   lcd.print("sec ");
   lcd.print(sec2b,1);
   lcd.print("sec");
   delay(1500);
   for (int positionCounter = 0; positionCounter < 11; positionCounter++)
   {
    // scroll one position left:
    lcd.setCursor(positionCounter, 0);
    lcd.print(" Mode 2");
    lcd.scrollDisplayLeft();
    // wait a bit:
    delay(150);
   }
   delay(1000);
   lcd.setCursor(0, 0);
   lcd.print("Mode 2");
   break;
   case 3:
   lcd.setCursor(0, 0);
   lcd.print("Mode 3");
   lcd.setCursor(0, 1); //confirm the timing that user entered
   lcd.print("Entered ");
   lcd.print(sec3a,1);
   lcd.print("sec ");
   lcd.print(sec3b,1);
   lcd.print("sec ");
   lcd.print(sec3c,1);
   lcd.print("sec");
   delay(1500);
   for (int positionCounter = 0; positionCounter < 20; positionCounter++)
   {
    // scroll one position left:
    lcd.setCursor(positionCounter, 0);
    lcd.print(" Mode 3");
    lcd.scrollDisplayLeft();
    // wait a bit:
    delay(150);
   }
   delay(1000);
   lcd.setCursor(0, 0);
   lcd.print("Mode 3");
   break;
  }
  // next level after timing is selected -- ask to start and show total
time**********************************************************************
  lcd.begin(16, 2);
```

```
while (enterstart==false) //checks if the user began the swim pacer
{
 lcd.setCursor(0, 0);
 lcd.print("Press 3 to Start");
 switch (mode)
 {
  case 1:
  lcd.setCursor(0, 1); //confirm the timing that user entered
  lcd.print("Total time:");
  lcd.print(sec,1);
  lcd.print("s");
  break;
  case 2:
  lcd.setCursor(0, 1); //confirm the timing that user entered
  lcd.print("Total time:");
  sec2t = sec2a + sec2b;
  lcd.print(sec2t,1);
  lcd.print("s");
  break;
  case 3:
  lcd.setCursor(0, 1); //confirm the timing that user entered
  lcd.print("Total time:");
  sec3t = sec3a + sec3b +sec3c;
  lcd.print(sec3t,1);
  lcd.print("s");
  break;
 }
 analog0 = analogRead(analogPin0);
 analog1 = analogRead(analogPin1);
 analog2 = analogRead(analogPin2);
 analog3 = analogRead(analogPin3);
 if (analog2 <threshold)
 {
  switch (mode)
  {
   case 1:
   spd =25/sec;
   break;
   case 2:
   spd =25/sec2t;
   break;
   case 3:
   spd =25/sec3t;
   break;
  }
  enterstart = true;
 }
 if (analog3 <threshold) // exit is pressed
 {
  reset();
 }
}
switch (mode)
{
 case 1:
 lcd.begin(16, 2);
 lcd.print("Start Pacer");
 lcd.setCursor(0, 1);
 lcd.print(spd);
 lcd.print("m/s ");
 lcd.print(sec,1);
 lcd.print("s");
 while (enterstop==false) //swim pacer begins, display the speed and time
 {
  analog3 = analogRead(analogPin3);
  if (analog3 <threshold)
```

```
    {
      reset();
    }
    else
      pacer();
  }
  break;
  case 2:
  lcd.begin(16, 2);
  lcd.print("Start Pacer");
  lcd.setCursor(0, 1);
  lcd.print(sec2a,1);
  lcd.print("s ");
  lcd.print(sec2b,1);
  lcd.print("s");
  while (enterstop==false) //swim pacer begins, display the speed and time
  {
    analog3 = analogRead(analogPin3);
    if (analog3 <threshold)
    {
      reset();
    }
    else
      pacer2();
  }
  break;
  case 3:
  lcd.begin(16, 2);
  lcd.print("Start Pacer");
  lcd.setCursor(0, 1);
  lcd.print(sec3a,1);
  lcd.print("s ");
  lcd.print(sec3b,1);
  lcd.print("s ");
  lcd.print(sec3c,1);
  lcd.print("s ");
  while (enterstop==false) //swim pacer begins, display the speed and time
  {
    analog3 = analogRead(analogPin3);
    if (analog3 <threshold)
    {
      reset();
    }
    else
    pacer3();
  }
  break;
 }
 }
////**********************************************************************End
loop**********************************************************************//////
////**********************************************************************End
loop**********************************************************************//////
////**********************************************************************End
loop**********************************************************************//////
////**********************************************************************End
loop**********************************************************************//////

void pacer()
{
 int x = 1;
 int y = 1;
 time2= ((4/(25/sec))/2)*1000/8/2;
 time= (sec-((4/(25/sec))/2))*1000/42/2;
 for(x = 1; x < 9; x++)
 {
```

```
 digitalWrite(select, HIGH);  //select right shift mode
 digitalWrite(select2, LOW);
 resetcheck();
 if (x==1)
   digitalWrite(led1, HIGH);
 else
   digitalWrite(led1, LOW);
 if (x==2)
   digitalWrite(led2, HIGH);
 else
   digitalWrite(led2, LOW);
 if (x==3)
   digitalWrite(led3, HIGH);
 else
   digitalWrite(led3, LOW);
 if (x==4)
   digitalWrite(r1, HIGH);
 else
   digitalWrite(r1, LOW);

 digitalWrite(slclk, HIGH);   // set the clock high
 delay(time2);            // wait for half second
 digitalWrite(slclk, LOW);    // set the clock low
 delay(time2);            // wait for half second
}
 for(x = 9; x < 52; x++)
{
 digitalWrite(select, HIGH);  //select right shift mode
 digitalWrite(select2, LOW);
 resetcheck();
 if (x==12)
   digitalWrite(r2, HIGH);
 else
   digitalWrite(r2, LOW);
 if (x==20)
   digitalWrite(r3, HIGH);
 else
   digitalWrite(r3, LOW);
 if (x==28)
   digitalWrite(r4, HIGH);
 else
   digitalWrite(r4, LOW);
 if (x==36)
   digitalWrite(r5, HIGH);
 else
   digitalWrite(r5, LOW);
 if (x==44)
   digitalWrite(r6, HIGH);
 else
   digitalWrite(r6, LOW);

 digitalWrite(slclk, HIGH);   // set the clock high
 delay(time);            // wait for half second
 digitalWrite(slclk, LOW);    // set the clock low
 delay(time);            // wait for half second
}
x=1;
digitalWrite(select, LOW);   //select left shift mode
digitalWrite(select2, HIGH);

for(y = 2; y < 9; y++)
{
  resetcheck();
 digitalWrite(slclk, HIGH);
 delay(time2);
 digitalWrite(slclk, LOW);
```

```
  delay(time2);
 }
 for(y = 9; y < 52; y++)
 {
  resetcheck();
  if (y==9)
   digitalWrite(r5, HIGH);
  else
   digitalWrite(r5, LOW);
  if (y==17)
   digitalWrite(r4, HIGH);
  else
   digitalWrite(r4, LOW);
  if (y==25)
   digitalWrite(r3, HIGH);
  else
   digitalWrite(r3, LOW);
  if (y==33)
   digitalWrite(r2, HIGH);
  else
   digitalWrite(r2, LOW);
  if (y==41)
   digitalWrite(r1, HIGH);
  else
   digitalWrite(r1, LOW);
  if (y==49)
   digitalWrite(led3, HIGH);
  else
   digitalWrite(led3, LOW);
  if (y==50)
   digitalWrite(led2, HIGH);
  else
   digitalWrite(led2, LOW);

  if (y==51)
   digitalWrite(led1, HIGH);
  else
   digitalWrite(led1, LOW);
  if (y==51)
  {
   digitalWrite(slclk, HIGH);
   delay(10);
   digitalWrite(slclk, LOW);
   delay(10);
   x=1;
  }
  else
  {
   digitalWrite(slclk, HIGH);
   delay(time);
   digitalWrite(slclk, LOW);
   delay(time);
   x=1;
  }
 }
}
////////////////////////////////////////////////////////////////////////**************************************************************************

void pacer2()
{
 int x = 1;
 int y = 1;
 time2= ((4/(12.5/sec2a))/2)*1000/8/2;
 time2a= (sec2a-((4/(12.5/sec2a))/2))*1000/17/2;
 time2b = sec2b *1000 /25/2;
 digitalWrite(select, HIGH);  //select right shift mode
```

```
digitalWrite(select2, LOW);
for(x = 1; x < 9; x++)
{
 resetcheck();
 if (x==1)
   digitalWrite(led1, HIGH);
 else
   digitalWrite(led1, LOW);
 if (x==2)
   digitalWrite(led2, HIGH);
 else
   digitalWrite(led2, LOW);
 if (x==3)
   digitalWrite(led3, HIGH);
 else
   digitalWrite(led3, LOW);
 if (x==4)
   digitalWrite(r1, HIGH);
 else
   digitalWrite(r1, LOW);

 digitalWrite(slclk, HIGH);   // set the clock high
 delay(time2);            // wait for half second
 digitalWrite(slclk, LOW);    // set the clock low
 delay(time2);            // wait for half second
}
for(x = 9; x < 26; x++)
{
 resetcheck();
 digitalWrite(select, HIGH);  //select right shift mode
 digitalWrite(select2, LOW);
 resetcheck();
 if (x==12)
   digitalWrite(r2, HIGH);
 else
   digitalWrite(r2, LOW);
 if (x==20)
   digitalWrite(r3, HIGH);
 else
   digitalWrite(r3, LOW);
 digitalWrite(slclk, HIGH);   // set the clock high
 delay(time2a);           // wait for half second
 digitalWrite(slclk, LOW);    // set the clock low
 delay(time2a);           // wait for half second
}
for(x = 26; x < 52; x++)
{
 resetcheck();
 if (x==20)
   digitalWrite(r3, HIGH);
 else
   digitalWrite(r3, LOW);
 if (x==28)
   digitalWrite(r4, HIGH);
 else
   digitalWrite(r4, LOW);
 if (x==36)
   digitalWrite(r5, HIGH);
 else
   digitalWrite(r5, LOW);
 if (x==44)
   digitalWrite(r6, HIGH);
 else
   digitalWrite(r6, LOW);
 digitalWrite(slclk, HIGH);   // set the clock high
 delay(time2b);           // wait for half second
```

```
 digitalWrite(slclk, LOW);    // set the clock low
 delay(time2b);              // wait for half second
}
x=1;
digitalWrite(select, LOW);   //select left shift mode
digitalWrite(select2, HIGH);
for(y = 2; y < 9; y++)
{
  resetcheck();
 digitalWrite(slclk, HIGH);
 delay(time2);
 digitalWrite(slclk, LOW);
 delay(time2);
}
for(y = 9; y < 26; y++)
{
 resetcheck();
 if (y==9)
   digitalWrite(r5, HIGH);
 else
   digitalWrite(r5, LOW);
 if (y==17)
   digitalWrite(r4, HIGH);
 else
   digitalWrite(r4, LOW);
 if (y==25)
   digitalWrite(r3, HIGH);
else
   digitalWrite(r3, LOW);

digitalWrite(slclk, HIGH);
delay(time2a);
digitalWrite(slclk, LOW);
delay(time2a);
x=1;
}
for(y = 26; y < 52; y++)
{
 resetcheck();
 if (y==25)
   digitalWrite(r3, HIGH);
else
   digitalWrite(r3, LOW);
 if (y==33)
   digitalWrite(r2, HIGH);
 else
   digitalWrite(r2, LOW);
 if (y==41)
   digitalWrite(r1, HIGH);
 else
   digitalWrite(r1, LOW);
 if (y==49)
   digitalWrite(led3, HIGH);
 else
   digitalWrite(led3, LOW);
 if (y==50)
   digitalWrite(led2, HIGH);
 else
  digitalWrite(led2, LOW);
 if (y==51)
   digitalWrite(led1, HIGH);
 else
   digitalWrite(led1, LOW);
 if (y==51)
 {
   digitalWrite(slclk, HIGH);
```

```
    delay(10);
    digitalWrite(slclk, LOW);
    delay(10);
   }
   else
   {
    digitalWrite(slclk, HIGH);
    delay(time2b);
    digitalWrite(slclk, LOW);
    delay(time2b);
   }
 }
}

////////////////////////////////////////////////////////////////////**************************************************************************
void pacer3()
{
 int x = 1;
 int y = 1;
  time2= ((4/((25/3)/sec3a))/2)*1000/8/2;
  time3a= (sec3a-((4/((25/3)/sec3a))/2))*1000/9/2;
  time3b = sec3b *1000 /17/2;
  time3c = sec3c*1000/16/2;
  digitalWrite(select, HIGH);  //select right shift mode
  digitalWrite(select2, LOW);
  for(x = 1; x < 9; x++)
  {
   resetcheck();
   if (x==1)
    digitalWrite(led1, HIGH);
   else
    digitalWrite(led1, LOW);
   if (x==2)
    digitalWrite(led2, HIGH);
   else
    digitalWrite(led2, LOW);
   if (x==3)
    digitalWrite(led3, HIGH);
   else
    digitalWrite(led3, LOW);
   if (x==4)
    digitalWrite(r1, HIGH);
   else
    digitalWrite(r1, LOW);
   digitalWrite(slclk, HIGH);  // set the clock high
   delay(time2);            // wait for half second
   digitalWrite(slclk, LOW);   // set the clock low
   delay(time2);            // wait for half second
  }
  for(x = 9; x < 18; x++)
  {
   resetcheck();
   if (x==4)
    digitalWrite(r1, HIGH);
   else
    digitalWrite(r1, LOW);
   if (x==12)
    digitalWrite(r2, HIGH);
   else
    digitalWrite(r2, LOW);
   digitalWrite(slclk, HIGH);  // set the clock high
   delay(time3a);            // wait for half second
   digitalWrite(slclk, LOW);   // set the clock low
   delay(time3a);            // wait for half second
  }
  for(x = 18; x < 35; x++)
```

```
{
  resetcheck();
  if (x==20)
    digitalWrite(r3, HIGH);
  else
    digitalWrite(r3, LOW);
  if (x==28)
    digitalWrite(r4, HIGH);
  else
    digitalWrite(r4, LOW);
  digitalWrite(slclk, HIGH);   // set the clock high
  delay(time3b);            // wait for half second
  digitalWrite(slclk, LOW);    // set the clock low
  delay(time3b);            // wait for half second
}
for(x = 35; x < 52; x++)
{
  resetcheck();
  if (x==36)
    digitalWrite(r5, HIGH);
  else
    digitalWrite(r5, LOW);
  if (x==44)
    digitalWrite(r6, HIGH);
  else
    digitalWrite(r6, LOW);
  digitalWrite(slclk, HIGH);   // set the clock high
  delay(time3c);            // wait for half second
  digitalWrite(slclk, LOW);    // set the clock low
  delay(time3c);            // wait for half second
}
x=1;
digitalWrite(select, LOW);   //select left shift mode
digitalWrite(select2, HIGH);
for(y = 2; y < 9; y++)
{
   resetcheck();
  digitalWrite(slclk, HIGH);
  delay(time2);
  digitalWrite(slclk, LOW);
  delay(time2);
}
for(y = 9; y < 18; y++)
{
  resetcheck();
  if (y==9)
    digitalWrite(r5, HIGH);
  else
    digitalWrite(r5, LOW);
  if (y==17)
    digitalWrite(r4, HIGH);
  else
    digitalWrite(r4, LOW);
  digitalWrite(slclk, HIGH);
  delay(time3a);
  digitalWrite(slclk, LOW);
  delay(time3a);
}
for(y = 18; y < 35; y++)
{
  resetcheck();
  if (y==17)
    digitalWrite(r4, HIGH);
  else
    digitalWrite(r4, LOW);
  if (y==25)
```

```
    digitalWrite(r3, HIGH);
   else
    digitalWrite(r3, LOW);
   if (y==33)
    digitalWrite(r2, HIGH);
   else
    digitalWrite(r2, LOW);
   digitalWrite(slclk, HIGH);
   delay(time3b);
   digitalWrite(slclk, LOW);
   delay(time3b);
  }
  for(y = 35; y < 52; y++)
  {
   resetcheck();
   if (y==41)
    digitalWrite(r1, HIGH);
   else
    digitalWrite(r1, LOW);
   if (y==49)
    digitalWrite(led3, HIGH);
   else
    digitalWrite(led3, LOW);
   if (y==50)
    digitalWrite(led2, HIGH);
   else
    digitalWrite(led2, LOW);
   if (y==51)
    digitalWrite(led1, HIGH);
   else
    digitalWrite(led1, LOW);
   if (y==51)
   {
    digitalWrite(slclk, HIGH);
    delay(10);
    digitalWrite(slclk, LOW);
    delay(10);
   }
   else
   {
    digitalWrite(slclk, HIGH);
    delay(time3c);
    digitalWrite(slclk, LOW);
    delay(time3c);
   }
  }
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

void resetcheck()
{
  analog3 = analogRead(analogPin3);
  if (analog3 <threshold)
  {
   digitalWrite(22,LOW );
   digitalWrite(led1, LOW);
   digitalWrite(led2, LOW);
   digitalWrite(led3, LOW);
   digitalWrite(r1, LOW);
   digitalWrite(r2, LOW);
   digitalWrite(r3, LOW);
   digitalWrite(r4, LOW);
   digitalWrite(r5, LOW);
   digitalWrite(r6, LOW);
   asm volatile (" jmp 0");
```

```
   delay(300);
   reset();
  }
}

void reset()
{
  digitalWrite(22,LOW );
  asm volatile (" jmp 0");
  delay(300);
}

void modeReset()
{
  entered = false;
  entersec=false;
  entersec2a=false;
  entersec2b=false;
  entersec3a=false;
  entersec3b=false;
  entersec3c=false;
  enterstart= false;
  enterstop = false;
}
```