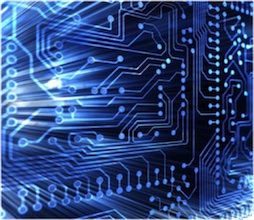
**Engineering Report**

****

**Course: Engineering**

**Code: Mr. C. D’Arcy**

**Author: Jiashi Zhu**

**Date: June 8, 2012**

**Table of Contents**

Activity 1. Build a Gamer PC 6

Purpose 6

Parts 6

Media 6

Activity 2. LED Patterns 8

Purpose 8

Media 8

Procedure 8

Parts 9

Code 10

Activity 3. Intersection Simulator 1. 12

Purpose 12

(Part 1) 12

Media 12

Parts 13

Code 13

(Part 2) 15

Media 15

Procedure 16

Parts 16

Activity 4. Intersection Simulator 2 17

Purpose 17

(Part 1) 17

Media 17

Procedure 17

Parts 18

Layout 19

Code: 19

(Part 2) 22

Media 22

Procedure 23

Activity 5. Problems 1.14~1.18 24

1.14 24

1.15 24

1.16 25

1.17 26

1.18 26

Activity 6. Altoids Speaker Project 28

Purpose 28

Parts of Circuit Board 28

BreadBoard-Prototype 29

Final Assembly 30

Activity 7. Altoids USB Charger Project 33

Purpose 33

Media 33

Parts 34

Procedure 34

2011 Exam: Conway’s Game of Life 36

Purpose: 36

Media 36

Parts 37

Procedure 37

TEI4M. Arduino 39

Project 1. A BiColor LED Flasher 39

Purpose 39

Reference 39

Method 39

Sketch 40

Media 41

Project 2. High Brightness Luxeon LED 42

Purpose 42

Reference 42

Method 42

Sketch 43

Media 44

Project 3. S.A.D. Light 45

Purpose 45

Reference 45

Parts List 45

Procedure 46

Sketch 48

Media 49

Project 4: Alphanumeric (16-Segment) Display 50

Purpose 50

Reference 50

Pin/Segment Map 50

Sketch 51

Media 53

Custom Sketch 53

Appendix C. Monthly Projects 55

October 2011: XBee Wireless Solution 55

Reference 55

Purpose 55

Parts List 55

Description 55

Summary 56

Project 5. Thermal Management 57

Part 1. NTC Thermistor 57

Reference 57

Purpose 57

Principle 57

Parts List 57

Media 58

Part 2. Serial Communications Using Processing 59

Reference 59

Purpose 59

Parts List 59

Procedure 59

Media 60

Sketch 60

Part 3. Precision Centigrade Sensor (LM35DZ) 63

Reference 63

Purpose 63

Procedure 63

Parts List 64

Sketch 64

Media 67

Part 4. Multiple Sensors: Polling 68

Reference 68

Purpose 68

Procedure 68

Parts List 68

Sketch 68

Media 73

Part 5. Multiple Sensors: Multiplexing 74

Purpose 74

Procedure 74

Sketch 74

Media 76

5V/12V Breadboard-Compatible Regulator 77

Reference 77

Purpose 77

Parts List 77

Procedure 77

Media 79

Part 6: Air Flow Management 80

Reference 80

Purpose 80

Parts List 80

Procedure 80

Sketch 81

Media 82

Part 7. Fan Speed 83

Reference 83

Purpose 83

Parts List 83

Procedure 83

Sketch 84

Media 85

Part 8. ATtiny85 Port 87

Reference 87

Purpose 87

Parts List 87

Procedure 87

Sketches 88

Media 89

Project 6. The Tune Player 91

Reference 91

Purpose 91

Parts List 91

Procedure 91

Sketches 92

Media 94

2012 Final Exam: Servo-Controlled Laser 97

Reference 97

Purpose 97

Parts List 97

Procedure 97

Sketches 98

Media 100

**2010-2011 TEI3M: Computer Engineering**

# Activity 1. Build a Gamer PC

### Purpose

The purpose of this activity was to build a personal computer that specializes in graphic and gaming performance.

### Parts

|  |  |
| --- | --- |
| **COMPONENT** | **DETAILS** |
| CPU | [Intel Core i5 650 Processor BX80616I5650 - 3.20GHz, LGA 1156, 4MB L3 Cache](http://www.tigerdirect.ca/applications/searchtools/item-details.asp?EdpNo=5581941&csid=ITD&body=MAIN) |
| Motherboard | [ASUS P7P55D-E LX Motherboard - Intel P55, Socket LGA1156, USB, DDR3, PCIe, LAN](http://www.tigerdirect.ca/applications/SearchTools/item-details.asp?EdpNo=15121&CatId=4725) |
| RAM | [Corsair PC12800 RAM - 4GB, DDR3, 1600MHz](http://www.tigerdirect.ca/applications/searchtools/item-Details.asp?EdpNo=5082127&sku=C13-8250&srkey=C13-8250) |
| Graphics card | [EVGA 01G-P3-1452-TR GeForce GTS 450 SuperClocked Video Card](http://www.tigerdirect.ca/applications/searchtools/item-Details.asp?EdpNo=6655890&sku=E145-0450&srkey=E145-0450) |
| Hard Drive | [Western Digital WD1002FAEX Caviar Black Hard Drive - 1TB, 7200RPM, 64MB, SATA 6Gbs](http://www.tigerdirect.ca/applications/searchtools/item-Details.asp?EdpNo=6076939&sku=TSD-1000FAEX&srkey=TSD-1000FAE%20SY) |
| Power supply | [Corsair TX650W Power Supply](http://www.tigerdirect.ca/applications/searchtools/item-Details.asp?EdpNo=3438607&sku=C13-2502&srkey=C13-2502) |
| DVD Writer | [LG Super Multi DVD Rewiter](http://www.tigerdirect.ca/applications/SearchTools/item-details.asp?EdpNo=6613479&CatId=1624) |
| Case | [Cooler Master HAF 922M](http://www.tigerdirect.ca/applications/searchtools/item-Details.asp?EdpNo=4631382&sku=C283-1223&srkey=C283-1223) |
| Mouse | [Microsoft Mouse](http://www.tigerdirect.ca/applications/searchtools/item-Details.asp?EdpNo=4171790&sku=M17-1799&srkey=M17-1799) |
| Keyboard | [Microsoft Keyboard](http://www.tigerdirect.ca/applications/searchtools/item-Details.asp?EdpNo=4171793&sku=M17-1871&srkey=M17-1871) |
| Speakers | [Logitech X540](http://www.tigerdirect.ca/applications/searchtools/item-Details.asp?EdpNo=2507686&sku=L23-7250&srkey=L23-7250) |
| O/S | [Microsoft Windows 7 Professional 64BIT OS](http://www.tigerdirect.ca/applications/searchtools/item-Details.asp?EdpNo=5213934&sku=M17-7306&srkey=M17-7306) |
| Warranty | 1 yr |

Media

Overview:

The following graph shows a detailed look of the motherboard and the components installed on it.



Procedure

ASUS Motherboard

GeForce GTS 450

Graphic Card

Core I5 CPU (with the CPU fan)

4GB DDR3 RAM x2

1. Preparation:

a. Get to know basic knowledge about building a personal computer and each required component.

1. Search information and computer components retailers and parts. Considering price, efficiency and performance, provide several doable combinations. (link [here](http://darcy.rsgc.on.ca/ACES/TEI3M/RecommendedComputerComponents.xls))
2. Decide the final configuration. Purchase parts from retailer store.
3. Build up
4. Motherboard: Place the motherboard into the case and fixate properly with screws.
5. RAM & graphic card: Place into the grooves on the motherboard.
6. CPU: remove the protector on the motherboard, place the CPU gently onto the motherboard, close the protector, place the CPU fan over the protector and fixate.
7. Hard drive, DVD burner & Power supply installation.
8. Wire connection and arrangement: Following the instruction menu comes with the motherboard, connect wires of power supply, fans and hard drives to the motherboard. Clean up wires and place them backwards of the case to save more space and reduce heating problems.
9. Sticker placement etc.

3. Operating system installation, performance testing.

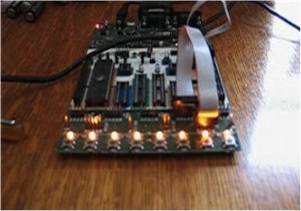
# Activity 2. LED Patterns

### Purpose

The purpose of this LED Pattern project is to program the AVR-tiny24 chip so that it can turn the 8 LED lights on the board on and off in certain patterns.

(Sample videos at [here](http://darcy.rsgc.on.ca/ACES/TEI3M/CircuitTasks/1011Tasks.htm#LEDPatterns), LED1~LED6)

### Media



The above picture shows the last second of pattern 1, when the value of output is 0x00, which means all the lights will be turned on.

### Procedure

The programming work is based on the AVR studio. 

The following program on next page uses regisiter 16 and 17 for data storage, and regisiter 20~22 for delay method.

The program is designed to set all 8 ports of Port A on the chip for output use. For each pattern, different numbers in 16 hex is applied to Port A and change in certain orders so that those lights on the board displays the same. There is a one second delay between each two patterns.

(For the lights, 1 in binary means off and 0 means on.)

Pattern 1: the default value of Port A is set to 0xFF(11111111, all off), and decrease by 1 until the value reaches 0x00(all on).

Pattern 2: the default value is set to 0x55(01010101), and then the exchange of 1 and 0 is processed.

Pattern 3: the default value is set to 0xFE(1111110), then rotates the number to left by 1 each time.

Pattern 4: the opposite of Pattern 3, default value is 0xFF(11111111), then rotates the number to right by 1 each time.

Pattern 5: alike to Pattern 3, but instead default value is set to 0xFC(11111100)

Pattern 6: alike to Pattern 2, but instead default value is set to 0xF0(11110000).

### Parts

1. ATtiny24
2. STK500

Code

;Purpose: This project is used to demonstrate 6 specific patterns.

;Authors: Reuben Sagman & Jiashi Zhu

;Date: 10.13.2010

;=============================================================

.include "tn24def.inc"

;=============================================================

.def temp = R16

.def data = R17

.def count = R18

.org 0x00

rjmp Reset

Reset: ;Reset

ldi data,0xFF ;set data to 11111111

out DDRA,data ;apply all the PORTA for output

Pat1: ;pattern 1 starts from here

out PORTA,data ;ouput 11111111

rcall Delay004s ;delay for 0.4 s

dec data ;decrease data by 1

brne Pat1 ;run Pattern1 again until data=0

rcall delay1s ;delay for 1 s

Pat2: ;pattern 2 starts from here

ldi count,10 ;set count to 10

ldi data,0x55 ;set data to 01010101

Loop:

out PORTA,data ;output 01010101

com data ;complements data(10101010)

rcall Delay06s ;delay for 0.6 s

dec count ;decrease count by 1

brne Loop ;run Pattern2 until count=0

rcall Delay1s ;delay for 1s

Pat3: ;Pattern 3 starts from here

ldi data,0xFE ;set data to 11111110

ldi count,44 ;set count to 44

Loop1:

out PORTA,data ;output data

rol data ;rotate data to left by 1

rcall Delay02s ;delay for 0.2s

dec count ;decrease count by 1

brne Loop1 ;run Pattern3 until count=0

rcall Delay1s ;delay for 1s

Pat4: ;Pattern 4 starts from here

ser data ;set data to 11111111

ldi count,45 ;set count to 45

Loop2:

out PORTA,data ;output data

ror data ;rotate data to right by 1

rcall Delay02s ;delay for 0.2s

dec count ;decrease count by 1

brne Loop2 ;run Pattern4 until count=0

rcall Delay1s ;delay for 1s

Pat5: ;Pattern 5 starts from here

sec ;set C to 1

ldi data,0xFC ;set data to 11111100

ldi count,43 ;set count to 43

Loop3:

out PORTA,data ;output data

rol data ;rotate data to left by 1

rcall Delay02s ;delay for 0.2s

dec count ;decrease count by 1

brne Loop3 ;run Pattern5 until count=0

rcall Delay1s ;delay for 1s

Pat6: ;Pattern 6 starts from here.

ldi count,10 ;set count to 10

ldi data,0xF0 ;set data to 11110000

Loop4:

out PORTA,data ;output data

com data ;complements data

rcall Delay06s ;delay for 0.6s

dec count ;decrease count by 1

brne Loop4 ;run Pattern6 until count=0

rcall Delay1s ;delay for 1s

rcall Reset ; jump back to Reset

;------- D E L A Y R O U T I N E --------------------------

Delay004s:

; =============================

; delay loop generator

; 40000 cycles:

; -----------------------------

; delaying 39999 cycles:

ldi R20, $43

WGLOOP0: ldi R21, $C6

WGLOOP1: dec R21

brne WGLOOP1

dec R20

brne WGLOOP0

; -----------------------------

; delaying 1 cycle:

nop

; =============================

ret

Delay02s:

rcall Delay004s

rcall Delay004s

rcall Delay004s

rcall Delay004s

rcall Delay004s

;=====================================

ret

Delay06s:

rcall Delay02s

rcall Delay02s

rcall Delay02s

;=====================================

ret

Delay1s:

rcall Delay02s

rcall Delay02s

rcall Delay02s

rcall Delay02s

rcall Delay02s

;=====================================

ret

Delay3s:

rcall Delay1s

rcall Delay1s

rcall Delay1s

;======================================

ret

# Activity 3. Intersection Simulator 1.

### Purpose

The purpose of this project is to build a 12-LED device that simulates the situation of an intersection. Four groups of three LED lights (green+red+yellow) are used to display the patterns of traffic lights at an intersection.

### (Part 1)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **State** | **R1** | **A1** | **G1** | **R2** | **A2** | **G2** | **Duration (s)** |
| **1** | 0 | 1 | 1 | 1 | 1 | 0 | 5 |
| **2** | 0 | 1 | 1 | 1 | 0 | 1 | 2 |
| **3** | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| **4** | 1 | 1 | 0 | 0 | 1 | 1 | 5 |
| **5** | 1 | 0 | 1 | 0 | 1 | 1 | 2 |
| **6** | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| **Return to Pattern 1 and so on...** | | | | | | | |

### image7.jpegMedia

Power line \_

Lines that transfer the data in PORTA to the LEDs

These LEDs are used to replace the lights on the board

Procedure

The 12 LEDs should be turned on and off in the following patterns. (Only 6 are shown below for all the LEDs are divided into 2 groups.)

1. A look up table (LUT) is set up to store the data that used to control LEDs. In each row of the table, the low value shows the on/off status of the LEDs, and the higher value is set to decide how long each pattern should be displayed.
2. Because technically only 6 of the LEDs are needed to be modified, the last 2 bits of PORTA are settled off.
3. The value Z is programmed as a pointer to read data from the LUT. When it’s pointed at a low value, the system will pass the value to PORTA; when it’s pointed at a high value, the system will jump to the Hold label and call Delay1s certain times according to the high value.
4. When value Z is detected to reach the end of the LUT, the system will reset its position back to the beginning of the LUT and process the previous steps again.

### Parts

1. ATtiny24.
2. STK500
3. Green, yellow, and red LEDs, 4 for each colour.
4. Power line, several wires
5. Optional: resistors (depends on how the LEDs are arranged.)

### Code

;-------------------------------------------------------

; PROJECT: Lookup Table Template (LUT)

; PURPOSE: To provide a (somewhat) standard framework for

; implementing all-important lookup tables

; AUTHOR: Reuben Sagman and Jiashi Zhu

; DATE: 10 10 29

;-------------------------------------------------------

.include "tn24def.inc"

;-------------------------------------------------------

;---- D E F I N E S ------------------------------------

.def temp = R16

.def duration= R17

;-----V E C T O R J U M P T A B L E ----------------

.org 0x00

rjmp Reset

LUT:

.db 0b01111011,0x05 ;The Look Up Table starts from here

.db 0b01110111,0x02 ;This table gives the situations of R+G, R+A,R+R

.db 0b01101111,0x01 ;The last two bits of each low value is ignored

.db 0b11001111,0x05 ;Low value stands for the off/on situation

.db 0b10101111,0x02 ;High value stands for the duration time

.db 0b01101111,0x01 ;

LUTEND:

;--------------------------------------------------------

Reset:

ldi temp, low(RAMEND) ; Point the low byte of the Stack Pointer (SPL)

out SPL, temp ; to the end of SRAM

SetUp:

ldi temp,0b11111100 ;Take 6 ports from PORTA for output

out DDRA,temp ;Set up DDRA

ldi XH,high(LUT<<1) ;Load the look up table

ldi XL,low(LUT<<1) ;Continue loading...

ldi YH,high(LUTEND<<1) ;Continue loading...

ldi YL,low(LUTEND<<1) ;Complete

Repeat:

movw Z,X ;Position the pointer to the first byte of the LUT

Display:

lpm temp,Z ;Fetch the next byte from the lookup table....

out PORTA,temp ;Output it to PORTA

inc ZL ;Move Z to the next value

lpm duration,Z ;pass the value of duration time to Z

rcall Hold ; Delay to admire!

inc ZL ; increment pointer...

cp ZL,YL ; is the pointer at LUTEND?

brne Display ; if not go back to Display

rjmp Repeat

;-------------------------------------------------------

Delay1s:

; =============================

; delay loop generator

; 1000000 cycles:

; -----------------------------

; delaying 999999 cycles:

ldi R20, $09

WGLOOP0: ldi R21, $BC

WGLOOP1: ldi R22, $C4

WGLOOP2: dec R22

brne WGLOOP2

dec R21

brne WGLOOP1

dec R20

brne WGLOOP0

; -----------------------------

; delaying 1 cycle:

nop

; =============================

ret

Hold:

rcall Delay1s ;Delay for 1 sec

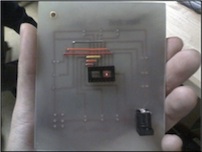
dec duration ;Decrease duration time value by 1

brne Hold ;Keep reapeating calling Delay1s until duration

ret ; reaches 0

### (Part 2)

### Media

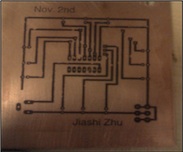


Jump wires

Power supply

all the components





final

product

### Procedure

1. Using EAGLE™ the circuit diagram is complete.
2. Print the diagram on an oil-sensitive paper, take a piece of copper board with roughly same size. Use an electric iron to print the diagram on the paper over the copper board. If succeeded, the board should be as Picture 3 shows above.
3. Place the board in a pool of copper-eating compounds. After several minutes, except the traces covered by oil, all the other copper will be gone. Clean the board.
4. Drill; set up jump wires; place power input plug, micro- controller seat and switch.
5. Cut a piece of plastic board with the same size as the copper board.
6. Drill; Connect LED lights according to the circuit diagram.
7. Place the programmed micro-controller on the chip seat; Connect lights and switch between copper board and plastic board. Use four nails to fix the four corners of the two boards. The intersection simulator is complete.

### Parts

1. LEDs ×12
2. ATtiny24
3. Chip seat
4. Copper circuit board
5. Power port
6. Jump wires
7. Switch
8. Screws ×4

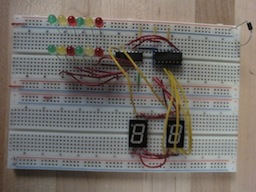
# Activity 4. Intersection Simulator 2

### Purpose

The purpose of this project is to upgrade the work done in previous activity and adds the function of doing a countdown at certain stage when the light is about to change. The picture below shows the prototype of the intersection simulator. The ATtiny24 chip on the left is connected to all the 6 LEDs. 2 of its spare ports are connected to a decoder on the right, which is charge of controlling which counter to turn on and outputting the value of in how many seconds the light is going to change.

### (Part 1)

### Media



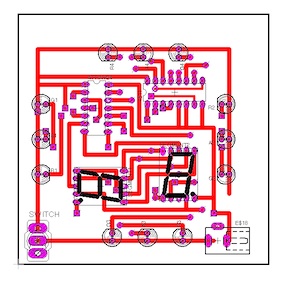
### Procedure

1. The whole system is archived by using interrupts.
2. Previously, the LUT are settled and the base lines to show the count down for each kind of light are added into the LUT.
3. A time counter is settled up as a interrupt. By setting up the starting bit value and pre-time scale, based on the counter of the chip( which increases 1 million per second), the time counter can function to decrease one per second.
4. In the main method, the method that keeps functioning is actually a wait loop doing nothing. Once the counter’s value reaches the base line, the program will display the value.
5. After the value reaches 0, the program will go to the next line in the LUT to generate a new counter and do all the work done above all over again.

### Parts

|  |  |  |
| --- | --- | --- |
| **Quantity** | **Part** | **Description** |
| 1 | ATtiny24 | 14-pin DIP |
| 1 | Chip Seat | 14-pin DIP |
| 4 | Red LEDs | SMM radial |
| 4 | Green |  |
| 4 | Amber |  |
| 2 | 7-Segment Displays |  |
| 1 | BCD ECoder | 16-pin CMOS 4511 |
| 1 | Chip Seat | 16-pin DIP |
| -- | Resistors |  |
| 1 | Toggle switch | SPST |
| 1 | DC jack | 5V |
| 1 | CCB Accrylic, Standoffs |  |

### Layout



### Code:

;Project: Intersection Simulator 2

;AUTHOR: Reuben Sagman and Jiashi Zhu

.include "tn24def.inc"

;-----------Vector Jump Table--------------------------

.org 0x00

rjmp main

.org OVF1addr

rjmp TIM1\_OVF

;-----------LookUp State Table-------------------------

LUT:

.db 0b00100001,10,5,0

.db 0b00100010,3,0,0

.db 0b00100100,1,0,0

.db 0b00001100,10,5,0

.db 0b00010100,3,0,0

.db 0b00100100,1,0,0

LUTEND:

;-----------DEFINE VARIABLES---------------------------

.def temp = R16

.def count = R17

.def segment = R18

.def toggle = R19

;------EQUATES----------------------------------------

.equ CLK = 0x01

.equ CLK8 = 0x02

.equ CLK64 = 0x03

.equ CLK256 = 0x04

.equ CLK1024 = 0x05

;------------------------------------------------------

Main:

ldi temp,low(RAMEND) ;Load the size of the RAM into temp

out SPL,temp ;Establish pointer to top of Stack in RAM

rcall InitPorts ;Initialize PORTA and PORTB

rcall InitLUT ;Set up pointers for the Look Up Table

rcall InitTimerCounter1 ;Initialize time counter, set starting point and scale

rcall NextTrafficState ;Start the

sei Enable Global Interrupts

Wait:

rcall Wait

;------------------------------------------------------

TIM1\_OVF:

rcall ResetCnt1 ;Set up the starting point of the time counter

dec count ;Count down

brne Continue ;If Count reahes the segment, continue.

rcall NextTrafficState ;Go to the next stage of the trafic light system

reti ;return

Continue:

out PORTB,count ;Output the count down to PORTB

cp count,segment ;compare if count has reached segment default value.

breq Activate ;If it has, Activiate the segment

reti

Activate:

tst toggle ;If toggle equals 0

brne Inverse ;If not, do inverse

sbi PORTA,PA6 ;If it does, turn off bit of Segment A and

cbi PORTA,PA7 ; turn on bit of Segment B

ret

Inverse:

sbi PORTA,PA7 ;If toggle = 0, turn on SegmentA.

cbi PORTA,PA6 ; turn off SegmentB

ret

InitPorts:

ser temp ;Set all 8 bits of PORTA and 4 bits of

out DDRA,temp ;PORTB for output use

ldi temp,0x0F

out DDRB,temp

ret

InitLUT:

ldi XH,high(LUT<<1) ;Load the LUT and point the pointer

ldi XL,low(LUT<<1) ;Z at the beginning of the LUT

ldi YH,high(LUTEND<<1) ;Set Y at the end of the LUT

ldi YL,low(LUTEND<<1)

movw Z,X ;Move the pointer to the beginning of LUT

ret

InitTimerCounter1:

ldi temp,CLK;8 ;Set up counter scale

out TCCR1B,temp ;

rcall ResetCnt1 ;Set up the starting point of the time counter

ldi temp,1<<TOIE1

out TIMSK1,temp

clr toggle ;Set the toggle(boolean) to 0

ret

NextTrafficState:

sbi PORTA,PA6 ;Ground the bit on PortA for Segment A,

sbi PORTA,PA7 ;Ground the bit on PortA for Segment A,

com toggle ;Switch the toogle's boolean value(T to F or F to T)

lpm temp,Z+ ;Put the byte referenced by Z on PortA

out PORTA,temp ;Output the PortA value to turn on the LEDs

inc ZL ;Point the pointer to the next position

lpm count,Z+ ;Set up the byte to count

inc ZL ;Point the pointer to the next position

lpm segment,Z+ ;Set up the value as deafult segment

inc ZL ;Point the pointer to the next position

cp ZL,YL ;Compare Z with Y( if the pointer is at the LUTEND or not)

brne If ;If not, return directly

movw Z,X ;If it is, point the pointer back to the LUT beginning

If: ret

ResetCnt1:

ser temp

out TCNT1H,temp

ldi temp,0xF0

out TCNT1L,temp

ret

### (Part 2)

### Media

|  |  |
| --- | --- |
| moto_0187.jpg | moto_0188.jpg |
| moto_0189.jpg drilling | moto_0190.jpg parts |
| moto_0191.jpg final product(top) | moto_0192.jpg final product(bottom) |

### Procedure

1. The copper circuit board is made according to the layout in previous section.
2. Cut a piece of acrylic board with the same shape, four holes with appropriate size are drilled on both the copper board and acrylic board for the screws to go through.
3. In this intersction simulator, a time counter is added to both traffic ways. The counter starts at 10 seconds before the light shows up and the number appears at the beginning of the last 5 seconds.
4. Unlike the previous intersection project, the parts are welded directly onto the copper board.
5. \*\* In this project, unfortunately the acrylic board is missing. Also, because of the defective design of the layout, the over use of jump wires made it impossible to fit all the parts onto the board and work properly.

# Activity 5. Problems 1.14~1.18

p.32~33

### 1.14

a)

b)

|  |  |  |
| --- | --- | --- |
|  | 1111 1111 |  |
| - | 1000 0000 | Base |
|  | 0111 1111 | 1’s complement |
| + | 1 |  |
|  | 1000 0000 | 2’s complement |

|  |  |  |
| --- | --- | --- |
|  | 1111 1111 |  |
| - | 0000 0000 | Base |
|  | 1111 1111 | 1’s complement |
| + | 1 |  |
|  | 0000 0000 | 2’s complement |

c)

d)

|  |  |  |
| --- | --- | --- |
|  | 1111 1111 |  |
| - | 1101 1010 | Base |
|  | 0010 0101 | 1’s complement |
| + | 1 |  |
|  | 0010 0110 | 2’s complement |

|  |  |  |
| --- | --- | --- |
|  | 1111 1111 |  |
| - | 0111 0110 | Base |
|  | 1000 1001 | 1’s complement |
| + | 1 |  |
|  | 1000 1010 | 2’s complement |

e)

f)

|  |  |  |
| --- | --- | --- |
|  | 1111 1111 |  |
| - | 1000 0101 | Base |
|  | 0111 1010 | 1’s complement |
| + | 1 |  |
|  | 0111 1011 | 2’s complement |

|  |  |  |
| --- | --- | --- |
|  | 1111 1111 |  |
| - | 1111 1111 | Base |
|  | 0000 0000 | 1’s complement |
| + | 1 |  |
|  | 0000 0001 | 2’s complement |

### 1.15

a)

b)

|  |  |  |
| --- | --- | --- |
|  | 99 999 999 |  |
| - | 63 325 600 | Base |
|  | 36 674 399 | 9’s complement |
| + | 1 |  |
|  | 36 674 400 | 10’s complement |

|  |  |  |
| --- | --- | --- |
|  | 99 999 999 |  |
| - | 52 784 630 | Base |
|  | 47 215 369 | 9’s complement |
| + | 1 |  |
|  | 47 215 370 | 10’s complement |

c)

d)

|  |  |  |
| --- | --- | --- |
|  | 99 999 999 |  |
| - | 00 000 000 | Base |
|  | 99 999 999 | 1’s complement |
| + | 1 |  |
|  | 00 000 000 | 2’s complement |

|  |  |  |
| --- | --- | --- |
|  | 99 999 999 |  |
| - | 25 000 000 | Base |
|  | 74 999 999 | 1’s complement |
| + | 1 |  |
|  | 75 000 000 | 2’s complement |

### 1.16

a)

b) B2FA=1011 0010 1111 1010

|  |  |
| --- | --- |
|  | FF FF |
| - | B2 FA |
|  | 4D 05 |
| + | 1 |
|  | 4D 06 |

c)

|  |  |  |
| --- | --- | --- |
|  | 1111 1111 1111 1111 |  |
| - | 1011 0010 1111 1010 | Base |
|  | 0100 1101 0000 0101 | 1’s complement |
| + | 1 |  |
|  | 0100 1101 0000 0110 | 2’s complement |

d) 0100 1101 0000 0110=4D 06

a)=d)

### 1.17

a) 6428-3409=3019

b) 0125-1800=-1675

|  |  |  |
| --- | --- | --- |
|  | 9 999 |  |
| - | 3 409 | Base |
|  | 6 590 | 9’s complement |
| + | 1 |  |
|  | 6 591 | 10’s complement |
| + | 6 428 |  |
|  | (1)3 019 |  |

|  |  |  |
| --- | --- | --- |
|  | 9 999 |  |
| - | 1 800 | Base |
|  | 8 199 | 9’s complement |
| + | 1 |  |
|  | 8 200 | 10’s complement |
| + | 0 125 |  |
|  | 8 325 |  |
| => | -1 675 |  |

c) 2043-6152=-4109

d) 1631-745=886

|  |  |  |
| --- | --- | --- |
|  | 9 999 |  |
| - | 6 152 | Base |
|  | 3 847 | 9’s complement |
| + | 1 |  |
|  | 3 848 | 10’s complement |
| + | 2 043 |  |
|  | 5 891 |  |
| => | -4 109 |  |

|  |  |  |
| --- | --- | --- |
|  | 9 999 |  |
| - | 0 745 | Base |
|  | 9 254 | 9’s complement |
| + | 1 |  |
|  | 9 255 | 10’s complement |
| + | 1 631 |  |
|  | (1)0 886 |  |

### 1.18

a)

b)

|  |  |  |
| --- | --- | --- |
|  | 11 111 |  |
| - | 10 001 | Base |
|  | 01 110 | 1’s complement |
| + | 1 |  |
|  | 01 111 | 2’s complement |
| + | 10 011 |  |
|  | (1)00 010 |  |

|  |  |  |
| --- | --- | --- |
|  | 111 111 |  |
| - | 100 011 | Base |
|  | 011 100 | 1’s complement |
| + | 1 |  |
|  | 011 101 | 2’s complement |
| + | 100 010 |  |
|  | 111 111 |  |
| => | -000 001 |  |

c)

d)

|  |  |  |
| --- | --- | --- |
|  | 111 111 |  |
| - | 101 000 | Base |
|  | 010 111 | 1’s complement |
| + | 1 |  |
|  | 011 000 | 2’s complement |
| + | 1 001 |  |
|  | 100 001 |  |
| => | -011 111 |  |

|  |  |  |
| --- | --- | --- |
|  | 111 111 |  |
| - | 010 101 | Base |
|  | 101 010 | 1’s complement |
| + | 1 |  |
|  | 101 011 | 2’s complement |
| + | 110 000 |  |
|  | (1)011 011 |  |

# Activity 6. Altoids Speaker Project

### Purpose

The purpose of this project is to build a portable speaker device for iPod and computers. The product and detailed procedures can be found in this website <http://www.instructables.com/id/Altoids-Tin-Speaker/>. The device is based on a circuit board powered by one AA battery fitting into an empty candy tin.

### Parts of Circuit Board

2



9

8

7

5

4

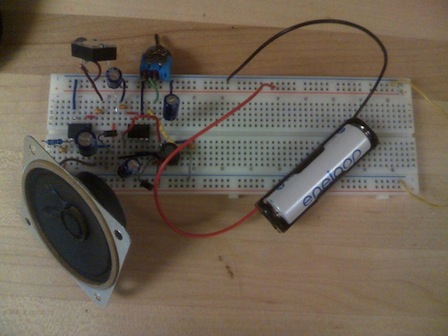
3

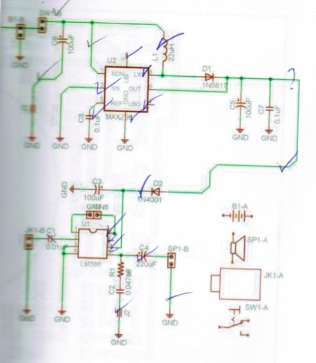
1

6

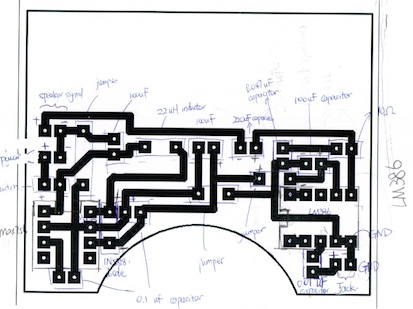
1. 100uF capacitor x3 + 220uF capacitor
2. chip seats x2
3. speaker
4. 0.1uF capacitor(104) x2 + 0.01 capacitor(103) + 0.047 capacitor(473)
5. 10Ω resistor
6. 1N 5818 diode + 1N4001 diode(missing)
7. MAX756 + LM386 + 22uH inductor
8. Audio jack
9. Battery pack

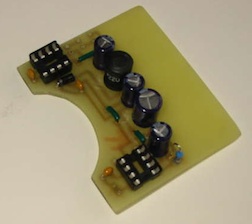
### BreadBoard-Prototype



Above are the rendering and the photo of the prototype. The MAX756 works as a power boost to increase the voltage from the battery. The jack accepts signal from TRS connectors and pass it to the LM386. By connecting Pin 1 and Pin 5 of LM386 the volume can be increased.

### Final Assembly

The process of the entire assembly is shown as follows: 



Here is the circuit diagram of the main board and a photo of the board with all the components installed as comparison.

|  |  |  |
| --- | --- | --- |
| Screen shot 2011-05-12 at 12.18.51 AM.jpg | Screen shot 2011-05-12 at 12.19.38 AM.jpg | Screen shot 2011-05-12 at 12.32.20 AM.jpg |

For the vulnerable components such as switch, battery holder, and the audio jack, pieces of heat shrink tubes are attached at the connections of the wire and the component Heat shrink tubes can tie various wires together to keep the circuit clear and also protect wires from breaking apart during the installation.



Note that a wire is attached to connect the two signal lugs to merge the channels together.

|  |  |
| --- | --- |
| moto_0197.jpg | moto_0199.jpg |
| moto_0201.jpg | moto_0200.jpg |

1. The components are connected to the board as the picture shows.
2. The candy tin is drilled with 29 small holes on the surface for sound to pass through, and 2 larger holes at side for the switch and 3.5mm audio jack to fit in. The whole bottom is covered with paper to prevent the metal surface from creating short circuits on the board.
3. All the connected parts are installed into the tin box as the picture shows.
4. Final product. Tested with various input, the speaker works well and stable.

# Activity 7. Altoids USB Charger Project

### Purpose

This project is created to build a iPod charger with two AA batteries using a Altoids chew gum tin. More detailed descriptions and parts are available at <http://www.ladyada.net/make/mintyboost/index.html>.

### Media

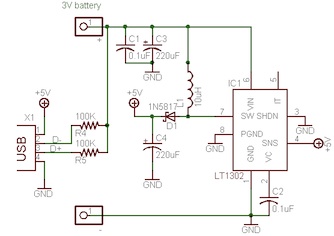
|  |  |
| --- | --- |
| moto_0207.jpg  Overall look | moto_0205.jpg  Battery holder and USB circuit |
| moto_0204.jpg  Circuit board (front) | moto_0206.jpg  Circuit board(back) |

### Parts

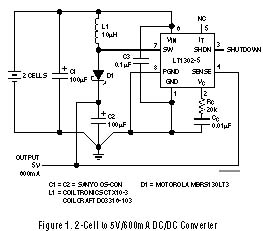
1. 5V boost converter LT1302VN8-5
2. 8-pin socket
3. Power supply capacitor 220uF x2
4. Bypass capacitor 0.1uF x2
5. 100k 1/4w 5% resistor x2
6. 1N5818 diode
7. 10uH power inductor
8. USE type A female pack
9. 2x AA battery holder
10. Circuit board

### Procedure

The working principle of the circuit is shown as the schematic：



The most important part of the whole circuit board is the [boost converter](http://en.wikipedia.org/wiki/Boost_converter) chip that boosts the 2.4 voltage provided by the two AA batteries to 5V for the devices.

A power boost usually functions with at least one transistor and one diode. The quality of a power boost largely depends on its efficiency during transmitting.

In this project, the boost converter is selected to be [LT1302VN8-5](http://www.linear.com/product/LT1302). (Detailed schematic on the left).

LT1302 can operate from a supply voltage as low as 2V and feature automatic shifting between Burst Mode operation at light load, and current mode operation at heavy load. The internal low loss NPN power switch can handle current in excess of 2A and switch at frequencies up to 400kHz.

# 2011 Exam: Conway’s Game of Life

### Purpose:

The purpose of this project is to build two 4 by 4 LED matrix models to simulate the Game of Life.( <http://en.wikipedia.org/wiki/Conway%27s_Game_of_Life>) The whole project is supported by ladyada.net( <http://www.ladyada.net/make/conway/solder.html>).

### Media

|  |  |
| --- | --- |
| moto_0209.jpg  raw board | moto_0211.jpg  Resistors (x16) |
| moto_0214.jpg  Green LED (x16) | moto_0217.jpg  Board with LED and resistors soldered |
| moto_0227.jpg  The connecting model with ports to connect to other models(final) | moto_0229.jpg  Independent model, with battery seat connected and protective case(final) |

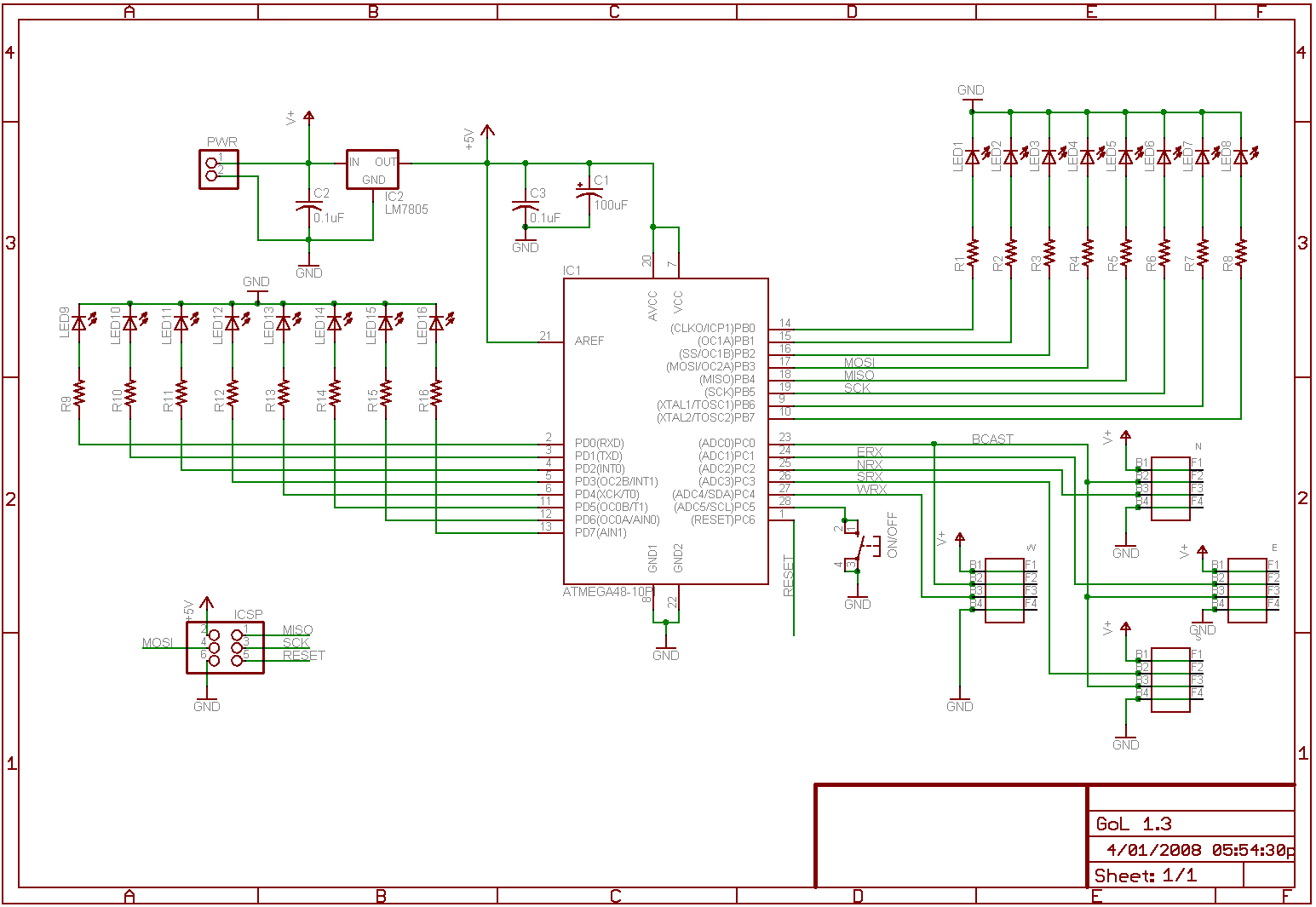
### Parts

1. Microcontroller
2. 28-pin socket
3. Ceramic 0.1uF capacitor
4. 100 ohm 1/4W 5% resistor x16
5. 5mm Green LED x16
6. 6mm tact switch button
7. 2 x AA battery holder
8. Circuit board
9. right-angle headers

### Procedure

More detailed instructions can be found at <http://www.ladyada.net/make/conway/solder.html>.

1. The Schematic of the circuit board is shown as following:



1. Find all the ports marked R1, R2, R3... Place the resistors at those spots and solder their legs onto the board from the backside.
2. Use the cutters to remove the expletory legs. Repeat to all 16 resistors.
3. Find all the ports marked LED1, LED2, LED3... Place LEDs at those spots. Notice that the longer leg of the LED is supposed to go to the positive pole.
4. Use the cutters to remove legs of LEDs.
5. Using a jump wire( can be replaced with one of the expletory legs), connect the two ports at side of IC2. Solder it onto the board.
6. Place the capacitor at the Port C3. Solder it.
7. Solder the chip seat onto the spot IC1, solder it. Place the chip onto the chip seat.
8. Place the ON/OFF/Reset button. Solder the four legs.
9. For the independent model, connect the wires of battery holder to the pwr spot. Solder it. For the connecting model, Solder the right-angle headers at the four sides of the board.

# TEI4M. Arduino

# Project 1. A BiColor LED Flasher

### Purpose

The purpose of this project is to develop a Arduino application that flashes a Red/Green Bicolor LED in response to commands entered into the Serial Monitor. The LED used in this project is the T-1 ¾ HLMP-40xx.

### Reference

Details about the HLMp-40xx can be found here. <http://darcy.rsgc.on.ca/ACES/TEI3M/Datasheets/BiColorLED.pdf>

The project can be found here.

<http://darcy.rsgc.on.ca/ACES/TEI3M/Tasks/1112Tasks.html#BiColorLEDTransmitter>

Arduino Official

<http://arduino.cc/en/>

### Method

The way that the program is designed is very easy. The cathode of the HLMP-40xx is connected to the ground pin on the Arduino. The red and green anodes are connected to two of the 13 pins available. In this project I used pin 12 &13. The program just simply recognizes the input character and sets the corresponding pin to high to light the led up, delays for 0.5 second and sets it to low to turn it off.

### Sketch

int redPin= 12;

int greenPin=13;

int Delay=500;

void setup(){

pinMode(redPin, OUTPUT);

pinMode(greenPin, OUTPUT);

Serial.begin(9600);

}

void loop(){

char ch;

if(Serial.available())

{

ch = Serial.read();

if (ch=='G' || ch=='g')

flash(greenPin);

else if (ch== 'r' || ch=='R')

flash(redPin);

}

}

void flash(int led)

{

digitalWrite(led, HIGH);

delay(Delay);

digitalWrite(led, LOW);

delay(Delay);

}

### Media

|  |  |
| --- | --- |
| BiColorLED.jpg  Red Anode  Green Anode  the HLMP-40xx  Cathode | ::Downloads:moto_0250.jpg  the Arduino Uno |
|  |  |

# Project 2. High Brightness Luxeon LED

### Purpose

This project is based on the Morse Code Project and an improved version with a supportive Arduino shield. (Morse Code Project is from the book: 30 Arduino Projects for the Evil Genius)

### Reference

Arduino Evil Genius

<http://www.arduinoevilgenius.com/>

Arduino proto shield

<http://www.ladyada.net/make/pshield/>

LM117 regulator

<http://darcy.rsgc.on.ca/ACES/TEI3M/Datasheets/LM117.pdf>

### Method

There are several external parts needed to make the 1W Luxeon LED to function.

1 W 4 Ω resistor

330 Ω resistor

LM117 3 Terminal Adjustable Regulator

To build the proto shield, following components are needed.

Printed circuit board

3mm Red LED

470-1.0K Resistors for LED

0.1uF 50V ceramic capacitor

36 pin male 0.1" header

Procedures to assemble the shield can be found here. <http://www.ladyada.net/make/pshield/solder.html> After the shield is ready, the LED is arranged onto the shield as the picture in media section below shows. The power source for the whole circuit comes out from the standard 5V pin and ground pin on the Arduino (shield).

The 1W 4Ω resistor is connected directly to the 5V power pin and the other side goes to the positive side of the Luxeon LED.

The negative side of the LED is connected to the middle leg of the regulator, the output.

For the regulator, its adjustment leg is connected to be grounded and its input leg is connected to the 330Ω resistor and then the pin 12 on the Arduino board, which controls the pattern of the light.

### Sketch

|  |
| --- |
| int ledPin= 12;  char\* letters[]={  ".-", "-...", "-.-.", "-..", ".", "..-.", "--.", "....", "..", //A-I  ".---", "-.-", ".-..", "--", "-.", "---", ".--.", "--.-", ".-.", //J-R  "...", "-", "..-", "...-", ".--", "-..-", "-.--", "--.."    };  char\* numbers[]= {  "-----", ".----", "..---", "...--", "....-", ".....", "-....", "--...",  "---..", "----." };    int dotDelay=200;    void setup(){  pinMode(ledPin, OUTPUT);  Serial.begin(9600);  }    void loop(){  char ch;  if(Serial.available())  {  ch = Serial.read();  if (ch>='a' && ch<='z')  flashSequence(letters[ch-'a']);    else if (ch>='A' && ch<= 'Z')  flashSequence(letters[ch-'A']);  else if (ch>='0' && ch<= '9')  flashSequence(numbers[ch-'0']);  else if(ch==' ')  delay(dotDelay\*4);    }  }    void flashSequence(char\* sequence)  {  for (int i=0; sequence[i]!=NULL; i++)  flashDotOrDash(sequence[i]);    delay(dotDelay\*3);  }    void flashDotOrDash(char i){  digitalWrite(ledPin, HIGH);  if(i =='.')  delay(dotDelay);  else delay(dotDelay\*3);  digitalWrite(ledPin, LOW);  delay(dotDelay);  } |

### Media

|  |  |
| --- | --- |
| protoshielddone.jpg  Assembled Proto Shield([click](http://farm2.static.flickr.com/1267/1341944959_d39a197f27_b.jpg) to zoom) | HighBrightnessMorseCode.jpgThe overview of tested breadboard  ([click](http://darcy.rsgc.on.ca/ACES/TEI3M/images/HighBrightnessMorseCode.jpg) to zoom) |
| Luxeon1WLED.jpg  The 1 W Luxeon LED used in this project | F03-13.jpg  The finished product (on the website) |
| :Desktop:moto_0252.jpg  final product |  |

# Project 3. S.A.D. Light

### Purpose

The purpose of this project is to create a S.A.D light panel with the Luxeon high-brightness LED, the same used in last project. An S.A.D light is a device that provides white light that mimic daylight to make benefit to people that suffers seasonal affective disorder.

### Reference

This project is based on the instruction of **30 Arduino Projects for the Evil Genius**

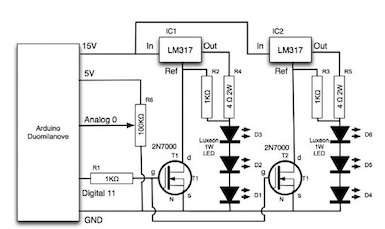
By Simon Monk.

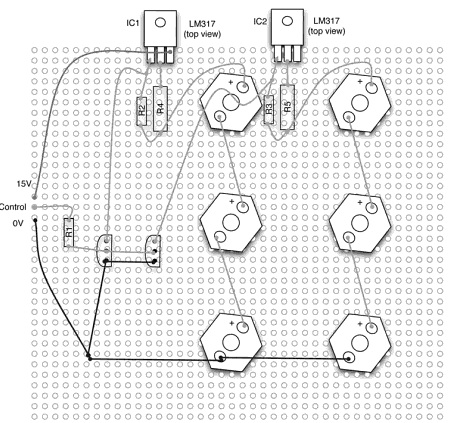
<http://www.arduinoevilgenius.com/home>

### Parts List

|  |  |
| --- | --- |
| Quantity | Components |
| 1 | Arduino |
| 6 | Luxeon 1W LED |
| 3 | 1 KΩ 0.5W metal film resistor |
| 2 | 4 Ω 2W resistor |
| 1 | 100K linear potentiometer |
| 2 | LM317 Voltage regulator |
| 2 | 2N7000 FET |
| 1 | Regulated 15V 1A power supply |
| 1 | Perf board |
| 1 | Three-way screw terminal |

### Procedure





The picture above shows a very clear layout for the arrangement of the board in schematic diagram and a perf board preview.

1. This project requires 15V for the six Luxeon LEDs and 5V for the regulators. Both power sources are provided by different pins on the Arduino.
2. In order to give the light the effect of fading on and off, normal digital pins cannot be used. To access PVM for maximum efficiency and power savery, analog pin 11 is settled to write mode and provides an output from 0 to 255, which each represents off and fully on.
3. Two 2N7000 FETs receive the output from pin 11 and work as a switch. Only when a high enough voltage is passed, it will connect and turn off all the LEDs.
4. The time period of the six Luxeon LEDs is depended on variable resistor. The value of the resistor is transferred to analog pin 2 on the Arduino. By increasing the resistor value, the period for one cycle becomes longer.

### Sketch

|  |
| --- |
| int ledPin = 11;  int analogPin = 2;  int startupSeconds = 20;  int turnOffSeconds = 10;  int minOnSeconds = 300;  int maxOnSeconds = 1800;  int brightness = 0;  void setup()  {  pinMode(ledPin, OUTPUT);  digitalWrite(ledPin,HIGH);  pinMode(14, OUTPUT);  pinMode(18, OUTPUT);  digitalWrite(18, HIGH);  int analogIn = analogRead(analogPin);  int onTime = map(analogIn, 0, 1023, minOnSeconds, maxOnSeconds);  turnOn();  delay(onTime \* 1000);  turnOff();  }  void turnOn()  {  brightness = 0;  int period = startupSeconds \* 1000 / 256;  while (brightness < 255);  {  analogWrite(ledPin, 255 - brightness);  delay(period);  brightness ++;  }  }  void turnOff()  {  int period = turnOffSeconds \* 1000 / 256;  while (brightness >= 0)  {  analogWrite(ledPin, 255 - brightness);  delay(period);  brightness --;  }  }  void loop()  {} |

### Media

|  |  |
| --- | --- |
| :Desktop:IMG_0132.jpg back | :Desktop:IMG_0130.JPG front |
| :Desktop:IMG_0133.jpg Rendering |  |

# Project 4: Alphanumeric (16-Segment) Display

### Purpose

The purpose of this project is to show the English alphabet and Arabic numbers with a 16-segemnt display kit.



### Reference

Alphanumeric LED Display Library <http://17segment.blogspot.com/>

Arduino Library <http://arduino.cc/en/Hacking/Libraries>

### Pin/Segment Map

|  |  |
| --- | --- |
| Segment map  pins’ position(up to down):  Left: 1~9  Right:18~11, dot pin:10 | :Desktop:Screen Shot 2011-10-23 at 11.29.53 PM.png  Matched sequence of the pins on the segment |

### Sketch

The program blueprint used in this project is from <http://17segment.blogspot.com/>.

The knowledge of “Libraries” in Arduino is required. The language used to build the library is in c.

|  |
| --- |
| BasicUseLoop.pde  /\*  Created by Ben Duncan - 7 April, 2009  Shared under Creative Commons  Give easy control of Alphanumeric LED displays  (Kingbright type)  \*/  #include <Sts.h> // Include the Seventeen Segment library  Sts sts; // Create the Sts object  void setup(){  }  void loop() {  for(int i=0; i<26; i++){  sts.displayChar(i); // Display the capital character  delay(500); // Wait 1/2 second  }  for(int i=0; i<10;i++){  sts.displayNumb(i); // Display the number  delay(500); // Wait 1/2 second  }  } |

|  |
| --- |
| Sts.h  /\*  Created by Ben Duncan - 7 April, 2009  Shared under Creative Commons  Give easy control of Alphanumeric LED displays  (Kingbright type)  \*/  #ifndef Sts\_h  #define Sts\_h  #include "WProgram.h"  class Sts  {  public:  int \_charNum;  void displayChar(int \_charNum);  void displayNumb(int \_charNum);  byte charactersCap[26][16];  byte numbers[10][16];  private:  void setPins();  };  #endif |

|  |
| --- |
| Sts.cpp  /\* Created by Ben Duncan - 7 April, 2009  Shared under Creative Commons  Give easy control of Alphanumeric LED displays  (Kingbright type)  \*/  #include "WProgram.h"  #include "Sts.h"  byte characters[26][16] = {  {1,0,0,1,1,0,0,1,0,0,1,0,1,1,0,1}, // A  {1,0,0,1,0,0,0,1,1,1,0,1,0,0,1,1}, // B  {1,0,0,1,0,0,0,1,1,1,0,0,0,0,0,1}, // C  {1,0,0,1,0,0,0,1,1,1,0,1,0,1,0,1}, // D  {1,0,0,1,0,0,0,1,1,1,0,0,0,1,0,1}, // E  {1,0,0,1,1,0,0,1,0,0,0,0,0,0,0,1}, // F  {1,0,0,1,0,0,0,1,1,1,1,0,1,0,0,1}, // G  {0,0,0,1,1,0,0,1,0,0,1,0,1,1,0,0}, // H  {1,1,0,0,0,1,0,0,1,1,0,0,0,0,0,1}, // I  {1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,1}, // J  {0,0,0,1,1,0,0,1,0,0,0,1,0,0,1,0}, // K  {0,0,0,1,0,0,0,1,1,1,0,0,0,0,0,0}, // L  {1,1,0,1,0,0,0,1,0,0,1,0,0,1,0,1}, // M  {0,0,1,1,0,0,0,1,0,0,1,1,0,1,0,0}, // N  {1,0,0,1,0,0,0,1,1,1,1,0,0,1,0,1}, // O  {1,0,0,1,1,0,0,1,0,0,0,0,1,1,0,1}, // P  {1,0,0,1,0,0,0,1,1,1,1,1,0,1,0,1}, // Q  {1,0,0,1,1,0,0,1,0,0,0,1,1,1,0,1}, // R  {1,0,0,1,1,0,0,0,1,1,1,0,1,0,0,1}, // S  {1,1,0,0,0,1,0,0,0,0,0,0,0,0,0,1}, // T  {0,0,0,1,0,0,0,1,1,1,1,0,0,1,0,0}, // U  {0,0,1,0,0,0,0,0,0,0,1,1,0,1,0,0}, // V  {0,0,0,1,0,1,0,1,1,1,1,0,0,1,0,0}, // W  {0,0,1,0,0,0,1,0,0,0,0,1,0,0,1,0}, // X  {0,0,1,0,0,1,0,0,0,0,0,0,0,0,1,0}, // Y  {1,0,0,0,0,0,1,0,1,1,0,0,0,0,1,1}}; // Z  byte numbers[10][16] = {  {1,0,0,1,0,0,0,1,1,1,0,1,0,1,0,1}, // 0  {1,1,0,0,0,1,0,0,1,1,0,0,0,0,0,0}, // 1  {1,0,0,0,1,0,0,0,1,1,0,0,1,0,0,1}, // 2  {1,0,0,0,1,0,0,0,1,1,1,0,1,1,0,1}, // 3  {0,0,0,1,1,0,0,0,0,0,1,0,1,1,0,0}, // 4  {1,0,0,1,1,0,0,0,1,1,1,0,1,0,0,1}, // 5  {1,0,0,1,1,0,0,1,1,1,1,0,1,0,0,1}, // 6  {1,0,0,0,0,0,0,0,0,0,1,0,0,1,0,1}, // 7  {1,0,0,1,1,0,0,1,1,1,1,0,1,1,0,1}, // 8  {1,0,0,1,1,0,0,0,1,1,1,0,1,1,0,1}}; // 9  void Sts::setPins(){  for(int i=2; i<17;i++){  pinMode(i, OUTPUT);  }  }  void Sts::displayChar(int \_charNum)  {  for(int i=0; i<15;i++){  digitalWrite((i+2), characters[\_charNum][i]);  }  }  void Sts::displayNumb(int \_charNum)  {  for(int i=0; i<15; i++){  digitalWrite((i+2), numbers[\_charNum][i]);  }  } |

### Media

|  |  |
| --- | --- |
| :Desktop:moto_0255.jpg | :Desktop:moto_0254.jpg |
|  |  |

### Custom Sketch

|  |
| --- |
| String chars="ABCDEFGHIJKLMNOPQRSTUVWXYZ 0123456789";  String input="";  int index;  /\*  Created by Ben Duncan - 7 April, 2009  Shared under Creative Commons  Give easy control of Alphanumeric LED displays  (Kingbright type)  \*/  #include <Sts.h> // Include the Seventeen Segment library  Sts sts; // Create the Sts object  void setup(){  Serial.begin(9600);  sts.setPins();  sts.blank();    }  void loop() {  while(Serial.available()){  char recv=Serial.read();  input=recv;  input=input.toUpperCase();  Serial.print(input);  displayString(input);  sts.blank();  }  }  void displayString(String str){  for (int i=0;i<str.length();i++){  index=chars.indexOf(str.charAt(i));  if (index<26)  sts.displayChar(index);  else if(index==26)  sts.blank();  else sts.displayNumb(index-27);  delay(500);  sts.blank();  }  } |

This code above allows the segment to display texts(letters and numbers) input from the serial monitor of the Arduino program.

# Appendix C. Monthly Projects

## October 2011: XBee Wireless Solution

### Reference

This project is powered by kits and instructions from sparkfun.com.

Link for tutorial: <http://www.sparkfun.com/tutorials/194>

### Purpose

Sometimes the Arduino needs to function while placed in some portable devices. Then it would hard for the programmer to connect the Arduino to computers with USB wires. The purpose of this project is to solve such problems by using two XBee chips to transfer data wirelessly between Arduino and computers.

### Assembly Guide-11.jpgParts List

XBee 1mW Chip Antenna Modules (XBee s1) x2

Arduino Stackable Header x4

XBee Shield PCB

XBee Explorer USB

### Description

Sparkfun.com provides Arduino shield PCB and USB explorer set.

The assembly procedures can be found in the link in Reference Section.

Either chip can work as the receiver or the sender.

The USB explorer connects to the computer through a mini-USB cable.



Primary pins (the picture on the left) are settled at the back of the board for convenience of advanced development to the XBee chip such as wireless bootloading. Unfortunately the Arduino model used (Uno) has too high a baud rate for the XBee to trace. So the advanced functions may not be achievable in this project.

With the shield PCB the XBee kit can stack up on the Arduino and become compatible with other projects based on it.



By adding the XBee shield to the Arduino, which is programmed to perform the last project -- 16-segemnt displays, the kit is now able to receive text from computers on the other side of the room. Technically the maximum range of the s1 XBee chips is up to 300 feet.

### Summary

Wireless function doesn’t necessarily show its advantage and significance when working with 16-segment displays that are fixed to a breadboard. However, in the following projects for the greenhouse, the same function will be applied to various sensors so that data of temperature or moisture in the greenhouse can be collected wirelessly from the computer room several feet away.

# Project 5. Thermal Management

## Part 1. NTC Thermistor

### Reference

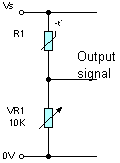
<http://darcy.rsgc.on.ca/ACES/TEI3M/Tasks/1112Tasks.html#Thermal1>

<http://en.wikipedia.org/wiki/Thermistor>

<http://www.arduino.cc/playground/ComponentLib/Thermistor2>

<http://www.arduino.cc/en/Math/H>

### Purpose

The purpose of this project is to add a NTC Thermistor to the Arduino and receives resistance value that changes according to the temperature (in a negative exponential pattern) The current temperature can be calculated based on the given resistance value.

### Principle

The principle of calculation of the changes of resistance value is shown as the picture beside.

The NTC Thermistor is connected to a variable resistor, whose value is fixed at 10k Ω, in a serial connection. In the middle a wire connects the NTC Thermistor to the analog0 pin on Arduino for the reading of the voltage travels through the thermistor.

The resistor value then can be calculated with such formula:



From the information provided by Wikipedia, the temperature should be:





### Parts List

NTC Thermistor(TOC310)

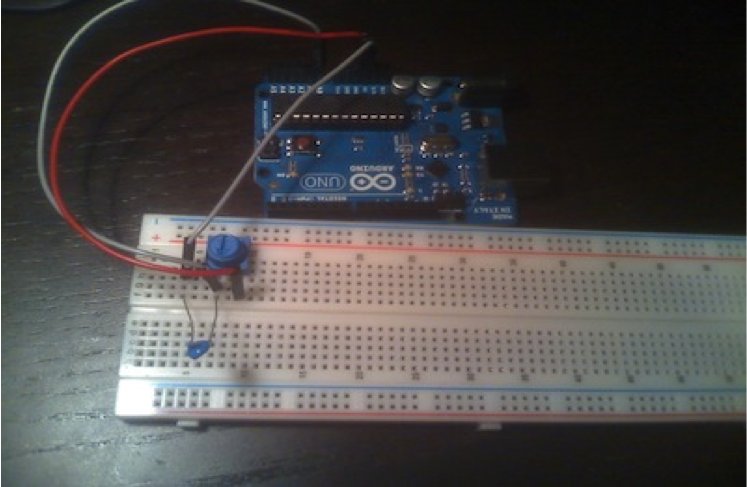
Arduino Uno

Variable resistor

Wires

Breadboard

### Media



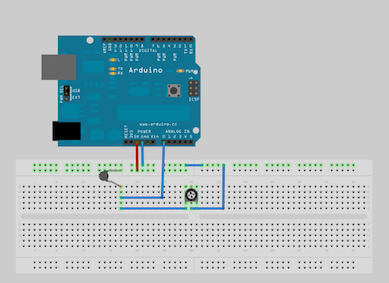
NTC Thermistor

Variable resistor

Power

Ground

Analog0

Breadboard Layout(powered by [Fritzing](http://fritzing.org/))

## Part 2. Serial Communications Using Processing

### Reference

<http://pscmpf.blogspot.com/2008/12/arduino-lm35-sensor.html>

<http://www.processing.org/>

<http://darcy.rsgc.on.ca/ACES/TEI3M/Tasks/1112Tasks.html#Thermal2>

### Purpose

As an advanced version of the previous project, the purpose of this project is to print the results from last project on a graphic interface on the computer. Arduino program is too limited for such function. Therefore a driver coded in Processing language (the language Arduino code based on). It receives the value of temperature instead of the serial monitor of Arduino and then draws a scale bar and dot diagram to display the change of the temperature in a period of time.

### Parts List

NTC Thermistor (TOC310)

Arduino Uno

Variable resistor

Wires

Breadboard

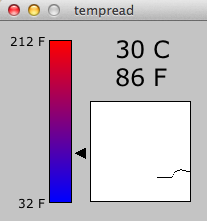
(Though this project is still based on the NTC Thermistor, soon it will be replaced with a more efficient and effective Precision Centigrade Sensor (LM35DZ).

### Procedure

The program on the Arduino is almost the same as the one in last project. The only difference is: when the temperature data is passed to the computer, it’s transferred into byte format for Processing driver to do the calculation more easily. Note that when the data is sent, the method print() must be used instead of println()to avoid the data interruption.

The Processing code is provided by <http://pscmpf.blogspot.com/2008/12/arduino-lm35-sensor.html>. It has two major functions. It draws a rectangle as the scale bar and a square as the frame for the dot diagram. After the temperature value is calculated, it’s displayed on both the scale bar and the diagram. An array is created to store all the values to show the pattern of temperature change on the dot diagram.

### Media

The breadboard layout is the same as last project.

A video demo of this project running can be found here.

<http://mail.rsgc.on.ca/~azhu/tempread2.mov>

### Sketch

Arduino part:

|  |
| --- |
| #include <Math.h>  int ana0=0;  double R1;  double rinf=0.01066;  double temp;  void setup() {  Serial.begin(9600);  }  void loop() {  ana0 = analogRead(A0);  R1 =(1023.0-ana0)\*10000/ana0;  temp =4100/log(R1/rinf);  temp-=273;  Serial.print((byte)temp);  delay(1000);  } |

Processing Part:

|  |
| --- |
| //import Serial communication library  import processing.serial.\*;  //init variables  Serial commPort;  float tempC;  float tempF;  int yDist;  PFont font12;  PFont font24;  float[] tempHistory = new float[100];  void setup()  {  //setup fonts for use throughout the application  font12 = loadFont("Verdana-12.vlw");  font24 = loadFont("Verdana-24.vlw");    //set the size of the window  size(210, 200);    //init serial communication port  String output= Serial.list()[0];  commPort = new Serial(this, output, 9600);    //fill tempHistory with default temps  for(int index = 0; index<100; index++)  tempHistory[index] = 0;  }  void draw()  {  //get the temp from the serial port  while (commPort.available() > 0)  {  tempC = commPort.read();    //refresh the background to clear old data  background(123);  //draw the temp rectangle  colorMode(RGB, 160); //use color mode sized for fading  stroke (0);  rect (49,19,22,162);  //fade red and blue within the rectangle  for (int colorIndex = 0; colorIndex <= 160; colorIndex++)  {  stroke(160 - colorIndex, 0, colorIndex);  line(50, colorIndex + 20, 70, colorIndex + 20);  }    //draw graph  stroke(0);  fill(255,255,255);  rect(90,80,100,100);  for (int index = 0; index<100; index++)  {  if(index == 99)  tempHistory[index] = tempC;  else  tempHistory[index] = tempHistory[index + 1];  point(90 + index, 180 - tempHistory[index]);  }    //write reference values  fill(0,0,0);  textFont(font12);  textAlign(RIGHT);  text("212 F", 45, 25);  text("32 F", 45, 187);    //draw triangle pointer  yDist = int(160 - (160 \* (tempC \* 0.01)));  stroke(0);  triangle(75, yDist + 20, 85, yDist + 15, 85, yDist + 25);    //write the temp in C and F  fill(0,0,0);  textFont(font24);  textAlign(LEFT);  text(str(int(tempC)) + " C", 115, 37);  tempF = ((tempC\*9)/5) + 32;  text(str(int(tempF)) + " F", 115, 65);  }  } |

## Part 3. Precision Centigrade Sensor (LM35DZ)

### Reference

<http://www.ectinschools.org/page.php?ps=2&p=928>

<http://darcy.rsgc.on.ca/ACES/TEI3M/Datasheets/LM35.pdf>

<http://pscmpf.blogspot.com/2008/12/arduino-lm35-sensor.html>

### Purpose

A more efficient and simple thermistor is used to replace the NTC thermistor. The LM35DZ provides a more straight-forward reading as a response to the temperature change. The LM35DZ is also used by the original producer of previous project.

|  |  |
| --- | --- |
| ttp://shop.rabtron.co.za/catalog/images/ntcx.jpg  NTC Thermistor | ttp://www.modtronix.com/images/to92.jpg  LM35DZ |
| -Rate to temperature change is exponential  -Complex formula in program for temperature  -Requires a variable resistor to regulate the voltage passed through | -Rate to temperature change is linear  -Easy calculation in program  -Requires no extra component’s assistance |

This purpose of this project is to assemble the LM35DZ onto the breadboard and modify the code to display temperature properly. Preparation for multiple sensors (our next project) is also done.

### :Desktop:Screen Shot 2011-11-28 at 1.23.26 AM.pngProcedure

According to the datasheet of DM35LZ, the left pin connects to the 5V output of Arduino, and middle connects to pin Anlog0 for data input. The right pin goes to the Ground. After that the assembly is done.

In Arduino, the original code suggests such formula for temperature:

T(ºC)=byte(Analog0\*5\*100.00/1024)

In Processing, lots of changes need to be made to meet the needs of multiple sensors. The following is my design of the interface.

|  |  |
| --- | --- |
| :Desktop:Screen Shot 2011-11-21 at 2.04.13 AM.png | :Desktop:Screen Shot 2011-11-28 at 1.36.27 AM.png |

Approximately 9 sensors will be used in the following project. Because there are several readings displaying at the same time, number value of the temperature is deleted. The diagram is enlarged and scaled for more accurate readings. 9 temperature bars are color-coded and will match with the lines in the diagram. The scale of the temperature (minimum and maximum) can be changed to zoom to the preferable range.

### Parts List

Less: NTC Thermistor (TOC310)

Arduino Uno

Less: Variable resistor

Wires

Breadboard

Add: LM35DZ

### Sketch

Arduino:

|  |
| --- |
| //declare variables  float tempC;  int tempPin = 0;  void setup()  {  Serial.begin(9600); //opens serial port, sets data rate to 9600 bps  }  void loop()  {  tempC = analogRead(tempPin); //read the value from the sensor  tempC = (5.0 \* tempC \* 100.0)/1024.0; //convert the analog data to temperature  Serial.print((byte)tempC); //send the data to the computer  delay(100); //wait one second before sending new data  } |

Processing:

|  |
| --- |
| //import Serial communication library  import processing.serial.\*;  //init variables  Serial commPort;  int numSensors=9;  float tempC;  float tempF;  int yDist;  PFont font12;  PFont font24;  PFont font8;  int bkgrd = 123;  int width=600;  int height=400;  //graph values  int sqrx=240;  int sqry=50;  int sqrLength=300;  float[] tempHistory = new float[sqrLength];  //scale values  int cMin=20;  int cMax=50;  int cNum=5;  int timeNum=10;  int timeMax=50;  //temp bar values  int barWidth=20;  int barLength=100;  int barSpace=30;  int startx=50;  int starty=20;  //palatte  int[][] colors= {{0,0,0},{255,36,0},{255,246,143},  {0,238,0},{0,245,255},{0,0,156},  {185,211,238},{205,127,50},{105,89,205}  };  void setup()  {  //setup fonts for use throughout the application  font12 = loadFont("Verdana-12.vlw");  font24 = loadFont("Verdana-24.vlw");  font8 = loadFont("Verdana-8.vlw");    //set the size of the window  size(width, height);    //init serial communication port  String output= Serial.list()[0];  commPort = new Serial(this, output, 9600);        //fill tempHistory with default temps  for(int index = 0; index<100; index++)  tempHistory[index] = 0;    background(bkgrd);  stroke (0);  for (int i=0; i<3; i++) {  for (int j=0;j<3;j++){  int x=startx+j\*(barWidth+barSpace);  int y=starty+i\*(barSpace+barLength);    //draw the temp rectangle  rect(x,y,barWidth,barLength);    //fade red and blue within the rectangle  colorMode(RGB, barLength); //use color mode sized for fading  for (int colorIndex = 0; colorIndex <= barLength; colorIndex++){  stroke(barLength - colorIndex, 0, colorIndex);  line(x, colorIndex+y, x+20, colorIndex+y);  }    //color code the rectangles  colorMode(RGB, 255); //reset color mode  stroke (bkgrd);  fill(colors[3\*i+j][0],colors[3\*i+j][1],colors[3\*i+j][2]);  rect(x,y-15,20,10);  }  }  //write reference values  fill(0,0,0);  textFont(font12);  textAlign(RIGHT);  text(cMax+" C", startx-5, starty+5);  text(cMin+" C", startx-5, starty+barLength+5);    //scales &Title  textFont(font24);  text("J.Z:Multiple LM35DZs",sqrx+sqrLength/2+120,sqry-20);  text("C",sqrx-15,sqry+sqrLength/2);  text("Time(s)",sqrx+sqrLength/2+30,sqry+sqrLength+45);    textFont(font8);  textAlign(RIGHT);  for(int i=0;i<cNum+1;i++){  int stage=(cMax-cMin)/cNum;  text(cMin+stage\*i,sqrx-5,sqry+3+sqrLength/cNum\*(cNum-i));  }  for(int i=0;i<timeNum+1;i++){  int step=sqrLength/timeNum;  text(i\*timeMax/timeNum,sqrx+5+i\*step,sqry+sqrLength+15);  }  }  void draw() {  stroke(0);  //get the temp from the serial port  while (commPort.available() > 0) {  tempC = commPort.read();    //refresh the background to clear old data        //draw graph  fill(255,255,255);  rect(sqrx,sqry,sqrLength,sqrLength);  smooth();  strokeWeight(2);  for (int index = 0; index<sqrLength; index++) {  stroke(colors[0][0],colors[0][1],colors[0][2]);  if(index == 299)  tempHistory[index] = tempC;  else  tempHistory[index] = tempHistory[index + 1];  point(sqrx + index, sqry+sqrLength - (tempHistory[index]-cMin)/cMax\*sqrLength);  }  strokeWeight(1);    //draw triangle pointer  fill(bkgrd);  noStroke();  for(int i=0; i<3;i++){  int x=72+i\*50;  rect(x,0,25,height);  }  yDist =starty+barLength-int((tempC-cMin)/cMax\*barLength);  stroke(0);  fill(0);  int trix=startx+barWidth+5;  triangle(trix, yDist, trix+10, yDist-5, trix+10, yDist+5);    }    } |

### Media

|  |
| --- |
| moto_0262.jpg  Assembly |
| Screen Shot 2011-11-27 at 11.48.32 PM.png  Breadboard Layout(powered by [Fritzing](http://fritzing.org/)) |

## Part 4. Multiple Sensors: Polling

### Reference

<http://www.ectinschools.org/page.php?ps=2&p=928>

<http://darcy.rsgc.on.ca/ACES/TEI3M/Datasheets/LM35.pdf>

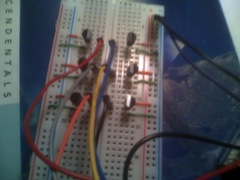
<http://pscmpf.blogspot.com/2008/12/arduino-lm35-sensor.html>

### Purpose

This section is a more advanced version based on Part 3. In this project, the maximum number of analog pins on the Arduino Uno to receive data from 6 thermometers and display the changes of temperature in one graph on the desktop.

### Procedure

As the picture beneath shows, 6 LM35DZ thermometers are each connects to one of the 6 analog pins on the Arduino. Detailed assembly for the LM35DZ can be found in the previous project. Most modifications are made on the software. Clearly such arrangement has limited resource and the wiring is rather complicated. In the next project a more efficient resolution will be introduced.



### Parts List

Arduino Uno

Wires

Breadboard

LM35DZ X6

### Sketch

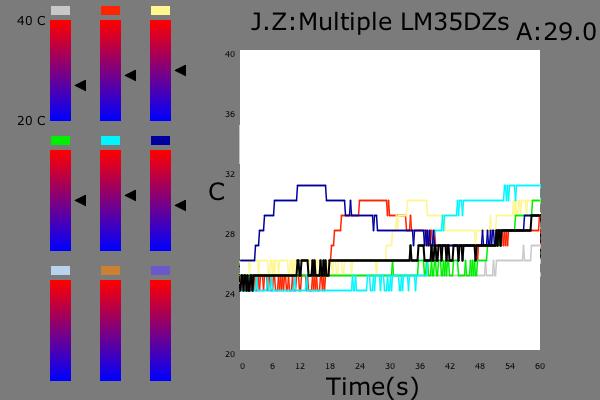
Arduino

|  |
| --- |
| float tempC; //declare variables  int numSensors=6;  int sensor = 0;  int sensors[]= {0,1,2,3,4,5};  void setup()  {  Serial.begin(9600); //opens serial port, sets data rate to 9600 bps  Serial.flush();  }  void loop()  {  tempC = analogRead(sensor); //read the value from the sensor  tempC = (5.0 \* tempC \* 100.0)/1024.0; //convert the analog data to temperature  Serial.print((byte)tempC); //send the data to the computer  sensor=(sensor+1)%numSensors;  delay(50); //wait one second before sending new data  } |

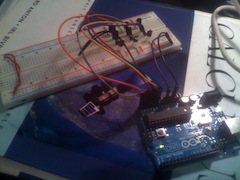
Processing

|  |
| --- |
| //import Serial communication library  import processing.serial.\*;  //init variables  Serial commPort;  int numSensors=6;  int sensor=0;  float tempC;  float tempF;  int yDist;  PFont font12;  PFont font24;  PFont font8;  int bkgrd = 123;  int width=600;  int height=400;  boolean firstrun=true;  int firstcount=0;  //graph values  int sqrx=240;  int sqry=50;  int sqrLength=300;  float beginy;  float endy;  float[][] tempHistory = new float[numSensors+1][sqrLength];  int [][] trix=new int[9][3];  //scale values  int cMin=20;  int cMax=40;  int cNum=5;  int timeNum=10;  int timeMax=60;  //temp bar values  int barWidth=20;  int barLength=100;  int barSpace=30;  int startx=50;  int starty=20;  //palatte  color[] colors= {  color(200, 200, 200), color(255, 36, 0), color(255, 246, 143),  color(0, 238, 0), color(0, 245, 255), color(0, 0, 156),  color(185, 211, 238), color(205, 127, 50), color(105, 89, 205),  };  void setup()  {  //setup fonts for use throughout the application  font12 = loadFont("Verdana-12.vlw");  font24 = loadFont("Verdana-24.vlw");  font8 = loadFont("Verdana-8.vlw");  //set the size of the window  size(width, height);  //init serial communication port  String output= Serial.list()[0];  commPort = new Serial(this, output, 9600);  //fill tempHistory with default temps  for (int c=0; c<numSensors+1;c++)  for (int index = 0; index<sqrLength; index++)  tempHistory[c][index] =0;  //triangle coordinates  for (int r=0;r<3; r++)  for (int c=0; c<3;c++) {  trix[r\*3+c][0]=startx+(c+1)\*barWidth+c\*barSpace+5;  trix[r\*3+c][1]=starty+(r+1)\*barLength+r\*barSpace;  trix[r\*3+c][2]=trix[r\*3+c][1];  }  background(bkgrd);  stroke (0);  for (int i=0; i<3; i++) {  for (int j=0;j<3;j++) {  int x=startx+j\*(barWidth+barSpace);  int y=starty+i\*(barSpace+barLength);  //draw the temp rectangle  rect(x, y, barWidth, barLength);  //fade red and blue within the rectangle  colorMode(RGB, barLength); //use color mode sized for fading  for (int colorIndex = 0; colorIndex <= barLength; colorIndex++) {  stroke(barLength - colorIndex, 0, colorIndex);  line(x, colorIndex+y, x+20, colorIndex+y);  }  //color code the rectangles  colorMode(RGB, 255); //reset color mode  stroke (bkgrd);  fill(colors[3\*i+j]);  rect(x, y-15, 20, 10);  }  }  //write reference values  fill(0, 0, 0);  textFont(font12);  textAlign(RIGHT);  text(cMax+" C", startx-5, starty+5);  text(cMin+" C", startx-5, starty+barLength+5);  //scales &Title  textFont(font24);  text("J.Z:Multiple LM35DZs", sqrx+sqrLength/2+120, sqry-20);  text("C", sqrx-15, sqry+sqrLength/2);  text("Time(s)", sqrx+sqrLength/2+30, sqry+sqrLength+45);  textFont(font8);  textAlign(RIGHT, CENTER);  for (int i=0;i<cNum+1;i++) {  int stage=(cMax-cMin)/cNum;  text(cMin+stage\*i, sqrx-5, sqry+3+sqrLength/cNum\*(cNum-i));  }  for (int i=0;i<timeNum+1;i++) {  int step=sqrLength/timeNum;  text(i\*timeMax/timeNum, sqrx+5+i\*step, sqry+sqrLength+15);  }  }  void draw() {  stroke(0);  //get the temp from the serial port  if (commPort.available() > 0) {  tempC = commPort.read();  //refresh the background to clear old data  //draw graph  noStroke();  fill(255, 255, 255);  rect(sqrx, sqry, sqrLength, sqrLength);  smooth();  strokeWeight(1.5);  //first run  if (firstrun) {  tempHistory[sensor][firstcount]= tempC;  if (sensor==numSensors-1)  firstcount++;  if (firstcount==sqrLength && sensor==numSensors-1)  firstrun=false;  }  //after first run  else for (int index = 0; index<sqrLength; index++) {  if (index == sqrLength-1)  tempHistory[sensor][index] = tempC;  else  tempHistory[sensor][index] = tempHistory[sensor][index + 1];  }  for (int c=0; c<firstcount;c++) {  int total=0;  for (int r=0; r<numSensors;r++)  total+=tempHistory[r][c];  tempHistory[numSensors][c]=total/numSensors;  }  //print  for (int r=0;r<numSensors+1;r++) {  stroke(colors[r]);  if (r==numSensors) {  stroke(color(0));  strokeWeight(2);  }  for (int i=0;i<firstcount; i++) {  if (i<sqrLength-1 && tempHistory[r][i+1]>0 )  line(sqrx + i, convert(tempHistory[r][i]), sqrx + i+1, convert(tempHistory[r][i+1]));  }  }  strokeWeight(1);  //draw triangle pointer & text  fill(bkgrd);  noStroke();  for (int i=0; i<3;i++) {  int x=72+i\*50;  rect(x, 0, 25, height);  }  rect( 0.86\*width, 0, 0.15\*width, 0.11\*height);  trix[sensor][2] =trix[sensor][1]-int((tempC-cMin)/(cMax-cMin)\*barLength);  stroke(0);  fill(0);  for ( int i=0; i<numSensors;i++)  triangle(trix[i][0], trix[i][2], trix[i][0]+10, trix[i][2]-5, trix[i][0]+10, trix[i][2]+5);  sensor= (sensor+1) % numSensors;  fill(0, 0, 0);  textFont(font24);  textAlign(LEFT);  if (firstcount!=0)  text("A:"+str(tempHistory[numSensors][firstcount-1]), 0.86\*width, 0.1\*height);  }  }  float convert(float temp) {  float initial=sqry+sqrLength-(temp-cMin)/(cMax-cMin)\*sqrLength;  return max(min(initial, sqry+sqrLength), sqry);  } |

### Media



Display Frame



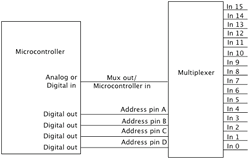
The thermometer panel

## Part 5. Multiple Sensors: Multiplexing

### Purpose

During previous process, a problem is revealed: when the number of thermometers is increased, the Arduino eventually doesn’t contain enough ports for input reading. A multiplexer is used as a solution. It allows the Arduino to read values for multiple pins through one pin and matches the value to the correct port.

### Procedure

As the graph on the left states, the multiplexer used in this project has 20 pins available. 4 of them are connected to four digital pins on Arduino to address the port that is supposed to receives the value. One pin is connected to Analog0 to pass the actual value. And the rest of the pins are open to the thermometers (LM35DZ).

Since all the values are sent from the same Analog pin now, the code on Arduino needs to be adjusted to match the selection of sensors. The Processing code only works as a receiver, therefore can remain the same.

### Sketch

Arduino:

|  |
| --- |
| // multiplexing LM35DZ  //uses CMOS CD4067BE Analog Multiplezer  //JIASHI ZHU  //2011.12.21  float tempC;  int numSensors=6;  int sensor=0;  int sensorPin = A0;  int addressPins[]= {2,3,4,5};  void setup()  {  Serial.begin(9600); //opens serial port, sets data rate to 9600 bps  Serial.flush();  for (int i=0; i<4; i++)  pinMode(addressPins[i],OUTPUT);    }  void loop()  {  selectNextSensor(sensor);  tempC = analogRead(sensorPin); //read the value from the sensor  tempC = (5.0 \* tempC \* 100.0)/1024.0; //convert the analog data to temperature  Serial.print((byte)tempC); //send the data to the computer  sensor=(sensor+1)%numSensors;  delay(50); //wait one second before sending new data  }  void selectNextSensor(int channel){  for( int n=0; n<4; n++)  digitalWrite(addressPins[n],bitRead(channel,n));    } |

### Media

|  |  |
| --- | --- |
| :Desktop:photo 2.JPG Four pins connect to the digital pins on Arduino for addressing the right sensor;  One pin connects to Analog0 for value transporting.  Two wires connect to the 5V and ground for power supply. | :Desktop:photo 1.JPG 6 pins are connected to the 6 thermometers. Readings are received in turns to match up the sensors setup in processing. The addressing information is controlled by a byte value passed through 4 pins. |

### :.Trash:temp:temp-2214.jpg

Display Frame

## 5V/12V Breadboard-Compatible Regulator

### Reference

<http://darcy.rsgc.on.ca/ACES/TEI3M/Tasks/1112Tasks.html#5VRegulator>

<http://www.sparkfun.com/products/114> (similar projects)

[www.fairchildsemi.com/ds/LM/**LM7805**.pdf](http://www.fairchildsemi.com/ds/LM/LM7805.pdf)

### Purpose

Sometimes a power source needs to feed different devices which each requires certain voltage supply, for example, a Arduino and a cooling fan. This voltage regulator is a easy resolution for such situations by transferring the original power source into 2 streams: a fixed 5V stream and a original voltage stream (in this project that is 12V).

### Parts List

LM7805 Voltage Regulator (major component)

DC Barrel Connector

10V 100uF capacitor

16V 10uF capacitor

220Ω resistor

650Ω resistor

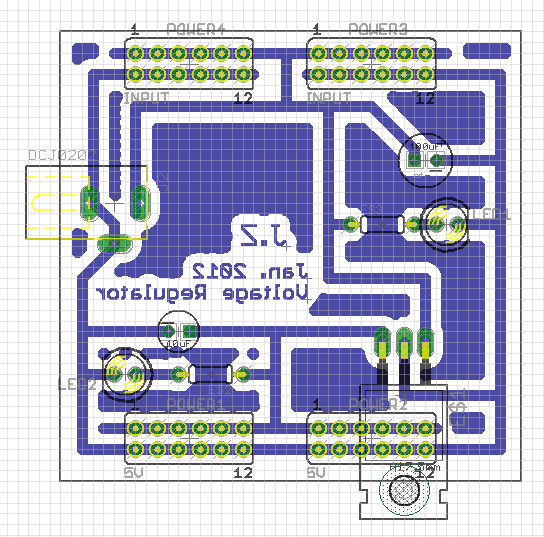
2x6 header pins X2

2x6 female Pins X2

Copper board

### :Desktop:Screen Shot 2012-02-01 at 12.34.20 AM.pngProcedure

The designing of the copper board is completed on EAGLE©.

Here is the brief digital layout of the board.

Noticeable Things:

1. In order to make this regulator breadboard compatible, the voltages circuits and the common ground are placed to fit a breadboard. Therefore, voltage (positive) is always on top of the ground (negative).
2. The blue print area is extra coverage of copper that remains on the board. This adjustment is to save the copper-consuming chemicals during the process of building wires.

After the board design is finished, the building process is pretty much straightforward: (Pictures in media section)

1. The circuit is printed on a small piece of Mimeograph paper.
2. Place the sheet over a copper board. When the board is heated up the oil print will come off to cover the copper.
3. Place the copper board in the chemicals tub. The oil saves the copper on the circuit traces from being removed.
4. Drilling, placing components, wiring and soldering.

Brief Explanation on the components

|  |  |
| --- | --- |
| DC Barrel Connector | LM7805 |
| ttp://www.adafruit.com/images/medium/21mmdcjack_MED.jpg  The voltage (positive) pin is the one at the end of the jack.  The other two pins are both ground for additional support. They are usually connected in a series to the common ground in the board designing. | ttp://www.seekic.com/uploadfile/ic-data/200916185259579.jpg  A brief sketch of the LM7805. As the picture marked, the left leg is for voltage input, the right leg is for modified voltage output( 5V), and the middle one connects to ground. |

### Media

|  |  |
| --- | --- |
| :Desktop:ss:IMG_0086.JPGThe copper board sheet. During the heating up process the oil doesn’t come off completely, which causes several breaks in the circuit. It’s later fixed by soldering patch. | :Desktop:ss:IMG_0085.JPG  Bread Board trial |
| :Desktop:ss:Screen Shot 2012-02-01 at 1.22.28 AM.png  Front of the board | Untitled.png  Back of the board. (lots of solder patches can be seen.) |
| Video:  <http://www.youtube.com/watch?v=zFlXWtezzSI&feature=g-upl&context=G227ac8cAUAAAAAAAAAA> | |

## Part 6: Air Flow Management

### Reference

Getting Started with Arduino p.70-71

<http://darcy.rsgc.on.ca/ACES/TEI3M/Tasks/1112Tasks.html#Thermal6>

<http://darcy.rsgc.on.ca/ACES/TEI3M/Datasheets/IRF520.pdf>

### Purpose

The purpose of this project is to set up a fan station, which the strength of the airflow can be controlled by a LM35DZ thermometer. The voltage regulator will be used to supply different voltages to the Arduino and the fan at the same time.

### Parts List

TA350DC Nidec fan

IRF520 MOSFET (TO-220)

5V/12V voltage regulator

LM35DZ thermometer

Arduino Uno

### Procedure

Two new components are added in this project. Here is a brief description on each.

|  |  |
| --- | --- |
| TA350DC Nidec fan | IRF520 MOSFET |
| :Desktop:nidecFan.jpg  This CPU fan requires a 12V to function at normal rate, which is harmful for the Arduino(that’s what the voltage regulator is for).  3 wires are connected in the fan’s circuit.  Power, Tach Signal Wire, and Ground.  For now only the Power and Ground wires are used. | :Desktop:Untitled.png  The MOSFET looks identical comparing to the LM7805 used for the regulator. However it has complete different function. The three pins are each called Gate, Drain, and Source.  Basically the MOSFET works as a switch. If a value is passed to the Gate pin, the Drain and Source are then connected. The amount of voltage allowed out of the total amount in the circuit at the connection is determined by the value to Gate pin. |

The 5V/12V voltage regulator supplies 5V and 12V each to the Arduino and a breadboard.

The IRF520 is placed on the breadboard. The Gate pin is connected to pin9 on the Arduino Uno. The Arduino is programmed to send certain values( out of 255) to the Gate pin to control the voltage supplied to the fan. The Drain pin is connected to the ground wire of the fan and the Source pin is connected to the common ground. So when a value is passed, the MOFSET will ground the fan to finish the circuit and the fan will start functioning.

At side, the LM35DZ is connected to the common power and ground, and the middle pin is connected to analog0 on Arduino to pass the temperature value.

The Arduino is programmed to have a TempOFF and a TempON value. When the temperature hits the TempON, the fan is turned on and cools down the thermometer. And when the temperature drops down and hit the TempOFF value, the fan will be de-grounded and shut down. The temperature is mapped from the scale of [TempOFF, TempON] to a scale of [0,255] to control the voltage passed through the fan circuit, which will eventually control the fan’s functioning rate.

### Sketch

|  |
| --- |
| float tempC;  int sensor=0; //pin to read temperature value  int fanOFFTemp=25; // shut down the fan at this temperature  int fanONTemp=30; //turn on the fan at this temperature  int fanPin=9; //pin to control the voltage to the fan  double voltage;  int time=0;  int vprint;  int vmax=11.5; //voltage input(max)  void setup() {  pinMode(9, OUTPUT);  Serial.begin(9600);  Serial.flush();  }  void loop() {  tempC=analogRead(sensor); //take the temperature  tempC=(int)((5.0 \* tempC \* 100.0)/1024.0); //convert to celsius  voltage=min(max(tempC,fanOFFTemp),fanONTemp); //convert the temp into the scale of [fanOFF, fanON]  voltage=(voltage-fanOFFTemp)/(fanONTemp-fanOFFTemp)\*255; //calculate voltage value  vprint=(int)(voltage\*vmax/255);  analogWrite(fanPin,voltage);  display();  time+=5;    delay(5000);    }  void display() {  Serial.print("Time:");  Serial.print(time);  Serial.print('\t');  Serial.print("Temp:");  Serial.print(tempC);  Serial.print('\t');  Serial.print("Event:");  if(tempC>fanOFFTemp)  Serial.print("ON");  else  Serial.print("OFF");  Serial.print('\t');  Serial.print("Supply:");  Serial.print(vprint);  Serial.println("V");  } |

### Media

|  |  |
| --- | --- |
| :Desktop:IMG_0090.JPG | Circuit overview  (breadboard +fan+ Arduino + voltage regulator) |
| :Desktop:IMG_0091.JPG | Breadboard layout |
| Video: <http://www.youtube.com/watch?v=5FXuwDk6vJk&feature=youtu.be> | |

## Part 7. Fan Speed

### Reference

<http://darcy.rsgc.on.ca/ACES/TEI3M/Datasheets/ta350dc.pdf>

<http://darcy.rsgc.on.ca/ACES/TEI3M/Tasks/1112Tasks.html#Thermal7>

<http://arduino.cc/playground/Main/ReadingRPM>

<http://www.arduino.cc/en/Reference/AttachInterrupt>

### Purpose

Based on the previous achievements, the purpose of this project is to add the function to detect the speed of the fan’s rotation. This is doable with the Nidec DC Fan Tach Signal that comes with the fan (the yellow wire).

### Parts List

TA350DC Nidec fan

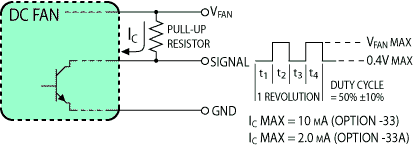
IRF520 MOSFET (TO-220)

5V/12V voltage regulator

LM35DZ thermometer

Arduino Uno

### Procedure

The code to detect and calculate the fan speed rate is modified based on the version provided [here](http://arduino.cc/playground/Main/ReadingRPM). The basic concept is the built-in **attachInterrupt** function provided by Arduino. As the picture below shows, when the fan is powered and spinning, through every two full rotations the Fan Tach Signal sends a voltage impulse (to whatever it is connected to).

And in this project, the signal wire is connected to the Interrupt Pin 0(the Pin 2 on the digital pins side) of the Arduino.

Every time the signal is sent, the number is recorded. When it reaches certain level ( the max number is custom. The large it is, the more accuracy it will reveal), the Interrupt method basically calls the microchip to pause from its current tasks and determine the time it takes through the recording. Then by simple dividing, the fan speed can be calculated. After that the microchip goes back to the task it leaves before.

Notable things:

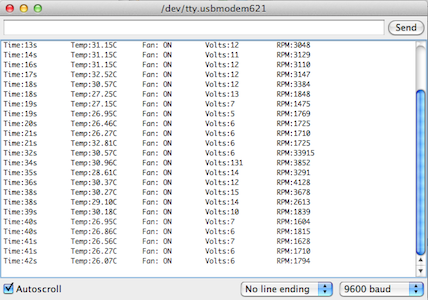
1. Because of the characteristics of Interrupt function, delay() method cannot be used in the project(when the Interrupt method calls it will cut in during the delaying and cause errors in the output). The replacement solution is to include the display() method in the Interrupt method. When a Interrupt signal is sent, the program will detach the Interrupt to prevent further interruption, calculate and display all the required values, and then reattach the Interrupt to start the next round.
2. When a voltage is applied, it doesn’t quite come as a clean ON/OFF signal, which is usually expected. Therefore, a debounce circuit is used to smooth out the multiple voltage changes. It can be simply archived by adding a LED (indicator) and a resistor to the circuit.

### Sketch

|  |
| --- |
| // Full Featured Fan Control System  // 1. LM35DZ Celsius Temperature Sensor on A0  // 2. IRF520 MOSFET on PWM Pin 9 for High Voltage Switching  // 3. Hardware Interrupt 0 on Pin 2 to monitor Fan Speed  // Taken from:  //http://arduino.cc/playground/Main/ReadingRPM  //-----------------------------------------------  int pinFAN = 9; // Control Voltage to Fan  int pinINTR = 2; // Int0 for fan speed  int rpmLimit = 100; //max number for recording of rotations  int maxFanRPM = 3100; //the maximum fan speed  int fanOFFTemp=25;  int fanFULLTemp=30;  float celsius;  int reading; //integer to hold the scaled temperature  int pwm; //mapped voltage depending on the temperature  volatile byte rpmcount;//rotations records, major Interrrupt signal  unsigned long rpm; //calculated fan speed  unsigned long timeold; //for time calculation  void setup(){  Serial.begin(9600);  pinMode(pinFAN,OUTPUT);  pinMode(2,OUTPUT);  digitalWrite(2,HIGH);  analogReference(INTERNAL);  attachInterrupt(0, rpm\_fun, RISING); //attach Interrupt  rpmcount = 0;  timeold = millis(); //time record begins  }  void loop(){  celsius = analogRead(0)\* 100.0/1024.0;  reading = min(max(fanOFFTemp,celsius),fanFULLTemp);  pwm = map(reading,fanOFFTemp,fanFULLTemp,0,255);  analogWrite(pinFAN,pwm); //temperature and voltage calculations  if (rpmcount>=rpmLimit){  detachInterrupt(0);  display();  timeold = millis();  rpmcount = 0;  attachInterrupt(0, rpm\_fun, RISING); //when reaches the maximum rotations, work out the rpm  }    }  void rpm\_fun()  {  rpmcount++;  }  void display(){  Serial.print("Time:");  Serial.print(millis()/1000);  Serial.print("s \tTemp:");  Serial.print( celsius);  Serial.print("C\t Fan:");  if (pwm>0)  Serial.print(" ON");  else  Serial.print("OFF");  Serial.print("\tVolts:");  int volt = map(rpm,0,3100,0,12);  Serial.print(volt);  Serial.print(" \t RPM:");  rpm=0.03\*(millis() - timeold)\*rpmcount;  Serial.print(rpm);  } |

### Media

|  |  |
| --- | --- |
| :Desktop:IMG_0092.JPG | The three wire that come out of the fan: voltage, ground and Nidec DC Fan Tach Signal. The signal wire is connected to a debouncing circuit, and then to the Interrupt Pin 0(Pin 2) on the Arduino. |
| :Desktop:IMG_0093.JPG | The complete layout of the breadboard. At the very left is the thermometer circuit, which connects to Analog pin0.  Then it’s the rpm Interrupt circuit to Interrupt Pin 0 (Pin 2).  And at the right side is the MOSFET voltage regulation circuit that controls the rate of the fan, which connects to Pin 9 on Arduino. |



From the output, despite the errors at beginning (because the output begins to show in the middle of data flows), the voltages and rpms basically match the temperature value.

Video: <http://www.youtube.com/watch?v=U7HSs8LAw9k&feature=youtu.be>

## Part 8. ATtiny85 Port

### Reference

<http://www.youtube.com/watch?v=30rPt802n1k&feature=related>

<http://hlt.media.mit.edu/?p=1229>

<http://darcy.rsgc.on.ca/ACES/TEI3M/Tasks/1112Tasks.html#Thermal8>

### Purpose

In order to apply the auto-fan system to the RSGC’s greenhouse, the Arduino Uno set must be replaced with a more lighter and efficient device. The solution is using the ATtiny85 to work as the microchip on the Arduino Uno. This requires access to upload code from the Arduino board to a separate chip. In this project, based on the software support from the high-low tech group, the ATtiny85 chip will determine the room temperature and apply certain voltage to the CPU fan.

### Parts List

ATtiny85

TA350DC Nidec fan

IRF520 MOSFET (TO-220)

5V/12V voltage regulator

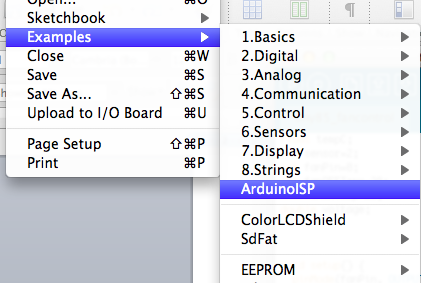
LM35DZ thermometer

Arduino Uno

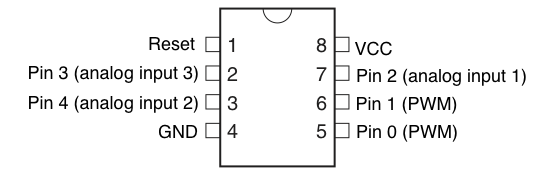
### Procedure

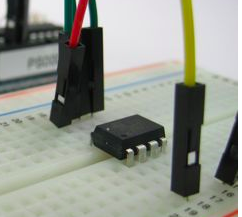
The first thing required is to setup the Arduino Uno as the station to modify other chips.

1. Go to Tools > Board, and select “Arduino Uno”, and upload the “ArduinoISP” sketch.

1. Download the support files for TItiny85, and copy them into the Arduino folder.
2. Go back to Tools > Board, and the ATtiny85 configurations should be present. Choose ATtiny85( w/ Arduino as ISP).

The software is all set for uploading code.





Above is the pinout of ATtiny85. In order to upload code into the chip, the pins in the picture on the left must connect to the Uno:

|  |  |
| --- | --- |
| Arduino Uno | ATtiny85 |
| 10 | Reset |
| 11 | Pin 0(PWM) |
| 12 | Pin 1(PWM) |
| 13 | Pin 2(analog input 1) |

Then connecting power and ground to ATtiny85 (either from Uno or separately), and the chip is fully ready to be programmed.

**Notable Thing:**

The supporting file provided seems to handle received data in a unique way. Therefore in the code the analogRead() value is modified based on repeated testing.

After the code is uploaded to ATtiny85, the connections have to be modified to fit into the fan project. Analog Pin 2( Pin 4 on pinout) is connected to the thermometer to receive temperature value; Pin 0 (PWM) is responsible to send digital value to the MOSFET to control the voltage supplied to the fan.

### Sketches

|  |
| --- |
| float tempC;  int sensor=2;  int fanPin=0;  int fanOFFTemp=20;  int fanONTemp=25;  double voltage;  void setup() {  pinMode(fanPin, OUTPUT);  }  void loop() {  tempC=analogRead(sensor)/4;  voltage=min(max(tempC,fanOFFTemp),fanONTemp);  voltage=map(voltage, fanOFFTemp, fanONTemp, 0, 255);  analogWrite(fanPin,voltage);  } |

### Media

|  |  |
| --- | --- |
| :Desktop:IMG_0099.JPG | Code uploading section  (Uno + ATtiny85) |
| :Desktop:IMG_0100.JPG | Thermal management fan project (w/ ATtiny85) overview |
| :Desktop:IMG_0101.JPG | Fan with power regulator circuit |
| :Desktop:IMG_0102.JPG | Attiny85 overview( code uploading station) |
| DEMO video link: <http://youtu.be/htt-9mzKjlo> | |

# Project 6. The Tune Player

### Reference

<http://darcy.rsgc.on.ca/ACES/TEI3M/Tasks/1112Tasks.html#TunePlayer>

<http://www.phy.mtu.edu/~suits/NoteFreqCalcs.html>

### Purpose

This project is inspired by the Project 19: the Tune Player in 30 Arduino Projects for the Evil Genius. By using Arduino and a DAC made by combing several resistors to approximate a sine wave, this project can play a series of musical notes through a mini speaker. Based on the project provided, further improvements are made, such as more accurate notes, wider range of notes and addition of sharp notes.

### Parts List

Arduino Uno

100uF non-polarized capacitor

100uF 16V electrolytic capacitor

10kΩ 0.5W resistor x5

4.7kΩ 0.5W resistor x3

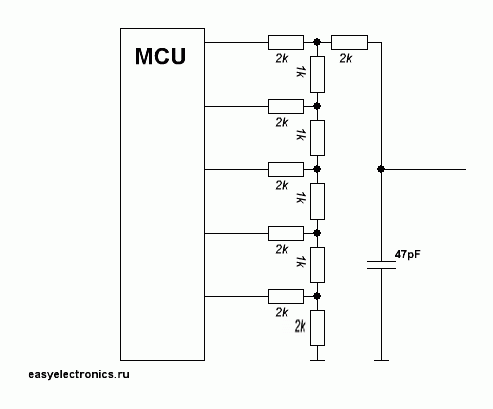
1mΩ 0.5W resistor

100kΩ linear potentiometer

8Ω speaker

TDA7052 1W audio amplifier

### Procedure

Sound is generated from waves. Sound can be generated from an Arduino by simply turning one of its pins on and off at certain frequency, which is rough and grating because it appears as a square wave. A better tone is performed when the wave is similar to the sine wave.

In order to produce a sine wave of change in frequency, a DAC (digital-to-analog converter) is used. As the schematic on the right shows, it is made by arranging resistors in such order. The red-highlight resistors in the parts list are the ones used to as DAC in this project.

By connecting the DAC to an audio amplifier, a series of rather natural tones can be generated with the right frequency provided. Based on the calculations done here, (<http://mail.rsgc.on.ca/~azhu/MusicalNotes.slsx> ) and some modifications in testing, the values are settled and restored in the toneDurations matrix in sketches.

Notes are referred to with a 3- character-based system. The alphabet letter determines the note (A-G). If there is an # following then it’s the sharp of that note. The following number represents its position in the scale (out of 3 scales). High number means higher pitch. Then the last number represents its duration. The larger the number, shorter the note is. A sample note would be C#01. By putting several notes in a string, and reads the string note by note, a simple melody can be performed.

### Sketches

|  |
| --- |
| // Project 19 Tune Player  // Orginal Author: Simon Monk  // Tuning Improvements: K. McMillan  // Coding Modifications: Jiashi Zhu  //-------------------------------------  //Enhanced Encoding: Note/Octave/Duration  // Star Wars Cantina Theme Song  char\* song = "A12D12A12D12A12D14A12\*G04A12 A15G05A15G01F02F01D01 A12D12A12D12A12D14A12\*G04A12 G02G01F03G02D13C14 \*B14\*A11 A12D12A12D12A12D14A12\*G04A12 C14 C11A13G03F01 D01 D01 F01 A11 C11 D14C12\*G04A14 F01 ";  //-------------------------------------  int DACPins[] = {2, 4, 7, 8};  int sin16[] = {  7, 8, 10, 11, 12, 13, 14, 14, 15, 14, 14, 13, 12, 11,  10, 8, 7, 6, 4, 3, 2, 1, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6};  long tempo = 12000;  int note;  int octave;  float duration;  // Three octave tone matrix  String scale = "A#BC#D#EF#G#";  int toneDurations [3][12]= { //12  {262, 253 ,235, 220, 208, 195, 180, 171, 159, 149, 140, 132},//12a  // A0 A#0 B0 C0 C#0 D0 D#0 E0 F0 F#0 G0 G#0  {120,114,108,100,94, 88, 82, 77, 70, 66, 61, 57},//12  //A1 A#1 B1 C1 C#1 D1 D#1 E1 F1 f#1 G1 G#1  { 53,49,46,42,39,36,34,31,27,25,22, 19}};//12  //A2 A#2 B2 C2 C#2 D2 D#2 E2 F2 F#2 G2 G#2  // Song for a headache....  //char\* song = " ";  void setup()  {  Serial.begin(9600);  for (int i = 0; i < 4; i++)  {  pinMode(DACPins[i], OUTPUT);  }  }  void loop()  {    int i = 0;  int ch = song[0];  while (ch != 0)  {  if (ch == ' ')  {  delay(75);  }  else {  note = scale.indexOf(ch);  i++;  if (song[i] == '#')  note++;  i++;    octave = song[i]-'0';  i++;  if (song[i]=='S')  duration = 16;  else if (song[i]=='0')  duration = 0.5;  else duration = song[i]-'0';    Serial.print("Note: ");  Serial.println(note);  Serial.print("Octave: ");  Serial.println(octave);  Serial.print("Duration: ");  Serial.println(duration);  Serial.print("Tone Duration: ");  Serial.println(toneDurations[octave][note]);  Serial.println();  playNote(toneDurations[octave][note]);    }  i++;  ch = song[i];  }  delay(1000);  }  void setOutput(byte value)  {  digitalWrite(DACPins[3], ((value & 8) > 0));  digitalWrite(DACPins[2], ((value & 4) > 0));  digitalWrite(DACPins[1], ((value & 2) > 0));  digitalWrite(DACPins[0], ((value & 1) > 0));  }  void playNote(int pitchDelay)  {  long numCycles = tempo/pitchDelay/duration;  for (int c = 0; c < numCycles; c++)  {  for (int i = 0; i < 32; i++)  {  setOutput(sin16[i]);  delayMicroseconds(pitchDelay);  }  }  } |

### Media

|  |  |
| --- | --- |
| :Desktop:IMG_0114.JPG | Layout of the tune player project (schematic diagram) |
| :Desktop:IMG_0115.JPG | The Tune player project |
| :Desktop:Screen Shot 2012-05-16 at 1.00.11 AM.png | Schematic diagram of the DAC system using an R-2R ladder |
| :Desktop:IMG_0116.JPG | A close shot of the resistors.  All ten resistors combine to work as an DAC to perform different notes. |
| :Desktop:IMG_0117.JPG | A close shot at the Arduino. All the digital pins used are D2, D4, D7 and D8 as it shows on the picture. |
| Video Demo of playing Cantina Theme from Star Wars:  <http://www.youtube.com/watch?v=UzH4SB9vD8Y&feature=youtu.be> | |

# 2012 Final Exam: Servo-Controlled Laser

### Reference

<http://darcy.rsgc.on.ca/ACES/TEI3M/Exams/2012TEI3MFinalExam.docx>

### Purpose

The purpose of this project is to build a Laser shooter with two servo motors to control the beam’s coordination. It’s done by sticking one servo motor vertically to the side of another horizontally placed servo motor. With the program sketches attached, the programmer is able to control the moving of the beam (with two scales of movement, large and small) and turn the beam on and of as they want.

### Parts List

Arduino Uno

0.8mW red laser card

9g servo motor x2

270Ω resistor

Breadboard

Project case

### Procedure

New Parts:

|  |  |
| --- | --- |
| Servo Motor | Laser Card |
| The Servo Motor contains 3 wires.  The red wire is for voltage input.  (Operating Voltage: 4.8V~6.0V)  The black wire is for ground.  The white wire is the signal/control wire. | The Laser card has two wires: voltage and ground.  It is a light practical device working on a 3.1V and 0.8mW. |

The assembly of this device is simple.

1. One servo motor is attached to a corner of the project case, with the reel part sticking out of the side of the case.
2. The second servo motor is attached to the reel side of the 1st motor horizontally. Make sure the reel of the 2nd motor is pointing upward.
3. Staple the laser card to the reel of the 2nd motor.

Wire Arrangement

1. The red and black wires of both servo motors go to the power(5V) and ground of the bread board.
2. White wire of the 1st (horizontal) motor goes to pin2 on the Arduino, to control the y value of the beam. The white wire of the 2nd (vertical) motor goes to pin3, for x value control.
3. Optional: Replace the laser card’s default wires with stronger and longer wires.
4. Connect the black wire of the laser card to ground. Connect the red wire to the 270Ω resistor, then leads the resistor to pin 4 of the Arduino. Then by changing the value passing to pin 4(0/1), the laser beam can be turned on and off.

### Sketches

|  |
| --- |
| #include <Servo.h>  int laserPin = 4;  Servo servoV;  Servo servoH;  int x = 90;  int y = 90;  int minX = 10;  int maxX = 170;  int minY = 50;  int maxY = 130;  void setup()  {  servoH.attach(3);  servoV.attach(2);  pinMode(laserPin, OUTPUT);  Serial.begin(9600);  }  void loop()  {  char ch;  if (Serial.available())  {  ch = Serial.read();  if (ch == '0')  {  digitalWrite(laserPin, LOW);  }  else if (ch == '1')  {  digitalWrite(laserPin, HIGH);  }  else if (ch == '-')  {  delay(100);  }  else if (ch == 'c')  {  x = 90;  y = 90;  }  else if (ch == 'l' || ch == 'r' || ch == 'u' || ch == 'd')  {  moveLaser(ch, 1);  }  else if (ch == 'L' || ch == 'R' || ch == 'U' || ch == 'D')  {  moveLaser(ch, 5);  }  }  servoH.write(x);  servoV.write(y);  }    void moveLaser(char dir, int amount)  {  if ((dir == 'r' || dir == 'R') && x > minX)  {  x = x + amount;  }  else if ((dir == 'l' || dir == 'L') && x < maxX)  {  x = x - amount;  }  else if ((dir == 'u' || dir == 'U') && y < maxY)  {  y = y + amount;  }  else if ((dir == 'd' || dir == 'D') && x > minY)  {  y = y - amount;  }  } |

### Media

|  |  |
| --- | --- |
| :Desktop:IMG_0127.JPG | Overall Layout (Arduino +Bread Board + servo-laser unit (on a box) |
| :Desktop:IMG_0126.JPG | Servo-Controlled Laser unit  Two Servo unit combine together to control x & y values of the laser beam. |
| :Desktop:IMG_0131.JPG | Pins used on the Arduino.  5V and Ground for power  Pin2 for 1st servo (x value)  Pin3 for 2nd servo (y value)  Pin4 for laser Card (switch of the laser beam) |
| :Desktop:IMG_0129.JPG | Layout of the breadboard  Two pairs of yellow wires on the far side are power supply for the servos.  Three green wires in the middle are power control for the laser card. A 270Ω resistor is used to lower the voltage. |
|  |  |