Introduction to Digital Electronics

Background information:

Course names:
► MEEN 685 (independent research)
► ENTC 219 Digital Electronics listed in engineering technology department
► ESET 219 Digital Electronics (as listed in electronic systems degree program)

Timing: Spring 2015 (third semester of graduate school)
Supplies necessary: access to basic hands-on electronics & hand-tools.

Background: in the master's program for mechanical engineering the student's advisor may assign a unique topic of study for a 3 credit hour course that supports the student's research area.

For this situation, my advisor Dr. Joe Morgan was also an undergraduate instructor for the Digital Electronics course, a junior-level course for the undergraduates in the Electronic Systems Engineering Technology Program. I collected the course materials and independently studied and met the course objectives with the professor's oversight.

Outcome:
For a mechanical engineer, this course contains a nearly-complete bridge to understanding all of electronic systems. Since mechanical engineers already understand circuits, this single course was within reach, and filled almost every major blind area in my understanding of applied electronics. I would suggest that taking this course is a shortcut to upgrade a mechanical engineer into a mechtronics engineer.

Updated 2026.06 by David Malawey

ENTC 219 – Digital Electronics
Spring 2015
TR 11:10 – 12:25
Thompson 121

Course Description:

Survey of digital applications, number systems, digital logic devices and circuits, sequential logic.

Prerequisite: None

Learning Outcomes/Course Objectives:

Students successfully completing ENTC 219 will:

1. Demonstrate ability to use digital design principles including binary codes, binary, octal, and hexadecimal number systems, and unsigned and signed binary arithmetic.
2. Design, implement, and test combinatorial logic circuits using standard design methods and computer-based design and analysis tools.
3. Design, implement, and test sequential logic circuits using standard design methods and computer-based design and analysis tools.
4. Design, build, test, optimize and document a complete digital system in a small team environment
5. Prepare detailed technical reports on all lab projects including test data necessary for design validation.

Instructor: Joseph A. Morgan, D.E., P.E.
Email: jmorgan@tamu.edu
Phone: 979-575-2^{7} (text and voice)
Office: Fermier 111
Office Hours: MW 5:30 – 7:00 PM

Lab Assistant: James Belcher
Email: jyb93@tamu.edu (Primary)
Phone: **(713) 898-3019** (Secondary)

Textbook: Recommended -- Not Required
Digital Design (3rd Edition or newer), Mano and Mano available from Amazon.com.

Engineering Notebook: Students are encouraged to maintain a professional, bound engineering notebook for this and other courses. The notebook entries will be

made using the original notes taken by the student during the normal class meetings. Each student will extract the fundamental concepts and generic examples from their class notes and enter these into their engineering notebook in a neat and orderly fashion. Approximately 10 pages of the engineering notebook should be reserved for ENTC 219. Notebooks will be turned in to the ESET Program Administrative Coordinator (Emma) the day of an examination prior to the exam. If the student's engineering notebook complies with the ENTC 219 standards, the student will receive up to 5 additional points on their exam based on the content and professional quality of the notebook. No loose papers, etc. should be in the engineering notebook, and no information can be inserted via taped, stapled or other attachment forms to the notebook. No pages can be removed from the engineering notebook, either. Again, this is a resource that the student can create which can then be used to prepare for the examinations if completed properly.

Lab Text: Lab assignments will be provided by the Lab Assistant – see ENTC 219 Lab Syllabus

Wiki Resource: A wiki page is available for additional information for ENTC 219 class and laboratory.

Grading: In this course, homework assignments, laboratory assignments, quizzes, and exams will be used for evaluation of your performance. Students should reference http://student-rules.tamu.edu/rule07 for student responsibility for attendance.

Midterm Exam	25 %
Final Exam	35 %
Laboratory	15 % *
Project	15 % **
Quizzes	10 % - Homework modifies this grade
Total	100 %

*Students will work in teams of two for all lab exercises. One student lab grade will be computed from the even numbered labs and the other student's lab grade will be based on the odd numbered labs. Students are encouraged to work together on all aspects of the lab including the lab report. **To receive a C or better grade in ENTC 219 all lab assignments must be completed and an acceptable lab report submitted.**

**Students will work in groups of two to complete the course project, but each student will be required to design and build his/her own robot. One robot will be used for the drag race and the other for the road race during the competition. Each student will prepare and submit a report on the lab project

Course Grade:

100 – 90 == A
89 – 80 == B

79 – 70 == C
69 – 60 == D
59 or less ==F

Topics:

Week	Chapter	Topic	Video Bytes
1		Number systems, codes, 2's complement	1-8, 10-12, 17
2		Logic gates, truth tables, equations	9, 13-16
3		Boolean Algebra	
4		K-Maps , Reduced SoP, PoS	18
5		Comparators	19
6		Mux, Selectors	20-23
7		Decoders	24-26
8		Midterm Exam	
9		Flip Flops	
10		Counters	
11		State machines and SLD processes	
12		State machines and SLD processes	
13		State machines and SLD processes	
14		IC Logic Families, interfacing, packaging	
	FINAL EXAM	Comprehensive	
	Optional	Xilinx Race of Champions	

The table indicates the planned schedule that will be used in the Spring 2015 semester. Some variation to this schedule may be required. In addition, the course will use "VideoBytes" to augment the lectures so that students can understand a number of basic digital logic/design concepts prior to attending class/lab. All of the VideoBytes are available on the ESETwiki under the ENTC 219 page. The VideoBytes shown in Table 1 should be viewed prior to class and/or lab scheduled for the indicated date. Quizzes will be given on a regular basis and will contain material from the appropriate VideoBytes. Students are also encouraged to ask questions on material covered in the VideoBytes.

Exams: There will be a Midterm exam and a comprehensive Final Exam. The final exam will be comprehensive and will be given at the time specified by the university. Unless there is a university approved absence, no make-up exams will be given. Unexcused absence from an exam or quiz will result in a grade of 0.

Opportunities will be provided to improve the midterm test grade by participating in ESET activities. These opportunities, which will be outside of normal class/lab times, will be announced in class.

For those students who complete the necessary application process and compete in the Xilinx Race of Champions, the final test grade (if higher than the Midterm grade (modified as indicated above) will be used as the Midterm grade.

Homework: Developing good circuit design and analysis techniques requires substantial practice. Therefore, homework problems will be assigned in class. In addition, a number of assignments will be provided as part of the VideoBytes to help the student validate their understanding of the fundamental digital concepts. Being able to work each homework problem is necessary to do well in the class. Homework assignments will be completed using the Multisim design environment. Computer-generated schematics and printouts showing the simulation of the circuit design with tools such as the Word Generator and Logic Analyzer will be required. In addition, all designs should be saved to thumb drive so that students can be called upon to present, explain and defend their design.

Grading

As requested by the course instructor, students will turn one or more of the homework problems (chosen randomly) to be graded. Selected homework assignments may also be used as quizzes. Homework will not normally be returned to students – making copies prior to turning in homework is a good idea. Homework assignments will be due at the **START** of class. Arriving late will result in homework not being accepted.

Laboratory: The lab exercises have been developed to provide the students with the design capabilities and laboratory practices to successfully complete the course project. Each team of two students must successfully design and implement mechanical and electronic subsystems necessary to guide a mobile platform over a random route (road race) and a straight route (drag race). During the final week of classes (reading days), the Xilinx Race of Champions will be conducted. The winning team will be based on the lowest total time to successfully navigate both routes. This competition is optional, but participation is required as one of the requirements for Final Exam grade substitution for Midterm (see above).

Laboratory preparation, performance, and documentation requirements can be found in the ENTC 219 laboratory syllabus.

Quizzes: Random quizzes will be given in class and may be given in lab sessions. In-class quizzes on a given subject may be given before the due date of homework covering that subject. Unless there is a university approved absence, no makeup quizzes will be given and the student will receive a 0 grade for any missed quizzes. Quizzes will be given at the START of class. Arriving after the quiz begins will result in student not taking the quiz. Quizzes will include information contained in the VideoBytes which should be viewed prior to the class/lab date shown in Table 1.

Tardiness: Both homework and lab assignments are due at the beginning of class on the date assigned. Late assignments will not be accepted without either a valid university approved excuse or prior consent from the professor. (The lab teaching assistant does not have the authority to excuse a late assignment or an absence.)

Attendance: Attendance is strongly required, but not mandatory. Generally speaking, a student who misses a number of classes does not achieve his/her potential level of performance. Students should reference http://student-rules.tamu.edu/rule07 for student responsibility for attendance.

Students with Disabilities: The Americans with Disabilities Act (ADA) is a federal anti-discrimination statute that provides comprehensive civil rights protection for persons with disabilities. Among other things, this legislation requires that all students with disabilities be guaranteed a learning environment that provides for reasonable accommodation of their disabilities. If you believe you have a disability requiring an accommodation, please contact the Department of Student Life, Services for Students with Disabilities, in Cain Hall or call 845-1637.

Honor Code: "The Aggie Honor Code:
"An Aggie does not lie, cheat, or steal or tolerate those who do." Academic Integrity is essential for the university environment of academic inquiry and learning and the accurate recognition of each student's achievement in that endeavor. Collaboration and information sharing are characteristics of a university education; however, academic integrity is violated when student conduct involves dishonesty or ways that give a student an unfair advantage. Academic dishonesty includes the commission of any of the following acts: cheating, fabrication, falsification, multiple submissions, plagiarism, complicity, abuse and misuse of access and unauthorized access, violation of departmental or college rules and university rules on research. (This listing is not exhaustive.) Clarification of these terms is at: www.tamu.edu/aggiehonor/definitions.php . Students may report violations of the honor code to the Aggie Honor System Office (AHSO), www.tamu.edu/aggiehonor or to the instructor. The report procedure is described at: www.tamu.edu/aggiehonor/reporting.php . Self-reporting is encouraged and may be considered a mitigating circumstance in the sanctioning phase of a particular case. The honor code process is described at: www.tamu.edu/aggiehonor/process.php and www.tamu.edu/aggiehonor .
The following is a highlighting of some of your student responsibilities:

* You are responsible to be fully acquainted with and to comply with the Aggie Honor Code, Honor Code Rules and Procedures.
* You are responsible to seek clarification from the instructor if you are in doubt whether an action constitutes academic dishonesty.
* You are to actively promote academic integrity. Apathy or acquiescence in the presence of academic dishonesty is not a neutral act-failure to confront and deter it will reinforce, perpetuate and enlarge the scope of such misconduct.

ESET 219 Notes

Pointers & Arrays

`int a[10]` – defines array of size 10, containing elements a[0] thru a[9]

`int *pa` – declares pa, a pointer to an integer

`pa = &a[0]` – sets pa to point to the zeroth element of a, so pa contains the address of a[0]

`x = pa` – ~~an~~ copies contents of a[0] into x

`*(pa+1)` – refers to the contents of a[1]

printf

`printf("%4.1f", fahr)`

- prints the variable stored as fahr
- goes in a space at least 4 characters wide
- has 1 digit after the decimal
- is a floating point number

Symbolic Constants

With #define, the compiler will replace all unquoted occurrences of a name by the corresponding string

`#define UPPER 300 /* upper limit */`

- here, UPPER is a symbolic name or symbolic constant

Constants

Syntax	meaning
037	number 31 in octal
0x1f	number 31 in hexadecimal
'O'	the character zero which is 48 in ASCII - if we write the character in single quotes then
'x'	- produces the numeric value of the letter x in the machine's character set (probably ASCII)
"x"	- a character string containing the letter x and a \0

operators

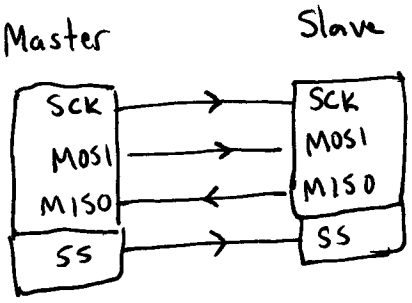
relational operators > >= < <=

equality operators == !=

logical connectives: && (and)
|| (or)

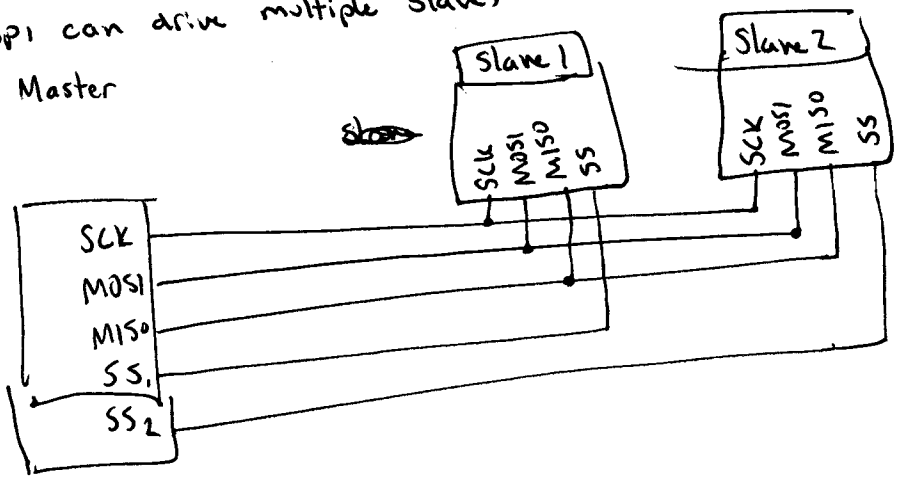
SPI (Synchronous) Serial Peripheral Interface

- can have a shift register as receiving hardware
  - (cheaper & simpler than asynchronous hardware like UART)



CLK or SCK = Synchronous clock
MOSI = master out, slave in
MISO = master in, slave out
SS = slave select

- SS is usually active @ low voltage
- SPI can drive multiple slaves



C programming

#include – directs the C preprocessor to scan the included file as an input before continuing with the rest of the file

#ifndef <TOKEN> – checks whether the given token has been #defined earlier and if not, includes the code all the way to #endif

extern – applies to c variables and functions. Only functions are automatically treated as extern.

declaration: does not allocate memory but declares the data type.
definition: allocates memory for the variable/function. defining a function means writing the body of the function

- declaration can be done more than once
- definition can only be done 1x

Example:

extern int var – declares ~~and defines~~ the integer type of variable named var WITHOUT defining it.

int var – declares AND defines the variable

extern int var = 0; – is taken as a declaration AND definition since it's declared with its initialization

Ternary operator

variable > 9600 ? 4 : 2 ; if line A is true then 4 is returned, otherwise 2 is returned

Breakdown of

String Parsing function

```
Void   parse_data (void) {
```

- parse_data is the function
- it returns nothing: void
- it requires no input argument: (void)

```
int delimiter[4] = {0};
```

- declares a 4 element array and stores zeros

```
:
IN1 = 0;
```

- a declared variable without a type is assumed to be of type INT

```
for (i=0; i<sizeof(recstring); ++i){
  if (recstring[i] == ',') {

    delimiter[j] = i;
    ++j;
```

- Set i to zero, check the condition, run whats in the loop, then increment i
- check if the i^{th} character in recstring is ","
- Set the j^{th} character of delimiter to the value in i. now delimiter knows where in the recstring the comma occurs

```
command = recstring[delimiter[0]+1]
```

- the character after the 1st delimiter is stored in "command"

```
for (i= delimiter[1]+1 ; i< delimiter[2]; ++i)
  IN1 = IN1 *10 + recstring[i] - '0';
```

- for as many spaces there are between delimiters
- move IN1 digit over to tens and place the i^{th} character in receivestring in the ones, then subtract ASCII character "0" in 8 bits
- now all digits from 2nd delimiter to 3rd delimiter is IN1

NESI UART Function Breakdown

return var ↓

Static BOOLEAN sendFrom(const Byte* startOfData, Uint16 bytesToSend)

↑ Type   ↑ function

function name: sendFrom()

name of variable returned: Boolean

type of variable returned: static

Parameters: startOfData = pointer to string of bytes to send

bytesToSend = number of bytes to send from the byte array

* Notes: return – terminates the execution of a function.
can also return a variable to the calling function

void – means that a function doesnt return anything

const – a qualifier. it's added to a declaration
and means the something is not modifiable

pointer – a variable whose value is the address of
another variable (direct address of memory location)

- designated as a pointer by *
- must have a type such as: int, double, float, char
  - ~ type specifies the type of variable being pointed to

```
int var = 20;    /* actual variable declaration */
int *ip;         /* pointer variable declaration */

ip = &var;       /* store address of var in pointer variable */
```

& – an operator that denotes an address in memory

for loop syntax:
```
for ( init; condition; increment)
{
  statement(s)
}
```

~ init: executed only once, condition is evaluated, loop executes, then
updates are made ... the increment statement then repeat

Assignments Section

3 Bit Comparator Logic Table

A2	A1	A0	B2	B1	B0	G	E	L
0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	1	0
0	0	0	0	1	0	1	1	0
0	0	0	0	1	1	1	1	0
0	0	0	1	0	0	1	1	0
0	0	0	1	0	1	1	1	0
0	0	0	1	1	0	1	1	0
0	0	0	1	1	1	1	1	0
0	0	1	0	0	0	0	1	1
0	0	1	0	0	1	0	0	0
0	0	1	0	1	0	1	1	0
0	0	1	0	1	1	1	1	0
0	0	1	1	0	0	1	1	0
0	0	1	1	0	1	1	1	0
0	0	1	1	1	0	1	1	0
0	0	1	1	1	1	1	1	0
0	1	0	0	0	0	0	1	1
0	1	0	0	0	1	0	1	1
0	1	0	0	1	0	0	0	0
0	1	0	0	1	1	1	1	0
0	1	0	1	0	0	1	1	0
0	1	0	1	0	1	1	1	0
0	1	0	1	1	0	1	1	0
0	1	0	1	1	1	1	1	0
0	1	1	0	0	0	0	1	1
0	1	1	0	0	1	0	1	1
0	1	1	0	1	0	0	1	1
0	1	1	0	1	1	0	0	0
0	1	1	1	0	0	1	1	0
0	1	1	1	0	1	1	1	0
0	1	1	1	1	0	1	1	0
0	1	1	1	1	1	1	1	0
1	0	0	0	0	0	0	1	1
1	0	0	0	0	1	0	1	1
1	0	0	0	1	0	0	1	1
1	0	0	0	1	1	0	1	0
1	0	0	1	0	0	0	0	0
1	0	0	1	0	1	1	1	0
1	0	0	1	1	0	1	1	0
1	0	0	1	1	1	1	1	0
1	0	1	0	0	0	0	1	1
1	0	1	0	0	1	0	1	1
1	0	1	0	1	0	0	1	1
1	0	1	0	1	1	0	1	1
1	0	1	1	0	0	0	1	1
1	0	1	1	0	1	0	0	0
1	0	1	1	1	0	1	1	0
1	0	1	1	1	1	1	1	0
1	1	0	0	0	0	0	1	1
1	1	0	0	0	1	0	1	1
1	1	0	0	1	0	0	1	1
1	1	0	0	1	1	0	1	1
1	1	0	1	0	0	0	1	1
1	1	0	1	0	1	0	1	1
1	1	0	1	1	0	0	0	0
1	1	0	1	1	1	1	1	0
1	1	1	0	0	0	0	1	1
1	1	1	0	0	1	0	1	1
1	1	1	0	1	0	0	1	1
1	1	1	0	1	1	0	1	1
1	1	1	1	0	0	0	1	1
1	1	1	1	0	1	0	1	1
1	1	1	1	1	0	0	1	1
1	1	1	1	1	1	0	0	0

MAX (POS) "0"

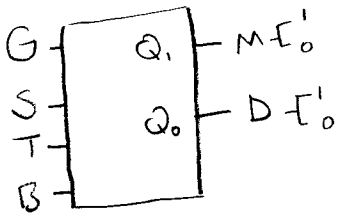
$E = (\quad)(\quad)$

"0"

$E = (A_2 + A_1 + A_0 + B_2 + B_1 + B_0)($

$(A_2 + A_1 + A_0' + B_2 + B_1 + B_0')$

Garage Door Opener



Garage button
- 0 Not pressed
- 1 pressed

Safety curtain
- 0 unobstructed
- 1 obstructed

Top stop
- 1 reached
- 0 no

Bottom stop
- 1 reached
- 0 not

Moving
- 1 yes
- 0 no

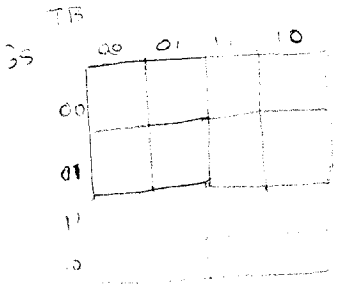
Direction
- 1 up
- 0 down

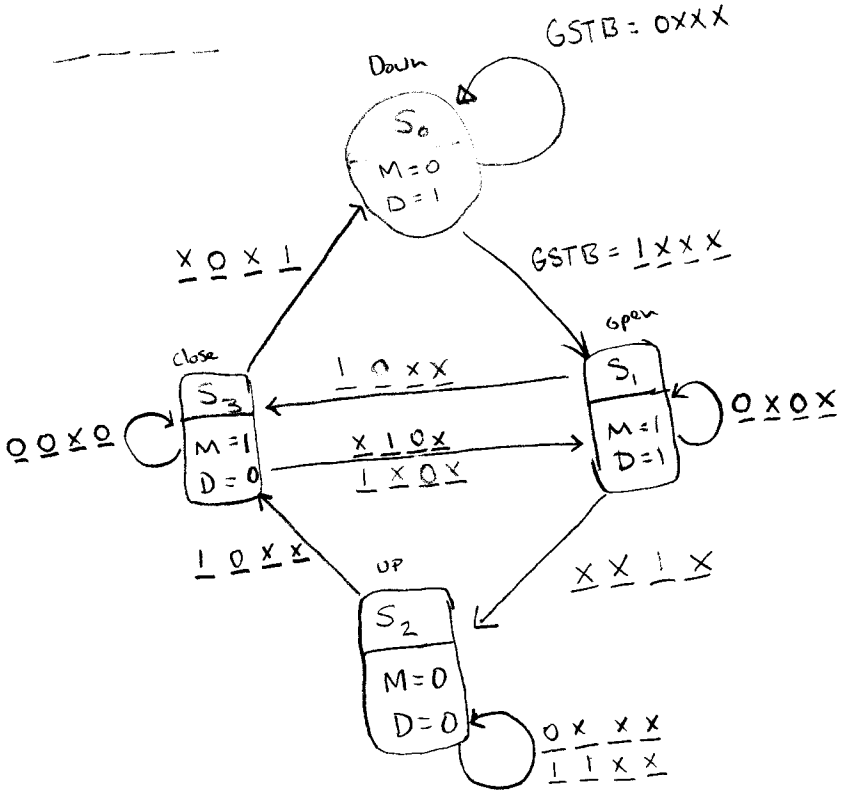
S_0 = Down

S_1 = Door ↑

S_2 = UP

S_3 = Door ↓



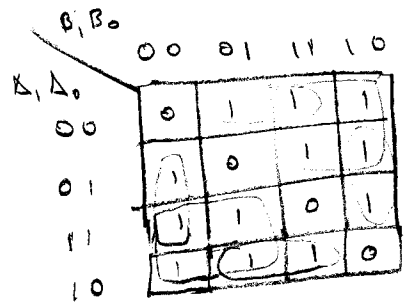


Current State		Input	Next State	
	M D	GSTB	M D	
5	01	1XXX	11	
3	01	0XXX	01	
4	11	0X0X	11	
4	11	10XX	10	
8	11	XX1X	00	
8	00	0XXX	00	
4	00	11XX	00	
4	00	10XX	10	
2	10	00X0	10	
2	10			

Design a 2-bit comparator with AND & OR gates

E

K-map



$E = (A_1 B_1') + (A_0 B_1' B_0') + (B_0 A_1 A_0') + (B_1 A_1') + (A_1' A_0' B_0') + (A_0 B_1 B_0')$

"A Karnaugh map (K-map) is a graphical tool to simplify and minimize boolean algebra expressions. This process reduces the number of logic gates in digital circuit design, as it reduces the number of logic gates and inputs required, leading to more efficient circuits."

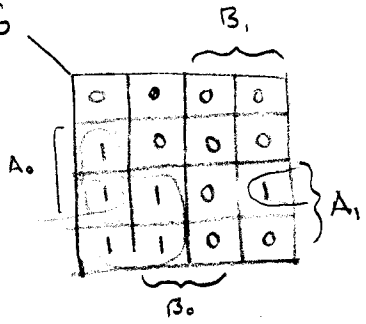
Truth Table

A_1	A_0	B_1	B_0	G	E	L
0	0	0	0	0	1	
0	0	0	1	0	0	
0	0	1	0	0	0	
0	0	1	1	0	0	
0	1	0	0	1	0	
0	1	0	1	0	1	
0	1	1	0	0	0	
0	1	1	1	0	0	
1	0	0	0	1	0	
1	0	0	1	1	0	
1	0	1	0	0	1	
1	0	1	1	0	0	
1	1	0	0	1	0	
1	1	0	1	1	0	
1	1	1	0	1	0	
1	1	1	1	0	1	

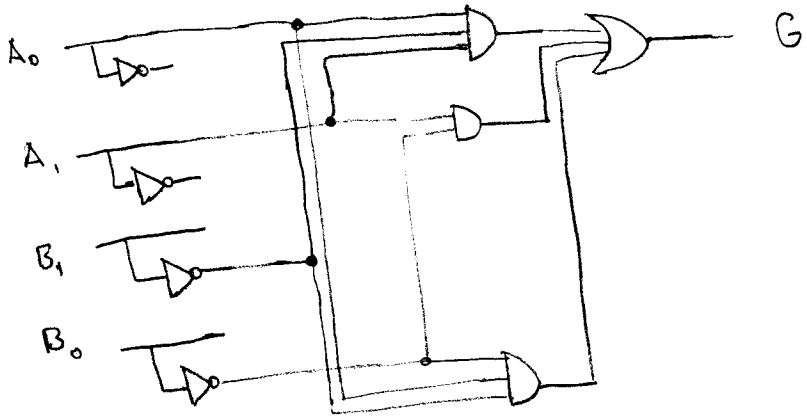
$G = (A_1' A_0 B_1' B_0') + (A_1 A_0' B_1' B_0') + (A_1 A_0' B_1 B_0') + (A_1 A_0 B_1' B_0') + (A_1 A_0 B_1 B_0')$

$E = (A_1 A_0 B_1 B_0) + (A_1 A_0' B_1 B_0') + (A_1' A_0 B_1' B_0) + (A_0 B_1 B_0')$

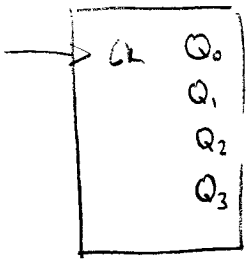
K-map

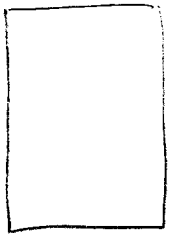


$G = (A_1 \overline{B_1}) + (A_0 \overline{B_1}\,\overline{B_0}) + (\overline{B_0} A_1 A_0)$



PWM Preliminary Design



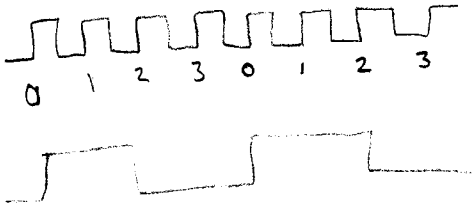


6 bits

5kHz out

5 kHz = $\frac{1}{2^6}$ (clock)

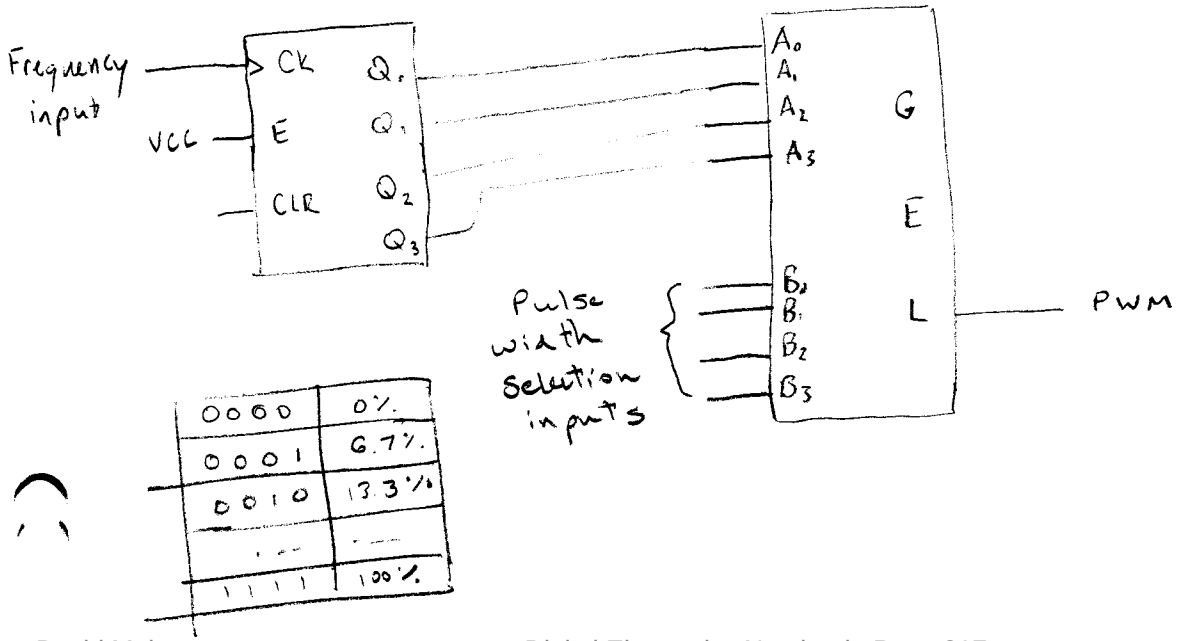
clock = 320 kHz



1, 2 ON

3, 4 OFF

- if counter output is > x, then signal OFF
- else, OFF
- x is signal from



0000	0%
0001	6.7%
0010	13.3%
...	...
1111	100%

ESET 219

Lab1 Working with Multisim

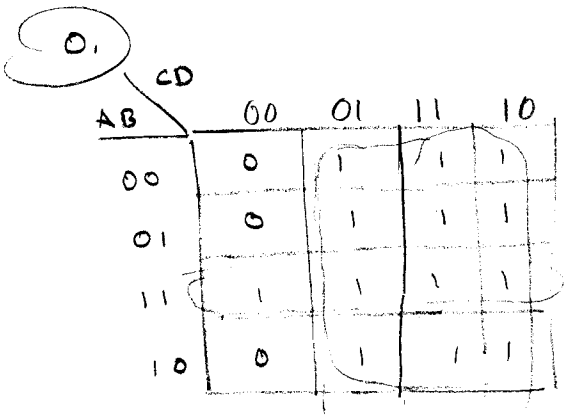
Adding components

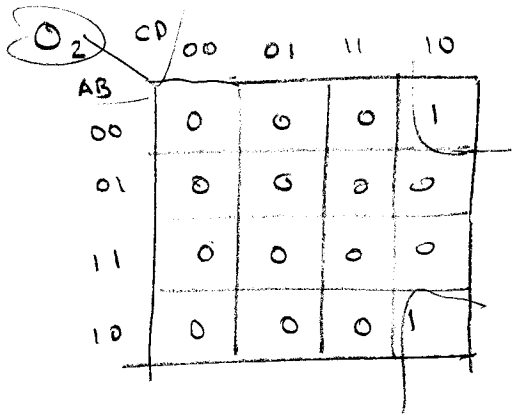
Group → misc digital → TIL (this gets you logic gates)

Circuit 3

A	B	C	D	O_1	O_2
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	1	1
0	0	1	1	1	0
0	1	0	0	0	0
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	1	1	0

A	B	C	D	O_1	O_2
1	0	0	0	0	0
1	0	0	1	1	0
1	0	1	0	1	1
1	0	1	1	1	0
1	1	0	0	1	0
1	1	0	1	1	0
1	1	1	0	1	0
1	1	1	1	1	0





$O_1 = AB + \overline{C'D'}$

$O_2 = B' \cdot C \cdot D'$

Lecture Notes

ESET 219

Videos

Overview Lessons

1 ✓

Codes

2 ✓ ASCII codes

3 ✓ BCD & 7 segment display codes

Systems

4 ✓ Decimal # system

5 ✓ binary # system

6

7

8

Basic Logic Gates

9 ✓ Basic Logic Gates

Two's Comp

10 ✓ Twos Compliment. With N bits, range is (-2^{n-1}) to $(2^{n-1}-1)$

11 ✓ 2's comp addition

12 ✓ 2's comp subtraction

Min/max term eqns

13 ✓ generating MINTERM eqns

14 ✓ generating MAXTERM eqns

15 ✓ Minterm to Circuit

16 ✓ Maxterma to Circuit

Octal & Hexadecimal

17 ✓ octal & hex

18 ✓ Kmaps & reduced SOP & POS equations

19 ✓ Comparators & cascading comparators

20

21

22

23

24

25

26

L2

ASCII codes

- has 8 bits, 256 codes
- "A" = 0100 0001

"a" = 0110 0001

"b" = 0110 0010

"3" = 0011 0011

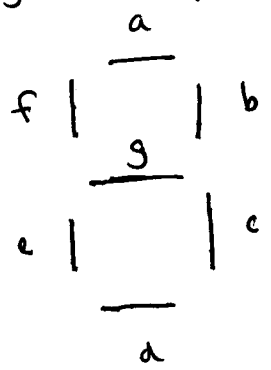
"7" = 0011 0111

L3 Binary Coded Decimal & 7-Segment

0-9 digits (10), so in binary n=4

Decimal	BCD
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

7-seg display



1	0	1	1	0	1	1
a	b	c	d	e	f	g

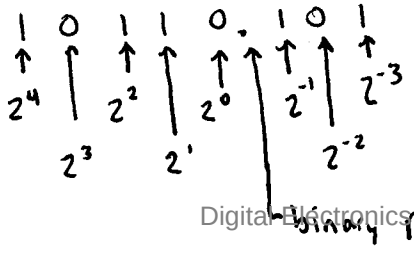
ex, 5 =

L4 Decimal # system: no notes

L5 Binary # System

n bits

c combinations



L6

L7

L10, L11

2's compliment Addition

n = 5 bits C = 15, range = -16 to 15

```
 0 1 1 1
 0 0 1 0 1   5   msb = 0 means positive
+0 0 1 1 1  +7
-----------  --
 0 1 1 0 0   12
```

↳ pos + pos = pos, no (O/F)

```
  1 1 1 1
 0 1 0 1 1
+0 0 1 1 1
-----------
 1 0 0 1 0   pos + pos = neg, (O/F)
```

- pos + neg has Never O/F
- final carry out digit Never has a value in 2's complement

L12

2's complement Subtraction

- A + (-B) is the method used
- (-B) is 2's complement of B

```
  00101          1
- 00111   ⇒   0 0 1 0 1
              1 1 0 0 1
              ---------
              1 1 1 1 0
```

↳ pos + neg = no overflow

```
  10011          1 1
- 11010   ⇒   1 0 0 1 1    -13
              0 0 1 0 0  -  -6
              ---------    ---
              1 1 0 0 1    -7
```

neg + pos = no O/F

L17 Octal & Hex

Octal = shorthand for Binary

use groups of 3

001 011 011 → 7 segment display code ⇒ 5

1 3 3

* Zeros added to finish octal notation

101 111 011 011 . 100 100

5 7 3 3 . 4 4 octal

- Hexadecimal

(base-16) 0~9, A~F

* 0001011 1100 1111 . 1011 1000

1 7 9 F . B 8

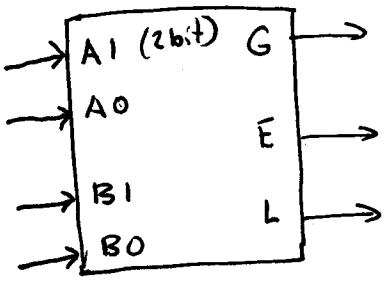
ASCII	OCTAL	HEX
"A", 01,00 0,001	101	41
OCTAL 753	111 101 011	1 E B

HEX AF13 ⇒

L19

Comparators & cascading comparators

A > B (G)
A = B (E)
A < B (L)



G

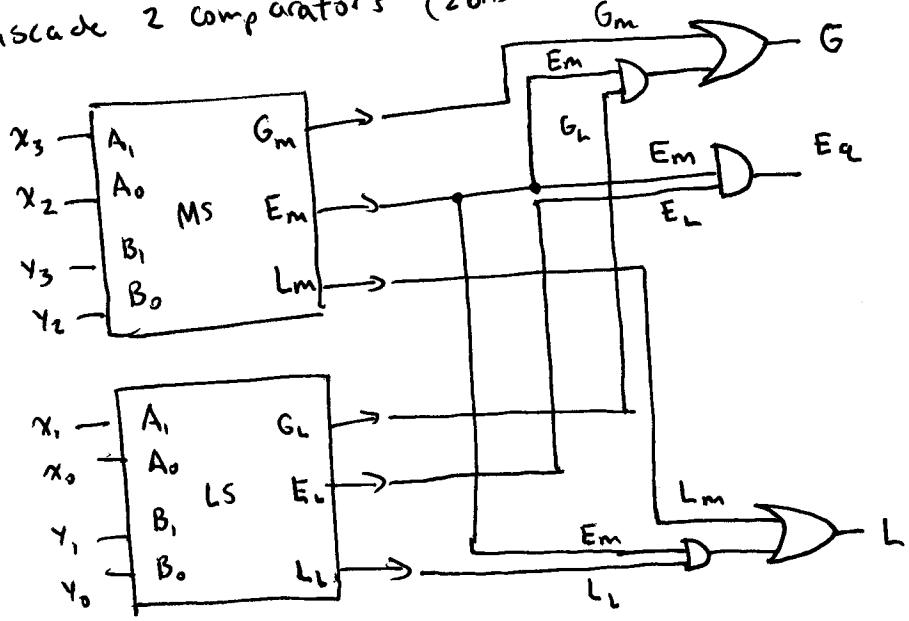
A_1A_0 \ B_1B_0	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	0	1
10	1	1	0	0

L = inverse of G

E

	00	01	11	10
00	1	0	0	0
01	0	1	0	0
11	0	0	1	0
10	0	0	0	1

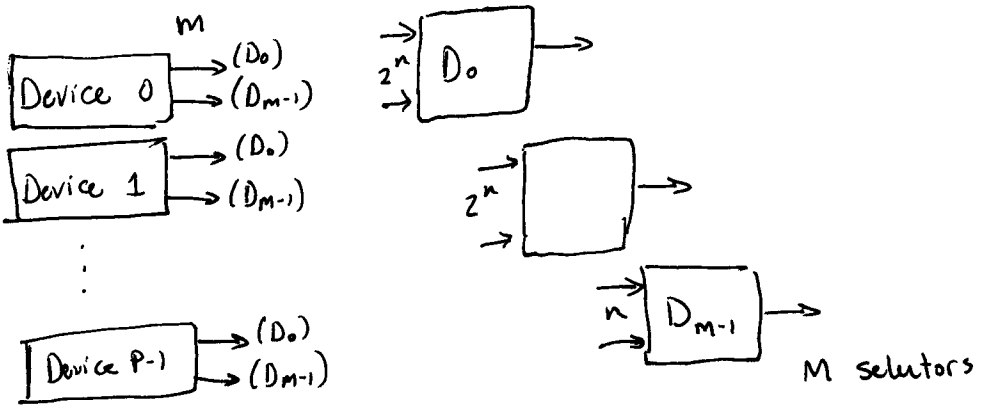
Cascade 2 comparators (2bits)



L22

Parallelizing selectors

- use multiple selectors to choose M outputs from 1 of P devices



P devices

Ex.) to select 13 outputs from one of 27 devices

13 selectors are needed

27 devices are needed

Selectors will have at least P inputs. $\Rightarrow 2^n \geq P$ P=

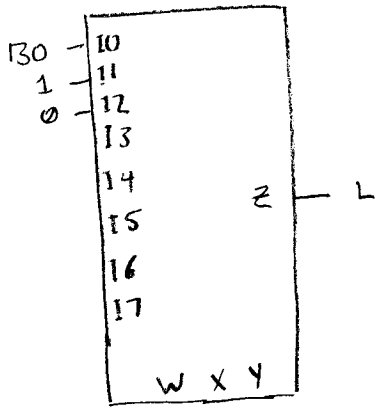
$P=27,\ 2^5 = 32$

- So, 5 inputs will be unused

What code will select the last device? (n-digits)

L23 Using Selectors to implement CLC design

Associate the



- Data inputs: 0, 1 LSB, LSB'
- one selector per truth table output will be required
- use selector with one less select input than no. inputs in truth table
- Connect the MS bits of TT to the select ~~of~~ lines of selector

L2

ASCII codes

- has 8 bits, 256 codes
- "A" = 0100 0001

"a" = 0110 0001

"b" = 0110 0010

"3" = 0011 0011

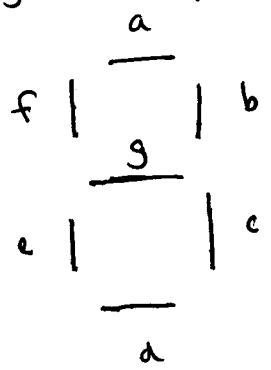
"7" = 0011 0111

L3 Binary Coded Decimal & 7-Segment

0-9 digits (10), so in binary n=4

Decimal	BCD
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

7-seg display



1	0	1	1	0	1	1
a	b	c	d	e	f	g

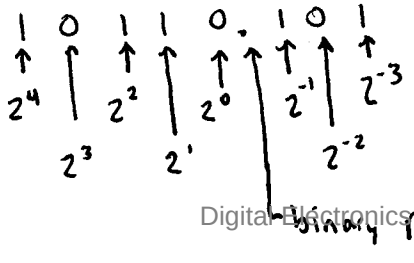
ex, 5 =

L4 Decimal # system: no notes

L5 Binary # system

n bits

c combinations



L6

L7

L10, L11

2's compliment Addition

n = 5 bits C = 15, range = -16 to 15

```
 0 1 1 1
 0 0 1 0 1   5   msb = 0 means positive
+0 0 1 1 1  +7
-----------  --
 0 1 1 0 0   12
```

↳ pos + pos = pos, no (O/F)

```
  1 1 1 1
 0 1 0 1 1
+0 0 1 1 1
-----------
 1 0 0 1 0   pos + pos = neg, (O/F)
```

- pos + neg has Never O/F
- final carry out digit Never has a value in 2's complement

L12

2's complement Subtraction

- A + (-B) is the method used
- (-B) is 2's complement of B

```
  00101          1
- 00111   ⇒   0 0 1 0 1
              1 1 0 0 1
              ---------
              1 1 1 1 0
```

↳ pos + neg = no overflow

```
  10011          1 1
- 11010   ⇒   1 0 0 1 1    -13
              0 0 1 0 0  -  -6
              ---------    ---
              1 1 0 0 1    -7
```

neg + pos = no O/F

L17 Octal & Hex

Octal = shorthand for Binary

use groups of 3

001 011 011 → 7 segment display code ⇒ 5

1 3 3

101 111 011 011 . 100 100 octal

5 7 3 3 . 4 4

* Zeros added to finish octal notation

- Hexadecimal

(base-16) 0~9, A~F

* 0001011 1100 1111 . 1011 1000

1 7 9 F . B 8

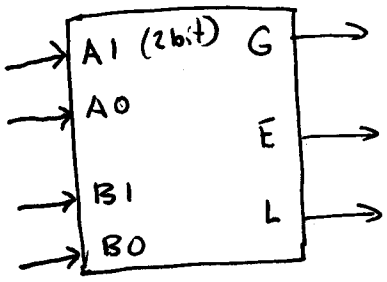
ASCII	OCTAL	HEX
"A", 01,00 0,001	101	41
OCTAL 753	111 101 011	1 E B

HEX AF13 ⇒

L19

Comparators & cascading comparators

A > B (G)
A = B (E)
A < B (L)



G

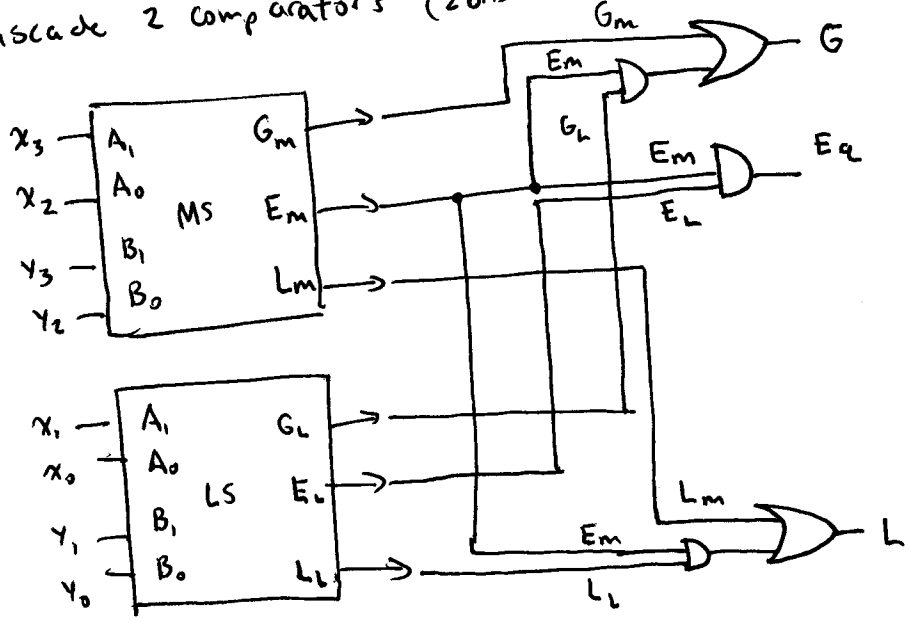
A_1A_0 \ B_1B_0	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	0	1
10	1	1	0	0

L = inverse of G

E

	00	01	11	10
00	1	0	0	0
01	0	1	0	0
11	0	0	1	0
10	0	0	0	1

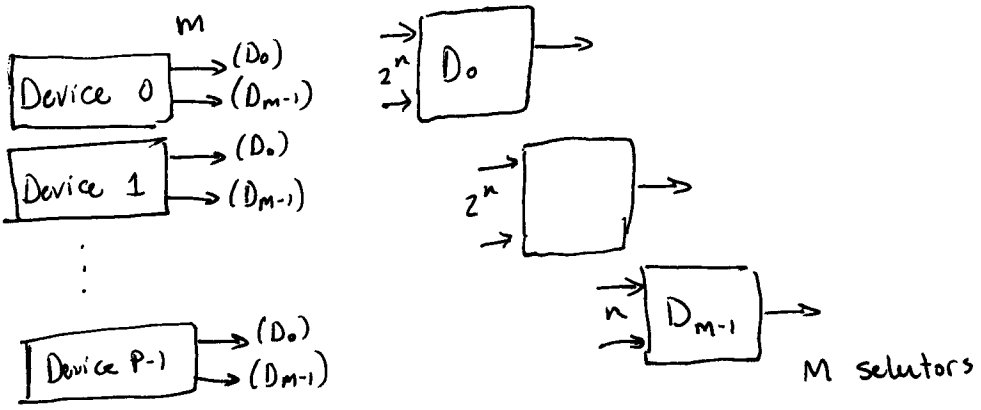
Cascade 2 comparators (2bits)



L22

Parallelizing selectors

- use multiple selectors to choose M outputs from 1 of P devices



P devices

Ex.) to select 13 outputs from one of 27 devices

13 selectors are needed

27 devices are needed

Selectors will have at least P inputs. $\Rightarrow 2^n \geq P$ P=

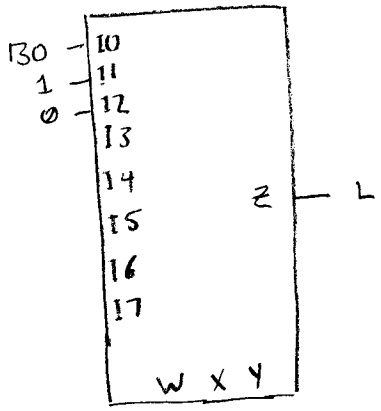
$P=27,\ 2^5 = 32$

- So, 5 inputs will be unused

What code will select the last device? (n-digits)

L23 Using Selectors to implement CLC design

Associate the



- Data inputs: 0, 1 LSB, LSB'
- one selector per truth table output will be required
- use selector with one less select input than no. inputs in truth table
- Connect the MS bits of TT to the select ~~of~~ lines of selector

Notes from Lab Exercises

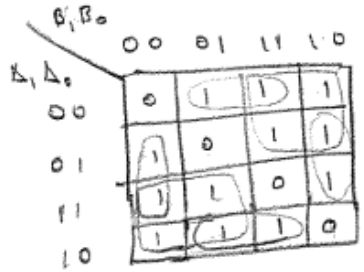
David Malawey
2015/07/29
MEEN 685 (& ESET219)

2 Bit Comparator Design

Design a 2-bit comparator with AND & OR gates

E

K-map



$E = (A_1 B_1') + (A_0 B_1' B_0') + (B_0 A_1 A_0') + (B_1 A_1') + (A_1' A_0' B_0') + (A_0 B_1 B_0')$

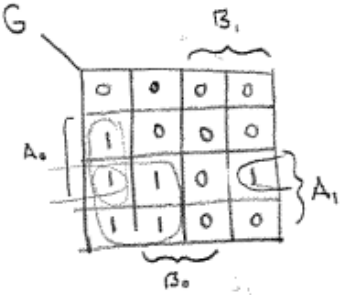
Truth Table

$A_1 A_0 B_1 B_0$	G	E	L
0 0 0 0	0	1	
0 0 0 1	0	0	
0 0 1 0	0	0	
0 0 1 1	0	0	
0 1 0 0	1	0	
0 1 0 1	0	1	
0 1 1 0	0	0	
0 1 1 1	0	0	
1 0 0 0	1	0	
1 0 0 1	1	0	
1 0 1 0	0	1	
1 0 1 1	0	0	
1 1 0 0	1	0	
1 1 0 1	1	0	
1 1 1 0	1	0	
1 1 1 1	0	1	

$G = (A_1' A_0 B_1' B_0') + (A_1 A_0' B_1' B_0') + (A_1 A_0' B_1' B_0) + (A_1 A_0 B_1' B_0') + (A_1 A_0 B_1 B_0') + (A_1 A_0 B_1' B_0)$

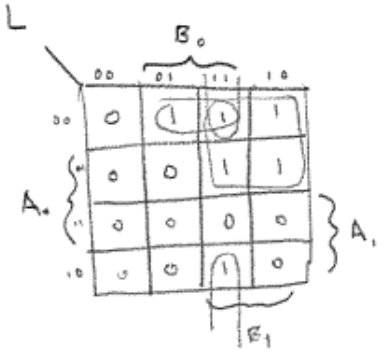
$E = (A_1 A_0 B_1 B_0) + (A_1 A_0' B_1 B_0') + (A_1' A_0 B_1' B_0) + (A_1' A_0' B_1 B_0') + (A_1' A_0' B_1' B_0')$

K-map



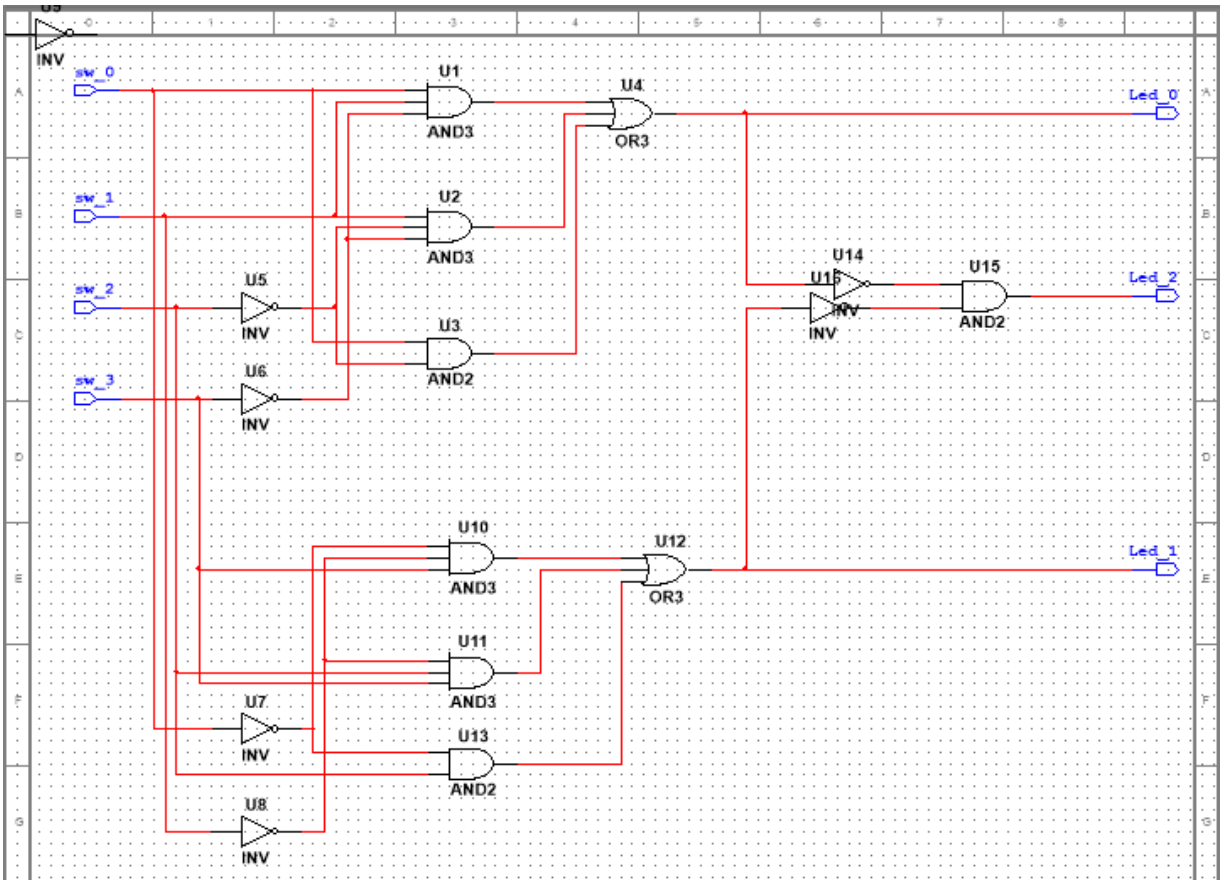
$G = (A_1 \overline{B_1}) + (A_0 \overline{B_1}\,\overline{B_0}) + (\overline{B_0} A_1 A_0)$

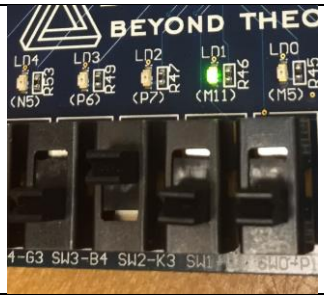
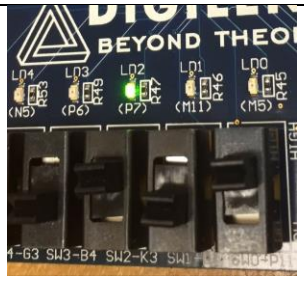
K-map



$L = (B_1 \overline{A_1}) + (B_0 \overline{A_0}\,\overline{A_1}) + (\overline{A_0} B_1 B_0)$

① ②



		
LED1 indicates bits 0,1 are less than bits 2,3	LED0 indicates bits 0,1 are greater than bits 2,3	LED2 indicates the bits are equal

David Malawey
8/3/2015
MEEN 685 / ESET219
Pulse Width Modulation Circuit

In this design a clock signal is fed into a binary 4 bit counter and the output bits of this counter are compared with a 4 bit constant signal that is chosen by the user. This configuration results in an active High less-than signal which has a constant frequency and a selectable width.

Using a 4 bit counter, the circuit frequency is $1/(2^4)$ * clock frequency.
At 50Hz clock signal, the PWM frequency is 3.125Hz.

Using the Pulse Width Selection bits, the PWM signal can be discretized into 15 widths.
Table1 shows the inputs and resulting output power:

Table 1

Input	PWM power
0000	6.67%
0001	13.3%
0010	20.0%
…	…
1111	100%

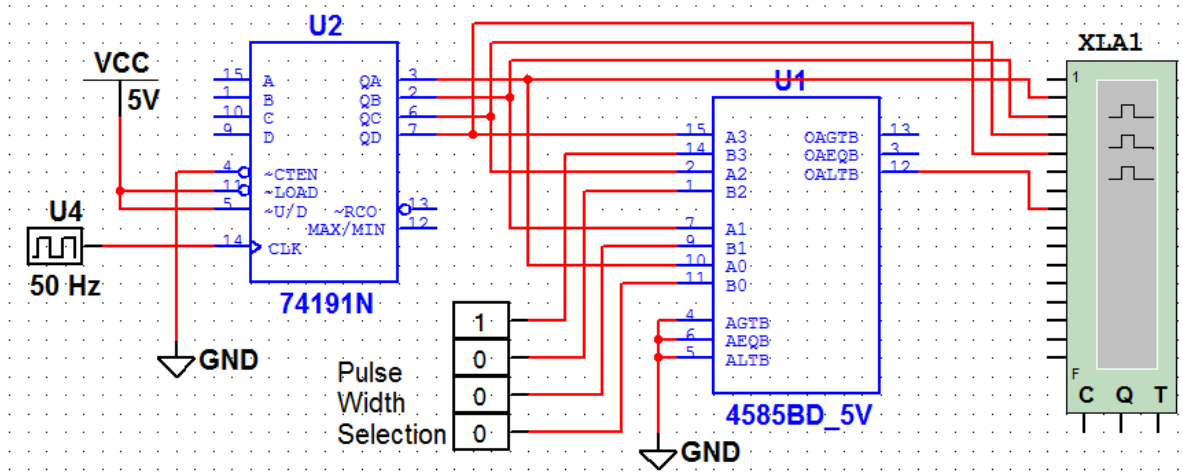


Figure 1. Circuit Diagram

The following figures show the changing counter bits and the output of the circuit (ALTB) meaning A is less than B. This signal can be used to drive an electric motor, but not directly. Since it is a digital signal it cannot deliver much power. This signal could be used to drive a relay which carries power to a motor. A low-pass filter may be necessary to smooth the final output.

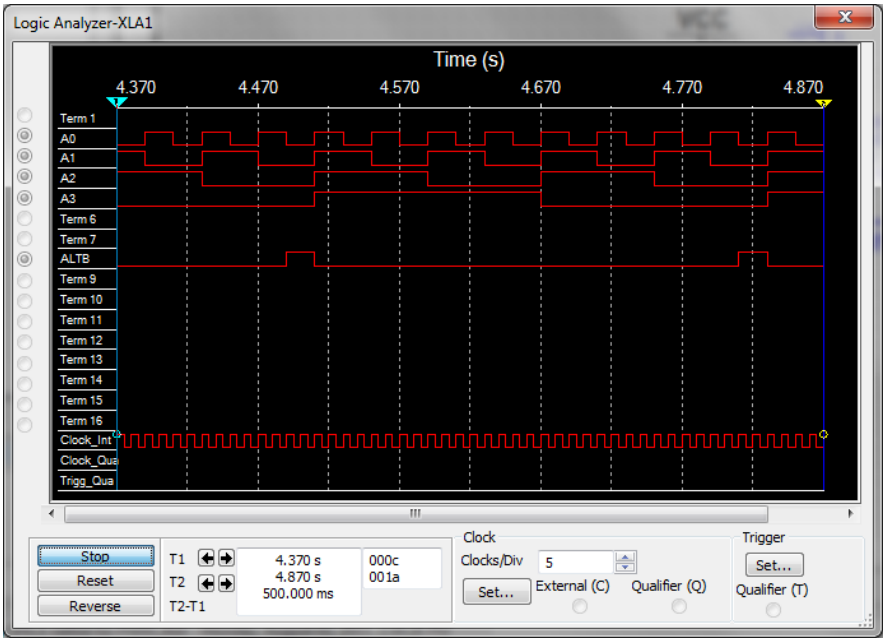


Figure 2. PWM Input = 0000

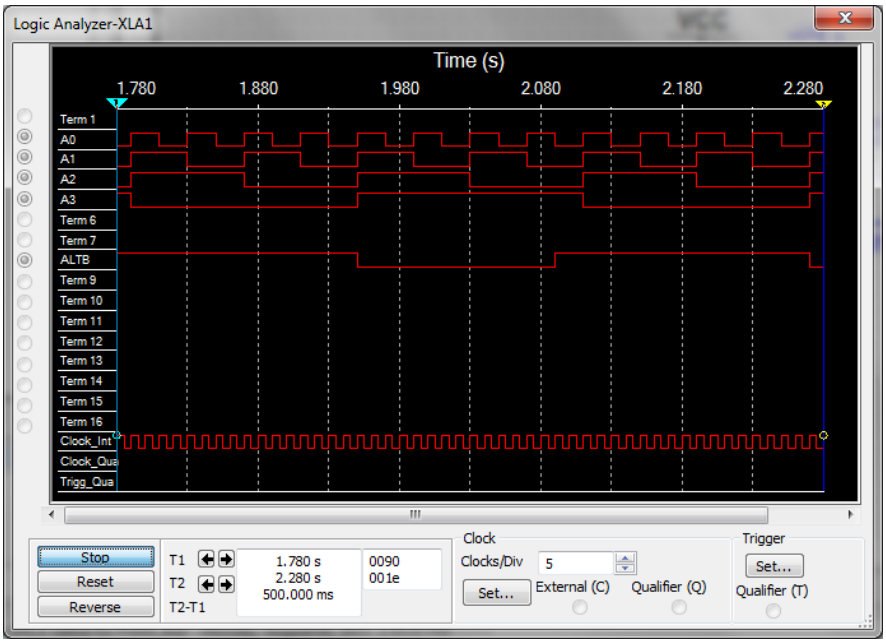


Figure 3. PWM input = 1000

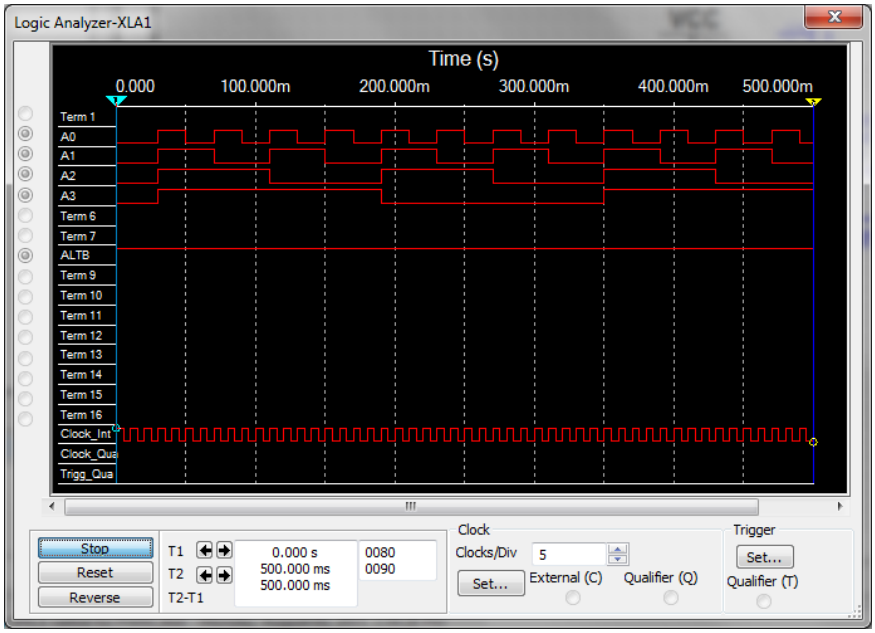
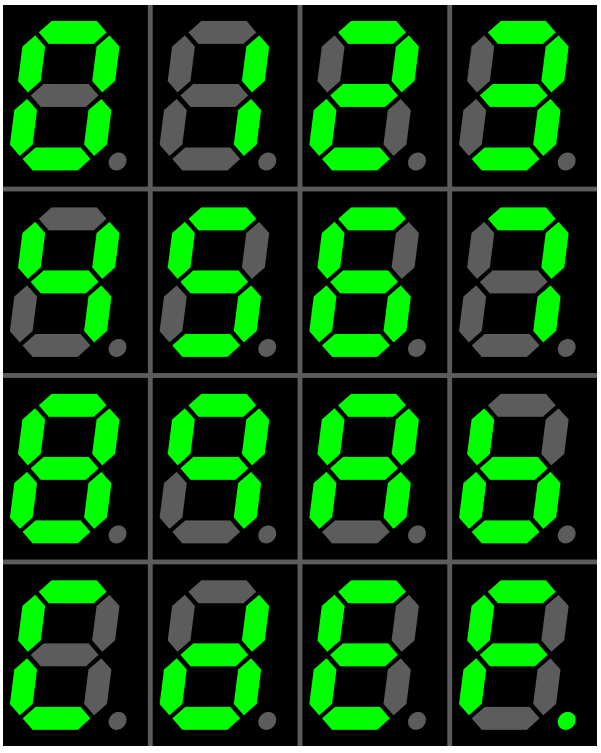


Figure 4. PWM input = 1111

Lecture 15
Seven segment display

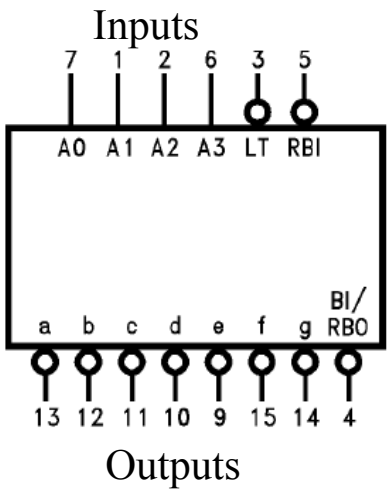
- Decoding BCD to seven segment
- The seven-segment LED display

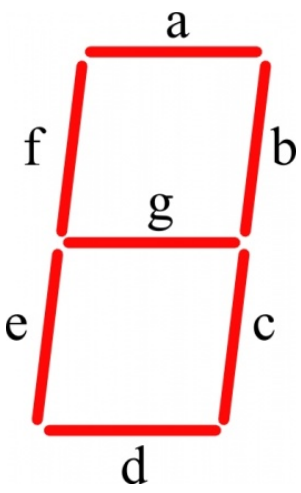


http://commons.wikimedia.org/wiki/File:Seven_segment_display-gallery.png

7447 7-segment decoder

- Input: 4 bits of BCD
- Output > 1001 not useful
- Output 7 segment control
- Outputs inverted!





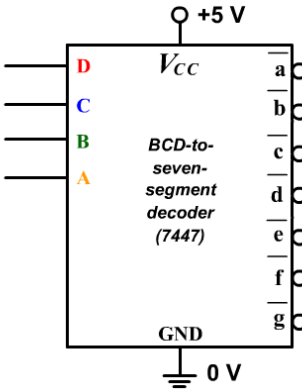
http://www.easyvectors.com/browse/other/seven-segment-display-clip-art

BCD inputs				segment outputs							display
D	C	B	A	a	b	c	d	e	f	g	
0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	1	0	1	1	0	0	0	0	1
0	0	1	0	1	1	0	1	1	0	1	2
0	0	1	1	1	1	1	1	0	0	1	3
0	1	0	0	0	1	1	0	0	1	1	4
0	1	0	1	1	0	1	1	0	1	1	5
0	1	1	0	0	0	1	1	1	1	1	6
0	1	1	1	1	1	1	0	0	0	0	7
1	0	0	0	1	1	1	1	1	1	1	8
1	0	0	1	1	1	1	0	0	1	1	9

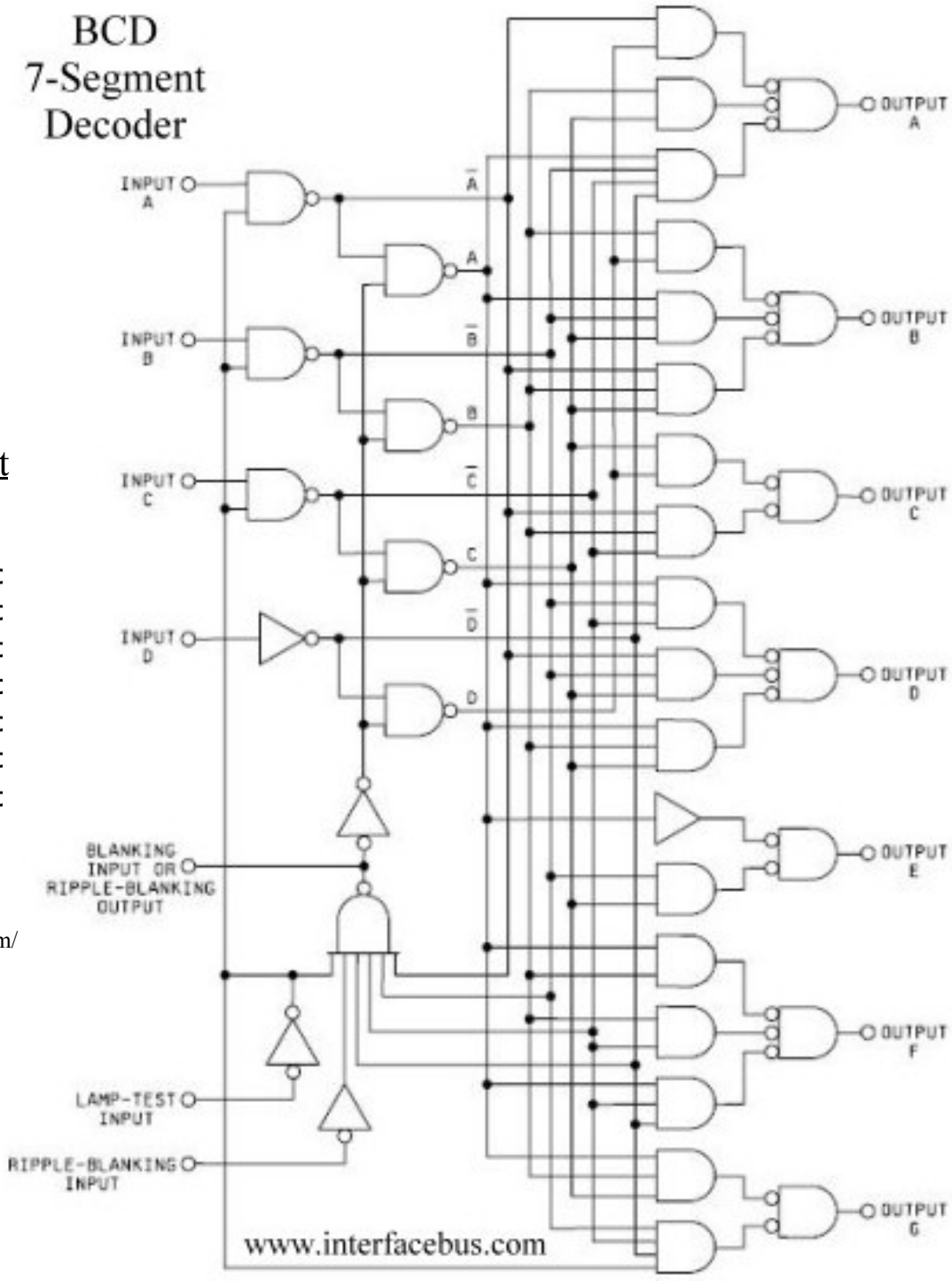
http://www.doctronics.co.uk/4511.htm#truth_table

7447 7-segment decoder

Functional layout



http://www.wisc-online.com/Objects/ViewObject.aspx?ID=DIG7307

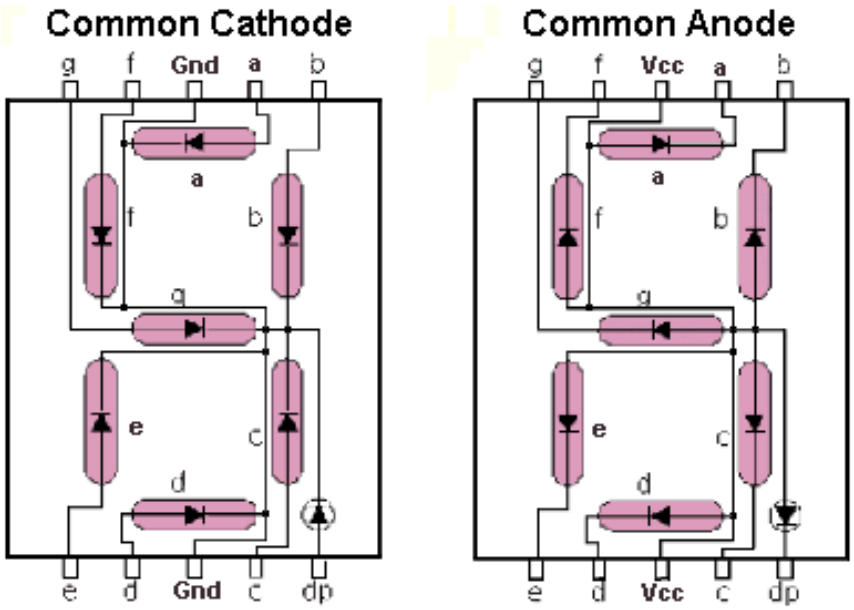


http://www.interfacebus.com/ic-bcd-to-7-segment-decoder-schematic.html

Note inverted outputs

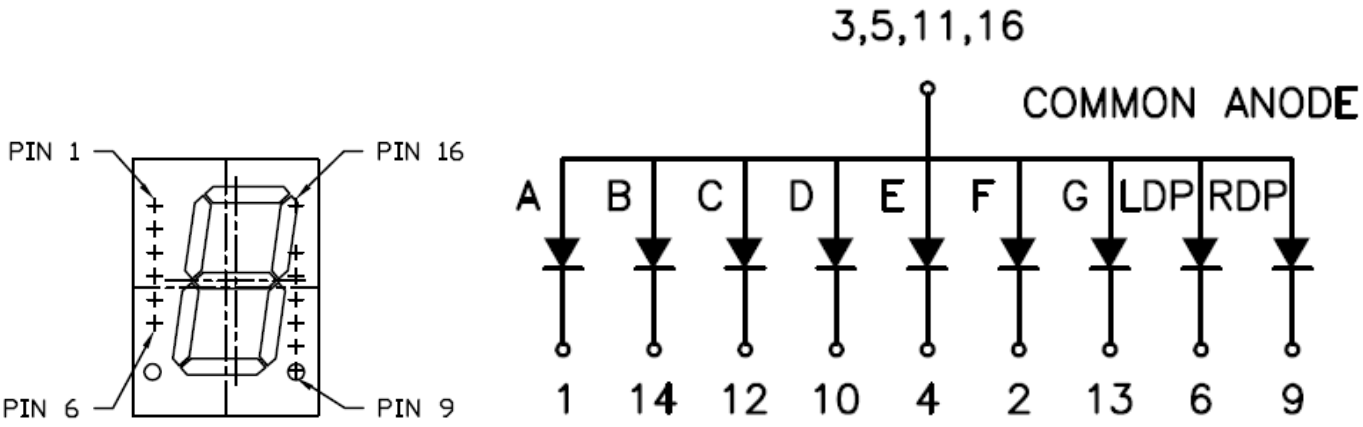
7-segment LED display

Common cathode vs. common anode



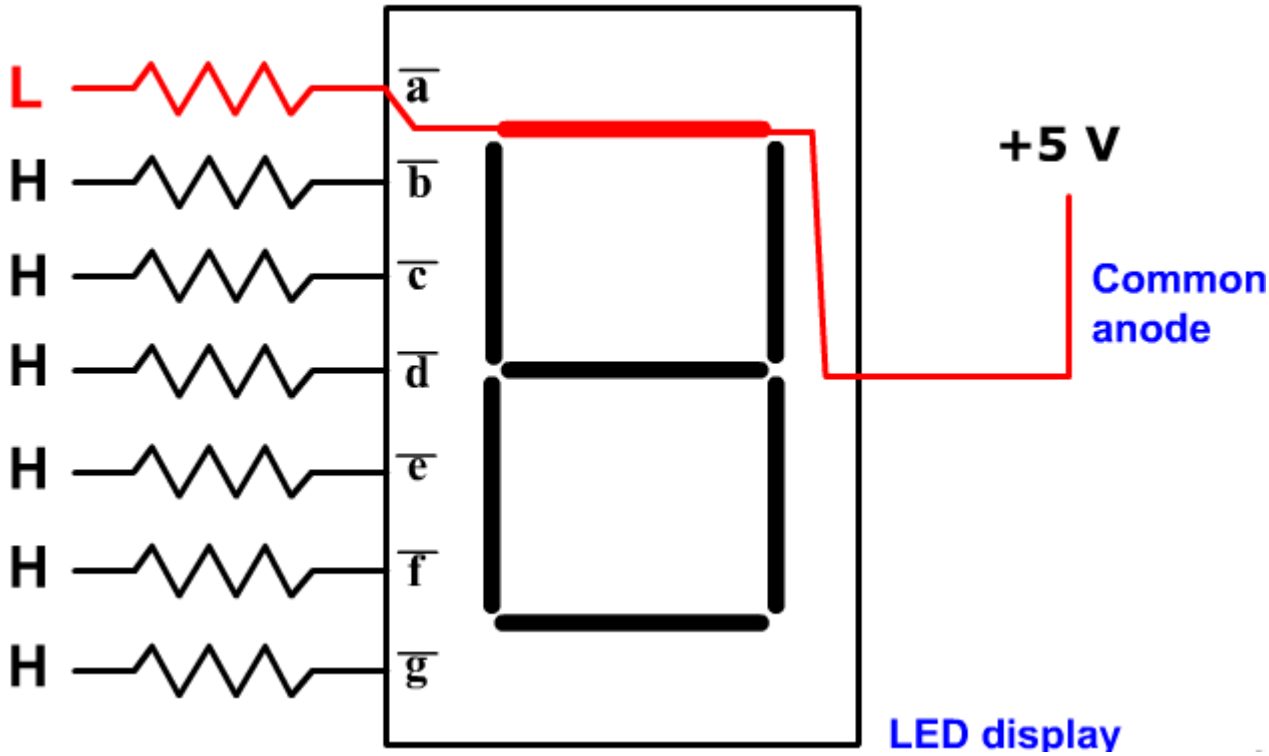
http://www.thelearningpit.com/lp/doc/7seg/7seg.html

Our common anode chip



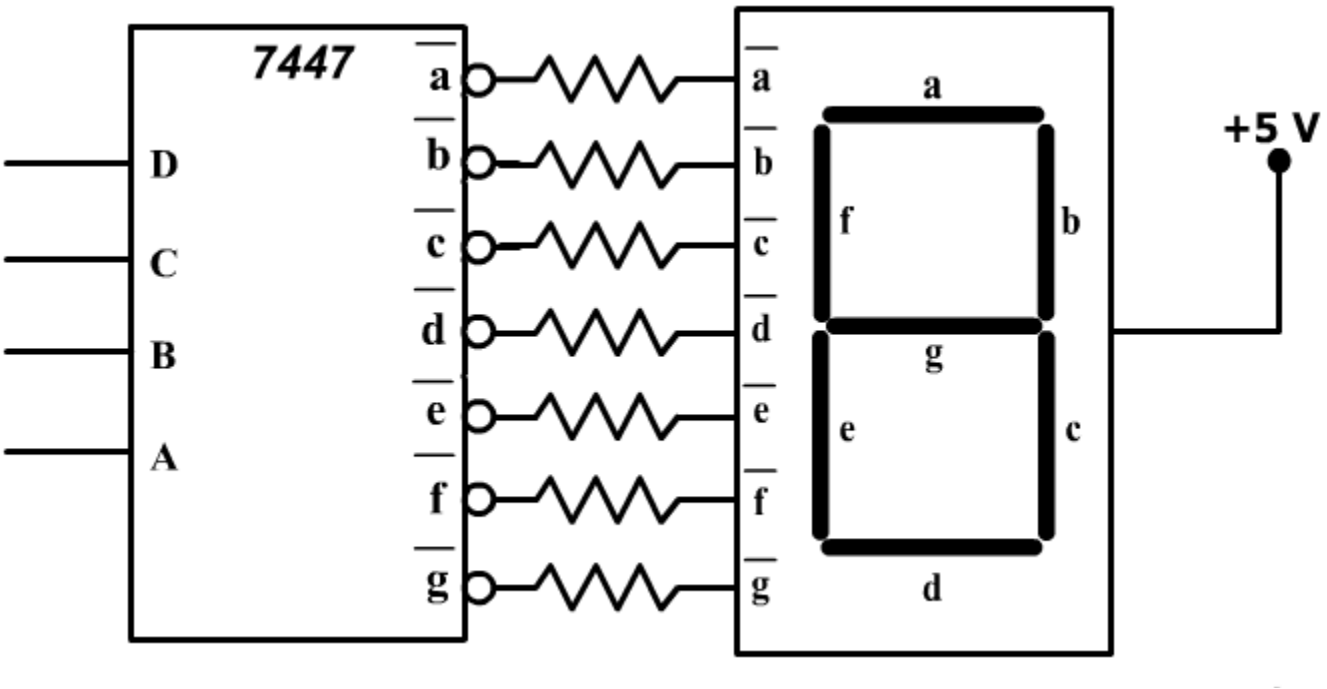
Common anode function

Drive control pin low to light segment

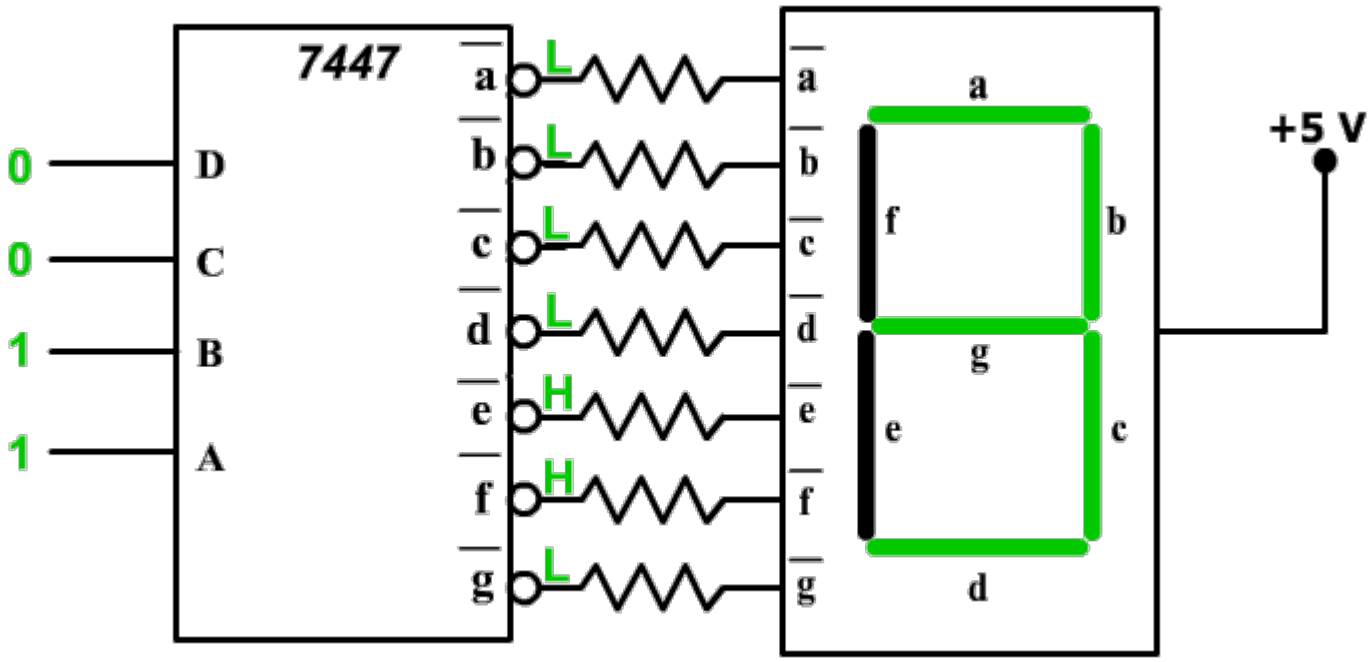


Note current limiting resistors.

Decoder + display chip



Example operation

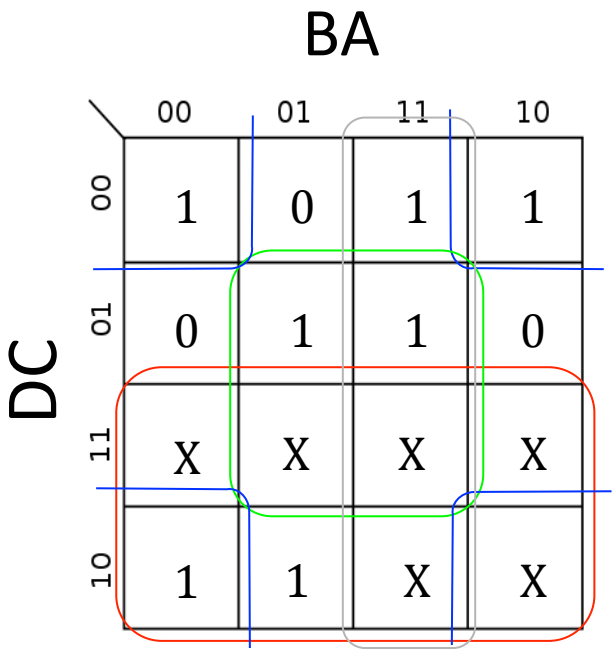


Quiz 15.1

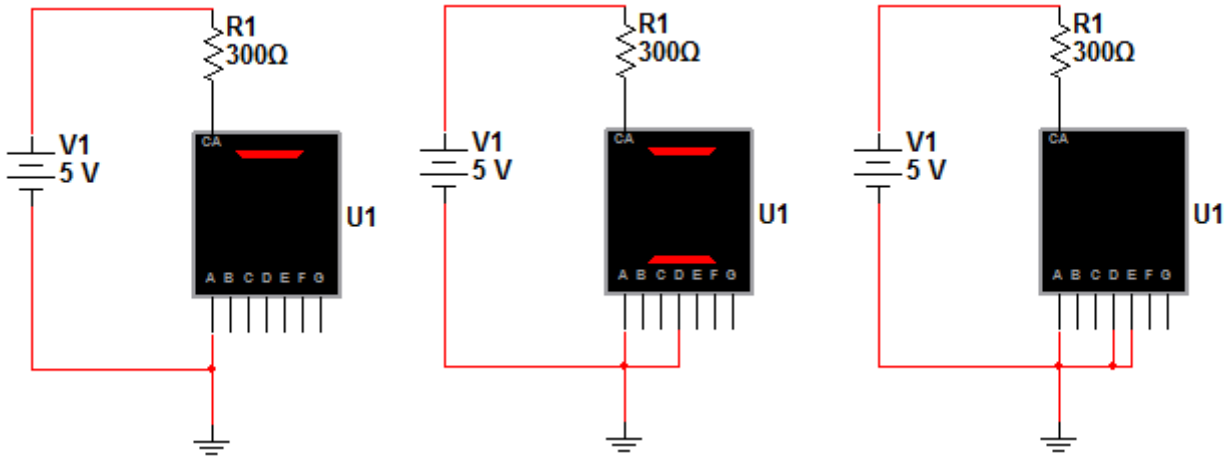
Q: Which term would appear as one of the product terms in a sum-of-products expression for the decoder logic circuit for the "a" segment of a BCD to 7-segment decoder, assuming all terms beyond BCD 9 are "don't care"? This would not be the only term, but would be one of them. You can use the truth table given in the notes. Hint: You do not need to use a Karnaugh map, although you can if you wish.

A: A
B: B
C: C
D: D

This can be found by observation from the logic table or by filling out the Karnaugh map



Quiz 15.2



Q: Current limiting resistors are usually placed on the cathodes of each diode in a common anode seven segment display. Your lab partner suggests reducing the component count by putting the current limiting resistor on the common anode instead. The three images above show the multisim results of trying this with 1, 2 and 3 segments. Note that two can be made to emit light, but not three. The reason for this is…

A: The LED voltage is less than the required ~2 V in the third case.

B: The LED current is less than the required ~5 mA in the third case.

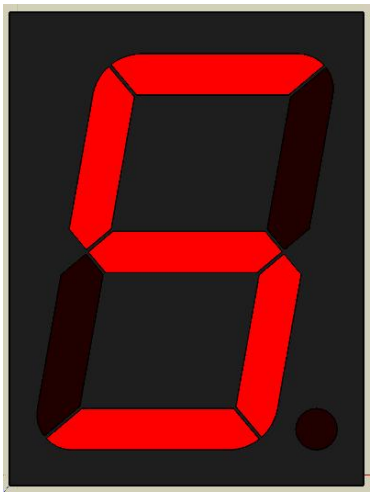
C: The LED voltage is too high in the third case.

D: The LED current is too high in the third case

E. The LEDs are no longer forward biased in the third case.

The current is limited to about 10 mA assuming a diode turn-on voltage of 2 V. But the current is split between all diodes, so drops every time another diode is turned on.

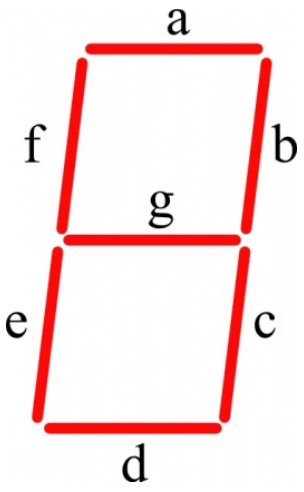
Quiz 15.3



Q: The image above is from a common cathode seven segment LED. The abcdefg lines to the chip were what? H = 5 V, L = 0 V.

A: abcdefg =	H	L	H	H	L	H	H
B: abcdefg =	L	H	L	L	H	L	L
C: abcdefg =	L	H	L	L	H	H	H
D: abcdefg =	H	H	L	L	H	L	H
E: abcdefg =	H	L	H	H	L	H	L

Common cathode means lines are taken high to turn on segments since the cathode is connected to ground.



Bonus quiz 15.1

Short answer Q: A 4-bit counter is wired to a seven-segment decoder which in turns drives a seven segment display. The output of the 4-bit counter is DCBA = 0101. The clock cycles once, increasing the count by one. Which segment of the seven segment display turns from ON to OFF at this point?

Answer: a. The count is 5 which then goes to 6. The only segment which turns off is right at the top, which is a. This can be seen in the truth table by looking for a transition (going down, from row 5 to row 6) from 1 to 0.

Bonus quiz 15.2

Short answer Q: Imagine designing one of the logic circuits that computes the output (a-g) of a seven-segment decoder. The inputs are the binary coded decimal digits D,C,B,A. Which input states would be "DON'T CARE" on the Karnaugh map? An example input state is 1001.

Answer: All states representing decimal 10 through 15. That is, 1010, 1011, 1100, 1101, 1110, 1111

Bonus quiz 15.3

Short answer Q: In HW problem **1**, you need to design a circuit that detects your counter has reached the BCD value 6 base 10. Write a simple truth table starting at 0000 and ending at 1001 (BCD 9). As the output, write in the appropriate 0, 1 or X in each row. What is the simplest logic function that will implement this truth table? Since pin 9 of the 161 counter is active low, this would then need to be inverted before being wired to ~LOAD.

Answer:

D	C	B	A	f
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	X
1	0	0	0	X
1	0	0	1	X

States 7,8,9 are don't care because the counter will never reach these values. It will roll over from 6 to zero. By observation f = C.B

David Malawey
MEEN 685 Directed Studies (digital electronics)
KRISYS Robot documentation
9/17/2015

Overview

Krysis is a two-wheeled robot which performs the task of continuously driving along a path on the ground given by a wire carring AC voltage. Figure 1 shows the complete robot. The components are described in the following paragraphs.

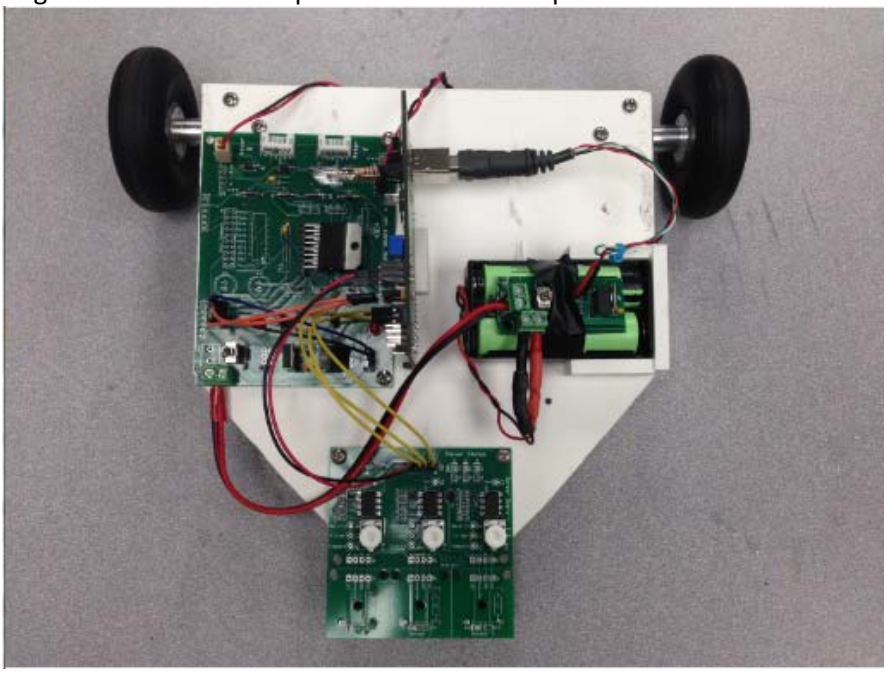


Figure 1. Krisys Robot

Components

The motor driver board Figure 2 contains a dual H-bridge chip which delivers power to two wheels according to a strength signal and a direction signal. The board receives PWM signals and battery power and output independent DC power to the two connectors shown on the left hand of the figure.

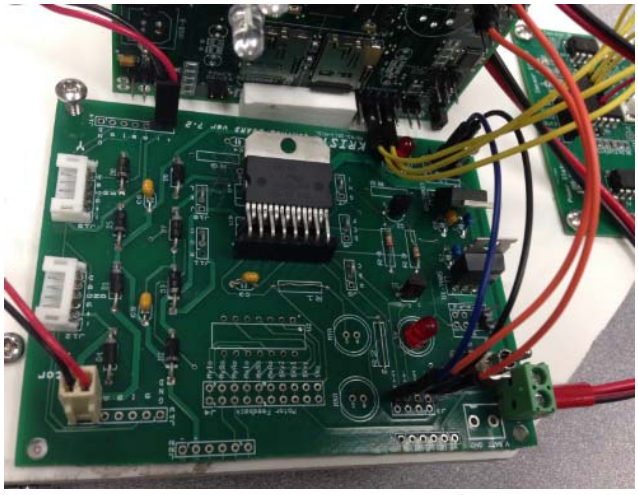


Figure 2. Motor Driver Board

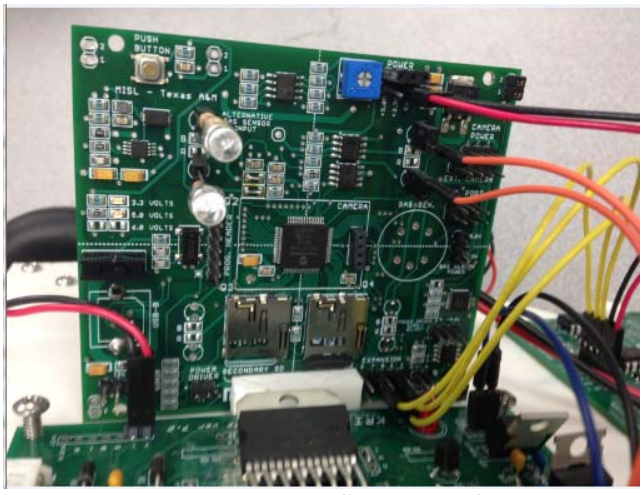


Figure 3. NESI intelligence Board

The brain of the Krisys robot is shown in Figure 3. The NESI board runs a Microchip PIC24 microcontroller which is programmed with the c-code in the attached document. The pins are configured such that three inputs are taken as digital signals to indicate the condition of the sensors on the front of the vehicle. The PIC24 calculates what power signal to send to the motor driver board via two digital voltage outputs configured as PWM pins. The connections to the board are all on the right hand in Figure 3 as follows:

- Power (R&B) from the battery @ 5V
- PWM (orange) digital voltage outputs for motor driving
- Digital (yellow) input pins from the sensor board

The power supply (Figure 4) regulates the battery source voltage to 5.0 volts for safe delivery to the NESI board. Two 3.6V Li-ion batteries in series give a source voltage of 7.2 volts at full charge. The raw, unregulated voltage is delivered to the motor driver board to give more power to the DC motors. The power has a shutoff switch, as do all boards except for the sensor board.

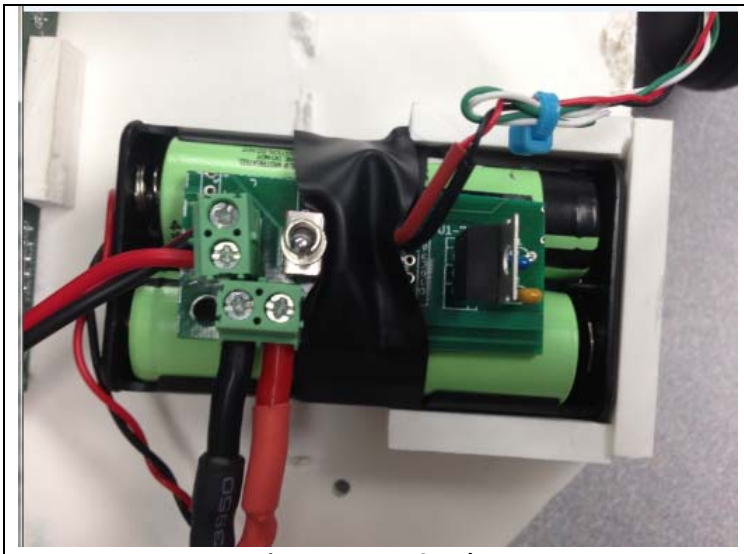


Figure 4. Power Supply

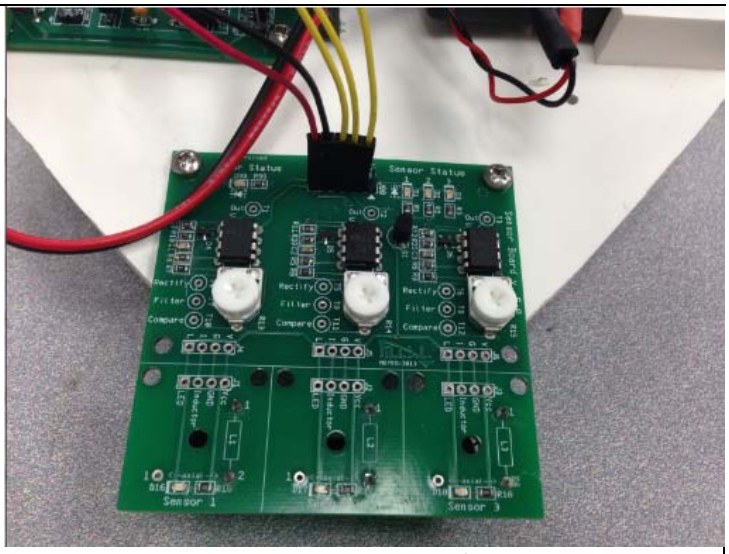


Figure 5. Sensor Board

The sensor board connects three sensors to digital output signals (yellow wires) shown in Figure 5. The sensors are inductive coils that respond to electric fields generated by nearby AC current. The board is arranged to deliver an active low signal for each sensor, and the sensors are calibrated for sensitivity by the white dials. In operation, the coil which lies closest to the AC wire will deliver a low signal and when a wire is between two coils, both coils are activated. This gives 7 possible states as described in Table 1. Essentially, 6 degrees of accuracy are given for the robot to navigate by.

Table 1. Navigation States

Description	Number of possible states
Wire lies under a coil	3
Wire lies between two coils	2
Wire lies outside the sensitivity zone of all coils	1
Total	**6**

Navigation strategy

The robot is programmed to drive the motors according to the input condition and drive along the wire as quickly as possible. For all conditions, the robot tries to align itself so the center sensor is activated. If the robot lies to the right of the wire, the left wheel slows down momentarily until the state change is satisfied. In this experiment, only 100% and 80% duty cycles are used (see attached code). In future programming, the system could be written to use a more continuous signal for smoother operation or advanced control strategy to reduce overshoot and other undesired characteristics. As for the current configuration, the robot navigates fairly quickly and does not lose track of its track!

C:/Users/David/MPLABXProjects/NESI+Core/src/app/main-david_motor.c

```
/*
 * File:   main-david_motor.c
 * Author: David Malawey
 * Created on August, 2015, H:MM xM
 */

#include <nesi.h>
#include "system.h"

/*
 * This program runs the crisis motors based on sensor inputs.
 * It creates a case statement to power the motors using PWM
 * based on the combination of inputs (from inductive sensors)
 * which tell the robot where it is pointed along the track wire.
 */

int main(void)
{
       nesi.init();          // initialize all modules
       int L M R
       char state = 'B';

        Boolean readSL(void)
        {
            _TRISB12 = 1;  // configure port as input (expansion pin 7)
            _PCFG12 = 1;
            return !_RB12; // return the inverse value on the pin (active low)
        }

        Boolean readSM(void)
        {
            _TRISG9 = 1;  // configure port as input (expansion pin 3)
            _PCFG9 = 1;
            return !_RG9; // return the inverse value on the pin (active low)
        }

        Boolean readSR(void)
        {
            _TRISB5 = 1;  // configure port as input (expansion pin 4)
            _PCFG5 = 1;
            return !_RB5; // return the inverse value on the pin (active low)
```

C:/Users/David/MPLABXProjects/NESI+Core/src/app/main-david_motor.c

```
        }

    while(1)
    {
        if(readSL() && !readSM()){
            state='A';
        }
        else if (readSL() && readSM()){
            state='B';
        }
        else if (readSM() && !readSL() && !readSR){
            state='C';
        }
        else if (readSM() && readSR()){
            state='D';
        }
        else if (readSR() && !readSM()){
            state='E';
        }
        else{
            state='F';
        }

           switch(state)
           {
            case 'A' :
                 ledR.dutycycle(80);
                 ledB.dutycycle(100);
              break;
            case 'B' :
                 ledR.dutycycle(80);
                 ledB.dutycycle(100);
               break;
            case 'C' :
                 ledR.dutycycle(100);
                 ledB.dutycycle(100);
              break;
            case 'D' :
                 ledR.dutycycle(100);
                 ledB.dutycycle(80);
            case 'E' :
```

C:/Users/David/MPLABXProjects/NESI+Core/src/app/main-david_motor.c

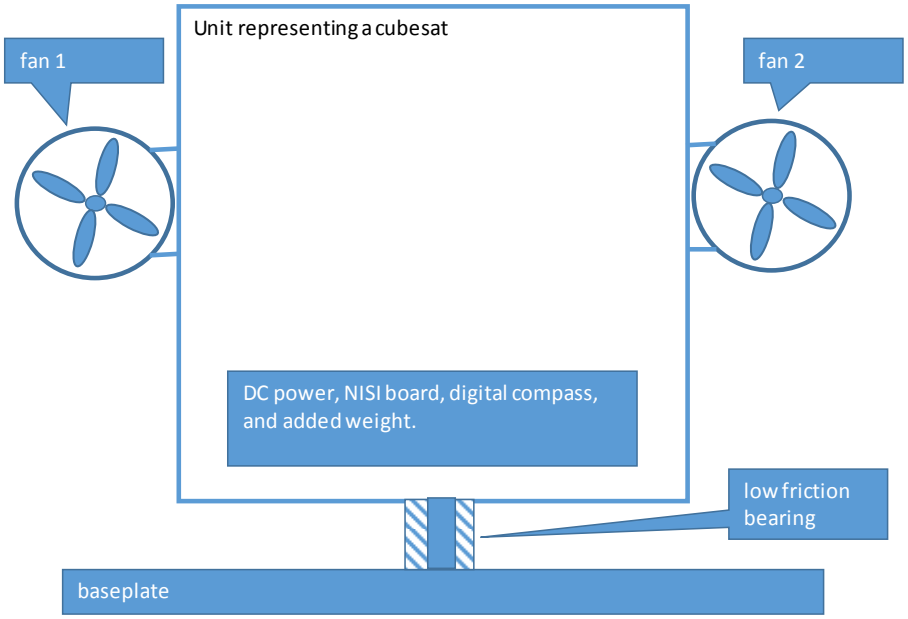
```
                ledR.dutycycle(100);
                ledB.dutycycle(80);
           case 'F' :
                ledR.dutycycle(0);
                ledB.dutycycle(0);
             break;
           }
    }
    return 0;
}
```

David Malawey
ME689 Directed Studies/Digital Electronics
2015/06/29
Project Proposal

Title: single axis orientation controller

Abstract: The design will measure and control the orientation of a device with similar mass and size to a cubesat in one axis. The orientation of the z-axis will be measured magnetically using a digital compass and controlled by 2 dc motors coupled to fans.

Apparatus:



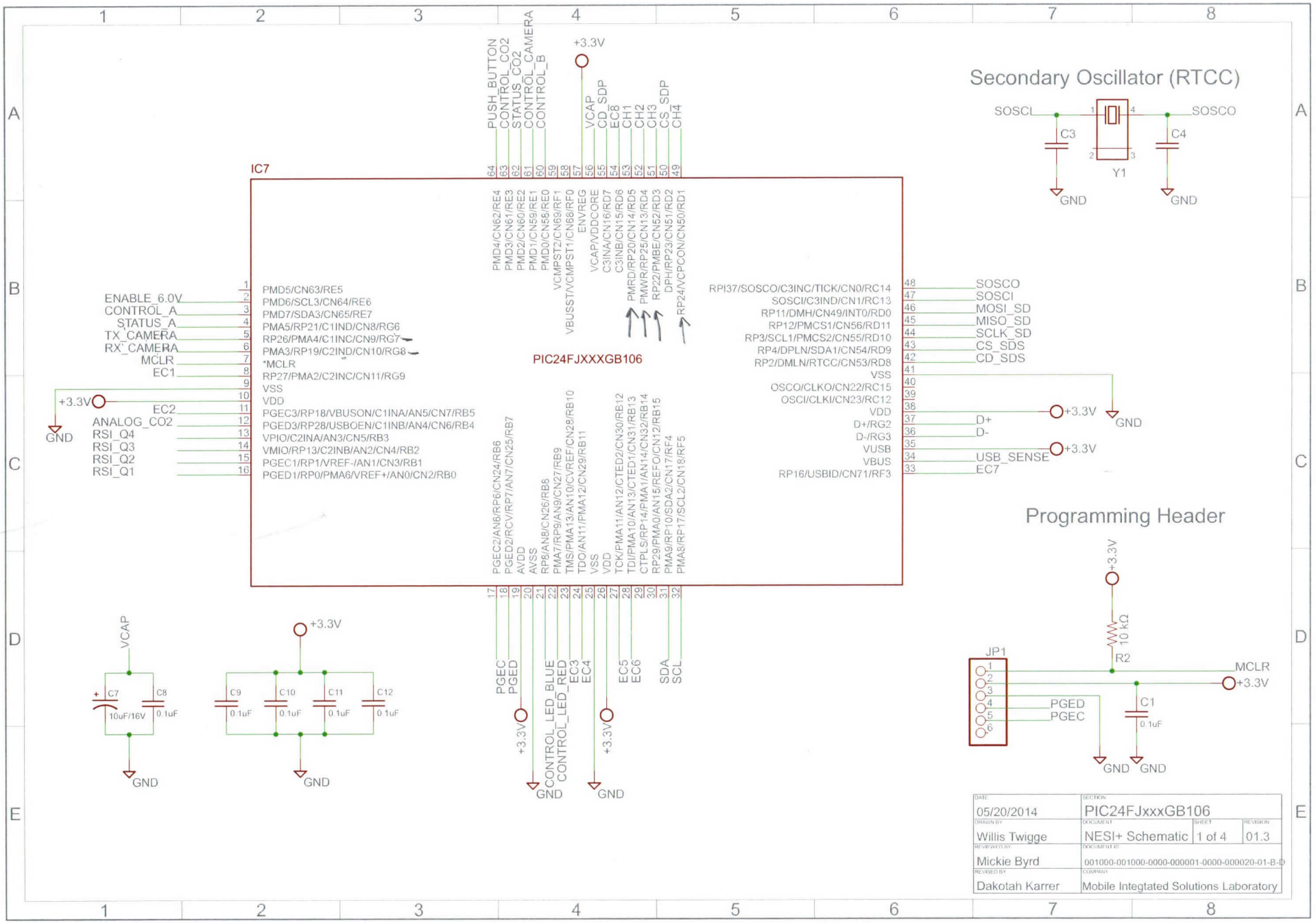
Coding:
The embedded code will have the following modules:

- Determining direction from sensor samples
- Determining angular velocity from sensor samples
- Driving 2 DC motors
- Control algorithm to reduce overshoot, settling time, and oscillations
- Input available for the user to request an orientation and velocity

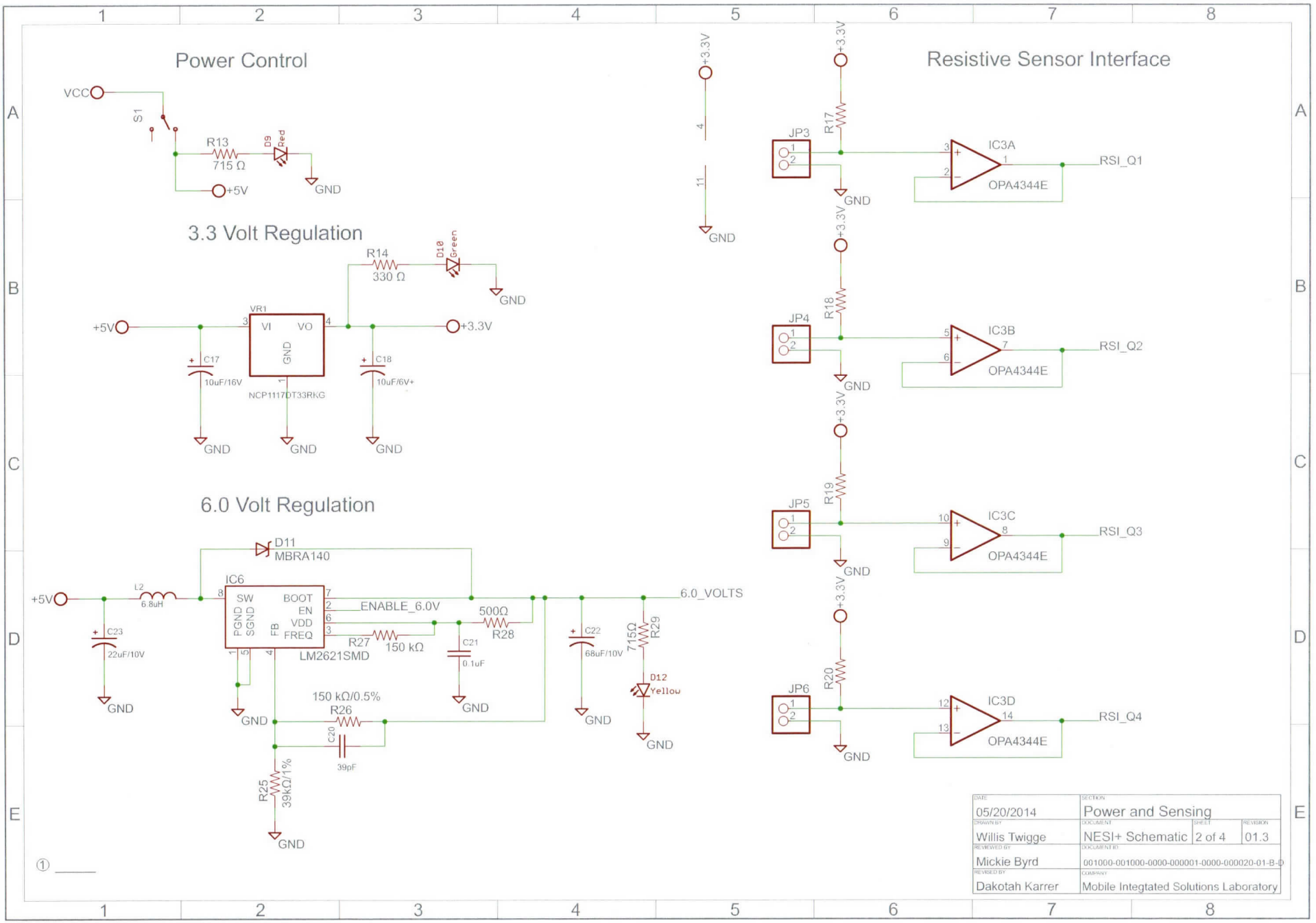
Strategy:
This experiment is designed to build skills for cubesat propulsion. The fans will only turn one direction because cold gas thrusters would have this same limitation. The system aims to be a very low friction system with no mechanical damping, so controlling overshoot will be important. An encoder would be a much more precise way to measure the orientation, but in space only magnetic fields will be available for taking bearings. Addressing these challenges in the project will improve capabilities for designing cubesat attitude control in space.

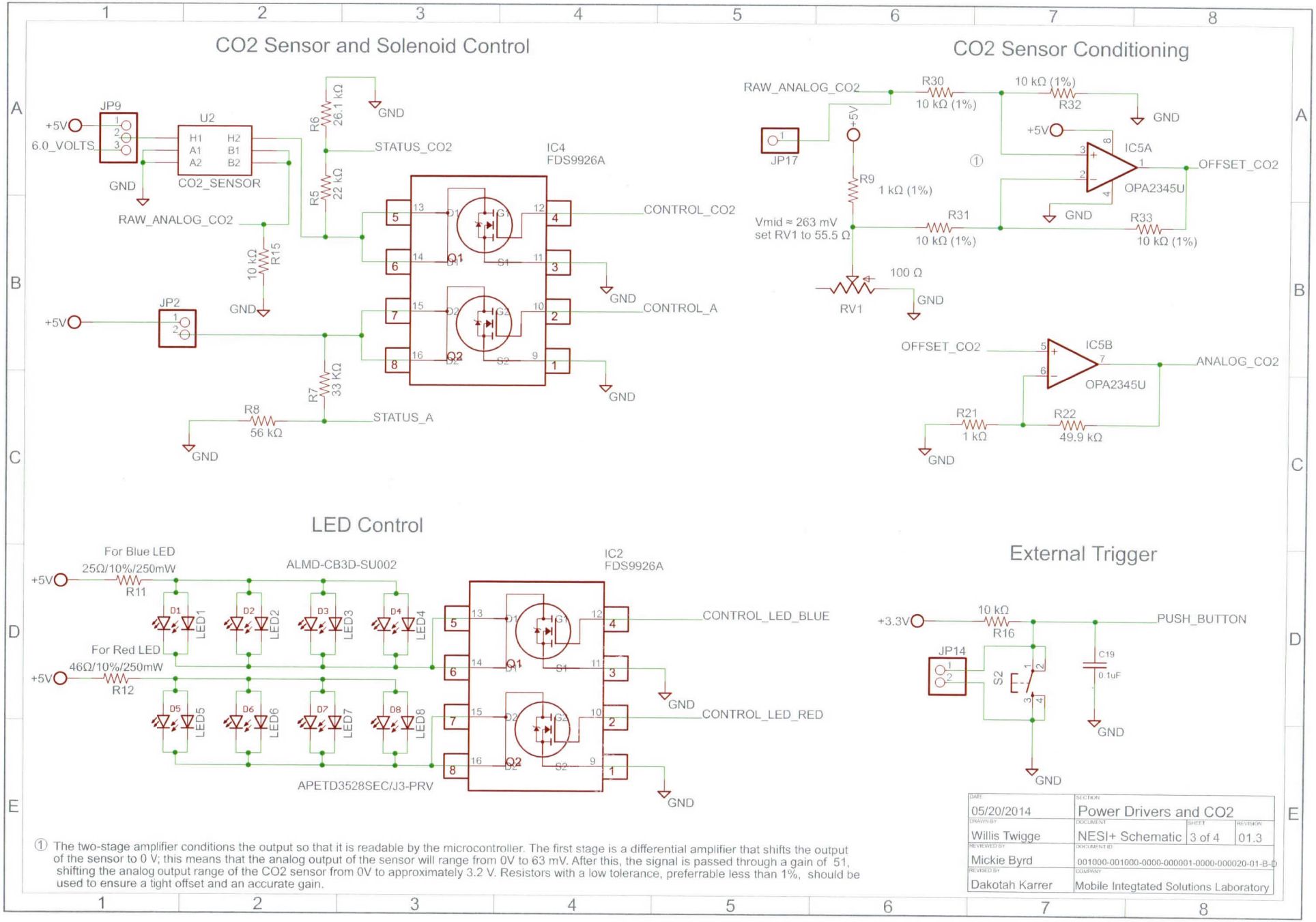
Bill of Materials

Item	Price	store	PN	qty	subtotal
SparkFun Triple Axis Magnetometer Breakout - HMC5883L	14.95	sparkfun	HMC5883L	1	14.95
NEEWER 30mm brushless fan (DC5v)	6.5	amazon	BEF0312MS	2	13.00
Motors, 3.7v					



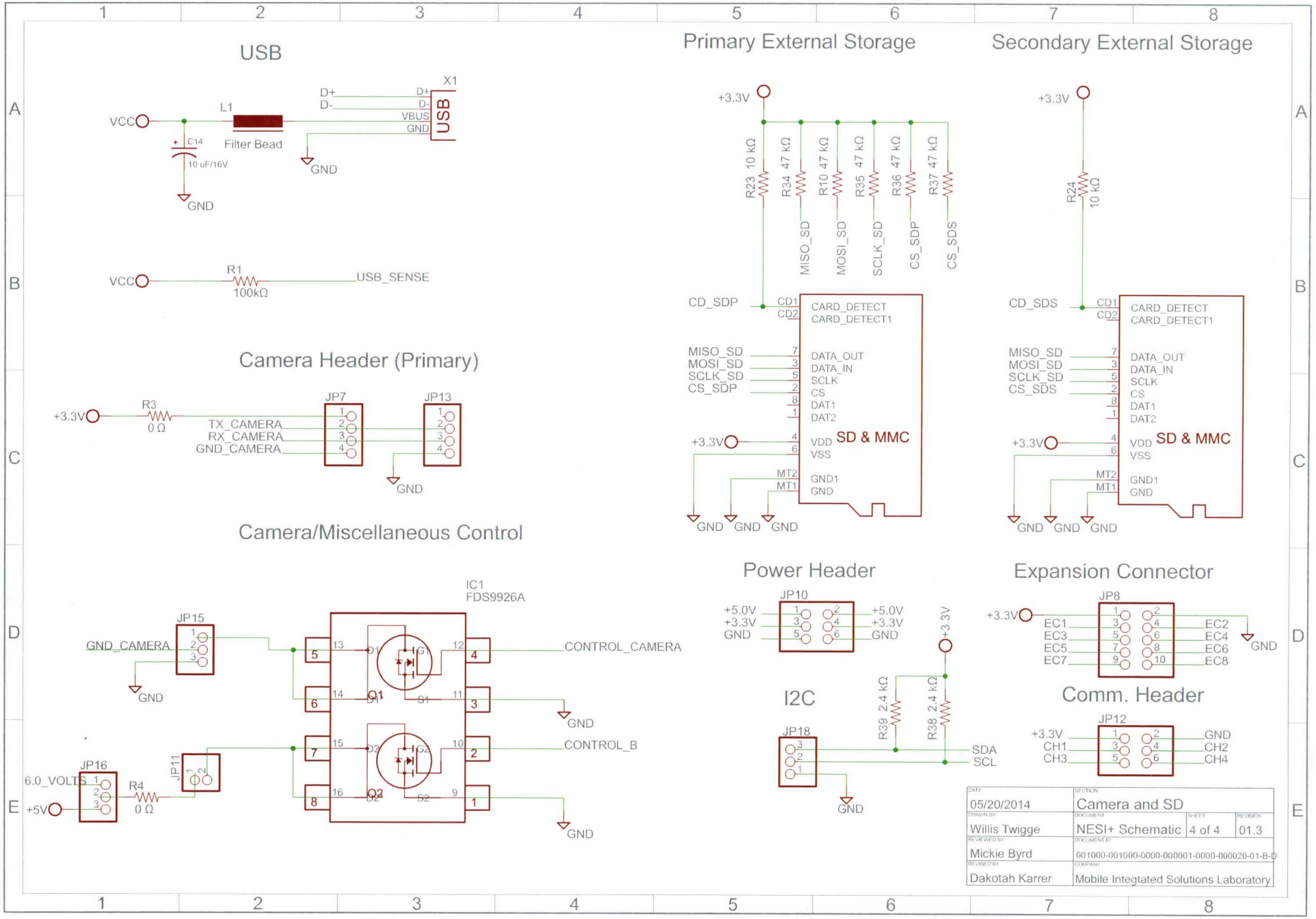
NESI Schematic


Power Control
VCC
S1
R13
715 Ω
D9
Red
+5V
GND
3.3 Volt Regulation
R14
330 Ω
D10
Green
VR1
VI
VO
GND
+5V
+3.3V
C17
10uF/16V
C18
10uF/6V+
NCP1117DT33RKG
6.0 Volt Regulation
D11
MBRA140
IC6
L2
6.8uH
SW
BOOT
EN
VDD
FREQ
PGND
SGND
FB
ENABLE_6.0V
R27
150 kΩ
LM2621SMD
C21
0.1uF
500Ω
R28
C22
68uF/10V
715Ω
R29
D12
Yellow
6.0_VOLTS
C23
22uF/10V
150 kΩ/0.5%
R26
C20
39pF
R25
39kΩ/1%
Resistive Sensor Interface
JP3
JP4
JP5
JP6
R17
R18
R19
R20
IC3A
IC3B
IC3C
IC3D
OPA4344E
RSI_Q1
RSI_Q2
RSI_Q3
RSI_Q4
DATE
05/20/2014
SECTION
Power and Sensing
DRAWN BY
Willis Twigge
DOCUMENT
NESI+ Schematic
SHEET
2 of 4
REVISION
01.3
REVIEWED BY
Mickie Byrd
DOCUMENT ID
001000-001000-0000-000001-0000-000020-01-B-D
REVISED BY
Dakotah Karrer
COMPANY
Mobile Integtated Solutions Laboratory



① The two-stage amplifier conditions the output so that it is readable by the microcontroller. The first stage is a differential amplifier that shifts the output of the sensor to 0 V; this means that the analog output of the sensor will range from 0V to 63 mV. After this, the signal is passed through a gain of 51, shifting the analog output range of the CO2 sensor from 0V to approximately 3.2 V. Resistors with a low tolerance, preferrable less than 1%, should be used to ensure a tight offset and an accurate gain.

DATE: 05/20/2014	SECTION: Power Drivers and CO2
DRAWN BY: Willis Twigge	DOCUMENT: NESI+ Schematic / SHEET: 3 of 4 / REVISION: 01.3
REVIEWED BY: Mickie Byrd	DOCUMENT ID: 001000-001000-0000-000001-0000-000020-01-B-D
REVISED BY: Dakotah Karrer	COMPANY: Mobile Integtated Solutions Laboratory


USB
X1
D+
D-
VBUS
GND
USB
VCC
L1
Filter Bead
C14
10 uF/16V
GND
R1
100kΩ
USB_SENSE
Primary External Storage
+3.3V
R23 10 kΩ
R34 47 kΩ
R10 47 kΩ
R35 47 kΩ
R36 47 kΩ
R37 47 kΩ
MISO_SD
MOSI_SD
SCLK_SD
CS_SDP
CS_SDS
CD_SDP
CD1
CD2
CARD_DETECT
CARD_DETECT1
DATA_OUT
DATA_IN
SCLK
CS
DAT1
DAT2
VDD
VSS
SD & MMC
GND1
GND
MT2
MT1
Secondary External Storage
R24 10 kΩ
CD_SDS
Camera Header (Primary)
R3
0 Ω
JP7
JP13
TX_CAMERA
RX_CAMERA
GND_CAMERA
Camera/Miscellaneous Control
IC1
FDS9926A
JP15
GND_CAMERA
CONTROL_CAMERA
CONTROL_B
JP16
6.0_VOLTS
+5V
R4
0 Ω
JP11
Power Header
JP10
+5.0V
+3.3V
I2C
JP18
R39 2.4 kΩ
R38 2.4 kΩ
SDA
SCL
Expansion Connector
JP8
EC1
EC2
EC3
EC4
EC5
EC6
EC7
EC8
Comm. Header
JP12
CH1
CH2
CH3
CH4
05/20/2014
Camera and SD
Willis Twigge
NESI+ Schematic
4 of 4
01.3
Mickie Byrd
001000-001000-0000-000001-0000-000020-01-B-D
Dakotah Karrer
Mobile Integtated Solutions Laboratory

SparkFun Packing List

#2234337

6333 Dry Creek Parkway • Niwot, CO 80503
1-303-284-0979

ered: 2015-07-07 01:35:42 pm

Billing
David Malawey

College Station, Texas 77840
United States

Delivery
David Malawey

Shipping Method: USPS First Class Mail

Payment Method: Credit Card
Order Status: Picking

Line #	Qty	SKU	Product Name
1	2	WRL-10532	RF Link Receiver - 4800bps (434MHz)
2	2	WRL-10534	RF Link Transmitter - 434MHz

SUPPLIER'S CERTIFICATION OF COMPLIANCE
It is hereby certified that the product(s) provided in this shipment conform to the requirements and the manufacturer's part number identified in the customer's purchase order and these parts have been received, stored and shipped by SparkFun Electronics.

Casey DeLio, SparkFun Customer Service Manager

These commodities, technology or software were exported from the United States in accordance with the Export Administration Regulations. Diversion contrary to U.S. law is prohibited.

Please direct any questions or concerns to cservice@sparkfun.com or (303) 284-0979.

Did you know SparkFun's packaging is recyclable?
Both our red boxes and brown paper packaging can be recycled in most areas. Thanks for doing your part!

Parts

~~valQ1 = resistiveSensors.getQ1(5,100)~~

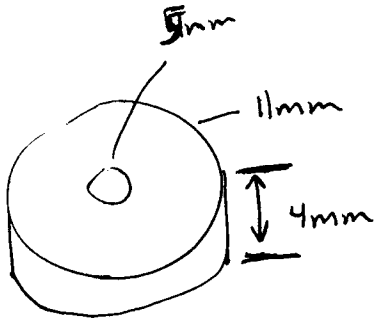
~~bright light 220 Ohm~~

light 12 kOhm

~~no light 1.3 kOhm~~

Transducer bearing

TRA 5116



Preliminary Attitude Control Design

measure direction at t_1 → dt_1

measure direction at t_2 → dt_2

$S_1 = (dt_2 - dt_1) \div (t_2 - t_1)$ // speed = Δ angle / Δ time

// x_1 is a slow positive speed

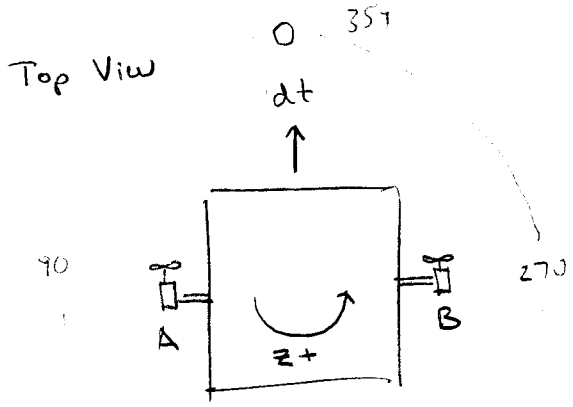
// dcA = duty cycle A

```
if S1 > x1
   { Power A = dcA
     Power B = OFF
   }
```

// x_2 is a small negative spd

```
else if S1 < x2
    { Power B = dcB
      Power A = OFF
    }

else
    {
    if dt2 ≥ 180
        { Power A = low      // Set 10% duty cycle
        }
    if dt2 ≤ 180
        { power B = low
        }
```



NESI 9DOF IMU (connections to NESI)